****

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

using Open-Source software

Eindwerk voorgedragen door: Frederik Van Den Hof en Beerend Lauwers

**Interne promotor: Jan Martens**

**Academiejaar: 2009-2010**

Doel: tot het bekomen van het diploma Hoger Onderwijs, één cyclus, volledig leerplan – Handelswetenschappen & bedrijfskunde, opleiding Informatiemanagement en -systemen)

****

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

using Open-Source software

Eindwerk voorgedragen door: Frederik Van Den Hof en Beerend Lauwers

**Interne promotor: Jan Martens**

**Academiejaar: 2009-2010**

Doel: tot het bekomen van het diploma Hoger Onderwijs, één cyclus, volledig leerplan – Handelswetenschappen & bedrijfskunde, opleiding Informatiemanagement en -systemen)

***"Ik, Frederik Van Den Hof ,  verklaar dat, voor zover ik er weet van heb, deze scriptie geen materiaal bevat dat ooit in eender welke instelling is gebruikt om een diploma, van welke aard ook, te behalen of dat eerder werd gepubliceerd of geschreven door een ander persoon, behalve daar waar deze scriptie referenties bevat naar andere werken.”***

***"Ik, Beerend Lauwers ,  verklaar dat, voor zover ik er weet van heb, deze scriptie geen materiaal bevat dat ooit in eender welke instelling is gebruikt om een diploma, van welke aard ook, te behalen of dat eerder werd gepubliceerd of geschreven door een ander persoon, behalve daar waar deze scriptie referenties bevat naar andere werken.”***

Table of contents

[Types of Single Sign-on 6](#_Toc262483078)

[Client based 6](#_Toc262483079)

[Server based 8](#_Toc262483080)

[Kerberos Authentication Protocol 9](#_Toc262483081)

[Kerberos terminology 9](#_Toc262483082)

[Kerberos Operation 10](#_Toc262483083)

[Benefits 13](#_Toc262483084)

[Drawbacks 14](#_Toc262483085)

[Implementing a Kerberos V SSO system 15](#_Toc262483086)

[Pre-configuration 15](#_Toc262483087)

[Operating System 15](#_Toc262483088)

[Network details 15](#_Toc262483089)

[Users 15](#_Toc262483090)

[Services 16](#_Toc262483091)

[Useful utilities 16](#_Toc262483092)

[Step 1: Berkeley Internet Name Domain (BIND) 17](#_Toc262483093)

[DNS Operation 17](#_Toc262483094)

[BIND Installation 19](#_Toc262483095)

[BIND Configuration 19](#_Toc262483096)

[named.conf.local 19](#_Toc262483097)

[named.conf.options 19](#_Toc262483098)

[Zone files 19](#_Toc262483099)

[resolv.conf 20](#_Toc262483100)

[hostname files 20](#_Toc262483101)

[Testing BIND 21](#_Toc262483102)

[Step 2: NTP-server 23](#_Toc262483103)

[NTP Installation 23](#_Toc262483104)

[NTP Configuration 23](#_Toc262483105)

[Setting up NTP client 24](#_Toc262483106)

[Step 3: MIT Kerberos V 25](#_Toc262483107)

[Kerberos Installation 25](#_Toc262483108)

[Kerberos Configuration 25](#_Toc262483109)

[krb5.conf 25](#_Toc262483110)

[kdc.conf 26](#_Toc262483111)

[kadm5.acl 27](#_Toc262483112)

[Kerberos Administration 27](#_Toc262483113)

[Initializing the Kerberos realm 27](#_Toc262483114)

[kadmin.local 27](#_Toc262483115)

[listprincs 27](#_Toc262483116)

[addprinc 28](#_Toc262483117)

[kadmin 28](#_Toc262483118)

[Adding users 28](#_Toc262483119)

[Setting up Kerberos clients 28](#_Toc262483120)

[Obtaining tickets manually 28](#_Toc262483121)

[Obtaining tickets automatically 29](#_Toc262483122)

[Step 3: Kerberized services 30](#_Toc262483123)

[telnet 30](#_Toc262483124)

[telnet installation 30](#_Toc262483125)

[telnet configuration 30](#_Toc262483126)

[Testing telnet 31](#_Toc262483127)

[rlogin 32](#_Toc262483128)

[Testing rlogin 32](#_Toc262483129)

[Secure Shell (SSH) 32](#_Toc262483130)

[SSH installation 32](#_Toc262483131)

[SSH configuration 32](#_Toc262483132)

[Testing SSH 33](#_Toc262483133)

[File Transfer Protocol 33](#_Toc262483134)

[FTP installation 33](#_Toc262483135)

[FTP configuration 33](#_Toc262483136)

[Testing FTP 33](#_Toc262483137)

[Step 5: Web SSO 35](#_Toc262483138)

[Web server 35](#_Toc262483139)

[Web server installation 35](#_Toc262483140)

[Web server configuration 36](#_Toc262483141)

[mod\_auth\_kerb 37](#_Toc262483142)

[Introduction 37](#_Toc262483143)

[The Negotiate mechanism 37](#_Toc262483144)

[mod\_auth\_kerb installation 37](#_Toc262483145)

[mod\_auth\_kerb configuration 38](#_Toc262483146)

[Client configuration 39](#_Toc262483147)

[Testing mod\_auth\_kerb 41](#_Toc262483148)

[Benefits and drawbacks 42](#_Toc262483149)

[php\_krb5 43](#_Toc262483150)

[php\_krb5 installation 43](#_Toc262483151)

[php\_krb5 configuration 44](#_Toc262483152)

[Testing php\_krb5 44](#_Toc262483153)

[Benefits and drawbacks 44](#_Toc262483154)

[Web portal 45](#_Toc262483155)

[Portal for mod\_auth\_kerb 46](#_Toc262483156)

[Portal for php\_krb5 50](#_Toc262483157)

[Sample applications 57](#_Toc262483158)

[Personal notepad 58](#_Toc262483159)

[MediaWiki 66](#_Toc262483160)

[Roundcube web mail 71](#_Toc262483161)

[References 80](#_Toc262483162)

# Prelude

This thesis is the result of our research regarding single-sign on. The subject was proposed to us by our internal promoter and Networking professor Jan Martens who was interested in the way single sign-on actually works and in how to actually set up a single sign-on enabled network.

We were both quite interested in the topic and decided to take on the challenge.

Configuring Kerberos properly was not the easiest of tasks, we experienced quite some setbacks but thankfully there were always people on mailing lists, forums and chatrooms who were ready to assist us with our problems. Not to mention all the community-written documentation that aided us in achieving our goal.

Our greatest gratitude goes out to the open-source community. In order to give something back to them we plan on publishing our findings online. This is also why we decided to write this thesis entirely in English, because we believe we can reach more people that way.

We would ofcourse also like to thank our promoters Jan Martens and Tom Van Kerckhove for the advice and equipment they provided us with.

Frederik Van Den Hof  
Beerend Lauwers

# Summary

# Introduction

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. It can be agreed upon that this proves to be detrimental for the overall efficiency of a company.

This paper will briefly discuss the different single sign-on alternatives available on the market and henceforth focus on the single sign-on opportunities the Kerberos authentication protocol has to offer.

The part concerning Kerberos will conclude:

* Information about how the Kerberos protocol operates
* Instructions on installing and maintaining a Kerberos server
* Instructions on installing single sign-on enabled services
* Instructions on installing a web-based single sign-on platform

Furthermore this paper will address other issues that play a significant role in a Kerberos enabled environment such as name resolution and time synchronization.

Upon having thoroughly read this paper one should have a fairly good understanding of the modus operandi of the Kerberos protocol and should be capable of setting up one’s own Kerberos enabled network.

# Types of Single Sign-on

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

## 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.



Figure 1:. http://www.novell.com/products/securelogin/media.html



Figure 2: 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 the same 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:



Figure 1: Kerberos authentication steps

Each step is described in detail on the following 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 3: 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 4: 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 5: 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 6: 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 7: 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.



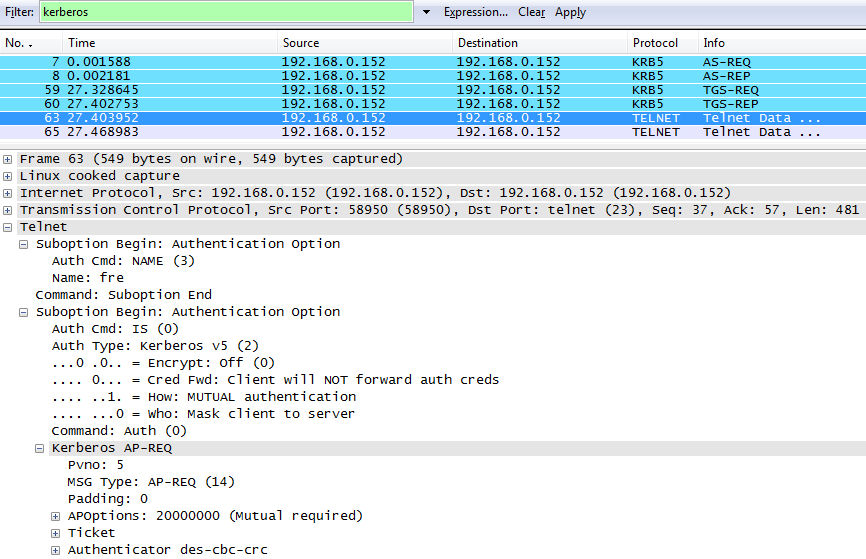
Figure 8: Step 12 and 13

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



Figure 9: Step 14

Upon taking a look at a Wireshark capture of a kerberized telnet session being initiated the aforementioned steps can easily be retrieved. Notice how in the last 2 messages Kerberos authentication data is imbedded within the telnet protocol. The configuration of Kerberized services will be explained later on.

Figure 10: kerberized telnet wireshark capture

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 a Kerberos V SSO system

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 allows 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

# 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 A 192.168.1.200

mail IN A 192.168.1.200  
krb IN A 192.168.1.200

And reverse-1.168.192 contained this configuration:

$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 (found in the /etc directory) 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. The hostname command is also used to determine and modify the hostname of the computer.

/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: NTP-server

When looking back upon the way Kerberos works it can be concluded that the authentication process relies on synchronized system clocks. Desynchronized time in between systems willresult in issues with the different timestamps and validity periods .

In order to synchronize the system time in our network the Network Time protocol or NTP will be made use of. In a nutshell, the NTP-server will be configured in such a way to check several reliable sources for the actual time and then broadcast these across the assigned network.

## NTP Installation

In order to proceed the ntp-server package must be installed.

sudo apt-get install ntp-server

## NTP Configuration

After installation the main configuration that can be found at /etc/ntp.conf should look something like this:

# /etc/ntp.conf, configuration for ntpd; see ntp.conf(5) for help  
  
driftfile /var/lib/ntp/ntp.drift  
  
  
# Enable this if you want statistics to be logged.  
#statsdir /var/log/ntpstats/  
  
statistics loopstats peerstats clockstats  
filegen loopstats file loopstats type day enable  
filegen peerstats file peerstats type day enable  
filegen clockstats file clockstats type day enable  
  
  
# You do need to talk to an NTP server or two (or three).  
server ntp.ubuntu.com  
  
  
# Access control configuration; see /usr/share/doc/ntp-doc/html/accopt.html for  
# details.  The web page <http://support.ntp.org/bin/view/Support/AccessRestrictions>  
# might also be helpful.  
#  
# Note that "restrict" applies to both servers and clients, so a configuration  
# that might be intended to block requests from certain clients could also end  
# up blocking replies from your own upstream servers.  
  
# By default, exchange time with everybody, but don't allow configuration.  
restrict -4 default kod notrap nomodify nopeer noquery  
restrict -6 default kod notrap nomodify nopeer noquery  
  
# Local users may interrogate the ntp server more closely.  
restrict 127.0.0.1  
restrict ::1  
  
# Clients from this (example!) subnet have unlimited access, but only if  
# cryptographically authenticated.  
#restrict 192.168.123.0 mask 255.255.255.0 notrust  
  
  
# If you want to provide time to your local subnet, change the next line.  
# (Again, the address is an example only.)  
broadcast 192.168.1.255  
  
# If you want to listen to time broadcasts on your local subnet, de-comment the  
# next lines.  Please do this only if you trust everybody on the network!  
#disable auth  
#broadcastclientsudo touch krb5/kdc.log

Note that in this configuration only the server ntp.ubuntu.com is addressed, ideally it is best to supply a list of servers that lay close to the server’s geographical location. The only parameter changed in this configuration is the broadcast parameter. It Is set now to the broadcast address of our network.

## Setting up NTP client

On the client the ntpdate package must be installed.

sudo apt-get install ntpdate

Once installed the following command needs to be issued in order to synchronize the time.

sudo ntpdate –dsv krb.khm.lan

The added parameters perform the command in verbose and debugging mode. It takes quite some time for the NTP-server to sync with the servers, even with a broadband connection speed this can take up to 10 minutes. Most errormessages regarding “strata too high” are usually too blame on a unsynchronized ntp-server. It is advised in that case to try again later.

# Step 3: 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 installed with the krb5-kdc and krb5-admin-server packages, the latter contains several applications required for proper Kerberos administration.

sudo apt-get install krb5-admin-server

sudo apt-get install krb5-kdc

The MIT Kerberos development files were also installed using the libkrb5-dev package. These files will be used for compiling several modules for the web SSO.

sudo apt-get install libkrb5-dev

## 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 installation

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

sudo apt-get install krb5-telnetd

### telnet configuration

After installation, 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 negotiation of authentication failed. 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

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. Services use a keytab file to store its secret keys. 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 also houses 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 whether 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.

# Step 5: Web SSO

## Web server

### Web server installation

For development purposes, the XAMPP Web Server was used as a test bed. XAMPP contains a multitude of pre-configured packages and plugins [TODO09], but is weakly secured for development purposes. A special tool can be used to harden the server [TODO10]. The following packages were downloaded and installed: XAMPP Linux 1.7.3a, XAMPP Linux 1.7.3a Development package and XAMPP Linux 1.7.4 Beta 3. The latter package was installed because of a bug in the PHP distribution of XAMPP Linux 1.7.3a that prevented MediaWiki from functioning correctly.

First, the files were downloaded from the XAMPP website:

<http://www.apachefriends.org/download.php?xampp-linux-1.7.3a.tar.gz>

<http://www.apachefriends.org/download.php?xampp-linux-devel-1.7.3a.tar.gz>

<http://www.apachefriends.org/xamppbetazz/xampp-linux-1.7.4-beta3.tar.gz>

Then, each file was unpacked in the order in which they were downloaded using the following command:

sudo tar xvfz xampp-linux-1.7.3a.tar.gz -C /opt

sudo tar xvfz xampp-linux-devel-1.7.3a.tar.gz -C /opt

sudo tar xvfz xampp-linux-1.7.4-beta3.tar.gz -C /opt

Finally, the web server was started with the following command:

sudo /opt/lampp/lampp start

Starting XAMPP for Linux 1.7.4-beta3...

XAMPP: Starting Apache with SSL (and PHP5)...

XAMPP: Starting MySQL...

XAMPP: Another FTP daemon is already running.

XAMPP for Linux started.

Browsing to www.khm.lan gave the following webpage:

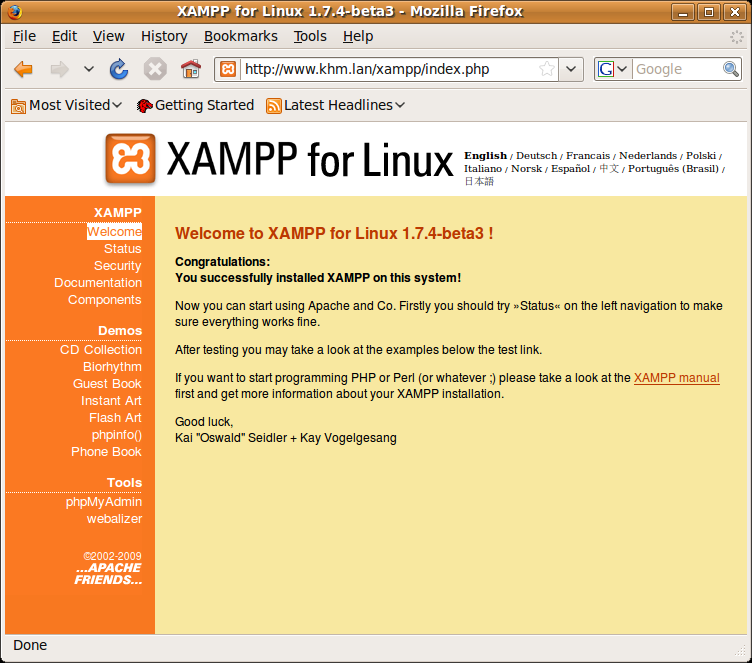


Figure TODO: Screenshot of the main web page of XAMPP

### Web server configuration

All configuration files for XAMPP are located in the /opt/lampp/etc directory. The file httpd.conf, which is Apache’s configuration file in XAMPP, was modified to contain the correct server name:

ServerName www.khm.lan

The rest of the values were left untouched.

Out of the box, XAMPP does not start or stop automatically. Hence, the following commands were issued to gain this functionality:

cd /etc/rc2.d

sudo ln -s /opt/lampp/lampp S99lampp

sudo ln -s /opt/lampp/lampp K01lampp

These commands create symbolic links to the XAMPP initialization script to start and stop XAMPP on boot and shutdown, respectively.

## mod\_auth\_kerb

### Introduction

mod\_auth\_kerb is an Apache module that makes authentication against a Kerberos server possible. It has two mechanisms:

* Basic Auth: Using a prompt, user credentials are requested and authenticated against a Kerberos server. The credentials are sent from the browser to the Apache server unencrypted.
* Negotiate: A Negotiate-capable web browser automatically requests a Service Ticket for the Apache server and sends an encrypted Application Request to the Apache server. The user is not prompted for his credentials.

The module can also temporarily save the credentials of a client and act as a delegate in CGI scripts [TODO15, TODO16].

This module will be compiled, installed on the XAMPP web server and configured to use the Negotiate mechanism to achieve single sign-on capabilities.

### The Negotiate mechanism

The Negotiate mechanism implements the SPNEGO (Simple and Protected GSSAPI Negotiation Mechanism) protocol, which allows a web client and a web server to negotiate which kind of GSSAPI authentication to use. This protocol was created by Microsoft to overcome the lack of authentication negotiation mechanism of the original GSSAPI protocol. SPNEGO currently houses two different authentication mechanisms [TODO11]:

* NTLM (NT LAN Manager): This authentication protocol was created by Microsoft and was used by the NT family of Windows operating systems. Starting from Windows 2000, Kerberos V was used in favor of NTLM.
* Kerberos: The Kerberos V authentication protocol.

The Negotiate mechanism works by adding a HTTP “WWW-Authenticate: Negotiate” header to the response to a “GET / HTTP/1.1” request of a client. If the client supports this method of authentication, negotiation of the authentication protocol is executed with use of the SPNEGO protocol: the client sends back a “GET / HTTP/1.1” request, together with a HTTP “Authorization” header which contains the initial authentication token (in the case of Kerberos, an Application Request). The server then uses this token to authenticate the client. If successful, the server sends back a “HTTP/1.1 200 Success” response. It is possible that the server also includes some GSSAPI data in the response, which can be used for mutual authentication [TODO11].

### mod\_auth\_kerb installation

The source of the module was downloaded from the development site:

<http://sourceforge.net/projects/modauthkerb/files/>

The module was untarred using the following command:

sudo tar xvfz mod\_auth\_kerb-5.4.tar.gz -C .

The module was configured, compiled and installed with the following commands:

sudo ./configure --with-apache=/opt/lampp/ --without-krb4

sudo make

sudo make install

The make install command automatically copied and chmod’ed the Apache extension in the correct folder, /etc/lampp/modules.

### mod\_auth\_kerb configuration

The mod\_auth\_kerb module can be configured to use either the Negotiate or the Basic Auth mechanism, but can also be configured to use both. In this case, the module will attempt Negotiate authentication. If this fails, it will resort to the Basic Auth mechanism.

The mod\_auth\_kerb module was added to httpd.conf:

LoadModule auth\_kerb\_module modules/mod\_auth\_kerb.so

Furthermore, the directory directive for the directory /opt/lampp/htdocs was modified in httpd.conf:

<Directory "/opt/lampp/htdocs">

Options Indexes FollowSymLinks MultiViews

AllowOverride None

Order allow,deny

allow from all

AuthType Kerberos

KrbMethodNegotiate on

KrbMethodK5Passwd on

AuthName "Katholieke Hogeschool Mechelen Login"

KrbAuthRealms EINDWERK.LAN

Krb5Keytab /opt/lampp/etc/apache.keytab

require valid-user

</Directory>

* KrbMethodNegotiate indicates that the module will attempt to use the Negotiate mechanism to authenticate a client.
* KrbMethodK5Passwd means that the module will take the username/password pair from the client and attempt to authenticate it against a Kerberos realm.
* The AuthName parameter is a string that is shown above the login prompt shown on the client.
* KrbAuthRealms contains all the realms that will be used for authentication, separated by a space. By default, this value is the default realm of the local Kerberos configuration.
* Krb5Keytab specifies the location of the keytab file of the Apache server service. By default, it uses the default keytab specified in the local Kerberos configuration. It was overridden to give an example of a stand-alone keytab for a single service.

Other configuration options can be found on the module website:

<http://modauthkerb.sourceforge.net/configure.html>

Lastly, a new principal, HTTP/krb.khm.lan, was created and added to an external keytab called apache.keytab file using the following commands:

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

kadmin.local: ktadd -k /opt/lampp/etc/apache.keytab HTTP/krb.khm.lan

Entry for principal HTTP/krb.khm.lan with kvno 4, encryption type AES-256 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/opt/lampp/etc/apache.keytab.

Entry for principal HTTP/krb.khm.lan with kvno 4, encryption type ArcFour with HMAC/md5 added to keytab WRFILE:/opt/lampp/etc/apache.keytab.

Entry for principal HTTP/krb.khm.lan with kvno 4, encryption type Triple DES cbc mode with HMAC/sha1 added to keytab WRFILE:/opt/lampp/etc/apache.keytab.

Entry for principal HTTP/krb.khm.lan with kvno 4, encryption type DES cbc mode with CRC-32 added to keytab WRFILE:/opt/lampp/etc/apache.keytab.

The newly created keytab was then chmod’ed to an appropriate level:

sudo chmod 755 /opt/lampp/etc/apache.keytab

After these configuration steps, the XAMPP web server was restarted:

sudo /opt/lampp/lampp restart

### Client configuration

In order to automatically send an authentication token to the web server, web browsers must support the Negotiate mechanism and be configured to only send this token to a certain domain. Mozilla Firefox was used for this test, but any version of Internet Explorer above v5.0 also supports the Negotiate mechanism [TODO12].

To configure Mozilla Firefox, about:config was typed into the browser bar to access the configuration menu:

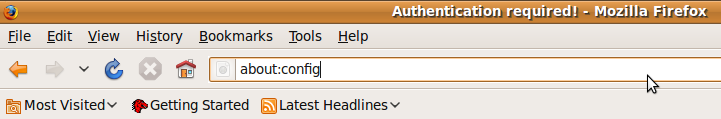


Figure TODO: Typing “about:config” in the browser bar to access the configuration menu.

After reading and confirming a warning message, the configuration menu was shown:

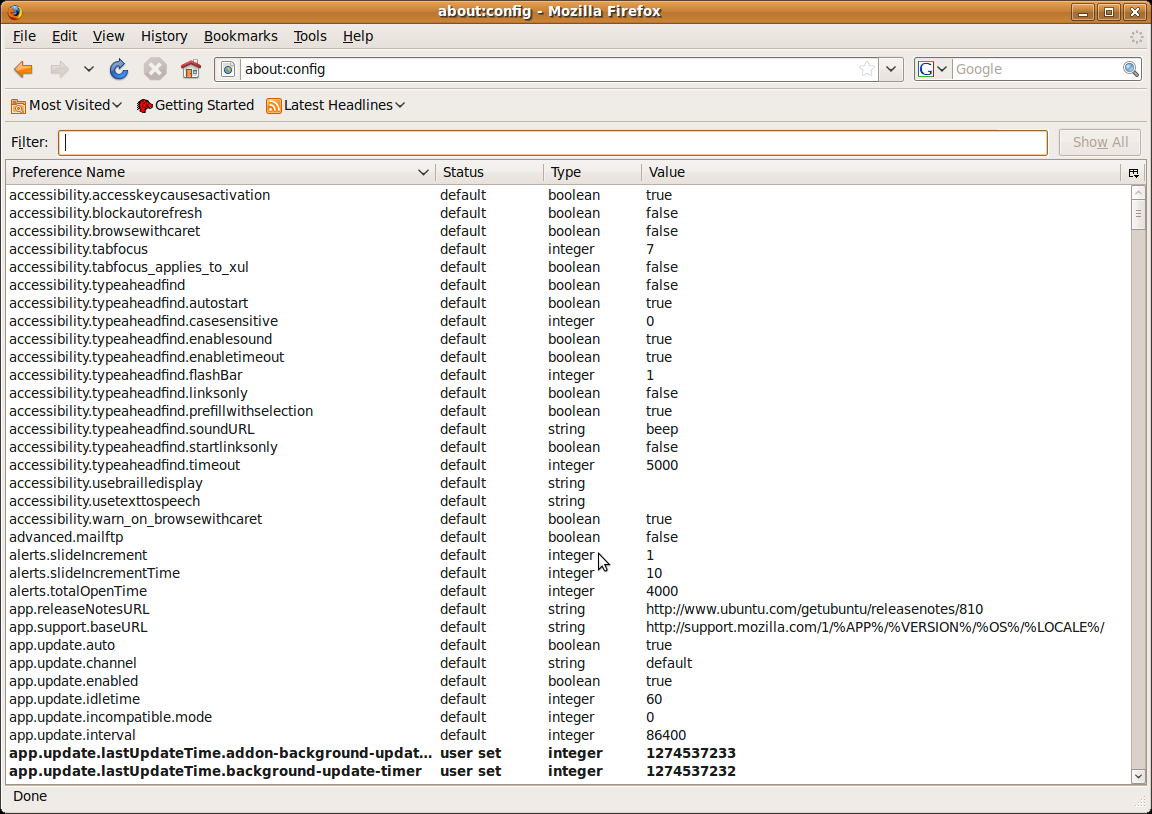


Figure TODO: Mozilla Firefox configuration menu.

The configuration items were filtered by the keyword network.negotiate-auth:

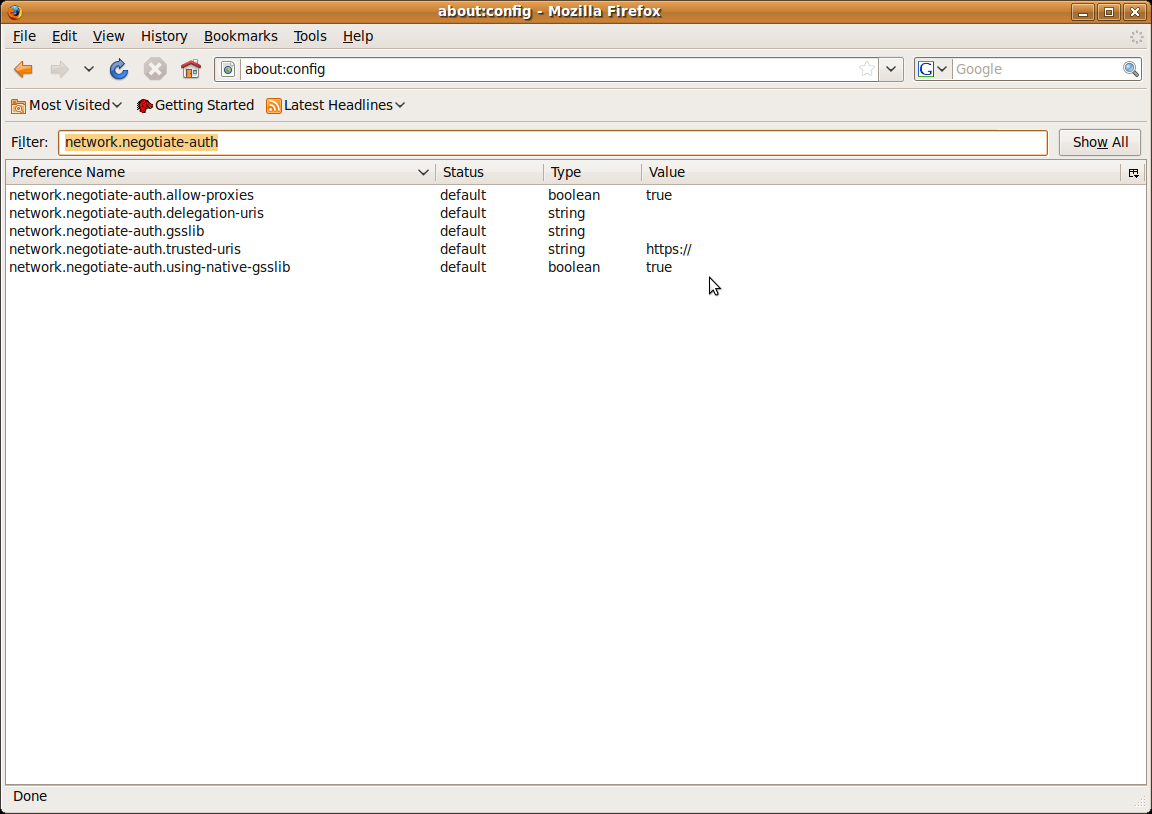


Figure TODO: All configuration items after filtering by the keyword “network.negotiate-auth”.

The values of network.negotiate-auth.trusted-uris and network.negotiate-auth.delegation-uris were changed to khm.lan.

### Testing mod\_auth\_kerb

First of all, the kdestroy and kinit commands were issued to create a new credential cache:

kdestroy

kinit

Then, the klist command was issued to confirm a valid Ticket Granting Ticket was in the credential cache of the client:

klist

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

Default principal: khmuser@KHM.LAN

Valid starting Expires Service principal

05/22/10 13:21:37 05/22/10 23:21:37 krbtgt/KHM.LAN@KHM.LAN

renew until 05/23/10 13:21:36

Mozilla Firefox was used to browse to the website www.khm.lan. The client was automatically granted access to the site, and the klist command was issued again to confirm this:

klist

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

Default principal: khmuser@KHM.LAN

Valid starting Expires Service principal

05/22/10 13:21:37 05/22/10 23:21:37 krbtgt/KHM.LAN@KHM.LAN

renew until 05/23/10 13:21:36

05/22/10 13:24:50 05/22/10 23:21:37 HTTP/krb.khm.lan@KHM.LAN

renew until 05/23/10 13:21:36

Now that the basic premise was working, a few examples for single sign-on web applications, accessible through a portal page, were created:

* A simple notepad application that was created with PHP and MySQL.
* A MediaWiki that automatically logs a user in using his authenticated username.
* A web mail interface that automatically logs a user in into his mail account.

The installation and configuration of these sample applications is explained in their subchapters.

### Benefits and drawbacks

The mod\_auth\_kerb portal is not reliant on session variables, but when using it, clients have to request a Service Ticket with the TGS each time they wish to gain access to secure resources. It is unclear if this is a bug in Mozilla Firefox or expected behavior of the mod\_auth\_kerb module. Below is a screenshot of a Wireshark capture of communication between a Mozilla Firefox client and the XAMPP web server. Note how the client needs to communicate with the TGS for each HTTP/1.1 401 it receives from the web server before sending back a GET/HTTP/1.1 with an Authentication header.

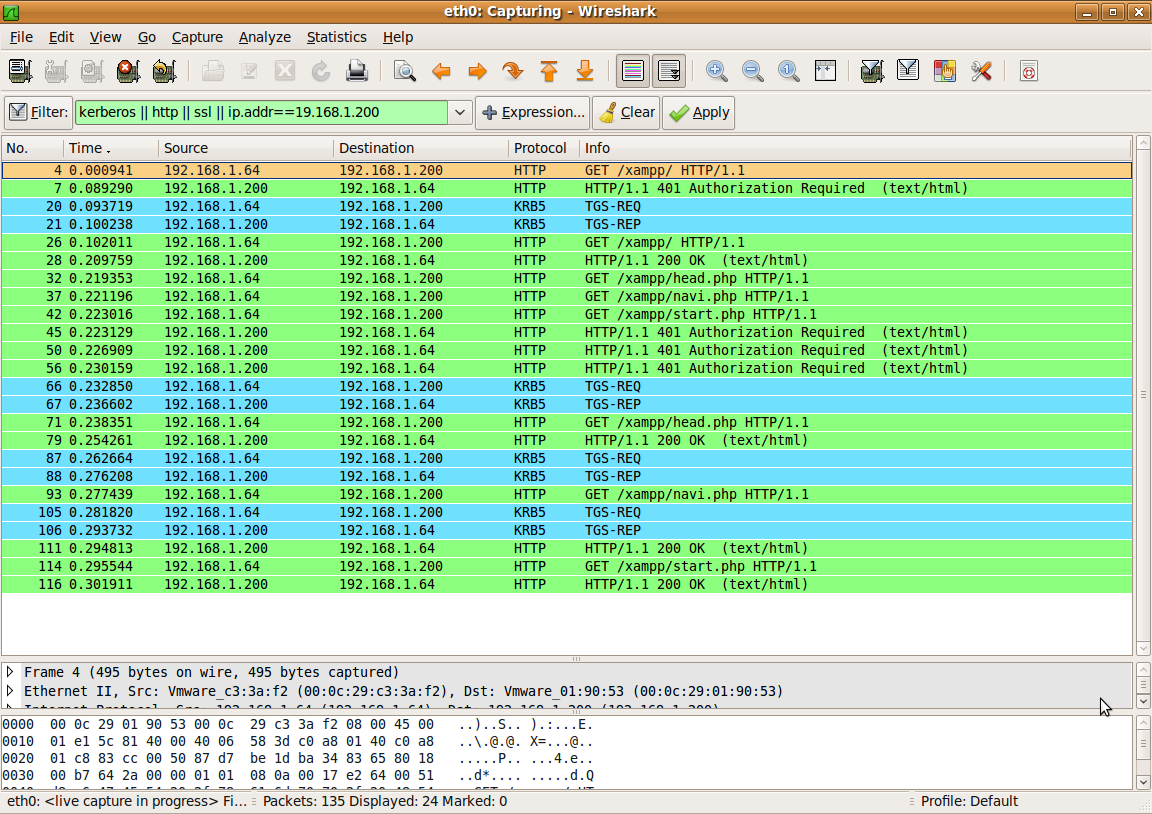


Figure TODO: Wireshark capture of client requesting four different resources from the web server.

A KDC may be overwhelmed by the many TGS requests that only a small number of clients could potentially produce. While this can be mitigated by using a Master/Slaves KDC setup, it would greatly increase the cost, complexity and vulnerability of the authentication system, ideally requiring an LDAP backend that keeps principals synchronized between the KDCs. However, redundancy would increase significantly and Denial of Service attacks would become far harder to execute [TODO17]. Balancing these tradeoffs should be done carefully by the Kerberos administrator.

## php\_krb5

php\_krb5 is a PHP module that offers the same functionality as mod\_auth\_kerb, but uses the PHP scripting language to perform its authentication, saving client credentials in memory or in a temporary folder. What sets this module apart, however, is its ability to perform GSSAPI bindings inside the PHP scripting language. This could allow a web server to perform GSSAPI authentication, acting as a delegate for the original client and contacting other services with the client’s credentials. This would allow further secure communication between the web server and other services while retaining single sign-on capabilities [TODO13, TODO14].

This thesis will utilize the ability of the module to save client credentials in a temporary folder to cut down on the amount of communication that must be made with the KDC in order to stay authenticated, prompting the user for his username and password only once. The GSSAPI features of the module also allow for more information concerning a user’s credentials, such as the remaining lifetime of the ticket, the ways the credentials may be utilized for GSSAPI security contexts and a list of OIDs (Object Identifiers) that can be used [TODO18].

As such, this module will not be used for single-sign on capabilities, but for simulating an integrated Kerberos authentication mechanism in a website. Construction of GSSAPI security contexts with other applications will not be covered, but will be investigated in the discussion chapter (rewrite? Also add chapter number).

In order to compile this PHP module, the autoconf package needs to be installed. This package dynamically creates a configuration script based on several features of the operating system.

autoconf was installed using the following command:

sudo apt-get install autoconf

### php\_krb5 installation

The php\_krb5 module was created by Moritz Bechler, who has posted a link to the second release candidate of the source code of the module on his personal blog at <http://mbechler.eenterphace.org/blog/>.

The source code was downloaded from the blog via the link [http://mbechler.eenterphace.org/  
php\_krb5-1.0rc2.tar.bz2](http://mbechler.eenterphace.org/php_krb5-1.0rc2.tar.bz2).

Then, it was compiled using the following commands:

sudo /opt/lampp/bin/phpize

sudo ./configure --with-php-config=/opt/lampp/bin/php-config

sudo make

sudo make install

The created module was then copied over to the PHP extensions directory of XAMPP, which is located at /opt/lampp/lib/php/extensions/no-debug-non-zts-20090626/:

sudo cp modules/krb5.so /opt/lampp/lib/php/extensions/no-debug-non-zts-20090626/

### php\_krb5 configuration

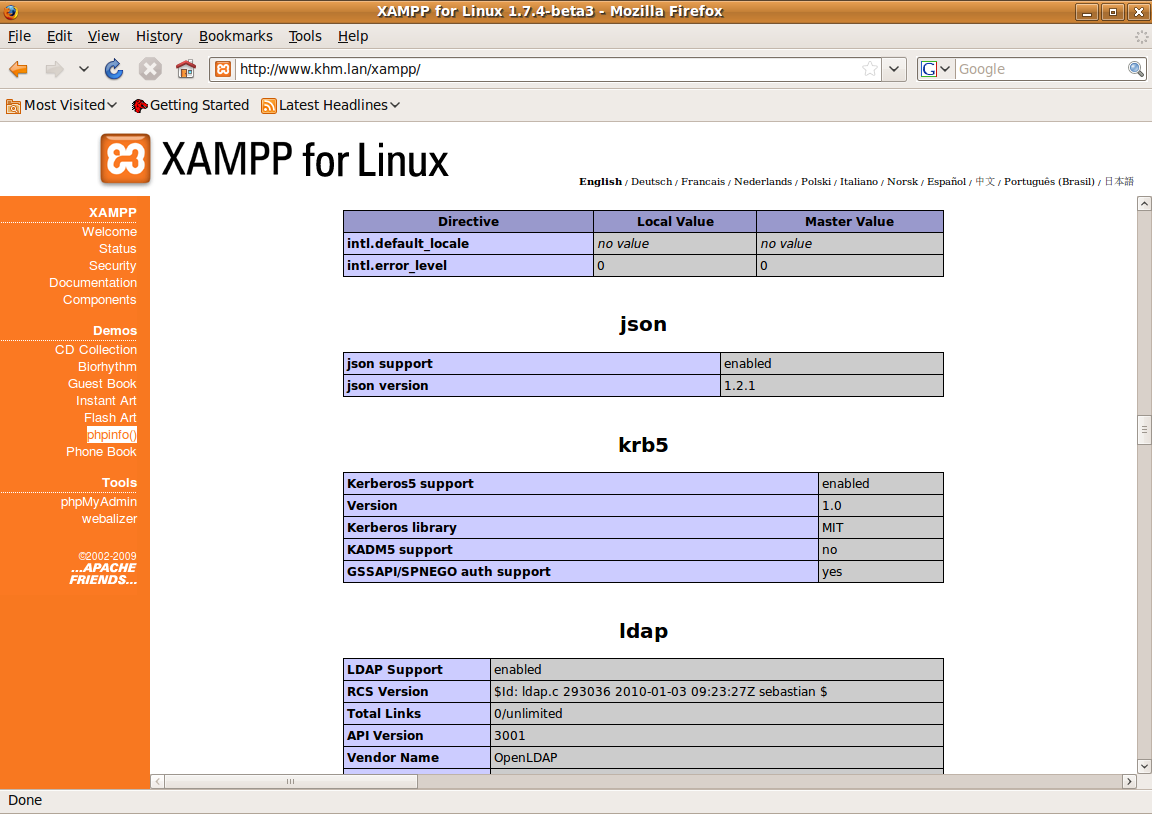
The PHP configuration file php.ini, located at /opt/lampp/etc, was modified to include the following line:

extension="krb5.so"

Then, the XAMPP web server was restarted with following command:

sudo /opt/lampp/lampp restart

Lastly, Mozilla Firefox was used to browse to the phpinfo page of XAMPP to check if the module was correctly configured and up and running:



### Testing php\_krb5

In order to test the functionality of the php\_krb5 module, a simple login page was created that posted back a username and password to a PHP script. If the username and password resulted in a valid set of credentials, the user was given access to a portal page which contained the same sample applications as the mod\_auth\_kerb portal page. The source code for this page can be found under the “Portal for X” subchapters (change this to the actual numbers).

### Benefits and drawbacks

Integrating the php\_krb5 module with a portal site is somewhat more complex compared to the relative ease of the mod\_auth\_kerb module: the source code for the php\_krb5 portal is significantly larger. However, the php\_krb5 portal does contain a few more features, such as its reduced communication with the KDC (pictured below) and the ability to automatically inform the user that his or her session will expire in a certain time period. Its potential for GSSAPI communication with other GSSAPI-enabled applications also makes it very interesting for further secure communication with other services located in the intranet of the web server.

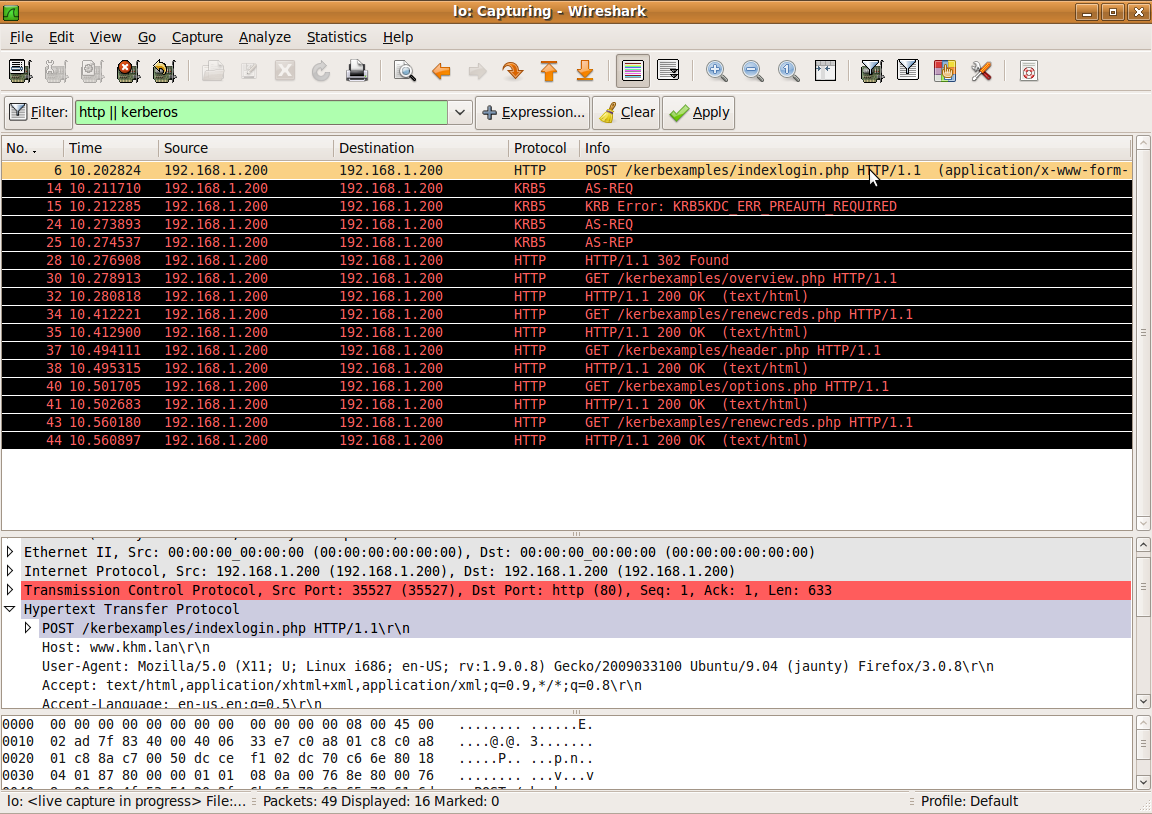


Figure TODO: Wireshark capture of the login process for the php\_krb5 portal.

However, this method of authentication also has a significant disadvantage: it cannot be used across server farms, as the credential caches are stored locally on the web server or in its memory. As such, a user can not be transferred to another web server and expect him or her to be automatically authenticated. The use of State Servers or encrypted cookies can be used to solve this problem [TODO19].

## Web portal

Both modules were outfitted with a portal page that contained three sample applications: A personal notepad, a MediaWiki and a web mail interface.

Note that, by default, only the Authentication header sent by the client is encrypted. Therefore, it is advised to use SSL encryption (“https”) for enhanced security. The source code has been modified to take this into consideration.

First, the source code for both the mod\_auth\_kerb and php\_krb5 portals will be presented. This source code was colored using the Generic Syntax Highlighter (GeSHI). Then, the installation, configuration and integration of the sample applications will be explained on a per-subchapter basis.

**Note:** This web portal was created as a proof-of-concept. It is incomplete and does not incorporate best practices such as error handling and data validation, nor does it provide protection from cross-site scripting and SQL injection. Barring those and similar methods, it should be secure: non-malignant users that have not logged in successfully will not be able to access any of the services that are available in the portal.

### Portal for mod\_auth\_kerb

The following pages contain the source code of the portal for the mod\_auth\_kerb module, and are housed under the /opt/lampp/htdocs/sso\_examples directory. Source code for modifications to sample applications can be found in the subchapters for these applications.

The source code is arranged roughly in the manner it will be executed on the web server.

index.html: This page simply redirects the user to the overview.php page.

<[**HTML**](http://december.com/html/4/element/html.html)>  
<[**HEAD**](http://december.com/html/4/element/head.html)>  
<[**TITLE**](http://december.com/html/4/element/title.html)>  
 Redirecting, please wait...  
</[**TITLE**](http://december.com/html/4/element/title.html)>  
</[**HEAD**](http://december.com/html/4/element/head.html)>  
<[**BODY**](http://december.com/html/4/element/body.html) onload="if (parent.location != location) { parent.location.href = document.location.href; }">  
<[**SCRIPT**](http://december.com/html/4/element/script.html) type="text/javascript">  
 window.location = "overview.php";  
</[**script**](http://december.com/html/4/element/script.html)>  
 If you are not redirected automatically, please click <[**a**](http://december.com/html/4/element/a.html) href="overview.php">here</[**a**](http://december.com/html/4/element/a.html)>.<[**br**](http://december.com/html/4/element/br.html)/>  
</[**BODY**](http://december.com/html/4/element/body.html)>  
</[**HTML**](http://december.com/html/4/element/html.html)>

overview.php: A frameset page with a header, which defaults to header.php, and a main page, which defaults to options.php.

**<?php**  
  
// Check user credentials  
include 'checkcreds.inc';  
  
// Show any errors  
if( $CREDENTIALSOK == -1000 ) { echo "An error occured: **\n**".$CREDENTIALCHECKERROR; [exit](http://www.php.net/exit); }  
  
// Return to main page if cache could not be found  
if( $CREDENTIALSOK == -1 ) { [header](http://www.php.net/header)('Location: index.html'); }  
  
**?>**

<[**html**](http://december.com/html/4/element/html.html)>  
  
<[**frameset**](http://december.com/html/4/element/frameset.html) rows="100,95%">  
  <[**frame**](http://december.com/html/4/element/frame.html) src="header.php" frameborder="0" noresize scrolling="no" name="header">  
  <[**frame**](http://december.com/html/4/element/frame.html) src="options.php" frameborder="0" name="main">  
  
<[**noframes**](http://december.com/html/4/element/noframes.html)>  
<[**body**](http://december.com/html/4/element/body.html)>Your browser does not handle frames!</[**body**](http://december.com/html/4/element/body.html)>  
</[**noframes**](http://december.com/html/4/element/noframes.html)>  
  
</[**frameset**](http://december.com/html/4/element/frameset.html)>  
  
</[**html**](http://december.com/html/4/element/html.html)>

checkcreds.inc: A PHP file that is included in every page that is inside the portal. It checks for the presence of the REMOTE\_USER environment variable of Apache. This code is far simpler compared to its php\_krb5 equivalent. The $CREDENTIALSOK variable can be used by the calling PHP page to make further decisions.

**<?php**  
  
if( ![isset](http://www.php.net/isset)( $\_SERVER['REMOTE\_USER'] ) )  
{  
        $CREDENTIALSOK = -1;  
}  
else  
{  
        $CREDENTIALSOK = 1;  
}  
**?>**

header.php: A header that is always visible in a strip on the top of the page. It contains the username and the realm the user is currently authenticated as.

**<?php**  
  
// Check user credentials  
include 'checkcreds.inc';  
  
// Return to main page if variable could not be found  
if( $CREDENTIALSOK == -1 ) { [header](http://www.php.net/header)('Location: index.html'); }  
  
**?>**

<[**HTML**](http://december.com/html/4/element/html.html)>  
<[**HEAD**](http://december.com/html/4/element/head.html)>  
<[**link**](http://december.com/html/4/element/link.html) href="portal\_styles.css" rel="stylesheet" type="text/css">  
</[**HEAD**](http://december.com/html/4/element/head.html)>  
<[**BODY**](http://december.com/html/4/element/body.html)>  
<[**div**](http://december.com/html/4/element/div.html) id="header">  
<[**div**](http://december.com/html/4/element/div.html) id="currentuser">

**<?php** if( $CREDENTIALSOK )  
        {  
                $userinfo = [explode](http://www.php.net/explode)( '@', $\_SERVER['REMOTE\_USER'] );  
                echo "Currently logged in as <b>".$userinfo[0]."</b><br/>**\n**";  
                echo "Realm: ".$userinfo[1]."<br/>";  
        }  
        **?>**

<[**a**](http://december.com/html/4/element/a.html) href="options.php" target="main">Back to portal</[**a**](http://december.com/html/4/element/a.html)>  
</[**div**](http://december.com/html/4/element/div.html)>  
</[**div**](http://december.com/html/4/element/div.html)>  
  
</[**BODY**](http://december.com/html/4/element/body.html)>  
</[**HTML**](http://december.com/html/4/element/html.html)>

portal\_styles.css: A simple CSS stylesheet that describes the lay-out of the header and options page.

#header  
{  
        **border-bottom**: 2px solid **black**;  
        **height**:75px;  
        **padding-bottom**: 2px;  
        **margin-bottom**: 5px;  
}  
  
#timer  
{  
        **padding**: 1px 1px 1px 1px;  
        **border**: 1px dashed **black**;  
        **float**:**right**;  
        **height**:75px;  
}  
  
.sso\_option  
{  
        **width**: 300px;  
        **height**: 300px;  
        **display**: block;  
        **margin**: 5px 5px 5px 5px;  
        **border**: 1px solid **black**;  
        **padding**: 3px 3px 3px 3px;  
        **text-align**: center;  
        **float**:**left**;  
}  
  
.sso-option-img  
{  
        **min-width**: 160px;  
        **min-height**: 160px;  
}

options.php: The main options page which gives access to three sample applications: A notepad, a MediaWiki and a web mail interface.

**<?php**  
  
// Check user credentials  
include 'checkcreds.inc';  
  
// Return to main page if cache could not be found  
if( $CREDENTIALSOK == -1 ) { [header](http://www.php.net/header)('Location: index.html'); }  
  
**?>**

<[**html**](http://december.com/html/4/element/html.html)>  
<[**head**](http://december.com/html/4/element/head.html)>  
<[**link**](http://december.com/html/4/element/link.html) href="portal\_styles.css" rel="stylesheet" type="text/css">  
</[**head**](http://december.com/html/4/element/head.html)>  
<[**body**](http://december.com/html/4/element/body.html)>  
<[**div**](http://december.com/html/4/element/div.html) id="overview">  
        <[**div**](http://december.com/html/4/element/div.html) class="sso\_option">  
        <[**a**](http://december.com/html/4/element/a.html) href="notes">  
        <[**div**](http://december.com/html/4/element/div.html) class="sso-option-img">  
                <[**img**](http://december.com/html/4/element/img.html) src="img/Notepad-icon.png" width="150" height="150" border="0" align="center" />  
        </[**div**](http://december.com/html/4/element/div.html)>  
        <[**br**](http://december.com/html/4/element/br.html)/>Notepad</[**a**](http://december.com/html/4/element/a.html)><[**br**](http://december.com/html/4/element/br.html)/>  
        Make notes in your personal notepad.  
        </[**div**](http://december.com/html/4/element/div.html)>  
  
        <[**div**](http://december.com/html/4/element/div.html) class="sso\_option">  
        <[**div**](http://december.com/html/4/element/div.html) class="sso-option-img">  
                <[**a**](http://december.com/html/4/element/a.html) href="mediawiki"><[**img**](http://december.com/html/4/element/img.html) src="img/Mediawiki.png" width="150" height="141" border="0" align="center" />  
        </[**div**](http://december.com/html/4/element/div.html)>  
        <[**br**](http://december.com/html/4/element/br.html)/>Wiki</[**a**](http://december.com/html/4/element/a.html)><[**br**](http://december.com/html/4/element/br.html)/>  
        Gain access to a local MediaWiki.  
        </[**div**](http://december.com/html/4/element/div.html)>  
  
                <[**div**](http://december.com/html/4/element/div.html) class="sso\_option">  
        <[**div**](http://december.com/html/4/element/div.html) class="sso-option-img">  
                <[**a**](http://december.com/html/4/element/a.html) href="roundcube"><[**img**](http://december.com/html/4/element/img.html) src="img/rcube\_logo.gif" style="padding:45px;" width="210" height="55" border="0" align="center" />  
        </[**div**](http://december.com/html/4/element/div.html)>  
        <[**br**](http://december.com/html/4/element/br.html)/>Webmail</[**a**](http://december.com/html/4/element/a.html)><[**br**](http://december.com/html/4/element/br.html)/>  
        Access your local webmail.  
        </[**div**](http://december.com/html/4/element/div.html)>  
</[**div**](http://december.com/html/4/element/div.html)>  
</[**body**](http://december.com/html/4/element/body.html)>  
</[**html**](http://december.com/html/4/element/html.html)>

A screenshot of the final portal page:

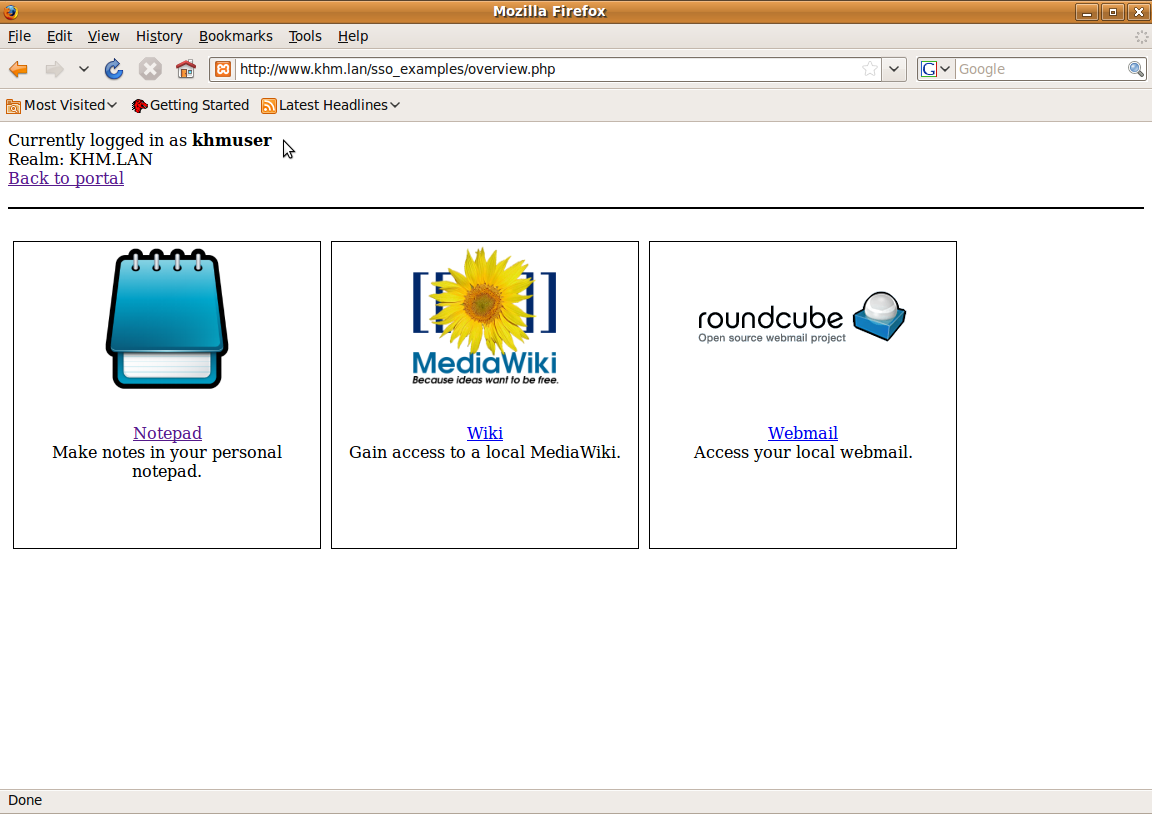


Figure TODO: A screenshot of the mod\_auth\_kerb portal page.

### Portal for php\_krb5

The next pages contain the source code for the portal of the php\_krb5 module, and are located in the /opt/lampp/htdocs/kerbexamples directory. It will become obvious that its workings are somewhat more complex than the mod\_auth\_kerb portal, which especially translates in the volume of the source code. The portal does boast a few more capabilities over the mod\_auth\_kerb module, namely:

* A one-time communication with the KDC per session.
* Ability to renew credentials.
* More credential information, the most important one being the remain lifetime of a ticket.

The source code is arranged roughly in the manner it will be executed on the web server.

index.html: This is the first page that is loaded when browsing to the directory. It asks for a username, password and realm, and posts it to a PHP script called indexlogin.php when the submit button is pressed. It also checks if it is inside a frameset. If so, it will break out of the set and reload the page.

<[**HTML**](http://december.com/html/4/element/html.html)>  
<[**HEAD**](http://december.com/html/4/element/head.html)>  
<[**TITLE**](http://december.com/html/4/element/title.html)>  
Please log in  
</[**TITLE**](http://december.com/html/4/element/title.html)>  
</[**HEAD**](http://december.com/html/4/element/head.html)>  
<[**BODY**](http://december.com/html/4/element/body.html) onload="if (parent.location != location) { parent.location.href = document.location.href; }">  
        <[**FORM**](http://december.com/html/4/element/form.html) method="post" action="indexlogin.php">  
                <[**table**](http://december.com/html/4/element/table.html)>  
                        <[**tr**](http://december.com/html/4/element/tr.html)>  
                        <[**td**](http://december.com/html/4/element/td.html)>Username:</[**td**](http://december.com/html/4/element/td.html)>  
                        <[**td**](http://december.com/html/4/element/td.html)><[**input**](http://december.com/html/4/element/input.html) type="text" size="20" name="login\_username" /></[**td**](http://december.com/html/4/element/td.html)>  
                        </[**tr**](http://december.com/html/4/element/tr.html)>  
                        <[**tr**](http://december.com/html/4/element/tr.html)>  
                        <[**td**](http://december.com/html/4/element/td.html)>Password:</[**td**](http://december.com/html/4/element/td.html)>  
                        <[**td**](http://december.com/html/4/element/td.html)><[**input**](http://december.com/html/4/element/input.html) type="password" size="20" name="login\_password" /></[**td**](http://december.com/html/4/element/td.html)>  
                        </[**tr**](http://december.com/html/4/element/tr.html)>  
                        <[**tr**](http://december.com/html/4/element/tr.html)>  
                        <[**td**](http://december.com/html/4/element/td.html)>Realm:</[**td**](http://december.com/html/4/element/td.html)>  
                        <[**td**](http://december.com/html/4/element/td.html)><[**input**](http://december.com/html/4/element/input.html) type="text" size="20" name="login\_realm" /></[**td**](http://december.com/html/4/element/td.html)>  
                        </[**tr**](http://december.com/html/4/element/tr.html)>  
                        <[**tr**](http://december.com/html/4/element/tr.html)>  
                        <[**tr**](http://december.com/html/4/element/tr.html)>  
                        <[**td**](http://december.com/html/4/element/td.html) colspan="2"><[**input**](http://december.com/html/4/element/input.html) type="submit" value="Log In"></[**td**](http://december.com/html/4/element/td.html)>  
                        </[**tr**](http://december.com/html/4/element/tr.html)>  
                </[**table**](http://december.com/html/4/element/table.html)>  
        </[**FORM**](http://december.com/html/4/element/form.html)>  
</[**BODY**](http://december.com/html/4/element/body.html)>  
</[**HTML**](http://december.com/html/4/element/html.html)>

indexlogin.php: This page is called when submitting the above HTML form. Due to either a bug in the php\_krb5 module or an incompatibility with the XAMPP web server, authentication errors could not be caught using a try…catch block. If the authentication was successful, the user is redirected to the portal overview page called overview.php. Note that a ticket lifetime of merely 30 seconds is requested when running this script. This is used to test out a JavaScript feature that shows a pop-up window when a user’s credentials will be invalid within a time span of five minutes.

**<?php**  
  
// Get values  
$username = $\_POST["login\_username"];  
$password = $\_POST["login\_password"];  
$realm = $\_POST["login\_realm"];  
  
// Create credential cache for user  
$ccache = **new** KRB5CCache();  
$flags = [array](http://www.php.net/array)('tkt\_life' => 30);  
  
// Try to authenticate  
$ccache->initPassword($username."@".$realm, $password, $flags);  
  
// Everything ok, save it to a file  
$ccache->save('FILE:/tmp/'.$username.'@'.$realm.'.ccache');  
  
// Save this user and the realm in a session variable  
[session\_start](http://www.php.net/session_start)();  
$\_SESSION['username'] = $username;  
$\_SESSION['realm'] = $realm;  
  
// Redirect user to overview page  
[header](http://www.php.net/header)("Location: overview.php");  
**?>**

overview.php: This page is very similar to the mod\_auth\_kerb overview.php page. The only difference is the error-checking code below the include statement.

**<?php**  
  
// Check user credentials  
include 'checkcreds.inc';  
  
// Show any errors  
if( $CREDENTIALSOK == -1000 ) { echo "An error occured: **\n**".$CREDENTIALCHECKERROR; [exit](http://www.php.net/exit); }  
  
// Return to main page if cache could not be found  
if( $CREDENTIALSOK == -1 ) { [header](http://www.php.net/header)('Location: index.html'); }  
  
**?>**

<[**html**](http://december.com/html/4/element/html.html)>  
  
<[**frameset**](http://december.com/html/4/element/frameset.html) rows="100,95%">  
  <[**frame**](http://december.com/html/4/element/frame.html) src="header.php" frameborder="0" noresize scrolling="no" name="header">  
  <[**frame**](http://december.com/html/4/element/frame.html) src="options.php" frameborder="0" name="main">  
  
<[**noframes**](http://december.com/html/4/element/noframes.html)>  
<[**body**](http://december.com/html/4/element/body.html)>Your browser does not handle frames!</[**body**](http://december.com/html/4/element/body.html)>  
</[**noframes**](http://december.com/html/4/element/noframes.html)>  
  
</[**frameset**](http://december.com/html/4/element/frameset.html)>  
  
</[**html**](http://december.com/html/4/element/html.html)>

checkcreds.inc: A PHP file that is included in every page that is inside the portal. It checks for the presence of the username and realm of the user, and uses it to attempt to read a matching file in the /tmp directory. If successful, the remaining lifetime of the ticket is verified to be above five minutes. If not, code to display a JavaScript pop-up window that prompts for a renewal of the ticket is written in the HTTP response.

**<?php**  
  
// Try and load the credential cache of user  
  
// Load session variables  
[session\_start](http://www.php.net/session_start)();  
  
if( ![isset](http://www.php.net/isset)($\_SESSION['username']) || ![isset](http://www.php.net/isset)($\_SESSION['realm']) )  
{  
        $CREDENTIALSOK = -1;  
}  
else  
{  
  
$USERNAMECREDENTIALCHECK = $\_SESSION['username'];  
$REALMCREDENTIALCHECK = $\_SESSION['realm'];  
  
// Load credential cache  
$CREDENTIALCHECKCACHE = **new** KRB5CCache();  
  
// The credentials are invalid by default.  
$CREDENTIALSOK = -1;  
  
try  
{  
  
$cachefile = "/tmp/**$USERNAMECREDENTIALCHECK**@**$REALMCREDENTIALCHECK**.ccache";  
  
if( [file\_exists](http://www.php.net/file_exists)( $cachefile ) )  
{  
        $CREDENTIALCHECKCACHE->open('FILE:'.$cachefile);  
        // Credentials loaded, check if they're still valid  
        if( $CREDENTIALCHECKCACHE->isValid( 300 ) )  
        {  
                // Credentials are fine.  
                $CREDENTIALSOK = 1;  
        }  
        else  
        {  
                // Credentials will need to be reloaded in a few moments.  
                $CREDENTIALSOK = 0;  
  
                // We'll need to show a pop-up window to renew our credentials, so let's find the correct path to that script  
                $renewcredslocation = $\_SERVER['DOCUMENT\_ROOT'].'/kerbexamples/renewcreds.php';  
                  
                if( [isset](http://www.php.net/isset)( $\_SERVER['HTTPS'] ) )  
                {  
                        // We're using HTTPS.  
                        $renewcredslocation = [str\_replace](http://www.php.net/str_replace)( "/opt/lampp/htdocs/", "https://".$\_SERVER['HTTP\_HOST']."/", $renewcredslocation );  
                }  
                else  
                {  
                        // We're not using HTTPS.  
                        $renewcredslocation = [str\_replace](http://www.php.net/str_replace)( "/opt/lampp/htdocs/", "http://".$\_SERVER['HTTP\_HOST']."/", $renewcredslocation );  
                }  
  
                // Display the pop-up window.             
                echo '<SCRIPT LANGUAGE="JavaScript">  
                        window.open("'.$renewcredslocation.'", "Renew Credentials", "width=350, height=300");   
                      </SCRIPT>';  
        }  
}  
else  
{  
        // We couldn't open the credential cache.  
        $CREDENTIALSOK = -1;  
}  
  
} catch (Exception $CREDENTIALCHECKERROR) { $CREDENTIALSOK = -1000; }  
  
}  
**?>**

portal\_styles.css: This CSS stylesheet is identical to the stylesheet used in the mod\_auth\_kerb portal and is thus not displayed here.

header.php: This page is always visible in a strip at the top of the page. Compared to its mod\_auth\_kerb equivalent, it does much more, using a JavaScript function to dynamically show the remaining lifetime of a ticket and displaying a pop-up window to renew a user’s credentials when needed.

**<?php**  
// Check user credentials  
include 'checkcreds.inc';  
  
// Show any errors  
if( $CREDENTIALSOK == -1000 ) { echo "An error occured: **\n**".$CREDENTIALCHECKERROR; [exit](http://www.php.net/exit); }  
  
// Return to main page if cache could not be found  
if( $CREDENTIALSOK == -1 ) { [header](http://www.php.net/header)('Location: index.html'); }  
  
**?>**<[**HTML**](http://december.com/html/4/element/html.html)>  
<[**HEAD**](http://december.com/html/4/element/head.html)>  
<[**link**](http://december.com/html/4/element/link.html) href="portal\_styles.css" rel="stylesheet" type="text/css">  
<[**script**](http://december.com/html/4/element/script.html) type="text/javascript">  
**var** ticketTimer = 0;

// Reduces the ticket time by one and checks if it goes below 300 seconds.  
// If so, check if the credentials are still valid. If not, show a pop-up window.  
**function** reduceTicket ( )  
{  
        **var** currentTicket = parseInt( document.getElementById('timeleft').innerHTML );  
  
        **if**( isNaN(currentTicket) )  
        {  
                **return** **false**;  
        }  
  
        currentTicket = currentTicket - 1;  
  
        **if**( currentTicket < 300 )  
        {

**<?php**  
                $renewcredslocation = $\_SERVER['DOCUMENT\_ROOT'].'/kerbexamples/renewcreds.php';  
                if( [isset](http://www.php.net/isset)( $\_SERVER['HTTPS'] ) )  
                        $renewcredslocation = [str\_replace](http://www.php.net/str_replace)( "/opt/lampp/htdocs/", "https://".$\_SERVER['HTTP\_HOST']."/", $renewcredslocation );  
                else  
                        $renewcredslocation = [str\_replace](http://www.php.net/str_replace)( "/opt/lampp/htdocs/", "http://".$\_SERVER['HTTP\_HOST']."/", $renewcredslocation );  
                **?>**  
                // Get the location of the script  
                **var** renewcredslocation = "**<?php** echo renewcredslocation; **?>**";  
  
                // Open window  
                window.open(renewcredslocation, "Renew Credentials", "width=350, height=300");  
  
                clearInterval( ticketTimer );  
        }  
  
        **if**( currentTicket < 0 )  
        {  
                document.getElementById("timeleft").innerHTML = "Credentials Expired.";  
                document.getElementById("currentuser").innerHTML = "Credentials Expired.";  
        }  
        **else**  
        {  
                document.getElementById("timeleft").innerHTML = currentTicket;  
        }    
}  
</[**script**](http://december.com/html/4/element/script.html)>  
</[**HEAD**](http://december.com/html/4/element/head.html)>  
<[**BODY**](http://december.com/html/4/element/body.html) onload="ticketTimer = setInterval  ( 'reduceTicket()', 1000 );"> **<?php**  
if( $CREDENTIALSOK == 1 )  
{  
        $cgssapi = **new** GSSAPIContext();  
        $cgssapi->acquireCredentials($CREDENTIALCHECKCACHE);  
        $information = $cgssapi->inquireCredentials();  
}  
**?>**<[**div**](http://december.com/html/4/element/div.html) id="header">  
<[**div**](http://december.com/html/4/element/div.html) id="timer">Time left:  
<[**div**](http://december.com/html/4/element/div.html) id="timeleft">  
        **<?php**  
        if( $CREDENTIALSOK == 1 )  
        {  
                echo $information['lifetime\_remain'];  
        }  
        else  
        {  
                echo "Credentials Expired.";  
        }  
        **?>**  
</[**div**](http://december.com/html/4/element/div.html)>  
<[**a**](http://december.com/html/4/element/a.html) href="logout.php" target="\_self">Log out</[**a**](http://december.com/html/4/element/a.html)>  
</[**div**](http://december.com/html/4/element/div.html)>  
<[**div**](http://december.com/html/4/element/div.html) id="currentuser">  
        **<?php**  
        if( $CREDENTIALSOK == 1 )  
        {  
                $userinfo = [explode](http://www.php.net/explode)( '@', $information['name'] );  
                echo "Currently logged in as <b>".$userinfo[0]."</b><br/>**\n**";  
                echo "Realm: ".$userinfo[1]."<br/>";  
        }  
        else  
        {  
                echo "Credentials Expired.<br/>";  
        }  
        **?>**  
        <[**a**](http://december.com/html/4/element/a.html) href="options.php" target="main">Back to portal</[**a**](http://december.com/html/4/element/a.html)>  
</[**div**](http://december.com/html/4/element/div.html)>  
</[**div**](http://december.com/html/4/element/div.html)>  
</[**BODY**](http://december.com/html/4/element/body.html)>  
</[**HTML**](http://december.com/html/4/element/html.html)>

options.php: This page gives an overview of the available sample applications and is very similar to its mod\_auth\_kerb counterpart. The only difference is that this version checks for errors that occurred in the checkcreds.inc script.

**<?php**  
  
// Check user credentials  
include 'checkcreds.inc';  
  
// Show any errors  
if( $CREDENTIALSOK == -1000 ) { echo "An error occured: **\n**".$CREDENTIALCHECKERROR; [exit](http://www.php.net/exit); }  
  
// Return to main page if cache could not be found  
if( $CREDENTIALSOK == -1 ) { [header](http://www.php.net/header)('Location: index.html'); }  
  
**?>**

<[**html**](http://december.com/html/4/element/html.html)>  
<[**head**](http://december.com/html/4/element/head.html)>  
<[**link**](http://december.com/html/4/element/link.html) href="portal\_styles.css" rel="stylesheet" type="text/css">  
</[**head**](http://december.com/html/4/element/head.html)>  
<[**body**](http://december.com/html/4/element/body.html)>  
<[**div**](http://december.com/html/4/element/div.html) id="overview">  
        <[**div**](http://december.com/html/4/element/div.html) class="sso\_option">  
        <[**a**](http://december.com/html/4/element/a.html) href="notes">  
        <[**div**](http://december.com/html/4/element/div.html) class="sso-option-img">  
                <[**img**](http://december.com/html/4/element/img.html) src="img/Notepad-icon.png" width="150" height="150" border="0" align="center" />  
        </[**div**](http://december.com/html/4/element/div.html)>  
        <[**br**](http://december.com/html/4/element/br.html)/>Notepad</[**a**](http://december.com/html/4/element/a.html)><[**br**](http://december.com/html/4/element/br.html)/>  
        Make notes in your personal notepad.  
        </[**div**](http://december.com/html/4/element/div.html)>  
  
        <[**div**](http://december.com/html/4/element/div.html) class="sso\_option">  
        <[**div**](http://december.com/html/4/element/div.html) class="sso-option-img">  
                <[**a**](http://december.com/html/4/element/a.html) href="mediawiki"><[**img**](http://december.com/html/4/element/img.html) src="img/Mediawiki.png" width="150" height="141" border="0" align="center" />  
        </[**div**](http://december.com/html/4/element/div.html)>  
        <[**br**](http://december.com/html/4/element/br.html)/>Wiki</[**a**](http://december.com/html/4/element/a.html)><[**br**](http://december.com/html/4/element/br.html)/>  
        Gain access to a local MediaWiki.  
        </[**div**](http://december.com/html/4/element/div.html)>  
  
                <[**div**](http://december.com/html/4/element/div.html) class="sso\_option">  
        <[**div**](http://december.com/html/4/element/div.html) class="sso-option-img">  
                <[**a**](http://december.com/html/4/element/a.html) href="roundcube"><[**img**](http://december.com/html/4/element/img.html) src="img/rcube\_logo.gif" style="padding:45px;" width="210" height="55" border="0" align="center" />  
        </[**div**](http://december.com/html/4/element/div.html)>  
        <[**br**](http://december.com/html/4/element/br.html)/>Webmail</[**a**](http://december.com/html/4/element/a.html)><[**br**](http://december.com/html/4/element/br.html)/>  
        Access your local webmail.  
        </[**div**](http://december.com/html/4/element/div.html)>  
</[**div**](http://december.com/html/4/element/div.html)>  
</[**body**](http://december.com/html/4/element/body.html)>  
</[**html**](http://december.com/html/4/element/html.html)>

renewcreds.php: This PHP page is called in a pop-up window when a user’s credentials are no longer valid (or no longer will be in a few minutes). It calls the renewcreds\_action.php script to actually renew the user’s credentials.

<[**HTML**](http://december.com/html/4/element/html.html)>  
<[**HEAD**](http://december.com/html/4/element/head.html)>  
<[**TITLE**](http://december.com/html/4/element/title.html)>  
Renew credentials  
</[**TITLE**](http://december.com/html/4/element/title.html)>  
</[**HEAD**](http://december.com/html/4/element/head.html)>  
<[**BODY**](http://december.com/html/4/element/body.html)>  
**<?php** [session\_start](http://www.php.net/session_start)(); **?>**  
        <[**FORM**](http://december.com/html/4/element/form.html) method="post" action="renewcreds\_action.php">  
        Your credentials need to be renewed. Please fill in your password.<[**br**](http://december.com/html/4/element/br.html)/>  
        If you wish to log in as another user, close this window and go to the main portal page.<[**br**](http://december.com/html/4/element/br.html)/>  
                <[**table**](http://december.com/html/4/element/table.html)>  
                        <[**tr**](http://december.com/html/4/element/tr.html)>  
                        <[**td**](http://december.com/html/4/element/td.html)>Username:</[**td**](http://december.com/html/4/element/td.html)>  
                        <[**td**](http://december.com/html/4/element/td.html)>**<?php** echo $\_SESSION['username']; **?>**</[**td**](http://december.com/html/4/element/td.html)>  
                        </[**tr**](http://december.com/html/4/element/tr.html)>  
                        <[**tr**](http://december.com/html/4/element/tr.html)>  
                        <[**td**](http://december.com/html/4/element/td.html)>Realm:</[**td**](http://december.com/html/4/element/td.html)>  
                        <[**td**](http://december.com/html/4/element/td.html)>**<?php** echo $\_SESSION['realm']; **?>**</[**td**](http://december.com/html/4/element/td.html)>  
                        </[**tr**](http://december.com/html/4/element/tr.html)>  
                        <[**tr**](http://december.com/html/4/element/tr.html)>  
                        <[**td**](http://december.com/html/4/element/td.html)>Password:</[**td**](http://december.com/html/4/element/td.html)>  
                        <[**td**](http://december.com/html/4/element/td.html)><[**input**](http://december.com/html/4/element/input.html) type="password" size="20" name="login\_password" /></[**td**](http://december.com/html/4/element/td.html)>  
                        </[**tr**](http://december.com/html/4/element/tr.html)>  
                        <[**tr**](http://december.com/html/4/element/tr.html)>  
                        <[**tr**](http://december.com/html/4/element/tr.html)>  
                        <[**td**](http://december.com/html/4/element/td.html) colspan="2"><[**input**](http://december.com/html/4/element/input.html) type="submit" value="Renew Credentials"></[**td**](http://december.com/html/4/element/td.html)>  
                        </[**tr**](http://december.com/html/4/element/tr.html)>  
                </[**table**](http://december.com/html/4/element/table.html)>  
        </[**FORM**](http://december.com/html/4/element/form.html)>  
</[**BODY**](http://december.com/html/4/element/body.html)>  
</[**HTML**](http://december.com/html/4/element/html.html)>

renewcreds\_action.php: This script takes the password of a user and uses it to renew his or her credentials. If successful, it saves the new credentials in a file in the /tmp directory, reloads the parent page and closes the pop-up window.

**<?php**  
  
// If some values are missing, return to the login page  
if( ![isset](http://www.php.net/isset)( $\_POST["login\_password"] )   
  ) { [header](http://www.php.net/header)('Location: renewcreds.php'); }  
  
// Get values  
[session\_start](http://www.php.net/session_start)();  
$username = $\_SESSION['username'];  
$password = $\_POST["login\_password"];  
$realm = $\_SESSION['realm'];  
  
// Get credential cache for user  
$ccache = **new** KRB5CCache();  
$flags = [array](http://www.php.net/array)('tkt\_lifetime' => 3600);  
  
// Try to authenticate  
try {  
        $ccache->initPassword($username."@".$realm, $password, $flags);  
} catch (Exception $error) { echo "ERROR: ".$error; [exit](http://www.php.net/exit); }  
  
// Everything ok, save it to a file  
$ccache->save('FILE:/tmp/'.$username.'@'.$realm.'.ccache');  
  
// Save this user and the realm in a session var  
$\_SESSION['username'] = $username;  
$\_SESSION['realm'] = $realm;  
  
// Reload the parent page!  
echo '<SCRIPT LANGUAGE="JavaScript">window.opener.parent.header.document.location.href=window.opener.parent.header.document.location.href;</SCRIPT>';  
  
// All done, close window.  
echo '<SCRIPT LANGUAGE="JavaScript">window.close();</SCRIPT>';  
  
**?>**

## Sample applications

As a proof-of-concept, three sample applications were created: A personal notepad, a customized MediaWiki and a customized Roundcube web mail interface. All three use either the $\_SESSION variables populated by the php\_krb5 module or the $\_SERVER[‘REMOTE\_USER’] environment variable provided by the mod\_auth\_kerb module to authenticate a user. This method of authentication will be shown to be inadequate especially for web mail purposes and anything else that requires contacting another server. Each sample application was chosen to demonstrate a certain feature:

* The personal notepad shows how web applications can be easily created in-house with single sign-on authentication in mind.
* The MediaWiki shows how existing web applications can be easily modified to work with a single sign-on feature.
* The Roundcube web mail interface displays how Kerberos single sign-on capabilities can be used to access e-mail services.

### Personal notepad

This simple application was built from the ground up, and allows a user to add, edit and delete notes. It automatically fetches the notes of the authenticated user and creates an “account” for the user if he or she does not yet exist in the MySQL database.

This application resides in the /notes directory of the portal and uses virtually the same files for both portals: only the error handling for a credential cache is not present in the mod\_auth\_kerb example.

#### MySQL commands

First, the database needs to be created. The following commands were executed in PHPMyAdmin:

CREATE DATABASE `kerbnotes`;   
  
CREATE TABLE `kerbnotes`.`users` (  
`userID` INT [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) [**AUTO\_INCREMENT**](http://search.mysql.com/search?site=refman-%35%31&q=AUTO_INCREMENT) PRIMARY KEY ,  
`user` VARCHAR( 255 ) [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) ,  
[**UNIQUE**](http://search.mysql.com/search?site=refman-%35%31&q=UNIQUE) (  
`user`  
)  
) ENGINE = MYISAM ;  
  
CREATE TABLE `kerbnotes`.`notes` (  
`noteID` INT [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) [**AUTO\_INCREMENT**](http://search.mysql.com/search?site=refman-%35%31&q=AUTO_INCREMENT) PRIMARY KEY ,  
`userID` INT [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) ,  
`note` LONGTEXT [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL)  
) ENGINE = MYISAM ;

#### Source code

Then, the following files were placed in the application directory:

index.php: This file shows a welcome message to the user and creates a new account in the MySQL database if one does not yet exist.

**<?**php  
  
// Check user credentials  
include '../checkcreds.inc';  
  
// Show any errors  
if( $CREDENTIALSOK == -1000 ) { echo "An error occured: **\n**".$CREDENTIALCHECKERROR; exit; }  
  
// Return to main page if cache could not be found  
if( $CREDENTIALSOK == -1 ) { header('Location: ../index.html'); }  
  
// Connect to database and check if user exists  
mysql\_connect("localhost","root","");  
mysql\_select\_db('kerbnotes') or die( "Unable to select database");  
  
if( $CREDENTIALSOK == 1 )  
{  
        $cgssapi = new GSSAPIContext();  
        $cgssapi->acquireCredentials($CREDENTIALCHECKCACHE);  
        $information = $cgssapi->inquireCredentials();  
}  
  
$result = mysql\_query( "SELECT \* FROM users U WHERE U.user = '".$information['name']."'" ) or die(mysql\_error());  
if( !mysql\_fetch\_array($result) )  
{  
        // This user does not yet exist.  
        mysql\_query("INSERT INTO users(user) VALUES('".$information['name']."');") or die(mysql\_error());  
}  
  
mysql\_close();  
  
**?>**

<html>  
<head>  
<link href="styles.css" rel="stylesheet" type="text/css">  
</head>  
<body>  
<div id="sidebar">  
My Notes<br/><br/>  
  
<b>Actions</b><br/>  
<a href="newnote.php">New Note</a><br/>  
<a href="viewnotes.php">View all notes</a><br/>  
  
</div>  
<div>  
<div id="main">Welcome to your personal notepad. Use the sidebar on the left to navigate.</div>  
</div>  
</body>  
</html>

newnote.php: This PHP page allows a user to create a new note. When the “Add Note” button is pressed, the form is posted to the addnote.php script.

**<?**php  
  
// Check user credentials  
include '../checkcreds.inc';  
  
// Show any errors  
if( $CREDENTIALSOK == -1000 ) { echo "An error occured: **\n**".$CREDENTIALCHECKERROR; exit; }  
  
// Return to main page if cache could not be found  
if( $CREDENTIALSOK == -1 ) { header('Location: ../index.html'); }  
  
// Connect to database  
mysql\_connect("localhost","root","");  
mysql\_select\_db('kerbnotes') or die( "Unable to select database");  
**?>**

<html>  
<head>  
<link href="styles.css" rel="stylesheet" type="text/css">  
</head>  
<body>  
<div id="sidebar">  
My Notes<br/><br/>  
  
<b>Actions</b><br/>  
<a href="newnote.php">New Note</a><br/>  
<a href="viewnotes.php">View all notes</a><br/>  
  
</div>  
<div>  
<div id="main">  
  
<BODY>  
        <h1>New note</h1>  
        <FORM method="post" action="addnote.php">  
                <?php  
  
               **<?**php  
  
                if( $CREDENTIALSOK == 1 )  
                {  
                        $cgssapi = new GSSAPIContext();  
                        $cgssapi->acquireCredentials($CREDENTIALCHECKCACHE);  
                        $information = $cgssapi->inquireCredentials();  
                }  
  
                $result = mysql\_query( "SELECT \* FROM users U WHERE U.user = '".$information['name']."'" ) or die(mysql\_error());  
                $row = mysql\_fetch\_array($result);  
                **?>**                <input type="hiden" value="**<?**php echo $row['userID']; **?>**" name="user\_id" style="display:none;" />  
  
                <table>  
                        <tr>  
                        <td>Note:</td>  
                        </tr>  
                        <tr>  
                        <td><textarea name="new\_note" cols=40 rows=10></textarea></td>  
                        </tr>  
                        <tr>  
                        <tr>  
                        <td><input type="submit" value="Add Note"></td>  
                        </tr>  
                </table>  
  
        </FORM>  
  
</div>  
</div>  
</body>  
</html>

addnote.php: A PHP script that adds a note to the list of notes of a certain user and then redirects the browser to the note overview.

**<?**php  
  
mysql\_connect("localhost","root","");  
mysql\_select\_db('kerbnotes') or die( "Unable to select database");  
  
$note = $\_POST['new\_note'];  
$userid = $\_POST['user\_id'];  
  
$note = mysql\_real\_escape\_string($note);  
mysql\_query("INSERT INTO notes(userid, note) VALUES( **$userid**, '**$note**' );");  
  
header('Location:viewnotes.php');  
  
mysql\_close();  
  
**?>**

viewnotes.php: This PHP page displays all the notes of a user, or a replacement text if he or she has currently no notes. Existing notes can be edited or deleted here.

**<?**php  
// Check user credentials  
include '../checkcreds.inc';  
  
// Show any errors  
if( $CREDENTIALSOK == -1000 ) { echo "An error occured: **\n**".$CREDENTIALCHECKERROR; exit; }  
  
// Return to main page if cache could not be found  
if( $CREDENTIALSOK == -1 ) { header('Location: ../index.html'); }  
  
// Connect to database  
mysql\_connect("localhost","root","");  
mysql\_select\_db('kerbnotes') or die( "Unable to select database");  
**?>**

<html>  
<head>  
<link href="styles.css" rel="stylesheet" type="text/css">  
</head>  
<body>  
<div id="sidebar">  
My Notes<br/><br/>  
  
<b>Actions</b><br/>  
<a href="newnote.php">New Note</a><br/>  
<a href="viewnotes.php">View all notes</a><br/>  
</div>  
<div>  
<div id="main">  
  
**<?**php  
// Show all notes  
echo "<h1>All notes</h1>";  
  
if( $CREDENTIALSOK == 1 )  
{  
        $cgssapi = new GSSAPIContext();  
        $cgssapi->acquireCredentials($CREDENTIALCHECKCACHE);  
        $information = $cgssapi->inquireCredentials();  
}  
  
$result = mysql\_query("SELECT N.\* FROM notes N, users U WHERE U.user = '".$information['name']."' AND U.userID = N.userID") or die(mysql\_error());  
  
if( mysql\_num\_rows( $result ) > 0 )  
{  
        echo "<table border='1'>";  
        echo "<tr>";  
        echo "<td>noteID</td>";  
        echo "<td>note</td>";  
        echo "</tr>";  
        while ($row = mysql\_fetch\_array($result) )  
        {  
                echo "<tr>";  
                echo "<td>";  
                echo $row['noteID'];  
                echo "</td>";  
                echo "<td>";  
                echo $row['note'];  
                echo "</td>";  
                echo "<td>";  
                echo '<form method="post" action="editnote.php"><input type="hidden" name="note\_id" value="'.$row['noteID'].'" style="display:none;"/><input type="submit" value="Edit"/></form>';  
                echo "</td>";  
                echo "<td>";  
                echo '<form method="post" action="delnote.php"><input type="hidden" name="note\_id" value="'.$row['noteID'].'" style="display:none;"/><input type="submit" value="Delete"/></form>';  
                echo "</td>";  
                echo "</tr>";  
        }  
        echo "</table>";  
}  
else  
{  
        echo "You currently have no notes.<br/>";  
}  
  
mysql\_close();  
**?>**  
  
</div>  
</div>  
</body>  
</html>

editnote.php: This page takes a note ID as an argument and loads the note in a textbox. On submit, the values are posted to the changepost.php script.

**<?**php  
// Check user credentials  
include '../checkcreds.inc';  
  
// Show any errors  
if( $CREDENTIALSOK == -1000 ) { echo "An error occured: **\n**".$CREDENTIALCHECKERROR; exit; }  
  
// Return to main page if cache could not be found  
if( $CREDENTIALSOK == -1 ) { header('Location: ../index.html'); }  
  
// Connect to database  
mysql\_connect("localhost","root","");  
mysql\_select\_db('kerbnotes') or die( "Unable to select database");  
**?>**

<html>  
<head>  
<link href="styles.css" rel="stylesheet" type="text/css">  
</head>  
<body>  
<div id="sidebar">  
My Notes<br/><br/>  
  
<b>Actions</b><br/>  
<a href="newnote.php">New Note</a><br/>  
<a href="viewnotes.php">View all notes</a><br/>  
  
</div>  
<div>  
<div id="main">  
  
<BODY>  
        <h1>New note</h1>  
        <FORM method="post" action="changenote.php">  
                **<?**php  
                if( $CREDENTIALSOK == 1 )  
                {  
                        $cgssapi = new GSSAPIContext();  
                        $cgssapi->acquireCredentials($CREDENTIALCHECKCACHE);  
                        $information = $cgssapi->inquireCredentials();  
                        $result = mysql\_query( "SELECT \* FROM users U WHERE U.user = '".$information['name']."'" ) or die(mysql\_error());  
                        $userrow = mysql\_fetch\_array($result);  
  
                        if( isset( $\_POST['note\_id'] ) )  
                        {  
                                $result = mysql\_query("SELECT \* FROM notes WHERE noteID = ".$\_POST['note\_id'].";") or die(mysql\_error());  
                                $noterow = mysql\_fetch\_array($result);  
  
                                if( $userrow['userID'] == $noterow['userID'] )  
                                {  
                                        echo '<input type="hiden" value="'.$\_POST['note\_id'].'" name="note\_id" style="display:none;" />';  
                                        echo "<table>";  
                                        echo "<tr>";  
                                        echo "<td>Note:</td>";  
                                        echo "</tr>";  
                                        echo "<tr>";  
                                        echo '<td><textarea name="edit\_note" cols=40 rows=10>';  
                                        echo $noterow['note'];  
                                        echo "</textarea></td>";  
                                        echo "</tr>";  
                                        echo "<tr>";  
                                        echo "<tr>";  
                                        echo '<td><input type="submit" value="Edit Note"></td>';  
                                        echo "</tr>";  
                                        echo "</table>";  
                                } else { echo "Invalid note specified."; }  
                        } else { echo "Invalid note specified."; }  
                }  
                **?>**  
        </FORM>  
</div>  
</div>  
</body>  
</html>

changenote.php: This script updates a note’s text based on a certain note ID and then redirects the browser to the note overview.

**<?**php  
  
mysql\_connect("localhost","root","");  
mysql\_select\_db('kerbnotes') or die( "Unable to select database");  
  
$note = $\_POST['edit\_note'];  
$noteID = $\_POST['note\_id'];  
  
$note = mysql\_real\_escape\_string($note);  
mysql\_query("UPDATE notes SET note = '**$note**' WHERE noteID = **$noteID**;");  
  
header('Location:viewnotes.php');  
  
mysql\_close();  
  
**?>**

delnote.php: This script deletes a note based on a note ID, but first checks if the authenticated user is the owner of the note.

**<?**php  
  
mysql\_connect("localhost","root","");  
mysql\_select\_db('kerbnotes') or die( "Unable to select database");  
  
$noteID = $\_POST['note\_id'];  
  
// Check user credentials  
include '../checkcreds.inc';  
  
if( $CREDENTIALSOK == 1 )  
{  
        // Check if the authenticated user is the owner of this note  
        $cgssapi = new GSSAPIContext();  
        $cgssapi->acquireCredentials($CREDENTIALCHECKCACHE);  
        $information = $cgssapi->inquireCredentials();  
  
        $result = mysql\_query( "SELECT \* FROM users U WHERE U.user = '".$information['name']."'" ) or die(mysql\_error());  
        $userrow = mysql\_fetch\_array($result);  
  
        $result = mysql\_query("SELECT \* FROM notes WHERE noteID = ".$noteID.";") or die(mysql\_error());  
        $noterow = mysql\_fetch\_array($result);  
  
        if( $userrow['userID'] == $noterow['userID'] )  
                mysql\_query("DELETE FROM notes WHERE noteID = **$noteID**");  
  
        header('Location:viewnotes.php');  
        mysql\_close();  
}  
**?>**

styles.css: A simple CSS stylesheet.

body  
{  
        width: 100%;  
        padding: 0px 0px 0px 0px;  
        margin: 0px 0px 0px 0px;  
}  
  
#sidebar  
{  
        margin-left: 20px;  
        float:left;  
        width: 250px;  
        min-height: 500px;  
        background-color: #EEE;  
        border: 1px solid black;  
        text-align: center;  
}  
  
#main  
{  
        padding: 5px 5px 5px 5px;  
        min-height: 500px;  
        margin-right: 20px;  
        margin-left: 290px;  
        background-color: lightgray;  
}

### MediaWiki

MediaWiki is the platform that the Wikimedia Foundation uses for its online encyclopedias and is used on many other websites as a collaboration tool [TODO20]. It has a very active development community and hundreds of extensions available [TODO21]. One of these extensions, Automatic REMOTE\_USER authentication, looks for the $\_SERVER[‘REMOTE\_USER’] environment variable and uses it to automatically log in a client, creating a new user account if one is not yet available [TODO22]. It can easily be modified to use other variables, which will prove to be quite useful for the php\_krb5 portal.

#### Installation and configuration

The latest version of MediaWiki was downloaded from the download page at <http://www.mediawiki.org/wiki/Download>. At the time of writing, this was v1.15.3. The package was then untarred and installed in the mod\_auth\_kerb portal with the following commands:

sudo tar xvzf mediawiki-1.15.3.tar.gz -C .

sudo cp mediawiki-1.15.3 /opt/lampp/htdocs/sso\_examples/mediawiki -R

Then, the configuration was file chmod’ed to be readable by MediaWiki:

sudo chmod a+w /opt/lampp/htdocs/sso\_examples/mediawiki/config

The following MySQL statements were executed in PHPMyAdmin to prepare the database (password expunged):

CREATE [USER](http://dev.mysql.com/doc/refman/%35%2E%31/en/information-functions.html) 'wikidb'@'localhost' IDENTIFIED BY '\*\*\*';  
GRANT USAGE ON \* . \* TO 'wikidb'@'localhost' IDENTIFIED BY '\*\*\*' WITH MAX\_QUERIES\_PER\_HOUR 0 MAX\_CONNECTIONS\_PER\_HOUR 0 MAX\_UPDATES\_PER\_HOUR 0 MAX\_USER\_CONNECTIONS 0 ;  
CREATE DATABASE [IF](http://dev.mysql.com/doc/refman/%35%2E%31/en/control-flow-functions.html) [**NOT**](http://dev.mysql.com/doc/refman/%35%2E%31/en/non-typed-operators.html) EXISTS `wikidb` ;  
GRANT ALL PRIVILEGES ON `wikidb` . \* TO 'wikidb'@'localhost';

Mozilla Firefox was used to browse to the configuration page of MediaWiki, which was located at <http://www.khm.lan/sso_examples/mediawiki/config/index.php>. The following settings were modified:

* Wiki name: KerberosWiki
* Copyright/License: Public domain
* Password of WikiSysop was changed to password of khmuser
* All e-mail functions were disabled
* DB username was changed to wikidb
* DB password was changed to match password used in the MySQL statement

Once the configuration was finished, the LocalSettings.php file was moved from /opt/lampp/htdocs/sso\_examples/mediawiki/config to /opt/lampp/htdocs/sso\_examples/  
mediawiki/:

sudo cp /opt/lampp/htdocs/sso\_examples/mediawiki/config/LocalSettings.php /opt/lampp/htdocs/sso\_examples/mediawiki/

#### Integration with mod\_auth\_kerb portal

The Automatic REMOTE USER Authentication extension was copied from the MediaWiki extension page at <http://www.mediawiki.org/wiki/Extension:AutomaticREMOTE_USER> and pasted into a file located at /mediawiki/extensions/Auth\_remoteuser.php, with the following modifications:

* The line function initUser(&$user) was changed to function initUser(&$user, $autocreate=false ).
* The line function addUser($user, $password) was changed to function addUser( $user, $password, $email='', $realname='' ).
* The lines

// Do nothing if session is valid

$user = User::newFromSession();

if (!$user->isAnon()) {

return; // User is already logged in and not anonymous.

were changed to:

// Do nothing if session is valid

$user = User::newFromSession();

if (!$user->isAnon()) {

$test\_username = $user->getName();

if( strcmp( strtolower($test\_username), strtolower( $\_SERVER['REMOTE\_USER'] ) ) == 0)

return; // User is already logged in and not anonymous.

The first two modifications were made to prevent MediaWiki warning from appearing. The last modification ensured that the MediaWiki account of a previously authenticated user was logged out and the new user logged on in his or her place. After adding the extension, a new directory called autologin was created in /mediawiki, and a new file named autologin.php was created with the following contents:

**<?**php  
/\* Optional settings \*/  
if( isset( $\_SERVER['REMOTE\_USER'] ) )  
{  
        $wgAuthRemoteuserAuthz = true;  
}  
else  
{  
        $wgAuthRemoteuserAuthz = false;  
}  
  
//$wgAuthRemoteuserMail = $\_SERVER["AUTHENTICATE\_MAIL"]; // Not going to use this  
$wgAuthRemoteuserNotify = false; /\* Do not send mail notifications \*/  
//$wgAuthRemoteuserDomain = "NETBIOSDOMAIN"; /\* Remove NETBIOSDOMAIN\ from the beginning of a IWA username \*/  
/\* User's mail domain to append to the user name to make their email address \*/  
$wgAuthRemoteuserMailDomain = "KHM.LAN"; // Change this to Kerberos realm  
// Don't let anonymous people do things...  
$wgGroupPermissions['\*']['createaccount']   = false;  
$wgGroupPermissions['\*']['read']            = false;  
$wgGroupPermissions['\*']['edit']            = false;  
  
/\* This is required for Auth\_remoteuser operation \*/  
require\_once('extensions/Auth\_remoteuser.php');  
$wgAuth = new Auth\_remoteuser();  
  
**?>**

Then, the following lines were added to the end of the LocalSettings.php configuration file:

// Load the auto-login script.  
include 'autologin/autologin.php';

Finally, Mozilla Firefox was used to browse to the sso\_examples/mediawiki directory of the website to test the feature:

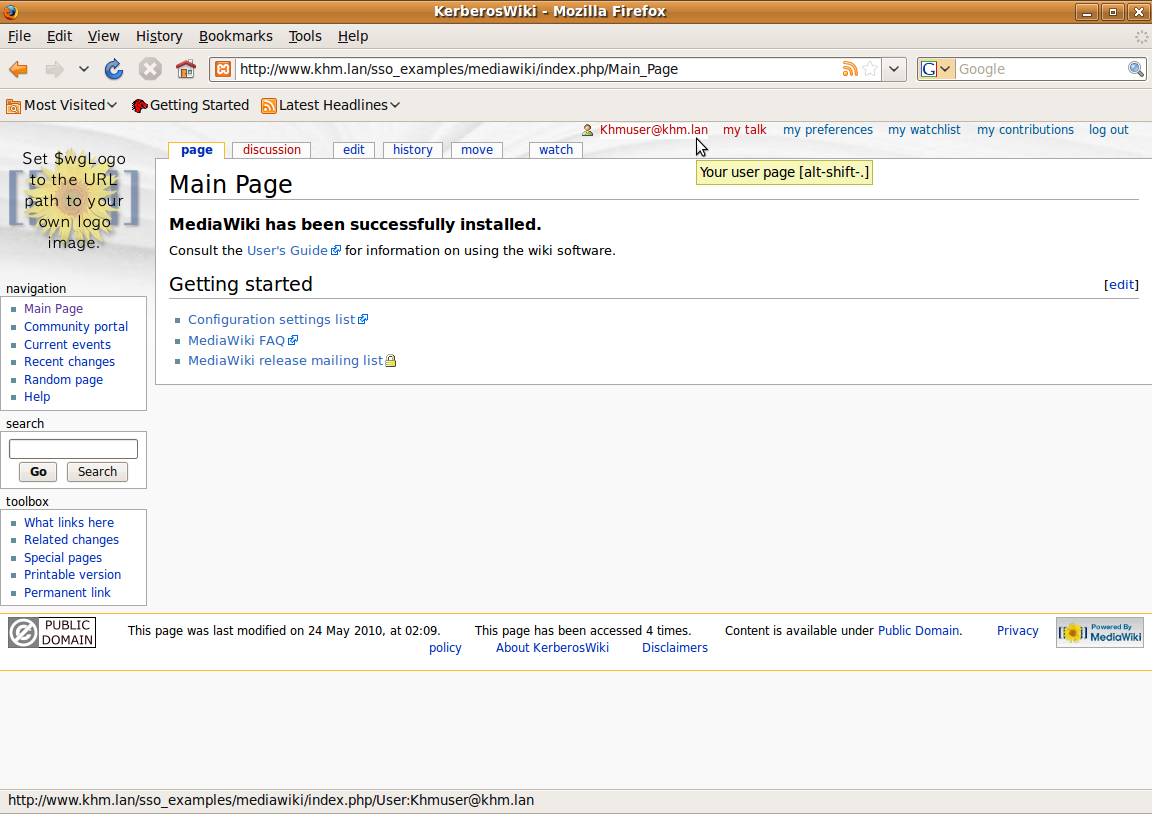


Figure TODO: Screenshot of single sign-on onto a MediaWiki.

#### Integration with php\_krb5 portal

The entire sso\_examples/mediawiki directory was copied over to the kerbexamples/mediawiki directory:

sudo cp /opt/lampp/htdocs/sso\_examples/mediawiki/ /opt/lampp/htdocs/kerbexamples/mediawiki –R

In /kerbexamples, several variables in the LocalSettings.php configuration file were changed to the following values:

$wgScriptPath = "/kerbexamples/mediawiki";

$wgCacheEpoch = gmdate( 'YmdHis' );

The $wgScriptPath variable was changed in order to prevent requests going to the sso\_examples/mediawiki directory, and $wgCacheEpoch was modified to prevent page caching as this caused issues when attempting to redirect an unauthenticated user to the login page.

Modifications to the autologin.php and Auth\_remoteuser.php scripts were also made so they would use the $\_SESSION[‘REMOTE\_USER’] session variable instead of the environment variable.

After modifications, the autologin.php script looked like this:

**<?**php  
  
// Check user credentials  
include $\_SERVER['DOCUMENT\_ROOT'].'/kerbexamples/checkcreds.inc';  
  
// Show any errors  
if( $CREDENTIALSOK == -1000 ) { echo "An error occured: **\n**".$CREDENTIALCHECKERROR; exit; }  
  
**?>**  
  
**<?**php  
  
/\* Optional settings \*/  
  
if( $CREDENTIALSOK == 1 )  
{  
        // Alright, let's get user information  
        $cgssapi = new GSSAPIContext();  
        $cgssapi->acquireCredentials($CREDENTIALCHECKCACHE);  
        $information = $cgssapi->inquireCredentials();  
  
        // Assign variables  
        $\_SESSION['REMOTE\_USER'] = $information['name'];  
        $wgAuthRemoteuserName = $information['name'];  
        $wgAuthRemoteuserAuthz = true;  
}  
elseif( $CREDENTIALSOK == 0 ) // Credentials need to be refreshed.  
{  
        $wgAuthRemoteuserAuthz = false;  
}  
else // Credentials are non-existant, send them back to portal page  
{  
        // Return to main page if cache could not be found  
        // We're using JavaScript because MediaWiki likes to cache pages  
  
        // UGLY HACK: Because of the caching, MediaWiki even remembers the PHP-values, which means it will not execute this code  
        // even when logged out. To get rid of this, pages are not cached using $wgCacheEpoch = gmdate( 'YmdHis' ); in LocalSettings.php.  
  
        $portalpage = $\_SERVER['DOCUMENT\_ROOT'].'/kerbexamples/index.html';  
        if( isset( $\_SERVER['HTTPS'] ) )  
                $portalpage = str\_replace( "/opt/lampp/htdocs/", "https://".$\_SERVER['HTTP\_HOST']."/", $portalpage );  
        else  
                $portalpage = str\_replace( "/opt/lampp/htdocs/", "http://".$\_SERVER['HTTP\_HOST']."/", $portalpage );  
  
        echo '<SCRIPT LANGUAGE="JavaScript">window.location = "'.$portalpage.'";</SCRIPT>';  
}  
  
//$wgAuthRemoteuserMail = $\_SERVER["AUTHENTICATE\_MAIL"]; // Not going to use this  
$wgAuthRemoteuserNotify = false; /\* Do not send mail notifications \*/  
//$wgAuthRemoteuserDomain = "NETBIOSDOMAIN"; /\* Remove NETBIOSDOMAIN\ from the beginning of a IWA username \*/  
/\* User's mail domain to append to the user name to make their email address \*/  
$wgAuthRemoteuserMailDomain = "KHM.LAN"; // Change this to kerberos realm  
// Don't let anonymous people do things...  
$wgGroupPermissions['\*']['createaccount']   = false;  
$wgGroupPermissions['\*']['read']            = false;  
$wgGroupPermissions['\*']['edit']            = false;  
  
/\* This is required for Auth\_remoteuser operation \*/  
require\_once('extensions/Auth\_remoteuser.php');  
$wgAuth = new Auth\_remoteuser();  
  
**?>**

In the Auth\_remoteuser.php script, every instance of $\_SERVER[‘REMOTE\_USER’] was replaced by $\_SESSION[‘REMOTE\_USER’]. Mozilla Firefox was then used to browse to the /kerbexamples/mediawiki directory of the website:

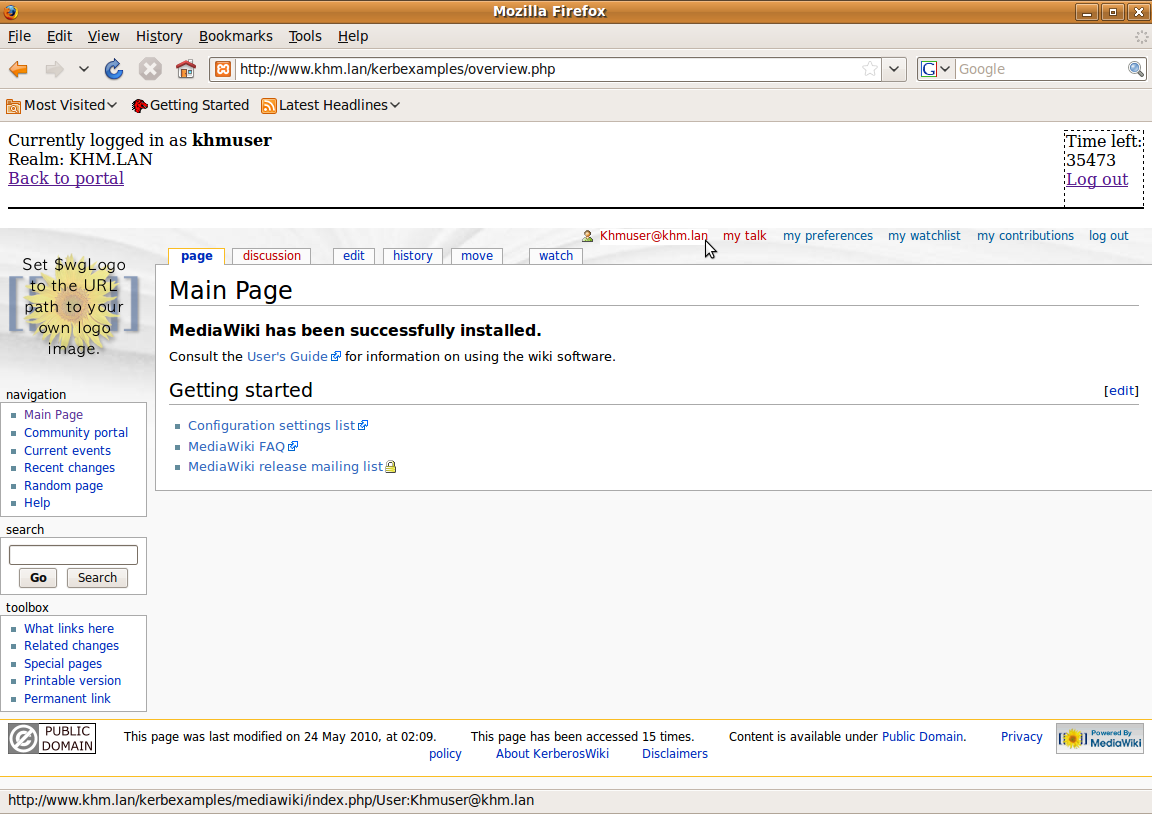


Figure TODO: Screenshot of Kerberos-based credential authentication on a MediaWiki.

### Roundcube web mail

Before being able to integrate Roundcube into the portals, a mail server must first be installed on the system. Parts of the “Ubuntu + Postfix + Courier IMAP + MySQL + Amavisd-new + SpamAssassin + ClamAV + SASL + TLS + SquirrelMail/Roundcube + Postgrey” How-To by Ivar Abrahamsen was used to install and configure the mail server [TODO23]. For more configuration details, please refer to the How-To itself.

The following components were installed and configured:

* Postfix, a Mail Transfer Agent (MTA),
* Courier, an SMTP/IMAP/POP3 daemon,
* Roundcube, a web mail interface.

#### Mail server installation

First, the Postfix packages were installed:

sudo apt-get install postfix postfix-mysql

The option “Internet Site” was chosen in the installation prompt and mail.khm.lan was used as the mail server name.

Then, the Courier packages were downloaded and installed:

sudo apt-get install courier-base courier-authdaemon courier-authlib-mysql courier-imap

When prompted for the creation of directories for web-based administration, the option “No” was chosen.

#### Mail server configuration

All packages were then configured, starting with Postfix:

1. The file /etc/mailname was confirmed to contain mail.khm.lan.
2. Several variables in the main.cf configuration file of Postfix were changed or added to contain the following values:

(Changed) myhostname = mail.khm.lan

(Changed) smtpd\_banner = $myhostname ESMTP $mail\_name

(Added) mynetworks\_style = host

(Added) local\_recipient\_maps =

(Added) mydestination =

(Added) # how long if undelivered before sending warning update to sender

(Added) delay\_warning\_time = 4h

(Added) # will it be a permanent error or temporary

(Added) unknown\_local\_recipient\_reject\_code = 450

(Added) # how long to keep message on queue before return as failed.

(Added) # some have 3 days, I have 16 days as I am backup server for some people

(Added) # whom go on holiday with their server switched off.

(Added) maximal\_queue\_lifetime = 7d

(Added) # max and min time in seconds between retries if connection failed

(Added) minimal\_backoff\_time = 1000s

(Added) maximal\_backoff\_time = 8000s

(Added) # how long to wait when servers connect before receiving rest of data

(Added) smtp\_helo\_timeout = 60s

(Added) # how many address can be used in one message.

(Added) # effective stopper to mass spammers, accidental copy in whole address list

(Added) # but may restrict intentional mail shots.

(Added) smtpd\_recipient\_limit = 16

(Added) # how many errors before back off.

(Added) smtpd\_soft\_error\_limit = 3

(Added) # how many max errors before blocking it.

(Added) smtpd\_hard\_error\_limit = 12

(Added) # not sure of the difference of the next two

(Added) # but they are needed for local aliasing

(Added) alias\_maps = hash:/etc/postfix/aliases

(Added) alias\_database = hash:/etc/postfix/aliases

(Added) # this specifies where the virtual mailbox folders will be located

(Added) virtual\_mailbox\_base = /var/spool/mail/virtual

(Added) # this is for the mailbox location for each user

(Added) virtual\_mailbox\_maps = mysql:/etc/postfix/mysql\_mailbox.cf

(Added) # and their user id

(Added) virtual\_uid\_maps = mysql:/etc/postfix/mysql\_uid.cf

(Added) # and group id

(Added) virtual\_gid\_maps = mysql:/etc/postfix/mysql\_gid.cf

(Added) # and this is for aliases

(Added) virtual\_alias\_maps = mysql:/etc/postfix/mysql\_alias.cf

(Added) # and this is for domain lookups

virtual\_mailbox\_domains = mysql:/etc/postfix/mysql\_domains.cf

(Added) # this is how to connect to the domains (all virtual, but the option is there)

(Added) # not used yet

(Added) # transport\_maps = mysql:/etc/postfix/mysql\_transport.cf

1. The file /etc/aliases was copied to /etc/postfix/aliases.
2. The postalias command was executed: sudo postalias /etc/postfix/aliases
3. The virtual mailbox was created with the command sudo mkdir /var/spool/mail/virtual.
4. The virtual group and user were added with the commands sudo groupadd virtual -g 5000 and sudo useradd virtual -u 5000 -g 5000.
5. The virtual group was made owner of the virtual mailbox with the command sudo chown -R virtual:virtual /var/spool/mail/virtual.
6. The links with the MySQL database were created by creating the following files:
   1. /etc/postfix/mysql\_mailbox.cf:

user=mail

password=[PASSWORD]

dbname=maildb

table=users

select\_field=maildir

where\_field=id

hosts=127.0.0.1

additional\_conditions = and enabled = 1

* 1. /etc/postfix/mysql\_uid.cf

user=mail

password=[PASSWORD]

dbname=maildb

table=users

select\_field=uid

where\_field=id

hosts=127.0.0.1

* 1. /etc/postfix/mysql\_gid.cf

user=mail

password=[PASSWORD]

dbname=maildb

table=users

select\_field=gid

where\_field=id

hosts=127.0.0.1

* 1. /etc/postfix/mysql\_alias.cf

user=mail

password=[PASSWORD]

dbname=maildb

table=aliases

select\_field=destination

where\_field=mail

hosts=127.0.0.1

additional\_conditions = and enabled = 1

* 1. /etc/postfix/mysql\_domains.cf

user=mail

password=[PASSWORD]

dbname=maildb

table=domains

select\_field=domain

where\_field=domain

hosts=127.0.0.1

additional\_conditions = and enabled = 1

1. The MySQL database and tables were created with the following commands:

CREATE DATABASE maildb;  
GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP ON maildb.\* TO 'mail'@'localhost' IDENTIFIED by '[PASSWORD]';  
GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP ON maildb.\* TO 'mail'@'**%**' IDENTIFIED by '[PASSWORD]';

USE `maildb`;

CREATE TABLE `aliases` (  
`pkid` smallint(3) [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) [**auto\_increment**](http://search.mysql.com/search?site=refman-%35%31&q=AUTO_INCREMENT),  
`mail` varchar(120) [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) default '',  
`destination` varchar(120) [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) default '',  
`enabled` tinyint(1) [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) default '1',  
PRIMARY KEY  (`pkid`),  
[**UNIQUE**](http://search.mysql.com/search?site=refman-%35%31&q=UNIQUE) KEY `mail` (`mail`)  
) ;  
  
CREATE TABLE `domains` (  
`pkid` smallint(6) [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) [**auto\_increment**](http://search.mysql.com/search?site=refman-%35%31&q=AUTO_INCREMENT),  
`domain` varchar(120) [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) default '',  
`transport` varchar(120) [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) default 'virtual:',  
`enabled` tinyint(1) [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) default '1',  
PRIMARY KEY  (`pkid`)  
) ;  
  
CREATE TABLE `users` (  
`id` varchar(128) [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) default '',  
`name` varchar(128) [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) default '',  
`uid` smallint(5) [**unsigned**](http://search.mysql.com/search?site=refman-%35%31&q=UNSIGNED) [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) default '5000',  
`gid` smallint(5) [**unsigned**](http://search.mysql.com/search?site=refman-%35%31&q=UNSIGNED) [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) default '5000',  
`home` varchar(255) [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) default '/var/spool/mail/virtual',  
`maildir` varchar(255) [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) default 'blah/',  
`enabled` tinyint(3) [**unsigned**](http://search.mysql.com/search?site=refman-%35%31&q=UNSIGNED) [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) default '1',  
`change**\_**password` tinyint(3) [**unsigned**](http://search.mysql.com/search?site=refman-%35%31&q=UNSIGNED) [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) default '1',  
`clear` varchar(128) [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) default 'ChangeMe',  
`crypt` varchar(128) [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) default 'sdtrusfX0Jj66',  
`quota` varchar(255) [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) default '',  
`procmailrc` varchar(128) [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) default '',  
`spamassassinrc` varchar(128) [**NOT NULL**](http://search.mysql.com/search?site=refman-%35%31&q=NOT%20NULL) default '',  
PRIMARY KEY  (`id`),  
[**UNIQUE**](http://search.mysql.com/search?site=refman-%35%31&q=UNIQUE) KEY `id` (`id`)  
) ;

1. Finally, the MySQL configuration file, located at /opt/lampp/etc/my.cnf, was modified to contain the following line:

bind-address = 127.0.0.1

Then, Courier was configured:

1. The Courier authentication daemon was modified in the following manner:

(Changed) authmodulelist="authmysql"

1. The Courier MySQL daemon was modified to contain the following:

(Changed) MYSQL\_USERNAME mail

(Changed) MYSQL\_PASSWORD [PASSWORD]

(Changed) MYSQL\_DATABASE maildb

(Changed) MYSQL\_USER\_TABLE users

(Uncommented, changed) MYSQL\_MAILDIR\_FIELD concat(home,'/',maildir)

(Uncommented, changed) MYSQL\_WHERE\_CLAUSE enabled=1

(Uncommented, changed) MYSQL\_SOCKET /opt/lampp/var/mysql/mysql.sock

**Note:** Ensure that there is no whitespace to the left of each variable.

After configuring Postfix and Courier, SQL test data was added in PHPMyAdmin:

USE `maildb`;

INSERT INTO domains (domain) VALUES  
        ('blobber.org'),  
        ('mail.blobber.org'),  
        ('whopper.nu'),  
        ('lala.com');  
  
INSERT INTO aliases (mail,destination) VALUES  
        ('@lala.com','@whupper.nu'),  
        ('@mail.blobber.org','@blobber.org'),  
        ('postmaster@whopper.nu','postmaster@localhost'),  
        ('abuse@whopper.nu','abuse@localhost'),  
        ('postmaster@blobber.org','postmaster@localhost'),  
        ('abuse@blobber.org','abuse@localhost');  
  
INSERT INTO users (id,name,maildir,crypt) VALUES   
        ('xandros@blobber.org','xandros','xandros/', [encrypt](http://dev.mysql.com/doc/refman/%35%2E%31/en/encryption-functions.html)('[PASSWORD]') ),  
        ('vivita@blobber.org','vivita','vivita/', [encrypt](http://dev.mysql.com/doc/refman/%35%2E%31/en/encryption-functions.html)('[PASSWORD]') );  
  
INSERT INTO aliases (mail,destination) VALUES  
        ('xandros@blobber.org','xandros@blobber.org'),  
        ('vivita@blobber.org','vivita@blobber.org');  
  
INSERT INTO aliases (mail,destination) VALUES  
                ('@whopper.nu','xandros@blobber.org');  
  
INSERT INTO aliases (mail,destination) VALUES  
        ('karl@blobber.org','beerdude26@hotmail.com');

Finally, all services were restarted:

sudo /etc/init.d/postfix restart

sudo /etc/init.d/courier-authdaemon restart

sudo /etc/init.d/courier-imap restart

#### Testing the mail services

Issuing the nmap localhost command gave the following output:

Starting Nmap 4.76 ( http://nmap.org ) at 2010-05-23 21:31 PDT

Warning: Hostname localhost resolves to 2 IPs. Using 127.0.0.1.

Interesting ports on localhost (127.0.0.1):

Not shown: 987 closed ports

PORT STATE SERVICE

21/tcp open ftp

22/tcp open ssh

23/tcp open telnet

25/tcp open smtp

53/tcp open domain

80/tcp open http

143/tcp open imap

443/tcp open https

544/tcp open kshell

631/tcp open ipp

749/tcp open kerberos-adm

2105/tcp open eklogin

3306/tcp open mysql

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds

The SMTP and IMAP ports were open. telnet was used to test the functionality of SMTP first:

telnet localhost 25

Trying ::1...

telnet: connect to address ::1: Connection refused

Trying 127.0.0.1...

Connected to localhost (127.0.0.1).

Escape character is '^]'.

220 mail.khm.lan ESMTP Postfix

EHLO mail.khm.lan

250-mail.khm.lan

250-PIPELINING

250-SIZE 10240000

250-VRFY

250-ETRN

250-STARTTLS

250-ENHANCEDSTATUSCODES

250-8BITMIME

250 DSN

MAIL FROM: khmuser@khm.lan

250 2.1.0 Ok

RCPT TO: xandros@blobber.org

250 2.1.5 Ok

data

354 End data with <CR><LF>.<CR><LF>

This is a test e-mail.

.

250 2.0.0 Ok: queued as 2D16C100111

quit

221 2.0.0 Bye

Connection closed by foreign host.

The virtual mailbox directory, located at /var/spool/mail/virtual, was checked to see if the mailbox for the user xandros had been created, and if the mail had arrived:

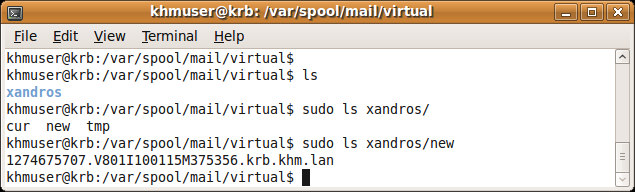


Figure TODO: Screenshot of the newly created mailbox for the user xandros.

Then, IMAP was tested with telnet:

telnet localhost 143

Trying ::1...

Connected to localhost (::1).

Escape character is '^]'.

\* OK [CAPABILITY IMAP4rev1 UIDPLUS CHILDREN NAMESPACE THREAD=ORDEREDSUBJECT THREAD=REFERENCES SORT QUOTA IDLE ACL ACL2=UNION] Courier-IMAP ready. Copyright 1998-2008 Double Precision, Inc. See COPYING for distribution information.

a login xandros@blobber.org [PASSWORD]

a OK LOGIN Ok.

The /var/log/mail.log file was consulted to confirm that IMAP was working correctly:

May 23 21:48:05 krb imapd: Connection, ip=[::1]

May 23 21:48:17 krb authdaemond: received auth request, service=imap, authtype=login

May 23 21:48:17 krb authdaemond: authmysql: trying this module

May 23 21:48:17 krb imapd: LOGIN, user=xandros@blobber.org, ip=[::1], port=[32907], protocol=IMAP

May 23 21:48:17 krb authdaemond: authmysqllib: connected. Versions: header 50067, client 50075, server 50146

May 23 21:48:17 krb authdaemond: SQL query: SELECT id, crypt, "", uid, gid, home, concat(home,'/',maildir), "", name, "" FROM users WHERE id = 'xandros@blobber.org'

May 23 21:48:17 krb authdaemond: password matches successfully

May 23 21:48:17 krb authdaemond: authmysql: sysusername=<null>, sysuserid=5000, sysgroupid=5000, homedir=/var/spool/mail/virtual, address=xandros@blobber.org, fullname=xandros, maildir=/var/spool/mail/virtual/xandros/, quota=<null>, options=<null>

May 23 21:48:17 krb authdaemond: authmysql: clearpasswd=<null>, passwd=Q.f7dVhHCLJGI

May 23 21:48:17 krb authdaemond: Authenticated: sysusername=<null>, sysuserid=5000, sysgroupid=5000, homedir=/var/spool/mail/virtual, address=xandros@blobber.org, fullname=xandros, maildir=/var/spool/mail/virtual/xandros/, quota=<null>, options=<null>

May 23 21:48:17 krb authdaemond: Authenticated: clearpasswd=[PASSWORD], passwd=Q.f7dVhHCLJGI

#### Roundcube introduction

Roundcube is an open-source project to build an extensible web mail interface with similar functionalities to other web mail interfaces such as Outlook Web Access, and works very well with Postfix and Courier.

While Cyrus and Postfix both support GSSAPI authentication [TODO25], a user’s credentials cannot be forwarded from a web mail interface to the IMAP server to perform authentication without patches to the source code of several applications [TODO24, TODO26, TODO27] . In addition, many of these patches are outdated and no longer work on more recent versions of the applications. This situation results in using a workaround or “hack” to achieve single-sign on capabilities for a web mail interface, namely using a static, secret password for all intranet users and the authenticated user’s username to log onto the web mail interface. Of course, this workaround is far from ideal. Possible solutions will be discussed in the discussion chapter. For now, however, this workaround will be used to illustrate the point of using user credentials for cross-service authentication and communication.

#### Roundcube installation

The latest version of Roundcube was downloaded from the Roundcube website (<http://roundcube.net/>), which at the time of writing was v0.4 Beta. It was untarred and copied to the mod\_auth\_kerb portal using the following commands:

sudo tar xvzf roundcubemail-0.4-beta.tar.gz -C .

sudo cp roundcubemail-0.4-beta /opt/lampp/htdocs/sso\_examples/roundcube –R

File permissions for the /temp and /logs directories were given:

sudo chown nobody /opt/lampp/htdocs/sso\_examples/roundcube/logs -R

sudo chown nobody /opt/lampp/htdocs/sso\_examples/roundcube/temp -R

Using PHPMyAdmin, a MySQL user and assorted database was created:

CREATE [USER](http://dev.mysql.com/doc/refman/%35%2E%31/en/information-functions.html) 'roundcubemail'@'localhost' IDENTIFIED BY '\*\*\*';  
GRANT USAGE ON \* . \* TO 'roundcubemail'@'localhost' IDENTIFIED BY '\*\*\*' WITH MAX\_QUERIES\_PER\_HOUR 0 MAX\_CONNECTIONS\_PER\_HOUR 0 MAX\_UPDATES\_PER\_HOUR 0 MAX\_USER\_CONNECTIONS 0 ;  
CREATE DATABASE [IF](http://dev.mysql.com/doc/refman/%35%2E%31/en/control-flow-functions.html) [**NOT**](http://dev.mysql.com/doc/refman/%35%2E%31/en/non-typed-operators.html) EXISTS `roundcubemail` ;  
GRANT ALL PRIVILEGES ON `roundcubemail` . \* TO 'roundcubemail'@'localhost';

Then, Mozilla Firefox was used to browse to the Roundcube installer at [http://www.khm.lan/  
sso\_examples/roundcube/installer/](http://www.khm.lan/sso_examples/roundcube/installer/). All default installation options were kept, except for product\_name which was changed to KerbCube Webmail, and the IMAP and SMTP servers, both of which were set to localhost. The MySQL connection options were also modified to reflect the correct user/password pair. The “CREATE CONFIG” button was clicked and the files main.inc.php and db.inc.php were created and filled with the provided configuration. When this was done, the “CONTINUE” button was clicked. The “Initialize Database” was clicked and the SMTP and IMAP testing tools were used to confirm that everything was working correctly:

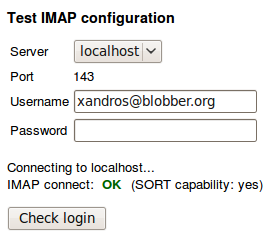
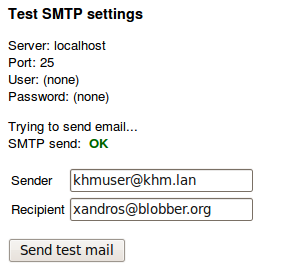


Figure TODO: Results of SMTP and IMAP testing tools.

# 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>

Todo09 <http://www.apachefriends.org/en/xampp-linux.html#374>

Todo10 <http://www.apachefriends.org/en/xampp-linux.html#381>

Todo11 <http://tools.ietf.org/html/rfc4559>

Todo12 <http://msdn.microsoft.com/en-us/library/ms995329>

Todo13 <http://mbechler.eenterphace.org/blog/index.php?/archives/10-php_krb5-release-candidate.html>

Todo14  [http://tools.ietf.org/html/rfc2744.html](http://www.faqs.org/rfcs/rfc2744.html)

Todo15 <http://modauthkerb.sourceforge.net/configure.html#saving>

Todo16 <http://thread.gmane.org/gmane.comp.apache.mod-auth-kerb.general/978>

Todo17 <http://docs.sun.com/app/docs/doc/816-4557/setup-9?a=view>

Todo18 php\_krb5 manual

Todo19 <http://en.wikipedia.org/wiki/Load_balancing_(computing)>

Todo20 <http://www.mediawiki.org/wiki/MediaWiki>

Todo21 <http://www.mediawiki.org/wiki/Category:All_extensions>

Todo22 <http://www.mediawiki.org/wiki/Extension:AutomaticREMOTE_USER>

Todo23 <http://flurdy.com/docs/postfix/>

Todo24 <http://fixunix.com/kerberos/60293-webmail-gssapi-authentication-imapd.html>

Todo25 <http://www.codealias.info/technotes/single_sign-on_e-mail_solution_with_gssapi_kerberos_authentication>

Todo26 <http://cyrusimap.web.cmu.edu/twiki/bin/view/Cyrus/SquirrelMailKerberos>

Todo27 <http://www.mail-archive.com/cosign-discuss@lists.sourceforge.net/msg00044.html>

[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

[14] “Kerberons on Ubuntu”, <http://www.alittletooquiet.net/text/kerberos-on-ubuntu/>,