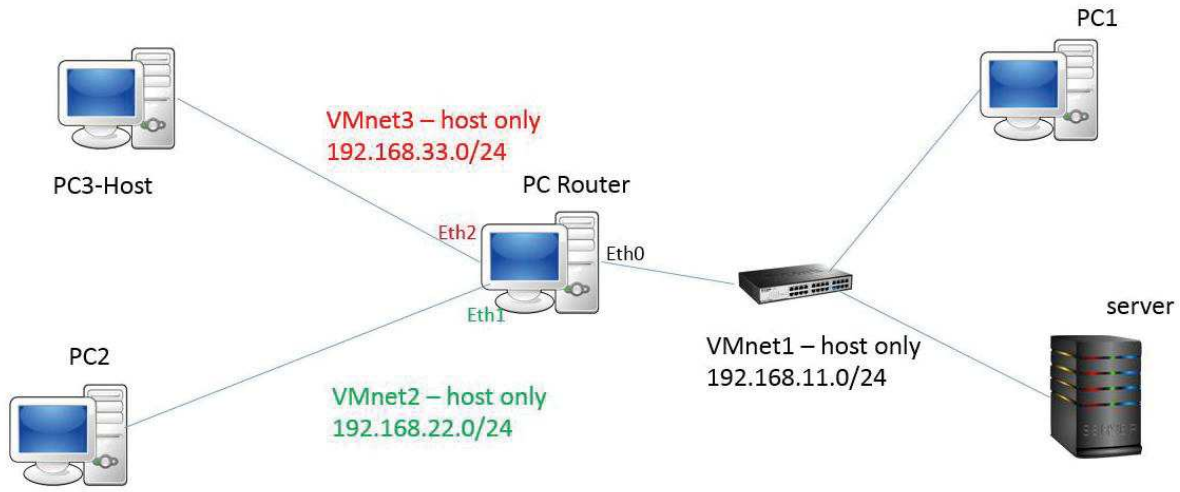
Part 1:

Activity objective:

The learning objective of this lab is to understand how to create a Certificate Authorities, request Certificate, sign certificates, hierarchy CA (**for the same Organization *ECELAB* in the same City Paris**), verify certification chain, configure apache server using SSL and the certificate delivered by CA to make a secure connection between the client and the server.

Visual Objective

You will use the following topology that you have made the last lab (Network virtualization with VirtualBox):



1. Before turn-on PC Router, change the Adapter configuration to NAT where the interface (eth2) or ens**xx** is plugged in VirtualBox and configure the interface (eth2) or ens**xx** on DHCP after booting the VM.
2. To have Internet access in all hosts :
   You have to add a NAT rule into PC Router:
   a. iptables -t nat -A POSTROUTING -o eth2 –j MASQUERADE
   b. apt-get install iptables-persistent.
   c. Save the NAT rule : iptables-save > /etc/iptables/rules.v4
   d. Assign a static IP in all VM and use **PC Router** like a default gateway of **PC1**, **PC2** and **server** (without interface eth2 on PC Router), and give IP address you will be used

VM	IP address
PC2	
PC1	
Server	

   e. Add a name server in /etc/resolv.conf in the others hosts :

nameserver 8.8.8.8

A. CA ROOT:

1. Use PC2 to host the CA ROOT
2. Prepare PC2 to create and sign certificates
    - Create a new directory CA-ROOT where you will make:
        a. mkdir newcerts
        b. mkdir private (you will place all private keys in this directory)
        c. mkdir certs (you will place all the certificates and Certificate request in this directory)
        d. touch index.txt
        e. echo 01 > serial
        f. Create file openssl.cnf (file configuration) (you find the openssl.cnf file in the campus)
3. Generate private key RSA **privcaroot.key** (in private directory) with length of 2048 bits with password (use option –des3) (provide screenshoot)
    - ***Note the password in your rapport and use the same password into the entire lab.***
4. Create a Certificate Self-signed **certcaroot.crt** (valid for 365 days)), Use CommonName = CA ROOT and CA_ROOT like extensions (provide screenshoot)
5. Copy the result in text format of ***openssl x509 –in certcaroot.crt -text -noout***
6. Update the configuration (CA_default section) of openssl.cnf.

B. CA LAB

1. Use PC1 to host the CA LAB
2. Prepare PC1 to create and sign certificates
    - Create a new directory CA-LAB where you will make:
        a. mkdir newcerts
        b. mkdir private (you will place all private keys in this directory)
        c. mkdir certs (you will place all the certificates and certificate request in this directory)
        d. touch index.txt
        e. echo 01 > serial
        f. Create file openssl.cnf (you find the openssl.cnf file in the campus)
3. Generate private key RSA **privcalab.key** with length of 2048 bits with password (provide screenshoot)
4. Create a request of certificate **certcalab.csr**, use CommonName = CA LAB**.**
5. Copy the result in text format of ***openssl req –in certcalab.csr -text –noout***
6. Send the request to the CA Root (use SFTP or SCP).
7. Sign the request of certificate by the CA Certificate and generate **certcalab.crt** , use CA_LAB like extensions.
8. Copy the result in text format of ***openssl x509 –in certcalab.crt -text -noout***
9. Send the certificate signed to PC1 (use SFTP or SCP)

10. Update the configuration (CA_default section) of openssl.cnf in the PC1.

C. Server Certificate

1. Create a private key RSA for the server **privsrv.key** with length of 2048 bits (**without password**) in server (provide screenshoot)
2. Create a request of certificate **certsrv.csr** ,use CommonName = ***<your Name>*** .
3. Copy the result in text format of ***openssl req –in certsrv.csr -text –noout***
4. Send the certificate request to PC1 for singing.
5. Update the configuration of openssl.cnf (SERVER section) with the Server IP address.
6. Sign the certificate by CA LAB and generate the **certsrv.crt** use SERVER like extensions (provide screenshoot)
7. Copy the result in text format of ***openssl x509 –in certsrv.crt -text –noout***
8. Copy the in text format the content of ***index.txt***
9. Send the certificate signed to server.

D. HTTPS Server

1. Active ssl mode in the server Apache2 and configure your site with https.
   a. Use Server Certificate created in the previous section.
   b. (refer to the following page to learn more about the HTTPS configuration **without Certificate generate STEP 2** : https://wiki.debian.org/Self-Signed_Certificate)
   c. Provide the configuration in your rapport.
2. Connect to the server from the PC3-Host using https
   a. Do you get a warning or error? What is the security problem?
   b. Provide a screenshot
3. Import CA ROOT and CA LAB Certificates in your browser (use WinSCP to get certificates).
   a. Restart your browser.
   b. Restart Apache2 service.
   c. Connect to the server from the PC3-Host using https
   d. Provide a screenshot.
   e. What is the difference from question D.2?
4. Capture and analyze with Wireshark an HTTPS traffic on PC3-Host (Provide a screen shot showing the browser getting the page and a sequence diagram that shows the basic SSL exchanges (Handshake Protocol) between the client and the server)
   a. Turn-on Wireshark
   b. Restart Apache2 service.
   c. Reconnect to the https server.

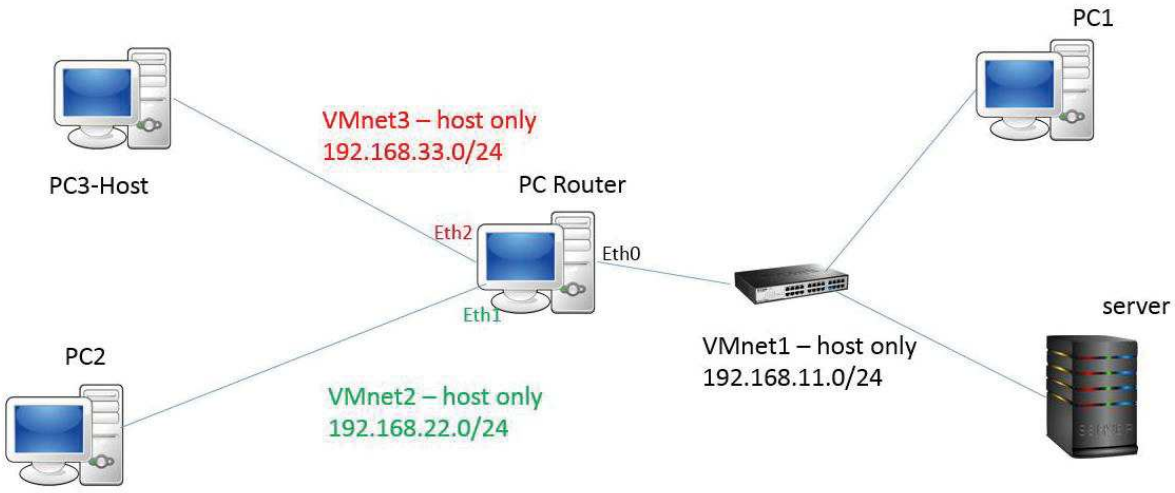
Part 2:

Activity objective:

The learning objective of this part is to understand how to install and configure a proxy/cache web Squid.

Visual Objective

You will use the following topology that you have used in part 1, in this part you have to use just PC Router (Proxy Server) and Server.



1. Install squid on PC Router (squid3 package).

   PS: you have to make all configuration in squid.conf, (make a copy squid.conf.default), you have to reload squid service after each configuration.

   To display the configuration in squid.conf without comment, you can use :
   cat squid.conf | grep -v '^#' | grep -v "^$"

2. Configure ***visible_hostname*** : proxy.lab.com

3. Configure the cache with the default values.
4. Configure the browser on PC3-Host to use the Proxy Server (use IP address on eth1)
5. Connect to server use http (provide screenshot)
   a. Copy the squid log that you have in accecc.log matches the http request and explain the squid result code.

6. Create an access list acl ***labnet*** to allow the LAN (the segment PC3-Host PC Router) to use proxy squid.

a. From PC3-Host, connect to server use http and https. (Provide screenshot)
b. From PC3-Host, connect to a www.google.fr (Provide screenshot)
c. Copy the squid log that you have in accecc.log matches previous requests and explain the squid result code.

7. Clear the cache in the browser and reconnect to server use http
   a. Copy the squid log that you have in accecc.log matches request, and explain the squid result code.

8. Configure the basic NCSA authentication on squid and use the authentication to the LAN (the segment PC3-Host PC Router)
   a. Refer to the following page to learn how to configure Squid proxy authentication server : http://www.cyberciti.biz/tips/linux-unix-squid-proxy-server-authentication.html
   b. Install apache2-utils package to use ***htpasswd***

9. Configure Squid to block the access to the domain .facebook.com
   a. Connect to www.facebook.com site (Provide screenshot)
   b. Copy the squid log that you have in accecc.log matches request, and explain the squid result code.
10. Create a PAC (proxy.pac) file that allow sending a traffic to server direct without Proxy and using Proxy to send traffic web to the Internet (Copy the function PAC in your report)
    a. Use proxy.pac in your browser.
    b. Capture and analyze with Wireshark an HTTP traffic on PC-Host (Provide a screen shot showing the browser getting the page and a sequence diagram that shows the basic HTTP exchanges between the client and servers for the tow requests:
       i. Connect to the server http (clear cache browser before your request)
       ii. Connect to www.google.fr

Aids:
$openssl genrsa -out <file_rsa.priv> -des3 <size>
Certificate Self-signed:
$openssl req -new -x509 -days <how_long_to_certify_for> –key <file_rsa.priv> -out <file_certificate> -config ./openssl.cnf -extensions <Extension_Section>
Certificate Request:
$openssl req -new -key <file_rsa.priv> -out <file_certificate_request> -config ./openssl.cnf
Sign a certificate by a CA:
$openssl ca –out <file_certificate> -config openssl.cnf -extensions <Extension_Section> -infiles <file_certificate_request>
Verify a certificate information:
$openssl x509 -in <file_certificate> -text -noout