

Architecture réseau et filtrage des flux

Filtrage de flux

Traduction d'adresse et redirection

Cloisonnement des sous-réseaux

Ajustement des services

Commutateurs administrables

Fabrice HARROUET

École Nationale d'Ingénieurs de Brest

harrouet@enib.fr

<http://www.enib.fr/~harrouet/>

Architecture réseau et filtrage des flux

▷ Propos

- ◇ Il ne suffit pas de faire fonctionner correctement le réseau (ça c'est facile !)
- ◇ Répond essentiellement à une problématique de sécurité
 - Antagoniste avec le “confort” d'utilisation (compromis à trouver)
- ◇ Limiter l'effet et la propagation des actions malveillantes
 - Risques techniques et juridiques
- ◇ Repose sur les fonctionnalités de filtrage, traduction et redirection
 - Cloisonnement des sous-réseaux
 - N'autoriser que le trafic strictement nécessaire
 - Décoreller les architectures physique et logique
- ◇ Influe sur la configuration des services
 - Restrictions sur l'usage des services existants
 - Introduction de services spécifiques

Architecture réseau et filtrage des flux

- ▷ **Où sont les méchants “hackers-pédophyles-nazys” ?**
 - ◇ À l’extérieur, sur *Internet* ?
 - On ne maîtrise rien, on y accède à nos risques
 - Limiter les sorties (*blacklist* ...)
 - On peut limiter les entrées à quelques services exposés (ou rien)
 - ◇ À l’intérieur, quelle confiance accorder aux utilisateurs, aux visiteurs ?
 - Ils peuvent être malveillants à leur insu !
 - Systèmes trop permissifs infestés de programmes malveillants
 - Destruction ou évocation d’informations sensibles
 - Hébergement ou relai d’informations “*tendancieuses*”
 - Difficile à contrer sans leur interdire complètement le trafic !
 - **Vous êtes responsable de ce qui sort de chez vous !**
 - (cf <http://sebsauvage.net/safehex.html>)

Filtrage, traduction et redirection

- ▷ **Faire respecter une politique de sécurité dans un réseau**
 - ◇ On utilise un dispositif *pare-feu* (*firewall*)
 - ◇ Filtrage : bloquer ou autoriser le trafic réseau
 - À destination de la machine elle-même
 - Émanant de la machine elle-même
 - Traversant la machine (routeur)
 - ◇ Traduction, redirection : modifier les sources/destinations du trafic
 - ◇ Repose sur l’analyse des entêtes *IP*, *ICMP*, *UDP*, *TCP* ...
 - ◇ Fonctionnalité du noyau du système, dans la pile *TCP/IP*
 - Configurable (voire interopérable) depuis le mode utilisateur

Filtrage, traduction et redirection

- ▷ **Pare-feu sans états (*stateless*)**
 - ◊ Dispositif de filtrage peu évolué (obsolète)
 - ◊ Un ensemble de règles à confronter aux packets *IP*
 - Porte sur les adresses, le protocole, les ports ...
 - ◊ L'analyse de chaque paquet est indépendante des paquets précédents
 - Ne filtre pas grand chose dans la pratique !
 - ◊ Exemple un poste client veut consulter des serveurs *HTTP*
 - Laisser sortir les paquets *TCP* vers le port 80 (\forall destination)
 - Le port source du client est dans la plage 49152–65535
 - Il faudrait laisser entrer tous les paquets dans cette plage !
(serveur \rightarrow client : acquitement ...)

Filtrage, traduction et redirection

- ▷ **Pare-feu à états (*statefull*)**
 - ◊ L'analyse de chaque paquet provoque une mémorisation
 - Ces informations aident à décider du sort des prochains paquets
 - La mémorisation a une durée maximale
 - ◊ On parle de *suivi de communication* (*conntrack*)
 - ◊ Exemple un poste client veut consulter des serveurs *HTTP*
 - Laisser sortir les paquets *TCP* vers le port 80 (\forall destination)
 - Le port source du client est dans la plage 49152–65535
 - Mémorisation de *TCP/@_src/port_src/@_dest/port_dest*
 - Autorisation implicite de *TCP/@_dest/port_dest/@_src/port_src*
 - Le serveur peut répondre au client
 - ◊ De même pour *UDP* et *ICMP*
(nb : une connexion *TCP* est facilement identifiable)

Filtrage, traduction et redirection

▷ Séquence de traitement des paquets

- ◇ Exemple de *NetFilter* (*Linux*) ici
- ◇ Plusieurs *chaînes* :
 - PREROUTING : ce qui arrive sur une interface
 - INPUT : ce qui est à destination de la machine elle-même
 - OUTPUT : ce qui émane de la machine elle-même
 - FORWARD : ce qui traverse la machine (routeur)
 - POSTROUTING : ce qui sort par une interface
- ◇ Plusieurs *tables* :
 - **filter** : filtrage des paquets
 - **nat** : traduction d'adresse et redirection
 - **mangle** : altération des paquets
 - **raw** : exceptions au suivi de communication (rare !)

Filtrage, traduction et redirection

▷ Séquence de traitement des paquets

- ◇ Chaque règle de *firewall* concerne une chaîne dans une table
 - Contient un critère de correspondance avec les paquets (interface d'entrée/sortie, protocole, ports/adresses source/dest. ...)
 - Contient une action à effectuer en cas de correspondance (accepter, détruire, traduire, rediriger ...)
- ◇ Pour chaque paquet à traiter dans une chaîne d'une table :
 - Les règles sont confrontées une à une au paquet
 - La première qui correspond emporte la décision
 - Décision par défaut en cas de non correspondance

Filtrage, traduction et redirection

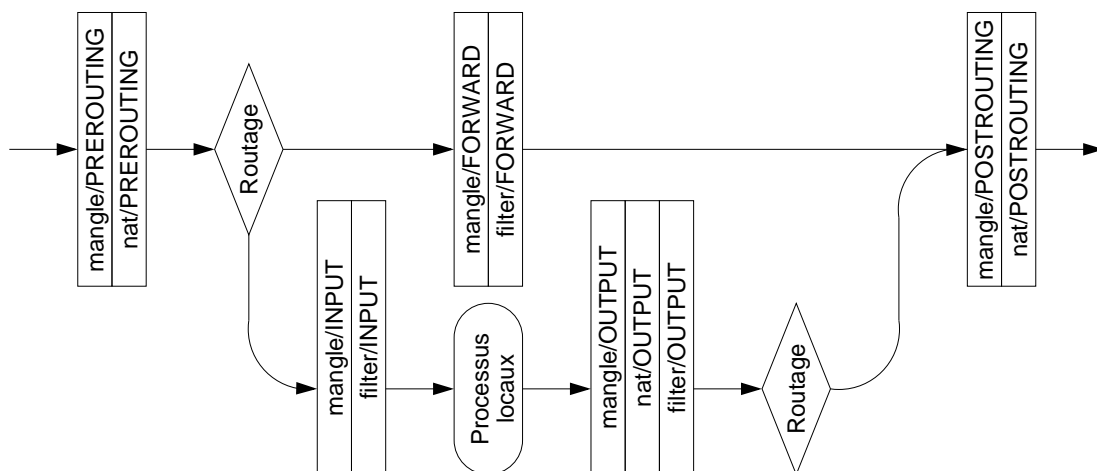
▷ Séquence de traitement des paquets

- ◇ Chaque table ne peut influencer que sur quelques chaînes
 - `filter` : INPUT, OUTPUT et FORWARD
 - `nat` : PREROUTING, OUTPUT et POSTROUTING
 - `mangle` : PREROUTING, INPUT, OUTPUT, FORWARD et POSTROUTING
 - `raw` : PREROUTING et OUTPUT
- ◇ Possibilité de créer de nouvelles chaînes
 - Peuvent être invoquées depuis d'autres chaînes
 - Permet de structurer les règles en cas configuration élaborée

Filtrage, traduction et redirection

▷ Séquence de traitement des paquets

- ◇ Pour chaque chaîne, dans l'ordre : `mangle` puis `nat` puis `filter`



Filtrage, traduction et redirection

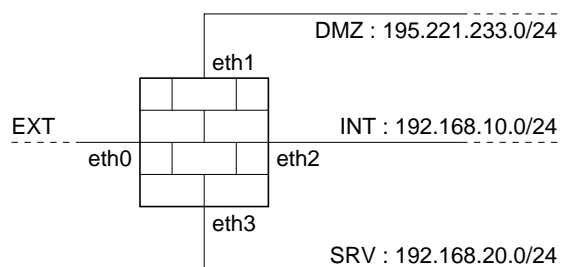
▷ Configuration de *NetFilter*

- ◇ Utilisation de la commande `iptables` (privilèges requis)
- ◇ Action par défaut : `iptables -t table -P chaîne action`
 - Action **ACCEPT** : autorisation par défaut
 - Action **DROP** : blocage par défaut
- ◇ Retirer les règles : `iptables -t table -F`
- ◇ Retirer les chaînes ajoutées : `iptables -t table -X`
- ◇ Ajout : `iptables -t table -A chaîne critère -j action`
- ◇ Table **filter** par défaut si option `-t` omise
- ◇ Commandes généralement regroupées dans un fichier *script*
 - Tout réinitialiser et bloquer par défaut
 - Si désactivation du *firewall* autoriser par défaut
 - Ajout des règles spécifiques

Filtrage, traduction et redirection

▷ Configuration de *NetFilter*

```
# cat /etc/rc.d/rc.firewall
#!/bin/sh
IFCFG="/sbin/ifconfig"
IPTBL="/usr/sbin/iptables"
EXT_IF=eth0
DMZ_IF=eth1
INT_IF=eth2
SRV_IF=eth3
case "$1" in
'stop')
    disable_firewall="true"
    ;;
*)
    disable_firewall="false"
    IPRE="[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}"
    EXT_IP='${IFCFG} ${EXT_IF} | grep "inet addr:" | \
        sed -re s/.*inet addr:(${IPRE}).*/\1/'
    if [ -z "${EXT_IP}" ] ; then
        echo "Cannot determine IP address for ${EXT_IF} !"
        disable_firewall="true"
    fi
    # ... determine DMZ_IP from DMZ_IF (same way) ...
    # ... determine INT_IP from INT_IF (same way) ...
    # ... determine SRV_IP from SRV_IF (same way) ...
    ;;
esac
```



Filtrage, traduction et redirection

```

##### Clear and set default policy to DROP #####
${IPTBL} -t filter -P INPUT DROP
${IPTBL} -t filter -P FORWARD DROP
${IPTBL} -t filter -P OUTPUT DROP
${IPTBL} -t filter -F
${IPTBL} -t filter -X
${IPTBL} -t nat -P PREROUTING ACCEPT
${IPTBL} -t nat -P POSTROUTING ACCEPT
${IPTBL} -t nat -P OUTPUT ACCEPT
${IPTBL} -t nat -F
${IPTBL} -t nat -X
${IPTBL} -t mangle -P PREROUTING ACCEPT
${IPTBL} -t mangle -P INPUT ACCEPT
${IPTBL} -t mangle -P FORWARD ACCEPT
${IPTBL} -t mangle -P OUTPUT ACCEPT
${IPTBL} -t mangle -P POSTROUTING ACCEPT
${IPTBL} -t mangle -F
${IPTBL} -t mangle -X

##### Allow everything and exit if disabled #####
if [ "${disable_firewall}" = "true" ] ; then
  echo "!!!! FIREWALL IS DISABLED !!!!!"
  ${IPTBL} -t filter -P INPUT ACCEPT
  ${IPTBL} -t filter -P FORWARD ACCEPT
  ${IPTBL} -t filter -P OUTPUT ACCEPT
  return 0 2>/dev/null || exit 0
fi

# ... specific rules ...

```

Filtrage, traduction et redirection

▷ Critères de correspondance des règles

- ◇ ∃ de nombreux critères, ceux qui ne sont pas spécifiés sont ignorés
 - i [!] *name* interface recevant le paquet
 - o [!] *name* interface émettant le paquet
 - s [!] *address* [/mask] adresse *IP* source
 - d [!] *address* [/mask] adresse *IP* destination
 - p [!] *protocol* icmp, udp, tcp ...
 - sport [!] *port* [:port] port source
 - dport [!] *port* [:port] port destination
 - Il en existe beaucoup d'autres ...
- ◇ Il faut toujours spécifier le plus finement possible !
 - Permet de n'autoriser que le strict nécessaire

Filtrage, traduction et redirection

▷ Correspondance des règles et suivi de communication

- ◇ `-m state --state NEW`
 - Le paquet ne correspond à aucune communication mémorisée
- ◇ `-p tcp --tcp-flags ALL SYN -m state --state NEW`
 - Une nouvelle connexion *TCP* doit nécessairement avoir le drapeau **SYN**
- ◇ `-m state --state ESTABLISHED,RELATED`
 - Le paquet correspond à une communication déjà mémorisée
- ◇ Démarche systématique de filtrage
 - Autoriser la suite des communications déjà établies
 - Autoriser “*finement*” les nouvelles communications
- ◇ Les interfaces de réception et d’émission ne sont pas mémorisées (`cat /proc/net/nf_conntrack`)
 - Il faut donc les spécifier pour le suivi des communications

Filtrage de flux

▷ Expression des règles de filtrage

- ◇ Concerne les chaînes **INPUT**, **OUTPUT** ou **FORWARD** de la table **filter**
- ◇ L’action (`-j`) est généralement
 - **ACCEPT** : on laisse le paquet continuer
 - **DROP** : on détruit le paquet
 - **REJECT** : **DROP** + envoi d’un message *ICMP* à la source
 - Il en existe bien d’autres ...
- ◇ Choisir les nouvelles communications à autoriser dans une direction
- ◇ Autoriser le suivi des communications établies
 - Dans la même direction pour les paquets suivants
 - Dans la direction opposée pour les paquets en retour
- ◇ L’ordre des règles est important
 - La première correspondance emporte la décision

Filtrage de flux

▷ Expression des règles de filtrage

- ◇ Utiliser des variables pour éviter les erreurs et faciliter la maintenance

```
# cat /etc/rc.d/rc.firewall
#!/bin/sh

... IPTBL EXT_IF/EXT_IP DMZ_IF/DMZ_IP INT_IF/INT_IP SRV_IF/SRV_IP ...

##### Usefull variables #####
NEW_ICMP_PACKET="-p icmp -m state --state NEW"
NEW_UDP_PACKET="-p udp -m state --state NEW"
NEW_TCP_PACKET="-p tcp --tcp-flags ALL SYN -m state --state NEW"

ALLOW_EXISTING="-m state --state ESTABLISHED,RELATED -j ACCEPT"
ALLOW_NEW_ICMP="{NEW_ICMP_PACKET} -j ACCEPT"
ALLOW_NEW_UDP="{NEW_UDP_PACKET} -j ACCEPT"
ALLOW_NEW_TCP="{NEW_TCP_PACKET} -j ACCEPT"
ALLOW_NEW_HTTP="{NEW_TCP_PACKET} --dport 80 -j ACCEPT"
ALLOW_NEW_DNS_T="{NEW_TCP_PACKET} --dport 53 -j ACCEPT"
ALLOW_NEW_DNS_U="{NEW_UDP_PACKET} --dport 53 -j ACCEPT"

WWW_HOST="195.221.233.2"
DMZ_NET="195.221.233.0/24"
INT_NET="192.168.10.0/24"
SRV_NET="192.168.20.0/24"

# ... specific rules ...
```

Filtrage de flux

▷ Filtrage du trafic sur un poste

- ◇ Ne concerne que le poste lui-même, chaînes **INPUT** et **OUTPUT** (\neq routeur)
- ◇ Au préalable, l'action par défaut est **DROP**
 - Il ne reste plus qu'à expliciter ce qu'on autorise
- ◇ Généralement on autorise tout sur l'interface *loopback*
 - L'interface est le seul critère de correspondance
- ◇ Autoriser l'accès à un service (rôle de serveur)
 - Aussi spécifique que possible
- ◇ Autoriser l'utilisation de services (rôle de client)
 - Aussi spécifique que possible

Filtrage de flux

▷ Filtrage du trafic sur un poste

```
# cat /etc/rc.d/rc.firewall

# ...

##### Clear and set default policy to DROP #####
${IPTBL} -t filter -P INPUT DROP
${IPTBL} -t filter -P FORWARD DROP
${IPTBL} -t filter -P OUTPUT DROP

# ...

##### Allow loopback #####
${IPTBL} -A INPUT -i lo -j ACCEPT
${IPTBL} -A OUTPUT -o lo -j ACCEPT

##### Allow input #####
CMD="${IPTBL} -A INPUT -i ${EXT_IF} -d ${EXT_IP}"
${CMD} ${ALLOW_EXISTING}
${CMD} ${ALLOW_NEW_HTTP} # http server

##### Allow output #####
CMD="${IPTBL} -A OUTPUT -o ${EXT_IF} -s ${EXT_IP}"
${CMD} ${ALLOW_EXISTING}
${CMD} ${ALLOW_NEW_HTTP} # http client
${CMD} ${ALLOW_NEW_DNS_U} # dns client
${CMD} ${ALLOW_NEW_DNS_T} # dns client (zone transfer)
```

enib, F.H ... 19/68

Filtrage de flux

▷ Filtrage du trafic à travers un routeur

- ◇ Ne concerne pas le poste lui-même, chaîne **FORWARD**
- ◇ Au préalable, l'action par défaut est **DROP**
 - Il ne reste plus qu'à expliciter ce qu'on autorise
- ◇ Autoriser la "*sortie*" vers des services extérieurs
- ◇ Autoriser l' "*entrée*" vers des services internes
- ◇ Aussi spécifique que possible
 - Selon le rôle attribué à chaque sous-réseau
 - Plus ou moins sûr, hostile, vulnérable ...
- ◇ nb : le routeur peut lui même être client ou serveur (voir l'utilisation des chaînes **INPUT** et **OUTPUT**)

enib, F.H ... 20/68

Filtrage de flux

▷ Filtrage du trafic à travers un routeur

```
# cat /etc/rc.d/rc.firewall
# ...
##### INT-->EXT #####
CMD="${IPTBL} -A FORWARD -i ${INT_IF} -s ${INT_NET} -o ${EXT_IF}"
${CMD} ${ALLOW_EXISTING}
${CMD} ${ALLOW_NEW_HTTP} # reach external http servers
##### EXT-->INT #####
CMD="${IPTBL} -A FORWARD -i ${EXT_IF} -o ${INT_IF} -d ${INT_NET}"
${CMD} ${ALLOW_EXISTING}

##### INT-->SRV #####
CMD="${IPTBL} -A FORWARD -i ${INT_IF} -s ${INT_NET} -o ${SRV_IF} -d ${SRV_NET}"
${CMD} ${ALLOW_EXISTING}
${CMD} ${ALLOW_NEW_DNS_U} # internal use of our own servers
${CMD} ${ALLOW_NEW_DNS_T}
##### SRV-->INT #####
CMD="${IPTBL} -A FORWARD -i ${SRV_IF} -s ${SRV_NET} -o ${INT_IF} -d ${INT_NET}"
${CMD} ${ALLOW_EXISTING}

##### EXT-->WWW_HOST #####
CMD="${IPTBL} -A FORWARD -i ${EXT_IF} -o ${DMZ_IF} -d ${WWW_HOST}"
${CMD} ${ALLOW_EXISTING}
${CMD} ${ALLOW_NEW_HTTP} # expose our public http server
##### WWW_HOST-->EXT #####
CMD="${IPTBL} -A FORWARD -i ${DMZ_IF} -s ${WWW_HOST} -o ${EXT_IF}"
${CMD} ${ALLOW_EXISTING}
```

enib, F.H ... 21/68

Traduction d'adresse

▷ Les plages d'adresses privées

- ◇ Permettent d'avoir plus de nœuds que d'adresses publiques disponibles
 - Pas assez d'adresses *IPv4* disponibles pour tout le monde !
- ◇ Par convention elles ne sont pas routées sur *Internet* (juste en interne)
 - De nombreux sous-réseaux utilisent en interne les mêmes adresses
 - 10.0.0.0/8 : 1 seul sous-réseau de classe A
 - 172.16.0.0/12 : 16 sous-réseaux de classe B
 - 192.168.0.0/16 : 256 sous-réseaux de classe C
 - Peuvent être découpés/agrégés librement (*CIDR*)
- ◇ Communication possible dans le réseau local (routage classique)
- ◇ Comment communiquer avec les adresses publiques d'*Internet* ?
 - Même si les paquets "*sortent*", le routage en retour est impossible (de multiples réseaux locaux utilisent les mêmes adresses privées)

enib, F.H ... 22/68

Traduction d'adresse

▷ Principe du NAT (*Network Address Translation*)

- ◇ Le routeur relié à *Internet* modifie la source du paquet sortant
 - Remplacée par une adresse publique (celle du routeur généralement)
- ◇ Les paquets en retour subissent la modification inverse
 - Adresse destination publique → privée
- ◇ Tout le trafic du réseau local semble émaner du routeur lui-même
 - Multiplexage des ports sources et suivi des communications
- ◇ Opération effectuée dans la chaîne **POSTROUTING** de la table **nat**
 - Le filtrage “*voit*” les adresses privées (cf page 10)
 - L'opération inverse est implicite (*statefull*)

Traduction d'adresse

▷ Expression des règles *SNAT* (*Source-NAT*)

- ◇ Action **-j SNAT** avec l'option **--to-source**
- ◇ L'ordre des règles est important
 - La première correspondance emporte la décision
 - Possibilité de “*court-circuit*” par une action **ACCEPT**

```
# cat /etc/rc.d/rc.firewall
# ...

##### First solution #####
# no translation for public addresses (DMZ)
${IPTBL} -t nat -A POSTROUTING -s ${DMZ_NET} -o ${EXT_IF} -j ACCEPT
# translate everything else
${IPTBL} -t nat -A POSTROUTING -o ${EXT_IF} -j SNAT --to-source ${EXT_IP}

##### Second solution #####
# ${IPTBL} -t nat -A POSTROUTING -s ! ${DMZ_NET} -o ${EXT_IF} -j SNAT --to-source ${EXT_IP}

##### Third solution #####
# ${IPTBL} -t nat -A POSTROUTING -s ${SRV_NET} -o ${EXT_IF} -j SNAT --to-source ${EXT_IP}
# ${IPTBL} -t nat -A POSTROUTING -s ${INT_NET} -o ${EXT_IF} -j SNAT --to-source ${EXT_IP}
```

Traduction d'adresse

▷ La variante *IP-Masquerading*

- ◇ Traduction utilisant implicitement l'adresse publique du routeur
 - `-j MASQUERADE` \equiv `-j SNAT --to-source ${EXT_IP}`
- ◇ À n'utiliser qu'en cas d'adresse publique variable (*FAI*)
 - L'adresse de l'interface est évaluée à chaque traduction
 - Oubli des traductions en cours en cas de changement d'adresses (elles ne peuvent se poursuivre)

Traduction d'adresse

▷ Filtrage des fausses adresses privées

- ◇ Aucune adresse privée ne doit provenir d'*Internet*
 - Ce serait certainement une tentative de malveillance
- ◇ De même pour les adresses de *loopback* recues sur une interface \neq `lo`
- ◇ Filtrage au plus tôt : `-t mangle -A PREROUTING` (cf page 10)
 - Ne sert normalement pas au filtrage, mais fonctionne tout de même
 - Sinon marquage des paquets puis filtrage dans `filter` (plus compliqué)

```
# cat /etc/rc.d/rc.firewall
# ...

##### Deny fake private/loopback addresses #####
${IPTBL} -t mangle -A PREROUTING -i ! lo -d 127.0.0.0/8 -j DROP
${IPTBL} -t mangle -A PREROUTING -i ! lo -s 127.0.0.0/8 -j DROP
${IPTBL} -t mangle -A PREROUTING -i ${EXT_IF} -d 192.168.0.0/16 -j DROP
${IPTBL} -t mangle -A PREROUTING -i ${EXT_IF} -s 192.168.0.0/16 -j DROP
${IPTBL} -t mangle -A PREROUTING -i ${EXT_IF} -d 172.16.0.0/12 -j DROP
${IPTBL} -t mangle -A PREROUTING -i ${EXT_IF} -s 172.16.0.0/12 -j DROP
${IPTBL} -t mangle -A PREROUTING -i ${EXT_IF} -d 10.0.0.0/8 -j DROP
${IPTBL} -t mangle -A PREROUTING -i ${EXT_IF} -s 10.0.0.0/8 -j DROP
```

Redirection

▷ Principe et intérêt

- ◇ Seules les adresses publiques peuvent être atteintes depuis *Internet*
- ◇ Affectation directe des adresses publiques aux machines ?
 - Notre parc ne le permet peut-être pas
 - Solution peu souple (migration, mise à jour *DNS* ...)
- ◇ Affectation d'adresses privées aux machines publiques
 - Nécessite une redirection *adresses publiques* → *adresses privées*
 - Souplesse dans la réorganisation du parc
- ◇ Similaire à la traduction mais modifie la destination du paquet entrant
- ◇ Les paquets en retour subissent la modification inverse
- ◇ Opération effectuée dans la chaîne **PREROUTING** de la table **nat**
 - Le filtrage "*voit*" les adresses privées (cf page 10)
 - L'opération inverse est implicite (*statefull*)

Redirection

▷ Expression des règles *DNAT* (*Destination-NAT*)

- ◇ Action **-j DNAT** avec l'option **--to-destination**
- ◇ Le port destination est un discriminant intéressant
- ◇ L'ordre des règles est important ici aussi

```
# cat /etc/rc.d/rc.firewall
# ...
##### EXT_IP is the only public address (ISP) #####
CMD="${IPTBL} -t nat -A PREROUTING -i ${EXT_IF} -d ${EXT_IP}"
${CMD} -p tcp --dport 80 -j DNAT --to-destination ${PRIV_WWW_HOST}
${CMD} -p udp --dport 53 -j DNAT --to-destination ${PRIV_NS_HOST}
${CMD} -p tcp --dport 53 -j DNAT --to-destination ${PRIV_NS_HOST}
# ...

# cat /etc/rc.d/rc.firewall
# ...
##### Public address range #####
# redirect access to public NS_HOST to private PRIV_NS_HOST
${IPTBL} -t nat -A PREROUTING -i ${EXT_IF} -d ${NS_HOST} \
-j DNAT --to-destination ${PRIV_NS_HOST}
# redirect access to public WWW_HOST to private PRIV_WWW_HOST
${IPTBL} -t nat -A PREROUTING -i ${EXT_IF} -d ${WWW_HOST} \
-j DNAT --to-destination ${PRIV_WWW_HOST}
# ...
```

Redirection et traduction

▷ Principe du *BiNAT*

- ◇ Associer complètement une adresse privée et une adresse publique
- ◇ Permet à un nœud d'être atteint depuis l'extérieur (redirection)
 - Le nœud est désigné par son adresse publique
- ◇ Lui permet d'établir un dialogue avec l'extérieur (traduction)
 - Le nœud est visible avec la même adresse publique
- ◇ nb : **NETMAP** \equiv **SNAT** ou **DNAT** pour des plages d'adresses (souplesse ?)

```
# cat /etc/rc.d/rc.firewall
# ...
# redirect access to public WWW_HOST to private PRIV_WWW_HOST
${IPTBL} -t nat -A PREROUTING -i ${EXT_IF} -d ${WWW_HOST} -j DNAT --to-destination ${PRIV_WWW_HOST}
# translate private PRIV_WWW_HOST as public WWW_HOST
${IPTBL} -t nat -A POSTROUTING -o ${EXT_IF} -s ${PRIV_WWW_HOST} -j SNAT --to-source ${WWW_HOST}

# cat /etc/rc.d/rc.firewall
# ...
# redirect access to whole public PUB_NET to private DMZ_NET
${IPTBL} -t nat -A PREROUTING -i ${EXT_IF} -d ${PUB_NET} -j NETMAP --to ${DMZ_NET}
# translate whole private DMZ_NET as public PUB_NET
${IPTBL} -t nat -A POSTROUTING -o ${EXT_IF} -s ${DMZ_NET} -j NETMAP --to ${PUB_NET}
```

enib, F.H ... 29/68

Filtrage, traduction et redirection

▷ Contrôler l'état des tables

- ◇ Commande `iptables -t table -L -v -n`
 - Peu lisible mais permet de constater l'effet des commandes
- ◇ Astuce : insérer la commande **echo** au début de la variable `IPTBL`
 - L'exécution du *script* affiche les commandes sans les exécuter

```
# iptables -L -v -n
Chain INPUT (policy DROP 117 packets, 38376 bytes)
 pkts bytes target prot opt in out source destination
  4 340 ACCEPT all -- lo * 0.0.0.0/0 0.0.0.0/0
 135 11424 ACCEPT all -- eth0 * 0.0.0.0/0 192.168.7.5 state RELATED,ESTABLISHED
  1 60 ACCEPT tcp -- eth0 * 192.168.4.8 192.168.7.5 tcp dpt:22 flags:0x3F/0x02 state NEW
  0 0 ACCEPT icmp -- eth0 * 192.168.4.8 192.168.7.5 state NEW

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target prot opt in out source destination

Chain OUTPUT (policy DROP 96 packets, 10013 bytes)
 pkts bytes target prot opt in out source destination
  4 340 ACCEPT all -- * lo 0.0.0.0/0 0.0.0.0/0
 85 10880 ACCEPT all -- * eth0 192.168.7.5 0.0.0.0/0 state RELATED,ESTABLISHED
  0 0 ACCEPT tcp -- * eth0 192.168.7.5 192.168.4.8 tcp dpt:22 flags:0x3F/0x02 state NEW
```

enib, F.H ... 30/68

Filtrage, traduction et redirection

▷ Bilan

- ◇ Le filtrage permet de n'autoriser que le trafic légitime
 - Cloisonnement des sous-réseaux
- ◇ Les traductions et redirections donnent de la souplesse
 - Adresses perçues décorellées des adresses effectives
- ◇ Seules les opérations de base ont été vues ici
 - Accéder à **Internet** et être accessible depuis **Internet**
- ◇ Bien d'autres choses sont envisageables
 - Utilisation de critères plus spécifiques
 - Modification, marquage de paquets et routage alternatif
 - Création de nouvelles chaînes
 - Interaction avec un processus
 - Architecture modulaire, beaucoup d'actions et d'options ...

Filtrage, traduction et redirection

▷ Bilan

- ◇ La solution **NetFilter** de **Linux** est présentée ici à titre d'illustration
 - Très employée, beaucoup de documentation
 - Très complète, permet des opérations très "*subtiles*"
 - Peut devenir très verbeuse et illisible, il faut s'efforcer de clarifier (variables, démarche systématique ...)
- ◇ Les autres sont très similaires dans le principe
 - Règles plus ou moins verbeuses
 - Interactions plus ou moins fortes avec un processus (**Window\$**)
 - Comportement plus ou moins implicite (ex : dans les **ACL** de **Cisco**, la poursuite des communications établies est implicite)
- ◇ Il existe des interfaces graphiques pour faciliter la mise au point

Filtrage, traduction et redirection

- ▷ **PacketFilter d'OpenBSD : la référence en matière de *firewall***
 - ◇ Système principalement orienté vers la sécurité
 - Beaucoup d'innovation dans ce domaine
 - ◇ *scrub* : normalisation des paquets (rejet des incohérences)
 - ◇ *antispoof* : rejet des adresses incohérentes
 - ◇ *synproxy* : ne laisser parvenir aux serveurs que les connexions établies
 - ◇ *modulate* : dissimulation du modèle des systèmes
 - ◇ *osfp* : détection passive des systèmes pour influencer sur le filtrage
 - ◇ *authpf* : adapter le filtrage à l'utilisateur (pas au poste)
 - ◇ *altq* : gestion des priorités
 - ◇ *carp/pfsync* : redondance et synchronisation des états
 - ◇ ...

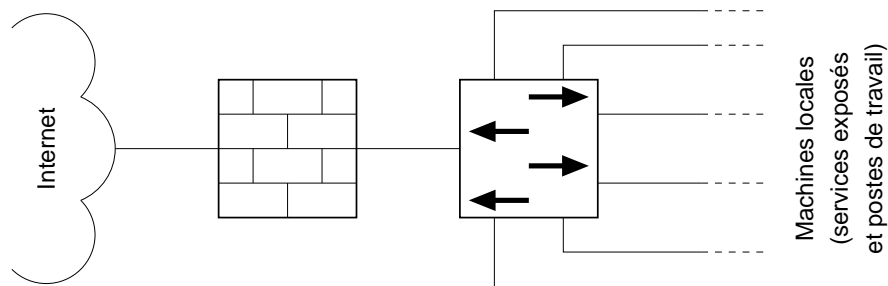
Architecture réseau

- ▷ **La notion de DMZ (*DeMilitarized-Zone*)**
 - ◇ Partie du réseau contenant les services accessibles depuis l'extérieur
 - ◇ Doit correspondre à des adresses publiques (redirections éventuelles)
 - ◇ Si un service est attaqué et corrompu, il reste isolé du réseau local
 - ◇ Peut être constituée de plusieurs sous-réseaux
 - Pour éviter la propagation aux autres services
 - ◇ La sécurisation de cette zone est très importante
 - Directement exposée aux attaques extérieures

Architecture réseau

▷ Architecture *bastion*

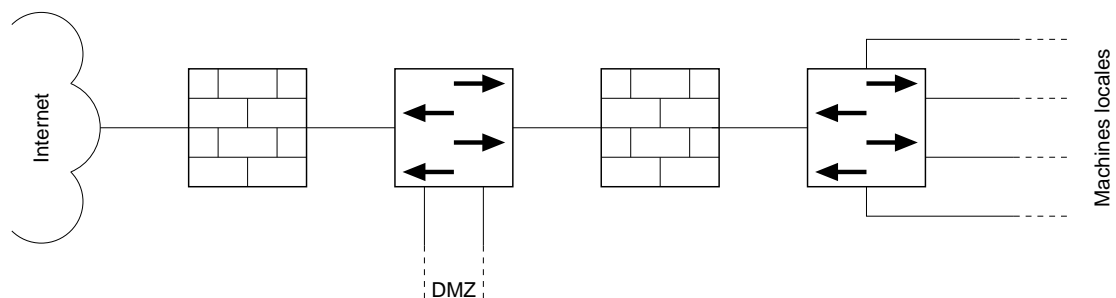
- ◇ (+) Solution économique pour un particulier
 - Boîtier *modem-routeur-firewall-switch*
- ◇ (-) Solution minimale, cloisonnement faible
 - Toutes les machines sont dans le même sous-réseau
 - Accès à l'une → accès potentiel à toutes



Architecture réseau

▷ Architecture *dos-à-dos*

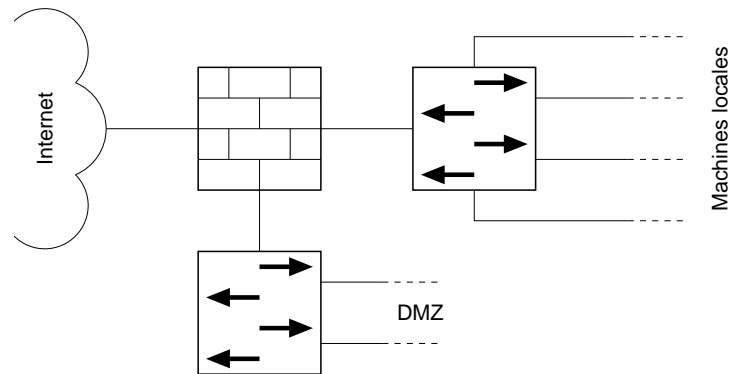
- ◇ (+?) Deux *firewalls* à “casser” pour accéder au réseau local (?)
 - Choisir deux modèles/systèmes différents (plus difficile)
 - Ne “casser” que le deuxième (le premier donne accès à la *DMZ* ...)
- ◇ (-) Nécessite du matériel : deux *firewalls*
- ◇ (-) Une *DMZ* corrompue peut interférer avec le trafic *local* ↔ *Internet* !



Architecture réseau

▷ Architecture à-trois-branches (ou N branches)

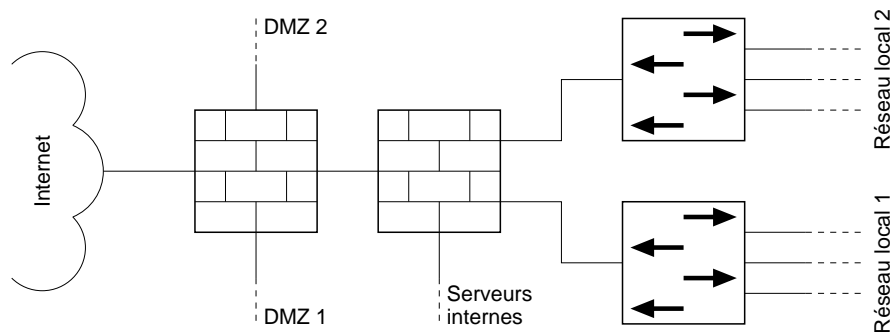
- ◇ (+) Nécessite peu de matériel : un seul *firewall*
- ◇ (+) Le cloisement est strict, une *DMZ* corrompue n'a accès qu'à elle-même
- ◇ (-) Tout repose sur l'unique *firewall*
 - S'il est vulnérable il n'y a plus du tout de cloisonnement



Architecture réseau

▷ Cascade d'architectures à-N-branches

- ◇ (-) Nécessite du matériel : plusieurs *firewalls*
- ◇ (+) Le cloisement est toujours aussi strict
 - Ajustement "*fin*" de l'accès à chaque branche
- ◇ (+) Ne repose pas que sur un seul *firewall* (différents modèles/systèmes)



Architecture réseau

- ▷ **Recommandations pour une *DMZ*, vis-à-vis d'*Internet***
 - ◇ Dissimuler ou falsifier les versions des serveurs et des systèmes
 - ◇ Utiliser des adresses privées et des redirections
 - Dissimuler les détails de l'architecture réelle (et plus souple)
 - ◇ Ne laisser entrer en *DMZ* que le strict nécessaire
 - Désactiver tous les services inutiles !
 - Atteindre les ports spécifiques des machines spécifiques
 - Un filtrage redondant sur chaque machine est encore mieux
 - Bloquer l'accès depuis une machine voisine corrompue
 - ◇ Rien ne doit sortir de la *DMZ* ! (juste le suivi)
 - Les serveurs ne sont là que pour répondre à des requêtes
 - Un serveur corrompu ne doit pas pouvoir atteindre *Internet* (télécharger des données, envoyer du *spam* ...)

Architecture réseau

- ▷ **Recommandations pour une *DMZ*, vis-à-vis du réseau local**
 - ◇ La *DMZ* peut être vue comme *Internet*, elle en fait partie
 - ◇ Il peut être nécessaire d'y accéder selon d'autres services
 - Dépôt de fichiers, mise à jour d'une base de données ...
 - Ne doit contenir que ce qu'on a l'intention de publier
 - Ne pas accorder une confiance excessive à ces machines (elles peuvent être corrompues)
 - ◇ Rien ne doit sortir de la *DMZ* ! (juste le suivi)
 - Aucune propagation ne doit être possible vers le réseau local

Architecture réseau

▷ Cloisonnement du réseau local

- ◇ Différents statuts d'utilisateurs (visiteurs, élèves, profs, administration ...)
 - Filtrage des accès des uns vers les autres
 - Différents droits d'accès aux services internes ou externes
- ◇ Les serveurs à usage interne doivent être sûrs
 - Cibles privilégiées pour la propagation des malveillances
 - Un poste ne doit pas pouvoir se faire passer pour un serveur
 - isoler les serveurs dans un(des) sous-réseau(x)
 - Mettre en œuvre un filtrage rigoureux ($\simeq DMZ$)
 - S'il y a des données sensibles, l'accès physique doit être restreint !

Ajustement des services

▷ Problématique : où placer tel service dans notre réseau ?

- ◇ La réponse dépend d'autres questions
 - Qui doit y avoir accès ?
 - Quelles relations ces services doivent-ils entretenir ?
- ◇ La réponse va au delà du simple placement
 - Ajustement des règles de filtrage
 - Configuration des services eux-mêmes
- ◇ Il ne s'agit pas que d'un problème technique
 - Quantité de matériel disponible ? Investissement envisageable ?
 - Habitudes, bonne volonté des partenaires ?
- ◇ Il n'y a pas de démarche systématique et sûre à 100%
 - Chaque cas mérite une étude détaillée
 - On se contente de limiter les entorses aux principes précédents :-)

Ajustement des services

- ▷ **Exemple : le serveur *web* public utilise une base de données**
 - ◇ Les employés de l'entreprise travaillent sur la base de données
 - Tout n'est pas censé être publié (résultats publiés, savoir faire dissimulé)
 - ◇ Placer la base dans le réseau local ?
 - (-) La *DMZ* devrait alors accéder au réseau local !
 - ◇ Placer la base en *DMZ* ?
 - (+) La *DMZ* n'accède pas au réseau local
 - (-) Les informations privées sont potentiellement accessibles
 - (-) Travail sur des données potentiellement corrompues
 - ◇ Réplication partielle de la base (réseau local → *DMZ*) ?
 - (+) Seules les données à publier sont en *DMZ*
 - (+) La *DMZ* n'accède pas au réseau local, le travail en local est sûr
 - (-) Coût de la duplication des serveurs et procédure de réplication

Ajustement des services

- ▷ **Le cas du service *DNS***
 - ◇ Certainement le service le plus utilisé
 - Une très grande proportion d'applications et de services en ont besoin
 - Une défaillance dans les résolutions *DNS* est très pénalisante (temps de réponse très long ou échec de l'application ou du service)
 - ◇ La tentation est grande de laisser le trafic *DNS* circuler librement !
 - "Comme ça au moins, ça fonctionne !"
 - ◇ Cible privilégiée pour les malveillances
 - Permet de détourner le trafic vers un "piège"
 - ◇ Comme pour notre exemple précédent, on duplique les serveurs
 - Un serveur *DNS* en *DMZ*
 - Un autre parmi les serveurs à usage interne

Ajustement des services

▷ Le serveur *DNS* en *DMZ*

- ◇ Sert à résoudre les noms de domaines de notre *DMZ* depuis *Internet*
 - Autoritaire sur notre domaine et nos adresses publiques
- ◇ Les machines en *DMZ* peuvent éventuellement l'interroger
 - Pour résoudre les voisines (multi-vues ?) (ex: *HTTP* → *BdD*)
- ◇ Ne doit pas permettre de récursion vers les serveurs racines
 - Aucune raison de sortir sur *Internet* → pas besoin de résoudre
- ◇ Ne doit pas permettre de résoudre dans notre domaine privé
 - Aucune raison d'entrer dans le réseau local → pas besoin de résoudre
- ◇ Les machines locales n'en ont pas besoin et ne doivent pas l'interroger
 - *DNS* privé et adresses privées, même pour atteindre la *DMZ*

Ajustement des services

▷ Le serveur *DNS* à usage interne

- ◇ Sert à résoudre dans notre domaine privé
 - Autoritaire sur notre domaine et nos adresses privées (y compris la *DMZ*)
- ◇ Sert également aux récursions vers les serveurs racines
 - Permet de résoudre vers *Internet*
- ◇ Interrogé par les serveurs internes et les postes de travail
 - Des restrictions (multi-vues) peuvent être mises en place
 - ex : Pas accès direct à *Internet* → domaine local uniquement
 - ex : Les élèves n'accèdent pas aux profs → domaine local partiel

Ajustement des services

- ▷ **Utilisation d'un serveur mandataire HTTP (*proxy*)**
 - ◇ Mise en *cache* des pages *web* rapatriées
 - Meilleures performances pour les pages souvent consultées
 - ◇ Constitution de journaux d'activité (*logs*)
 - Les autorités peuvent les demander (archivage !)
 - ◇ Interdiction de visite de certains sites (*blacklist*)
 - ex : base pour **squidguard** sur <http://cri.univ-tlse1.fr/blacklists/>
(mise à jour par un "robot" et des contributeurs)
 - ◇ Filtrage éventuel du contenu selon des mots-clefs (raciste, cochon ...)

Ajustement des services

- ▷ **Utilisation d'un serveur mandataire HTTP (*proxy*)**
 - ◇ Fonctionnement du *proxy*
 - Le *client* fait sa requête au *proxy*
 - Le *proxy* analyse la requête et la relaie sur **Internet**
 - Le *proxy* analyse la réponse et la relaie vers le *client*
 - *Cache, log*, filtrage éventuels dans les étapes de relais
 - ◇ Seul le *proxy* résoud les noms de domaines sur **Internet**
 - Les clients n'ont pas besoin de récursion vers les *DNS* racines
 - Uniquement si tout passe par le *proxy* (difficile)
 - Permet également de s'affranchir du fichier **hosts** corrompu
(`C:\system32\drivers\etc\hosts` ou `/etc/hosts`)
 - ◇ Placer le *proxy* parmi les serveurs : pas d'usurpation pour sortie directe
 - ◇ Jamais en *DMZ* : utilisé depuis l'extérieur pour rebondir (*open-proxy*) !!!

Ajustement des services

▷ *Proxy transparent*

- ◇ Normalement les clients se connectent explicitement au *proxy*
 - Sur un port spécifique, **3128** par exemple
 - Les logiciels doivent être configurés en conséquence
- ◇ Pour plus de confort le *firewall* peut effectuer une redirection
 - Intercepter les tentatives de sortie sur le port **80 TCP**
 - Les rediriger vers le *proxy*
 - Celui-ci doit être configuré pour comprendre la requête
 - Les clients n'ont plus à connaître le *proxy*

```
#{IPTBL} -t nat -A PREROUTING -i #{INT_IF} -s #{INT_NET} \
-p tcp --dport 80 -j DNAT --to-destination #{PROXY_HOST}:3128
```

Ajustement des services

▷ *Proxy transparent*

- ◇ La solution précédente est très simple mais limitée
 - L'entête de la requête *HTTP* doit contenir un champ **Host**
 - Sans ça, le *proxy* ne sait pas où se connecter
- ◇ Solution plus élaborée : routage alternatif
 - Marquer les paquets *HTTP*
 - Router les paquets marqués selon une table alternative
 - Le *proxy* doit capturer des paquets qui ne lui sont pas adressés !
 - La destination de ces paquets lui indiquent où se connecter

```
#{IPTBL} -t mangle -A PREROUTING -i #{INT_IF} -s #{INT_NET} \
-p tcp --dport 80 -j MARK --set-mark 3
ip rule add fwmark 3 table 2
ip route add default via #{PROXY_HOST} dev #{SRV_IF} table 2
```

Ajustement des services

▷ Agent de relais *DHCP*

- ◇ Lorsque le serveur *DHCP* n'est pas dans le sous-réseau des clients
 - Fonctionne sur le routeur pour relayer entre les clients et le serveur
- ◇ Différent types de dialogues (initial, renouvellement, abandon)
 - Client (0.0.0.0:68) → relais (255.255.255.255:67)
 - Client (*adresse_client*:68) → relais (*adresse_relais*:67)
 - Relais (*adresse_relais*:67) → client (*adresse_client*:68)
 - Relais (*adresse_relais*:67) ↔ serveur (*adresse_serveur*:67)
 - Client (*adresse_client*:68) ↔ serveur (*adresse_serveur*:67)
- ◇ Le relais capture des paquets avant filtrage ! (le serveur aussi)
- ◇ Messages relativement décorellés les uns des autres
 - Le suivi de session n'apporte pas grand chose
 - On autorise uniquement en fonction des source/destination/ports

Ajustement des services

▷ Agent de relais *DHCP*

```
# ... Allow input from client to relay ...
${IPTBL} -A INPUT -i ${INT_IF} -s 0.0.0.0 -d 255.255.255.255 \
-p udp --sport 68 --dport 67 -j ACCEPT
${IPTBL} -A INPUT -i ${INT_IF} -s ${INT_NET} \
-p udp --sport 68 --dport 67 -j ACCEPT

# ... Allow output from relay to clients ...
${IPTBL} -A OUTPUT -o ${INT_IF} -d ${INT_NET} \
-p udp --sport 67 --dport 68 -j ACCEPT

# ... Allow input from server to relay ...
${IPTBL} -A INPUT -i ${SRV_IF} -s ${SRV_NET} \
-p udp --sport 67 --dport 67 -j ACCEPT

# ... Allow output from relay to server ...
${IPTBL} -A OUTPUT -o ${SRV_IF} -d ${SRV_NET} \
-p udp --sport 67 --dport 67 -j ACCEPT

# ... Allow forward from clients to server ...
${IPTBL} -A FORWARD -i ${INT_IF} -s ${INT_NET} -o ${SRV_IF} -d ${DHCP_HOST} \
-p udp --sport 68 --dport 67 -j ACCEPT

# ... Allow forward from server to clients ...
${IPTBL} -A FORWARD -i ${SRV_IF} -s ${DHCP_HOST} -o ${INT_IF} -d ${INT_NET} \
-p udp --sport 67 --dport 68 -j ACCEPT
```

Ajustement des services

▷ Fonctionnement du protocole *FTP* (*File Transfer Protocol*)

- ◇ Le client se connecte sur le port *TCP* 21 du serveur
- ◇ Cette connexion permet d'échanger des commandes
- ◇ Pour échanger un fichier, *FTP* utilise une deuxième connexion
- ◇ Fonctionnement en mode *actif*
 - Le client écoute sur un nouveau port arbitraire
 - Il envoie ce numéro de port au serveur par la première connexion
 - Le serveur se connecte au port du client pour l'échange de fichier
- ◇ Fonctionnement en mode *passif*
 - Le serveur écoute sur un nouveau port arbitraire
 - Il envoie ce numéro de port au client par la première connexion
 - Le client se connecte au port du serveur pour l'échange de fichier

Ajustement des services

▷ Filtrage du protocole *FTP*

- ◇ On autorise la connexion au port 21 du serveur
- ◇ Comment autoriser la deuxième connexion ?
 - En mode *actif* : laisser entrer vers tous les ports du client !?!
 - En mode *passif* : laisser sortir vers tous les ports à l'extérieur !?!
- ◇ Nécessité d'un module "*applicatif*"
 - Analyse le dialogue sur la première connexion
 - Repère l'échange du numéro de port et autorise le trafic vers ce port
 - Correspond à l'état **RELATED** dans le suivi des communications

```

/sbin/modprobe ip_conntrack_ftp
ALLOW_NEW_FTP="${NEW_TCP_PACKET} --dport 21 -j ACCEPT"
# ... proxy --> FTP servers ...
CMD="${IPTBL} -A FORWARD -i ${SRV_IF} -s ${PROXY_HOST} -o ${EXT_IF}"
${CMD} ${ALLOW_EXISTING}
${CMD} ${ALLOW_NEW_FTP}
# ... FTP servers --> proxy ...
CMD="${IPTBL} -A FORWARD -i ${EXT_IF} -o ${SRV_IF} -d ${PROXY_HOST}"
${CMD} ${ALLOW_EXISTING}

```

Commutateurs administrables

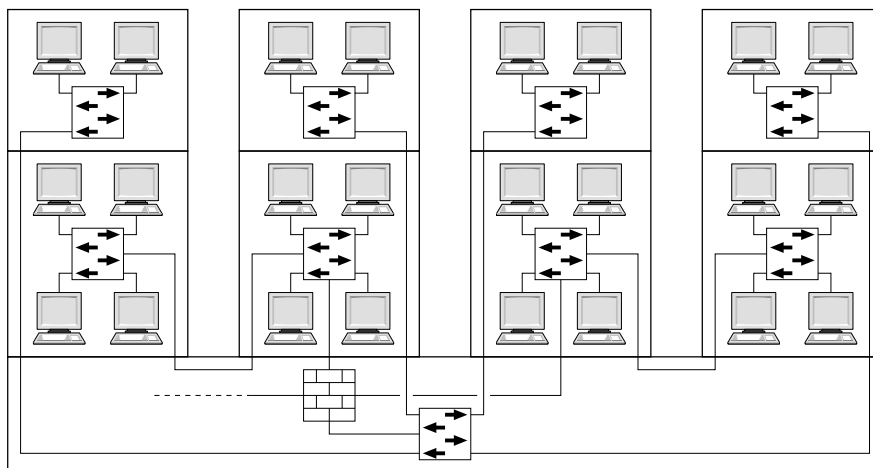
▷ Rappels sur les commutateurs

- ◇ Équipement de niveau 2 (couche liaison/liens)
 - Pas d'adresse *IP* à ce niveau
 - Plusieurs commutateurs reliés → **un seul** sous réseau
 - Si besoin de plusieurs sous-réseaux → intercaler des routeurs
- ◇ Limites en terme d'architecture
 - Architecture logique ≡ architecture physique !
 - Reconfigurer ≡ tirer de nouveaux câbles !

Commutateurs administrables

▷ Exemple d'architecture : structure relativement figée

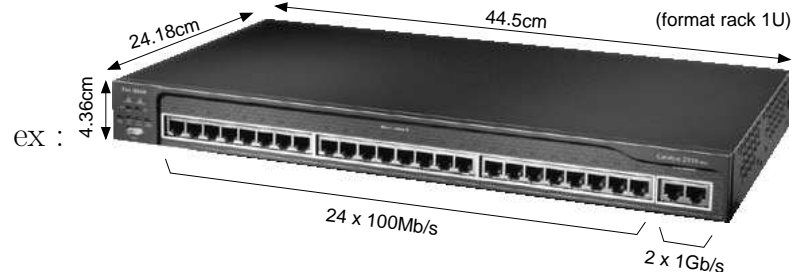
- ◇ Comment subdiviser plus finement ?
- ◇ Nécessite beaucoup d'équipement



Commutateurs administrables

▷ Ce sont toujours des commutateurs (niveau 2)

- ◇ Fonctionnement par défaut sans configuration
- ◇ Disposent généralement de nombreux ports (24, 48 ...)



▷ Fonctionnalités configurables (niveau 2+ ou 3)

- ◇ *VLANs* : réseaux virtuels
- ◇ *ACLs* : filtrage (*Access Control List*)

Commutateurs administrables

▷ Configuration individuelle des ports

- ◇ Affectation d'un numéro de *VLAN*
- ◇ Lien vers un autre commutateur administrable (*Tronçon/Trunk*)

▷ Un *VLAN* \equiv un commutateur classique

- ◇ Fonctionnement au niveau 2
- ◇ Pas de communication implicite entre les différents *VLANs*

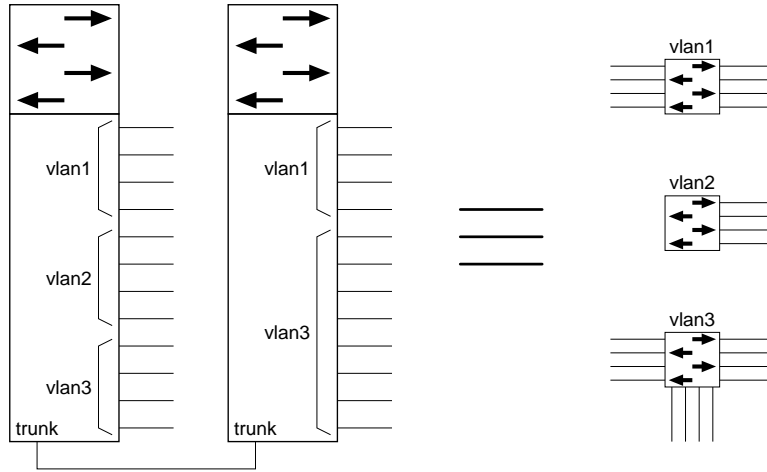
▷ Association de commutateurs administrables

- ◇ Ports du même *VLAN* \equiv un **unique** commutateur virtuel
- ◇ Mise en commun des tables *MAC*/port d'un même *VLAN*

Commutateurs administrables

▷ Association de commutateurs administrables

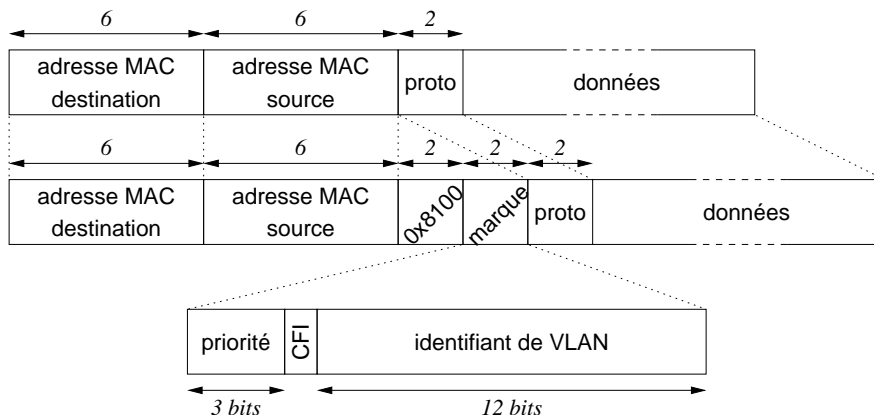
◇ N VLANs \equiv N commutateurs virtuels



Commutateurs administrables

▷ Liaisons *trunk*

- ◇ Encapsulent le trafic de plusieurs brins
- ◇ Préférer les ports les plus rapides pour cet usage
- ◇ Paquets marqués par l'identifiant de *VLAN* (802.1Q)

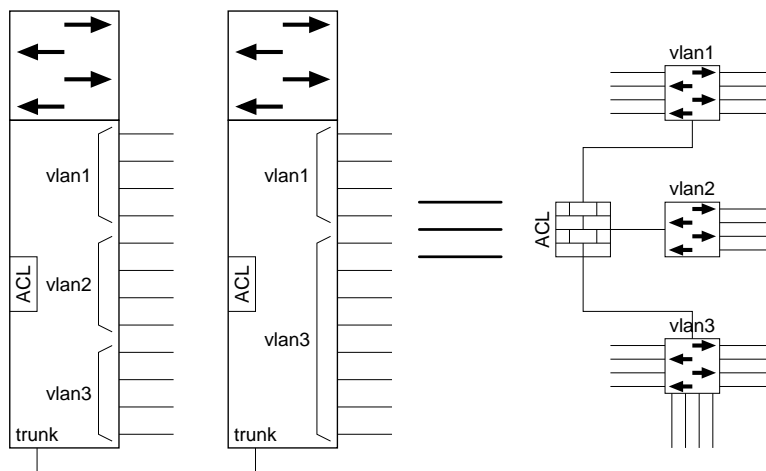


Commutateurs administrables

- ▷ **Un VLAN \equiv un sous-réseau (domaine de diffusion)**
 - ◊ Routage nécessaire pour communiquer entre les VLANs
 - ◊ Possibilité de relier un routeur à plusieurs VLANs (lourd !)
- ▷ **Un VLAN \equiv une interface réseau virtuelle**
 - ◊ Affectation d'une adresse *IP* compatible avec le sous-réseau
 - L'interface est une passerelle du sous-réseau
 - ◊ Règles de communication explicites (*ACL*)
 - Pas de *forward* implicite
 - Règles semblables à un *firewall*
 - Peuvent être définies sur un commutateur *cœur de réseau* puis répliquées automatiquement sur les autres

Commutateurs administrables

- ▷ **Association de commutateurs administrables**
 - ◊ N VLANs + *ACLs* \equiv un routeur filtrant à N interfaces



Commutateurs administrables

▷ Interfaces d'administration

- ◇ Système d'exploitation avec langage de commandes (ex : *Cisco IOS*)
- ◇ Éventuellement une interface *web*

▷ Accès aux interfaces d'administration

- ◇ Généralement protégées par mot de passe
- ◇ Port *console*, liaison série (solution de secours)
 - Uniquement en ligne de commandes
- ◇ Attribution d'une adresse *IP* d'administration
 - *telnet*, *ssh*, *HTTP* ...

Commutateurs administrables

▷ Comment placer et relier les équipements ?

- ◇ Éviter les choix forts sur les installations terminales
 - Autoriser les reconfigurations logiques
 - Offrir de nombreuses possibilités de branchement : **le brassage**
- ◇ Utilisation de locaux techniques
 - Regrouper les équipements spécialisés
(*racks* de commutateurs, de routeurs ...)
 - Un **tableau de brassage** relié à chaque installation terminale
(longs cables, boitiers de sol, goulottes ...)

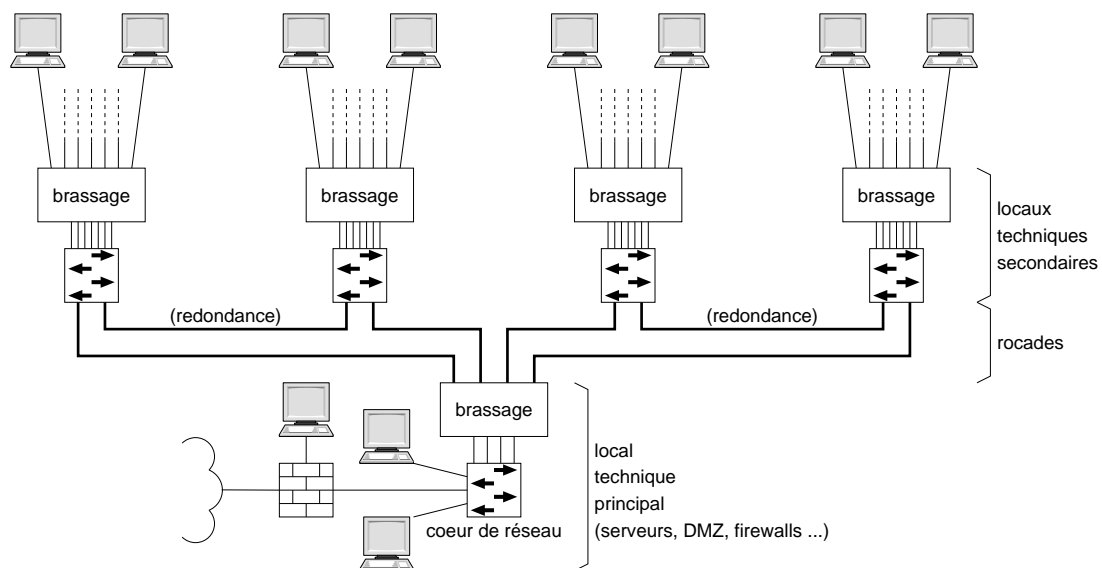
Commutateurs administrables

▷ Comment relier les locaux techniques ?

- ◇ Quelques cables seulement, à très haut débit : **les rocares**
- ◇ Utilisées notamment pour les liaisons *trunk*
- ◇ Possibilité de redondance des rocares
 - Minimiser le risque de coupure
 - Les redondances forment des boucles
 - défaillance des commutateurs classiques
- ◇ Les commutateurs administrables stoppent ces boucles
 - Utilisation du protocole *spanning-tree*
 - Désactiver les liaisons redondantes les plus longues
 - En cas de défaillance d'une liaison
 - réactivation d'une autre après un délais

Commutateurs administrables

▷ Exemple d'architecture : structure relativement flexible



Commutateurs administrables

▷ Intérêt principal

- ◇ Grande souplesse dans l'architecture du réseau
→ reconfiguration logique sans intervention physique
- ◇ Limitation du nombre d'équipement spécialisé à maintenir

▷ Limitations

- ◇ Pas des équipements de sécurité (malgré les *ACLs*)
 - Ne s'applique qu'aux réseaux locaux
 - Il faut utiliser des *firewalls* en amont
- ◇ Prix ! (de qq 100€ à qq 1000€)

▷ De nombreuses fonctionnalités non vues ici ...

- ◇ Filtrage des adresses *MAC* port par port
- ◇ Optimisation, qualité de service ...
- ◇ Très dépendant des marques et des modèles

Architecture réseau et filtrage des flux

▷ Toujours bloquer ce qui n'est pas nécessaire

"Pourtant je ne vois vraiment pas le danger avec ce trafic ..."

- ◇ D'autres le voient peut-être ! (*"Il y'a des gens plus malins que moi !"*)

▷ Rendre difficile la prise d'information

- ◇ Dissimuler/falsifier les versions des services/systèmes
- ◇ Bloquer ping/traceroute
- ◇ Les *DNS* ne doivent donner que le strict nécessaire

▷ Redondance et hétérogénéité

- ◇ La sécurité doit reposer sur plusieurs équipements
- ◇ Les systèmes identiques ont les mêmes vulnérabilités

▷ Tenir son parc à jours vis-à-vis des correctifs de sécurité

- ◇ Sans tarder car ils peuvent avoir un effet pervers !
- ◇ Leur publication divulgue l'existence de la vulnérabilité !