

VIMINAL (*)

Virtual Model for Ip Network Architecture Lab

(*) The **Viminal Hill** (Latin *Collis Viminalis*, Italian *Viminale*) is the smallest and least important of the famous [seven hills](http://en.wikipedia.org/wiki/Seven_hills_of_Rome) of [Rome](http://en.wikipedia.org/wiki/Rome), and as such always referred to as *collis* rather than *mons*. (wikipedia : http://en.wikipedia.org/wiki/Viminal_Hill)

Introduction

VIMINAL (Virtual Model for Ip Network Architecture Lab) plate-form is an autonomous network and system lab environment. Available on a liveCD, it offers network models on which you have extended rights. It integrates all the materials needed to securely play system and IP network labs on common computers. The main goal is to play such labs, with no installation or configuration modifications on your computer. VIMINAL liveCD is based on VNUML (Virtual Network User-Mode Linux <http://jungla.dit.upm.es/~vnuml/>), a neat project of Telematics Engineering Department (DIT) of the Technical University of Madrid (UPM).

What is VNUML ?

"VNUML stands for Virtual Network User Mode Linux. It is an open-source general purpose virtualization tool designed to quickly define and test complex network simulation scenarios based on the great User Mode Linux (UML) virtualization software. It has been initially developed in the context of Euro6IX research project to simulate IPv6 IX scenarios based on Linux and zebra/quagga routing daemons (see our recent article on IEEE Comms. Magazine). However, it is a useful tool that can be used to simulate general Linux based network scenarios. VNUML is aimed to help in testing network applications and services over complex testbeds made of several nodes (even tenths) and networks inside one Linux machine, without involving the investment and management complexity needed to create them using real equipment. VNUML tool is made of two main components: the VNUML language used for describing simulations in XML; and the interpreter of the language (vnuml command), that builds and manages the simulation, hiding all UML complex details to the user. VNUML has been developed by the Telematics Engineering Department (DIT) of the Technical University of Madrid (UPM) in Spain. This software is released under GNU Public Licence. It has been developed with the partial support from the European Commission under the Euro6IX IST research project."

More information can be found on the [VNUML web site](#)

Authorisations and privileges considerations

- VIMINAL liveCD boots the computer under the « viminal » user, with low level rights. So

user viminal has no privilege on the real host. Super-user account (root) having a new random password on each liveCD reboot, there's no simple way to gain super user privileges on the real computer.

- On each VNUML virtual machine you have super-user privileges. I know root with no password, it's bad ! But remember user-mode linux virtual machines are sandbox with no particular privilege on the real host.

Environments

- VIMINAL is based on Gentoo. Why ? First because that's my usual distribution. For me there's no bad GNU/Linux distribution. The good one, is the distribution you use, you know how to configure and you are able to easily update. Second Gentoo has a good liveCD generator tool named Catalyst2. With Catalyst2 creating and updating dedicated liveCD is two conf files and some basic bash scripts.
- As a convenience VIMINAL liveCD embeds two graphical desktops. The host desktop is KDE, the virtual machine desktop is Fluxbox. We choose two different desktop window managers to easily distinguish real host desktop and virtual machine desktop.
- Two modes are available to have direct virtual machine access : Console mode with SSH and graphic desktop with VNC. VNC is used instead of a direct X-window export because VNC has a native session mode. You can leave the virtual machine desktop, and find it in the same state when you reconnect.

Getting started

First of all, you have to boot your computer on the liveCD. Depending on your computer configuration, you may have to change the boot sequence order in the bios (see your computer notice to find how to change your boot sequence order if not booting first on cdrom). Boot step automatically starts linux, and ask your keyboard preference for the tty terminal (choose you keyboard preference by typing your two letters iso country code if you don't have an 'us' keyboard which is the default).

```

:: Scanning for sata_qstor...sata_qstor loaded.
:: Scanning for ahci...ahci loaded.
:: Scanning for ata_piix...ata_piix loaded.
:: Scanning for dm-mod...dm-mod loaded.
:: Scanning for dm-mirror...dm-mirror loaded.
>> Activating udev
>> Making tmpfs for /newroot
>> Attempting to mount CD:- /dev/hdc
>> CD medium found on /dev/hdc
>> Loading keymaps
Please select a keymap from the following list by typing in the appropriate
name or number. Hit Enter for the default "us/41" US English keymap.

 1 azerty   7 cf       13 es      19 il      25 mk      31 ru      37 trf
 2 be       8 croat    14 et      20 is      26 nl      32 se      38 trq
 3 bg       9 cz       15 fi      21 it      27 no      33 sg      39 ua
 4 br-a    10 de      16 fr      22 jp      28 pl      34 sk-y     40 uk
 5 br-l    11 dk      17 gr      23 la      29 pt      35 sk-z     41 us
 6 by      12 dvorak  18 hu      24 lt      30 ro      36 slovene  42 wangbe

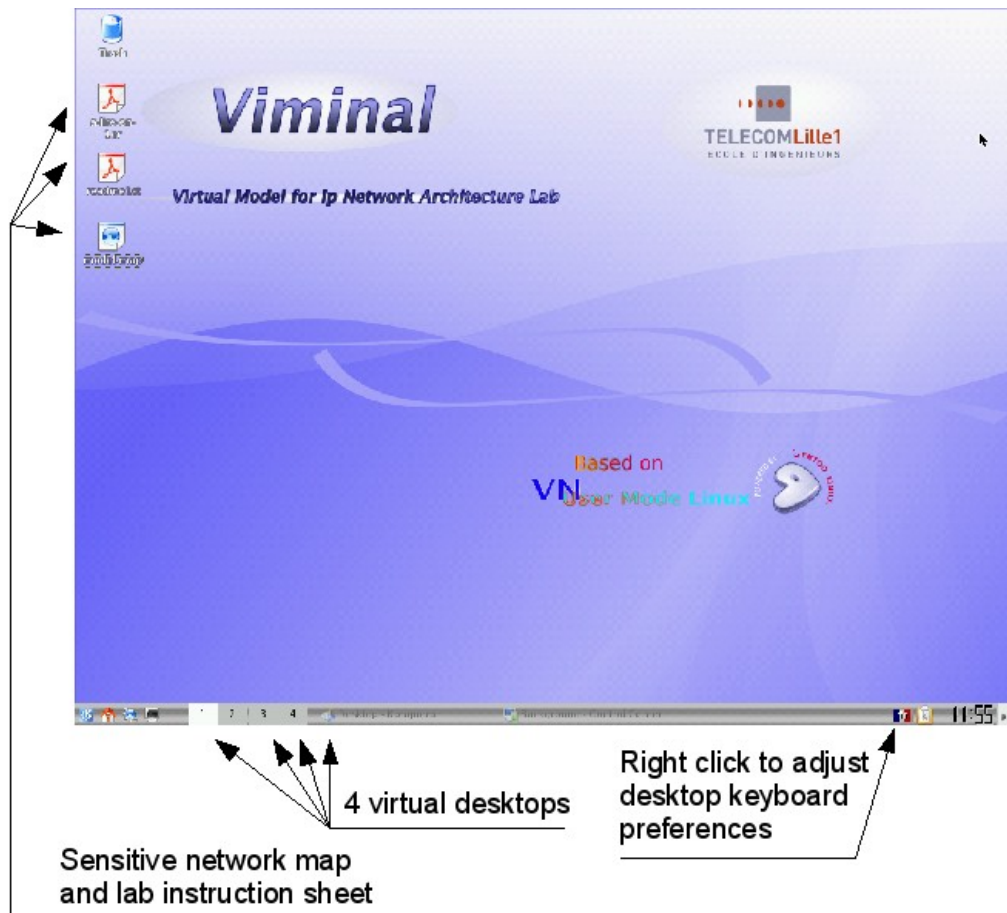
<< Load keymap (Enter for default): fr_

```

Playing the lab

The VIMINAL desktop

The liveCD automatically logs the viminal user and starts a graphical desktop session (KDE desktop). At the end of the boot step you must have a graphic screen looking like this one :



All what you need to play the lab is directly available on this desktop :

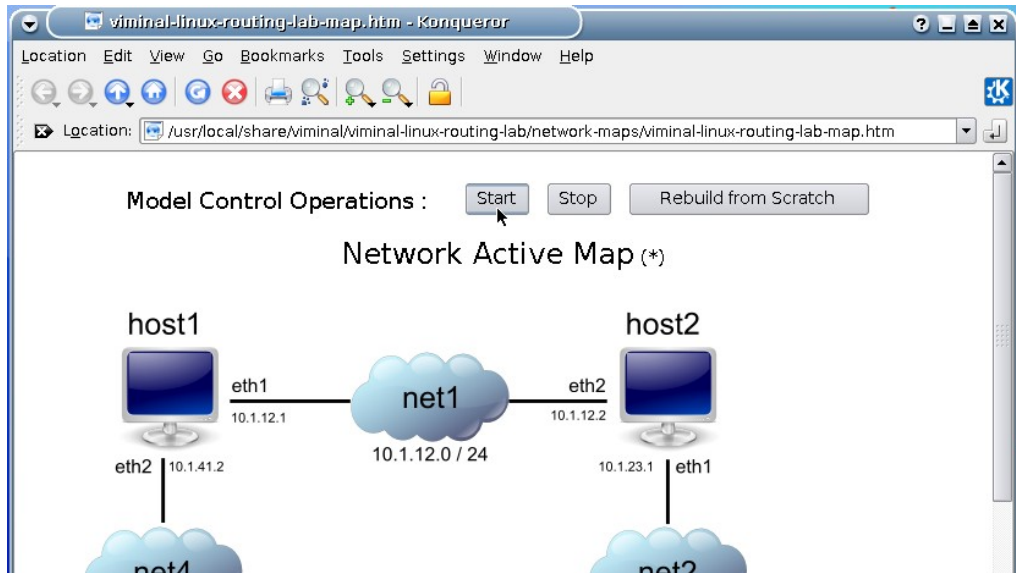
- this read me first instruction file,
- the sensitive network map to interact with the network model (a graphical representation of the network model on which you can do start, stop control operations, and have direct access on the virtual machines of the model),
- the lab instruction sheet to play the lab.

Nota : you can adjust your desktop keyboard preferences by right clicking on the flag in right bottom of the screen (default is french azerty keyboard).

Starting the network model

There's two ways to start the network model :

- The simplest one : just open the sensitive network model map, and click on the “start” button.
- Manually : open a shell terminal (konsole on KDE), and manually launch the `/usr/local/share/viminal/name-of-the-lab/model-control-scripts/name-of-the-lab-start.sh` script

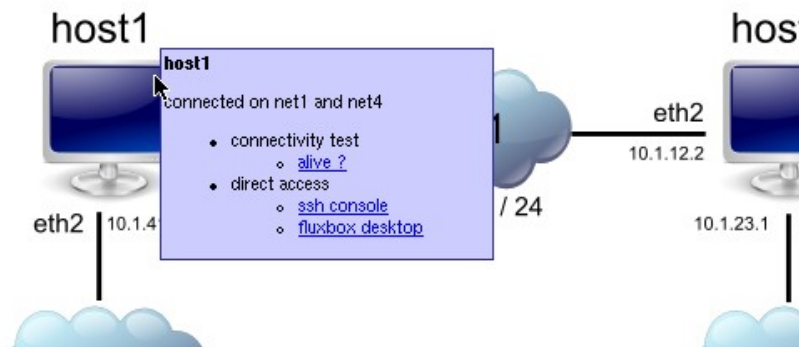


This starts all the networks and virtual machines of the model. It can be quite long (depending of your computer capacity) and open a lot of consoles. Be patient and read the lab instruction sheet during this step.

When the model is up, your graphical desktop displays a set of consoles (for each machine of the model console #0 displays log messages of the virtual machine, and console #1 is a direct login console), which can be ugly for the usability. You can switch on another virtual desktop (remember you have 4 virtual desktops on KDE, so you can switch between them by a simple mouse click in the taskbar). Or you can minimize all these consoles by right clicking on the console reference in the taskbar and then select "Minimize All".

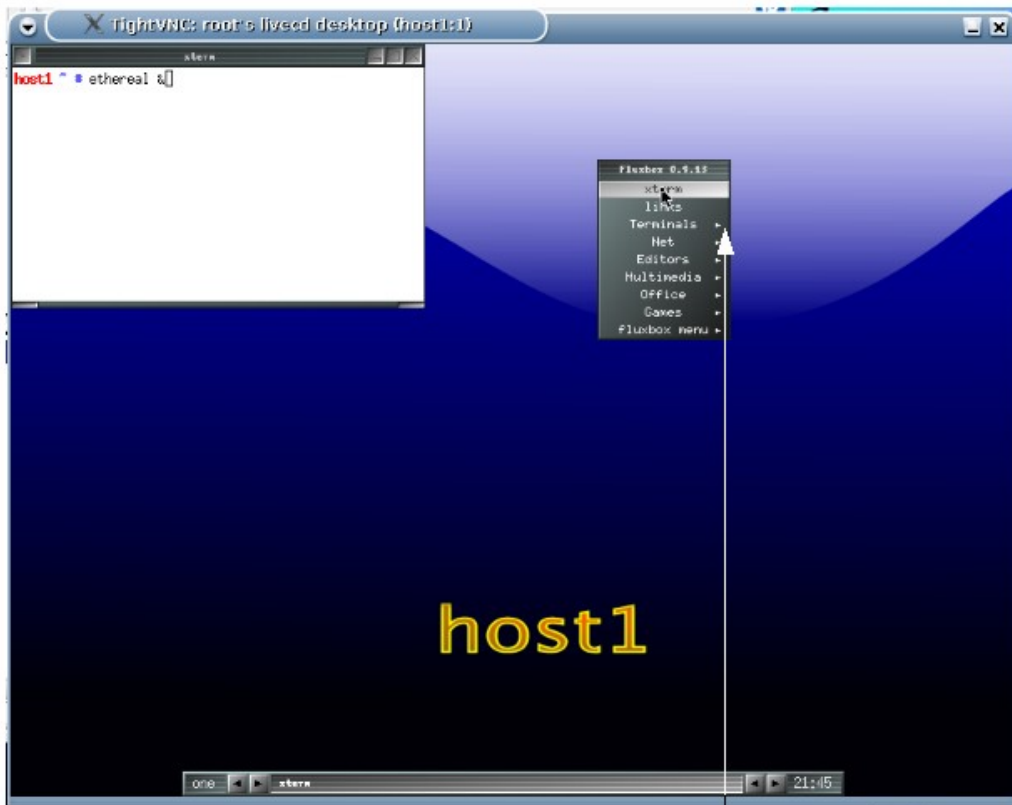
In the sensitive network map, a mouse rollover on each host of the model open a small menu which allows a ping test (alive ?), a ssh access (on first access you have to respond yes to generate ssh keys) and a graphical Fluxbox desktop access.

Network Active Map (*)



On a Fluxbox desktop, a right button click on the mouse gives you an access to a set of tools with super-user rights.

Host1 fluxbox desktop



Mouse right click

Nota : In some circumstances, on one or several virtual machines, this Fluxbox desktop is not fully operational. In that case right clic access to the set of tools does not work. To restart the Fluxbox desktop just proceed as follows. Close the invalid Fluxbox desktop window. Using the sensitive map open a ssh console on that virtual machine and then restart X-Window and Fluxbox environments chaining the following commands in that order.

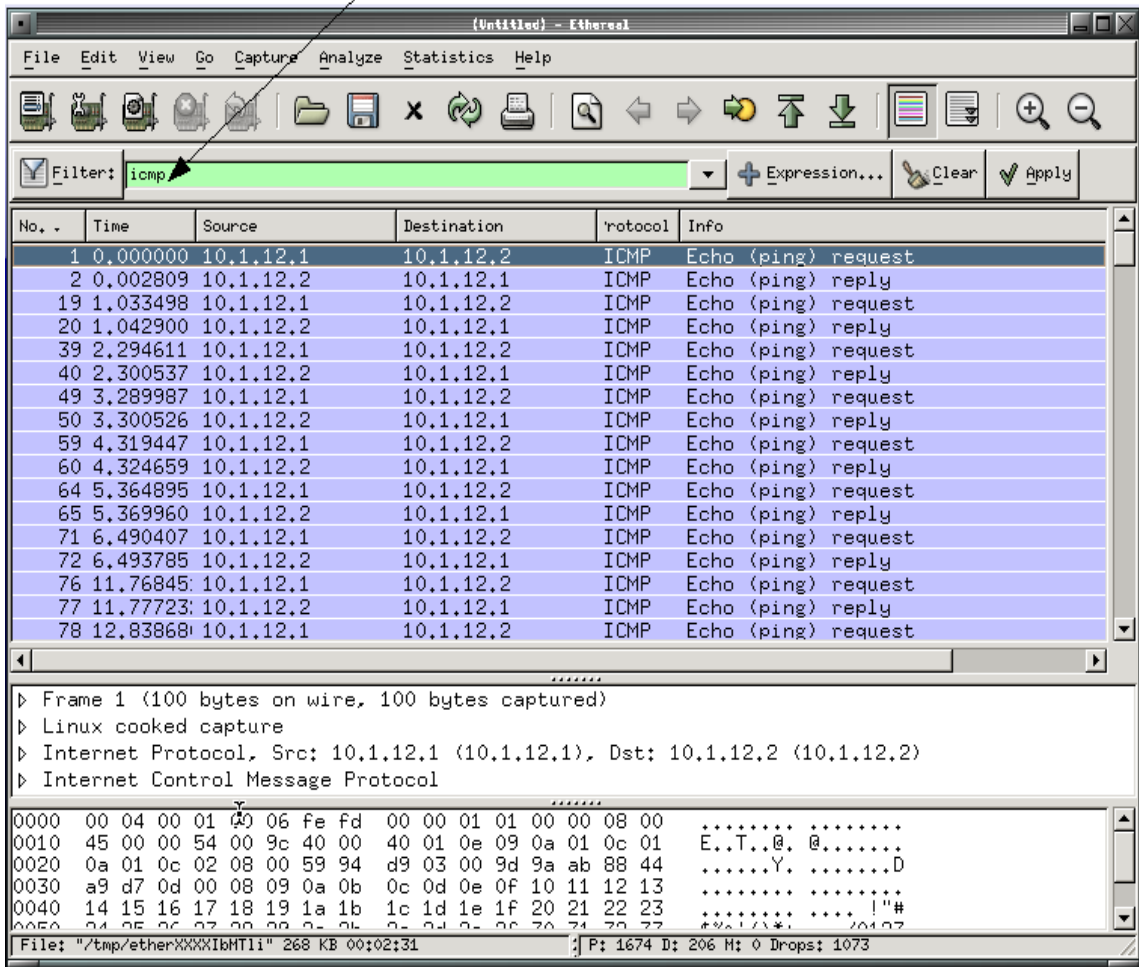
```
~# /etc/init.d/vm-fluxbox stop
~# /etc/init.d/vm-xvnc restart
~# /etc/init.d/vm-fluxbox start
```

The machine then displays a set of messages before giving back the shell prompt (~#). Then you can close the ssh terminal, and open the Fluxbox desktop using the sensitive map.

Capturing network traffic

Each virtual machine of the model, embeds Wireshark, the great network traffic capture tool. As you have super-user rights on, you can observe traffic on all interfaces of the virtual machine. To launch Wireshark, just type 'wireshark &' at the shell prompt in an xterm. In the 'Capture' menu, launch traffic capture on all ethernet interfaces. Let capture run several seconds, then stop. Wireshark will then display all captured datagrams on each ethernet interface of the virtual machine. To make reading easier, filter the displayed trace by applying a display filter in the filter field (icmp display filter in the example bellow).

Wireshark display filter



-----oOo-----

VIMINAL
mobidik lab

(**M**odel to **O**bserve **B**asic Infrastrucuture **D**oing **I**dentification with **K**erberos)

Linux mobidik lab
Model to **O**bserve **B**asic Infrastructure **D**oing **I**dentification with **K**erberos

Préambule : Content of this lab is a derived work from an initial one proposed by Mathieu Blondel et Anthony Legrand for a final project at TELECOM Lille1. Jacques Landru made a complement and an adaptation for VIMINAL (Virtual Model for Ip Network Achitecture Lab) platform. Several documents were used to build this lab like MIT Kerberos 5 Release 1.6.3 documentation and an [HOWTO : Kerberos for small networks, without LDAP or AD] from Gentoo forum. A copy of these elements is joined to this document :

- Initial Mr. Blondel and Mr. Legand proposition
[./tp-blondel-legrand/fr/kerberos.html](#)
- MIT Kerberos 5 reference documents
Introduction
[./MIT-docs/MIT-krb5-1.6-documentation.pdf](#)
Kerberos V5 Installation Guide :
[./MIT-docs/MIT-krb5-1.6-documentation-installation-guide.pdf](#)
Kerberos V5 UNIX User's Guide
[./MIT-docs/MIT-krb5-1.6-documentation-user-s-guide.pdf](#)
Kerberos V5 System Administrator's Guide
[./MIT-docs/MIT-krb5-1.6-documentation-administrator-s-guide.pdf](#)
- Gentoo HOWTO from <http://forums.gentoo.org/viewtopic-t-565180.html>
[./Gentoo-howto/_Gentoo_Forum-View-topic-HOWTO-Kerberos-for-small-networks-without-LDAP-or-AD.pdf](#)

Jacques Landru

Ingénieur d'Etudes Institut TELECOM
TELECOM Lille1
Dept Informatique et Réseaux.

Table des matières

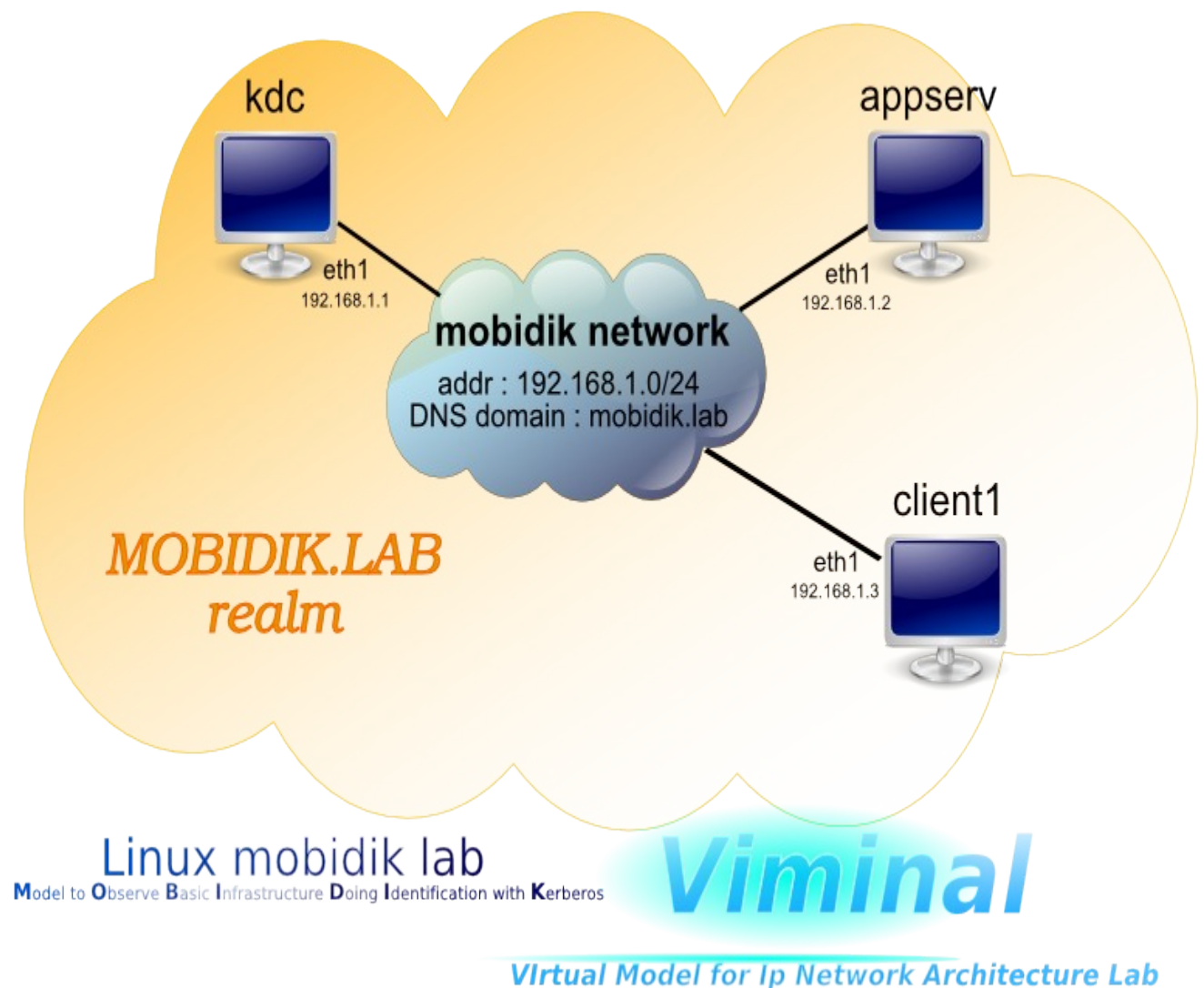
| | |
|---|----|
| 1) Lab context..... | 7 |
| 2) Lab sequence..... | 8 |
| 2.1) Model start..... | 8 |
| 2.2) Which Kerberos ?..... | 8 |
| 2.3) Kerberos ecosystem..... | 8 |
| 2.3.a) NTP (Network Time Protocol)..... | 8 |
| 2.3.b) DNS (Domain Name Service)..... | 8 |
| 2.3.b.1) DNS server configuration..... | 8 |
| 2.3.b.2) Starting DNS service..... | 9 |
| 2.3.b.3) DNS checking..... | 10 |
| 2.4) MOBIDIK.LAB Kerberos realm configuration..... | 11 |
| 2.4.a) MOBIDIK.LAB realm protagonists..... | 11 |
| 2.4.b) KDC configuration..... | 11 |
| 2.4.b.1) /var/lib/krb5kdc/kdc.conf file..... | 12 |
| 2.4.b.2) /var/lib/krb5kdc/kadm5.acl file..... | 12 |
| 2.4.c) MOBIDIK.LAB kerberos client configuration..... | 12 |
| 2.4.d) Initial database creation the Kerberos realm..... | 13 |
| 2.4.e) administrator principals creation..... | 14 |
| 2.4.f) Bunch of keys creation for kadmin service..... | 15 |
| 2.4.g) KDC services startup..... | 16 |
| 2.4.h) User account creation..... | 16 |
| 2.5) Kerberos use..... | 17 |
| 2.5.a) Ask a ticket for castafiore..... | 17 |
| 2.5.b) Telnet versus kerberized Telnet | 19 |
| 2.5.b.1) Telnet is prying with your personal informations..... | 19 |
| 2.5.b.2) Kerberized telnet application service..... | 21 |
| 2.5.b.3) Kerberized telnet session, where is the password ?..... | 22 |
| 2.5.b.4) Encrypted kerberized telnet session..... | 24 |
| 2.5.c) Preauthentication activation..... | 24 |
| 2.5.d) Kerberos, on time !..... | 25 |
| 3) To go further..... | 26 |

VIMINAL
mobidik lab

(**M**odel to **O**bserve **B**asic Infrastructure **D**oing Identification with **K**erberos)

1) Lab context

This lab is based on VIMINAL (**V**irtual **M**odel for **I**p **N**etwork **A**rchituectue **L**ab). Mobidik (**M**odel to **O**bserve **B**asic Infrastructure **D**oing Identification with **K**erberos) model set up three virtual machines (kdc, appsrv, and client1) interconnected on 192.168.1.0/24 mobidik lan. The lab goal is to configure MOBIDIK.LAB Kerberos realm and observe, using wireshark network analyzer, traffic between the different protagonists during Kerberos authentication. The picture below show the MOBIDIK.LAB realm, the mobidik LAN ant its address space.



2) Lab sequence

2.1) Model start

To learn how to use VIMINAL, see “readme-1st” document on the computer desktop. Open the sensitive model map, and click on the start button. While model is starting, which can be long depending of you computer power, you can read the “readme-1st” document.

2.2) Which Kerberos ?

Several editors develop Kerberos implementations. About free software, there 're two references :

- Heimdal version developed by Sweden,
- MIT (Massachusetts Institute of Technology) historical version. That's this version we use for Mobidik lab.

Kerberos ecosystem

2.2.a) NTP (Network Time Protocol)

Kerberos is time shifting sensitive, maximal toleration is about five minutes. Fine clock synchronization, between realm machines is needed, due to Kerberos client authenticity checking challenge. NTP server can be used to synchronize network computers. Such NTP server is not needed on VIMINAL, as the clock of all the virtual machines derives from the host clock. To limit the number of virtual machines, we haven't installed NTP server on Mobidik model.

2.2.b) DNS (Domain Name Service)

Kerberos make heavy usage of the DNS (Domain Name Service). As Kerberos v5 principal names are based on the Fully Qualified Domain Name (FQDN), each operation has to resolve hostname. DNS must be capable of forward and reverse lookups. A good healthy naming service is essential for a Kerberos realm. An erroneous configuration or a naming service problem can bring about Kerberos failure, which can be difficult to solve as log entries are sometimes not clear.

To limit number of virtual machines of the Mobidik model, DNS service will be held on kdc.mobidik.lab (address is 192.168.1.1).

2.2.b.1) DNS server configuration

On the sensitive map open a command xterm on the KDC machine (left button on the fluxbox

desktop), or a ssh terminal.

DNS service is held on KDC machine by the Bind/Named daemon. Verify the conformity between the DNS configuration with the address plan of Mobidik lan.

Nota : On VIMINAL platform you have “ cat “, “ more “ or “ less “ commands to display file contents, and “ nano “, “ vi “ or “ vim “ to modify files. To leave “ less “ display, just type ” q “ (which means “ quit “).

All configuration files of named daemon are in /etc/bind. Using “ cat “ (or “ more “ or “ less “) command verify content of the main DNS config file /etc/bind/named.conf.

```
kdc~# less /etc/bind/named.conf
```

On which zones, is the server authoritative ?

What are mobidik.lab and 1.168.192.in-addr.arpa zones ?

In which files are configured these zones ?

Verify conformity of these zone files with the address plan of the Mobidik lan map.

```
kdc~# less /etc/bind/pri/mobidik.lab.zone
```

```
kdc~# less /etc/bind/pri/1.168.192.in-addr.arpa.zone
```

Nota : Each character of the file has a syntax meaning. For example, a final dot on a DNS name means an absolute DNS name. Final dot points then to the DNS root. If that dot is erroneously missing (or present), it can make the name service erratic. See that final dot on entries of 1.168.192.in-addr.arpa table.

2.2.b.2) Starting DNS service

DNS service daemon is called “ named “. “ named “ system user, which has daemon control, must have the “ write ” permissions on the /var/run/named directory. Before starting the service, give the directory ownership to the named user, using the “ chown “ (change owner) command.

```
kdc~# chown named /var/run/named
```

A startup script, in /etc/init.d directoy, makes some control before starting the service. Launch the service with “ /etc/init.d/named start ” command.

```
kdc~# /etc/init.d/named start
```

Virtual machine then displays.

```
starting Named [ok]
```

Verify all is OK, displaying the tail of the system log.

```
kdc~# tail /var/log/messages
```

An error message means a misconfiguration, you have to correct before going further.

To start at boot up, DNS service has to be integrated in the startup sequence of the machine. On Gentoo distribution, to automatically start a service you just have to add the corresponding service init script in the default runlevel with the rc-update command.

```
kdc~# rc-update add named default
```

You can verify the bootup sequence with the command.

```
kdc~# rc-update -s
```

2.2.b.3) DNS checking

On each machine of the model, you need to verify DNS client configuration. A name lookup must return the Fully Qualified Domain Name, and reverse lookup must be OK.

On the sensitive map, open a command xterm, clicking the left button on the fluxbox desktop, or just open a ssh terminal.

Check DNS resolver configuration of your IP stack.

```
kdc~# cat /etc/resolv.conf
```

Search domain has to be set to "mobidik.lab". Primary server address has to be 192.168.1.1.

Check name lookup for each hostname on the Mobdik lan.

```
kdc~# nslookup kdc
kdc~# nslookup kdc.mobidik.lab
kdc~# nslookup client1.mobidik.lab
kdc~# nslookup appsrv.mobidik.lab
```

Check reverse lookups.

```
kdc~# nslookup 192.168.1.1
kdc~# nslookup 192.168.1.2
kdc~# nslookup 192.168.1.3
```

“hostname -f” command has to display the FQDN of the machine.

```
kdc~# hostname -f
```

Don't go further if all these checks are not successful.

2.3) MOBIDIK.LAB Kerberos realm configuration

2.3.a) MOBIDIK.LAB realm protagonists

For this lab we'll create and manage MOBIDIK.LAB Kerberos realm, bound to “mobidik.lab” DNS domain. As said in Kerberos fundamentals the Key Distribution Center (KDC) is the central repository of a Kerberos realm. This donjon is the depository or the guardian of the keys of all the subjects of the kingdom. It's the trusty element on which depends all the strength of the realm. Its configuration must be carefully designed. The kdc.mobidik.lab machine (address 192.168.1.1) will have this role. As the MOBIDIK.LAB realm administrator, you'll have the responsibility to set it up.

Kerberos realm configuration has two parts :

- KDC server configuration and management,
- Kerberos clients configuration : “kerberized” application servers and their clients. For Mobidik lab we'll setup one “kerberized” telnet application service on appsrv.mobidik.lab machine.

KDC machine (kdc.mobidik.lab) has to be set up with the two roles. Other machines of the model (appsrv.mobidik.lab holding ktelnet kerberized service and client1.mobidik.lab) are kerberos client only.

2.3.b) KDC configuration

Kerberos has two server daemons :

- one for the KDC (as AS et TGS functions, see Kerberos fundamentals), it's /usr/sbin/krb5kdc program.
- The second for KDC keys database management, called kadmind and held by /usr/sbin/kadmind program.

/var/lib/krb5kdc/kdc.conf file

These service daemons are set up with parameters in `/var/lib/krb5kdc/kdc.conf` file.

Nota : The file location can vary, depending of the GNU/Linux distribution or the different BSD/unix flavors. An environment variable `KRB5_KDC_PROFILE=/path/to/kdc.conf` can be used before starting daemons.

On `kdc.mobidik.lab` machine, consult `/var/lib/krb5kdc/kdc.conf` file.

```
kdc~# less /var/lib/krb5kdc/kdc.conf
```

Using “`man kdc.conf`” command, the installation guide, </usr/local/share/viminal/viminal-mobidik-lab/instruction-sheets/MIT-docs/MIT-krb5-1.6-documentation-installation-guide.pdf>, give an interpretation of each different rubrics and parameters of this file for MOBIDIK.LAB realm.

2.3.b.1) /var/lib/krb5kdc/kadm5.acl file

KDC database management rights are controlled by an access control list (ACL) in the `/var/lib/krb5kdc/kadm5.acl` file. Consult this file content.

```
kdc~# less /var/lib/krb5kdc/kadm5.acl
```

Using the installation guide, </usr/local/share/viminal/viminal-mobidik-lab/instruction-sheets/MIT-docs/MIT-krb5-1.6-documentation-installation-guide.pdf>, give your interpretation of the content of this file.

Who (principal names) have KDC management rights for MOBIDIK.LAB realm ?

2.3.c) MOBIDIK.LAB kerberos client configuration

Each MOBIDIK.LAB realm machine (even KDC machine) has to be configured as Kerberos client, setting up `/etc/krb5.conf` file.

On each machine of Mobidik lan, consult `/etc/krb5.conf` file content.

```
kdc~# less /etc/krb5.conf
```

Using “`man krb5.conf`” command and the installation guide, </usr/local/share/viminal/viminal-mobidik-lab/instruction-sheets/MIT-docs/MIT-krb5-1.6-documentation-installation-guide.pdf>, give your interpretation of the content of this file.

2.3.d) Initial database creation the Kerberos realm

KDC database creation is done only once, when the realm is funded. The `kdb5_util` command is used for low level database management. It is also used to create the database. A set of sensitive files, located in `/var/lib/krb5kdc` directory will be created.

Display the directory content before initializing the database.

```
kdc~# ls -al /var/lib/krb5kdc
```

During the database creation a master password will be asked. This master key, can be saved in a stash file. From the installation guide, we can learn that : “ The stash file is a local copy of the master key that resides in encrypted form on the KDC's local disk. The stash file is used to authenticate the KDC to itself automatically before starting the “ kadmind “ and “ krb5kdc “ daemons (e.g. as part of the machine's boot sequence). The stash file, like the keytab file, is potential point-of-entry for a break-in, and if compromised, would allow unrestricted access to the Kerberos database. If you choose to install a stash file, it should be readable only by root, and should exist only on the KDC's local disk. The file should not be part of any backup of the machine unless access to the backup data is secured as tightly as access to the master password itself”. A copy of this stash file would be securely put in a safe place. “If you choose not to install a stash file, the KDC will prompt you for the master key each time it starts up”. This means that the KDC will not be able to start automatically, such as after a system reboot. The trust we put in the KDC depends on the precautions we took to protect the master key.

For Mobidik lab, we choose “ `m0b1d1kim121k` “ as password value for the master key. The key can be any string. A good key is one you can remember, but that no one else can guess, even when using a dictionary attack. A good key would mix upper and lower case letters and digits. In Mobidik example, we start from the sentence “ **mobidik is my lab to learn kerberos** “. We use the first letter of each word, and for mobdik acronym we replace “o” letter with zero digit, “i” letter with one digit. We also replace the “to” word with the two digit. This value will be asked twice during database creation.

To initialize the database just type “ `kdb5_util create -r MOBIDIK.LAB -s` “ command. Note the “ `s` “ parameter to create the stash file.

```
kdc~# kdb5_util create -r MOBIDIK.LAB -s
Loading random data
Initializing database '/var/lib/krb5kdc/principal' for realm
'MOBIDIK.LAB',
master key name 'K/MOBIDIK.LAB'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:
Re-enter KDC database master key to verify:
kdc~#
```

Display again the `/var/lib/krb5kdc` directory. Note that, only the root has read/write rights on the newly created files.

```
kdc~# ls -al /var/lib/krb5kdc
```

2.3.e) administrator principals creation

This step is, also, done once, when the realm is funded. You need to add administrative principals to the Kerberos database. You must add at least one. By convention, an administrative principal has the instance attribute value set to " admin ".As seen before, ACL file has to specify which principals have the administrative rights on the KDC's database.

As an allusion of the famous summer TV game show, I mean Fort Boyard (see : [http://en.wikipedia.org/wiki/Fort_Boyard_\(TV_series\)](http://en.wikipedia.org/wiki/Fort_Boyard_(TV_series)) and http://en.wikipedia.org/wiki/Fort_Boyard) , we'll create user "fouras", to pay homage to father Fouras which is the master of the keys in this TV game. We'll create two principals for that user, one with the " admin " instance attribute set for administrative role, in conformance with the `kadm5.acl` rules, and another one without the instance attribute for a daily usage as every user.

For the fouras user we'll choose the following passwords

- for [fouras@MOBIDIK.LAB](#) principal : « fFitrs0fB » « father Fouras is the riddle spécialiste of fort Boyard ». replace "o" letter with zero digit.
- for [fouras/admin@MOBIDIK.LAB](#) principal : « fFitm0tk! » « father Fouras is the master of the keys ! ». replace "o" letter with zero digit, don't forget the exclamation point to harden the password.

As `kadmind` is not yet started, we'll use the `kadmin.local` utility. This tool is used to directly manage the local database on the KDC. The command prompt " `kadmin.local` " replaces the bash prompt. As for that latter the tabulation key can be used for command completion. The " ? " displays all the available commands.

Before creating the new principals create a default policy. A policy is a name group of restrictions on passwords like their minimum length and maximum lifetime. Each principal is assigned a policy when it is created. If no explicit policy is given during principal creation, the policy " default " is assigned. In Mobdikik lab the password will have ten years validity by default.

```
kdc~# kadmin.local
authenticating as principal root/admin@MOBIDIK.LAB with password
kadmin.local: add_policy -maxlife "10 years" default
kadmin.local: add_principal fouras
NOTICE: no policy specified for fouras@MOBIDIK.LAB; assigning
"default"
Enter password for principal "fouras@MOBIDIK.LAB":
Re-enter password for principal "fouras@MOBIDIK.LAB":
Principal."fouras@MOBIDIK.LAB" created
kadmin.local: add_principal fouras/admin
NOTICE: no policy specified for fouras/admin@MOBIDIK.LAB; assigning
"default"
Enter password for principal "fouras/admin@MOBIDIK.LAB":
Re-enter password for principal "fouras/admin@MOBIDIK.LAB":
Principal."fouras/admin@MOBIDIK.LAB" created
kadmin.local:
```

List all the principals in the KDC's database using `list_principals` command of `kadmin.local.tool`.

```
kadmin.local: list_principals
fouras@MOBIDIK.LAB
fouras/admin@MOBIDIK.LAB
K/M@MOBIDIK.LAB
kadmin/admin@MOBIDIK.LAB
kadmin/changepw@MOBIDIK.LAB
kadmin/history@MOBIDIK.LAB
krbgt@MOBIDIK.LAB
kadmin.local: quit
kdc~#
```

[K/M@MOBIDIK.LAB](#), [kadmin/admin@MOBIDIK.LAB](#), [kadmin/changepw@MOBIDIK.LAB](#), [kadmin/history@MOBIDIK.LAB](#), [krbgt@MOBIDIK.LAB](#) principals were created by default during KDC database creation.

2.3.f) Bunch of keys creation for kadmin service

Kerberized application services run in background as daemons. Such program can't have a password, which will be derived as a key. It's not possible to type the password each time the service start at server boot up. Each Kerberos application server needs local on-disk-copy of its own host's key. This copy is stored in a file named `krb5.keytab` located in `/etc` directory. It's, again, a sensitive file with rights restricted to the superuser (root), or the system user dedicated to the service daemon. File `/etc/krb5.keytab` can be seen as a the bunch of keys for the application services hosted on the machine. " Like the stash file, this file should not be part of any backup of the machine, unless access to the backup data is secured as tightly as access to the machine's root password itself ".

Kadmin is KDC database remote management service. When activated, realm administrators can remotely create or update the database entries, users can remotely change their password from every machine of the realm. Kadmin service is the first kerberized application service launched on the realm. It resides on the KDC. We have to create its bunch of keys. " `ktadd` " command of " `kadmin.local` " utility is used to generate or complete the `keytab` file. For the kadmin service, we'll have to generate the `/etc/krb5.keytab` for `kadmin/admin` and `kadmin/changepw` principals.

```
kdc~# kadmin.local
authenticating as principal root/admin@MOBIDIK.LAB with password
kadmin.local : ktadd kadmin/admin kadmin/changepw
Entry for principal kadmin/admin with kvno 3, encryption type Triple
DES cbc mode with HMAC/sha1 added to WRFILE:/etc/krb5.keytab
Entry for principal kadmin/admin with kvno 3, encryption type DES cbc
mode with HMAC/sha1 added to WRFILE:/etc/krb5.keytab
Entry for principal kadmin/changepw with kvno 3, encryption type
Triple DES cbc mode with HMAC/sha1 added to WRFILE:/etc/krb5.keytab
Entry for principal kadmin/changpw with kvno 3, encryption type DES
cbc mode with HMAC/sha1 added to WRFILE:/etc/krb5.keytab
kadmin.local quit
```

Check that keytab file is created in `/etc` directory.

```
kdc~# ls -al /etc/krb5*
```

What are this keytab files rights settings ?

2.3.g) KDC services startup

We can now start KDC services. Startup scripts have been installed in the `/etc/init.d` directory by the package manager.

Nota : Names an localization of those scripts can vary, depending of the GNU/Linux distribution or the different BSD/unix flavors.

Start `kdc` and `kadmin` services.

```
kdc~# /etc/init.d/mit-krb5kdc start
* Starting MIT Kerberos 5 KDC . . . [OK]
kdc~# /etc/init.d/mit-krb5kadmind start
* Starting MIT Kerberos 5 Admin daemon . . . [OK]
```

Have a look to the end of the system log file to check operation success.

```
kdc~# tail -20 /var/log/messages
```

MOBIDK.LAB realm's KDC is now operational !

2.3.h) User account creation

Let's create our first user account. For kerberized application service tests we create "castafiore" user (see http://en.wikipedia.org/wiki/Bianca_Castafiore for Castafiore biography). Its password will be "fFshiiP" from the mnemonic sentence "father Fouras's hairdresser is in prison"

Nota : A french joke, when your hair is a mess. You didn't see your hairdresser for a long time as he is in jail. :o)).

Using "kadmin.local" create castafiore user's principal.

```
kdc~# kadmin.local
authenticating as principal root/admin@MOBIDIK.LAB with password
kadmin.local : add_principal castafiore
NOTICE: no policy specified for castafiore@MOBIDIK.LAB; assigning
"default"
Enter password for principal "castafiore@MOBIDIK.LAB":
Re-enter password for principal "castafiore@MOBIDIK.LAB":
Principal "castafiore@MOBIDIK.LAB" created.
Kadmin.local: quit
kdc~#
```

2.4) Kerberos use

2.4.a) Ask a ticket for castafiore

From every host of MOBIDIK.LAB realm, you can now get an initial ticket (TGT Ticket Granting Ticket) or a service ticket. A set of Kerberos client tools (“`kinit`”, “`klist`”, “`kdestroy`” ...) can be used to manage tickets.

Using the sensitive map, on the client1 machine open a command xterm, clicking the left button on the fluxbox desktop, or just open a ssh terminal.

“`kinit`” command is used to get an initial ticket called TGT (Ticket Granting Ticket). It's the “open sesame” for the realm access. Once authenticated, with this TGT you can get service ticket for kerberized application service access. “`kinit`” prompts for user password. Only the right password can be used to decrypt the message containing the ticket. Once the message decrypted, the TGT and its associated key is stored in `/tmp/krb5cc_xxx` local cache file.

As the client config file points to MOBIDIK.LAB, realm parameter in the “`kinit`” command is facultative. So «`kinit castafiore`» ou «`kinit castafiore@MOBIDIK.LAB`» are equivalent.

```
client1~# kinit castafiore
Password for castafiore@MOBIDIK.LAB:
client1~#
```

No error message display means success.

“`klist`” can be used to list the Kerberos tickets held in a credentials cache or the keys held in a keytab file.

```
client1~# klist
```

What is the content of the credentials cache ?

Which validity has the ticket ?

What is the ticket principal ?

For which service is this ticket needed ? (see Kerberos lecture) ?

Type “`man klist`” command, to find which parameters can be used to know more about this ticket

What are the ticket flags ? Is this ticket forwardable ? renewable ?

“`kdestroy`” command destroys the user's active Kerberos authorization tickets by writing zeros to the credentials cache.

Nota : Your tickets are proof you are indeed yourself, and tickets can be stolen. If this happens, the person who has them can masquerade as you until they expire. For this reason, MIT recommends that users explicitly destroy before logging out. One way to help ensure that this happens is to add the `kdestroy` command to your `.logout` script file. This script file is systematically run when user close his session.

Empty your credentials cache. Check the file is deleted.

```
client1~# ls -al /tmp
.
.
.
client1~# kdestroy
client1~# ls -al /tmp
.
.
.
client1~# klist
klist: No credentials cache found (ticket cache FILE:/tmp/krb5cc_0)
client1~#
```

Let's observe Kerberos exchanges during the “`kinit`” process. On `client1.mobidik.lab` machine “`castafiore`” user ask again a TGT from the KDC. Before starting the `kinit` process, start a wireshark capture on **eth1** interface.

Attention : Don't launch capture on the pseudo “any” interface. Too many ethernet frames will be caught making capture readability difficult.

```
client1~# wireshark &
client1~#
```

On wireshark interface, to start a capture chain the following menus « captures → interfaces → eth1 → start »

Open a new xterm window and ask for a new TGT.

```
client1~# kinit castafiore
Password for castafiore@MOBIDIK.LAB:
client1~#
```

Stop the wireshark capture (menu interfaces → stop, ou click on red icon with white cross). Observe traffic between `client1.mobidik.lab` and `kdc.mobidik.lab` machines.

What are the two Kerberos message types ?

What is their content ?

Unfold message headers.

In AS-REQ message :

- *which options are requested ?*
- *what is the client principal value ?*
- *Which service is requested ?*
- *What is the validity period ?*

In AS-REP message :

- *what is the ticket version number ?*
- *which cryptographic algorithm is used for that ticket ?*

Empty again your credentials cache with “ `kdestroy` ” command before going further.

```
client1~# kdestroy
```

2.4.b) Telnet versus kerberized Telnet

To show advantages of Kerberos, we are going to compare a legacy telnet version versus its kerberized version.

The remote terminal service, known as telnet, allows command session on remote machine through the network. It was a very popular service but it has, nowadays, been replaced by a more secure tool, I mean ssh (Secure SHell). The main drawback of telnet service is all information are clearly transmitted over the network. SSH use a secure transport layer to offer a encrypted tunnel through the network. We'll see that the kerberized version of telnet can also offer a secure remote session, almost as good as ssh.

2.4.b.1) Telnet is prying with your personal informations

In this step we are opening a classical telnet session (unkerberized), which will be caught by a wireshark capture.

Until now, mobidik network user accounts are not unified on a central network repository like a NIS or LDAP directory. Setting up such a directory is out of the scope of this lab. We are just going to replicate accounts on the server (appsrv.mobidik.lab) local account database (`/etc/passwd` and `/etc/shadow`) for the users already created on the KDC.

Using the sensitive map, on the appsrv.mobidik.lab machine open a command xterm, clicking the left button on the fluxbox desktop, or just open a ssh terminal.

Create a local account for castafiore user.

```
appsrv~# useradd -c "Bianca Castafiore" -d /home/castafiore -m -p  
"fFshiip" -s /bin/bash castafiore
```

Check account and its home directory are successfully created, castafiore user can open a session on appsrv.mobidik.lab machine.

```
appsrv~# su -l castafiore
castafiore@appsrv~$ whoami
castafiore
castafiore@appsrv~$ pwd
/home/castafiore
castafiore@appsrv~$ exit
appsrv~#
```

Telnet service is usually automatically activated using xinetd super-daemon. In this lab, to observe its behavior, we'll manually launch it in debug mode each time we need it.

On appsrv.mobidik.lab machine, open a command terminal and launch `ktelnetd` daemon in debug mode using “`ktelnetd -debug -a debug`” command.

The service is then waiting an incoming connection, it will stop as soon as the session will be closed.

```
appsrv~# ktelnetd -debug -a debug
.
.
.
```

Attention : `ktelnetd` the final “d” is important, it means server daemon program, whereas `ktelnet` is the client program.

Let's go now on the telnet client side. On client1.mobidik.lab, castafiore user will open a remote session on appsrv.mobidik.lab machine. Before starting that remote session, start a wireshark capture on **eth1** interface.

Attention : Don't launch capture on the pseudo “any” interface. Too many ethernet frames will be caught making capture readability difficult.

```
client1~# wireshark &
client1~#
```

On wireshark interface, to start a capture chain the following menus « captures → interfaces → eth1 → start »

Open a new command terminal and launch telnet client in unkerberized mode.

```
client1~# ktelnet appsrv.mobidik.lab
.
.
.
```

At login prompt, respond castafiore, type her password. Once connected on remote host, chain this command sequence.

```
castafiore@appsrv~$ whoami
castafiore
castafiore@appsrv~$ pwd
/home/castafiore
castafiore@appsrv~$ ls -al
.
.
.
castafiore@appsrv~$ exit
Connection closed by foreign host.
client1~#
```

Stop the wireshark capture (menu interfaces → stop, or click red icon with white cross). Observe traffic between client1.mobidik.lab and kdc.mobidik.lab machines.

Apply a display filter on the wireshark capture, to hide ARP and DNS traffic. In the filter field type the following filter “ !dns and !arp ”. You'll then only see the tcp/telnet traffic.

Select one of the telnet datagram and right click to choose “ follow TCP Stream “ option. In a new window, you'll see all your telnet session built up again.

Note the inquisitiveness of the whireshark capture tool. All you user session has been watched, password is in clear texte. Is this acceptable ?

On appsrv.mobidik.lab, telnet session is now closed. Note that the daemon is stopped.

2.4.b.2) Kerberized telnet application service

Every kerberized application service needs an entry in the KDC's database as a service principal with its associated key. Principal name is “ host “, instance attribute is the FQDN of the host. So in our case [host/appsrv.mobidik.lab@MOBIDIK.LAB](#). The key, attached to this principal, will be exported in the keytab file of the machine appsrv.mobidik.lab on which the application service is hosted. “ kadmin “ utility will be used, as previously said, to manage remotely the KDC's database. KDC access will be done with realm administrator credentials.

The host principal will be created, asking KDC to generate a random key. Since this principal will be added to a keytab, the password will never be used by a human. The “ -randkey “ option of the “ add_principal “ command randomize the key. This key will be then exported to the application server's bunch of keys (the keytab fille), using the “ ktadd “ command.

Nota : There's an alternative method, which can be used but needs some precautions. Instead of using the remote admin tool (kadmin), you can generate the principal and its random key directly on the KDC machine using the kadmin.local tool. You can then to export the key in a temporary KDC local file with the “ -k /path/to/temporary.keytab “ parameter on the “ ktadd “ command. You need to safely transfer the keytab file on the application server using a secure channel (scp/ssh for example). Don't forget to delete the sensitive temporary keytab file on the KDC.

Using the sensitive map, on the appsrv.mobidik.lab machine open a command xterm, clicking the left button on the fluxbox desktop, or just open a ssh terminal.

```
appsrv~# kadmin -p fouras/admin
Authentication as principal fouras/admin with password.
Password for fouras/admin@MOBIDIK.LAB
kadmin: add_principal -randkey host/appsrv.mobidik.lab
NOTICE: no policy specified for host/appsrv.mobidik.lab@MOBIDIK.LAB;
assigning "default"
Principal "host/appsrv.mobidik.lab" created.
Kadmin: ktadd host/appsrv.mobidik.lab
Entry for principal host/appsrv.mobidik.lab with kvno 3, encryption
type Triple DES cbc mode with HMAC/sha1 added to
WRFILE:/etc/krb5.keytab
Entry for principal host/appsrv.mobidik.lab with kvno 3, encryption
type DES cbc mode with HMAC/sha1 added to WRFILE:/etc/krb5.keytab
kadmin: list_principals
.
.
.
kadmin: quit
appsrv~#
```

“list_principals” command of « kadmin » tool is used to verify that the new entry is successfully created. Check also that /etc/krb5.keytab exists.

```
appsrv~# ls -al /etc/krb5*
.
.
.
appsrv~#
```

2.4.b.3) Kerberized telnet session, where is the password ?

We're now going to open a kerberized telnet session. On the appsrv.mobidik.lab machine, restart ktelnetd daemon with the same options that the previous step.

```
appsrv~# ktelnetd -debug -a debug
.
.
.
```

On the client1.mobidik.lab, restart a wireshark traffic capture (without saving the previous capture). Don't forget to clear the display filter (click on clear button)..

We request again an initial ticket (TGT) for castafiore user.

```
client1~# kinit castafiore
Password for castafiore@MOBIDIK.LAB:
client1~# klist
.
.
.
client1~#
```

Launch now the telnet client with Kerberos authentication using the following command : "ktelnet -f -a -l castafiore appsrv.mobidik.lab". " -f " option ask to transfer the tickets on the remote host, " -a " option try an automatic authentication.

```
client1~# ktelnet -f -a -l castafiore appsrv.mobidik.lab
.
.
.
```

Is single sign on (SSO) successful ?

Observe the telnet banner

Once connected on the remote machine, chain the following commands.

```
castafiore@appsrv~$ whoami
castafiore
castafiore@appsrv~$ pwd
/home/castafiore
castafiore@appsrv~$ ls -al
.
.
.
castafiore@appsrv~$ exit
Connection closed by foreign host.
client1~#
```

Stop the wireshark capture (menu interfaces → stop, ou red icon with white cross).

Apply a display filter on the wireshark capture, to hide ARP and DNS traffic. In the filter field type the following filter " !dns and !arp ". You'll then only see the tcp/telnet traffic.

Select one of the telnet datagram and right click to choose " follow TCP Stream " option. In a new window, you'll see all your telnet session built up again.

Note there's no password exchange.

But telnet session is always in clear text as we start the kerberized telnet session without encrypting with the session key.

On appsrv.mobidik.lab, telnet session is now closed, the daemon is stopped. See the displayed debug messages about the kerberized session for castafiore user.

Clear the wireshark display filter (click on the clear button) and apply again the « !arp and !dns » filter.

Locate the service ticket request response exchange (noted KRB5 TGS-REQ and KRB5 TGS-REP)

Unfold headers of these datagrams.

In request message :

- Which is the requested service ?
- What is the nonce value ?
- What are the flag values ?

In TGS response message :

- What is the client principal of the ticket ?

On the client1 machine, what is the content of ticket cache ?

```
client1~# klist -f
.
.
.
client1~#
```

2.4.b.4) Encrypted kerberized telnet session

Replay the operations of the previous step, to open an encrypted kerberized telnet session. You just have to add the “-x” parameter when launching the telnet client. Don't forget to restart a new wireshark capture.

```
client1~# ktelnet -f -x -a -l castafiore appsrv.mobidik.lab
.
```

Stop the capture and observe, using “Follow TCP Stream”, observe that all the session is now encrypted.

2.4.c) Preauthentication activation.

Preauthentication is used to challenge the client for each TGT request. KDC deliver then an initial ticket only to its known users. Without preauthentication, the KDC distributes a TGT to everyone, who asks one, with no verification. So a fake user can try a offline dictionary attack to crack the password, on a ticket requested with the principal of the user he wants to usurp.

Preauthentication can be globally activated on the KDC, in the “/var/lib/krb5kdc.kdc.conf” configuration file, or at the database creation or modification using the “kdb5_util” command. It can also be activated on a per policy or on a per principal basis.

If preauthentication is active on a service principal, a ticket for that service can be delivered only if the client's TGT has the “preauth” flag on.

For Mobidik lab we'll activate preauthentication for castafiore user.

In a command terminal on kdc.mobidik.lab, modify [castafiore@MOBIDIK.LAB](#) principal using “`kadmin.local`” command.

```
kdc~# kadmin.local
authenticating as principal root/admin@MOBIDIK.LAB with password
kadmin.local : modify_principal +requires_preauth castafiore
Principal "castafiore@MOBIDIK.LAB" modified.
Kadmin.local: quit
kdc~#
```

Observe exchanges during kinit process when preauthentication is active. On client1.mobidik.lab user castafiore request again a TGT from the KDC. Before starting the kinit process, start a wireshark capture on **eth1** interface.

Attention : Don't launch capture on the pseudo “any” interface. Too many ethernet frames will be caught making capture readability difficult.

```
client1~# wireshark &
client1~#
```

On wireshark interface, to start a capture chain the following menus « captures → interfaces → eth1 → start »

Open a new command terminal, empty ticket cache using “`kdestroy`” command and request a new TGT.

```
client1~# kdestroy
client1~# kinit castafiore
Password for castafiore@MOBIDIK.LAB:
client1~#
```

Stop the wireshark capture (menu interfaces → stop, or red icon with white cross).

Apply a display filter on the wireshark capture, to hide ARP and DNS traffic. In the filter field type the following filter “`!dns and !arp`”.

Observe the error message in response to the initial TGT request.

What is the challenge submitted to the client ?

What are the new information when client makes its second request ?

2.4.d) Kerberos, on time !

We are going to observe now, what happen in case of time desynchronization. On client1.mobidik.lab, using “`date`” command change clock value for 15 minutes before or after the current date.

Nota : display bug : Your virtual machine display may freeze when you change clock value in the past. To unfreeze, just select again the host display on the sensitive map. This refresh will restore virtual machine display.

Display current date, using “date” command without parameter, then shift date for a quarter (“date MMJJhhmm” command). Launch a new wireshark capture on eth1 interface, empty ticket cache and request a new TGT for castafiore user.

```
client1~# date
.
.
.
client1~# date MMJJhhmm
.
.
.
client1~# wireshark &
client1~# kdestroy
client1~# client1~# kinit castafiore
Password for castafiore@MOBIDIK.LAB:
kinit(v5): Clock skew too great while getting initial credentials
client1~#
```

Stop the wireshark capture (menu interfaces → stop, ou red icon with white cross).

Apply a display filter on the wireshark capture, to hide ARP and DNS traffic. In the filter field type the following filter “!dns and !arp”.

Observe the error message in response to the initial TGT request.

What is the message displayed by the kinit command ?

What is the error code in kerberos message datagram ?

Nota: Strangely, KDC don't care about clock shift when preauthentication is inactive. If you unset preauthentication for castafiore principal using “modify_principal -requires_preauth castafiore” command under kadmin tool, you can get TGT whatever clock drift.

3) To go further...

In the documentation quoted in the preamble, you can find elements to :

- create a secondary (slave) KDC, which can supply the primary (master) KDC when it is unavailable,
- Storing KDC database in a LDAP directory back end,
- manage client configuration using DNS,
- ...

----- oOo -----