Question 1

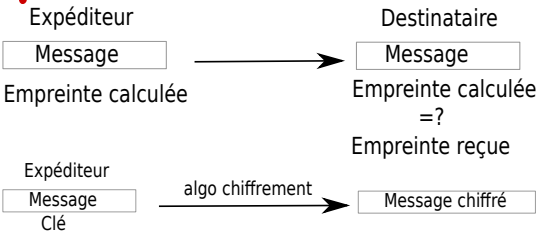
Non car intégrité = vérifier que la donnée est restée identique -> inutile de savoir qui s'en sert.

- A t'on besoin d'assurer l'intégrité pour pouvoir authentifier ? Non
- A t'on besoin d'authentification pour assurer l'intégrité ? Non

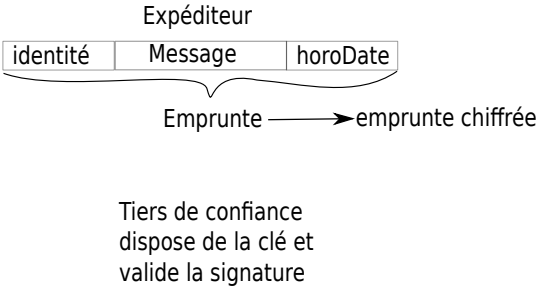
Question 2

- confidentialité Non
- authentification Oui car pour attribuer une action à quelqu'un il faut l'avoir authentifié
- intégrité Oui car il faut que les preuves soient intègres
- contrôle d'accès Non

Question 3



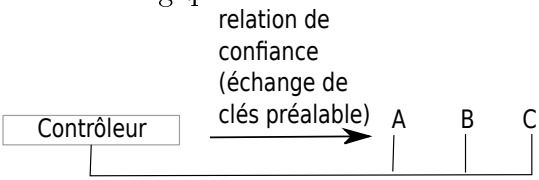
Autre méthode de confidentialité : authentification + contrôle d'accès



Question 4

Une contrôleur de domaine est un serveur chargé de l'authentification.

Domaine logique



Question 5 Domaine physique

Question 6

- Casiers judiciaires
- Dossiers médicaux
- Relevés bancaires
- Enquêtes de police
- Enquêtes fiscales
- Documents industriels confidentiels
- Délibérations du gouvernement

Les personnes concernées ont toujours le droit d'accéder à leurs informations personnelles.

Question 7

Commission Nationale Informatique et Liberté. La CNIL est une institution indépendante qui a pour rôle de faire respecter la loi informatique et libertés.

Question 8

La "politique de sécurité" est un document qui détermine les objets à sécuriser, identifie les menaces à prendre en compte, définit le périmètre de sécurité et spécifie l'ensemble des lois, règlements et pratiques à respecter pour assurer la sécurité.

Question 9

On écrit le plan, puis on le teste (on provoque une panne et on essaye de faire une reprise d'activité). C'est cette phase qui est la plus dure à réaliser.

- Vérifier l'intégrité des données
- Réparer les données corrompues
- Récupérer les données à partir d'archives

Question 10

La sécurité est l'affaire de tous. L'élément humain est le plus problématique car :

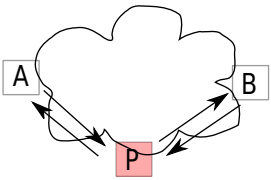
- Pas conscient des enjeux
- Sécurité = contraignant (ex : interdiction de clé USB, mdp complexes, sites web inaccessibles etc)

Question 11

- Perte de l'appareil ⇒ perte des données stockées ⇒ perte de confidentialité
- Utilisation de services cloud pour stocker des données sensibles
- Présence de virus sur la machine
- etc

Exemple : ARP spoofing

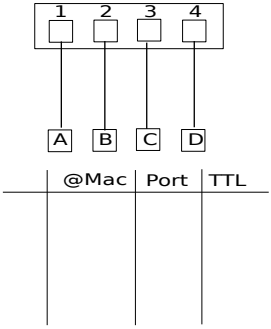
Définition : Man in the Middle est un terme générique désignant une attaque où l'attaquant peut accéder en lecture et en écriture à toutes les données circulant entre deux machines.



A et B pensent qu'ils communiquent directement alors que toutes les données passent par P.

ARP spoofing A et B sont reliés par l'intermédiaire d'un commutateur.

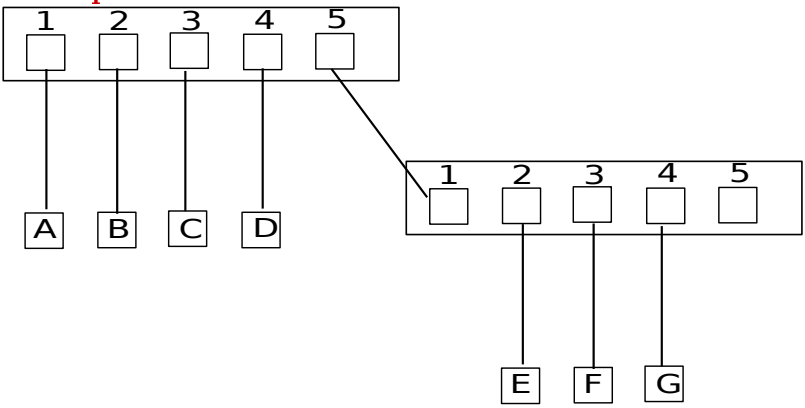
Rappel : fonctionnement d'un commutateur



Lorsqu'on aliment le commutateur la table est vide.
Le commutateur remplit sa table en inspectant les adresses MAC sources des trames qui circulent.

Si l'adresse MAC source n'est pas présente dans la table, il renvoie la trame sur tous les ports.

Exemple :



- A envoie à B
- E envoie à F
- B envoie à E
- G envoie à E
- Table de SW0

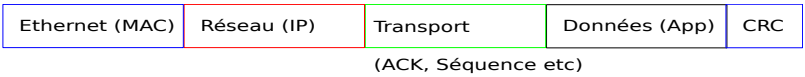
MAC	Port	TTL	Méthode
@A	1	100	Flooding
@E	5	100	Flooding
@B	2	100	Forwarding port 5

- Table de SW1

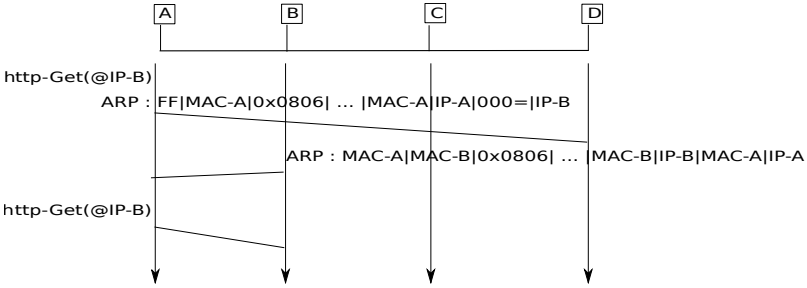
MAC	Port	TTL	Méthode
@A	1	100	Flooding
@E	2	100	Flooding
@B	1	100	Forwarding port 2
@G	4	100	Forwarding port 2

Rappel : fonctionnement du protocole ARP et du modèle OSI

- Couches :
  - Application
  - Transport - Fiabilité de bout en bout - TCP
  - Réseau - Trouver le chemin - IP
  - Liaison - Dépend du média et permet d'assurer une transmission fiable sur ce média - Ethernet, Wifi etc.
  - Physique - Caractéristiques mécaniques et électriques du média et es signaux circulant dessus



- ARP(Adress Resolution Protocol) permet de connaitre l'adresse MAC d'une machine, connaissant son adresse IP. Le fonctionnement générique : envoi d'une requête ARP en broadcast, la machine qui reconnait son IP répond.



ARP spoofing : spoofing ou cache poisonnig = usurpation

Méthode 1 : Le pirate envoie des réponses ARP en continu.

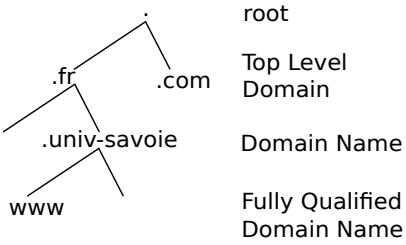
- Réponse émise à destination de A : @MAC-A|@MAC-P|0x0806| . . . | @MAC-P | @IP-B | @MAC-A | @IP-A | . . .
- Exemple de trame envoyée par A à destination de B @MAC-C|@MAC-A|0x0800| ....|@IP-A|@IP-B| ....
- Réponse émise à destination de B : @MAC-B|@MAC-P|0x0806| . . . | @MAC-P | @IP-A | @MAC-B | @IP-B | . . .
- Exemple de trame envoyée par B à destination de A @MAC-C|@MAC-B|0x0800| ....|@IP-B|@IP-A| ....

Méthode 2 : Le pirate envoie des requêtes ARP en continu. En effet, dès qu'une machine reçoit une requête, elle enregistre les infos sur la source dans son cache ARP.
Trames émises par C

- @MAC-A | @MAC-C | 0x0806 | . . . | @MAC-C | @IP-B | 00 | IP-A
- @MAC-B | @MAC-C | 0x0806 | . . . | @MAC-C | @IP-A | 00 | IP-B

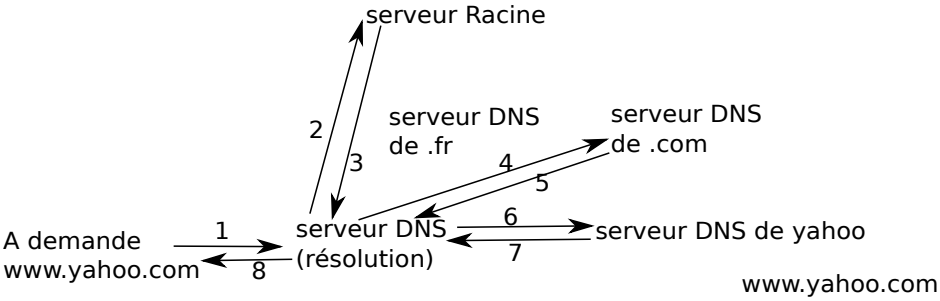
Comment l'éviter ? Segmenter le réseau (ne pas rester tous sur le même média). On isole les différents groupes d'utilisateurs (60 machines maximum sur un même segment).

Rappel : DNS Domain Name System = Résolution de nom (facilite la mémorisation et permet de mémoriser le type de service). Ajoute un niveau d'abstraction permettant d'identifier une machine indépendemment de sa localisation.



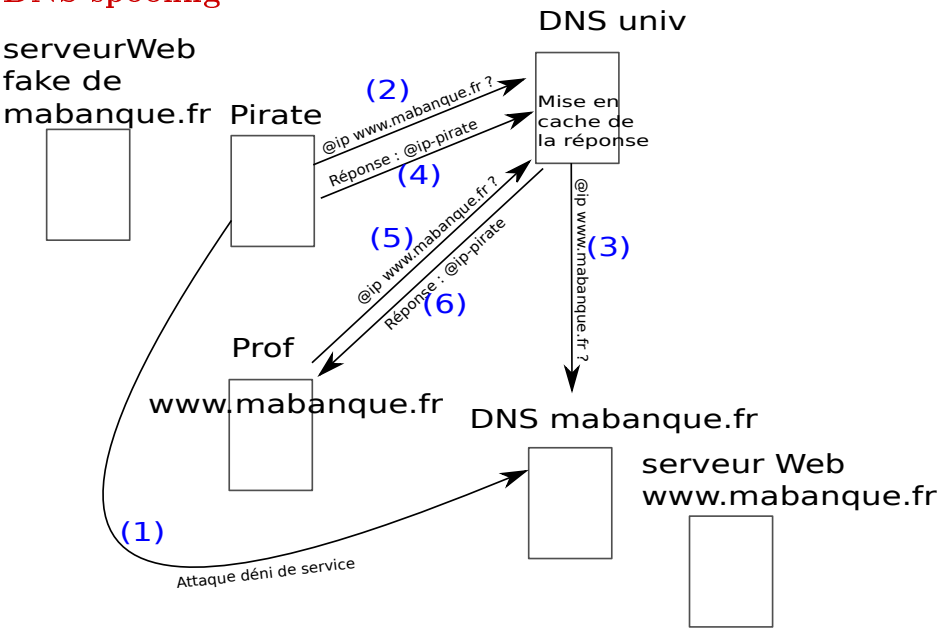
Les serveurs DNS ont 2 rôles :

- Faire de la résolution
- Héberger des informations de zone



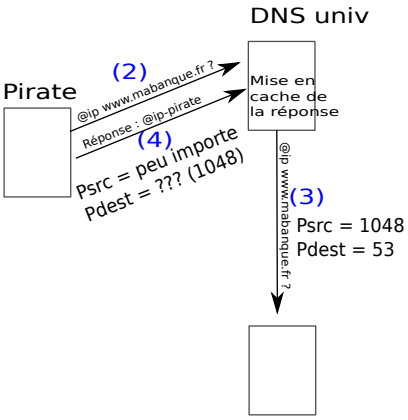
Rappel : La durée de mise en cache est renvoyée avec la réponse du serveur DNS hébergeant l'information.

DNS-spoofing



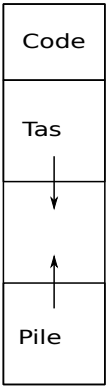
Remarque : les concepteurs de DNS ont prévu cette attaque : une id est ajoutée à la demande afin de s'assurer que celui qui répond est bien celui à qui on a demandé. Problème : ces id sont codés sur 16 bits donc il est devenu facile d'envoyer les 65 535 réponses possibles en un temps très court.

Parade : On fait intervenir la couche transport pour "compléter" l'id : les 2 protocoles disponibles pour cette couche sont UDP et TCP, ces 2 protocoles ont des champs communs (port source : identifie l'application sur la machine émettrice / port dest : identifie l'application sur la machine réceptrice). Le pirate doit deviner les 16 bits à mettre dans port dest => ça fait 16 + 16 = 32 bits à deviner => 4 milliards de réponses possibles.



Remarque : Avant les serveurs DNS mettaient toujours 53 en port dest.

Organisation de la RAM pendant l'exécution d'un processus



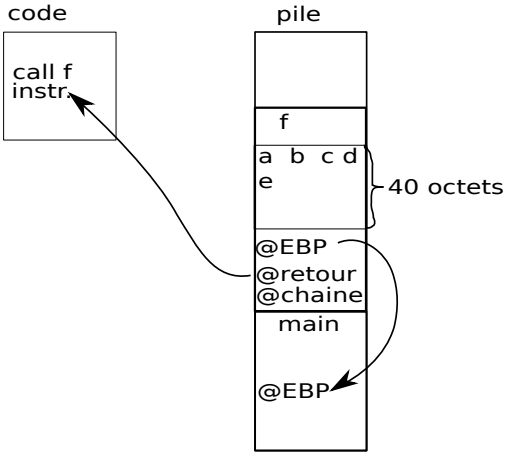
EIP Pointeur d'instructions.

EBP Pointeur de base : les variables locales sont référencées par rapport à EBP (ex : la variable x est à EBP - 3)

Organisation de l'espace d'une fonction dans la pile :

Variables locales
EBP
Adresse de retour (instruction suivante dans le code)
Arguments de la fonction

Exemple :


code
pile
call f
instr
f
a b c d
e
40 octets
@EBP
@retour
@chaine
main
@EBP

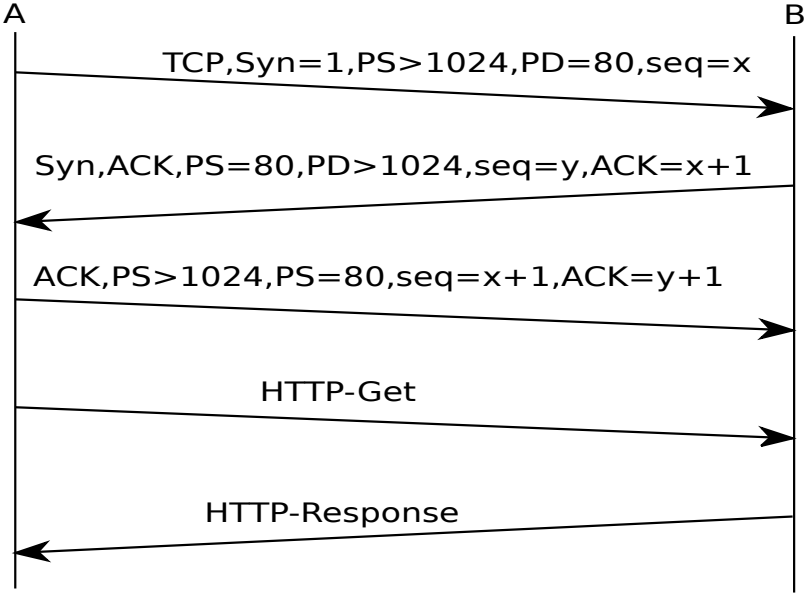
Définition : Logiciel ou boitier matériel conçu pour filtrer les échanges de données entre un réseau / une application de confiance et une réseau / une application extérieur.

Couches concernées :

- entête Réseau (IP)
- entête Transport (TCP, UDP)
- charge utile (filtrage d'url, type de fichier)

On peut placer un pare feu en entrée d'un réseau de confiance et/ou en entrée d'une machine.

Rappel : Connexion http classique

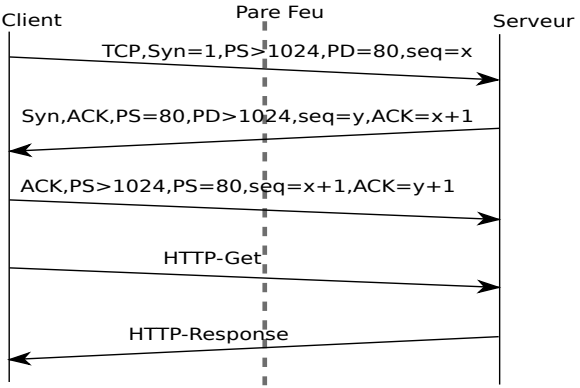


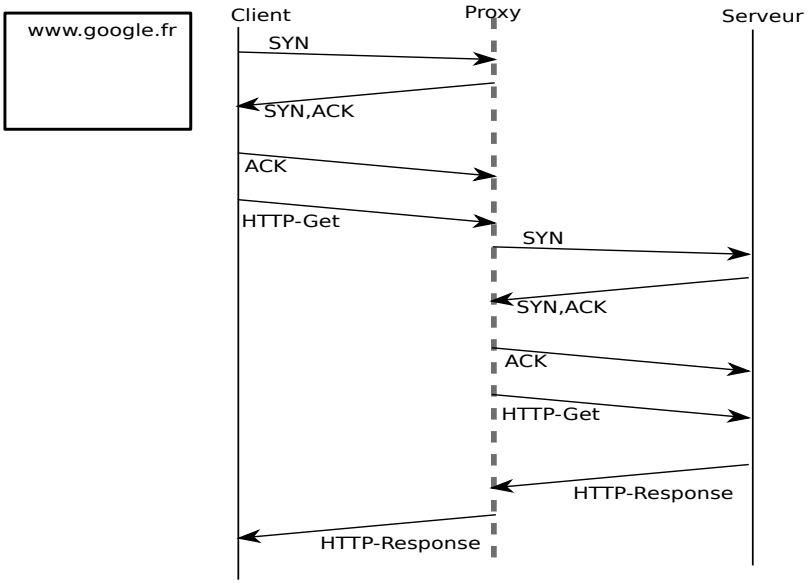
Autorisation de filtrage dans B:

1. Autoriser PS>1024, PD=80, SYN=1
2. Autoriser PS=80, PD>1024, SYN=1, ACK=1, @IP-B, @IP-A(toutes du réseau autorisé)
3. Autoriser PS>1024, PD=80, ACK=1, @IP-A(toutes du réseau autorisé) @IP-B
4. Autoriser PS=80, PD>1024, ACK=1, @IP-B, @IP-A(toutes du réseau autorisé)

Historique :

- Filtrage statique : ils n'avaient pas de mémoire pour enregistrer l'état d'une connexion (on ne peut pas vérifier les numéros de séquence par exemple). Ce type de filtre est plus efficace en bloquage qu'en autorisation.
- Proxy : Apparu en même temps que le filtrage statique. A l'époque, ils ont été créés pour économiser les adresses IP (avant, les IP étaient organisées par classes). Le proxy permet de donner un accès au web à toutes les machines d'un réseau local avec une seule adresse publique. Puisque toutes les requêtes passent par lui, il peut les filtrer. Actuellement les proxy sont encore utilisés comme serveurs de cache, filtres d'url et de contenu.





Il faut un Proxy par type d'application car il doit connaitre les protocoles de la couche application (ex: HTTP).

Cas particulier : les proxy socks :

IP	UDP	Socks @client @serveurfinal @serveursocks protocole transport port dest	HTTP-GET

- Filtrage dynamique (autorise le premier paquet, et ceux qui suivent logiquement). Le pare feu possède une mémoire qui enregistre un suivi de connexion.

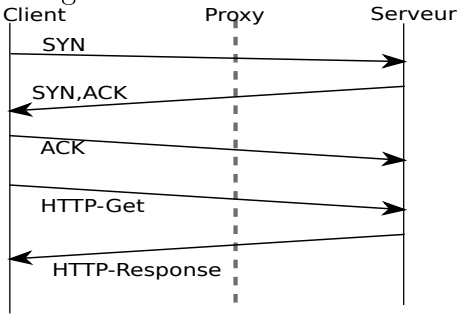


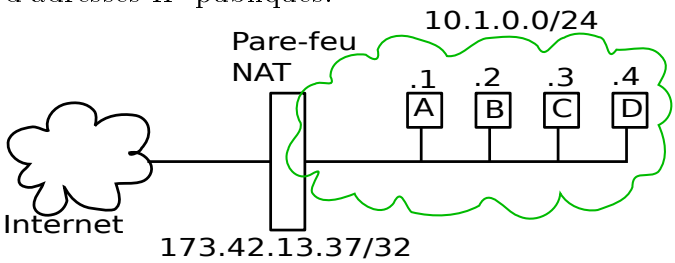
Table de suivi de connexion
IP-S = IP-A
IP-Dest = IP-B
Port-Source = 1025
Port-Dest = 80 =
Num Seq =
Num ACK =
SYN = 1

Lorsque le pare feu reçoit la réponse il l'examine et vérifie que tous les champs sont en correspondance dans la mémoire.

Remarque : puisque le pare feu doit enregistrer les IP et les ports, c'est généralement sur lui qu'on implémente la translation d'adresses.

Rappel : Translation d'adresse Elle sert à palier la pénurie

d'adresses IP publiques.



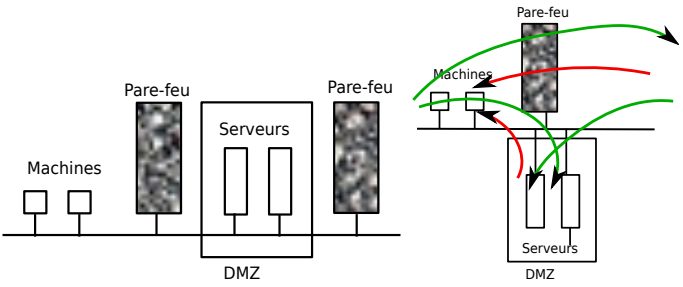
Scénario :

- A envoie :
  |4 . . . |10.1.0.1|1.2.3.4|1025|80|. . . | à l'adresse IP 1.2.3.4.
- Le paquet est transformé :
  |4 . . . |173.42.13.37|1.2.3.4|1025|80|. . . |
- B envoie :
  |4 . . . |10.1.0.2|1.2.3.4|1025|80|. . . | à l'adresse IP 1.2.3.4.
- Le paquet est transformé :
  |4 . . . |173.42.13.37|1.2.3.4|1026|80|. . . | à l'adresse IP 1.2.3.4.
- 1.2.3.4 répond :
  |4 . . . |1.2.3.4|173.42.13.37|80|1025|. . . |
- Le pare feu sait que c'est pour A grâce au port-dest 1025.
- Le paquet est transformé et renvoyé à A :
  |4 . . . |1.2.3.4|10.1.0.1|80|1025|. . . |
- 1.2.3.4 répond :
  |4 . . . |1.2.3.4|173.42.13.37|80|1026|. . . |
- Le pare feu sait que c'est pour B grâce au port-dest 1026.

- Le paquet est transformé et renvoyé à A :
  |4 . . . |1.2.3.4|10.1.0.2|80|1025|. . . |

Interne		Externe
IP-Src	Port-Src	Port-Src
10.1.0.1	1025	1025
10.1.0.2	1025	1026

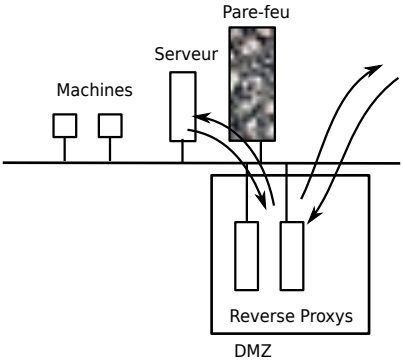
En lui même, le translateur d'adresses constitue déjà une sécurité, car une machine à l'extérieur du réseau privé ne peut pas contacter une machine à l'intérieur si la machine à l'intérieur n'a pas initié de connection.
Si on veut héberger un serveur dans le réseau interne, il faut saisir manuellement une ligne IP-Serveur, Port 80, Port 80 dans la table de translation et autoriser les connections entrantes sur le port 80.
Problème : Un pirate qui accède à notre serveur http entre dans le réseau local, de là il peut potentiellement accéder aux autres machines du même réseau. Il faut un autre pare feu entre les serveurs et le reste du réseau local.



Reverse Proxy : Serveurs cache spécialisés dans l'hébergement web. Ils vont avoir 2 rôles :

- Augmenter la sécurité et la fiabilité

- Faire de la répartition de charge



SYN Flooding : Déni de service (peut être coupé par un pare-feu dynamique mais pas simple)

Principe du chiffrement asymétrique :

- Je donne ma clé publique
- L'autre donne sa clé publique.
- Je chiffre avec la clé publique du destinataire et il déchiffre avec sa clé privée
- Le destinataire chiffre avec ma clé publique et je déchiffre avec ma clé privée

Différences :

Symétrique	Asymétrique
plus rapide (la clé est sur 128bits)	lent (la clé est sur 2048bits)
phase critique : échange de la clé	échange des clés facilités
la confidentialité est assurée	il faut s'assurer de l'identité de l'émetteur de la clé publique

Code chiffré : On dit qu'il est pseudo-aléatoire :

- pseudo : ce n'est pas aléatoire
- aléatoire : ça a l'air aléatoire (on ne peut pas reconnaitre des motifs)

DES : Data Encryption Service

AES : Advanced Encryption Service

Problématique du changement d'algorithme : Ce sont des puces qui réalisent le cryptage, donc il faut du matériel équipé de ces puces.

Pièges : Attention : il ne faut pas chiffrer par caractère, car il est alors facile de trouver la correspondance 'caractère' / 'caractère chiffré'.

CBC : Cipher Bloc Chaining

Exemple : Chiffrement d'une connexion WIFI.

@MAC-D	@MAC-Src	Ethertype	VI	... zone chiffrée ...

Côté récepteur, on prend la clé et on la concatène avec le VI reçu et on génère le même R que côté émetteur.
On fait un XOR entre le R et le message reçu, on obtient le message en clair.
Il faut connaitre la clé pour générer le R, et ensuite faire le XOR.
Le pirate connait le VI mais pas la clé, donc il ne peut pas déchiffrer.
Le R est toujours différent donc le pirate ne peut pas reconnaitre des motifs en se basant sur le format classique des trames.

Faille principale du WEP : Le VI est sur 24 bits, donc environ 16 millions de possibilités. On a à peu près 20 Mbits/s de débit utile sur le réseau, la taille moyenne d'une trame est 1000 octets = 8000 bits.
Combien de trames par secondes ? $20.10^6/8.10^3 = 2500$
Au bout de combien de temps les VI rebouclent ? $16.10^6/2500 =$

$6400s = 1h45$

Comme ça reboucle fréquemment, il est impossible d'empêcher de rejouer les VI.

Exemple : Le pirate va capturer une trame ARP avec un VI donné. Il capture C = M XOR R. Il devine M car il connait les trames ARP. Il calcule R = C XOR M. Il connait R_{VI_1}. Il émet des trames en utilisant ce R_{VI_1}. Lorsque le point d'accès répond avec un autre VI, or le pirate connait la réponse donc il peut calculer R_{VI_2}. A la fin il peut posséder une table de correspondance pour chaque VI (sans avoir eu besoin de découvrir la clé).