R401
Infrastructures de sécurité

Sami Evangelista
IUT de Villetaneuse
Département Réseaux et Télécommunications
2023–2024

http://www.lipn.univ-paris13.fr/~evangelista/cours/R401

Plan

Contexte

- On voit le réseau comme un ensemble de ressources connectées.
- une ressource :
  - un service
  - un fichier
  - un poste de travail
  - une imprimante
  - un équipement réseau (switch, routeur, ...)

  ...
- On s'intéresse dans ce module aux techniques permettant de sécuriser ces ressources.

Évaluer la sécurité - Les critères DIC

- Qu'est-ce qui définit une ressource "sécurisée" ?
- On retient généralement les 3 critères DIC + éventuellement le critère T.

Disponibilité

La ressource doit être disponible en temps voulu avec un temps de réponse acceptable.

Intégrité

Exactitude de la ressource. Seules les personnes habilitées peuvent modifier son contenu. Toute modification illégitime doit pouvoir être détectée.

Confidentialité

Seules les personnes habilitées peuvent accéder à la ressource.

Traçabilité

Pouvoir retrouver l'historique des accès à la ressource.

Atteintes à la sécurité

Comment la sécurité d'un réseau (au sens des critères DIC) peut-elle être compromise ?

Problèmes physiques/de gestion

- accident, incendie, vol, catastrophe naturelle, panne matérielle, `rm -rf` malheureux, ...
- solution : redondance des données et équipements, sauvegardes automatiques, journalisation, ...

  (hors du champ de ce cours)

Attaques réseau

- attaque = action d'un tiers malveillant exploitant une **faille** de sécurité
- solution : mise en œuvre d'une politique de sécurité via des techniques de protection (pare-feu, chiffrement, authentification, ...), la formation des utilisateurs, ...

Sources des failles de sécurité

Diverses faiblesses peuvent exposer le réseau à des failles :

- faiblesses de conception (= systèmes/protocoles non conçus pour assurer la sécurité ou l'assurant de manière incorrecte)

  ex : DHCP, ARP
- faiblesses d'implémentation (= bogue)

  ex : ping of death
- faiblesses de configuration (= mauvaise configuration des services, équipements de sécurité, ...)

  ex : mauvaise topologie de réseau

  ex : laisser entrer (par erreur) sur le réseau des flux non autorisés
- faiblesses d'utilisation

  ex : mots de passe trop simples

⇒ La sécurité doit être intégrée à tous les niveaux.

Veille technologique

- importance de la veille technologique en sécurité : se tenir informé des vulnérabilités connues
- Plusieurs bases de données publiques recensent ces vulnérabilités :
  - **CVE** — Common Vulnerabilities and Exposures
    - `https://www.cve.org/`
    - maintenue par la MITRE corporation, organisation états-unienne
  - la base du **CERT-FR** (Computer Emergency Response Team)
    - `https://www.cert.ssi.gouv.fr/`
    - centre rattaché à l'ANSSI (Agence Nationale de la Sécurité des Systèmes d'Information)
- Après découverte et publication des vulnérabilités, des correctifs sont ajoutés aux logiciels.

  ⇒ nécessité de mise à jour régulière
- Mais ces bases peuvent aussi être exploitées par les attaquants ...

Qui sont les attaquants ?

Quelques profils :

- cracker (pour le défi technique, pas d'objectif malfaisant)
- ethical hacker (trouve des failles et aide à la résolution)
- cyber criminel (p.ex., vol/rançon de données)
- états (cyber guerre)
- hacktivistes (p.ex., lanceurs d'alerte)
- script kiddies (utilise des scripts d'attaque clé en main sans les comprendre)

Déroulement d'une attaque réseau

Comment s'y prend un attaquant pour compromettre une ressource ?

1. recherche d'informations sur la cible

   ports ouverts, services rendus, OS utilisés, numéros de version des logiciels, logins, ...

   comment ?
   - capture du trafic réseau
   - utilisation d'outils de scan (p.ex., nmap)
   - demande de mot de passe par mail, par chat, ...

   ...
2. identification des failles potentielles

   protocole non sécurisé, version d'un service contenant des bogues, ...
3. **exploit** : tirer parti de ces failles pour
   - gagner un accès privilégié (p.ex., root) sur la machine (**élévation** ou **escalade** de privilèges)
   - paralyser la machine
   - installer un maliciel
   - installer une porte dérobée

   ...

Sans laisser de trace !

Plan

2. Les attaques

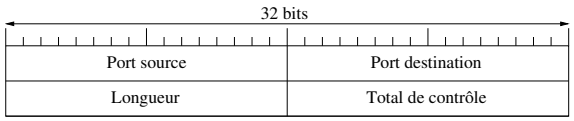
Plan

Caractéristiques de TCP et d'UDP

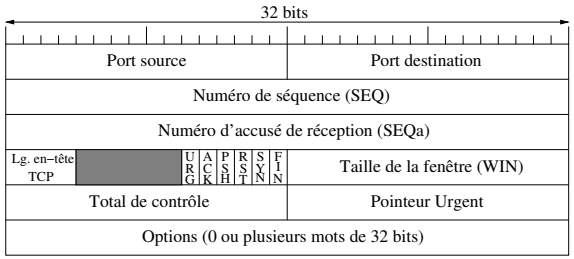
- protocoles de transport (au-dessus d'IP)
- notion de port
  - identifie un processus ($\approx$ adresse du processus sur la machine)
  - port source et destination présents dans les en-têtes TCP et UDP
  - entier sur 16 bits
- UDP
  - mode non connecté
  - pas de correction des erreurs réseau (p.ex., en cas de paquet détruit)
- TCP
  - mode connecté (échange de données balisé par une connexion et une déconnexion)
  - correction des erreurs réseau via l'utilisation d'acquittements et de temporisations

Les en-tête UDP et TCP

- UDP (8 octets)

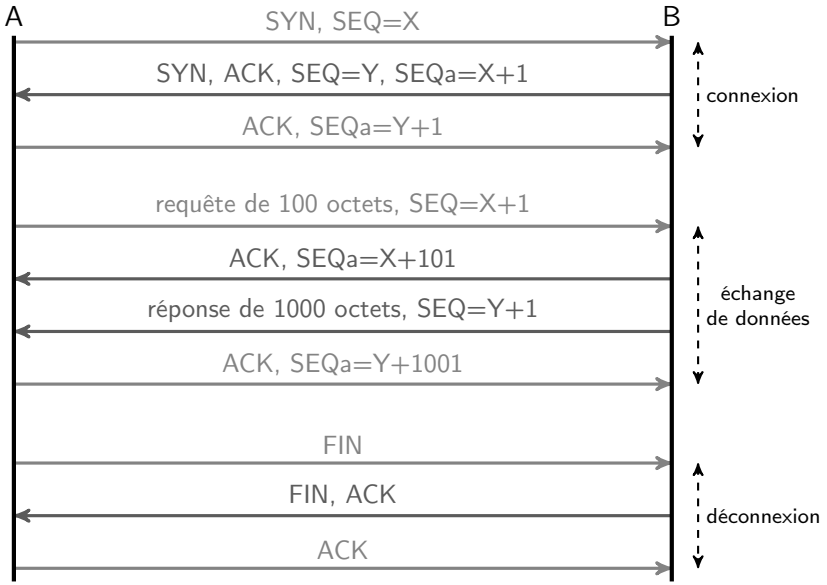


- TCP ($\geq$ 20 octets)



SEQ, SEQa : numéros de séquence et d'acquittement
SYN, FIN, ACK, RST, PSH, URG : bits utilisés pour le contrôle de l'échange

Exemple d'échange TCP



X et Y : num. de séquence initiaux

Plan

Attaques par logiciel

installation de logiciel malveillants (**maliciels**) sur la machine ciblée, généralement via l'ouverture de mails frauduleux, la consultation de sites web contaminés, ...

- virus
  - auto-répliquant, conçu pour se propager sur le réseau
  - fonctionne en infectant (s'insérant dans) un logiciel légitime (dit hôte)
- ver
  - auto-répliquant, conçu pour se propager sur le réseau
  - pas besoin de logiciel hôte
- wabbit
  - auto-répliquant mais sans propapagation sur le réseau
  - ex : fork bomb
- cheval de Troie
  - logiciel apparemment légitime mais qui contient une fonctionnalité malveillante (p.ex., envoyer des données confidentielles à un tiers)
- backdoor
  - logiciel qui va ouvrir un accès sur la machine à un attaquant
  - généralement installé par un cheval de Troie

Attaque par force brute/par dictionnaire

- utilisées pour trouver (**casser**) un mot de passe
- attaque par force brute
  - tenter toutes les combinaisons possibles de mot de passe
  - très simple à programmer
  - mais très peu susceptible de fonctionner pour des mots de passe pas trop courts
  - et pas très discret si l'attaque se fait sur le réseau
- attaque par dictionnaire
  - tenter tous les mots de passe dans un dictionnaire donné : admin, 123456, azerty, ..., combiné éventuellement avec des infos personnelles sur la victime (date de naissance, surnom, ...), ...

Attaque par déni de service (DoS : Denial of Service)

- objectif : paralyser une machine/un réseau en saturant ses ressources
- exemples :
  - ping of death
    - ICMP request de taille $>$ à la taille max. (65 536 octets)
    - un bug dans certains OS (avant 1998) avait pour conséquence de bloquer complètement la machine
  - land
    - segment TCP avec IP source = IP dest. et port source = port dest.
    - conséquence : victime paralysée (la victime se renvoie le paquet à elle même en boucle) sur des vieux OS (avant 1997)
  - SYN flood (rafale de SYN)
    - envoi d'une rafale de paquets SYN sur un port ouvert de la victime avec IP source = IP aléatoire
    - mémoire de la victime saturée par des connexions semi-ouvertes
  - smurf
    - envoi d'une rafale d'ICMP request avec IP source = IP de la victime et IP destination = IP de diffusion
    - conséquence : toutes les machines dans l'adresse de diffusion vont envoyer un ICMP reply à la victime
- déni de service distribué (DDoS) : lancer un DoS depuis plusieurs machines préalablement compromises (machines zombies)

Attaque de type man-in-the-middle

- L'attaquant se trouve sur le chemin entre deux victimes.
  (Ou a fait en sorte que les paquets échangés entre les deux victimes passent par lui via, p.ex., une usurpation d'IP.)
- Il peut ensuite capturer ou altérer les paquets échangés.
- Peut également servir à outrepasser les mécanismes d'autentification par chiffrement asymétrique.
- difficile à détecter car du point de vue des victimes, tout semble OK

Attaque par usurpation

- principe : usurper l'adresse IP, l'adresse MAC ou le nom d'une victime
- Pas une attaque en soit, mais peut être un préalable à d'autres attaques.
- Sur un réseau local : empoisonnement de cache ARP
  - objectif : l'attaquant (A) veut capter le trafic entre deux victimes V et W
  - comment ? A envoie à V un paquet ARP avec @MAC = A et @IP = W

  $\Rightarrow$ les paquets de V pour W sont reçus par A (car envoyés sur son @MAC)
- Usurpation de nom : empoisonnement de cache DNS
  - objectif : falsifier l'adresse IP associée par un serveur DNS à un nom
  - fonctionnement (on suppose que A veut que le serveur DNS V associe le nom `ma-banque.fr` à l'IP $A.B.C.D$ sur lequel s'exécute un site web frauduleux) :
    1. A$\rightarrow$V : req. DNS : quelle est l'@IP de `ma-banque.fr` ?
    2. V$\rightarrow$X : relai de la req. DNS à un autre serveur DNS X
    3. A$\rightarrow$V : rép. DNS : l'@IP de `ma-banque.fr` est $A.B.C.D$.
  - fonctionne si V accepte (3) comme une réponse valide à (1), c'est-à-dire :
    - IP source = IP de X
    - port dest. = port depuis lequel V a envoyé sa requête
    - Id. de requête DNS = Id. de requête utilisé par V

    $\Rightarrow$ A doit forger de multiples réponses pour espérer qu'une soit acceptée.

Attaques sur les serveurs web

Les failles des serveurs web sont souvent dûes à des faiblesses de programmation (sites web mal programmés) :

- injection de commandes
  faire exécuter au serveur web une commande shell malicieuse
- injection SQL
  faire exécuter au serveur web une requête SQL malicieuse
- XSS (Cross-site scripting)
  - L'attaquant insére du code javascript malicieux dans la base de données du serveur web via un formulaire mal programmé.
    - ex de code malicieux : envoi des cookies du client vers un serveur web contrôlé par l'attaquant
  - Quand un utilisateur légitime charge une page du site, le code est exécuté.
    $\Rightarrow$ l'utilisateur envoie ses cookies au serveur web de l'attaquant

Plan

3. Outils et architectures de sécurité

Quelques outils et techniques de sécurité

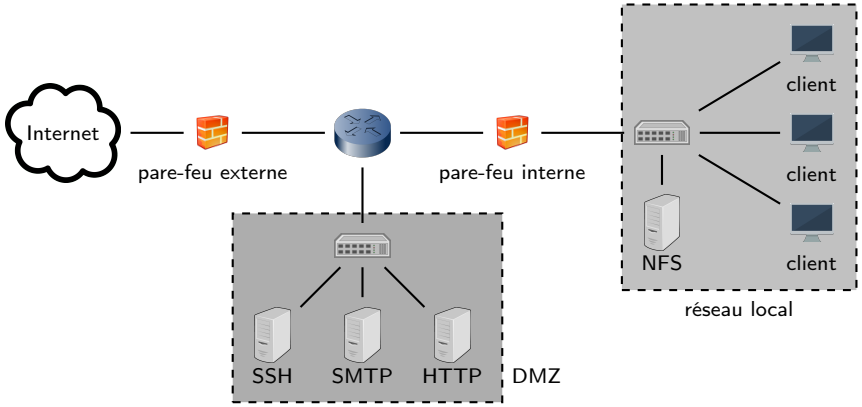
- pare-feu
  protège un réseau/un hôte en bloquant les flux non autorisés
- NAT (Network Address Translation)
  cache un réseau en utilisant une seule (ou quelques) IP pour tous ses hôtes
- serveur VPN
  crée des tunnels sécurisés entre hôtes/réseaux distants
- IDS (Intrusion Detection System)
  écoute le trafic et détecte les attaques potentielles
- IPS (Intrusion Prevention System)
  un IDS capable de prendre des mesures de protection (p.ex., bloquer des flux)
- VLAN (Virtual Local Area Network)
  sépare les flux sur un réseau local
- proxy et reverse proxy
  passerelle de niveau applicatif permettant, p.ex., de contrôler les flux (p.ex., bloquer l'accès à un site web frauduleux)
- anti-virus
  identifie et élimine les logiciels malveillants
- honeypot
  système avec des failles volontaires, utilisé pour attirer/leurrer des attaquants

Quelques bonnes pratiques

- protéger au maximum les équipements dédiés à la sécurité
  par exemple, pour un pare-feu :
  - ne pas créer d'utilisateurs
  - installer le moins de logiciels possibles ($\Rightarrow$ moins de failles potentielles), p.ex., pas d'interface graphique
  - ne lancer aucun service
  - bloquer tous les paquets destinés au (ou émis par le) pare-feu
- utiliser des équipements/logiciels de différents constructeurs/développeurs
  - Si une faille critique est découverte sur un pare-feu X de la marque Y, et que le réseau contient uniquement des pare-feu X, tout le réseau est compromis.
- segmenter le réseau selon le degré de sécurité/confidentialité attendu
  - réseau des serveurs
  - réseau d'administration
  - réseau des utilisateurs
  - réseau public

  ...
- journaliser tous les évènements sur les serveurs et équipements réseau

Les DMZ (DeMilitarized Zone)



- Objectif : protéger un réseau local tout en rendant un certain nombre de services accessibles depuis l'extérieur (Internet).
- Le pare-feu externe laisse entrer le trafic vers les serveurs de la DMZ.
- Le pare-feu interne bloque tout les flux initiés depuis l'extérieur ou la DMZ.
- Avantage : le réseau local reste protégé même si la DMZ est compromise.

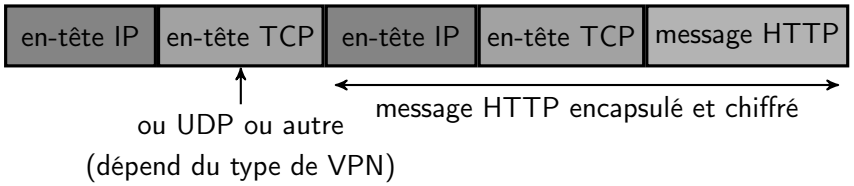
Plan

Principe des VPN

- VPN = Virtual Private Network
- Objectif : offrir un **tunnel**
  - virtuel ⇔ pas de liaison physique dédiée
  - privé ⇔ **confidentialité** et **intégrité** des échanges

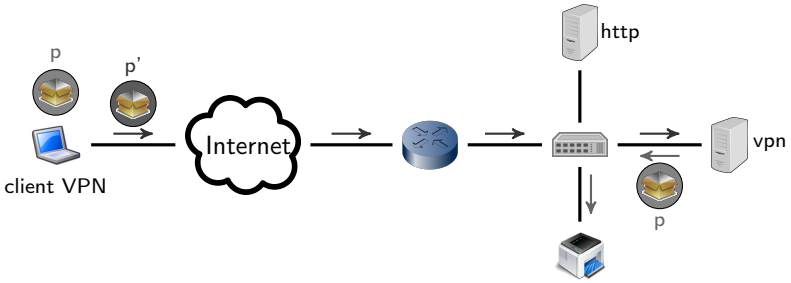
  sur un réseau public (p.ex., Internet)
- tunnel = chemin réseau sur lequel vont circuler des paquets encapsulant
  - des trames (tunnel de niveau 2) ;
  - ou des paquets (tunnel de niveau 3) ;
  - ou ...

  chiffrés.
- Ex : structure d'un message HTTP circulant sur un tunnel de niveau 3 :



VPN — Exemple 1

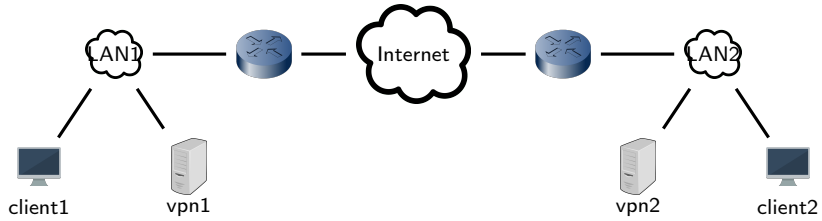
- accès sécurisé d'un utilisateur (le client) à un réseau distant



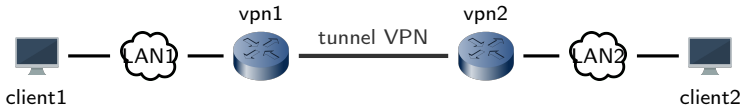
- Le client (C) peut accéder aux services et équipements du réseau distant comme s'il se trouvait sur ce réseau.
- Quand C veut envoyer un paquet p à l'imprimante (I) :
  - C chiffre p puis l'encapsule dans un nouveau paquet p'
  - C envoie p' au serveur VPN (V)
  - V décapsule le paquet puis déchiffre son contenu
  - V envoie p à l'imprimante

VPN — Exemple 2

- interconnexion sécurisée de réseaux distants



- Les clients peuvent communiquer par l'intermédiaire des serveurs VPN.
- Avec un tunnel de niveau 3, l'architecture logique est la suivante :



OpenVPN

https://openvpn.net

- logiciel libre
- multi-plateformes (Linux, Mac, Android, Windows, ...)
- port par défaut : 1194 avec UDP (par défaut) ou TCP
- intègre de nombreux algorithmes de chiffrement
- possibilité de compression des données circulant dans le tunnel
- possibilité de créer des tunnels
  - de niveau 2 (les serveurs VPN fonctionnent comme des switchs) ;
  - ou de niveau 3 (les serveurs VPN fonctionnent comme des routeurs).
- possibilité de chiffrement
  - symétrique (avec une clé partagé entre serveurs) ;
  - ou asymétrique (avec TLS).
- en pratique : création d'une interface virtuelle représentant le tunnel
  - de type tap pour un tunnel de niveau 2 ;
  - et de type tun pour un tunnel de niveau 3.

(plus de détails dans le TP 1)

Plan

Principe et application du NAT

- NAT = Network Address Translation
- Principe : modification par le routeur des IP et/ou n° de port apparaissant dans les en-têtes IP, TCP, ou UDP des paquets routés.
- Application : gestion de la pénurie d'adresse IP.
  - Les machines du réseau interne ont des IP dans un pool d'adresses **privées**.
  - Ces adresses privées sont valables uniquement sur le réseau interne.
  - les plages d'adresses privées (RFC 1918) :
    - 10.0.0.0/8
    - 172.16.0.0/12
    - 192.168.0.0/24
  - L'interface du routeur reliée à l'extérieur a une IP **publique** fournie par son fournisseur d'accès à Internet.
  - Cette IP publique est utilisable sur Internet.

  ⇒ diminution du nombre d'IP utilisées : un réseau peut contenir des milliers de machines utilisant des IP privées mais utilise une seule IP publique

Différents types de NAT

- NAT 1-à-1 ou NAT statique : N IP publiques ↔ N IP privées
  ⇒ ne répond pas au problème de pénurie d'adresses IP
- NAT 1-à-N ou NAT dynamique : 1 IP publique ↔ N IP privées
- PAT : translation de port — Le routeur modifie aussi les numéros de port TCP ou UDP.
- solution courante : NAT dynamique + PAT (on parle aussi de NAPT)
  - 1 seule IP publique, la passerelle NAT modifie IP et ports
  - C'est la solution que nous utiliserons.

Fonctionnement de la passerelle NAT (1/2)

1 Pour les paquets sortants, la passerelle :
  1.1 modifie l'IP et le port source (privés) par son IP publique et par un nouveau numéro de port qu'elle choisit (p.ex., aléatoirement dans une plage donnée) ;
  2.2 et mémorise l'association (IP + port privés, IP + port publics) dans une table de correspondance.

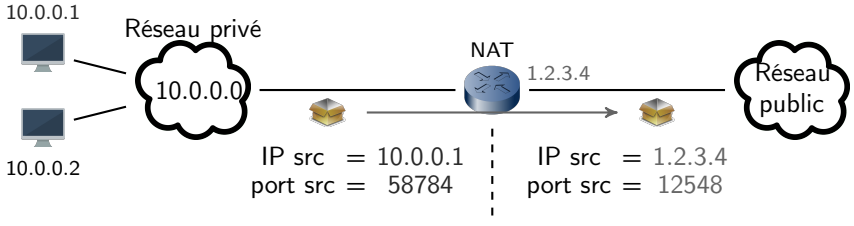


Table de correspondance

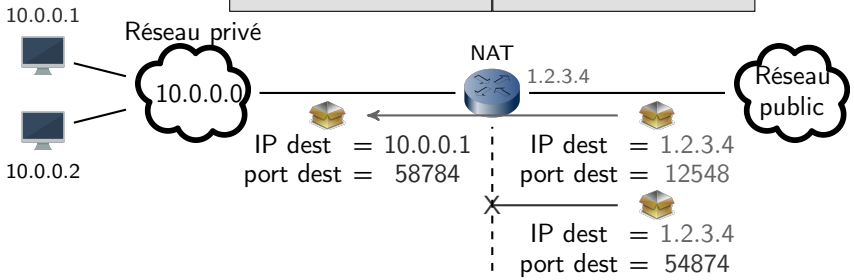
IP + port privés	IP + port publics
10.0.0.1:58784	1.2.3.4:12548

Fonctionnement de la passerelle NAT (2/2)

2 Pour les paquets entrants, la passerelle cherche dans sa table.
  2.1 si association trouvée : IP et port destination modifiés
  2.2 sinon : paquet filtré

Table de correspondance

IP + port privés	IP + port publics
10.0.0.1:58784	1.2.3.4:12548



Remarques sur le fonctionnement de la passerelle NAT

- La sortie d'un paquet ouvre le port choisi par la passerelle.
- La passerelle joue le rôle de filtre : elle
  - laisse entrer les paquets envoyés en réponse à des paquets sortis du réseau ;
  - et bloque les autres.
- Tout client sur le réseau public qui essaie de contacter un serveur sur le réseau privé verra donc ses paquets rejetés.
- Donc, si on veut qu'un serveur du réseau interne soit accessible depuis l'extérieur, il faut rajouter statiquement une ligne dans la table.
  - Ex : l'administrateur ajoute (priv. = 10.0.0.2:80, pub. = 1.2.3.4:80) pour rendre le serveur web sur 10.0.0.2 accessible depuis l'extérieur.
  - Cela se fait par l'ajout d'une règle au pare-feu.

Avantages et inconvénients du NAT

Avantages

- solution à la pénurie d'adresses IPv4
  - Un réseau de grande taille peut utiliser une seule IP publique.
- en terme de sécurité :
  - permet de cacher tout un réseau derrière une seule IP
  - filtre certains paquets entrant non sollicités

Inconvénients

- plus de travail pour le pare-feu : recalcul des codes d'erreur IP, TCP, ...
- Comment faire pour les protocoles de type peer-to-peer ?
  N'importe quelle IP du réseau peut alors être contactée depuis l'extérieur.
- Comment faire si les données contiennent elles aussi des IP privées ?
  Ces IP ne sont pas modifiées par le pare-feu !

⇒ besoin de protocoles/solutions supplémentaires (p.ex., STUN)

Plan

Rôle d'un pare-feu

- équipement à la frontière entre
  - un réseau local (ou interne)
  - et un réseau externe
- C'est généralement le routeur qui relie le réseau interne à l'extérieur.
- fonctions :
  - protéger le réseau interne
  - translation d'adresses
- comment ?
  - en examinant les paquets traversés et en leur appliquant des **règles** (laisser passer, bloquer, modifier, ...) définies par une **politique de sécurité**
- par exemple :

- On a aussi des pare-feux protégeant un seul hôte.

Les catégories de pare-feu : sans état (*stateless*)

- aucun historique des paquets traités

⇒ décisions de routage prises indépendamment les unes des autres

- décisions de routage statiques définies uniquement sur les en-têtes du paquet (IP, n° de port, drapeaux TCP (SYN, ACK, ...), ...)
- avantages
  - rapide
  - ne nécessite pas de mémoire
- inconvénients
  - faible protection
    Un attaquant peut envoyer des paquets sur les ports ouverts.
  - ne peut pas faire de NAT

Les catégories de pare-feu : à état (*stateful*)

- maintient une table des connexions ouvertes
  (extension du concept de connexion aux protocoles ICMP et UDP)
- peut bloquer/accepter le trafic selon l'état de la connexion
- p.ex. : autoriser les paquets entrants si la connexion a déjà été établie
- avantages :
  - rapide
  - moins de règles et règles plus simples qu'en mode sans état
  - plus sécurisé que le mode sans état
- inconvénients :
  - le traçage des connexions nécessite de la mémoire
  - pas de compréhension des données transportées ($\Rightarrow$ attaques possibles sur les protocoles autorisés)

Les catégories de pare-feu : applicatif

- pare-feu de niveau 7
- regarde dans les données du paquet pour prendre une décision
- pare-feu spécifique à un type d'application
  - Exemple : WAF = Web Application Firewall
- peut bloquer des attaques spécifiques à un protocole
  - pour le WAF : bloquer les injections SQL
- avantage :
  - peut bloquer des attaques exploitant des failles de protocole
- inconvénient :
  - peut consommer beaucoup de ressources (en calcul et en mémoire)

Présentation d'iptables

- logiciel libre sous licence GPL
- présent dans le noyau Linux depuis la version 2.4 (2001)
- successeur de ipchains
- fonctionnalités :
  - filtrage stateful
  - translation d'adresses
  - modification des paquets traversés

Les tables

- Les règles appliquées par iptables sont stockées dans des **tables**.
- À chaque table correspond un type de règle.
- Les deux tables les plus importantes :
  - **filter** — règles de filtrage de paquets
  - **nat** — règles de translation d'adresse (voir plus loin)
- iptables propose d'autres tables, par exemple :
  - **mangle** — règles de modification des paquets (ex : modification de la durée de vie d'un paquet)

Dans ce cours nous n'utiliserons que les tables filter et nat.

Les chaînes et règles

Chaînes

- À chaque table est associée un ensemble de **chaînes**.
- Une chaîne indique dans quel contexte des règles sont appliquées.
- 1 chaîne ⇔ séquence de règles à appliquer sur le paquet
- La chaîne s'arrête dès qu'une règle peut être appliquée (sauf pour l'action **LOG**, cf. plus bas).
- Si aucune règle ne s'applique ⇒ on applique une **politique par défaut**.

⇒ L'ordre des règles dans une chaîne a de l'importance.

Les règles

- 1 **règle** = 1–N condition(s) sur le paquet + une action à effectuer
- exemples de conditions :
  - L'adresse IP source du paquet est 192.168.1.1.
  - Le port de destination est le port 80.
  - Le paquet contient un message ICMP.
- L'action à effectuer dépend de la table dans laquelle la règle se trouve.

La table filter

Les chaînes

- **INPUT** — appliquée à tout paquet destiné à un processus local (s'exécutant sur le routeur)
- **OUTPUT** — appliquée à tout paquet envoyé par un processus local
- **FORWARD** — appliquée à tout paquet routé par (qui traverse) le routeur

Les actions

- **DROP** — détruire le paquet
- **REJECT** — détruire le paquet + envoyer un message d'erreur ICMP à la source
- **ACCEPT** — laisser passer le paquet
- **LOG** — sauvegarder une trace dans le journal

La table nat

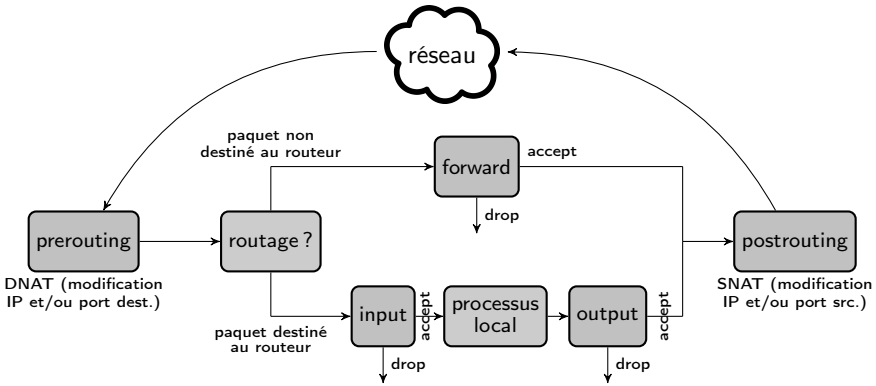
Les chaînes

- **PREROUTING** — règles DNAT appliquées sur le paquet immédiatement après sa réception
- **POSTROUTING** — règles SNAT appliquées sur le paquet avant son émission

Les actions

- **DNAT** — modification de la destination (IP et/ou port)
  valable uniquement dans la chaîne PREROUTING
- **SNAT** — modification de la source (IP et/ou port)
  valable uniquement dans la chaîne POSTROUTING
- **MASQUERADE** — modification de l'IP source par l'IP de l'interface de sortie (voir exemples plus loin)
  valable uniquement dans la chaîne POSTROUTING

Traversée des chaînes



table/chaîne à utiliser pour quelques cas de figure

- Filtrer les paquets destinés au/émis par le routeur.
  filter/input et **filter/output**
- Filtrer les paquets qui traversent le routeur (quel que soit le sens).
  filter/forward
- Autoriser l'accès à un serveur local depuis l'extérieur.
  nat/prerouting
- Modifier l'IP source (interne) des paquets allant vers l'extérieur par l'IP externe.
  nat/postrouting


```
# -L : afficher le contenu d'une table (de toutes ses chaînes)
iptables -t filter -L

# afficher le contenu d'une chaîne
iptables -t nat -L PREROUTING

# -F (flush) : vider le contenu d'une table (de toutes ses chaînes)
iptables -t filter -F

# vider le contenu d'une chaîne
iptables -t filter -F FORWARD

# -P : fixer la politique par défaut
iptables -t filter -P FORWARD DROP
```


Syntaxe générale :

```
iptables -t <table> -A <chaîne> <cond> -j <action> <options>
```

avec :

- `cond` = conditions que doit remplir le paquet pour que la règle s'applique
- `action` = action à réaliser si les conditions sont remplies (-j = jump)
- `options` = options de l'action

Remarques :

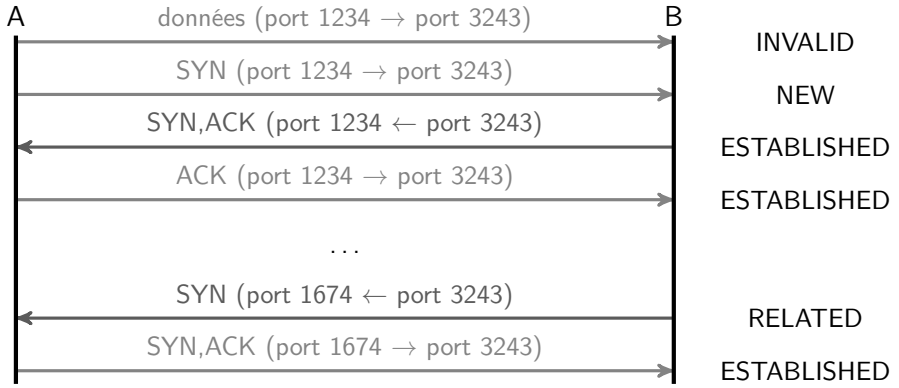
- Avec `-A`, la règle est ajoutée en fin de chaîne.
  Pour ajouter une règle à la position i dans la chaîne :
  remplacer `-A <chaîne>` par `-I <chaîne> <i>`
- Pour supprimer la règle à la position i : option `-D <chaîne> i`.
- Si les conditions sont vérifiées par le paquet, on effectue l'action et on sort de la chaîne (sauf si `action` = LOG).


```
# refuser de recevoir ou d'émettre des paquets ICMP
iptables -t filter -A INPUT -p icmp -j DROP
iptables -t filter -A OUTPUT -p icmp -j DROP

# accepter de router les paquets ICMP provenant de l'interface eth0
# et redirigés vers eth1
iptables -t filter -A FORWARD -p icmp -i eth1 -o eth1 -j ACCEPT

# sauvegarder dans le journal les paquets TCP dest. au port 80
iptables -t filter -A FORWARD -p tcp --dport 80 -j LOG

# rejeter les paquets à router de 10.0.1.1 vers le réseau 1.2.0.0/16
iptables -t filter -A FORWARD -s 10.0.1.1 -d 1.2.0.0/16 -j REJECT
```

Suivi des connexions

- Les paquets peuvent être filtrés selon l'état de la connexion.
- On utilise alors l'option `-m state --state <etat(s)>`
- Les différents états de connexion qui peuvent être associés à un paquet :
  - **NEW** — paquet initiant une nouvelle connexion
  - **INVALID** — paquet ne pouvant pas être associé à une connexion
  - **ESTABLISHED** — paquet faisant partie d'une connexion déjà établie
  - **RELATED** — paquet initiant une nouvelle connexion associée à une autre connexion déjà établie
    - État utilisé pour certains protocoles qui vont utiliser plusieurs ports.
    - Ex : lors d'un transfert de fichiers avec FTP le serveur peut se connecter sur un autre port du client.

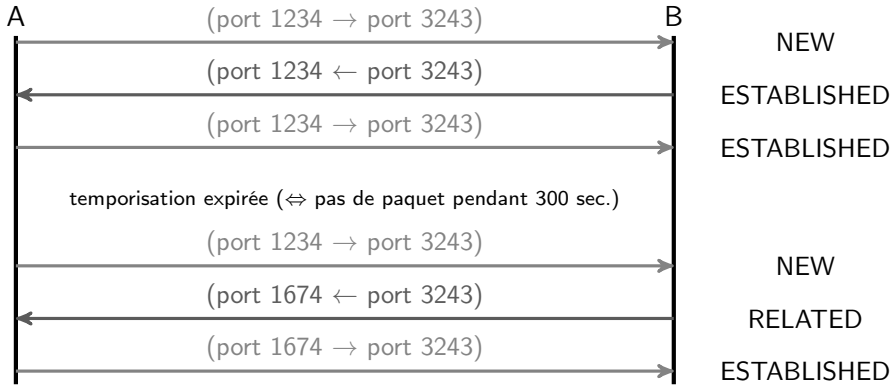
Suivi des connexions — Cas de TCP



RELATED : ouverture d'une nouvelle connexion TCP par B alors qu'une connexion est déjà active sur d'autres ports

Suivi des connexions — Cas d'UDP et d'ICMP

Utilisation de temporisations pour étendre le concept de connexion aux protocoles non connectés.



Plan

7. Systèmes de détection et de prévention des intrusions

Pare-feux, IDS et IPS

Le pare-feu

- protège un réseau/une machine en appliquant des règles de filtrage

L'IDS (**I**ntrusion **D**etection **S**ystem)

- analyse le trafic réseau à la recherche d'attaques potentielles
- génère une alarme (log, mail, ...) en cas d'attaque présumée
- mais n'effectue aucune action de protection

L'IPS (**I**ntrusion **P**revention **S**ystem)

- un IDS capable de prendre des mesures de protection
- p.ex. : changer dynamiquement les règles de filtrage, bloquer l'IP source d'une attaque ($\approx$ pare-feu dynamique)

Résultat d'inspection d'un IDS

- Un événement (p.ex. l'arrivée d'un paquet)
  - peut être une attaque ou pas ;
  - peut générer une alarme ou pas.

  $\Rightarrow$ 4 situations possibles :

		Attaque ? Oui	Attaque ? Non
Alarme levée ?	Oui	Vrai positif	Faux positif
	Non	Faux négatif	Vrai négatif

- Objectif : minimiser les
  - les faux négatifs — attaque non détectée ;
  - et les faux positifs — alarme levée alors qu'aucune attaque n'a lieu.

Méthodologies de détection d'un IDS

- Détection basée sur les **signatures**
  - recherche de **signatures** d'attaques connues dans les paquets capturés

    signature = élément (port particulier, mots-clés dans les données du paquet, longueur du paquet, ...) caractéristique d'une attaque
  - alarme en cas de signature reconnue dans un paquet
  - avantages : rapide, peu de faux positifs
  - inconvénients : ne détecte que les attaques connues
- Détection basée sur les **spécifications** (actions autorisées)
  - définition par un expert de la spécification de chaque protocole observé
  - alarme en cas d'événement déviant de la spécification
  - avantages : peut détecter de nouvelles attaques, peu de faux positifs
  - inconvénients : écrire les spécifications prend du temps
- Détection basée sur les **anomalies**
  - IDS intégrant de l'IA
  - phase d'apprentissage : au début, l'IDS observe le trafic normal (autorisé)
  - alarme en cas d'événement anormal (par rapport à ce qui a été appris)
  - avantages : automatique, peut détecter de nouvelles attaques
  - inconvénients : peut générer beaucoup de faux positifs/négatifs, peut manquer des attaques connues

NIDS et HIDS

NIDS = Network IDS

- IDS orienté réseau
- analyse tous les paquets circulant sur le réseau
- souvent une machine dédiée vers laquelle on copie tout le trafic

HIDS = Host IDS

- IDS orienté hôte
- installé sur la machine à protéger
- analyse les paquets destinés à l'hôte

NIDS et HIDS — Comparaison

- Le NIDS peut détecter des attaques sur tout le réseau
  mais il ne comprend pas les flux chiffrés
  et il peut être submergé en cas de trafic important.
- L'HIDS analyse les paquets reçus après déchiffrement
  et a accès à plus d'infos (journaux de l'hôte, état courant : espace disque, mémoire libre, ...) pour détecter les attaques.
  ⇒ beaucoup plus efficace
  mais il protège un seul hôte.

Le NIDS Snort

`https://www.snort.org`

- logiciel libre
- disponible sur de nombreux OS (linux, windows, mac OS, ...)
- détection d'attaques basée sur une base de données de signatures
  base de données constamment enrichie par les développeurs et utilisateurs
- détection en temps réel (snort écoute le trafic, détecte les attaques sur les paquets capturés et lève des alarmes en cas d'attaque)

Le NIDS Snort — Architecture

- Pre-processeurs
  - ensemble de plug-ins préparant le paquet pour la détection, p.ex. :
    - frag : défragmentation du paquet
    - stream : réassemblage de flux, inspection stateful (comme iptables)
    - http_inspect : normalisation des requêtes HTTP (p.ex. : réécrire `/rep1/../rep2` en `/rep2`)
  - Certains plug-ins peuvent également générer des alarmes.
- Moteur de détection
  - Des règles sont appliquées sur le paquet pour détecter les attaques.
  - une règle = une signature d'attaque
- Modules de sortie
  - différents plug-ins pour générer l'alarme, p.ex. :
    - écriture dans une base de données MySQL
    - envoi d'un message de log

Le NIDS Snort — Les règles

Syntaxe des règles :

```
action proto src psrc -> dst pdst (options)
```

- `action` : alert, log, ...
- `proto` : tcp, udp ou icmp
- `src` ou `dst` : `10.0.2.3`, `10.2.0.0/16`, any (n'importe quelle IP), `!10.0.0.0/24` (toute IP hors du réseau) ...
- `psrc` ou `pdst` : 22, `[10:50]` (10 à 50), any, `!22` (port différent de 22), ...
- `options` : critères additionels, paramétrage de l'alarme en sortie

Exemples :

```
alert icmp any any -> 255.255.255.255 any (
    msg: "message ICMP vers une adresse de broadcast";)
alert tcp any any -> 192.168.0.100 !22 (
    msg: "accès au serveur ssh sur un port non conventionnel";)
```

- Remarque : on peut écrire <> à la place de -> ⇒ la règle fonctionne alors quel que soit le sens.

Le NIDS Snort — Les règles — Options de sortie

- `msg` — préfixe du message d'alarme généré
- `rev` — numéro de révision de la règle snort
- `classtype` — identifie le type d'attaque
  - ex : `classtype: denial-of-service;`, `classtype:trojan-activity;`
- `sid` — identifiant de signature
- `reference` — référence vers la source (p.ex., une URL) décrivant l'attaque.

Le NIDS Snort — Les règles — Options de filtrage

critères TCP :

- `seq` — numéro de séquence
- `ack` — numéro d'acquittement
- `flags` — drapeaux TCP activés dans le paquet
- `flow` — informations sur l'état de la connexion

critères ICMP :

- `itype` — type ICMP
- `icode` — code ICMP
- `icmp_id` — identifiant ICMP
- `icmp_seq` — numéro de séquence ICMP

critères sur les données :

- `dsize` — taille du paquet (à l'intérieur de TCP, UDP ou ICMP)
- `content` — motif à rechercher dans les données
- `depth` — sur combien d'octets rechercher
- `nocase` — recherche insensible à la casse
- `uricontent` — motif à rechercher dans l'URI de la ressource (pour HTTP)

Le NIDS Snort — Les règles — Exemples


```
alert udp any any -> any 5632 ( # paquet udp envoyé sur le port 5632
 msg:"APP-DETECT PCAnywhere server response"; # description
 content:"ST"; # "ST" doit apparaître dans les données ...
 depth:2; # ... parmi les deux premiers octets
 sid:566; rev:10; # identifiant de signature et num. de révision
)

alert icmp any any -> any any ( # paquet ICMP
 msg:"PROTOCOL-ICMP Unusual L3retriever Ping detected"; # description
 icode:0; # code ICMP = 0
 itype:8; # type ICMP = 8 (requête echo)
 dsize:>32; # le paquet ICMP contient plus de 32 octets de données
 content:"ABCDEFGHIJKLMNOPQRSTUVWABCDEFGHI";
 depth:32;
 sid:29454; rev:2;
)

alert tcp any any -> any $HTTP_PORTS ( # paquet TCP vers un port HTTP
 msg:"WEB-CGI webdriver access"; # description
 flow:to_server,established; # la connexion doit être établie et le
     paquet envoyé du client vers le serveur
 uricontent:"/webdriver"; nocase; # l'URI demandée doit contenir la
     chaîne "/webdriver" (en majuscules ou minuscules)
 sid:808; rev:8;
)
```