Ensimag 1ère année

TP n°1 — Sécurité réseau

Il est recommandé de prendre des notes. Se référer à la version numérique de ce document pour les liens (sur Chamilo).
Une question sur l'effet ou l'utilisation d'une commande ?
`man nom_de_la_commande` !!

1 Utilisation de SSH

L'objectif de cette partie de TP est d'étudier l'outil de connexion à distance `ssh` et l'authentification sous-jacente. Vous découvrirez aussi quelques utilitaires associés à `ssh`, comme `scp`, `ssh-keygen`, `ssh-copy-id`.

L'utilitaire `ssh` permet d'ouvrir une session de terminal à distance sécurisée. Si vous avez des problèmes au cours de cet exercice, consultez `man ssh`. Les répertoires à retenir : `$HOME/.ssh` sur la machine cliente et serveur, ainsi que `/etc/ssh` sur la machine serveur (voir aussi la section « FILES » dans la page du manuel de `ssh`). Ces répertoires contiennent entre autres les différentes clés publiques et privées.

Vous pouvez vous reporter au schéma du cours donnant l'exemple d'une connexion ssh avec les fichiers sur la machine locale et la machine distante.

Les informations les plus utiles se trouvent dans la documentation de chaque outil ainsi que sur les pages `man` (`ssh-keygen`...). D'autres informations peuvent se trouver dans des tutoriels, sur Wikipédia ou via des moteurs de recherche.

Dans la suite du TP, lorsqu'on vous demande d'ouvrir une session sur `ensipcNNN.ensimag.fr`, utilisez l'outil `ssh` (ex : `ssh votre_login@ensipcNNN.ensimag.fr`). L'indication du nom de login `login@` est optionnelle (non nécessaire) si vous avez le même nom de login sur les deux machines. De même, comme vous l'avez vous pour le DNS, il n'est pas nécessaire de donner un nom de machine distante complet avec `.ensimag.fr` (FQDN) si vous êtes dans le même domaine.

Il sera utile en général d'avoir deux fenêtres de type "terminal" : une dans laquelle vous ne ferez pas de SSH pour voir ce qui se passe sur votre PC local, et une autre dans laquelle vous travaillerez sur une machine distante après y avoir lancé une commande SSH.

1.1 Premiers pas

Q 1 *Connectez-vous par SSH au PC indiqué par votre enseignant (ou à défaut à celui de votre voisin) par `ssh -K ensipcNNN.ensimag.fr`. Quelle différence observez-vous sur le souffleur (prompt) de votre shell, i.e. les caractères en début de ligne à gauche de l'invite de commande juste avant là où vous allez pouvoir taper votre commande ?*

Q 2 *Vérifiez qui est actuellement connecté à ce serveur, à l'aide de la commande « `w` ». Comparez avec votre propre machine dans un autre terminal.*

Q 3 *Essayez de lancer un programme graphique comme `xclock` (ou en plus rigolo, `xeyes` qui permet de suivre le pointeur de la souris) d'abord sur un terminal de votre machine, puis sur le PC distant par le shell par lequel vous vous êtes connecté en ssh. Cela fonctionne-t-il ?*

Q 4 *Trouvez l'option de `ssh` qui active l'affichage graphique à travers le réseau (chercher « X11 forwarding » dans le `man ssh`). Réessayez ensuite de lancer `xclock` sur `pcserveur`. N'oubliez pas de vous déconnecter de la session SSH entre chaque essai de connexion !*

Q 5 *Dans un autre terminal sur votre machine, essayez de terminer ce processus `xclock` grâce à la commande `killall`. Cela fonctionne-t-il ? Pouvez-vous en déduire sur quelle machine s'exécute le processus `xclock` ?*

Pensez bien à vous déconnecter de pcserveur

1.2 Authentification du serveur distant

Pour démarrer d'un environnement vide pour SSH **sur votre machine** (requis pour cette partie du TP), vous devez exécuter la commande :

```
mv ~/.ssh ~/.ssh.sav
```

À la fin de la séance, vous pourrez au choix conserver l'environnement du TP et supprimer l'ancien (`rm -r ~/.ssh.sav`) ou restaurer l'ancien (`rm -r ~/.ssh; mv ~/.ssh.sav ~/.ssh`).

Ouvrez une session SSH sur `gitlab.ensimag.fr`, sans préciser de login. N'acceptez pas la connexion immédiatement.

Le serveur `gitlab.ensimag.fr` utilise une clé publique ED25519, dont les empreintes SHA256 et MD5 sont (en hexadécimal) :

```
ED25519 key fingerprint is SHA256:1WMnjPktwTkHJe3ZkK16iIjWZZRE5x4/24hEBHWLsIg.
ED25519 key fingerprint is MD5:c4:12:90:e4:f1:ca:d5:16:74:68:33:83:c8:90:41:d7.
```

On obtient l'empreinte md5 (deprecated depuis 2008) avec `ssh -o FingerprintHash=md5 gitlab.ensimag.fr`.

Q 6 *Quel est le nombre de bits de l'empreinte MD5 ?*

Q 7 *Quelle est la question que l'on vous pose pour savoir si vous acceptez ou non la connexion ? A quelle condition acceptez-vous ?*

Q 8 *Si les empreintes de la clé publique de l'énoncé et celles données sur votre ordinateur sont égales, que peut-on en déduire (en supposant que la cryptographie est robuste et n'a pas été compromise, par exemple par un vol de clés) ? Si elles sont différentes, qu'a-t-il pu se passer (au moins 2 possibilités) et quelles sont les conséquences si l'on saisit ses identifiants ?*

Q 9 *Après avoir accepté la connexion (*), quelles modifications ont été apportées au fichier `~/.ssh/known_hosts` sur votre machine cliente ?*

(*) Nota Bene : En fait, vous ne pouvez pas aller au bout d'un login sur gitlab, car vous n'avez pas de compte de login vous permettant d'ouvrir un shell sur gitlab. Ainsi la suite de ssh échouera car il n'y a pas de login à votre nom, donc quel que soit le mot de passe que vous mettiez, vous ne pourrez pas en donner un valide et vous n'aurez pas de shell. Donc à la demande de mot de passe, répondez par CTRL-C pour arrêter là votre tentative de connexion.

L'utilitaire ssh-keygen (grâce à la commande : `ssh-keygen -l -f <fichier clé>`) permet d'afficher l'empreinte de la clé contenue dans le fichier `<fichier clé>`.

Q 10 *Utilisez la commande* `ssh-keygen` *avec les bonnes options pour afficher l'empreinte de la clef pour* `gitlab.ensimag.fr` *stockée dans* `~/.ssh/known_hosts` *et vérifiez que cette empreinte est toujours celle qui vous a été fournie précédemment.*

L'utilitaire `ssh-keygen` génère par défaut une paire de clés RSA (équivalent à l'option `-t RSA`) de taille 3072 bits (minimum 1024)

Q 11 *Générez une paire de clés RSA de taille 4096 bits. Lorsque l'utilitaire vous demande où stocker les clés générées, vous pouvez laisser le chemin proposé par défaut (il suffit de valider avec Entrée).*

N.B. : La passphrase est un "mot de passe" (généralement un texte de 4 mots donne un mot de passe robuste et facile à mémoriser sans le noter) qui permet de chiffrer/déchiffrer localement votre clé privée. En effet, la clé privée est stockée dans un fichier de votre compte (.ssh/id_rsa par défaut), pour que les applications comme ssh puissent la lire lorsqu'elles en ont besoin. Mais cela fait qu'un indiscret (profitant de votre compte ouvert, ou un root non scrupuleux) pourrait la lire et la copier. Donc la clé elle-même peut être chiffrée, pour que ce qui reste stocké sur l'ordinateur ne puisse être utilisé en votre absence. La passphrase vous sera demandée par ssh pour "déverrouiller" votre clé, i.e. la déchiffrer avant de l'utiliser. Si vous décidez de mettre une passphrase sur votre clé privée, il ne faudra pas l'oublier, sinon votre clé privée deviendra inutilisable. Si vous décidez d'appuyer sur la touche d'entrée lors de cette étape, aucune passphrase ne sera utilisée, et votre clé privée ne sera pas chiffrée sur votre disque dur, et donc n'importe quelle personne qui aurait accès à votre ordinateur ou votre compte pourrait utiliser cette clé et se faire passer pour vous.

Q 12 *Vérifiez le contenu du dossier où vos nouvelles clés ont été stockées. Quels fichiers ont été générés et quel est le rôle de chacun ?*

Q 13 *Générez une paire de clés de taille 16385 bits. Que se passe-t-il ?*

Q 14 *Générez une paire de clés RSA de taille maximum. Que remarquez-vous ?*

Q 15 *Générez une paire de clés par défaut (sans spécifier de taille) utilisant le système de signature ed25519 (car pour ed25519 la taille est fixe)*

Q 16 *Comparez la taille des clés publiques générées avec RSA et ed25519*

N.B. Nous verrons plus loin que l'utilitaire `openssl` permet non seulement de créer des clés comme `ssh-keygen`, mais aussi de réaliser d'autres opérations cryptographiques.

1.3 Authentification de l'utilisateur par clés asymétriques

Pour l'instant, vous pouvez vous connecter à `ensipcNNN.ensimag.fr` en donnant votre mot de passe. Nous allons maintenant utiliser un mécanisme évitant de saisir votre mot de passe, par une authentification avec une clé publique (et la clé privée associée) que vous allez vous créer.

Q 17 *Fabriquez une paire de clés RSA à l'aide de l'utilitaire* `ssh-keygen`*. Vous pouvez également utiliser la paire de clés RSA de taille 4096 bits fabriquée précédemment.*

Nous allons maintenant transférer votre clé publique sur `ensipcNNN.ensimag.fr` afin que celui-ci puisse l'utiliser afin de vous authentifier. Pour cela, nous allons utiliser la commande `ssh-copy-id`.

D'après le `man` (allez le consulter !), la commande `ssh-copy-id` « use locally available keys to authorise logins on a remote machine », ou, en d'autres termes, va copier votre clé publique au bon endroit sur l'hôte distant. La section `DESCRIPTION` donne des détails sur sa façon de procéder.

Q 18 *Utilisez la commande* `ssh-copy-id` *pour copier votre clé publique sur* `ensipcNNN.ensimag.fr`*.*

Vérifiez ensuite que lorsque vous essayez de vous connecter à `ensipcNNN.ensimag.fr`*, vous n'avez plus besoin d'utiliser de mot de passe.*

Q 19 *Sur* `ensipcNNN.ensimag.fr`*, regardez le contenu de* `$HOME/.ssh/authorized_keys` *et comparez le avec le contenu de votre clé publique. Que remarquez-vous ?*

Q 20 *Revenez maintenant sur votre machine, et ouvrez de nouveau une connexion cette fois-ci en mode "verbeux" par* `ssh -v ensipcNNN.ensimag.fr` *tout en regardant ce qui se passe avec Wireshark. Identifiez la suite des opérations faites par votre client ssh, et notamment l'utilisation de votre clé privée pour vous authentifier auprès du serveur. Vous pourrez aussi vous référer à la diapositive du cours sur le déroulement du protocole ssh.*

INFO : L'utilisation d'une passphrase non vide chiffre votre clé privée, ce qui fait que le client ssh est obligé de vous demander votre passphrase pour répondre au défi d'authentification. Pour éviter d'avoir à saisir cette passphrase à chaque nouvelle connexion ssh, vous pouvez consulter les commandes `ssh-agent`, puis `ssh-add`, et rajouter ces commandes dans votre fichier `.bash_profile`.

1.4 Ajout de clés SSH sur GitLab

Connectez-vous sur votre compte sur le gitlab de l'ensimag (`https://gitlab.ensimag.fr`), puis dans les réglages de votre compte, ajoutez votre clé publique pour que Gitlab autorise celle-ci.

Q 21 *Connectez-vous à GitLab en SSH* ***avec l'utilisateur*** `git` *en utilisant cette clé. Cela fonctionne-t-il ?*

2 Utilisation de OpenSSL

L'objectif de cette partie de TP est d'étudier l'outil OpenSSL une implémentation d'une grande variété de primitives cryptographiques symmétriques et asymétriques.

L'outil en ligne de commande est `openssl` suivi d'une commande (dont la liste s'obtient par `openssl --help`).

Pour lister les options possibles pour une commande et l'aide associée, ajouter l'option `-h`. Par exemple `openssl genrsa -h` vous donnera la liste des options pour la commande `genrsa` en charge de la génération de clés RSA.

2.1 Premiers pas : étude des composants d'une clé

Q 22 *Générez une clé (privée) RSA de 2048 bits dans un fichier* `privkey.pem`.

Vous pouvez protéger le fichier de votre clé privée RSA ainsi généré en le chiffrant lui-même (ce fichier) par un chiffrement de votre choix utilisant comme clé de protection une passphrase à saisir en mode interactif. Par exemple nous allons produire un nouveau fichier `privkey.pem` mais qui sera cette fois chiffré avec l'algorithme aes256 (alors qu'il était en clair précédemment lors de la question précédente).

Q 23 *Générez maintenant une clé privée RSA de 2048 bits chiffrée en aes256, que vous stockerez dans un fichier* `privkey2.pem`.

Revenons à la première clé privée RSA (celle que vous avez produite en clair).

Q 24 *La commande* `rsa` *permet la manipulation et l'inspection de clés RSA. Affichez le modulo de votre clé. Vérifiez sa taille en bits.*

Q 25 *Quel est l'exposant public de chiffrement ?*

Vous pouvez maintenant générer votre clé publique correspondante à partir de votre clé privée avec la commande

```
openssl rsa -in privkey.pem -pubout -out pubkey-rsa-<VOTRE LOGIN>.pem
```

2.2 Signatures et vérifications

La commande `dgst` permet de gérer les fonctionnalités de hachage et de signature.

Q 26 *Récupérez dans le dossier* `/matieres/3MMSECU/TP1/OpenSSL` *de* `pcserveur.ensimag.fr` *les 3 fichiers* `v1.txt,v2.txt` *et* `v3.txt`. *Parmis ces trois fichiers, un seul est authentique et a été signé par l'un de vos professeurs dans le fichier signature* `sign-pernetc.bin`. *Sa clé publique RSA est disponible dans ce dossier :* `pubkey-rsa-pernetc.pem`

En utilisant la commande `openssl dgst -verify`, *retrouver lequel de ces 3 fichiers est authentique.*

3 Manipulations en machine virtuelle

Dans la suite de ce TP, il sera de temps en temps nécessaire de manipuler en `root`. Vous allez donc manipuler dans une machine virtuelle (VM) lancée depuis votre compte Ensimag habituel.

Pour lancer la VM, la commande à utiliser est :

```
/matieres/3MMIRC9/lance-vm-3MMIRC.sh
```

Puis sélectionner "use a temporary machine folder" (option 1). La machine virtuelle démarre alors sur une interface graphique simplifiée. Aucun mot de passe n'est demandé, et vous êtes automatiquement connecté à l'interface graphique avec l'utilisateur `root`. Il ne vous reste donc plus qu'à ouvrir un terminal pour pouvoir manipuler en `root` (Cliquez sur `Activities` en haut à gauche de votre VM, puis tapez `Terminal` dans la barre de recherche).

Si jamais vous avez besoin du mot de passe de la VM, par exemple pour vous connecter en SSH, le mot de passe associé au compte `root` est "`root./`".

4 Etude d'un pare-feu sous Linux

Sauf indication contraire, dans cette partie du TP, toutes les manipulations doivent être effectuées dans la machine virtuelle et non sur votre compte Ensimag habituel.

L'objectif de cette partie de TP est de comprendre le fonctionnement d'un pare-feu (*firewall*), à travers l'usage d'`iptables` sous Linux.

Rappel : il faut être `root` pour configurer le pare-feu. N'oubliez donc pas de travailler à l'intérieur de la VM.

4.1 Exploration

Le but d'un pare-feu est de classifier les paquets qui entrent et sortent d'une machine, puis d'appliquer des *actions* sur chaque paquet, selon des règles de politique pré-établies. Généralement, les actions se limitent à laisser passer ou bloquer le paquet, mais il est possible de mettre en place des actions plus complexes.

Dans ce TP, nous nous intéresserons uniquement aux paquets générés ou reçus par la machine locale. Un routeur pourrait également agir comme pare-feu sur les paquets qu'il route, mais ce n'est pas le but du TP.

Q 27 *Affichez l'état actuel du pare-feu grâce à `iptables -L -n`. Combien y a-t-il de "chaînes" ? D'après leur nom, sur quels paquets s'applique chaque chaîne ? Repérez la politique par défaut pour chaque chaîne (`POLICY`).*

Une bonne pratique de sécurité consiste à bloquer par défaut tous les paquets, et n'autoriser que les types de paquets souhaités.

Q 28 *Changez la politique par défaut des chaînes `INPUT` et `OUTPUT` de façon à bloquer tous les paquets entrants et sortants :*

```
# iptables -P INPUT DROP
# iptables -P OUTPUT DROP
```

4.2 Premières politiques de pare-feu

Nous allons commencer par autoriser le protocole ICMP, puisqu'il est indispensable pour gérer les cas d'erreurs et pour pouvoir tester le bon fonctionnement d'un réseau avec `ping`. Pour cela, nous allons utiliser l'option `-A` d'`iptables` pour rajouter (A*ppend*) des règles :

```
# iptables -A INPUT -p icmp -j ACCEPT
# iptables -A OUTPUT -p icmp -j ACCEPT
```

Q 29 *Ajoutez ces règles et testez un `ping` vers `129.88.47.4`. Cela fonctionne-t-il ? Et un ping vers `delos.imag.fr` ? Expliquez la différence de comportement : quel(s) protocole(s) sont mis en jeu dans chacun de ces deux cas ?*

Q 30 *Obtenez des statistiques sur le pare-feu grâce à `iptables -L -n -v` : cela affiche pour chaque règle le nombre de paquets traités. Vérifiez que plusieurs paquets ICMP ont utilisé les règles rajoutées précédemment, et que plusieurs paquets ont été bloqués. Dans quel(s) cas la politique par défaut est-elle utilisée ?*

Nous aimerions également autoriser l'utilisation de DNS, qui fonctionne avec le protocole UDP sur le port 53. Pour cela, ajoutez la règle suivante :

```
# iptables -A OUTPUT -p udp --destination-port 53 -j ACCEPT
```

Remarquez la nouvelle option `--destination-port`, qui est spécifique à UDP. Les options de base d'`iptables` sont décrites dans `man iptables`, tandis que les options spécifiques à chaque protocole (par exemple `--destination-port` ici) sont décrites dans `man iptables-extensions`.

Q 31 *Est-ce que le ping vers `delos.imag.fr` fonctionne mieux ? Si non, quelle pourrait être la cause ? Utilisez Wireshark et `iptables -L -n -v` pour essayer de comprendre d'où vient le problème.*

Q 32 *Ajoutez la règle manquante pour résoudre le problème.* ***Indice :*** *il est nécessaire d'utiliser une nouvelle option spécifique au protocole UDP.*

Pour aller plus loin...

Partie OPTIONNELLE uniquement pour les étudiants voulant aller plus loin après avoir maîtrisé les parties précédentes.

Jusqu'ici, nous avons traité de la sécurité défensive. Pour eux qui veulent tâter de la sécurité "offensive", voici une manipulation d'usurpation d'identité de machine.

5 Usurpation ARP (optionnel)

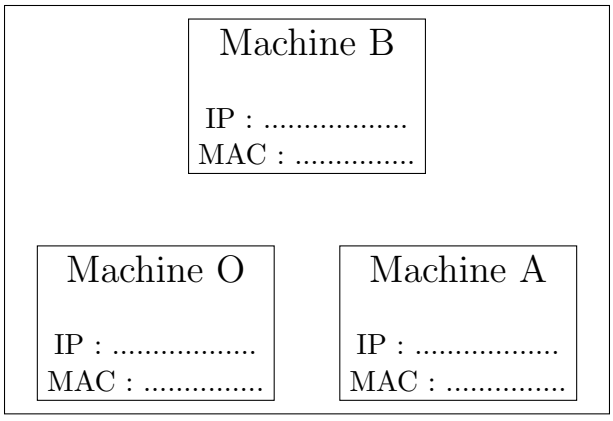
L'attaque que nous allons réaliser durant ce TP est présentée seulement dans un cadre pédagogique. Vous devez absolument penser aux conséquences légales si vous décidez de la mettre en place dans des réseaux qui ne vous appartiennent pas.

L'objectif de cette partie de TP est de réaliser une usurpation ARP simple. Pour cela, vous allez avoir besoin de trois machines en réseaux, dont deux seront virtuelles :

1. La machine `A`, qui cherchera à contacter `B`,
2. La machine `B`, qui attendra un message de `A`,
3. La machine `O`, qui se fera passer pour `B` aux yeux de `A`, et interceptera donc le message.

5.1 Mise en place

Pour ne pas s'embrouiller dans cette partie, il est important d'être organisé, car nous aurons trois machines *sur un même écran*. Si vous vous trompez de machine lors de l'exécution d'une commande, cela n'aura pas l'effet escompté. L'idée est d'avoir les terminaux des trois machines visibles en même temps, et de ne plus déplacer les fenêtres, selon la configuration (que vous remplirez dans les questions à venir) suivante :



Nous commençons par relancer la machine `O`, qui correspond à la VM utilisée pour la partie sur les pare-feu.

Si vous ne la relancez pas, remettez les politiques du pare-feu par défaut à `ACCEPT`, car sinon vos paquets seront bloqués.

Q 33 *Relancez la machine virtuelle avec la commande* `/matieres/3MMIRC9/lance-vm-3MMIRC.sh` *en choisissant à nouveau l'option 1.*

Placez la fenêtre de la machine virtuelle en bas à gauche comme sur le schéma, ouvrez un terminal et utilisez la commande `ifconfig` *pour trouver l'adresse IP et l'adresse MAC de l'interface* `enp0s8`*.*

Nous lançons maintenant la machine `A`.

Lancez `/matieres/3MMIRC9/lance-vm-3MMIRC-prime.sh` (**Attention à bien lancer -prime**).

Q 34 *Placez la fenêtre en bas à droite comme sur le schéma, ouvrez un terminal et utilisez la commande* `ifconfig` *pour trouver l'adresse IP et l'adresse MAC de l'interface* `enp0s8`*.*

Finalement, la machine hôte sera la machine `B`.

Q 35 *Hors des machines virtuelles, ouvrez un terminal, placez le en haut de l'écran comme sur le schéma, et utilisez la commande* `ifconfig` *pour trouver l'adresse IP et l'adresse MAC de l'interface* `vboxnet0`*.*

Q 36 *Faites votre schéma représentant les trois machines avec leur adresse IP et MAC, en reprenant la position des terminaux sur l'écran. Sur chaque machine, regardez le cache ARP avec la commande* `arp`*.* ***Considérez uniquement les entrées du cache ARP correspondant à l'interface qui vous intéresse*** *(*`vboxnet0` *ou* `enp0s8`*).*

Q 37 *Lancez une capture* `wireshark` *sur la machine* `B`*, puis testez la connection en faisant un* `ping` *de la machine* `A` *vers la machine* `B`*. Regardez les paquets ARP capturés, et inspectez à nouveau le cache ARP de la machine* `A`*. Vérifiez que les informations sont cohérentes.*

5.2 Usurpation d'identité

Nous allons voir comment dans un protocole de communication "naif" un serveur est susceptible de se faire usurper son identité, et comment `ssh` empêche cela.

Q 38 *Sur les machines* `B` *et* `O`*, écoutez sur le port 8081 avec la commande* `nc -l -p 8081`*. Depuis la machine* `A`*, établissez une connection TCP à la machine* `B` *sur le port 8081 avec la commande* `telnet <ipMachineB> 8081`*. Envoyez par ce canal TCP un message confidentiel d'une grande importance, et vérifiez qu'il apparaît bien sur la machine* `B`*.*

Nous allons maintenant usurper l'identité de la machine `B` avec la machine `O`. Pour cela, nous allons envoyer à `A` un paquet ARP disant que l'adresse IP de `B` est à l'adresse MAC de `O`. Pour falsifier un tel paquet, nous allons avoir besoin d'un outils en ligne de commande de falsification de paquets nommé *scapy*.

Q 39 *Sur la machine* `O`*, installez scapy avec la commande* `apt install python-scapy`*. Une fois l'installation terminée, lancez le avec la commande* `scapy`*.*

Scapy est en fait une bibliothèque pour le langage de programmation python. Vous allez créer un paquet et le stocker dans une variable, en spécifiant les couches du paquet, avec des commandes comme `packet = Ether()/ICMP()` ou `packet = Ether()/IP()`. Pour spécifier les valeurs des champs de chaque couche, vous avez deux possibilités :

1. Vous pouvez passer les valeurs en argument des couches, par exemple :
   `packet = IP(dst="<ip destination>", src="<ip source">)`
2. Vous pouvez remplir les valeurs après création du paquet, par exemple :
   `packet[IP].dst = "<ip destination>"`

Une fois que vous avez créé un paquet, vous pouvez voir ses champs avec la commande :
`packet.show()`
et l'envoyer par l'interface réseau de votre choix avec la commande :
`sendp(packet, iface="<interface>")`

Q 40 *Afin de bien comprendre l'utilisation de scapy, donnez lui les commandes suivantes qui envoyent un ping à la machine `B` (que vous pourrez donc voir dans la capture wireshark) :*

1. *`packet = Ether(src="<adr MAC de O>", dst="<adr MAC de B>")/IP()/ICMP()`*
2. *`packet.show()`*
3. *`packet[IP].src = "<adr IP de O>"`*
4. *`packet[IP].dst = "<adr IP de B>"`*
5. *`packet.show()`*
6. *`sendp(packet, iface="enp0s8")`*

Q 41 *Depuis la machine `O`, envoyez avec scapy un paquet ARP à la machine `A`, indiquant malicieusement qu'un paquet pour l'adresse IP de `B` doit être envoyé à l'adresse MAC de `O`. Pour ceci, nous vous indiquons les noms dans scapy des champs de la couche ARP que vous aurez besoin de remplir :*

1. *`op` qui indique quel type de paquet est-ce : `1` pour un paquet "requête" demandant une association MAC/IP, et `2` pour un paquet "réponse" associant une adresse MAC et IP.*
2. *`hwsrc` qui doit contenir l'adresse MAC à associer,*
3. *`hwdst` qui doit contenir l'adresse MAC de destination du paquet,*
4. *`psrc` qui doit contenir l'adresse IP à associer,*
5. *`pdst` qui doit contenir l'adresse IP de destination du paquet.*

Vérifiez l'envoi du paquet à l'aide wireshark.

Q 42 *Une fois le paquet envoyé, inspectez à nouveau la table ARP de la machine `A`. Comment a-t-elle été modifiée ?*

Q 43 *Depuis la machine `A`, tentez de faire un ping vers la machine `B`. Que se passe-t-il ? (regardez wireshark sur la machine `O`). Arrêtez le ping et regardez à nouveau la table ARP de `A`. Retentez le ping, regardez à nouveau la table ARP.*

La machine `O` seulement reçoit les paquets du ping, mais elle pense que ces paquets ne lui sont pas destinés car ils sont pour une autre adresse IP que la sienne. Nous allons donc lui indiquer de considérer que ces paquets sont pour elle en changeant ce qu'elle considère être son adresse IP.

Q 44 *Depuis la machine `O`, renvoyez le paquet ARP de la question précédente, puis changez son adresse avec la commande `ifconfig enp0s8 <nouvelle IP> up`. Testez à nouveau le ping.*

.

Q 45 *Écoutez avec la commande `nc -l -p 8081` sur le port 8081 sur les machines `B` et `O`. Avec la machine `A`, connectez-vous sur le port 8081 et l'adresse IP de la machine `B`. Envoyez des informations très importantes par ce canal. Qui reçoit les informations ?*

5.3 Protection contre l'usurpation d'identité en SSH

Commencez par rétablir la situation initiale : remettez à la machine `O` son adresse IP initiale, et refaîtes des ping vers l'IP de la machine `B` jusqu'à ce que ceux-ci arrivent à la machine `B`. Nous allons maintenant vérifier que le mécanisme vu en cours de l'authentification du serveur fait par SSH marche réellement.

Q 46 *Depuis la machine `A`, connectez-vous en SSH à la machine `B`. Acceptez le certificat, et vérifiez que vous êtes bien sur la bonne machine.*

Q 47 *Refaites la manipulation de la partie précédente pour que `O` usurpe l'identité de la machine `B`. Retentez la connexion SSH vers l'IP de la machine `B`. Que se passe-t-il ?*

Q 48 *Pourquoi est-ce que la donnée de la clé publique de la machine `B` ne permet pas à la machine `O` de se faire passer pour la machine `B` ?*