

Forwarding: Inspecter le paquet et l'aiguiller (Couche 3)

Couche 4: Transport:

- End to End protocol
- Destinataires finaux.

Routeur:

- TTL
- IP destination

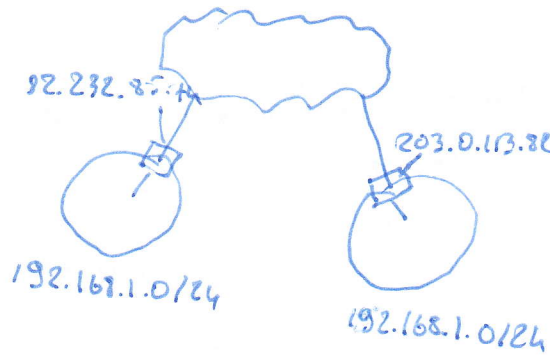
RFC 1918: Ilya 3 prefix d'adresse pour un usage privé only

10.0.0.0/8

172.16/12

(172.16.0.0 ↔ 172.31.255.255)

192.168.0.0/16



Bon: lors d'un paquet sortant, tu remplace l'ip locale par l'ip publique et vice versa pour le packet retour

SNAT: Source Network Address Translation.

La box doit se souvenir des packet: => break End to End principe.

Appariation des state.

Pour distinguer les pc, la box y attribue un port par pc.
↳ la violation de l'encapsulation

DMZ: Route par défauts Redirection en réseau local

Les routeurs regardent ainsi la couche 3/4 et 7.

DNAT: Destination NAT

Configurer NAT:

Linux: iptables

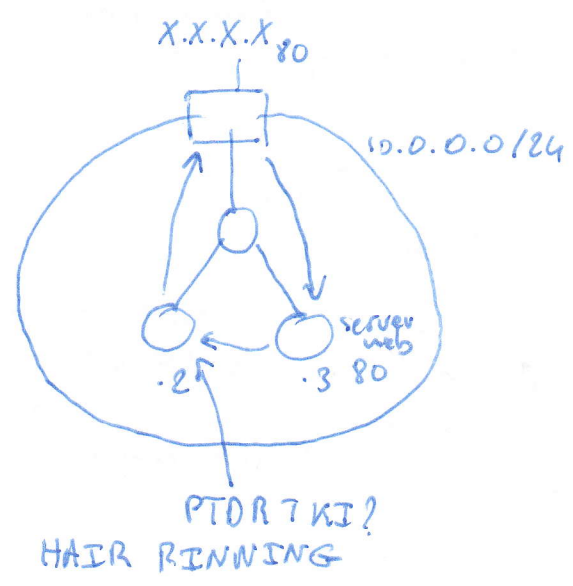
IPTABLES

↳ FW ↳ NAT

Routing decision: Inspection de la table de routage

3 grands chemins:

- Input
- Output
- Forward (Repropagation)



FILTER: Firewall

NAT

Port forwarding: Destination NAT

Tables NAT: 4 checkpoints

- Pre routing ← DNAT
- Post routing ← SNAT (Masquerade)
- Input
- Output

DNAT localhost: 80 → 8080 OUTPUT chain.

7: APPLICATION

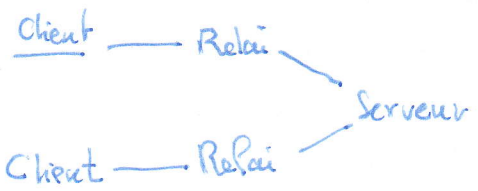
- ↳ DHCP (HTTP (Cours de 06/11))
- ↳ DNS (Cours de 13/11)

DHCP: Dynamic Host Configuration Protocol

- ↳ Auto-configuration Réseau,
- ↳ Fournit:

- Adresse IP
- Mask du sous réseau
- Default gate: IP du routeur pour se connecter à internet
- Adresse serveur DNS.
- NTP: Serveur temps

Pour pouvoir réaliser la connexion de manière centralisée malgré le broadcast. On utilise des relais qui peuvent contacter un serveur. Le relais ne renvoie que le packet du serveur. Il ne contient pas d'information



Le serveur DHCP n'adresse pas les IP aléatoirement. Il peut s'appuyer sur les infos (@MAC, hostname) pour tenter de reconnaître un client et lui donner la même adresse IP que à la dernière connexion.

DHCP

Broadcast ⇒ UDP

- ↳ Appel à plusieurs machines.
- ↳ Port 67/68
- ↳ Message DHCP
- ⇒ Premier message sur le réseau local
- ↳ DISCOVER

Reponse du serveur: OFFER

- ↳ IP pour le client
- ↳ IP Gateway
- ↳ Durée de vie du contrat

Le client accepte: REQUEST

Le serveur renvoie un ACKNOWLEDGE

- Nemotechnique:
- DISCOVER
 - OFFER
 - REQUEST
 - ACKNOWLEDGE
 - DORA.

HTTP : Format Request / Reponse
TCP ≠ DHCP en UDP
Stateless ⇒ On peut traiter les requêtes sans avoir de mémoire.

Version: 1.0, 1.1, 2.0 → DRAMA

URLC URI
↑ ↑ ce que c'est
location

URL:
http://Goto.com/Pop?mdir=42
SCHEME AUTHORITY PATH QUERY
 OR HOST NAME STRING

Hypertext: Document connecté avec d'autres document.
Pour passer de l'un à l'autre, il n'est pas nécessaire de repasser par la barre de recherche.

GET: Accès à un document on read
POST: Envoyer un formulaire (login)
PUT
DELETE

Request HTTP:
HTTP/1.1 200 OK
Content-type:
Content-length:

Header: Host: nom de domaine avant traduction
User-Agent

Aujourd'hui, 95% des applications sont sur HTTP

Les requêtes HTTP des apps permet de diriger des applications via des portes logiques (ex boutons)
Elles ne sont pas faites pour être lu par des organiques.

⇒ API: ex:
• SOAP: Echange d'XML
• XMLRPC
• REST:

GET: Ne doit pas faire de modif, au effet de bord. En effet elle est mis en cache.
PUT
DELETE

⇒ IDEMPOTENT (f(f(x)) = f(x))
On peut l'appliquer autant de fois, le résultat sera le même que d'application une seule fois.

POST créer une ressource, on ne sait pas où elle va exister ⇒ Il n'est pas IDEMPOTENT

Status code HTTP

- 1xx: Info
- 2xx: Succes
- 3xx: Redirection
- 4xx: Erreur client
- 5xx: Erreur serveur