Quelle sécurité appliquer sur le réseau

Analyse de réseau wireshark, arpscan

►BUT :

- On va récupérer et analyser les trames qui passent sur notre réseau au moyen d'un outil simple « Wireshark ».
- On va aussi débusquer des utilisateurs du réseau en utilisant un scanner « arpscan » de couche 2 (contrairement à nmap qui est plutôt de couche 3 donc utilisant des adresses IP) et distinguer les applications sécurisées et celles qui ne le sont pas, ce qui permet de faire des choix permettant de protéger sa vie privée.

Remarque : comme tout couteau qui peut être utilisé comme une arme ou comme une manière élégante de manger de la viande ou une pomme, les outils que nous allons manipuler servent autant à se connaître (informatiquement parlant) qu'à espionner des réseaux inconnus (et ceci est une action répréhensible par la loi).

►GLOSSAIRE :

Wireshark : C'est un analyseur de paquets open source (licence GNU). Il est gratuit et fonctionne sur tous les systèmes (Windows, Linux, Mac...). Ce scanner graphique est la pierre angulaire de l'analyse de réseau.

arp-scan : Scanner de couche 2, il sollicite les périphériques sur votre réseau et les obligent à répondre car l'adresse de destination est une adresse broadcast arp (adresse : ff:ff:ff:ff:ff:ff). Même une protection type pare feu ne peut empêcher cette réponse. Une exception cependant, un périphérique qui ne fait qu'écouter le réseau et qui ne communique pas, ou un natel peuvent être rigides aux requêtes ARP.

Scapy : "Scapy est un logiciel libre de manipulation de paquets réseau écrit en python.Il est capable, entre autres, d'intercepter le trafic sur un segment réseau, de générer des paquets dans un nombre important de protocoles, de réaliser une prise d'empreinte de la pile TCP/IP, de faire un traceroute, d'analyser le réseau informatique… Créé par Philippe Biondi, actuellemnt développé par Pierre Lalet, Guillaume Valadon & Gabriel Potter" (fr.wikipedia.org/wiki/Scapy)

Autre références :

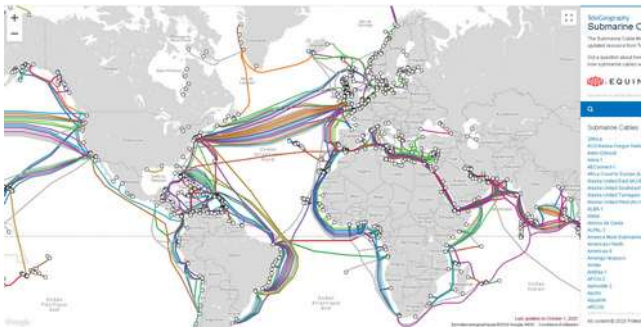
Rodofile, N., Radke, K., & Foo, E. (2015). Real-time and interactive attacks on DNP3 critical infrastructure using Scapy,

Biondi, P. (2010). Scapy documentation (!)

Maxwell, A. (2012). The very unofficial dummies guide to Scapy. *The IT Geek Chronicles*.

►La réponse à la question :

Il y a différents types de câbles sous-marins, les anciens câbles sont de type série et avec des débits lents, ils sont remplacés actuellement par des fibres optiques beaucoup plus rapides ce qui explique des temps très différents pour traverser l'Atlantique ou pour descendre le long de l'Afrique.
Réf : https://www.submarinecablemap.com/
En cliquant sur le câble on connaît les lieux d'extrémité.



Les câbles sous-marins sont aussi espionnés : https://bfmbusiness.bfmtv.com/entreprise/les-cables-sous-marins-meilleurs-amis-de-ceux-qui-nous-espionnent-897455.html (2015)
https://ltemagazine.com/la-securite-des-cables-sous-marins-un-enjeu-majeur (2018)

►C'EST PARTI :

— Telnet est-il sécurisé ?—

Le Raspberry Pi est branché sur votre box, votre ordinateur est aussi sur le réseau filaire ou bien en Wifi peu importe.

Sur votre ordinateur connectez vous en VNC sur le Raspberry Pi. Vous pouvez fermer l'application p1 du TPA1, elle a servi simplement à lancer l'application Telnet, on verra à la fin du TP comment l'empêcher de ce lancer au démarrage.
Pour installer Wireshark il faut que la base d'information des packages soit à jour. Pour cela il faut, dans un terminal, passer la commande suivante : sudo apt-get update
puis la commande sudo apt-get install wireshark
Une fois l'installation terminée vous retrouvez wireshark dans le menu Applications→Internet→ wireshark

Si vous cliquez le programme, on se trouve face à un problème, on ne retrouve pas notre interface Ethernet eth0 dans la liste des interfaces possibles à écouter (en interface graphique on n'est pas administrateur/root)
Deux solutions sont possibles :
1) Lancer wireshark en tant qu'admin dans un terminal : sudo wireshark & (& pour être en tâche de fond)
2) On modifie le contenu du menu pour lancer wireshark correctement.

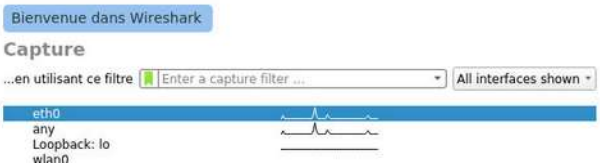
Les descriptions des applications des menus xfce4 (l'environnement graphique) sont dans le répertoire `/usr/share/applications` et sont nommées *nomApplication*.desktop
On va ouvrir le fichier wireshark.desktop : sudo nano /usr/share/applications/wireshark.desktop
et modifier la ligne Exec=wireshark %f en Exec=sudo wireshark %f .
puis sauvegarder et quitter : Ctrl+O Entrée Ctrl+x
Pour rajouter ou modifier un menu il faut au minimum le fichier suivant :

```
[Desktop Entry]
Version=1.0
Type=Application
Name=<Le_nom_de_l_application>
Exec=<Le_chemin_vers_l_application>
Icon=<L_icone_de_l_application>
Categories=<La_catégorie_normalisée_du_menu>
```

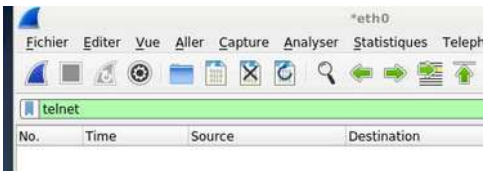
Exemple de catégories : Network, Development... lire la catégorie du menu déjà installé dans lequel on veut raccrocher son propre menu.

On peut maintenant lancer wireshark correctement et on va analyser nos échanges telnet.

Lancez wireshark
et
double cliquer
sur la ligne eth0.

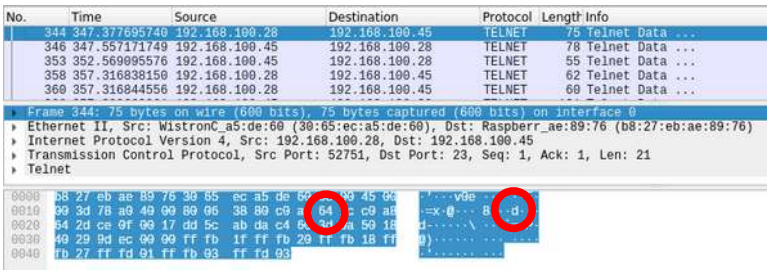


Pour éviter d'avoir trop d'informations, on va filtrer les entrées de wireshark en écrivant telnet dans la barre de filtre et en validant par la touche Entrée



On se connecte en telnet (comme au TPA1) on entre son nom, on peut voir que ça enregistre côté Wireshark. Une fois que c'est terminé on clique sur le carré rouge de Wireshark pour arrêter l'enregistrement.

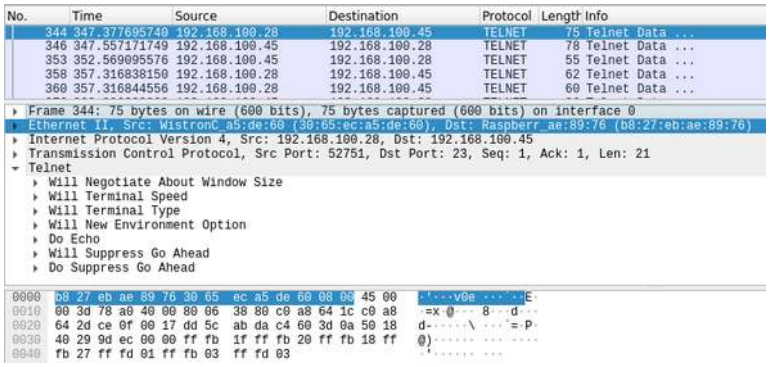
Quand on clique sur une trame (premier cadre en haut) on retrouve son découpage en couche osi (***O**pen **S**ystems **I**nterconnection*) dans le cadre en dessous.



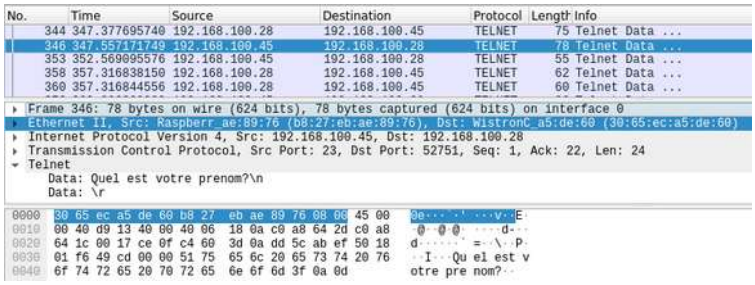
Si on clique sur « Frame » on « bleuit » la zone correspondante (tout) dans le troisième cadre qui fournit la trame en binaire à gauche et sa traduction en valeurs ASCII à droite par exemple 64 (hexadécimal) est représenté par d en ASCII.

En cliquant sur la deuxième ligne on sélectionne la partie correspondant à la couche Ethernet,

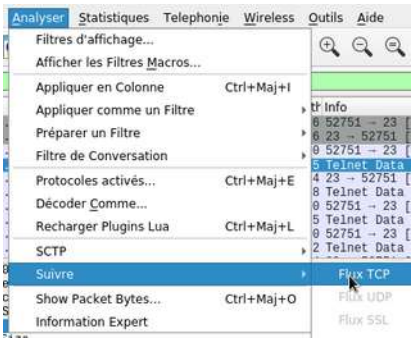
On y retrouve les adresses MAC source Src et destinataire Dst et le type. C'est une trame Ethernet.

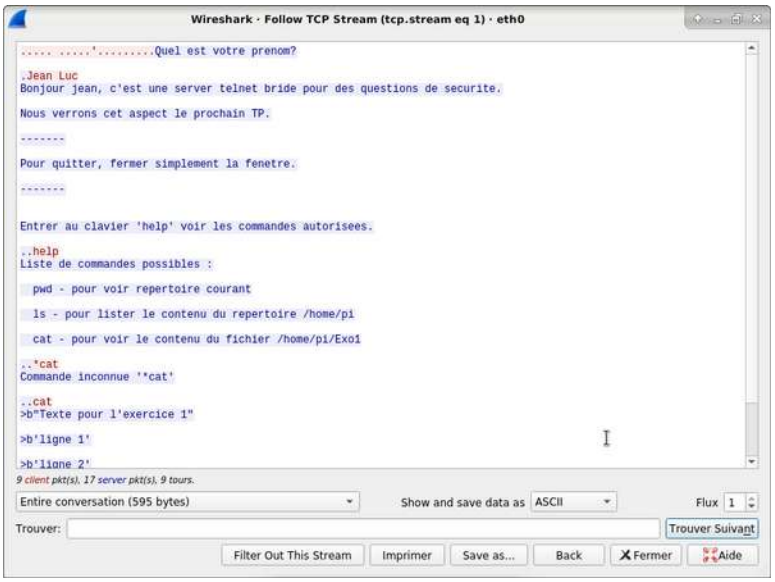


En choisissant sa trame on peut voir la demande du prénom de la part du serveur telnet.
On voit en bas à droite QUE LE DIALOGUE EST EN CLAIR (on peut lire « Qu el est votre pre nom ? »



Sélectionner une trame et cliquer le menu « Flux TCP », on peut voir la totalité des échanges de la session.





On imagine très bien que lors de la demande de mot de passe par le switch ou le routeur, si la liaison est en Telnet toute personne lisant les trames aura votre mot de passe. On évitera donc, quand on est à distance, d'utiliser Telnet qui n'est pas sécurisé.

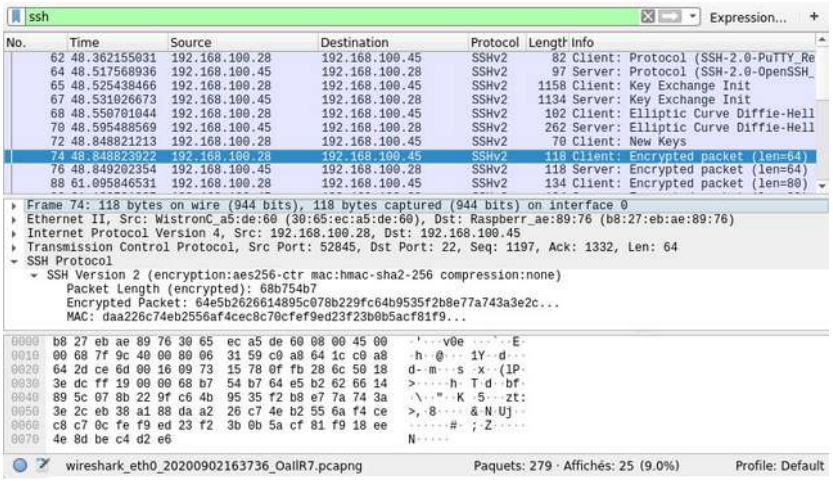
Effacer le fichier /home/pi/Exo1 pour supprimer le démarrage automatique de l'application p1

— Ssh est-il sécurisé ?—

On relance une capture de trame (Menu Capture → Démarrer), on ne veut pas sauvegarder les derniers échanges Telnet donc on clique « Continue without saving »). On filtre en écrivant « ssh ».

On lançe une connexion ssh depuis Putty sur votre PC comme au TPA1.

On peut voir qu'après une phase de discussion en clair (non chiffrée pour choisir le chiffrement) les échanges sont ensuite chiffrés (« Encrypted paquet ») et donc indéchiffrables

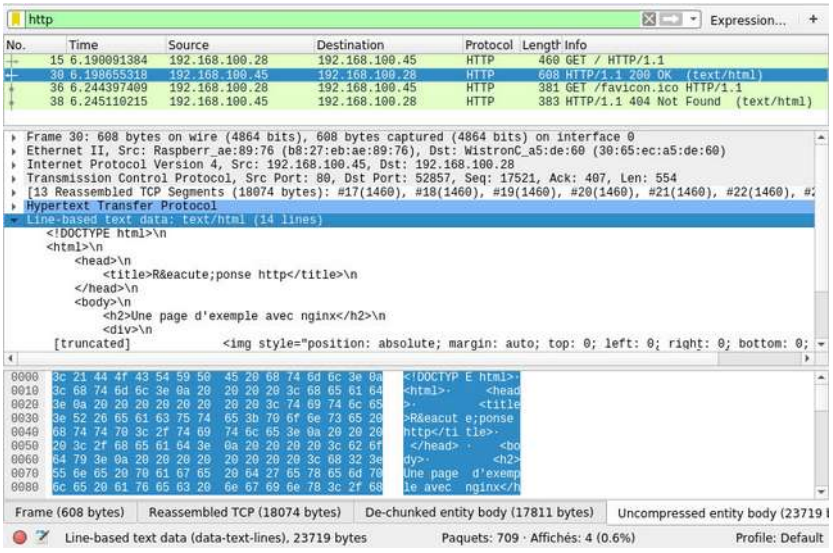


Conclusion : Il faut préférer une connexion Ssh lorsqu'on se connecte à distance à un périphérique qu'il soit switch, routeur, PC, IoT … Vous bénéficiez alors d'une intégrité des données (elles ne sont pas modifiées, sinon on le détecte), de la confidentialité des données (à cause du chiffrement), et de l'authentification (vous avez sauvegardé la clé publique du périphérique lors de la première connexion). Cette clé publique identifie de manière quasiment unique votre périphérique et ne vous sera plus proposée sauf si elle a été changée.

— Qu'en est-il en http ?—

On lance une connexion http depuis un navigateur sur votre PC comme au TPA1 (en utilisant l'adresse IP de votre RaspberryPi dans la barre d'adresse).

On peut voir que la discussion est en clair.
On reconnaît très bien le code html de la page visualisée dans le navigateur.

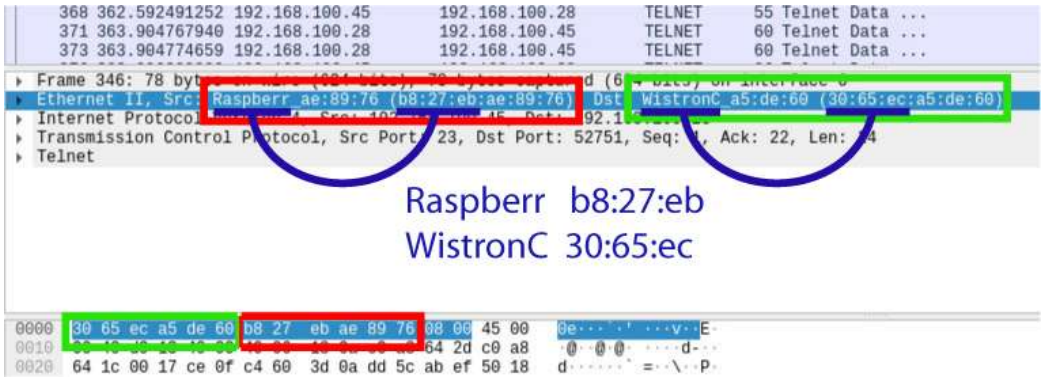


Conclusion : Le protocole http est en clair, pour conserver la confidentialité de vos données utilisez le protocole https qui est sécurisé (un peu comme le ssh).

Question : Dans le protocole https quelle est l'utilité du cadenas en dehors du fait qu'il vous offre une conversation sécurisée (chiffrée) ?

— ARP Address Resolution Protocol —

On vient de voir qu'un analyseur de réseau peut nous fournir beaucoup d'informations sur les trames qui circulent sur un réseau. On peut aussi débusquer des périphériques en utilisant un scanner ARP (Protocole de Résolution d'Adresses).
Dans le TPA précédent nous avions utilisé un scanner IP (nmap) qui travaille principalement avec les adresses IP mais la réponse des périphériques peut être masquée par un firewall ou un dispositif de filtrage.
L'intérêt de travailler avec le protocole ARP c'est que toute machine voulant parler sur le LAN est obligée d'utiliser les adresses MAC et de répondre au protocole ARP (en IPV4). On retrouve les adresses physiques (MAC) dans la trame Ethernet et Wireshark nous les affiche dans la partie hexadécimale, il traduit la partie constructeur de l'adresse et recopie les trois autres octets de numéro de série, dans la fenêtre du milieu concernant le protocole Ethernet.



Les adresses MAC sont importantes pour le fonctionnement des trames Ethernet (couche 2). Chaque périphérique va donc les mémoriser pour une utilisation ultérieure.
Un routeur ou un périphérique de type PC va mémoriser le couple (adresse IP, adresse MAC) dans une table. Ils gagneront du temps pour envoyer une trame à un destinataire si le couple (ad IP dest, ad MAC dest) existe.
Un switch (couche 2) va mémoriser le couple (ad MAC, numéro du port du switch d'où provient cette adresse), de manière à trouver rapidement vers quel port envoyer une trame ayant l'adresse MAC associée.

— Scanner un réseau en utilisant la couche 2 —

Pour scanner un réseau en couche 2, donc en utilisant les adresses physiques (MAC), on va dans un premier temps utiliser l'application arp-scan.
Dans une fenêtre de commande après avoir mis à jour la base de connaissances,on télécharge/installe arp-scan.

```
sudo apt-get update
sudo apt-get install arp-scan
```

```
pi@raspberrypi:~ $ sudo arp-scan 192.168.100.0/24
Interface: eth0, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.9.5 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.100.53	30:65:ec:a5:02:60	Wistron (ChongQing)
192.168.100.31	e8:6a:64:f1:d7:2d	(Unknown)
192.168.100.65	14:0c:76:4b:04:17	FREEBOX SAS
192.168.100.21	b8:27:eb:d0:12:53	Raspberry Pi Foundation
192.168.100.27	00:21:b7:3d:a5:e7	Lexmark International Inc.
192.168.100.78	00:01:e6:2f:c8:05	Hewlett Packard
192.168.100.25	14:0c:76:af:51:c4	FREEBOX SAS
192.168.100.30	88:29:9c:5a:bd:61	(Unknown)
```

Pourquoi scanner en couche 2 ? A priori les appareils voulant communiquer doivent se faire connaître au niveau du LAN et donc répondent aux sollicitations en couche 2. Ce type de scan est complémentaire au

scanner de niveau 3 de type nmap. Il faut noter que les natels n'ont pas tendance à répondre à ces demandes qu'elles arrivent en couche 2 ou en couche 3.
Pour connaître le fonctionnement de ce type de scan on peut regarder les échanges avec Wireshark, avec un filtrage "arp".

```
►Frame 725: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▼Ethernet II, Src: Raspberr_08:df:38 (b8:27:eb:08:df:38), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
        ►Destination: Broadcast (ff:ff:ff:ff:ff:ff)
        ►Source: Raspberr_08:df:38 (b8:27:eb:08:df:38)
        Type: ARP (0x0806)
▼Address Resolution Protocol (request)
        Hardware type: Ethernet (1)
        Protocol type: IPv4 (0x0800)
        Hardware size: 6
        Protocol size: 4
        Opcode: request (1)
        Sender MAC address: Raspberr_08:df:38 (b8:27:eb:08:df:38)
        Sender IP address: 192.168.100.22
        Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
        Target IP address: 192.168.100.1
```

En résumé, en couche 2 on met en destination une adresse physique de broadcast (ff:ff:ff:ff:ff:ff) et notre adresse MAC comme source. En couche 3 on utilise le protocole arp avec une demande "arp request" dans laquelle le couple (ad MAC source/ad IP source) c'est vous (Rpi), et le couple de destination (ad MAC/ad IP) possède une ad MAC inconnue (00:00:00:00:00:00).
La réponse comportera le couple de destination (ad MAC/ad IP) renseigné.

— Scanner "fait maison" avec SCAPY —

Il faut installer le bibliothèque Scapy en admin car beaucoup d'actions liées au réseau demandent à être administrateur.

```
sudo pip3 install scapy
ou sudo -s puis pip3 install scapy
```

Pour tester les éléments dont on a besoin dans la bibliothèque scapy, dans un premier temps on va simplement utiliser le shell de python3

```
sudo python3
```

La bibliothèque scapy doit être importée pour pouvoir l'utiliser

```
>>> from scapy.all import *
```

Chaque protocole possède une classe qui lui est dédiée. En regardant la sortie Wireshark du paragraphe précédent, nous aurons besoin de la classe qui gère la couche 2 (Ethernet) et celle qui gère le protocole ARP. On commence par Ethernet, la classe est Ether() et pour voir les propriétés de l'objet on utilise la méthode show()

```
>>> c2 = Ether()
>>> c2.show()
###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = b8:27:eb:08:df:38
  type      = 0x9000
```

On remarque que l'adresse MAC de destination est déjà une adresse de boadcast, l'adresse source est celle du Raspberry Pi. Le type "Ethertype" à 0x9000 correspond à la description de la loopback (https://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xhtml). Pour mettre la valeur on peut modifier l'objet une fois créé : `>>> c2.type=0x0806`
ou lors de la création : `>>> c2 = Ether(type=0x0806)`
On procède de la même façon pour la couche ARP :

```
>>> arp = ARP()
>>> arp.show()
###[ ARP ]###
  hwtype    = 0x1
  ptype     = IPv4
  hwlen     = None
  plen      = None
  op        = who-has
```

```
  hwsrc     = b8:27:eb:08:df:38
  psrc      = 192.168.100.22
  hwdst     = 00:00:00:00:00:00
  pdst      = 0.0.0.0
```

On remarque que l'on a déjà une requête ARP de découverte (who-has dans la propriété op). L'adresse MAC et l'adresse IP source sont déjà remplies, l'adresse MAC de destination aussi (inconnue donc 00:00:00:00:00:00). L'adresse IP de destination devra être complétée, les autres propriétés seront complétées automatiquement.

On construit notre trame en utilisant l'opérateur / qui encapsule les objets:

```
>>> trame = Ether()/ARP(pdst="192.168.100.0/24")
>>> trame.show()
###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = b8:27:eb:08:df:38
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = None
     plen      = None
     op        = who-has
     hwsrc     = b8:27:eb:08:df:38
     psrc      = 192.168.100.22
     hwdst     = 00:00:00:00:00:00
     pdst      = Net('192.168.100.0/24')
```

Au lieu de lui demander une adresse IP de destination spécifique on lui donne la plage à scanner avec le CIDR (/24), avec le show() on voit que l'objet à bien compris qu'il s'agit d'un réseau (pdst = Net...). La propriété EtherType a été modifiée automatiquement (type = ARP).

Il ne reste plus qu'à envoyer la trame et à récupérer les trames en retour. Plusieurs méthodes existent :

- envoi/réception de couche 3
  La méthode sr() qui retourne le couple (paquets avec réponse / paquets sans réponse).
  La méthode sr1() qui est identique à sr() mais pour un seul paquet envoyé.
- envoi/réception de couche 2
  La méthode srp() qui retourne le couple (paquets avec réponse / paquets sans réponse) Dans les paquets sans réponse il y a None quand le délai (timeout) est atteint.

  L'utilisation de scp() s'impose. Le timeout impose aussi le temps total du scan, plus il sera long plus le scan durera. J'ai choisi un timeout de 4 secondes. On se limitera aux trames reçues srp(...) **[0]**

Dans un premier temps pour voir les objets retournés, on va envoyer une trame uniquement à la passerelle 192.168.100.254 (pour moi, indiquez la votre)

```
>>> trame = Ether()/ARP(pdst="192.168.100.254")
>>> result = srp(trame, timeout=4,verbose=0)[0]
```

On ne regarde le premier élément du tableau (qui n'en comporte qu'un)

```
>>> result[0]
(<Ether  type=ARP |<ARP  pdst=192.168.100.254 |>>, <Ether  dst=b8:27:eb:08:df:38
src=14:0c:76:a3:51:c1 type=ARP |<ARP  hwtype=0x1 ptype=IPv4 hwlen=6 plen=4 op=is-at
hwsrc=14:0c:76:a3:51:c1 psrc=192.168.100.254 hwdst=b8:27:eb:08:df:38 pdst=192.168.100.22
| <Padding  load='\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00' |>>>)
```

C'est aussi un tableau, on fait afficher le premier élément puis le second

```
>>> result[0][0]
<Ether  type=ARP |<ARP  pdst=192.168.100.254 |>>

>>> result[0][1]
<Ether  dst=b8:27:eb:08:df:38 src=14:0c:76:a3:51:c1 type=ARP |<ARP  hwtype=0x1 ptype=IPv4 hwlen=6
plen=4 op=is-at hwsrc=14:0c:76:a3:51:c1 psrc=192.168.100.254 hwdst=b8:27:eb:08:df:38
pdst=192.168.100.22 |<Padding  load='\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\ x00\x00\x00' |>>>
```

Le premier élément est la trame d'envoi, le deuxième élément est la réponse que l'on peut afficher plus clairement avec la méthode show()

```
>>> result[0][1].show()
###[ Ethernet ]###
  dst       = b8:27:eb:08:df:38
  src       = 14:0c:76:a3:51:c1
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = 6
     plen      = 4
     op        = is-at
     hwsrc     = 14:0c:76:a3:51:c1        <-- adresse MAC de réponse
     psrc      = 192.168.100.254          <-- adresse IP de réponse
     hwdst     = b8:27:eb:08:df:38
     pdst      = 192.168.100.22
###[ Padding ]###
        load      = '\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
```

On regroupe tout cela dans un fichier monscan.py en important seulement ce dont on a besoin et en remplaçant par votre adresse de réseau.

```
from scapy.all import ARP, Ether, srp

trame = Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(pdst="192.168.100.0/24")
resultat = srp(trame, timeout=4,verbose=0)[0]

for envoi, reponse in resultat:
   print("{:16}    {}".format(reponse.psrc, reponse.hwsrc))
```

Pour exécuter le script : `sudo python3 monscan.py`

— Pour aller plus loin —

Utiliser les objets IP() et ICMP() pour faire un scan de niveau 3 en pinguant votre cible.

Si un firewall protège votre cible quel scan serait possible en niveau 3 qui pourrait le démasquer (ce type de scan existe avec nmap). Après avoir trouvé la réponse vous pouvez la programmer avec scapy. (la réponse sera donnée au prochain TPA)

— Petit complément pour clôturer ce TPA —

On a apprendre, depuis un terminal, à charger un fichier à partir d'une adresse web. Depuis un terminal du Raspberry Pi exécuter la commande suivante : `wget https://www.idreso.org/TPA/p3`
Il faut rendre le fichier exécutable : `chmod +x ./p3`

► ✉ :

Afin de nous donner un petit signe d'approbation de ce TPA, vous pouvez vous signaler en exécutant ce programme (`./p3`).

Au plaisir de vous retrouver pour le prochain TPA la semaine prochaine...