Kerberos and SESAME

Mats Persson and Ulf Lindbergh

September 6, 1999

Abstract

Today many users have to login to each computer they want to access and each time their passwords is sent in clear text over the network. This is a serious security problem and it would be much simpler to authenticate the user only once. To solve this problem two security systems have been developed, Kerberos and SESAME. We have installed and tried these two systems and this report contains our observations.

1 Introduction

In the last decades the organization of computers and their networks has shifted from mainframes to smaller computers connected in heterogenous networks, even involving the global internet. A consequence of this was that people had to login at each computer in the network to perform their tasks, and they had to remember a password for every login. These passwords were often sent in clear over the network, which is a security problem because other people can listen to the network and catch the passwords. This can be a serious problem even in a internal network like an intranet. One solution is to use Kerberos or SESAME.

Kerberos and SESAME have several things in common and basically offers the same service. They are two security systems that allows users to securely login once to a computer network at a single point. The users are authenticated by giving a password, and then they get a ticket which they can use to access other applications in the network. In SESAME the ticket even contain access rights. Also password is only needed when SESAME is used in Kerberos compatability mode.

Both systems are based on the client-server model. There are authentication servers, a ticket server, and application servers. These servers have exchanged cryptographic keys in advance so they can transfer the users authentication and privilege credentials. This means that users do not have to send passwords over the network at all.

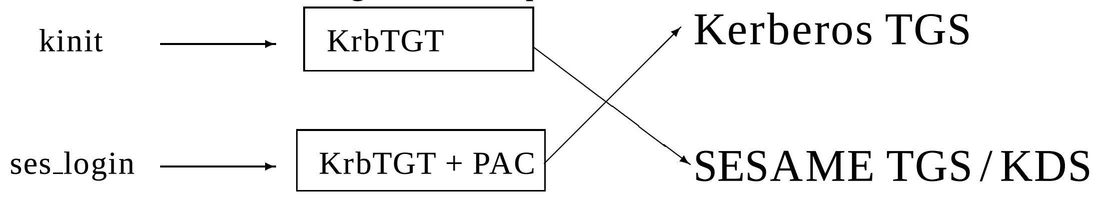
2 Description of project

This project is a part of the Applied Network Security course at the university in Linkoping. The task was to install and configure Kerberos and SESAME, and to see if they are compatible.

It was not very hard to understand the first part of the task, how to install and configure either Kerberos or SESAME. It was the second part about compatibility that was a bit unclear. Are the protocols compatible or is the current implementations compatible with each other? If it is the first question, how could two different protocols be compatible?

We got some advice from our supervisors and figure 1 shows how the compatible tickets in SESAME and Kerberos were going to be used. The user logs in with the "kinit" program and gets a "Kerberos ticket granting ticket" which the user then should use to get access to applications from the SESAME ticket granting server. And vice versa when starting from "ses_login".

Figure 1: Compatible tickets?



3 Kerberos

This section describes Kerberos and the installation and configuration.

3.1 Purpose

The primary purpose of Kerberos was to provide a secure network authentication service. Kerberos has two main goals:

1. To allow a user single sign-on to the network.
2. To protect the authentication information so that masquerading is far more difficult.

3.2 Description

Kerberos is a trusted third-party service. That means that there is a third party (the kerberos server) that is trusted by all the entities on the network (users and services, usually called ”principals”). All principals share a secret password (or key) with the kerberos server and this enables principals to verify that the messages from the Kerberos server are authentic. Thus trusting the kerberos server, users and services can authenticate each others.

In order to make this work the following assumptions was made:

- The time is trusted
- The user trusts their workstation completely
- The security server is on-line
- The servers are stateless
- Because of the existing patents on public-key cryptography, only symmetric key cryptography would be used
- The time that the user’s pasword is available on the client machine needs to be minimized

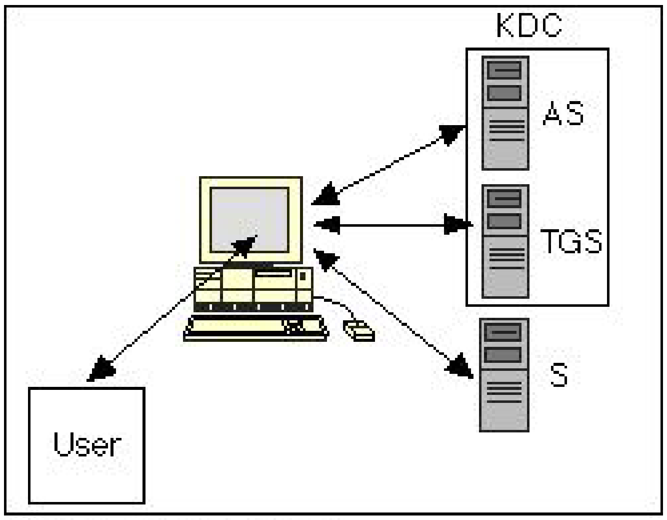


Fig. 1 Overview of Kerberos

Figure 1 shows the components of the Kerberos system: the User, their Client workstation, the Authentication Server (AS), the Ticket Granting Server (TGS) and the Application Server (S).

The purpose of the authentications server is to define which users are entitled to log into the system and to provide these users with the possibility to authenticate to the system.

The Ticket Granting Server (TGS) generates the server application ticket for the client using the information obtained from the client's Ticket Granting Ticket (TGT). TGT is obtained from the authentication procedure and the session key in it prove the clients identity so the user never has to re-enter a password. TGS also generates and distributes session keys so that the communication between clients and applications server can be crypthographically protected. AS + TGS = Security Server or Key Distribution Center (KDC).

The Application Server (S) may provide a number of applications to the various client machines.

Before the protocol starts, long term symmetric keys are shared between each user A and the authentications server (K_A), between the authentic server and the ticket granting server (K_{AS-TGS}) and between the ticket granting server and an application server (K_{TGS-S}). The administrative (security) domain of Kerberos is called realm.

For a detailed description of the Kerberos protocol refer to chapter 5.5 in [1].

3.3 Installation

We installed one precompiled release of the free version of Kerberos 5 called “Heimdal” on a Sun Solaris and compiled one later release on a Linux machine. On both systems we had to be root to install it.

1. From Internet we located the path, http://www.pdc.kth.se/heimdal/

2. As root we created a new directory called `local` placed in the directory `/opt`, which is placed on our local machine. In `/opt/local` we created the directory `kerberos`.

3. From Internet we downloaded into the directory `/opt/local/kerberos` a precompiled (bin) release of Kerberos called *Heimdal*. Heimdal is a free implementation of Kerberos 5.

4. The file you get is tar-compressed and unpacking this will give you a new directory called `heimdal-0.1e`.

   In this directory you now find the directory `bin, include, info, lib, libexec, man` and `sbin`.

   In the directory you will find the document called `heimdal.info` which describes this version of Kerberos.

3.4 Configuration and running

The configuration and running was fairly simple once you understood the configuration file `krb5.conf`.

1. **Setting up realm**. Give your realm (security domain) a name. We gave it the name LIN.FOA.SE

2. Create a configuration file: `/etc/krb5.conf`. There is a sample `krb5.conf` file supplied with the download (appendix 1).

3. **Creating the database**. Create the directory `/var/heimdal`

   This is where all the principals are stored in the database. These can be encrypted with a master key but we chose not to encrypt it at this stage because we wanted to see how they were stored. Hopefully it will be possible to encrypt the database later because encrypting this database is vital for security.

4. Make the right path

   ```
   LD_LIBRARY_PATH=/opt/local/kerberos/heimdal-0.1e/lib
   export LD_LIBRARY_PATH
   ```

5. Initialise the database

   ```
   #sbin/./kadmin -l
   kadmin>init MY.REALM
   Realm max ticket life [unlimited]:
   Realm max renewable ticket life [unlimited]:
   ```

6. Add yourself to the database

   ```
   kadmin>add me
   Max ticket life [unlimited]:
   Attributes []:
   Passwords:
   Verifying password - Password:
   ```

7. Start the Key Distribution Center (KDC)

   ```
   #libexec/./kdc &
   ```

8. Get a ticket

   ```
   #./kinit me
   me@MY.REALMS's password:
   #bin/./klist
   ```

   To extract a service ticket from the database and put it in a keytab you need first to create the principal in the database and then extract it.

9. The **keytab** is a key table file containing one or more keys. A host or service uses a keytab file in much the same way as a user uses his/her password.

   First we need to create the principal with `add` and then extract it with `ext`.

   ```
   kadmin>add --random host/my.host.name
   Max ticket life [unlimited]:
   Max renewable life [unlimited]:
   Attributes []:
   kadmin>ext host/my.host.name
   ```

10. Extract a service ticket

    ```
    kadmin>ext host/my.host.name
    #sbin/./ktutil list
    ```

11. **Remote administration**. Add the following line to `/etc/inetd.conf`

    ```
    kerberos-adm stream tcp nowait root /usr/heimdal/libexec/kadmind k
    ```

    Add kerberos-adm to `/etc/services` as 749/tcp

We could not continue this update because Heimdal was an alpha-release of the implementation. There wasn't any demo-application that worked either.

3.5 Comments

There has been a lot of discussion about Kerberos and its security. One longer article [3] pointed out several limitations of Kerberos, especially in the protocols but also in the whole design.

Another article about a different security architecture [2] mentioned two weaknesses in Kerberos.

- Kerberos has no redundancy, ie if one server is compromised the whole system is undermined.

- Requires synchronous communication with the servers, which can mean low performance. A stateless protocol would have made it more robust.

4 SESAME

This section describes SESAME and the installation and configuration process. It also contain some comments and opinions about SESAME.

4.1 Description

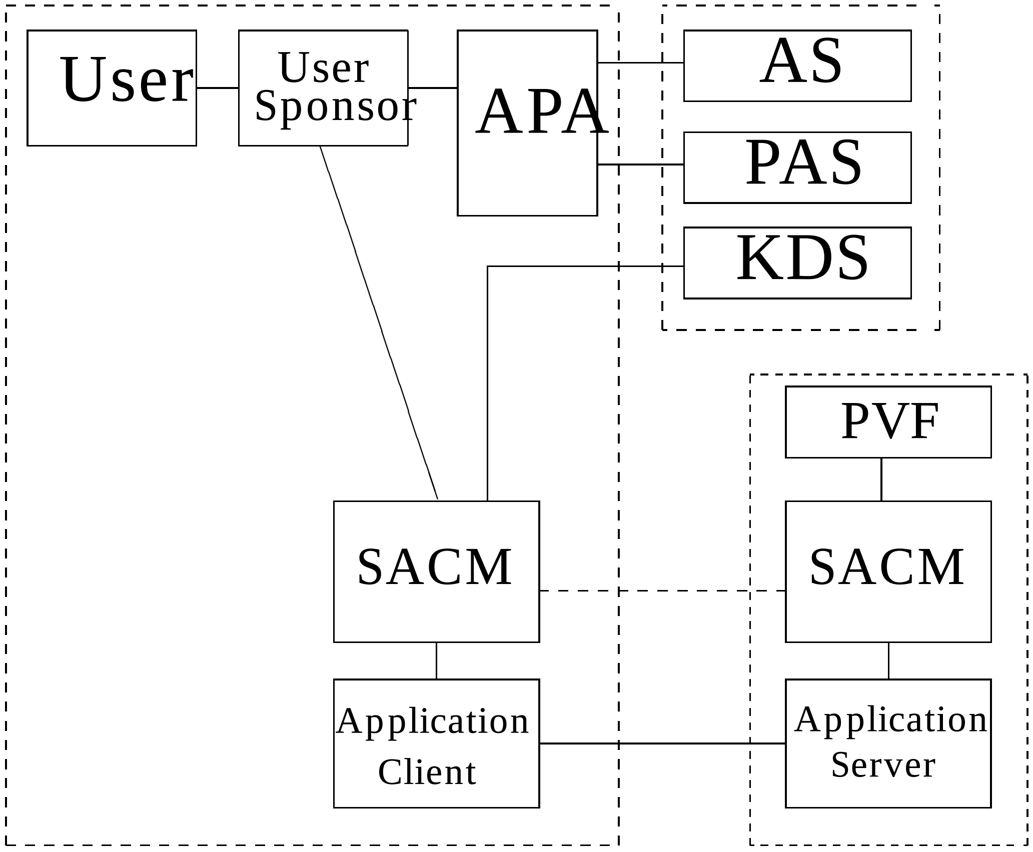
SESAME is an acronym for “Secure European System for Applications in a Multi-vendor Environment”. It is a network authentication service based on the same general ideas as Kerberos, with its single login and encrypted communications. There are several improvements in SESAME over Kerberos. It uses public-key cryptography, has role based access control, separated authentication and confidentiality keys, and many other improvements.

As shown in figure 2 below, the user logs in by connecting to a User Sponsor (US) client which then contacts the Authentication Server (AS) via the Authentication Privilege Attribute (APA) client. The US authenticates itself to the AS, and then the US contacts the Privilege Attribute Server (PAS) and receives a Privilege Attribute Certificate (PAC) which contains the user’s privileges and this PAC is used when making access control decisions. The user is now authenticated and has a PAC which can be used when starting application clients. Note that the user gets no ticket like in the Kerberos case, although it is possible to run in Kerberos mode with tickets.

If the user wants to start an application the user, actually the User Sponsor, must contact the Secure Association Context Manager (SACM) for the application client. This SACM then contacts the SACM for the application server and exchange user credentials. Next the SACM on the server side contacts the PAC Validation Facility to check the users PAC. After this the user can start the application client and exchange data with the server.

A more thorough and detailed description of SESAME can be found in [1]. An overview of the SESAME system can be seen in figure 2 below.

Figure 2: SESAME



4.2 Installation

There is a public release of Sesame version 4, and it is not simple to install it. The reason is that it was probably first developed for a Bull DPX/20 computer running AIX operating system, and therefore it has to be ported to other systems. Fortunately it has been ported to Linux and there are patches for the source code.

The installation and configuration procedure had to be followed exactly according to the project supervisors. This was true most of the time but some things didn't work as intended.

First we had to create accounts for "sesadm" (Sesame Administrator), "ca" (Sesame Certification Authority), and "sesuser" (Sesame User). Also a unix group called "sesame" had to be created. Next we had to download the libdes library and compile and install it. After this we started a build script that compiled SESAME, including some extra libraries.

During the installation there was several problem, even if it was said to compile without problems.

- First, there was a number of minor glitches in the installation instructions, for example that it worked better with typing the command "make gcc" than just "make" when installing libdes.

- In order to compile the sourcecode, ncurses-devel and XFree86-devel must be installed. These are development libraries for curses and the X-windows system.

- It uses "imake" to compile the source code. Unfortunately, "imake" is a Makefile generator made for the X-windows applications, and not really a generalized generator, and is thus unsuited for other application development. It would have been better to use GNU Autoconf instead.

- The whole package is compiled as userid "sesadm". It is also installed as that userid, instead of the normal way of installing as root. It is not really installed either because the binaries are still in the source tree and the users have to set up paths into that tree. It would have been better to install it into /usr/local/bin or some other common directory.

Once the above problems was solved, the apply of patches and compilation of the source code went fairly smoothly.

4.3 Configuration

The configuration of SESAME is a fairly tedious process, even if it is described in detail in the building guide. First you had to create a couple of user accounts, in addition to the “sesadm” account that was created earlier. Next you had to set up some environment variables that each user of SESAME must have. After this you had to edit and then run an overall configuration script that sets up certain things like the hostname of the computer and the name of the realm. The last and most lengthy step is to create all the public keys for the CA, AS, PAS, KDS, PVF and users including their roles and privileges.

The configuration process had the following problems:

- The process of configuration could have been simplified considerably if there had been a graphical user interface, where you could easily type in the names. Also there could have been a script which automatically created all the public keys.

- If lots of users and roles have to be configured, it would have been convenient with a configuration file where all new users are listed. This file then could have been run through a script and all the users and roles would have been created.

- There were no explanation of how the realm name and hostname had to be setup. The building guide was too much of a step by step instruction instead of a guide.

4.4 Running

In order to run SESAME three servers must be started, and this is done with a script called “rc.sesame”. The servers are kdc (AS, PAS and KDS), the pvf server and the audit server, which record all security related events. All three of these servers are started by the user sesadm.

The pvf must be run as root and so does “ses_login”, which is the program (User Sponsor) that logs in a user. By setting the setuid bit on these two programs and setting the owner to root they can be run with root privileges by any user that is a member of the sesame group. This means that any user can start the pvf.

When we tried to run “ses_login” it gave an error message. After some searching we found out that the homedirectory of root must be readable by everyone because SESAME searches that directory for keys.

4.5 Comments

There are several good things about SESAME. Most notably is the facts that it works well in a heterogenous system with different operating systems, it has rolebased access control, it uses public key cryptography when authenticating users and servers, and it is an open system.

But unfortunately there are some not so good things, in our opinion. SESAME is based on Kerberos and tries to be backward compatible, and all applications have to be sesamized. Also, it is dependent on Unix security, where instead this type of security should be an integral part of the operating system.

- SESAME uses a lot of acronyms, which doesn’t make it easier to understand. It does in some way tell us that SESAME is a quite complex system. In our opinion it is too complex for being a security system, and that Kerberos is as complex as such a system can be.
- Just like Kerberos, SESAME also requires synchronous communication and has even more states than Kerberos. This type of communication has some advantages, but also the disadvantage of having lower performance and less robust. This can be a serious disadvantage in a large heterogenous system. [2]
- There was documentation for SESAME, but it was a bit disorganized. For example we did not find any guide for setting up multiple realms.
- The servers should only be possible to start from root, and the servers should change their effective userid to sesadm. Also the applications should be installed in the normal unix directories. This is to prevent other users to do bad things, and also to integrate it more into unix.

- SESAME uses the client-server model. One result of this is that access to other filesystems must be done via an application. Perhaps this is not really a bad idea, and it could make a more coherent system.

5 SESAME/Kerberos Compatibility

So far we haven't figured out how compatible they are, but it looks like SESAME can handle Kerberos V5 key exchanges in the Key Distribution Server (KDS), so it can probably handle requests from Kerberos applications. Also, SESAME can be setup to make its authentications in Kerberos style.

It is also possible that they will be compatible because both uses the GSS-API interface. Though, neither SESAME or the free version of Kerberos (Heimdal) are really finished products. Therefore proper testing couldn't be done.

6 Discussion and conclusions

In older computer systems everyone logged into the mainframe and users naturally had one single login. When computer networks arrived users had to login at each computer to run an application. They also often sent passwords in clear text over the network. Kerberos and SESAME solves this current and real problem, and they do it well. It is not just a design and not future speculation.

The Kerberos / SESAME type of security system has its base in the client-server paradigm. New paradigms, like distributed mobile object and computer immune systems [4], are entering the stage and it is unknown whether Kerberos and SESAME have a place in the new cast. Also, there are many other instances where you do not need to use the client-server paradigm but instead just simply start a program, or where you do not need all this extra security, for example in web servers or anonymous ftp servers.

It is always important to note that Kerberos / SESAME is not the only solution to security. Even if SESAME can be very secure there can be other weak links in the chain.

The lessons we learned was that this is large systems and requires a substantial work to understand and operate.

References

[1] P Ashley and M Vandenwauver, *Practical Intranet Security*, Kluwer Academic Publishers, ISBN 0-7923-8354-0, 1999

[2] E. Belani, A Vahdat, T Anderson, M Dahlin, *The CRISIS Wide Area Security Architecture*, Proceedings of the 7th USENIX Security Symposium, 1998

[3] Bellowin and Merritt, *Limitations of the Kerberos Authentication System*, Proceedings of the USENIX Winter '91 Conference

[4] Bob Blakley, *The Emperor's Old Armour*, Proceedings of New Security Paradigms Workshop, 1996