# Global Information Assurance Certification Paper

**Roberto Sabbi**

# Running a Secure Kerberos Server on FreeBSD

## GCUX Practical Assignment version 2.0

Option 1 – Securing Unix Step by Step

**15 April 2004**

## Table of contents


**1 Abstract ..... 4**

**2 Introduction ..... 4**

**3 Specification ..... 5**

3.1 Server role ..... 5
3.2 Hardware requirements ..... 5
3.2.1 CPU ..... 5
3.2.2 Memory ..... 7
3.2.3 Storage ..... 7
3.2.4 Networking ..... 7
3.3 Operating system ..... 8
3.4 Software ..... 9
3.4.1 Kerberos ..... 9
3.4.2 Cracklib: a proactive Password Sanity Library ..... 11
3.5 Operating System and Application processes ..... 11
3.6 Network services ..... 11
3.7 User management ..... 12

**4 Risk analysis and mitigation plan ..... 12**

4.1 Threat analysis ..... 12
4.2 Data classification ..... 12
4.2.1 Confidentiality ..... 13
4.2.2 Integrity ..... 13
4.2.3 Availability ..... 13
4.3 Network location ..... 13
4.4 Risk analysis ..... 14
4.5 Mitigation plan ..... 14

**5 Installation and hardening ..... 15**

5.1 Physical installation ..... 15
5.2 BIOS setup ..... 16
5.3 Operating system installation ..... 16
5.3.1 Booting ..... 16
5.3.2 Disk configuration ..... 19
5.3.3 Distribution sets ..... 22

5.3.4 Final configuration ..... 24
5.3.5 Network services ..... 27
5.4 Software installation ..... 45
5.4.1 Package build ..... 45
5.4.2 Package Installation ..... 46
5.4.3 Configuration ..... 46
5.4.4 Startup ..... 48

**6 Maintenance ..... 57**
6.1 Operating system update ..... 57
6.2 Application update ..... 57
6.3 Backups and redundant storage ..... 57

**7 Auditing ..... 58**

**End of document ..... 58**

# 1 Abstract

This paper will discuss the use of the FreeBSD operating system to implement a Kerberos Key Distribution Center.

We will:

- perform an analysis of the hardware requirements and show that they are satisfied by the available machine
- introduce the operating system with some background on the version chosen
- introduce the software and the protocol it implements
- perform a risk analysis detailing the measures taken to mitigate the risks given the company requirements
- discuss in detail both the operating system and application installation
- accompany the installation procedures with the corresponding verification steps
- quickly describe the maintenance and audit procedures

# 2 Introduction

SabbiCom Servizi is a 20 people company working in the GIS sector that operates on a tight budget. The most common activity consists in creating and updating maps and drawings from a wide variety of customers, exchanged in electronic form. For compatibility reasons SabbiNet needs to work with a number of different platforms among which Solaris, HP-UX, Linux and FreeBSD.

Over time, SabbiCom acquired and developed software for many tasks. In total 40 hosts are in use although occasionally additional machines are rented to cope with the workload.

Indeed, in order to keep machines under maximum utilisation, SabbiCom also employs temp workers (mostly students) for the more clerical tasks such as data entry. These people work from Monday to Saturday in the following two shifts: 7h-14h and 14h-19h.

The workflow is managed by supervisors with a smart scheduling that also allocates machine slots according the software installed in order to limit the number of licenses. The supervisors are also tasked to manage the temp workers accounts. As the business expands, this is increasingly difficult since the supervisors' working hours do not necessarily overlap with the temps' schedule. Furthermore, accounts are created, locked and deleted quite often since the students adjust their presence on their academic planning.

Regarding the data, the files contain information that must be protected for privacy and national security reasons (some customers are government agencies). SabbiCom's management is therefore unhappy with the current compliance to the customer's security standards. The account management takes too much time from the supervisors and full compliance is not strictly guaranteed.

At first it is considered to hire a dedicated IT person but the idea is abandoned: the users are quite technical and do not generate enough support demand to justify the cost.

Upon advice from the consultant currently working for SabbiCom, the management decides to deploy a centralised authentication system and the choice falls on Kerberos. The consultant will also be in charge of maintaining the server, taking care of the security updates for both the operating system and the application.

This paper will describe the processes to setup, configure and maintain the Kerberos server.

# 3 Specification

## 3.1 Server role

The server will act as a single KDC with no slave replication, providing the authentication services needed to access the workstations and the services on the network.

## 3.2 Hardware requirements

The machine chosen is a single Pentium II 400MHz with 384 megabytes of memory and 20 gigabytes of hard disk. A cd-writer is installed for the backups. The network interface is a 100Mbit. The motherboard has an onboard dual channel ATA66 controller.

The machine is obsolete as a CAD workstation will therefore be devoted to the task of running the KDC. We will do a quick analysis to determine whether the hardware is sufficient.

### 3.2.1 CPU

The Pentium II CPU will handle low network traffic since it will only handle the authentication traffic and will access the network services needed (mostly DNS and NTP).

Most of the CPU load will be caused by the crypto calculation necessary to handle Kerberos tickets. Although there will probably be peaks localized at the beginning of the working shifts, the CPU described above is sufficient.

We can perform a “quick and dirty” verification by using the benchmarking capabilities of OpenSSL on similar PII machine:

> **openssl speed des -elapsed**

```
You have chosen to measure elapsed time instead of user CPU time.
To get the most accurate results, try to run this
program when this computer is idle.
Doing des cbc for 3s on 16 size blocks: 896402 des cbc's in 3.00s
Doing des cbc for 3s on 64 size blocks: 232714 des cbc's in 3.01s
Doing des cbc for 3s on 256 size blocks: 59623 des cbc's in 3.01s
Doing des cbc for 3s on 1024 size blocks: 14963 des cbc's in 3.01s
Doing des cbc for 3s on 8192 size blocks: 1866 des cbc's in 3.01s
Doing des ede3 for 3s on 16 size blocks: 246772 des ede3's in 3.01s
Doing des ede3 for 3s on 64 size blocks: 63019 des ede3's in 3.01s
Doing des ede3 for 3s on 256 size blocks: 15823 des ede3's in 3.01s
Doing des ede3 for 3s on 1024 size blocks: 3954 des ede3's in 3.01s
Doing des ede3 for 3s on 8192 size blocks: 496 des ede3's in 3.01s
OpenSSL 0.9.7c-p1 30 Sep 2003
built on: Tue Apr  6 23:45:21 CEST 2004
options:bn(64,32) md2(int) rc4(idx,int) des(ptr,risc1,16,long) aes(partial) idea(int) blowfish(idx)
compiler: cc
available timing options: USE_TOD HZ=128 [sysconf value]
timing function used: gettimeofday
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes
des cbc          4778.93k     4950.07k     5073.04k     5092.42k     5080.27k
des ede3         1312.37k     1340.49k     1346.23k     1345.46k     1349.50k
>
```

If we compare these results with the size of a sample credential file:

-rw------- 1 root wheel 461B Apr 5 23:39 krb5cc_0

We see that for the encryption that will be used (des3-cbc) the machine should suffice the target load.

### 3.2.2 Memory

The application executables running on the system typically have a small memory footprint and there will not be many of them given the narrow role of the server. The database used by Kerberos is a simple “dbm” file and most of the network services do not run as multi-process. The memory described above will be more than sufficient.

### 3.2.3 Storage

At any given time it is unlikely that the Kerberos database will hold more than 50 entries, therefore most of the storage space will be occupied by the logs. Neither the source files nor the “ports” tree will be installed leaving the base installed system at approximately 200 MB. Thus, the 20 GB will be more than sufficient for extended log retention.

A boot with a live bootable cdrom confirms the operating system support for the ATA controller, as can be seen from this excerpt from the dmesg output.

…

atapci0: <VIA 82C596 ATA66 controller> port 0xe000-0xe00f at device 7.1 on pci0

…

### 3.2.4 Networking

The Kerberos server will serve quite a light traffic, mostly “ntp” updates on a regular basis and Kerberos traffic with a peak at the beginning of shifts, when the staff logs in for the first time. The interface is thus adequate to handle the expected traffic.

We also repeated the same compatibility test for the network card with similar success:

…

dc0: <ADMtek AN985 10/100BaseTX> port 0xe800-0xe8ff mem 0xd7000000-0xd70003ff irq 11 at device 8.0 on pci0

dc0: Ethernet address: 00:50:bf:98:fb:1c

miibus0: <MII bus> on dc0

ukphy0: <Generic IEEE 802.3u media interface> on miibus0

ukphy0: 10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto

…

## 3.3 Operating system

The operating system chosen is FreeBSD for the following reasons:

- the consultant working for SabbiCom and most of the supervisors are familiar with it
- this ensures basic troubleshooting without the consultant intervention or by phone support
- no licenses costs will be sustained given the BSD license that covers FreeBSD
- the software used, Heimdal Kerberos and Cracklib, are available both as a port and as a binary package under FreeBSD, making installation and support easier
- FreeBSD has an excellent reputation for security, stability and network performance
- FreeBSD runs on the popular Intel platform
- In general, the hardware requirements for running FreeBSD are low

The candidates considered for this implementation are respectively version 5.2.1 and 4.8 of FreeBSD.

FreeBSD 5.2.1 would provide some useful security features not present in 4.8, like filesystem ACLs and Mandatory Access Control (an extension to the kernel access control capabilities).

Unfortunately at the time of deployment the version 5 of FreeBSD is not yet considered as production-ready. This means that the FreeBSD team still plans some fundamental changes before version 5.3 which is planned to be the Production Release.

This could complicate maintenance since system update via “make world” could break or require elaborate manual intervention. Additionally it contains much new code that needs to be exercised for security and stability[1].

For more information see: The FreeBSD Release Engineering Team. “Early Adopter's Guide to FreeBSD 5.2.1-RELEASE”. 25 Feb. 2004.

URL: http://www.freebsd.org/releases/5.2.1R/early-adopter.html (09 Apr. 2004).

The chosen version will be 4.8, in particular following the 4.8-RELEASE cvs tag, thus only the security updates will be added. Also, according to the FreeBSD

[1] The FreeBSD Release Engineering Team. “Early Adopter's Guide to FreeBSD 5.2.1-RELEASE”. 25 Feb. 2004. URL: http://www.freebsd.org/releases/5.2.1R/early-adopter.html (09 Apr. 2004).

website[2] it will be probably the 4.x release with the longest support (March 2005), meaning that a RELEASE upgrade (that constitutes heavy maintenance) will not be required for some time.

## 3.4 Software

### 3.4.1 Kerberos

#### 3.4.1.1 The Kerberos protocol

The Project Athena was established in May 1983 with support from a consortium of computer vendors to develop strategies and software aimed at integrating the use of computers in the MIT curriculum. MIT Kerberos was one of the various packages developed in the context of the project; its goal was providing authentication services to a distributed network consisting of servers and untrusted client machines.

The goal of the Kerberos protocol is to centralise authentication on a single server, the Key Distribution Center. The users (both human and applications) are called *principals* and their authentication domain is called *realm*.

The KDC stores each principal data including the pass-phrase in a database. This server, after authenticating the users or applications with their respective pass-phrase, provides them a TGT (for “Ticket Granting Ticket”).

This ticket is actually two copies of the same information: the identity (*principal*), the lifetime of the ticket and a session key. One copy is encrypted with the principal’s pass-phrase, the other with the KDC pass-phrase. The client verifies the identity of the server by verifying that he can decrypt the first copy and will use the other to prove his identity on all further ticket requests to the KDC.

Now the client is ready to connect to other servers within the same *realm* (that is the “jurisdiction” of the KDC):

- the client contacts the KDC sending a service ticket request. To prove his identity, he also sends the encrypted copy of the TGT. He also sends along an Authenticator: this is basically his name and IP address with a timestamp encrypted with the session key from the TGT.
- The KDC verifies he can decrypt the TGT with his key (which proves the client has been authenticated). He also decrypts the Authenticator to make sure the timestamp falls within the allowed time skew (usually 5 minutes).
- The KDC sends back a service ticket to the client, again encrypted in two copies: one with the client key, the other with the server key. The session key will be used to communicate between the client and the server.

[2] The FreeBSD Release Engineering Team. “FreeBSD Security Information”. 25 Feb. 2004. URL: http://www.freebsd.org/security/ (09 Apr. 2004).

- The client decrypts the first copy of the service ticket to verify the KDC identity and sends the other to the server along with an authenticator.
- If the client requests the server to prove his identity, the server increments the timestamp by one and sends it back to the client.

At this point, the client and the server are mutually authenticated and share a session key. This key can be used directly within an application or to exchange further sub-session keys.

While Kerberos provides the framework for all successive exchanges, they are entirely application dependent.

A few points to retain for the security analysis later:

- all the pass-phrases are stored in the KDC
- service tickets follow the same concept and structure of the TGT
- the KDC is the centre of all authentication exchanges as said above
- the authenticators ensure protection against replay attacks

#### 3.4.1.2 Heimdal Kerberos

Protocols are amazing creations, but they would be useless without software implementing them.

The current reference Kerberos implementations are the MIT Kerberos, from the original creators of the protocol and the Heimdal Kerberos distribution, a free Swedish implementation.

They are both complete implementation of the basic authentication protocol. However, they differ in the administration part regarding replication, password change and database manipulation since these topics were not covered in the initial specification[3].

They both support the same encryption algorithms such as des, triple des and the newest aes. However, MIT Kerberos is subject to export restrictions outside the U.S. since cryptography is considered as weaponry by the U.S. government.

Since SabbiCom is an Italian company, the implementation chosen for this project is Heimdal. The current version is 0.6.1, recently released to fix a vulnerability.

[3] The book used for this paper covers Kerberos in detail: Garman, Jason. Kerberos, The Definitive Guide. Sebastopol: O'Reilly & Associates, Inc, August 2003. Another interesting reading is: Steiner, Neuman,Schiller. Kerberos:An Authentication Service for Open Network System. 30 Mar. 1988. URL: http://www.pdc.kth.se/kth-krb/kerberos.ps (09 Apr. 2004)

### 3.4.2 Cracklib: a proactive Password Sanity Library

Cracklib is a library that can be easily used by applications to ensure adequate password strength[4].

The FreeBSD port of Heimdal can be compiled with cracklib support, which results in the “kpasswdd” daemon to use it for password quality check.

In our case this is very useful since the users themselves are responsible for periodically changing their password.

## 3.5 Operating System and Application processes

The server will run the following operating system processes:

- The init process
- The adjkerntz utility for local time correlation
- the syslogd daemon for logging
- the cron daemon for task scheduling
- the sshd daemon for remote login
- sendmail for local mail delivery
- the inetd daemon for running the kadmind application daemon

And the following application processes:

- the kdc daemon that is the authentication server
- the kpasswd daemon used for providing the password change service to users
- the kadmind daemon for the Kerberos database administration

## 3.6 Network services

The server will be listening on the following ports:

- 22/tcp for ssh connections
- 88/tcp and udp for Kerberos authentication request
- 464/udp for password change requests
- 749/tcp for administrative connections
- 25/tcp *only locally* for mail delivery

[4] For more information: Muffet, Alec. “CrackLib: A ProActive Password Sanity Library.” 14 Oct. 2003.URL: http://www.crypticide.org/users/alecm/security/cracklib,2.7.txt (09 Apr. 2004)

### 3.7 User management

The only users allowed interactive access through “ssh” will be the supervisors. They will then be able to check the logs for problems and the Kerberos logs. They will not be included in the “wheel” group, to prevent them from “su-ing” to root.

The supervisors will also be allowed to connect with the “kadmin” utility to perform account management.

The other users will only be able to change their passwords connecting via the “kpasswd” utility.

## 4 Risk analysis and mitigation plan

### 4.1 Threat analysis

The outer security layer of any host is represented by the network services. The highest threat is an attack to the “sshd” server since password enumerating activity could provide a shell to an attacker that would then try to escalate to the root account. This potential threat is mitigated by the fact that once the user has successfully logged in, the “sshd” daemon runs with the user privileges. We must also keep into account that the “sshd” daemon provides file transfer capabilities that are useful to an attacker.

An alternative attack would be trying to directly exploit the service to run code as root or obtain a shell. This requires the service to be vulnerable.

The Kerberos services do not provide any direct shell access, therefore they can only be attacked if remotely vulnerable. An exception could be “kadmind” since it is a client/server application that accepts user input.

All the network services can also be simply attacked with a denial of service.

Moving closer to the inside we note that the supervisors will have shell access via “ssh”. The mitigating factor is that they do not have root access nor they can “su” by default, even if having root password.

Then we can list the physical attack, starting from account enumeration at the console, to the possibility of inserting medias and/or trying to reboot the machine and enter into sinlge user mode.

The denial of service attack can also be physical.

### 4.2 Data classification

The KDC itself does not contain any customer files nor internally developed software (both of which are the most valuable asset for the company). However, it contains the credentials allowing access to the hosts on the network.

Thus, the data are by the KDC database.

### 4.2.1 Confidentiality

A compromise of the credentials confidentiality would give access to the data and software.

As said above, the company has a contractual obligation to privacy given the fact that most files being worked contain demographic information.

If some of the information contained was revealed this would cause great embarrassment for the company. For a small company like SabbiCom, this is more sensitive than financial information.

If, for example, one file was posted on the Internet, the impact could be the loss of a contract, immediately threatening the company survival.

### 4.2.2 Integrity

The introduction of forged credentials, basically recreates the scenario discussed above.

### 4.2.3 Availability

The availability of the data is not a critical factor. SabbiCom is currently operating without a central authentication server and would for sure be able to continue operating in case of unavailability. Considering that procedures exist to operate without central authentication, the unavailability of the server would cause loss of productivity but would not be a blocking factor.

## 4.3 Network location

Except for some servers containing data and therefore well hardened, most of the hosts are workstations. These are allowed to talk to the KDC server since they need to use its authentication services.

These workstations are not allowed to talk directly to the Internet. They fetch software updates from a server.

The only server connecting outside the firewall protecting the network is the one that co-hosts “dns” and time server. For its role, it connects to the upstream services provided by the ISP.

This host is used by the KDC for “dns” and time services.

Other hosts are the executive’s laptops that fetch mail from the ISP.

## 4.4 Risk analysis

Let us first consider an attack coming from the outside. This would have necessarily to come through the DNS/Time server.

Unfortunately they are the only instances available and must be relied upon. Also, this scenario would not pose more risk to the KDC than to the servers actually containing the data.

Second we have the managers' laptops: since they cannot be used to surf the Internet, the only possible attack would be an e-mail worm. The ISP is therefore relied upon to make sure that e-mails are properly scanned for viruses in addition to the antivirus installed locally on the laptops. This would also be an off-line attack and therefore is not of great concern.

The most probable scenario is thus the internal attack. Occasionally some customers come on site to preview the work being done and connect to the network.

The rest of the users are the supervisors and the temp workers using the workstations. These workstations are also used for development therefore, although not connected to the Internet, there is the potential for installing malicious tools and attacking the server.

Given the role of the server, though, it is not possible to prevent the supervisors from accessing it. In fact, all the traffic happens on the same network and almost all the hosts need to access the KDC.

The features of the Kerberos protocol should prevent theft of in-transit credentials.

The supervisors will also be accessing the KDC as administrators and their credentials could be stolen and used to create or modify accounts. Therefore we must make sure that their admin accounts as well as any other are well protected.

The supervisors will also be interactively accessing the KDC to do log monitoring, thus even the interactive accounts will have to be protected. With one of their accounts, an attacker could try to wander around in the system, transfer data to or from it, or filling the disk.

The host will not have other interactive accounts beside those and the consultant one. This and the root password will be changed after every maintenance activity.

The remaining risks will have to be mitigated with an adequate logging policy.

Finally, the physical exposure of the host cannot be tolerated and measures will have to be taken.

## 4.5 Mitigation plan

In order to mitigate the remaining risks, the following measures will be taken:

- the host will be physically secured
- the BIOS will be adequately configured
- the unused drivers will be disabled
- the partitions will be well sized and spread to prevent denial of service attacks and ensure long term log retention
- the operating system components and the server software installed will be the minimum required to run the server as a KDC
- unused network services will be turned off
- necessary network services will be hardened
- time synchronization will be used to prevent replay attacks
- unused local services will be turned off
- the console will be hardened against password-guessing activity or session hijacking
- single user mode will be password protected
- remote logins will also be hardened
- the default password policy will be tighetened
- the filesystem mount parameters will be hardened
- this will be complemented by a “securelevel” setup, making this immutable without reboot and preventing drastic time changes
- the software installed will also enforce password quality and provide access on a need-to-have basis

# 5 Installation and hardening

## 5.1 Physical installation

SabbiCom does have a dedicated server room but other servers and telecom equipment are located in it.

This is taken as an opportunity to install a wire cage on which this system and the ones containing the data will be placed. The room has no windows which prevents shoulder surfing from the outside.

The chassis has a door with lock protecting access to media drives.

During the installation, the host will only be connected to a small hub, together with a laptop used to test the system.

## 5.2 BIOS setup

The BIOS, after the first reboot of the installation will be configured with hard disk as the first booting media, followed by cdrom and floppy. Unfortunately, this BIOS does not offer the possibility of directly excluding devices. On the other hand, the BIOS is rather old and does not allow booting from the network or usb ports.

After this modification, a setup (not boot) password is entered and also put in an envelope that will be put in a locked tray.

The boot password will not be set since this would prevent a quick recovery in case of system crash.

The parallel port and the floppy drive are also disabled.

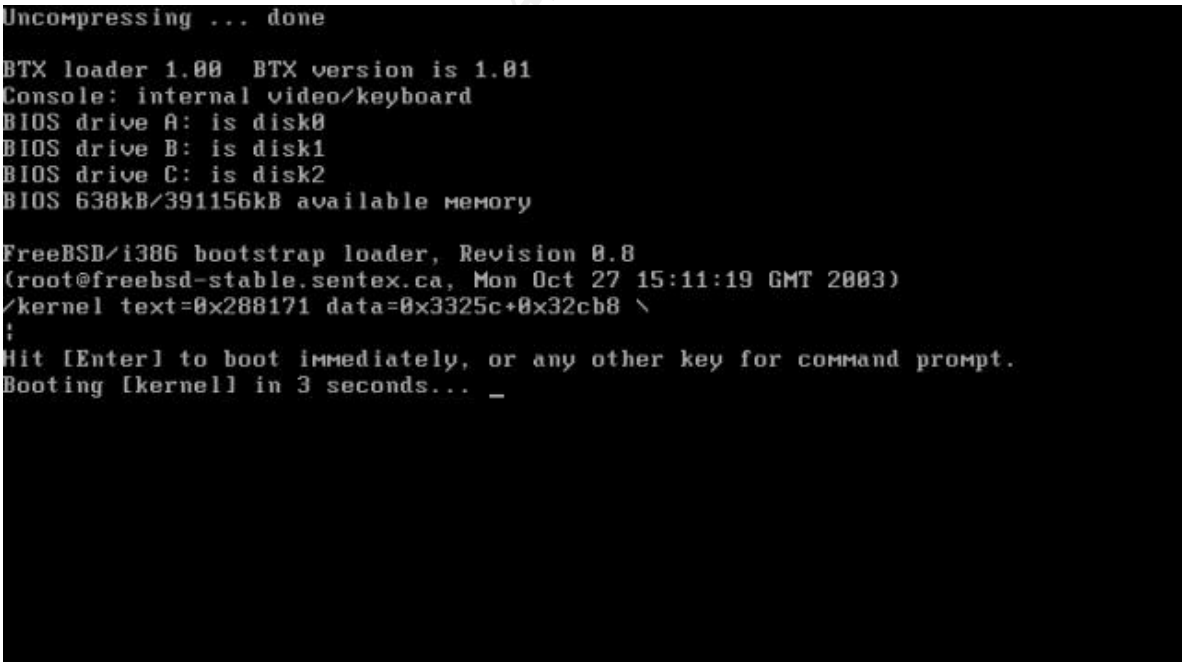
The setup password is kept in the company safe.

These measures, together with the physical ones, prevent an attack based on booting on a removable media or dumping the credentials.

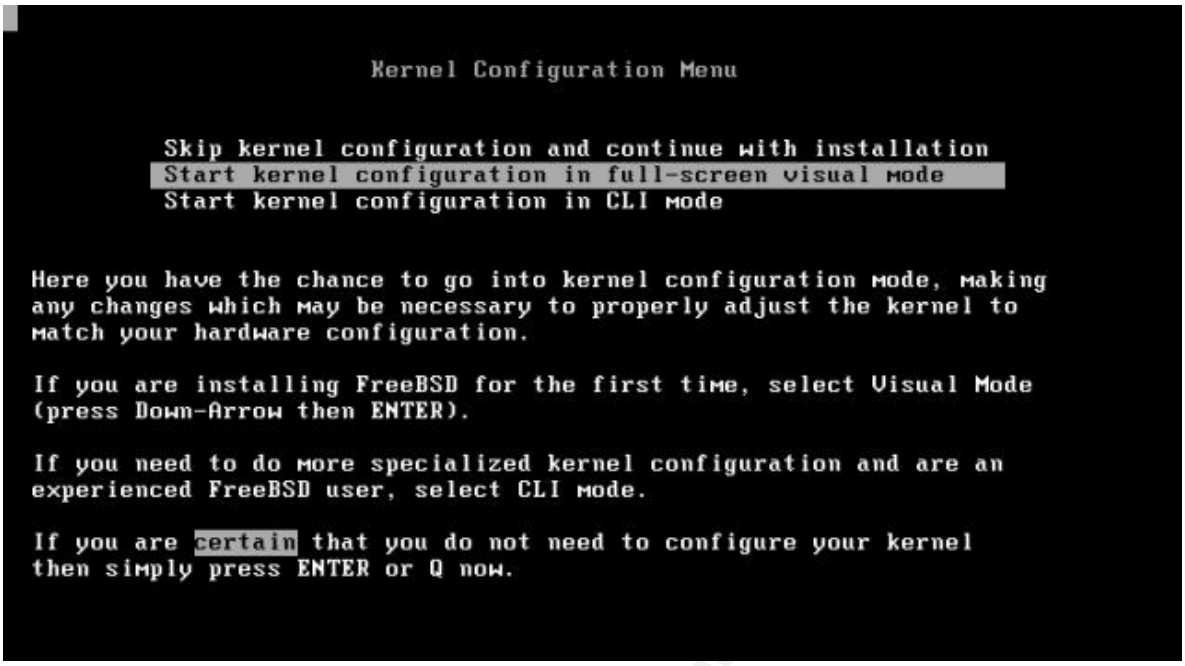
## 5.3 Operating system installation

### 5.3.1 Booting

It is now time to insert the FreeBSD cd and boot the system; after some hard disk activity, some console messages and some beeping,

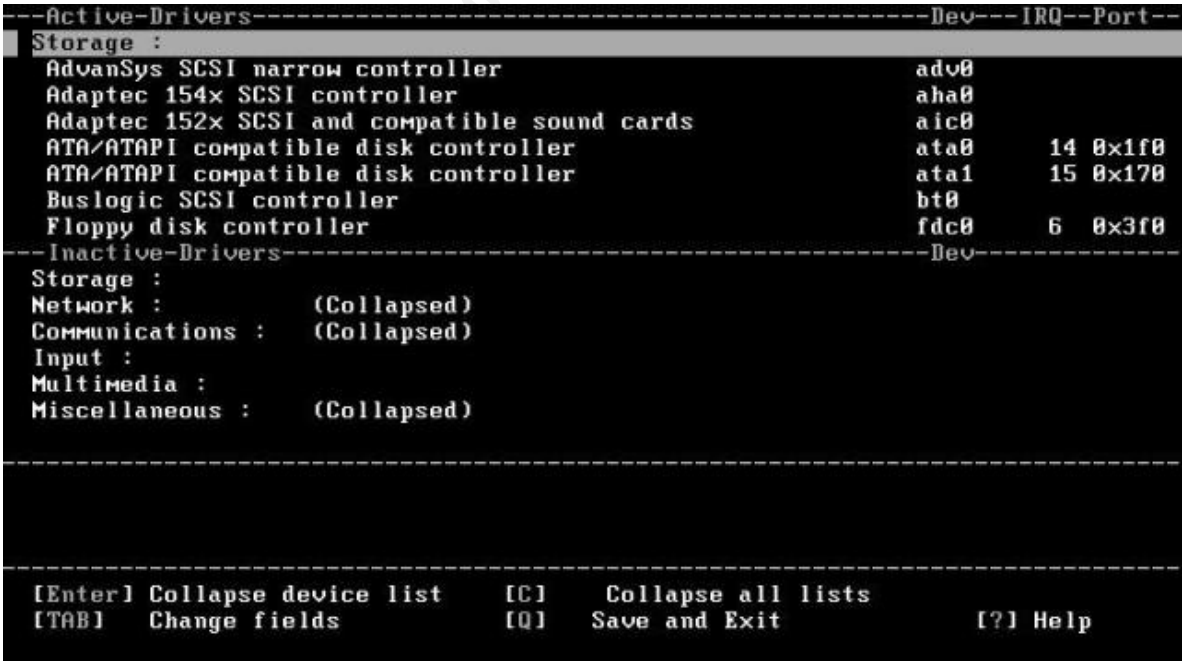



we are presented with three options to do kernel configuration:




We will move on the second option: "**Start kernel configuration in full-screen visual mode**", and press "Enter".

On the next screen we will press "X" which will present us with the list of the detected hardware grouped by driver category.



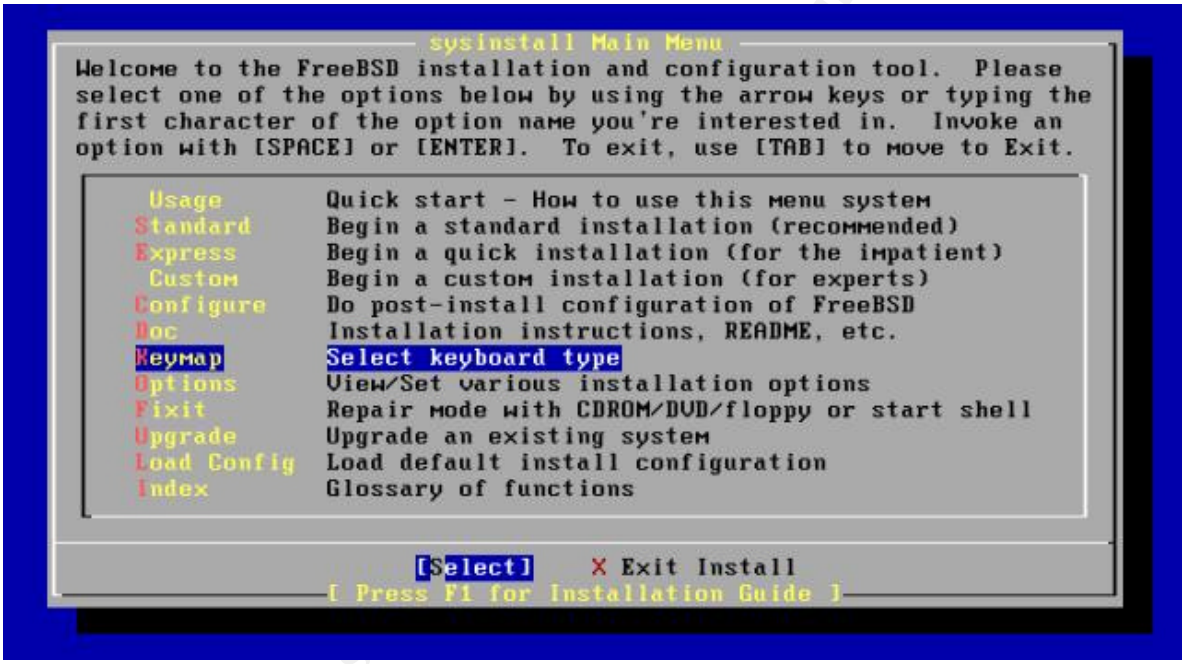
The kernel used for installation is compiled with support for several devices and this menu is used to enable/disable in-kernel drivers that may conflict.

It is useful to perform this step initially because the configuration will be saved and we will not have to do it later. We also minimise the chances of an aborted install.

Moving with the arrow keys and removing devices with the delete key we will delete the scsi devices, the network controllers and the pc-card controller.

We will not delete the ATA/ATAPI entries or our machine will not boot further.

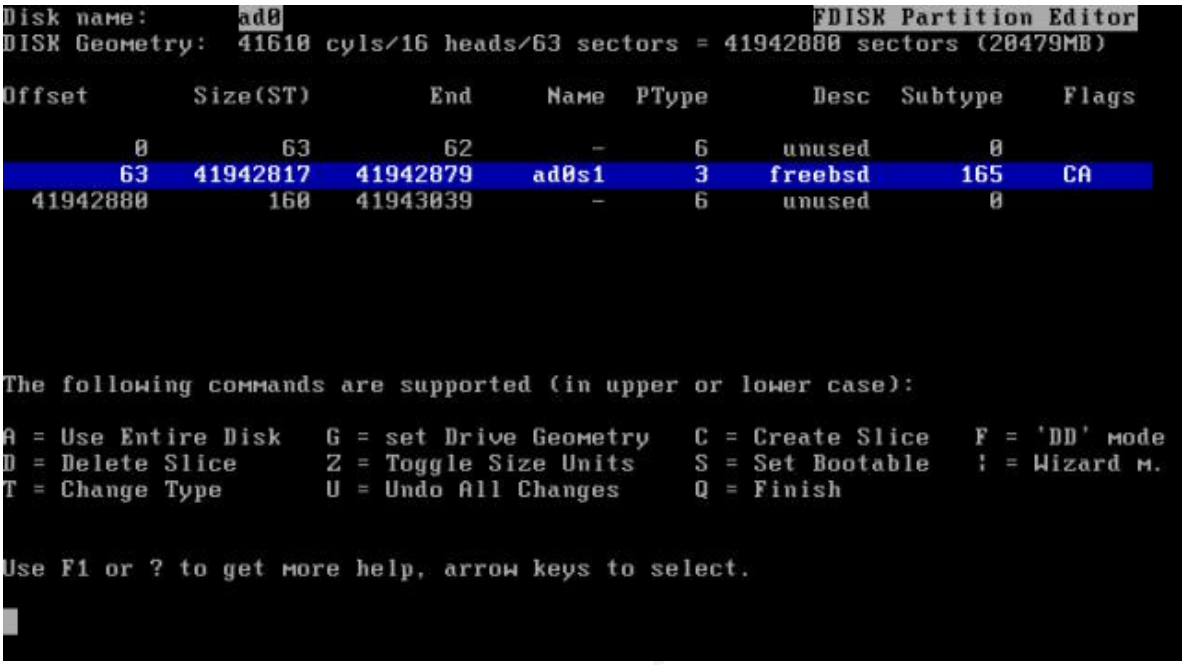
When done, we will press “Q” to exit and “Y” to save the configuration and the boot sequence will continue until the main installation menu.




Note that even if we connected a non-US keyboard, the OS still uses a US layout thus, on the next screen we will move over the “Keymap” option and select the appropriate layout.

We will then exit and move over the “Express” installation option and hit enter and will be presented with the FDISK partition editor.

### 5.3.2 Disk configuration

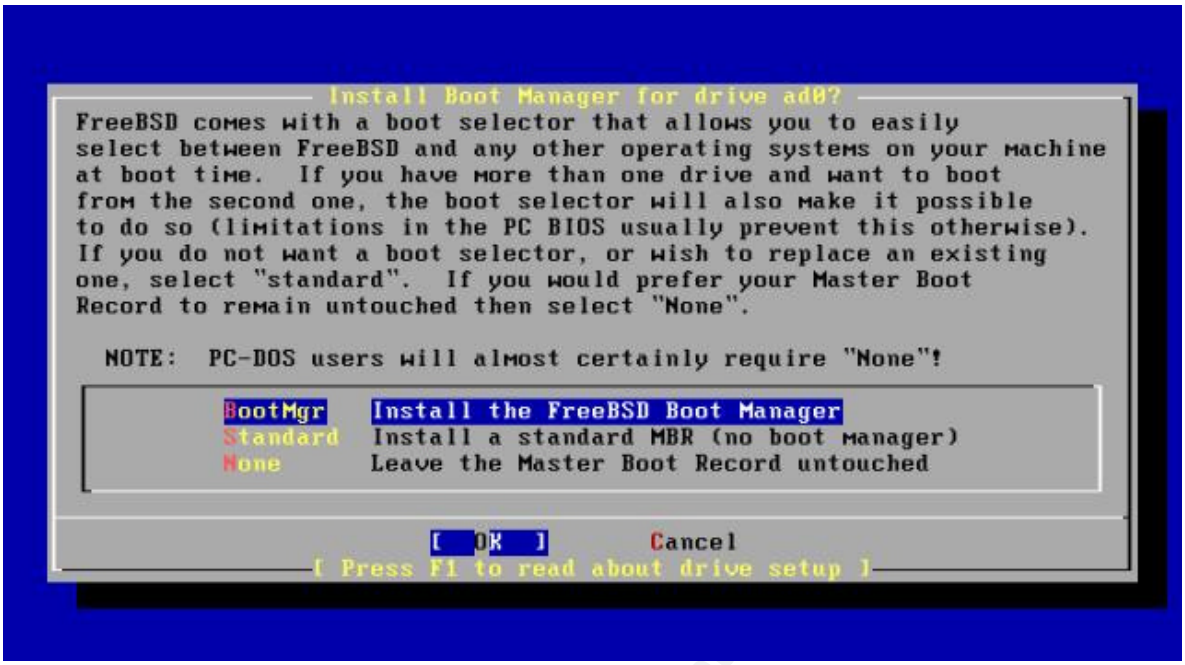



In this screen we will create the FreeBSD partition: using the “A” key, we will allocate the entire disk to the FreeBSD partition.

We move now over the newly allocated partition, hit “S” to make it bootable and exit with “Q”.

Note that in the BSD operating systems family, all the installation is contained in a partition further divided into slices. The slices will be initialised and mounted under the root tree.

The next menu will prompt us for the boot manager installation:




We hit “Enter” to confirm installation of the boot manager.

Having the boot manager installed has an indirect security implication. We may need to recompile the kernel for example to exclude a vulnerable subsystem for which no patch is still out. If something went wrong and the newly compiled kernel does not boot we still have the option of booting the old one. Without a boot loader, we would have to perform a complex recovery procedure, prolonging the unavailability of the system.

We are now on the disklabel editor menu, where we will divide the disk in slices. We now have to decide the number and size of our filesystems.

Keeping into account our mitigation plan:

- we will allocate 1G of space to the root partition. Typically FreeBSD will not use more than 50~70 Mb of space even with multiple kernels and modules installed (upon install of a new kernel, the old one and the modules directory are renamed to “.old”. We want to make sure that it is not easy to fill the root partition, be it because of malicious intent or system manipulation error.

- to calculate the swap file size for a server a good rule of thumb is to make it the double of the physical memory. Actually a network server, especially when providing interactive services, should never have to swap. This would make response time unbearable for the user. A Kerberos server does not provide interactive services but would cause the same effect for a user requesting a service ticket to access one.

- the home directory will not be used much since the server will only be accessed for monitoring or maintenance purposes. Again, 1Gb will provide a basic assurance against denial of service. Furthermore, having it as a separate partition, prevents a

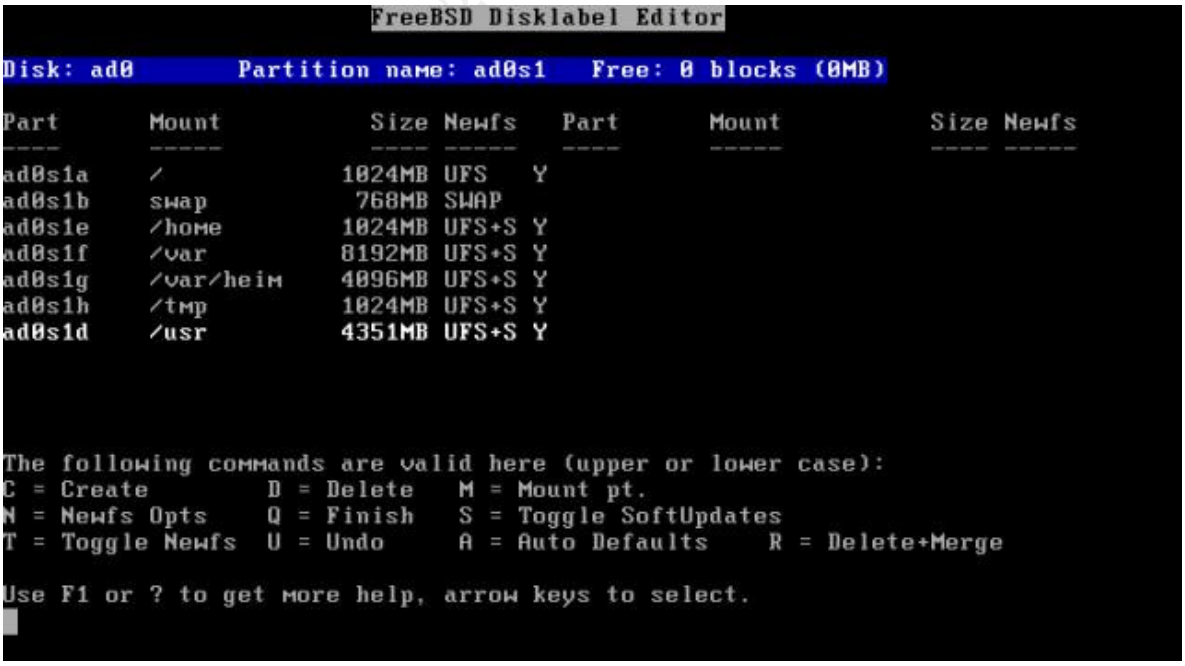
denial of service opportunity: by default FreeBSD creates the “/home” tree as a symbolic link of the “/usr/home” partition. A user could then fill up the “/usr” partition and at the next “make installworld” run, the system would be only partially update. This could leave the machine in an inconsistent or unbootable state.

- since the system will undergo long periods without qualified maintenance from the consultant and for eventual historical research, it is decided to keep the logs for as long as possible. Therefore 8Gb will be allocated to the “/var” filesystem that also contains mail files that may accumulate over rtime.

- we will dedicate Heimdal a separate filesystem for the data directory, “/var/heimdal”, that will contain only the principals database, the master key and the kadmin acl file. The log will go into “/var/log”. For this reason 4Gb is more than enough. We stay on the very safe side, because filling up this partition would likely cause the KDC to crash.

- FreeBSD uses “/bin”,“/sbin” and “/usr” (and some other) trees for the base system and “/usr/local” for the third party packages. This, among other advantages, prevents any dangerous overlapping between a third party package and the base system. Our case is a good example: the FreeBSD base system already comes with a basic Heimdal Kerberos installation. Even if most executables have different names, we will be sure that the daemons under “/usr/local/libexec” belong to our Heimdal package. The same goes for “/usr/local/bin” and “/usr/local/sbin”. The remaining space will be then allocated to the “/usr” filesystem. This tree will remain stable in size since no application will be installed other than those initially planned.

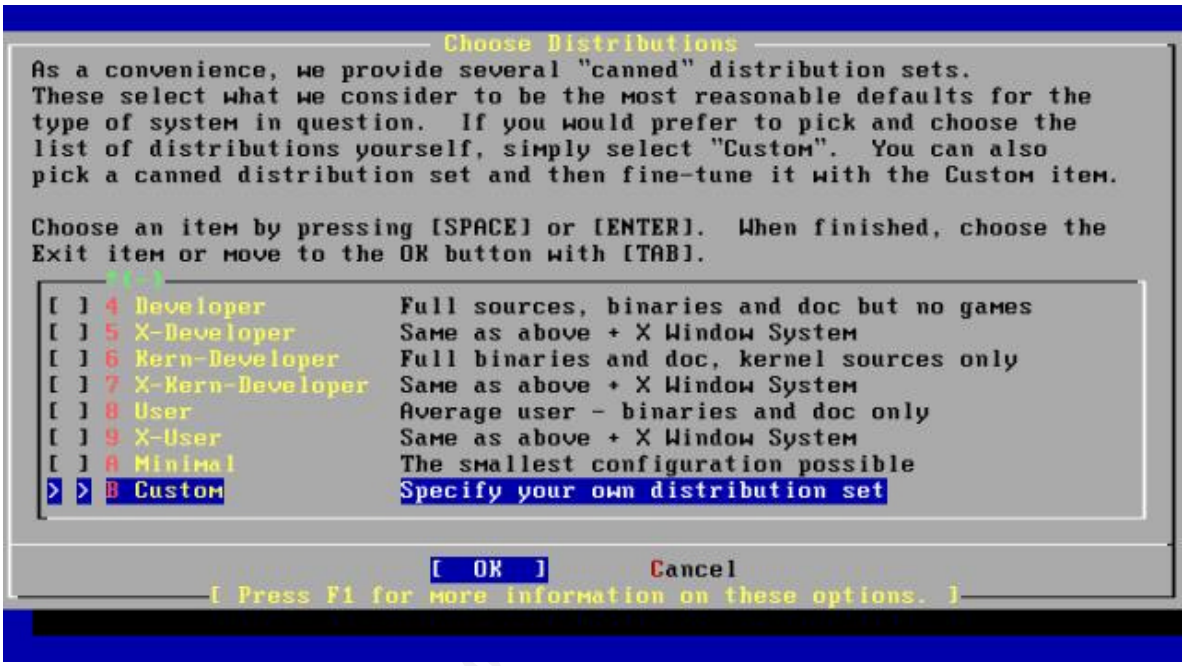
Before committing our layout we can have one last look:



After having allocated all the slices and setup the mount points, we exit hitting ‘Q’.

### 5.3.3 Distribution sets

In the next step we can apply the “minimal installation” policy discussed earlier. The installation presents us the choice of predefined sets of the base installation.




We will chose the “Custom” option that will allow us to specify exactly which components we want to install.

Of the options in the next menu we will choose the following:

**Bin** – the binary base distribution, it is mandatory.

**Crypto** – encryption services that we will need. OpenSSH is an example of a component that will use those services.

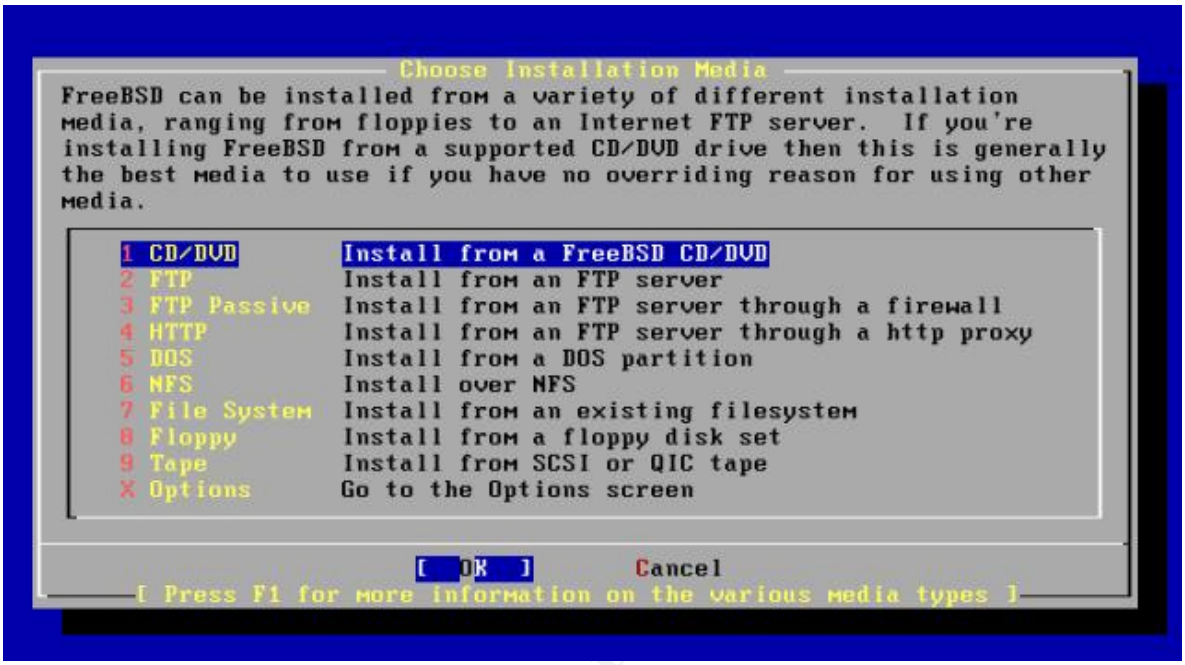
**Dict** – contains spell checker dictionaries we will need for password quality checking.

**Info** – the Info format documentation system, we will need it since part of the Heimdal documentation comes in this format.

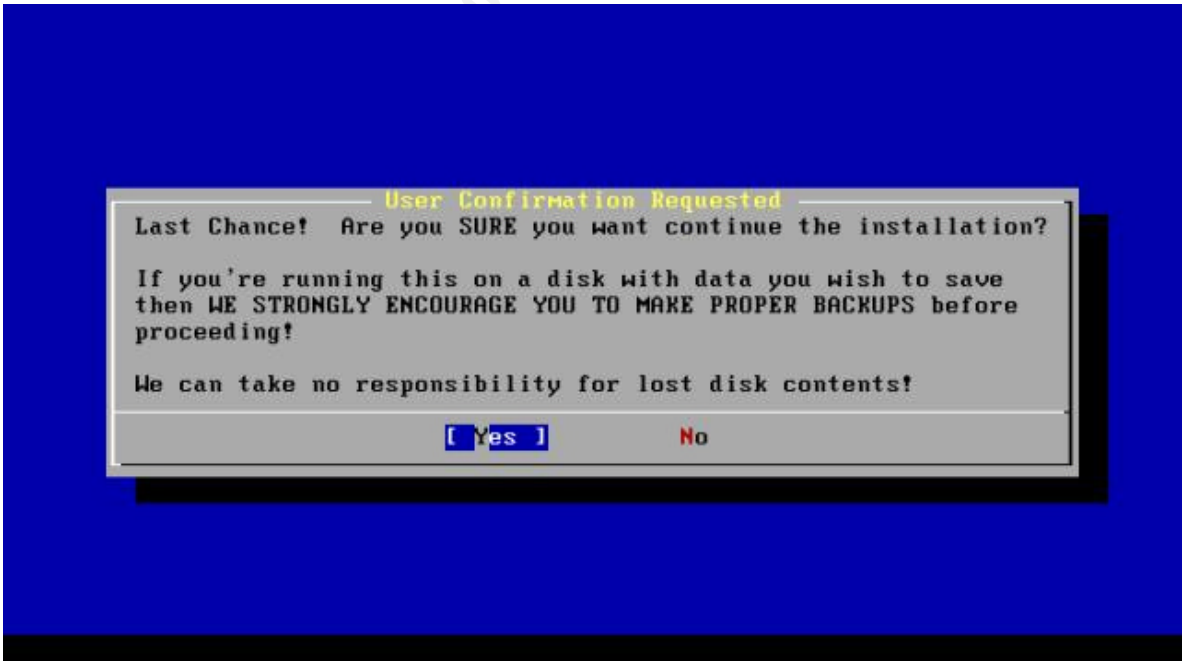
**Man** – the manual pages, needed for both the base system and the software.

Even though info and man are not strictly required they add only a few binaries between executables and libraries and are contained in dedicated directories

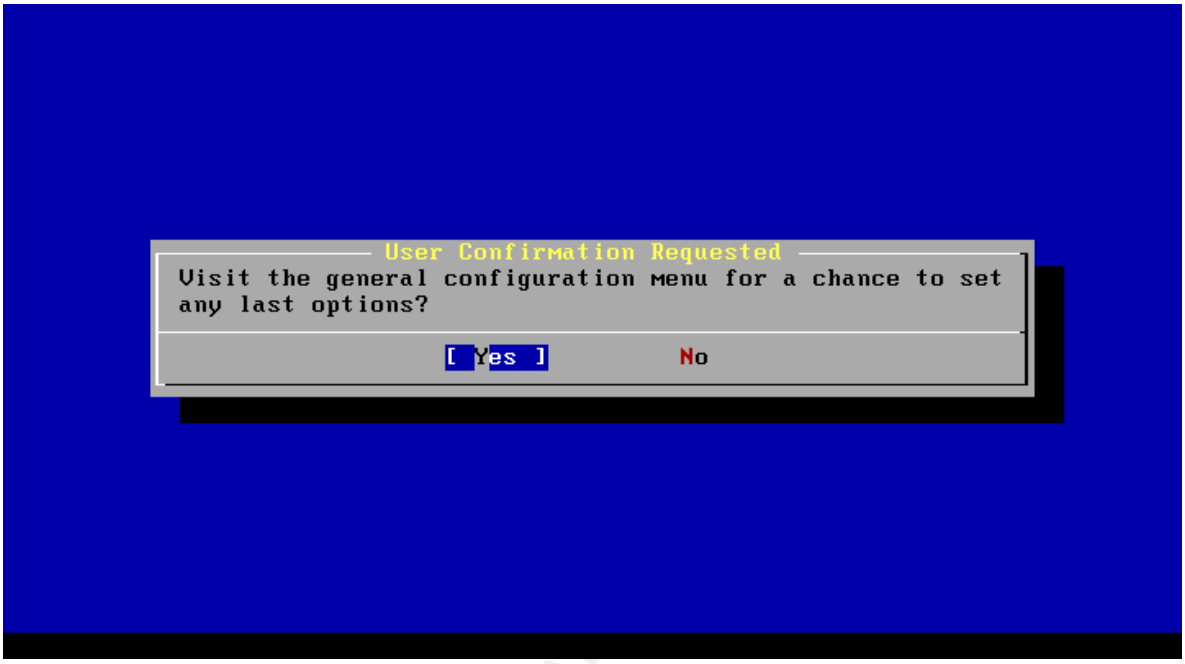
After completing the selection, we will exit selecting the “Exit” option twice and select the “CD/DVD” media option:




At the next screen we are given our last chance to back out before any changes are made,



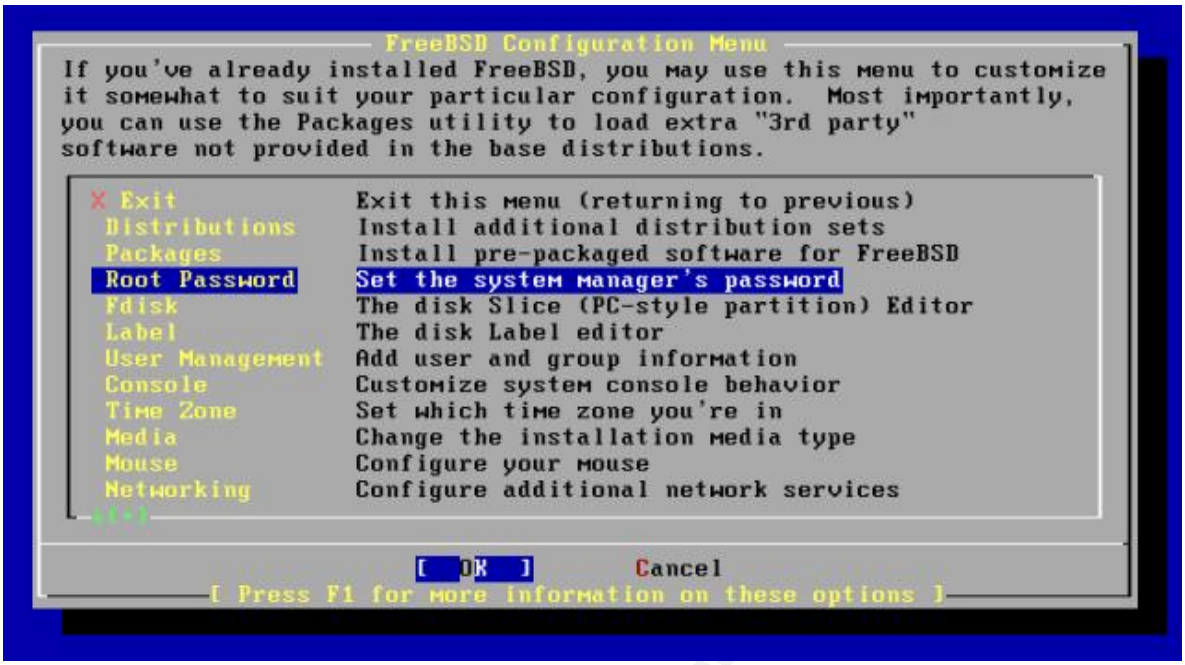
We hit the “Y” key and the installation starts first initialising the disk layout previously configured then the distribution sets are installed and the devices created.




The installation asks now if we want to do more configuration. We will choose “Yes” in order to configure all we can before rebooting.

### 5.3.4 Final configuration

Our last choice brings us back to the Main installation screen where we will set the root password.

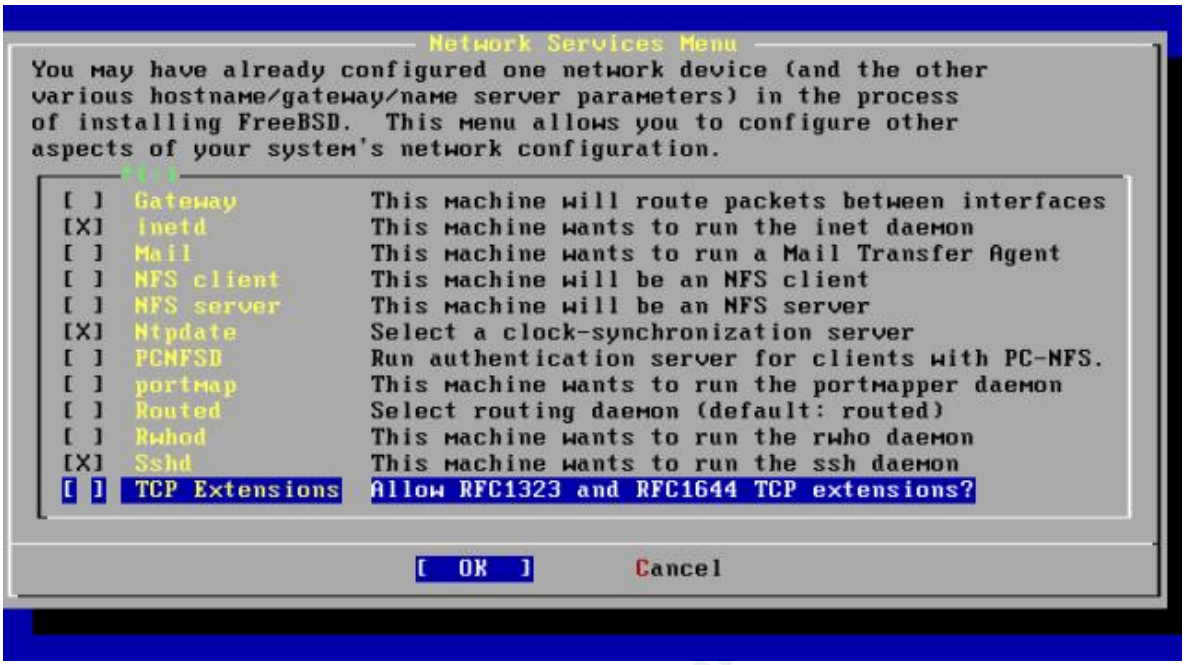



Next, we will choose “Networking” and will enable “ntpdate”. From the list of time servers, we will choose “Other” and we will enter “time.sabbi.it”, the hostname of the internal timeserver.

Now, every time the system will boot, the system clock will be synchronised. After having configured the “securelevel” later on, no clock adjustment of more than one second will be allowed. This ntpdate run, though, is done before the system enters the “securelevel” (which is irreversible). In other words, if the system clock goes beyond the limit where successive adjustments are not able to bring it back on time, a simple reboot will fix it, waiting for more troubleshooting.

This can be done by simply accessing the console of the system and hitting “ctrl+alt+del”. In this option there is of course the potential for misuse, but being able to reboot cleanly the system without using the root password can be an advantage. If for some reason the system becomes unstable, a supervisor will be able to gain physical access under control from the management (who has sole access to the safe holding the key of the cage and the IT room) and perform the operation.

In conclusion: the root password will not be exposed and physical access control is ensured.




Finally we will disable the last item in the list, named “TCP Extensions”.

The RFCs 1323 and 1644 define extensions to the TCP protocol in order to improve network performance by looking into reducing TCP overhead and adapting to changing traffic conditions. According to “The FreeBSD Handbook:”:

> This enables the TCP Extensions defined in RFC 1323 and RFC 1644. While on many hosts this can speed up connections, it can also cause some connections to be dropped. It is not recommended for servers, but may be beneficial for stand alone machines[5].

Furthermore this has some benefit only if both ends implement the extensions.

We are now ready to reboot. We exit from the configuration menu, then we choose “Exit install” and after a confirmation prompt, the machine reboots.

We will follow carefully the boot messages to make sure no errors are reported and we will login with root. So far it is the only account present.

[5] The FreeBSD Documentation Project. “The FreeBSD Handbook” System Documentation. URL: http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/book.html#INSTALL-POST (01 Apr. 2004)

### 5.3.5 Network services

It is time now to have a look the network services running on the host:

```
# netstat –na
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  *.587                  *.*                    LISTEN
tcp4       0      0  *.25                   *.*                    LISTEN
tcp4       0      0  *.22                   *.*                    LISTEN
tcp46      0      0  *.22                   *.*                    LISTEN
udp4       0      0  *.514                  *.*
udp6       0      0  *.514                  *.*
Active UNIX domain sockets
Address  Type   Recv-Q Send-Q    Inode     Conn     Refs  Nextref Addr
d659cc80 dgram       0      0        0 d6569f00        0 d659cf00
d659cf00 dgram       0      0        0 d6569f00        0 d659cd20
d659cd20 dgram       0      0        0 d6569f00        0 d659ce60
d659ce60 dgram       0      0        0 d6569f00        0        0
d6569f00 dgram       0      0 d6563b80        0 d659cc80        0 /var/run/log
#
```

The first thing to note is that sendmail is running and listens both on port 25 and 587. As said earlier, we only want local mail to be delivered for system reporting purposes.

Another thing to note is that “sshd” and “syslogd” are also listening for both IPv4 and IPv6 addresses (we will address that later on).

#### 5.3.5.1 Sendmail

Let’s start from sendmail configuration:

In FreeBSD the “/etc/defaults/rc.conf” file, controls among other things, what services are started at boot. Let’s have a look at the sendmail section:

```
##############################################################

###  Mail Transfer Agent (MTA) options  ######################

##############################################################


mta_start_script="/etc/rc.sendmail"

sendmail_enable="YES"

sendmail_flags="-L sm-mta -bd -q30m"
```

```
sendmail_submit_enable="YES"
sendmail_submit_flags="-L sm-mta -bd -q30m
-ODaemonPortOptions=Addr=localhost"

sendmail_outbound_enable="YES"
sendmail_outbound_flags="-L sm-queue -q30m"

sendmail_msp_queue_enable="YES"
sendmail_msp_queue_flags="-L sm-msp-queue -Ac -q30m"
```

The “sendmail_..._enable” variables control the start of the different sendmail instances, in the following order:

- the one listening for incoming e-mail from the outside
- the one listening for incoming mail from the localhost
- the one flushing the outbound queue
- the one flushing the submission queue

We do not want the first one or we would have port 25 listening on the network.

We need a delivery agent so we need to enable the second, and it listens only on localhost.

We do not want mail to go outside so the third is to be disabled.

We would need the 4th one but there is already a script in the “/etc/periodic/daily/500.queuerun” doing the same work once per day. Since we do not need immediate delivery, we can run one daemon less.

Before changing the options, we need to stop sendmail. In FreeBSD there is a practical makefile that is used to generate the “.cf” files and to stop/start the daemons. Since the makefile uses the rc.conf variables, we need to issue the “make stop” before changing them.

We execute:

```
# cd /etc/mail
# make stop
```

The system confirms the daemons have stopped.

Now we can turn off those options by putting the following in “/etc/rc.conf”:

```
sendmail_enable="NO"
sendmail_submit_enable="YES"
sendmail_outbound_enable="NO"
sendmail_msp_queue_enable="NO"
```

We will try as much as possible not to touch “/etc/default/rc.conf” because when updating the system, this file is updated from time to time. Putting the variables in “/etc/rc.conf” ensures the integrity of the configuration.

We can now restart the mail system:

```
# make start
Starting: sendmail-submit.
```

We verify again the listening ports:

```
# netstat –na
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  127.0.0.1.25           *.*                    LISTEN
tcp4       0      0  *.22                   *.*                    LISTEN
tcp46      0      0  *.22                   *.*                    LISTEN
udp4       0      0  *.514                  *.*
udp6       0      0  *.514                  *.*
Active UNIX domain sockets
Address  Type   Recv-Q Send-Q    Inode     Conn     Refs  Nextref Addr
d659cf00 dgram      0      0        0 d6569f00        0 d659cd20
d659cd20 dgram      0      0        0 d6569f00        0 d659ce60
d659ce60 dgram      0      0        0 d6569f00        0        0
d6569f00 dgram      0      0 d6563b80        0 d659cf00        0 /var/run/log
```

We verify that only one instance of sendmail is running:

```
# ps -ax -o pid,user,command

  PID USER COMMAND
    0 root  (swapper)
    1 root /sbin/init --
    2 root  (taskqueue)
    3 root  (pagedaemon)
```

```
  4 root  (vmdaemon)
  5 root  (bufdaemon)
  6 root  (syncer)
  7 root  (vnlru)
 70 root /usr/sbin/syslogd -s
 78 root /usr/sbin/inetd -wW
 80 root /usr/sbin/cron
 82 root /usr/sbin/sshd
 85 root sendmail: accepting connections (sendmail)
101 root login [pam] (login)
112 root -csh (csh)
178 root ps -ax -o pid,user,command
102 root /usr/libexec/getty Pc ttyv1
103 root /usr/libexec/getty Pc ttyv2
104 root /usr/libexec/getty Pc ttyv3
105 root /usr/libexec/getty Pc ttyv4
106 root /usr/libexec/getty Pc ttyv5
107 root /usr/libexec/getty Pc ttyv6
108 root /usr/libexec/getty Pc ttyv7
#
```

#### 5.3.5.2 Syslogd

Now, beside sshd we still have port 514 used by syslogd. FreeBSD syslogd has an interesting feature: by supplying the flag “-ss” it is possible to disable all network capabilities. Thus syslog does not even connect to remote hosts.

We just have to add the line below to “/etc/rc.conf”:

```
syslogd_flags=”-ss”
```

#### 5.3.5.3 OpenSSH

With the setup done so far and after configuring the networking we will have only SSH listening after the next reboot. So, before we proceed to configure sshd, let’s review the default FreeBSD configuration. We will only discuss the parameters that need to be changed and those with particular relevance for security:

```
# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options change a
# default value.
```

This configuration file is useful because we can compare inline the default settings with our target configuration.

#Protocol 2,1

We have no use for the version 1 of the protocol therefore we will modify this option to prevent downgrading to version 1. It will be:

# Changed on 04/04/2004
Protocol 2

We added the comment date because on system update this file could be subject to merging and having the comment will help deciding what to do.

#ListenAddress 0.0.0.0

The “Listen” statement is useful to decide to which address and port sshd will bind. We do not really want to change that (it could be useful if we had multiple interfaces) but un-commenting that line has the positive effect of preventing sshd from listening for IPv6 addresses.
We will then change it to:

# Changed on 04/04/2004
ListenAddress 0.0.0.0

We will only review the next parameter:

# Logging
#obsoletes QuietMode and FascistLogging
#SyslogFacility AUTH
#LogLevel INFO

The level of logging and the facility is appropriate. We will instead change the next one:

# Authentication:
#LoginGraceTime 120

To:

# Changed on 04/04/2004
LoginGraceTime 60

with the effect of aborting the login if not completed within 1 minute. This makes the network login a little bit more resistant. In case of malicious activity the attacker has less time to fiddle with the initiated login. Next parameter:

#PermitRootLogin no

This is a sensible default setting; we will leave it as it is. We want that, if the root account has to be used, there is a record of who did it. The next one too, is appropriate:

#StrictModes yes

With this setting sshd accepts login only if no world-writable directories or files are in the user home directory. We will change the next one:

#PubkeyAuthentication yes

To:

# Changed on 04/04/2004
PubkeyAuthentication no

Since we are not going to use Public Key authentication there is no reason to leave it in place. An attacker could create a backdoor on a compromised account by placing a public key in the account home directory known_hosts and later login without being detected by the owner of the account. Note that this also prevents host-based authentication since it works together with public key authentication.

# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no

In line with what was said above, we leave password authentication enabled and, of course, prevent the creation of null password accounts. Furthermore we will disable the authentication method controlled by the next parameter:

# Change to no to disable PAM authentication
#ChallengeResponseAuthentication yes

To:

# Change to no to disable PAM authentication
# Changed on 04/04/2004
ChallengeResponseAuthentication no

The default value allows authentication to go through the Pluggable Authentication Modules mechanism. This could, in fact, override the authentication we chose to

use (password-based) with another one. A similar motivation applies to the next parameter:

```
# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
```

This controls whether the Kerberos protocol can be used to authenticate users. We decided to dissociate the account used by supervisors to do their daily work (authenticated through Kerberos), from the one they use to check logs on this machine. They will also have a different one to manage Kerberos accounts (via a different Kerberos instance). This way, if their daily account gets compromised, the attacker will not be able to compromise the Kerberos database. If the Kerberos admin account is compromised they will not be able to compromise the KDC.

We will then leave these parameters untouched.

Note that from a user perspective, this adds only the inconvenience of further password to remember for the supervisors because even if we used Kerberos authentication, ssh would re-authenticate users to establish the connection and they would have to retype their Kerberos password. As said when describing the Kerberos protocol, a kerberised application can still decide to re-authenticate the client.

```
#X11Forwarding yes
```

Even though we did not install the X11 software, we disable forwarding for good measure.

```
# Changed on 04/04/2004
X11Forwarding no
```

We also want ssh to use the login and session limitations that we will define later on:

```
#UseLogin no
```

Will become:

```
# Changed on 04/04/2004
UseLogin yes
```

Finally, we also disable the sftp subsystem since we do not want someone to be able to upload anything to the server. Thus the next parameter:

```
# override default of no subsystems
Subsystem    sftp    /usr/libexec/sftp-server
```

Will be changed to:

```
# override default of no subsystems
# Changed on 04/04/2004
#Subsystem  sftp    /usr/libexec/sftp-server
```

There is one more parameter we need to change, not listed in the default config: “AllowTcpForwarding”. This controls the tunneling of connections over ssh and is enabled by default. While a useful feature in certain cases, it could be used maliciously to attack another hosts while disguising as the KDC.

We will turn it off by adding:

```
# Added on 04/04/2004
AllowTcpForwarding no
```

#### 5.3.5.4 Network Interface

We will now configure the network interface first by adding the lines below to “/etc/rc.conf”:

```
ifconfig_lnc0="inet 192.168.1.5  netmask 255.255.255.0"
defaultrouter="192.168.1.1"
hostname="giac.sabbi.net"
```

then creating “/etc/resolv.conf” with the following content:

```
domain        sabbi.net
nameserver   192.168.1.20
```

And finally adding the following to “/etc/hosts”:

```
192.168.1.5          giac.sabbi.net giac
192.168.1.5          giac.sabbi.net.
```

At the next reboot, the interface will be brought up.

#### 5.3.5.5 Adding a user

Now that the networking is configured, we will reboot the host to make sure that our network configuration is correct. After reboot the machine will be reachable via

ssh and we will be able to leave the console (usually located in an uncomfortable location). Before that, though, we need to create a normal user account or we will not be able to login. Remember that we have disabled direct root logins and root is still the only interactive account in the host.

Below is the input sequence to create the account, with comments on interesting points:

giac# **adduser**

Use option ``-silent'' if you don't want to see all warnings and questions.

Since it is the first time we create a user, the system asks questions in order to set default parameters for user account creation.

Check /etc/shells

Check /etc/master.passwd

Check /etc/group

Usernames must match regular expression:

[^[a-z0-9_][a-z0-9_-]*$]: **<enter>**

Enter your default shell: csh date no sh tcsh [sh]: **tcsh**

Your default shell is: tcsh -> /bin/tcsh

Enter your default HOME partition: [/home]: **<enter>**

Copy dotfiles from: /usr/share/skel no [/usr/share/skel]: **<enter>**

Send message from file: /etc/adduser.message no

[/etc/adduser.message]: **<enter>**

Use passwords (y/n) [y]: **<enter>**

Write your changes to /etc/adduser.conf? (y/n) [n]: **y**

We confirm we want to make these the defaults for user creation. Now we can go on and create the user:

Ok, let's go.

Don't worry about mistakes. I will give you the chance later to correct any input.

Enter username [^[a-z0-9_][a-z0-9_-]*$]: **roby**

Enter full name []: **Roberto Sabbi**

Enter shell csh date no sh tcsh [tcsh]: **<enter>**

Enter home directory (full path) [/home/roby]: **<enter>**

Uid [1000]: **<enter>**

Enter login class: default []: **<enter>**

Login group roby [roby]: **<enter>**

Login group is ``roby''. Invite roby into other groups: guest no

[no]: **wheel**

We have included the user into the wheel group otherwise he will not be allowed to "su" to root.

Enter password []: **<hidden>**

Enter password again []: **<hidden>**

Name: roby

Password: ********

Fullname: Roberto Sabbi

Uid: 1000

Gid: 1000 (roby)

Class:

Groups: roby wheel

HOME: /home/roby

Shell: /bin/tcsh

OK? (y/n) [y]: **y**

Added user ``roby''

Send message to ``roby'' and: no root second_mail_address [no]: **<enter>**

Roberto Sabbi,

your account ``roby'' was created.

Have fun!

See also chpass(1), finger(1), passwd(1)

Add anything to default message (y/n) [n]: **<enter>**

Send message (y/n) [y]: **n**

Copy files from /usr/share/skel to /home/roby

Add another user? (y/n) [y]: **n**

Goodbye!

giac#

After rebooting the machine we login as the normal user created above and “su” to root. We will start by checking once again if our configuration was successful and then move into local services:

giac# **netstat -na**

Active Internet connections (including servers)

| Proto | Recv-Q | Send-Q | Local Address | Foreign Address | (state) |
|---|---|---|---|---|---|
| tcp4 | 0 | 0 | 192.168.1.5.22 | 192.168.1.100.1041 | ESTABLISHED |
| tcp4 | 0 | 0 | 127.0.0.1.25 | *.* | LISTEN |
| tcp4 | 0 | 0 | 192.168.1.5.22 | *.* | LISTEN |

Active UNIX domain sockets

| Address | Type | Recv-Q | Send-Q | Inode | Conn | Refs | Nextref | Addr |
|---|---|---|---|---|---|---|---|---|
| d658fc80 | stream | 0 | 0 | 0 | d658fd20 | 0 | 0 | |
| d658fd20 | stream | 0 | 0 | 0 | d658fc80 | 0 | 0 | |
| d658fe60 | dgram | 0 | 0 | 0 | d6569f00 | 0 | d658ff00 | |
| d658ff00 | dgram | 0 | 0 | 0 | d6569f00 | 0 | 0 | |
| d6569f00 | dgram | 0 | 0 | d6563ac0 | 0 | d658fe60 | 0 | /var/run/log |

giac#

The output confirms that sendmail is listening only locally on port 25 for IPv4, sshd on 22 and syslogd in no longer listening. We can also see our ssh connection.

#### 5.3.5.6 Time Syncronisation

The typical time drift observed on another machine of the same model is in the order of 5 seconds per day. Since later we will configure the securelevel to allow only 1 second time correction, we need to adjust at least 5 times per day. We will stay on the safe (but reasonable) side and adjust every hour.

The following entry will be put in the root crontab:

```
@hourly     root    /usr/sbin/ntpdate time.sabbi.net > /dev/null
```

We also need to set the timezone since the system clock will be running in localtime:

```
giac# cp /usr/share/zoneinfo/Europe/Rome /etc/localtime
giac#
```

We have decided to run on local time rather than GMT to make life easier for the users when checking logs.

#### 5.3.5.7 Local Processes

It is time to turn off any unneeded local services, we will first list them:

```
giac# ps ax -o pid,user,command
  PID USER COMMAND
    0 root  (swapper)
    1 root /sbin/init --
    2 root  (taskqueue)
    3 root  (pagedaemon)
    4 root  (vmdaemon)
    5 root  (bufdaemon)
    6 root  (syncer)
    7 root  (vnlru)
   73 root /usr/sbin/syslogd -ss
```

```
  81 root /usr/sbin/inetd -wW
  83 root /usr/sbin/cron
  85 root /usr/sbin/sshd
  88 root sendmail: accepting connections (sendmail)
 112 root sshd: roby [priv] (sshd)
 114 roby sshd: roby@ttyp0 (sshd)
 115 roby -tcsh (tcsh)
 117 root _su (csh)
 134 root ps ax -o pid,user,command
 104 root login [pam] (login)
 122 root -csh (csh)
 105 root /usr/libexec/getty Pc ttyv1
 106 root /usr/libexec/getty Pc ttyv2
 107 root /usr/libexec/getty Pc ttyv3
 108 root /usr/libexec/getty Pc ttyv4
 109 root /usr/libexec/getty Pc ttyv5
 110 root /usr/libexec/getty Pc ttyv6
 111 root /usr/libexec/getty Pc ttyv7
giac#
```

A first observation: the services inside round brackets are actually kernel tasks (e.g. the syncer is the task that flushes the disk buffer).

The second thing to note is that, although not listening on any port, inetd is still running. We will disable it by adding:

inetd_enable="NO"

To "/etc/rc.conf".

#### 5.3.5.8 Console and accounts settings

In the output above we then notice that there are 7 virtual consoles running (the "getty" processes). They are too many and we do not want someone operating on

the console forgetting a session logged in because he switched to another virtual console. If the console freezes, there is still the option of connecting via the network and viceversa. If this is not possible then the whole operating system is frozen.

A further concern is the single user mode: with the present settings, no password is asked. This means that having access to the console and the power button or cable (thus not necessarily the chassis), yields root access. We will prevent that as a complement to the physical measures already in place. Also we want to prevent someone that has stolen the root password (and is not in the wheel group) to use it. In other words, we want to prevent direct root login at the console even in multi-user mode.

On a higher level, we want to make sure that every use of the root password, passes from the envelope stored in the safe.

To summarise we replace the corresponding lines on the “/etc/ttys” file with the ones below:

```
# If console is marked "insecure", then init will ask for the root password
# when going to single-user mode.
console none                            unknown off insecure
#
ttyv0   "/usr/libexec/getty Pc"         cons25  on  insecure
# Virtual terminals
ttyv1   "/usr/libexec/getty Pc"         cons25  off secure
ttyv2   "/usr/libexec/getty Pc"         cons25  off secure
ttyv3   "/usr/libexec/getty Pc"         cons25  off secure
ttyv4   "/usr/libexec/getty Pc"         cons25  off secure
ttyv5   "/usr/libexec/getty Pc"         cons25  off secure
ttyv6   "/usr/libexec/getty Pc"         cons25  off secure
ttyv7   "/usr/libexec/getty Pc"         cons25  off secure
ttyv8   "/usr/X11R6/bin/xdm -nodaemon"  xterm   off secure
```

We have turned off all virtual consoles except one, and marked it insecure together with the single user mode.

We will now harden the login security parameters by editing the “default:” section in “/etc/login.conf”. We enforce a minimum password length of 8 characters. Remember that the most recurring logins will be those of the supervisors to check logs.

The first addition will be:

```
# Changed on 04/04/2004
:login-backoff=5:\
```

After 3 unsuccessful login attempts, the system will introduce a delay after each new login attempts. This is very useful to prevent password guessing activity.

```
:login-retries=5:\
```

We limit the number of wrong password typed in before returning to the login prompt. Note that the two parameters described above are overridden by sshd (they will be enforced at the console).

```
:sessiontime=1h:\
:warntime=5m:\
:sessionlimit=1:\
```

The supervisors normal activity will not require more than 1 session open for a maximum of 1 hour. They will be warned 5 minutes before their session expires.

```
:autodelete=1w:\
:minpasswordlen=8:\
:passwordtime=90d:\
:warnexpire=1w:\
```

With the above settings, accounts will be automatically deleted after one week from expiration if not re-activated. Passwords must have 8 characters minimum, must be changed every 90 days and a warning will be given one week in advance. These are reasonable settings for md5 passwords and take into account the human factor: if we forced the supervisors (that also do normal CAD work) to change more often, we would create the risk of having them write passwords.

This takes into account the fact that they will also have their normal Kerberos login plus the administrative one. The “autodelete” feature, avoids having “zombie” accounts in the host.

:idletime=5:\

This parameter would be very useful since it would automatically logoff idle sessions. Unfortunately the “login.conf” man page informs us that this parameter is not enforced.

#### 5.3.5.9 Filesystems

We will now try to enforce some measures to protect the integrity of the filesystems by modifying the “/etc/fstab” mount options for the following entries:

```
/dev/ad0s1e          /home          ufs  rw,nodev,noexec,nosymfollow
```

In our installation, home directories should not be used very much, preventing device files from functioning, any executable from being run, and symlinks from being followed. These are all opportunities for malicious activity.

```
/dev/ad0s1h          /tmp           ufs  rw,nodev,noexec,nosymfollow
```

We only have one application running thus the temporary system directory will not be used much, if at all.

```
/dev/ad0s1d          /usr           ufs  rw,nodev
```

There is not much we can restrict on “/usr”, since most executables are located there and symlinks are used, among other, for libraries. However, device files have no purpose in there.

```
/dev/ad0s1f          /var           ufs  rw,nodev,noexec,nosymfollow
```

Under “/var”, there should only be data and symlinks in our machine will not be used.

```
/dev/ad0s1g          /var/heimdal   ufs  rw,nodev,noexec,nosymfollow
```

The Heimdal directory is similar to the rest of the “/var” filesystem.

```
/dev/acd0c          /cdrom      cd9660  ro,noauto,nosuid
```

We will use the cdrom for system updates, with a privileged account, therefore we will disable suid so that when the disc will be mounted, unprivileged users will not be able to run executables with more privilege then their own.

To confirm our settings we will reboot and verify:

```
giac# mount
/dev/ad0s1a on / (ufs, local)
/dev/ad0s1e on /home (ufs, local, nodev, noexec, nosymfollow, soft-updates)
/dev/ad0s1h on /tmp (ufs, local, nodev, noexec, nosymfollow, soft-updates)
/dev/ad0s1d on /usr (ufs, local, nodev, soft-updates)
/dev/ad0s1f on /var (ufs, local, nodev, noexec, nosymfollow, soft-updates)
/dev/ad0s1g on /var/heimdal (ufs, local, nodev, noexec, nosymfollow, soft-updates)
procfs on /proc (procfs, local)
giac#
```

#### 5.3.5.10 Securelevel

These settings could be changed on the fly by root, by remounting the filesystems. To prevent this from happening we have to use the “securelevel” feature. It is a state on which certain kernel runtime parameters become unalterable even by root. Only by modifying the “securelevel” settings and rebooting, the situation can be modified.

We will use a value of 2 which prevents disc devices from being written. In other words it prevents also unmounting and remounting them and using newfs. It also prevents the time from being changed by more than one second. This is very important in our case since validation of Kerberos tickets relies on accurate time synchronisation. We will enable securelevel 2 by adding the following to /etc/rc.conf:

```
kern_securelevel_enable="YES"
kern_securelevel="2"
```

Let us verify if the setup is effective (via ssh):

giac# **mount -u –o rw /var/heimdal/**

mount: not currently mounted /var/heimdal/

giac#

A word of caution: repeating the same command sequence directly from the console still works (note the "/var/heimdal" entry below).

giac# **mount -u –o rw /var/heimdal/**

giac# **mount**

/dev/ad0s1a on / (ufs, local)

/dev/ad0s1e on /home (ufs, local, nodev, noexec, nosymfollow, soft-updates)

/dev/ad0s1h on /tmp (ufs, local, nodev, noexec, nosymfollow, soft-updates)

/dev/ad0s1d on /usr (ufs, local, nodev, soft-updates)

/dev/ad0s1f on /var (ufs, local, nodev, noexec, nosymfollow, soft-updates)

/dev/ad0s1g on /var/heimdal (ufs, local, soft-updates)

procfs on /proc (procfs, local)

giac#

Let us see if we can change the time by putting it back five minutes as if we were trying a replay attack:

giac# **date**

Tue Apr 13 08:20:52 GMT 2004

giac# **date 200404130815**

Tue Apr 13 08:15:00 GMT 2004

giac# **date**

Tue Apr 13 08:21:40 GMT 2004

giac# **dmesg**

…

<Output omitted>

…

Time adjustment clamped to -1 second

giac#

This time repeating from the console yields the same result.

## 5.4 Software installation

### 5.4.1 Package build

We can proceed now to software installation. This host does not have access to the Internet, so the installation can only be done from the laptop of the installer. This computer is running the same version of FreeBSD and has “cvsup” installed. It is the most practical way to update the FreeBSD sources. One of this source tree is the ports source tree, a sophisticate system of scripts to compile software automatically from sources.

Basically, by “cd”-ing into a port main directory and issuing a “make && make install” the source code of the corresponding software is downloaded from the Internet, patched for the FreeBSD platform (if needed) and installed on the system. All dependencies are also recursively built using the same mechanism.

It is also possible to pass compile options along with those defined in the system-wide” /etc/defaults/make.conf” or in the “/etc/make.conf”.

Note also that the integrity of the downloaded sources is verified by calculating an md5 hash to be compared with the one pre-existing in the distinfo file located in the port directory.

Alternatively by passing the “package” target to the make utility, the port is also packaged in a “.tgz” file that can be installed in a system without ports and access to the Internet.

On the laptop we will run the following:

cvsup –g –L 2 –h cvsup.fr.freebsd.org /usr/share/examples/cvsup/ports-supfile

After much output we will have the whole /usr/ports tree up-to-date to the latest version.

We have said that we will use Cracklib support option for Heimdal. We will simply include it by issuing:

frodo # cd /usr/ports/security/cracklib

frodo # make

frodo # make package

and then:

frodo # cd /usr/ports/security/heimdal

frodo # make –DWITH_CRACKLIB

frodo # make –DWITH_CRACKLIB package

In both cases we will see the sources downloading and building. In the end in each port directory a ".tgz" package will be present.

### 5.4.2 Package Installation

We will now copy these packages on a directory on our server and install them:

giac # pkg_add cracklib-2.7_2.tgz

giac # pkg_add heimdal-0.6.1.tgz

giac# pkg_info

cracklib-2.7_2 Password-checking library

heimdal-0.6.1 A re-implementation of Kerberos V

giac#

The "pkg_info" utility, invoked with no options reports the installed packages.

### 5.4.3 Configuration

The two packages are installed so we can proceed to the Kerberos configuration. There is no default Kerberos configuration file so we will create one:

```
[libdefaults]
    default_realm = SABBI.NET
    forwardable = yes
    krb4_get_tickets = no
[realms]
```

```
   SABBI.NET = {
      kdc = giac.sabbi.net
      admin_server = giac.sabbi.net
   }
[logging]
   kdc = FILE:/var/log/kdc.log
   kadmind = FILE:/var/log/kadmind.log
   kpasswdd = FILE:/var/log/kpasswd.log
   default = FILE:/var/log/Kerberos.log


[kdc]
   enable-kerberos4 = no
   require-preauth = yes
   check-ticket-addresses = yes
   allow-null-ticket-addresses = no
   kdc_warn_pwexpire = 7d
[domain_realm]
   .sabbi.net = SABBI.NET
[kadmin]
   require-preauth = yes


[libdefaults]
```

Even the KDC needs to use Kerberos authentication for its operations so, like any other client, it must know the default realm it should assume to belong to.

He will try to obtain “forwardable” tickets so they will be usable directly for application authentication (subject to applications will).

No Kerberos IV tickets will be requested: since we have are setting up a realm from scratch there is no need to support the previous version of the protocol. This is because all client hosts support Kerberos v5.

[realms]

Again a section in common with clients, its purpose being the addresses of the KDC and the Kadmin server (usually they are on the same machine). The Kadmin service allows remote management of principals without having to interactively login on the KDC.

The Kpasswd service allowing users to change their password, if not specified it is assumed to be co-located with the Kadmin service.

[logging]

The KDC logs will also be used to check temp workers presence and activity, and regular change of password by the supervisors. Thus we want to have all the logs centralised in 4 dedicated files.

[kdc]

Our KDC will not provide tickets to v4 clients. This may be needed in the future to support some operating system. For now we avoid it since we would be giving out tickets encrypted with the 56-bit single DES algorithm with a risk of brute force, offline attack. Even to v5 hosts.

We will also mandate clients pre-authentication: by default Heimdal provides a ticket to any client requesting it for a given principal. This can be used to mount an offline attack. With the above setting an additional exchange is requested to the client to prove its identity.

As said in the protocol description, IP addresses are part of the tickets. The last two options make sure that only tickets with addresses are used and that those addresses are the ones for which the ticket can be used. This option provides a marginal security improvement since IP addresses can be faked and it is turned off when NAT has to be traversed. In our case we operate on the same network.

We also give a warning to users when their password is about to expire.

[domain_realm]

Kerberos needs to map the realm to a DNS domain. By convention the realm is the domain name all in uppercase. Beside an evident need to avoid confusion, it is also useful when client are only partially configured: the Kerberos libraries can figure out the realm from the domain.

[kadmin]

We obviously want to require pre-authentication for administrative connections too.

It has to be noted that no entries about realms or domains would be needed with an appropriate DNS setup. In our case the choice was deliberately made to rely as little as possible on network services.

### 5.4.4 Startup

Now we need to make the necessary in order to automatically startup the KDC. The installation has copied the startup script in the “/usr/local/etc/rc.d/” directory.

We need to edit the sample kdc.sh.sample script.

# One of `none', `master', or `slave'

KDC_ROLE=none

This will become:

KDC_ROLE=master

Next we will enable Cracklib support:

# Uncomment to use CrackLib

#KPASSWDD_FLAGS="--check-library=${PREFIX}/lib/kpasswdd-cracklib.so"

As suggested we will uncomment the above line.

Apparently, either the Heimdal developers or the FreeBSD port maintainer do not envisage the possibility of running a single KDC with no slave. Thus we need to comment out the next line:

IPROPD_MASTER="${PREFIX}/libexec/ipropd-master"

In order to prevent “ipropd-master” from starting. This is the daemon responsible for propagating the Kerberos database to the slave KDC.

For automatic startup we need to remove the “.sample” extension and to make the script executable for both the owner and group.

The script does not start “kadmind” the remote administration daemon. This could be due to the fact that it can be both run standalone or from “inetd”. We will go for the first option, since the Heimdal info page advises against the inetd option. We will then insert:

# Changed on 04/04/2004

/usr/local/libexec/kadmind &

before the last “exit 0” statement, on the kdc.sh startup script.

As usual we will verify immediately with a reboot the effectiveness of our configuration:

```
giac# ps ax -o pid,user,command
  PID USER COMMAND
    0 root  (swapper)
    1 root /sbin/init --
    2 root  (taskqueue)
    3 root  (pagedaemon)
    4 root  (vmdaemon)
    5 root  (bufdaemon)
    6 root  (syncer)
    7 root  (vnlru)
   28 root adjkerntz -i
   72 root /usr/sbin/syslogd -ss
   82 root /usr/sbin/cron
   84 root /usr/sbin/sshd
   87 root sendmail: accepting connections (sendmail)
  126 root sshd: roby@ttyp0 (sshd)
  128 roby -tcsh (tcsh)
  130 root _su (csh)
  132 root ps ax -o pid,user,command
  110 root login [pam] (login)
  111 roby -tcsh (tcsh)
  113 root _su (csh)
  103 root /usr/local/libexec/kdc
104 root /usr/local/libexec/kpasswdd --check-library=/usr/local/lib/kpasswdd-
giac#
```

As we were expecting both the “kdc and “kpasswdd” are running. We do not see kpasswd since we run it with “inetd”. We need again “netstat”:

```
giac# netstat -a
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
```

```
tcp4       0      0  giac.ssh              192.168.1.100.1109     ESTABLISHED
tcp4       0      0  localhost.kerberos-sec *.*                    LISTEN
tcp4       0      0  giac.kerberos-sec      *.*                    LISTEN
tcp6       0      0  localhost.kerberos     *.*                    LISTEN
tcp4       0      0  *.kerberos-adm         *.*                    LISTEN
tcp46      0      0  *.kerberos-adm         *.*                    LISTEN
tcp4       0      0  localhost.smtp         *.*                    LISTEN
tcp4       0      0  giac.ssh               *.*                    LISTEN
udp4       0      0  localhost.kerberos-sec *.*
udp4       0      0  localhost.kpasswd5     *.*
udp6       0      0  localhost.kpasswd5     *.*
udp4       0      0  giac.kpasswd5          *.*
udp4       0      0  giac.kerberos-sec      *.*
udp6       0      0  localhost.kerberos     *.*
Active UNIX domain sockets
Address  Type   Recv-Q Send-Q    Inode     Conn     Refs  Nextref Addr
d6593e60 dgram       0      0        0 d656df00        0 d6593f00
d6593f00 dgram       0      0        0 d656df00        0        0
d656df00 dgram       0      0 d6568940        0 d6593e60        0 /var/run/log
giac#
```

Everything is running as expected.

There is just one more step needed in order to have proper passwords. We will setup a symbolic link named /etc/krb5.dict pointing at /usr/share/dict/words.

According to the FreeBSD Handbook:

> When setting up a krb5.dict file to prevent specific bad passwords from being used (the manual page for kadmind covers this briefly), remember that it only applies to principals that have a password policy assigned to them[6].

[6] The FreeBSD Documentation Project. "The FreeBSD Handbook" System Documentation. URL: http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/book.html#KERBEROS5 (01 Apr. 2004)

Unfortunately, the author has not found any documentation on how to set policies for users.

We can now test our server by creating the master key and then the Kerberos database:

giac# **kstash**

Master key:

Verifying - Master key:

kstash: writing key to `/var/heimdal/m-key'

giac#

The “kstash” command creates a key that will be used by Kerberos to encrypt/decrypt the database. Being in the same directory it is clear that the security provided against an attacker is relatively little. This, in fact it is aimed at backups security, provided that the key is not stored in the same media.

Now we will initialise the database; this also gives us an opportunity to test the “kadmin” service:

giac# **kadmin -l**

kadmin> **init SABBI.NET**

Realm max ticket life [unlimited]:8h

Realm max renewable ticket life [unlimited]:8h

kadmin>

We have set the default ticket life to the typical duration of a working day for the supervisors. We also allow another 8 hours extension if the client renews the ticket before its expiry, since sometime the temps work two shifts in a row. Note that this does not lower security since a new ticket is issued on the trust of the current one.

kadmin> **list ***

default@SABBI.NET

kadmin/admin@SABBI.NET

kadmin/hprop@SABBI.NET

kadmin/changepw@SABBI.NET

krbtgt/SABBI.NET@SABBI.NET

changepw/kerberos@SABBI.NET

kadmin>

Here is a list of the principals needed by the KDC for the services it provides. We will now be able to create a normal user, an admin user and a admin that will be able to manage admins.

kadmin> **add roby**

Max ticket life [1 day]:**8h**

Max renewable life [1 week]:**8h**

Principal expiration time [never]:**2004-05-01**

Password expiration time [never]:**2004-04-20**

Attributes []:**<enter>**

roby@SABBI.NET's Password:

Verifying - roby@SABBI.NET's Password:

kadmin> **add roby/admin**

Max ticket life [1 day]:**8h**

Max renewable life [1 week]:**8h**

Principal expiration time [never]:**2004-05-01**

Password expiration time [never]:**2004-04-20**

Attributes []:**<enter>**

roby@SABBI.NET's Password:

Verifying - roby@SABBI.NET's Password:

kadmin> **add roby2/admin**

Max ticket life [1 day]:**8h**

Max renewable life [1 week]:**8h**

Principal expiration time [never]:**2004-05-01**

Password expiration time [never]:**2004-04-20**

Attributes []:**<enter>**

roby@SABBI.NET's Password:

Verifying - roby@SABBI.NET's Password:

kadmin> **list ***

roby@SABBI.NET

```
default@SABBI.NET
roby/admin@SABBI.NET
roby2/admin@SABBI.NET
kadmin/admin@SABBI.NET
kadmin/hprop@SABBI.NET
kadmin/changepw@SABBI.NET
krbtgt/SABBI.NET@SABBI.NET
changepw/kerberos@SABBI.NET
kadmin>
```

The three users are created. Now we will setup the ACL file ("/var/heimdal/kadmind.acl").

```
giac# cat /var/heimdal/kadmind.acl
roby/admin@SABBI.NET all *
roby2/admin@SABBI.NET all */admin
giac#
```

This is an example of how the supervisors account will be setup. The principal "roby/admin" is an example of a normal supervisor. He will be able to perform all accounts modification but only on users without instance. The principal "roby2/admin" will be able to manage "/admin" principals.

We will now try to acquire a ticket from another host:

```
frodo# kinit roby
roby@SABBI.NET's Password:
kinit: NOTICE: ticket renewable lifetime is 8 hours
frodo# klist
Credentials cache: FILE:/tmp/krb5cc_0
        Principal: roby@SABBI.NET

  Issued           Expires          Principal
Apr 13 17:52:51  Apr 14 01:52:51  krbtgt/SABBI.NET@SABBI.NET
```

frodo# **ls -la /tmp/**

total 8

drwxrwxrwt 3 root wheel 512 Apr 13 17:55 .

drwxr-xr-x 17 root wheel 512 Apr 8 12:24 ..

-rw------- 1 root wheel 451 Apr 13 17:55 krb5cc_0

frodo# **kdestroy**

Our KDC seems to work well. Now we will look at some kdc.log output:

frodo# **cat kdc.log**

2004-04-13T18:11:18 AS-REQ roby/admin@SABBI.NET from IPv4:192.168.1.5 for kadmin/admin@SABBI.NET

2004-04-13T18:11:18 No PA-ENC-TIMESTAMP -- roby/admin@SABBI.NET

2004-04-13T18:11:18 sending 216 bytes to IPv4:192.168.1.5

2004-04-13T18:11:18 AS-REQ roby/admin@SABBI.NET from IPv4:192.168.1.5 for kadmin/admin@SABBI.NET

2004-04-13T18:11:18 Looking for pa-data -- roby/admin@SABBI.NET

2004-04-13T18:11:18 Pre-authentication succeded -- roby/admin@SABBI.NET

2004-04-13T18:11:18 Using des3-cbc-sha1/des3-cbc-sha1

2004-04-13T18:11:18 sending 627 bytes to IPv4:192.168.1.5

frodo#

The AS-REQ entry is for Authentication Service Request. This is the TGT request. The “PA-ENC-TIMESTAMP” shows that KDC effectively requires pre-authentication (PA) and the negotiated encryption is triple des.

Now we will try changing a password into a 4 character one:

frodo# **kinit roby**

roby@SABBI.NET's Password:

frodo# **kpasswd**

roby@SABBI.NET's Password:

New password:

Verifying - New password:

Soft error : it is too short
frodo#

Now we will use the word “electric”:

frodo# **kpasswd roby**
roby@SABBI.NET's Password:
New password:
Verifying - New password:
Soft error : it is based on a dictionary word
frodo#

Now a valid one:

frodo# **kpasswd**
roby@SABBI.NET's Password:
New password:
Verifying - New password:
Success : Password changed
frodo#

Cracklib seems to do its work. Below the corresponding “kpasswd.log” entries:

2004-04-13T18:24:17 Changing password for roby@SABBI.NET
2004-04-13T18:24:17 it is too short
2004-04-13T18:25:06 Changing password for roby@SABBI.NET
2004-04-13T18:25:06 it is based on a dictionary word
2004-04-13T18:28:07 Changing password for roby@SABBI.NET

# 6 Maintenance

## 6.1 Operating system update

FreeBSD features an automatic build process that allows rebuilding the system from the source files.

Similarly to the software update process we will update the sources on the laptop:

frodo# **cvsup –g –L 2 –h cvsup.fr.freebsd.org \ /usr/share/examples/cvsup/stable-supfile**

Frodo# cd /usr/src

Frodo# make buildworld

Frodo# make buildkernel

Then we will copy the “/usr/obj” and the “/usr/src” directories on a cd that we will mount on the KDC on the respective directories.

Giac# make installkernel

Reboot the KDC:

Giac# make installworld

Followed by another reboot.

## 6.2 Application update

The application patching process will consist in repeating the one used for the installation. The only additional step will be to execute a “pkg_delete” prior to the “pkg_add” of the new package.

## 6.3 Backups and redundant storage

The following script will be used to backup the Kerberos database. Note that the master is not copied therefore the backup is encrypted.

```
# !/bin/sh
cd /root
cdcontrol -vf /dev/acd0c close
sleep 30
burncd -v -f /dev/acd0c -s 4 -e blank
sleep 30
cdcontrol -v -f /dev/acd0c close
rm /root/tmp/*
/usr/local/etc/rc.d/kdc.sh stop
cp /var/heimdal/heimdal.db /root/tmp/SC_heimdal.db
/usr/local/etc/rc.d/kdc.sh start
mkisofs -r -J -o /root/tmp
rm /root/tmp/*
sleep 30
burncd -v -f /dev/acd0c -s 4 -e data /root/bck.iso fixate
rm /root/bck.iso
```

# 7 Auditing

The verification of the system configuration was done during the installation. The only configuration not thoroughly tested was the OpenSSH one although some of the verifications were performed both on the console and via “ssh”. Thus, at least the accounts parameters can be considered as valid.

# 8 List of references

The FreeBSD Release Engineering Team. “Early Adopter's Guide to FreeBSD 5.2.1-RELEASE”. 25 Feb. 2004. URL: http://www.freebsd.org/releases/5.2.1R/early-adopter.html (09 Apr. 2004).

The FreeBSD Release Engineering Team. “FreeBSD Security Information”. 25 Feb. 2004. URL: http://www.freebsd.org/security/ (09 Apr. 2004).

Garman, Jason. Kerberos, The Definitive Guide. Sebastopol: O’Reilly & Associates, Inc, August 2003.

Steiner, Neuman,Schiller. Kerberos:An Authentication Service for Open Network System. 30 Mar. 1988. URL: http://www.pdc.kth.se/kth-krb/kerberos.ps (09 Apr. 2004)

Muffet, Alec. “CrackLib: A ProActive Password Sanity Library.” 14 Oct. 2003.URL: http://www.crypticide.org/users/alecm/security/cracklib,2.7.txt (09 Apr. 2004)

The FreeBSD Documentation Project. “The FreeBSD Handbook” System Documentation. URL: http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/book.html#KERBEROS5 (01 Apr. 2004)

Also, not directly referenced in the text but of great inspiration in getting started:

Patterson,D. Securing FreeBSD step by step. GCUX Practical (30 May 2003)

## End of document