Sécurité des réseaux IP

Benoît Morgan †, Vincent Nicomette ‡, Éric Alata ‡

† IRIT, ENSEEIHT, INP Toulouse
‡ LAAS-CNRS, INSA Toulouse, Université de Toulouse

7 avril 2019

Introduction – *motivation*

Sécurité des réseaux

- Augmentation de l'utilisation des systèmes informatiques et des réseaux
- Augmentation de l'utilisation des ressources réseau Internet
- Logiciels de plus en plus complexes : systèmes d'exploitation IOT
  $\Rightarrow$ Vite fait, mal fait...
- De plus en plus d'utilisateurs légitimes
  $\Rightarrow$ et d'utilisateurs malhonnêtes

Introduction – *sécurité des réseaux – propriétés de sécurité*

Confidentialité

- Des messages
- Fuite d'information
- Identité : vie privée ; anonymat

Intégrité

- De la source / destination
- Identité des entités
- De l'information

Disponibilité

- Du service : équipement réseau ; hôte réseau ; ...
- Du réseau, du médium de communication

Plan du cours

Attaques des couches OSI

Filtrage

Sécurisation des communications Réseau

Plan du cours

Attaques des couches OSI

Filtrage

Sécurisation des communications Réseau

Attaques – *sécurité des réseaux – les attaquants*

- Les motivations
  - Challenge intellectuel ; vandalisme
  - . . . (*idem* introduction)
- Qui sont-ils ? (**important**)
  - Personne extérieure → intrusion réseau
  - Personne intérieure → augmentation des privilèges
  - Administrateur → abus d'autorité

Constat similaire : beaucoup d'attaques sont exécutées par des utilisateurs authentifiés et autorisés !

- Écoute passive
  - On ne modifie pas l'information
  - Écoute réseau passive (*Sniffing*) ; scan réseau ; *wiretapping* (boucle locale *DSLAM*)
- Interception
  - L'attaquant modifie, crée ou détruit l'information
  - Rejeu ; insertion ; substitution ; suppression ; vol de session TCP ; etc.
- Déguisement
  - L'attaquant se fait passer pour quelqu'un d'autre
  - *Spoofing* IP, MAC ; *phishing* ; harponnage ; ...

Attaques – *sécurité des réseaux – principales attaques réseau*

- Cryptanalyse
  - Récupération d'information secrètes à partir d'information pulbique
  - Déchiffrement de message ; récupération de clé de chiffrement / tag d'intégrité ; ...
- Dénis de service
  - L'attaquant empêche la victime à accéder au service cible
  - Spamming (si inondation) ; inondation réseau (*flood*) ; schtroumpfage (*smurfing*) ; Dénis de service distribué (*DDOS*)
- *Logiciels malveillants / bombes logiques réseau*
  - Portes dérobées ; chevaux de troie
    ex : rumeurs CISCO / NSA
  - Protocoles volontairement faibles / vulnérables
    ex : espionnage industriel

Attaques – *couches du modèle OSI*

1. Physique
2. Liaison de données
3. Réseau
4. Transport
5. Session
6. Présentation
7. Application

Attaques – *couches du modèle OSI*

1. **Physique**
2. **Liaison de données**
3. **Réseau**
4. **Transport**
5. Session
6. Présentation
7. Application

← ce cours

Attaques – *le modèle TCP / IP*

Propriétés

- IP est un protocole réseau permettant de relayer et de router des paquets sur des média de communication quelconque.
- Principales qualités
  - Support de réseaux de grande tailles
  - Excellente résilience (aux fautes : routage, code détecteur d'erreur)
- Propriétés de sécurité garanties
  - *Aucune*
  - Construit sur un modèle de confiance dans les hôtes et média

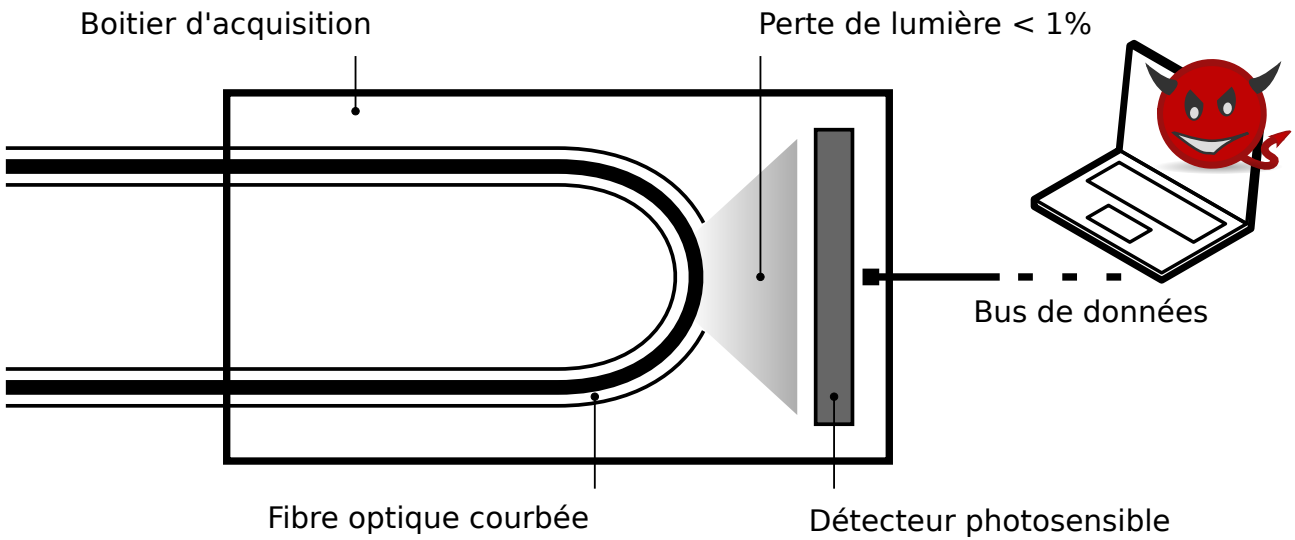
Problèmes principaux

- Pas de confiance dans le champ IP source
- Les routeurs peuvent modifier le contenu des paquets
- Les routes annoncées par les protocoles de routage ne sont pas authentifiées

Attaques – *couche physique (1)*

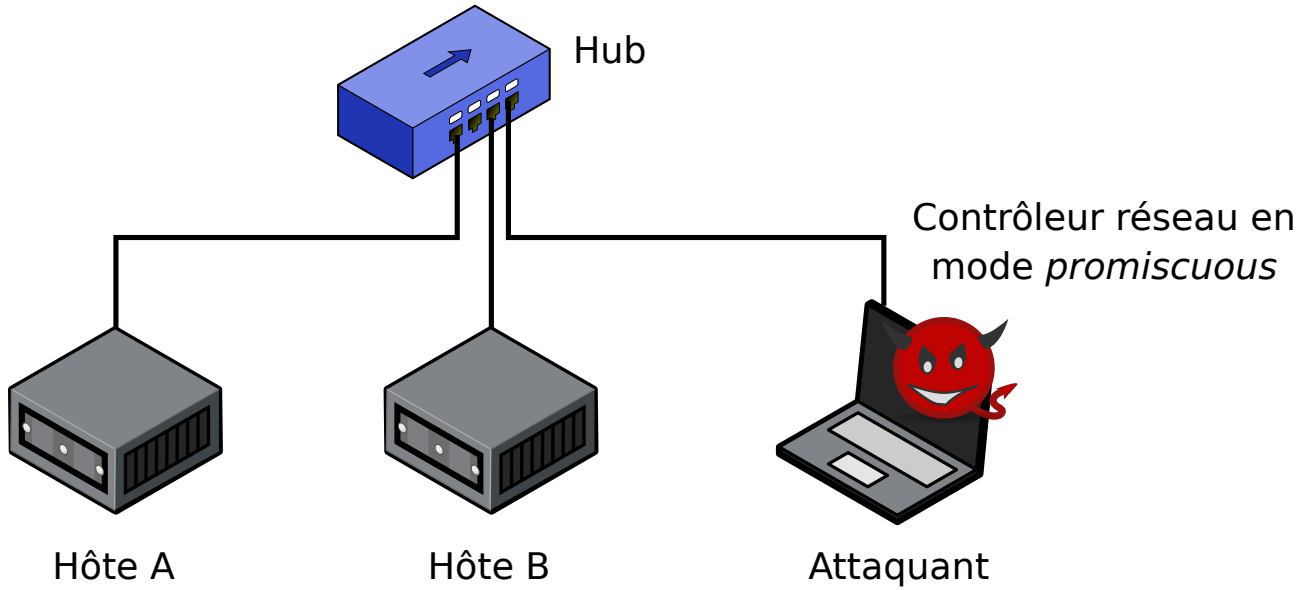
Réseau câblés

- Couper les bus et insérer un équipement pour maintenir le lien
  - Facile avec les paires cuivrées : une pince a sertir, 2 RJ45, un hub
  - Plus difficile mais faisable avec des fibres optiques
- Insertion sur routeurs, hôtes et commutateur réseau
  - Accès physique nécessaire aux baies de serveurs ou de switchs
- Canaux auxiliaires
  - Émission électromagnétique des câbles (TEMPEST)
  - Consommation électrique des équipements



Attaques – *couche liaison de données (2) - écoute réseau (sniffing)*

- Réseau local Ethernet



- L'attaquant peut écouter tout le traffic échangé par les hôtes A et B
  ⇒ ex : mots de passes en clair ; données personnelles ; . . .
- Switch réseau principalement pour diminuer la charge
  ⇒ + isolation des communications entre les hôtes
  Suffisant ? → non : attaques ARP, débordement de tables CAM, . . .

Attaques – *couche liaison de données (2) - écoute réseau (sniffing)*

- L'écoute réseau est passive, donc difficile a détecter
- Détection par cohérence du protocole
  1. Machine M_A écoute, $M_V \rightarrow$ `echo-request(@IP`$_A$, $@MAC_X$)
  2. Si $M_V \leftarrow$ `echo-reply(@IP`$_M$, $@MAC_A$)
  3. $\Rightarrow$ mode *promiscuous* sur M_A
- Détection par canal auxiliaire temporel
  1. Mesurer t_1 le *Round Trip Time* (RTT) moyen à l'aide de simples $\rightarrow$ `echo-request`, $\leftarrow$ `echo-reply` avec M_A
  2. Innonder le traffic M_A
  3. Mesurer t_2 le RTT à nouveau (1)
  4. Si $t_2 >> t_1 \Rightarrow$ mode *promiscuous*

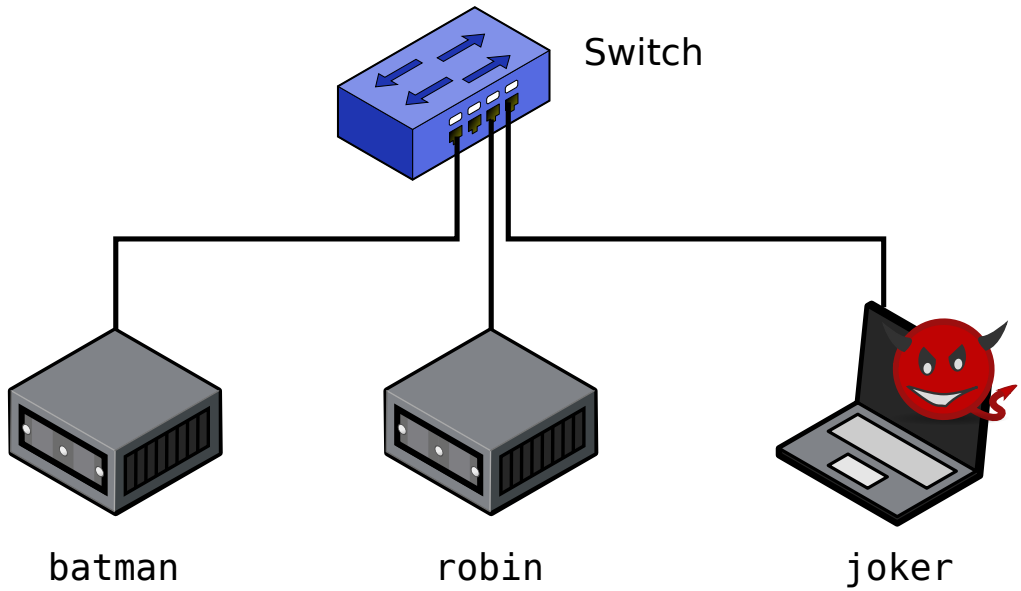
Attaques – *couche liaison de données (2)*

Address Resolution Protocol (ARP)

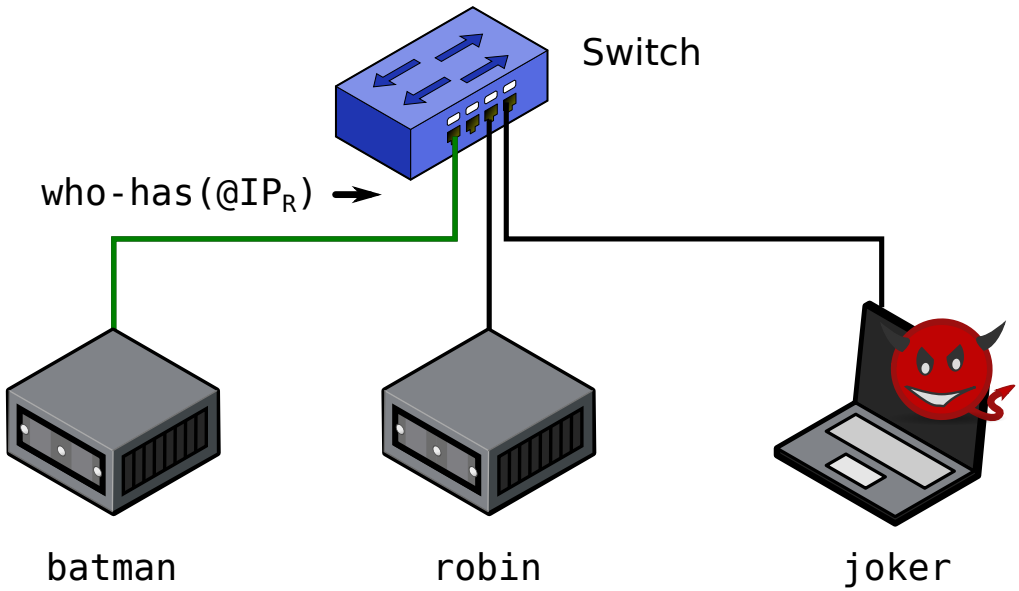
- Chaque paquet IP est encapsulé dans une trame `Ethernet`(@MAC$_{dst}$, @MAC$_{src}$)
- Le protocole ARP sert résoudre l'adresse IP d'une machine en adresse MAC
- **Comportement simplifié** : si un hôte ne pas connaît le duet (@IP$_X$, @MAC$_X$), exécute le protocole ARP

Attaques – *couche liaison de données (2) - ARP spoofing*

Interaction

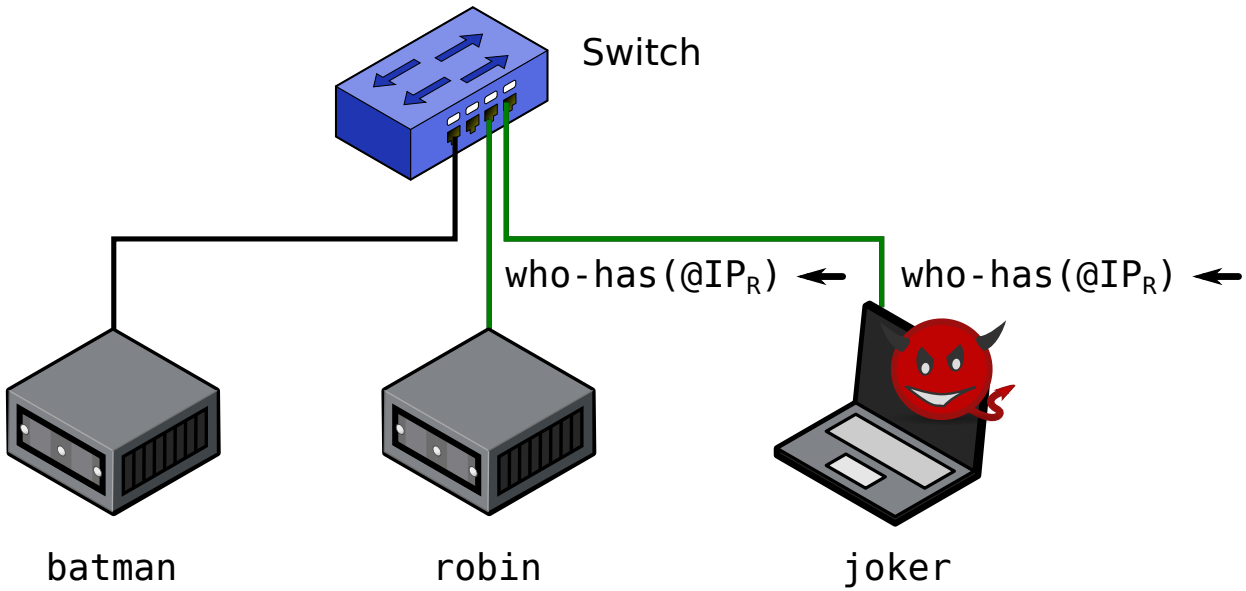


Interaction normale

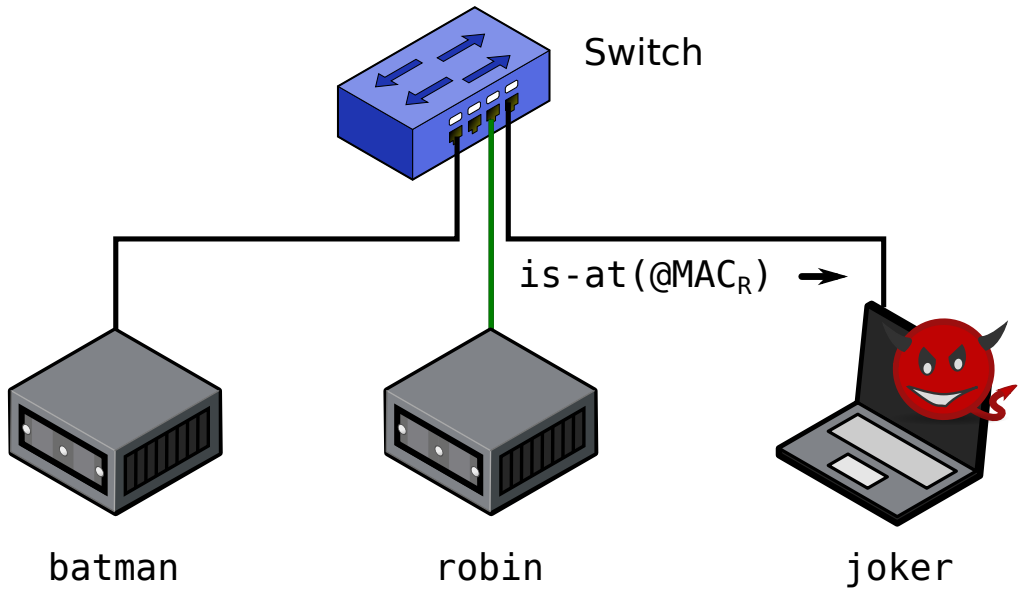


Attaques – *couche liaison de données (2) - ARP spoofing*

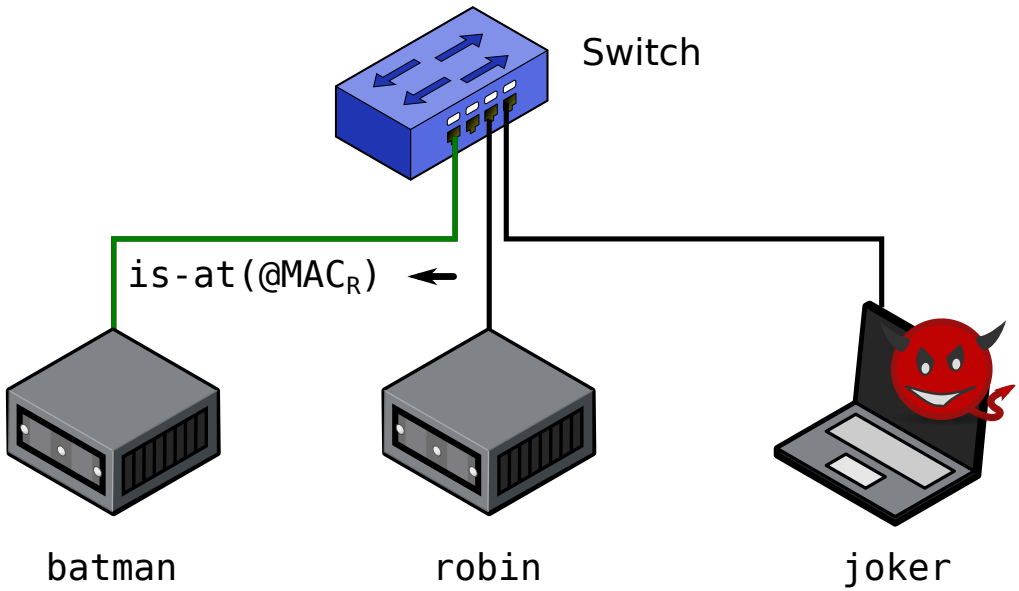
Interaction normale



Interaction normale



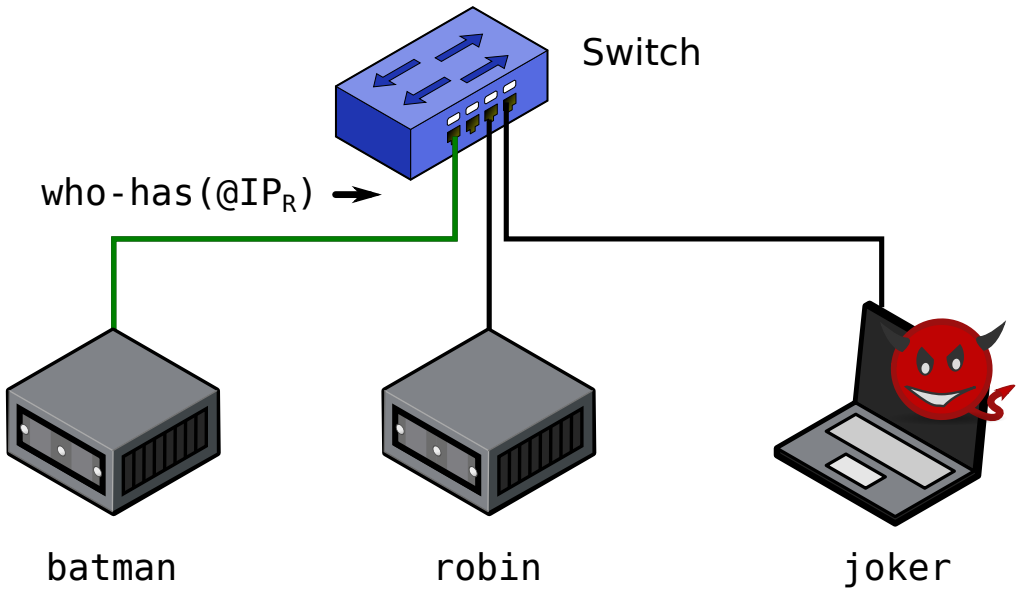
Interaction normale



Attaques – *couche liaison de données (2) - ARP spoofing*

Interaction **malveillante**

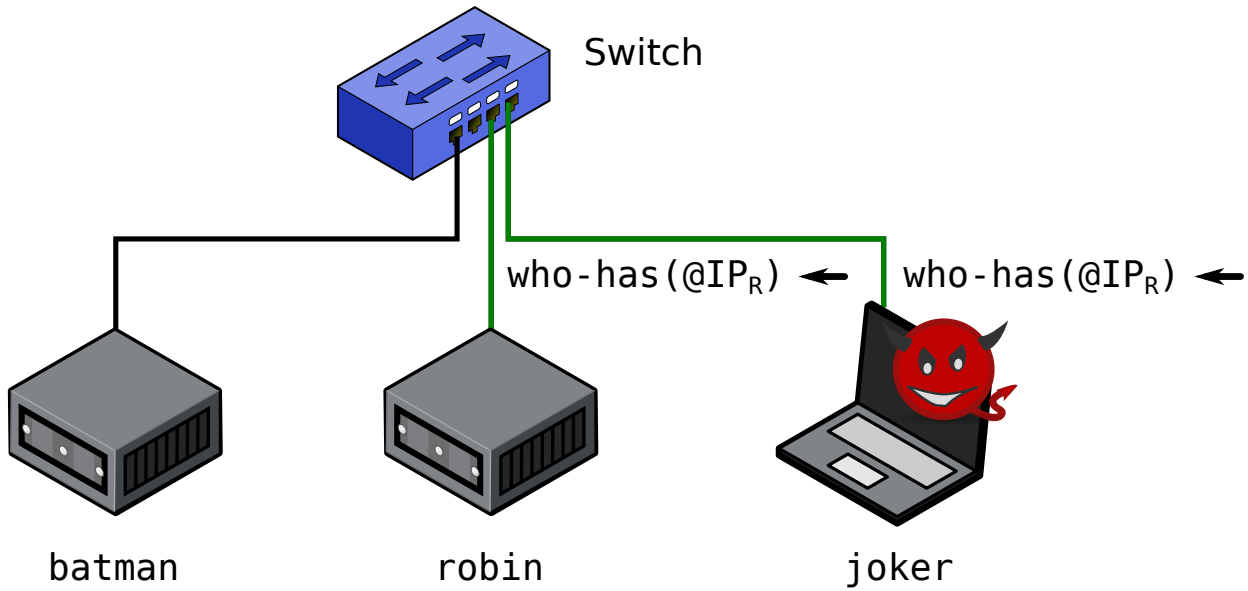
Objectif de l'attaque : vol du traffic `batman` → `robin`



Attaques – *couche liaison de données (2) - ARP spoofing*

Interaction **malveillante**

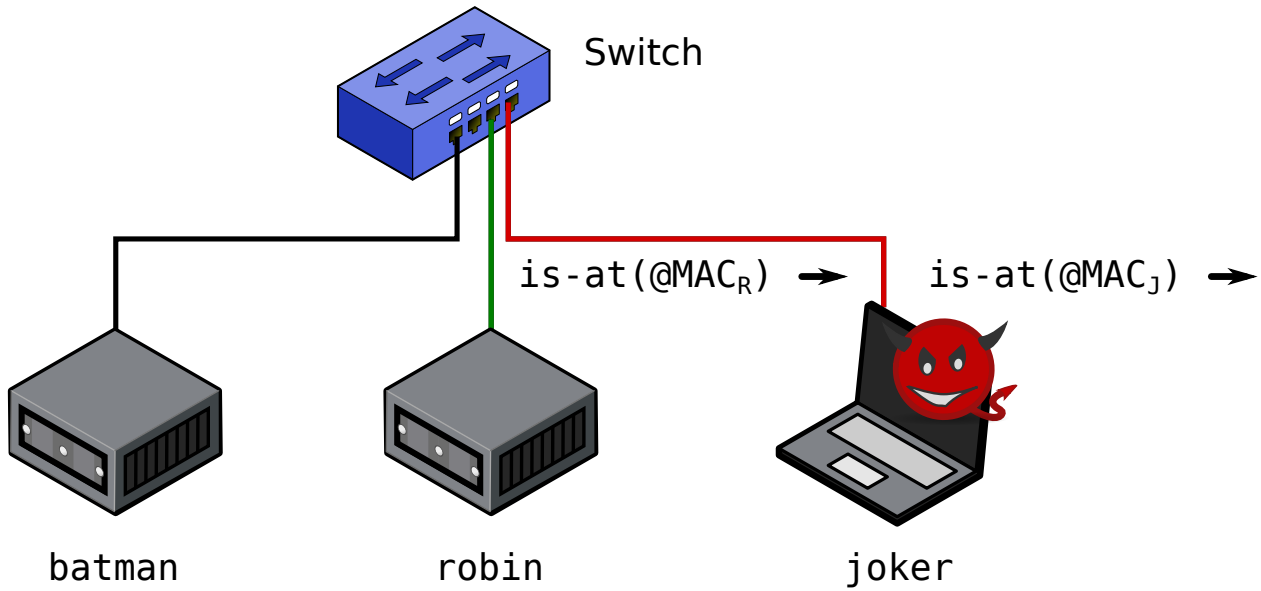
Objectif de l'attaque : vol du traffic `batman` → `robin`



Attaques – *couche liaison de données (2) - ARP spoofing*

Interaction **malveillante**

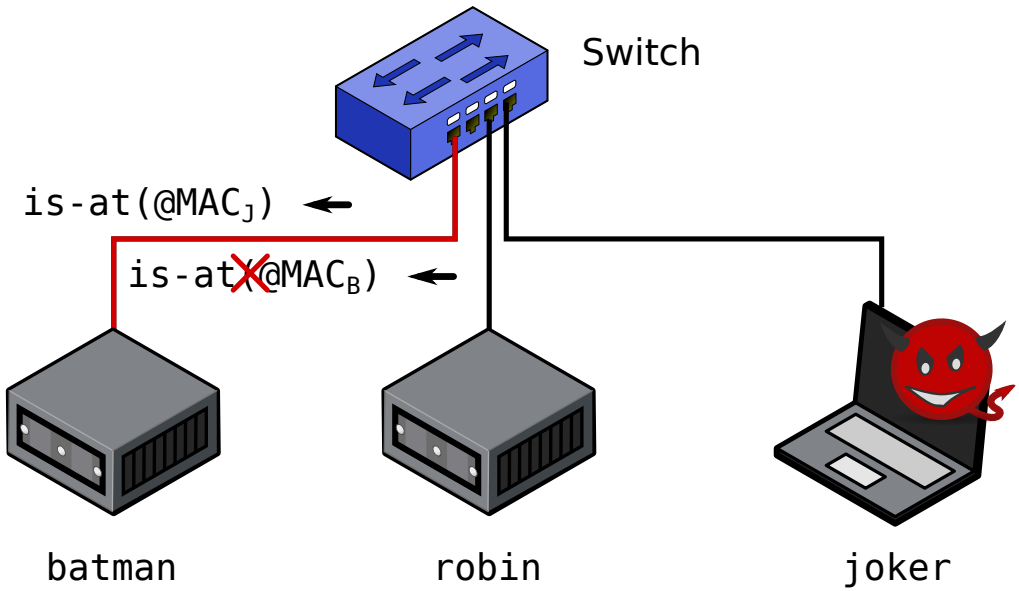
Objectif de l'attaque : vol du traffic `batman` → `robin`



Attaques – *couche liaison de données (2) - ARP spoofing*

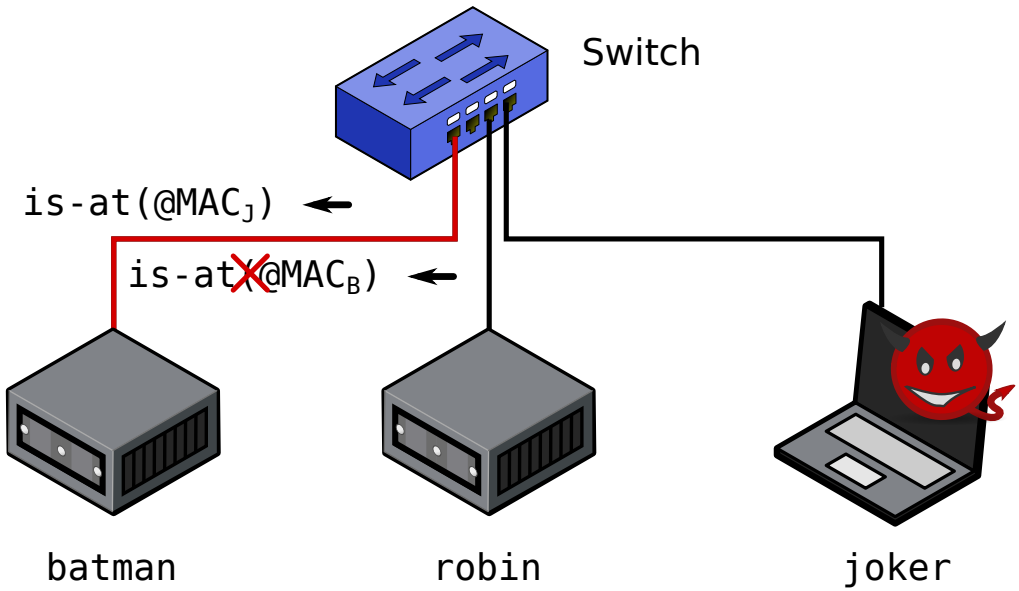
Interaction **malveillante**

Objectif de l'attaque : vol du traffic `batman` → `robin`



Attaques – *couche liaison de données (2) - ARP spoofing*

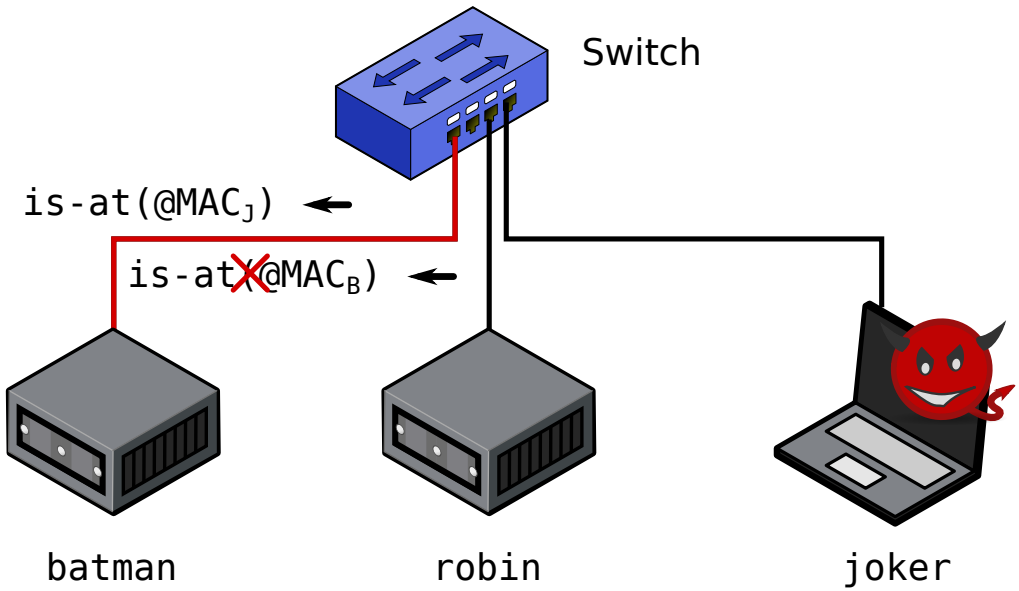
Interaction **malveillante**
Objectif de l'attaque : vol du traffic `batman` → `robin`



Situation de competition avec la réponse légitime → innondation
Cache ARP de `batman` corrompu (*ARP cache poisonning*)
Attaque a renouveller lors de l'expiration du cache → innondation

Attaques – *couche liaison de données (2) - ARP spoofing*

Interaction **malveillante**
Objectif de l'attaque : vol du traffic `batman` → `robin`



Situation de competition avec la réponse légitime → innondation
Cache ARP de `batman` corrompu (*ARP cache poisonning*)
Attaque a renouveller lors de l'expiration du cache → innondation
Démo !

Attaques – *couche liaison de données (2) - ARP spoofing*

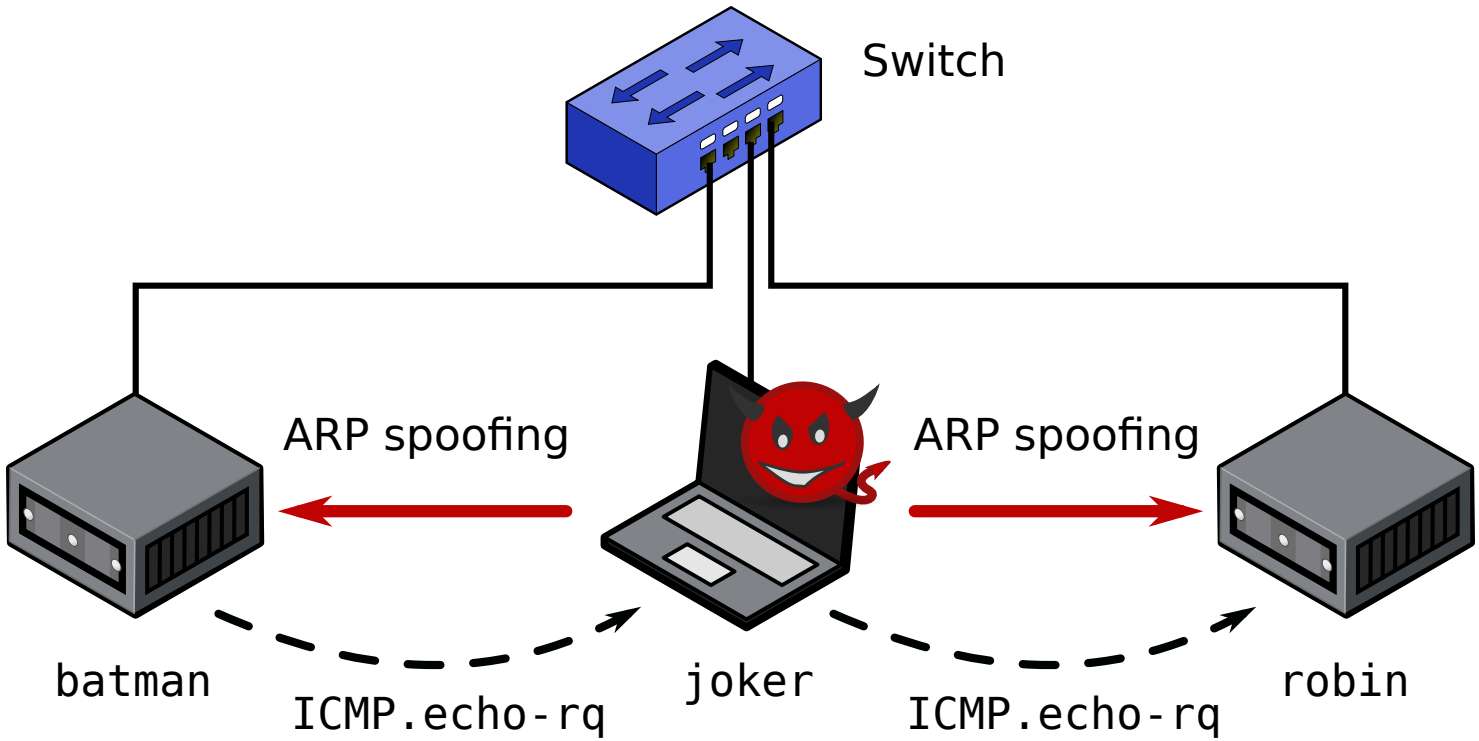
```
n7@batman:~$ ip neighbour
n7@batman:~$ ping robin -c 1
64 bytes from robin (192.168.0.2): icmp_seq=1 ttl=64 time=0.429 ms

--- robin ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.429/0.429/0.429/0.000 ms
n7@batman:~$ ip neighbour
192.168.0.2 dev eth0 lladdr ca:fe:ba:be:ca:02 REACHABLE
n7@robin:~$ ip neigh
192.168.0.1 dev eth0 lladdr ca:fe:ba:be:ca:01 REACHABLE
n7@joker:~$ sudo arp-sk -r -D 192.168.0.1 -d ca:fe:ba:be:ca:01 \
    -S 192.168.0.2 -s ca:fe:ba:be:ca:03
--- Start classical sending ---
TS: 20:24:00.308950
To: ca:fe:ba:be:ca:01 From: ca:fe:ba:be:ca:03 0x0806
    ARP For 192.168.0.1 (ca:fe:ba:be:ca:01):
        192.168.0.2 is at ca:fe:ba:be:ca:03
n7@batman:~$ ping robin -c 1
PING robin (192.168.0.2) 56(84) bytes of data.

--- robin ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

Attaques – *couche liaison de données (2) - homme dans le milieu ARP*

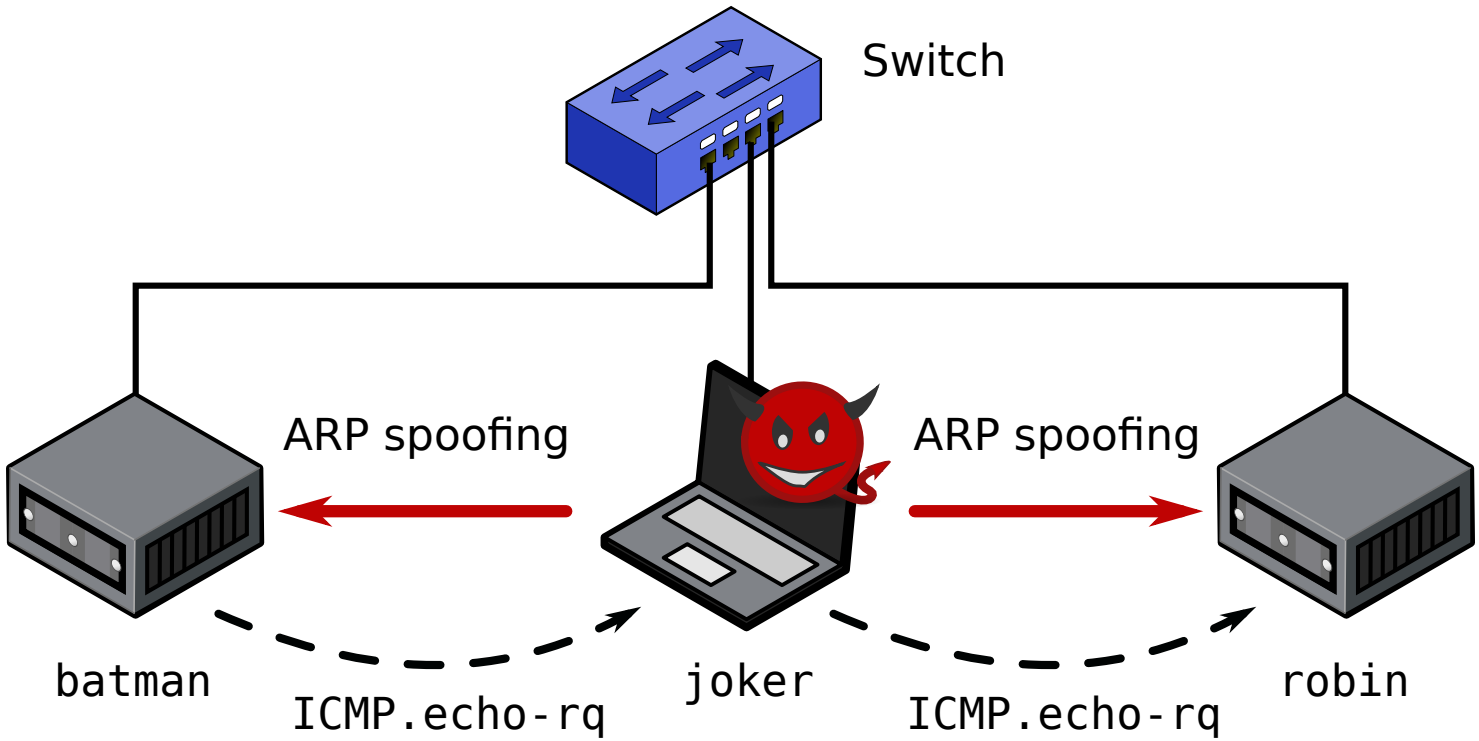
Homme dans le milieu par ARP spoofing : vol du traffic

1. ARP spoofing `batman` $\leftrightarrow$ `robin`
   - Cache ARP `batman` : $(\mathrm{IP}_{\mathrm{robin}}, \mathrm{MAC}_{\mathrm{joker}})$
   - Cache ARP `robin` : $(\mathrm{IP}_{\mathrm{batman}}, \mathrm{MAC}_{\mathrm{joker}})$
2. Joker active le relayage des paquets IP (mode router)
3. Configuration plus fine de la pile IP pour rester furtif

Homme dans le milieu par ARP spoofing : vol du traffic



Homme dans le milieu par ARP spoofing : vol du traffic



Démo !

Attaques – *couche liaison de données (2) - homme dans le milieu ARP*

```
n7@batman:~$ ping robin &> /dev/null &
n7@robin:~$ sudo tcpdump -i eth0
^C # Rien ...
n7@joker:~$ sudo arp-sk -r -D 192.168.0.1 -d ca:fe:ba:be:ca:01 \
    -S 192.168.0.2 -s ca:fe:ba:be:ca:03 &> /dev/null &
n7@joker:~$ sudo arp-sk -r -D 192.168.0.2 -d ca:fe:ba:be:ca:02 \
    -S 192.168.0.1 -s ca:fe:ba:be:ca:03 &> /dev/null &
n7@joker:~$ sudo tcpdump -i eth0
20:53:48.540416 IP batman > robin: ICMP echo request, id 558, seq 18, length 64
20:53:49.542344 IP batman > robin: ICMP echo request, id 558, seq 19, length 64
n7@robin:~$ sudo tcpdump -i eth0
^C # Rien ...
n7@joker:~$ sudo sysctl net.ipv4.ip_forward=1
n7@robin:~$ sudo tcpdump -i eth0
20:56:07.884689 IP batman > robin: ICMP echo request, id 558, seq 155, length 64
20:56:07.884736 IP robin > batman: ICMP echo reply, id 558, seq 155, length 64
n7@joker:~$ sudo tcpdump -i eth0
20:56:07.884689 IP batman > robin: ICMP echo request, id 558, seq 155, length 64
20:56:07.884736 IP robin > batman: ICMP echo reply, id 558, seq 155, length 64
```

Attaques – *couche liaison de données (2) - innondation ARP*

- Un switch Ethernet isole le traffic *unicast*

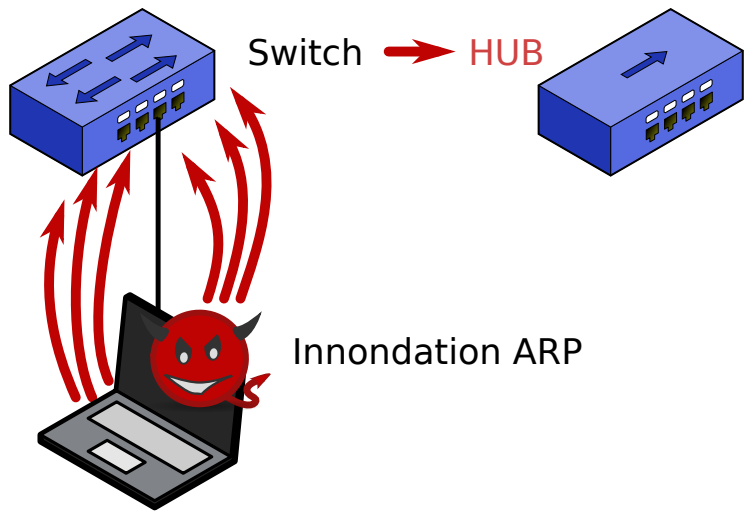
→ Commutation des trame uniquement sur le port concerné

- Le switch tient à jour une liste de duets $(\text{MAC}, \text{port\#})$
  Mémoire $\mathcal{M}$ adressable par contenu, de taille N :
  $\text{read} : \{0,1\}^{48} \rightarrow \{0,1\}^{log_2 N}$; $\text{MAC} \mapsto @|\mathcal{M}[@] = \text{MAC}$

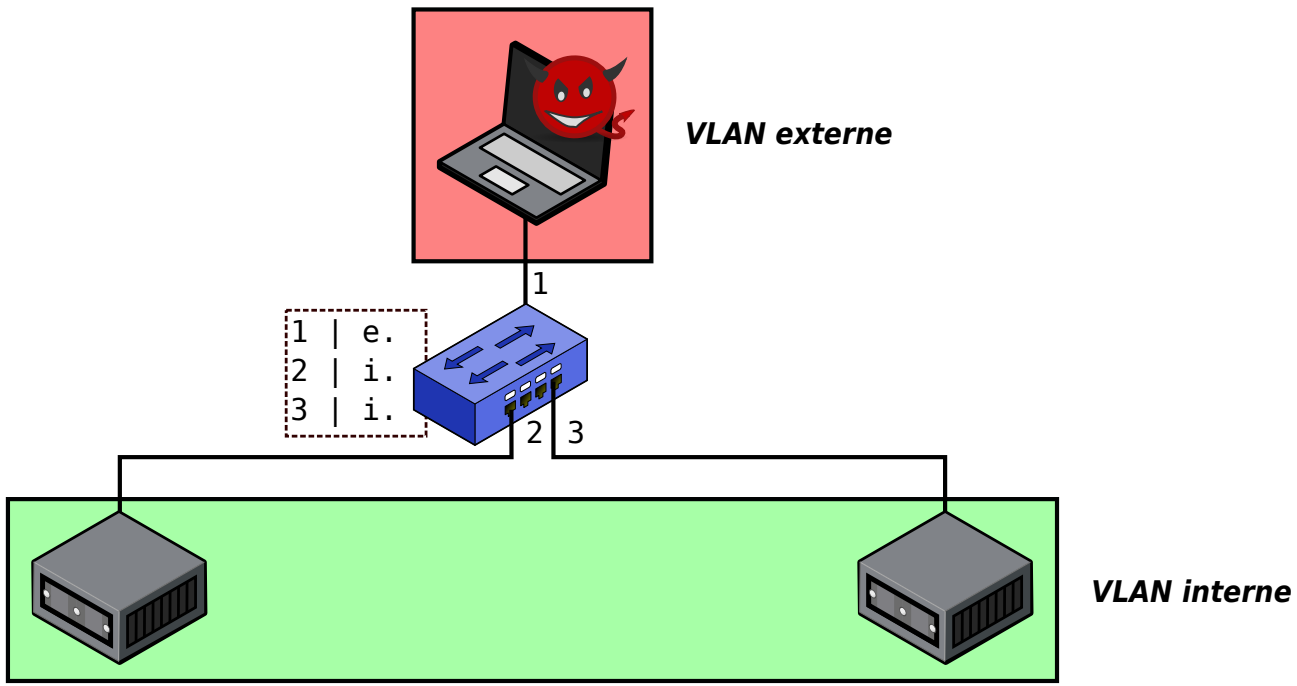
→ Mise à jour par observation du traffic réseau

- Taille limitée finie de $\mathcal{M} = N \rightarrow$ débordement possible

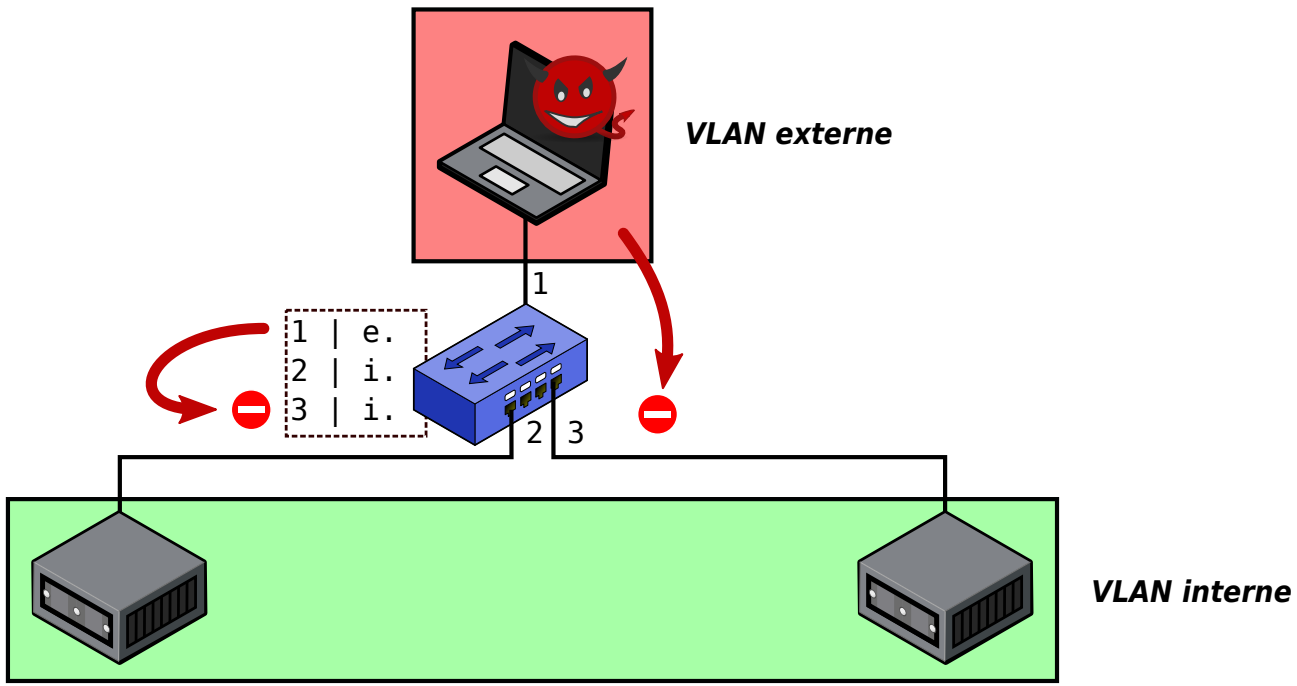
⇒ Retour au comportement du HUB Ethernet



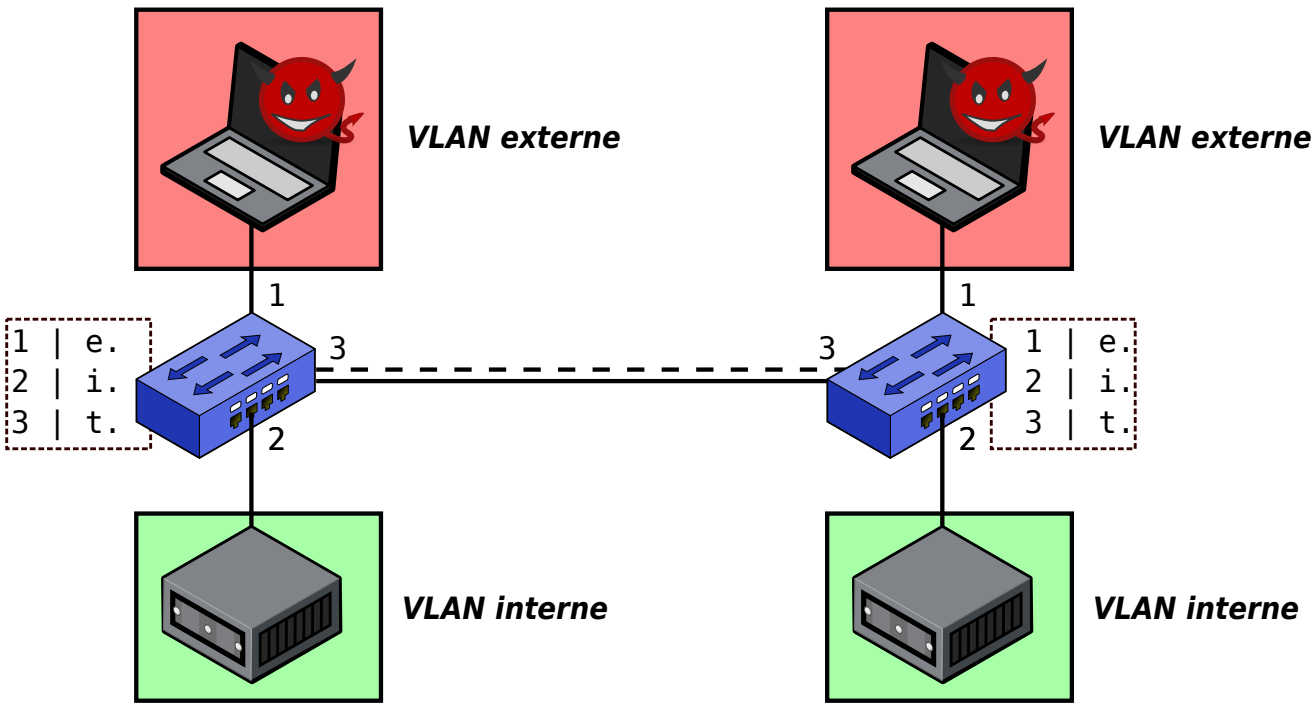
Attaques – *couche liaison de données (2) - VLANs 1/2*



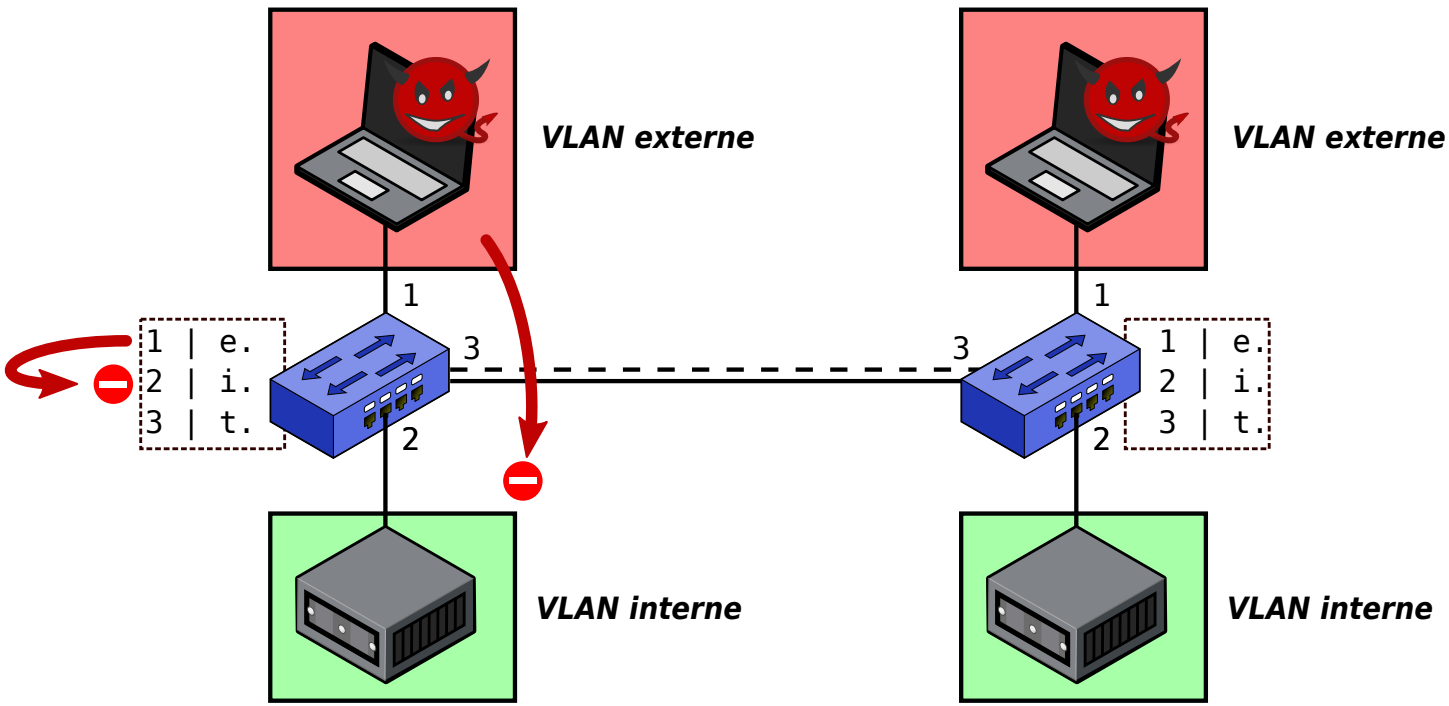
Attaques – *couche liaison de données (2) - VLANs 1/2*



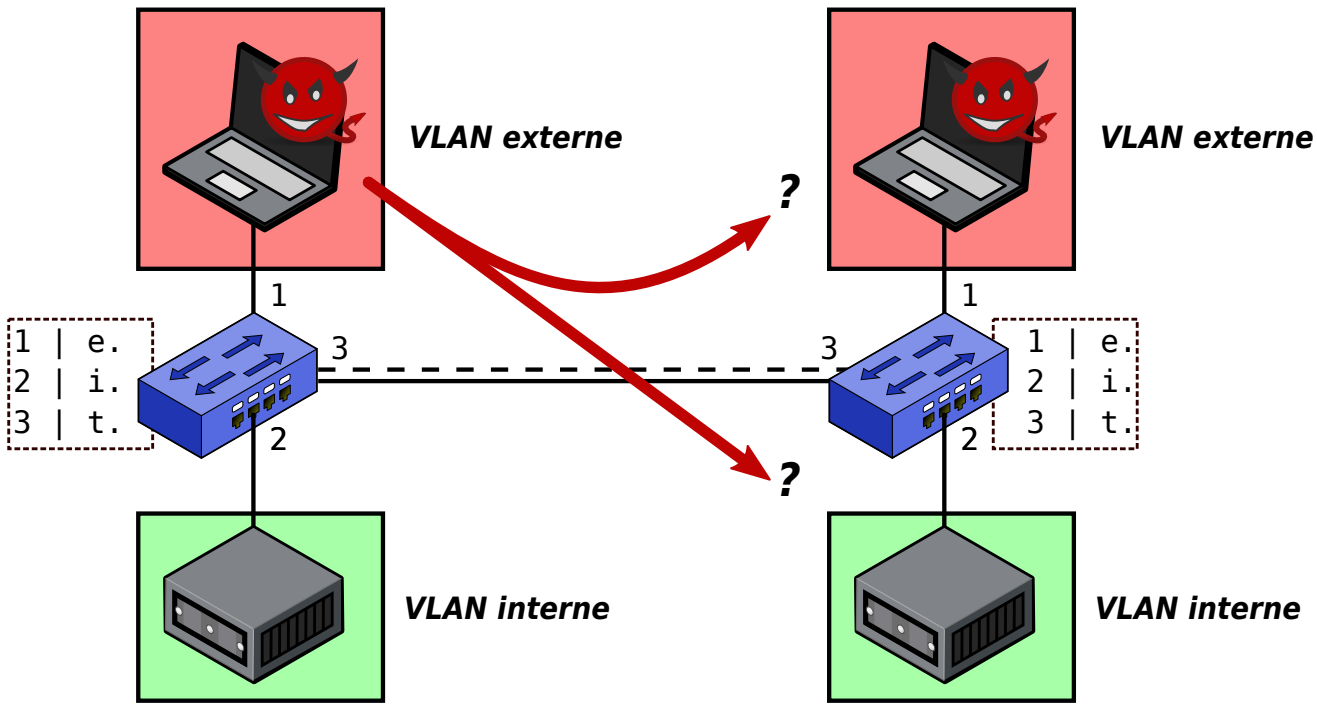
Attaques – *couche liaison de données (2) - VLANs 2/2*



Attaques – *couche liaison de données (2) - VLANs 2/2*

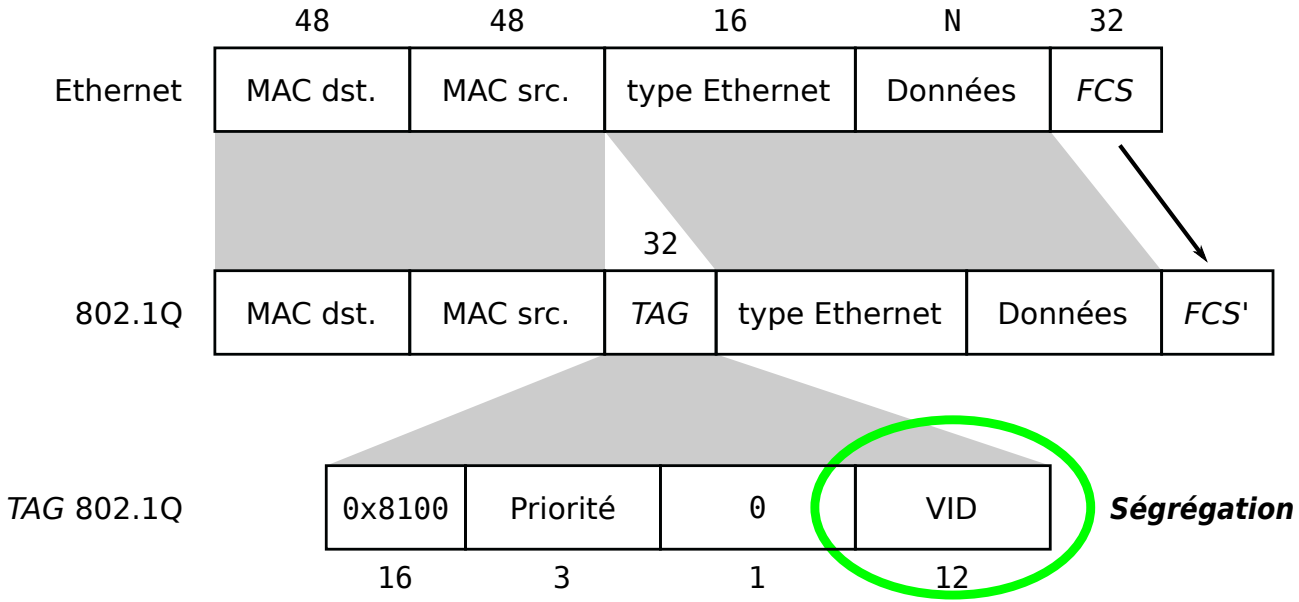


Attaques – *couche liaison de données (2) - VLANs 2/2*



Attaques – *couche liaison de données (2) - 802.1Q*

Multiplexage des VLANs basée sur l'insertion de tags



1. Ajout du TAG par le switch d'émission lors de la commutation sur une interface en mode *trunk*
2. Retrait du TAG et lecture du VLAN ID lors de la réception sur le switch de réception
3. Commutaiton de la trame sur le port si le VLAN ID sauvé correspond au VLAN ID du numéro de port cible

Attaques – *couche liaison de données (2) - saut de VLAN*

Saut de VLAN ou *VLAN hopping*

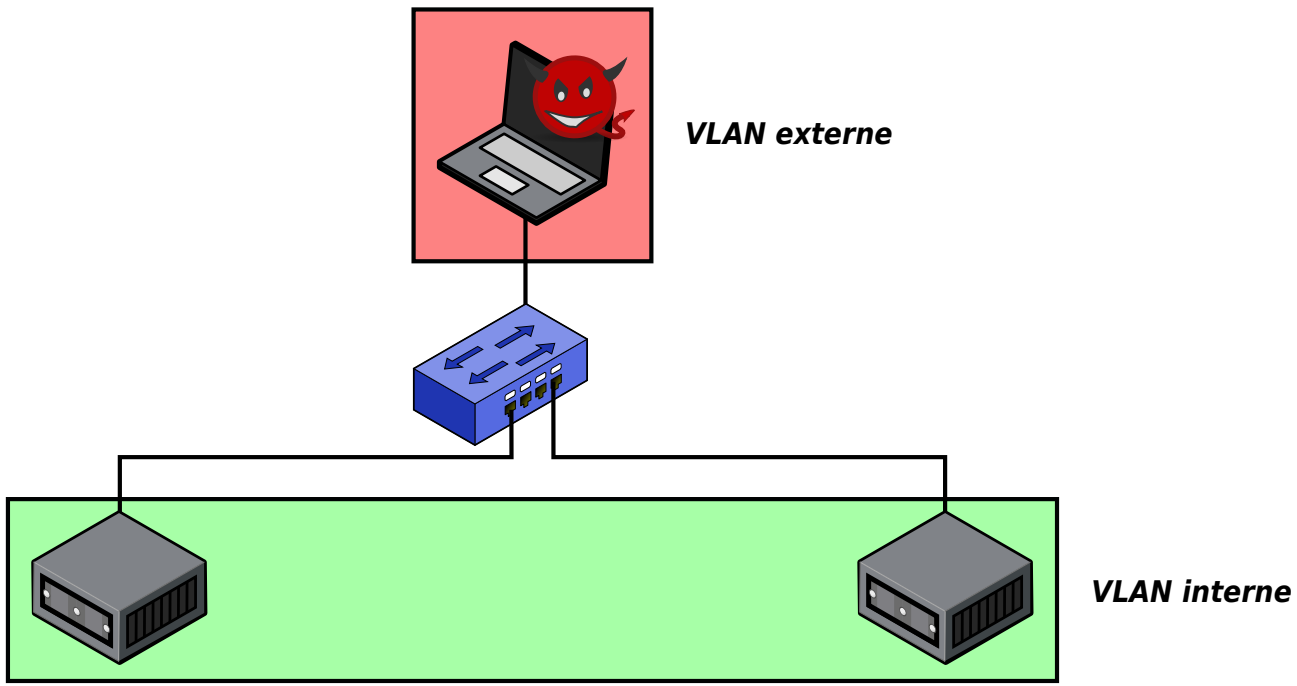
- Objectif : envoyer un message d'un sur un VLAN différent sur lequel on est pas connecté
- Deux méthodes :
  - déguisement en switch (*switch spoofing*) ou
  - *tag* double (*double tagging*)

Attaques – *couche liaison de données (2) - déguisement en switch (DTP)*

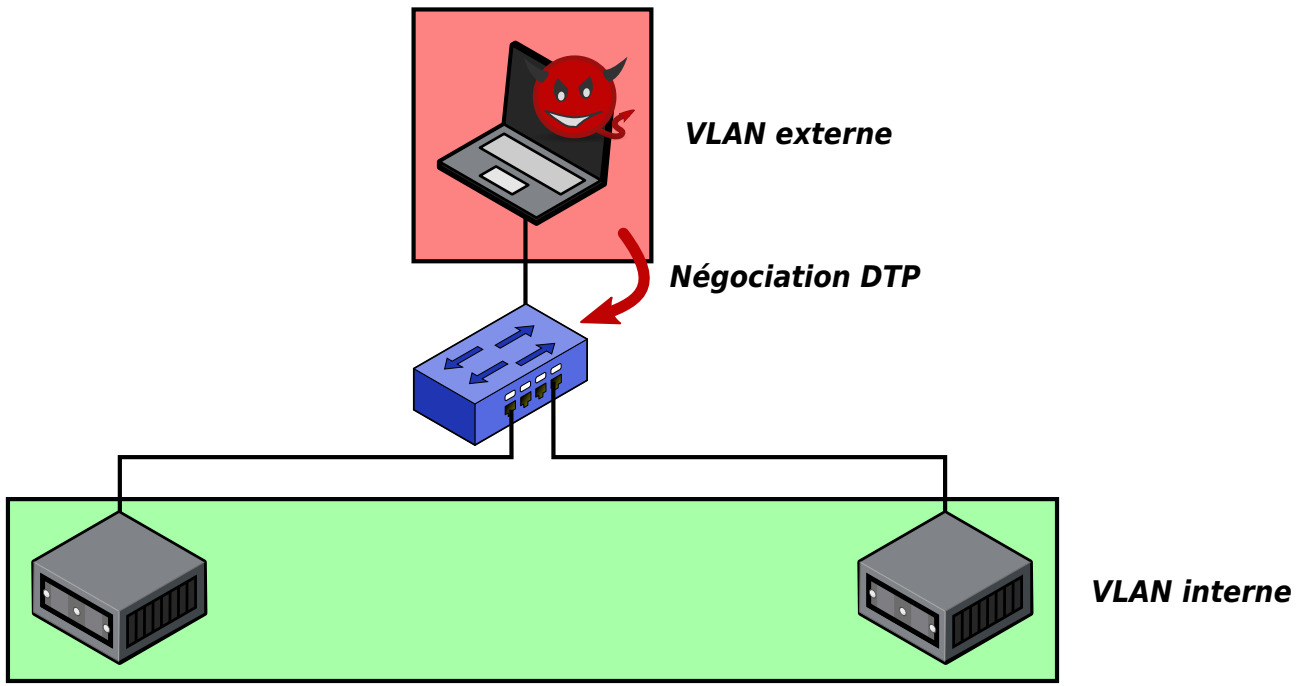
- Certain switchs proposent on protocole d'auto négociation du mode d'un port (normal ou *trunk*)
- C'est notamment le cas avec le protocole propriétaire *Dynamic Trunking Protocol* (DTP) de Cisco :

  `Switch(config-if)# switchport {no|}negotiate`
- **Problème de sécurité** : pas d'authentification de la classe de l'équipement
- **Méthode** :
  1. Se brancher sur le port d'un switch / VLAN non privilégié Cisco
  2. Négocier ce port en mode *trunk* avec DTP
  3. Émettre des trames marquées avec le VLAN cible

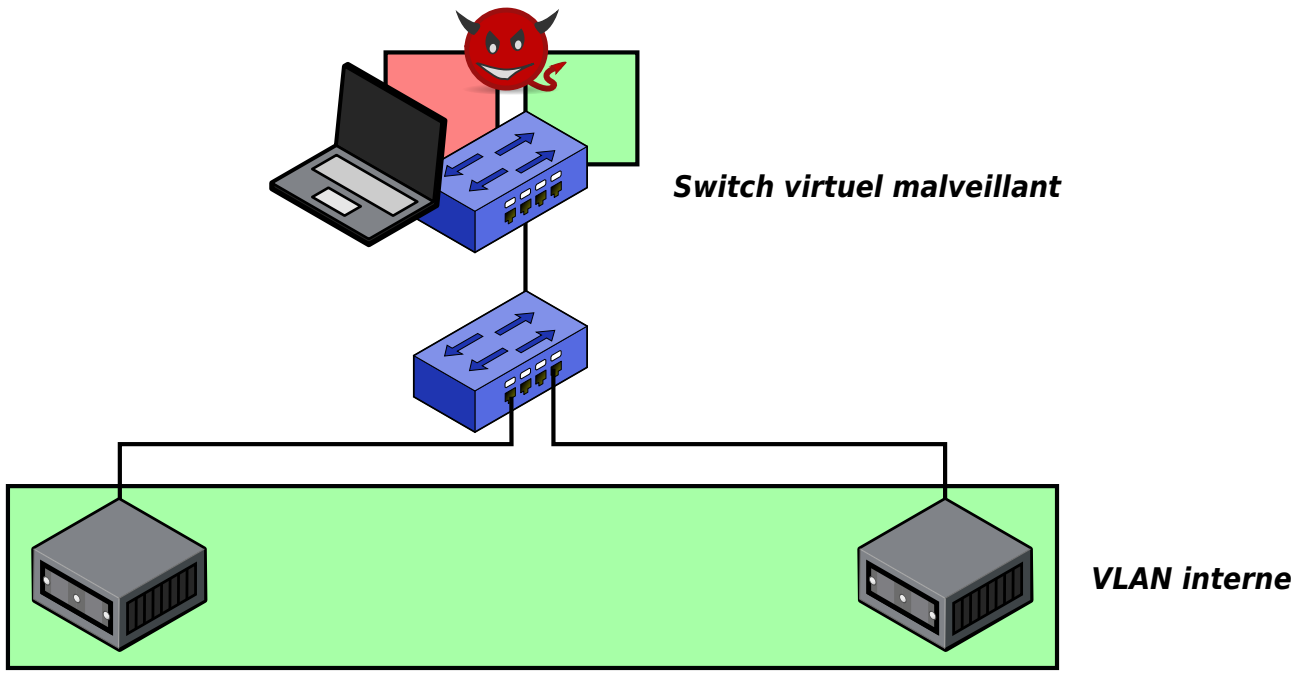
Attaques – *couche liaison de données (2) - déguisement en switch (DTP)*



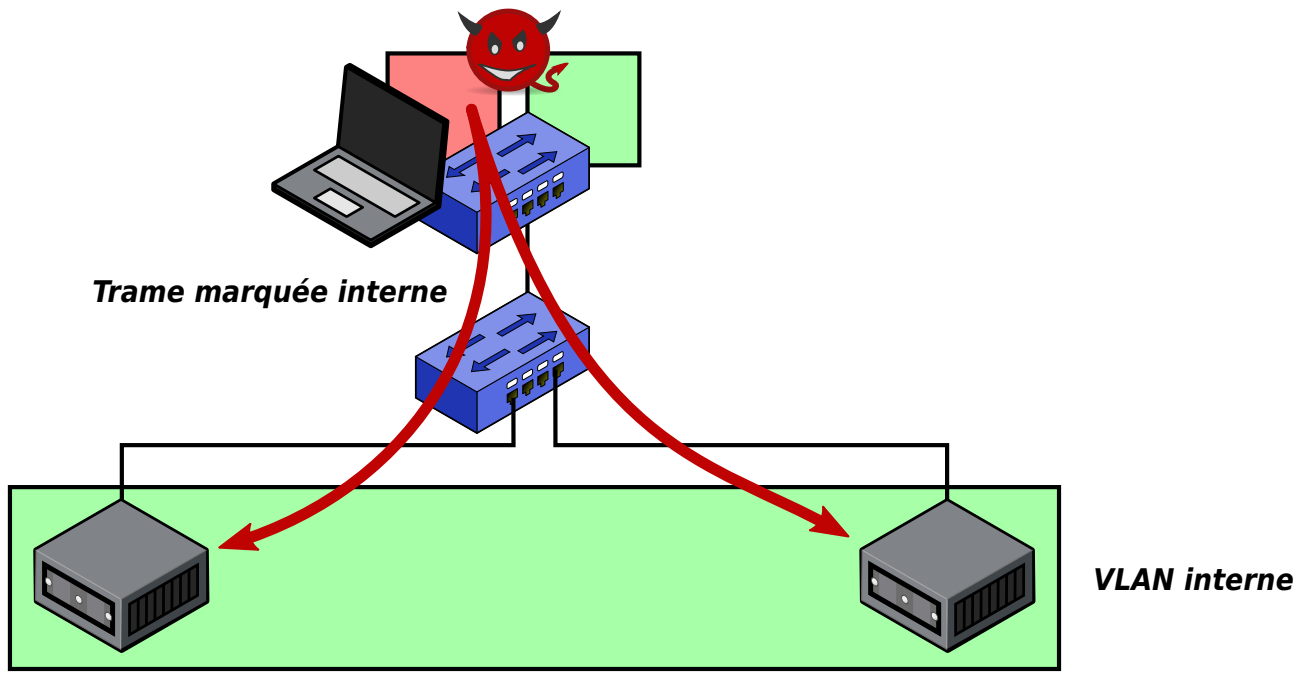
Attaques – *couche liaison de données (2) - déguisement en switch (DTP)*



Attaques – *couche liaison de données (2) - déguisement en switch (DTP)*

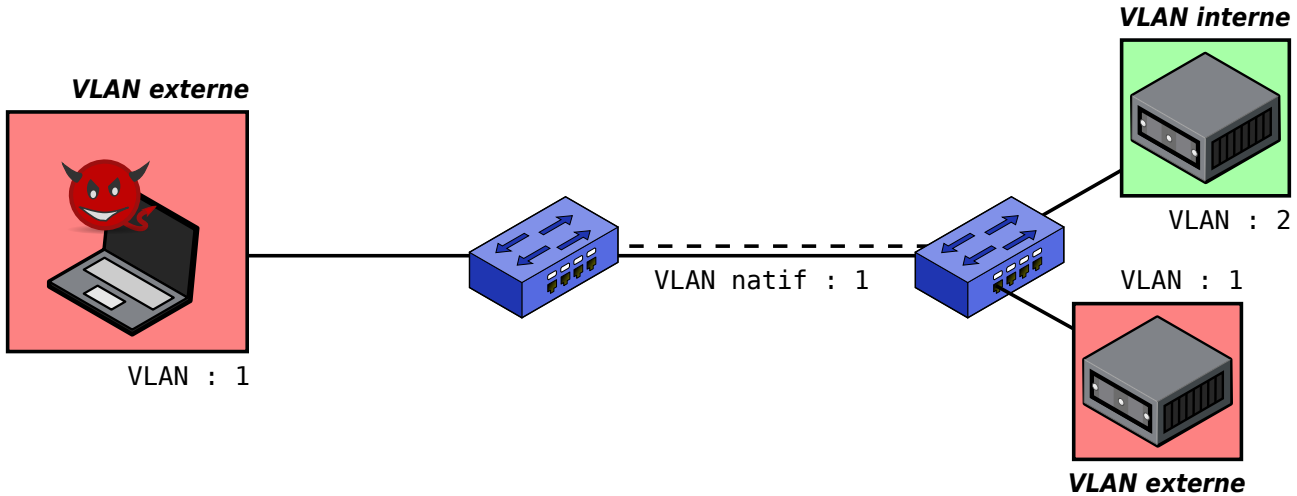


Attaques – *couche liaison de données (2) - déguisement en switch (DTP)*



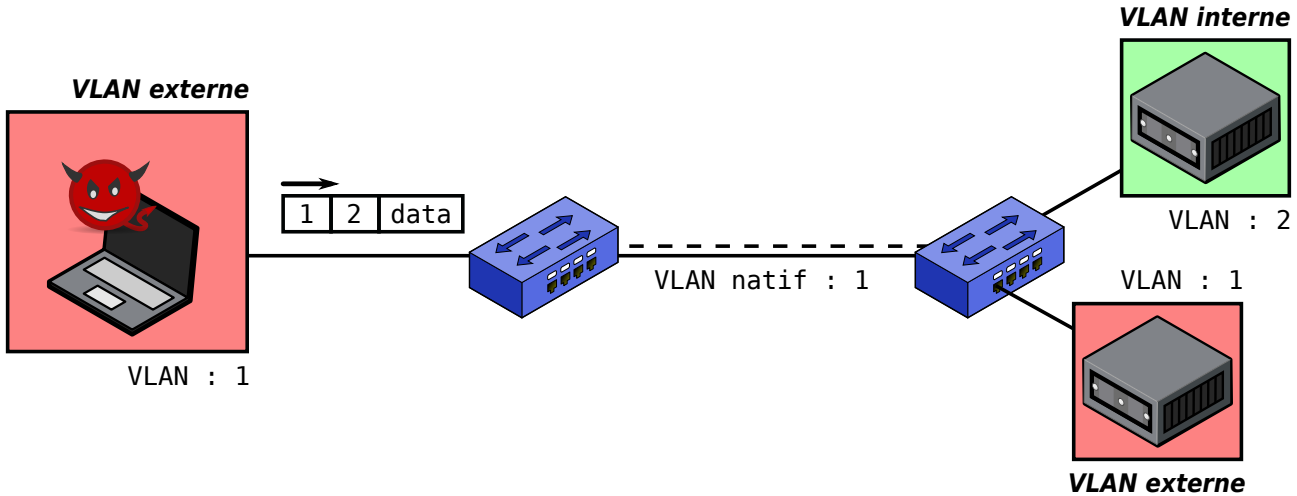
Attaques – *couche liaison de données (2) - tag double*

Prérequis de l'attaque : le VLAN extérieur doit être sur le VLAN natif des interfaces *trunk*



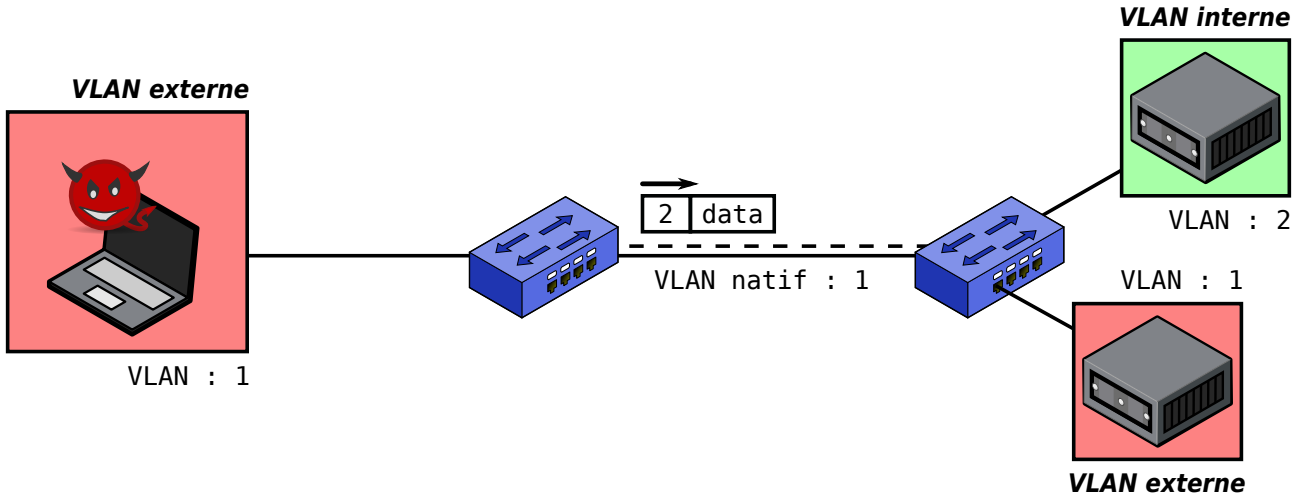
Attaques – *couche liaison de données (2) - tag double*

Prérequis de l'attaque : le VLAN extérieur doit être sur le VLAN natif des interfaces *trunk*



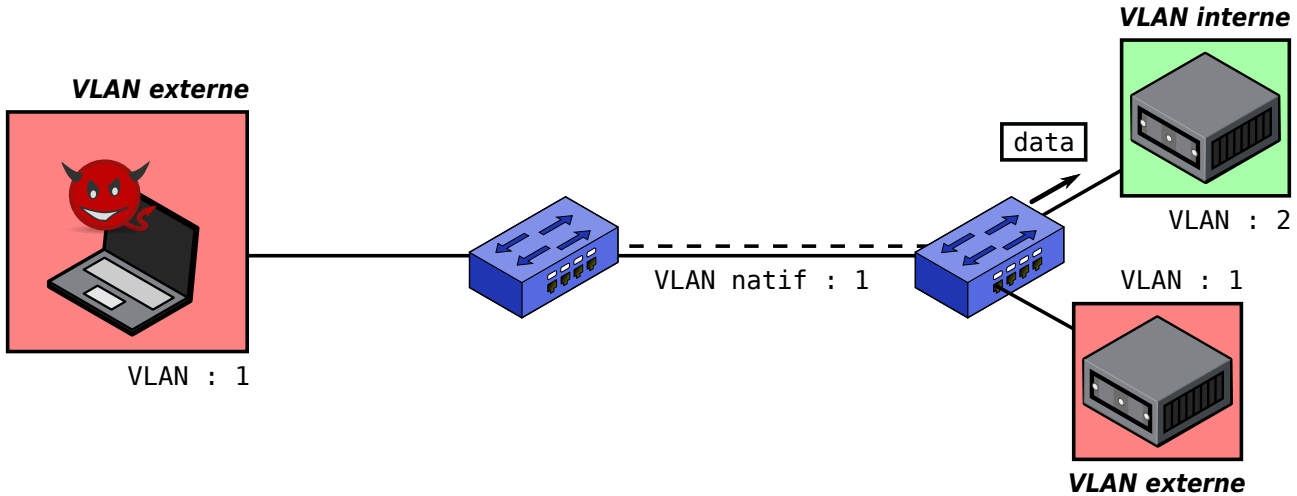
Attaques – *couche liaison de données (2) - tag double*

Prérequis de l'attaque : le VLAN extérieur doit être sur le VLAN natif des interfaces *trunk*



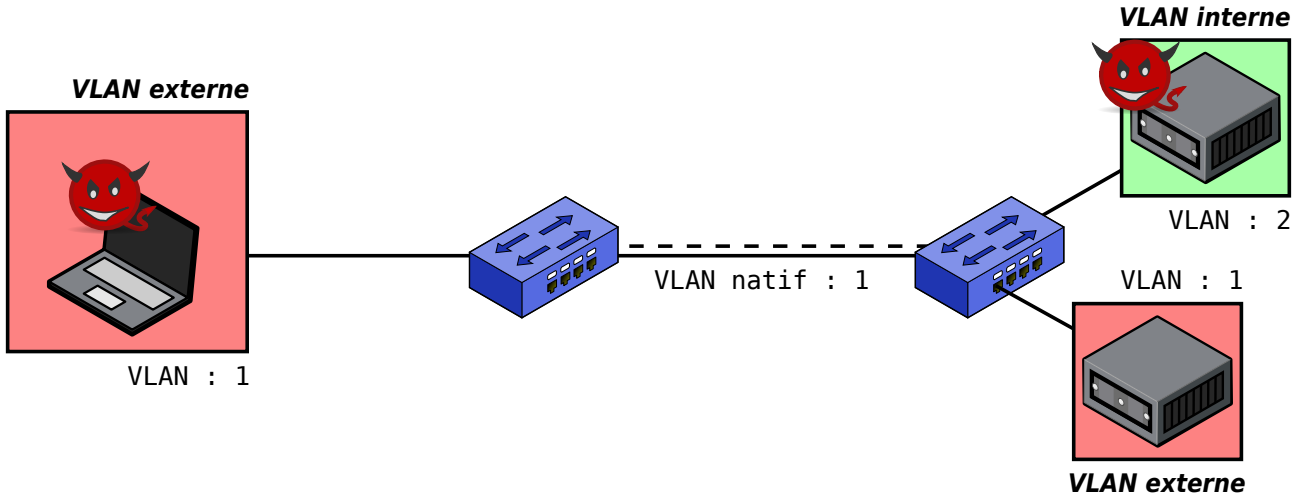
Attaques – *couche liaison de données (2) - tag double*

Prérequis de l'attaque : le VLAN extérieur doit être sur le VLAN natif des interfaces *trunk*



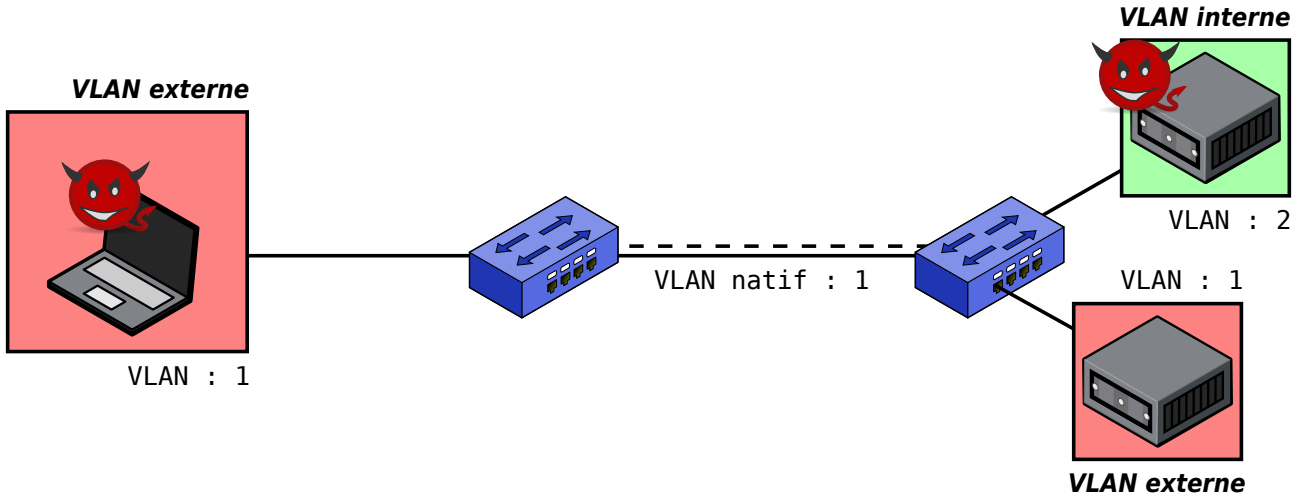
Attaques – *couche liaison de données (2) - tag double*

Prérequis de l'attaque : le VLAN extérieur doit être sur le VLAN natif des interfaces *trunk*



Attaques – *couche liaison de données (2) - tag double*

Prérequis de l'attaque : le VLAN extérieur doit être sur le VLAN natif des interfaces *trunk*



Contremesure simple : ne jamais placer un port d'accès sur le VLAN natif de ports *trunk*

Protocole Cisco Discovery Protocol (CDP)

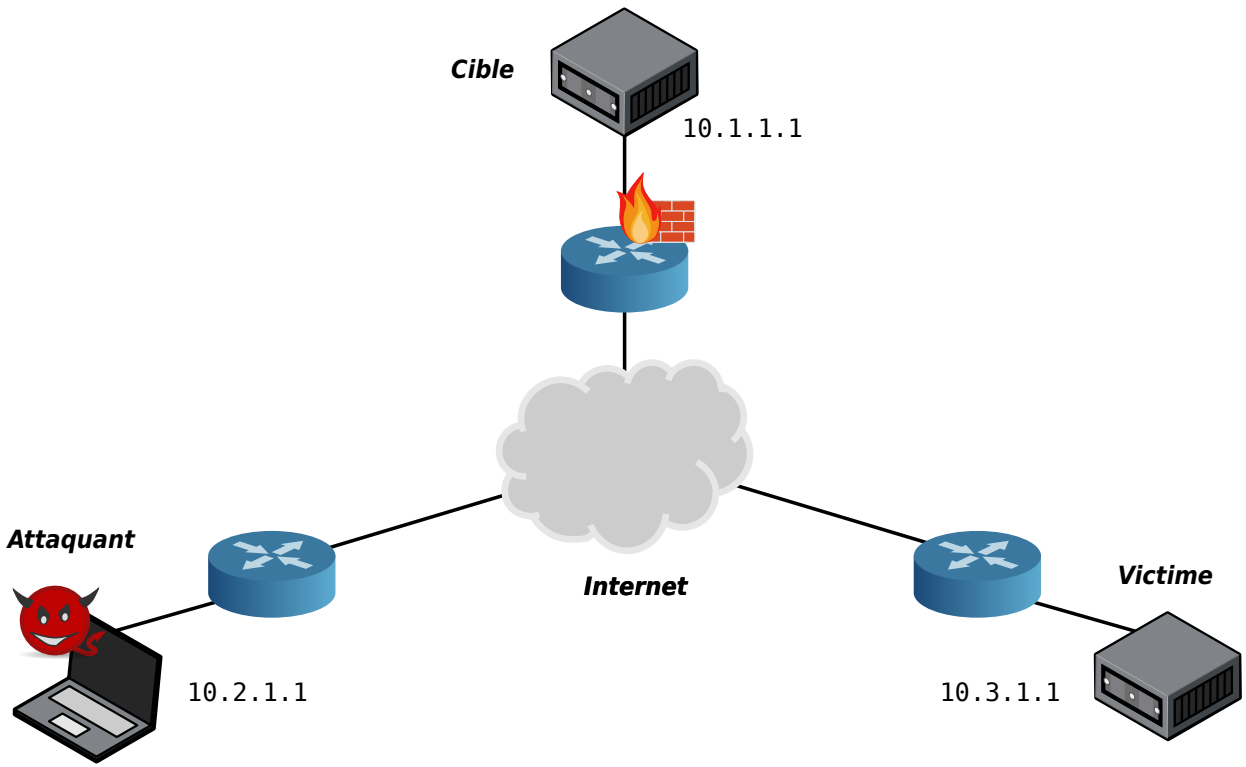
- Équivalent IEEE Link Layer Discovery Protocol (LLDP)
- Découvrir les équipement d'un même domaine de diffusion
- Peut substituer aux protocoles de routage (eq. IPv6 NDP)
- **Injection de messages** → modification de la vision du réseau
- **Innondation de messages** → denis de service

Protocole de l'arbre couvrant de poids minimal (STP)

- Forcer la réélection du switch racine → denis de service

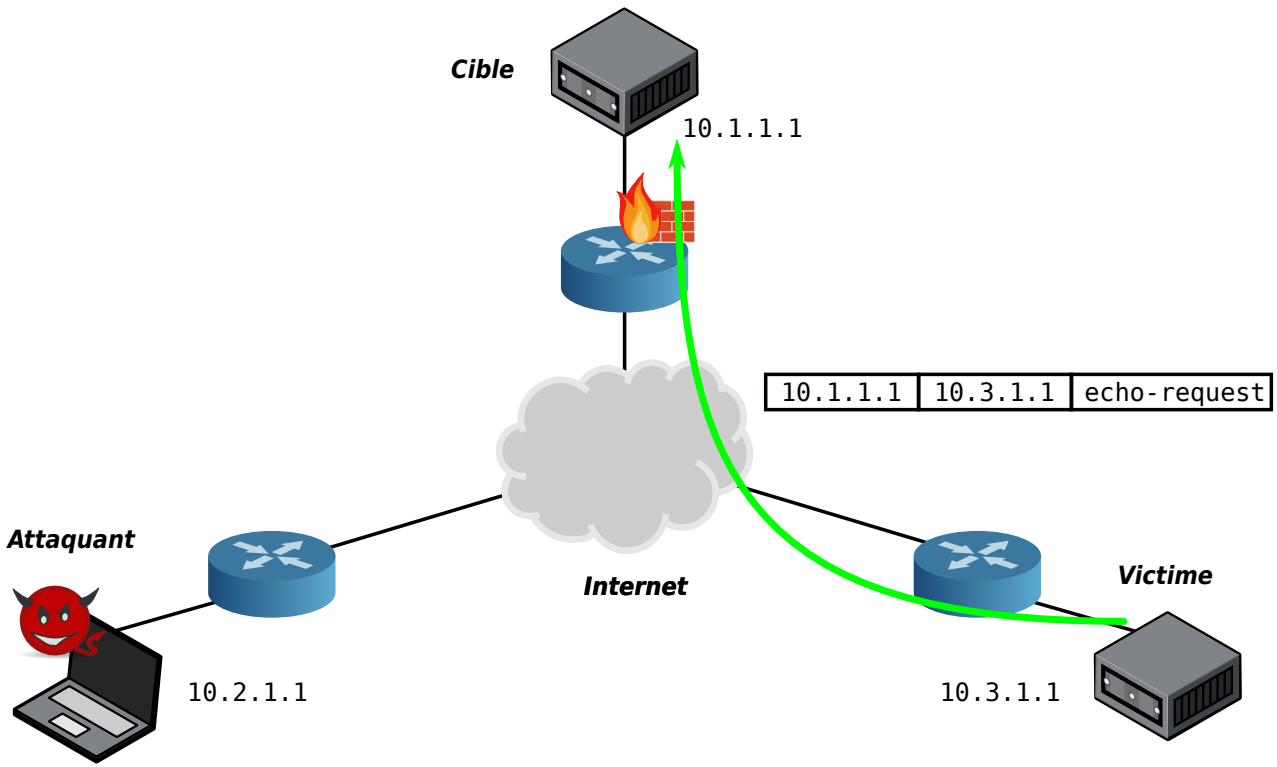
Attaques – *couche réseau (3) - déguisement IP*

Déguisement IP (*IP spoofing*) : emission de traffic avec une IP qui n'appartient pas à l'attaquant



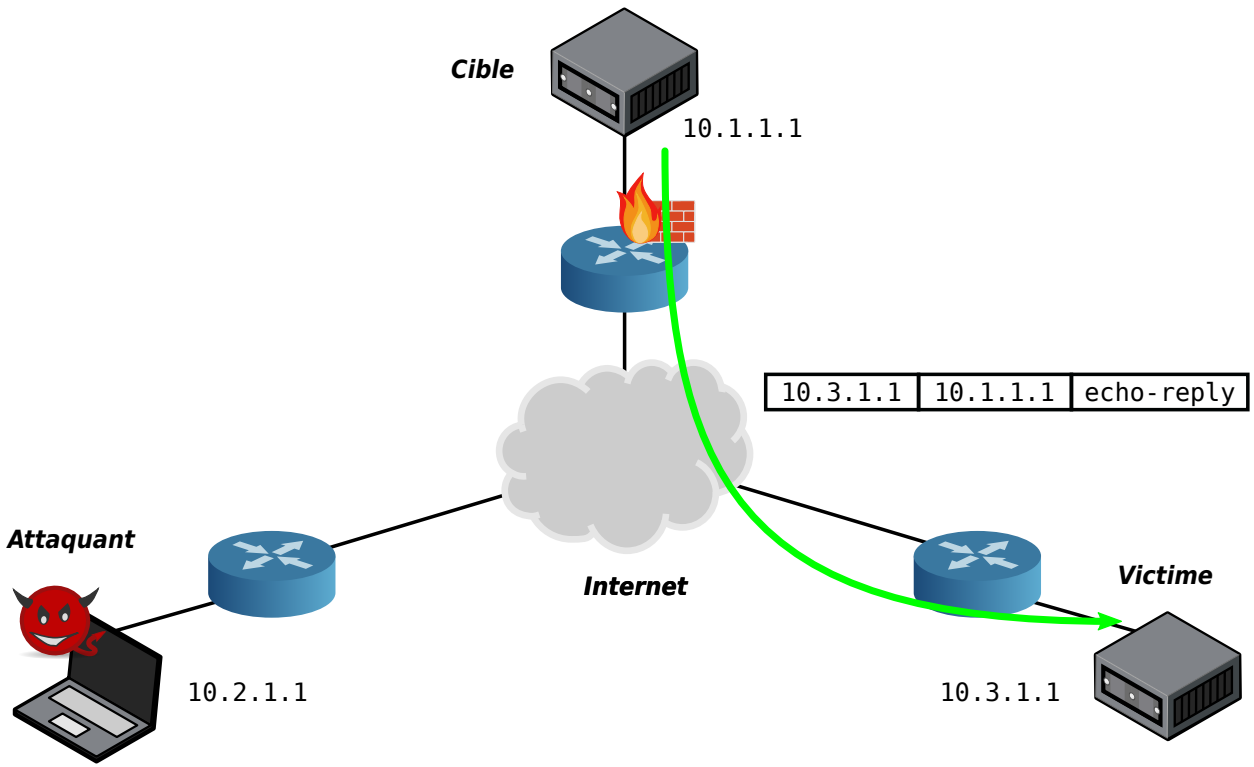
Attaques – *couche réseau (3) - déguisement IP*

Déguisement IP (*IP spoofing*) : emission de traffic avec une IP qui n'appartient pas à l'attaquant



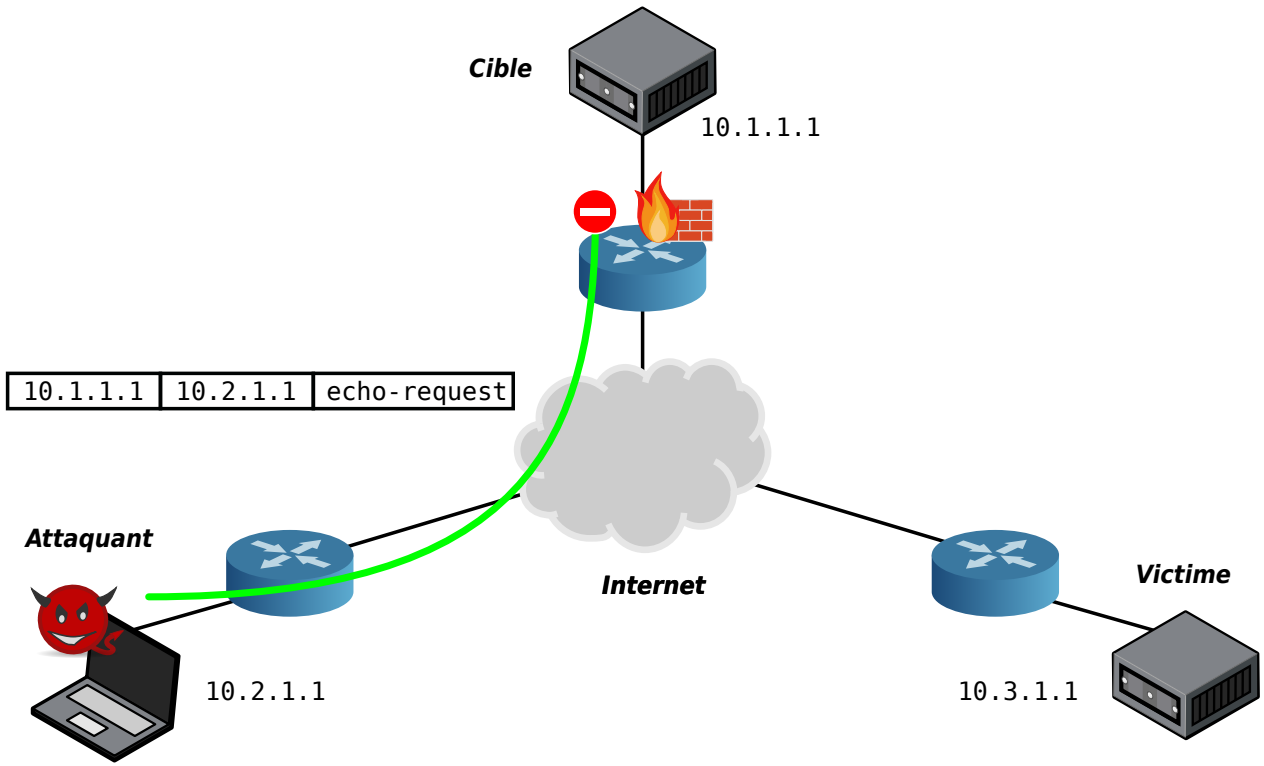
Attaques – *couche réseau (3) - déguisement IP*

Déguisement IP (*IP spoofing*) : emission de traffic avec une IP qui n'appartient pas à l'attaquant



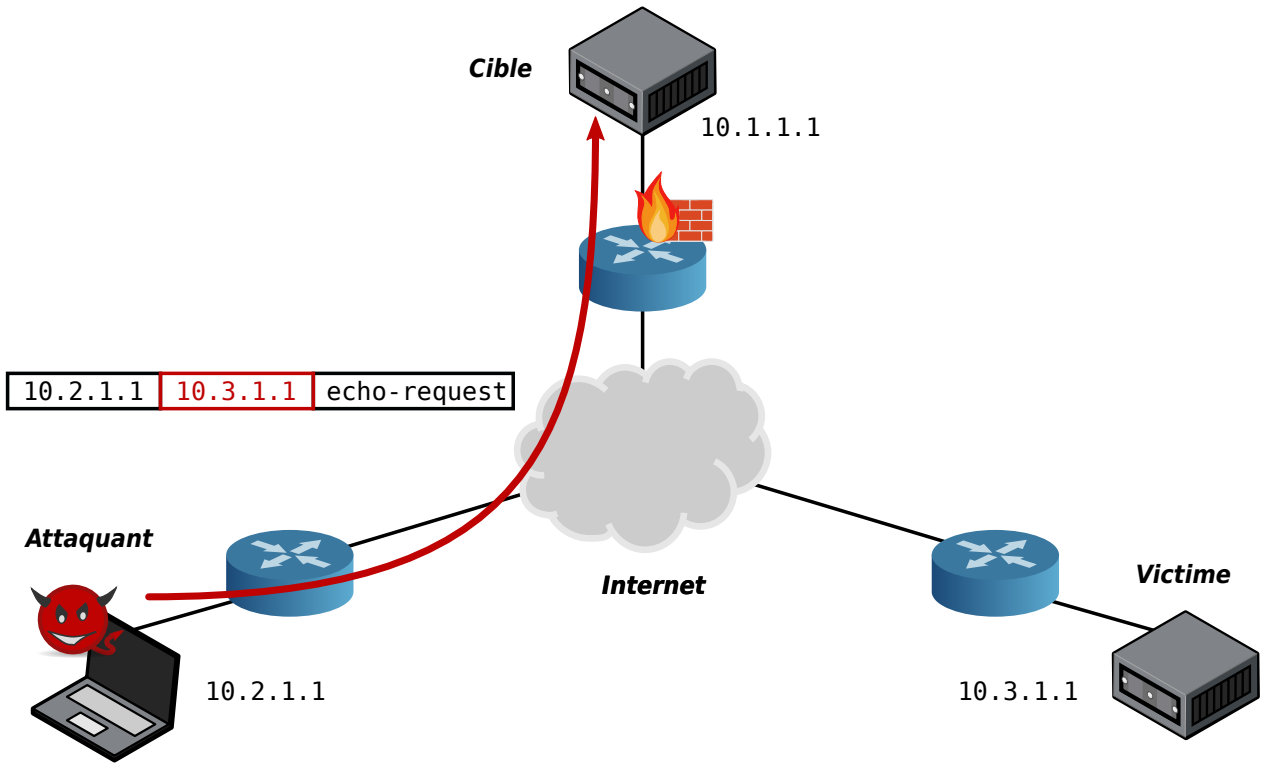
Attaques – *couche réseau (3) - déguisement IP*

Déguisement IP (*IP spoofing*) : emission de traffic avec une IP qui n'appartient pas à l'attaquant



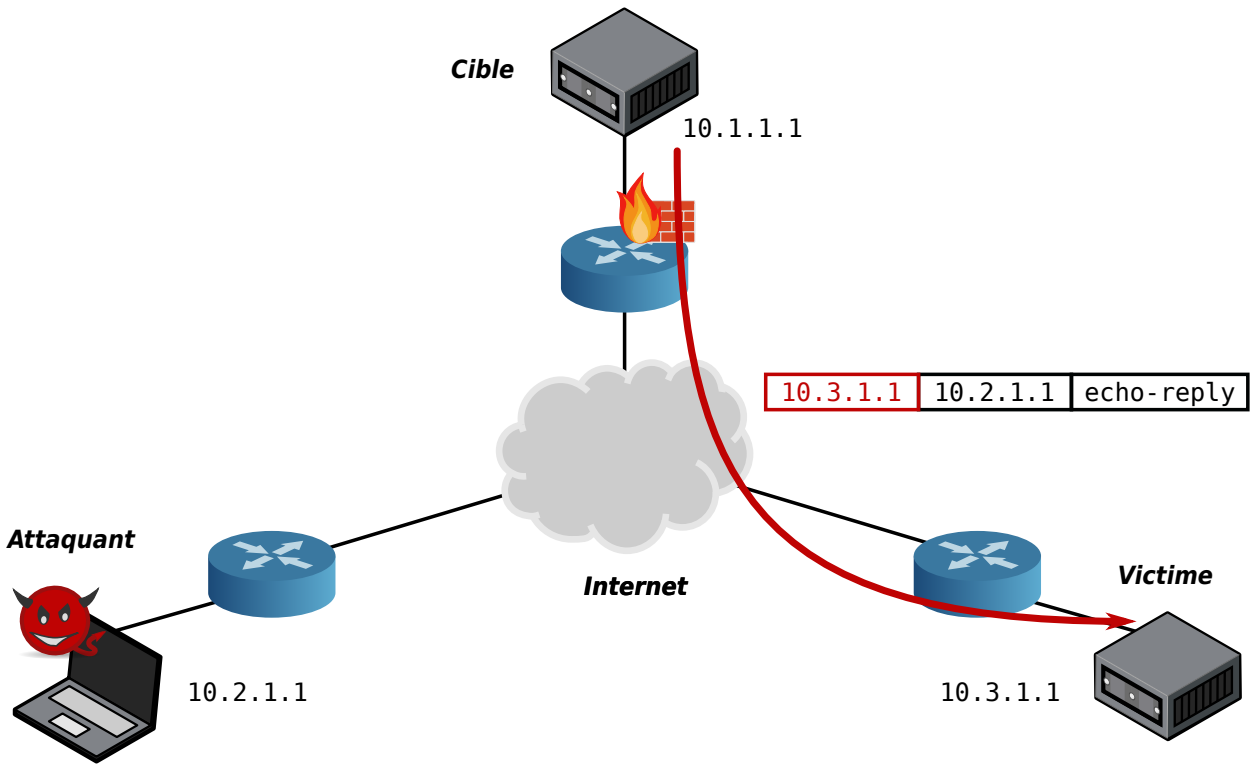
Attaques – *couche réseau (3) - déguisement IP*

Déguisement IP (*IP spoofing*) : emission de traffic avec une IP qui n'appartient pas à l'attaquant



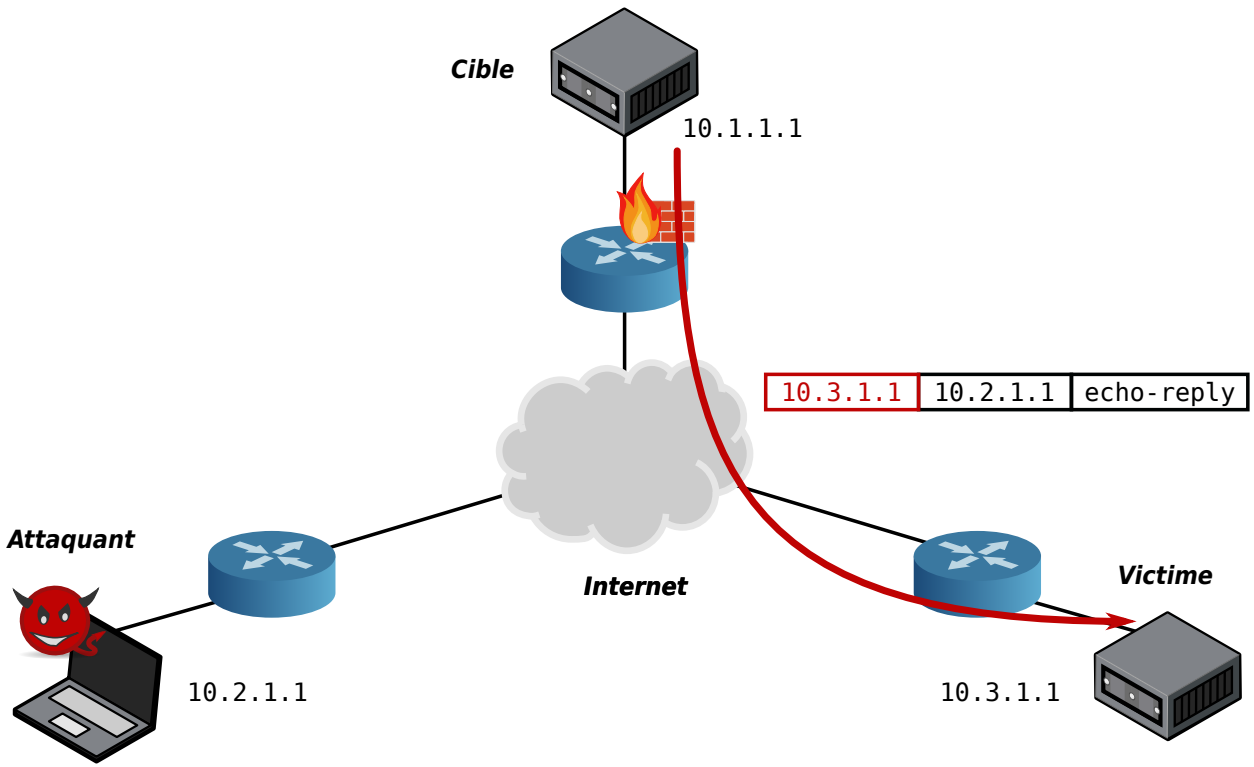
Attaques – *couche réseau (3) - déguisement IP*

Déguisement IP (*IP spoofing*) : emission de traffic avec une IP qui n'appartient pas à l'attaquant



Attaques – *couche réseau (3) - déguisement IP*

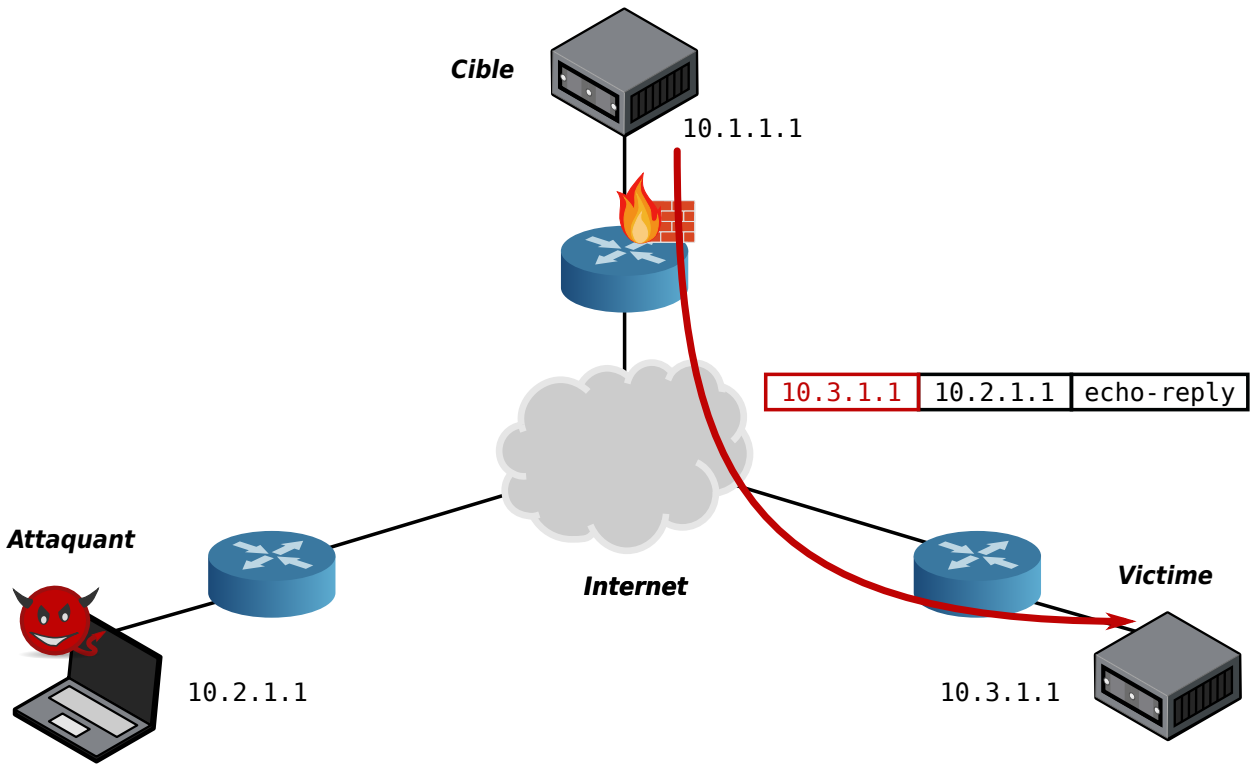
Déguisement IP (*IP spoofing*) : emission de traffic avec une IP qui n'appartient pas à l'attaquant



→ Contournement de filtrage réseau

Attaques – *couche réseau (3) - déguisement IP*

Déguisement IP (*IP spoofing*) : emission de traffic avec une IP qui n'appartient pas à l'attaquant



→ Contournement de filtrage réseau

→ Perte systématique de la réponse

Attaques – *couche réseau (3) - déguisement IP par source routing*

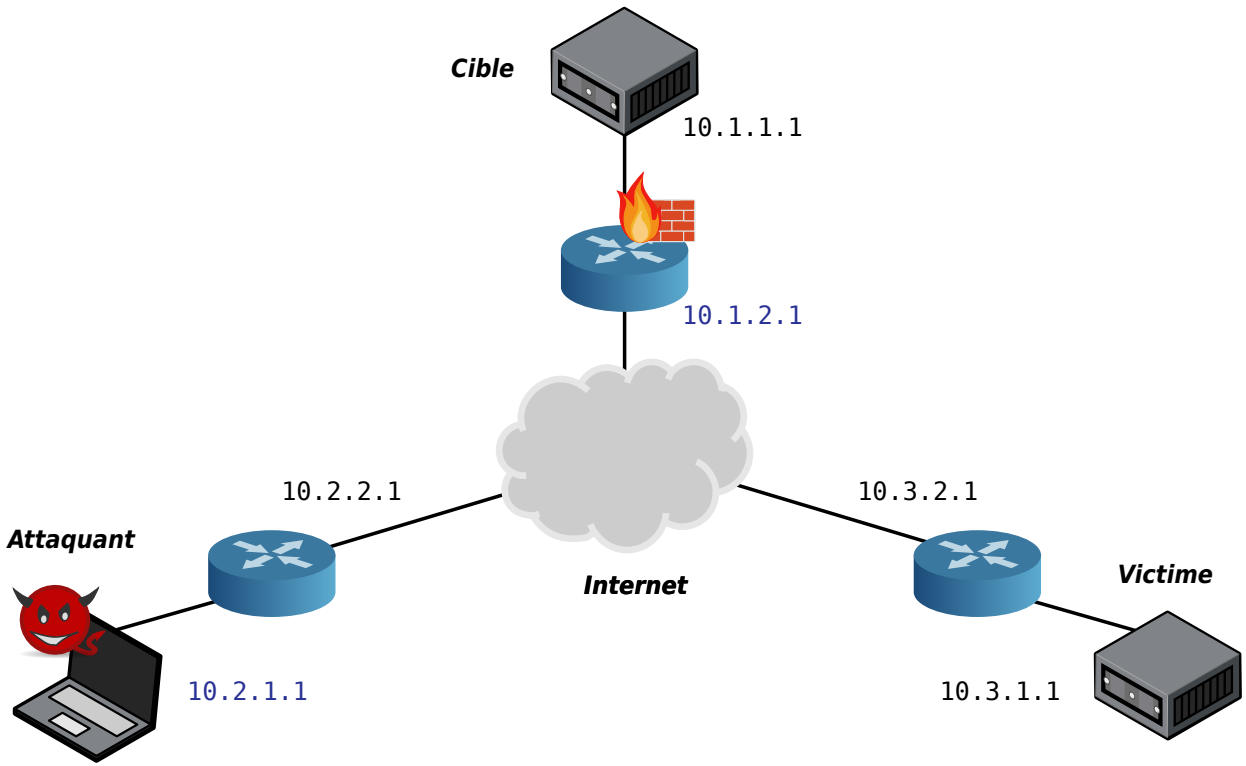
Attaque historique : *source routing* est désactivé par RFC

- *Source routing* : impose un routage strict d'un paquet
- *Loose source routing* : sous-ensemble des routeurs traversés

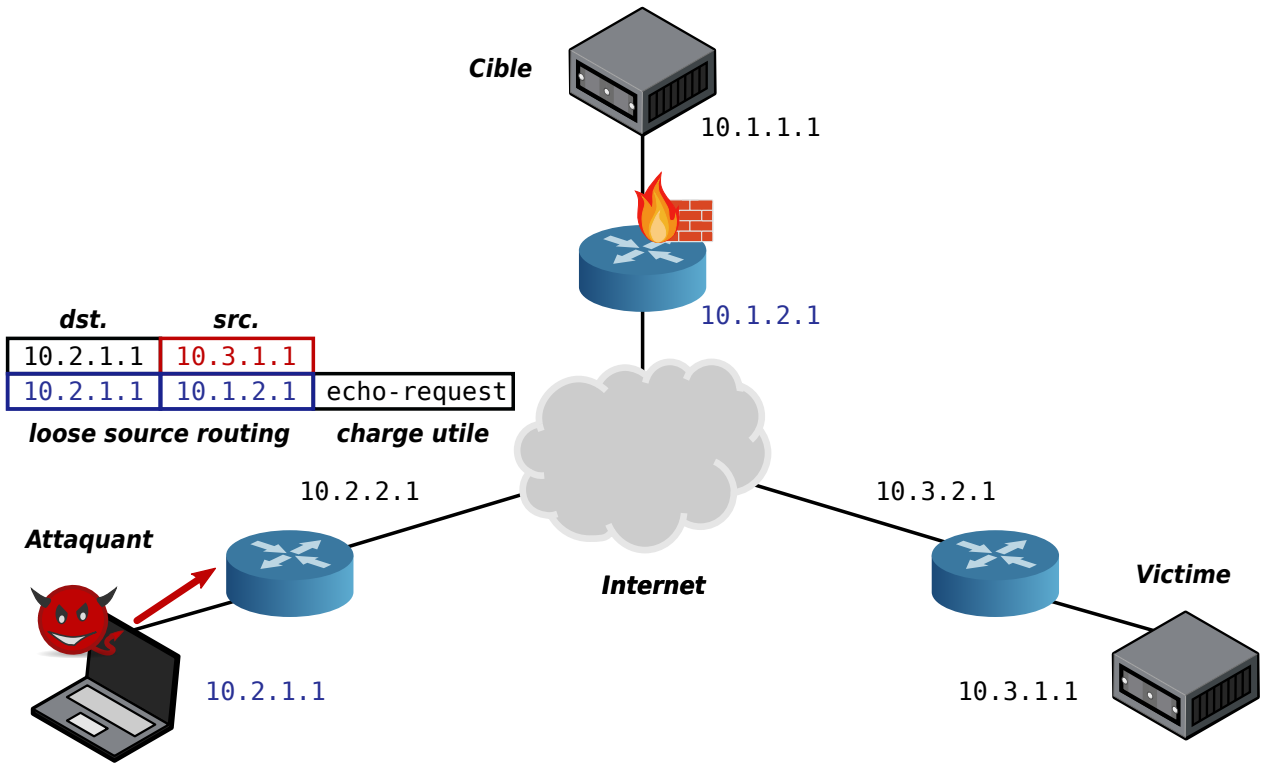
Attaque : se déguiser en victime aux yeux de cible

- L'attaquant forge un paquet avec l'adresse IP source de la victime
- L'attaquant active le (*loose*) *source routing*
- Il se place dans le liste des routeurs
- Il transmet le paquet qui va suivre le rest de la route décrite
- Cible répond avec la même route (*loose*) *source routing*
- Attaquant intercepte le paquet avant victime

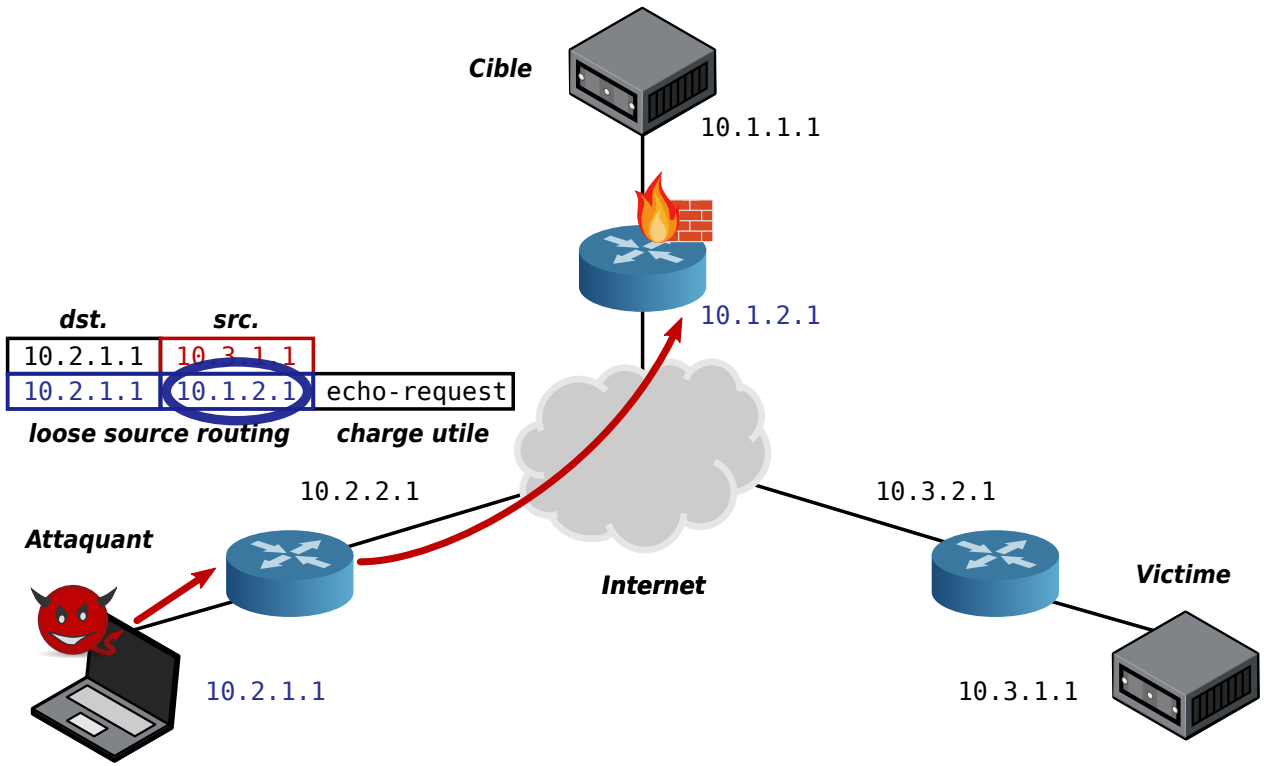
Attaques – *couche réseau (3) - déguisement IP par source routing*



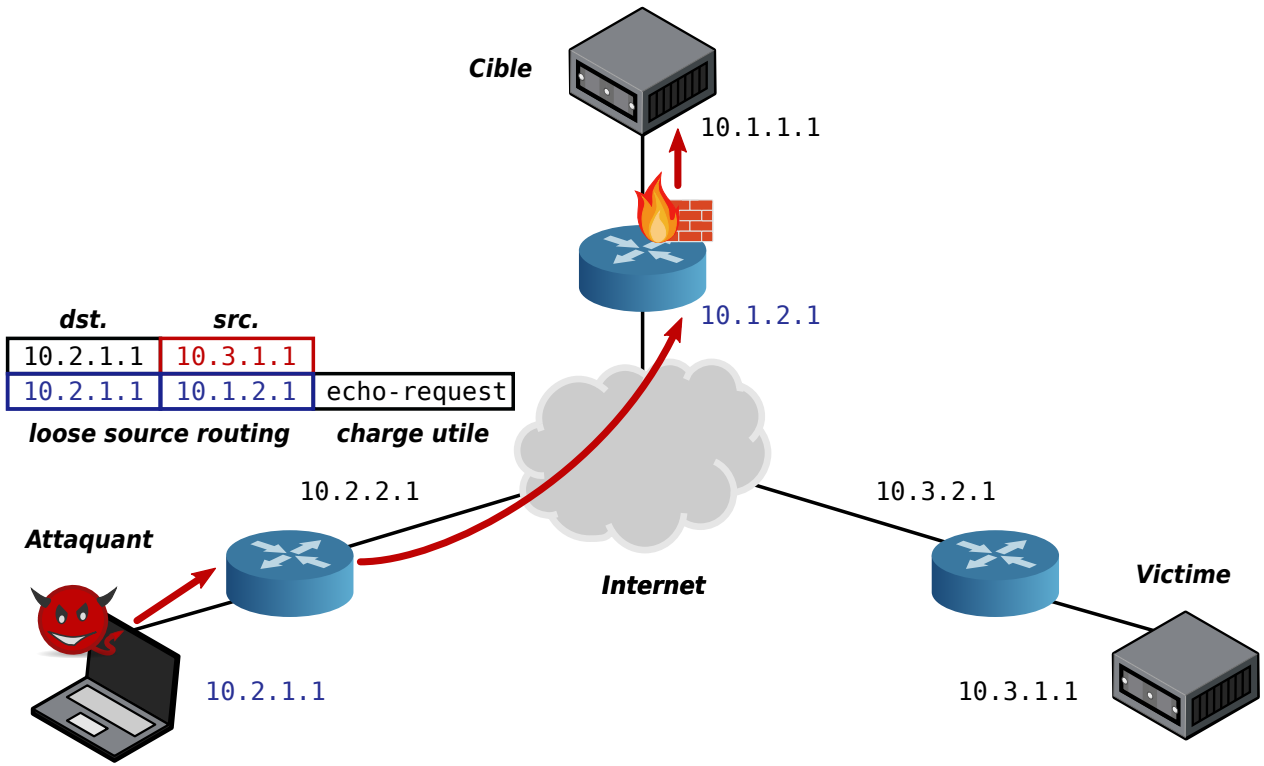
Attaques – *couche réseau (3) - déguisement IP par source routing*



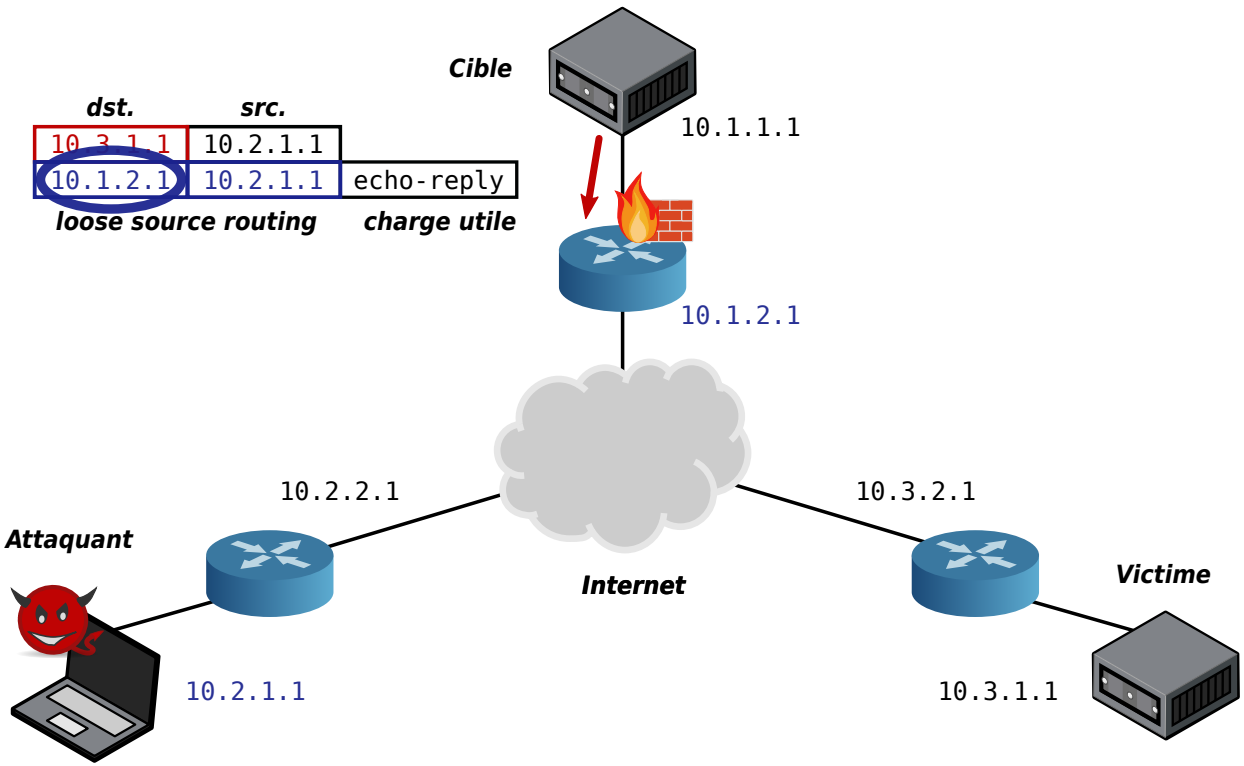
Attaques – *couche réseau (3) - déguisement IP par source routing*



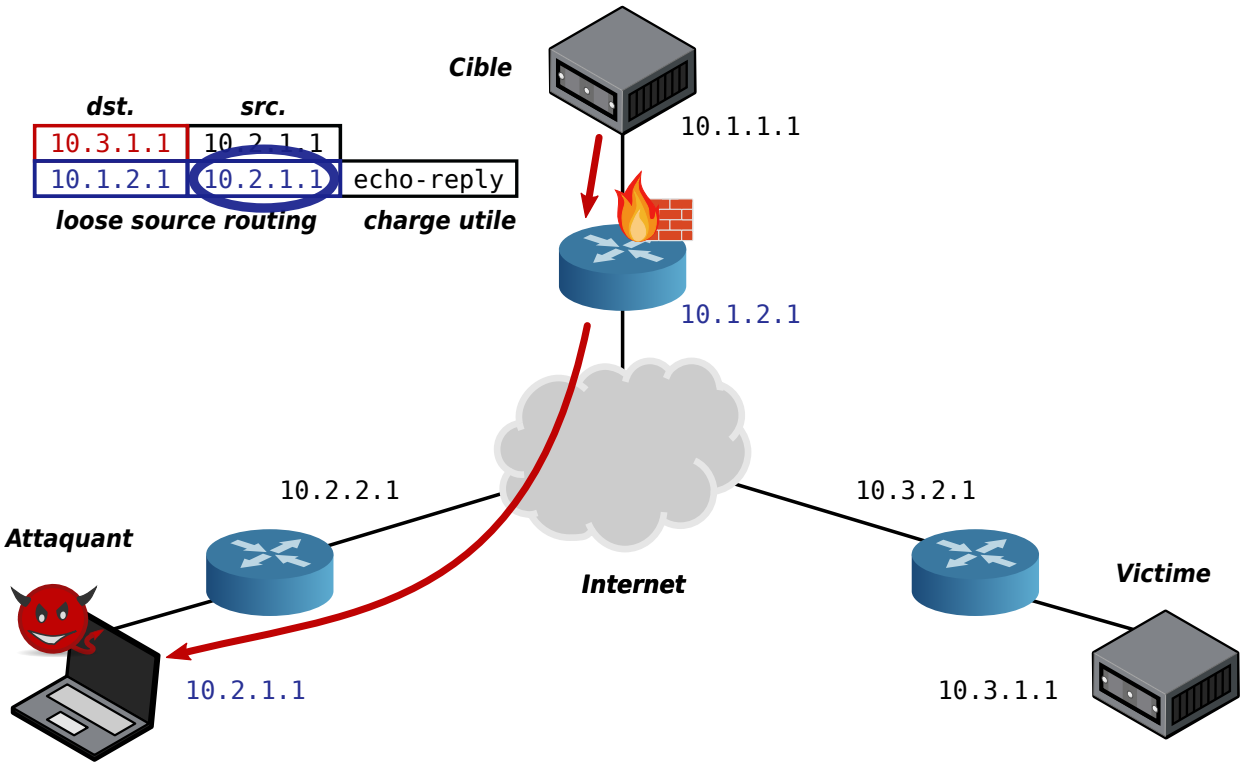
Attaques – *couche réseau (3) - déguisement IP par source routing*



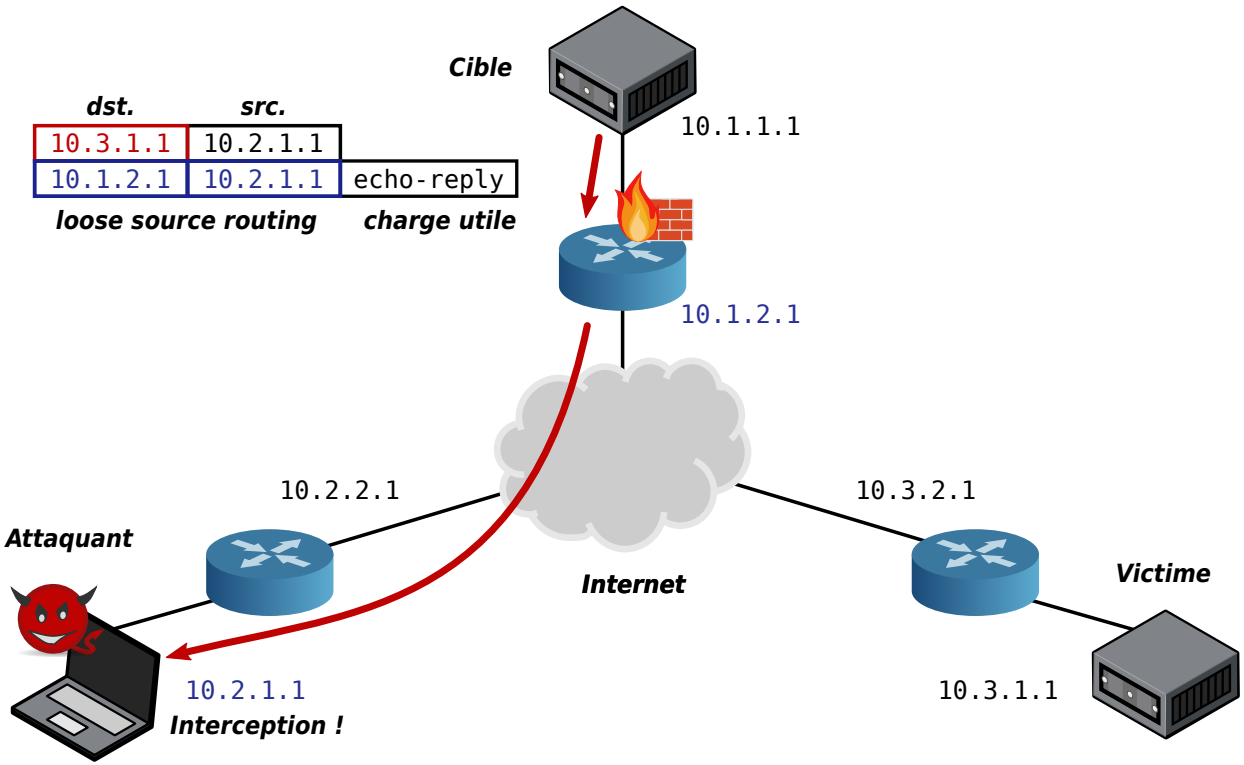
Attaques – *couche réseau (3) - déguisement IP par source routing*



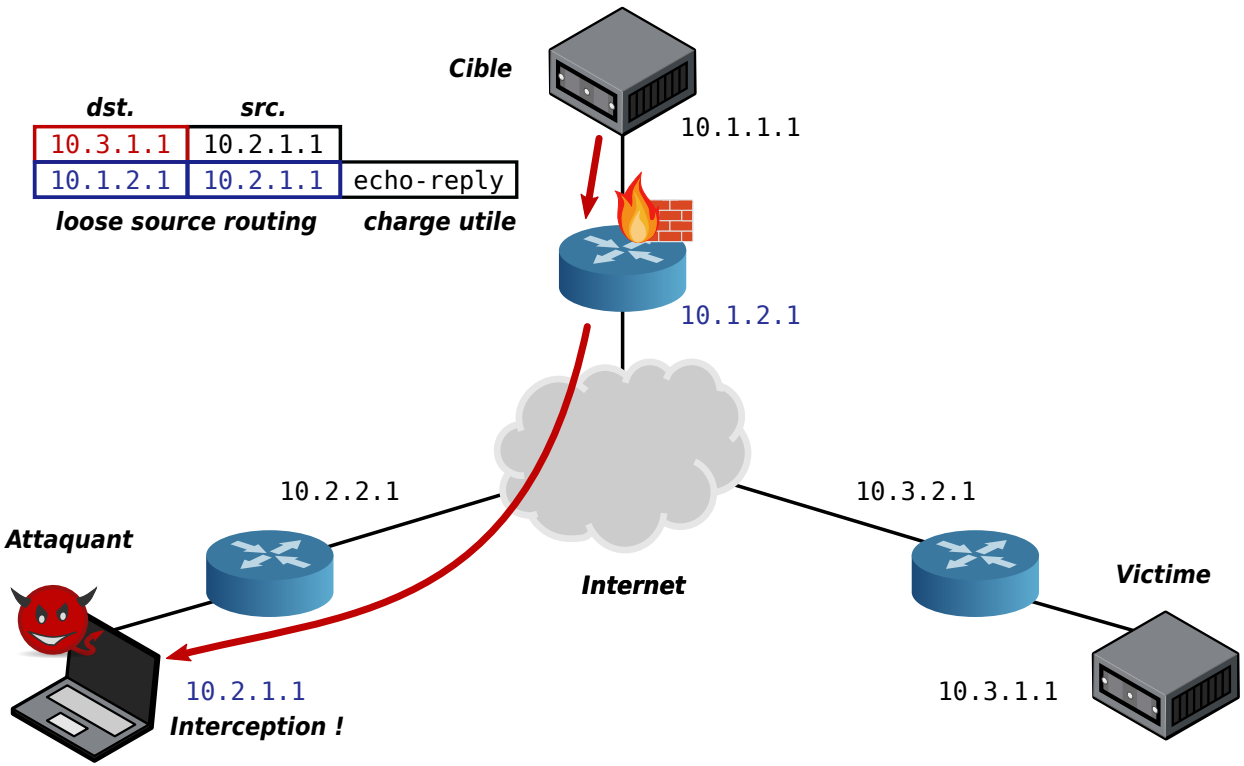
Attaques – *couche réseau (3) - déguisement IP par source routing*



Attaques – *couche réseau (3) - déguisement IP par source routing*

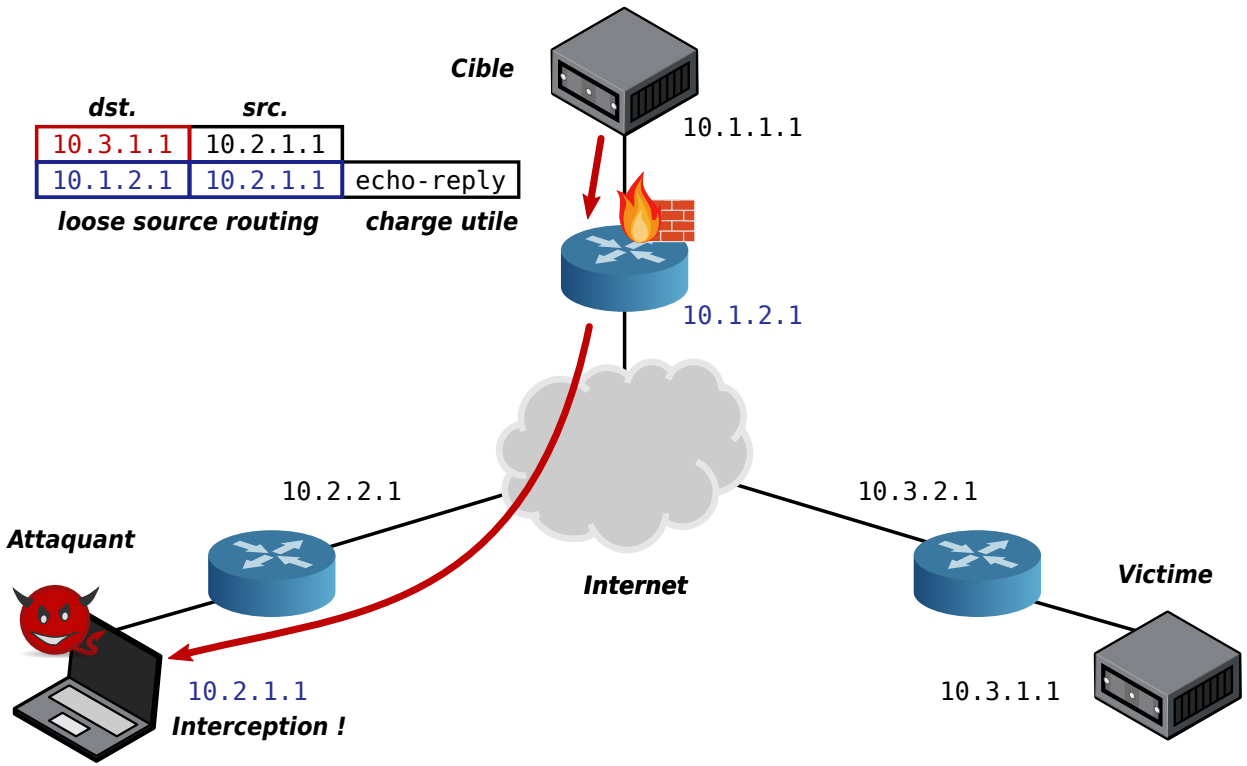


Attaques – *couche réseau (3) - déguisement IP par source routing*



→ Contournement de filtrage réseau

Attaques – *couche réseau (3) - déguisement IP par source routing*



→ Contournement de filtrage réseau

→ Homme dans le milieu possible !

Attaques – *couche réseau (3) - protocoles de routage*

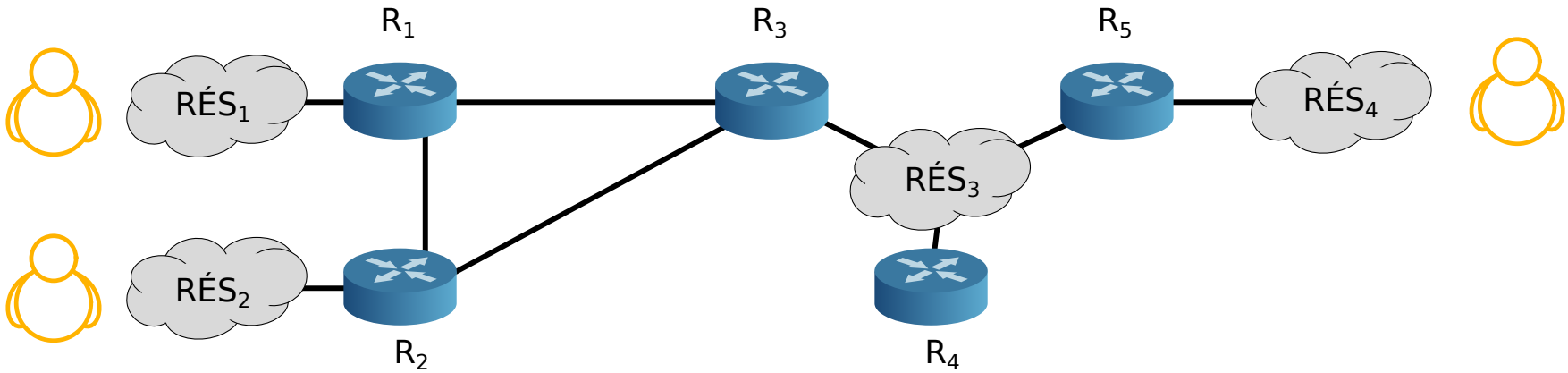
Services :

- Constituer la base de connaissance de la topologie réseau
- Calculer les routes nécessaires au relayage des paquets
- Support du dynamise de la topologie du réseau

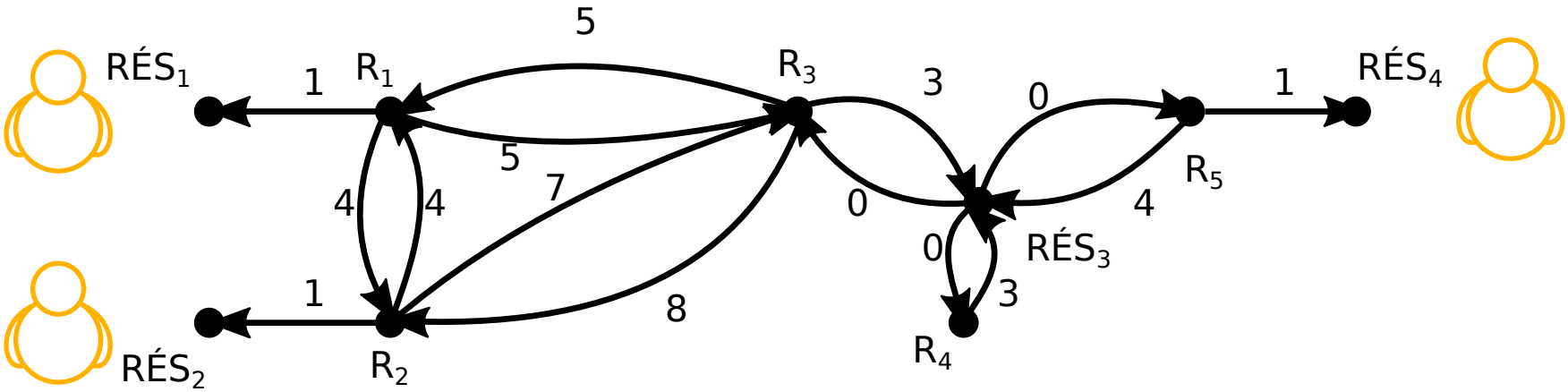
Types de protocoles de routages :

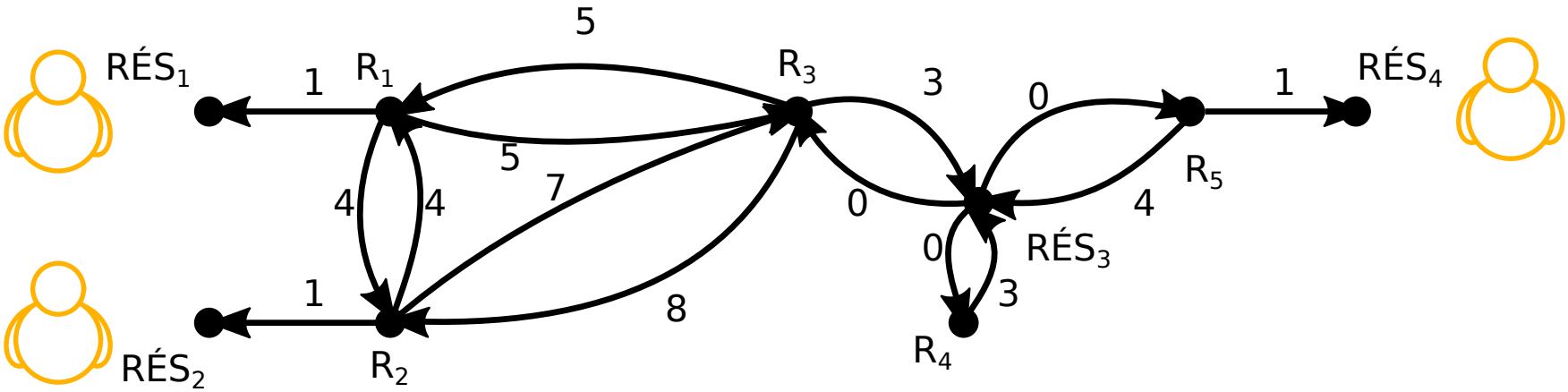
- Dépend du type de réseau à supporter
- Routage au sein d'un seul système autonome
  - Protocole de routage intradomaine, *Interior Gateway Protocol*
  - Exemples : *Routing Information Protocol* (RIP), *Open Shortest Path First* (OSPF)
- Routage entre plusieurs systèmes autonomes
  - Protocole de routage interdomaines, *Exterior Gateway Protocol*

Attaques – *couche réseau (3) - protocoles de routage - OSPF*



Attaques – *couche réseau (3) - protocoles de routage - OSPF*





Routes calculées par routeur 2 :

1 R2 → R1 = 5

3 R2 → R3 = 10

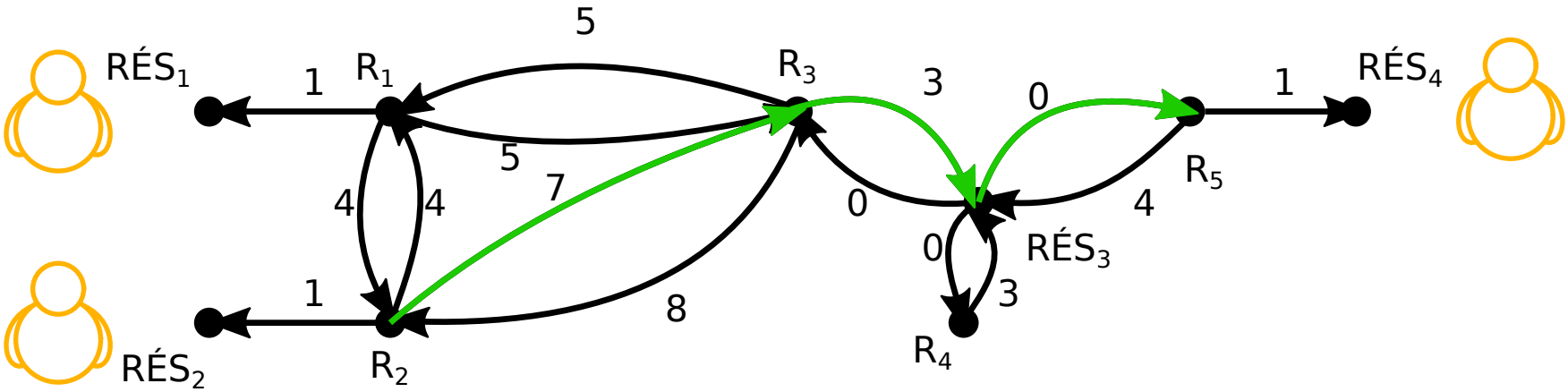
4 R2 → R3 → R5 = 11

Table de routage du routeur 2 :

1 → R1

3 → R3

4 → R3



Routes calculées par routeur 2 :

1 R2 → R1 = 5

3 R2 → R3 = 10

4 R2 → R3 → R5 = 11

Table de routage du routeur 2 :

1 → R1

3 → R3

4 → R3

Attaques – *couche réseau (3) - protocoles de routage*

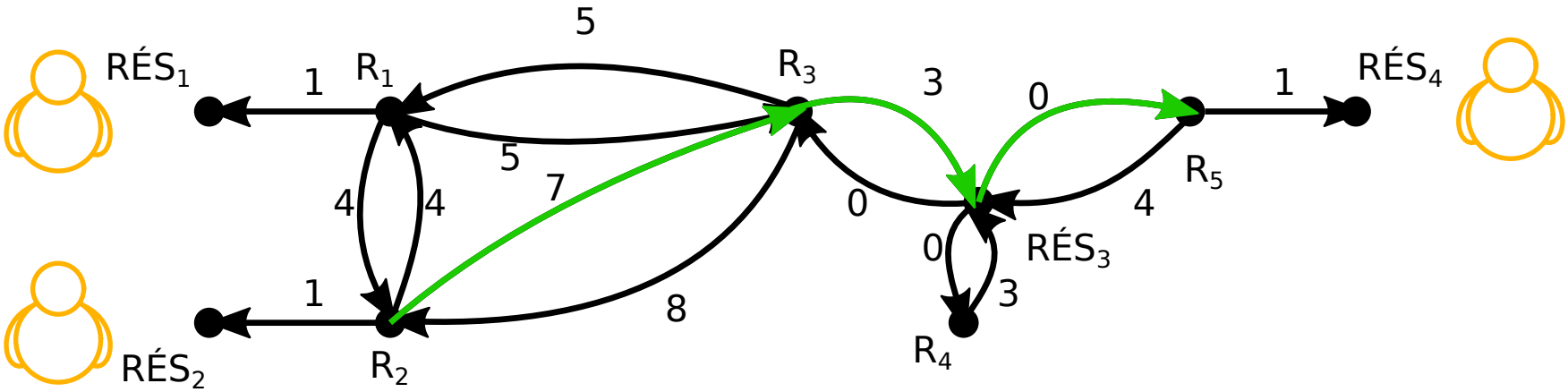
Quels sont les éléments critiques à protéger / à assurer ?

Attaques – *couche réseau (3) - protocoles de routage*

Quels sont les éléments critiques à protéger / à assurer ?

- Authenticité de l'identité des routeurs
- L'intégrité des accessibilités échangées
- L'intégrité des routeurs
- Leur confidentialité ?

Attaques – *couche réseau (3) - protocoles de routage - OSPF*



Routes calculées par R2 :

1 R2 → R1 = 5

3 R2 → R3 = 10

4 R2 → R3 → R5 = 11

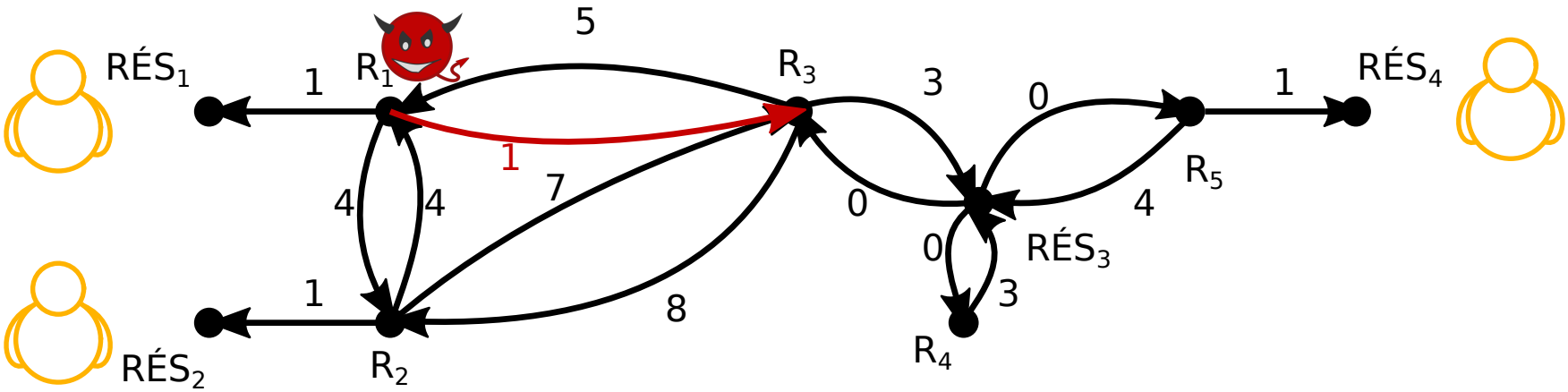
Table de routage de R2 :

1 → R1

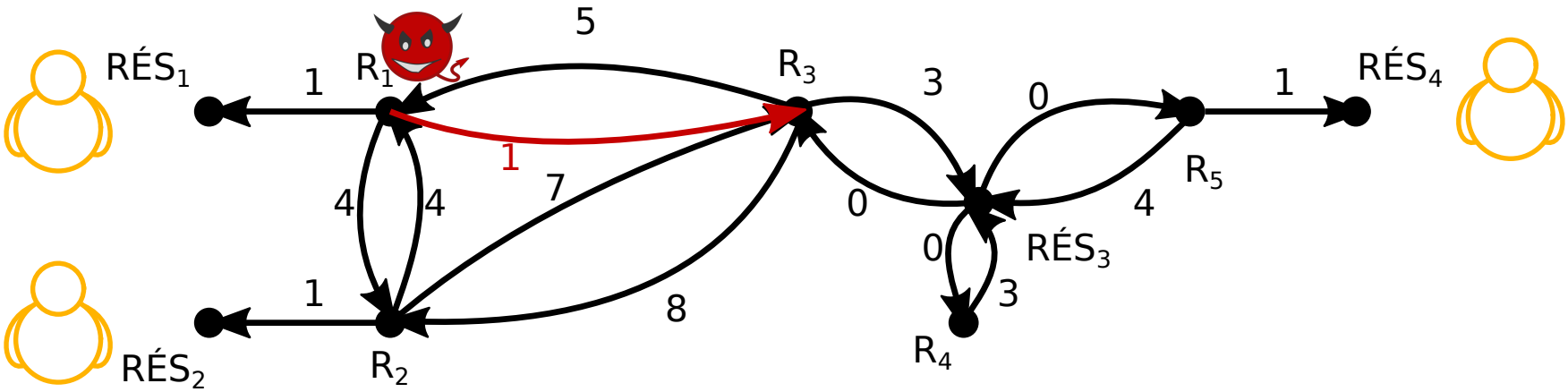
3 → R3

4 → R3

Attaques – *couche réseau (3) - protocoles de routage - OSPF*



Attaques – *couche réseau (3) - protocoles de routage - OSPF*



Routes calculées par R2 :

1 R2 → R1 = 5

3 R2 → R1 → R3 = 8

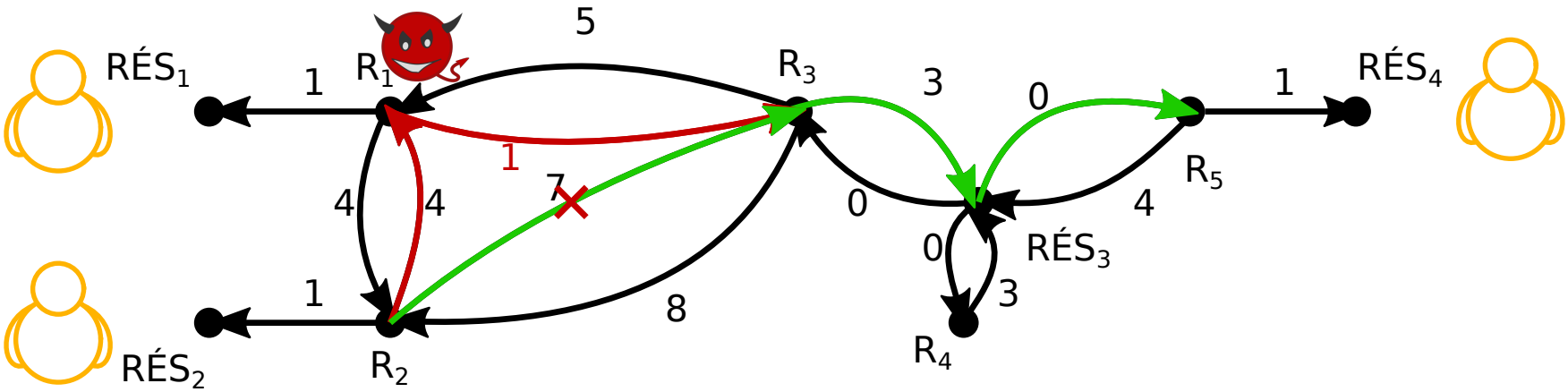
4 R2 → R1 → R3 → R5 = 9

Table de routage de R2 :

1 → R1

3 → R1

4 → R1



Routes calculées par R2 :

1 R2 → R1 = 5

3 R2 → R1 → R3 = 8

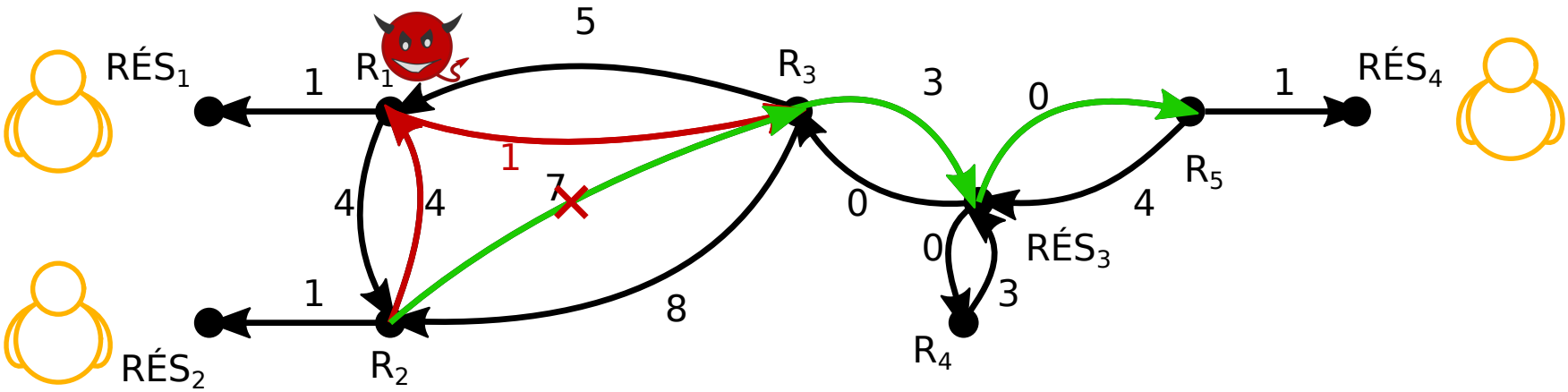
4 R2 → R1 → R3 → R5 = 9

Table de routage de R2 :

1 → R1

3 → R1

4 → R1



Routes calculées par R2 :

1 R2 → R1 = 5

3 R2 → R1 → R3 = 8

4 R2 → R1 → R3 → R5 = 9

Table de routage de R2 :

1 → R1

3 → R1

4 → R1

→ R1 est homme dans le milieu

Attaques – *couche réseau (3) - fragmentation IP*

Objectif :

- Le protocole IP permet la fragmentation des paquets qu'il relaye
- Cela permet le support de media aux tailles maximum de transfert variées

Utilisation par un routeur :

1. Réception d'un paquet P avec une MTU de 1500 (Ethernet)
2. Prochaine route : médium à la MTU de 1468 octets (ADSL)
3. $\text{MTU}_{\text{dst}} < \text{taille}(P) \rightarrow$ fragmentation en 2 fragments IP
4. Le routeur à l'autre bout du médium pourra éventuellement rassembler les fragments

Problème de sécurité : comment filtrer les fragments ?

Attaques – *couche réseau (3) - fragmentation IP*

Objectif :

- Le protocole IP permet la fragmentation des paquets qu'il relaye
- Cela permet le support de media aux tailles maximum de transfert variées

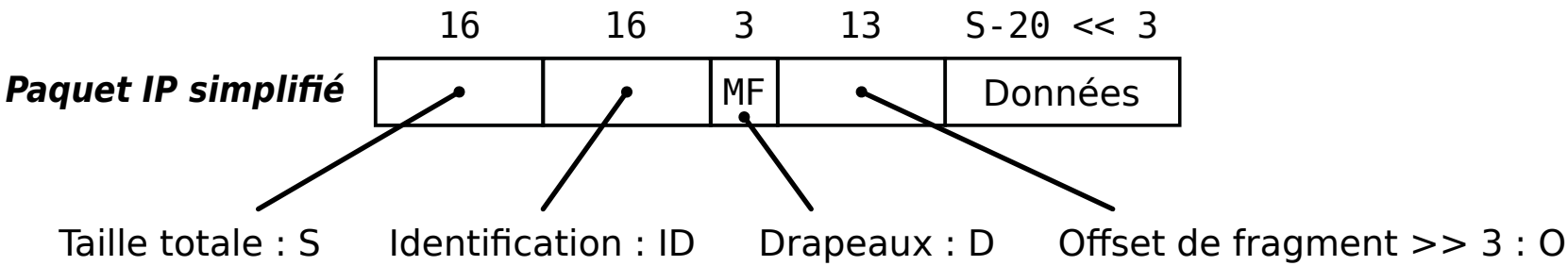
Utilisation par un routeur :

1. Réception d'un paquet P avec une MTU de 1500 (Ethernet)
2. Prochaine route : médium à la MTU de 1468 octets (ADSL)
3. $\text{MTU}_{\text{dst}} < \text{taille}(P) \rightarrow$ fragmentation en 2 fragments IP
4. Le routeur à l'autre bout du médium pourra éventuellement rassembler les fragments

Problème de sécurité : comment filtrer les fragments ?
$\rightarrow$ calcul du filtre sur le premier, puis idem pour les autres

Attaques – *couche réseau (3) - fragmentation IP*

Un paquet IP en provenance d'un réseau Ethernet doit être transporté sur un médium ADSL



Attaques – *couche réseau (3) - fragmentation IP*

Un paquet IP en provenance d'un réseau Ethernet doit être transporté sur un médium ADSL

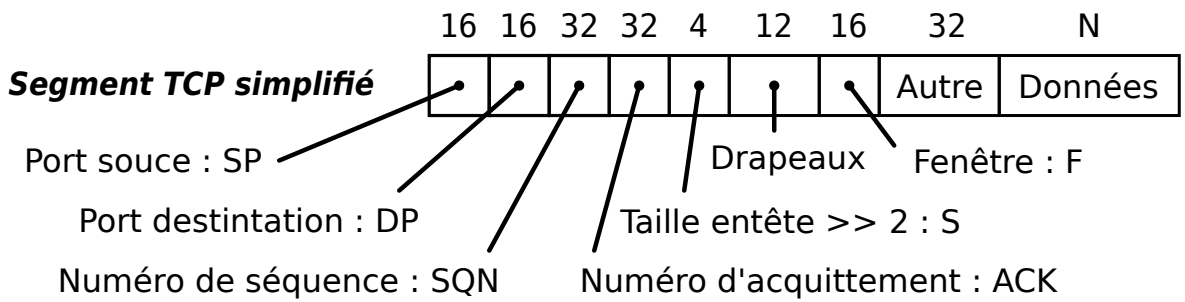
	S	ID	D	O	Données
Paquet IP simplifié	1500	0xcafe		0	[0:1480]

Attaques – *couche réseau (3) - fragmentation IP*

Un paquet IP en provenance d'un réseau Ethernet doit être transporté sur un médium ADSL

	S	ID	D	O	Données
Paquet IP simplifié	1500	0xcafe		0	[0:1480]
Fragment 1	1468	0xcafe	MF	0	[0:1447]
Fragment 2	52	0xcafe		181	[1448:1480]

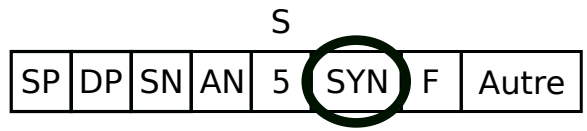
Objectif : contourner un pare feu qui filtre les segments SYN



Attaques – *couche réseau (3) - fragments minuscules*

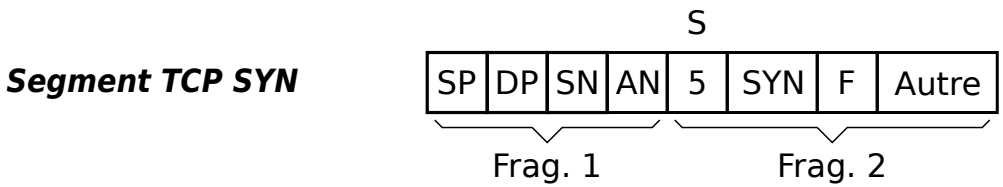
Objectif : contourner un pare feu qui filtre les segments SYN

Segment TCP SYN



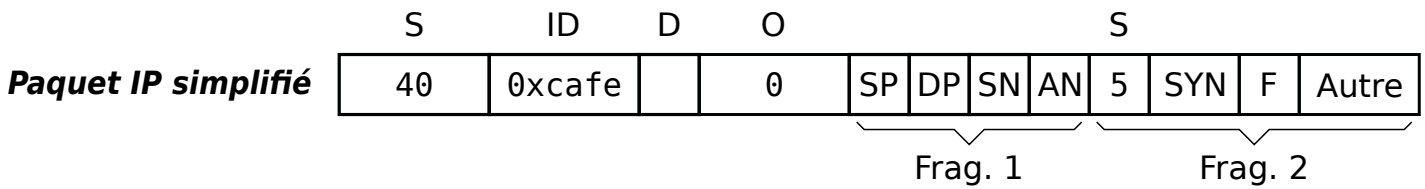
Attaques – *couche réseau (3) - fragments minuscules*

Objectif : contourner un pare feu qui filtre les segments SYN
Comment : cacher l'intention par fragmentation en plaçant les drapeaux dans un deuxième fragment



Attaques – *couche réseau (3) - fragments minuscules*

Objectif : contourner un pare feu qui filtre les segments SYN
Comment : cacher l'intention par fragmentation en plaçant les drapeaux dans un deuxième fragment

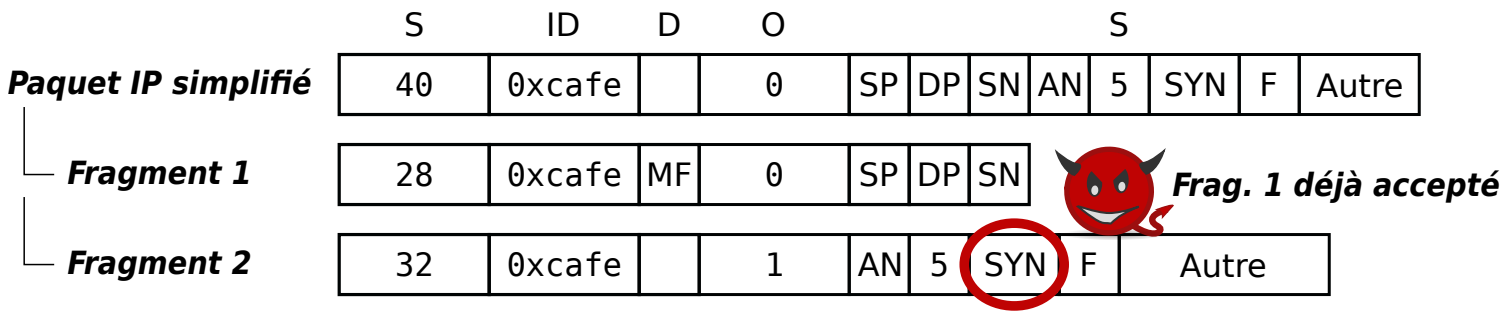


Attaques – *couche réseau (3) - fragments minuscules*

Objectif : contourner un pare feu qui filtre les segments SYN
Comment : cacher l'intention par fragmentation en plaçant les drapeaux dans un deuxième fragment

Attaques – *couche réseau (3) - fragments minuscules*

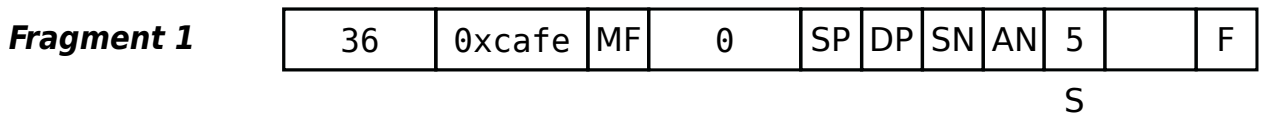
Objectif : contourner un pare feu qui filtre les segments SYN
Comment : cacher l'intention par fragmentation en plaçant les drapeaux dans un deuxième fragment



Conséquence : le premier fragment sera autorisé par erreur
→ le segment TCP entier est autorisé par erreur

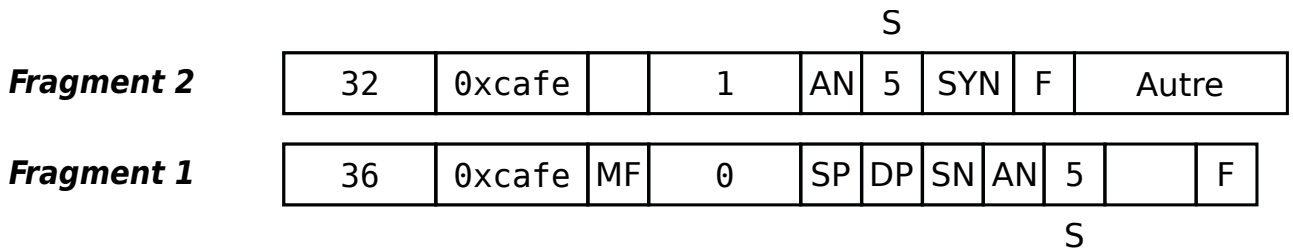
Attaques – *couche réseau (3) - fragments qui se chevauchent*

Objectif : contourner un pare feu qui filtre les segments SYN



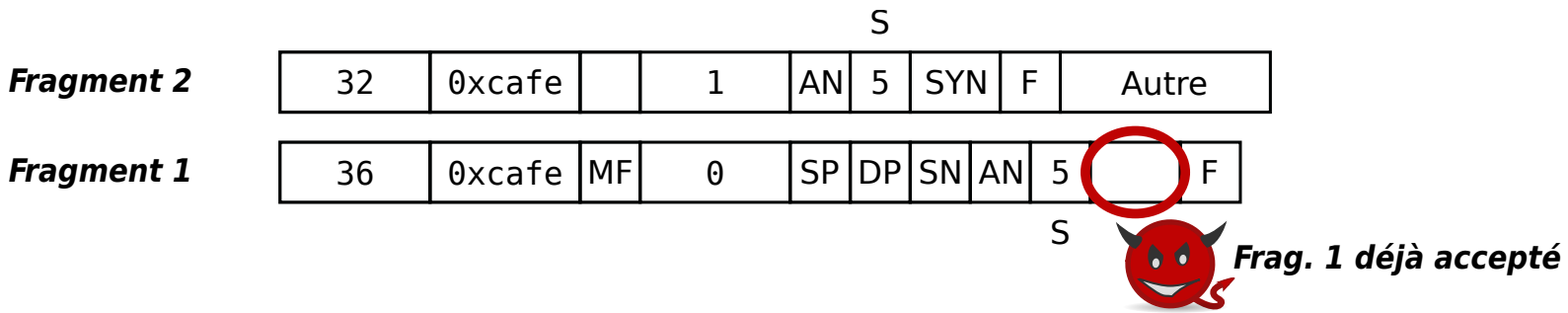
Attaques – *couche réseau (3) - fragments qui se chevauchent*

Objectif : contourner un pare feu qui filtre les segments SYN



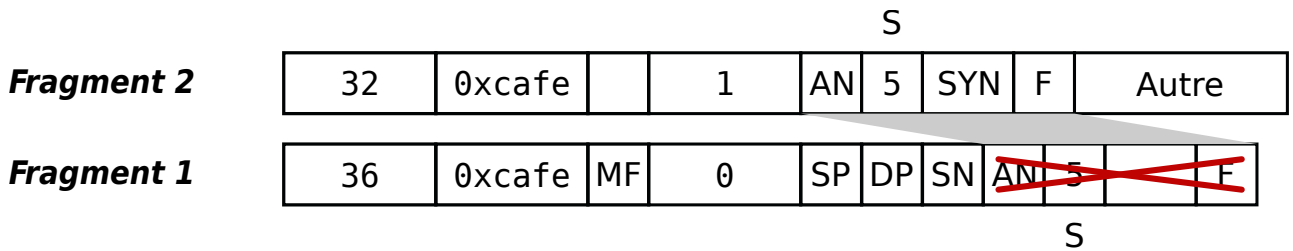
Attaques – *couche réseau (3) - fragments qui se chevauchent*

Objectif : contourner un pare feu qui filtre les segments SYN



Attaques – *couche réseau (3) - fragments qui se chevauchent*

Objectif : contourner un pare feu qui filtre les segments SYN
Comment : cacher l'intention par fragmentation en réécrivant les drapeaux par un deuxième fragment

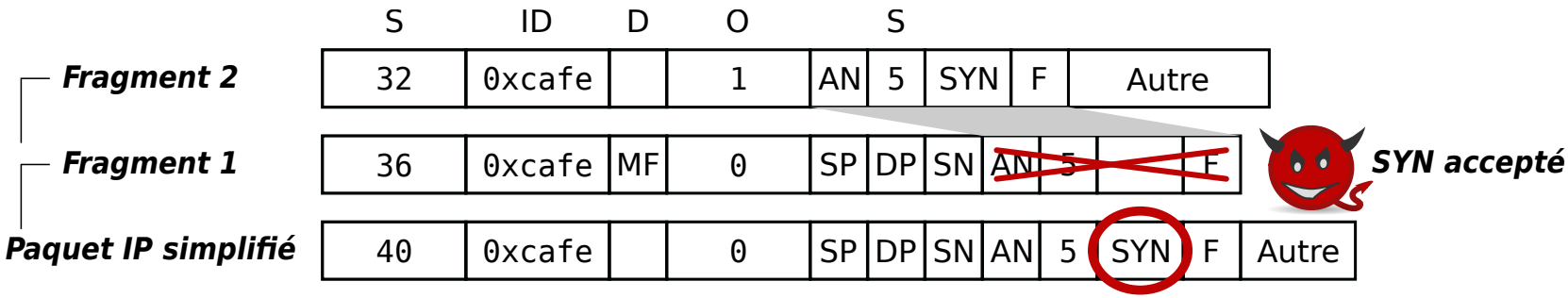


Attaques – *couche réseau (3) - fragments qui se chevauchent*

Objectif : contourner un pare feu qui filtre les segments SYN
Comment : cacher l'intention par fragmentation en réécrivant les drapeaux par un deuxième fragment

Attaques – *couche réseau (3) - fragments qui se chevauchent*

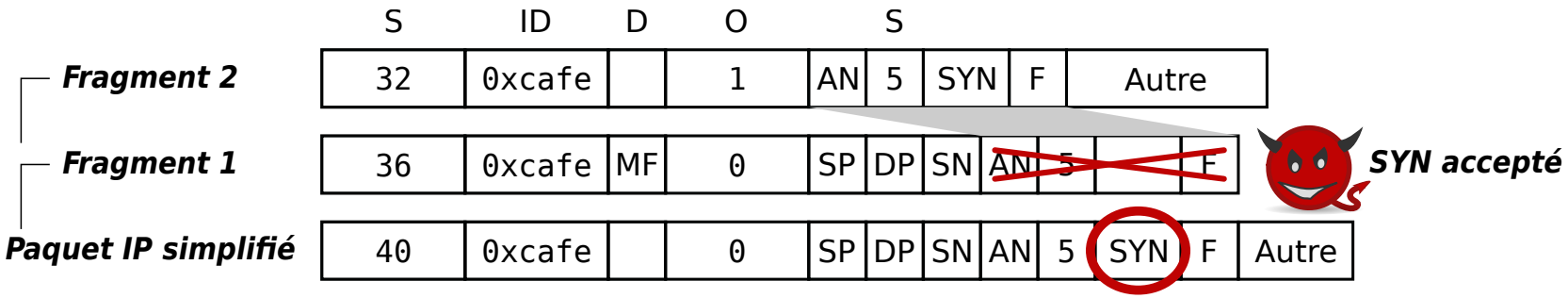
Objectif : contourner un pare feu qui filtre les segments SYN
Comment : cacher l'intention par fragmentation en réécrivant les drapeaux par un deuxième fragment



Conséquence : le premier fragment sera autorisé
$\rightarrow$ le segment TCP entier est autorisé par erreur

Attaques – *couche réseau (3) - fragments qui se chevauchent*

Objectif : contourner un pare feu qui filtre les segments SYN
Comment : cacher l'intention par fragmentation en réécrivant les drapeaux par un deuxième fragment



Conséquence : le premier fragment sera autorisé
→ le segment TCP entier est autorisé par erreur
Contremesure : systématiquement défragmenter dans le pare-feu

Attaques – *couche réseau (3) - fragments trop gros*

Objectif : dépassement de tampon dans l'équipement défragmentant

- Taille maximum d'un paquet IP $M = 2^{16} - 1 = 65535$ octets
- Les équipements sont dimensionnés en conséquence
- Or, offset de fragment maximum $OM = 2^{13} \times 8 = 65528$ octets

$\Rightarrow$ Large dépassement de M possible

- En effet, si MTU = 1500 octets et offset de fragment = OM
  $\Rightarrow$ taille totale paquet = $OM + 20 + 1500 = 67048 >> M$
- Exemple : ping de la mort

Attaques – *couche réseau (3) - schtroumpfage (smurfing)*

Objectif : dénis de service par inondation réseau

1. Envoyer une grosse quantité de requêtes `icmp-echo` vers l'addresse de diffusion (*broadcast*) d'un réseau (multiplicateur) et déguiser l'adresse source avec celle d'une machine victime
2. Toutes les machines IP du réseau multiplicateur vont répondre à la victime → grosse génération de trafic

⇒ Dénis de service sur le réseau ou la victime

Pour 1 paquet envoyé sur un réseau `/n` le trafic généré est :
$2^{32-n} - 2$ messages répondus, si toutes les machines sont actives.

Attaques – *couche réseau (3) - dénis de service distribué*

Caractéristiques

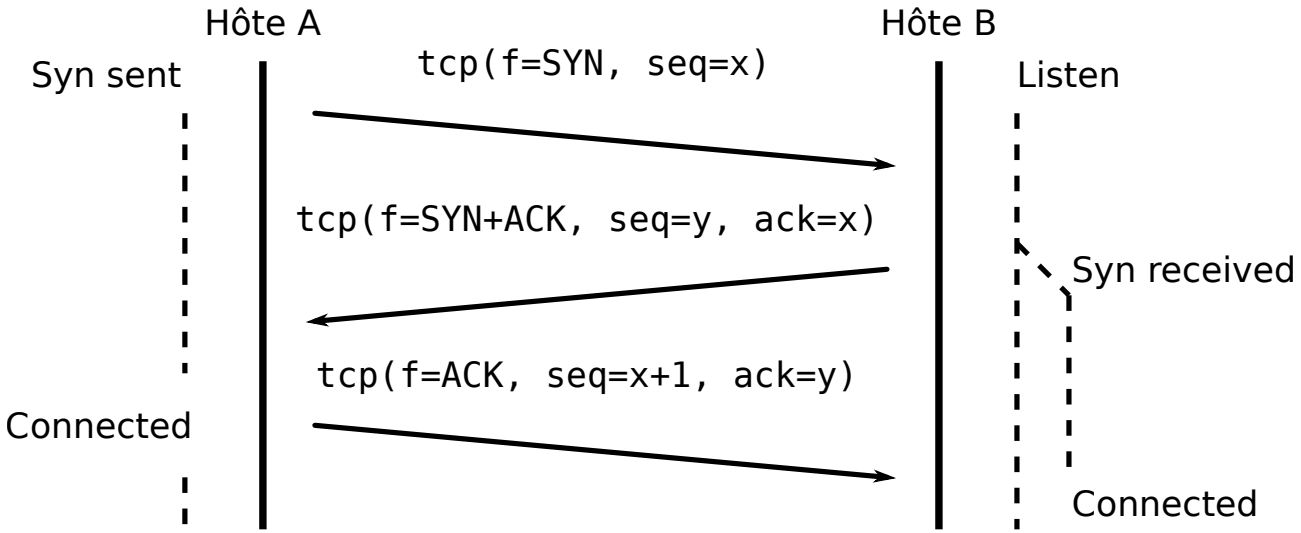
- Inondation réseau depuis des sources multiples
- Difficile à détecter :
  Par construction : IP source et port source aléatoires
  Amélioration : IP destination et port destination aléatoires
- Multiplication de trafic : schtroumpfage, requêtes récursives DNS, etc.

Condition d'un réseau pour l'inondation : *botnet*

- Infection de machines victimes contrôlées à distance
- Contrôle à distance via des protocoles aux ports ouverts : IRC
- Compromission par virus ou vers, etc.
- Réalité : certains spécialistes avancent qu'un machine sur 4 est un **zombie**, c'est à dire contrôlée et utilisée dans un botnet

Attaques – *couche transport (4) - dénis de service TCP*

Inondation TCP SYN : ouverture de connexions non fermés



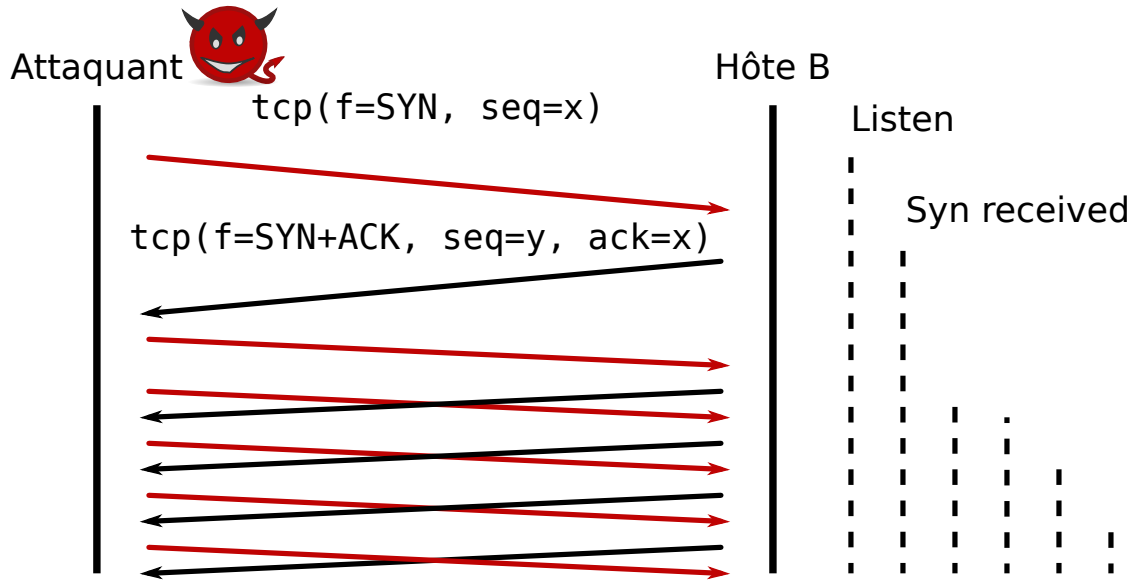
- Cf. attaque de Kevin Mitnick : inondation TCP de la victime

1. L'attaquant génère un certain nombre d'ouverture de connexions
2. La victime accepte la connexion et exécute la connexion inverse

Contremesure : file de connexion semi ouvertes : `listen(sock, 5)`

Attaques – *couche transport (4) - dénis de service TCP*

Inondation TCP SYN : ouverture de connexions non fermés



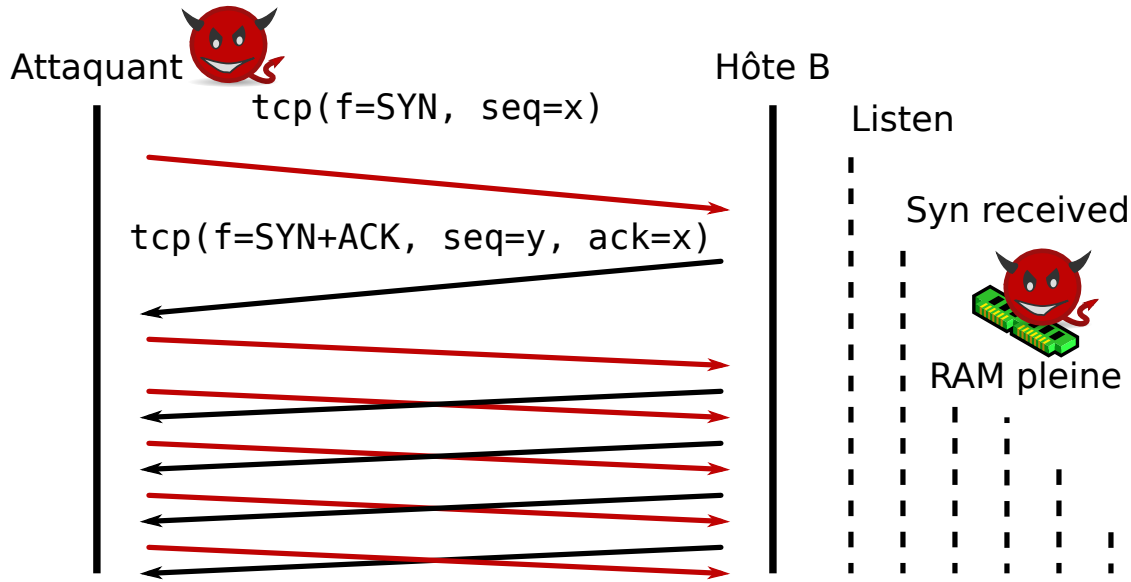
- Cf. attaque de Kevin Mitnick : inondation TCP de la victime

1. L'attaquant génère un certain nombre d'ouverture de connexions
2. La victime accepte la connexion et exécute la connexion inverse
   → consommation mémoire pour l'attente

Contremesure : file de connexion semi ouvertes : `listen(sock, 5)`

Attaques – *couche transport (4) - dénis de service TCP*

Inondation TCP SYN : ouverture de connexions non fermés



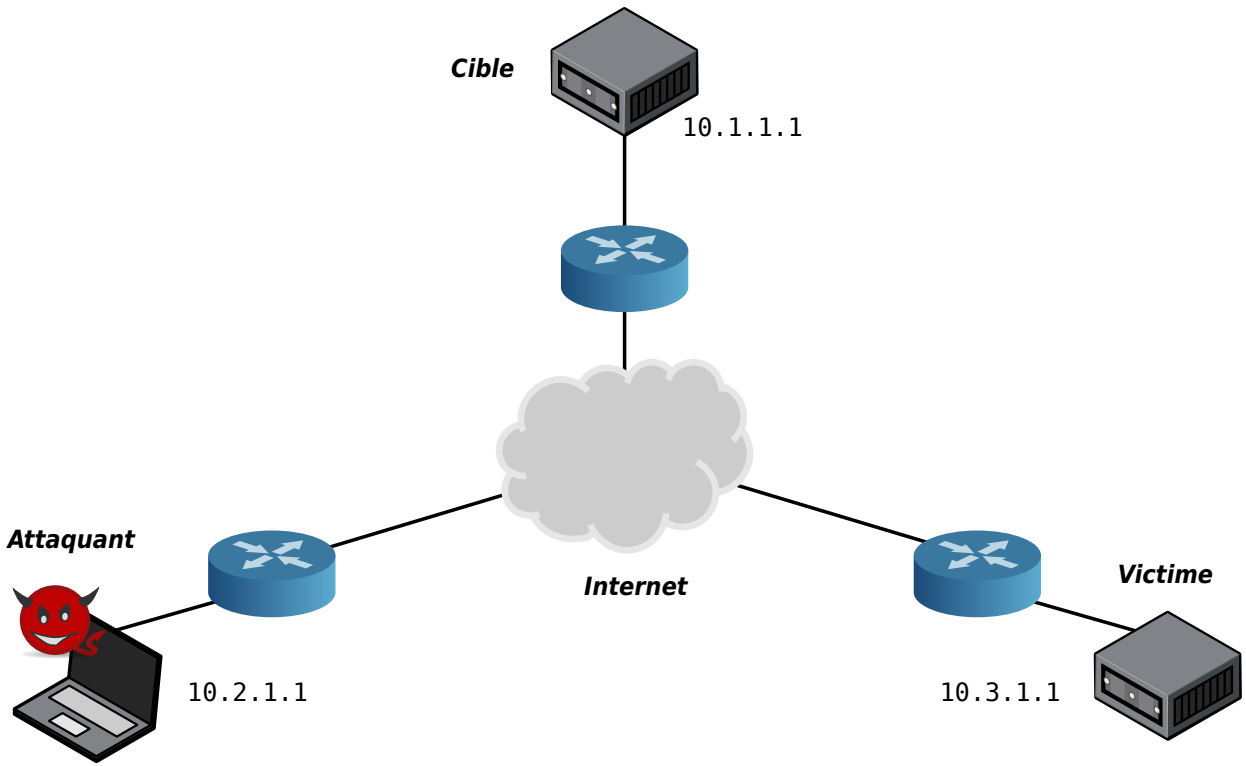
- Cf. attaque de Kevin Mitnick : inondation TCP de la victime

1. L'attaquant génère un certain nombre d'ouverture de connexions
2. La victime accepte la connexion et exécute la connexion inverse
   → consommation mémoire pour l'attente → dénis de service

Contremesure : file de connexion semi ouvertes : `listen(sock, 5)`

Attaques – *couche transport (4) - Déguisement TCP / IP*

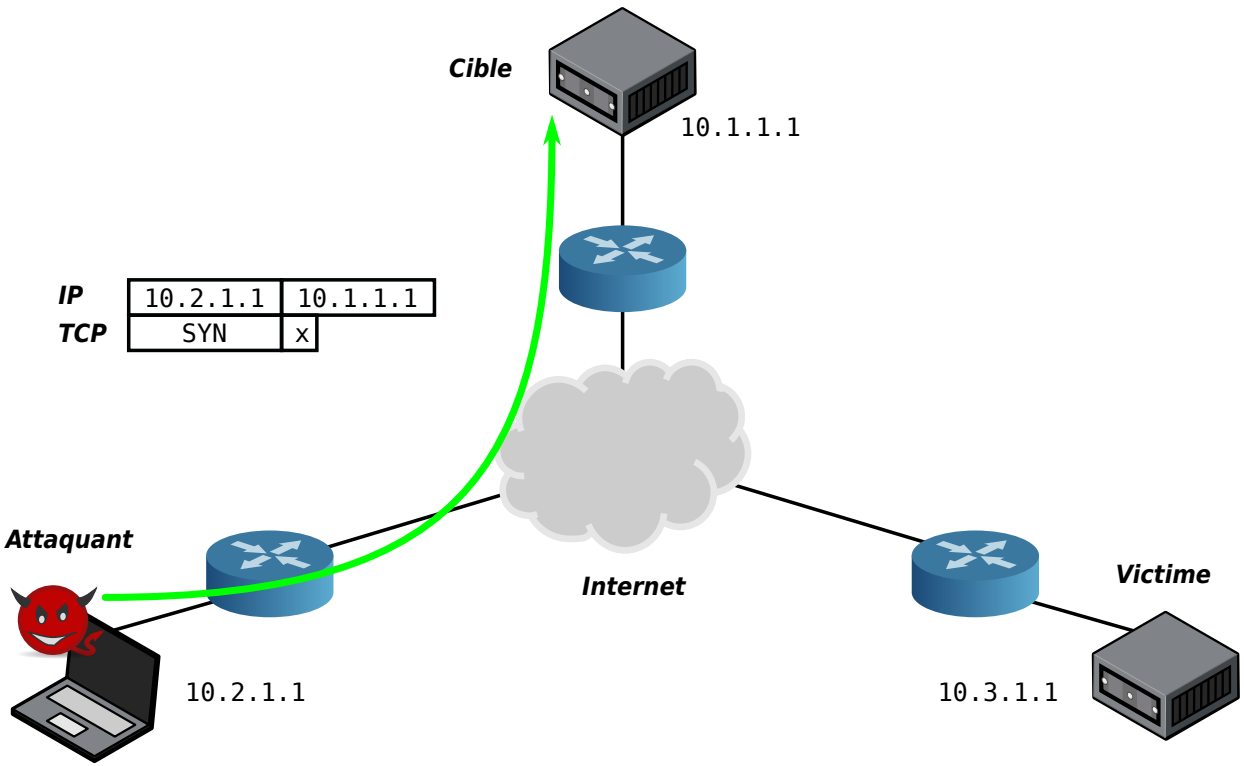
Principe : déguisement IP en devinant les numéros de séquence TCP choisis par la cible. Exemple : attaque de Kevin Mitnick
Cette attaque ne peut fonctionner que sur des systèmes cibles où les numéros de séquences choisis sont prédictibles.



Contremesure : randomisation des numéros de séquence utilisés

Attaques – *couche transport (4) - Déguisement TCP / IP*

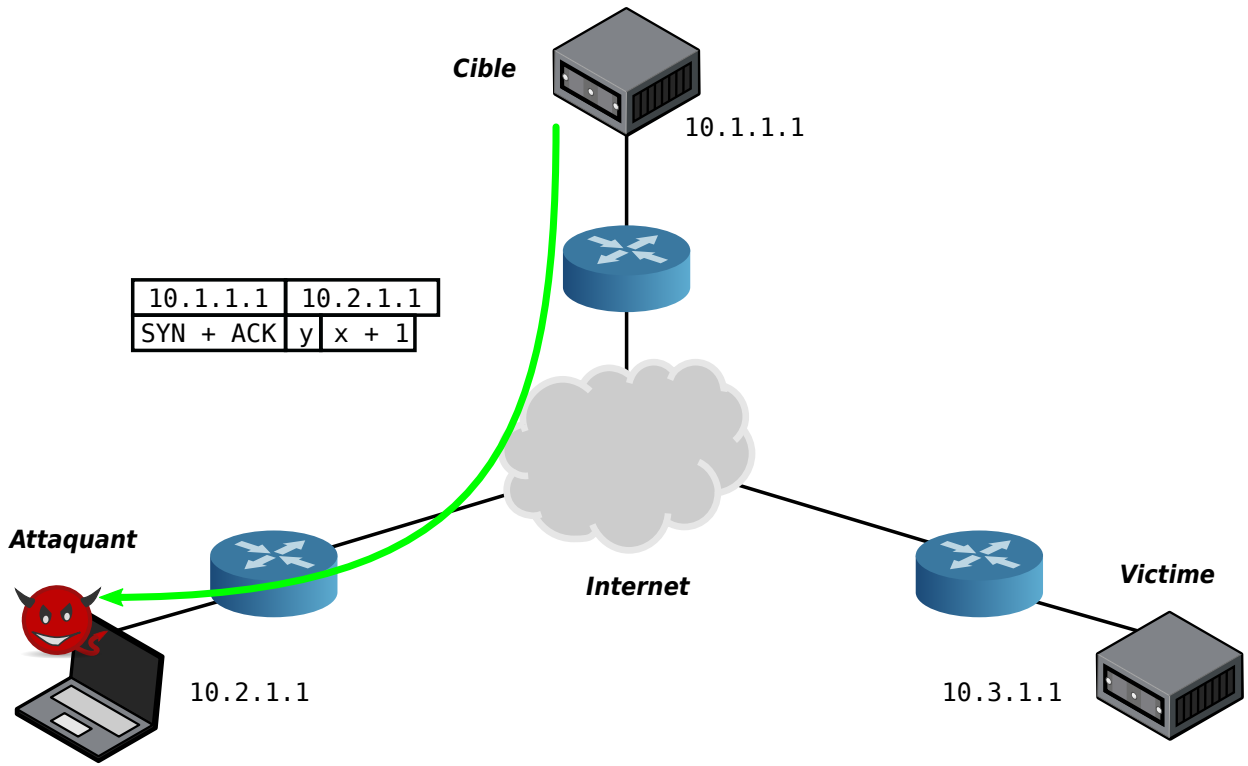
Principe : déguisement IP en devinant les numéros de séquence TCP choisis par la cible. Exemple : attaque de Kevin Mitnick
Cette attaque ne peut fonctionner que sur des systèmes cibles où les numéros de séquences choisis sont prédictibles.



Contremesure : randomisation des numéros de séquence utilisés

Attaques – *couche transport (4) - Déguisement TCP / IP*

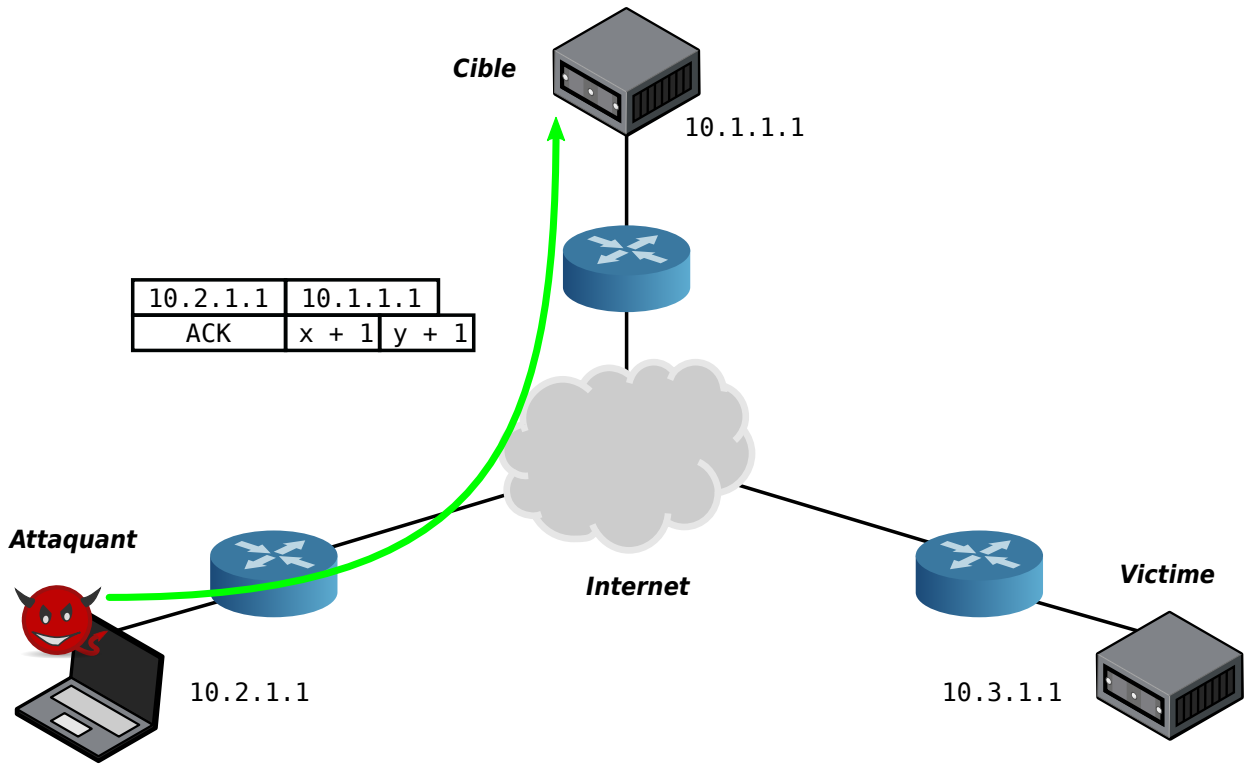
Principe : déguisement IP en devinant les numéros de séquence TCP choisis par la cible. Exemple : attaque de Kevin Mitnick
Cette attaque ne peut fonctionner que sur des systèmes cibles où les numéros de séquences choisis sont prédictibles.



Contremesure : randomisation des numéros de séquence utilisés

Attaques – *couche transport (4) - Déguisement TCP / IP*

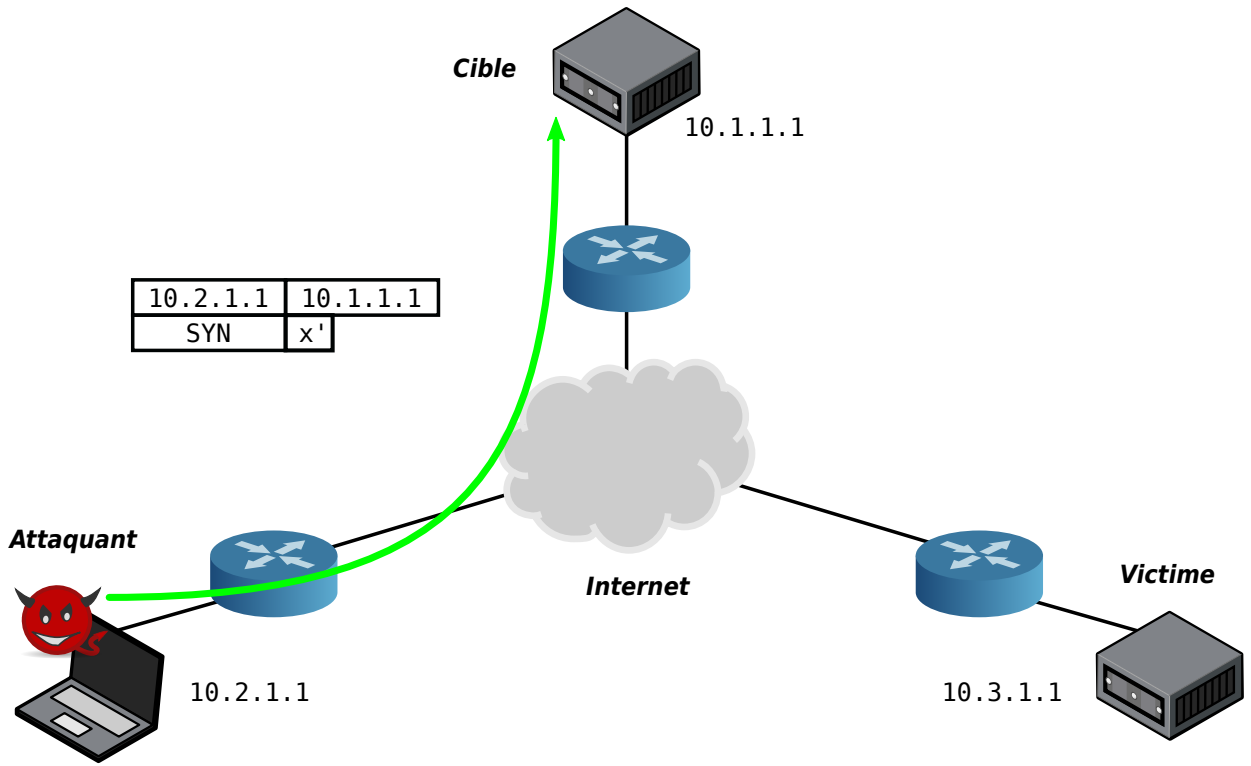
Principe : déguisement IP en devinant les numéros de séquence TCP choisis par la cible. Exemple : attaque de Kevin Mitnick
Cette attaque ne peut fonctionner que sur des systèmes cibles où les numéros de séquences choisis sont prédictibles.



Contremesure : randomisation des numéros de séquence utilisés

Attaques – *couche transport (4) - Déguisement TCP / IP*

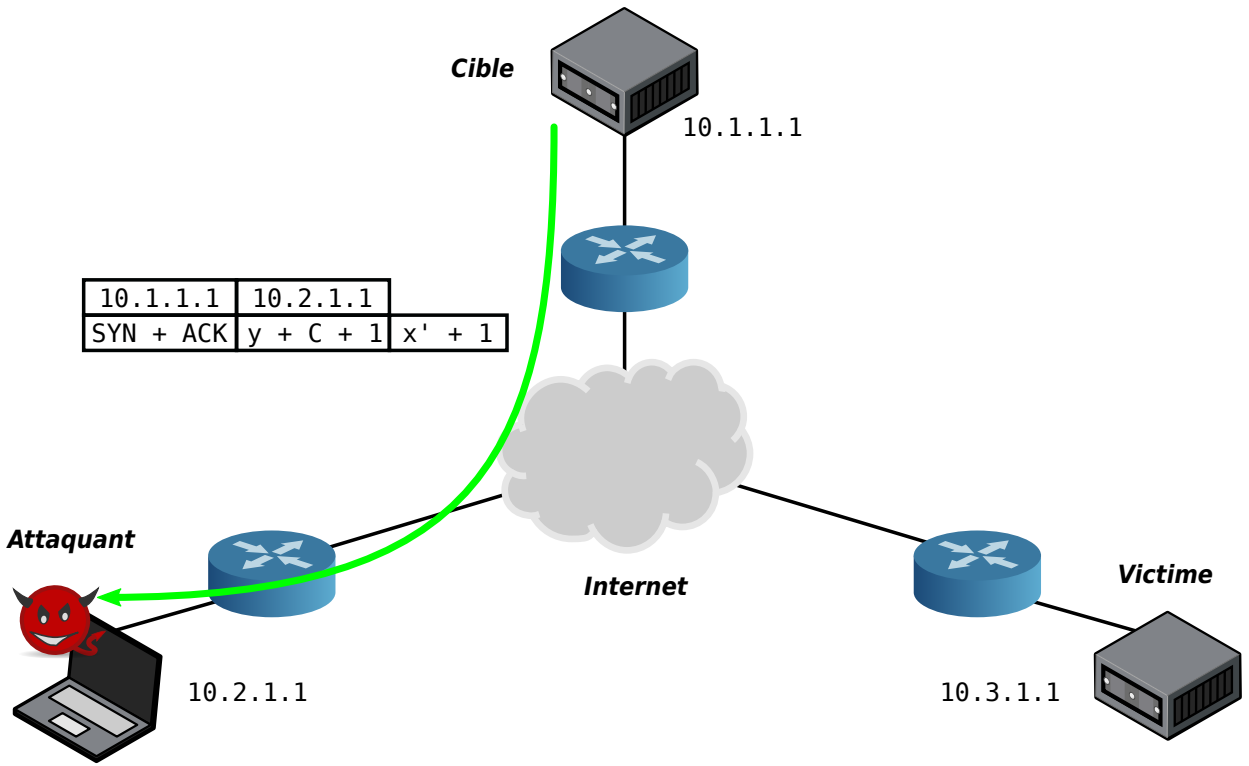
Principe : déguisement IP en devinant les numéros de séquence TCP choisis par la cible. Exemple : attaque de Kevin Mitnick
Cette attaque ne peut fonctionner que sur des systèmes cibles où les numéros de séquences choisis sont prédictibles.



Contremesure : randomisation des numéros de séquence utilisés

Attaques – *couche transport (4) - Déguisement TCP / IP*

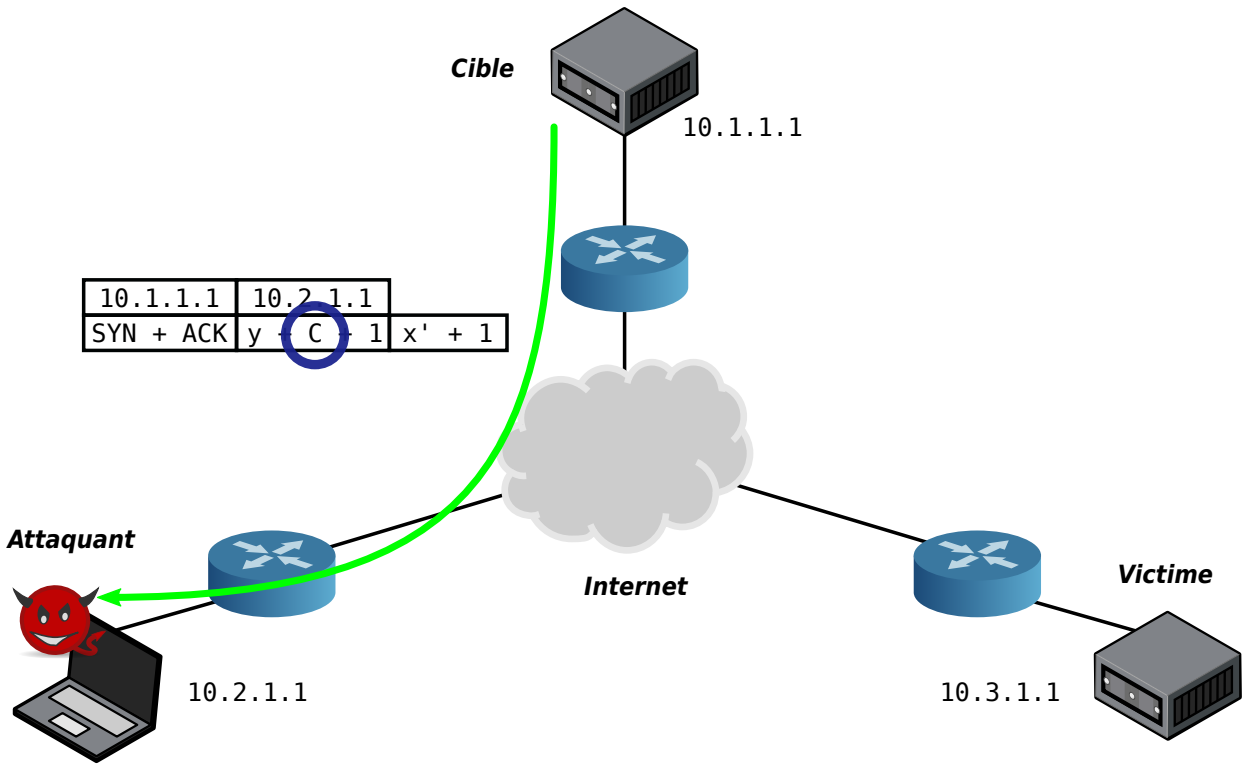
Principe : déguisement IP en devinant les numéros de séquence TCP choisis par la cible. Exemple : attaque de Kevin Mitnick
Cette attaque ne peut fonctionner que sur des systèmes cibles où les numéros de séquences choisis sont prédictibles.



Contremesure : randomisation des numéros de séquence utilisés

Attaques – *couche transport (4) - Déguisement TCP / IP*

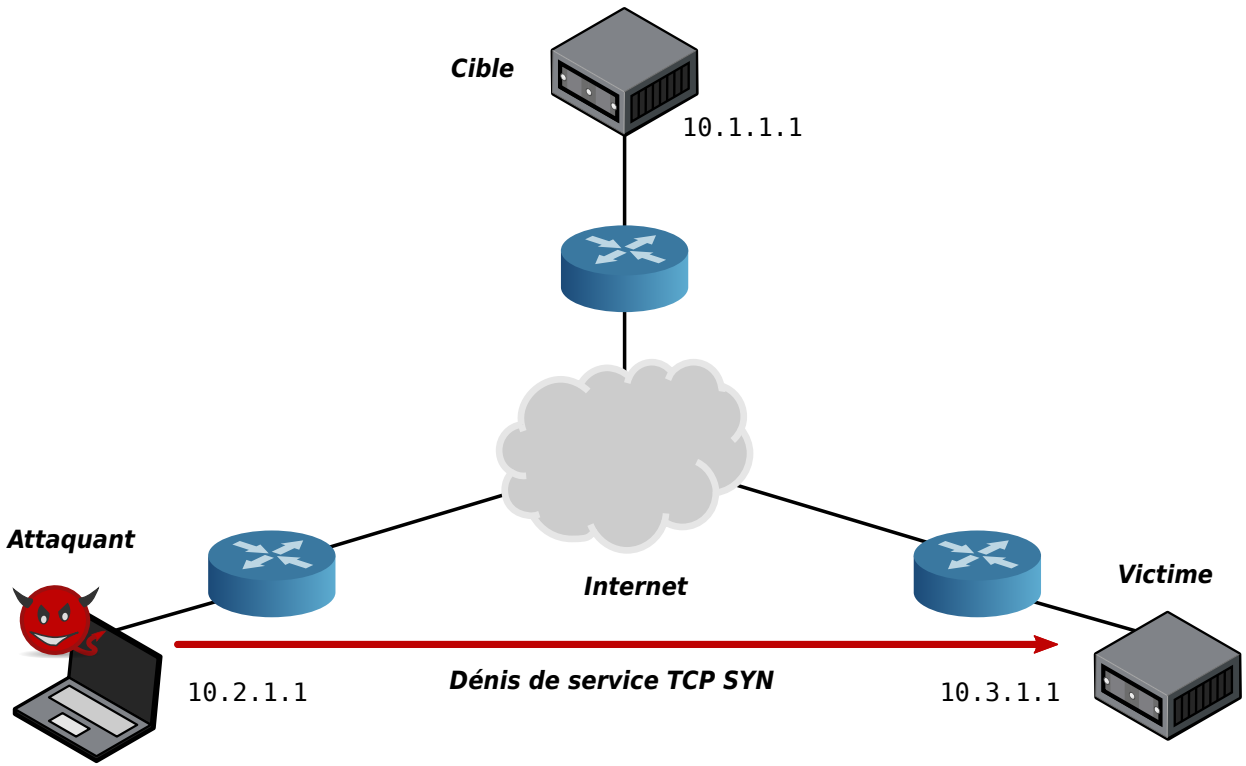
Principe : déguisement IP en devinant les numéros de séquence TCP choisis par la cible. Exemple : attaque de Kevin Mitnick
Cette attaque ne peut fonctionner que sur des systèmes cibles où les numéros de séquences choisis sont prédictibles.



Contremesure : randomisation des numéros de séquence utilisés

Attaques – *couche transport (4) - Déguisement TCP / IP*

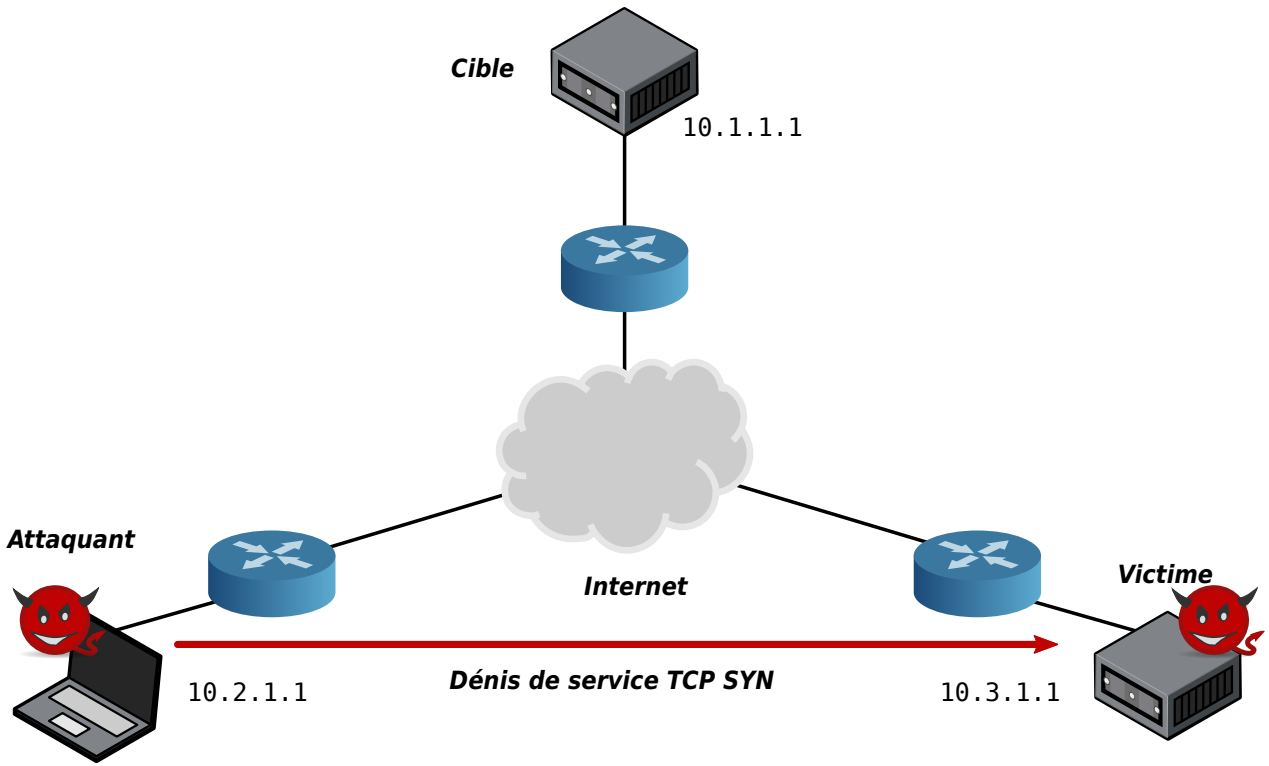
Principe : déguisement IP en devinant les numéros de séquence TCP choisis par la cible. Exemple : attaque de Kevin Mitnick
Cette attaque ne peut fonctionner que sur des systèmes cibles où les numéros de séquences choisis sont prédictibles.



Contremesure : randomisation des numéros de séquence utilisés

Attaques – *couche transport (4) - Déguisement TCP / IP*

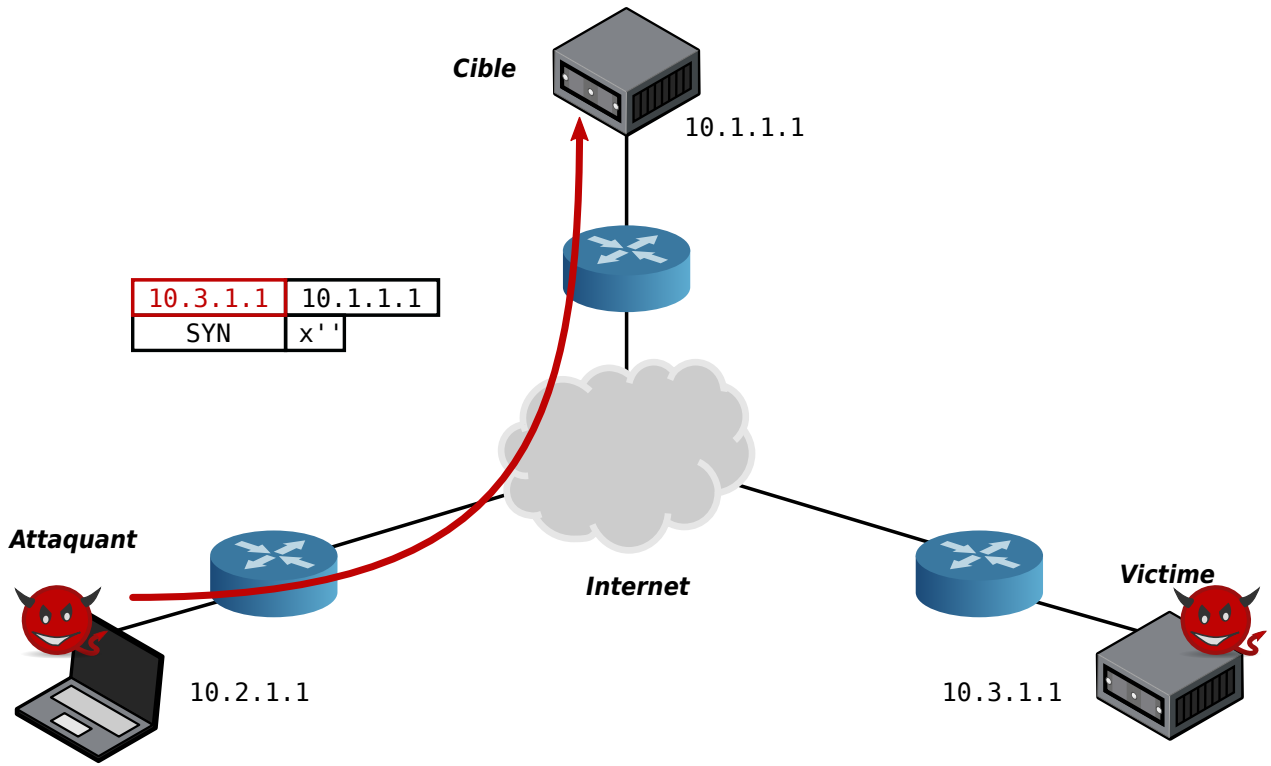
Principe : déguisement IP en devinant les numéros de séquence TCP choisis par la cible. Exemple : attaque de Kevin Mitnick
Cette attaque ne peut fonctionner que sur des systèmes cibles où les numéros de séquences choisis sont prédictibles.



Contremesure : randomisation des numéros de séquence utilisés

Attaques – *couche transport (4) - Déguisement TCP / IP*

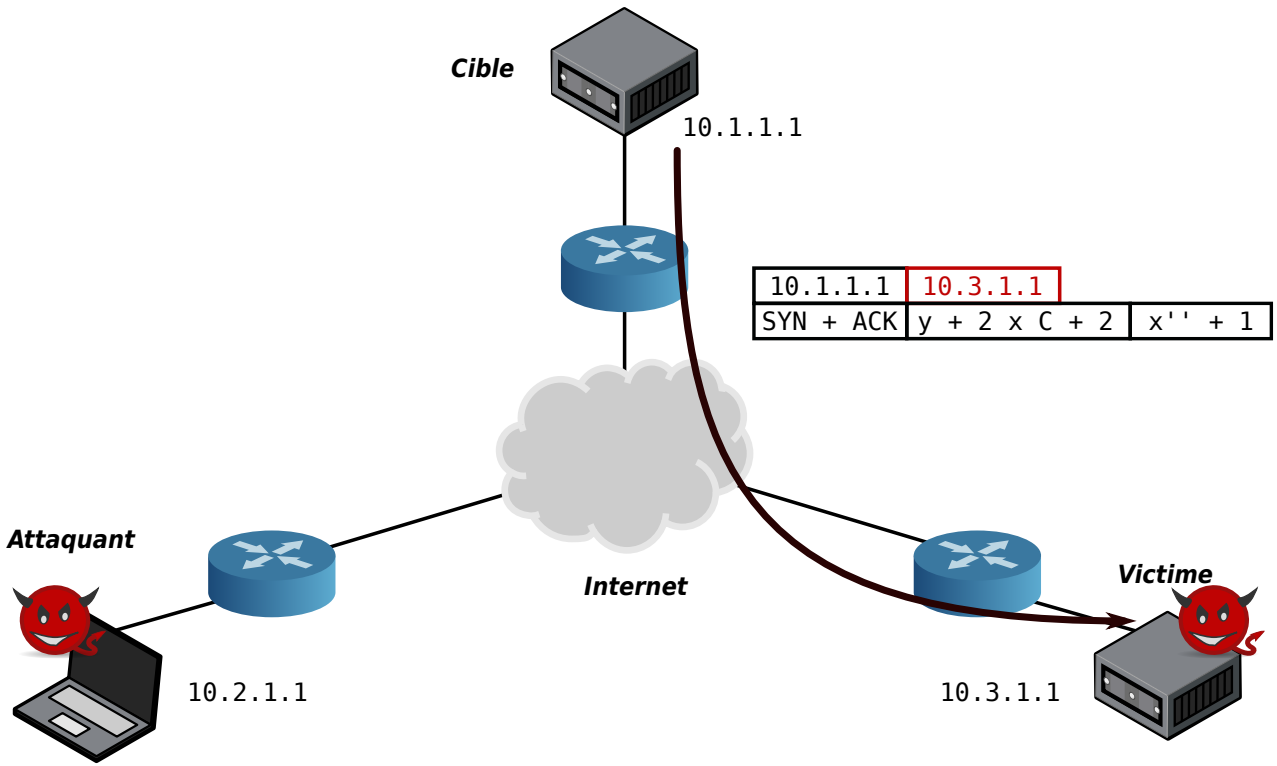
Principe : déguisement IP en devinant les numéros de séquence TCP choisis par la cible. Exemple : attaque de Kevin Mitnick
Cette attaque ne peut fonctionner que sur des systèmes cibles où les numéros de séquences choisis sont prédictibles.



Contremesure : randomisation des numéros de séquence utilisés

Attaques – *couche transport (4) - Déguisement TCP / IP*

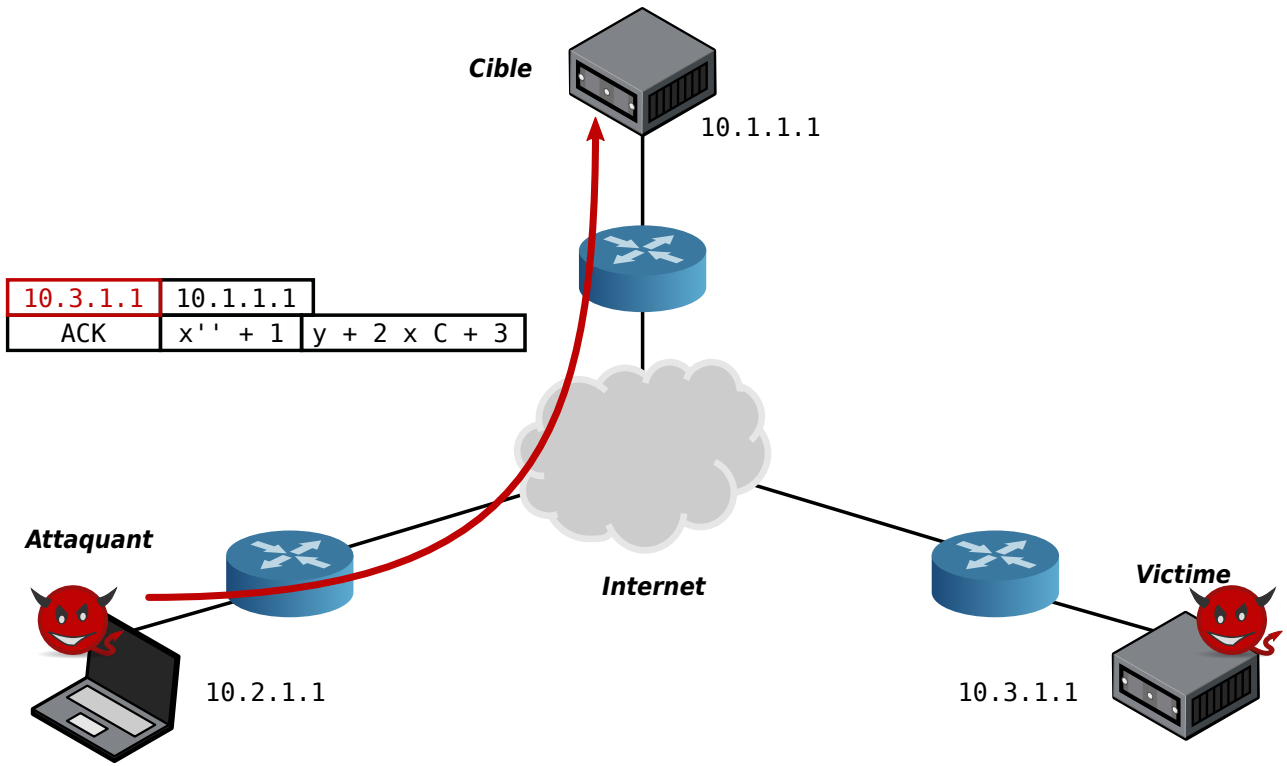
Principe : déguisement IP en devinant les numéros de séquence TCP choisis par la cible. Exemple : attaque de Kevin Mitnick
Cette attaque ne peut fonctionner que sur des systèmes cibles où les numéros de séquences choisis sont prédictibles.



Contremesure : randomisation des numéros de séquence utilisés

Attaques – *couche transport (4) - Déguisement TCP / IP*

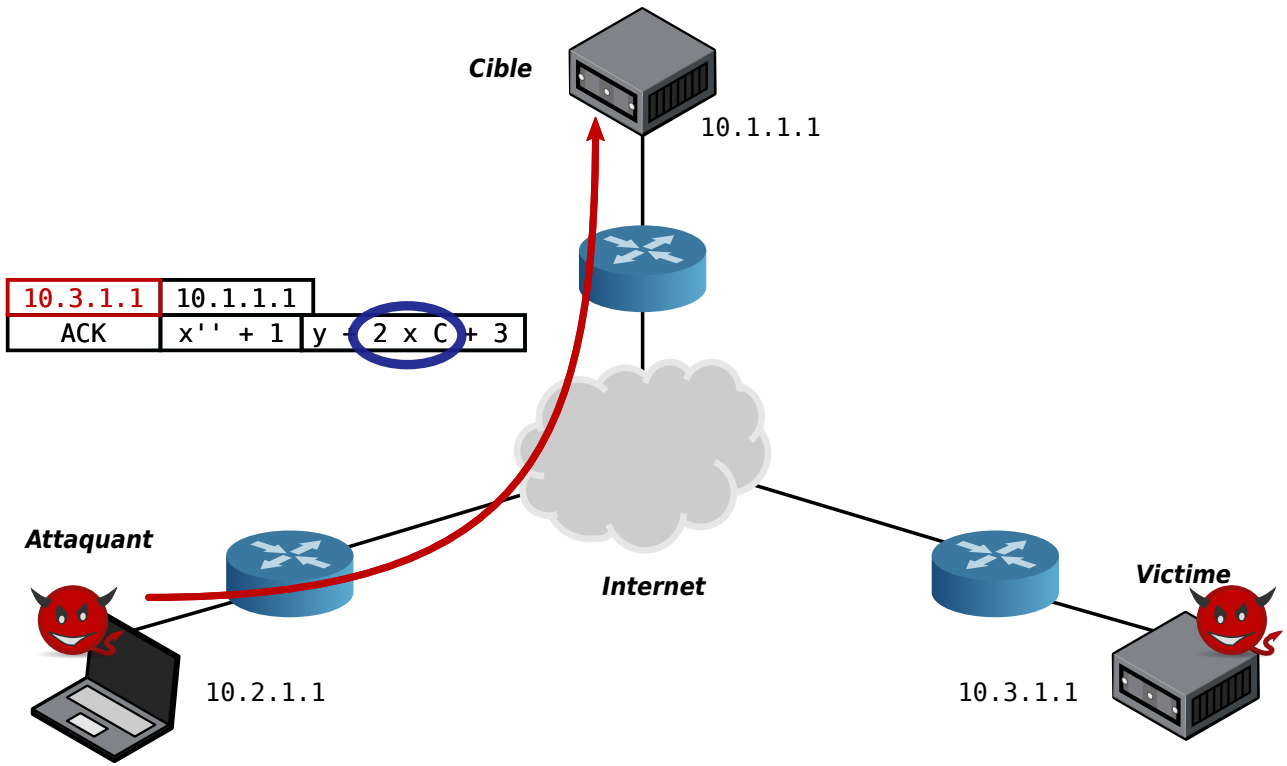
Principe : déguisement IP en devinant les numéros de séquence TCP choisis par la cible. Exemple : attaque de Kevin Mitnick
Cette attaque ne peut fonctionner que sur des systèmes cibles où les numéros de séquences choisis sont prédictibles.



Contremesure : randomisation des numéros de séquence utilisés

Attaques – *couche transport (4) - Déguisement TCP / IP*

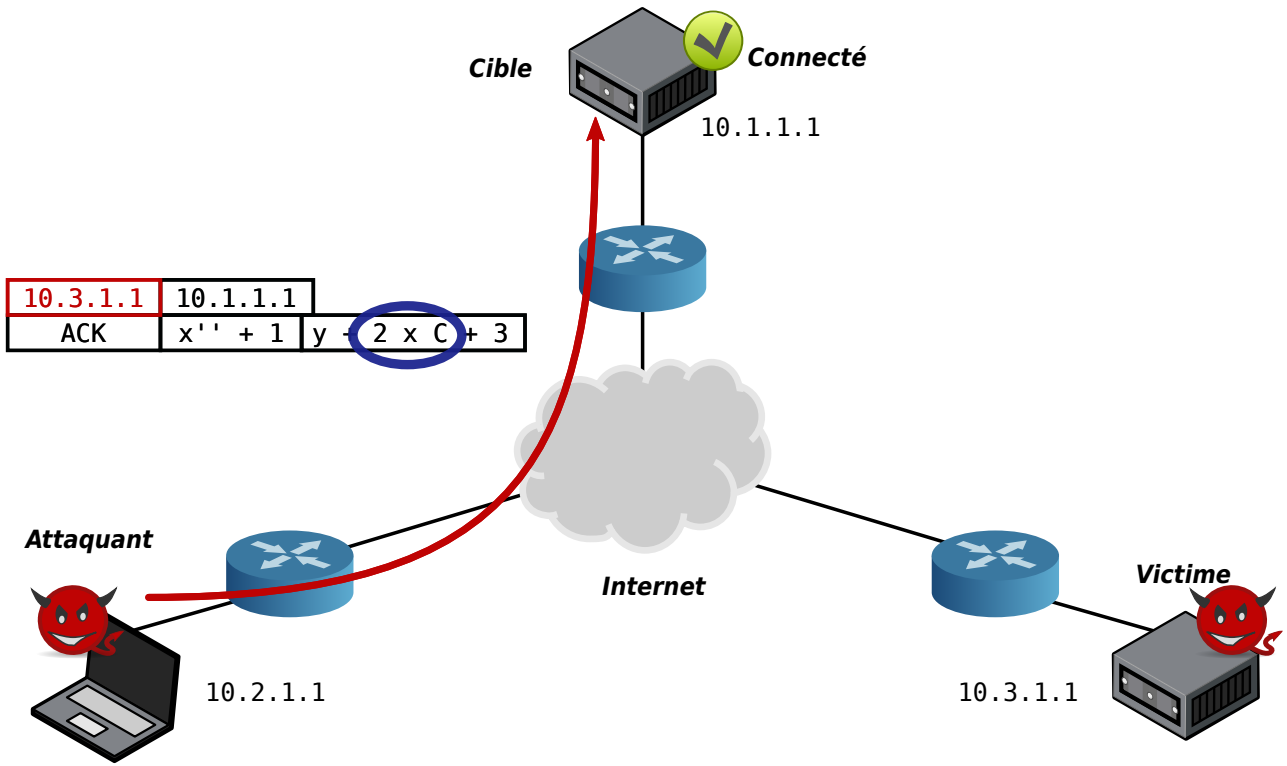
Principe : déguisement IP en devinant les numéros de séquence TCP choisis par la cible. Exemple : attaque de Kevin Mitnick
Cette attaque ne peut fonctionner que sur des systèmes cibles où les numéros de séquences choisis sont prédictibles.



Contremesure : randomisation des numéros de séquence utilisés

Attaques – *couche transport (4) - Déguisement TCP / IP*

Principe : déguisement IP en devinant les numéros de séquence TCP choisis par la cible. Exemple : attaque de Kevin Mitnick
Cette attaque ne peut fonctionner que sur des systèmes cibles où les numéros de séquences choisis sont prédictibles.



Contremesure : randomisation des numéros de séquence utilisés

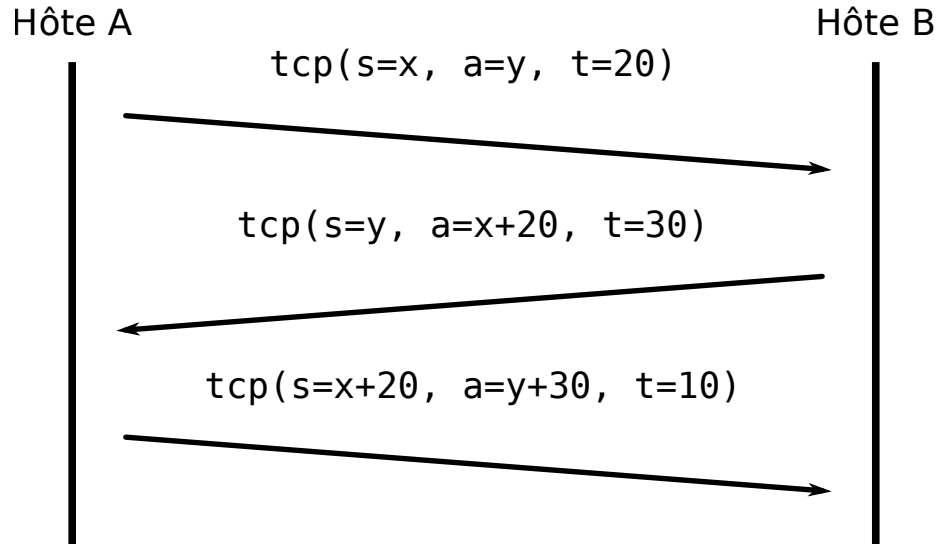
Attaques – *couche transport (4) - Interruption de session TCP*

Objectif : dénis de service par interruption de session TCP

- Envoi d'un segment TCP RST portant le bon numéro de séquence
- Déguisement IP nécessaire
- Attaqie triviale si placé en homme dans le milieu
- Dans les autres cas, attaque par force brute

Attaques – *couche transport (4) - désynchronisation de session TCP*

Objectif : vol de session TCP, dénis de service



Définition : Séssion désynchronisée si

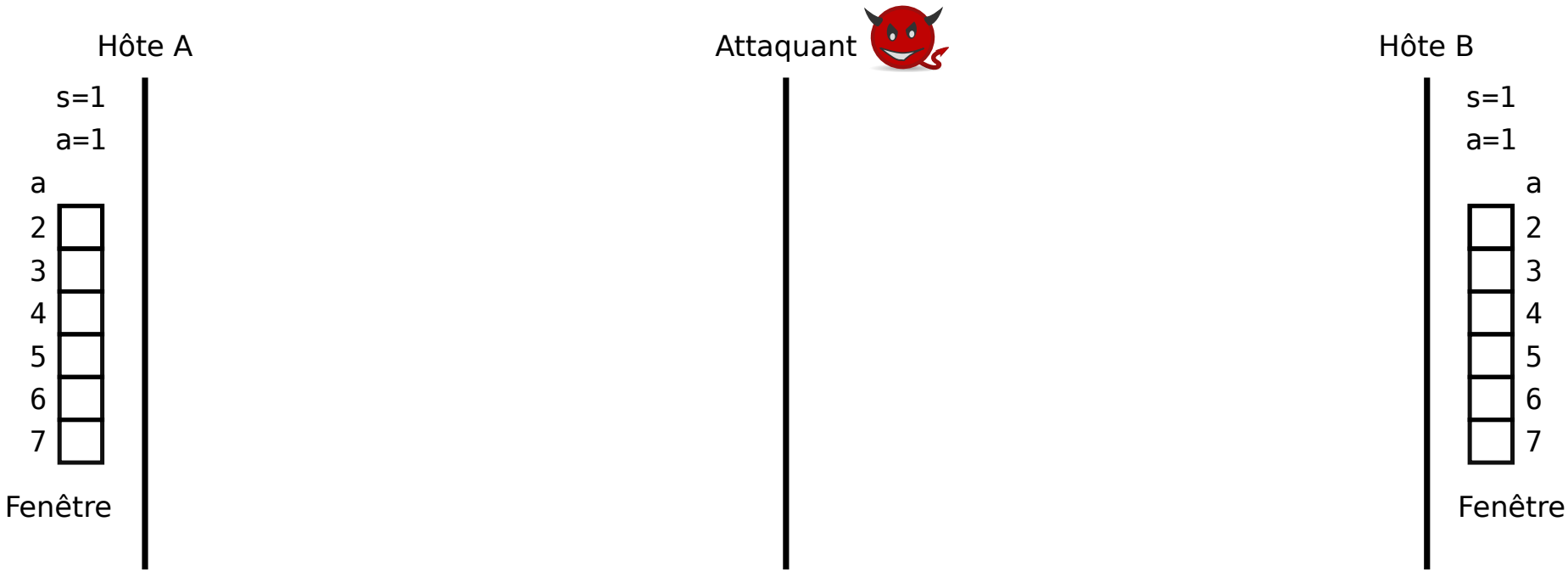
- $\text{ack}_A^i \neq \text{seq}_B^{i+1} \vee \text{ack}_B^i \neq \text{seq}_A^{i+1}$
- $\text{ack}_A^{i+1} \neq \text{seq}_B^i + \text{taille}(i) \vee \text{ack}_B^{i+1} \neq \text{seq}_A^i + \text{taille}(i)$

Conséquence : la cible refuse les paquets de la victime et peut éventuellement accepter des paquets de l'attaquant

Méthodes : insertion de segments *null data* durant la session ; désynchronisation lors de la poignée de main TCP (*early desynchronization*)

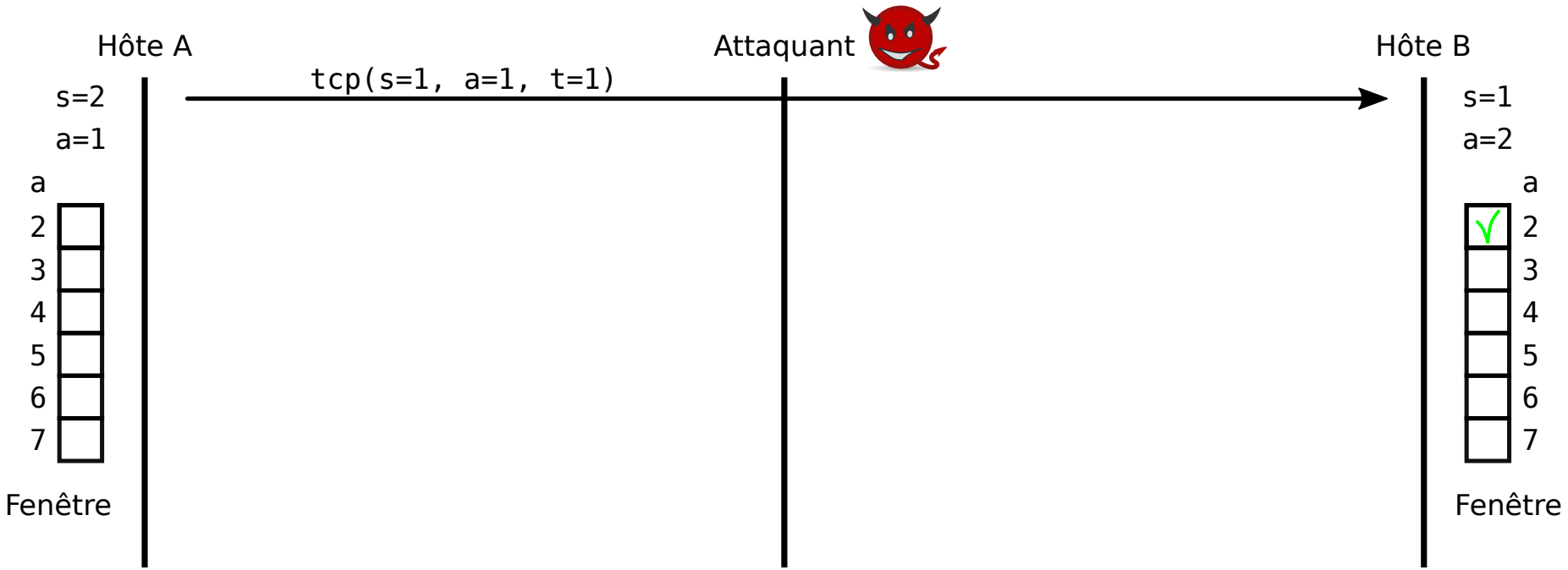
Attaques – *couche transport (4) - désynchronisation de session TCP*

Méthode : injection de segments *null data*
Exemple de donnée *null* Telnet : caractère `NOP` (`0x82`)



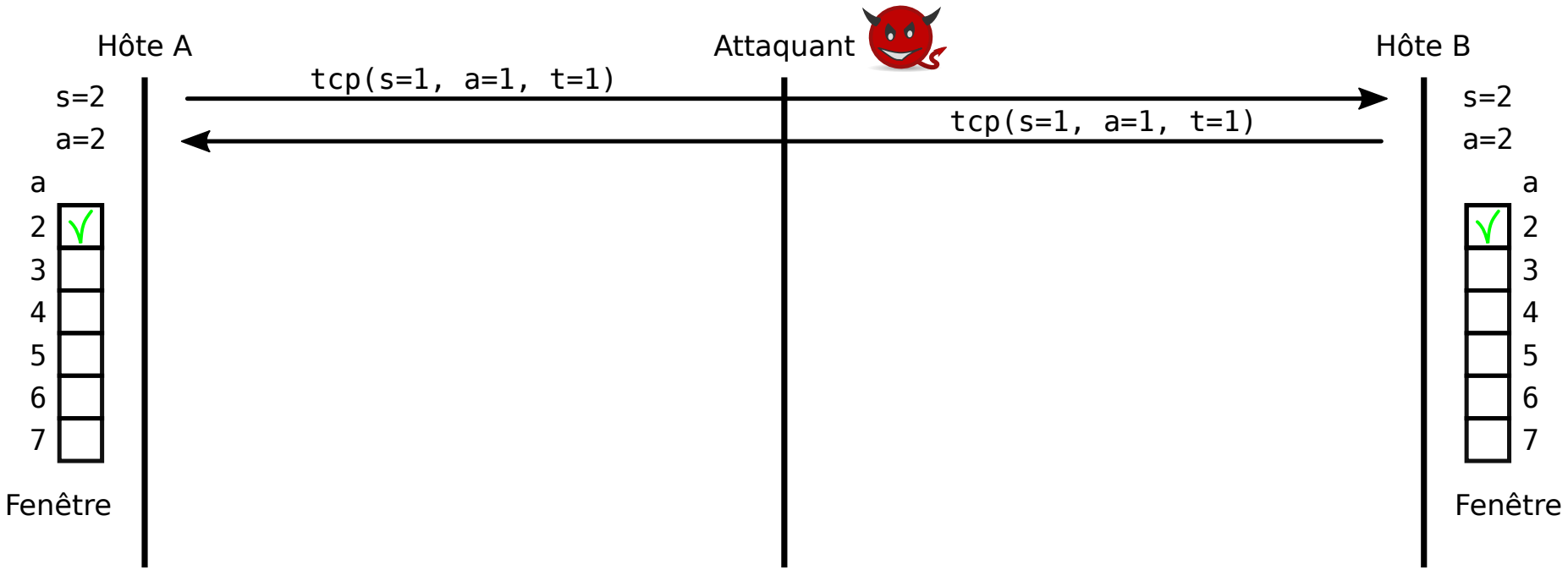
Attaques – *couche transport (4) - désynchronisation de session TCP*

Méthode : injection de segments *null data*
Exemple de donnée *null* Telnet : caractère `NOP` (`0x82`)



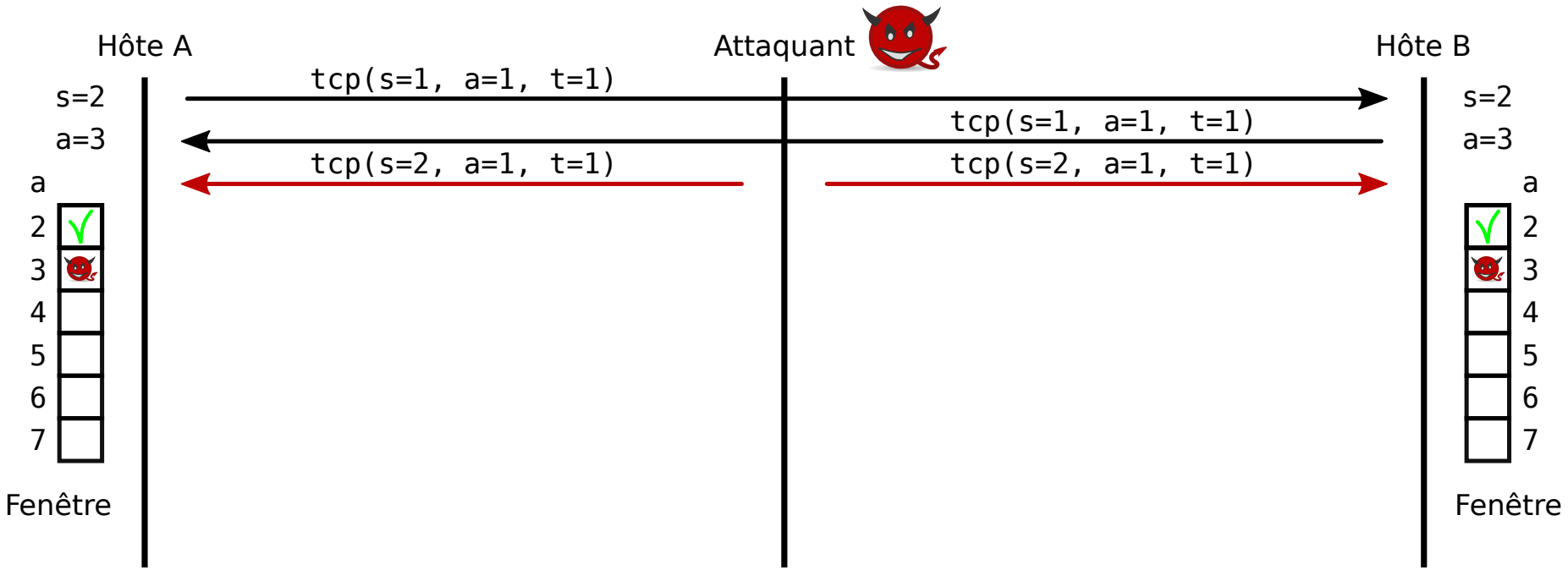
Attaques – *couche transport (4) - désynchronisation de session TCP*

Méthode : injection de segments *null data*
Exemple de donnée *null* Telnet : caractère `NOP` (`0x82`)



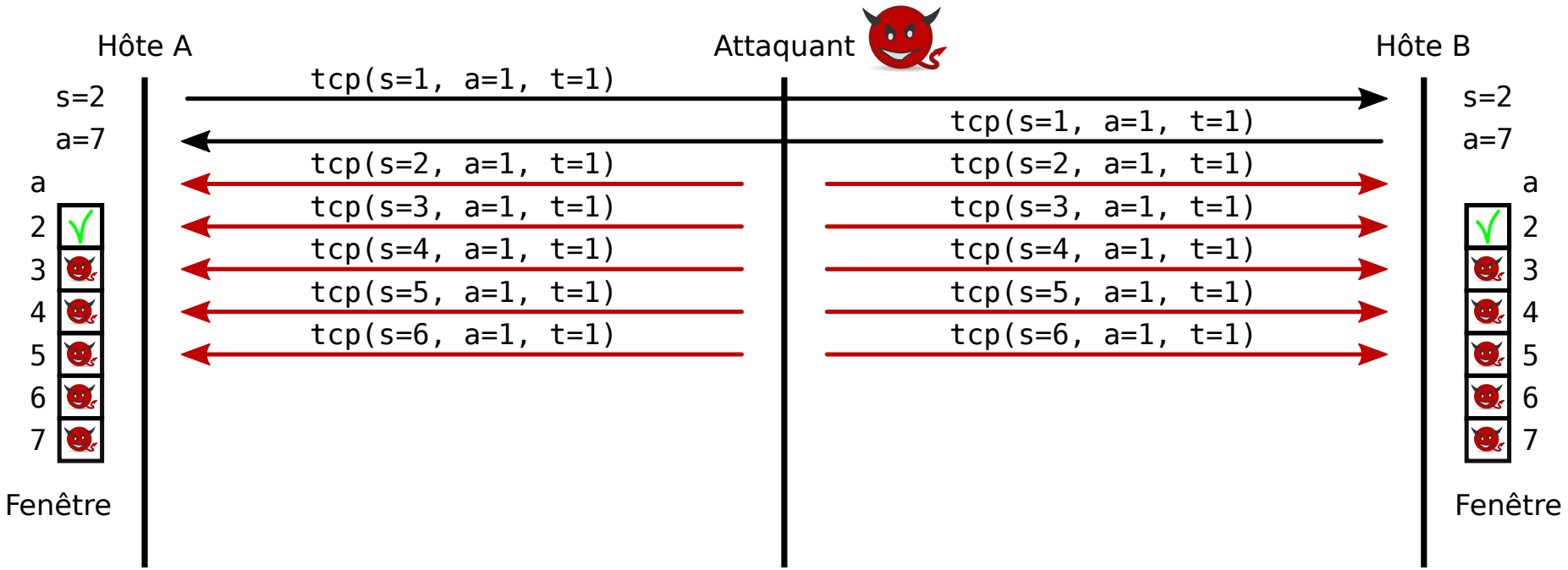
Attaques – *couche transport (4) - désynchronisation de session TCP*

Méthode : injection de segments *null data*
Exemple de donnée *null* Telnet : caractère `NOP` (`0x82`)



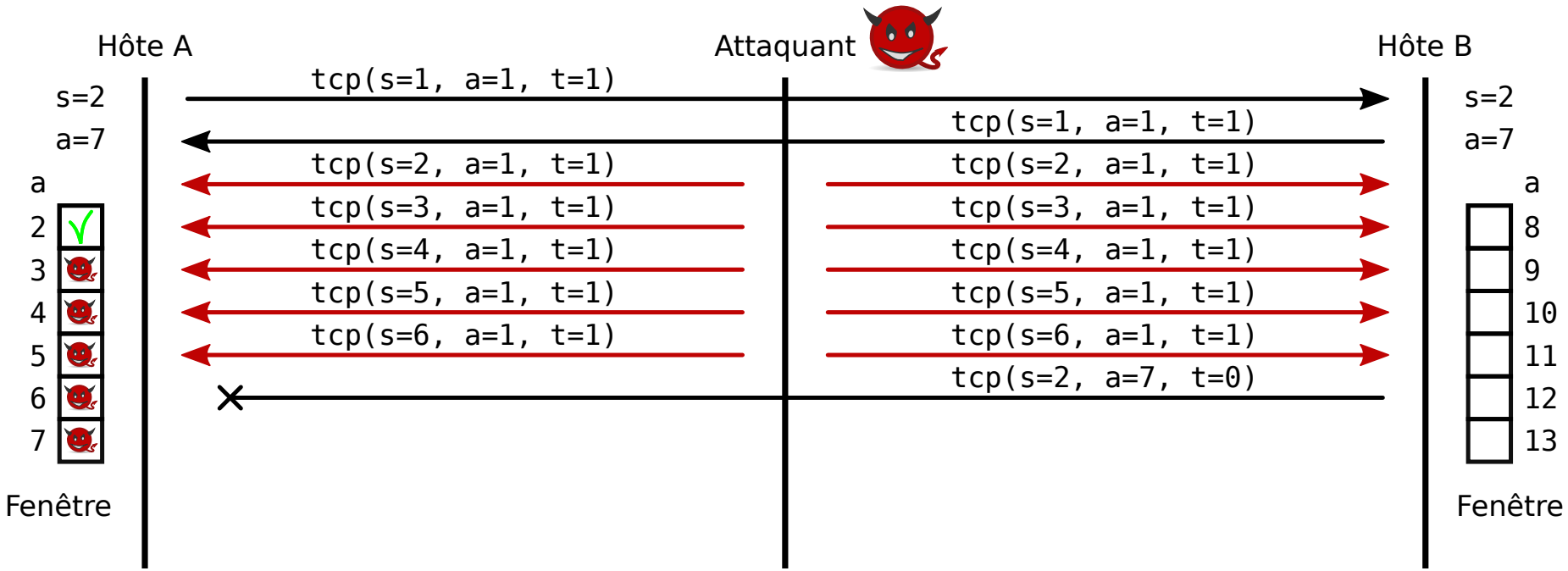
Attaques – *couche transport (4) - désynchronisation de session TCP*

Méthode : injection de segments *null data*
Exemple de donnée *null* Telnet : caractère `NOP` (`0x82`)



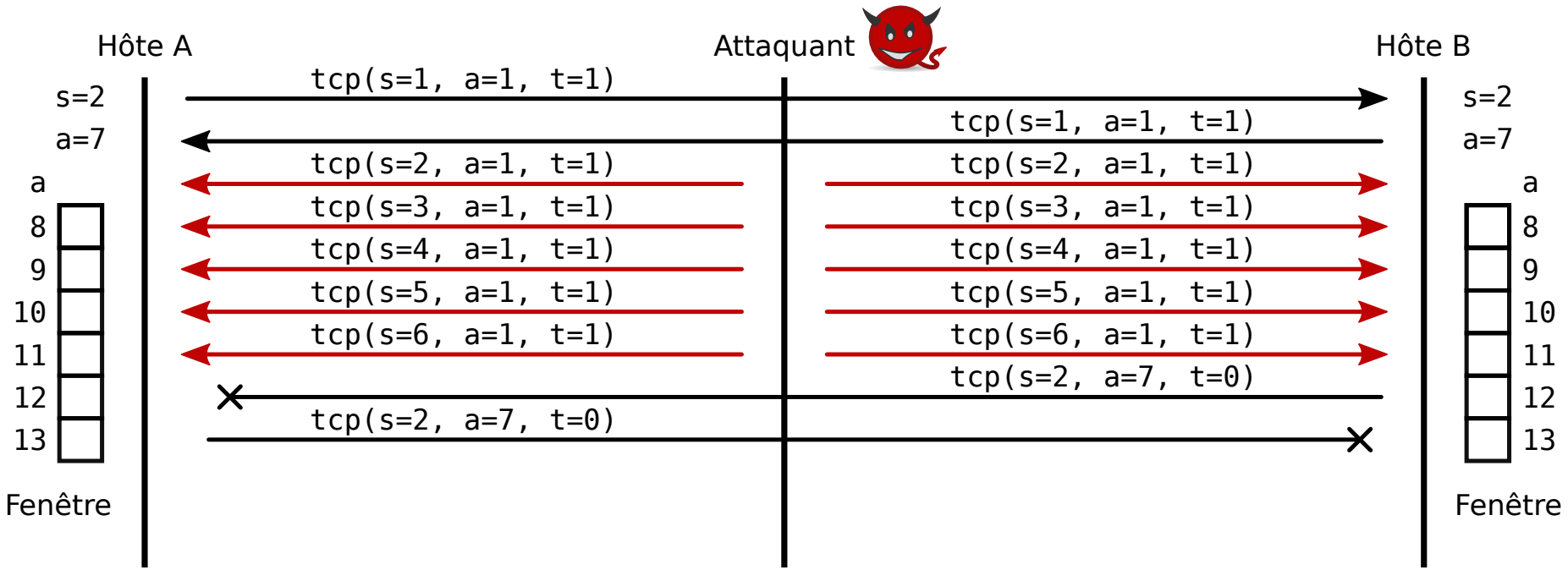
Attaques – *couche transport (4) - désynchronisation de session TCP*

Méthode : injection de segments *null data*
Exemple de donnée *null* Telnet : caractère `NOP` (`0x82`)



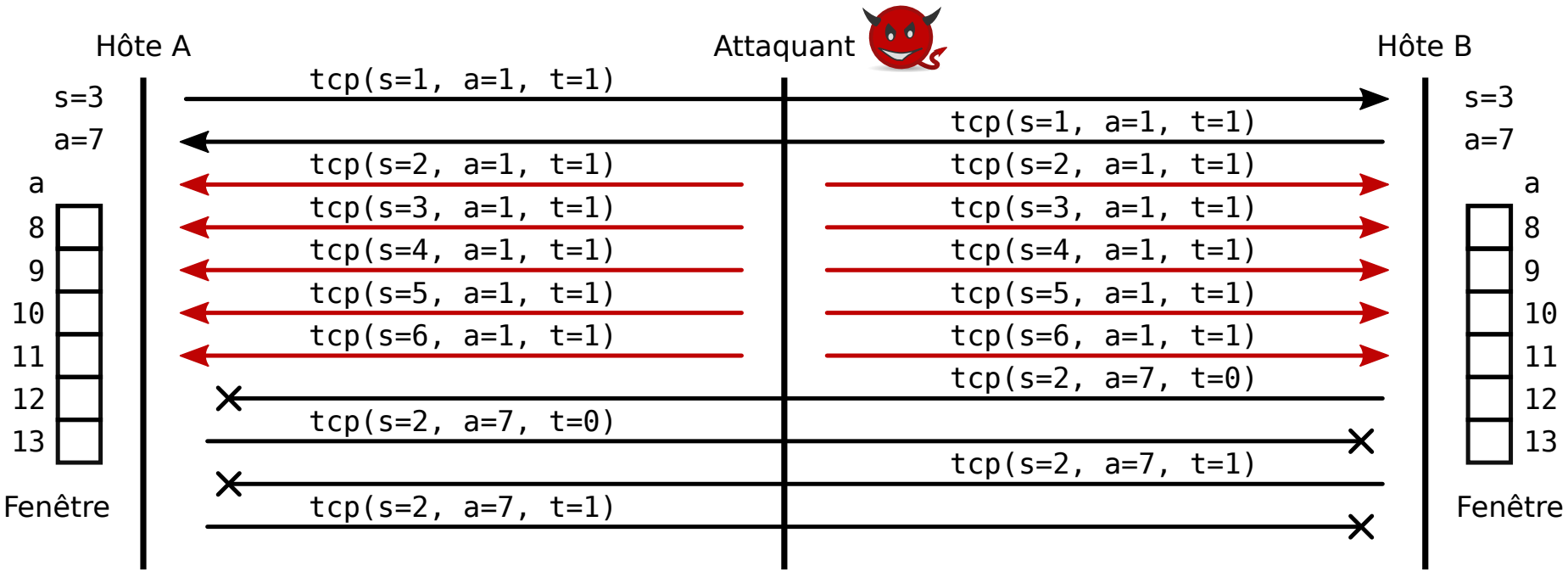
Attaques – *couche transport (4) - désynchronisation de session TCP*

Méthode : injection de segments *null data*
Exemple de donnée *null* Telnet : caractère `NOP` (`0x82`)



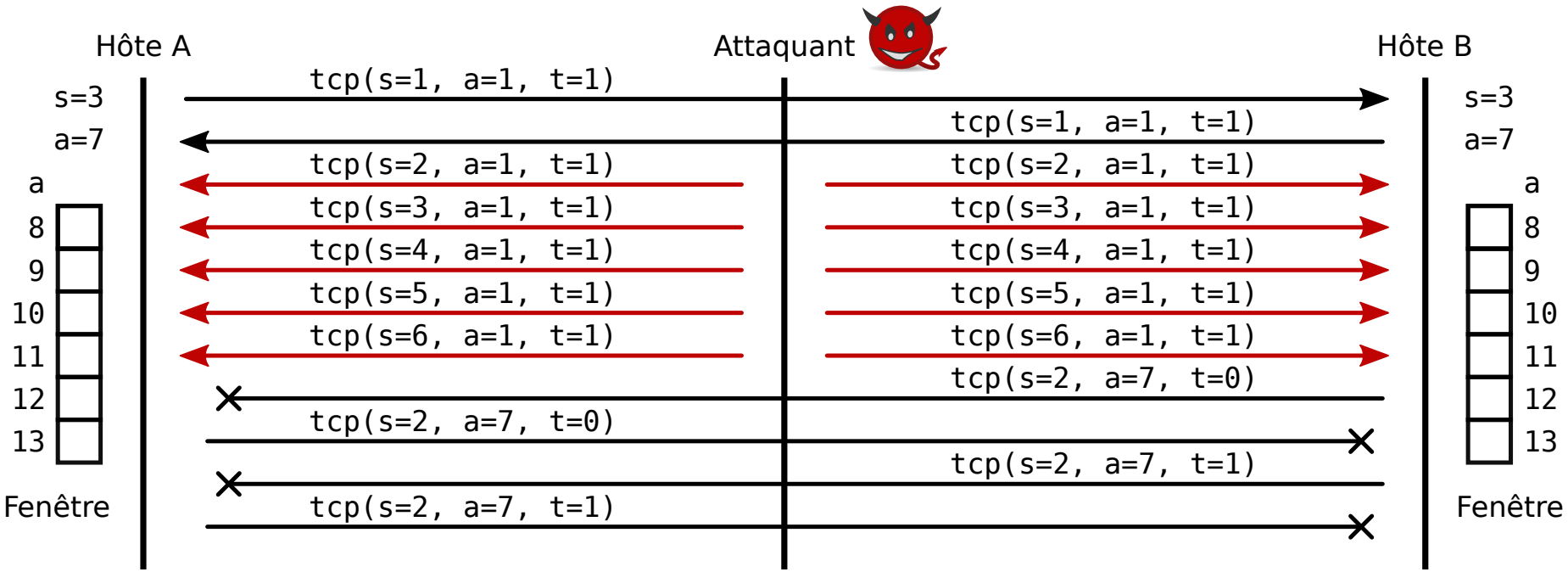
Attaques – *couche transport (4) - désynchronisation de session TCP*

Méthode : injection de segments *null data*
Exemple de donnée *null* Telnet : caractère `NOP` (`0x82`)



Attaques – *couche transport (4) - désynchronisation de session TCP*

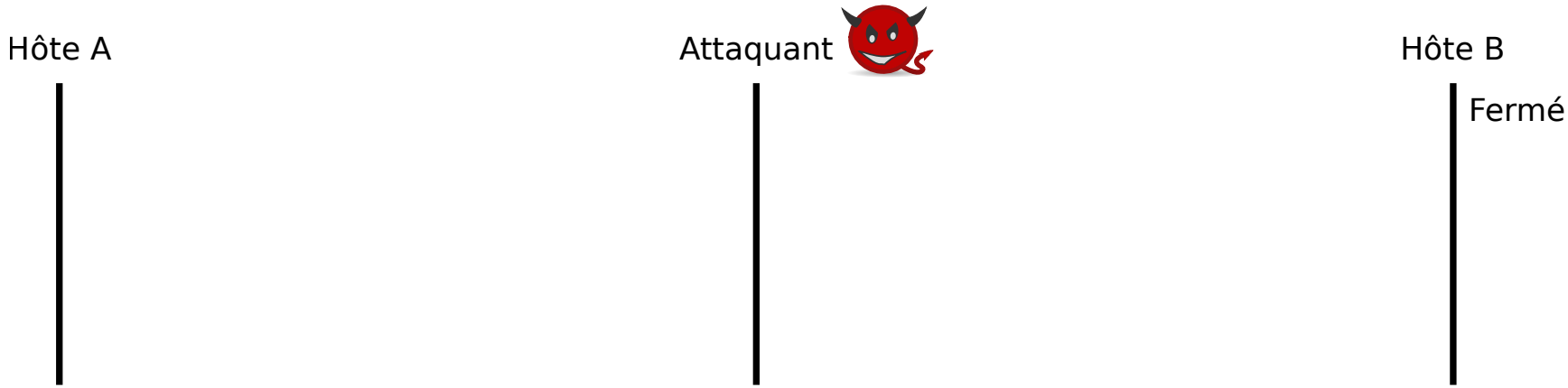
Méthode : injection de segments *null data*
Exemple de donnée *null* Telnet : caractère `NOP` (`0x82`)



Conséquence : Hôte A et hôte sont désormais désynchronisés de plus d'une fenêtre de réception. Ils ne peuvent plus s'échanger de segments.

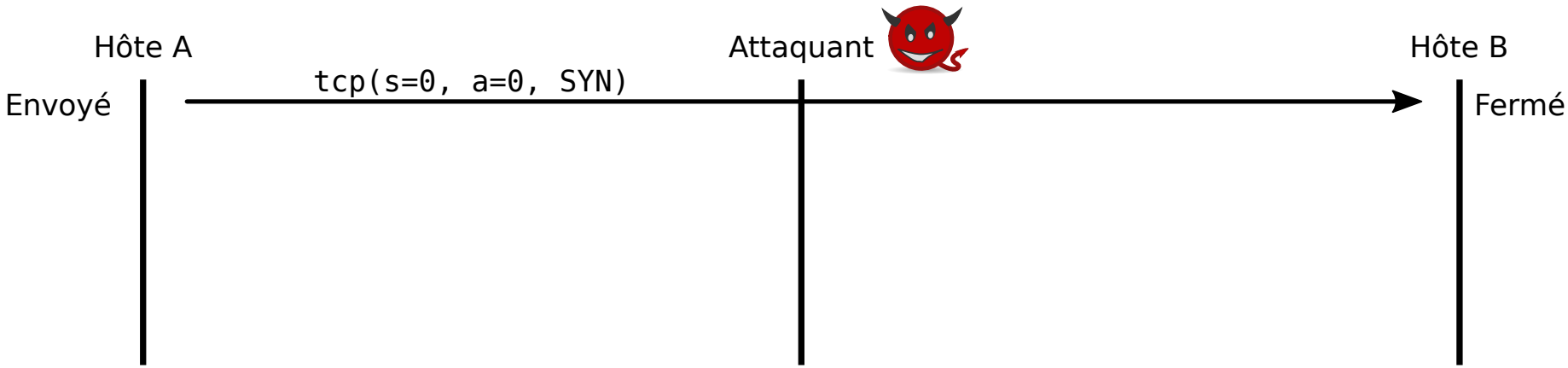
Attaques – *couche transport (4) - désynchronisation de session TCP*

Méthode : désynchronisation lors de la poignée de main



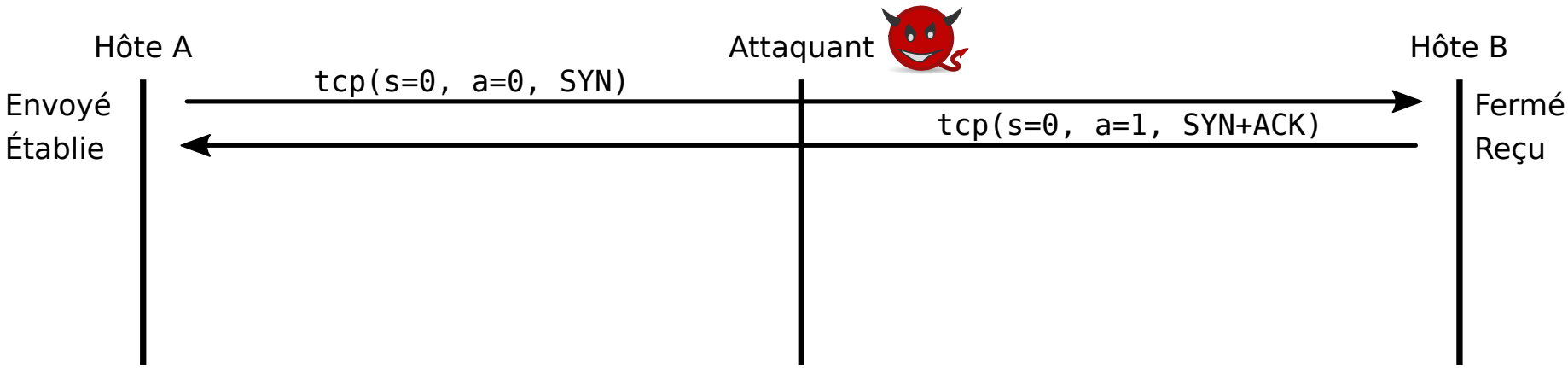
Attaques – *couche transport (4) - désynchronisation de session TCP*

Méthode : désynchronisation lors de la poignée de main



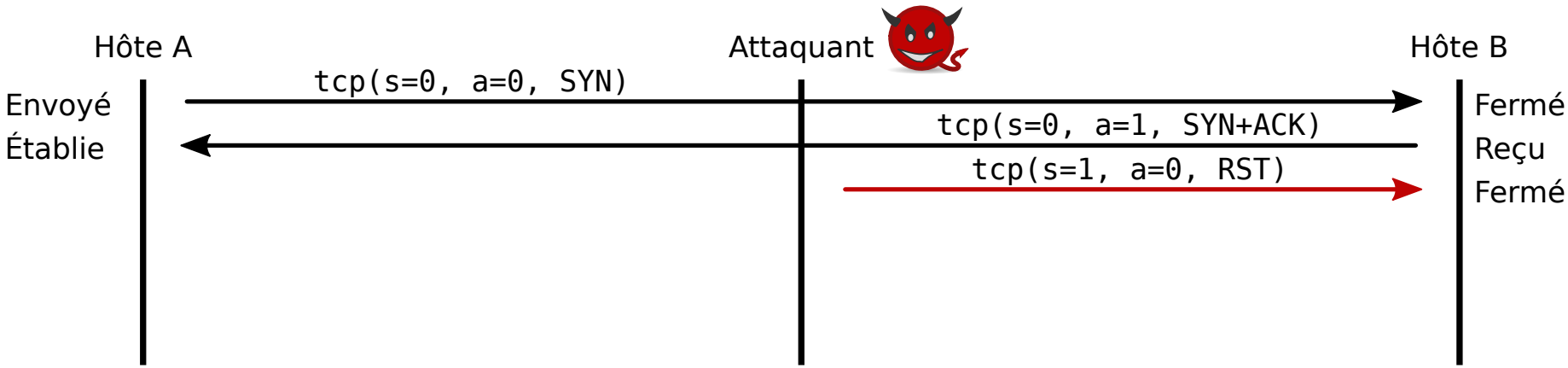
Attaques – *couche transport (4) - désynchronisation de session TCP*

Méthode : désynchronisation lors de la poignée de main



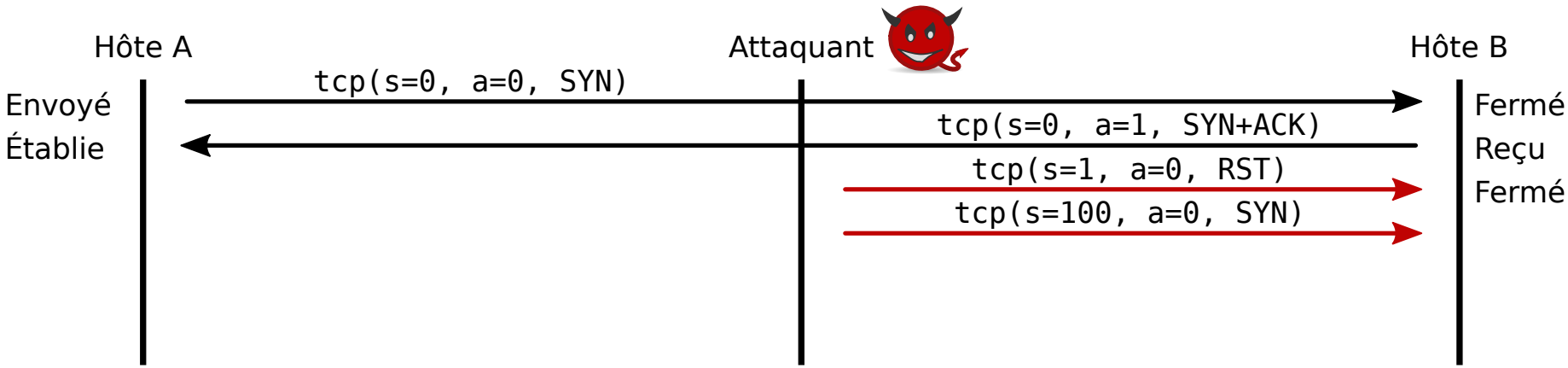
Attaques – *couche transport (4) - désynchronisation de session TCP*

Méthode : désynchronisation lors de la poignée de main



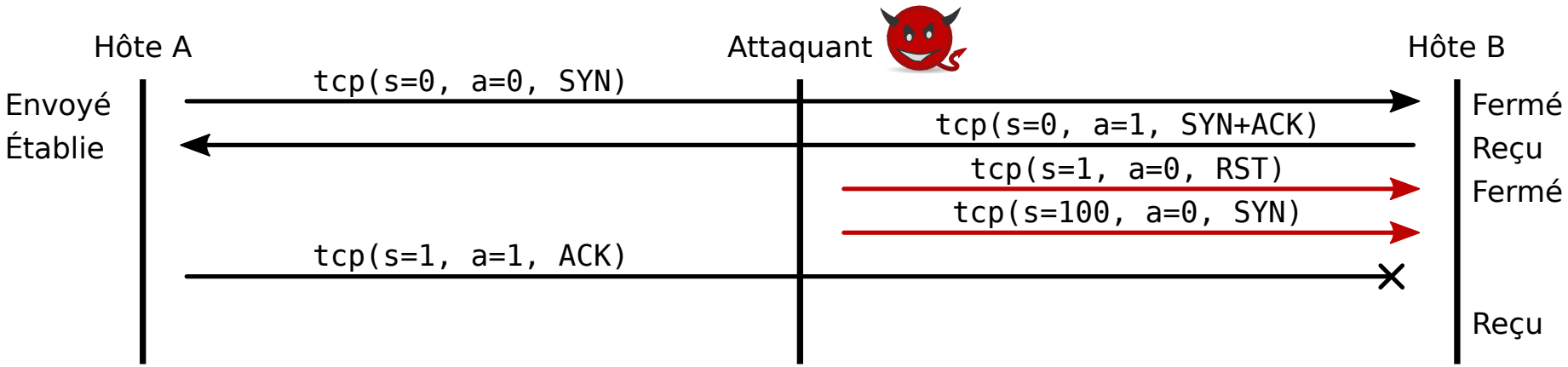
Attaques – *couche transport (4) - désynchronisation de session TCP*

Méthode : désynchronisation lors de la poignée de main



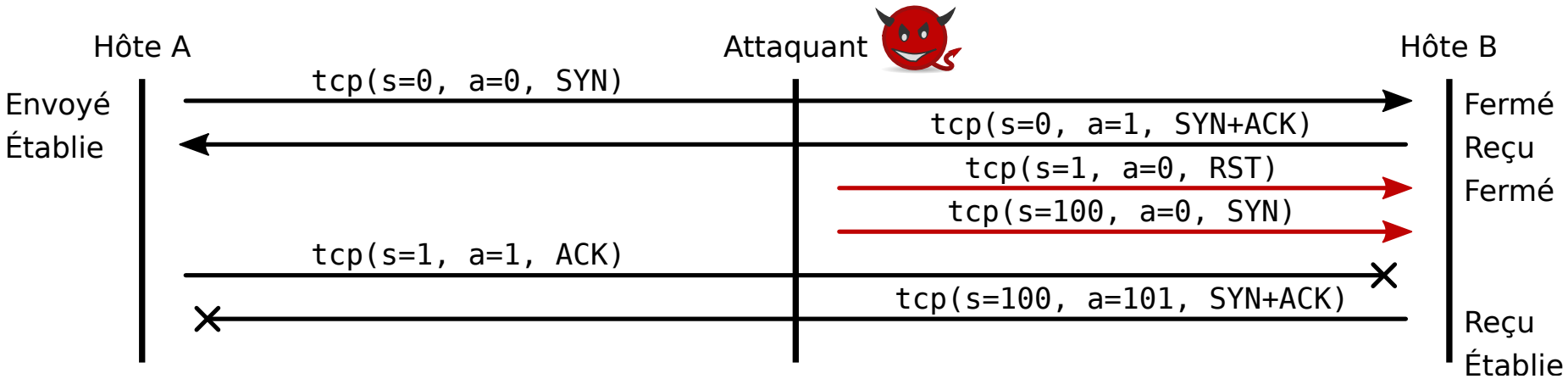
Attaques – *couche transport (4) - désynchronisation de session TCP*

Méthode : désynchronisation lors de la poignée de main



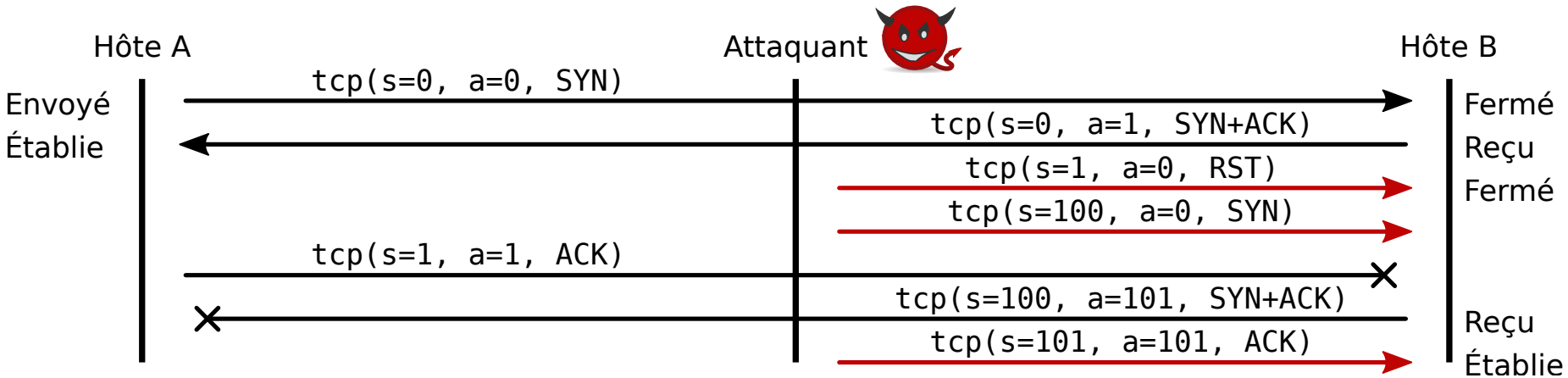
Attaques – *couche transport (4) - désynchronisation de session TCP*

Méthode : désynchronisation lors de la poignée de main



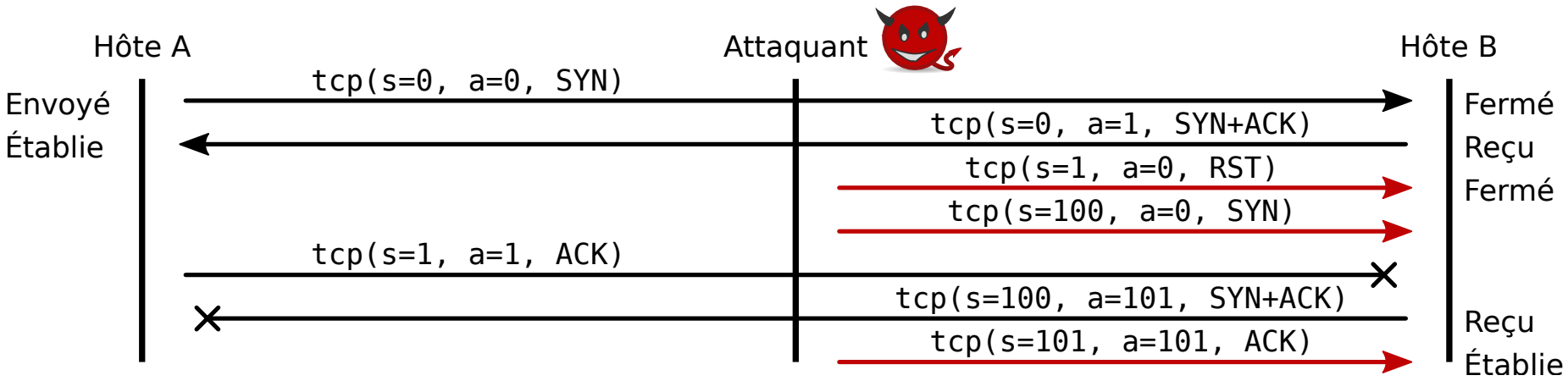
Attaques – *couche transport (4) - désynchronisation de session TCP*

Méthode : désynchronisation lors de la poignée de main



Attaques – *couche transport (4) - désynchronisation de session TCP*

Méthode : désynchronisation lors de la poignée de main



Conséquence : Hôte A et hôte B possèdent maintenant une session établie mais désynchronisée. Ils ne peuvent pas s'échanger de segments.

Attaques – *couche transport (4) - désynchronisation de session TCP*

Méthode : désynchronisation lors de la poignée de main

1. Hôte A envoie SYN à hôte B
2. B répond SYN + ACK à hôte A $\Rightarrow$ session établie A $\rightarrow$ B
3. Attaquant envoie RST à hôte B en se déguisant en hôte A
4. Attaquant envoie SYN à hôte B en se déguisant en hôte A
5. Hôte A envoie ACK à hôte B qui refuse le segment
6. Hôte B envoie SYN + ACK vers hôte A qui refuse le segment (provoqué par attaquant)
7. Attaquant envoie ACK à hôte B en se déguisant en hôte A $\Rightarrow$ session établie B $\rightarrow$ A

Attaques – *couche transport (4) - désynchronisation de session TCP*

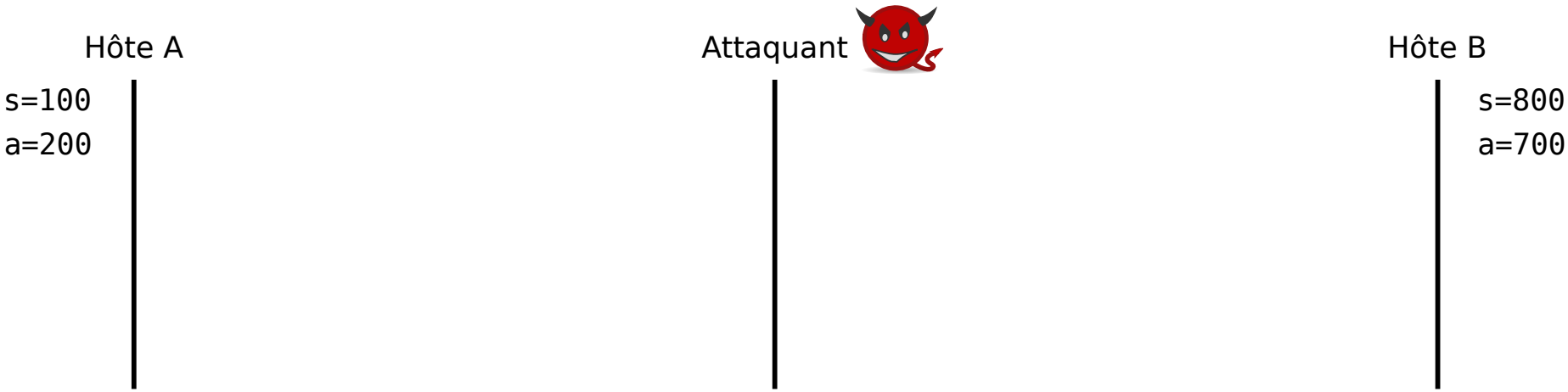
Méthode : désynchronisation lors de la poignée de main

1. Hôte A envoie SYN à hôte B
2. B répond SYN + ACK à hôte A $\Rightarrow$ session établie A $\rightarrow$ B
3. Attaquant envoie RST à hôte B en se déguisant en hôte A
4. Attaquant envoie SYN à hôte B en se déguisant en hôte A
5. Hôte A envoie ACK à hôte B qui refuse le segment
6. Hôte B envoie SYN + ACK vers hôte A qui refuse le segment (provoqué par attaquant)
7. Attaquant envoie ACK à hôte B en se déguisant en hôte A $\Rightarrow$ session établie B $\rightarrow$ A

Conséquence : session désynchronisée établie A $\leftrightarrow$ B

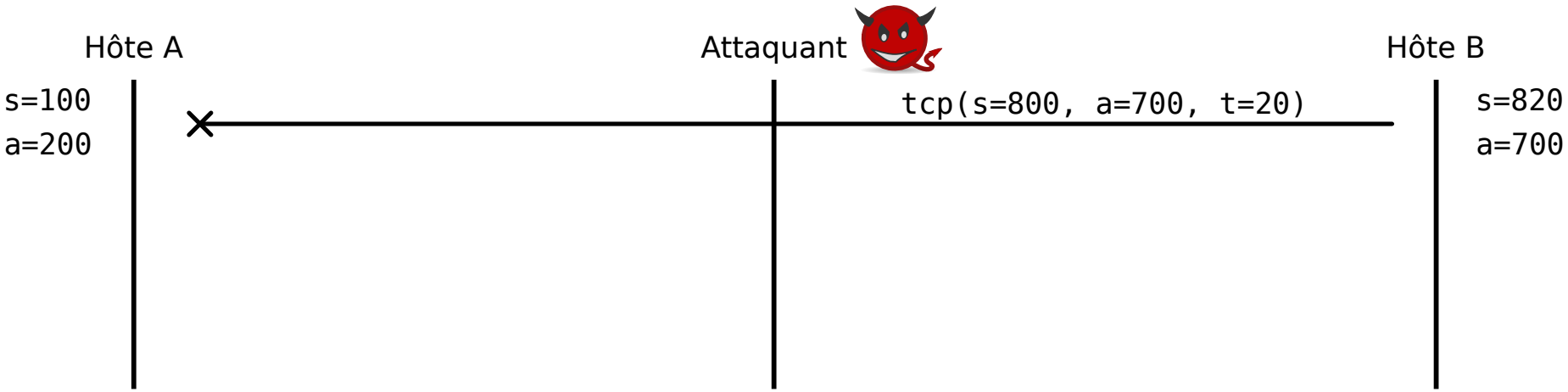
Attaques – *couche transport (4) - homme dans le milieu désynchronisé*

Méthode : Désynchroisation de session TCP et interaction curieuse avec les hôtes désynchronisés



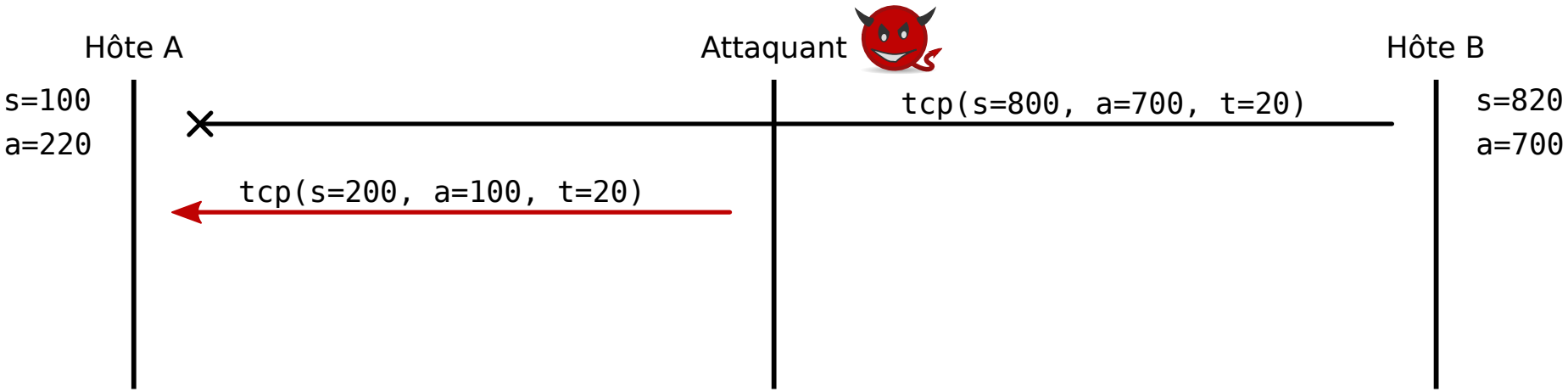
Attaques – *couche transport (4) - homme dans le milieu désynchronisé*

Méthode : Désynchroisation de session TCP et interaction curieuse avec les hôtes désynchronisés



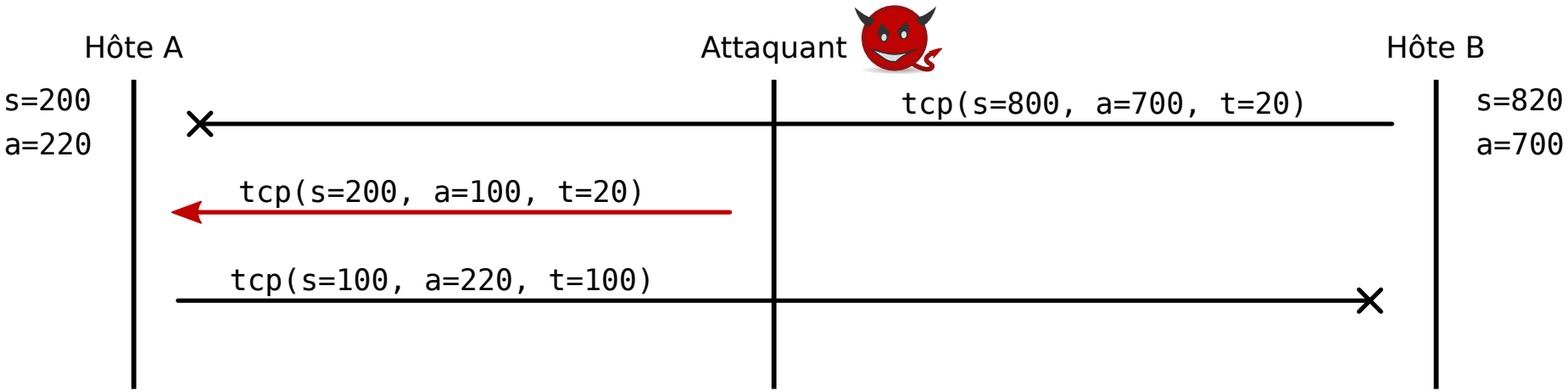
Attaques – *couche transport (4) - homme dans le milieu désynchronisé*

Méthode : Désynchroisation de session TCP et interaction curieuse avec les hôtes désynchronisés



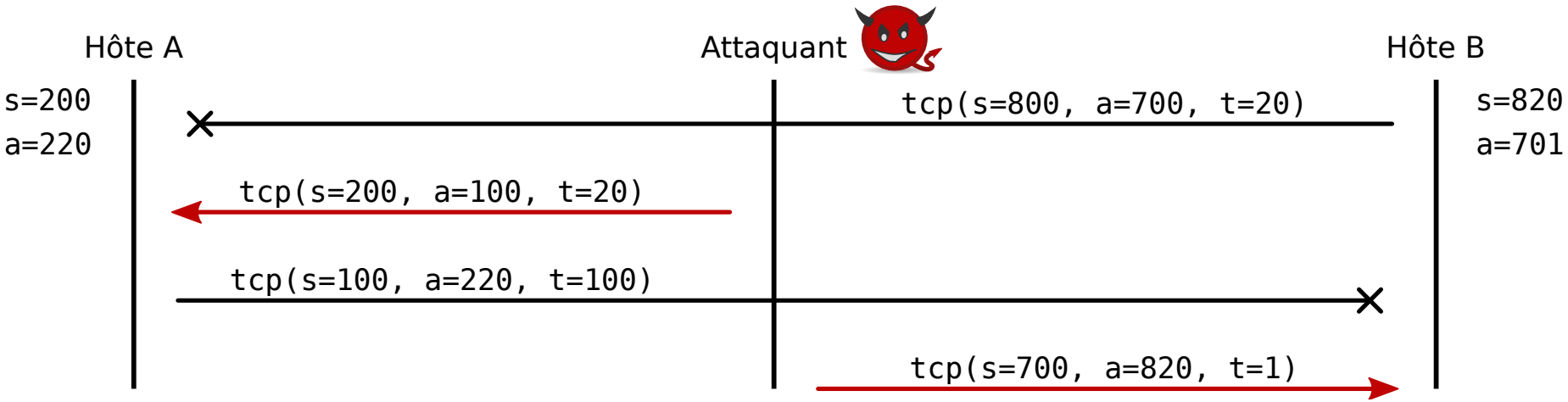
Attaques – *couche transport (4) - homme dans le milieu désynchronisé*

Méthode : Désynchroisation de session TCP et interaction curieuse avec les hôtes désynchronisés



Attaques – *couche transport (4) - homme dans le milieu désynchronisé*

Méthode : Désynchroisation de session TCP et interaction curieuse avec les hôtes désynchronisés



Attaques – *couche transport (4) - homme dans le milieu désynchronisé*

Méthode : Désynchroisation de session TCP et interaction curieuse avec les hôtes désynchronisés

1. La session entre les hôtes A et B est désynchronisée
2. Attaquant connait les paires numéros de séquence / acquittement
3. Il sniffe le trafic A $\rightarrow$ B refusé par B et inversement
4. Il relaie le trafic en se déguisant en source A ou B et en utilisant les bons numéros de séquence et d'acquittement. Il peut modifier le contenu des messages à sa convenance

Plan du cours

Filtrage – *introduction*

Modèle de menaces : On observe une augmentation des tentatives d'intrusions réseau

$\Rightarrow$ Construction d'architectures sécurité réseau

- Définition de domaines aux différents niveau de privilèges
- Filtrage des communications inter domaines.

Objectif d'une architectures de sécurité réseau :

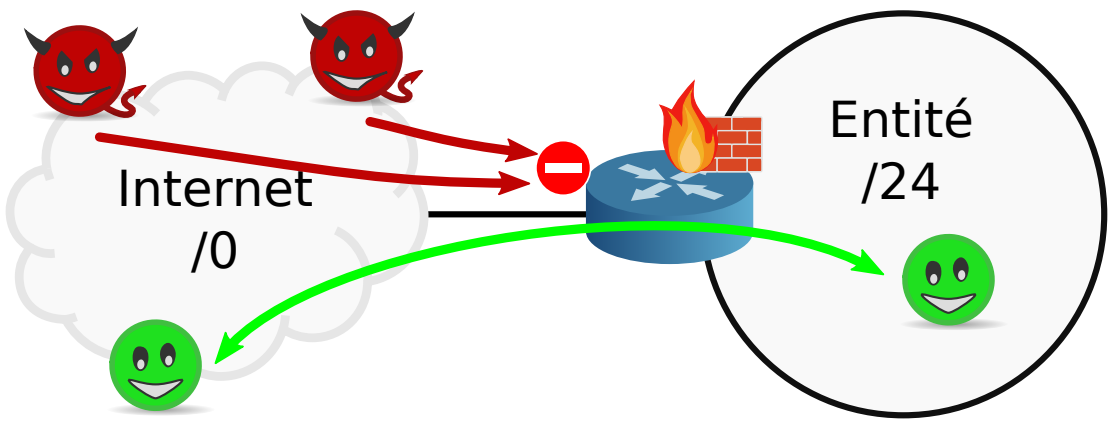
- Contrôler les flux entrant et sortant des domaines réseau
- Limiter la visibilité de la structure réseau d'une entité

$\rightarrow$ Conception du pare-feu

Filtrage – *introduction*

Principe de base des pare-feu

- Contrôle des flux réseau à l'aide de fonctions de filtrage (et parfois de traduction)
- Domaines réseau IP classiques :
  - Interne et de confiance
  - Externe, auquel on ne peut faire confiance (Internet)



Filtrage – *introduction*

Objectif de sécurité d'un pare-feu

- Filtrage configurable avec le "langage" du trafic légitime
- Uniquement le trafic légitime est relayé
- Le pare-feu analyse les flux réseau inter domaines
  → positionnement stratégique
- Limiter l'impact sur les performances
  → matériel dédié et filtrage au bon niveau

Catégories de pare-feu :

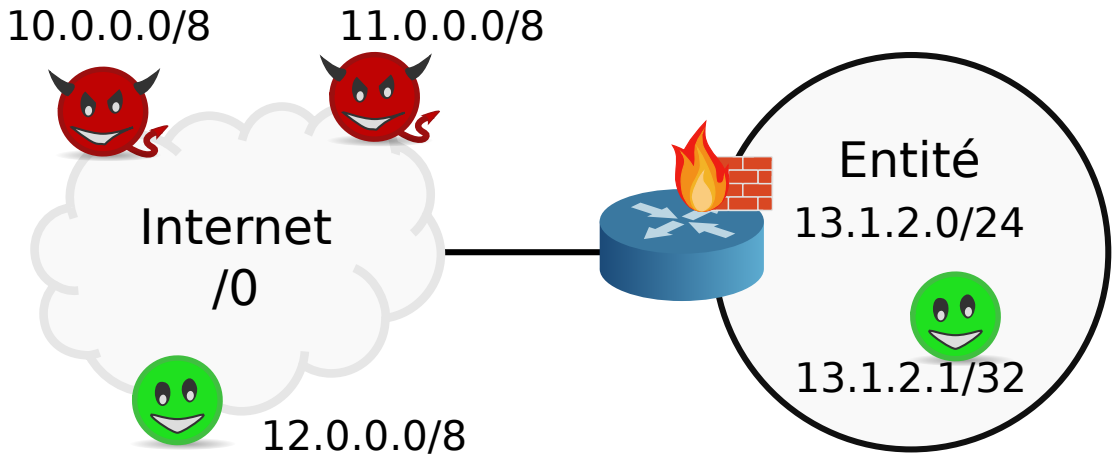
- Pare-feu (filtres réseau, *packet filter*) : routeur de sécurité capable de filtrer le trafic aux niveau OSI réseau et transport
- Serveur mandataires (proxies) : peuvent souvent filtrer le trafic jusqu'à la couche application. Mode de fonctionnement explicite classique ou transparent (bastion ou pare-feu applicatif)

Filtrage – *filtrage réseau*

Principe du filtrage réseau

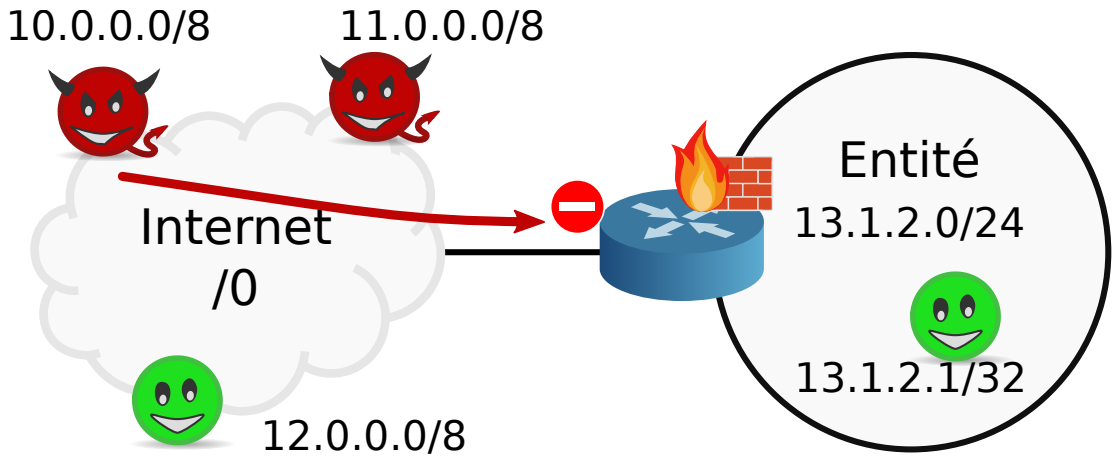
- Contrôle du flux réseau en fonction de règles de filtrage
- Le filtrage peut-être exécuté par du matériel spécifique ou des machines généralistes configurée avec plusieurs contrôleurs réseau
- Les règles de filtrage sont s'appliquent au moins sur les champs protocolaires de réseau et transport
  Par exemple : réseau IP et transport TCP

Exemple de règles de filtrage



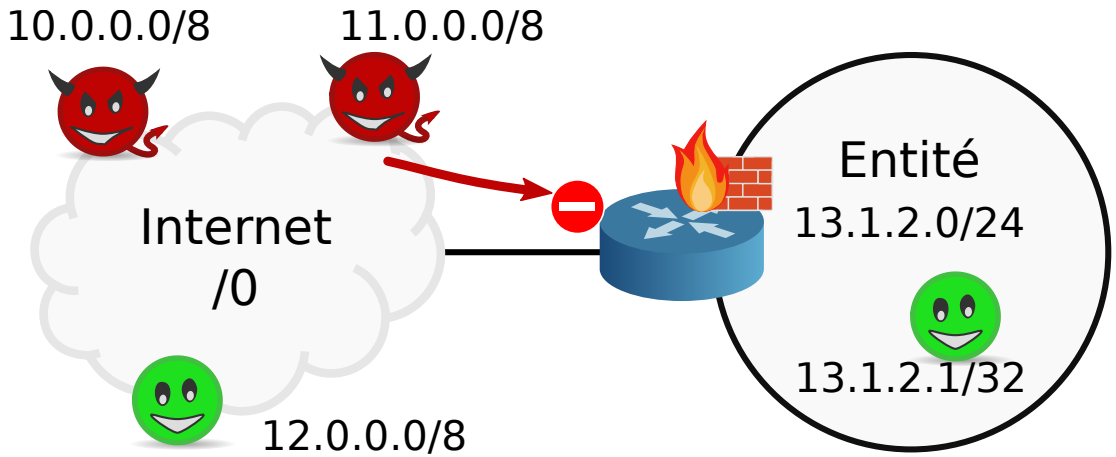
10.0.0.0/8	0.0.0.0	*	*	rejeter
11.0.0.0/8	0.0.0.0	*	*	rejeter
12.0.0.0/8	13.1.2.1/32	*	80	accepter
13.1.2.0/24	12.0.0.0/8	80	*	accepter

Exemple de règles de filtrage



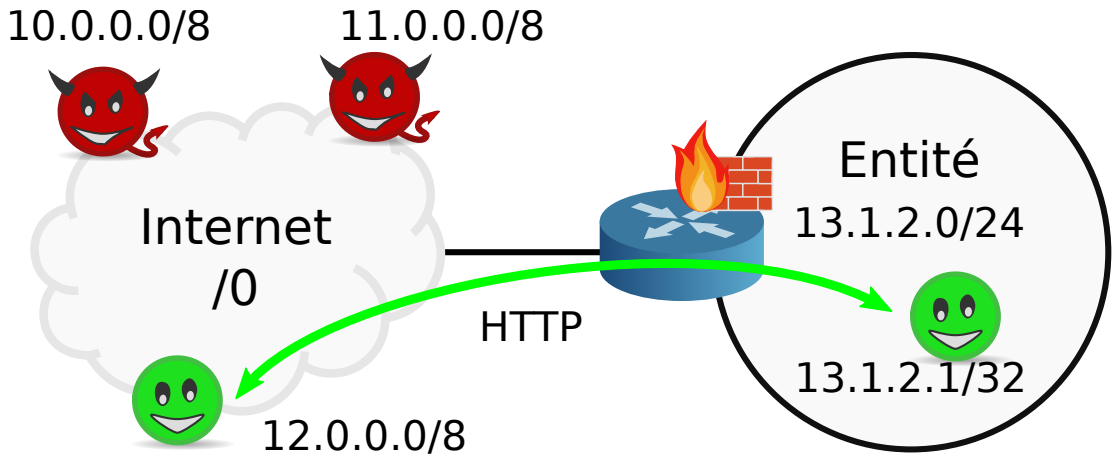
10.0.0.0/8	0.0.0.0	*	*	rejeter
11.0.0.0/8	0.0.0.0	*	*	rejeter
12.0.0.0/8	13.1.2.1/32	*	80	accepter
13.1.2.0/24	12.0.0.0/8	80	*	accepter

Exemple de règles de filtrage



10.0.0.0/8	0.0.0.0	*	*	rejeter
11.0.0.0/8	0.0.0.0	*	*	rejeter
12.0.0.0/8	13.1.2.1/32	*	80	accepter
13.1.2.0/24	12.0.0.0/8	80	*	accepter

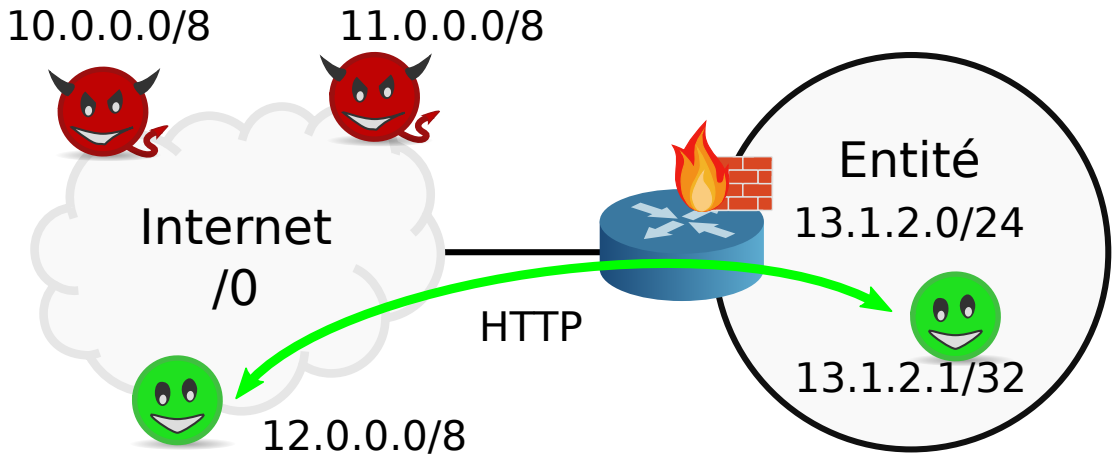
Exemple de règles de filtrage



10.0.0.0/8	0.0.0.0	*	*	rejeter
11.0.0.0/8	0.0.0.0	*	*	rejeter
12.0.0.0/8	13.1.2.1/32	*	80	accepter
13.1.2.0/24	12.0.0.0/8	80	*	accepter

Filtrage – *filtrage réseau*

Exemple de règles de filtrage



10.0.0.0/8	0.0.0.0	*	*	rejeter
11.0.0.0/8	0.0.0.0	*	*	rejeter
12.0.0.0/8	13.1.2.1/32	*	80	accepter
13.1.2.0/24	12.0.0.0/8	80	*	accepter

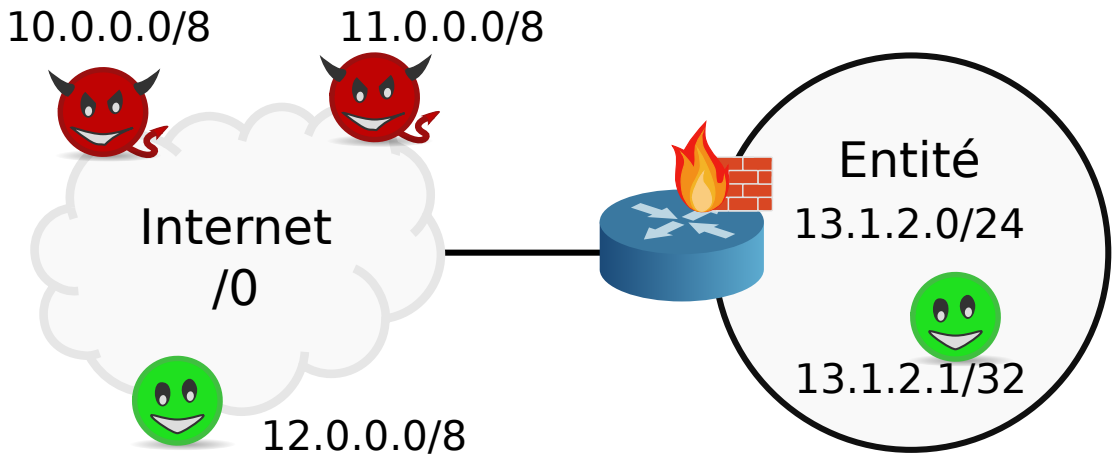
Inconvénient majeur : lister les règles d'interdiction de façon exhaustive...

Filtrage – *filtrage réseau*

Règle par défaut (politique)

- Comportement à adopter (*accepter* / *rejeter*) si aucune règle ne s'applique
- Deux politiques s'opposent :
  - Rejet par défaut → sécurité par défaut
    Exemple : banques, armée, ...
  - Acceptation par défaut → sûreté par défaut Exemple : systèmes critiques, temps réels, ...
- Dans la majorité systèmes d'informations non critiques on adopte en grande majorité le rejet et donc la **sécurité par défaut**
  Exemple : box de fournisseur d'accès à Internet (modem / routeur / switch personnel)

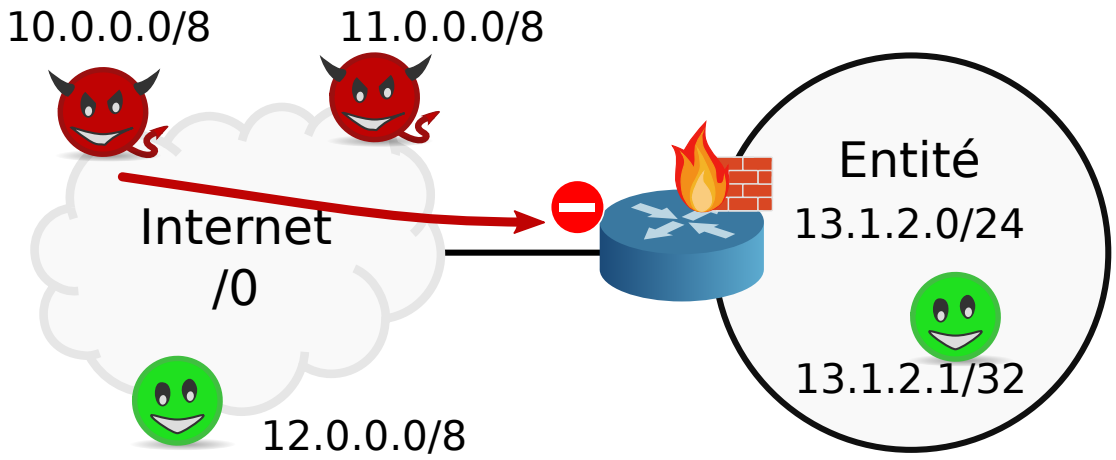
Exemple de filtrage avec règle par défaut (politique)



12.0.0.0/8	13.1.2.1/32	*	80	accepter
13.1.2.0/24	12.0.0.0/8	80	*	accepter
Règle par défaut				rejeter

Filtrage – *filtrage réseau*

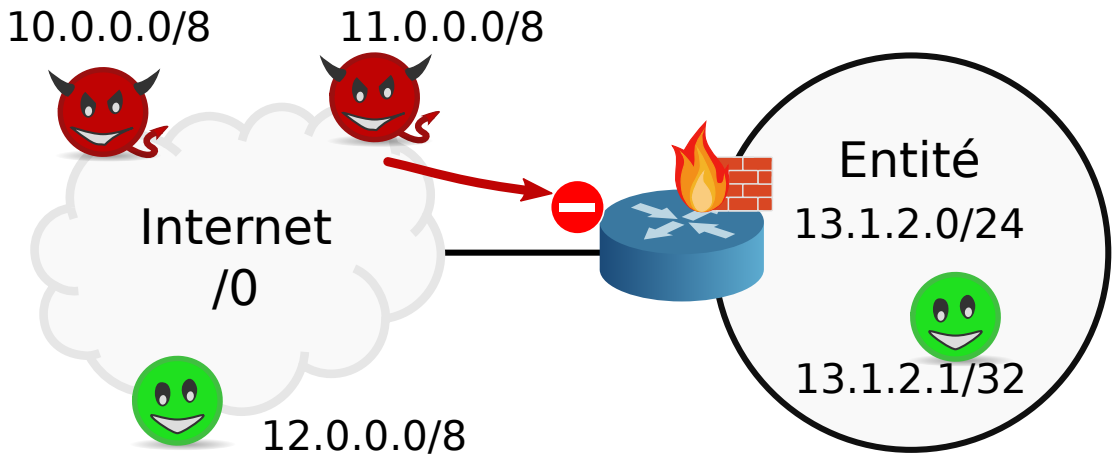
Exemple de filtrage avec règle par défaut (politique)



12.0.0.0/8	13.1.2.1/32	*	80	accepter
13.1.2.0/24	12.0.0.0/8	80	*	accepter
Règle par défaut				rejeter

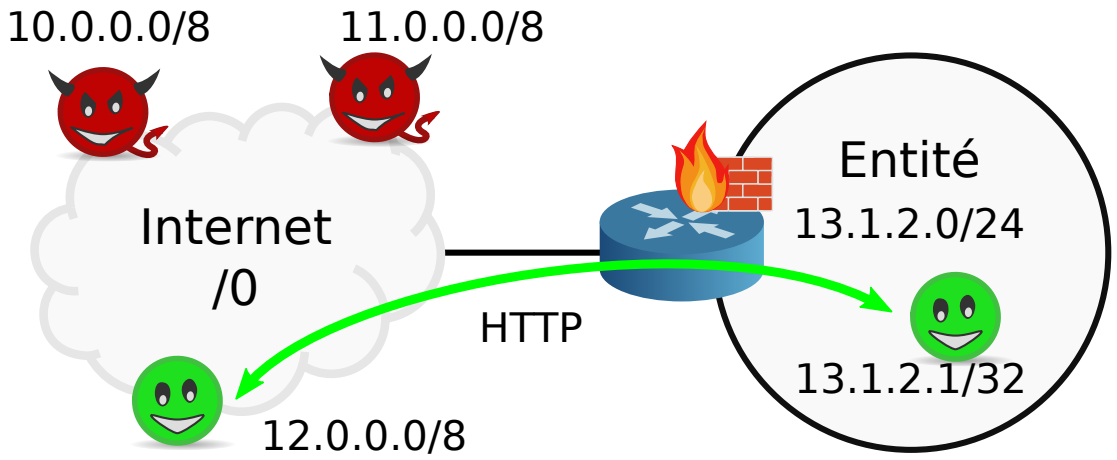
Filtrage – *filtrage réseau*

Exemple de filtrage avec règle par défaut (politique)



12.0.0.0/8	13.1.2.1/32	*	80	accepter
13.1.2.0/24	12.0.0.0/8	80	*	accepter
Règle par défaut				rejeter

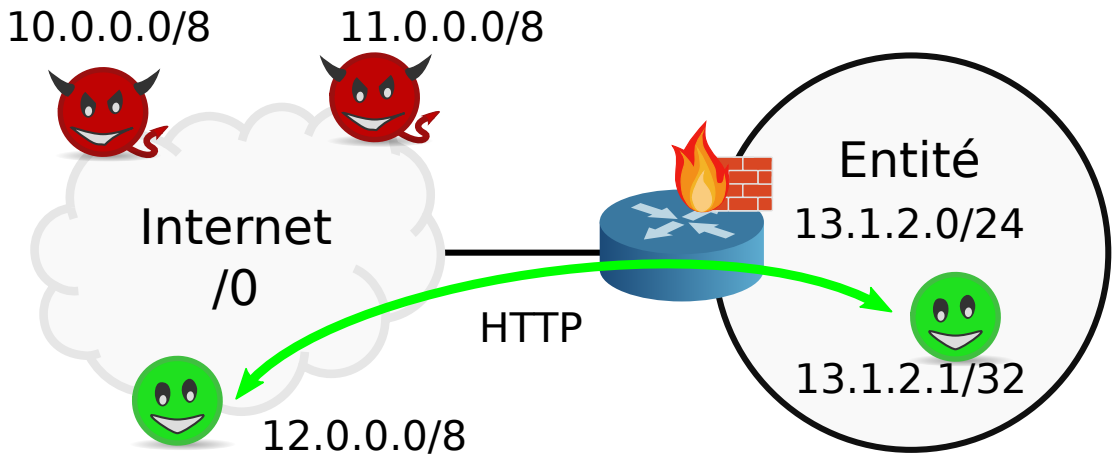
Exemple de filtrage avec règle par défaut (politique)



12.0.0.0/8	13.1.2.1/32	*	80	accepter
13.1.2.0/24	12.0.0.0/8	80	*	accepter
Règle par défaut				rejeter

Filtrage – *filtrage réseau*

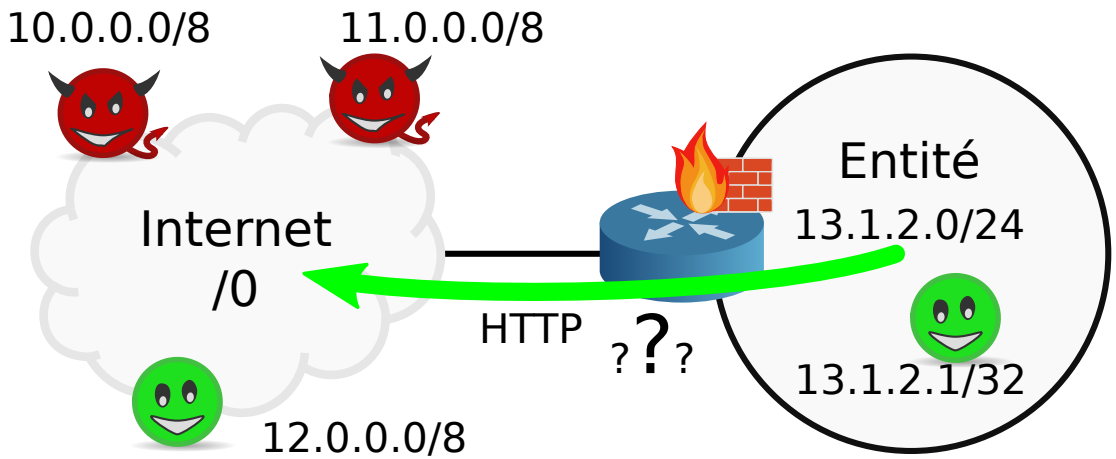
Exemple de filtrage avec règle par défaut (politique)



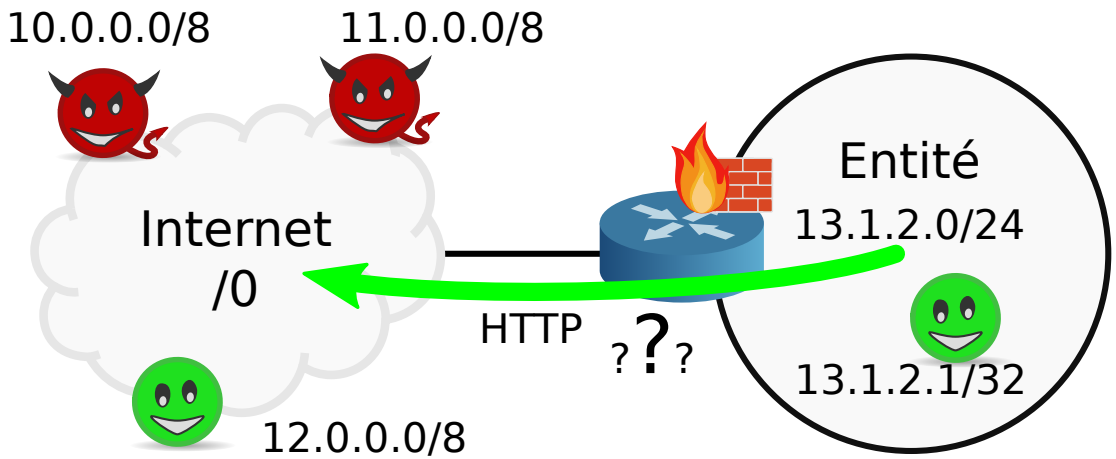
12.0.0.0/8	13.1.2.1/32	*	80	accepter
13.1.2.0/24	12.0.0.0/8	80	*	accepter
Règle par défaut				rejeter

Avantages majeurs : sécurité par défaut, même en cas de mauvaise configuration. Meilleures performances ⇒ toujours utiliser une règle par défaut !

Filtrage – *filtrage réseau*

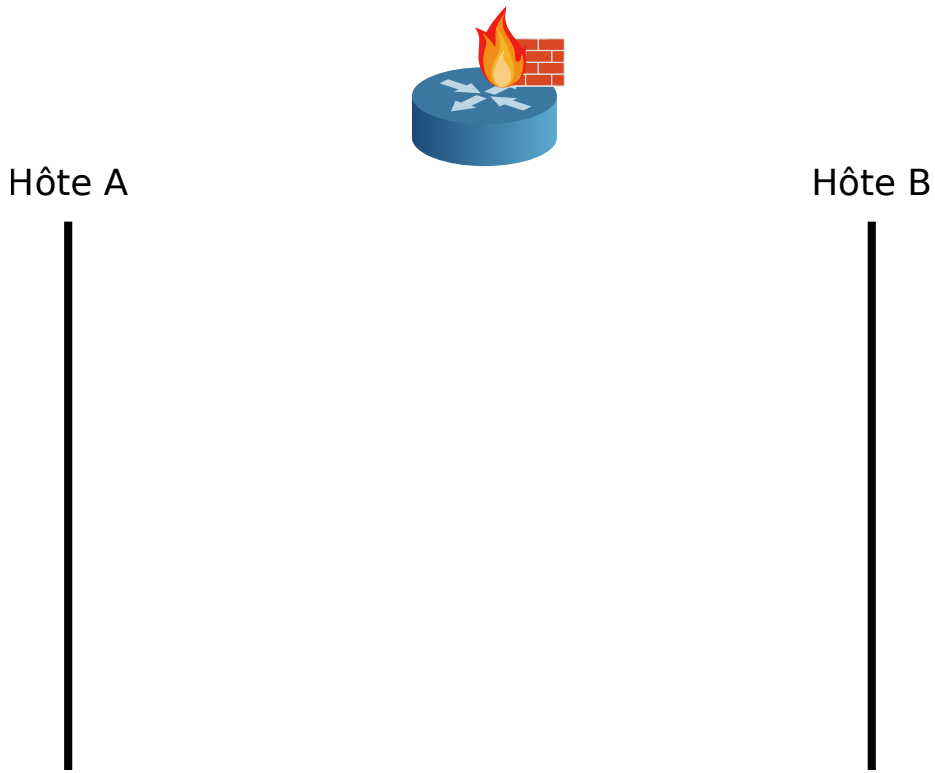


Filtrage – *filtrage réseau*

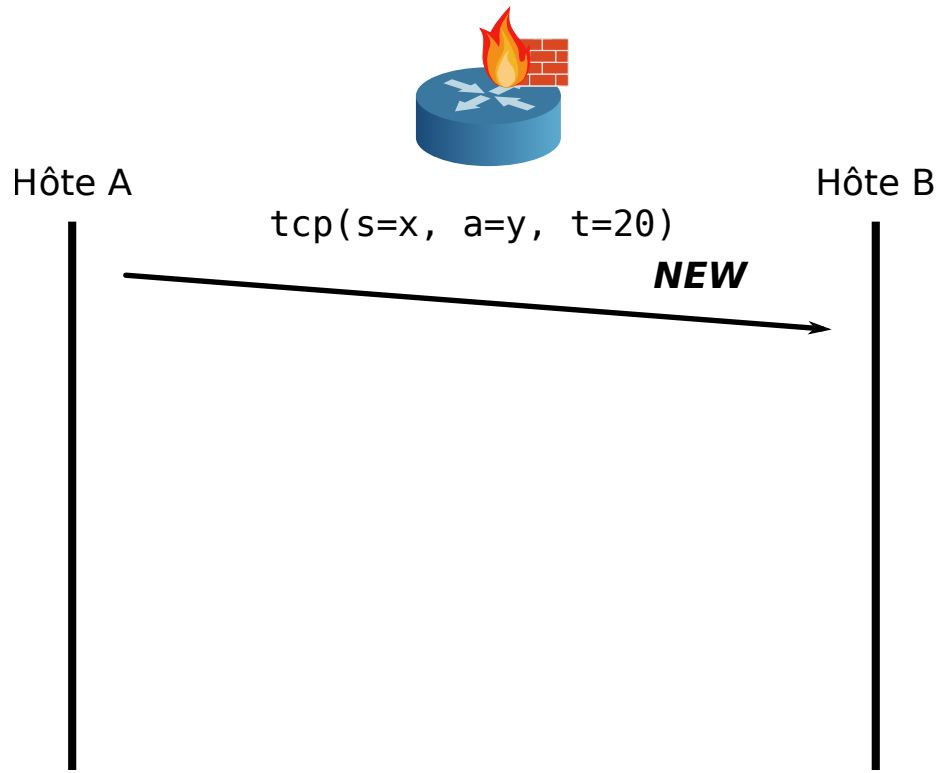


Pare-feu avec suivi d'état (*stateful*)
Méthode : suivi fin des interactions protocolaires niveau réseau à application pour filtrer plus finement → sens de connexion

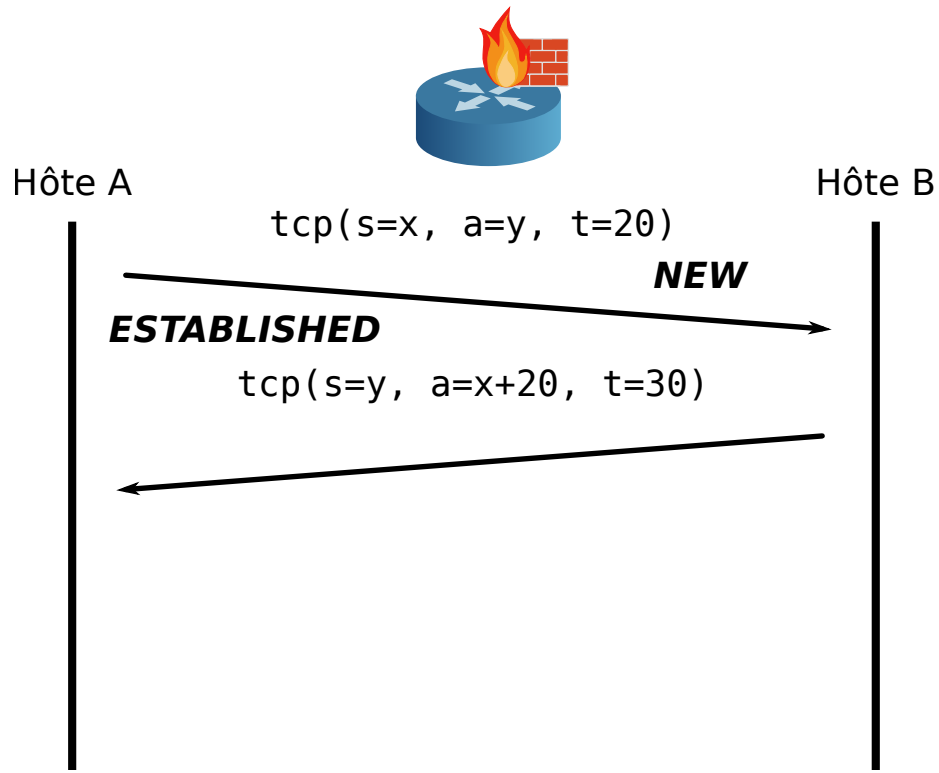
Exemple de suivi de session TCP



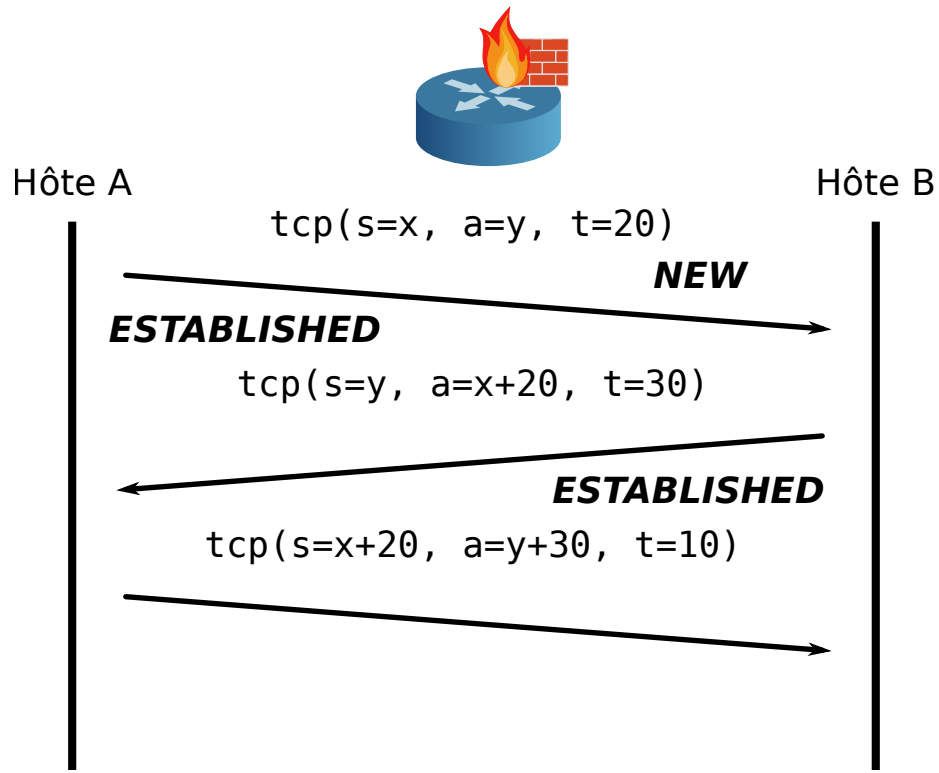
Exemple de suivi de session TCP



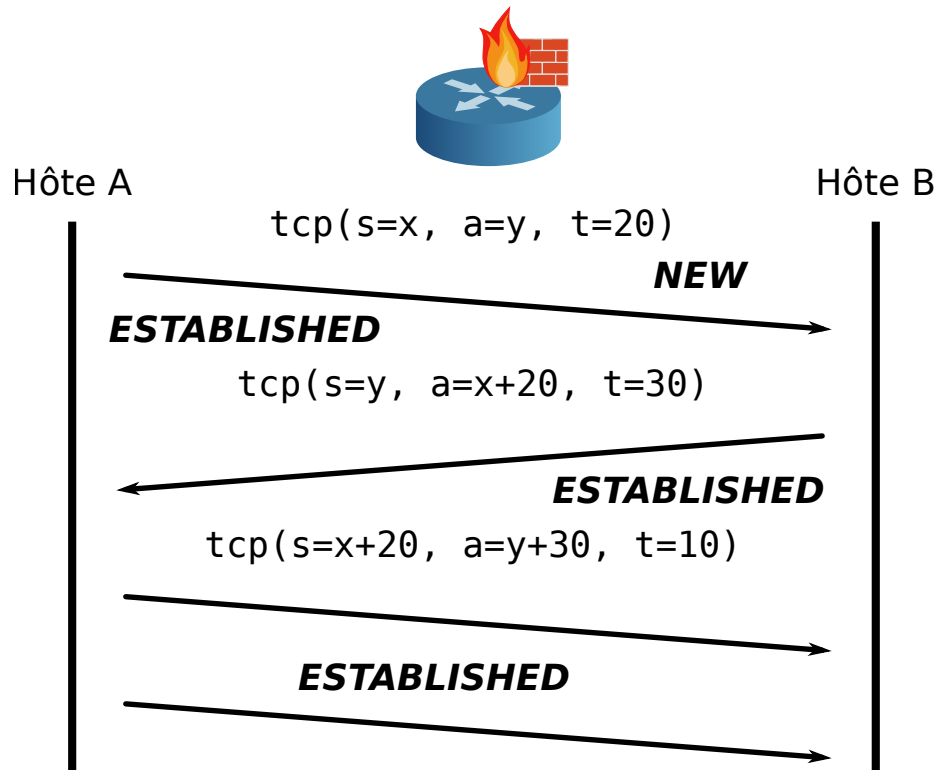
Exemple de suivi de session TCP



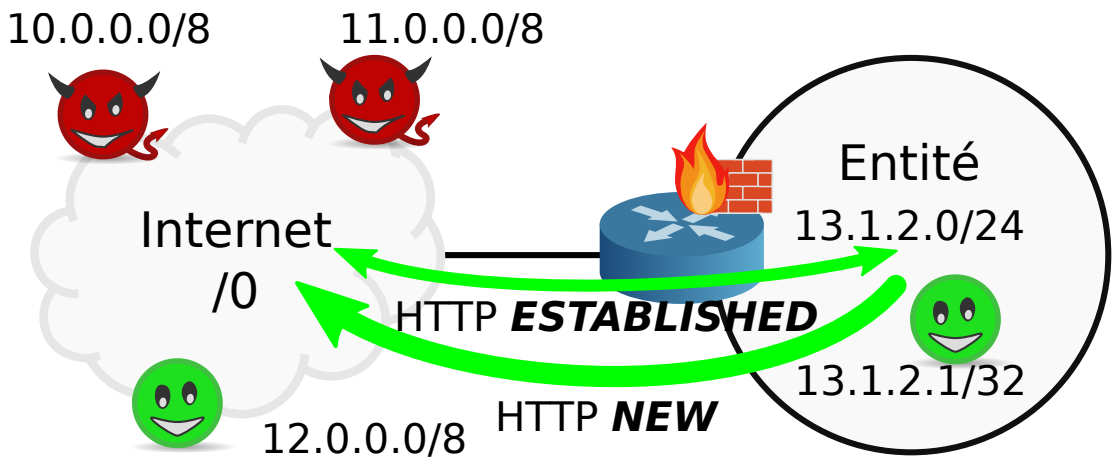
Exemple de suivi de session TCP



Exemple de suivi de session TCP



Filtrage – *filtrage réseau stateful*



Filtrage – *filtrage réseau anti déguisement*

Filtrage *Ingress* et *egress*

Pare-feux matériels

- Produits Cisco : ASA ; PIX
  Firmware Internetwork Operating System (IOS)
- Produits Juniper, NetASQ, Fortinet, . . . [1]

→ accés sur la performance réseau

Pare-feux logiciels

- Linux netfilter + (iptables | nftables (si >= linux 3.13))
- OpenBSD + Packet Filter

→ flexibilité infinie (100 % logiciel >= couche réseau)

Pare-feux matériels

- Produits Cisco : ASA ; PIX
  Firmware Internetwork Operating System (IOS)
- Produits Juniper, NetASQ, Fortinet, . . . [1]

→ accés sur la performance réseau

Pare-feux logiciels

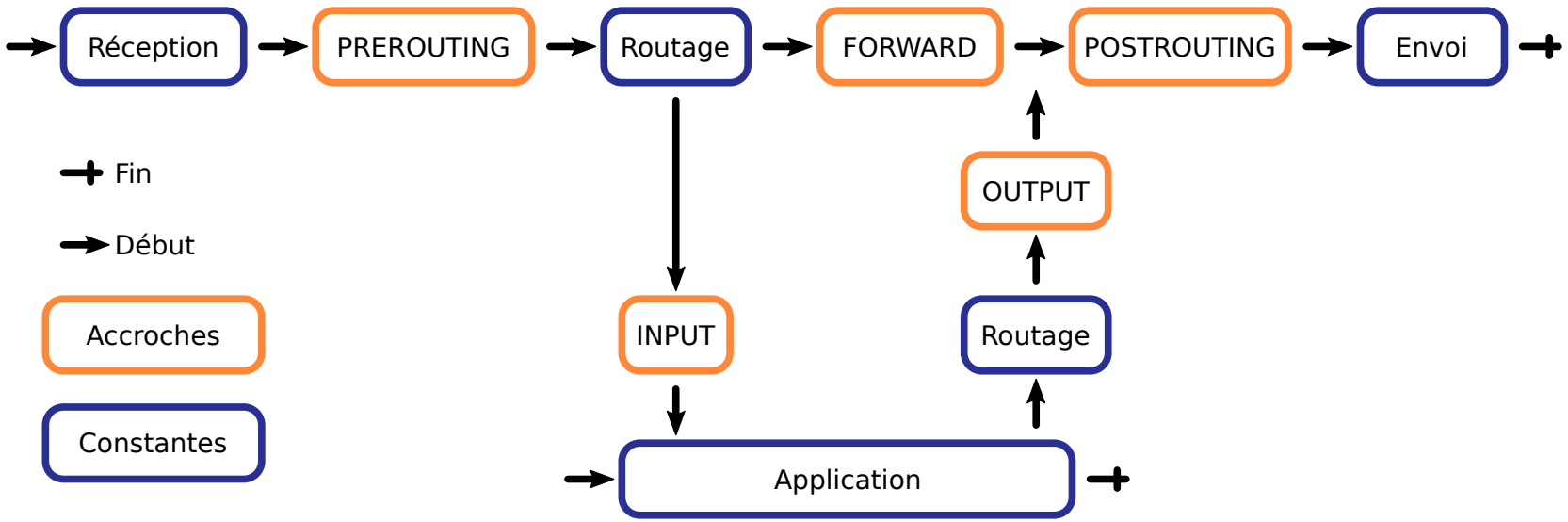
- Linux **netfilter** + (**iptables** | nftables (si >= linux 3.13))
- OpenBSD + Packet Filter

→ flexibilité infinie (100 % logiciel >= couche réseau)

Trame de ce cours

Filtrage – *filtrage réseau – framework Netfilter*

Netfilter est un *framework* réseau extensible à l'aide d'accroches (hooks).



- PREROUTING : réception d'un paquet sur une interface
- POSTROUTING : envoi d'un paquet sur une interface
- FORWARD : relayage d'un paquet réseau (mode routeur)
- INPUT : le paquet en entrée est adressé localement
- OUTPUT : une application envoie un paquet

Filtrage – *filtrage réseau – iptables (xtables)*

Iptables est utilisé par abus de langage pour le module xtables.

Principes basiques

- Xtables est un pare-feu logiciel qui s'appuie sur Netfilter
- Il définit 3 tables ordonnées : `filter` ; `nat` et `mangle`
- Qui contiennent chacune des chaînes de règles ($\simeq$ de filtrage)
- Les chaînes permettent d'appliquer des règles à chaque hook
- Chaque tables proposent au plus les 5 chaînes, parfois moins

Exéction d'une chaîne

1. Un paquet déclenche un crochet Netfilter
2. Xtables est enregistré sur ce crochet
3. Il exécute la chaîne de la table la plus prioritaire sur ce crochet
4. Chaque règle est évaluée jusqu'à ce qu'on trouve un match
5. L'action d'une règle (cible) est exécutée. Ex. : DROP ; ACCEPT

† Si aucune règle ne matche, on exécute la cible par défaut

Le module xtables est extensible

- L'utilisateur peut créer des chaînes pour des besoins spéciaux
  Ces chaînes ajoutées sont seulement exécutée à l'aide de cibles
- Des modules d'extensions de xtables peuvent définir de nouvelles tables et chaînes

Modules d'extensions les plus connus (et indispensables)

- `nf_conntrack` : pare-feu à suivi d'état de session TCP (*stateful*)
- `nf_conntrack_tfp` : pare-feu *stateful* FTP

Filtrage – *filtrage réseau – iptables (xtables)*

La table `mangle` : modification de paquets

- Exécutée en premier
- Modification de champs à la volée et sans suivi d'état (TTL, ...)
- Chaînes : PREROUTING ; POSTROUTING et FORWARD

La table `nat`

- Exécutée en deuxième
- Mise en place du *source NAT* ou *destination NAT*
- Chaînes : PREROUTING et POSTROUTING

La table `filter` : le pare-feu

- Exécutée en troisème
- Définition de règles de filtrage : cibles DROP ou ACCEPT
- Journalisation des évènements : cibles LOG
- Peut interagir avec toutes les chaînes

Filtrage – *filtrage réseau – iptables (xtables)*

Ici image avec le framework netfilter + iptables complet

Filtrage – *filtrage réseau – commande iptables*

Interface utilisateur *iptables*. Interface en ligne de commandes

- Gestion des règles des tables et chaînes xtables
- Ajout de chaînes utilisateurs
- Sauvegarde et restauration des règles
- Interaction avec les modules d'extension

Syntaxe de la commande (Extrait du manuel d'utilisation)

```
iptables [-t table] {-A|-D} chain rule-specification
iptables [-t table] {-I} chain [rulenum] rule-specification
iptables [-t table] {-D} chain rulenum
iptables [-t table] -P chain target
```

- `-A` : ajouter une règle à la fin d'une chaine
- `-D` : retirer de la chaîne la règle spécifiée ou à `rulenum` donné
- `-I` : ajouter une règle à l'index `rulenum` d'une chaîne
- `-P` : configure la cible par défaut de la chaîne `chain`
- `chain` : INPUT, OUTPUT, FORWARD, ...

Filtrage – *filtrage réseau – commande iptables*

Options principales de spécification de règle `rule-specification`

Sélection du protocole

```
-p proto
proto := {nombre | tcp | udp | icmp |...}
```

Sélection de numéros de port de couche transport

```
--dport nombre --sport nombre
```

Sélection des interfaces d'entrée et de sortie

```
-i interface-entree -o interface-sortie
```

Options principales de spécification de règle `rule-specification`

```
{-m | --match} match
match := {limit, state, conntrack}
```

Limitation du débit : `x` paquets par période `t` et rafale max. `y`

```
-m limit --limit x/t --limit-burst y
```

Spécification d'un état pour filtrage stateful

```
-m state --state states
-m conntrack --ctstate states
states := {states, | NEW | RELATED | ESTABLISHED}
```

Filtrage – *filtrage réseau – exemple iptables*

1	12.0.0.0/8	13.1.3.0/24	*	80	accepter
2	13.1.3.0/24	12.0.0.0/8	80	*	accepter
3	Règle par défaut				rejeter

Configuration de la règle 1

```
iptables -t filter -A FORWARD -s 12.0.0.0/8 -d 13.1.3.1
    -p tcp --dport 80 -j ACCEPT
```

Configuration de la règle 2

```
iptables -t filter -A FORWARD -s 13.1.3.0/24 -d 12.0.0.0
    -p tcp --sport 80 -j ACCEPT
```

Configuration de la règle 3 (par défaut)

```
iptables -t filter -P FORWARD DROP
```

Configuration de la règle d'acceptation si autorisé

```
iptables -t filter -A FORWARD -m state
    --state RELATED,ESTABLISHED -j ACCEPT
```

Configuration de la règle de sortie

Autorise l'entrée dans ESTABLISHED ou RELATED

```
iptables -t filter -A FORWARD -s 13.1.3.0/24 -d 12.0.0.0
    -p tcp --sport 80 -m state --state NEW -j ACCEPT
```

Configuration de la règle par défaut

```
iptables -t filter -P FORWARD DROP
```

Plan du cours

Attaques des couches OSI

Filtrage

Sécurisation des communications Réseau

Sécurisation – *introduction*

La cryptographie pour la sécurisation des communications réseau

- Fonctions de hashage : clés de chiffrement et de mot de passe à usage unique ; tags d'intégrité (si secret)
- Chiffrement symétrique : confidentialité de la charge utile ; tags d'intégrité
- Primitives à clé publique : authentification des entités ; échange de clés
- Générateurs d'aléa

Protocoles réseau sécurisés

- Mot de passe à usage unique (*One Time Password*) : S/KEY
- Shell distant sécurisé : *Secured SHell*
- Transport sécurisé : SSL (vulnérable), TLS (vulnérable < 1.3)
- Réseau privé virtuel : IPSec, OpenVPN et Wireguard

Sécurisation – *mot de passe à usage unique*

Problème : le mot de passe est un secret transmit sur le réseau. Le canal doit nécessairement être confidentiel.
Vol du mot de passe ⇒ usurpation d'identité.
Principe : un utilisateur U veut s'authentifier auprès du serveur S
Génération selon le schéma de Lamport :

1. S génère un secret s et un entier $N \rightarrow N-1$ mots de passe
2. S calcule $P =< \mathcal{H}(s), \mathcal{H}(s), \ldots, \mathcal{H}^N(s) > \rightarrow$ mots de passe
3. S envoie P à U, sauvegarde $P(N)$ et supprime s et P

Authentification numéro i si $i \in [1; N[$:

1. C envoie $P(N-i)$
2. S calcule si $\mathcal{H}(P(N-i)) = P(N-i+1)$ sauvé $\rightarrow C$ authentifié
3. S sauve $P(N-i)$

Sécurisation – *mot de passe à usage unique*

Authentification si $i = 1$

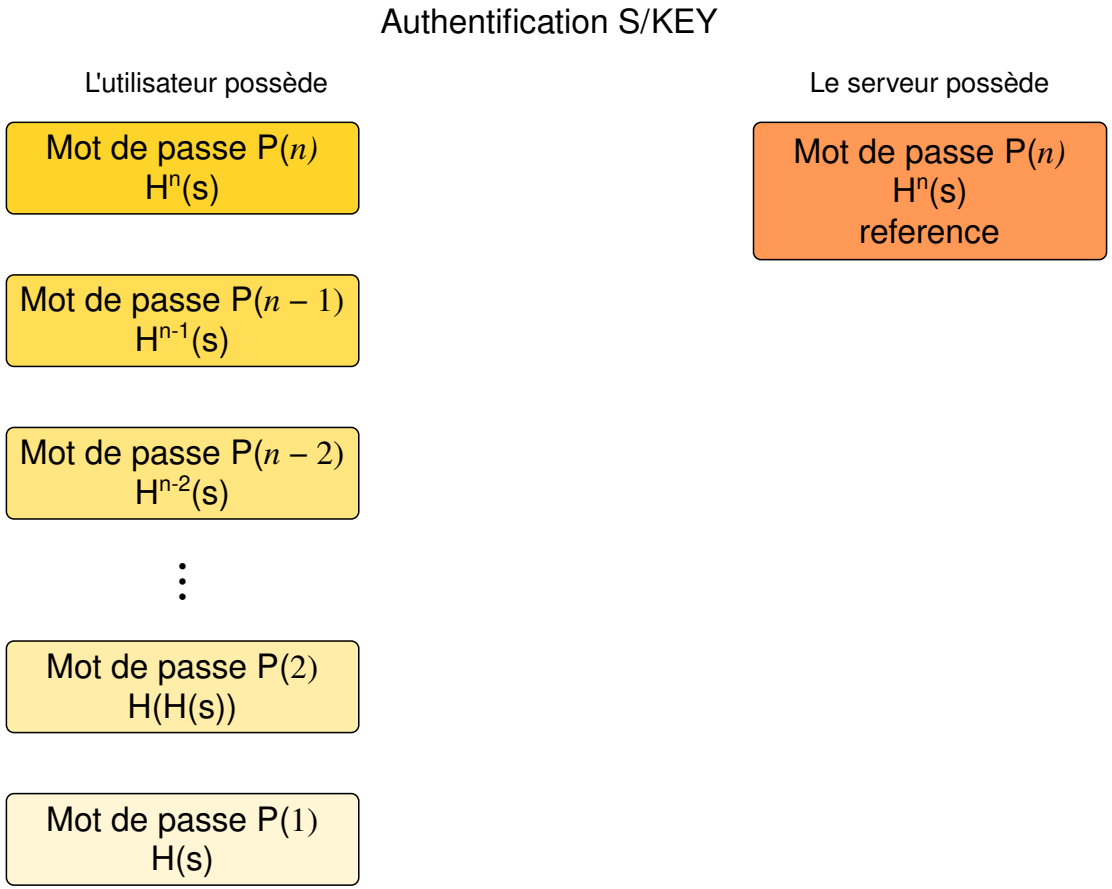


Image traduite et animée depuis

`https://upload.wikimedia.org/wikipedia/en/9/97/Skey_authentication.svg`

Sécurisation – *mot de passe à usage unique*

Authentification si $i = 1$

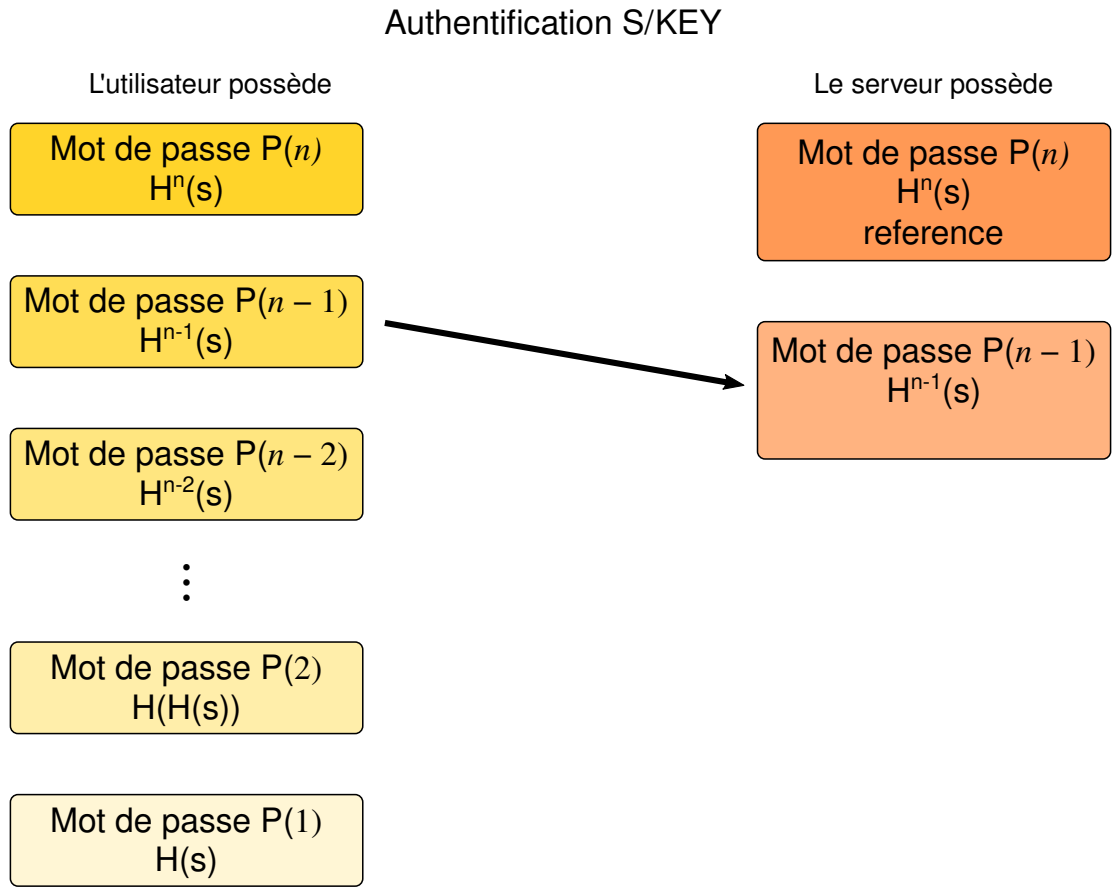


Image traduite et animée depuis

`https://upload.wikimedia.org/wikipedia/en/9/97/Skey_authentication.svg`

Sécurisation – *mot de passe à usage unique*

Authentification si $i = 1$

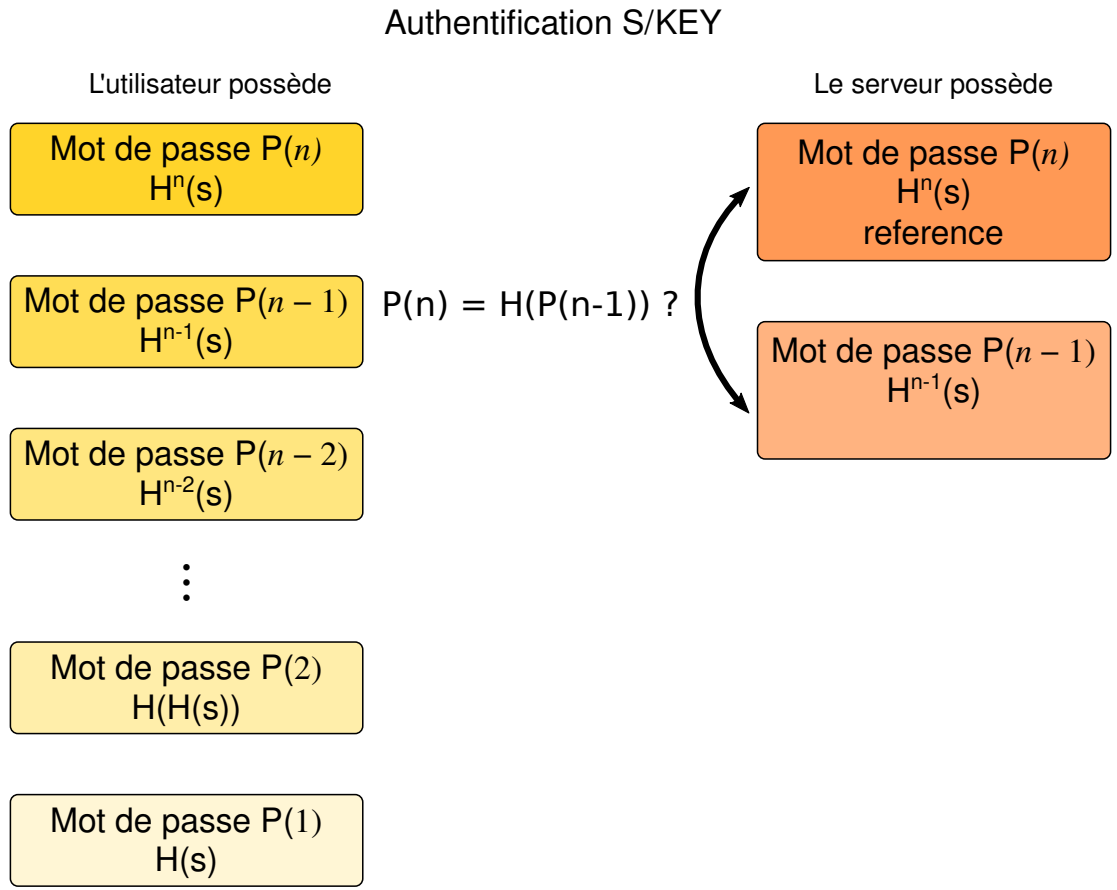


Image traduite et animée depuis

`https://upload.wikimedia.org/wikipedia/en/9/97/Skey_authentication.svg`

Sécurisation – *mot de passe à usage unique*

Authentification si $i = 1$

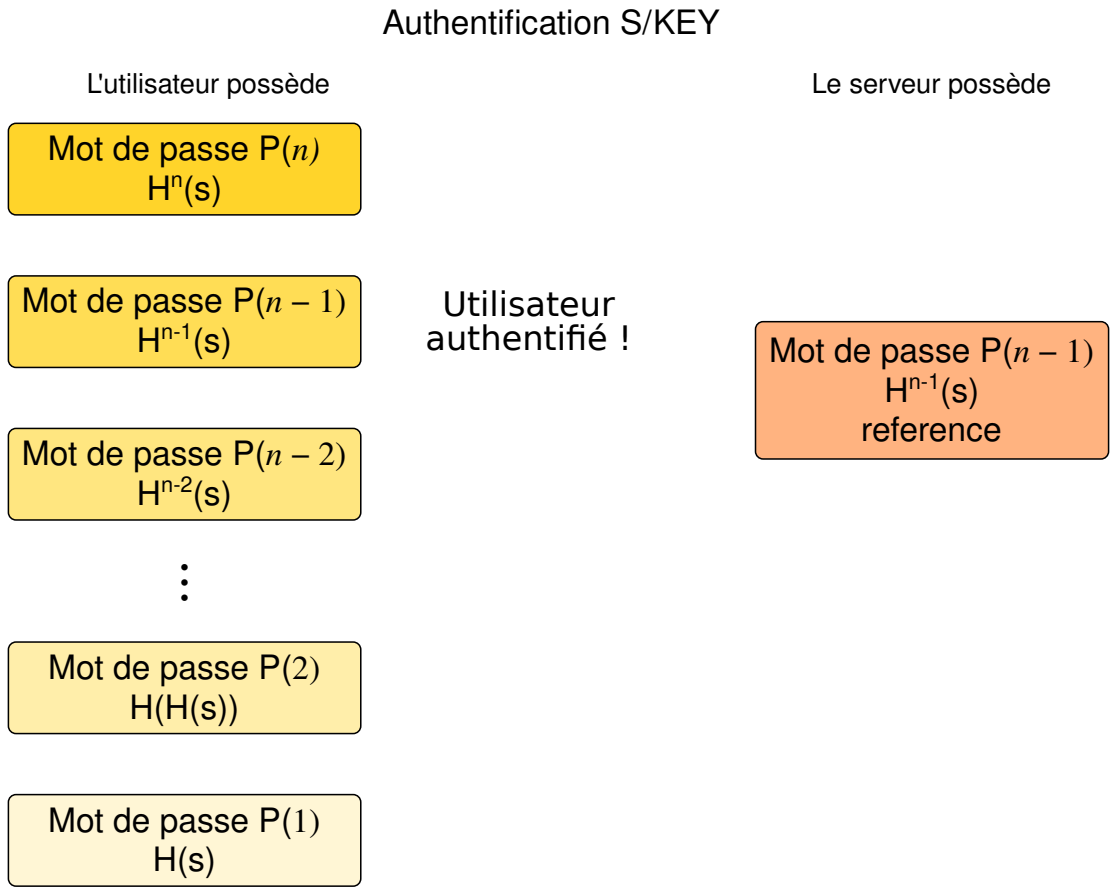


Image traduite et animée depuis

`https://upload.wikimedia.org/wikipedia/en/9/97/Skey_authentication.svg`

Sécurisation – *certificats numériques*

Définition

Preuve d'intégrité de l'association d'une entité E, d'une paire de clé asymétrique (K_{pub}, K_{priv}), de métadonnées d et de données métier D.

Authenticité

L'authenticité d'un certificat est prouvée à l'aide d'une signature S par un tiers et de la validé dans le temps de son utilisation. La valeur de la signature est égale à la confiance que l'on a dans le tiers.

L'authenticité est intégrité de :

- Identité
- Validité
- Intégralité
- Précision

Certificat : $C = (E, K_{pub}, d, D, S)$

Sécurisation – *certificats numériques*

Définition

Preuve d'intégrité de l'association d'une entité E, d'une paire de clé asymétrique (K_{pub}, K_{priv}), de métadonnées d et de données métier D.

Intégrité

L'intégrité d'un certificat est prouvée à l'aide d'une signature S par un tiers et de la validé dans le temps de son utilisation. La valeur de la signature est égale à la confiance que l'on a dans le tiers.

Intégrité (crypto) :

- Authenticité
- Validité
- Intégralité
- Précision

Certificat : $C = (E, K_{pub}, d, D, S)$

Sécurisation – *certificats numériques*

Validité

Date d'expiration $\in d$. Permet de prouver la validité $\Rightarrow$ un certificat expire

Données métiers

- HTTPS : entête `host` HTTP $\in D$, prouve que le site derrière le nom de domaine `host` appartient à E
- BGP et RPKI : routes $\in D$, prouve que le routeur appartement à E a le droit d'annoncer des routes
- IPSEC : $D = \emptyset$. Seule l'identité de l'entité est vérifiée

Un certificat à quoi ça sert ?

Prouver que K_{pub} est authentique dans la vue d'ouvrir une session cryptographique (HTTPS, IPSEC).
Vérifier la validité des information métiers annoncées dans un certificat par vérification d'une signature (BGP et RPKI).

Sécurisation – *modèles de confiance et signature asymétrique*

- *Trust On First Use* (TOFU) (SSH)
  Vérification lors de la première utilisation → puis confiance
- Toile de confiance ou *Web of trust* (PGP)
  Modèle de confiance distribué ou chaque utilisateur peut signer le certificat d'autres utilisateurs
  Intuition : plus de signatures ⇒ plus de confiance
- Certificats épinglés (HTTPS) → confiance par présence
  Installation *out-of-band* (paquetages, cérémonies de remise de clés)
- Autorités de certification (HTTPS)
  Certificat à vérifier signé par un autre certificat de confiance épinglé ayant autorité → autorité de certification

Sécurisation – *autorités de certification et certificat épinglés*

Processus de création d'un certificat : entité A veut créer un certificat, qui sera utilisé par entité B, qui fait confiance à l'autorité de certification AC. L'autorité AC peut déléguer la vérification de l'identité de A à une autorité d'enregistrement AE.

1. A génère une paire de clé (K_{priv}, K_{pub}) et cache K_{priv}
2. A génère une requête de signature $R = (E, K_{pub}, d, D)$
3. A envoie R à AE
4. AC vérifie l'identité de A (exemple *let's encrypt* : secret partagé)
5. AE certifie l'intégrité de R après de AC
6. AC signe le certificat $C = (E, K_{pub}, d, D, \mathbf{S})$ et le donne à A
7. A peut s'authentifier auprès de B, B à épinglé le certificat de AC

Sécurisation – *infrastructure à clés publiques*

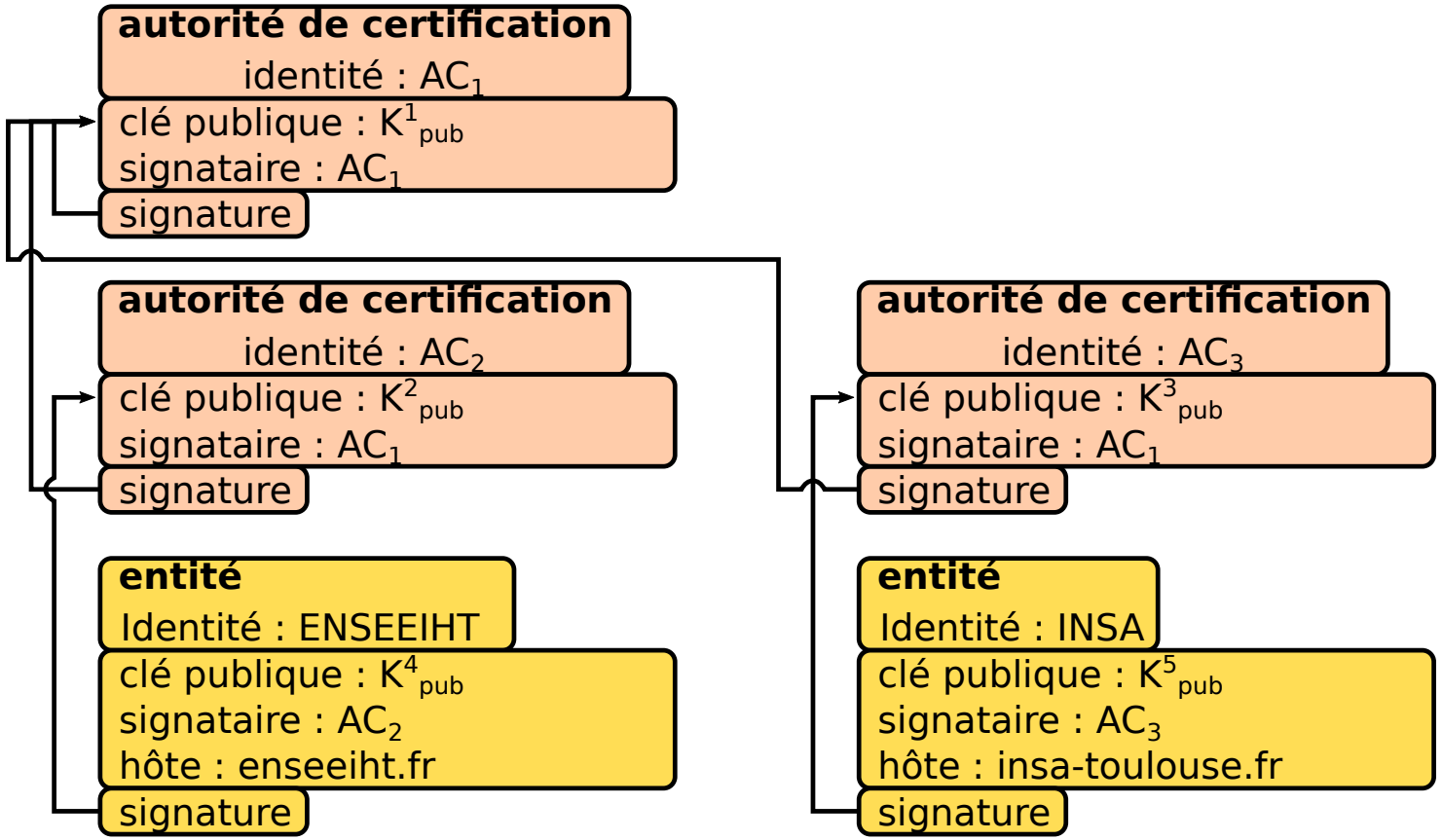
Généralisation du modèle précédent

Ensemble des logiciels, matériels, primitives cryptographiques qui permettent de créer, gérer, sauvegarder, distribuer et révoquer des certificat basés sur la cryptographie asymétrique.

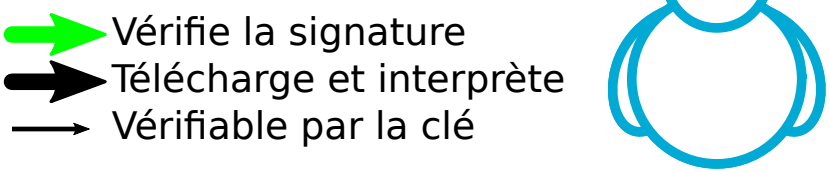
Éléments liés

- Listes de révocation ou *Certificate Revokation List* (CRL)
- *Hardware Security Modules* (HSM) : stockage sécurité des clés
- Bibliothèques de couches de transport sécurisées (OpenSSL)
- Bibliothèques gestion des certificats au `X509` (OpenSSL)

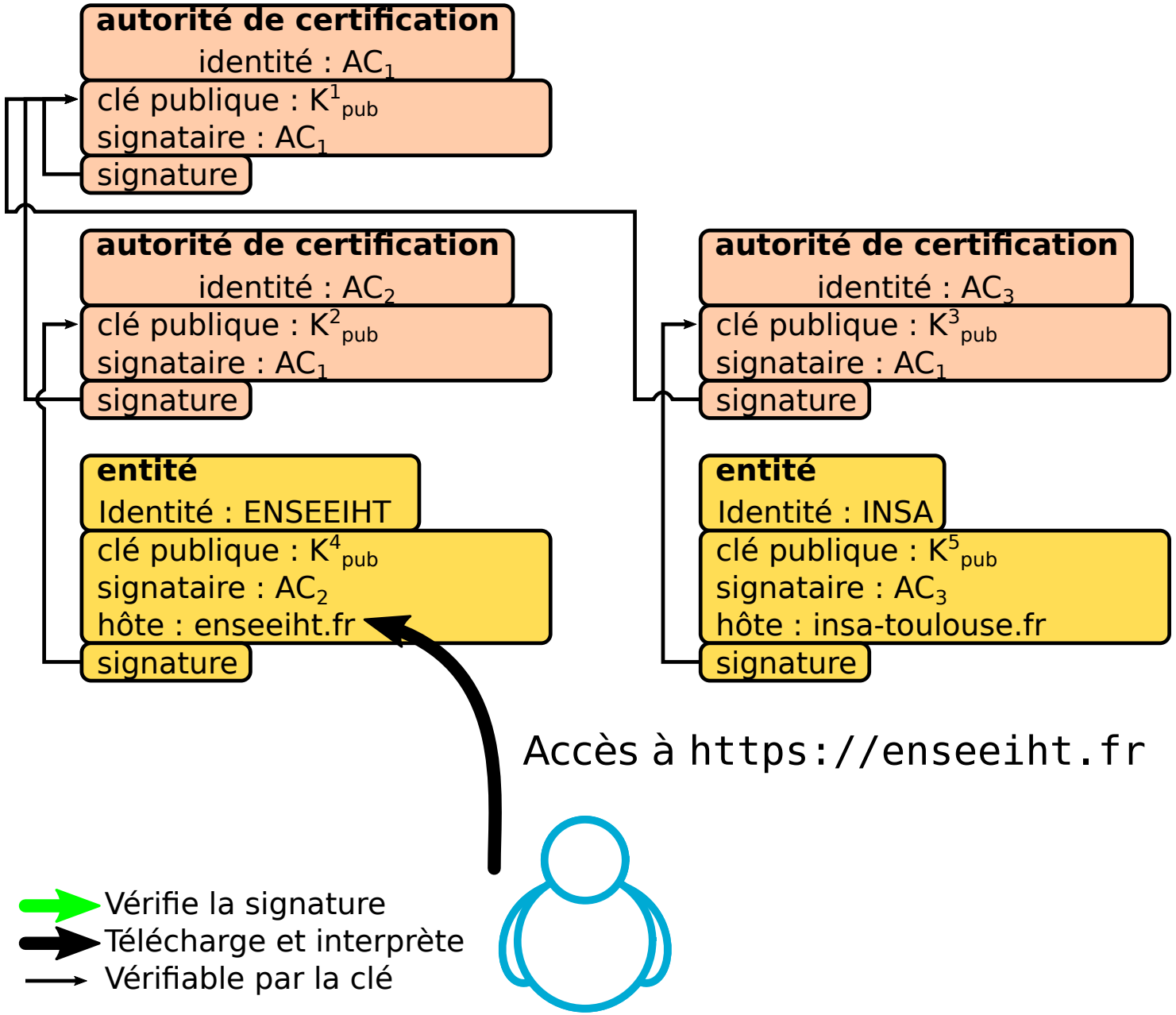
Sécurisation – *infrastructure à clés publiques HTTPS*



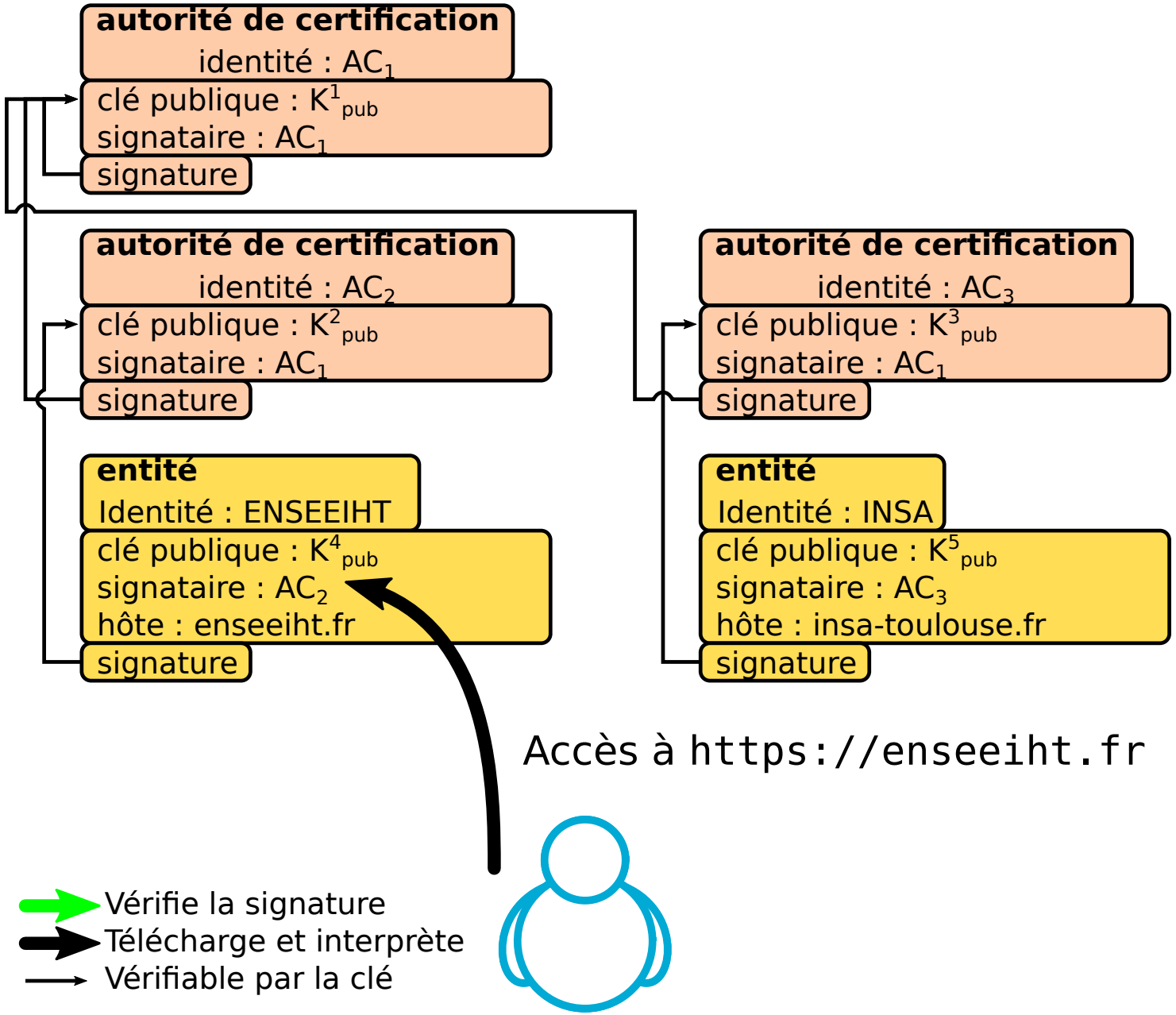
Accès à `https://enseeiht.fr`



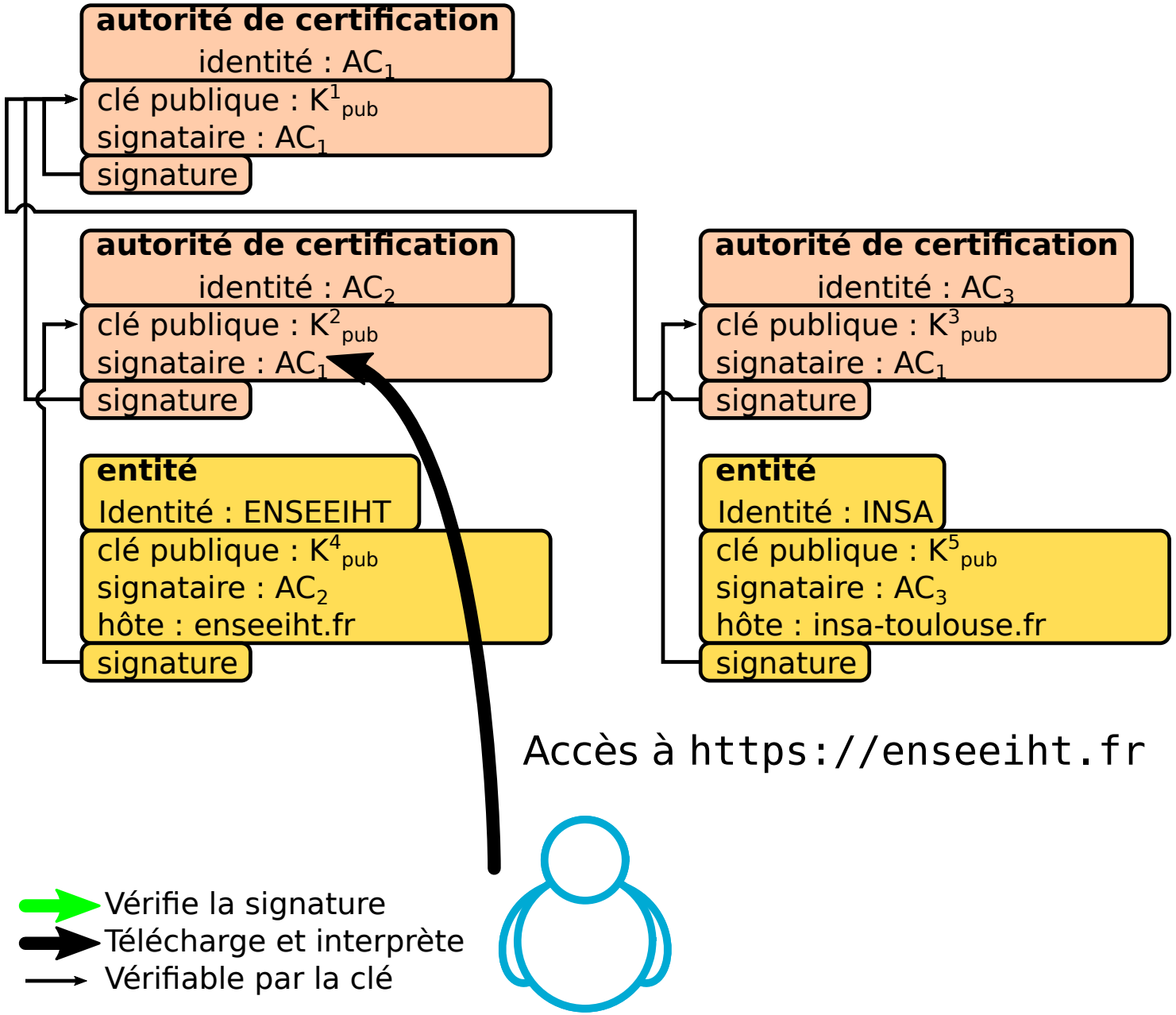
Sécurisation – *infrastructure à clés publiques HTTPS*



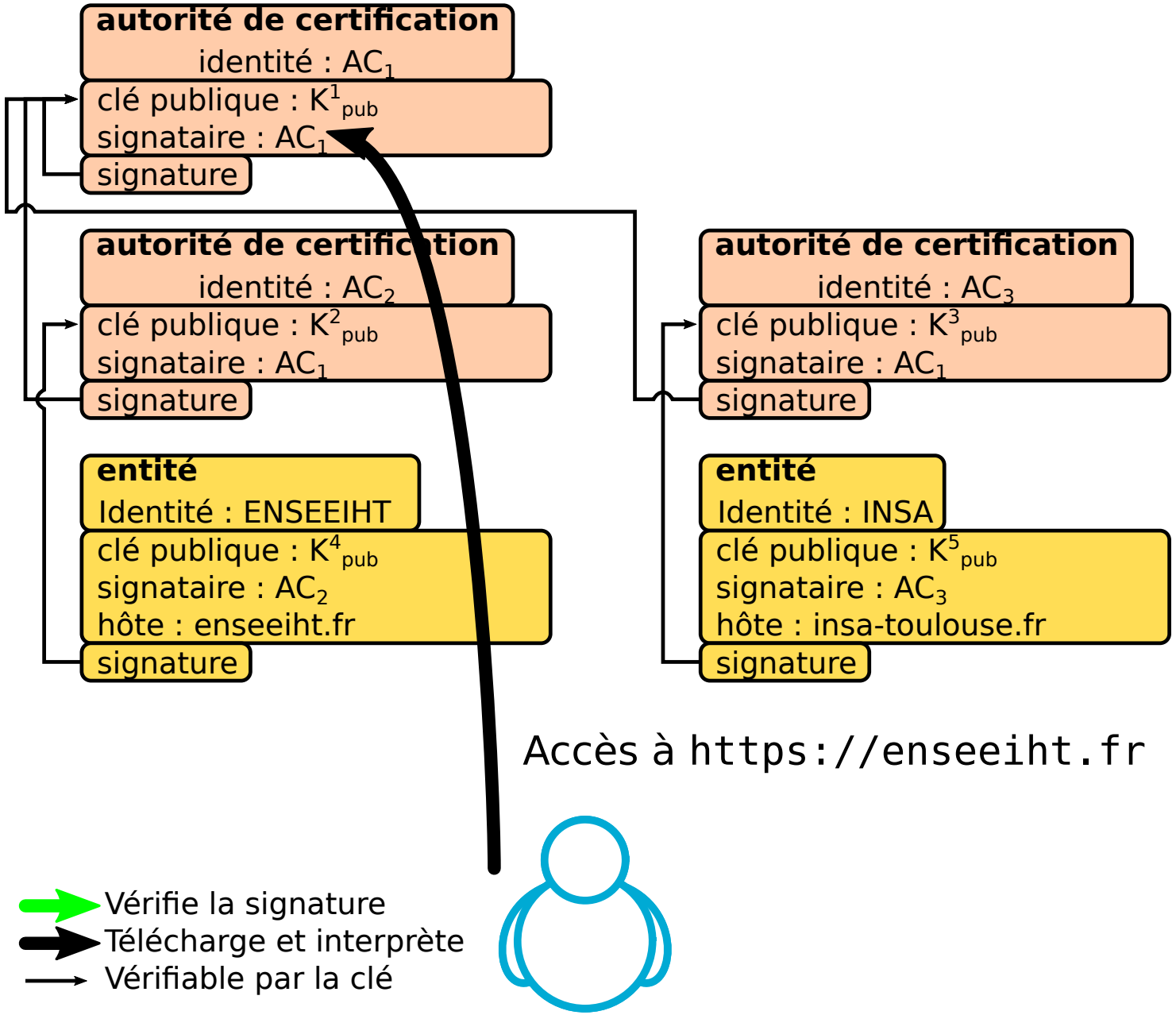
Sécurisation – *infrastructure à clés publiques HTTPS*



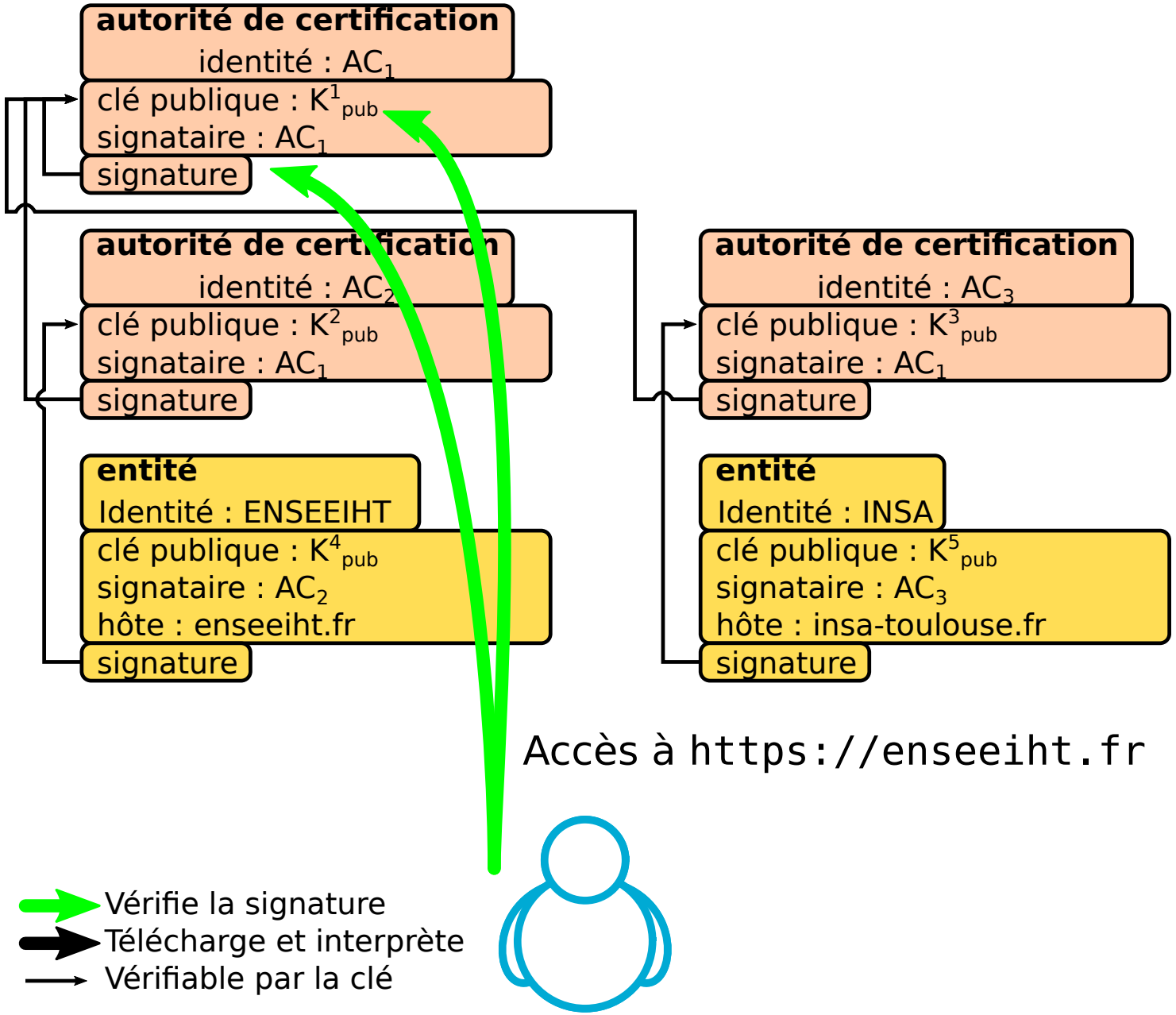
Sécurisation – *infrastructure à clés publiques HTTPS*



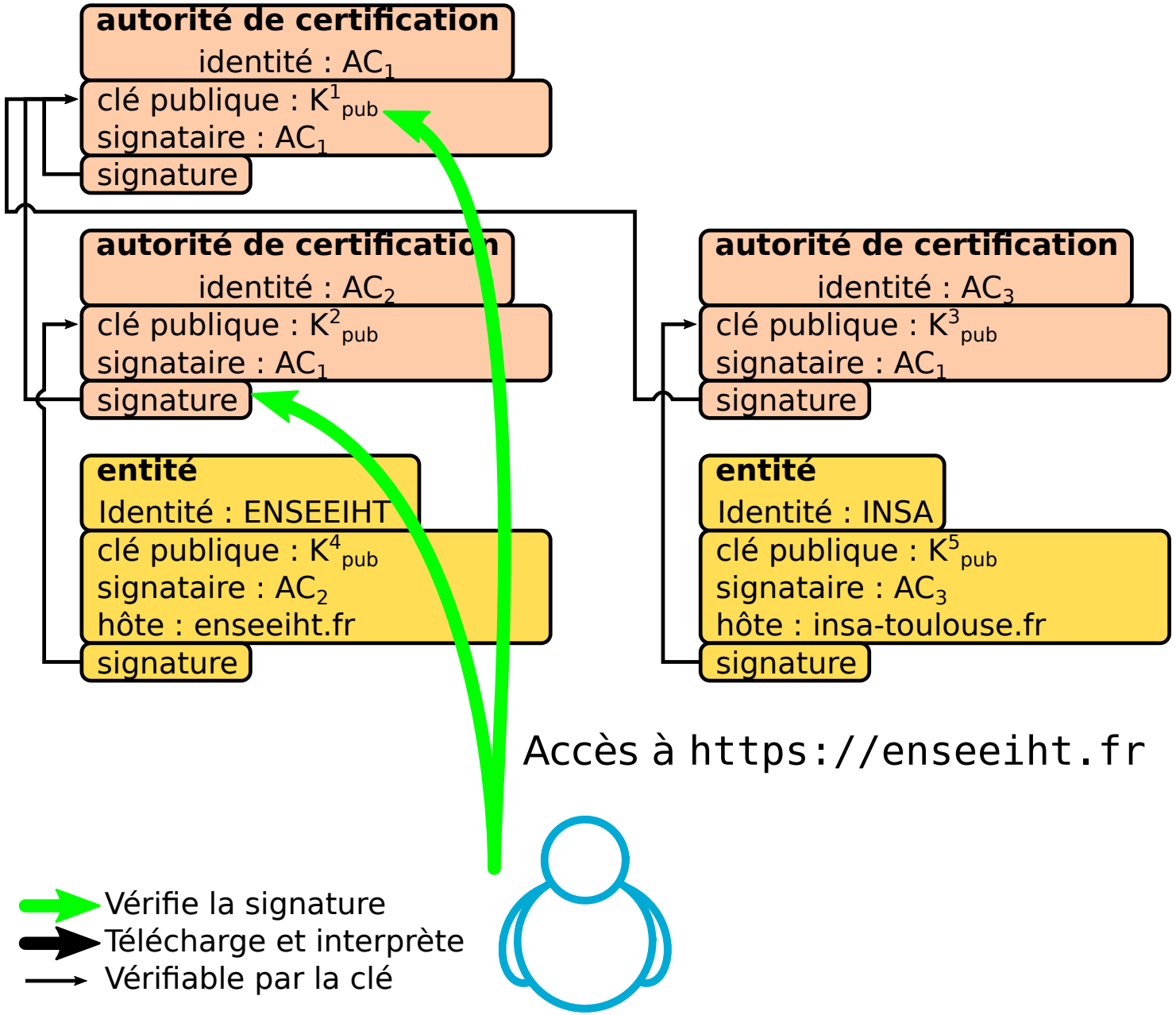
Sécurisation – *infrastructure à clés publiques HTTPS*



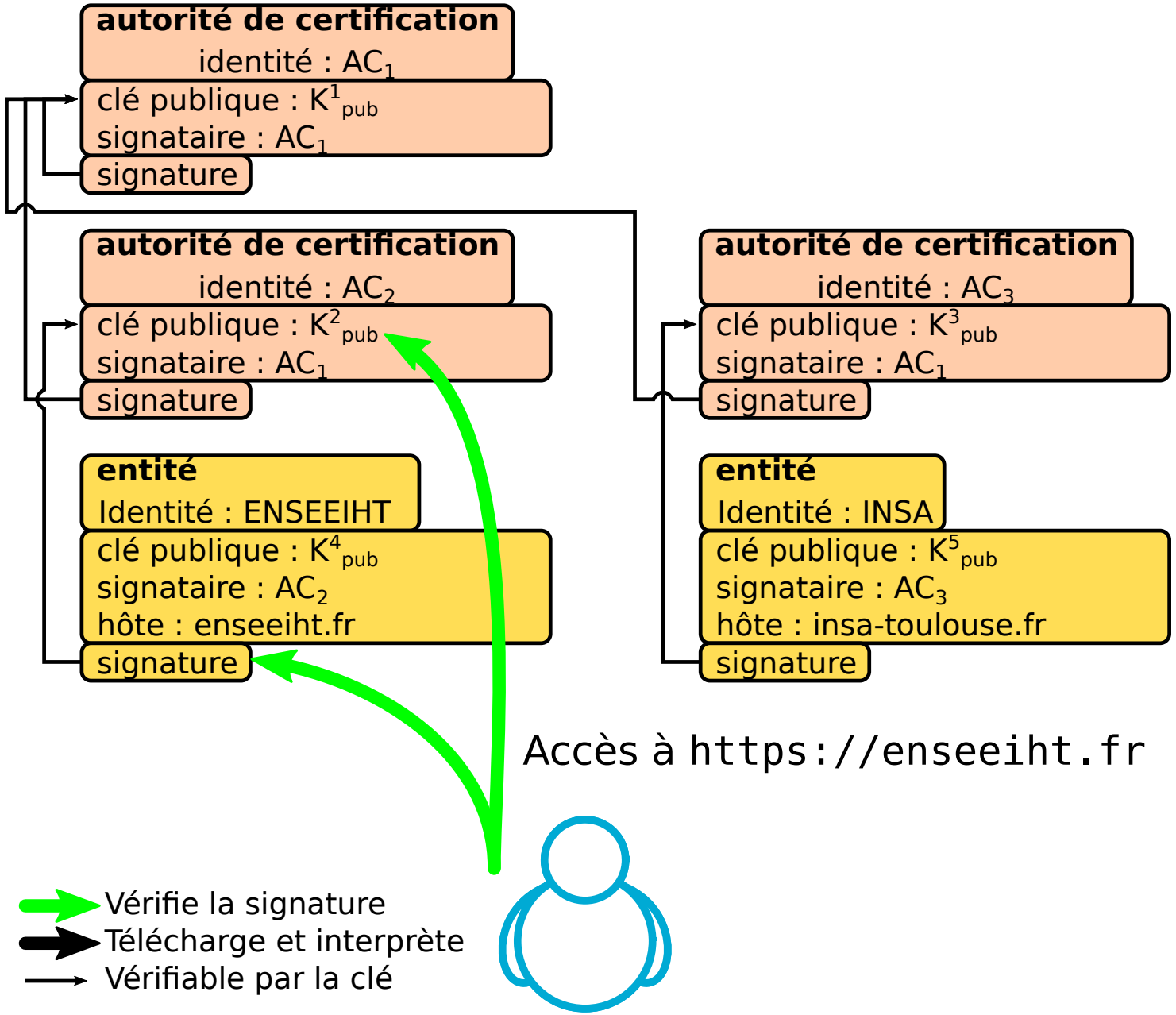
Sécurisation – *infrastructure à clés publiques HTTPS*



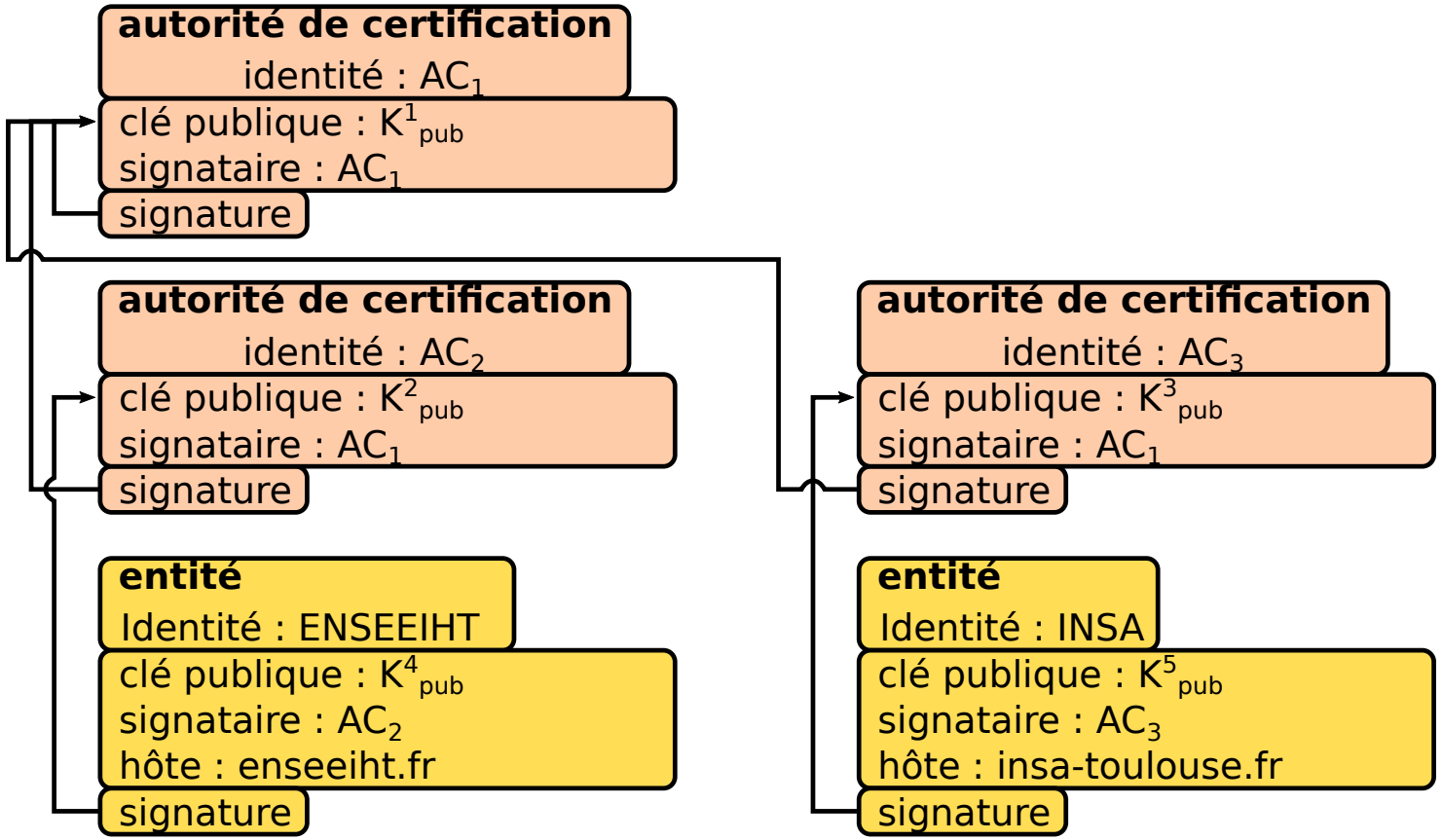
Sécurisation – *infrastructure à clés publiques HTTPS*



Sécurisation – *infrastructure à clés publiques HTTPS*

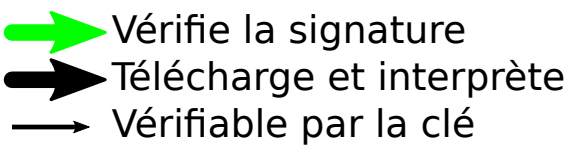


Sécurisation – *infrastructure à clés publiques HTTPS*



Accès à `https://enseeiht.fr`

Vérifié !



Sécurisation – *format des standardisé de certificats : x509*

Exemple : certificat HTTPS de `www.google.com`

```
$ openssl x509 -text -noout -in wwwgooglecom.crt
Certificate:
    Data:
        Issuer: C = US, O = Google Trust Services,
            CN = Google Internet Authority G3
        Validity
            Not Before: Mar  1 09:46:35 2019 GMT
            Not After : May 24 09:25:00 2019 GMT
        Subject: C = US, ST = California, L = Mountain View, O = Google LLC,
            CN = www.google.com
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                pub: [..] # clé
        X509v3 extensions:
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 Authority Key Identifier:
            keyid: [..] # identifiant de clé
            X509v3 CRL Distribution Points:
                Full Name:
                  URI:http://crl.pki.goog/GTSGIAG3.crl
    Signature Algorithm: sha256WithRSAEncryption
         [..] # signature
```

Sécurisation – *IPSEC*

Généralités

- À été créé dans le but de sécurité la couche réseau IPv4
- Standardisé par l'*Internet Engineering Task Force* (IETF)
- Créé en 1992, premier standard en 1995
- Natif dans IPv6, toutes les piles IP doivent le supporter

Propriétés de sécurité proclamées

- Confidentialité du trafic
- Intégrité des messages (prouvée, au sens cryptographique)
- Authentification des entités
- Protection contre le rejeu

Sécurisation – *IPSEC*

Protocoles proposés

- *Authentication Header* (AH)
- *Encapsulated Security Payload* (ESP)

Mise en œuvre dans IP

- IPv6 : *extension header* dédié
- IPv4 : encapsulation d'un protocole réseau à l'aide du champ protocole. AH et ESP possèdent eux-même un champ *Next Header* pour spécifier le prochain protocole encapsulé
  Exemple : `IP.p` → `TCP` devient
  `IP.p` → `AH.n-h` → `ESP.n-h` → `TCP`

Modes proposés

- Tunnel
- Transport

Sécurisation – *IPSEC – Authentication Header*

Propriétés garanties

- Intégrité des messages (au sens crypto avec authenticité)
  Intégrité de l'intégralité du paquet IP
- Pas de chiffrement ⇒ pas de confidentialité

Champs importants

- *Security Parameter Index* (SPI) → *Security Association* (SA)
- *Sequence Number* (SN) : SN + tag d'intégrité = anti-rejeu
- *Integrity Check Value* : tag d'intégrité
  Exemple : HMAC-SHA256

Sécurisation – *IPSEC – Security Association*

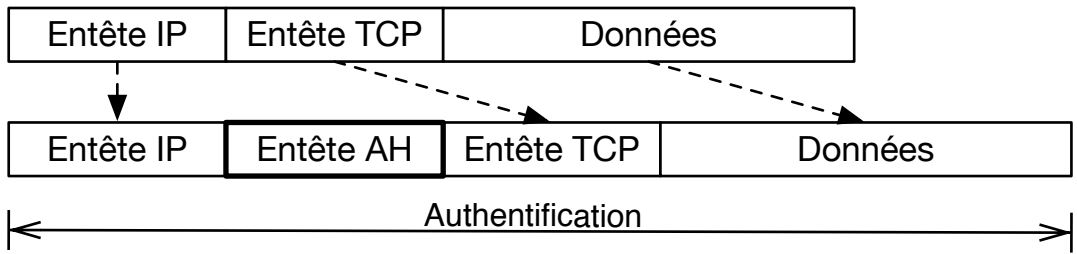
- Choix des algorithmes cryptographiques pour cette SA
- Dans le sens de communication courant
- Paramètres de configuration des algorithmes

Exemple Linux avec `setkey`

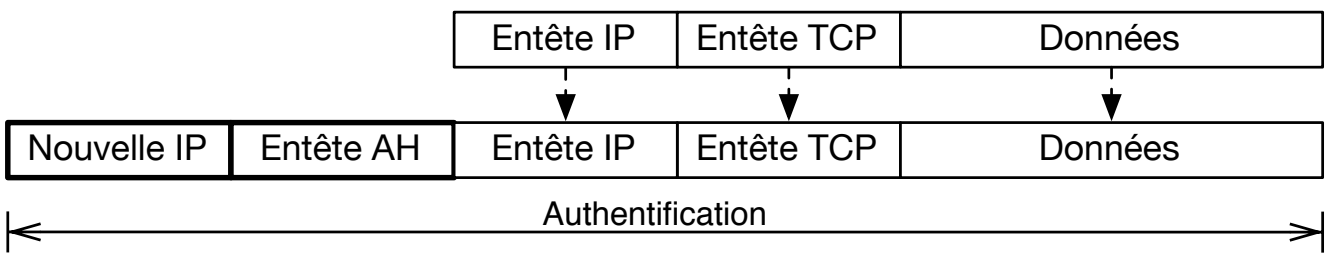
```
# AH SAs using 128 bit long keys
add 193.168.1.23 193.168.1.17 ah 0x200 -A hmac-md5
0x891aac72662ca73d424285ad586e2bde;
add 193.168.1.17 193.168.1.23 ah 0x300 -A hmac-md5
0x735a466744ae4a880753184f8b0568c0;

# ESP SAs using 192 bit long keys (168 + 24 parity)
add 193.168.1.23 193.168.1.17 esp 0x201 -E 3des-cbc
0xa4a5cda0b0e1f3f850bee59b51d9e7c4ddff9a923f94f036;
add 193.168.1.17 193.168.1.23 esp 0x301 -E 3des-cbc
0x6a47d3f709ffc990d3bdc0bec949bba1e913917deaeb27a0;
```

Sécurisation – *IPSEC – Authentication Header*

Mode transport



Mode tunnel



Sécurisation – *IPSEC – Authentication Header*

Propriétés garanties

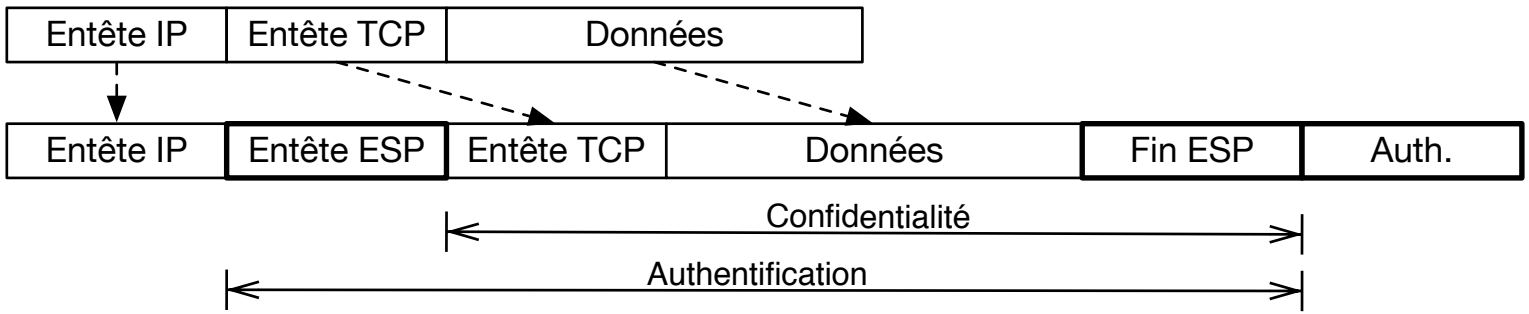
- Intégrité des messages (au sens crypto avec authenticité)
  À L'EXCEPTION DE l'entête IP
- Chiffrement ⇒ confidentialité des données

Champs importants

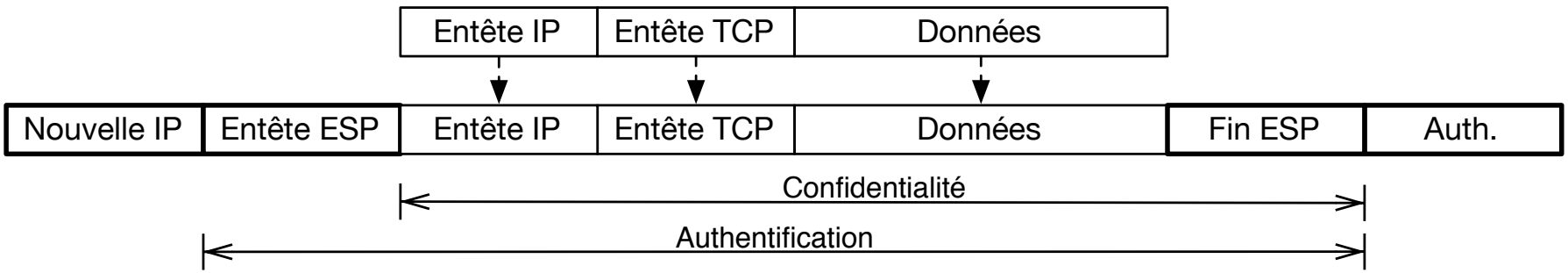
- *Security Parameter Index* (SPI) → *Security Association* (SA)
- *Sequence Number* (SN) : SN + tag d'intégrité = anti-rejeu
- *Integrity Check Value* (ICV) : tag d'intégrité
- *Padding length* : ICV est placé après la charge utile

Sécurisation – *IPSEC – Authentication Header*

Mode transport



Mode tunnel



Sécurisation – *IPSEC – Security Policy*

Security Policy (SP)

- Comment chaque nœud du réseau utilisant IPSEC définit comment traiter les paques
- Ces politiques définissent comment pour chaque paquet quelle configuration d'IPSEC doit être utilisée

Exemple Linux avec `setkey`

```
spdadd 193.168.1.23 193.168.1.17 any -P in ipsec
esp/transport//require
ah/transport//require;

spdadd 193.168.1.17 193.168.1.23 any -P out ipsec
esp/transport//require
ah/transport//require;
```

Sécurisation – *IPSEC – Internet Key Exchange*

Désavantages d'IPSEC seul

- Configuration lourde
- Pas de flexibilité
- Échange des clés

Protocole *Internet Key Exchange* (IKE)

- Négociation dynamique des paramètres AH et ESP dans des SA
- Association automatique des SPI avec les SA
- Échange automatique de clés de chiffrement et de tags d'intégrité
- **Répond** aux limites levées

Sécurisation – *IPSEC – Internet Key Exchange*

Désavantages d'IPSEC seul

- Configuration lourde
- Pas de flexibilité
- Échange des clés

Protocole *Internet Key Exchange* (IKE)

- Négociation dynamique des paramètres AH et ESP dans des SA
- Association automatique des SPI avec les SA
- Échange automatique de clés de chiffrement et de tags d'intégrité
- **Répond** aux limites levées
- La flexibilité est évidemment limité par les algorithmes supportés par les deux hôtes à la fois !

Sécurisation – *IPSEC – Internet Key Exchange*

Désavantages d'IPSEC seul

- Configuration lourde
- Pas de flexibilité
- Échange des clés

Protocole *Internet Key Exchange* (IKE)

- Négociation dynamique des paramètres AH et ESP dans des SA
- Association automatique des SPI avec les SA
- Échange automatique de clés de chiffrement et de tags d'intégrité
- **Répond** aux limites levées
- La flexibilité est évidemment limité par les algorithmes supportés par les deux hôtes à la fois !
- Mise en œuvre de référence sous linux : Racoon

Sécurisation – *IPSEC – teaser*

TPéééé