

# Visualiser les échanges de courriels chiffrés avec Wireshark

## Présentation

Un **analyseur de trame** est un outil **de base** de l'informaticien car il permet de comprendre ce qu'il se passe à un niveau très bas. Il permet aussi de mettre en évidence de nombreux concepts théoriques du cours.

**Wireshark** (anciennement Ethereal) est un logiciel libre d'**analyse de protocole**, ou **packet sniffer**, utilisé dans le **dépannage** et l'**analyse du fonctionnement** des réseaux informatiques. Il est utilisé pour **diagnostiquer** des dysfonctionnements dans un réseau informatique.

Un analyseur de protocoles (ou analyseur de réseaux ou de paquets) est un logiciel permettant **d'intercepter** et de consigner le trafic des données transférées sur un réseau de données. L'analyseur **capture** chaque **PDU** (protocol data unit - unité de données de protocole) des flux de données circulant sur le réseau.

Il permet de décoder et d'analyser leur contenu conformément aux spécifications **RFC** ou autres appropriées.

Wireshark est programmé pour reconnaître la structure de différents protocoles réseau.

## Installation de Wireshark dans la VM Ubuntu

Avant d'installer un nouveau logiciel, mettez à jour votre environnement Ubuntu.

- lancez un Terminal
- Mettez à jour votre environnement Ubuntu

```
$ sudo apt update  
$ sudo apt upgrade
```

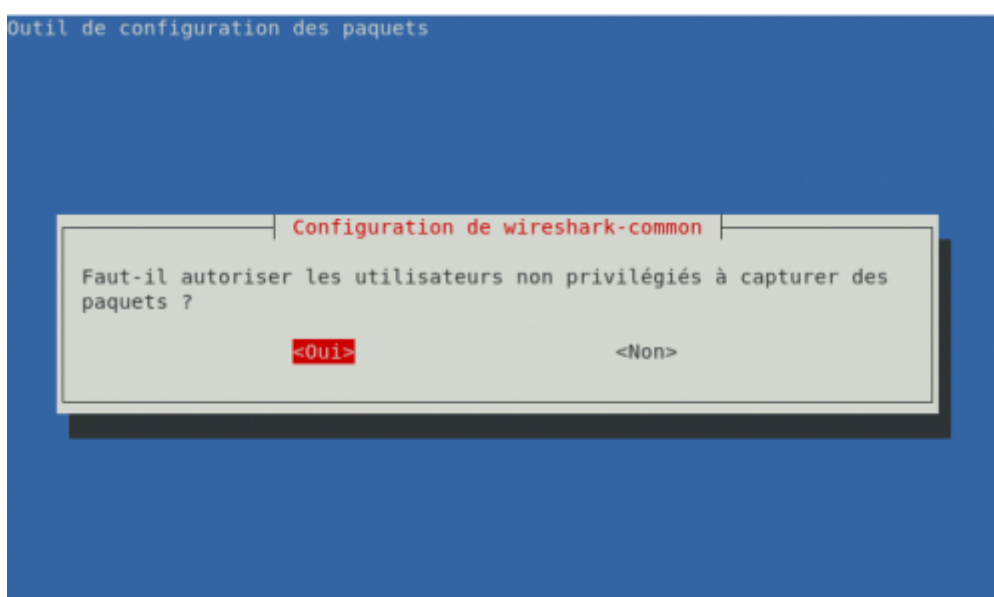
- installez Wireshark

```
$ sudo apt install wireshark
```

Durant l'installation il vous est demandé **d'autoriser** l'installation de **Dmccap**. Utilisez la touche **TAB** pour sélectionner le bouton **OK** et validez avec la touche entrée :



- **Autorisez** les utilisateurs non privilégiés à **utiliser Wireshark** :



- **Ajoutez** le compte utilisateur voulu au groupe **wireshark** pour lui permettre d'utiliser toutes les fonctionnalités de Wireshark :

```
$ sudo usermod -aG wireshark comptevoulu
```

ou le compte qui a ouvert la session (la commande whoami permet de connaître l'utilisateur qui a ouvert la session :

```
$ sudo usermod -aG wireshark $(whoami)
```

^

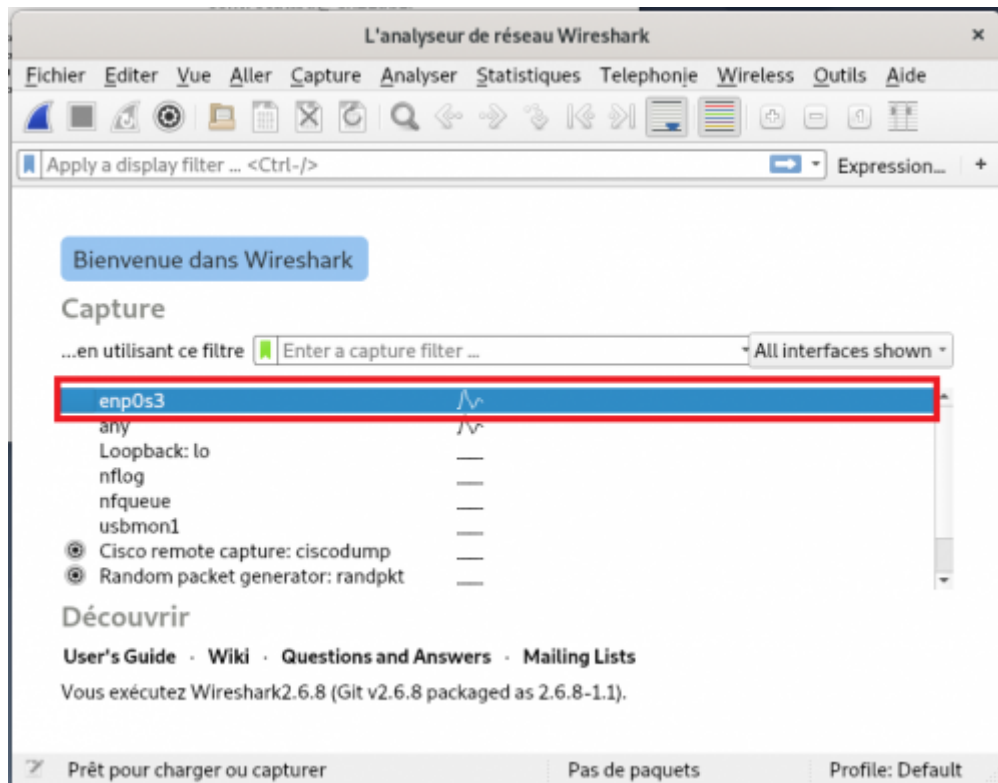
- Fermez puis réouvrez votre session pour actualiser vos droits.

- Lancez un Terminal et vérifiez que vous êtes bien dans le groupe wireshark avec la commande suivante :

```
$ groups
```

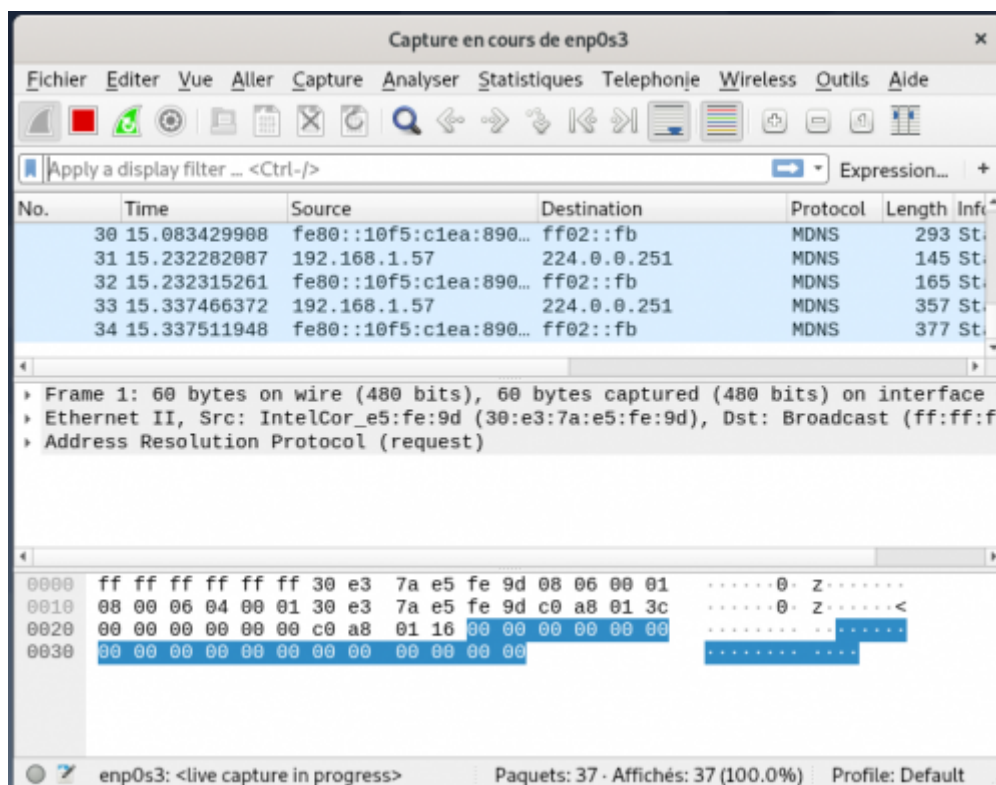
## Prise de contact

Lancez le logiciel qui se présente ainsi (la carte réseau **enp0s3** connectée au réseau est encadrée en rouge):

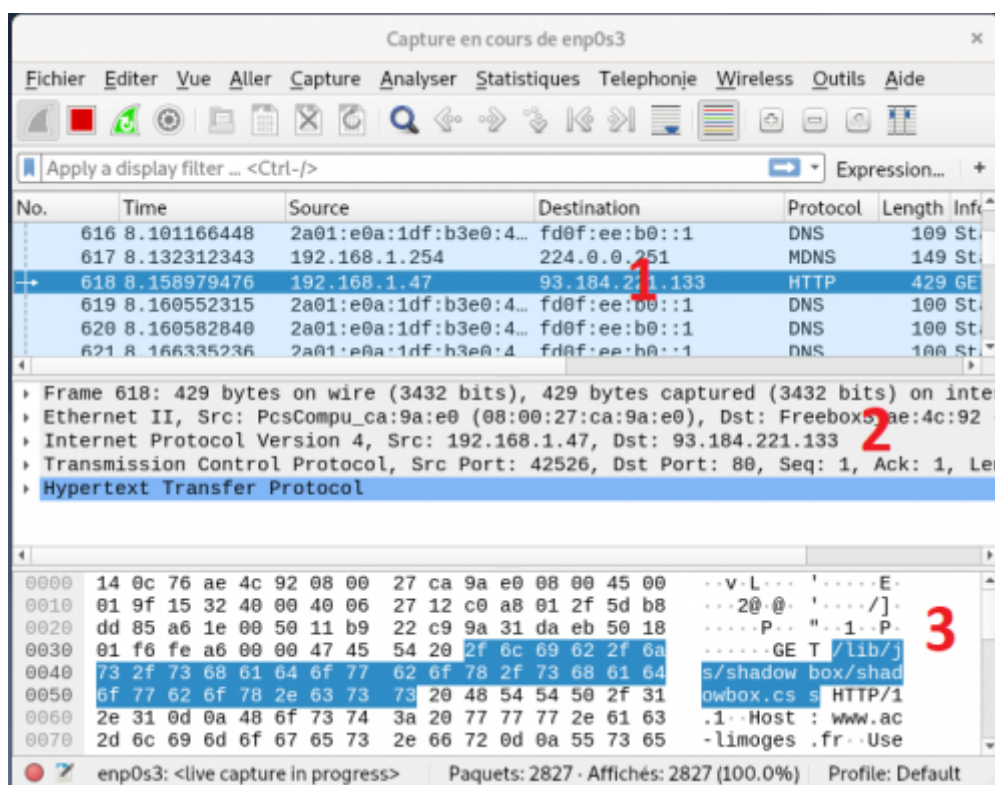


Vous pouvez démarrer une capture en cliquant tout simplement sur l'interface réseau qui vous intéresse. Vous ne verrez donc que le trafic réseau vu par cette carte réseau.

- **Démarrez une capture** avec l'icône en forme d'aileron de requin en haut à gauche.
- Au bout de quelques instants vous verrez des paquets réseau apparaître dans la fenêtre, ce qui montre que même si vous ne faites rien, il y a des informations qui circulent sur le réseau !
- **Arrêtez** la capture des trames :



Wireshark permet de donner des informations très détaillées. Examinez l'écran principal du logiciel :



On observe en trois parties :

- 1. La **liste des trames Ethernet capturées**. Elles sont chacune numérotées et horodatées par Wireshark (ces données ne figurent donc pas dans la trame d'origine).
- 2. Pour chaque trame, sa **structure** est présentée sous une forme **hiérarchique** (ainsi ce que vous voyez dans le volet 2 est le détail de la trame numéro 2)

- 3. Le volet 3 est la même chose que le volet 2 mais sous une forme **brute** non structurée avec une présentation ASCII et hexadécimale. Vous pouvez la présenter en binaire (clic droit dans le volet 3).

## Examen détaillé

Examinez en détail le volet 2. Vous pouvez cliquer sur les croix pour développer les contenus. Ce volet met en évidence le phénomène **d'encapsulation** :

Le premier élément concerne la **trame** proprement dite (taille, temps, etc.) :

```

▼ Frame 618: 429 bytes on wire (3432 bits), 429 bytes captured (3432 bits) on int
  ▸ Interface id: 0 (enp0s3)
    Encapsulation type: Ethernet (1)
    Arrival Time: Nov 12, 2020 21:30:46.030778283 CET
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1605213046.030778283 seconds
    [Time delta from previous captured frame: 0.026667133 seconds]
    [Time delta from previous displayed frame: 0.026667133 seconds]
    [Time since reference or first frame: 8.158979476 seconds]
    Frame Number: 618
    Frame Length: 429 bytes (3432 bits)
    Capture Length: 429 bytes (3432 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:tcp:http]
    [Coloring Rule Name: HTTP]
    [Coloring Rule String: http || tcp.port == 80 || http2]

```

Ensuite, en montant d'un cran est présentée la partie liée à **Ethernet**. On retrouve les adresses physiques de **destination** et de **source**, également le type trame de niveau supérieur (ici IP **0x0800**) :

```

▼ Ethernet II, Src: PcsCompu_ca:9a:e0 (08:00:27:ca:9a:e0), Dst: FreeboxS_ae:4c:92 (14:0c:76:ae:4c:92)
  ▸ Destination: FreeboxS_ae:4c:92 (14:0c:76:ae:4c:92)
    Address: FreeboxS_ae:4c:92 (14:0c:76:ae:4c:92)
    .... 0. .... = LG bit: Globally unique address (factory def)
    .... 0. .... = IG bit: Individual address (unicast)
  ▸ Source: PcsCompu_ca:9a:e0 (08:00:27:ca:9a:e0)
    Address: PcsCompu_ca:9a:e0 (08:00:27:ca:9a:e0)
    .... 0. .... = LG bit: Globally unique address (factory def)
    .... 0. .... = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800)

```

A la couche supérieure, c'est la partie IP :

```

▼ Internet Protocol Version 4, Src: 192.168.1.47, Dst: 93.184.221.133
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 415
    Identification: 0x1532 (5426)
  ▸ Flags: 0x4000, Don't fragment
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0x2712 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.1.47
    Destination: 93.184.221.133

```

On retrouve les **adresses IP source** et **destination du paquet**. De plus, certaines données correspondent à des bits d'un octet particulier (**differentiated services field**). Des données techniques comme la longueur du paquet, le numéro de séquence, le temps à vivre (**TTL** ou Time To Live), l'identité du protocole supérieur (**UDP**) sont nécessaires au fonctionnement de cette couche.

En montant encore d'un niveau on observe la partie **transport**. Ici il s'agit de **UDP** qui est un protocole simple sans gestion des erreurs, son contenu est beaucoup plus simple que **TCP** :

```
Transmission Control Protocol, Src Port: 42526, Dst Port: 80, Seq: 1, Ack: 1, L
  Source Port: 42526
  Destination Port: 80
  [Stream index: 10]
  [TCP Segment Len: 375]
  Sequence number: 1 (relative sequence number)
  [Next sequence number: 376 (relative sequence number)]
  Acknowledgment number: 1 (relative ack number)
  0101 .... = Header Length: 20 bytes (5)
  Flags: 0x018 (PSH, ACK)
  Window size value: 502
  [Calculated window size: 64256]
  [Window size scaling factor: 128]
  Checksum: 0xfea6 [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  [SEQ/ACK analysis]
  [Timestamps]
  TCP payload (375 bytes)
```

Comme à chaque fois, une information concernant le protocole de niveau supérieur (ici **ssdp** pour Simple Service Discovery Protocol) est intégré. Nous retrouvons également la notion de **port source** et de **destination** mais aussi de **checksum** qui permet le contrôle d'erreur.

Et enfin, on aborde la partie **application**. vous remarquerez que **Wireshark** sait mettre en relation les **données structurées** et les **données brutes**. Ainsi, sur n'importe quelle couche du paquet, si vous sélectionnez un élément, celui-ci est mis en évidence dans le dernier volet :

```
Hypertext Transfer Protocol
  GET /lib/js/shadowbox/shadowbox.css HTTP/1.1\r\n
  Host: www.ac-limoges.fr\r\n
  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78
  Accept: text/css,*/*;q=0.1\r\n
  Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3\r\n
  Accept-Encoding: gzip, deflate\r\n
  Connection: keep-alive\r\n
  Referer: http://www.ac-limoges.fr/\r\n
  Cookie: PHPSESSID=p3rggrk9j7thvjbd7sv1ajko96\r\n
  \r\n
  [Full request URI: http://www.ac-limoges.fr/lib/js/shadowbox/shadowbox.css]
  [HTTP request 1/6]
  [Response in frame: 624]
  [Next request in frame: 632]
```

Pour information, le paquet présenté ici correspond au **protocole HTTP** et au téléchargement de la feuille de style **shadowbox.css** de la page d'accueil du site l'Académie de Limoges.

## Filtres

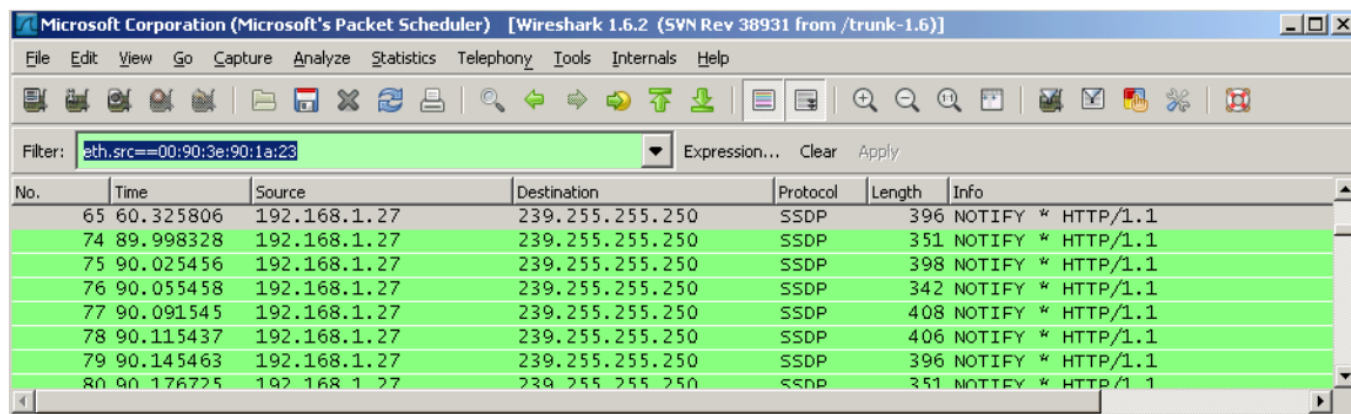
Lorsque vous capturez des trames sur un réseau, vous pouvez avoir beaucoup de **bruit** car tous les ordinateurs du réseau **communiquent en permanence**. Il est donc important de pouvoir **filtrer** une capture sur différents critères. Parmi les plus fréquents, nous avons les **adresses source** ou **destination** de niveau 2 ou 3 et le protocole.

Dans copie d'écran ci-dessous, vous voyez qu'il existe une zone **filter** qui permet justement de saisir des requêtes avec une **syntaxe** propre à Wireshark (mais qui s'inspire du langage C).

Par exemple, filtre sur l'adresse MAC source : 00:90:3e:90:1a:23 (à adapter à ce que vous avez



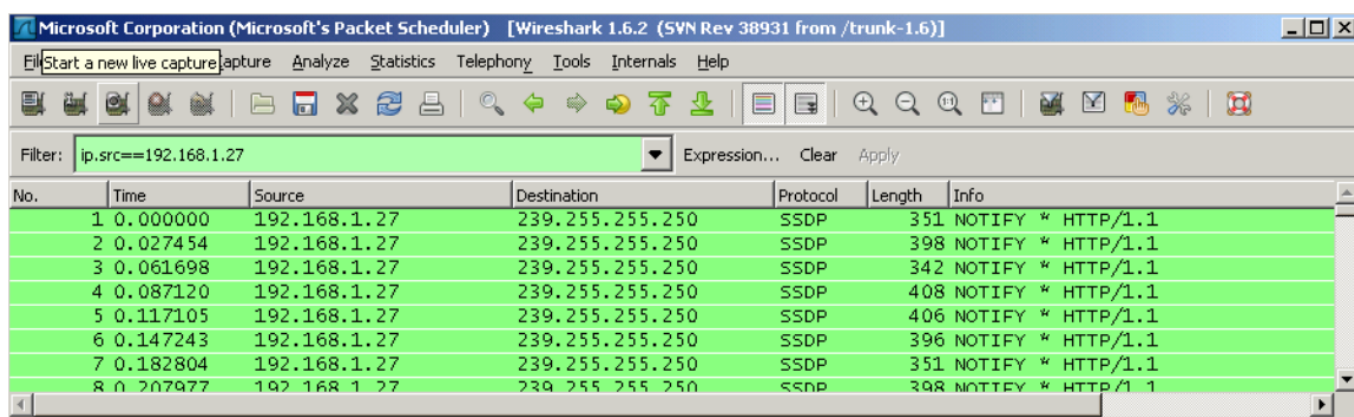
**réellement dans votre capture**) Dans le champ Filter saisissez : **eth.src==00:90:3e:90:1a:23** et cliquez sur le bouton **Apply**



Lorsque vous avez commencé à taper **eth**. Vous avez vu que de nombreux autres champs sont disponibles.

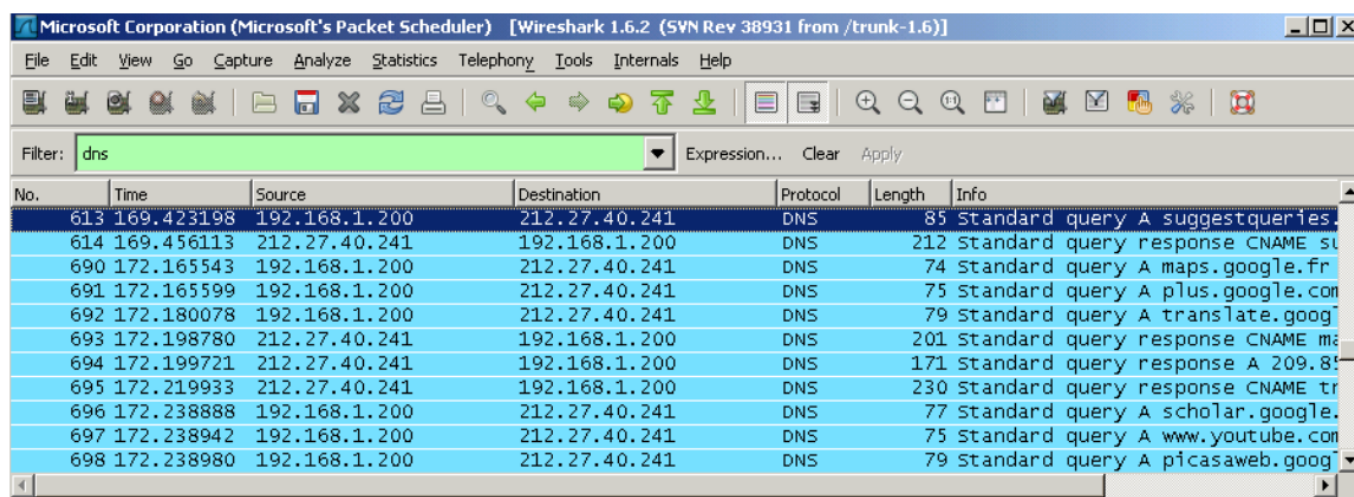
Vous pouvez faire de même avec les **adresses IP**, par exemple l'adresse source **192.168.1.27**.

Dans le champ Filter saisissez : **ip.src==192.168.1.27**



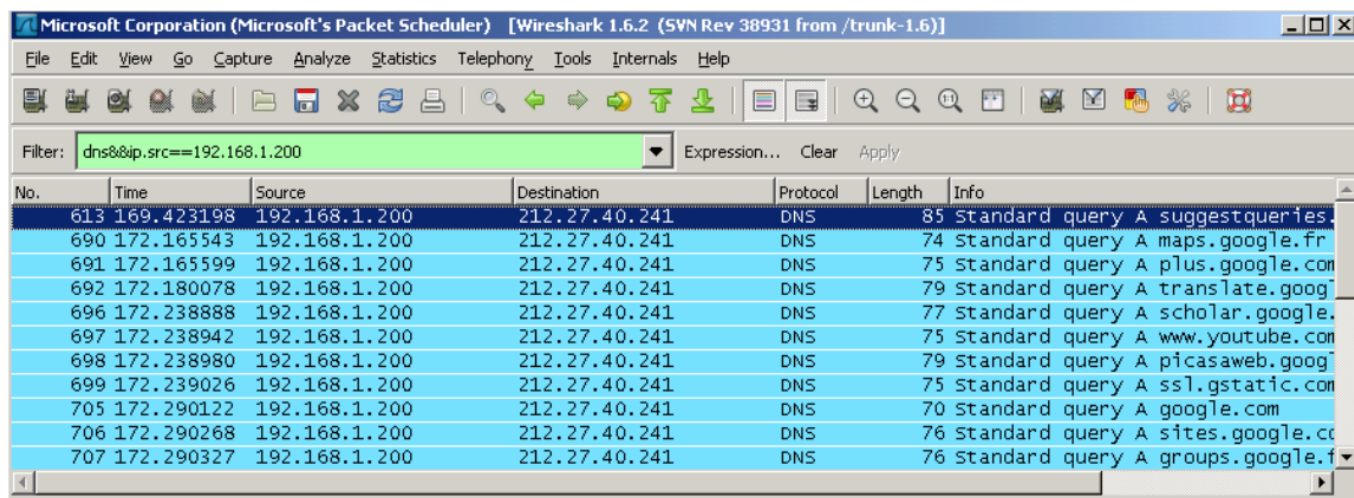
Pour un filtre basé sur les protocoles, saisissez tout simplement :

- Filter : **http** ou **ssh** ou **dns**



Bien sûr, les filtres peuvent être **cumulés**, par exemple **protocole** et **adresse source** :

- Filter : **dns&&ip.src==192.168.1.200**



Microsoft Corporation (Microsoft's Packet Scheduler) [Wireshark 1.6.2 (SVN Rev 38931 from /trunk-1.6)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: **dns&&ip.src==192.168.1.200** Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
613	169.423198	192.168.1.200	212.27.40.241	DNS	85	Standard query A suggestqueries.
690	172.165543	192.168.1.200	212.27.40.241	DNS	74	Standard query A maps.google.fr
691	172.165599	192.168.1.200	212.27.40.241	DNS	75	Standard query A plus.google.com
692	172.180078	192.168.1.200	212.27.40.241	DNS	79	Standard query A translate.google.
696	172.238888	192.168.1.200	212.27.40.241	DNS	77	Standard query A scholar.google.
697	172.238942	192.168.1.200	212.27.40.241	DNS	75	Standard query A www.youtube.com
698	172.238980	192.168.1.200	212.27.40.241	DNS	79	Standard query A picasaweb.google.
699	172.239026	192.168.1.200	212.27.40.241	DNS	75	Standard query A ssl.gstatic.com
705	172.290122	192.168.1.200	212.27.40.241	DNS	70	Standard query A google.com
706	172.290268	192.168.1.200	212.27.40.241	DNS	76	Standard query A sites.google.co
707	172.290327	192.168.1.200	212.27.40.241	DNS	76	Standard query A groups.google.f

## Retour Accueil Bloc3

- [Bloc3](#)

From:  
<https://siocours.lycees.nouvelle-aquitaine.pro/> - Les cours du BTS SIO

Permanent link:  
<https://siocours.lycees.nouvelle-aquitaine.pro/doku.php/bloc3s1/wiresharkmessagerie?rev=1605213788>

Last update: 2020/11/12 21:43

