Sécurité
Attaques OSI niveau 3-4

Carlos Aguilar
carlos.aguilar@enseeiht.fr

IRIT-IRT

Plan

Définition et objectifs

What ?

Utiliser la fragmentation IP à des fins autres que l'adaptation des MTU

Why ?

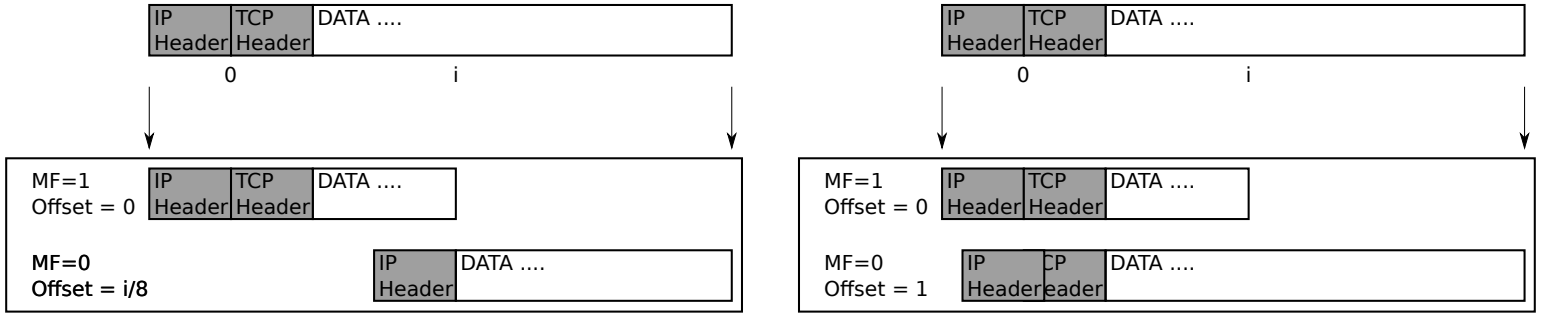
Contournement des moyens d'autorisation

- traverser les firewalls
- effectuer des actions interdites sans être détecté par un IDS

Déni de service

- saturer la mémoire d'une machine
- faire planter le driver réseau

Pourquoi ça marche ?



La fragmentation et les firewalls/IDS

Une application sans état analyse les paquets du réseau indépendemment.

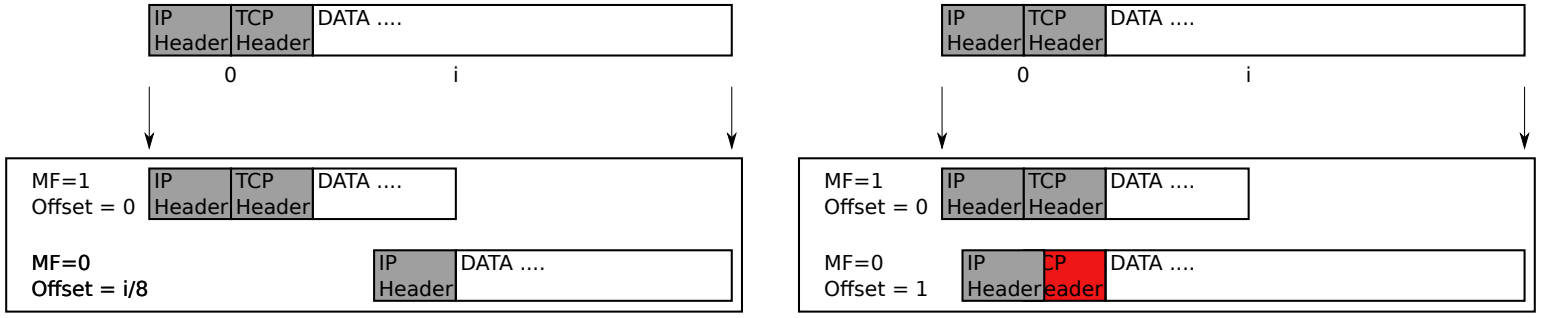
- Firewall : vérification de l'en-tête TCP/UDP du premier fragment
- IDS : recherche de signatures sur chaque fragment

Même une application stateful peut avoir une stratégie de reconstruction différente !

Deux voies d'attaque

Fragmentation à outrance
Messages superposés

Pourquoi ça marche ?



La fragmentation et les firewalls/IDS

Une application sans état analyse les paquets du réseau indépendemment.

- Firewall : vérification de l'en-tête TCP/UDP du premier fragment
- IDS : recherche de signatures sur chaque fragment

Même une application stateful peut avoir une stratégie de reconstruction différente !

Deux voies d'attaque

Fragmentation à outrance
Messages superposés

Première voie : la fragmentation à outrance

Idée

Envoyer dans chaque fragment huit octets (micro-fragmentation)
Firewall/IDS sans reconstruction verra même pas les en-têtes (sauf UDP !)

How ? Utilisation de Scapy

```
from scapy.all import *
destIP = ''123.123.123.123''
data = ''XXXXXXXXX''
ip = IP(dst=destIP, id=12345)/TCP(sport=65000,dport=25)/data
packetList = ip.fragment(francise=8)
for packet in packetList : send(packet)
```

Deuxième voie : superposition des fragments

Idée : plusieurs options de reconstruction

- First : Windows, MacOS, SUN
- Last : Cisco
- Linux : Linux (premier permettant de compléter)
- BSD : AIX, FreeBSD, Wireshark (dernier permettant de compléter)

Le firewall/IDS utilise une reconstruction et la victime une autre

How ? Utilisation de Scapy

```
from scapy.all import *
destIP = ''123.123.123.123''
data = ''XXXXXXXXX''
ip1 = IP(dst=destIP, id=12345, flags=1, frag=0)/TCP(sport=65000,dport=22)/data
ip2 = IP(dst=destIP, id=12345, flags=0, frag=0)/TCP(sport=65000,dport=25)/data

datas = ['' V'', ''AI'', ''BR'', ''CU'', ''DS'']; pL = []
pL[0] = IP(dst=destIP, id=12345, flags=1)/UDP(sport=65000,dport=25)/datas[0]
for i in range(1,4) : pL[i] = IP(dst=destIP, id=12345, flags=1,frag=i)/datas[i]
pL[4] = IP(dst=destIP, id=12345, flags=0,frag=4)/datas[4]
for packet in pL : send(packet)
```

Plan

L'usurpation d'adresse IP

What ?

Envoyer des paquets sur le réseau avec une adresse IP qui n'a pas été, ou n'est pas destinée à être, attribuée à l'émetteur

Why ?

Contournement des moyens d'authentification/autorisation
Éviter d'être tracé (quel que soit l'objectif)

Pourquoi ça marche ?

Aucun moyen d'authentification de l'émetteur mis en place (ni sur IP ni sur le routage)

Contre minimum

Filtrage `egress, ingress` au niveau des routeurs
Ne router que les paquets qui ont des IPs qui correspondent à leurs réseaux

Problèmes

ICMP, UDP

Aucun (pas d'ajout de sécurité par rapport à IP)

TCP

Circuit virtuel

- les paquets sont ordonnés
- les données sont acquittées

Difficile de spoofer en aveugle
Difficile d'insérer du trafic

TCP Connection Spoofing (1/2)

What ?

Établir une connexion TCP avec l'adresse IP d'autrui

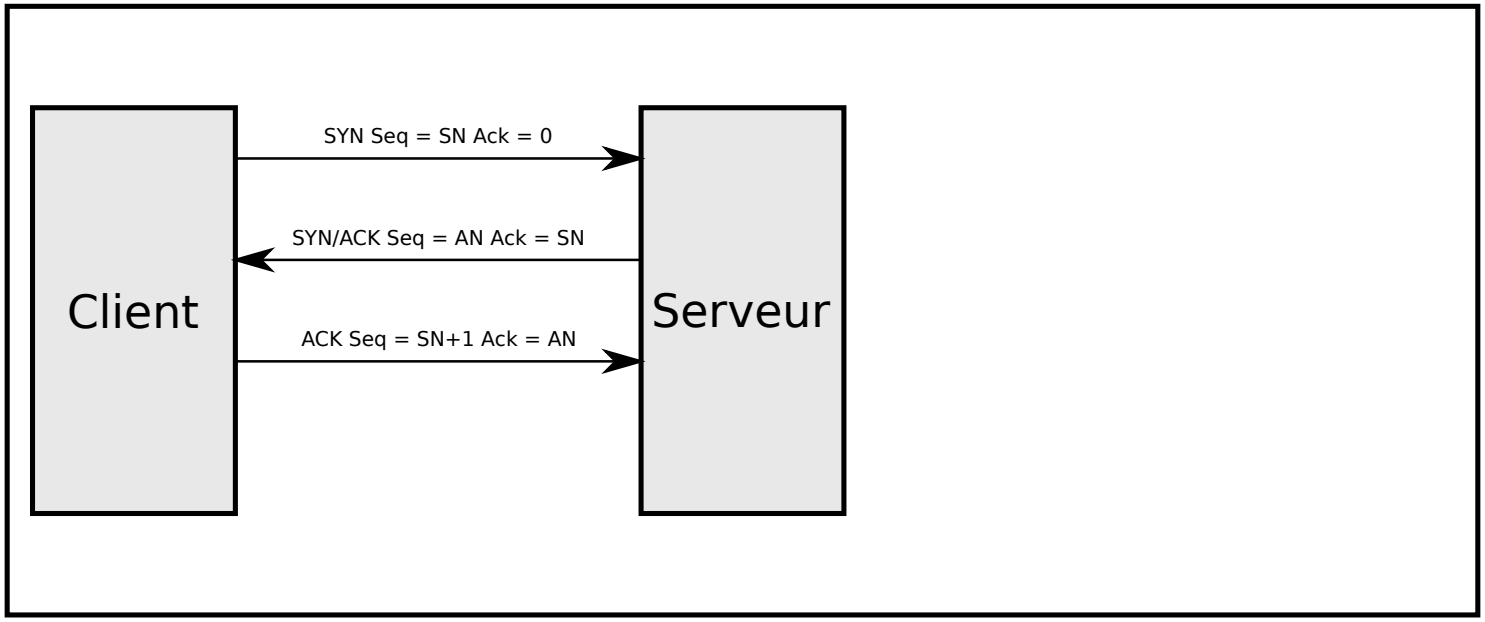
Why ?

Contourner les moyens d'authentification/autorisation

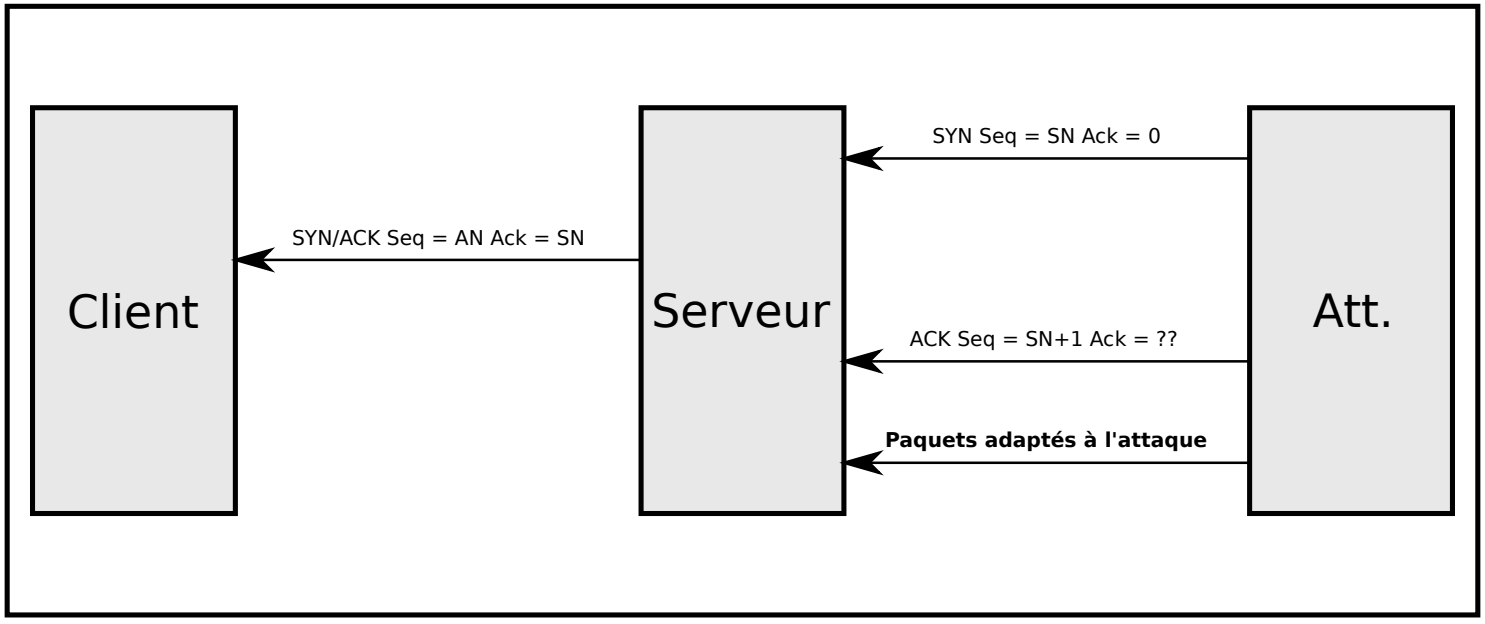
How ?

Pas évident ...

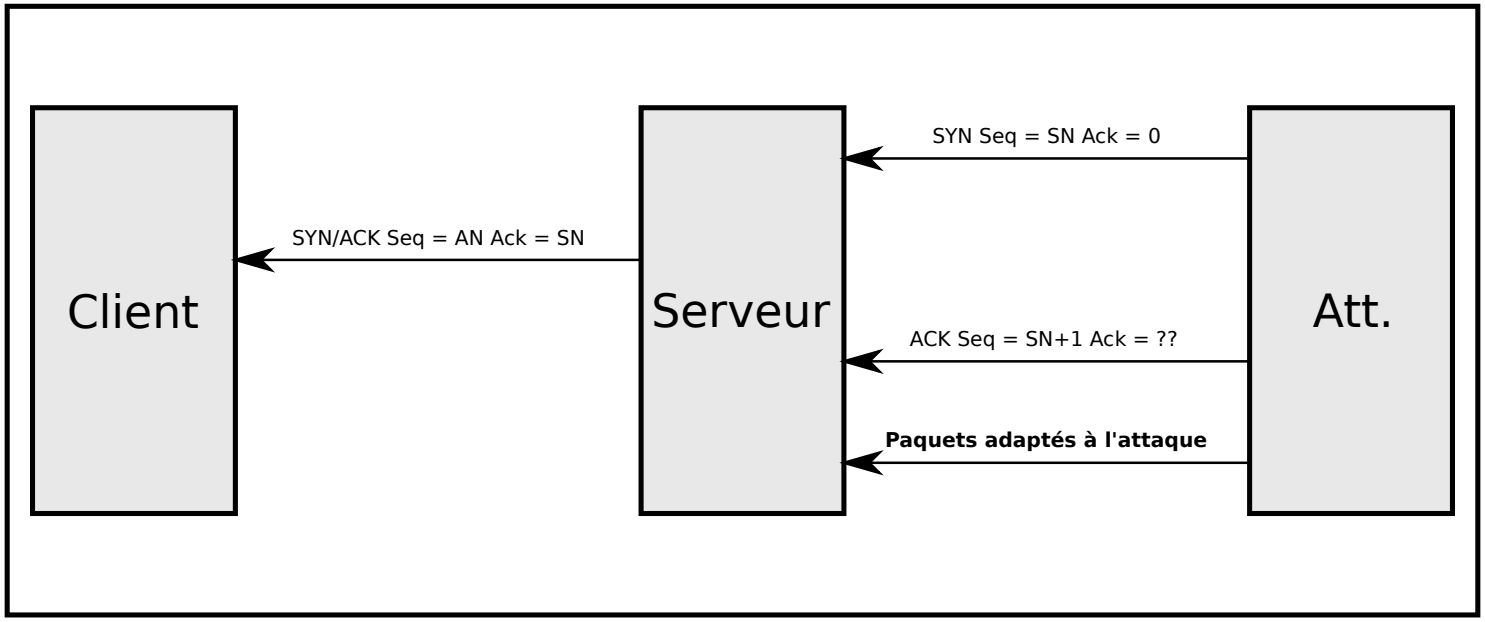
TCP Connection Spoofing (2/2)



TCP Connection Spoofing (2/2)



TCP Connection Spoofing (2/2)



Ça ne marche pas tel quel !

- Il faut connaître AN
  - En local on peut tenter l'écoute ou le détournement du trafic
  - En distant il faut trouver un moyen de le prévoir
- Le client va envoyer un RESET !

How ? L'attaque de Kevin Mitnick

Kevin Mitnick

Hacker de genie des années 90
Victimes : Pac. Bell, NorAD Command, Fujitsu, Motorola, etc.
Parmis les dix criminels les plus recherchés aux USA
Fait de la prison, maintenant co-fondateur de Defensive Thinking

Une époque dorée pour les hackers

On diffusait pas mal d'informations notamment par la commande `finger`
Il y avait la notion de noeuds de confiance /etc/rhosts (utilisé par `telnet`,`rsh`, `rcp`, `X`, etc.)

L'attaque de Tsutomu Shimomura (1995)

Attaque menée le jour de Noël 1994
Il apprend que l'utilisateur root de RIMMON est connecté sur OSIRIS sa cible
$\Rightarrow$ RIMMON est peut-être un noeud de confiance
Envoi de plusieurs SYN vers OSIRIS pour comprendre comment les AN sont générés
DoS sur RIMMON (SYN flood) puis TCP Connection Spoofing sur le port de `rsh`
Envoi un packet avec `echo + + > /etc/rhosts` qui s'exécute comme root ...

Et de nos jours ?

Découverte du réseau/services/connexions

Cartographie : cours RO
Plus compliqué mais faisable

Prédictions des numéros de séquence

Obtenus par des générateurs pseudo-aléatoires
À distance :

- Att. aux vieux systèmes comme NT !
- `nmap` → fingerprint OS puis recherche Internet :)

En local :

- Écoute (segment local, MAC flooding, ARP Spoofing) → AN

Et le payload ? `echo + + > /etc/rhosts` ?

Services "en r" fermés depuis longtemps ...
Mais en local on peut faire plus qu'envoyer un seul paquet !

Vol de session TCP

Limitations du connection spoofing

On peut envoyer des données mais pas recevoir les réponses
Le protocole au dessus (FTP, HTTP) peut demander une authentification

What ?

Émettre avec l'adresse IP d'autrui dans une connexion **déjà établie**

Pourquoi ça marche

On n'a pas besoin de deviner l'AN ou de s'authentifier
Pbs : SNs, ACKs, synchronisation
$\rightarrow$ Il faut être en local (au moins via un sniffer)

Cas le plus simple : MITM par ARP Spoofing déjà en place

Facile : une fois l'utilisateur authentifié on insère/modifie ce qu'on veut

Vol de session TCP : le Simple Active Hijack (1/2)

What ?

Désynchroniser les interlocuteurs au niveau des SNs

Why ?

Rendre la communication entre eux impossible et prendre la place

How ? Premier essai

Écouter pour obtenir le bon SN
Envoyer (au bon moment) un RST à A qui fermera la connexion
Envoyer des paquets à B avec l'adresse IP de A (vite)

Problèmes

B va générer des ACK et les envoyer à A ...
Qui va ... envoyer des RST (on est cuits)

Vol de session TCP : le Simple Active Hijack (2/2)

How ? L'attaque de Joncheray

Deux options :

- En début de connexion
  - Le client démarre une connexion avec le serveur
  - On coupe la connexion côté serveur (pas côté client)
  - On démarre la connexion côté serveur à nouveau

  $\Rightarrow$ Des nouveaux numéros de séquences sont générés

  Rq Il faut que le serveur accepte qu'une connexion plante
- À n'importe quel moment
  - On envoie des faux paquets au serveur avec les bons SN

  $\Rightarrow$ Le SN attendu par le serveur monte

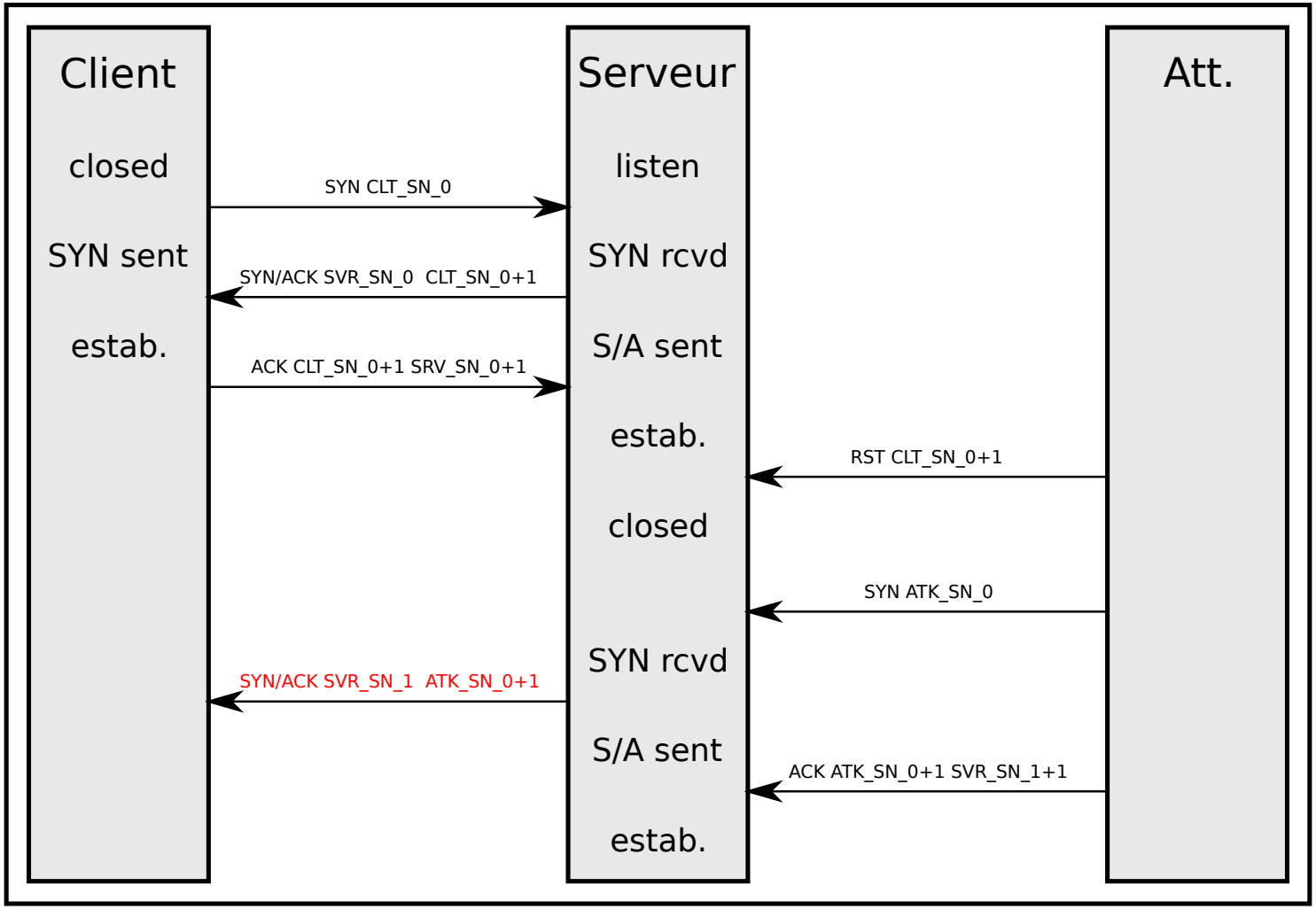
  Rq Il faut que ça ne fasse pas planter le serveur (plus dur)

Dans les deux cas on atteint une désynchronisation

$\Rightarrow$ Ils ne traitent pas leurs messages respectifs

$\Rightarrow$ L'attaquant traduit et remplace/introduit ce qu'il veut

Joncheray : en début de connexion



Joncheray : problèmes

ACK Storm

Paquet hors de la fenêtre TCP → ACK avec le SN et AN de récepteur
Même si c'est un ACK ! ! !

Si les interlocuteurs son désynchronisés chaque paquet de A est hors fenêtre pour B
Ce qui génère un ACK de B qui est hors fenêtre pour A
Ce qui génère un ACK de A qui est hors fenêtre pour B
...

ACKs pas renvoyés en cas de drop → processus fini (et auto-régulé)

Contres

Ingress filtering
Port Security

Fin !

Prochain cours

Wi-Fi ...