Lab Security: Cryptography

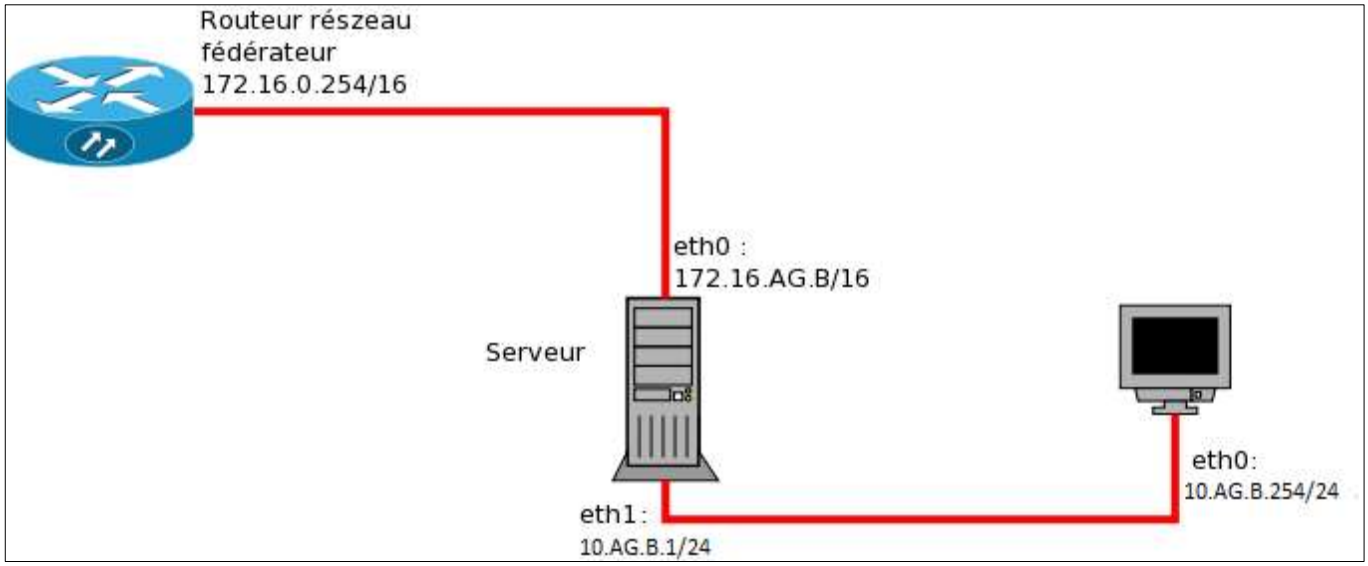
1 Installation of the platform

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)).

- The left computer is the server and ensures the following roles:
  - Router.
  - Mailing server.
- The computer on the right is the client:

1.1.1 Diagram of the platform



1.1.2 Network configuration

From the USB key install webmin

Rpm –ivh webminXXX.rpm

Some dependencies are required yum install perl-Net-SSLeay.x86_64

Connect with a webbrowser to the server https://@IP:10000

OS	Role	eth0	eth1	eth2	Hostname	Gateway
Centos 5.7	Server	172.16.AG.B/16	10.AG.B.1/24	N/A	gw.zAG-B.rt.iut	172.16.0.254
Centos 5.7	Client	10.AG.B.254/24	non activated	N/A	pcAG-B.zAG-B.rt.iut	10.AG.B.1

"IP Forwarding" will be activated on the server.
A=Année (year) (for you 3rd year).

G=Groupe (FI=1, Alt=2, RTWEB=3, WiNS = 4)
B=Binomial (LAB Cryptography: 1,2 and 3).

1.1.3 DNS resolution

Machine	DNS primary server	DNS secondary server	Default DNS Suffix
Server	127.0.0.1	172.16.0.200	
Client	10.AG.B.1	N/A	

1.1.4 Configuration of the users

Each student will have a user account on the server.
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”

1.2 CentOS installation and wiring

1.2.1 Wiring

The server eth0 card will be wired on the switch of the backbone network (172.16.X.X).
The server eth1 card will be connected on the switch available on your table.
The client eth0 card will be connected on the switch available on your table.
The other network cards will not be used.

The 2 machines are installed using CentOS Ghost images (ask the teacher).
The configuration of the server and the client are described in the following paragraphs.

1.2.2 Configurations files

- Hostname: /etc/sysconfig/network
- IP address(es): /etc/sysconfig/network-scripts/ifcfg-ethX
- DNS resolution: /etc/resolv.conf
- Default gateway: /etc/sysconfig/network
- IP Forwarding: /etc/sysctl.conf

Edit these files to configure your client and server using data given above.
If you cannot edit a file, please refer to the appendix “edition of files with vim”.

1.3 Validation

At this stage, your client must be able to contact:

- the CentOS server,
- the router of the backbone network,
- a public Internet server (ex: 152.77.24.211),

thanks to its public IP address.

At this stage, your server must be able to contact:

- the CentOS client,
- the router of the backbone network,
- an Internet server (ex 152.77.24.211).
- and to solve all DNS names (ex: www.google.fr).

You are asked to prepare and launch a list of commands allowing you to show the correct functioning.
Please show to the teacher.

2 Installation and configuration of the BIND DNS server

2.1 Installation of BIND

The installation of BIND is done using the “yum” command.

```
[root@gw ~] # yum install bind
```

2.2 BIND configuration

USE WEBMIN FOR THE 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: “zAG-B.rt.iut” in the adequate file.
- to declare the reverse zone: “B.AG.in-addr.arpa” in the adequate file.

Determine the location of the zone files in /etc/named.conf
Configure the BIND server using example files available in the appendix.

Launch the DNS server:

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

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)
- to solve the names in the zAG-B.rt.iut zone (ex: pop3.zAG-B.rt.iut).

From the server side:

- to solve the names in the zAG-B.rt.iut zone (ex: pcAB.zAG-B.rt.iut)
- to solve the names in the zAG-B.rt.iut zone (ex: pcAB.bB.gtr.iut)

You are asked to prepare and launch a list of commands allowing you to show the correct functioning.
Please show to the teacher.

3 Mailing installation and configuration

3.1 Postfix installation and configuration

3.1.1 Installation

The installation of postfix is achieved using the “yum” command.

3.1.2 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 = gw. zAG-B.rt.iut
mydomain= zAG-B.rt.iut
inet_interfaces = all
mydestination = $myhostname, localhost.$mydomain, localhost, $mydomain
mynetworks_style = subnet
```

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@gw named] # service postfix start
```

3.2 Dovecot installation and configuration

3.2.1 Installation

The installation of dovecot is done using the “yum” command.
COPY FROM THE USB KEY THE FOLDER DOVECOT AND REPLACE THE ONE IN /etc

3.2.2 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
```

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.AG.B.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?

4 Accounts creation, configuration of mailing clients

4.1 Creation of mailing accounts

On the server:
Create the “students” group:

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

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

```
[root@gw 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

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@zAG-B.rt.iut
Type of reception server	POP
Name of the reception server	Choose among the DNS recordings of the zAG-B.rt.iut zone the name which corresponds best.
Server of sending	Choose among the DNS recordings of the zAG-B.rt.iut zone the name which corresponds best.
Name of ingoing user	yourlogin
Name of outgoing user	yourlogin
Name of the account	yourlogin@zAG-B.rt.iut

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”.

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

6.1 OpenSSL

6.1.1 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 are exchanged are encrypted, compressed 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).

(source: www.wikipedia.fr)

6.1.2 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 symmetrical encryption

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

```
[etud1@pc31 ~] $ openssl rc4 -in text.clair -out text.rc4
```

Do again the operation with the same key:

```
[etud1@pc31 ~] $ openssl rc4 -in text.clair -out text.rc42
```

Compare these two output files:

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

Are these two files identical?
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@gw ~] $ openssl rc4 -d -in text.rc4.etud1 -out text.rc4.etud1.dec
```

Compare with the original file:

```
[etud2@gw ~] $ diff text.clair text.rc4.etud1.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@gw ~] $ openssl enc -base64 -d -in 1.b64 -out 1
[etud2@gw ~] $ 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 and SHA (SHA1,…).

Hash the text.clair file with md5:

```
[etud2@gw ~] $ openssl md5 text.clair
MD5 (text.clair) = 34dad5a2e10810d84be7d787edc212a8
[etud2@gw ~] $ openssl md5 text.clair > emptext.clair
```

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

```
[etud2@gw ~] $ openssl md5 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 symmetrical 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@gw 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.
(Already done on your Linux) Reconfigure your system to use MD5 passwords:

```
[root@gw etud2] #authconfig -enablemd5 -enableshadow --update
```

Change the password of a user. Start again the operations.
How the machine knows that the given password is the correct one if it is not stored on its harddisk?

Asymmetrical 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@gw 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@gw etud2] # openssl rsa -in username.priv -pubout -out username.pub
```

Encrypt text.clair with the public key:

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

Decrypt with the private key:

```
[root@gw 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 deny it.

The digital signature is possible only with asymmetrical cryptography.

Contrary to the written signature it is not visual, but corresponds to a succession of numbers.

Example:

Recover on the ftp server: /ftp.gtrgrenoble.fr/TpL3/crypto/ the UserGuide.pdf file, the UserGuide.pdf.sign file and the couple of keys from IUT: cle_iut.pub and cle_iut.priv

The Userguide.sign file is the digital signature of the UserGuide.pdf file.

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.

Check this signature using the public key of accountancy (comptabilité).

```
[etud2@gw 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@gw digitalsigning] $ openssl dgst -md5 -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@gw digitalsigning] $ openssl dgst -md5 -sign cle_iut.priv -out signature_num.sign signature_num.pdf
```

Verification of the file with its signature:

```
[etud2@gw digitalsigning] $ openssl dgst -md5 -verify cle_iut.pub -signature signature_num.signsign signature_num.pdf
```

6.3 PGP (Pretty Good Privacy)

6.3.1 Simulation of PGP with openSSL

You will use the couple of keys created previously for this simulation.
You will use the techniques of symmetrical and asymmetrical encryption. A student will encrypt with a symmetrical key of his choice (keep the key secret) a confidential file.
He will exchange this symmetrical key with his binomial thanks to his couple of asymmetrical keys and the public key of his binomial.
The recipient will use his couple of asymmetrical 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 symmetrical key and a file named confidential.txt which contains secret information.
3. The student on the right-hand side encrypts this file with the secret key key.txt:

```
[etud1@pc31 ~] $ openssl enc -bf -in confidential.txt -k key.txt -out confidential.txt.cryptsym
```

4. The encrypted message is hashed:

```
[etud1@pc31 ~] $ openssl dgst -md5 -binary confidential.txt.cryptsym >
confidential.txt.cryptsym.dgst
```

5. The message is signed:

```
[etud1@pc31 ~] $ openssl rsautl -in confidential.txt.cryptsym.dgst -sign -inkey etud1.priv >
confidential.txt.cryptsym.sign
```

6. The student on the right should pass the secret key, it will encrypt it with the public key of the student of left:
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@gw ~] $ openssl rsautl -verify -pubin -inkey etud1.pub -in
confidential.txt.cryptsym.sign -out dgst1
[etud2@gw ~] $ openssl dgst -md5 -binary -in confidential.txt.cryptsym > dgst2
[etud2@gw ~] $ diff dgst1 dgst2
```

9. The student on the right recovers the secret key thanks to his private key:

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

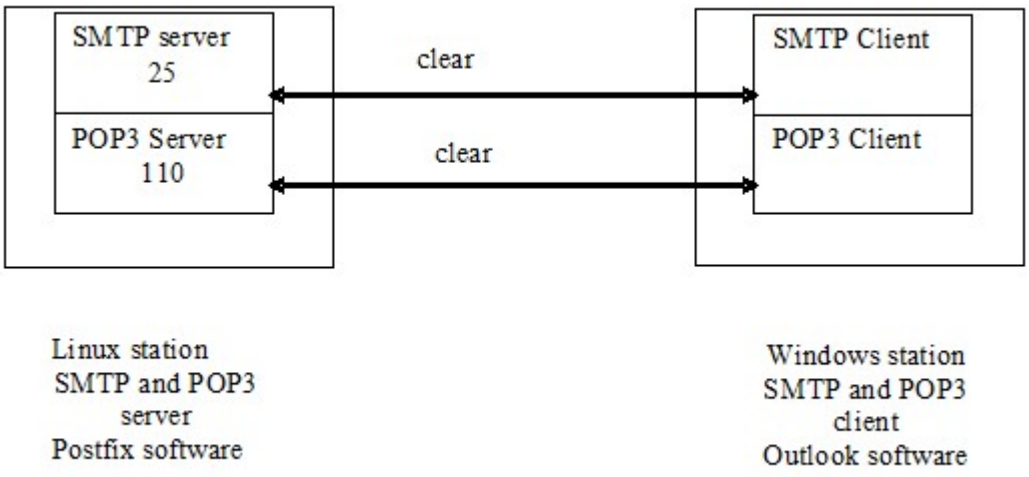
10. He decrypts the message:

```
[etud2@gw ~] $ 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)?).
In the present situation, there is an uncertainty, where?
Prepare a diagram of the protected transmission. This diagram should be given to the teacher.

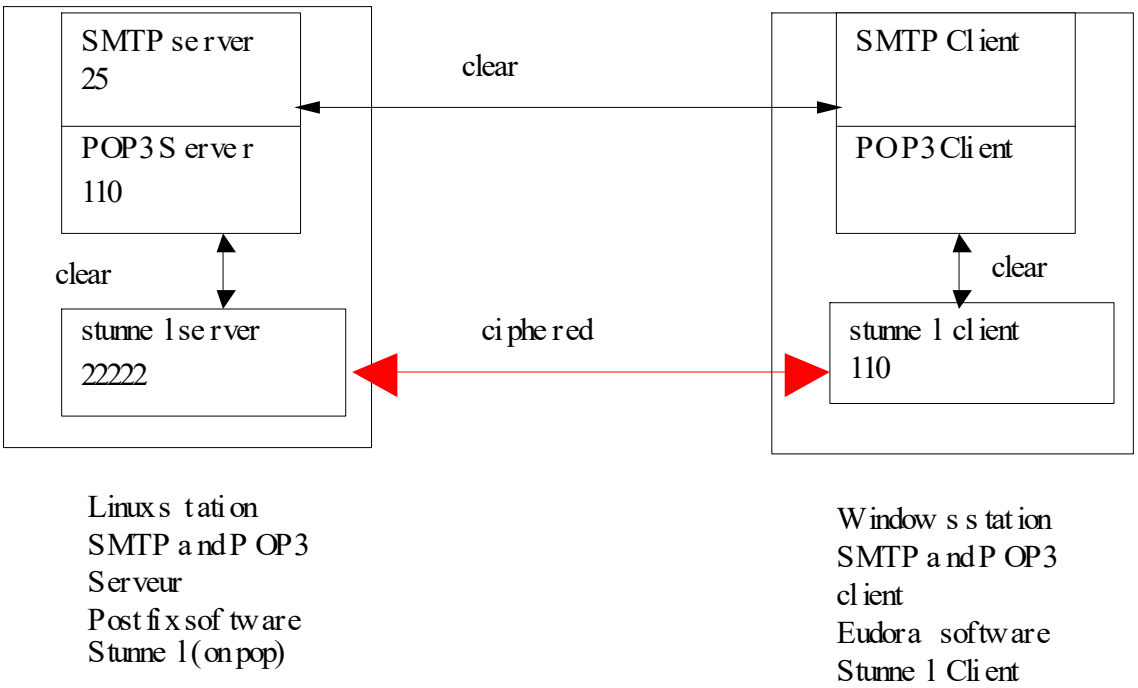
6.4 Security of the mailing system with stunnel

6.4.1 Current situation



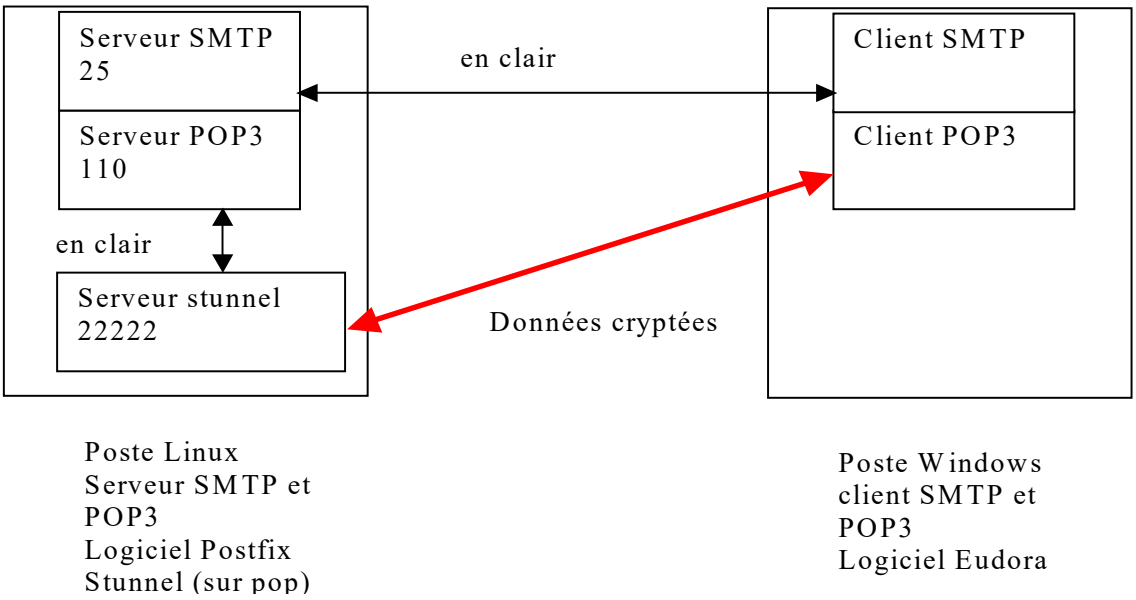
All the exchanges between the client and the server are done in clear.

6.4.2 First Solution for security



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.

6.4.3 Second solution for security



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.

6.4.4 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.

6.4.5 Configuration of the stunnel server

On the server:

- Create a private key for the stunnel server:

```
[root@gw etud2] # openssl genrsa -out /etc/stunnel/stunnel.key 1024
```

- Create a request to get a x509 certificate:

```
[root@gw 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]:Saint Martin d'Hères
Organization Name (eg, company) [My Company Ltd]:LPRO
Organizational Unit Name (eg, section) []:RSFS
Common Name (eg, your name or your server's hostname) []:pop3.b1.gtr.iut
Email Address []:root@b1.gtr.iut
```

- Sign the request for a certificate:

```
[root@gw 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=Saint Martin
d'H\xC3\x83\xC2\xA8res/O=LPRO/OU=RSFS/CN=pop3.b1.gtr.iut/emailAddress=root@b1.
gtr.iut
Getting Private key
```

- Configure stunnel on the server:

```
[root@gw etud2] # CP /usr/share/doc/stunnel-4, 29/stunnel.conf-sample /etc/stunnel
```

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

- Launch stunnel

```
[root@gw etud2] # stunne /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@pc31 ~] $ SSH root@gw.bB.gtr.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@pc31 ~] $ su -username
```

Generate a pair of RSA keys:

```
[etud1@pc31 ~] $ 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/autorized_keys (the format is explained in appendix).
Connect yourselves to the station of your binomial:

```
[etud1@pc31 ~] $ ssh root@gw.zAG-b.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).

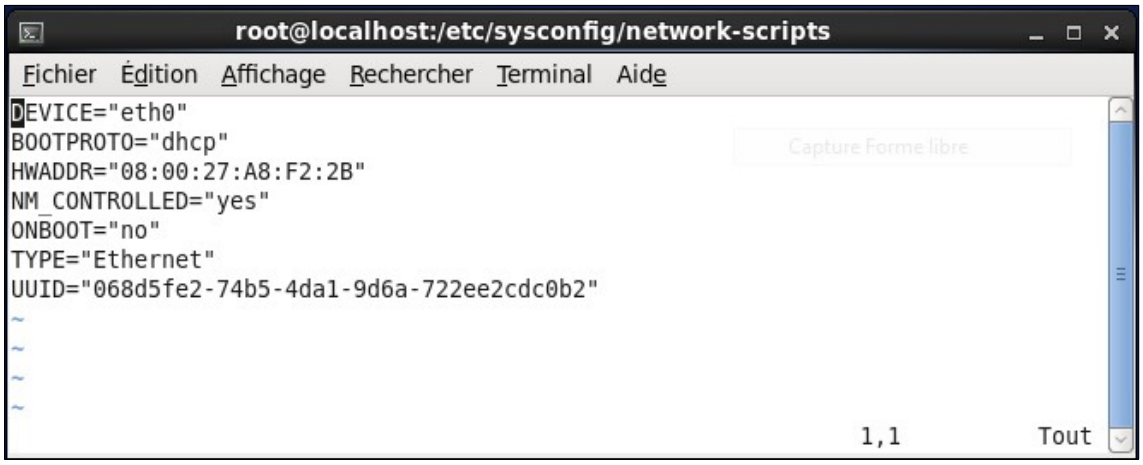
7 Appendices:

7.1 Edition of files with “VIM”

Open the file:

```
[root@localhost ~] # cd /etc/sysconfig/network-scripts/
[root@localhost ~] # vim ifcfg-eth0
```

Edit the file:



Press on the key “i” to pass in the “INSERTION” mode and edit your text.

Leave the “insertion” mode by pressing “esc”.

The commands are preceded by “: ”

To enter an order, to enter “: ” and to make follow order has to carry out.
Useful and sufficient commands:

- wq: write and quit (saving)
- Q!: quit without saving
- W: save without leaving.
- H: help.

7.2 Configuration of Bind:

/etc/named.conf.sample

```
// 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 {
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; };
forwarders { 192.168.1.1 };
recursion yes;
};

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

include "/etc/named.rfc1912.zones";
//include "/etc/named.root.key";
zone "b1.gtr.iut" in {
type master;
file "direct.b1.gtr.iut";
allow-update{none;};
};
zone "1.32.10.in-addr.arpa" in {
type master;
file "reverse.b1.gtr.iut";
allow-update{none;};
};
```

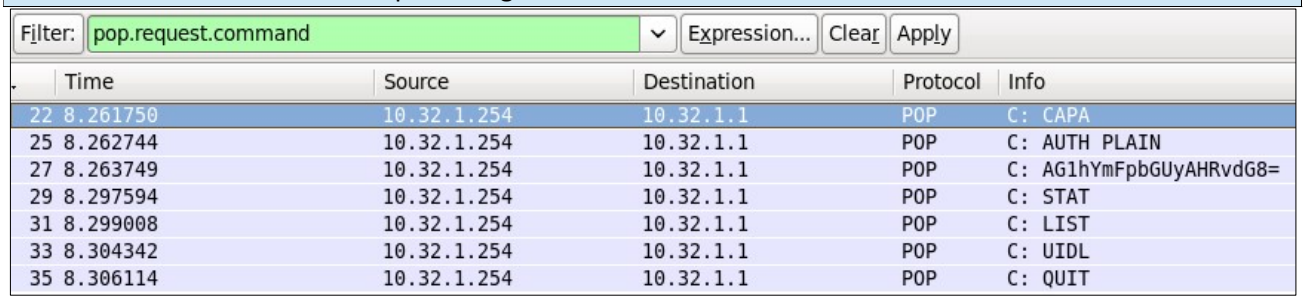
/var/named/direct.zAG-B

```
; /var/named/direct.bX.gtr.iut
$TTL 86400
$ORIGIN b1.gtr.iut.
@           IN    SOA    gw.b1.gtr.iut root.gw.b1.gtr.iut. (
                                        200107011; Serial
                                        8H    ; Refresh
                                        2H    ; Retry
                                        1W    ; Expire
                                        1D)   ; Minimum
                          IN    NS    gw.b1.gtr.iut.
localhost     IN    A           127.0.0.1
gw            IN    A           10.32.1.1
e-mail        IN    CNAME     gw
smtp          IN    CNAME     gw
IMAP          IN    CNAME     gw
pop3          IN    CNAME     gw
pcA1          IN    A    10.32.1.254
```

/var/named/reverse.zAG-B.rt.iut

```
; /var/named/reverse.bB.gtr.iut
$TTL 86400
$ORIGIN 1.32.10.in-addr.arpa.
@                   IN      SOA   gw.b1.gtr.iut root.b1.gtr.iut. (
                                        1288639578; Serial
                                        8H; Refresh
                                        2H; Retry
                                        1W; Expire
                                        1D); Minimum
                          IN      NS     gw.b1.gtr.iut.

254                 IN      PTR    gw.b1.gtr.iut.
1                   IN      PTR    pcA1.b1.gtr.iut.
```



Example of wireshark capture.

/root/.ssh/authorized_keys

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAtzgNCZhqVqdmRhCH6KCfbpoKL2dXK4vGjqT6ABT3LO2FMQKeKXvGOelZ
PupFU7pzessbAx9iWlpV24xSgvfYz2oYDEwR0yXdAvQSsUliZMuGZ/om+XgfUQQwILuhccvL6Ccx5k90UC5NKc5
gTXIlb8KhR4Rk8EmkD3CFuJh+RediY0sezlxgcGmPVdmVvV1YJ4LSPogPCRYojTbjPDECbLYQd/Z5GRsfFOCr4kYXx
05i9gInOET80Z8aplRX0qL2ou4ZrhOHgVxqTbFSEUbNp96BcspfogZu1xBG/KWjf9Z9DGb/pbsz/F5eol2EpmoZL8
tpj3uMVRt5C5DHSFIM4w== root@PCA1.b1.gtr.iut
```