Travaux Pratiques
Routage, filtrage (pare-feu) et NAT

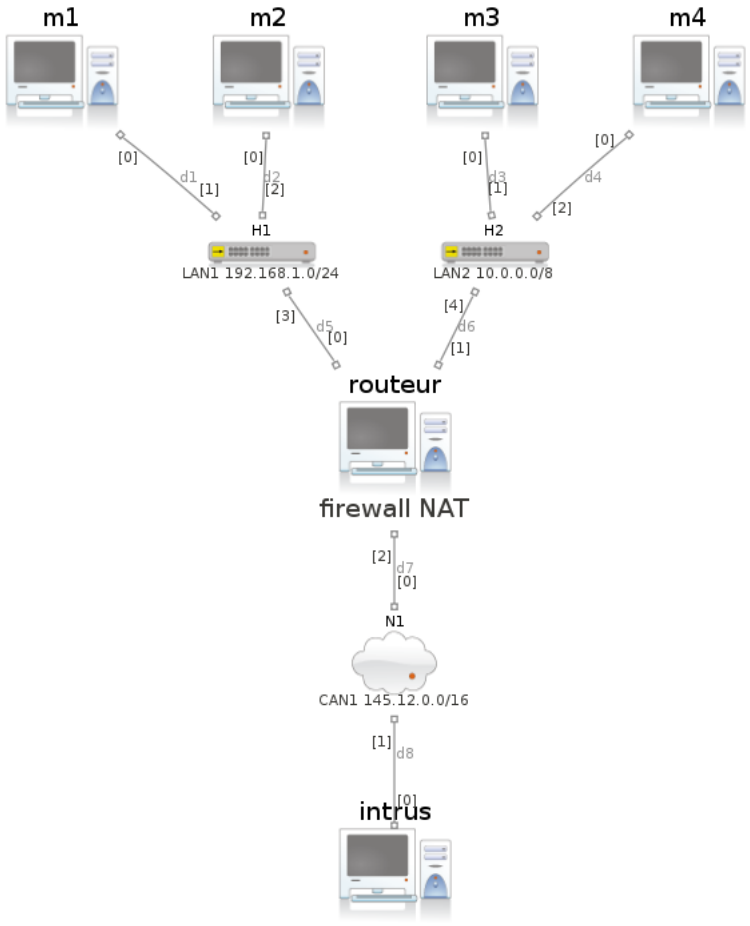
Séance de TP entièrement effectuée avec le logiciel Marionnet. Durée estimée : 2h - 2h30.

Prérequis. Notions de routage, filtrage et NAT (SNAT et DNAT) et leur interdépendance.

Câblage et configuration du réseau local

Deux machines, m_1 et m_2 et un concentrateur H_1 réalisent un réseau local $LAN_1 = \{m_1, m_2\}$ en 192.168.1.0/24. Deux autres machines m_3 et m_4 et un concentrateur H_2 réalisent un réseau local $LAN_2 = \{m_3, m_4\}$ en 10.0.0.0/8. Un troisième réseau CAN_1 (Campus Area Network) sera constitué d'une machine appelée *intrus* et d'une partie indéfinie (de niveau 2) représentée par le composant marionnet "nuage". Une machine faisant office de routeur assurera la laison (de niveau 3) entre LAN_1 (port 0), LAN_2 (port 1) et CAN_1 (port 2).

Distributions GNU/Linux. Utilisez n'importe quelle distribution : il suffira de pouvoir lancer les commandes basiques de configuration et observation du réseau (`ifconfig`, `route`, `tcpdump`, ...)



Attribution des IP. Par simplicité, la machine m_i aura l'adresse 192.168.1.i ou 10.0.0.i selon le réseau d'appartenance. Le routeur *routeur* doit avoir son port 0 branché au LAN_1 et configuré en 192.168.1.254. Concernant le réseau CAN_1 (145.12.0.0/16), la machine *intrus* prendra le 145.12.0.42, et le routeur prendra le 145.12.0.53 sur le port 2 (`eth2`).

Première partie
Routage

Configurer le routage sur la machine *routeur* et définissez-la comme passerelle pour toutes les autres machines du réseaux. Testez avec la commande `ping` que toutes les machines puissent communiquer avec toutes les autres.

Test et remarques : observez que la machine *intrus* reçoit (et répond) aux ping (ECHO REQUEST/REPLY du protocole ICMP) des machines du LAN_1 et du LAN_2, même si elles appartiennent à un réseau à priori **privé** :

```
m1# ping 145.12.0.42
intrus# tcpdump -i eth0
```

Cette situation n'est pas souhaitable pour plusieurs raisons :

1. *intrus* a défini 145.12.0.53 comme passerelle par défaut, ce qui est *abusif* : il devrait ignorer l'existence des réseaux privés ; ces derniers devraient, dans l'idéal, être *cachés* derrière le routeur ;
2. *intrus* peut lui même pinguer les réseaux privés, ce qui veut dire que *routeur* laisse passer toutes les trames (*pas de filtrage*), même celles qui correspondent à des *initiatives* de l'extérieur vers les réseaux privés (et dans ce contexte, *initiative* peut vouloir dire *attaque*) ;
3. lorsque l'initiative est prise par l'intérieur, comme dans le cas d'un ping depuis LAN_1 ou LAN_2 vers *intrus*, la machine *routeur* laisse passer les paquets IP sans les changer (*pas de NAT*) et *intrus* constate donc la réception de messages provenant d'adresse telles que 192.168.1.0/24 ou 10.0.0.8 ; s'il ne le sait pas déjà, il peut donc imaginer pouvoir utiliser *routeur* pour atteindre ces adresses. Autrement dit, s'il l'ignorait auparavant, il n'ignorera plus l'existence de ces réseaux, ce qui nous ramène au problème soulevé au point 1.

Il faut donc configurer le filtrage et la traduction d'adresses pour **protéger** et **cacher** la structure interne du réseau privé aux yeux de l'extérieur.

Deuxième partie
Filtrage et SNAT

Supposons que la machine *intrus* offre un service HTTP ; lancez donc une serveur http sur cette machine. Configurer le filtrage sur la machine *routeur* de façon qu'elle protège l'ensemble des réseaux privés LAN_1 et LAN_2, c'est-à-dire de façon que :

(a) *intrus* ne pourra pas avoir accès à ces réseaux privés

(b) toutes les machines des réseaux privés auront accès au serveur web de *intrus*

(c) *intrus* aura toujours l'impression que les requêtes proviennent de *routeur*

Suggestion : pour obtenir (a) et (b) utiliser le module `state` de `iptables` pour définir des règles selon l'état `NEW`, `ESTABLISHED` ou `RELATED` (cf. `man iptables`) ; pour obtenir (c) utiliser une règle SNAT.

Vérifier le résultat de votre configuration avec `tcpdump` ou `wireshark`.

Troisième partie
DNAT

Supposons à présent que la machine m_1 offre un service HTTP et que m_2 offre un service de connexion à distance SSH ; lancez donc ces services. Ajouter une règle au pare-feu de façon que :

(d) *intrus* pourra être client des services HTTP de m_1 et SSH de m_2 mais en ayant toujours l'impression que ces services soient rendus par *routeur* qui, à ses yeux, sera donc son seul possible interlocuteur.