LABO – Révisions

1 MATIERES A REVOIR

1.1. Commandes de gestion du réseau

Vous devez connaître les commandes de bases vues au **Labo 1** et savoir interpréter leurs résultats :

1. ipconfig (ifconfig ou ip addr sous Linux (n'oubliez pas sudo))
2. arp
3. ping
4. tracert (traceroute sous linux (sudo apt-get update && sudo apt-get install traceroute))
5. netstat (netstat -rn → affiche la table de routage; netstat -an → affiche les services actifs sur le réseau)

Vous devez pouvoir interroger un DNS avec la commande **dig** vue au **Labo4a** et savoir expliquer les réponses. (Sur Linux, dig mx nom_du_domaine → le noms des serveurs SMTP, dig nom_du_serveur → l'adresse IP)

Vous devez savoir utiliser l'outil netcat sous linux (nc) vu au **Labo5**, en mode client, mode serveur, sous le protocole TCP et UDP.

Vous devez également maîtriser la commande telnet vue au **Labo5,** savoir vous connecter sur un port particulier et savoir clôturer une session proprement (**CTRL+]** et **quit**)

1.2. Analyse sous Wireshark

Vous devez :

- Savoir réaliser une capture réseau sur une interface précise en fonction de l'adresse IP (voir Labo2, 3, 4 et 5).
- Savoir filtrer l'affichage sur une adresse IP, un protocole en particulier et/ou un numéro de port. ip.addr == 127.0.0.1 && tcp.port == 5000
- Repérer et pouvoir identifier précisément la couche du modèle TCP/IP et OSI sur base du protocole ou de l'informations (une MAC adresse, une IP, un port, ARP, DHCP, ICMP, un segment, un packet, une frame, une donnée, DNS, telnet, http, https, ssh, etc.)
- Savoir repérer et retrouver les données de sizing vous permettant d'expliquer le mécanisme d'encapsulation/décapsulation d'un modèle en couche.
- Pouvoir identifier le positionnement des flags TCP
- Pouvoir identifier une ouverture de session TCP, un envoi de donnée TCP et UDP, une fermeture de session TCP.
- Pouvoir compléter un schéma en flèche, en positionnant, les IP, le numéros de port, les flags et les numéros de séquence et d'accusé de réception sur un trafic TCP.
- Savoir identifier les données applicatives d'une communication dns, telnet, ssh, http, https.

1.3. Programmation des sockets

Savoir programmer une application simple client/serveur UDP et TCP dans le langage python sur base des sockets.

2 EXEMPLES DE QUESTIONS D'EXAMEN (LISTE NON EXAUTIVES)

2.1. Exercices sur les commandes de base

1. Quelle est l'adresse IP de votre système (sous windows et sous Linux) et quelle commande utilisez-vous pour obtenir l'information ?

2. Quelle est l'adresse IP de votre default route (sous windows et sous Linux) et quelle commande utilisez-vous pour obtenir l'information ?

3. Quelle est l'adresse mac liée à votre default route (sous windows et sous Linux) et quelle commande utilisez-vous pour obtenir l'information?

4. Précisez l'adresse IPv4 du premier serveur SMTP du domaine helmo.be ? Détaillez les commandes utilisées sur la Kali Linux pour obtenir ces informations.

5. Sur votre host (Windows ou MacOS), donnez la route vers le réseau virtuel NAT (192.168.254.0/24). Quelle commande utilisez-vous pour obtenir cette information ?

6. Quel commande utilisée pour tester qu'un système est connecté sur un réseau IPv4 et/ou IPv6 ?

7. Votre entreprise a été victime d'une attaque sur son DNS interne. Après restauration du service DNS, votre système Windows continue à donner des mauvaises IP.

    a. Quel est la raison de ces mauvaises réponses alors que le service est rétabli ?

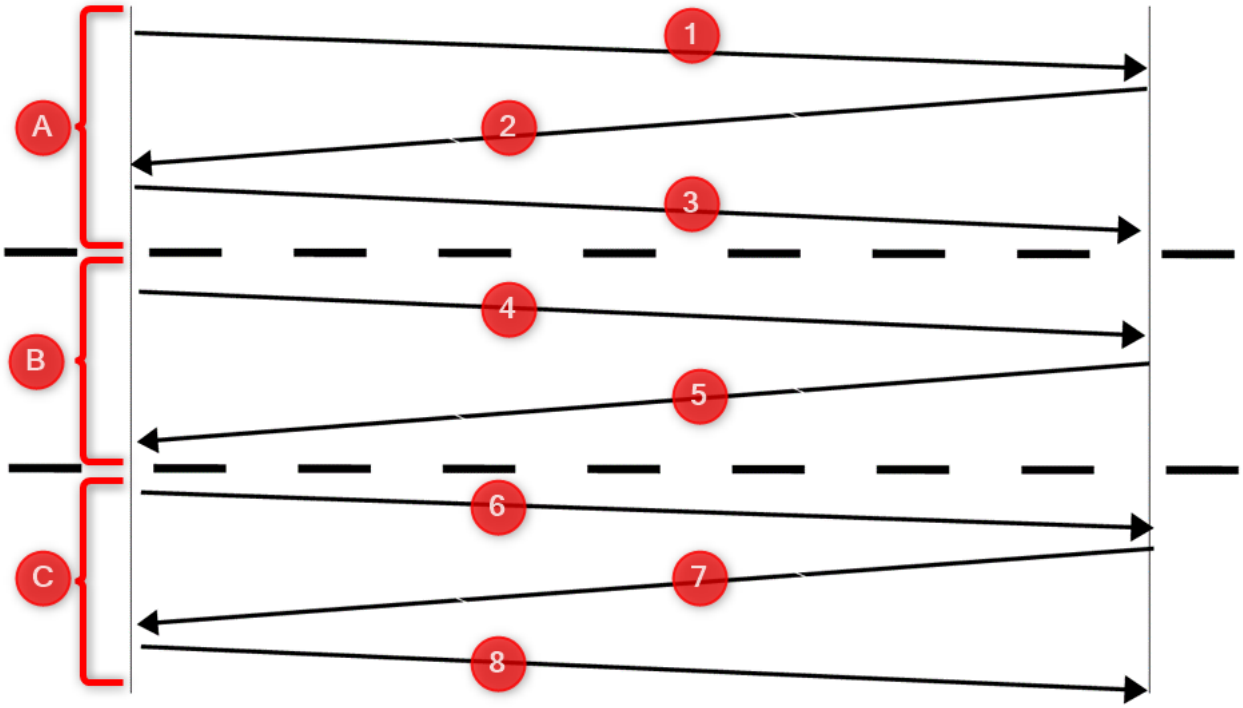
    b. Que faire en ligne de commande sous Windows pour visualiser la cause du problème ?

    c. Toujours en ligne de commande, que devez-vous exécuter pour résoudre la problématique ?

2.2. Exercices sur Wireshark

1. Sur votre VM Linux, réalisez la capture du trafic réseau sur l'adresse 127.0.0.1
2. A l'aide de la commande netcat (nc), exécutez un server en écoute sur le port 5000.
3. A l'aide de la commande **telnet**, connectez-vous à ce serveur sur le port 5000 et réalisez un seul envoi **Je suis *prénom NOM***, en plaçant votre nom et prénom.
4. Fermez la session telnet proprement.
5. Arrêtez la capture **WireShark** et sauvegardez votre capture sous **Exam01-NOM_Prénom.pcap** en plaçant votre nom et prénom.

Client Num port :	Serveur Num port :



A partir de votre capture de trafic réseau, détaillez le schéma en flèche en complétant chacun des points suivants :

1. flags :	num. de séq. (RAW et relatif) :
2. flags :	num. de séq. (RAW et relatif) : num. d'ACK (RAW et relatif) :
3. flags :	num. de séq. (RAW et relatif) : num. d'ACK (RAW et relatif) :
4. flags :	num. de séq. (RAW et relatif) : num. d'ACK (RAW et relatif) :
5. flags :	num. de séq. (RAW et relatif) : num. d'ACK (RAW et relatif) :
6. flags :	num. de séq. (RAW et relatif) : num. d'ACK (RAW et relatif) :
7. flags :	num. de séq. (RAW et relatif) : num. d'ACK (RAW et relatif) :
8. flags :	num. de séq. (RAW et relatif) : num. d'ACK (RAW et relatif) :

9. Dans une communication client/serveur TCP/IP, expliquez **en une phrase** à quoi correspond :

- la phase A dans le schéma ?
- la phase B dans le schéma ?
- la phase C dans le schéma ?

2.3. Questions développement de socket

1. Sur votre ordinateur, développez un script python **servUDP_echo.py**. Ce script devra utiliser le module socket pour exécuter un serveur UDP en écoute sur le port 8000.
2. Sur votre ordinateur, développez un script python **cliUDP_echo.py**. Ce script permettra la connexion vers votre serveur UDP développé au point 1.
3. Faite en sorte que les scripts communique en alternance les entrées clavier introduites par l'utilisateur. Exemple :

cliUDP_echo.py
```
Qui es-tu ?
Je suis ton serveur
Quel âge as-tu ?
J'ai moins d'une minute
quit
```

servUDP_echo.py
```
127.0.0.1 : Qui es-tu ?
Je suis ton serveur
127.0.0.1 : Quel âge as-tu ?
J'ai moins d'une minute
Aurevoir 127.0.0.1 !
```

4. Vos scripts doivent afficher Aurevoir et s'arrêter si le client ou le serveur envoi la commande **quit**
5. Déplacez votre code sur votre VM Linux et adaptez vos scripts pour réaliser la communication entre les deux machines.

6. Sur votre ordinateur, développez un script python **servTCP_echo.py**. Ce script devra utiliser le module socket pour exécuter un serveur TCP en écoute sur le port 9000.
7. Sur votre ordinateur, développez un script python **cliTCP_echo.py**. Ce script permettra la connexion vers votre serveur TCP développé au point 6.
8. Faite en sorte que les scripts communique en alternance les entrées clavier introduites par l'utilisateur. Exemple :

cliTCP_echo.py
```
Qui es-tu ?
Je suis ton serveur
Quel âge as-tu ?
J'ai moins d'une minute
quit
```

servTCP_echo.py
```
127.0.0.1 : Qui es-tu ?
Je suis ton serveur
127.0.0.1 : Quel âge as-tu ?
J'ai moins d'une minute
Aurevoir 127.0.0.1 !
```

9. Vos scripts doivent afficher Aurevoir et s'arrêter si le client ou le serveur envoi la commande **quit**
10. Déplacez votre code sur votre VM Linux et adaptez vos scripts pour réaliser la communication entre les deux machines.