Introduction à l'analyse réseau

Philippe Latu
philippe.latu(at)inetdoc.net

http://www.inetdoc.net

Résumé

L'analyseur de trafic est un outil pédagogique essentiel pour comprendre les mécanismes de fonctionnement des protocoles de communication sur les réseaux contemporains. Ce document comprend deux parties. Dans un premier temps, on trouve une introduction à l'utilisation de l'analyseur Wireshark, le logiciel libre incontournable en la matière. Dans un deuxième temps, les travaux pratiques permettent de découvrir la richesse des informations fournies par cet analyseur.

Table des matières

1. Copyright et Licence 2
    1.1. Méta-information 2
2. Analyse avec Wireshark 2
    2.1. Quels sont les protocoles supportés ? 2
    2.2. Quels sont les médias supportés ? 2
    2.3. Comment accéder aux interfaces ? 2
3. Interface utilisateur 4
4. Capture d'une série de trame 5
5. Filtrage de l'affichage après capture 6
    5.1. Isoler une connexion TCP 6
    5.2. Syntaxe du filtrage à postériori 7
    5.3. Documentation de référence sur les filtres d'affichage 9
6. Analyse à distance 9
7. Travaux pratiques : navigation Web (HTTP) 11
    7.1. Protocoles étudiés 11
    7.2. Marche à suivre 12
    7.3. Analyse des protocoles 12
        7.3.1. Protocoles capturés 12
        7.3.2. Trame Ethernet, paquet IP et datagramme UDP 12
        7.3.3. Service DNS 13
        7.3.4. Connexion TCP 13
        7.3.5. Requête HTTP GET 14
        7.3.6. Réponse HTTP 14
8. Travaux pratiques : messages de contrôle internet (ICMP) 15
    8.1. Protocoles et outils étudiés 15
    8.2. Marche à suivre 15
    8.3. Analyse avec **ping** 16
        8.3.1. Protocoles capturés 16
        8.3.2. Message ICMP «Echo Request» 16
        8.3.3. Message ICMP «Echo Reply» 16
        8.3.4. Messages ICMP restants 16
    8.4. Analyse avec **(tcp)traceroute** 17
        8.4.1. Protocoles capturés 17
        8.4.2. Message UDP 17
        8.4.3. Message ICMP «Time Exceeded» 17
        8.4.4. Evolution du champ TTL 17
        8.4.5. Variantes 18
9. Documents de référence 18

1. Copyright et Licence


```
Copyright (c) 2000,2015 Philippe Latu.
Permission is granted to copy, distribute and/or modify this document under the
terms of the GNU Free Documentation License, Version 1.3 or any later version
published by the Free Software Foundation; with no Invariant Sections, no
Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included
in the section entitled "GNU Free Documentation License".
```

```
Copyright (c) 2000,2015 Philippe Latu.
Permission est accordée de copier, distribuer et/ou modifier ce document selon
les termes de la Licence de Documentation Libre GNU (GNU Free Documentation
License), version 1.3 ou toute version ultérieure publiée par la Free Software
Foundation ; sans Sections Invariables ; sans Texte de Première de Couverture,
et sans Texte de Quatrième de Couverture. Une copie de la présente Licence est
incluse dans la section intitulée « Licence de Documentation Libre GNU ».
```


1.1. Méta-information

Cet article est écrit avec DocBook[1] XML sur un système Debian GNU/Linux[2]. Il est disponible en version imprimable au format PDF : `intro.analyse.pdf`[3].

2. Analyse avec Wireshark

Avec Wireshark[4], il est possible de capturer des paquets directement sur les interfaces du système utilisé ou de lire des fichiers de captures sauvegardées. Wireshark supporte les formats de fichiers de capture les plus courants : libpcap/tcpdump, Sun's snoop/atmsnoop, LanAlyzer, MS Network Monitor, HPUX nettl, AIX iptrace, Cisco Secure IDS, etc.

2.1. Quels sont les protocoles supportés ?

La liste des protocoles supportés par Wireshark est considérable. Elle évolue avec chaque nouvelle version. La page Protocol Reference[5] fournit un classement par famille de tous les protocoles dont les champs sont interprétés. Il est aussi possible d'accéder à ces informations via le menu Help → Supported Protocols.

2.2. Quels sont les médias supportés ?

Le logiciel Wireshark permet l'analyse de transmissions réseau sur presque toutes les technologies. Les limitations d'utilisation sont plutôt dues au système d'exploitation sur lequel on exécute ce logiciel. Pour obtenir un état des possibilités d'analyse en fonction du système utilisé, il faut consulter la page : Network media specific capturing[6]

2.3. Comment accéder aux interfaces ?

Lorsque l'on exécute **wireshark** en tant qu'utilisateur normal, on ne peut accéder à la liste des interfaces en lançant l'opération Capture. Sur un système d'exploitation correctement administré, un utilisateur normal ne doit pas avoir accès aux interfaces sans conditions. Il existe plusieurs solutions pour donner un accès direct à la liste des interfaces physiques. En voici trois :

En mode super-utilisateur
: Partant d'une connexion avec un compte utilisateur normal, celui-ci est propriétaire exclusif de son écran (display). Il doit donc autoriser le super utilisateur à accéder à son écran à l'aide de la commande **xhost**, passer en connexion super-utilisateur avec la commande **su** puis exécuter l'application Wireshark.

[1] http://www.docbook.org
[2] http://www.debian.org
[3] http://www.inetdoc.net/pdf/intro.analyse.pdf
[4] http://www.wireshark.org/
[5] http://wiki.wireshark.org/ProtocolReference
[6] http://wiki.wireshark.org/CaptureSetup/NetworkMedia

```
$ xhost +local:
$ su
Password:
# wireshark &
```

En mode utilisateur avec gksu
: L'application gksu est un frontal graphique de la commande **su**. Elle permet, après la saisie du mot de passe super-utilisateur, d'exécuter une application en mode super-utilisateur. Dans notre cas, on accède directement aux interfaces physiques en mode graphique sans passer par une manipulation à la console. Ce mode opératoire est proposé par défaut avec toutes les distributions GNU/Linux actuelles.

En mode utilisateur avec sudo
: L'application sudo permet de déléguer les droits du super-utilisateur avec une granularité très fine. L'optique de l'analyse réseau étant un cas particulier de l'administration système, on se limitera à la présentation du fichier de configuration de l'application : `/etc/sudoers`.

```
# sudoers file.
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the man page for details on how to write a sudoers file.
#

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL) ALL

etu    ALL = NOPASSWD: /usr/bin/wireshark, /usr/bin/tshark
```

C'est à la dernière ligne que se situe la partie intéressante. L'utilisateur normal `etu` dispose, sur n'importe quel hôte géré par ce système (`ALL`), d'un accès super-utilisateur aux applications Wireshark et tshark sans avoir à saisir son mot de passe. Pour lancer l'application, il faut préciser l'appel à l'application sudo de la façon suivante :

```
$ sudo wireshark &
```

En mode utilisateur via les Linux Capabilities
: On débute par la création d'un groupe système dédié à la capture de trafic réseau.

```
# addgroup --system pcap
Adding group `pcap' (GID 136) ...
Done.
```

On ajoute un ou plusieurs utilisateur(s) au groupe système.

```
# adduser phil pcap
Adding user `phil' to group `pcap' ...
Adding user phil to group pcap
Done.
```

Attention ! Cette nouvelle attribution n'est valable qu'après une nouvelle authentification. Nous sommes encore dans le cas classique de création du contexte de travail utilisateur au moment de l'authentification.

On modifie les propriétés du programme dumpcap qui est chargé de la collecte du trafic réseau.

Avant modification du groupe propriétaire, le masque des permissions est le suivant :

```
# ls -lh `which dumpcap`
-rwxr-xr-x 1 root root 62K  4 mars  18:04 /usr/bin/dumpcap
```

On change le groupe propriétaire et on applique un nouveau masque de permissions. Une fois cette opération faite, les membres du groupe système `pcap` seront les seuls utilisateurs à pouvoir exécuter le programme en mode non privilégié.

```
# chgrp pcap /usr/bin/dumpcap
# chmod 750 /usr/bin/dumpcap
# ls -lh /usr/bin/dumpcap
-rwxr-x--- 1 root pcap 62K  4 mars  18:04 /usr/bin/dumpcap
```

On indique au gestionnaire de paquets Debian que ces nouvelles propriétés doivent être conservées lors des mises à jour à venir.

```
# dpkg-statoverride --add root pcap 750 /usr/bin/dumpcap
# dpkg-statoverride --list /usr/bin/dumpcap
root pcap 750 /usr/bin/dumpcap
```

On modifie le contexte de travail du programme `dumpcap`.

```
# setcap cap_net_raw,cap_net_admin=eip /usr/bin/dumpcap
# getcap /usr/bin/dumpcap
/usr/bin/dumpcap = cap_net_admin,cap_net_raw+eip
```

Les bits `eip` correspondent aux attributs effective, inheritable et permitted.

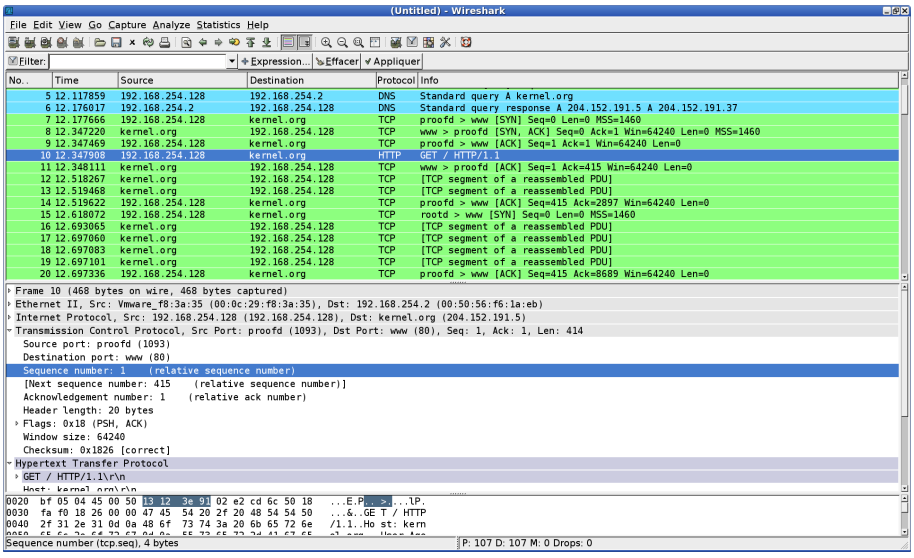
Avec l'attribut effective, le noyau ne vérifie pas si l'UID vaut 0 (mode privilégié) si le programme nécessite une opération en mode privilégié.

L'attribut inheritable transmet les aptitudes du processus actuel aux autres processus enfants.

L'attribut permitted indique que le processus peut utiliser les aptitudes étendues du noyau Linux.

La documentation sur les Linux Capabilities est disponible à partir de la page Not needing root to administer Linux[7].

3. Interface utilisateur



Capture d'écran Wireshark - vue complète[8]

L'interface de l'analyseur se décompose en plusieurs barres ou fenêtres :

Barre de menus
: On y retrouve la liste classique de menus. Voici une liste des fonctions remarquables accessibles à partir de ces menus.

- Le menu File sert à sauvegarder ou charger un fichier de capture réseau. Une capture peut très bien avoir été réalisée sur une sonde distante ou avec un autre outil et être analysée avec Wireshark à postériori.

- Le menu Capture sert à fixer les paramètres d'une nouvelle capture réseau. Voir Section 4, « Capture d'une série de trame ».

[7] http://sites.google.com/site/fullycapable/
[8] http://www.inetdoc.net/travaux_pratiques/intro.analyse/images/wireshark-complet.png

- Le menu Statistics sert à effectuer différents calculs sur les volumes de données et la répartition des protocoles.

Barre des icônes
: Cette barre regroupe tous les raccourcis sur les manipulations d'une capture.

Barre de filtrage
: Cette barre sert à saisir l'expression de filtrage à postériori d'une capture pour isoler tout ou partie d'un échange réseau.

Fenêtre contenant la liste des trames capturées
: Sur chaque ligne on retrouve :

  - le numéro du paquet,
  - son temps de capture,
  - sa source,
  - sa destination,
  - le protocole de plus haut niveau décodé,
  - le résumé des champs caractéristiques de ce protocole.

Fenêtre d'affichage de la pile des protocoles décodés pour la trame sélectionnée
: Avant toute opération de développement des champs d'un ou plusieurs protocoles, cette fenêtre donne la liste la pile de protocoles décodés allant du niveau physique (en haut) jusqu'au niveau le plus haut reconnu (en bas). Le protocole de niveau le plus haut reconnu apparaît est celui qui apparaît dans la colonne protocole de la Fenêtre contenant la liste des trames capturées.

  - La première ligne ou niveau Frame correspond à une pseudo couche physique. Comme il n'est pas possible de réaliser la capture directement à partir des composants électroniques qui pilotent l'interface réseau sans perturber le fonctionnement du système, l'opération a lieu au niveau liaison à l'aide de la bibliothèque libpcap.

    A ce niveau, les informations disponibles sont : la quantité de bits capturés et la date de capture.

  - La deuxième ligne correspond au niveau liaison. On y détaille le type et les champs de la trame et les adresses physiques.
  - La troisième ligne correspond au niveau réseau. On y détaille les champs du protocole réseau reconnu : adresses logiques et indicateurs d'état.
  - La quatrième ligne correspond au niveau transport. On y détaille les champs du protocole de transport reconnu : état de la connexion, numéros de ports utilisés et diverses options.
  - La cinquième ligne correspond au niveau application. On y trouve les données utilisateur.

  Pour le développement de chacun des champs de la trame, il faut cliquer sur le triangle situé à gauche au niveau de chaque couche.

Fenêtre d'affichage brut de la trame sélectionnée
: Cette fenêtre affiche tous les octets de la trame en hexadécimal.

4. Capture d'une série de trame

Après avoir lancé le logiciel Wireshark, suivre la séquence suivante pour capturer une série de 60 trames :

1. Sélectionner Capture puis Start.

2. La ligne Capture Filter, permet de préciser un filtrage à priori. La syntaxe de ce filtrage est identique à celle de la commande **tcpdump**. La documentation est disponible à partir des pages de manuels de cette commande : **`man tcpdump`**. Voici 3 exemples :

   - **`ip`** : en spécifiant le protocole réseau à analyser, on évite la capture des trames des autres protocoles de niveau réseau (IPX) et des protocoles de niveau liaison (STP, CDP, etc.).

   - **`host 192.168.0.1`** : en spécifiant l'adresse IP d'un hôte, on ne retient que le trafic émis et reçu par cette adresse.

   - **`host 192.168.0.1 and host 10.0.0.1`** : en spécifiant les adresses IP de 2 hôtes, on ne retient que le trafic entre ces 2 adresses.

   D'une façon plus générale, on peut combiner plusieurs critères avec les opérateurs logiques `and` et|ou `or`.

   - le type : `host`, `net` et `port`.

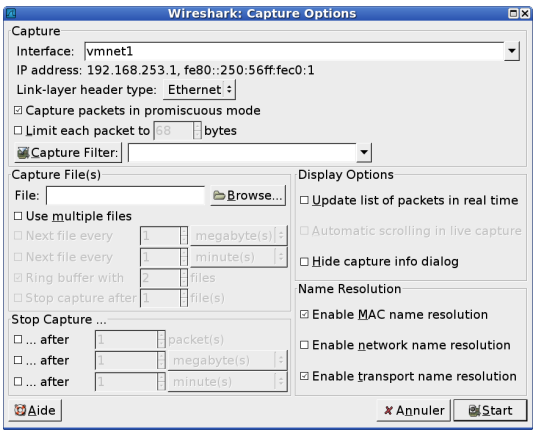
   - la direction : `src` et `dst`.

   - le protocole : `ether`, `fddi`, `tr`, `ip`, `ip6`, `arp`, `rarp`, `decnet`, `tcp` et `udp`.

   En règle générale, il faut limiter au maximum de filtrage à priori de façon à disposer du maximum d'information pour l'analyse. La syntaxe de la Section 5, « Filtrage de l'affichage après capture » offre beaucoup plus de possibilités.

3. La rubrique Stop Capture permet de fixer plusieurs critères d'arrêt en fonction du nombre de trames et|ou du volume de données capturées.

4. Clicker sur le bouton Valider pour lancer la capture.



Paramètres de capture - vue complète[9]

5. Filtrage de l'affichage après capture

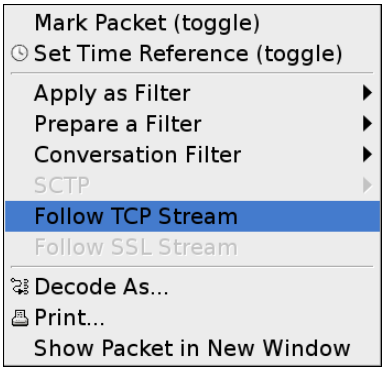
Le filtrage à postériori est certainement l'étape la plus importante dans l'analyse réseau. C'est cette opération qui permet d'isoler l'information pertinente. La granularité de la syntaxe de filtrage disponible avec Wireshark est très importante. Il est possible de retenir un champ unique parmi les 820 protocoles supportés. Voici quelques exemples de filtrage allant du plus général au plus détaillé.

5.1. Isoler une connexion TCP

Après avoir réalisé une capture, il est possible d'isoler une connexion TCP en repérant son établissement (le début) et sa libération (la fin). En cliquant sur le bouton droit de la souris après avoir

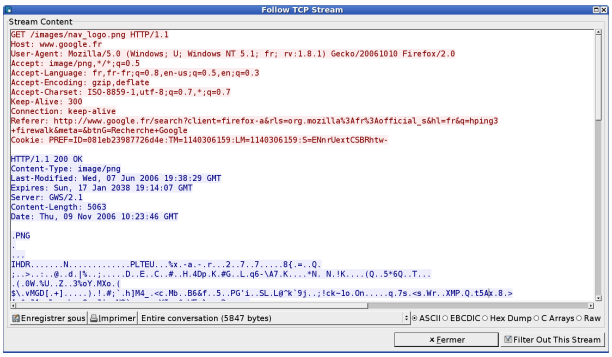
[9] http://www.inetdoc.net/travaux_pratiques/intro.analyse/images/wireshark-capture.png

sélectionné n'importe quelle trame appartenant à la connexion à isoler, il faut valider l'option Follow TCP Stream.



Isoler une connexion TCP - vue complète[10]

A la suite de cette opération, Wireshark ouvre une nouvelle fenêtre contenant les données vues de la couche transport.



Données vues de la couche transport - vue complète[11]

5.2. Syntaxe du filtrage à postériori

Comme indiqué ci avant, la granularité de la syntaxe de filtrage est très importante. Elle peut donc s'avérer très complexe à manipuler. Wireshark offre plusieurs solutions pour rendre l'apprentissage de cette syntaxe interactif.

Tout d'abord, l'opération précédente de filtrage simplifié (voir Section 5.1, « Isoler une connexion TCP ») n'était qu'un cas particulier de saisie interactive de filtre de capture. En sélectionnant l'option Follow TCP Stream, on a «saisi» un filtre avec la syntaxe suivante :

```
(ip.addr❶ eq 192.168.1.9❷ and ip.addr eq 80.247.225.35) \
and❸ (tcp.port❹ eq 32783❺ and tcp.port eq 80)
```

Cette expression est extraite de la Barre de filtrage. Elle doit tenir sur une ligne unique quel que soit sa longueur.

❶ **`ip.addr`** : sélection d'une adresse IP.

❷ **`eq 192.168.1.9`** : valeur particulière d'adresse IP. L'opérateur `eq` correspond à un test d'égalité. Il est aussi possible d'utiliser la syntaxe du langage C pour les tests :

- `==` : égalité,
- `!=` : différence,
- `>=` : supérieur ou égal,
- `<=` : inférieur ou égal.

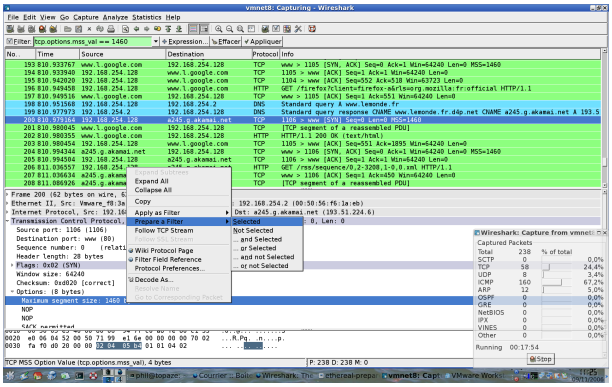
[10] http://www.inetdoc.net/travaux_pratiques/intro.analyse/images/wireshark-tcpstream.png
[11] http://www.inetdoc.net/travaux_pratiques/intro.analyse/images/wireshark-tcpdata.png

❸ Les opérateurs logiques tels que `and` et|ou `or` associés aux parenthèses servent à composer des expressions de sélection précises.
❹ **`tcp.port`** : sélection d'un numéro de port du protocole TCP de la couche transport.
❺ **`eq 32783`** : valeur particulière de port TCP. La syntaxe de test est identique pour tous les champs des différents protocoles reconnus.

La construction interactive des filtres d'affichage peut se faire à l'aide de la souris. Voici 2 exemples «simplistes» :

Option TCP MSS

Admettons que l'on veuille repérer toutes les trames capturées dans lesquelles l'option `MSS` (Maximum Segment Size) apparaît. On développe alors l'en-tête TCP d'un paquet correspondant à une demande de connexion pour faire apparaître cette option. En cliquant sur le bouton droit de la souris on accède au menu Prepare a Filter.



Préparation d'un filtre d'affichage à la souris - vue complète[12]

L'expression préparée apparaît dans la champ de la Barre de filtrage :

```
tcp.options.mss_val == 1460
```

Supposons maintenant que l'on veuille afficher toutes les trames ayant cette option indépendamment de sa valeur. Il suffit alors de supprimer le test :

```
tcp.options.mss_val
```

Fragmentation IP

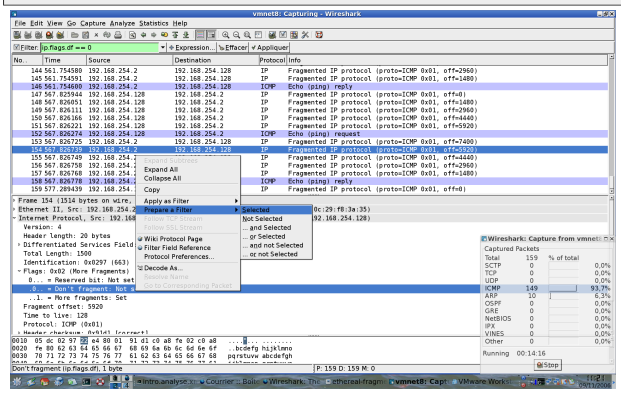
Admettons que l'on veuille observer la fragmentation IP en repérant les champs correspondants de l'en-tête des paquets IP. Tout d'abord, il faut «provoquer» la fragmentation IP artificiellement. On utilise deux hôtes avec chacun une interface Ethernet et un hub. En réduisant la taille maximum des données transmises par paquet (Maximum Transmit Unit) sur l'interface Ethernet d'un hôte, on observe plus facilement les effets de la fragmentation.

[12] http://www.inetdoc.net/travaux_pratiques/intro.analyse/images/wireshark-preparefilter.png

```
  __________
 |_=_=_=_=_// Hub
   |     |
  /       \_______
  |.....          \
.------,~          |.....
|Hote_A|'        .------,~
|      ||        |Hote_B|'
\------ /        |      ||
 ======/         \------ /
192.168.254.128   ======/
                 192.168.254.2

Hote_A # ifconfig eth0 mtu 256
Hote_A # ping -s 128 -c 5 192.168.254.2
PING 192.168.254.2 (192.168.254.2) 128(156) bytes of data.
136 bytes from 192.168.254.2: icmp_seq=1 ttl=64 time=0.591 ms
136 bytes from 192.168.254.2: icmp_seq=2 ttl=64 time=0.528 ms
136 bytes from 192.168.254.2: icmp_seq=3 ttl=64 time=0.554 ms
136 bytes from 192.168.254.2: icmp_seq=4 ttl=64 time=0.545 ms
136 bytes from 192.168.254.2: icmp_seq=5 ttl=64 time=0.546 ms

--- 192.168.254.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3999ms
rtt min/avg/max/mdev = 0.528/0.552/0.591/0.036 ms
Hote_A # ping -s 8192 -c 5 192.168.254.2
PING 192.168.254.2 (192.168.254.2) 8192(8220) bytes of data.
8200 bytes from 192.168.254.2: icmp_seq=1 ttl=64 time=15.4 ms
8200 bytes from 192.168.254.2: icmp_seq=2 ttl=64 time=15.4 ms
8200 bytes from 192.168.254.2: icmp_seq=3 ttl=64 time=15.4 ms
8200 bytes from 192.168.254.2: icmp_seq=4 ttl=64 time=15.4 ms
8200 bytes from 192.168.254.2: icmp_seq=5 ttl=64 time=15.4 ms

--- 192.168.254.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 15.444/15.462/15.481/0.079 ms
```

On observe ensuite le résultat sur l'affichage des trames capturées. La syntaxe du filtre est :

```
ip.flags.df == 0
```



Préparation d'un filtre d'affichage à la souris - vue complète[13]

Connaissant maintenant la syntaxe d'identification de la fragmentation IP, il sera toujours possible d'appliquer le même filtre sur une capture beaucoup plus importante en volume.

5.3. Documentation de référence sur les filtres d'affichage

La documentation sur l'ensemble des champs des protocoles reconnus utilisables dans les expressions de filtres d'affichage est disponible à l'adresse : Display Filter Reference[14].

6. Analyse à distance

Lorsque l'on exploite une infrastructure de serveurs avec plusieurs périmètres réseau cloisonnés, il est fréquent de devoir procéder à des captures réseau à distance. De plus, la plupart des serveurs récents

[13] http://www.inetdoc.net/travaux_pratiques/intro.analyse/images/wireshark-fragment.png
[14] http://www.wireshark.org/docs/dfref/

sont des lames qui n'ont ni clavier ni écran. Voici donc un exemple de scénario capture réseau réalisée sur un hôte distant exploitée ensuite sur un poste de travail ayant une interface graphique.

Dans la suite de copies d'écran suivante, on considère les éléments suivants :

- Le poste de travail sur lequel l'analyse est effectuée en mode graphique après collecte du fichier de capture est appelé `<my_laptop.myothernet>`.
- Le serveur lame sans écran ni clavier sur lequel la capture réseau est réalisée est appelé `<my_distant_server.mynet>`. On y accède via une console sécurisée SSH.
- On suppose que les deux hôtes ont un compte utilisateur `me`. Le compte utilisateur sur le serveur doit disposer des droits nécessaire à la capture de trames sur les interfaces réseau du serveur. Ces droits sont gérés avec sudo.
- On utilise l'application tshark qui permet d'exécuter l'analyse réseau directement à la console sans recours à une interface graphique. Cette application est fournie par le paquet Debian du même nom. Voir le résultat de la commande **`$ apt-cache show tshark`** pour obtenir les informations sur ce paquet.
- Les indications données ci-dessous ne peuvent se substituer aux pages de manuels de l'application. Il est vivement conseillé de les consulter pour adapter l'analyse réseau à ses besoins : **`man tshark`**.

Connexion au serveur depuis le poste de travail
: Comme indiqué ci-avant, on accède au serveur via une console sécurisée SSH. À partir le Windoze, l'outil `putty` permet d'effectuer la même opération.

```
me@<my_laptop>:~$ ssh me@<my_distant_server.mynet>

Linux <my_distant_server> 2.6.15 #1 SMP Mon Mar 13 14:54:19 CET 2006 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
No mail.
Last login: Tue Mar 21 10:45:38 2006 from <my_laptop.myothernet>

me@<my_distant_server>:~$
```

Lancement de la capture réseau dans un nouveau shell
: Un utilisateur «normal» n'ayant pas les droits suffisants pour accéder directement aux interfaces réseau, on doit lancer l'analyseur de réseau via sudo : **`$ sudo tshark`**.

  On lance cette commande dans un nouveau shell en ajoutant le symbole `&` à la fin de la ligne. De cette façon, on conserve la possibilité de lancer d'autres commandes sur la console obtenue lors de la connexion au serveur.

```
me@<my_distant_server>:~$ sudo tshark -q -i _eth0 -w distant.cap \
  -a filesize:4096 tcp and ! host <my_laptop.myothernet> &
```

  - L'option `-q` rend la capture «silencieuse». Il s'agit surtout de supprimer l'affichage du compte des paquets enregistrés pendant la capture. Cet affichage est gênant si l'on souhaite conserver la console pour effectuer d'autres manipulations en cours de capture.
  - L'option `-i _eth0` désigne l'interface réseau sur laquelle la capture est réalisée.
  - L'option `-w distant.cap` désigne le fichier dans lequel les paquets capturés sont enregistrés. Sans spécification du format de fichier avec l'option `-F`, les paquets capturés sont enregistrés directement (mode raw).
  - L'option `-a filesize:4096` donne le critère d'arrêt de l'enregistrement. Ici, le critère retenu est la taille du fichier de capture. Cette taille est comptabilisée en multiple du kilooctet (1024 octets) ; soit 4096ko dans cet exemple.

- Les options suivantes correspondent au filtrage à priori des paquets à enregistrer. On spécifie le protocole de transport `tcp` et on n'enregistre pas les paquets de l'hôte qui à ouvert la console sécurisée : `! host <my_laptop.myothernet>`. Sans cette dernière précaution, l'enregistrement ne contiendra pratiquement que les échanges SSH. Ces échanges sont sans intérêt puisqu'ils correspondent aux communications entre les deux hôtes utilisés pour l'analyse distante.

«Initiation» du trafic réseau à capturer.

Cette commande n'est qu'un prétexte pour remplir le fichier de capture. Avec le téléchargement d'une image des sources du noyau Linux, on est sûr de faire transiter un volume suffisant ;-).

```
me@<my_distant_server>:~$ wget \
  http://kernel.org/pub/linux/kernel/v2.6/linux-2.6.16.tar.bz2
--11:14:29--  http://kernel.org/pub/linux/kernel/v2.6/linux-2.6.16.tar.bz2
           => `linux-2.6.16.tar.bz2'
Résolution de kernel.org... 204.152.191.5, 204.152.191.37
Connexion vers kernel.org|204.152.191.5|:80...connecté.
requête HTTP transmise, en attente de la réponse...200 OK
Longueur: 40 845 005 (39M) [application/x-bzip2]

100%[===============//====================>] 40 845 005   296.19K/s    ETA 00:00

11:16:58 (292.09 KB/s) - « linux-2.6.16.tar.bz2 » sauvegardé [40845005/40845005]
```

Fin de la capture et visualisation du fichier

Comme indiqué ci-avant, l'enregistrement s'arrête lorsque le fichier atteint la taille de 4096ko.

```
[1]+  Done  sudo tshark -q -i _eth0 -w distant.cap \
                -a filesize:4096 tcp and ! host <my_laptop.myothernet>

me@<my_distant_server>:~$ ls -lAh
-rw-------  1 root latu   4,1M 2006-03-21 11:14 distant.cap
-rw-r--r--  1 latu latu    39M 2006-03-20 07:22 linux-2.6.16.tar.bz2

me@<my_distant_server>:~$ sudo chmod 640 distant.cap

me@<my_distant_server>:~$ exit
logout
Connection to <my_distant_server.mynet> closed.
```

L'enregistrement sur fichier ayant été réalisé avec l'identité du super-utilisateur via la commande **sudo**, il faut changer le masque des permissions de ce fichier ou son propriétaire. Dans cet exemple, c'est le masque des permissions d'accès qui a été étendu pour que l'utilisateur normal puisse lire le fichier de capture et le transférer sur son poste de travail.

Récupération du fichier de capture sur le poste de travail

```
me@<my_laptop>:~$ scp me@<my_distant_server.mynet>:~/distant.cap .
distant.cap                                   100% 4097KB 682.8KB/s   00:06

me@<my_laptop>:~$ wireshark -r distant.cap
```

La commande **scp** illustre le transfert du fichier de capture réseau via SSH. On peut effectuer la même opération à partir de Windoze avec l'outil `WinSCP`.

Enfin, il est possible de lire le fichier de capture directement au lancement de l'analyseur réseau avec l'option `-r`.

7. Travaux pratiques : navigation Web (HTTP)

7.1. Protocoles étudiés

- Adressage matériel (MAC|Ethernet) et logique (IP).
- Requête et réponse du service de noms de domaines (DNS).
- Établissement, maintien et libération de connexion TCP : procédure en trois étapes, numéros de séquence et d'acquittement.

- Requête et réponse HTTP.

7.2. Marche à suivre

1. Lancer Wireshark.
2. Lancer la capture des trames sans restrictions d'adresses, de protocoles ou de volume.
3. Lancer un navigateur Web et saisir une adresse de site (URL) de votre choix.
4. Une fois la page complètement chargée, arrêter la capture. Sauvegarder un fichier de capture.
5. Passer aux questions suivantes.

Suivant le contexte de connexion, le volume d'information capturé varie énormément : connexion DSL, réseau local commuté ou non, multiplicité des protocoles réseau, etc. Il est cependant préférable d'effectuer la première capture sans aucune restriction à priori de façon à avoir une image exacte du trafic. Si l'information utile est vraiment noyée dans du «bruit», il est toujours possible de reprendre la capture avec un filtre ; voir Section 4, « Capture d'une série de trame ».

7.3. Analyse des protocoles

Pour répondre aux questions suivantes, utiliser le résultat de la capture issue de l'étape précédente ou charger un fichier de capture.

7.3.1. Protocoles capturés

Q1. Quels sont les protocoles indiqués dans la colonne Protocol de la fenêtre de liste des trames capturées ?

Confirmer que la capture contient bien les protocoles DNS, TCP et HTTP.

7.3.2. Trame Ethernet, paquet IP et datagramme UDP

Analyser la trame correspondant au premier message DNS émis par le client Web.

Q2. Quels sont les adresses (MAC|Ethernet) et IP du client ?

Q3. Quel est le contenu du champ type de la trame Ethernet ?

Q4. Quelles sont les adresses destination (MAC|Ethernet) et IP ?

Q5. À quelles machines correspondent ces adresses ?

Analyser l'en-tête IP du premier message DNS émis par le client Web.

Q6. Quelle est la taille de l'en-tête ? Quelle est la longueur totale du paquet ?

Q7. Repérer le champ «type de protocole» dans l'en-tête. Quel est le numéro et le type de protocole présent dans les données du paquet ?

Analyser l'en-tête UDP du premier message DNS émis par le client Web.

Q8. Quels sont les numéros de ports du client et du serveur ? Quelles sont les particularités de ces valeurs ? Quel est le protocole de couche application présent dans les données du message ?

Q9. Quelle est la valeur indiquée dans le champ longueur de l'en-tête UDP ? Est-ce qu'elle correspond à l'information donnée dans l'en-tête du paquet IP ?

Faire un croquis des piles de protocoles des couches physique à application pour le client et le serveur ; identifier les unités de données de protocoles (PDUs) et les communications de bout en bout.

7.3.3. Service DNS

Analyser le message de requête DNS émis par le client Web.

Q10. Quel est le champ qui indique si le message est une requête ou une réponse ?

Q11. Quelle est l'information transportée dans le corps de la requête ? Identifier le type et la classe de la requête.

Q12. Quel est l'identificateur de transaction de la requête ?

On considère maintenant la réponse à la requête précédente.

Q13. Quelles devraient être les adresses (MAC|Ethernet) et IP de ce paquet ?

Vérifier que les adresses attendues sont présentes.

Q14. Quelle est la taille du paquet IP ; du message UDP ? Cette taille est-elle plus importante que celle du paquet de requête ?

Q15. Quel est l'identificateur de transaction de la réponse ? Est-ce qu'il correspond à la requête ?

Q16. Combien de réponses sont disponibles dans le message de réponse ? Comparer les réponses et leurs valeurs TTL (Time-to-live).

7.3.4. Connexion TCP

Identifier la trame qui correspond au premier segment TCP dans la procédure en trois étapes (three ways handshake) qui initie la connexion entre le client et le serveur HTTP.

Q17. Quelles sont les adresses (MAC|Ethernet) et IP attendues pour cette trame ? Quels sont les valeurs des champs `type` et `protocole` respectivement attendus pour cette trame et ce paquet ?

Vérifier que ces champs et adresses correspondent.

Q18. Expliquer les valeurs des adresses destination (MAC|Ethernet) et IP ? À quels hôtes correspondent ces adresses ?

Q19. Identifier les numéros de ports utilisés par le client. Pourquoi ces valeurs sont-elles utilisées ?

Q20. Quelle est la longueur du segment TCP ?

Q21. Quel est le numéro de séquence initial (Initial Sequence Number ou ISN émis par le client vers le serveur ? Quelle est la taille de fenêtre initiale ? Quelle est la taille maximale de segment (Maximum Segment Size ou MSS) ?

Q22. Trouver la valeur hexadécimale de l'octet qui contient l'indicateur d'état SYN ?

Identifier la trame qui correspond au second segment TCP dans la procédure en trois étapes (three ways handshake).

Q23. Combien de temps s'est écoulé entre la capture du premier et du second segment TCP ?

Q24. Relever les valeurs des champs suivants de cette trame :

- Adresses MAC source et destination de la trame Ethernet.
- Adresses source et destination du paquet IP.
- Numéros de séquence et d'acquittement du segment TCP.
- Valeurs des indicateurs d'état.

Vérifier que tout correspond aux valeurs attendues.

Q25. Quelle est la longueur du segment TCP ?

Q26. Quel est le numéro de séquence initial (Initial Sequence Number ou ISN émis par le serveur vers le client ? Quelle est la taille de fenêtre initiale ? Quelle est la taille maximale de segment (Maximum Segment Size ou MSS) ?

Identifier la trame qui correspond au dernier segment TCP dans la procédure en trois étapes (three ways handshake).

Q27. Combien de temps s'est écoulé entre la capture du second et du troisième segment TCP ? Comparer cette valeur avec celle relevée entre le premier et le second segment et expliquer la différence.

Q28. Relever les valeurs des champs suivants de cette trame :

- Numéros de séquence et d'acquittement du segment TCP.
- Valeurs des indicateurs d'état.
- Tailles de fenêtre.

Vérifier que tout correspond aux valeurs attendues.

Q29. Quelle est la longueur du segment TCP ?

7.3.5. Requête HTTP GET

Identifier la trame qui correspond au message HTTP `GET`.

Q30. Quelles sont les valeurs des numéros de séquence et d'acquittement de l'en-tête TCP ?.

Vérifier que tout correspond aux valeurs attendues.

Q31. Quels sont les indicateurs d'état actifs de l'en-tête TCP ? Expliquer pourquoi.

Q32. Quelles sont les longueurs de l'en-tête et de la «charge» du message TCP ?

On considère maintenant le contenu du message HTTP `GET`.

Q33. Comparer le texte décodé dans la fenêtre d'affichage de la pile de protocoles avec le contenu de la fenêtre d'affichage brut.

Q34. Compter le nombre d'octets du message et vérifier que ce nombre correspond au champ longueur de l'en-tête TCP.

Q35. Quel est le prochain numéro de séquence attendu dans le message suivant émis par le serveur HTTP ?

7.3.6. Réponse HTTP

Q36. Combien de temps s'est écoulé entre la capture du message `GET` et la capture du message de réponse correspondant ?

Q37. Déterminer si le serveur répond avec un message HTTP ou un segment TCP `ACK` ?

Q38. Quel est le numéro de séquence émis par le serveur HTTP ? Est-ce qu'il correspond à la valeur attendue ?

On considère maintenant l'en-tête du message réponse HTTP.

Q39. Quelle est la longueur de la «charge» indiquée dans l'en-tête TCP ?

Q40. Quels sont les indicateurs d'état actifs de l'en-tête TCP ? Expliquer pourquoi.

Q41. Quel est le prochain numéro de séquence attendu dans le message suivant émis par le client ?

On considère maintenant le corps du message réponse HTTP.

Q42. Quel est le code dans le message de réponse ?

Q43. Sélectionner ce code avec la souris dans la fenêtre d'affichage de la pile de protocoles et comparer avec ce qui est affiché sur la page du navigateur Web.

Cette opération revient à suivre la démarche présentée dans la Section 5.1, « Isoler une connexion TCP ».

8. Travaux pratiques : messages de contrôle internet (ICMP)

8.1. Protocoles et outils étudiés

- Internet Control Message Protocol ou ICMP ; messages de type : `Echo`, `Echo Reply` et `Time Exceeded`.
- Internet Protocol ou IP ; champ de l'en-tête IP : `Time to Live`.
- Commande **ping**.
- Commandes **traceroute** et **tcptraceroute**.

8.2. Marche à suivre

Commande ping

1. Lancer Wireshark.
2. Lancer la capture des trames sans restrictions d'adresses, de protocoles ou de volume.
3. Lancer une console et taper une commande du type **`ping -c10 www.phrack.org`**. L'option `-c10` limite le nombre de requêtes ICMP à 10. Bien sûr, le choix de l'adresse à contacter est totalement libre.
4. Arrêter la capture lorsque l'invite de commande réapparaît à la console.
5. Sauvegarder le fichier de capture.

Commande traceroute

1. Lancer Wireshark.
2. Lancer la capture des trames sans restrictions d'adresses, de protocoles ou de volume.
3. Lancer une console et taper une commande du type **`traceroute www.phrack.org`**. Bien sûr, le choix de l'adresse à contacter est totalement libre.
4. Arrêter la capture lorsque l'invite de commande réapparaît à la console.
5. Sauvegarder le fichier de capture.

La plage de ports UDP utilisée par défaut par la commande **traceroute** est de plus en plus fréquemment bloquée par les équipements d'interconnexion. Il est alors utile d'envisager l'emploi de la commande **tcptraceroute** avec laquelle on peut fixer les ports source et destination.

Commande tcptraceroute

1. Lancer Wireshark.
2. Lancer la capture des trames sans restrictions d'adresses, de protocoles ou de volume.
3. Lancer une console et taper une commande du type **`tcptraceroute -p 1024 www.phrack.org 80`**. Bien sûr, le choix de l'adresse à contacter est totalement libre.
4. Arrêter la capture lorsque l'invite de commande réapparaît à la console.
5. Sauvegarder le fichier de capture.

8.3. Analyse avec ping

Pour répondre aux questions suivantes, utiliser le résultat de la capture issue de l'étape précédente ou charger un fichier de capture.

8.3.1. Protocoles capturés

Q44. Quels sont les protocoles indiqués dans la colonne Protocol de la fenêtre de liste des trames capturées ?

Il est probable que les paquets ICMP soient précédés d'un jeu de question/réponse DNS.

Q45. Relever l'adresse IP renvoyée avec la réponse DNS.

8.3.2. Message ICMP «Echo Request»

Étude du paquet IP qui correspond au premier message ICMP `Echo Request`.

Q46. Quelle est l'adresse IP destination du paquet ? Quelle est la valeur du champ `Protocol Type` ? Quelle est la valeur du champ `Time to Live` ?

Étude du message ICMP.

Q47. Quel est le type de message ICMP ? Quel est l'identificateur de message ? Quel est le numéro de séquence ?

Q48. Sélectionner à la souris les octets de données du message de requête. Comparer ces données avec celles affichées dans la fenêtre d'affichage brut.

8.3.3. Message ICMP «Echo Reply»

Étude du paquet IP qui correspond au premier message ICMP `Echo Reply`.

Q49. Quelle sont les adresses IP source et destination du paquet ? Quelle est la valeur du champ `Protocol Type` ? Quelle est la valeur du champ `Time to Live` ?

Étude du message ICMP.

Q50. Quel est le type de message ICMP ? Comparer l'identificateur de message et le numéro de séquence du message de réponse avec les valeurs du message de requête.

Q51. Sélectionner à la souris les octets de données du message de requête. Comparer ces données avec celles affichées dans le message de requête.

8.3.4. Messages ICMP restants

Reprendre les 2 points précédents pour les messages ICMP `Echo Request` et `Echo Reply` restants.

Q52. Comment les champs d'identification et de numéro de séquence évoluent dans le temps ?

Q53. Est-ce que les séquences de données des requêtes et des réponses changent ?

Q54. Calculer l'écart de temps entre l'émission de chaque message `Echo Request` et la réception de chaque message `Echo Reply`. Comparer les résultats avec les valeurs maximum, moyenne et minium fournies par la commande **ping**.

8.4. Analyse avec (tcp)traceroute

Pour répondre aux questions suivantes, utiliser le résultat de la capture issue de l'étape précédente ou charger un fichier de capture.

8.4.1. Protocoles capturés

Q55. Quels sont les protocoles indiqués dans la colonne Protocol de la fenêtre de liste des trames capturées ?

Il est probable que les paquets ICMP soient précédés d'un jeu de question/réponse DNS.

Q56. Relever l'adresse IP renvoyée avec la réponse DNS.

8.4.2. Message UDP

Q57. Quelle est l'adresse IP destination du premier paquet contenant le message UDP ? Quelles sont les valeurs des champs `Protocol Type` et `Time to Live` ?

Comparer l'adresse IP destination relevée avec celle de la réponse DNS. Noter les valeurs caractéristiques de l'en-tête IP en vue d'une utilisation ultérieure.

Q58. Combien d'octets de données sont présents dans ce message de requête ?

Noter la séquence de caractères présente dans la troisième fenêtre.

8.4.3. Message ICMP «Time Exceeded»

Q59. Quelles sont les adresses IP source et destination du paquet de la première réponse ICMP `Time Exceeded` ?

Étude du message ICMP.

Q60. Quel est le type de message ICMP ?

Les champs `Type`, `Code` et `Checksum` sont suivis par plusieurs octets à zéro puis par l'en-tête IP du message ICMP `Echo Request`. Comparer les valeurs caractéristiques de cet en-tête avec celles notées ci-avant.

Q61. Est-ce que le message ICMP contient de nouveaux octets de données ?

8.4.4. Evolution du champ TTL

Q62. Combien de messages UDP sont émis avec la même valeur de champ `TTL` dans l'en-tête de paquet IP ?

Q63. Quelles sont les adresses IP source des paquets ICMP `Time Exceeded` ?

Comparer ces adresses avec celles données lors de l'exécution de la commande **traceroute**.

Q64. Quel est le type du message ICMP reçu lorsque l'hôte destinataire est atteint ?

Q65. Comment calculer les temps affichés par la commande **traceroute** à partir des valeurs données dans la colonne `Time` de la fenêtre des trames capturées ?

Utiliser les pages de manuels de la commande **traceroute** pour obtenir la signification des différentes valeurs de temps pour atteindre une destination.

8.4.5. Variantes

Il est possible de reprendre les questions ci-dessus en utilisant différentes options des commandes **traceroute** et|ou **tcptraceroute**.

- Analyse uniquement à base de messages ICMP avec l'option `-I` : **`traceroute -I www.phrack.org`**.
- Analyse à base de segments TCP en précisant le numéro de port visé : **`tcptraceroute www.phrack.org 80`**.

  Cette dernière variante est très utile pour vérifier si un service est ouvert ou non.

9. Documents de référence

Guide de l'utilisateur

Le Wireshark User's Guide[15] est la référence la plus complète sur l'utilisation de notre analyseur de trafic favori !

Protocoles

- Le fichier PDF TCP/IP and tcpdump Pocket Reference Guide[16] est une «antisèche» sur les champs des en-têtes des protocoles essentiels ; un document indispensable pour la pratique de l'analyse réseau.

Travaux pratiques

- Le support Configuration d'une interface de réseau local[17] présente les opérations de configuration d'une interface réseau et propose une exploitation des protocoles TCP/IP et ICMP sans recours à un analyseur réseau.
- Le chapitre Using Ethereal - Chapter 4 of Ethereal packet sniffing[18] est un extrait de livre consacré à la version antérieure de l'analyseur de trafic réseau.
- Le site Ethereal Labs[19] présente d'autres travaux pratiques basés sur Ethereal, la version antérieure de l'analyseur de trafic réseau.

[15] http://www.wireshark.org/download/docs/user-guide-a4.pdf
[16] http://www.sans.org/resources/tcpip.pdf
[17] http://www.inetdoc.net/travaux_pratiques/config.interface.lan/
[18] http://searchenterpriselinux.techtarget.com/searchEnterpriseLinux/downloads/284_EPS_04.pdf
[19] http://gaia.cs.umass.edu/ethereal-labs/