

# Travaux Pratiques

## Filtrage et NAT : scripts de contrôle

Copyright (C) 2012 Jean-Vincent Loddo  
Licence Creative Commons Paternité - Partage à l'Identique 3.0 non transposé.

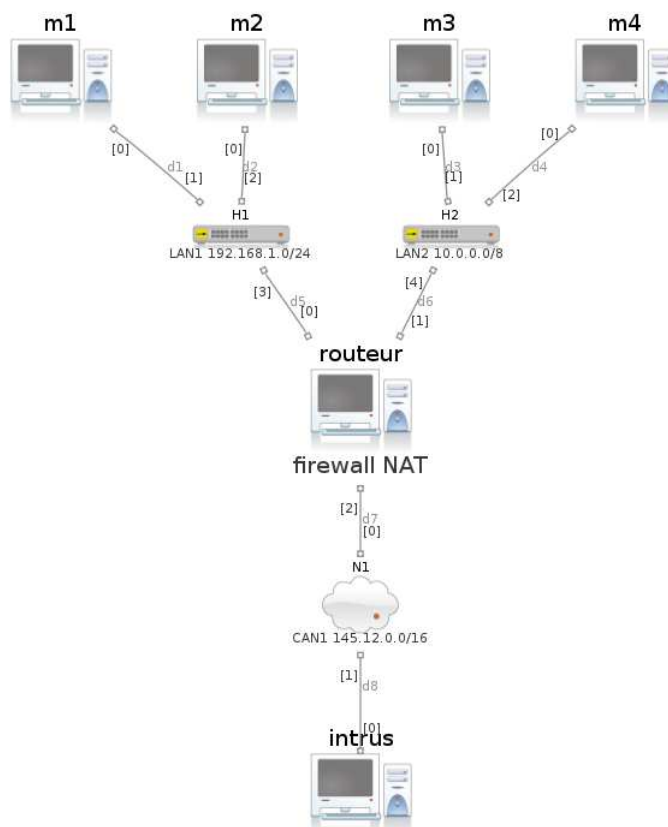
Séance de TP entièrement effectuée avec le logiciel Marionnet. Durée estimée : 2h - 2h30.

**Prérequis.** Notions de routage, filtrage et NAT (SNAT et DNAT) et leur interdépendance. Savoir écrire de simples scripts shell.

### 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 liaison (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_firewall` doit avoir son port 0 branché au  $LAN_1$  et configuré en 192.168.1.254. Concernant le réseau  $CAN_1$ , 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 et lancement des services

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

Services à activer :

- *intrus* offre un service HTTP
- *m<sub>1</sub>* offre un service HTTP
- *m<sub>3</sub>* offre un service de connexion à distance sécurisé SSH

## Deuxième partie

# Filtrage, SNAT et DNAT par script

Le but de cette partie est d'écrire un script sur *routeur\_firewall* de façon à obtenir le comportement qui est habituellement désiré :

1. *routeur\_firewall* doit prêter son identité publique aux machines des réseaux privés, lorsque ces dernières "sortent" sur le réseau publique (SNAT, action `MASQUERADE`)
2. l'extérieur, dont *intrus*, ne peut avoir accès, par son **initiative**, aux réseaux privés ; l'extérieur peut juste **répondre** aux initiatives (filtrage, chaîne `FORWARD`, module `state`)
3. les machines du réseaux privés, à l'occurrence *m<sub>1</sub>* et *m<sub>3</sub>*, doivent pouvoir offrir des services à l'extérieur, sans que l'extérieur en suppose l'existence ; *routeur\_firewall* "fera semblant " d'offrir ces services à l'extérieur (DNAT).

Écrire donc un script `bash` commençant par :

```
#!/bin/bash
EXTIF=eth2    # external interface
```

et contenant les fonctions suivantes :

```
function fw_start { .. }
function fw_stop  { .. }
function fw_open_port { .. }
function fw_close_port { .. }
```

- La fonction `fw_start` s'occupe des points 1. et 2. ci-dessus, c'est-à-dire elle s'occupe d'activer le filtrage de base (pas d'initiatives de l'extérieur) et le masquering. Elle opère à la fois sur la chaîne `FORWARD` et sur la chaîne `INPUT`.
- La fonction `fw_stop` remet toutes les chaînes manipulées dans leurs états initiaux.
- La fonction `fw_open_port` ouvre un port d'un protocole donné ; pour ce faire elle utilise au moins 2 arguments : le protocole (`$1`) et le port (`$2`) à ouvrir ; si on ne lui donne pas d'autres arguments, elle ouvre ce port en opérant sur la chaîne `INPUT` ; si on lui donne un troisième argument, qui est une adresse IP (`$3`), elle ouvre ce port en opérant sur la chaîne `FORWARD` et s'occupe de rediriger (DNAT) vers `$3` ; un quatrième argument (`$4`) optionnel précisera éventuellement le port de redirection. Exemples d'utilisation :

```
fw_open_port tcp 53
fw_open_port tcp 80 192.168.1.1
fw_open_port tcp 22 10.0.0.1
fw_open_port tcp 22 10.0.0.1 50022
```

- La fonction `fw_close_port` fera le travail inverse de `fw_open_port`, c'est-à-dire qu'elle éliminera les règles introduites avec `fw_open_port` ; elle aura donc elle aussi 2 arguments obligatoires et 2 autres optionnels. Exemples d'utilisation :

```
fw_close_port tcp 53
fw_close_port tcp 22 10.0.0.1 50022
```

Complétez le script pour configurer *routeur\_firewall* et vérifiez le résultat de votre configuration avec `tcpdump` ou `wireshark`.