Université de Strasbourg **Licence 2 Informatique**
UFR Mathématique Informatique **Année 2019-2020**

Réseaux IP

TP 1 - Analyse de trafic / Ethernet

1 Analyse de trafic avec Wireshark

Afin d'analyser les flux circulant sur un réseau, vous allez utiliser l'analyseur de trafic `Wireshark`. Avec cet outil il est possible de capturer des messages directement sur les interfaces réseaux du système utilisé et/ou de lire des captures pré-enregistrées.

`Wireshark`, comme l'outil `tcpdump` (son équivalent en mode texte), est basé sur la bibliothèque libpcap. Cette bibliothèque offre deux modes de fonctionnement :

- le mode normal : permet uniquement d'analyser les messages émis ou à destination du poste qui effectue l'analyse ;
- le mode promiscuous : permet d'analyser tout le trafic passant sur un lien, même si les messages ne sont pas à destination du poste qui effectue l'analyse.

Dans tous les cas, il faut avoir les droits d'un utilisateur privilégié pour analyser les trames circulant sur vos interfaces réseaux. Par défaut, un utilisateur simple ne pourra pas accéder à ces données sensibles.

La figure 1 présente un aperçu de l'interface de `Wireshark`. Une capture se décompose en 3 zones : la liste des messages capturés, les valeurs des différents champs, couche par couche, du message sélectionné et un dump hexadécimal du message sélectionné. La liste des messages capturés donne un aperçu rapide de chaque message : son numéro et sa date de réception (données locales à `Wireshark`), les adresses source et destination de la couche la plus haute trouvée dans le message, le protocole correspondant à la couche la plus haute trouvée dans le message, la longueur totale du message (tous les en-têtes compris) et des informations succinctes sur le contenu du message. Pour analyser sérieusement un message il faudra toujours se concentrer sur la seconde zone de l'interface.

Exercice : premières analyses

1. Lancez `Wireshark` en tant que super-utilisateur et démarrez une capture. Si vous ne disposez pas d'un compte administrateur, vous pouvez travailler avec la capture `trace-ex1.pcapng` fournie sur moodle.
2. Lancez un navigateur web et saisissez une adresse (URL) d'un site de votre choix.

   L'URL choisie est `http://dpt-info.u-strasbg.fr`

3. Lorsque la page est chargée, arrêtez la capture et sauvegardez cette dernière dans un fichier exploitable par `Wireshark`.

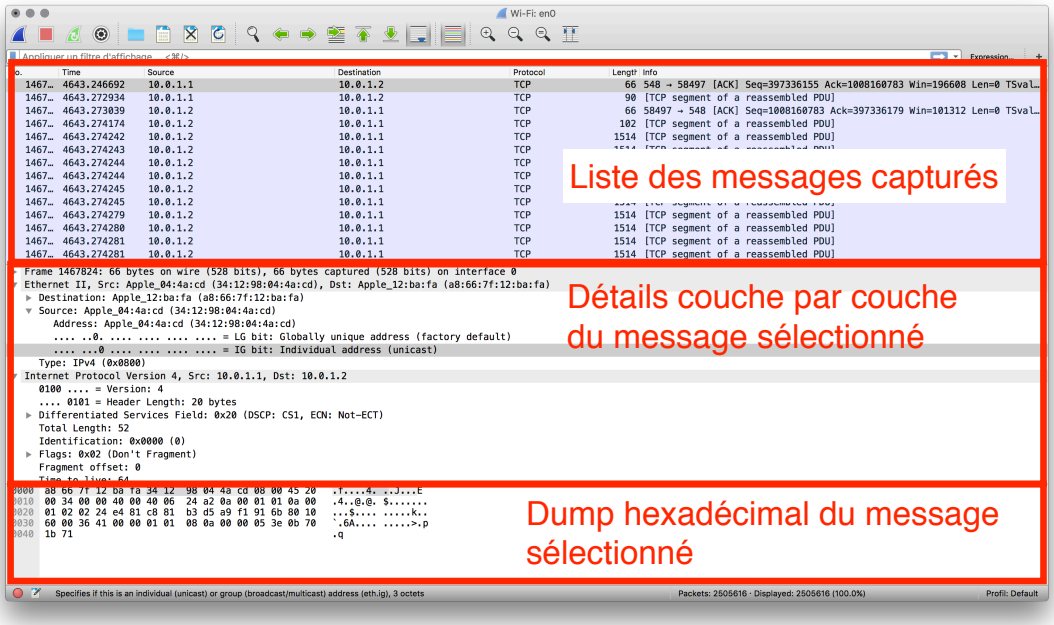

FIGURE 1 – Interface d'une capture dans Wireshark

4. À l'aide de votre capture, identifiez un message qui contient des données HTTP. Dans ce message, observez le modèle d'encapsulation et identifiez les protocoles utilisés sur chaque couche.

Le premier message qui contient des données HTTP est le numéro 4 (voir capture `trace-ex1.pcapng`). Dans ce message, le modèle d'encapsulation est TCP/IP qui comporte 5 couches :

— Couches physique et liaison → protocole Ethernet
— Couche réseau → protocole IP version 4
— Couche transport → protocole Transmission Control Protocol (TCP)
— Couche application → protocole HyperText Transfer Protocole (HTTP)

5. Identifiez le premier segment TCP de la procédure de connexion de votre poste vers le serveur web. Dans ce message, quelles sont les adresses de niveaux 2 et 3 utilisées ? À quels équipements appartiennent-elles ? Quelle est la valeur du champ `type` dans l'en-tête de niveau 2 ?

Le premier segment TCP envoyé vers ce serveur web est le numéro 1 dans la capture. Les adresses source et destination au niveau 2 sont respectivement `9c:b6:d0:8c:36:af` et `ec:88:92:32:82:73` Elles appartiennent respectivement à mon PC, source de la trame, et à un équipement intermédiaire (le routeur du réseau qui me sert de passerelle vers Internet), destinataire

de la trame. ***Le commutateur ne sera jamais la source ou le destinataire d'une trame Ethernet.***

Au niveau réseau, les adresses source et destination du paquet IP sont respectivement `192.168.43.6` et `130.79.44.193` Elles appartiennent respectivement à mon PC (source du paquet IP) et au destinataire final du paquet, c'est-à-dire le serveur web.

La valeur du champ type dans l'en-tête Ethernet vaut `0x0800` ce qui correspond au protocole IP version 4.

6. Identifiez le deuxième segment TCP transmis entre votre poste et le serveur web. Combien de temps s'est-il écoulé entre les captures des deux premiers segments TCP ? Dans ce message, relevez les paramètres suivants : adresses de niveaux 2 et 3 (source et destination), et numéros de ports source et destination (dans l'en-tête de niveau transport). Quelle est la longueur du paquet de niveau 3 ?

C'est le message numéro 2 qui nous intéresse ici. Il s'est écoulé $61,520$ ms entre les messages 1 et 2. Les adresses source et destination au niveau liaison sont respectivement `ec:88:92:32:82:73` et `9c:b6:d0:8c:36:af` (donc l'inverse du message précédent). Au niveau réseau, les adresses source et destination du paquet IP sont respectivement `130.79.44.193` et `192.168.43.6` (donc ici aussi l'inverse du message précédent). Les ports source et destination au niveau transport sont respectivement `80` (correspond à HTTP) et `33734` (port éphémère sur le PC). La taille du paquet est de 52 octets (en-tête IP comprise).

2 Manipulation sous Linux

Sous Linux les interfaces réseaux se manipulent avec la commande `ip`. Par exemple, la commande pour visualiser les interfaces réseaux présentes sur votre machine est la suivante :

```
[root@turing]# ip link show
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast state DOWN ...
    link/ether 00:80:c8:f8:4a:51 brd ff:ff:ff:ff:ff:ff
```

On peut voir dans cet exemple que le système comporte deux interfaces réseaux nommées `lo` et `eth0`. Vous pouvez notamment voir que :

- l'interface `lo` (qui est l'interface de boucle locale) est active (*UP*) et supporte des messages d'une taille maximale de 16436 octets (*mtu* pour *Maximum Transfer Unit*)
- l'interface `eth0` est inactive (*DOWN*), supporte des messages d'une taille maximale de 1500 octets et supporte les transmissions *broadcast* et *multicast*. Vous êtes également en mesure de visualiser les adresses unicast et broadcast associées à cette interface : `00:80:c8:f8:4a:51` et `ff:ff:ff:ff:ff:ff`

Exercice :

Un utilisateur ne dispose pas de connectivité réseau sur sa machine. Pour vérifier sa configuration il utilise la commande `ip` et obtient le résultat suivant. Identifiez le problème.

```
[root@turing]# ip link show
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN
    link/ether 00:80:c8:f8:4a:51 brd ff:ff:ff:ff:ff:ff
```

On peut voir que l'interface est activée de manière logicielle (`UP`) mais qu'au final elle reste dans le statut désactivé (`DOWN`) en raison de l'absence d'une connectivité physique (`NO-CARRIER`). Cela signifie qu'il n'y a pas de câble branché sur la carte, ou que ce dernier est défectueux, ou qu'il n'y a pas d'équipement active à l'autre extrémité du câble.

3 Forge de messages avec Scapy

Avant d'attaquer cette partie, vous devez installer le nécessaire pour monter un réseau virtuel avec l'outil `mininet`. Le réseau mis en place comportera 3 stations connectées via un commutateur. Suivez attentivement les étapes décrites dans la section suivante.

3.1 Pré-requis

1. Installez mininet sur votre machine.
   (a) Si vous disposez d'une machine sous Linux, mininet est probablement disponible via le gestionnaire de paquet :

   ```
   apt update
   apt install mininet
   ```

   (b) Sinon vous pouvez également l'installer via les sources :

   ```
   git clone https://github.com/mininet/mininet.git
   ```

   Ensuite suivez les indications contenues dans le fichier `INSTALL`.

   (c) Si vous ne disposez pas d'une machine sous Linux (sic), vous devez configurer une machine virtuelle sous Ubuntu 18.04 (la machine virtuelle fournie sur le site de mininet est trop ancienne et ne fonctionnera pas). Pour ce faire vous pouvez utiliser virtualbox :

   `https://virtualbox.org`

   et télécharger l'ISO d'Ubuntu depuis le site officiel :

   `https://ubuntu.com/#download`

   Une fois la machine virtuelle prête, installez dans cette dernière mininet via le gestionnaire de paquet :

   ```
   apt update
   apt install mininet
   ```

2. (optionnel) Si vous avez opté pour une installation locale (via apt ou les sources), il est possible que vous deviez (re)définir une variable d'environnement pour que le système puisse trouver les modules fournis par mininet :

```
find /usr/ -name "*mininet*"
=> affiche quelque chose comme /usr/lib/python2.7/dist-packages/mininet

PYTHONPATH=$PYTHONPATH:/usr/lib/python2.7/dist-packages/mininet
export PYTHONPATH
```

Cette étape est inutile si vous avez optez pour une machine virtuelle toute fraîche.

3. Installez le nécessaire (dans la VM ou directement sur votre poste suivant la méthode choisie en 1) pour créer des commutateurs Linux :

   ```
   apt install bridge-utils
   ```

4. Enfin, installez xterm, scapy et wireshark :

   ```
   apt install xterm scapy wireshark
   ```

5. Désactivez IPv6 (dans la VM ou directement sur votre poste suivant la méthode choisie en 1) afin que ce type de trafic ne pollue pas vos traces wireshark :

   ```
   sudo sysctl -w net.ipv6.conf.all.disable_ipv6=1
   sudo sysctl -w net.ipv6.conf.default.disable_ipv6=1
   ```

6. Récupérez le script python `topo.py` disponible sur moodle et lancez le via :

   ```
   sudo python topo.py
   ```

7. Dans la console mininet, ouvrez 3 terminaux afin d'avoir la main sur chacune des stations :

   ```
   mininet> xterm h1 h2 h3
   ```

8. Ouvrez wireshark en arrière plan sur chacune des stations (dans les terminaux que vous avez ouverts à l'étape précédente) et lancez une capture.

   ```
   wireshark &
   ```

Voila vous êtes prêts à continuer le tp !

3.2 Analyse de traces et table de commutation

Scapy est un outil écrit en Python qui permet de forger des messages. Nous allons utiliser cet outil pour construire des trames Ethernet depuis les stations et les transmettre sur le réseau afin d'analyser le trafic reçu et le comportement du commutateur.

1. Vérifiez l'état de la table de commutation du commutateur depuis un terminal de la machine hôte (votre propre machine ou la VM). Il ne doit pas y avoir d'entrée pour les stations 1, 2 et 3 :

   ```
   brctl showmacs s1
   ```

   On peut voir que le commutateur ne dispose en effet d'aucune entrée correspondant aux stations 1, 2 ou 3. Les adresses présentes correspondent aux différentes interfaces du commutateur (facilement vérifiable en tapant la commande `ip link show` sur la machine hôte de mininet). Attention, seuls les commutateurs administrables (sur lequels on peut se connecter pour les configurer) disposent d'adresses Ethernet sur les ports. Ce n'est pas le cas pour des commutateurs non administrable destinés au grand public.

```
port no  mac addr              is local?     ageing timer
2        2a:83:a8:44:a0:95     yes           0.00
2        2a:83:a8:44:a0:95     yes           0.00
3        ae:cc:cd:6b:a0:b0     yes           0.00
3        ae:cc:cd:6b:a0:b0     yes           0.00
1        ce:1a:38:b4:b8:53     yes           0.00
1        ce:1a:38:b4:b8:53     yes           0.00
```

2. Lancez scapy sur chaque station (depuis leur terminaux respectifs) :

   ```
   scapy
   ```

3. Créez une trame Ethernet depuis la station 1 à destination de l'adresse broadcast et envoyez cette dernière sur le réseau :

   ```
   msg = Ether()
   msg.dst = "ff:ff:ff:ff:ff:ff"
   sendp(msg)
   ```

4. À l'aide des captures wireshark (toujours en cours sur les stations 1, 2 et 3), identifiez les stations qui ont reçu cette trame. Quel est l'état de la table de commutation du commutateur après cette émission ?

   Cette trame étant à destination de l'adresse de diffusion Ethernet, le commutateur la retransmet sur tous ses ports actifs sauf le port source de la trame. Les stations 2 et 3 reçoivent donc cette trame (cf. traces wireshark trace-ex3-sta2.pcapng et trace-ex3-sta3.pcapng disponibles sur moodle). L'état de la table de commutation du commutateur contient désormais une entrée pour la station 1 :

   ```
   port no  mac addr              is local?     ageing timer
   1        00:00:00:00:00:01     no            2.48
   2        2a:83:a8:44:a0:95     yes           0.00
   2        2a:83:a8:44:a0:95     yes           0.00
   3        ae:cc:cd:6b:a0:b0     yes           0.00
   3        ae:cc:cd:6b:a0:b0     yes           0.00
   1        ce:1a:38:b4:b8:53     yes           0.00
   1        ce:1a:38:b4:b8:53     yes           0.00
   ```

5. Toujours depuis la station 1, envoyez une trame Ethernet destinée à la station 2. À l'aide des captures wireshark (toujours en cours sur les stations 1, 2 et 3), identifiez les stations qui ont reçu cette trame. Quel est l'état de la table de commutation du commutateur après cette émission ?

   Sur la station 1, dans scapy :

   ```
   msg = Ether()
   msg.dst = "00:00:00:00:00:02"
   sendp(msg)
   ```

La station 2 n'étant pas encore connue du commutateur, cette trame est retransmise sur tous les ports actifs du commutateur sauf le port source de la trame. Elle est donc reçue par les stations 2 et 3 (2e message dans les captures correspondantes). La table de commutation ne change pas, sauf mise à jour du temporisateur associé à l'entrée pour la station 1.

6. Envoyez maintenant une trame Ethernet depuis la station 2 vers la station 1. À l'aide des captures wireshark (toujours en cours sur les stations 1, 2 et 3), identifiez les stations qui ont reçu cette trame. Quel est l'état de la table de commutation du commutateur après cette émission ?

Sur la station 2, dans scapy :

```
msg = Ether()
msg.dst = "00:00:00:00:00:01"
sendp(msg)
```

La station 1 étant connue du commutateur, cette trame n'est retransmise que sur le port 1. Elle n'est donc reçue que par la station 1 (cf. trace trace-ex3-sta1.pcapng disponible sur moodle). Cette transmission permet au commutateur d'apprendre la localisation de la station 2 :

```
port no  mac addr              is local?     ageing timer
1        00:00:00:00:00:01     no            108.55
2        00:00:00:00:00:02     no            4.01
2        2a:83:a8:44:a0:95     yes           0.00
2        2a:83:a8:44:a0:95     yes           0.00
3        ae:cc:cd:6b:a0:b0     yes           0.00
3        ae:cc:cd:6b:a0:b0     yes           0.00
1        ce:1a:38:b4:b8:53     yes           0.00
1        ce:1a:38:b4:b8:53     yes           0.00
```

7. Envoyez maintenant une trame Ethernet depuis la station 3 vers la station 2 en usurpant l'identité de la station 1 (l'adresse source de la trame doit être celle de la station 1). À l'aide des captures wireshark, identifiez les stations qui ont reçu ce message. Quel est l'état de la table de commutation du commutateur après cette émission ?

Sur la station 3, dans scapy :

```
msg = Ether()
msg.src = "00:00:00:00:00:01"
msg.dst = "00:00:00:00:00:02"
sendp(msg)
```

Le commutateur reçoit cette trame depuis le port 3. La destination étant l'adresse de la station 2, elle n'est retransmise que sur le port 2 car une entrée dans la table de commutation correspond à cette adresse. C'est donc uniquement la station 2 qui reçoit cette trame. L'adresse source de la trame étant l'adresse de la station 1, le commutateur met à jour l'entrée correspondante. En effet, le commutateur n'a aucun moyen de savoir que la station 3 a usurpé l'adresse de la station 1. Dans l'état actuel de la

table, si une trame est envoyée à destination de la station 1, c'est uniquement la station 3 qui va la recevoir.

```
port no  mac addr                is local?     ageing timer
3        00:00:00:00:00:01       no            2.79
2        00:00:00:00:00:02       no            11.06
2        2a:83:a8:44:a0:95       yes           0.00
2        2a:83:a8:44:a0:95       yes           0.00
3        ae:cc:cd:6b:a0:b0       yes           0.00
3        ae:cc:cd:6b:a0:b0       yes           0.00
1        ce:1a:38:b4:b8:53       yes           0.00
1        ce:1a:38:b4:b8:53       yes           0.00
```