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

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."

1. Table of contents

2. Prelude

This thesis is the result of our research regarding Kerberos 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 chat rooms who were ready to assist us with our problems. Those sources of aide were extremely useful to us, not to even 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, as we believe we can reach more people that way.

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

<div align="right">

Frederik Van Den Hof
Beerend Lauwers

</div>

3. Abstract

In many companies, employees encounter password prompts quite regularly, be it to gain access to the company intranet, to view their webmail, to gain access to a fileserver, etc. It can be agreed upon that this endless stream of password prompts is more than just a nuisance and proves to be detrimental for the company's overall efficiency.

The solution to this problem lies in single sign-on methods. The idea behind single sign-on is to channel all authentication requests through one central framework and henceforth allow this framework to handle the authentication in the user's place.

The goal of this thesis is to explain and document the entire installation and configuration process of a single sign-on enabled network. To achieve this goal the MIT Kerberos v5 distribution was chosen, considering this is the most widespread open source single sign-on authentication protocol currently available.

Step by step instructions accompanied by a proper explanation are available throughout the thesis, which guide the reader in setting up his or her own Kerberos single sign-on enabled network. Several examples of Kerberos enabled services as well as examples of Kerberos enabled web applications are given. A limited integration with a Windows 2003 Active Directory domain is also done. Upon having thoroughly read this thesis, the reader will have obtained an adequate amount of knowledge to set up and manage his or her own Kerberos enabled network and expand upon this.

Furthermore, the possibilities for future web based single sign-on portals are discussed, with the very recently released `php_krb5` PHP module allowing for some very promising scenarios with its built-in support for GSSAPI security contexts. Further integration with Active Directory domains is also discussed, suggesting mixed work environments where an Active Directory user can login onto a Linux workstation and vice versa.

It can be concluded that the MIT Kerberos v5 implementation of Kerberos allows for an efficient and secure way of single sign-on. However, the importance of the Key Distribution Center means it is a single point of failure unless several master Key Distribution Centers are used. The largest drawback, however, is that the user himself proves to be a liability. The user proves to be the weakest link and a strict security policy needs to be in place to compensate for this.

4. 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.

5. Types of Single Sign-on

When looking at Single Sign-on systems they can be divided in two main groups, namely the client based and the server based ones

5.1. 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 having employees bring their authentication data with them everywhere they go on a USB stick, but this just creates further complications.

One big advantage of this method, however, is that once the application has been correctly configured, immediate access to multiple sites and applications that are not interconnected can be gained.

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 an example can be found of the Novell SecureLogin Single Sign-on tool **[1]**. Two screenshots are depicted on which the prompts presented to the end-user on the first time he authenticates to a new site can be seen.
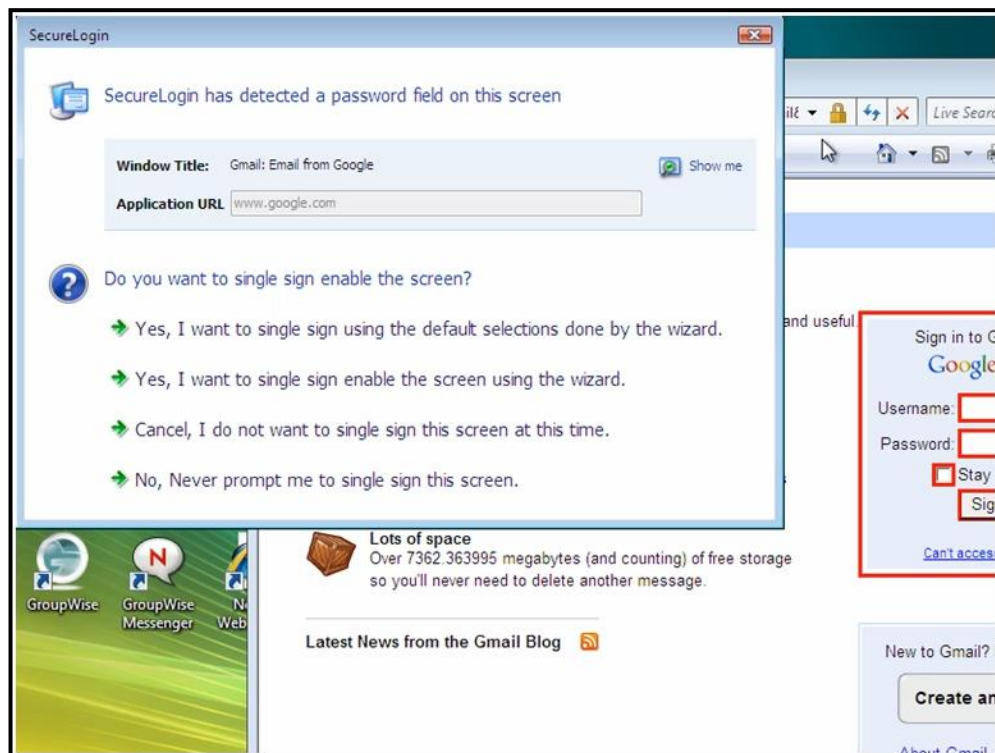
Figure 5.1: Screenshot of Novell SecureLogin.
Source: http://www.novell.com/products/securelogin/media.html
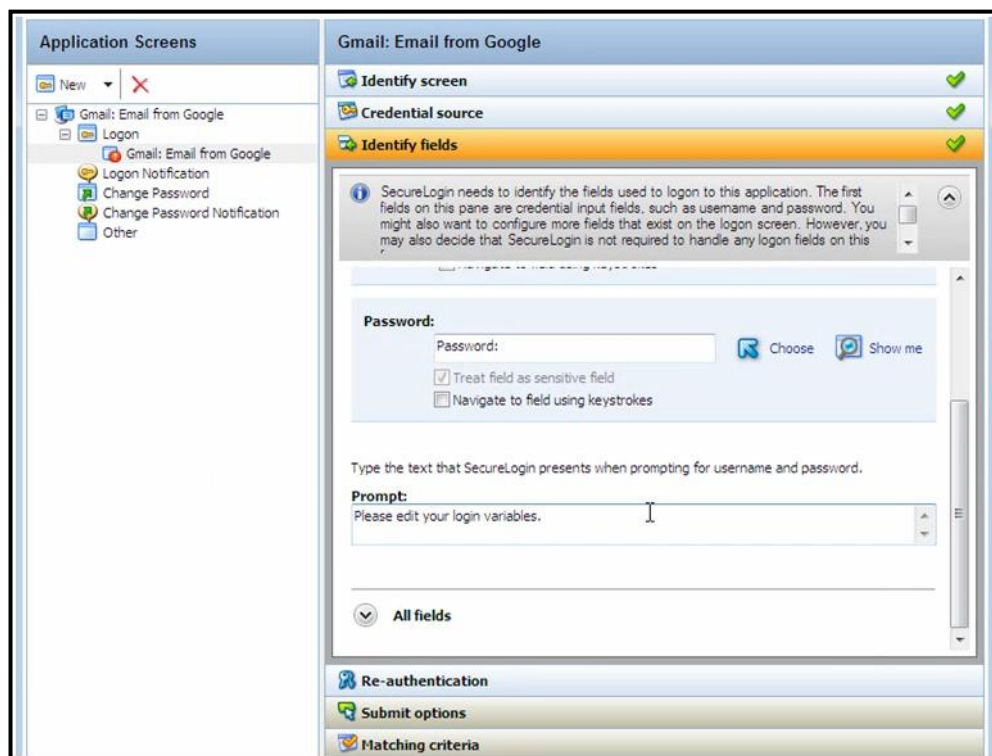
Figure 5.2: Screenshot of Novell SecureLogin configuration screen.
Source: http://www.novell.com/products/securelogin/media.html

5.2. 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 an employee has logged in to the authentication server he will possess a ticket. Whenever he tries to log in to a service, he will send that ticket to the service. Optionally, the service will crosscheck the ticket with the central authentication server to see if the employee has access to that service. Regardless, the employee will be authenticated automatically.

It can be compared to a ski resort: after paying for a ski pass, access is acquired to all the slopes.

Once the Single Sign-on system is in place an employee can for instance log into his workstation and from that point on 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 anything. He will just as well be logged in with the same smooth transition when he connects to the intranet web server, 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.

6. 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. In other words, Kerberos is an authentication protocol for trusted hosts on untrusted networks **[2]**. 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.

6.1. 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 **[3]**.

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.

6.2. 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 [2]:
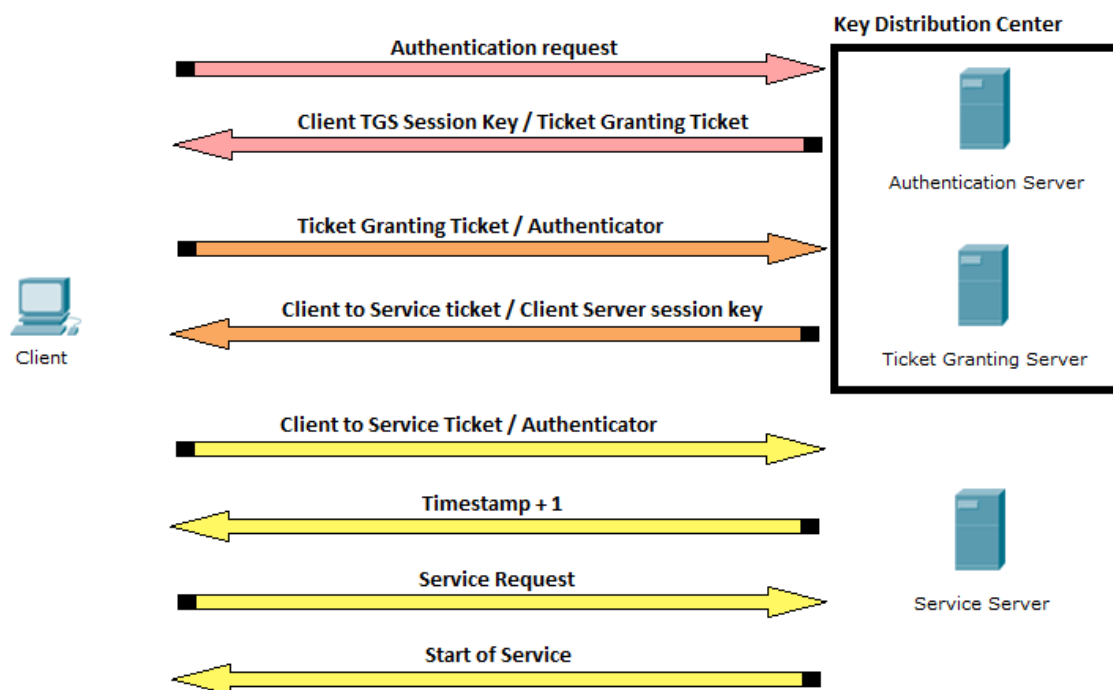
Figure 6.1: The Kerberos authentication steps.

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.
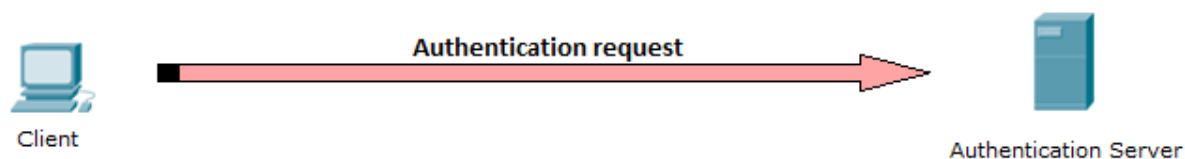
Figure 6.2: Step 1 and 2

3. 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.
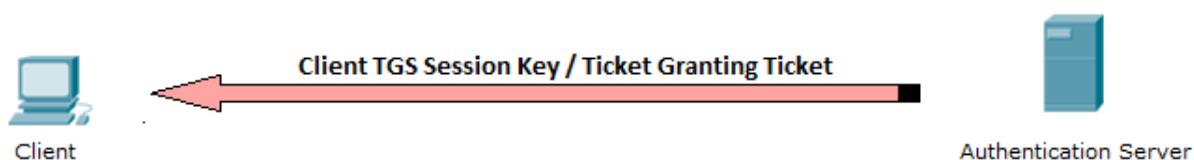
Figure 6.3: Step 3

4. The user is prompted for his password. The client then generates a client secret key based on the username and password of the user.
5. 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.
6. The user attempts to access a service (e.g. a FTP server).
7. 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 6.4: Step 7

8. 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.

9. 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.
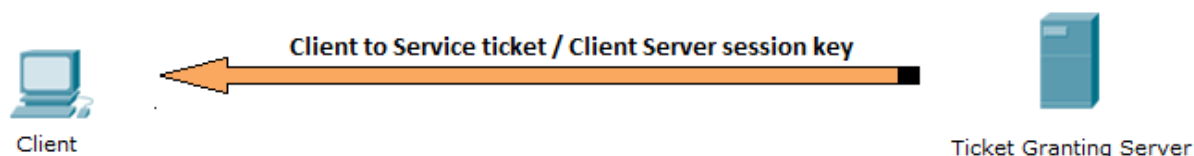
Figure 6.5: Step 8 and 9

10. The client receives the packet and decrypts it with the Client/TGS key to extract the Client/Service key. The Service Ticket remains encrypted.
11. 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.
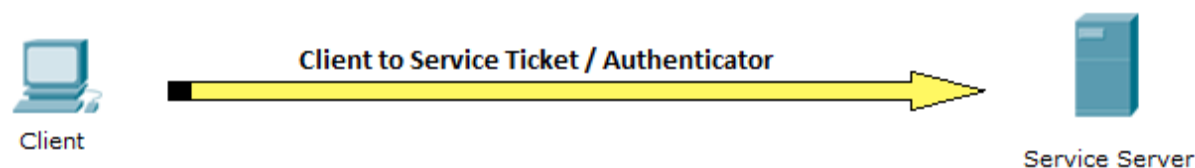
Figure 6.6: Step 10 and 11

12. 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.
13. (OPTIONAL) The application server send a Application Reply back to the client to confirm it is indeed the server the client wished to contact. This step is executed when mutual trust is required.
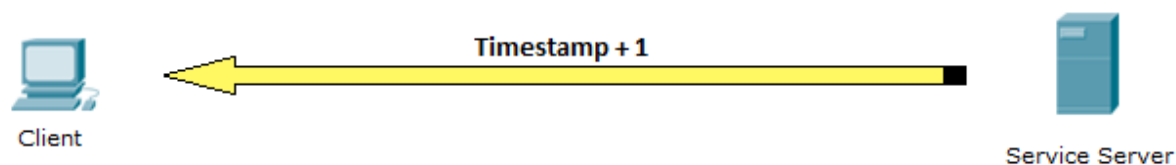
Figure 6.7: Step 12 and 13

14. Communication between the client and the service server can now commence.
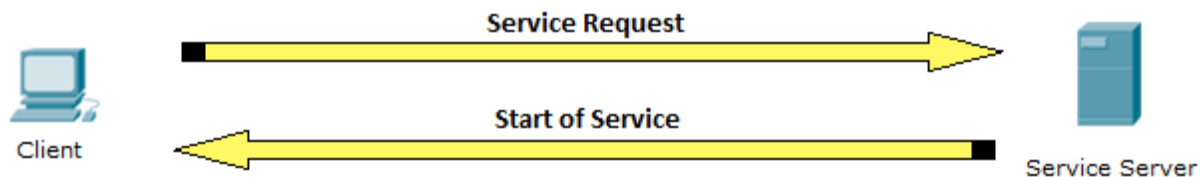
Figure 6.8: Step 14

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

Figure 6.9: Kerberized `telnet` *Wireshark capture.*

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

6.3. 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.

6.4. 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. These risks can however be reduced by setting up a master/slave or a multi-master setup. 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 **[4]**.

- 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 **[5]**.

- 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.

- Trojaned Kerberos (`kinit`, `klist`) commands could steal the unencrypted password and send it to a malicious user who can then assume the identity of a valid user.

7. 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.

7.1. 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.

7.2. 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.

7.3. 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
```

7.4. Users

The main user, called 'khmuser', was given `sudo` powers.

7.5. 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.

7.6. 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
```

8. 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 **[6]**.

8.1. 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 8.1: 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.               86400 IN      NS     dns.khm.be.
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 8.2: 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 **[6]**.
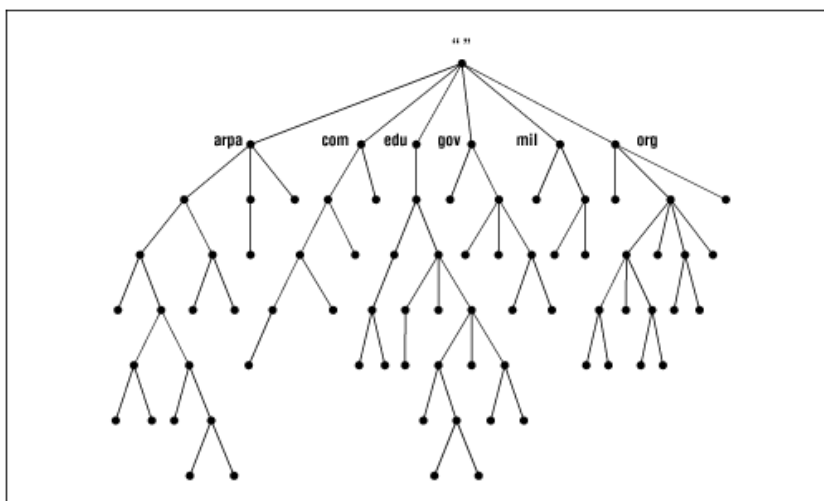
Figure 8.3: Representation of the hierarchical form of the DNS infrastructure.
Source: http://ninja4ru.wordpress.com/2008/02/25/ninja-lezingenavond-over-internet-infrastructuur/

8.2. BIND Installation

To install BIND, the following command was given:

```
sudo apt-get install bind9
```

8.3. BIND Configuration

8.3.1.   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. A reverse zone is a zone responsible for the reverse lookup; it looks up an FQDN that belongs to an IP-address. To create a reverse zone an "`in-addr.arpa`" domain will need to be created, to do this the network part of the IP-address needs to be reversed and "`in-addr.arpa`" needs to be added as a suffix to the zone name. Considering the /24 class C network address that was used, this will become 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.

8.3.2.   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 the local router:

```
forwarders    {
      192.168.1.254;
            };
```

8.3.3.   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 configuration:

```
$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
```

Some explanation concerning the various terms:

SOA: The first record in the zonefile is the SOA or the Start of Authority record. This record first lists the nameserver of the zone and next the e-mail address of the administrator of this zone. Note that the '@' character has been replaced by a period.

The next parameter of the SOA record is a serial number. The serial number helps slave servers identify which version of the zone file they have saved locally. For that reason the serial number needs to be increased every time a zone file is updated. It is best practice to use a date here in the form of [YEAR] + [MONTH] + [DAY] + [VERSION NUMBER USED THAT DAY].

All the other parameters of the SOA file are all expressed in seconds. The explanation can be found in the file itself.

A: An A or address record links a part of a domain name to an IP-address. The @ symbol refers to the origin of the domain, in this case specifieid as `khm.lan`.

It is of course possible to assign multiple addresses within the same zone in the following manner:

```
@                 IN    A     192.168.0.152
firewall          IN    A     192.168.0.1
switch            IN    A     192.168.0.3
replaytv          IN    A     192.168.0.200
```

In this example, `firewall.khm.lan` will refer to the IP-address 192.168.0.1.

MX: MX (Mail eXchange) records are used for mail servers. A priority can be specified in front of the alias that will be used for the name server. This wa,y mail traffic can be prioritized if more than one mail server is present.

NS: This record specifies the name server that is authoritative for a zone.

CNAME: A CNAME or Canonical name is used to assign an alias to already existing A records. There are no CNAME records in use in the used BIND configuration, but the following example will explain the concept:

```
www           IN          CNAME       khm.lan.
alias1        IN          CNAME       khm.lan.
alias2        IN          CNAME       khm.lan.
alias3        IN          CNAME       khm.lan.
```

In the above example, `alias1.khm.lan`, `alias2.khm.lan`, `alias3.khm.lan` and `www.khm.lan` would all refer to `khm.lan`.

The `reverse-1.168.192` zone 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 lookup use PTR records, which link IP addresses to hostnames.

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

8.3.4.   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
```

It is possible to specify multiple domains here, but there is currently no need for this considering our forwarders in `named.conf.options` are properly configured.

The resolv.conf file corresponds with the following interface in Windows, which might seem more familiar to Windows users.
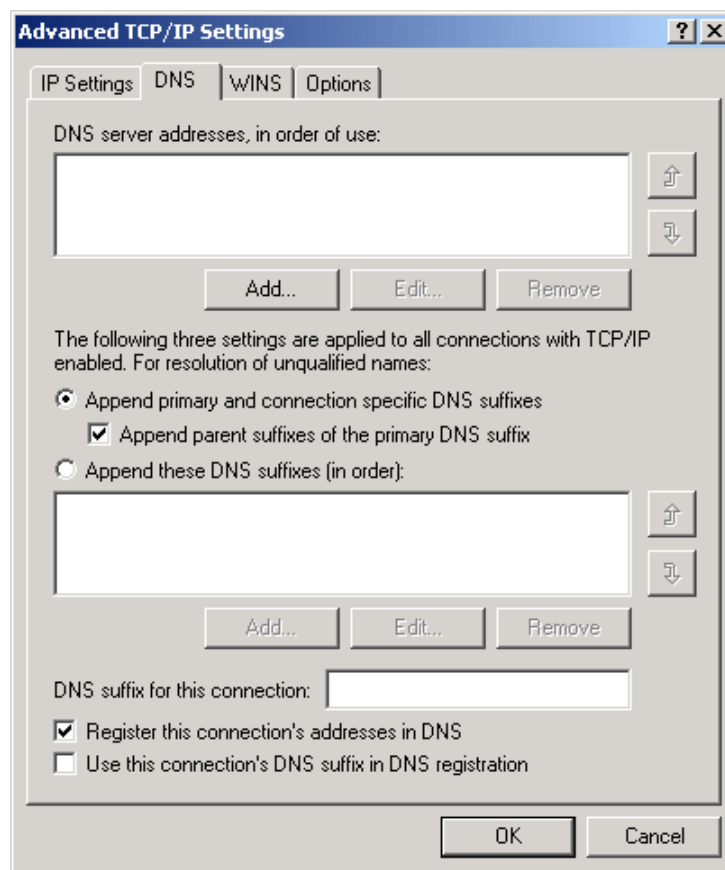
Figure 8.4: The DNS configuration tab on a Windows operating system.

8.3.5. 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
```

8.4. 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
```

9. Step 2: NTP

When thinking about the way Kerberos works it can be concluded that the authentication process relies on synchronized system clocks. Desynchronized time in between systems will result in issues with the different timestamps and validity periods, often referred to as "time skew".

In order to synchronize the system time in the network, the Network Time protocol or NTP will be utilized. 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.

9.1. NTP Installation

In order to proceed, the `ntp-server` package must be installed:

```
sudo apt-get install ntp-server
```

9.2. 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
#broadcastclient
```

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.

9.3. Setting up the 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 error messages regarding "strata too high" are usually to blame on a unsynchronized NTP server. It is advised in that case to try again later.

10.   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.

10.1.   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
```

10.2.   Kerberos Configuration

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

10.2.1.   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
```

10.2.2.   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
    }
```

10.2.3.   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`.

10.3.   Kerberos Administration

10.3.1.   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.

10.3.2.   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
```

10.3.3.   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
```

10.3.4.   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.
```

10.3.5.   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.

10.3.6.   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
```

10.4.   Setting up Kerberos clients

10.4.1.   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.

10.4.2.   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 **[18]**. 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
```

11.   Step 4: 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
```

11.1.   telnet

11.1.1.   telnet installation

The following command was issued to install a kerberized telnet daemon:

```
sudo apt-get install krb5-telnetd
```

11.1.2.   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)
```

11.1.3. 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.

11.2.   rlogin

11.2.1.   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.

11.3.   Secure Shell (SSH)

Secure Shell is preferred over telnet and rlogin considering the security it provides **[19]**. There is no preconfigured kerberized package for SSH. The OpenBSD SSH daemon will be downloaded and configured for use with Kerberos authentication.

11.3.1.   SSH installation

The following command was issued to install the SSH daemon:

```
sudo apt-get install openssh-server
```

11.3.2.   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
```

11.3.3.   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.

11.4.   File Transfer Protocol

11.4.1.   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
```

11.4.2.   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
```

11.4.3.   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.

12.   Step 5: Web SSO

12.1.   Web server

12.1.1.   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 **[20]**, but is weakly secured for development purposes. A special tool can be used to harden the server **[21]**. 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:
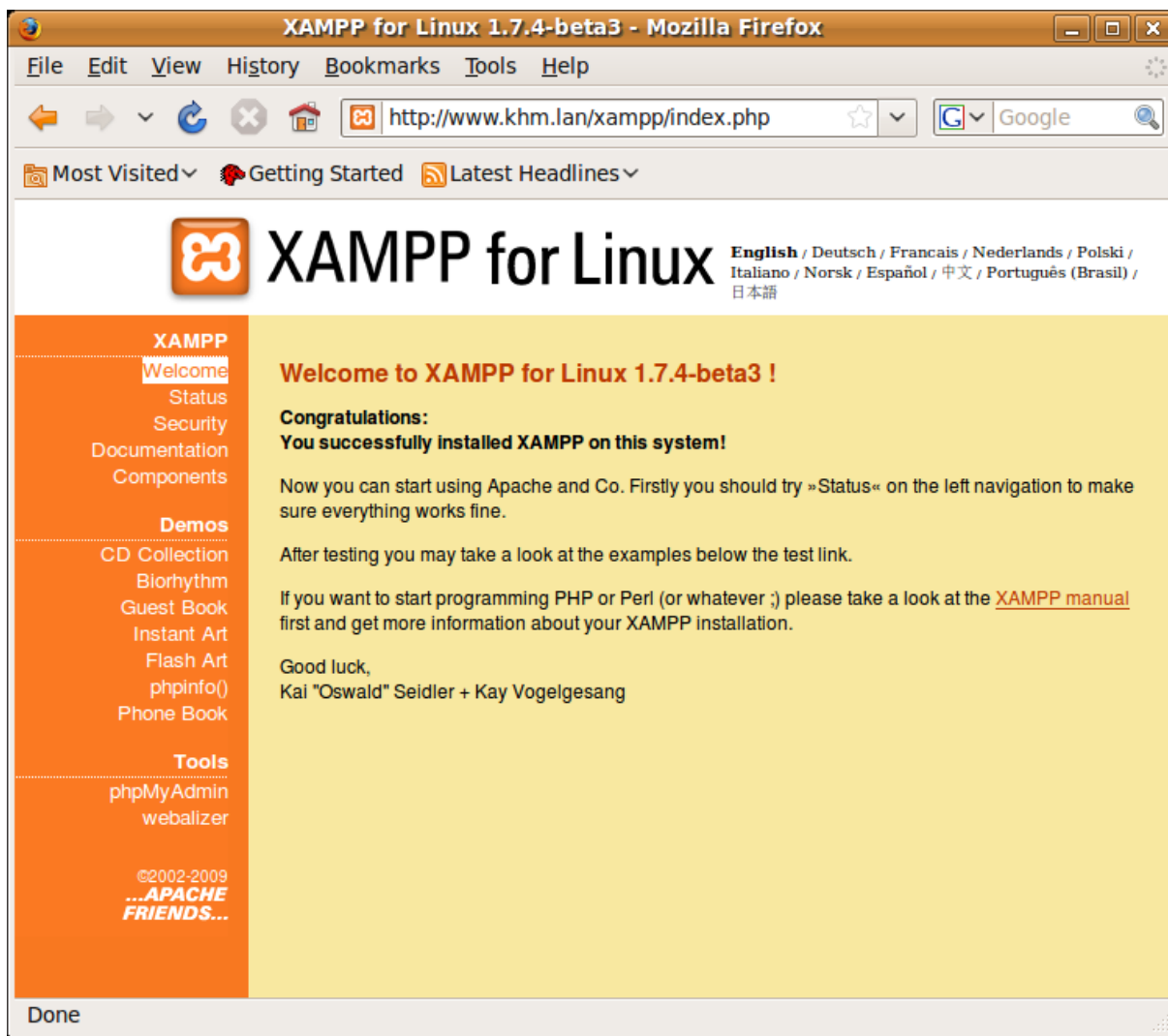
Figure 12.1: Screenshot of the main web page of XAMPP.

12.1.2.   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.

12.2.   mod_auth_kerb

12.2.1.   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 **[21, 22]**.

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

12.2.2.   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 **[23]**:

- 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 an 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 **[23]**.

12.2.3.   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`.

12.2.4.   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 KHM.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
```

12.2.5. 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 [24].

To configure Mozilla Firefox, `about:config` was typed into the browser bar to access the configuration menu:
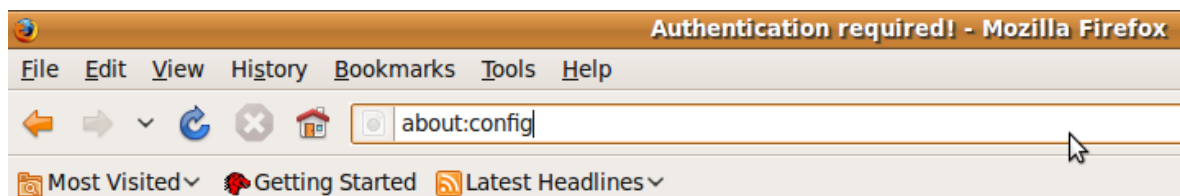
Figure 12.2: Typing "about:config" in the browser bar to access the configuration menu.

After reading and confirming a warning message, the configuration menu was shown:
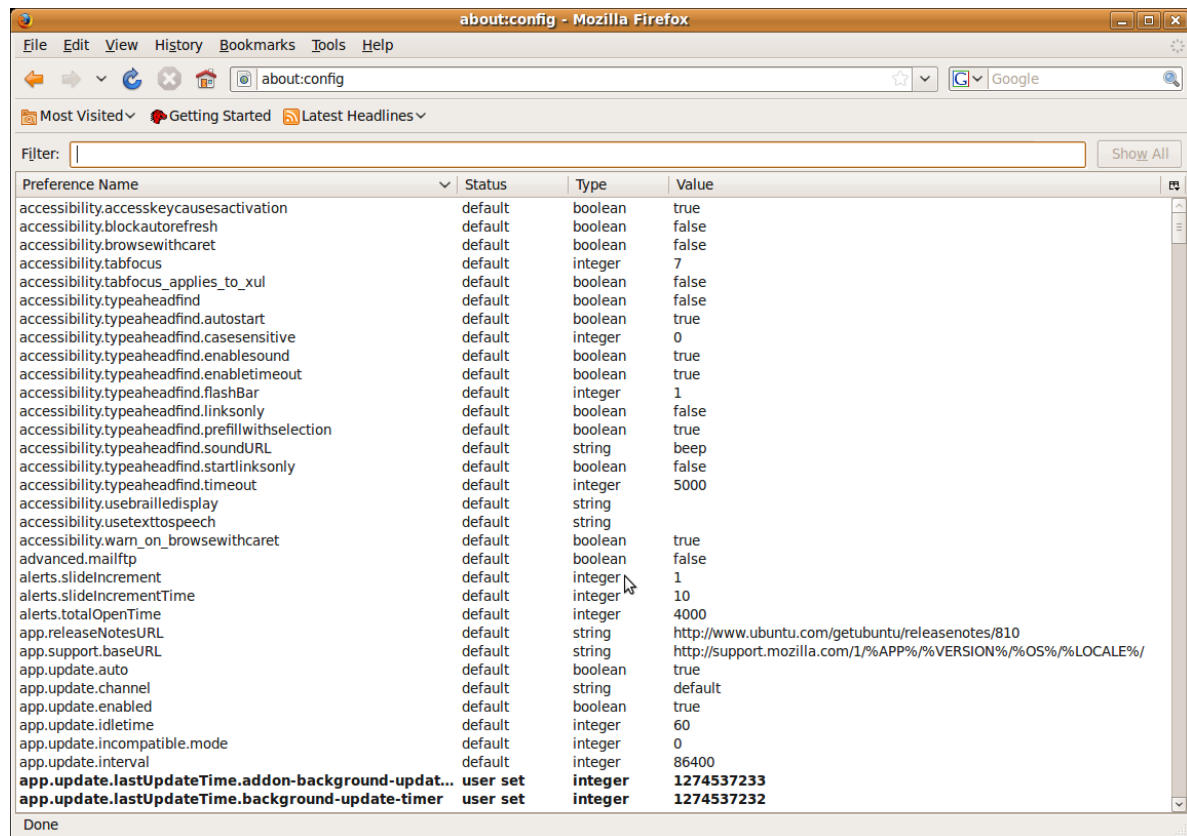
Figure 12.3: The Mozilla Firefox configuration menu.

The configuration items were filtered by the keyword `network.negotiate-auth`:
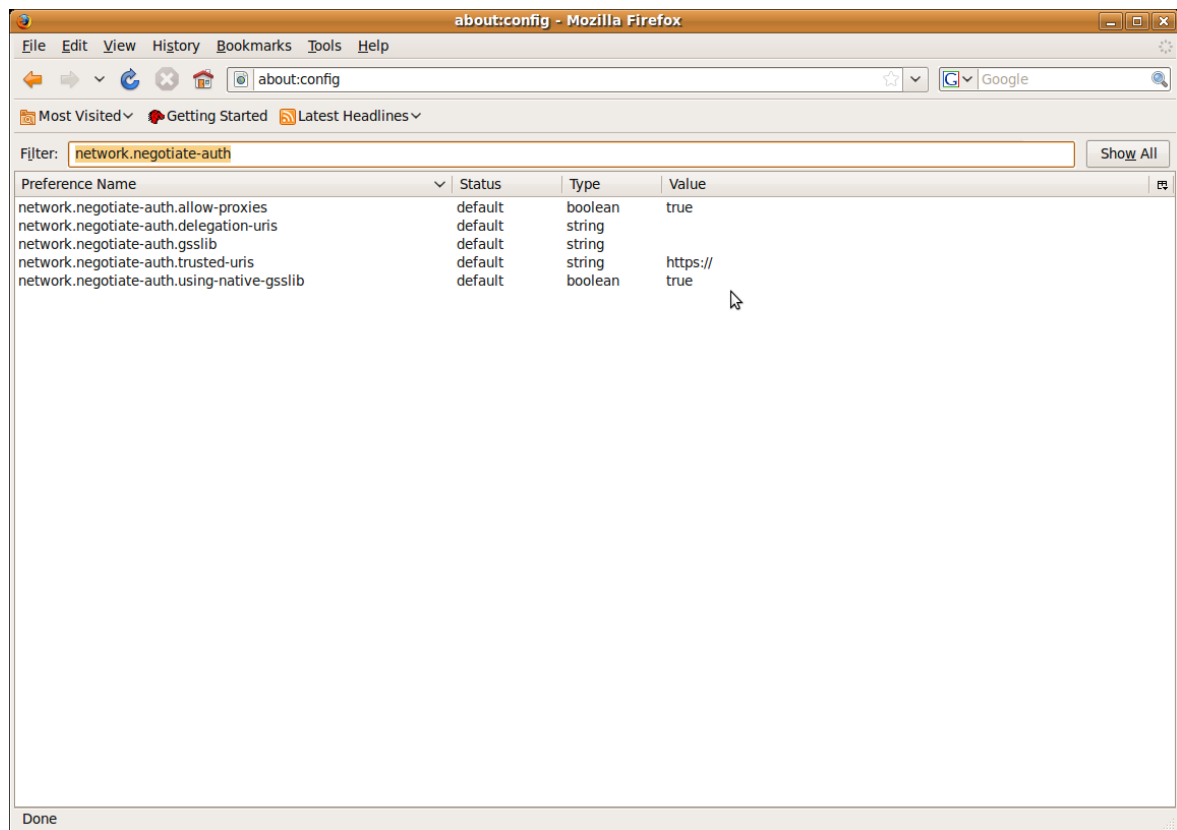
Figure 12.4: All configuration items after filtering by the keyword "network.negotiate-auth".

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

12.2.6.   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.

12.2.7.   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.
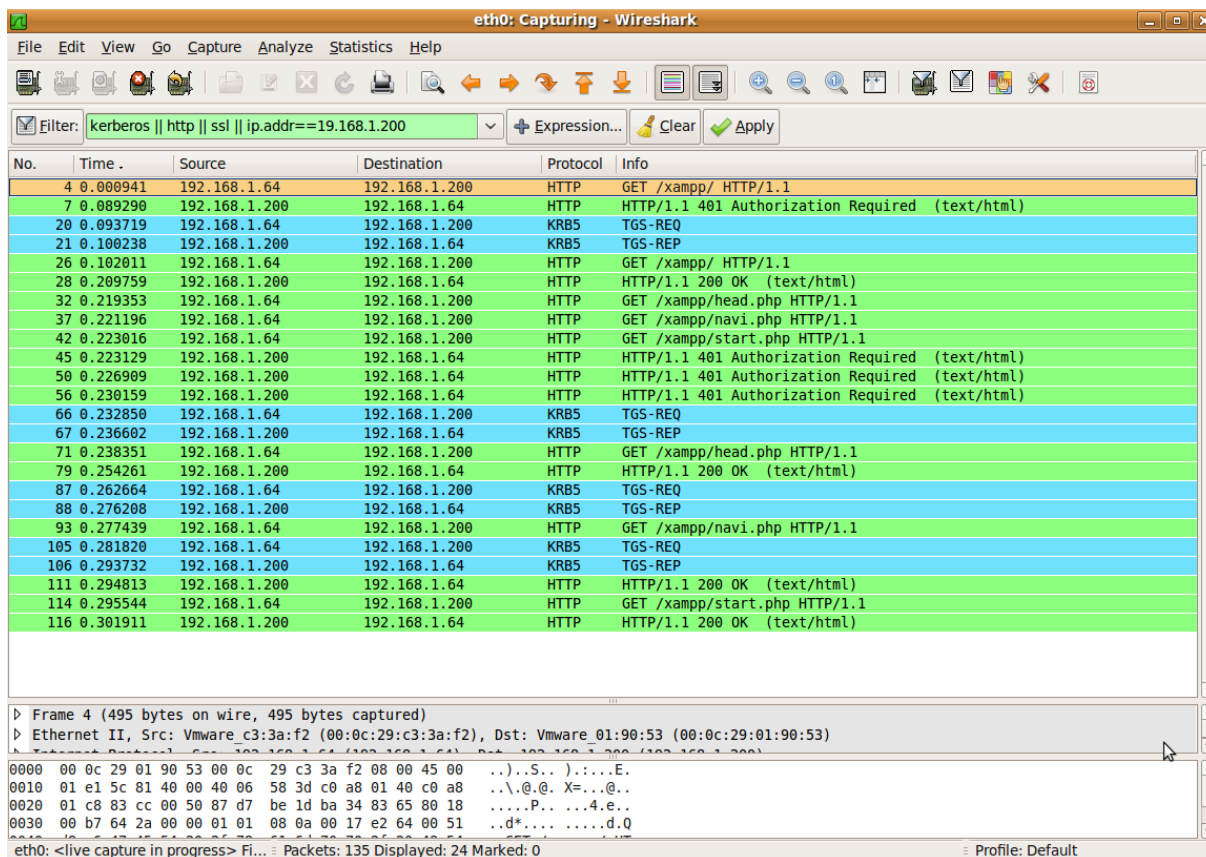
Figure 12.5: Wireshark capture of client requesting four different resources from the web server.

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 increase the cost, complexity and vulnerability of the authentication system. Another way to deal with this issue is to have all principals in an LDAP directory and have two or more Kerberos masters handle their share of requests using RRDNS load balancing. These measures would make the system more redundant and Denial of Service attacks would become far harder to execute **[25]**. Balancing these tradeoffs should be done carefully by the Kerberos administrator.

12.3.   php_krb5

12.3.1.   Introduction

`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 **[26, 27]**.

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 **[27]**.

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.

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
```

12.3.2.   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.

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/
```

12.3.3.   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:

Figure 12.6: Using phpinfo()

12.3.4.   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 …" subchapters.

12.3.5. 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.
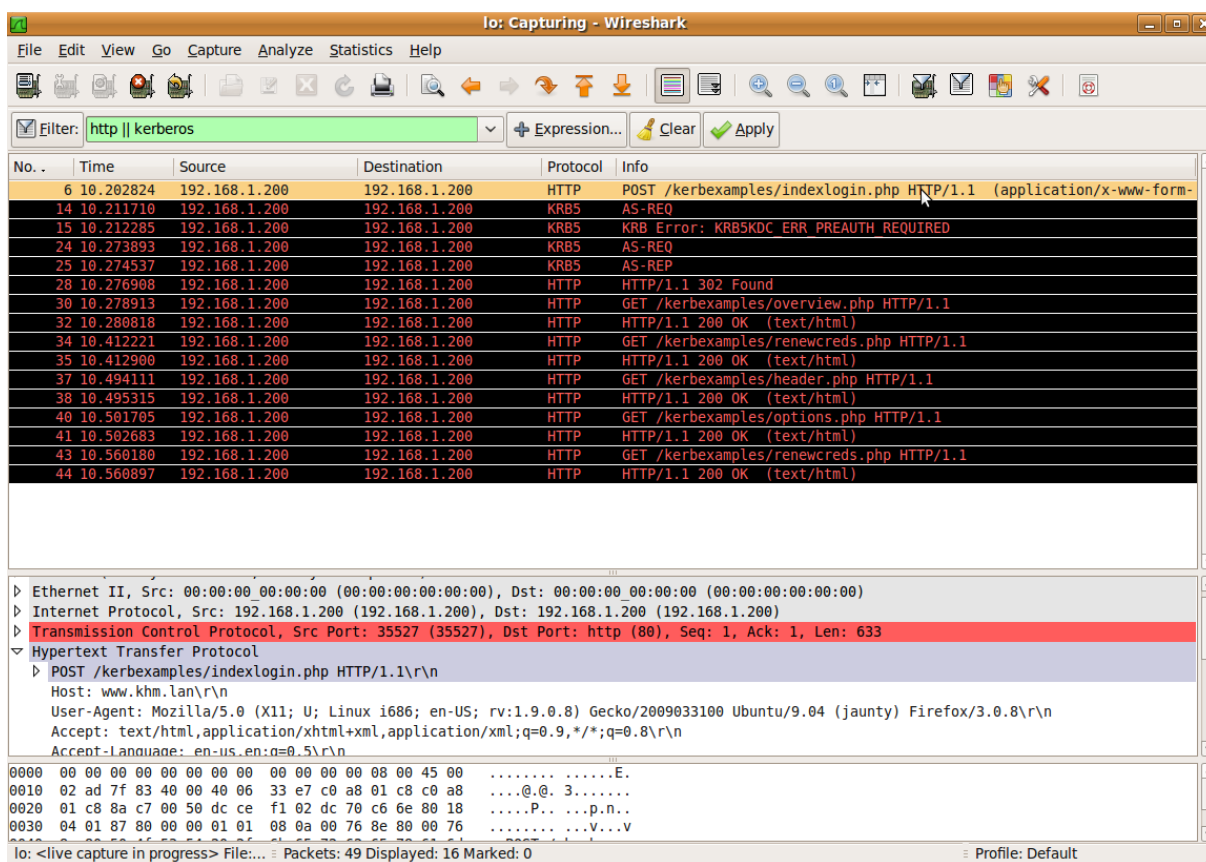
Figure 12.7: Wireshark capture of the login process for the `php_krb5` portal.

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 **[28]**.

12.4. 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.

12.4.1.   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.

After adding the files below, the `httpd.conf` configuration file was modified to change its Kerberos authentication to only the `/sso_examples` directory:

```
<Directory "/opt/lampp/htdocs/sso_examples">
        ...
</Directory>
```

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
<HTML>
<HEAD>
<TITLE>
      Redirecting, please wait...
</TITLE>
</HEAD>
<BODY onload="if (parent.location != location) { parent.location.href =
document.location.href; }">
<SCRIPT type="text/javascript">
      window.location = "overview.php";
</script>
      If you are not redirected automatically, please
click <a href="overview.php">here</a>.<br/>
</BODY>
</HTML>
```

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

```php
<?php

// Check user credentials
include 'checkcreds.inc';
```

```php
// 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'); }

?>
<html>

<frameset rows="100,95%">
  <frame src="header.php" frameborder="0" noresize scrolling="no" name="header">
  <frame src="options.php" frameborder="0" name="main">

<noframes>
<body>Your browser does not handle frames!</body>
</noframes>

</frameset>

</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
<?php

if( !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
<?php

// Check user credentials
include 'checkcreds.inc';

// Return to main page if variable could not be found
if( $CREDENTIALSOK == -1 ) { header('Location: index.html'); }

?>
<HTML>
<HEAD>
<link href="portal_styles.css" rel="stylesheet" type="text/css">
</HEAD>
<BODY>
```

```html
<div id="header">
<div id="currentuser">
<?php if( $CREDENTIALSOK )
        {
                $userinfo = explode( '@', $_SERVER['REMOTE_USER'] );
                echo "Currently logged in as <b>".$userinfo[0]."</b><br/>\n";
                echo "Realm: ".$userinfo[1]."<br/>";
        }
        ?>
<a href="options.php" target="main">Back to portal</a>
</div>
</div>

</BODY>
</HTML>
```

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

```css
#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
<?php
```

```
// Check user credentials
include 'checkcreds.inc';

// Return to main page if cache could not be found
if( $CREDENTIALSOK == -1 ) { header('Location: index.html'); }

?>
<html>
<head>
<link href="portal_styles.css" rel="stylesheet" type="text/css">
</head>
<body>
<div id="overview">
        <div class="sso_option">
        <a href="notes">
        <div class="sso-option-img">
                <img src="img/Notepad-
icon.png" width="150" height="150" border="0" align="center" />
        </div>
        <br/>Notepad</a><br/>
        Make notes in your personal notepad.
        </div>

        <div class="sso_option">
        <div class="sso-option-img">
                <a href="mediawiki"><img src="img/Mediawiki.png" width="150" height
="141" border="0" align="center" />
        </div>
        <br/>Wiki</a><br/>
        Gain access to a local MediaWiki.
        </div>

                <div class="sso_option">
        <div class="sso-option-img">
                <a href="roundcube"><img src="img/rcube_logo.gif" style="padding:45
px;" width="210" height="55" border="0" align="center" />
        </div>
        <br/>Webmail</a><br/>
        Access your local webmail.
        </div>
</div>
</body>
</html>
```

A screenshot of the final portal page:
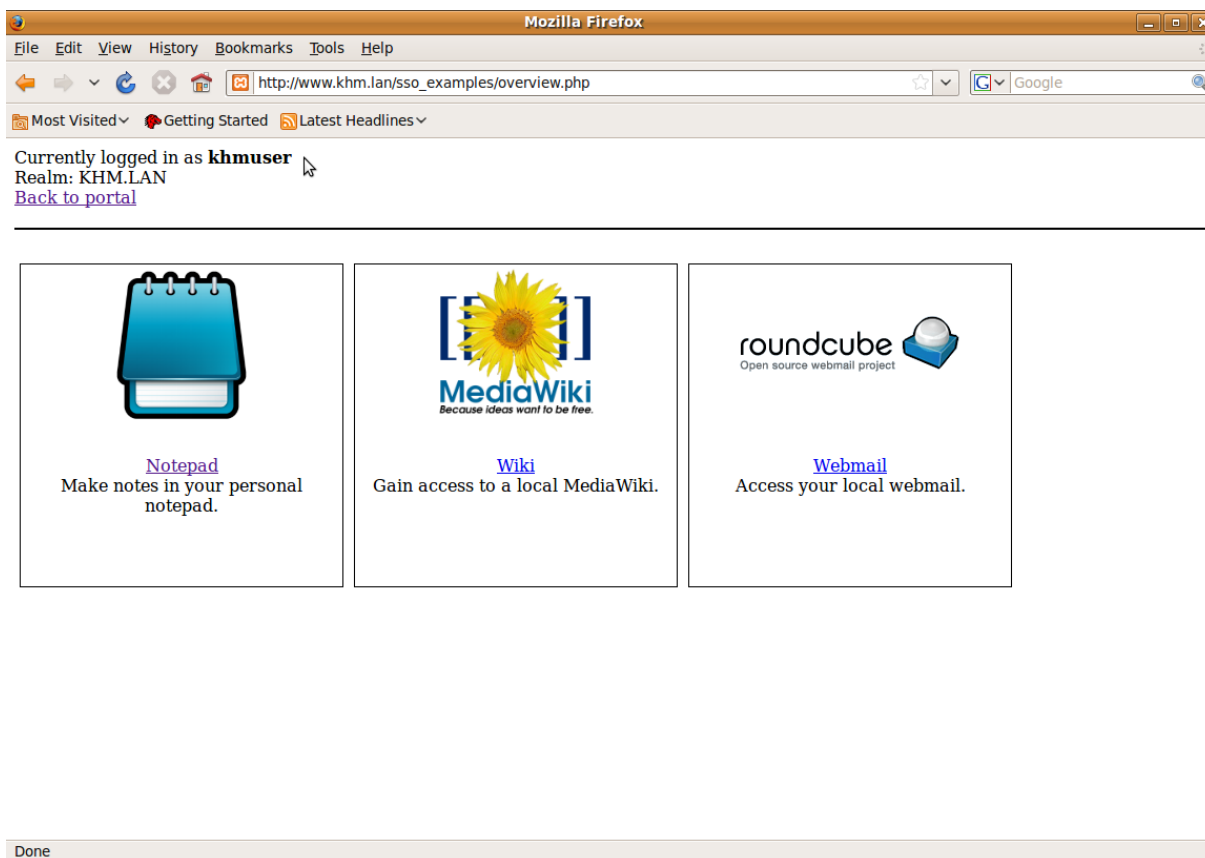
Figure 12.8: A screenshot of the `mod_auth_kerb` portal page.

12.4.2.   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 size 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 remaining 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>
<HEAD>
<TITLE>
Please log in
</TITLE>
```

```html
</HEAD>
<BODY onload="if (parent.location != location) { parent.location.href =
document.location.href; }">
        <FORM method="post" action="indexlogin.php">
                <table>
                        <tr>
                        <td>Username:</td>
                        <td><input type="text" size="20" name="login_username" /></
td>
                        </tr>
                        <tr>
                        <td>Password:</td>
                        <td><input type="password" size="20" name="login_password"
/></td>
                        </tr>
                        <tr>
                        <td>Realm:</td>
                        <td><input type="text" size="20" name="login_realm" /></td>
                        </tr>
                        <tr>
                        <tr>
                        <td colspan="2"><input type="submit" value="Log In"></td>
                        </tr>
                </table>
        </FORM>
</BODY>
</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
<?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('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();
$_SESSION['krb_username'] = $username;
$_SESSION['realm'] = $realm;
```

```php
// Redirect user to overview page
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
<?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'); }

?>
<html>

<frameset rows="100,95%">
  <frame src="header.php" frameborder="0" noresize scrolling="no" name="header">
  <frame src="options.php" frameborder="0" name="main">

<noframes>
<body>Your browser does not handle frames!</body>
</noframes>

</frameset>

</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
<?php

// Try and load the credential cache of user

// Load session variables
session_start();

if( !isset($_SESSION['krb_username']) || !isset($_SESSION['realm']) )
{
        $CREDENTIALSOK = -1;
}
else
{

$USERNAMECREDENTIALCHECK = $_SESSION['krb_username'];
$REALMCREDENTIALCHECK = $_SESSION['realm'];
```

```php
// Load credential cache
$CREDENTIALCHECKCACHE = new KRB5CCache();

// The credentials are invalid by default.
$CREDENTIALSOK = -1;


try
{

$cachefile = "/tmp/$USERNAMECREDENTIALCHECK@$REALMCREDENTIALCHECK.ccache";

if( 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/rene
wcreds.php';

                if( isset( $_SERVER['HTTPS'] ) )
                {
                        // We're using HTTPS.
                        $renewcredslocation = str_replace( "/opt/lampp/htdocs/", "h
ttps://".$_SERVER['HTTP_HOST']."/", $renewcredslocation );
                }
                else
                {
                        // We're not using HTTPS.
                        $renewcredslocation = str_replace( "/opt/lampp/htdocs/", "h
ttp://".$_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; }
```

```php
}
?>
```

`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
<?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'); }

?>
<HTML>
<HEAD>
<link href="portal_styles.css" rel="stylesheet" type="text/css">
<script 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/rene
wcreds.php';
                if( isset( $_SERVER['HTTPS'] ) )
                        $renewcredslocation = str_replace( "/opt/lampp/htdocs/", "h
ttps://".$_SERVER['HTTP_HOST']."/", $renewcredslocation );
                else
                        $renewcredslocation = str_replace( "/opt/lampp/htdocs/", "h
ttp://".$_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>
</HEAD>
<BODY onload="ticketTimer = setInterval  ( 'reduceTicket()', 1000 );">
<?php
if( $CREDENTIALSOK == 1 )
{
        $cgssapi = new GSSAPIContext();
        $cgssapi->acquireCredentials($CREDENTIALCHECKCACHE);
        $information = $cgssapi->inquireCredentials();
}
?>
<div id="header">
<div id="timer">Time left:
<div id="timeleft">
        <?php
        if( $CREDENTIALSOK == 1 )
        {
                echo $information['lifetime_remain'];
        }
        else
        {
                echo "Credentials Expired.";
        }
        ?>
</div>
<a href="logout.php" target="_self">Log out</a>
</div>
<div id="currentuser">
        <?php
        if( $CREDENTIALSOK == 1 )
        {
                $userinfo = explode( '@', $information['name'] );
                echo "Currently logged in as <b>".$userinfo[0]."</b><br/>\n";
                echo "Realm: ".$userinfo[1]."<br/>";
        }
        else
        {
                echo "Credentials Expired.<br/>";
        }
        ?>
```

```html
            <a href="options.php" target="main">Back to portal</a>
</div>
</div>
</BODY>
</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
<?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'); }

?>
<html>
<head>
<link href="portal_styles.css" rel="stylesheet" type="text/css">
</head>
<body>
<div id="overview">
        <div class="sso_option">
        <a href="notes">
        <div class="sso-option-img">
                <img src="img/Notepad-
icon.png" width="150" height="150" border="0" align="center" />
        </div>
        <br/>Notepad</a><br/>
        Make notes in your personal notepad.
        </div>

        <div class="sso_option">
        <div class="sso-option-img">
                <a href="mediawiki"><img src="img/Mediawiki.png" width="150" height
="141" border="0" align="center" />
        </div>
        <br/>Wiki</a><br/>
        Gain access to a local MediaWiki.
        </div>

                <div class="sso_option">
        <div class="sso-option-img">
                <a href="roundcube/roundcubelogin.php"><img src="img/rcube_logo.gif
" style="padding:45px;" width="210" height="55" border="0" align="center" />
        </div>
        <br/>Webmail</a><br/>
        Access your local webmail.
        </div>
</div>
```

```html
</body>
</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
<HTML>
<HEAD>
<TITLE>
Renew credentials
</TITLE>
</HEAD>
<BODY>
<?php session_start(); ?>
        <FORM method="post" action="renewcreds_action.php">
        Your credentials need to be renewed. Please fill in your password.<br/>
        If you wish to log in as another user, close this window and go to the main
portal page.<br/>
                <table>
                        <tr>
                        <td>Username:</td>
                        <td><?php echo $_SESSION['krb_username']; ?></td>
                        </tr>
                        <tr>
                        <td>Realm:</td>
                        <td><?php echo $_SESSION['realm']; ?></td>
                        </tr>
                        <tr>
                        <td>Password:</td>
                        <td><input type="password" size="20" name="login_password"
/></td>
                        </tr>
                        <tr>
                        <tr>
                        <td colspan="2"><input type="submit" value="Renew
Credentials"></td>
                        </tr>
                </table>
        </FORM>
</BODY>
</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
<?php

// If some values are missing, return to the login page
if( !isset( $_POST["login_password"] )
  ) { header('Location: renewcreds.php'); }

// Get values
session_start();
$username = $_SESSION['krb_username'];
$password = $_POST["login_password"];
```

```php
$realm = $_SESSION['realm'];

// Get credential cache for user
$ccache = new KRB5CCache();
$flags = array('tkt_lifetime' => 3600);

// Try to authenticate
try {
        $ccache->initPassword($username."@".$realm, $password, $flags);
} catch (Exception $error) { echo "ERROR: ".$error; 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['krb_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>';

?>
```

12.5.   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.

12.5.1.   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.

12.5.1.1. *MySQL commands*

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

```sql
CREATE DATABASE `kerbnotes`;

CREATE TABLE `kerbnotes`.`users` (
`userID` INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`user` VARCHAR( 255 ) NOT NULL ,
UNIQUE (
`user`
)
) ENGINE = MYISAM ;

CREATE TABLE `kerbnotes`.`notes` (
`noteID` INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`userID` INT NOT NULL ,
`note` LONGTEXT NOT NULL
) ENGINE = MYISAM ;
```

12.5.1.2. *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
<?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">
```

```html
<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
<?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();
```

```php
?>
```

`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
<?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>";
```

```php
        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
<?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'); }
```

```php
// 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'] == $noterow['userID'
] )
                                {
                                        echo '<input type="hidden"
value="'.$_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
<?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
<?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 )
{
```

```php
        // 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.

```css
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;
}
```

12.5.2.   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 **[30]**. It has a very active development

community and hundreds of extensions available **[31]**. 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 **[32]**. It can easily be modified to use other variables, which will prove to be quite useful for the `php_krb5` portal.

12.5.2.1. 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 '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_C
ONNECTIONS 0 ;
CREATE DATABASE IF NOT 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/
```

Lastly, the WikiSysop user was logged in and used to modify the `common.css` file in MediaWiki itself (http://www.khm.lan/sso_examples/mediawiki/index.php/MediaWiki:Common.css) to contain the following:

```
/* CSS placed here will be applied to all skins */
```

```
li#pt-logout { display: none; }
```

This hides the log-out button from a user, and will continue to work after upgrading.

12.5.2.2.   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
<?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...
```

```php
$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:

```php
// 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:
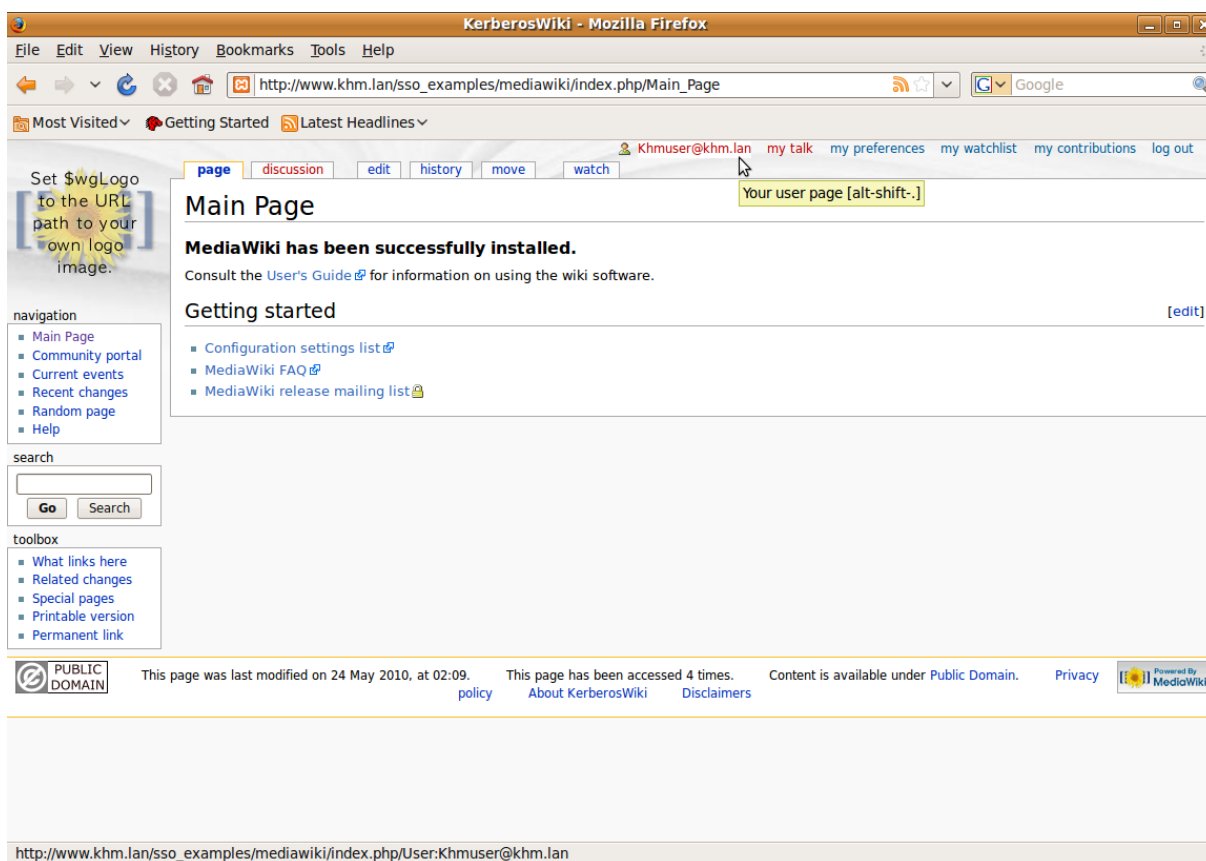
Figure 12.9: Screenshot of single sign-on authentication onto a MediaWiki.

12.5.2.3.   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 portal 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
<?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'] ) )
                $portalpage = str_replace( "/opt/lampp/htdocs/", "https://".$_SERVE
R['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:
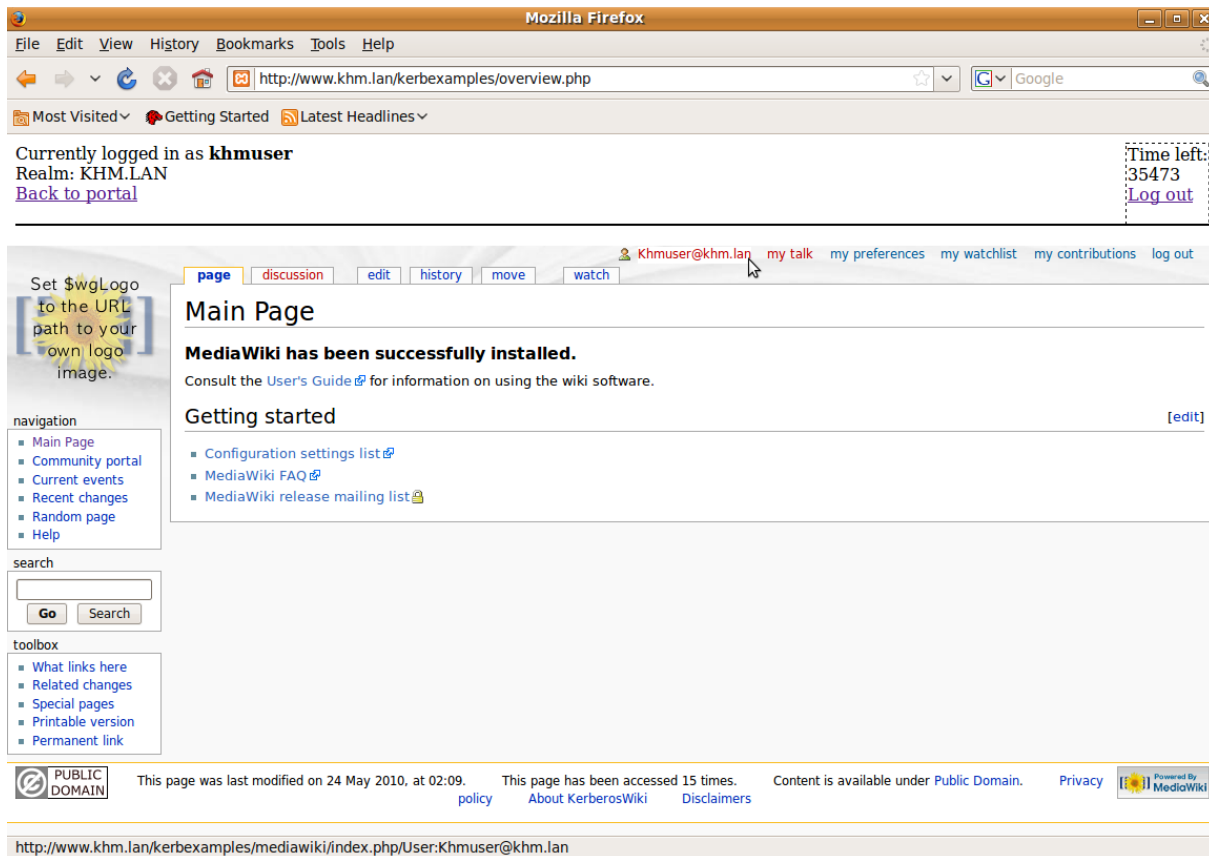
Figure 12.10: Screenshot of Kerberos-based credential authentication on a MediaWiki.

12.5.3. 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 were used to install and configure the mail server **[33]**. 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.

12.5.3.1. 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.

12.5.3.2. Mail server configuration

All packages were then configured, starting with Postfix:

The file `/etc/mailname` was confirmed to contain `mail.khm.lan`.

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
```

The file `/etc/aliases` was copied to `/etc/postfix/aliases`.

The `postalias` command was executed: `sudo postalias /etc/postfix/aliases`

The virtual mailbox was created with the command `sudo mkdir /var/spool/mail/virtual`.

The `virtual` group and user were added with the commands `sudo groupadd virtual -g 5000` and `sudo useradd virtual -u 5000 -g 5000`.

The `virtual` group was made owner of the virtual mailbox with the command `sudo chown -R virtual:virtual /var/spool/mail/virtual`.

The links with the MySQL database were created by creating the following files:

`/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
```

/etc/postfix/mysql_uid.cf

```
user=mail
password=[PASSWORD]
dbname=maildb
table=users
select_field=uid
where_field=id
hosts=127.0.0.1
```

/etc/postfix/mysql_gid.cf

```
user=mail
password=[PASSWORD]
dbname=maildb
table=users
select_field=gid
where_field=id
hosts=127.0.0.1
```

/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
```

/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
```

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 auto_increment,
`mail` varchar(120) NOT NULL default '',
`destination` varchar(120) NOT NULL default '',
`enabled` tinyint(1) NOT NULL default '1',
PRIMARY KEY  (`pkid`),
UNIQUE KEY `mail` (`mail`)
) ;
```

```
CREATE TABLE `domains` (
`pkid` smallint(6) NOT NULL auto_increment,
`domain` varchar(120) NOT NULL default '',
`transport` varchar(120) NOT NULL default 'virtual:',
`enabled` tinyint(1) NOT NULL default '1',
PRIMARY KEY  (`pkid`)
) ;

CREATE TABLE `users` (
`id` varchar(128) NOT NULL default '',
`name` varchar(128) NOT NULL default '',
`uid` smallint(5) unsigned NOT NULL default '5000',
`gid` smallint(5) unsigned NOT NULL default '5000',
`home` varchar(255) NOT NULL default '/var/spool/mail/virtual',
`maildir` varchar(255) NOT NULL default 'blah/',
`enabled` tinyint(3) unsigned NOT NULL default '1',
`change_password` tinyint(3) unsigned NOT NULL default '1',
`clear` varchar(128) NOT NULL default 'ChangeMe',
`crypt` varchar(128) NOT NULL default 'sdtrusfX0Jj66',
`quota` varchar(255) NOT NULL default '',
`procmailrc` varchar(128) NOT NULL default '',
`spamassassinrc` varchar(128) NOT NULL default '',
PRIMARY KEY  (`id`),
UNIQUE KEY `id` (`id`)
) ;
```

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"
```

2. 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('[PASSWORD]') ),
        ('vivita@blobber.org','vivita','vivita/', encrypt('[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
```

12.5.3.3.   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
```
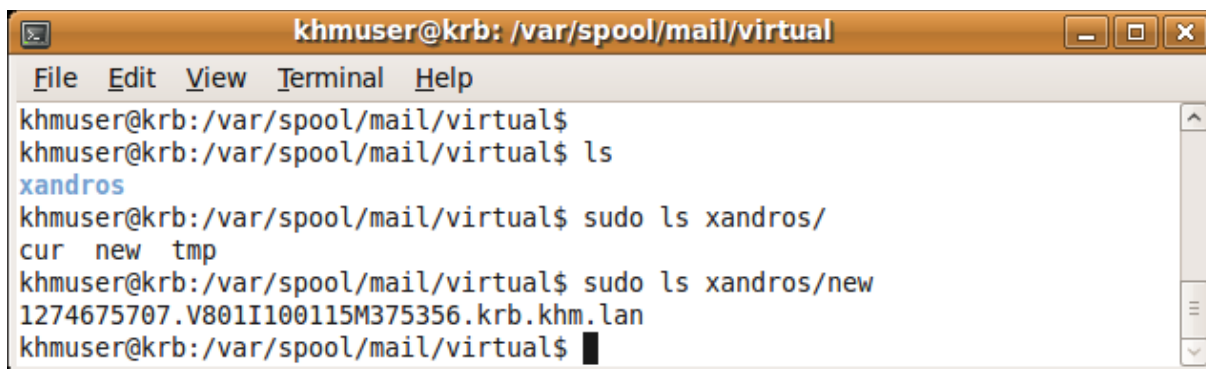
The SMTP and IMAP ports were open. `telnet` was used to test the functionality of SMTP first:

```
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 12.11: 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
```

12.5.3.4.   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 **[34]**, 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 **[35, 36, 37]**. 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.

12.5.3.5.   Roundcube installation and configuration

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 '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_C
ONNECTIONS 0 ;
CREATE DATABASE IF NOT 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/. 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 12.12: Results of SMTP and IMAP testing tools.

12.5.3.6.   Integration with mod_auth_kerb portal

A patch for Roundcube to use the `REMOTE_USER` Apache environment variable was found in a ticket on the Roundcube Trac (a project management platform) at http://trac.roundcube.net/ticket/ 1486689. This patch uses a static password for each automatically authenticated user, which severely limits the user's possibilities in using non-web mail clients for his e-mail. This issue is further dissected in the discussion chapter.

The patch was copied and pasted into the file remote_user_auth.php, which was placed in the /sso_examples/roundcube/plugins/remote_user_auth directory:

```php
<?php

/**
* remote user auth plugin
* set action to login if remote user is set
*/
class remote_user_auth extends rcube_plugin
{
        public $task = 'login';

        function init()
```

```
        {
          $this->add_hook('startup', array($this, 'startup'));
          $this->add_hook('authenticate', array($this, 'authenticate'));
        }

        function startup($args)
        {
          // change action to login if REMOTE_USER is set
          if ( $_SERVER['REMOTE_USER'] ) {
                $args['action'] = 'login';
          }

          return $args;
        }

        function authenticate($args)
        {
          if ( $_SERVER['REMOTE_USER'] ) {
                $args['user'] = $_SERVER['REMOTE_USER'];
                $args['pass'] = 'mailadmin';
                $args['host'] = 'localhost';
                $args['cookiecheck'] = false;
          }

          return $args;
        }
}
```

Then, the `main.inc.php` configuration file was modified to contain the following line:

```
$rcmail_config['plugins'] = array('remote_user_auth');
```

The logout button was removed by editing `/sso_examples/roundcube/skins/default/common.css` to contain the following:

```
#taskbar a.button-logout
{
  background-position: 0 -75px;
  display:none;
}
```

As e-mail accounts are not created automatically, some MySQL statements needed to be executed in order to have `khmuser@KHM.lan` log in automatically:

```
-- Insert domain
INSERT INTO `maildb`.`domains` (`pkid` ,`domain` ,`transport` ,`enabled`)
VALUES (NULL , 'khm.lan', 'virtual:', '1');
-- Insert user
INSERT
INTO `maildb`.`users` (`id` ,`name` ,`uid` ,`gid` ,`home` ,`maildir` ,`enab
led` ,`change_password` ,`clear` ,`crypt` ,`quota` ,`procmailrc` ,`spamassa
ssinrc`)
```

```
VALUES ('khmuser@khm.lan', 'khmuser', '5000', '5000','/var/spool/mail/virtu
al', 'khmuser/', '1', '1', 'ChangeMe','MGGKUVlyGU.K6', '', '', '');
```

A maildir for `khmuser@khm.lan` was created by issuing the following commands:

```
cd /var/spool/mail/virtual
sudo mkdir khmuser
sudo chmod 750 khmuser
sudo chown virtual:virtual khmuser
```

Finally, the feature was tested by browsing to http://www.khm.lan/sso_examples/roundcube/ with Mozilla Firefox:
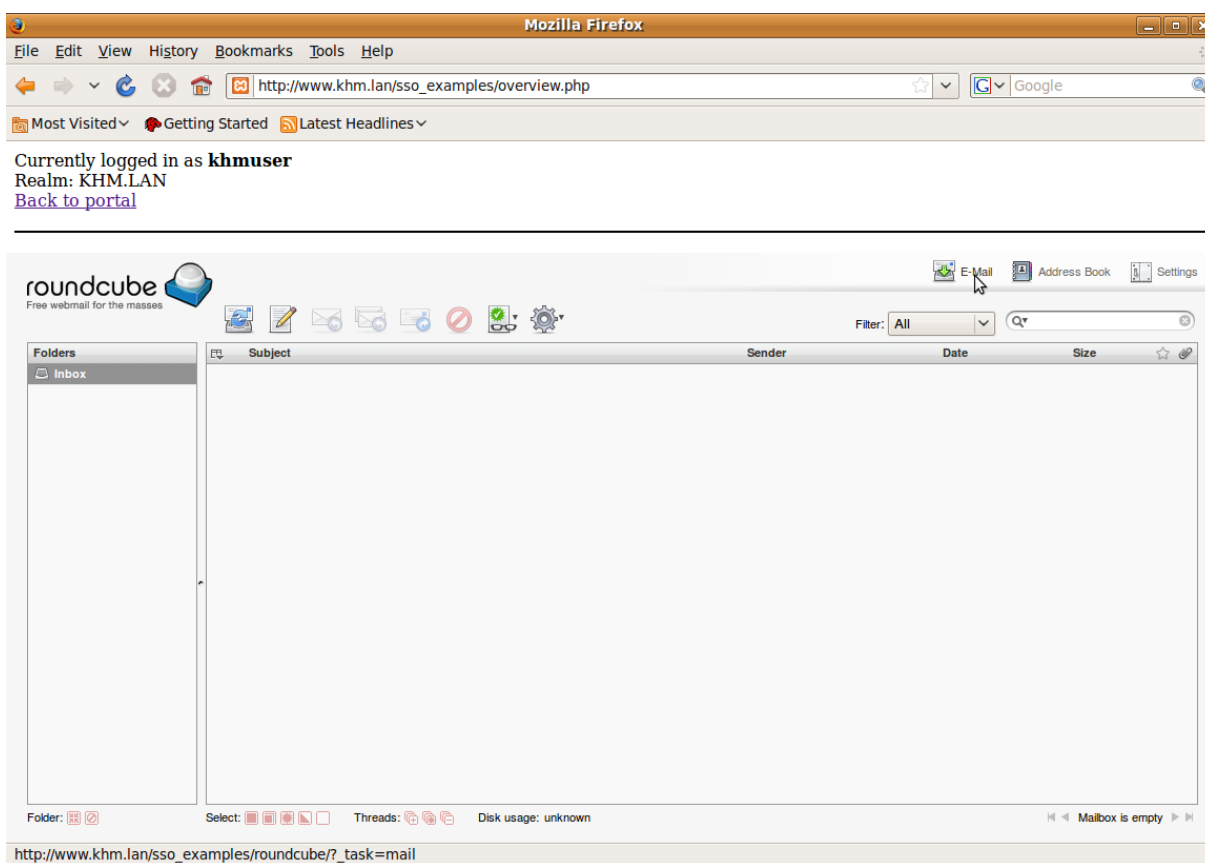
Figure 12.13: Screenshot of the Roundcube web mail interface in the `mod_auth_kerb` *portal.*

12.5.3.7.   Integration with php_krb5 portal

The entire /roundcube directory was copied over to the /kerbexamples directory:

```
sudo cp /opt/lampp/htdocs/sso_examples/roundcube opt/lampp/htdocs/kerbexamples –R
```

Due to the way Roundcube uses session variables, it was not possible to protect the `/roundcube` directory with the `checkcreds.inc` script, as that script relies on session variables in order to function correctly. A user that forgot to log out of the portal would remain logged into the Roundcube web mail interface. Other users would then be able to browse directly to the web mail interface and gain access to the original user's web mail. Closing the browser ends the mail session and will prevent this from happening.

For the same reason, it was not possible to use a modified version of the plugin used in the `mod_auth_kerb` portal. Instead, a PHP login script provided by Philipp C. Heckel was used. The scripts, `RoundcubeLogin.class.php` and `rclogin.php`, were downloaded from his blog at http://blog.philippheckel.com/2008/05/16/roundcube-login-via-php-script/ and placed in the `/kerbexamples/roundcube` directory. The redirect script was renamed to `roundcubelogin.php` and was modified to use the session variables set by the `php_krb5` portal to log in to Roundcube:

```php
<?php

include "RoundcubeLogin.class.php";

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'); }

$roundcube_login = strtolower( $USERNAMECREDENTIALCHECK.'@'.$REALMCREDENTIA
LCHECK );

# Create RC login object.
# Note: The first parameter is the URL-path of the RC inst.,
#       NOT the file-system path
# e.g. http://host.com/path/to/roundcube/ --> "/path/to/roundcube"
$debug = TRUE;

$rcl = new RoundcubeLogin("/kerbexamples/roundcube/", $debug);

try {

   # Try to login and simply redirect on success
  $rcl->login($roundcube_login, "mailadmin");

   if ($rcl->isLoggedIn())
      $rcl->redirect();

   # If the login fails, display an error message
  die("ERROR: Login failed due to a wrong user/pass combination.");
}
catch (RoundcubeLoginException $ex) {
   echo "ERROR: Technical problem, ".$ex->getMessage();
   $rcl->dumpDebugStack(); exit;
}

?>
```

Then, Mozilla Firefox was used to log on onto the `php_krb5` portal and browse to the Roundcube web mail interface:
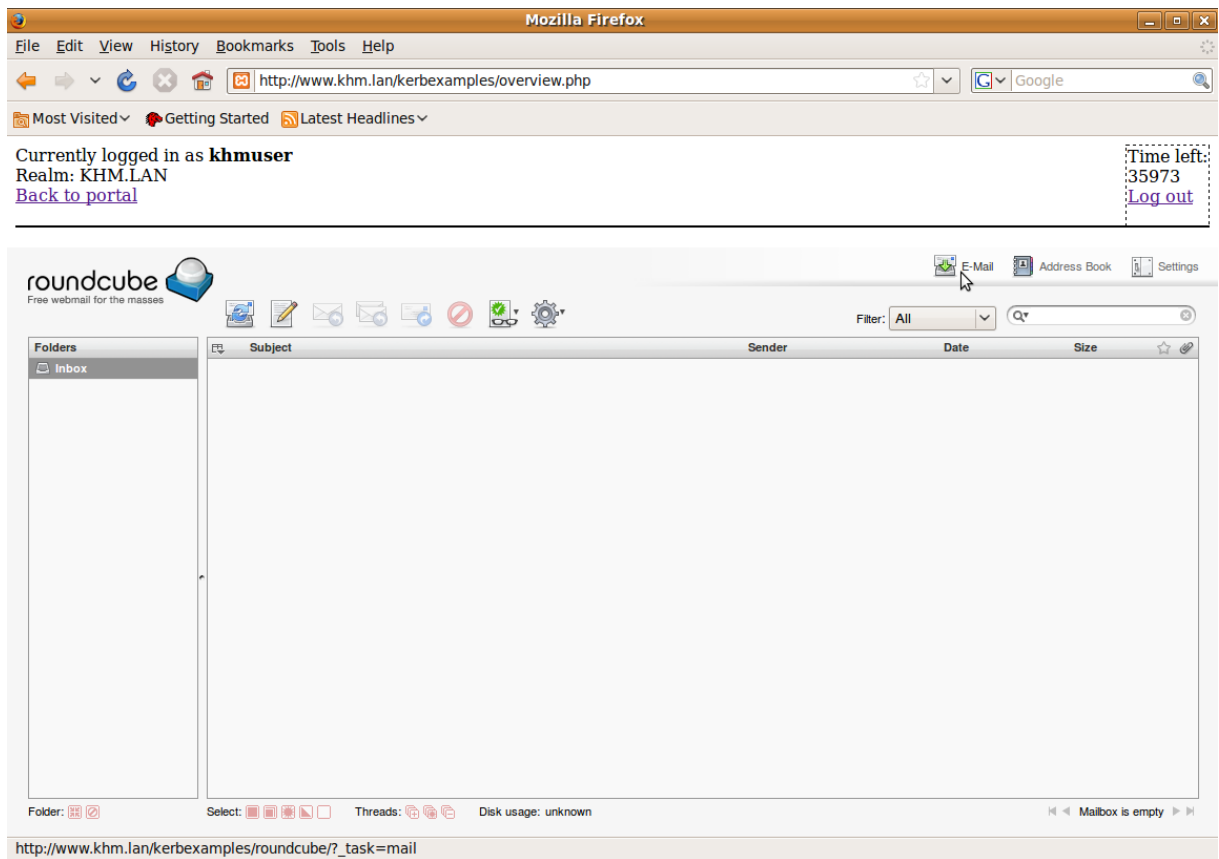
Figure 12.14: Screenshot of the Roundcube web mail interface in the `php_krb5` portal.

13.   Addendum: Linking with Active Directory

The ability to authenticate to a MIT Kerberos KDC to gain access to several kerberized services is a very interesting feature, but many organizations will often already have existing SSO infrastructure in the form of Windows 2000/2003/2008 servers. These operating systems contain an implementation of the Kerberos protocol in the form of Active Directory, a Kerberos KDC with an LDAP backend. It is possible to link Active Directory realms (called "domains") to MIT Kerberos realms.

As the scope of this thesis is to use open-source software for single sign-on services, only the following options will be explored:

- The ability to authenticate as an Active Directory principal on a Linux machine, gaining single sign-on capabilities;
- The ability to use an Active Directory username/password pair with an Apache web server and the `php_krb5` module to gain access to a web portal.

However, many other venues are possible. These will be touched upon in the discussion chapter.

Windows 2003 was used for this thesis, but Windows 2000 and Windows 2008 should also have the same functionalities.

13.1.   Windows 2003 installation

A new virtual machine was installed using Windows 2003 Enterprise Edition and was given a fixed IP address of 192.168.1.201 and 192.168.1.254 as its gateway and preferred DNS server:
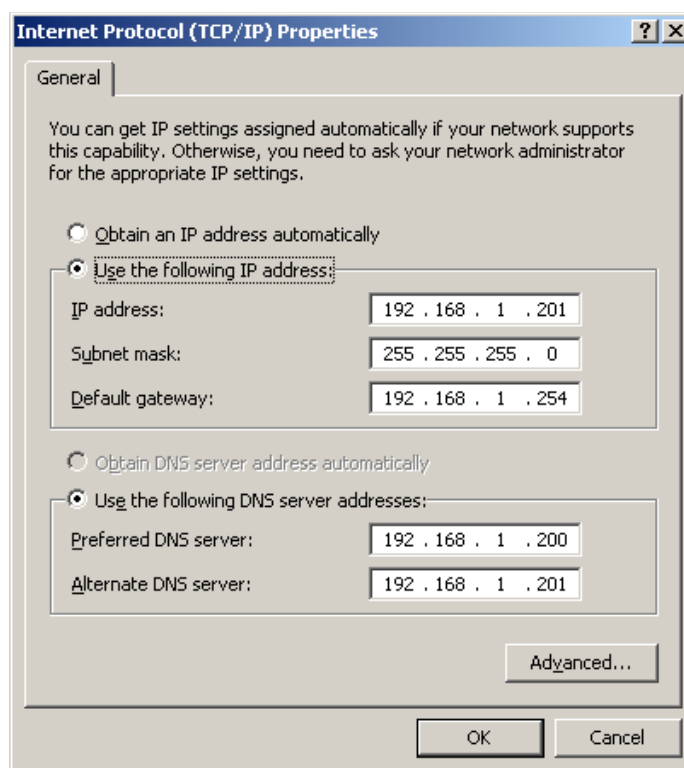
Figure 13.1: Screenshot of Windows 2003 Server IP settings.

The server was allowed to download and install all necessary security updates.

13.2.   Windows 2003 configuration

13.2.1.   Hostname configuration

To begin, the hostname of the computer was changed to `krb`, and the primary DNS suffix was changed to `windows.lan`:
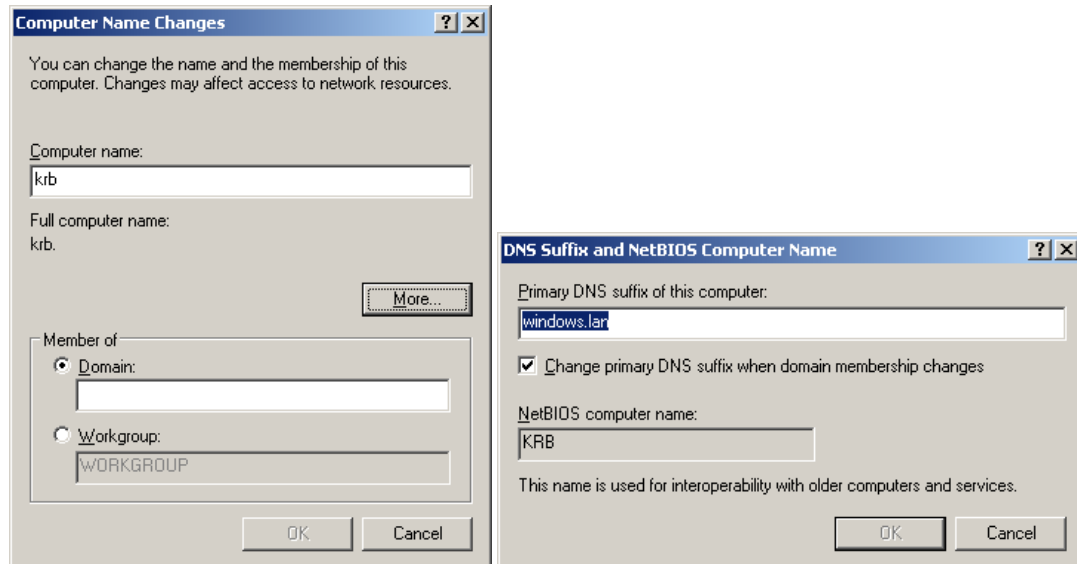
Figure 13.2: Hostname configuration settings for the Windows 2003 server.

The server was then restarted.

13.2.2.   DNS Installation

While Active Directory can be configured to use BIND as its DNS server **[38]**, this was not a necessity for the wanted features, so a Windows DNS server was set up. The DNS software was installed from the Windows 2003 Enterprise Edition disc:
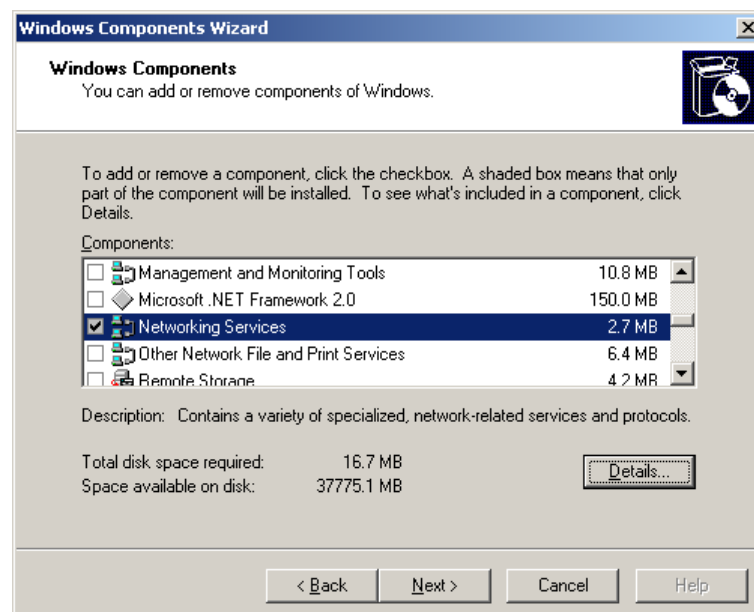
Figure 13.3: Installing a DNS server.

13.2.3.   Active Directory configuration

Now that everything was ready to configure the Active Directory domain, the `dcpromo` command was issued:
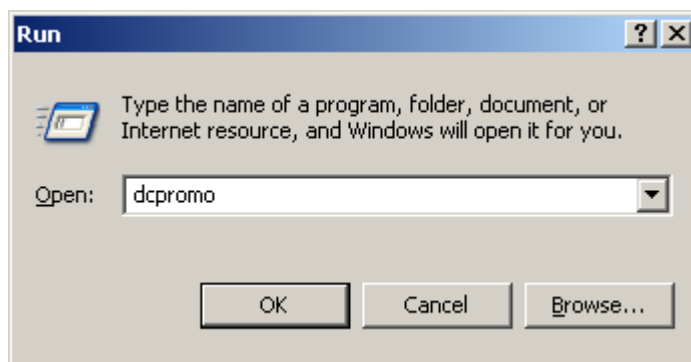
Figure 13.4: Issuing the `dcpromo` command.

While following the configuration wizard, the following options were chosen:

- A domain controller for a new domain was set up.
- This domain was in a new forest.
- The full DNS name was configured as `windows.lan`.
- The NetBIOS name was left as the default.
- The Active Directory database and log file locations were left as their defaults.
- The SYSVOL folder was left at its default value.
- The wizard was allowed to configure the DNS server so that the Active Directory server used it as its preferred DNS server.
- Permissions were set to be compatible only with Windows 2000/2003 operating systems.
- A fitting password for the Restore Mode administrator was entered.

When the wizard was done configuring the Active Directory services, the server was restarted.

A PTR record for `krb.khm.lan` was added to the `1.168.192.in-addr.arpa` zone:
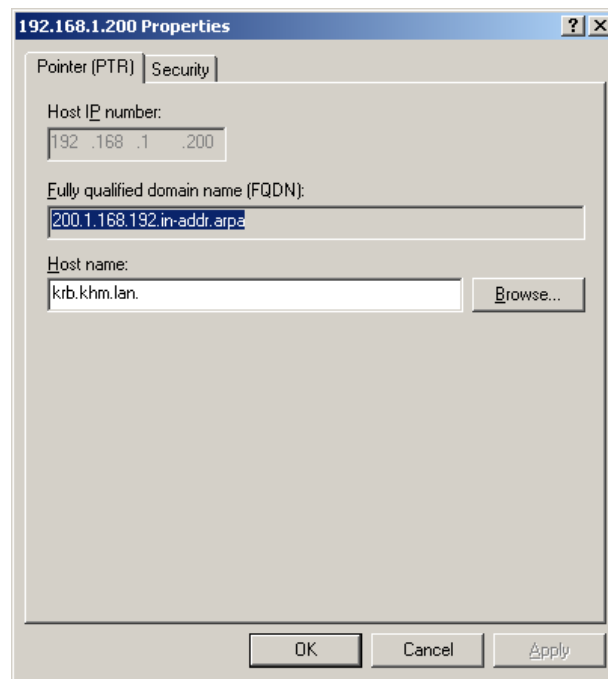
Figure 13.5: Adding a PTR record for krb.khm.lan.

13.2.4.   Configuring the trust

The following files were copied off of the Windows 2003 Enterprise Edition into the `C:\Documents and Settings\Administrator` directory:
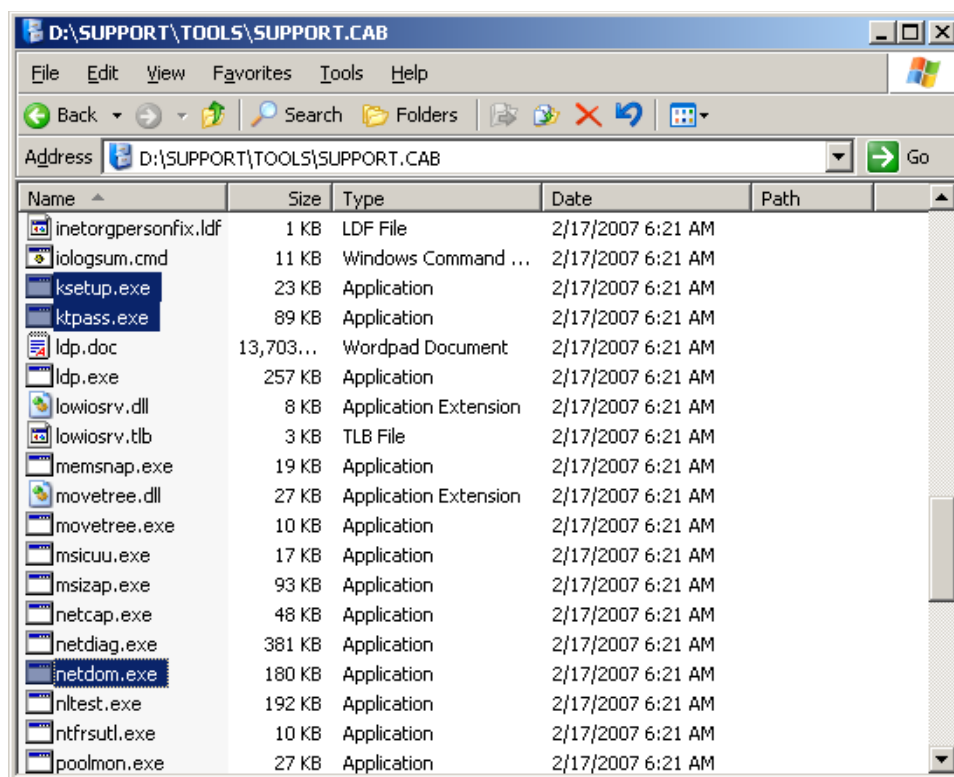
Figure 13.6: Copying the `ksetup.exe`, `ktpass.exe` and `netdom.exe` files to the local hard drive.

- `ksetup.exe` is used to set up a Kerberos realm and add KDCs.

- `ktpass.exe` can be used to map Active Directory users to Kerberos V principals using the `princ` and `mapuser` directives, among other things **[39]**.
- `netdom.exe` is used to verify trusts between Active Directory domains and Kerberos V realms.

Only `ksetup` will be used. Mapping an Active Directory user to a Kerberos V principal is out of the scope of this thesis, and the trust between `WINDOWS.LAN` and `KHM.LAN` will be proven via another way. Still, these tools are great for further experimentation, which is why they were explained and extracted.

A command line was brought up and the following command was issued to register a KDC for the `KHM.LAN` realm:

```
ksetup /addkdc KHM.LAN krb.khm.lan
```

Then, the "New Trust" configuration wizard was started, which can be found under Administrative Tools → Active Directory Domains And Trusts → Right-click on the desired domain → Properties → Tab "Trusts" → "New Trust" button:
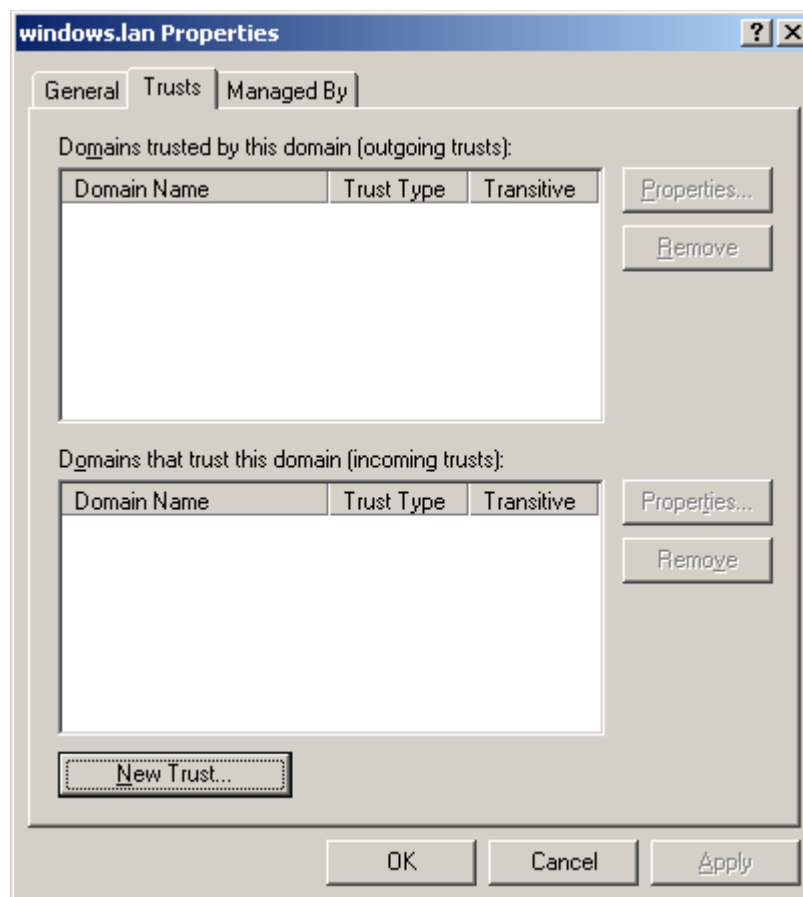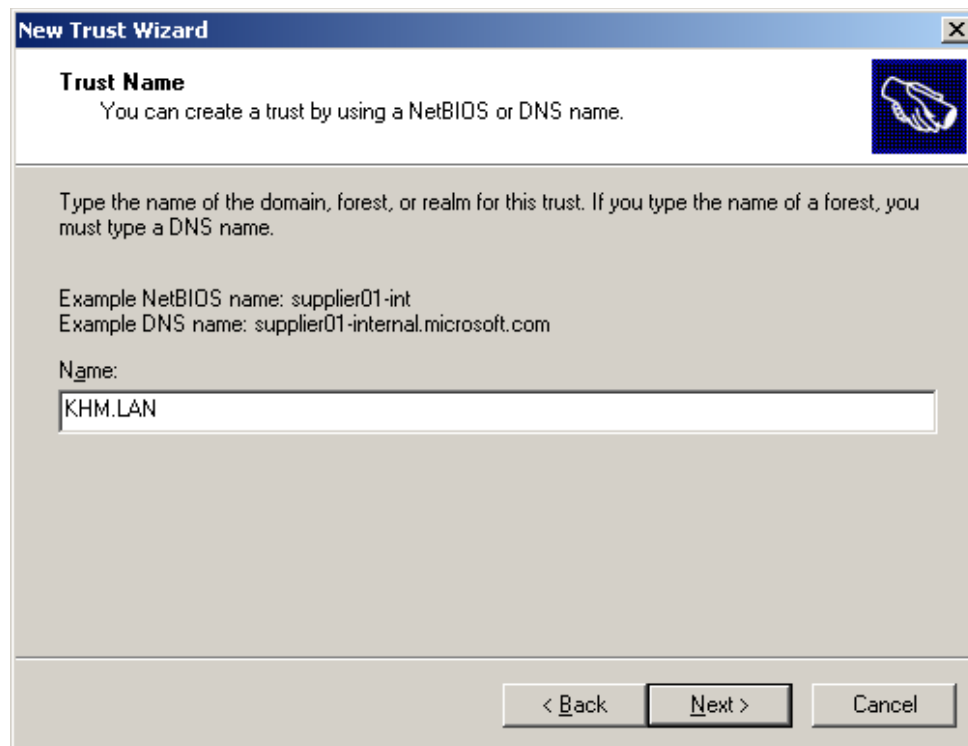
Figure 13.7: The Trusts window.
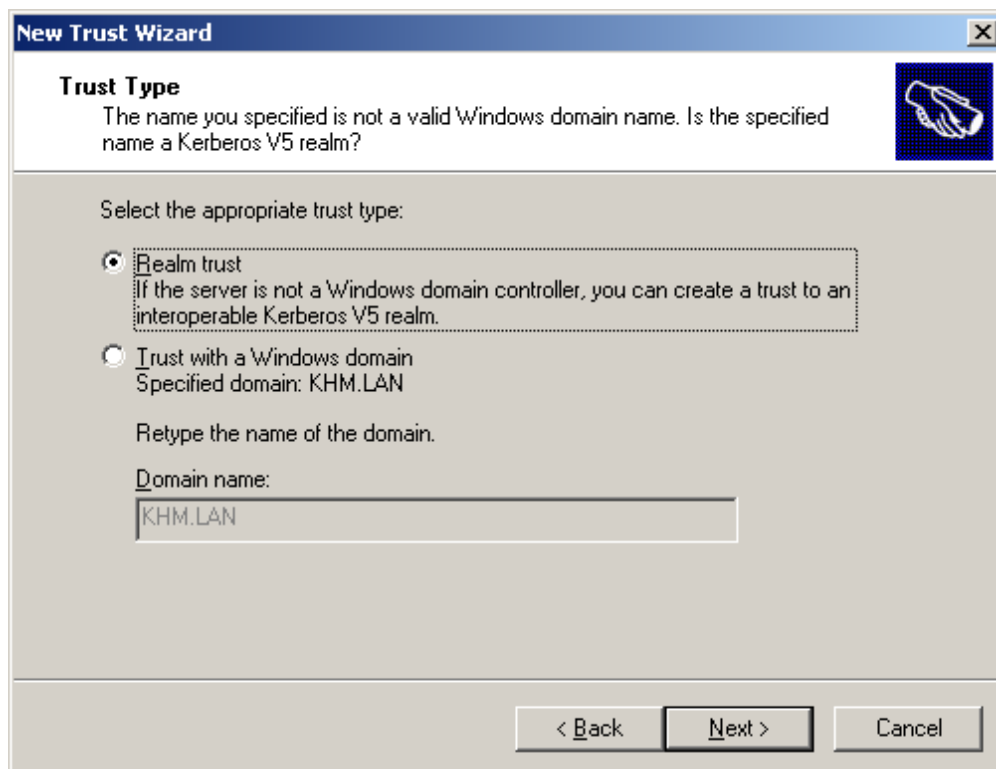
`KHM.LAN` was given as the name of the to be trusted domain. It is crucial that this name is exactly (case-sensitive) the same as the Kerberos realm:

Figure 13.8: Defining the name of the domain that is to be trusted.

This trust was a realm trust between a non-Windows domain controller:

Figure 13.9: Defining the type of trust.

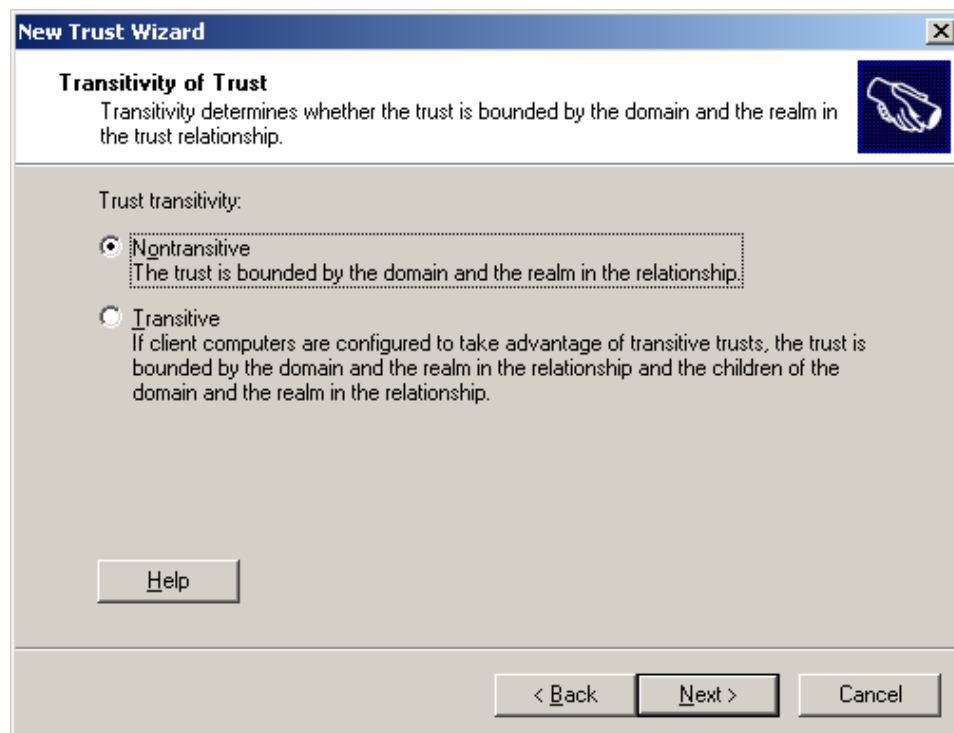This trust was nontransitive, as there was no hierarchical definition of the domains:

Figure 13.10: Defining the transitivity of trust.

This trust was made to be a two-way trust, so a Linux user could `kinit` using an Active Directory user, and the resulting authenticated user was automatically authenticated when accessing the Apache webserver:
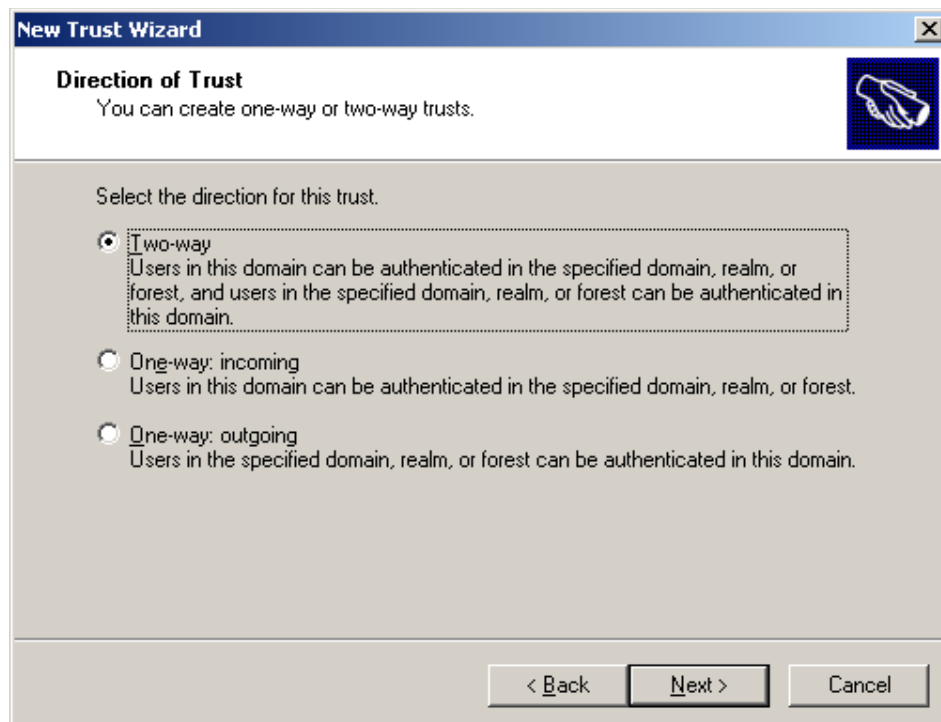

Figure 13.11: Determining the direction of trust.

The passwords for the trust were entered and the configuration wizard was allowed to finish configuring the trust. Then, a command line was brought up and the type of encryption for the trust was set to ArcFour, which is the preferred encryption type of Windows communication:

```
ktpass.exe /MITRealmName KHM.LAN /TrustEncryp rc4
```

13.2.5.   Adding a user

The user `windowsuser` was added to the `WINDOWS.LAN` domain:

Figure 13.12: Adding a user to the `WINDOWS.LAN` domain.

13.3.   Linux server configuration

13.3.1.   Modifying the DNS configuration

In order to communicate with the Windows 2003 server a zone file, `windows.lan.zone`, for the `windows.lan` network was added to BIND, as well as a new PTR record in the `reverse-1.168.192` file:

`named.conf.local`:

```
//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";

zone "khm.lan" in {
      type master;
      file "khm.lan.zone";
```

```
};

zone "windows.lan" in {
      type master;
      file "windows.lan.zone";
};

zone "1.168.192.in-addr.arpa" {
      type master;
      notify no;
      file "reverse-1.168.192";
};
```

`reverse-1.168.192`:

```
$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.
201            IN     PTR    ns.windows.lan.
```

`windows.lan.zone`:

```
$TTL 86400    ;max TTL
$ORIGIN windows.lan.
@      IN     SOA    ns.windows.lan. root.windows.lan. (
                          2010051201   ;SERIAL
                          28800        ;REFRESH 8 HOURS
                          7200         ;RETRY 2 HOURS
                          604800       ;EXPIRE 1 WEEK
                          3600 )       ;MIN TTL 1 HOUR
@            IN     A      192.168.1.201
@            IN     NS     windows.lan.
@            IN     MX     10     mail
mail         IN     A      192.168.1.201
krb          IN     A      192.168.1.201
```

The BIND daemon was then restarted with the following command:

```
sudo /etc/init.d/bind9 restart
```

13.3.2.   Modifying the Kerberos configuration

In order to reach the WINDOWS.LAN Active Directory domain, the following lines were added to /etc/krb5.conf:

```
[realms]
      WINDOWS.LAN = {
            kdc = krb.windows.lan
            admin_server = krb.windows.lan
            default_domain = windows.lan
      }
[domain_realm]
```

```
        .windows.lan = WINDOWS.LAN
       windows.lan = WINDOWS.LAN
```

13.3.3. Configuring the trust

The following commands were issued in the administration server interface:

```
addprinc krbtgt/WINDOWS.LAN@KHM.LAN
addprinc krbtgt/KHM.LAN@WINDOWS.LAN
```

The passwords used were the same ones used when configuring the trust on the Windows 2003 server.

13.4.   Testing the setup

The following commands were issued on the Linux server:

```
kdestroy
kinit windowsuser@WINDOWS.LAN
Password for windowsuser@WINDOWS.LAN:
klist
Ticket cache: FILE:/tmp/krb5cc_1000_OZzvRP
Default principal: windowsuser@WINDOWS.LAN


Valid starting     Expires            Service principal
05/24/10 19:10:25  05/25/10 05:10:07  krbtgt/WINDOWS.LAN@WINDOWS.LAN
       renew until 05/25/10 19:10:25
```

A Ticket Granting Ticket for `WINDOWS.LAN` was successfully received. To test if this ticket could be used to access the `mod_auth_kerb` portal, Mozilla Firefox was used to browse to `www.khm.lan/sso_examples/`:
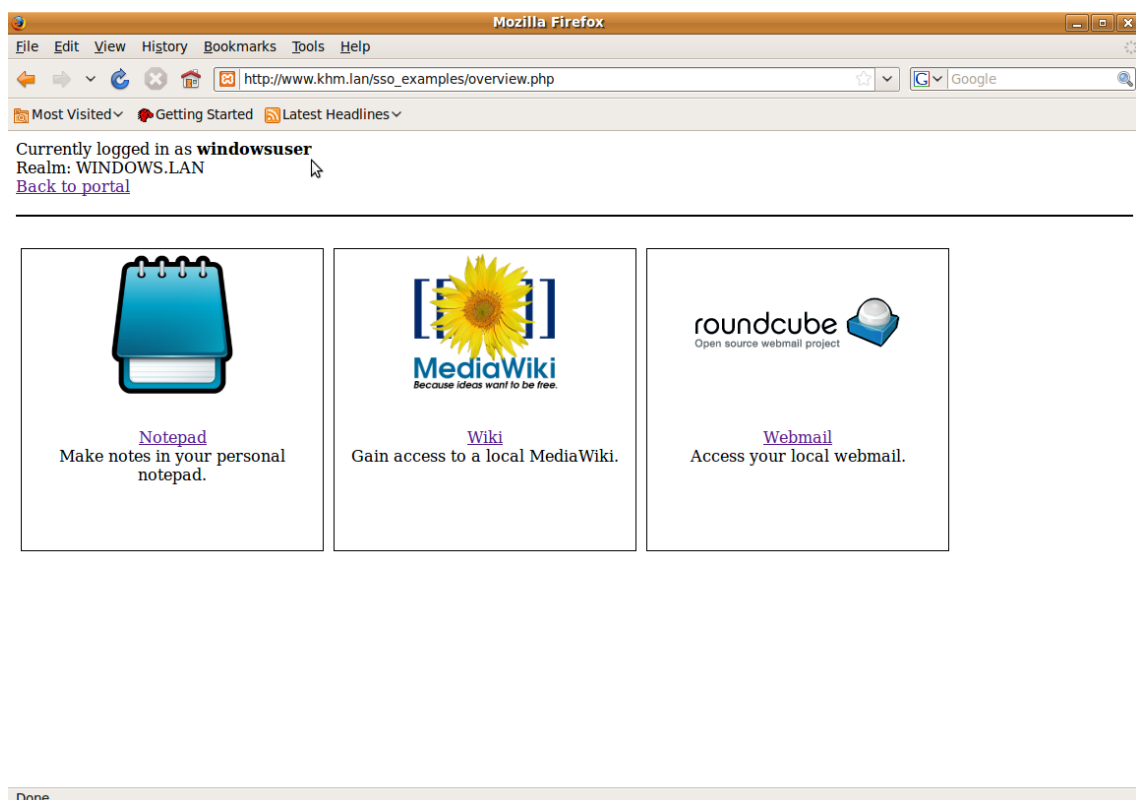
Figure 13.14: Screenshot of the `mod_auth_kerb` portal while authenticated as `windowsuser@WINDOWS.LAN`.

After issuing a klist and doing Wireshark capture, it was confirmed that the two-way trust between the Active Directory domain and the MIT Kerberos V realm was successful:

```
klist
Ticket cache: FILE:/tmp/krb5cc_1000_OZzvRP
Default principal: windowsuser@WINDOWS.LAN
Valid starting     Expires            Service principal
05/24/10 19:10:25  05/25/10 05:10:07  krbtgt/WINDOWS.LAN@WINDOWS.LAN
     renew until 05/25/10 19:10:25
05/24/10 19:13:15  05/25/10 05:10:07  krbtgt/KHM.LAN@WINDOWS.LAN
     renew until 05/25/10 19:10:25
05/24/10 19:13:37  05/25/10 05:10:07  HTTP/krb.khm.lan@KHM.LAN
     renew until 05/25/10 19:10:25
```
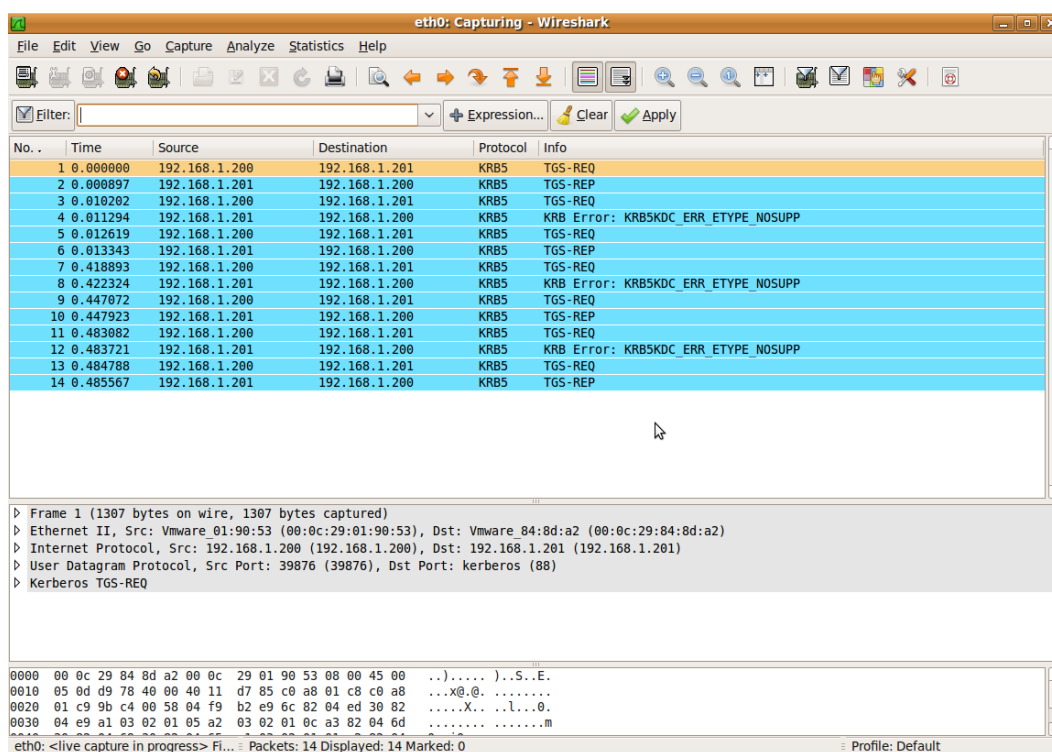

Figure 13.15: Wireshark capture of cross-realm authentication

In the above capture, the following steps occur:

1. The browser requests a Ticket Granting Ticket for the `KHM.LAN` realm, as this is where the Apache web server is located in.
2. The Active Directory domain sends back a Ticket Granting Ticket for the `KHM.LAN` realm.
3. The browser requests on a per-resource basis a Service Ticket for `HTTP/krb.khm.lan`, using the TGT ticket received in step 2.
4. The Active Directory domain sends back a Service Ticket.

The `KRB5KDC_ERR_ETYPE_NOSUPP` errors are caused by the browser asking for 256-bit AES encryption. Unfortunately, Windows 2003 can only communicate cross-realm using RC4 or DES encryption. The second TGS-request is more liberal and sends back a larger selection of encryption types for Windows 2003 to choose from. Windows 2008 Server does support AES **[40]**.

To check if a user was able to log in to the php_krb5 portal using Active Directory credentials, Google Chrome was used on an external Windows machine to browse to `192.168.1.200/kerbexamples/` and log in as `windowsuser` from the `WINDOWS.LAN` domain:
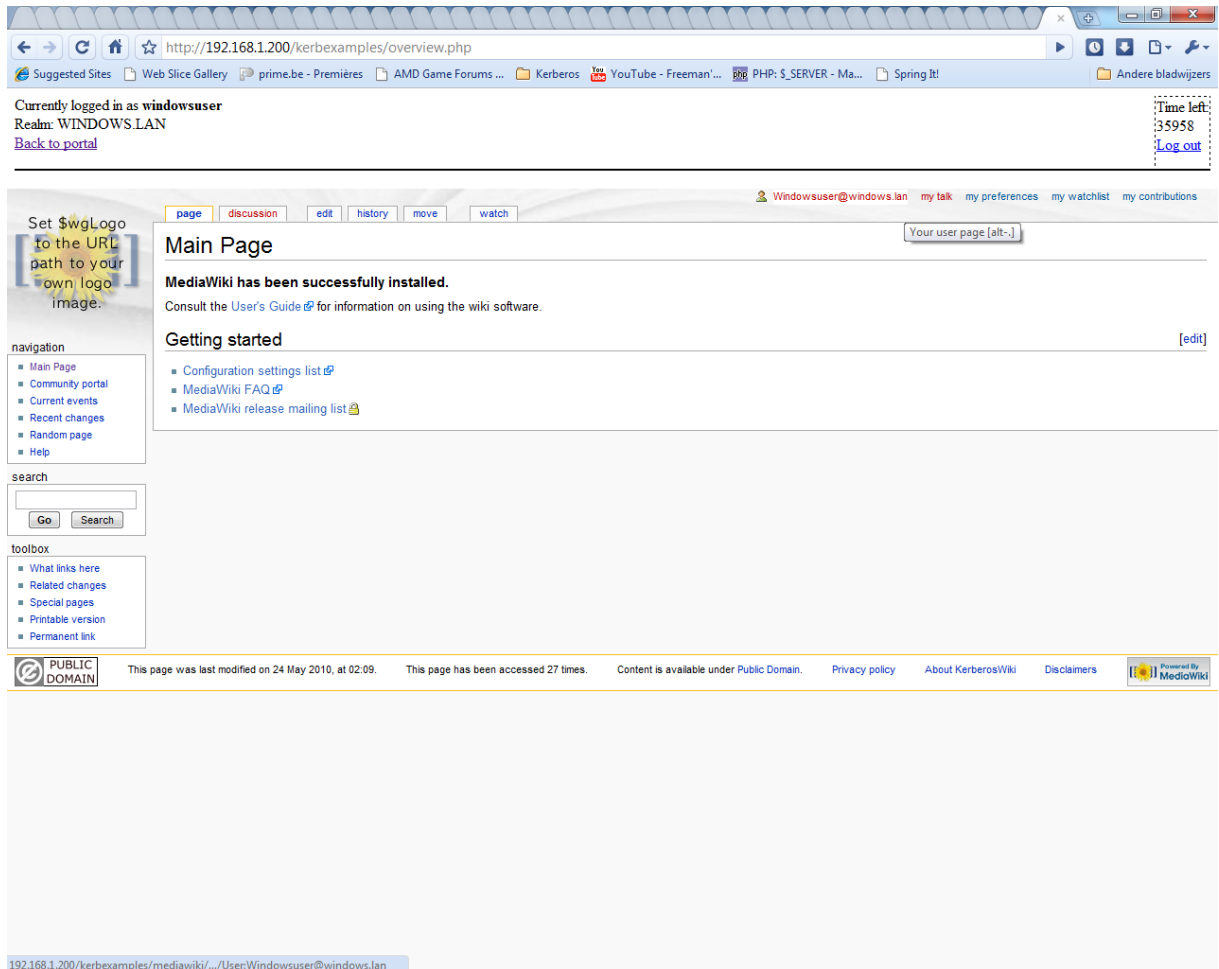

Figure 13.16: Screenshot of the `php_krb5` portal while authenticated as `windowsuser@WINDOWS.LAN`.

14.   Discussion

The created Kerberos authentication system is a viable proof-of-concept for providing intranet single sign-on services, but it could be further extended in several ways. This chapter discusses two such extensions:

1. The possibilities of using the `mod_auth_kerb` and `php_krb5` modules to offer single sign-on capabilities for web services in a secure manner using open-source solutions.
2. The venues of further integration with Windows services.

14.1.   Secure web SSO with mod_auth_kerb and php_krb5

Both `mod_auth_kerb` and `php_krb5` support saving the credentials of a user: `mod_auth_kerb` has the possibility of using the credential cache of a user in a CGI script to execute commands in name of the user, and `php_krb5` supports full-blown GSSAPI bindings as a delegate of the user. It is these GSSAPI bindings that are particularly interesting, as they can be used to communicate in a standardized manner with other applications:

- Java uses JAAS (Java Authentication and Authorization Service) to communicate via GSSAPI with other applications **[41]**;
- Cyrus and Postfix both support GSSAPI authentication **[34]**;
- There are several C plugins to make applications capable of GSSAPI communication, such as PGSSAPI **[42]** and Sun's Mechglue **[43]**, the latter being available in MIT Kerberos' source code.
- The SPNEGO protocol uses GSSAPI to negotiate a security mechanism between two peers **[44]**.
- The SASL (Simple Authentication and Security Layer) framework supports GSSAPI for Kerberos **[45]**.
- There is a Perl module available that provides access to the GSSAPI library **[46]**.

At the time of writing, there were no sample applications available that demonstrated communication between the `php_krb5` module and another GSSAPI-aware application. Some organizations have succeeded in using GSSAPI authentication with web mail services, but none of them used Kerberos as the primary authentication service, instead relying on a separate authentication service such as Pubcookie or Cosign and requesting Kerberos service tickets using the credentials from those authentication services **[36, 37]**.

To illustrate the flexibility of PHP-based GSSAPI authentication, a few scenarios were thought up:

- Web mail: Using `php_krb5`, a web mail interface could easily interface with GSSAPI-aware SMTP and IMAP servers such as Cyrus and Postfix using only Kerberos authentication.
- Web-based telnet or SSH clients could be modified to perform GSSAPI authentication using the authenticated user's credentials, allowing any user to easily and securely access devices remotely with his or her corresponding privileges.
- Web-based remote desktop applications could be modified to use GSSAPI authentication to allow secure access to devices in a graphical environment. For example, RealVNC uses a Java Applet to connect to a computer remotely in a graphical environment **[47]**.

- Existing web application frameworks such as Horde could be modified to authenticate to its services (web mail, calendars, groupware solutions, etc) using GSSAPI.
- Applications developed In-house could present a web interface that uses GSSAPI to authenticate users and send its data in a secure manner.

Native Kerberos authentication has never been used much for extensive web SSO. One reason is that web-capable modules that use Kerberos authentication have only recently been developed: `mod_auth_kerb` was conceived in 2002 **[48]**, `mod_spnego` started in March 2003 **[49]**, with SPNEGO only supporting Kerberos V5 authentication in 2005 **[44]**. Instead, it is used as a secure method of authentication for another central authentication service such as CAS, Pubcookie, CoSign or Shibboleth. These systems often had a head start of a few years **[50, 51]**.

While many of these central authentication services are also open-source, they are often limited to a certain programming language, infrastructure or both. CAS, for example, runs on a Java server component **[52]**. The extra abstraction layers these applications produce make it hard to combine resources from multiple central authentication services, unless they are stacked upon one another **[53]**. Using Kerberos authentication and GSSAPI in its native forms allow applications to be more easily kerberized without having to rely on an extra authentication service. It will be interesting to see how the open-source community will use this new PHP module to create GSSAPI-capable web portal sites and –applications.

14.2.   Integration with Windows services

While some limited integration with Windows 2003 server was achieved, further integration with Active Directory domains is possible, such as linking Linux principals to Windows users, allowing a Linux user to log into Windows workstations and the other way around. Unfortunately, a Windows user was not able to use his or her credentials to log onto a Linux machine: A Ticket Granting Ticket was correctly fetched for the `KHM.LAN` realm, but the login action still failed. It is possible that the `pam_krb5` module does not support authenticating to a Active Directory domain.

Internet Explorer supports integrated Windows authentication, which uses the SPNEGO mechanism to provide web single sign-on capabilities **[54]**. This feature was tested out on a Windows XP Professional workstation: a Ticket Granting Ticket for the `KHM.LAN` realm and a Service Ticket for the Apache web server (running on a Ubuntu server) were fetched by Internet Explorer, but the final HTTP request that contained the authentication header was never made. It is possible that this is related to the haphazard implementation of the DNS services of the Windows 2003 server, as Kerberos relies very heavily on correctly configured DNS services, even using the domain names as salts for encryption **[16]**.

MIT Kerberos also integrates well with Samba **[55]**, allowing Kerberos authentication to be used for file shares and printers for Windows and Unix/Linux alike.

Still, the simple integration that was achieved is ideal for organizations that wish to use GSSAPI-enabled web services that rely on the `php_krb5` module, allowing a user to authenticate him- or herself via a web portal and letting the web browser communicate with the other applications as a delegate for the user.

15.   Conclusion

Kerberos is a well-established authentication protocol that has been implemented in a wide array of software, protocols and IETF standards. Its continued development by MIT has allowed it to continue to be used in modern single sign-on environments.

Kerberos' security lies in its trusted third party: The Key Distribution Center (KDC) is trusted by all other parties to provide secure communication between everyone. It relies on symmetric keys and randomly generated session keys to achieve its high level of security. A side effect of the way Kerberos conducts its authentication is that single sign-on capabilities are achieved, which makes it very attractive for intranet use or in situations where many services rely on password authentication.

The trusted third party is also the greatest weakness, however: If the KDC is compromised, all encrypted passwords may be appropriated by the hacker and brute forced. A compromised KDC could also allow unsecure or malicious hosts to be trusted, which can be used to mislead users in thinking they are accessing a secure host. After all, Kerberos is meant to authenticate users to trusted hosts on unsecure networks. For these reasons, it is imperative the KDC is well protected.

The user itself is undoubtedly the biggest security issue: Compromised workstations could contain trojans or keyloggers that steal the unencrypted password and funnel it through to a malicious user, who can then authenticate as the victim to all the services the victim has access to. A strict password policy should be enforced and up-to-date antivirus and firewall software should be installed on every workstation to minimize this risk.

Installing and configuring a Kerberos environment is relatively easy, and allows for an administrator to manage passwords in one central location. Modules that extend Kerberos' single sign-on capabilities to the web continue to be developed, with `mod_auth_kerb` being a stable and easy to install module for Apache. The very recent `php_krb5` PHP module is the most interesting one: its capabilities to do GSSAPI bindings with other applications in PHP code may result in some very interesting kerberized web applications.

At the moment, however, web single sign-on using a native Kerberos and GSSAPI solution is still rather limited, especially in communicating with other services such as mail servers.

Interfacing a Linux Kerberos realm with a Windows Active Directory domain, is easily done, and allows many fruitful combinations to occur. Simple linking of domains and realms is an interesting venue when investing in web single sign-on portals, which the previously mentioned modules will hopefully make more attractive.

When implemented correctly, Kerberos is a robust and secure authentication protocol that easily interfaces with both native mechanisms such as GSSAPI and extra central authentication services such as CAS, Pubcookie, CoSign or Shibboleth. Its continued use in both federal as institutional networks is a testament to its legacy of stable and secure authentication.

16. References

[1] Novel, "Novel securelogin", http://www.novell.com/products/securelogin/media.html, 2010

[2] MIT, *"Kerberos: The network authentication protocol"*, http://web.mit.edu/Kerberos/, 8 april 2010

[3] Cheryl Gribble, *"Tutorial for Kerberos"*, http://www.hitmill.com/computers/kerberos.html, 1 Februari 2009

[4] *Kasslin K., Tikkanen A., Virtanen T., "Kerberos V Security: Replay Attacks",* http://users.tkk.fi/u/autikkan/kerberos/AIWSC03_kerberos_replay_attacks.pdf, 2003

[5] *Alex Brennen, "Kerberos Infrastructure HOWTO",* http://tldp.org/HOWTO/Kerberos-Infrastructure-HOWTO/overview.html, 29 May 2004

[6] András Salamon, *"DNS related RFC's"*, http://www.dns.net/dnsrd/rfc/, 2 June 2004

[7] Wiki community, *"Domain Name System"*, http://en.wikipedia.org/wiki/Domain_Name_System, 23 May 2010

[8] Raxxal, *"DNS Server with Suse Linux 10.0"*, http://en.opensuse.org/DNS_server_with_SUSE_Linux_10.0, 13 January 2006

[9] Lars M. Hansen, *"Writing zone files"*, http://www.hansenonline.net/Networking/zonefile.html, 16 May 2010

[10] Docelic, *"MIT Kerberos installation on Debian"*, http://www.debian-administration.org/articles/570#krb-install, 5 december 2007

[11] 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

[12] OpenSuse community, "Example Configuration of Kerberos 5 Server on OpenSUSE 10.2", http://en.opensuse.org/Howto_KRB_server, 25 April 2009

[13] Romain Lenglet, "How to install Kerberos 5", http://www.berabera.info/oldblog/lenglet/howtos/kerberoshowto/index.html, 8 November 2005

[14] Stone, "Automatic kinit upon login", http://yoten.blogspot.com/2009/04/automatic-kinit-upon-login.html, 22 April 2009

[15] "Kerberos on Ubuntu", http://www.alittletooquiet.net/text/kerberos-on-ubuntu/,

[16] Fulvio Ricciardi, "Kerberos Protocol tutorial », http://www.kerberos.org/software/tutorial.html, 11 November 2007

[18] Wiki community, "Pluggable authentication modules", http://en.wikipedia.org/wiki/Pluggable_Authentication_Modules, 15 April 2010

[19] Wiki community, "Secure Shell", http://en.wikipedia.org/wiki/Secure_Shell, 17 May 2010

[20] Kai Seidel, "xampp for Linux", http://www.apachefriends.org/en/xampp-linux.html#374, 22 Februari 2009

[21] Mod Auth Kerb Staff, "Ticket File/Credential Cache Saving", http://modauthkerb.sourceforge.net/configure.html#saving, 28 April 2004

[22] Achim Grolms, "KrbSaveCredentials on how can I use the credentials cache?", http://thread.gmane.org/gmane.comp.apache.mod-auth-kerb.general/978, 21 June 2006

[23] K. Jaganathan, "SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows", http://tools.ietf.org/html/rfc4559, June 2006

[24] Sanj Surati & Michael Muckin, "HTTP-Based Cross-Platform Authentication via the Negotiate Protocol", http://msdn.microsoft.com/en-us/library/ms995329, December 2002

[25]   Oracle, "Configuring KDC Servers", http://docs.sun.com/app/docs/doc/816-4557/setup-9?a=view, 7 October 2009

[26]   Moritz Bechler, "php krb5 release candidate", http://mbechler.eenterphace.org/blog/index.php?/archives/10-php_krb5-release-candidate.html, 28 november 2009

[27]   J. Wray, "RFC2744 - Generic Security Service API Version 2 : C-bindings", http://tools.ietf.org/html/rfc2744.html, January 2000

[28]   Wiki community, "Load Balancing (computing)", http://en.wikipedia.org/wiki/Load_balancing_(computing), 13 May 2010

[29]   "Round Robin DNS information and Services", http://www.rrdns.com, 2010

[30]   "MediaWiki", http://www.mediawiki.org/wiki/MediaWiki, 11 May 2010

[31]   "Category: All Extensions" http://www.mediawiki.org/wiki/Category:All_extensions, 18 September 2009

[32]   "Extension:AutomaticREMOTE_USER", http://www.mediawiki.org/wiki/Extension:AutomaticREMOTE_USER, 6 May 2010

[33]   Ivar Abrahamsen, "How to set up a mail server on a GNU / Linux system", http://flurdy.com/docs/postfix/, 10 May 2010

[34]   "Single sign-on e-mail with GSSAPI/Kerberos authentication", http://www.codealias.info/technotes/single_sign-on_e-mail_solution_with_gssapi_kerberos_authentication

[35]   "webmail and GSSAPI authentication to imapd – Kerberos", http://fixunix.com/kerberos/60293-webmail-gssapi-authentication-imapd.html, 10 February 2007

[36]   Dave McMurtrie, "Squirrelmail Kerberos", http://cyrusimap.web.cmu.edu/twiki/bin/view/Cyrus/SquirrelMailKerberos, 14 August 2008

[37]   Simon Wilkinson, "webmail using Kerberos tickets", http://www.mail-archive.com/cosign-discuss@lists.sourceforge.net/msg00044.html, 20 February 2008

[38]   Bob Carver, "Linux to Windows migration", http://technet.microsoft.com/en-us/library/dd316373.aspx, 1 June 2001

[39]   Microsoft, "Step-by-Step Guide to Kerberos 5 (krb5 1.0) Interoperability", http://technet.microsoft.com/en-us/library/bb742433.aspx, January 2000

[40]   Heimdal Organization, "Inter-Realm keys (trust) between Windows and a Heimdal KDC", http://www.h5l.org/manual/heimdal-1-3-branch/info/heimdal/Inter_002dRealm-keys-_0028trust_0029-between-Windows-and-a-Heimdal-KDC.html, 23 November 2009

[41]   Sun, "Introduction to JAAS and Java GSS-API Tutorials", http://java.sun.com/j2se/1.5.0/docs/guide/security/jgss/tutorials/index.html, 18 December 2007

[42]   "Pluggable GSSAPI", http://rc.quest.com/topics/pgssapi/, 10 April 2007

[43]   "GSSAPI Mechglue" http://grid.ncsa.illinois.edu/gssapi-mechglue/, 6 October 2009

[44]   L.Zhu, P. Leach, K. Jaganathan, "The Simple and Protected Generic Security Service Application Program Interface (GSS-API)Negotiation Mechanism", http://tools.ietf.org/html/rfc4178, October 2005

[45]   Melnikov, "The Kerberos V5 ("GSSAPI") SASL mechanismdraft-ietf-sasl-gssapi-08" http://tools.ietf.org/html/draft-ietf-sasl-gssapi-08, 1 September 2006

[46]   Achim Grolms, "Perl GSSAPI bindings", http://perlgssapi.sourceforge.net/, 5 May 2009

[47]   RealVNC , "Java VNC® Viewer", http://www.realvnc.com/support/javavncviewer.html, 26
        April 2010

[48]   mod_auth_kerb crew, "Kerberos Module For Apache",
        "http://sourceforge.net/projects/modauthkerb/files/?sort=date&sortdir=desc, 12 April 2008

[49]   Möller M., "Apache Kerberos/SPNEGO module",
        http://sourceforge.net/projects/modgssapache/files/?sort=date&sortdir=asc, 11 April 2007

[50]   Wikipedia community, "Shibboleth (Internet2)",
        http://en.wikipedia.org/wiki/Shibboleth_(Internet2), 25 March 2010

[51]   Wikipedia community, "Pubcookie", http://en.wikipedia.org/wiki/Pubcookie, 9 January 2010

[52]   Jasig, "CAS | Jasig Community", http://www.jasig.org/cas, 16 December 2009

[53]   Gilbert H., "Shibboleth 1.3", http://www.ja-sig.org/wiki/display/CAS/Shibboleth+1.3, 17
        March 2005

[54]   Wikipedia community, "Integrated Windows Authentication",
        http://en.wikipedia.org/wiki/Integrated_Windows_Authentication, 5 April 2010

[55]   Vinayak H., Premalatha, Escalante M., "MIT Kerberos Enhancements for Samba",
        http://www.sambaxp.org/uploads/media/04-Vinayak_Hedge_-
        _MIT_KDC_Enhancements_for_Samba.pdf, 12 May 2005