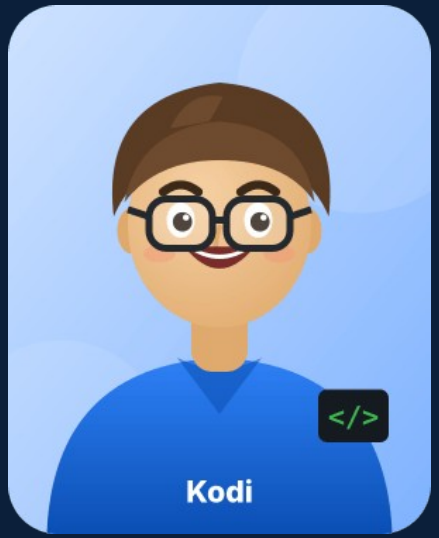
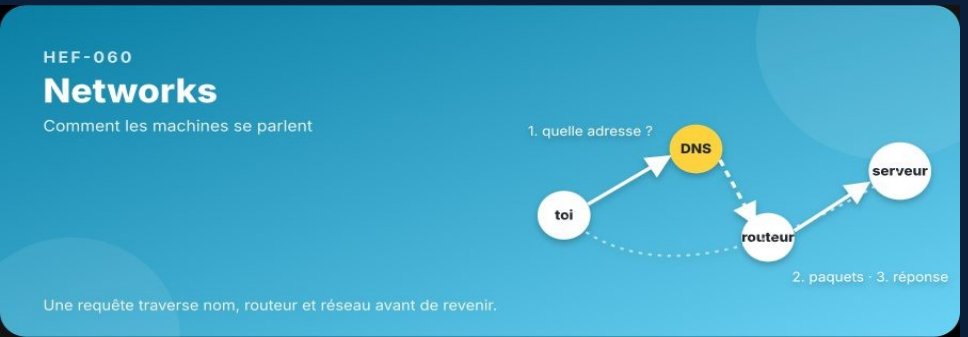
HEF-060 · MISSION 09 / 10

NAT, pare-feu et services

Comment des centaines d'appareils partagent une seule adresse publique, et comment on décide qui entre.



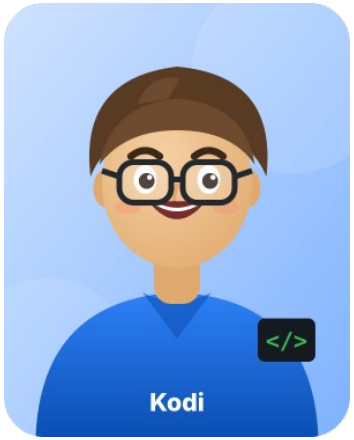


Badge : Gardien de la frontière

80 min · intermédiaire

L'histoire

À la maison, toute la famille de Kodi utilise la même connexion : une seule adresse IP publique, et pourtant dix appareils sont en ligne. Au bureau, un administrateur dit : « Le serveur de test n'est pas joignable de l'extérieur : c'est voulu. »



> *« Deux mécanismes expliquent ça, dit Sam : le NAT traduit les adresses, le pare-feu filtre le trafic. »*

Tu comprends comment on partage une adresse publique et comment on protège un réseau.

Ta mission

À la fin, je serai capable de…

1. expliquer le NAT/PAT et la table de traduction
2. expliquer pourquoi une machine privée n'est pas joignable de l'extérieur par défaut
3. décrire un pare-feu à règles ordonnées (première correspondance)
4. distinguer pare-feu sans état et avec état
5. concevoir une politique « tout refuser sauf le nécessaire »

Défi d'échauffement

Un immeuble n'a qu'**une seule adresse postale** (le numéro 12). Le concierge reçoit tout le courrier. Quand un habitant envoie une lettre, le concierge note « lettre de l'appartement 3 vers Paris » sur un carnet. Quand une réponse arrive, comment sait-il à quel appartement la donner ? Qu'arrive-t-il à une lettre qui arrive sans correspondre à aucune demande ?

Le concept

Pourquoi le NAT ?

Il n'y a pas assez d'adresses IPv4 publiques pour tous les appareils. Les réseaux locaux utilisent des adresses **privées** (`192.168.x.x`, `10.x.x.x`…) qui ne sont pas routées sur internet. Le **NAT** (*Network Address Translation*) permet à plusieurs machines privées de partager **une** adresse publique.

PAT : le NAT avec ports

En pratique, on traduit aussi les **ports** (*PAT*, *NAT overload*). Le routeur garde une **table de traduction** :

Interne (IP:port)	Externe (IP publique:port)	Destination
192.168.1.10:51000	203.0.113.5:40001	93.184.216.34:443
192.168.1.11:51000	203.0.113.5:40002	93.184.216.34:443

Le concept

PAT : le NAT avec ports (suite)

À l'**aller**, le routeur remplace la source par `203.0.113.5:40001`. Au **retour**, il consulte la table et remet l'adresse interne.

Conséquences

- Les connexions **sortantes** fonctionnent naturellement.
- Les connexions **entrantes** vers une machine privée ne correspondent à aucune entrée : elles sont **rejetées**. C'est un effet de protection (mais le NAT n'est **pas** un pare-feu à lui seul).
- Pour exposer un service : **redirection de port** (*port forwarding*) : « tout ce qui arrive sur le port 443 de l'adresse publique va vers 192.168.1.50:443 ».
- Les protocoles qui incluent des adresses dans leurs données, ou la communication directe entre deux machines derrière des NAT, sont plus compliqués.

(IPv6 offre assez d'adresses pour s'en passer ; le filtrage devient alors la seule protection.)

Le concept

Le pare-feu

Un **pare-feu** (*firewall*) filtre les paquets selon des **règles** :

#	Action	Protocole	Source	Destination	Port
1	ACCEPTER	TCP	10.0.0.0/8	serveur	22
2	ACCEPTER	TCP	tout	serveur	443
3	REFUSER	tout	tout	tout	tout

- Les règles sont évaluées **dans l'ordre** ; **la première qui correspond s'applique**.
- La dernière règle est la **politique par défaut**. Bonne pratique : **tout refuser** sauf ce qui est explicitement autorisé (*default deny*).

Le concept

Sans état ou avec état

- **Sans état** (*stateless*) : chaque paquet est jugé seul. Il faut autoriser explicitement les réponses.
- **Avec état** (*stateful*) : le pare-feu **mémorise les connexions** ; les réponses à une connexion autorisée passent automatiquement.

Où placer un pare-feu ?

- au bord du réseau (entre internet et le réseau interne) ;
- sur chaque machine (pare-feu local, `ufw`, `iptables`, `nftables`) ;
- entre les zones internes (**segmentation**) : une machine compromise ne peut pas atteindre tout le reste.

Le concept

Modéliser en Python

```
import ipaddress

REGLES = [
    ("ACCEPTER", "tcp", "10.0.0.0/8", 22),
    ("ACCEPTER", "tcp", "0.0.0.0/0", 443),
    ("REFUSER", "*", "0.0.0.0/0", None),
]

def decision(protocole, ip_source, port_dest):
    for action, proto, reseau, port in REGLES:
        if proto in ("*", protocole) and ipaddress.ip_address(ip_source) in ipaddr
   ess.ip_network(reseau) \
           and (port is None or port == port_dest):
            return action
    return "REFUSER"
```

Le concept

Modéliser en Python (suite)

```
print(decision("tcp", "10.1.2.3", 22))        # ACCEPTER
print(decision("tcp", "203.0.113.9", 22))     # REFUSER
print(decision("tcp", "203.0.113.9", 443))    # ACCEPTER
```

Exemple pas à pas

Problème : simuler un routeur NAT avec table de traduction.

```
class RouteurNAT:
    def __init__(self, ip_publique, port_depart=40000):
        self.ip_publique = ip_publique
        self.prochain_port = port_depart
        self.sortant = {}       # (ip_interne, port_interne, ip_dest, port_dest) -
   > port_externe
        self.entrant = {}       # port_externe -> (ip_interne, port_interne, ip_de
   st, port_dest)

    def sortir(self, ip_interne, port_interne, ip_dest, port_dest):
        cle = (ip_interne, port_interne, ip_dest, port_dest)
        if cle not in self.sortant:
            self.sortant[cle] = self.prochain_port
            self.entrant[self.prochain_port] = cle
            self.prochain_port += 1
        return (self.ip_publique, self.sortant[cle])
```

Exemple pas à pas (suite)

```
    def entrer(self, ip_source, port_source, port_externe):
        cle = self.entrant.get(port_externe)
        if cle is None or (cle[2], cle[3]) != (ip_source, port_source):
            return "rejeté (aucune connexion correspondante)"
        return (cle[0], cle[1])

nat = RouteurNAT("203.0.113.5")
a = nat.sortir("192.168.1.10", 51000, "93.184.216.34", 443)
b = nat.sortir("192.168.1.11", 51000, "93.184.216.34", 443)
print(a, b)                                          # deux ports externes différe
   nts
print(nat.entrer("93.184.216.34", 443, a[1]))        # réponse : ('192.168.1.10',
   51000)
print(nat.entrer("198.51.100.9", 4444, a[1]))        # inconnu : rejeté
print(nat.entrer("93.184.216.34", 443, 12345))       # port non traduit : rejeté
```

Lecture :

Exemple pas à pas (suite)

- deux machines utilisent **le même port interne** (51000), mais le routeur leur attribue des **ports externes différents** pour s'y retrouver ;
- une réponse légitime est correctement renvoyée à la bonne machine ;
- un paquet non sollicité est **rejeté** : le NAT agit comme un pare-feu à états rudimentaire.

Règles de pare-feu : l'ordre compte :

Ordre des règles	Effet pour le trafic depuis 203.0.113.9 vers le port 22
1. REFUSER tout · 2. ACCEPTER 10.0.0.0/8	refusé, et **personne** n'est jamais accepté (la règle 2 n'est jamais atteinte)
1. ACCEPTER 10.0.0.0/8 · 2. REFUSER tout	refusé pour cette adresse, accepté pour le réseau interne

À toi, avec aide

Écris les règles d'un pare-feu pour un serveur web :

- le public peut joindre les ports 80 et 443 ;
- l'administration SSH (port 22) n'est accessible que depuis `192.168.1.0/24` ;
- tout le reste est refusé.

Teste avec 6 paquets : `203.0.113.9:443`, `203.0.113.9:22`, `192.168.1.77:22`, `203.0.113.9:8080`, `192.168.2.5:22`, `198.51.100.1:80`.

À toi, avec aide · réponse

Indices progressifs

Indice 1

Mets les règles spécifiques d'abord, la règle `tout refuser` en dernier.

Indice 2

Utilise `ipaddress.ip_network` pour les réseaux.

Indice 3

Le résultat attendu : A, R, A, R, R, A.

À toi, avec aide · réponse

Réponse

```
import ipaddress

REGLES = [
    ("ACCEPTER", 80, "0.0.0.0/0"),
    ("ACCEPTER", 443, "0.0.0.0/0"),
    ("ACCEPTER", 22, "192.168.1.0/24"),
    ("REFUSER", None, "0.0.0.0/0"),
]

def decision(ip, port):
    for action, p, reseau in REGLES:
        if (p is None or p == port) and ipaddress.ip_address(ip) in ipaddress.ip_n
   etwork(reseau):
            return action

tests = [("203.0.113.9", 443), ("203.0.113.9", 22), ("192.168.1.77", 22),
         ("203.0.113.9", 8080), ("192.168.2.5", 22), ("198.51.100.1", 80)]
print([decision(*t)[0] for t in tests])     # ['A', 'R', 'A', 'R', 'R', 'A']
```

À toi, en solo

Pare-feu avec état. Étends le pare-feu pour qu'il :

1. autorise les connexions **sortantes** (réseau interne → externe) et mémorise chaque connexion ;
2. autorise les paquets **entrants** seulement s'ils appartiennent à une connexion mémorisée, ou s'ils visent un service public (80, 443) ;
3. oublie une connexion après un délai d'inactivité (temps simulé).

Teste : un client interne ouvre une connexion vers un serveur ; la réponse passe ; un autre paquet venant du même serveur vers un autre port est refusé ; après expiration, la réponse est refusée.

À toi, en solo · réponse

Pistes de correction

```
class PareFeuEtat:
    def __init__(self, delai=60):
        self.delai = delai
        self.connexions = {}                # (ip_int, port_int, ip_ext, port_ext
   ) -> dernière activité

    def sortant(self, t, ip_int, port_int, ip_ext, port_ext):
        self.connexions[(ip_int, port_int, ip_ext, port_ext)] = t
        return "ACCEPTER"

    def entrant(self, t, ip_ext, port_ext, ip_int, port_int):
        if port_int in (80, 443):
            return "ACCEPTER (service public)"
        cle = (ip_int, port_int, ip_ext, port_ext)
        if cle in self.connexions and t - self.connexions[cle] <= self.delai:
            self.connexions[cle] = t
            return "ACCEPTER (réponse)"
        return "REFUSER"
```

Pistes de correction (suite)

```
pf = PareFeuEtat(delai=60)
print(pf.sortant(0, "192.168.1.10", 51000, "93.184.216.34", 443))
print(pf.entrant(5, "93.184.216.34", 443, "192.168.1.10", 51000))      # réponse a
   cceptée
print(pf.entrant(6, "93.184.216.34", 8080, "192.168.1.10", 51000))     # autre por
   t : refusé
print(pf.entrant(100, "93.184.216.34", 443, "192.168.1.10", 51000))    # trop tard
    : refusé
```

Pièges classiques

Piège	Pourquoi c'est un problème	Comment l'éviter
Penser que le NAT protège	Il n'est pas conçu pour la sécurité	Toujours un vrai pare-feu
Règles dans le mauvais ordre	Une règle générale masque les spécifiques	Du plus précis au plus général, deny à la fin
Ouvrir trop de ports	Surface d'attaque élargie	N'ouvrir que le nécessaire
Oublier le trafic de retour	Services qui ne répondent pas (sans état)	Pare-feu avec état ou règles explicites

Mini-boss

Mini-boss

Architecture réseau de HashBox. Zones : internet, DMZ (serveur web), réseau interne (base de données, postes), administration.

1. Définis les plages d'adresses (avec `ipaddress`) et les flux autorisés dans une **matrice de flux** (qui peut parler à qui, sur quel port).
2. Écris une classe `Politique` qui évalue un flux `(source, destination, port)` et renvoie la règle appliquée.
3. Teste avec 12 flux, dont des flux interdits (internet → base de données, DMZ → réseau administration).
4. Décris ce qui se passe si le serveur web de la DMZ est compromis : que peut atteindre l'attaquant ?

Mini-boss · réponse

Pistes de correction

Un bon travail : internet → DMZ (80/443) ; DMZ → base de données (5432 seulement depuis le serveur web) ; administration → DMZ et interne (22) ; interne → internet (sortant) ; tout le reste refusé. La classe évalue les règles dans l'ordre et renvoie la règle appliquée pour l'explicabilité. Si le serveur web est compromis, il ne peut atteindre que le port de la base prévu, pas les postes ni l'administration : la segmentation limite le rayon d'action. Le dossier précise les journaux à surveiller sur les règles de refus.

Quiz éclair

Quiz éclair

1. Le NAT permet de :

A. chiffrer le trafic
B. faire partager une adresse publique à plusieurs machines privées
C. trouver l'adresse MAC

2. Dans un pare-feu à règles ordonnées, c'est :

A. la dernière règle qui s'applique
B. la plus longue règle qui s'applique
C. la première règle qui correspond qui s'applique

3. La politique « default deny » consiste à :

A. tout refuser sauf ce qui est explicitement autorisé

Quiz éclair

Quiz éclair (suite)

B. tout accepter sauf le danger
C. désactiver le pare-feu

4. 🔁 *Révision mission 08* : Une clé publique peut être :

A. gardée secrète
B. partagée avec tout le monde
C. utilisée uniquement pour déchiffrer

5. 🔁 *Révision mission 08* : Un certificat lie :

A. une adresse IP à une adresse MAC
B. un mot de passe à un utilisateur
C. un nom de domaine à une clé publique

Quiz éclair · réponse

Réponses

1. **B**.
2. **C**.
3. **A**.
4. **B**.
5. **C**.

Je réfléchis

1. Quels « pare-feu » personnels utilises-tu (filtrer les appels, les messages) ?
2. Pourquoi refuser par défaut est-il plus sûr qu'accepter par défaut ?

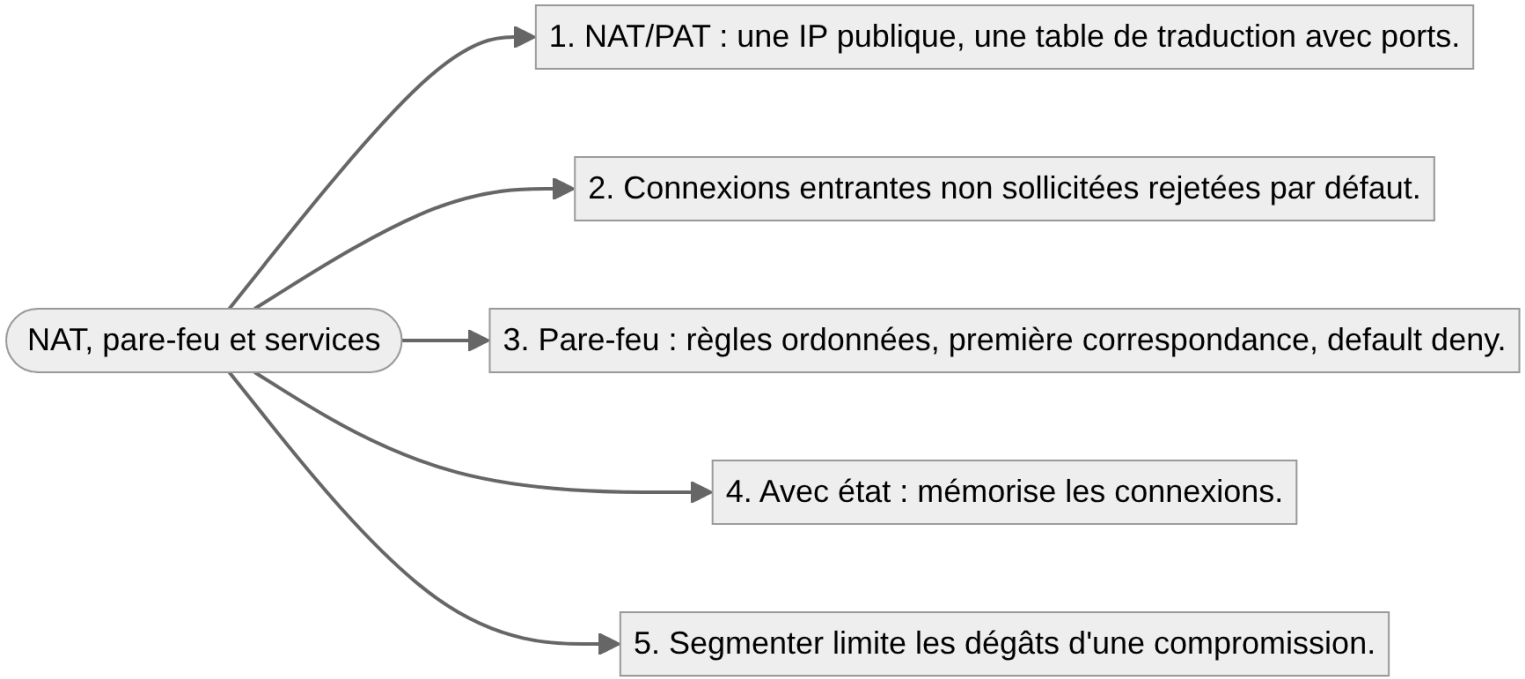
À retenir

À retenir

1. NAT/PAT : une IP publique, une table de traduction avec ports.
2. Connexions entrantes non sollicitées rejetées par défaut.
3. Pare-feu : règles ordonnées, première correspondance, default deny.
4. Avec état : mémorise les connexions.
5. Segmenter limite les dégâts d'une compromission.

À retenir

Le schéma de la mission



MISSION ACCOMPLIE

+100 XP

Badge débloqué : Gardien de la frontière

Prochaine mission : Analyse de trafic et dépannage

