Université Grenoble Alpes

IUT1 de Grenoble

Département RESEAUX et TELECOMMUNICATIONS

Professional Bachelor

Computer Networks, Mobility, Security CNMS

Course 25 :
Lab Security

CRYPTOGRAPHY
2023-2024

1 Installation of the platform

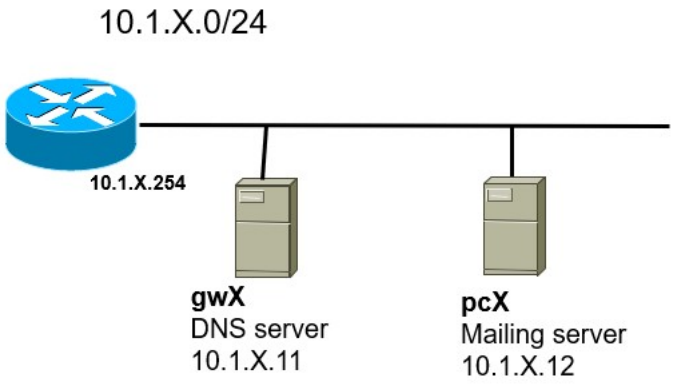
1.1 Description of the platform

For this LAB you have 2 computers using centOS (Community Enterprise OS, fork of Redhat Enterprise Linux (paying version)) in a Virtual Machine VM.

- The left computer manages the DNS of your internal domain zX.rt.iut.
- The computer on the right manages the mailing system.

Before to launch the VM, you need to validate/configure the network cards on VirtualBox. You have to check the network configuration: in order to do so, press "setting" from the top bar, then in adapter 1 select Intel and select bridge mode. When you will start your machine, you will be able to see ensp03(eth0) in network settings, so you will be able to configure the IP address. At first you will use 172.16.0.200 as the DNS server for the installations and preliminary tests.

Diagram of the platform



Network configuration

OS	Role	eth0	eth1	eth2	Hostname	Gateway
Centos 5.7	DNS Server	10.1.X.11/24	non activated	N/A	gwX.zX.rt.iut	10.1.X.254
Centos 5.7	Mailing Server	10.1.X.12/24	non activated	N/A	pcX.zX.rt.iut	10.1.X.254
If needed, LEFT Physical machine		10.1.X.1/24	non activated	N/A		10.1.X.254
If needed, LEFT Physical machine		10.1.X.2/24	non activated	N/A		10.1.X.254

DNS resolution

Machine	DNS primary server	DNS secondary server	Default DNS Suffix
Left machine	127.0.0.1	172.16.0.200	
Right machine	10.1.X.11	N/A	

Configuration of the users (RIGHT MACHINE)

Each student will have a user account on the mailing server (RIGHT MACHINE).
The user names will follow this rule: “2 first letters of the first name + 6 first letters of the family name”.
Ex: Michael Jordan will be “mijordan”.
Arnold Schwarzenegger will be “arschwar”

```
Linux command to create user: sudo adduser username
```

1.2 CentOS installation and wiring

Wiring

The eth0 cards of both machines will be wired on the switch of the student network (10.1.X.0).
The other network cards will not be used.
The 2 machines are installed on Virtual machines.
The configurations of both hosts are described in the following paragraphs.
Configurations files: **this configuration should be done on the graphical interface, top right, use the TAB “Shared” to replace “localhost.localdomain” by your own local host and domain names. You can type “hostname” in the terminal window to check the name and domain of your computer.**
If needed or if there are problems with the graphic interface: the files concerned are the following:

- Hostname: /etc/sysconfig/network (Instead of changing this file, use following command in terminal to change host name: hostnamectl set-hostname new-hostname. In order to check hostname type ‘hostnamectl’ and it will show you your machine name)
- IP address(es): /etc/sysconfig/network-scripts/ifcfg-ethX (Use GUI instead, it is available on top right corner, open network setting)
- DNS resolution: /etc/resolv.conf
- **Default gateway: /etc/sysconfig/network (Use GUI instead, it is available on top right corner, open network setting)**

Edit these files if needed to configure your client and server using data given above.
Use Gedit to edit the necessary files. To open file with gedit type following: `Gedit filename`
Another classical file editor being used is vim (To learn more about it please refer to the appendix “edition of files with vim”).
ATTENTION! Maybe you will have to use the superuser mode when editing (`sudo`).

2 Installation and configuration of the BIND DNS server (LEFT MACHINE)

2.1 Installation of BIND

The installation of BIND is done using the "yum" command.

```
[root@gwX ~] # yum install bind OR [user@gwX ~]# sudo yum install bind
```

If you are having trouble with installation you should try to clean YUM cache with the following command:
[root@gwX ~] # yum clean all

2.2 BIND configuration

The configuration files should be created. For that, you can consult example files in the appendix.
In the configuration file, you will have:

- to define the global options: /etc/named.conf
- to declare the direct zone: "zX.rt.iut" in the adequate file.
- to declare the reverse zone: "X.1.10.in-addr.arpa" in the adequate file.

Configure the BIND server using example files available in the appendix.
Determine the location of the zone files in /etc/named.conf
Before modifying the configuration, it is recommended to save the configuration
#cp /etc/named.conf /etc/named.conf.orig

Launch the DNS server:

```
[root@gwX ~] # service named start
```

The following command will start your service at boot each time
[root@gwX ~] # Service named enable
The following command is used to check the status of named service:

```
[root@gwX ~] # service named status
```

The following command is used to restart named service:

```
[root@gwX ~] # service named restart
```

If service is not starting you should check your configuration files. If you have difficulties, you can also check: `cat /var/log/messages`

2.3 Validation

Once your server has been configured and launched, you should be able:
From the client side: to solve all DNS names (ex: www.google.fr) and to solve the names in the zX.rt.iut zone (ex: pop3.zX.rt.iut or simply pop3). From the server side: to solve the names in the zX.rt.iut zone (ex: pcX.zX.rt.iut or simply pcX)
You are asked to prepare and launch a list of commands allowing you to show the correct functioning: for instance, about query tools for both windows and linux (see the reminder below). You might also be asked to display nameservers for specific domain name, to show full zone transfer or a query for reverse resolution.

Please show to the teacher.

REMINDER if necessary
host *is the simplest one:*
$ host www.rtgrenoble.fr # forward resolution
$ host 152.77.173.246 # reverse resolution
dig *accepts parameters and provides the reply directly in a format reminding a zone file.*
$ dig [@IP_or name_of_DNS_server] queried_name [question_type]
nslookup *can be used as a command or interactively. Under Linux, it is deprecated and dig is recommended. On Windows, it is available by default. C:\> nslookup [-q=question_type] queried domain. [IP_or_name]*

3 Mailing installation and configuration (RIGHT MACHINE)

3.1 Postfix installation and configuration

Installation

The installation of postfix is achieved using the "yum" command.

```
root@pcX]# yum install postfix OR user@pcX]# sudo yum install postfix
```

Configuration

The configuration of postfix is done by editing the /etc/postfix/main.cf file
Add and/or modify the parameters in this file to obtain the following parameters:

```
myhostname = pcX. zX.rt.iut
mydomain= zX.rt.iut
inet_interfaces = all
mydestination = $myhostname, localhost.$mydomain, localhost, $mydomain
mynetworks_style = subnet
home_mailbox = /var/mail/
my origin = $mydomain
```

There are many other parameters which can be used for the configuration of a postfix server. These parameters allow more security, the activation of more functionalities… in our case the above parameters are enough. **Launch postfix:**

```
root@pcX named] # service postfix start
```

3.2 Dovecot installation and configuration

Installation

The installation of dovecot is done using the "yum" command.

Configuration

The configuration of dovecot is done by editing the /etc/dovecot/dovecot.conf file.
Add and/or modify the parameters in this file to obtain the following parameters:

```
protocols = imap pop3
disable_plaintext_auth = no
```

Modify the file /etc/dovecot/conf.d/10-mail.conf

```
mail_location = maildir:~/var/mail
```

Modify **/etc/dovecot/conf.d/10-auth.conf**

```
disable_plaintext_auth = no
```

If needed (to ensure the compatibility with Outlook, in fact we don't need it for this lab), modify the file /etc/dovecot/conf.d/10-mail.conf by removing comment:

```
pop3_uidl_format = %08Xu%08Xv
```

There are many other parameters. It is for example possible to configure the setting of the "listen" directive for one or more protocols; this can allow us for instance to restrain the POP3 server to answer only requests from the 10.1.X.0/24 network or for requests coming from the server itself.
That makes it possible to forbid users to access their mail if they are not connected to the internal network.
Which are the protocols considered by our Dovecot server?
Start dovecot :

```
root@pcX]# service dovecot start
```

4 Accounts creation, configuration of mailing clients

4.1 Creation of mailing accounts (RIGHT MACHINE)

On the server:
Create the “students” group:

```
[root@pcX named] # groupadd students
```

Create the 2 user accounts as it is described in the paragraph 1:

```
[root@pcX named] # useradd -g students -d /home/username -m -s /bin/bash username
```

The creation of a user account on the CentOS server creates automatically an associated mailing account.

4.2 Thunderbird configuration (BOTH MACHINES)

The mailing client software used is Thunderbird on both machines.
Install Thunderbird using the “yum” command.
Launch Thunderbird.

Configuration Information:

Type of account	e-mail account
Your Name	your name
E-mail address	yourlogin@zX.rt.iut
Type of reception server	POP
Name of the reception server	Choose among the DNS recordings of the zX.rt.iut zone the name which corresponds best.
Server of sending	Choose among the DNS recordings of the zX.rt.iut zone the name which corresponds best.
Name of ingoing user	Yourlogin
Name of outgoing user	Yourlogin
Name of the account	yourlogin@zX.rt.iut

Make sure to provide right username and password, password must match the one which you have provided while configuring local users.
Then click on the manual configuration.
Select POP3.
Select right server name, it should be the one mentioned in the configuration file. Make sure DNS is working fine and you can ping that name from the client.
One important thing to consider here is while configuring when we click manual configurations, it automatically gives a server name matching domain of your email account, make sure to remove “.” before the name of that server.
An example of screenshot is given in the appendices.

4.3 Validation

At this stage you should be able to exchange e-mails.
Show to the teacher.

5 Some security problems

Analyze the frames:
On the CentOS server, install Wireshark using the “yum” command.
Launch a capture on the eth1 interface. Apply the following filter: “pop.request.command”.

TAKE CARE! It will be surely necessary to launch Wireshark using `sudo`.

An example of analysis of frames is proposed in the appendix. Make a capture of your own frame. Record it and keep it.

What can you deduce from this capture concerning the security of your mailing system?

6 Use of cryptography tools (BOTH MACHINES)

6.1 OpenSSL

Presentation of OpenSSL

The SSL (Secure Socket Layer) Protocol was developed by the Netscape Communications Corporation to make it possible the client/server applications to communicate in a protected way. TLS (Transport Layer Security) is an evolution of SSL achieved by the IETF.

SSL is a protocol which is above the TCP/IP layer and below the applications layers.
A SSL session is achieved in 2 steps:

- The handshake: it is a phase during which the client and the server agree on the encryption system and the key which will be used thereafter. The strongest encryption method known by the 2 systems will be used.
- The communication itself. During communication, the data exchanged are encrypted and signed.

During the handshake, the identification is achieved through X.509 certificates.

OpenSSL is a cryptographic toolbox which implements TLS and SSL protocols.
OpenSSL includes:

- A library of C programs which makes it possible to develop protected client/server applications (the communications are based on SSL/TLS).
- A tool (using command lines) which allows:
  - The creation of RSA/DSA keys.
  - The creation of X509 certificates.
  - The calculation of hashes (MD5, SHA…).
  - Encryption and decryption (DES, IDEA, RC2, RC4…).
  - The realization of tests for SSL/TLS clients and servers.
  - The signature and the encryption of e-mails (S/MIME).

Examples of use

For all these examples, the working directory is the user home directory.
Create a file named “text.clair” containing the text:

```
“To encrypt and decrypt data with openSSL is not so complicated!!”
```

RC4 symmetric encryption

To encrypt data means to avoid that any person reads it without knowing the decryption key. In the case of a symmetric encryption, one uses the same key to encrypt and decrypt.
Encrypt the text.clair file by applying the RC4 algorithm:

```
[etud1@pcX ~] $ openssl rc4 -in text.clair -out text.rc4
```

Do again the operation with the same key:

```
[etud1@pcX ~] $ openssl rc4 -in text.clair -out text.rc42
```

Compare these two output files:

```
diff text.rc4 text.rc42
```

Are these two files identical? (For that, you may use the `cmp` command).
More precisions on RC4:
The password is not used directly as a key. It is used for generating a key with a random number: “the salt”. Thus two messages encrypted with the same password will not be identical.

Send the file text.rc4 to your neighbor with the encryption key.
Decrypt the file of your neighbor:

```
[etud2@gwX ~] $ openssl rc4 -d -in text.rc4 -out text.rc4.dec
```

Compare with the original file:

```
[etud2@gwX ~] $ diff text.clair text.rc4.dec
```

What are your observations?

Base64 coding

Take again the information from your POP3 frames capture
Let’s create 2 files:

- 1.b64:

```
echo ZXR1ZDE= > /home/yourlogin/1.b64
```

- 2.b64:

```
echo dG90bw== > /home/yourlogin/2.b64
```

Decode these 2 files with base64:

```
[etud2@gwX ~] $ openssl enc -base64 -d -in 1.b64 -out 1
[etud2@gwX ~] $ openssl enc -base64 -d -in 2.b64 -out 2
```

Why base64 isn't an encryption algorithm?
What can we say about the security of the mailing system?

Hashing techniques

A hash function is a one-direction function. It makes it possible to calculate a print (hash) linked to the message to be transmitted. In theory it is impossible to determine the contents of a file from its hash.
The two most used techniques for hashing are md5 (deprecated nowadays) and SHA.

Hash the text.clair file with sha1:

```
[etud2@gwX ~] $ openssl sha1 text.clair
SHA1 (text.clair) = 34dad5a2e10810d84be7d787edc212a8
[etud2@gwX ~] $ openssl sha1 text.clair > emptext.clair
```

Modify temporarily the text.clair file.
Hash again the text.clair file:

```
[etud2@gwX ~] $ openssl sha1 text.clair > emptext.clair.2
```

Compare the two hashes:

```
diff emptext.clair emptext.clair.2
```

Remove the modifications done to text.clair and renew the operation.

What can we deduce from the comparison of hashes:

- In the case where they are identical?
- In is the case where they are different?

Linux passwords

Passwords are encrypted with the DES symmetric algorithm. Neither the password itself nor the encrypted version of the password is stored on the harddisk. One stores the password hash.
/etc/shadow: (more /etc/shadow | grep yourlogin)

```
etud2: $1$G1tcFI7A$EMH4ctom2PrQJ6HFC/A43/:15318:0:99999:7:::
```

user : -----SALT------$---------HASH---------:
Generation of the hash for the etud2 password “toto1” using the salt: G1tcFI7A
To be done:

```
[root@gwX etud2] # openssl passwd -1 -salt G1tcFI7A toto1
```

Which is the password of the system from where this example comes from?
On the recent distributions, the algorithm has been changed (6 instead of 1), so it doesn’t function any more.

Asymmetric methods with couple of public/private key

One generally uses:

- Diffie-Helman.
- RSA (Rivest Shamir Adleman).
- DSA (Digital Signature Algorithm).

Generate a couple of keys:

```
[root@gwX etud2] # openssl genrsa -out username.priv 4096
```

This command generates a private key with a 4096-bit length.

Generate the public key from the private key:

```
[root@gwX etud2] # openssl rsa -in username.priv -pubout -out username.pub
```

Encrypt text.clair with the public key:

```
[root@gwX etud2] # openssl rsautl -inkey username.pub -pubin -in text.clair -out text.username.pub -encrypt
```

Decrypt with the private key:

```
[root@gwX etud2] # openssl rsautl -inkey username.priv -in text.username.pub -out text.username.pub.decrypt -decrypt
```

Of course generally encryption and decryption are not carried out on the same computer.

Achieve the opposite operation:

One encrypts with the private key and decrypts with the public key.

What is the result? Does that appear logical? Explain why.

6.2 Digital signature

Digital signature (sometimes called electronic signature) is a mechanism making it possible to guarantee the integrity of a digital document and to authenticate the author of it, by analogy with the handwritten signature of a paper document.

A digital mechanism of signature must present the following properties:

- It must make it possible to the reader of a document to identify the person or the organization who affixed her/his signature.
- It must guarantee that the document was not altered between the time when the author signed it and the time when the reader consulted it.

For that, the following conditions must be met:

- Authentic: The identity of the sender should be able to be proved in an unquestionable way.
- Unfalsifiable: The signature cannot be falsified. Somebody cannot be made pass for another.
- Non reusable: The signature is not reusable. It belongs to the signed document and cannot be moved on another document.
- Inalterable: A signed document is inalterable. Once it is signed, one cannot any more modify it.
- Irrevocable: The person who signed cannot denies it.

The digital signature is possible only with asymmetric cryptography.
Contrary to the written signature it is not visual, but corresponds to a succession of numbers.
Thanks to the public key of the transmitter, the receiver can:

- check the received hash (option “verify”)
- calculate its own version of the hash for the received information
- compare the two versions of the hash.

The following lines give some examples on how to use these commands. Adapt these lines to your situation and exchange with your colleague some signed messages.
Check this signature using the public key.

```
[etud2@gwX digitalsigning] $ openssl rsautl -verify -in signature_num.sign -inkey cle_iut.pub -pubin > signature_num.emp
```

Calculate your own version of the hash from the file signature_num.pdf:

```
[etud2@gwX digitalsigning] $ openssl dgst -sha1 -out signature_num.emp2 signature_num.pdf
```

Compare the hashes, what do you deduce?
In practice we can proceed in a quicker way:
Signature of a file signature_num.pdf:

```
[etud2@gwX digitalsigning] $ openssl dgst -sha1 -sign cle_iut.priv -out signature_num.sign signature_num.pdf
```

Verification of the file with its signature:

```
[etud2@gwX digitalsigning] $ openssl dgst -sha1 -verify cle_iut.pub -signature signature_num.signsign signature_num.pdf
```

6.3 PGP (Pretty Good Privacy)

Simulation of PGP with openSSL

You will use the couple of keys created previously for this simulation.
You will use the techniques of symmetric and asymmetric encryption. A student will encrypt with a symmetric key of his choice (keep the key secret) a confidential file.
He will exchange this symmetric key with his binomial thanks to his couple of asymmetric keys and the public key of his binomial.
The recipient will use his couple of asymmetric keys and the public key of his binomial to recover the secret key and will decrypt the confidential file.
1. Exchange your public keys thanks to your e-mails.
2. The student on the right-hand side creates a key.txt file containing the symmetric key and a file named confidential.txt which contains secret information.
To generate symmetric key: Openssl rand 128 > key.txt
3. The student on the right-hand side encrypts this file with the symmetric key key.txt. The following command is used to encrypt with the symmetric key.

```
[etud1@pcX ~] $ openssl enc -bf -in confidential.txt -k key.txt -out confidential.txt.cryptsym
```

4. The encrypted message is hashed:

```
[etud1@pcX ~] $ openssl dgst -sha1 -binary confidential.txt.cryptsym > confidential.txt.cryptsym.dgst
```

5. The message is signed:

```
[etud1@pcX ~] $ openssl rsautl -in confidential.txt.cryptsym.dgst -sign -inkey etud1.priv > confidential.txt.cryptsym.sign
```

Generate public and private asymmetric keys for the other user:

openssl genrsa -out username.priv 4096

```
openssl rsa -in username.priv -pubout -out username.pub
```

6. The student on the right should pass the secret key, it will encrypt it with the public key of the student of left:

```
openssl rsautl -inkey username.pub -pubin -in key.txt -out key.txt.cryptasym -encrypt
```

7. The student on the right sends to the student on the left: *key.txt.cryptasym*, *confidential.txt.cryptsym.sign* and *confidential.txt.cryptsym* thanks to e-mail.

8. The student on the left will check that the file confidential.txt.cryptsym was not modified thanks to its signature:

```
[etud2@gwX ~] $ openssl rsautl -verify -pubin -inkey etud1.pub -in
confidential.txt.cryptsym.sign -out dgst1
[etud2@gwX ~] $ openssl dgst -sha1 -binary confidential.txt.cryptsym > dgst2
[etud2@gwX ~] $ diff dgst1 dgst2
```

9. The student on the left recovers the secret key thanks to her/his private key:

```
[etud2@gwX ~] $ openssl rsautl -decrypt -inkey etud2.priv -out key.txt -in key.txt.cryptasym
```

10. She/he decrypts the message:

```
[etud2@gwX ~] $ openssl enc -bf -d -in confidential.txt.cryptsym -k key.txt -out
confidential.txt
```

This simple method enables us to send files in a secure way.

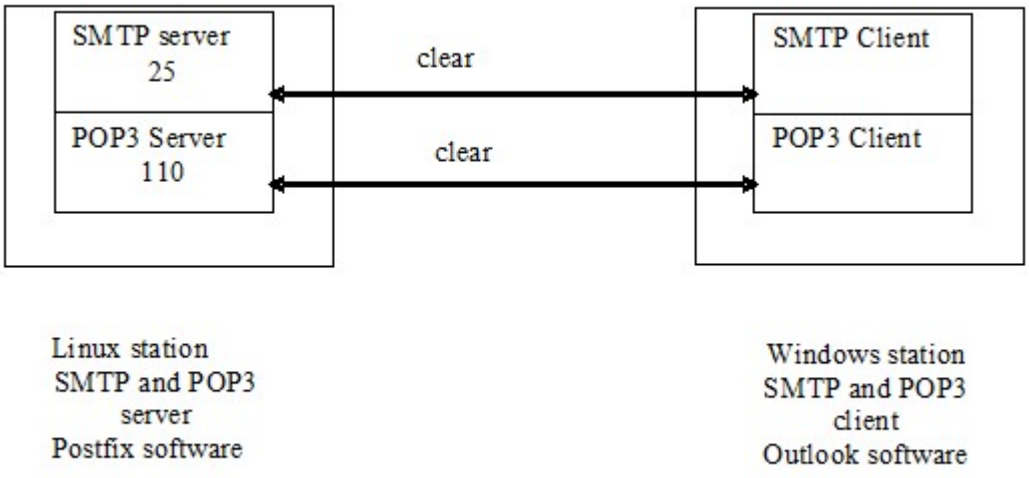
What are the security roles ensured? (At which stage(s)?).

At this stage, there is an uncertainty, where?

Prepare a diagram of the protected transmission. This diagram should be given to the teacher.

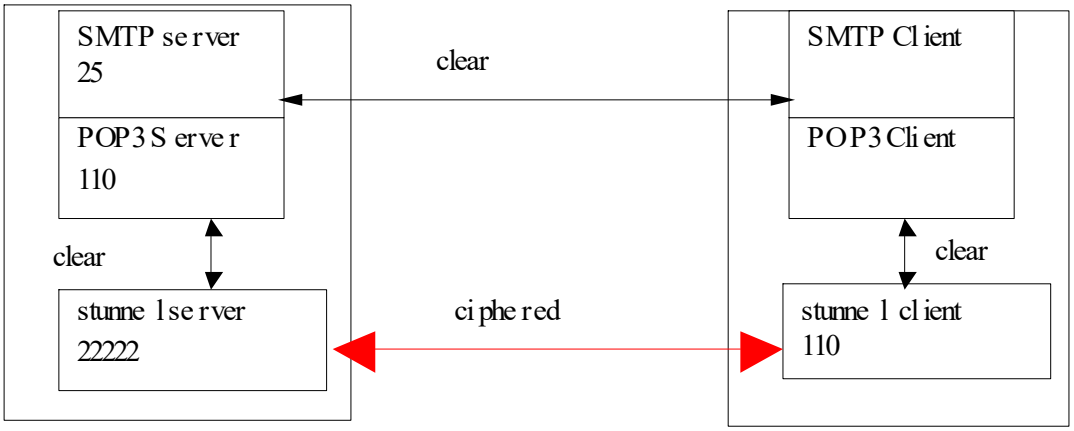
6.4 Security of the mailing system with stunnel

Current situation



All the exchanges between the client and the server are done in clear.

First Solution for security

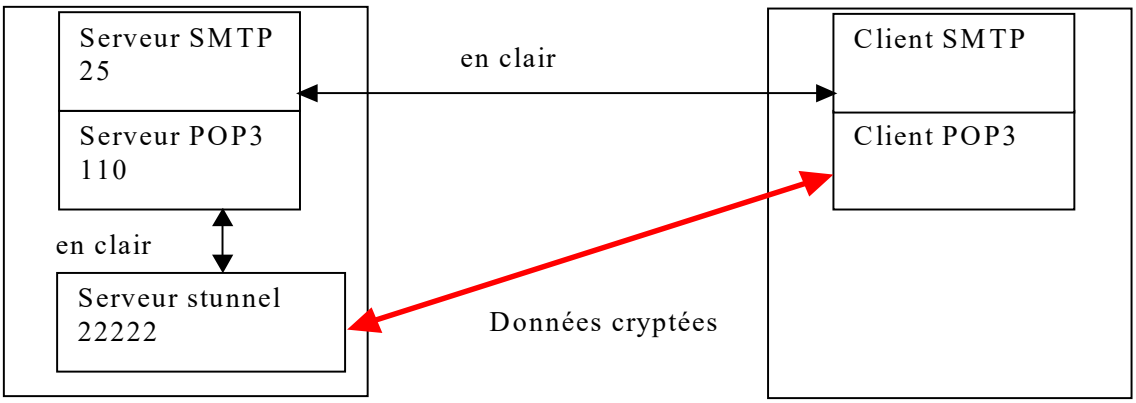


Linux station
SMTP and POP3
Serveur
Postfix software
Stunnel (on pop)

Window s station
SMTP and POP3
client
Eudora software
Stunnel Client

The POP3 Client connects itself on the stunnel client.
The stunnel client connects itself to the stunnel server by means of a protected connection.
The stunnel server transmits the request of the client to the POP3server. There are no password in clear on the network.

Second solution for security



Poste Linux
Serveur SMTP et
POP3
Logiciel Postfix
Stunnel (sur pop)

Poste Windows
client SMTP et
POP3
Logiciel Eudora

The POP3 Client is able to connect itself in a protected way to the stunnel server.
The stunnel server relays the request to the POP3server.

Choice of the solution

The initial solution is not protected thus unacceptable.
The second solution is protected. It implies the installation and the configuration of a stunnel server on the server but also the installation and the configuration of a stunnel client on all the computers having to host a mailing client.
The second solution for security is so the one which is chosen.

Configuration of the stunnel server

On the server: Install stunnel
Yum install stunnel

- Create a private key for the stunnel server:

```
[root@gwX etud2]# openssl genrsa -out /etc/stunnel/stunnel.key 1024 OR [user@gwX etud2]# sudo openssl genrsa -out /etc/stunnel/stunnel.key 1024
```

- Create a request to get a x509 certificate:

```
[root@gwX etud2]# openssl req -new -key /etc/stunnel/stunnel.key -out /etc/stunnel/stunnel.csr

Enter pass phrase for /etc/stunnel/stunnel.key:
You are about to be asked to enter information that will be incorporated into your certificate request. What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank. For some fields there will be a default value. If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:FR
State or Province Name (full name) [Berkshire]:Isere
Locality Name (eg, city) [Newbury]:Grenoble
Organization Name (eg, company) [My Company Ltd]:LPRO
Organizational Unit Name (eg, section) []:CNMS
Common Name (eg, your name or your server's hostname) []:pop3.zX.rt.iut
Email Address []:root@zX.rt.iut
```

- Sign the request for a certificate:

```
[root@gwX etud2] # openssl x509 -req -days 365 -in /etc/stunnel/stunnel.csr -signkey /etc/stunnel/stunnel.key -out /etc/stunnel/stunnel.crt

Signature ok
subject=/C=FR/ST=Isere/L=Grenoble/O=LPRO/OU=CNMS/CN=pop3.zX.rt.iut/emailAddress=root@zX.rt.iut
Getting Private key
```

- Configure stunnel on the server:

```
[root@gwX etud2] # cp /usr/share/doc/stunnel-4.56/stunnel.conf-sample /etc/stunnel.conf
```

Edit this file in order to have the same options as in the appendix.

```
cert = /etc/stunnel/stunnel.crt
key = /etc/stunnel/stunnel.key
```

- Launch stunnel

```
[root@gwX etud2] # stunnel /etc/stunnel.conf
```

On the Thunderbird client:
Change the configuration of the account to use SSL at the time of the downloading of e-mails.
Analyze the frames with wireshark during a connection. Is the security complete?

6.5 Security of SSH

6.5.1 SSH Connection

On the CentOS client:
- Connect yourselves using SSH on the server

```
[etud1@pcX ~] $ SSH root@gwX.zX.rt.iut
```

- Change the password for “WsD43%iop09=AZgh”.
- Disconnect and establish the connection again.

What do you think? How to facilitate the connection of an administrator? Which problem is posed?

6.5.2 Configuration of the authentication by a RSA key

Take the identity of the user of the machine:

```
[etud1@pcX ~] $ su -username
```

Generate a pair of RSA keys:

```
[etud1@pcX ~] $ ssh-keygen
```

Do not put a passphrase.
Send your public key to your binomial
Copy the public key of your binomial in the directory /root/.ssh/authorized_keys (the format is explained in appendix).
Connect yourselves to the station of your binomial:

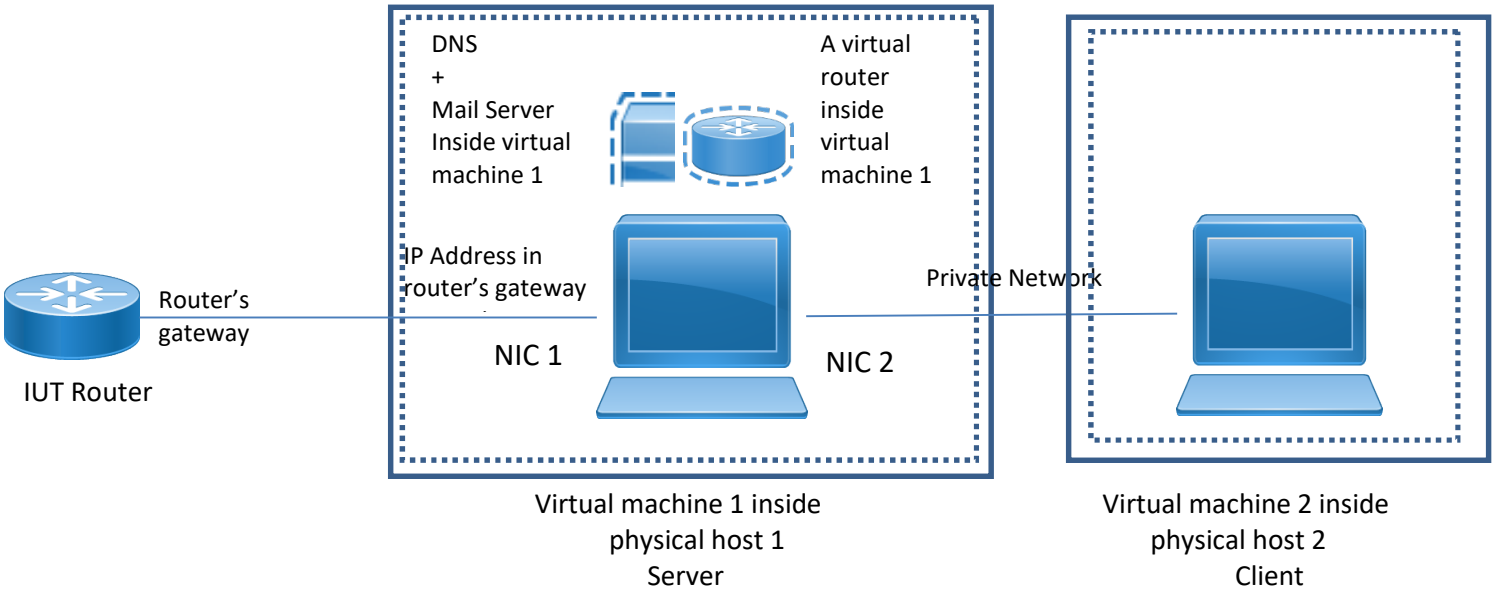
```
[etud1@pcX ~] $ ssh root@gwX.zX.rt.iut
```

What can you notice?
Which security risk are we exposed to?

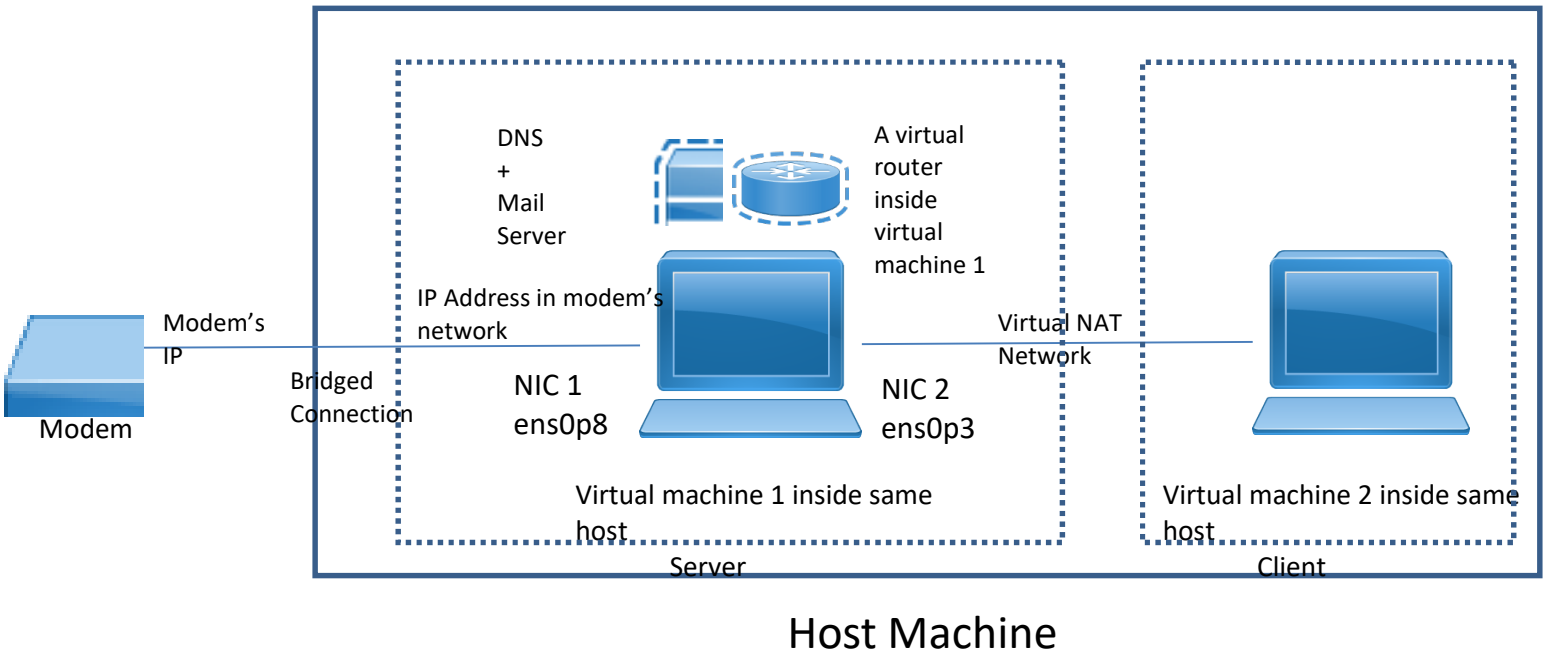
Improve the security with the SSH agent mechanism. (its functioning is largely described on Internet).

7 Appendices

Network Diagram for physical working in lab



Network Diagram for remote working:



Configuration of Bind:

/etc/named.conf

```
// named.conf
//
// Provided by Red Hat bind package to configure the ISC BIND named(8) DNS
// server as a caching only nameserver (as a localhost DNS resolver only).
//
// See /usr/share/doc/bind*/sample/ for example named configuration files.
//

options {
listen-on port 53 { 127.0.0.1; }; TAKE CARE ABOUT THE CONFIGURATION HERE !
directory "/var/named";
dump-file "/var/named/data/cache_dump.db";
statistics-file "/var/named/data/named_stats.txt";
memstatistics-file "/var/named/data/named_mem_stats.txt";
allow-query { 10.32.1.0/24; any; }; TAKE CARE ABOUT THE CONFIGURATION HERE !
forwarders { 192.168.1.1; }; TAKE CARE ABOUT THE CONFIGURATION HERE !
recursion yes;
pid-file "/run/named/named.pid";
session-keyfile "/run/named/session.key";
};

logging {
channel default_debug {
file "data/named.run";
severity dynamic;
};
};
zone "." IN {
type hint;
file "named.ca";
};

include "/etc/named.rfc1912.zones";
zone "zX.rt.iut" in {
type master;
file "direct.zX.rt.iut";
allow-update{none;};
};
zone "X.1.10.in-addr.arpa" in {
type master;
file "reverse.zX.rt.iut";
allow-update{none;};
};
```

/var/named/direct.zX.rt.iut
// be careful with that you put a point (dot) at the end of yellow highlighted otherwise named service will not start
These files are examples. Take care about the addresses!
Highlighted in red are important, if you are having errors, check if you have written those.
@ is replaced with the content of $ORIGIN, here: zX.rt.iut.

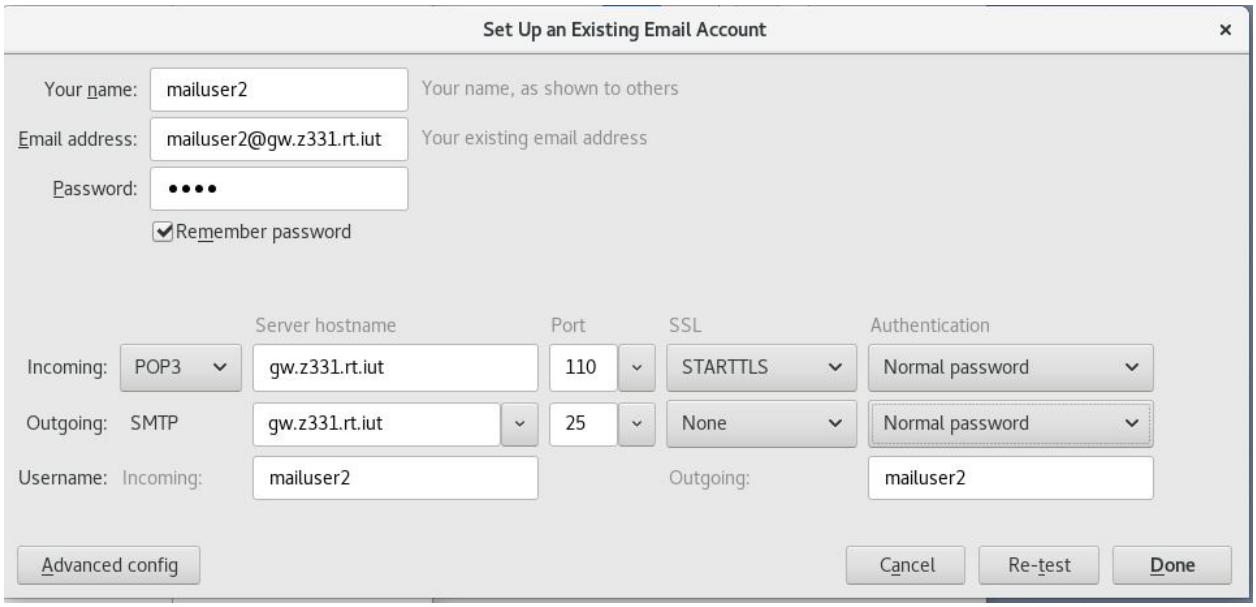
```
; /var/named/direct.zX.rt.iut
$TTL  30
$ORIGIN zX.rt.iut.
@           IN    SOA    gwX.zX.rt.iut. root.gwX.zX.rt.iut. (
                                        200107011; Serial
                                        8H    ; Refresh
                                        2H    ; Retry
                                        1W    ; Expire
                                        1D   ; Minimum
                                        )
                          IN    NS    gwX.zX.rt.iut.
localhost     IN    A          127.0.0.1
gwX           IN    A            10.32.1.1
e-mail        IN    CNAME    pcX
smtp          IN    CNAME    pcX
IMAP          IN    CNAME    pcX
pop3          IN    CNAME    pcX
pcX         IN    A    10.32.1.254
```

/var/named/reverse.zX.rt.iut

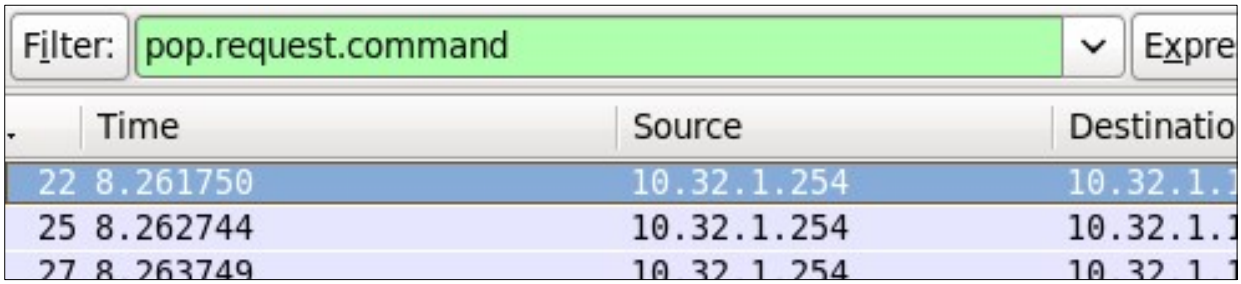
```
; /var/named/reverse.zX.rt.iut
$TTL  30
$ORIGIN 1.32.10.in-addr.arpa.     ; TAKE CARE ABOUT THE CONFIGURATION HERE !
@                    IN      SOA   gwX.zX.rt.iut. root.zX.rt.iut. (
                                        1288639578; Serial
                                        8H; Refresh
                                        2H; Retry
                                        1W; Expire
                                        1D; Minimum
                                    )
                          IN      NS     gwX.zX.rt.iut.

254                  IN      PTR    gwX.zX.rt.iut.
1                    IN      PTR    pcX.zX.rt.iut.
```

Modify /etc/named.conf as recommended.
Edit /etc/sysconfig/named so to run named only as an IPv4 server, otherwise it will run many IPv6 requests (and many errors in the log files)
OPTIONS="-4"

Thunderbird account settings:



Example of wireshark capture.
/root/.ssh/authorized_keys /root/.ssh/authorized_keys



ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAtzgNCZhqVqdmRhCH6KCfbpoKL2dXK4vGjqT6ABT3LO2FMQKeKXvGOel
ZPupFU7pzessbAx9iWlpV24xSgvfYz2oYDEwR0yXdAvQSsUliZMuGZ/om+XgfUQQwILuhccvL6Ccx5k90UC5NKc
5gTXIlb8KhR4Rk8EmkD3CFuJh+RediY0sezlxgcGmPVdmVvV1YJ4LSPogPCRYojTbjPDECbLYQd/Z5GRsfFOCr4kYX
x05i9gInOET80Z8aplRX0qL2ou4ZrhOHgVxqTbFSEUbNp96BcspfogZu1xBG/KWjf9Z9DGb/pbsz/F5eol2EpmoZL
8tpj3uMVRt5C5DHSFIM4w== root@pcX.zX.rt.iut