Travaux Pratiques – Rupture protocolaire et proxy

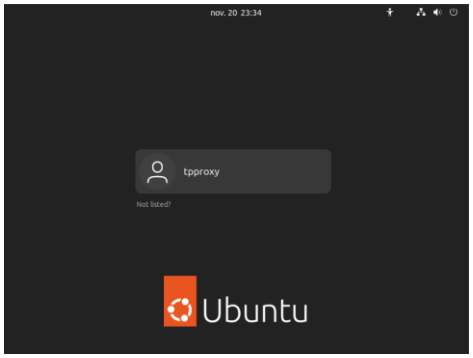
Prise en main de l'environnement de tests

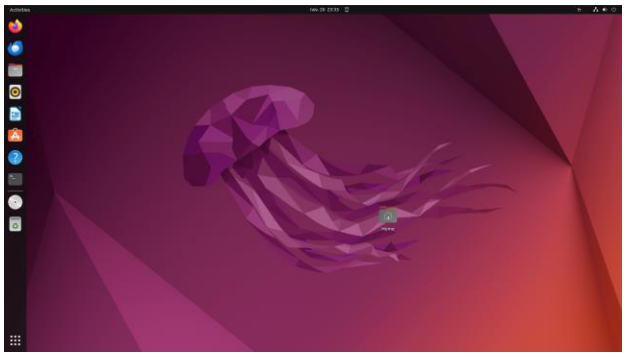
Les activités liées à ce TP sont à réaliser depuis la machine virtuelle Ubuntu Desktop dont les caractéristiques sont les suivantes :

- **Fichiers image VirtualBox** : *tp-proxy.vbox* + *tp-proxy.vdi*
- **CPU** : 4 vCPU
- **Mémoire** : 4 GB
- **OS** : Linux Desktop Ubuntu 22.04 LTS

Depuis l'hyperviseur VirtualBox, **démarrez la machine virtuelle** *tp-proxy* puis **ouvrez une session** avec les identifiants suivants :

```
Username: tpproxy
Password: enssatTP (/!\ clavier en qwerty par défaut)
```



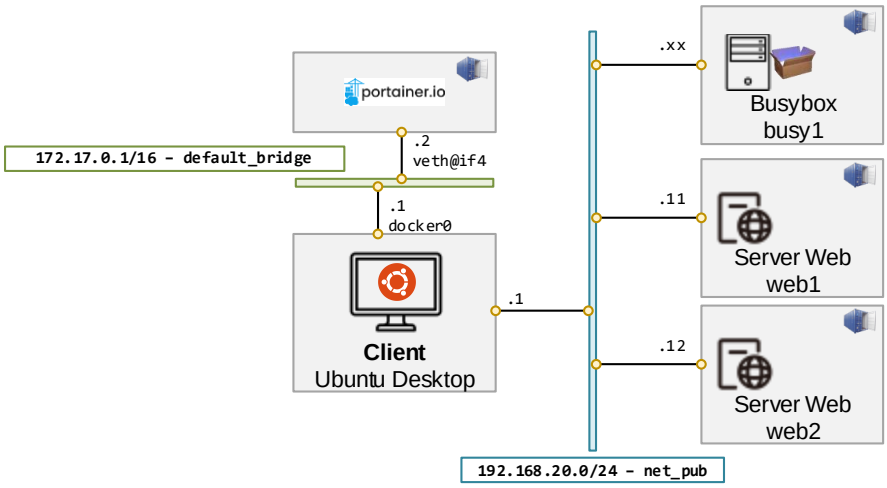


Docker est utilisé au cours de ce TP pour simuler les différents réseaux et services nécessaires.

Ouvrez un terminal puis **instanciez l'environnement de test** depuis le répertoire */home/tpproxy/exo1/*.

```
tpproxy@tp-proxy:~$ cd exo1
tpproxy@tp-proxy:~/exo1$ sudo docker-compose up -d
Creating network "exo1_net_pub" with the default driver
Creating exo1_web2_1  ... done
Creating exo1_web1_1  ... done
Creating exo1_busy1_1 ... done
```

L'environnement ainsi créé est composé de 3 containeurs : deux serveurs web et une Busybox reliés entre eux par un même réseau. Le schéma ci-après décrit l'architecture :



Listez les différents réseaux actuellement présents en utilisant la commande ci-dessous :

```
tpproxy@tp-proxy:~$ sudo docker network ls
```

Q1. Combien de réseaux sont actuellement créés ? Quels sont les différents types de réseaux listés par docker ?

A l'aide de la commande *docker inspect* **inspectez les propriétés** du réseau *net_pub*.

Q2. Quelle est l'adresse IP de la Busybox ?

Connectez-vous en CLI à la Busybox et **vérifiez son adresse IP**.

```
tpproxy@tp-proxy:~$ sudo docker exec -it <container_name> sh
```

Ensuite toujours depuis la CLI, **lancez un ping vers les containeurs** *web1* et *web2* en utilisant leur **adresse IP puis leur nom** (nom complet attribué par docker).

Q3. Que constatez-vous ? Quelle fonctionnalité est mise en avant en faisant ce test ?

Revenez au terminal du poste Ubuntu puis **lancez une capture réseau** pour obtenir dans un fichier « *.pcap* » les flux sur le réseau *net_pub* à l'aide de la commande *tcpdump*.

```
tpproxy@tp-proxy:~$ sudo tcpdump ...
```

Ensuite depuis le navigateur Firefox, **accédez à chacun des deux serveurs web** sur le port HTTP – TCP 80 : http://192.168.20.11 et http://192.168.20.12 (videz le cache navigateur au préalable). **Arrêtez la capture et commencez son analyse** à l'aide de Wireshark.

Q4. A partir de l'inspection des streams TCP/HTTP, quelles sont les informations notables dans les entêtes HTTP échangées avec les deux serveurs web ? Est-ce qu'il y une différence entre les deux serveurs Web ? Quels sont les codes de retours HTTP et qu'indiquent-ils ? Quelle est la réponse bonus ?

Avant de passer à l'étape suivante, **arrêtez/supprimez** les containeurs et le réseau.

```
tpproxy@tp-proxy:~/exo1$ sudo docker-compose down
Creating network "exo1_net_pub" with the default driver
Creating exo1_web2_1  ... done
Creating exo1_web1_1  ... done
Creating exo1_busy1_1 ... done
```

Mise en place d'un reverse proxy simple basé sur NGINX

NGINX sera utilisé dans la suite du TP en tant que Reverse Proxy.

A partir d'un terminal, **instanciez l'environnement de test** depuis le répertoire */home/tpproxy/exo2/*.

```
tpproxy@tp-proxy:~$ cd ~/exo2/
tpproxy@tp-proxy:~/exo1$ sudo docker-compose up -d
Creating network "exo2_net_pub" with the default driver
Creating network "exo2_net_priv" with the default driver
Creating exo2_rp1_1  ... done
Creating exo2_web1_1 ... done
```

Q5. Décrivez sous la forme d'un schéma l'environnement de test instancié en distinguant les différents éléments et les différents réseaux/adresses IP impliqués.

Connectez-vous en CLI au conteneur *rp1* NGINX et **lancez une capture réseau** sur toutes les interfaces pour récupérer dans un fichier **pcap l'intégralité des flux HTTP**.

```
tpproxy@tp-proxy:~$ sudo docker exec -it <container_name> sh
# tcpdump ...
```

Ensuite depuis le navigateur Firefox, **accédez au site web sur le port HTTP – TCP 80 via le reverse proxy** : http://192.168.20.2 (videz le cache navigateur au préalable).

Arrêtez la capture et **rapatrier la sur le poste Ubuntu** pour l'analyser avec Wireshark.

```
tpproxy@tp-proxy:~$ sudo docker cp exo2_rp1_1:/<xxx.pcap> ./
```

Q6. Représentez sous la forme d'un graphe de flux le déroulement des échanges (aidez-vous des fonctionnalités de Wireshark). Décrivez et expliquez les différents flux réseaux.

Réalisez depuis un terminal du poste Ubuntu **une connexion avec Telnet sur le port 80, requêtez la racine /**.

```
tpproxy@tp-proxy:~$ telnet 192.168.20.2 80
Trying 192.168.20.2...
Connected to 192.168.20.2.
Escape character is '^]'.
GET /
<html><body><h1>It works!</h1></body></html>
Connection closed by foreign host.
```

En CLI sur le reverse proxy ***rp1***, **parcourez le fichier des logs d'accès** (format des logs d'accès décrit dans le fichier *nginx.conf*).

```
tpproxy@tp-proxy:~$ sudo docker exec -it <container_name> sh
# more /var/log/nginx/rp1.access.log
```

Réalisez également la même chose sur le serveur ***web1*** (format des logs d'accès décrit dans le fichier *httpd.conf*).

```
tpproxy@tp-proxy:~$ sudo docker exec -it <container_name> sh
# more /usr/local/apache2/logs/access_log
```

Q7. Quelles sont les principales différences entre les logs de *rp1* et *web1* ? Quels champs sont manquants pour la connexion via Telnet ? Quelles informations pertinentes pourraient être ajoutées côté web1 vis-à-vis du client ?

En vous aidant de la documentation officielle NGINX (lien), **parcourez les fichiers de configurations** minimales *rp1-custom.conf* et *rp1-nginx.conf* (répertoire */home/tpproxy/exo2/*) qui ont été utilisé pour instancier l'environnement. **Identifiez** les éléments clés de configuration.

Complétez la configuration pour permettre l'ajout ou la modification d'entêtes HTTP *X-Real-IP*, *X-Forwarder-For* et *X-Forwarded-Proto*. **Relancez des tests** et confirmez le fonctionnement en analysant les entêtes HTTP depuis une nouvelle capture réseau.

Q8. Comparez les entêtes HTTP côté client et côté serveur *web1*. Quels sont les avantages d'une telle configuration ?

La documentation officielle reverse proxy NGINX propose une section « *Configuring Buffers* » : https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/

Q9. Quel est l'objectif de cette fonctionnalité ? Expliquez brièvement son fonctionnement.

Avant de passer à l'étape suivante, **arrêtez/supprimez** les containeurs et les réseaux.

Réécriture d'URL avec un reverse proxy basé sur NGINX

Les fonctions de réécriture d'URL sont des fonctionnalités apportées par le reverse proxy. Elles permettent par exemple de distinguer des accès, de masquer une topologie interne ou encore de faire des redirections sur mesure.

Rendez-vous dans le répertoire */home/tpproxy/exo3/* pour réaliser cet exercice.

L'objectif de cet exercice est de masquer à l'extérieur via un reverse proxy la topologie interne d'un site web. Voici les éléments à prendre en compte :

- Côté interne :
  - Deux serveurs web
  - *web1* héberge et expose avec le préfixe ***/internal*** la partie HTML du site web (i.e. URI */internal/index.html*)
  - *web2* héberge et expose avec le préfixe ***/images*** les fichiers images jpeg. A noter que les images sont nommées sous la forme ***img-<ref_number>.jpeg*** (e.g. URI */images/img-23.jpeg*)
- Côté externe :
  - Le reverse proxy doit exposer le site web sur le réseau *net_pub* avec le préfixe ***/external***
  - Le reverse proxy devra également rediriger les requêtes sur les images jpeg vers *web2* grâce au « *ref. number* »

Q10. En prenant en compte les éléments fournis, décrivez sous la forme d'un schéma l'environnement de test à instancier. Décrivez les réécritures à effectuer.

Q11. A partir de l'environnement de tests et du squelette de configuration fournis, mettez en œuvre les configurations requises. Répondez à la question posée sur le site web.

Q12. A partir d'une capture réseau sur *rp1*, représentez sous la forme d'un graphe de flux le déroulement des échanges. Observez les différentes requêtes HTTP.

Avant de passer à l'étape suivante, **arrêtez/supprimez** les containeurs et les réseaux.

Répartition de charge avec un reverse proxy basé sur NGINX

L'objectif de cet exercice est de mettre en place une répartition de charge entre des serveurs web au travers d'un reverse proxy. Plusieurs méthodes de répartition de charge ainsi que différents types de persistance de sessions seront testés.

Rendez-vous dans le répertoire */home/tpproxy/exo4/* pour réaliser cet exercice.

Q13. En prenant en compte les éléments fournis, décrivez sous la forme d'un schéma l'environnement de test à instancier.

Complétez la configuration pour instancier un **groupe de serveur** contenant *web1*, *web2* et *web3* avec une répartition de charge par défaut « *Round Robin* ».

Q14. Effectuez plusieurs accès sur le reverse proxy http://192.168.20.2. Quel comportement observez-vous ? Comment est réalisée la répartition de charge ?

Q15. Quelles sont les autres méthodes de répartition de charges proposées par NGINX ? Quelles sont leur fonctionnement ?

Modifiez votre configuration pour envoyer 60% des requêtes sur *web1* et 40% sur *web2*. **Positionnez** *web3* en tant que server de secours (« backup »).

Q16. Quelle est la configuration des poids par serveur ? Générez une dizaine de requêtes et confirmez le bon fonctionnement.

Arrêtez le conteneur *web2*.

Q17. Quel comportement observez-vous ?

Arrêtez le conteneur *web3*.

Q18. Quel comportement observez-vous ?

A partir de la documentation officielle : lien, **améliorez la configuration avec les options « *Passive health checks* »**.

Q19. Décrivez vos choix de configuration ? Testez votre configuration et observez les flux via une capture réseau.

Bien que NGINX Plus ne soit pas disponible dans notre environnement, **parcourez la documentation officielle relative aux « *Active health checks* »**.

Q20. Quelles sont les fonctionnalités apportées ? Quelles améliorations pourraient être mises en place dans notre cas ?

A partir de la documentation officielle : lien, **parcourez les différentes options de persistance de session**.

Q21. Quels sont le ou les intérêts d'une persistance de session ? Quels sont les principaux modes et fonctionnement offerts par NGINX ?

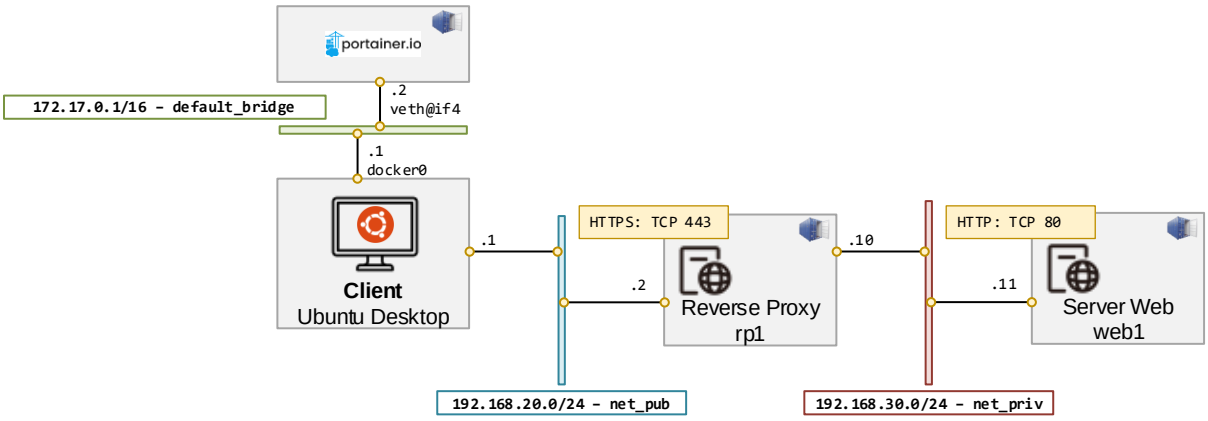
Testez la mise en place d'une persistance de type « *IP hash* ». **Analysez le comportement**.

Q22. Quel comportement observez-vous ?

Avant de passer à l'étape suivante, **arrêtez/supprimez** les containeurs et les réseaux.

Mise en place d'une terminaison SSL sur un reverse proxy NGINX

Afin de décharger les serveurs applicatifs (e.g. serveur web), le chiffrement est souvent déporté sur un équipement dédié comme un reverse proxy. L'environnement de test ci-dessous sera utilisé pour cet exercice.



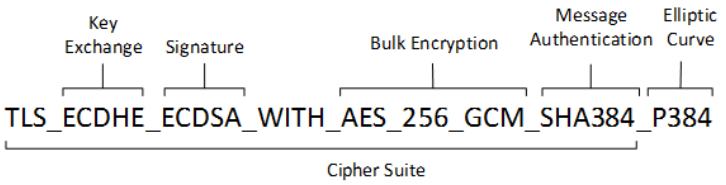
Rendez-vous dans le répertoire */home/tpproxy/exo5/* pour réaliser cet exercice.

L'objectif est de **mettre en place une connexion chiffrée HTTPS (SSL/TLS)** entre le client et le reverse proxy *rp1*. **Complétez la configuration** en prenant en compte les éléments ci-dessous :

- Certificat autosigné à l'aide d'OpenSSL (par simplification) ;
- Certificat au format X509 ;
- Clé privée 2048 bits RSA dans un fichier dédié ;
- Perfect Forward Secrecy Diffie-Hellman group 4096 bits ;
- Suite de chiffrement : « HIGH:!aNULL:!MD5 » ;
- TLS v1.2.

Q23. A partir du navigateur web, réalisez un test d'accès et confirmez les différents paramètres liés à la connexion sécurisée SSL/TLS (version protocole, cipher suite…)

Q24. A partir d'une capture réseau sur *rp1*, analysez et décrivez les échanges TLS v1.2. Combien d'échanges sont nécessaires pour établir la session TLS ? Combien de suite de chiffrement sont proposées dans l'échange ? Quelle suite a été retenue ?



Changez votre configuration reverse proxy pour prendre en compte les **suites de chiffrement** suivantes :

```
ssl_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384;
ssl_ecdh_curve secp384r1;
```

Q25. Quelle suite est maintenant retenue ? Vérifiez également le changement du groupe d'échange de clé « Key exchange group ».

Changez votre configuration reverse proxy pour forcer l'utilisation du **TLS v1.3**. Avant de tester à nouveau, **fermez et videz le cache de votre navigateur**.

Q26. A partir d'une capture réseau sur *rp1*, analysez les échanges TLS v1.3. Que demande le serveur via le message « Hello Retry Request » ?

Revenez à la configuration initiale des suites de chiffrement comme indiqué ci-dessous et **supprimez la configuration liée au « *ssl_ecdh_curve* »**.

```
ssl_ciphers HIGH:!aNULL:!MD5;
```

Q27. A partir d'une capture réseau sur *rp1*, analysez et décrivez les échanges TLS v1.3. Combien d'échanges sont nécessaires pour établir la session TLS ? Quel est le gain observé ?

Avant de passer à l'étape suivante, **arrêtez/supprimez** les containeurs et les réseaux.

Navigation web via un proxy Web – Squid

La navigation web sur internet depuis un réseau d'entreprise ou autre structure n'est quasiment jamais réalisée en direct. Les flux tels qu'HTTP/HTTPS doivent passer au travers de proxys afin d'y être contrôlés et filtrés.

L'objectif de cet exercice est de mettre en place un filtrage simple à l'aide du proxy-cache web Squid : lien et d'analyser notamment la gestion d'une session TLS.

Rendez-vous dans le répertoire */home/tpproxy/exo5/* pour réaliser cet exercice.

Parcourez les directives dans le fichier de configuration du proxy squid (fichier *proxy_squid.conf*, lignes non commentées).

Quelques directives :

- *http_port* : correspond à l'adresse et le port d'écoute de Squid ;
- *coredump_dir* : spécifie le répertoire dans lequel Squid peut écrire ses fichiers d'erreur ;
- *refresh_pattern* : indique les règles qui établissent si un fichier est "frais" ou "périmé". Un fichier "périmé" est supprimé du cache ;
- *acl* : définit des critères de contrôle d'accès (Access Control List) ;
- *http_access* : permet d'autoriser ou interdire les connexions (e.g. HTTP). Ces directives sont lues dans l'ordre d'apparition et sont exécutées dès qu'une règle correspond.

Q28. En quelques mots, décrivez la politique de sécurité présente dans le fichier ? Quel est l'intérêt de la dernière directive *http_access* ?

Instanciez l'environnement de test mis à disposition puis **configurez le navigateur web afin d'utiliser le proxy** pour les flux HTTP et HTTPS.

Confirmez le bon fonctionnement.

Q29. A partir d'une capture réalisée sur l'interface virtuelle du proxy, analysez la séquence des flux pour un accès au site https://www.perdu.com. Représentez sous forme d'une séquence de flux simplifiée les échanges. Comment est établie la session TLS ?

Modifiez la configuration pour bloquer le domaine *youtube.com*. Pour plus de facilité, vous pouvez utiliser le fichier *custom-px.conf*.

Q30. Lors d'une tentative d'accès, que pouvez-vous constater dans les échanges avec le proxy ?

Modifiez la configuration pour **autoriser** le domaine *youtube.com* **uniquement sur une plage horaire**. Testez votre configuration en ajustant la plage horaire.

Pour aller plus loin… filtrage avancé, gestion du cache, module ICAP pour filtrage antivirus (e.g. ClamAV) ou catégorisation d'URL.

Avant de passer à l'étape suivante, **arrêtez/supprimez** les containeurs et les réseaux.

Génération et capture de flux réseau avec Scapy

Scapy est un programme écrit en Python permettant d'envoyer, de capturer, d'analyser et de forger des paquets réseau. C'est une boite à outils tout-en-un qui permet de remplacer de nombreux utilitaires tels que hping, arpspoof, arp-sk, arping, p0f, Nmap ou encore tcpdump.

Scapy peut être directement utilisé via sa console interactive ou importé en tant que librairie dans un programme.

➔ Plus de détails sur le site et documentation officiels : https://scapy.net/ – Documentation

Rendez-vous dans le répertoire */home/tpproxy/exo7/* pour réaliser cet exercice.

Activez l'environnement Python présent dans « *venv* ».

```
tpproxy@tp-proxy:~/exo7$ source venv/bin/activate
(venv) tpproxy@tp-proxy:~/exo7$
```

Avant de lancer Scapy, **instanciez à nouveau l'environnement de l'exercice 1** (répertoire */home/tpproxy/exo1*). A noter, que si vous changez la configuration réseau (i.e. bridge, interface containeur…) il faudra relancer l'interpréteur Scapy pour prendre en compte les changements.

Lancez Scapy à partir de la ligne de commande ci-dessous.

/!\ Veillez à bien utiliser la version de Scapy 2.5.0 présente dans l'environnement virtuel.

```
(venv) tpproxy@tp-proxy:~/exo7$ sudo venv/bin/scapy
WARNING: IPython not available. Using standard Python shell instead.
AutoCompletion, History are disabled.
                    aSPY//YASa
            apyyyyCY//////////YCa       |
           sY//////YSpcs  scpCY//Pp     | Welcome to Scapy
 ayp ayyyyyyySCP//Pp           syY//C    | Version 2.5.0
 AYAsAYYYYYYYY///Ps              cY//S   |
         pCCCCY//p          cSSps y//Y   | https://github.com/secdev/scapy
         SPPPP///a          pP///AC//Y   |
              A//A            cyP////C   | Have fun!
              p///Ac            sC///a   |
              P////YCpc           A//A   | Craft packets like it is your last
       scccccp///pSP///p          p//Y   | day on earth.
      sY/////////y  caa           S//P   |                     -- Lao-Tze
       cayCyayP//Ya              pY/Ya   |
        sY/PsY////YCc          aC//Yp
         sc  sccaCY//PCypaapyCP//YSs
                  spCPY//////YPSps
                       ccaacs
>>>lsc()
```

Quelques commandes utiles :

```
# Lister les commandes utilisateur
>>> lsc()
# Description d'une commande
>>> help(<command_name>)
# Voir la table de routage de Scapy
>>> conf.route
```

Entrez les commandes suivantes et **analysez les résultats** :

```
>>> pkt = sr1(IP(dst="192.168.20.11")/ICMP()/"xyzXYZ")
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
>>> pkt
>>> pkt.show()

>>> answer, unanswer = sr(IP(dst=["192.168.20.11",
"192.168.20.15"])/TCP(dport=[80,443,8080]), timeout=1)
>>> answer.summary()
>>> unanswer.summary()
```

Q31. A l'aide de la documentation officielle, décrivez la composition et les actions réalisées par ces commandes. Quelles informations est-il possible de déduire des réponses réseaux ?

Réalisez un nouveau test avec la commande *srp()* :

```
>>> answer, unanswer = srp(Ether()/IP(dst=["192.168.20.11",
"192.168.20.15"])/TCP(dport=[80,443,8080]), timeout=1, iface='<net_pub bridge>')
>>> answer.show()
>>> unanswer.show()
```

Q32. Quelle est la différence entre les commandes *sr()* et *srp()* ?

Recherchez sur le réseau *net_pub* les hôtes présents à l'aide d'un scan ARP (ou « ARP ping »). Puis, **testez les ports TCP associés à HTTP pour chacun des hôtes** à l'aide d'un « *TCP Null scan* » ou d'un « *TCP XMAS scan* ».

Q33. Listez les commandes utilisées pour générer les requêtes. Quels sont les hôtes présents et lesquels ont un port HTTP d'ouvert ? Quelles sont les spécificités propres aux « *TCP Null scan* » et « *TCP XMAS scan* » ?

A l'aide de la commande *sniff*, **lancez une capture réseau** pour capturer le trafic HTTP à destination de *web1*, puis **accédez à** http://192.168.20.11 depuis Firefox. Arrêtez la capture avec CTRL^C et analysez les échanges.

```
>>> pkts = sniff(iface="<net_pub bridge>", filter="port 80 and host 192.168.20.11")
^C
>>> pkts.show()
>>> pkts[10].show()
```

Q34. Améliorez cette commande pour afficher en continu le trafic HTTP à destination de *web1* et *web2* sous cette forme :

IP Source:port TCP source ==> IP Destination:port TCP destination

Raw Data

A partir de la documentation officielle, parcourez quelques-unes des possibilités offertes par Scapy (e.g. scan, arp poisoning, vlan hopping…) :

➔https://scapy.readthedocs.io/en/latest/usage.html#simple-one-liners

Réalisez à l'aide de Scapy une requête HTTP sur la racine de *web1*. **Affichez et analysez la réponse**. Vous pouvez constater une soixantaine d'attributs dans l'entête HTTP.

Q35. Listez les commandes utilisées pour générer les requêtes.

Avant de passer à l'étape suivante, **arrêtez/supprimez** les containeurs et les réseaux liés à l'environnement de l'*exo1*. Puis **instanciez à nouveau l'environnement de l'*exo5*** (répertoire */home/tpproxy/exo5*). Vos devrez ré-utiliser vos configurations TLS 1.2 (Q24) et TLS 1.3 (Q27).

Q36. Réalisez à nouveau les analyses effectuées dans les questions Q24 à Q27 à l'aide de Scapy.

De plus, à partir de la session TLS v1.2, naviguez dans la capture pour afficher en détail votre certificat auto-signé.

Listez les commandes utilisées pour générer les requêtes.