

GNU / Linux — Routage, Firewall et Blacklisting

Netfilter • iptables • NAT • Fail2Ban • Null route

Objectifs du cours

Avoir le minimum de base pour filtrer du trafic.

Gérer un firewall.

Filtrer le trafic.

Créer des règles.

Accepter des packets.

Rejeter des paquets.

GNU / Linux – Routage et Firewall

- Objectifs
 - Avoir le minimum de base pour filtrer du trafic
- Gérer un firewall
 - Filtrer le trafic
- Créer des règles
 - Accepter des packets
 - Rejeter des paquets

GNU / Linux – Firewall

- 2 grandes familles
 - Netfilter
- Composant du noyau
- Travaille directement sur la pile TCP/IP
 - Généralement en couches 3/4 OSI
- Réseau : niveau paquet
- Transport : niveau Segment TCP et Datagramme UDP
- Couches 2/3 TCP/IP
 - Iptables
- Outils du Userland
- Envoie des règles à Netfilter
 - Dont il implémente presque toutes fonctionnalités
- Sauf le conntrack (à valider)

GNU / Linux – Firewall

- Iptable s'appuie sur des... tables (Surprise)
- 3 grandes fonctionnalités dans iptables
 - Filter
- Filtrage des paquets
- Tables par défaut

- NAT/NAPT
- Altération de l'entête IP
 - Paquet pour l'IP destination
 - Segment TCP ou Datagramme UDP pour le port
 - Mangle
- Modification et marquage des packets

GNU / Linux – Firewall

- Les chaînes et les tables
 - Tables :
- Le type d'action que l'on va effectuer
 - Chaînes :
- Une liste de règles dans un ordre précis
- Sur quel type de trafic on va agir
 - Différentes chaînes dans une même table
 - Filtres :
- INPUT
- OUTPUT
- FORWARD

GNU / Linux – Firewall

- Les règles et les cibles
 - Règles :
- Permettent de sélectionner les paquets
 - Une IP, un rang IP, un masque CIDR, un sous-réseau...
 - Un port, protocole TCP/UDP...
 - Cible :
- Qu'est ce qu'on va faire avec les paquets sélectionnés
 - Selon la table et la chaîne
- Certaines règles et chaînes ne seront pas disponibles
 - Par exemple
- Une interface réseau d'entrée
- Avec un filtre OUTPUT

GNU / Linux – Firewall

- Schéma de fonctionnement synthétique

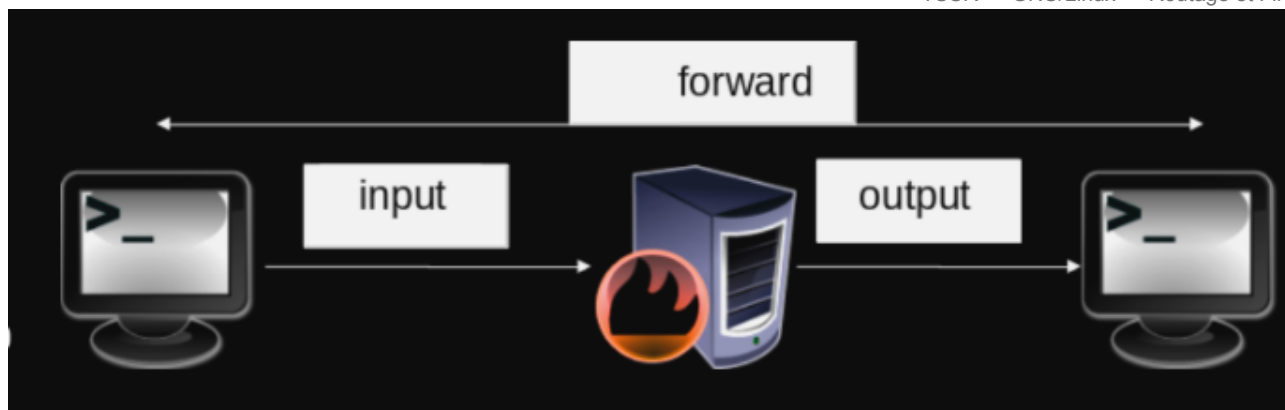


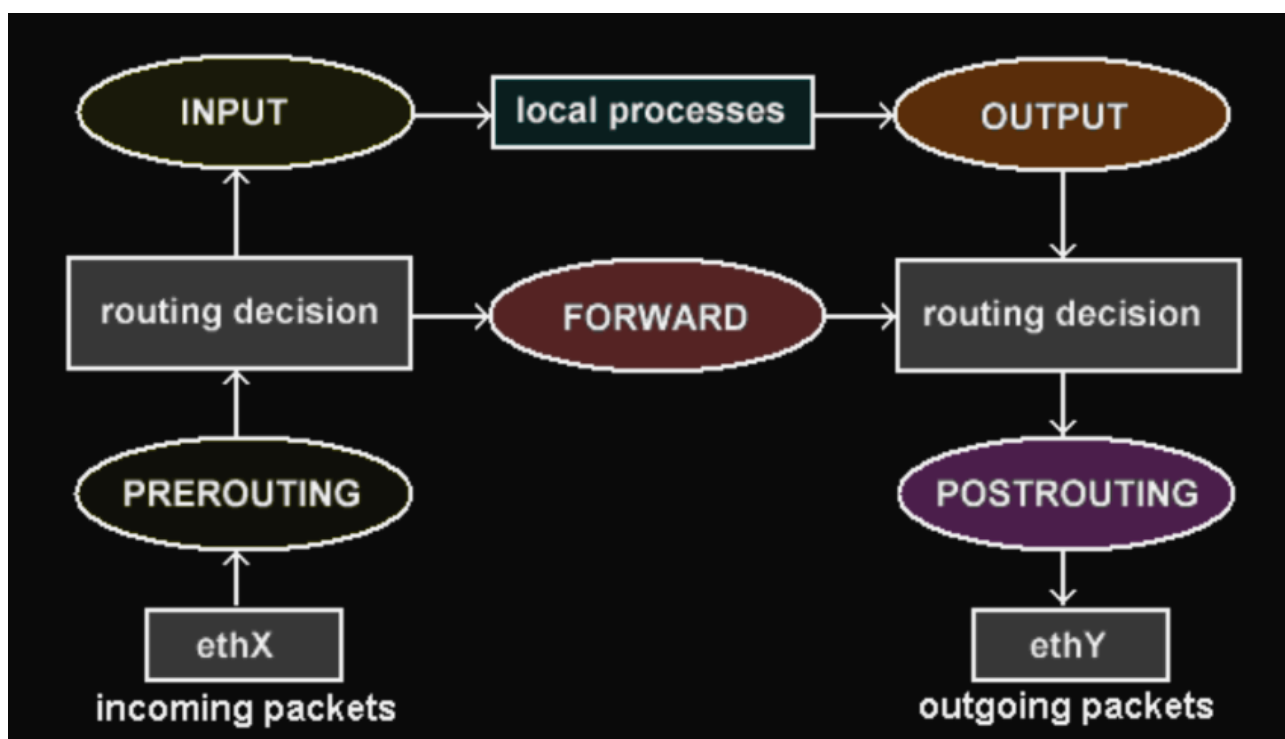
Schéma synthétique : INPUT, OUTPUT et FORWARD.

Lecture textuelle du schéma

INPUT : trafic destiné à la machine locale.

OUTPUT : trafic généré par la machine locale.

FORWARD : trafic traversant la machine.



Cheminement des paquets dans Netfilter : PREROUTING, INPUT, FORWARD, OUTPUT et POSTROUTING.

Lecture textuelle du schéma

Paquets entrants par ethX → PREROUTING → décision de routage.

Destination locale → INPUT → processus local → OUTPUT.

Paquet à transférer → FORWARD → décision de routage → POSTROUTING → ethY.

GNU / Linux – Firewall

- Les tables et les chaînes
 - Paquets entrants, sans forwarding
- Toute la partie en jaune clair et en jaune sur le diagramme

N° Tables Chaînes Explication

1 Couche Physique

2 Arrivée dans l'interface réseau

3 MANGLE PREROUTING Modification des paquets, drapeaux TOS, connection state tracking

4 NAT PREROUTING DNAT (Dynamic NAT - pas de filtrage)

5 Décision de routage (Forward ou Local?)

6 MANGLE INPUT Les paquets ont été routés et sont des INPUT. On peut les modifier à destination d'un processus local.

7 FILTER INPUT Filtrage du trafic entrant, tous les paquets passent par cette chaîne, peu importe d'où ils viennent (ifaces, lo)

8 Le processus en écoute de la couche application réceptionne les informations (httd, sshd, client/server...)

GNU / Linux – Firewall

- Les cibles de filtres
 - Que va-t'on faire des paquets ?
- Toute la partie en Orange et en violet sur le diagramme
 - Conseil :
- Ne pas utiliser les tables et les chaînes pour autre chose que leur positionnement dans la pile TCP/IP
 - Risque d'effet de bord

Target Explication

ACCEPT Le paquet passe à la règle suivante

DROP Le paquet est détruit en silence

REJECT Le paquet est bloqué

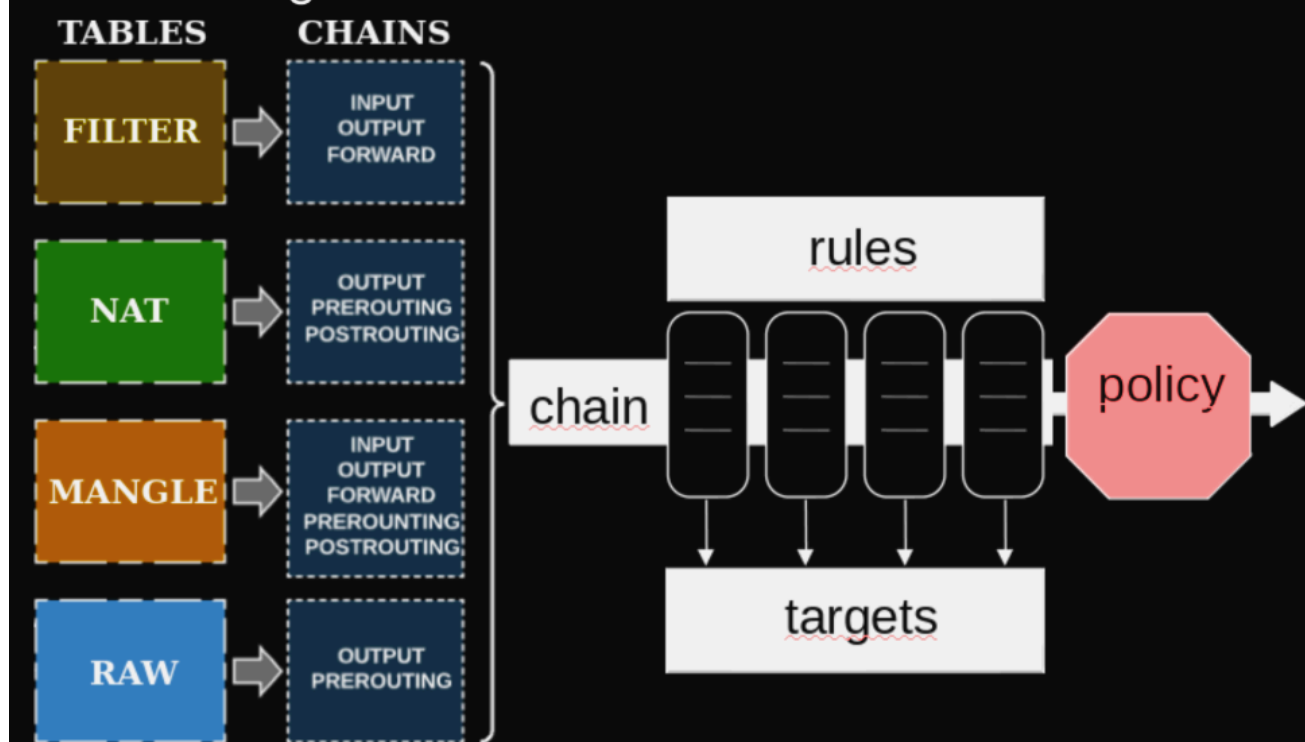
Une réponse ICMP définie par l'utilisateur est envoyée

LOG Une demande d'enregistrement est envoyé à Syslog

GNU / Linux – Firewall

- Schéma général de la structure de données

• Schéma général de la structure de données



Structure générale : tables, chaînes, règles, cibles et politique.

Transcription du schéma

TABLE FILTER → chaînes INPUT, OUTPUT, FORWARD.

TABLE NAT → chaînes OUTPUT, PREROUTING, POSTROUTING.

TABLE MANGLE → chaînes INPUT, OUTPUT, FORWARD, PREROUTING, POSTROUTING.

TABLE RAW → chaînes OUTPUT, PREROUTING.

Une chaîne contient des règles ; chaque règle renvoie vers une cible ; la politique s'applique si aucune règle ne correspond.

GNU / Linux – Firewall

• Les commandes de base

```
# Remise à zéro des iptables. Si on fait juste iptables -F,
# On peut bloquer l'accès à la machine. Toujours prévoir une porte
# de secours avec un crontab qui détruit les règles régulièrement
# Jusqu'à ce que vous validiez votre config !!!
# Le mieux c'est quand même de tester sur VM avant Epik Fail
iptables -P INPUT ACCEPT && iptables -P OUTPUT ACCEPT && iptables
```

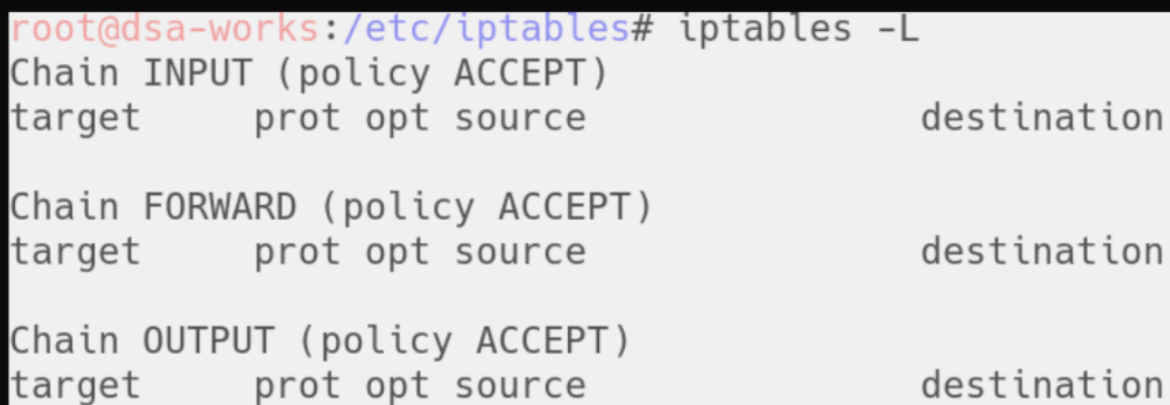
-P FORWARD ACCEPT && iptables -F

```
# Exemple pour supprimer toutes les règles depuis un fichier
# fichier remove-all-rules
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
COMMIT
# Exécution des règles, éventuellement dans la crontab de root
iptables-restore < remove-all-rules
```

GNU / Linux – Firewall

- Les commandes de base

```
# Maintenant qu'on sait à peu près revenir dans un truc qui marche
# Lister les règles de chaque chaîne de la table filter
iptables -L
```



```
root@dsa-works:/etc/iptables# iptables -L
Chain INPUT (policy ACCEPT)
target      prot opt source                destination

Chain FORWARD (policy ACCEPT)
target      prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target      prot opt source                destination
```

Résultat de la commande iptables -L.

Transcription essentielle

Chain INPUT (policy ACCEPT)

Chain FORWARD (policy ACCEPT)

Chain OUTPUT (policy ACCEPT)

Colonnes : target, prot, opt, source, destination.

GNU / Linux – Firewall

- Les commandes de base

```
# Lister les règles de la chaîne INPUT de la table filter
iptables -L INPUT
# C'est un exemple de chez moi, vous ne devriez rien avoir
```

```

root@dsa-works:/etc/iptables# iptables -L INPUT
Chain INPUT (policy DROP)
target     prot opt source                destination
ACCEPT     all  --  anywhere               anywhere
ACCEPT     all  --  anywhere               anywhere    state RELATED,ESTABLISHED
ACCEPT     icmp --  anywhere               anywhere    state NEW
ACCEPT     tcp  --  ip234.ip-51-77-173.eu  anywhere    tcp dpt:ssh
ACCEPT     tcp  --  anywhere               anywhere    tcp dpt:http
ACCEPT     tcp  --  anywhere               anywhere    tcp dpt:https
root@dsa-works:/etc/iptables#

```

Exemple de règles de la chaîne INPUT.

Transcription essentielle

Chain INPUT (policy DROP)

ACCEPT all anywhere anywhere

ACCEPT all anywhere anywhere state RELATED,ESTABLISHED

ACCEPT icmp anywhere anywhere state NEW

ACCEPT tcp ip234.ip-51-77-173.eu anywhere tcp dpt:ssh

ACCEPT tcp anywhere anywhere tcp dpt:http

ACCEPT tcp anywhere anywhere tcp dpt:https

GNU / Linux – Firewall

- Les commandes de base

```

# Lister les règles de chaque chaîne de la table nat
iptables -L -t nat

```

```

root@dsa-works:/etc/iptables# iptables -L -t nat
Chain PREROUTING (policy ACCEPT)
target      prot opt source                destination
DOCKER      all  --  anywhere               anywhere

Chain INPUT (policy ACCEPT)
target      prot opt source                destination

Chain POSTROUTING (policy ACCEPT)
target      prot opt source                destination
MASQUERADE  all  --  172.17.0.0/16          anywhere

Chain OUTPUT (policy ACCEPT)
target      prot opt source                destination
DOCKER      all  --  anywhere              !127.0.0.0/8

Chain DOCKER (2 references)
target      prot opt source                destination
RETURN      all  --  anywhere              anywhere

```

Exemple de la table NAT affichée avec iptables -L -t nat.

Transcription essentielle

Chaînes visibles : PREROUTING, INPUT, POSTROUTING, OUTPUT et DOCKER.

Exemple : MASQUERADE pour la source 172.17.0.0/16 dans POSTROUTING.

GNU / Linux – Firewall

• Les commandes de base

```

# Effacer une règle de la chaîne POSTROUTING de la table nat
# En utilisant l'index de la ligne
iptables -L -t nat --line-numbers
# Effacer une règle de la chaîne POSTROUTING de la table nat
# En utilisant l'index de la ligne
iptables -t nat -D POSTROUTING 1

```

```
# Effacer une règle de la chaîne POSTROUTING de la table nat
# En utilisant l'index de la ligne
iptables -L -t nat --line-numbers
```

```
Chain POSTROUTING (policy ACCEPT)
num  target      prot opt source      destination
1    MASQUERADE    all  --  172.17.0.0/16  anywhere
```

```
# En utilisant l'index de la ligne
iptables -t nat -D POSTROUTING 1
```

```
Chain POSTROUTING (policy ACCEPT)
num  target      prot opt source      destination

Chain OUTPUT (policy ACCEPT)
```

Suppression d'une règle par son numéro de ligne.

Transcription essentielle

```
iptables -L -t nat --line-numbers
```

La chaîne POSTROUTING affiche la règle numéro 1 : MASQUERADE 172.17.0.0/16.

```
iptables -t nat -D POSTROUTING 1
```

La règle numéro 1 est supprimée.

GNU / Linux – Firewall

- Les commandes de base

```
# Effacer toutes les règles de la chaîne OUTPUT de la table nat
iptables -F OUTPUT -t nat
# On affiche la chaîne OUTPUT de la table nat
# Effacer toutes les règles de toutes les chaînes de la table nat
iptables -t nat -F
# On affiche toutes les chaînes de la table nat
# on constate que la dernière chaîne n'existe plus
```

• Les commandes de base

```
# Effacer toutes les règles de la chaîne OUTPUT de la table nat
iptables -F OUTPUT -t nat
# On affiche la chaîne OUTPUT de la table nat
```

```
Chain OUTPUT (policy ACCEPT)
num target      prot opt source          destination
```

```
# Effacer toutes les règles de toutes les chaînes de la table nat
iptables -t nat -F
# On affiche toutes les chaînes de la table nat
# on constate que la dernière chaîne n'existe plus
```

```
Chain DOCKER (0 references)
num target      prot opt source          destination
root@dsa-works:/etc/iptables#
```

Vidage de la chaîne OUTPUT puis de toutes les chaînes de la table NAT.

Transcription essentielle

iptables -F OUTPUT -t nat : efface toutes les règles de OUTPUT dans la table NAT.

iptables -t nat -F : efface toutes les règles de toutes les chaînes de la table NAT.

```
root@dsa-works:/etc/iptables# iptables -L -t nat --line-numbers
Chain PREROUTING (policy ACCEPT)
num target      prot opt source          destination

Chain INPUT (policy ACCEPT)
num target      prot opt source          destination

Chain POSTROUTING (policy ACCEPT)
num target      prot opt source          destination

Chain OUTPUT (policy ACCEPT)
num target      prot opt source          destination
root@dsa-works:/etc/iptables#
```

Table NAT après vidage : chaînes par défaut affichées sans règles.

GNU / Linux – Firewall

• Les commandes de base

```
# Effacer la chaîne DOCKER de la table nat
iptables -X DOCKER -t nat
# On affiche la table nat
```

```
# Effacer toutes les chaines qui ne sont pas par défaut
iptables -X
```

GNU / Linux – Firewall

- Création de règles simples pour un serveur
 - On veut créer les filtres suivants
- On veut autoriser tout le trafic en sortie
- On ne veut pas autoriser le forwarding
- Je veux refuser tout le trafic entrant sauf :
 - Le port 22 en TCP pour l'IP : 15.77.173.234
- Mes bureaux sont en IP fixe
 - Le port 22 en TCP pour l'IP : 80.15.77.173
- J'ai un VPN avec une IP fixe quand je ne suis pas dans mes locaux
 - Port 80 et 443
- Autoriser pour tout le monde
 - Autoriser ICMP pour le diagnostic/monitoring
- Vous êtes libres d'adapter les IP

GNU / Linux – Firewall

- Création de règles simples pour un serveur

```
# étape 1
# définir les policies, appelés aussi des fallbacks, les règles
# qui seront exécutées si aucune des règles ne correspond
# Principe de White List, je refuse tout sauf...
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT

# étape 2
# On autorise le trafic entrant et connexions sur la boucle locale
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

# Certains services ont besoin de la loopback pour se connecter
# Par exemple MySQL (peut varier d'une distro à l'autre)
```

GNU / Linux – Firewall

- Création de règles simples pour un serveur

```
# étape 3
# Autorisation de ICMP (ping, echo et cie...)
iptables -A INPUT -p icmp -m state --state NEW -j ACCEPT

# étape 4
# Filtrage IP pour SSH et le port 8006 Proxmox
# Ne pas oublier iptables devant -A
-A INPUT -p tcp -m tcp -s 15.77.173.234 --dport 22 -j ACCEPT
-A INPUT -p tcp -m tcp -s 80.15.77.173 --dport 22 -j ACCEPT

# Notez qu'on peut utiliser des listes
```

```
-s 15.77.173.234,80.15.77.173 --dport 22,8006
```

```
# Ainsi que des ranges + notations CIDR
```

```
-s 80.15.77.0/24 --dport 22,8006-8010
```

GNU / Linux – Firewall

- Création de règles simples pour un serveur

```
# étape 5
# J'autorise tout le trafic en TCP sur 80 et 443
iptables -A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp -m tcp --dport 443 -j ACCEPT

# étape 6
# La configuration de iptables n'est pas persistante
# Je sauve ma configuration
iptables-save > /etc/iptables/iptables
# Elle doit être en place avant le chargement des interfaces
cd /etc/network/if-pre-up.d
echo "#!/bin/bash" > iptables
echo "iptables-restore /etc/iptables/iptables" >> iptables
chmod +x iptables
# Ceci peut aussi être fait avec init.d
# Pour restaurer en ligne de commande
iptables-restore < /etc/iptables/iptables
```

GNU / Linux – Firewall

- Fichier /etc/iptables/iptables

```
# Generated by iptables-save v1.4.14 on Wed Mar 5 23:27:13 2014
*filter

:INPUT DROP [13:888]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [35:5084]
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -o lo -j ACCEPT
# ICMP
-A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -m state --state NEW -j ACCEPT
-A INPUT -p tcp -m tcp -s 15.77.173.234,80.15.77.173 --dport 22 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80,443 -j ACCEPT

#####
COMMIT
# Completed on Wed Mar 5 23:27:13 2014
```

GNU / Linux – Détection des intrusions

- fail2ban
- Qu'est ce que c'est ?
 - Un outil qui analyse les logs
 - Maintient des statistiques
- Et prend des décision
 - S'appuie sur Iptables
- Inconvénient majeur
 - Assez lourd en processeur
 - Travail en autonomie :
- Installer et configurer Fail2Ban
 - Agir sur SSH : 3 tentatives ratés = 2h de ban

- Agir sur la HTTP Auth : 3 tentatives ratées = 2h de ban
- Trouver comment supprimer un BAN à la main

GNU / Linux – Blacklisting

- On a besoin de détecter les IP
 - Depuis les logs
 - De manière automatique
 - Depuis des emplacements géographiques
- Commande whois
- solution payante : <https://lite.ip2location.com/>
 - Depuis des fournisseurs
- DNSBL (Spam) / IP Blacklist
 - On peut utiliser iptables
- Mais il existe une commande magique
- Voir page suivante

GNU / Linux – Blacklisting

- Procédure de null route, cas pratique

étape 1, on repère une IP qui nous ennuie dans des logs

194.168.32.45

On commence à creuser les logs et on se rend compte qu'on a une
floppée d'adresse avec le même préfixe, je réalise un whois
whois 194.163.32.45

```
root@dsa-works:/etc/snort/rules# whois 194.163.32.45
% This is the RIPE Database query service.
% The objects are in RPSL format.
% Blablalabla
% Abuse contact for '194.163.32.0 - 194.163.39.255' is 'abuse@hostinger.com'
inetnum:          194.163.32.0 - 194.163.39.255
netname:          HOSTINGER-HOSTING
country:          SG
```

Exemple de recherche WHOIS sur une adresse IP.

Transcription essentielle

whois 194.163.32.45

inetnum : 194.163.32.0 - 194.163.39.255

netname : HOSTINGER-HOSTING

country : SG

GNU / Linux – Blacklisting

- Procédure de null route, cas pratique

```
# étape 2, je dois transformer le range en notation CIDR
ipcalc 194.163.32.0-194.163.39.255

# étape 3, Nullroute ou blackhole routing L'attaquant pense qu'il
# n'y a rien derrière l'IP : No route to host
ip route add blackhole 194.163.32.0/21

# étape 4, suppression de la null route
ip route del 194.163.32.0/21
```

• Procédure de null route, cas pratique

```
# étape 2, je dois transformer le range en notation CIDR
ipcalc 194.163.32.0-194.163.39.255
```

```
root@dsa-works:/etc/snort/rules# ipcalc 194.163.32.0-194.163.39.255
deaggregate 194.163.32.0 - 194.163.39.255
194.163.32.0/21
```

```
# étape 3, Nullroute ou blackhole routing L'attaquant pense qu'il
# n'y a rien derrière l'IP : No route to host
ip route add blackhole 194.163.32.0/21
```

```
root@dsa-works:/etc/snort/rules# ip route
default via 192.168.1.1 dev enp2s0 proto static metric 100
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
192.168.1.0/24 dev enp2s0 proto kernel scope link src 192.168.1.23 metric 100
blackhole 194.163.32.0/21
```

```
# étape 4, suppression de la null route
ip route del 194.163.32.0/21
```

Procédure de null route : conversion CIDR, ajout et suppression d'une route blackhole.

Transcription essentielle

```
ipcalc 194.163.32.0-194.163.39.255 → 194.163.32.0/21

ip route add blackhole 194.163.32.0/21

ip route del 194.163.32.0/21
```