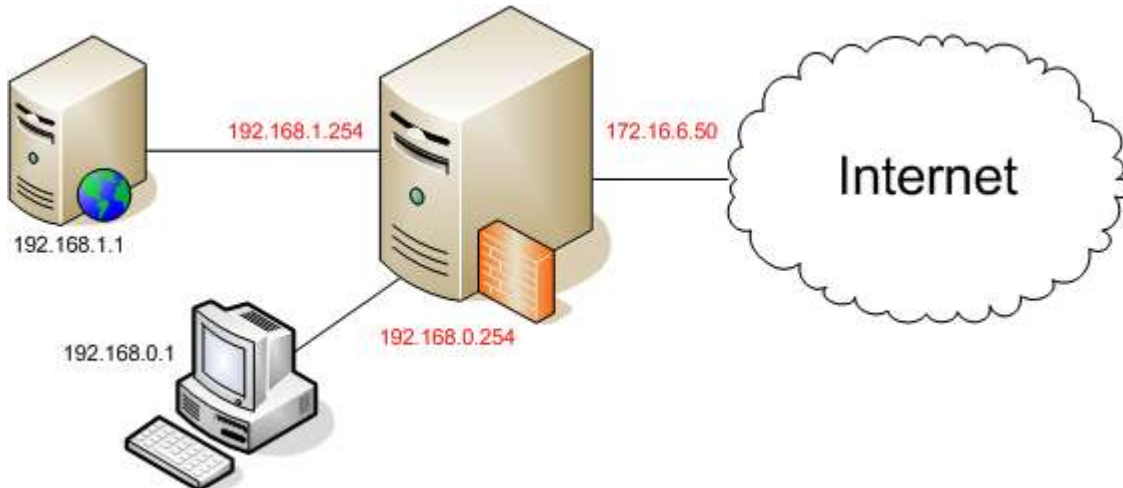


Compte- rendu PTI #03

Cette troisième PTI couvre le domaine du paramétrage réseau et de la sécurité du réseau par la mise en place d'un système de filtrage de paquets via Netfilter (iptables) sous GNU/Linux.

Principe

Le schéma suivi est composé de 3 parties :



Légende :

- 172.16.6.50 Interface du routeur vers le réseau externe (Internet)
- 192.168.0.254 Interface du routeur vers le réseau interne
- 192.168.1.254 Interface du routeur vers la zone démilitarisée (DMZ)
- 192.168.1.1 Serveur web (dans la DMZ)
- 192.168.0.1 Machine cliente (dans le réseau interne)

Compétences

Cette PTI couvre les compétences suivantes :

- C21 - Installer et configurer un microordinateur
- C22 - Installer et configurer un réseau
- C23 - Installer et configurer un dispositif de sécurité
- C31 - Assurer les fonctions de base de l'administration d'un réseau
- C32 - Assurer les fonctions de l'exploitation
- C34 - Surveiller et optimiser le trafic réseau

Outils utilisés

Au cours de la réalisation de la PTI, les outils suivants ont été utilisés :

Matériel :

- Micro- ordinateurs (un client, un serveur et un routeur logiciel)
- Câbles réseaux à paires torsadées croisées

Logiciels :

- Système d'exploitation GNU/Linux Slackware version 10
- Logiciel de filtrage Netfilter 1.2.11
- Serveur web Apache 1.3.31

Opérations préliminaires

La machine qui fait office de routeur et de pare-feu nécessite quelques réglages avant de pouvoir fonctionner comme souhaité. Tout d'abord il faut y activer le routage IP, comme ceci :

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

(Note : toutes les commandes doivent être exécutées par le root)

Ensuite, il convient de configurer et d'activer les interfaces réseau sur le routeur avec les paramètres réseau utilisés dans cette PTI, et enfin de paramétrer la route par défaut vers Internet et les DNS :

```
# ifconfig eth0 172.16.6.50
# ifconfig eth1 192.168.1.254
# ifconfig eth2 192.168.0.254
# route add default gw 172.16.1.254
# echo "nameserver 212.180.1.79" > /etc/resolv.conf
# echo "nameserver 212.180.0.137" >> /etc/resolv.conf
```

A présent que le routeur est opérationnel, nous pouvons nous intéresser au paramétrage du pare-feu.

Fonctionnement et configuration de Netfilter

Netfilter - et non pas *iptables* comme la majorité des gens le nomme- est un logiciel de filtrage de paquets (un pare-feu, ou *firewall*) intégré à Linux depuis le noyau 2.4, qui remplace les *ipchains* du noyau 2.2. A l'instar de la majorité des logiciels composants ce système d'exploitation il est libre et gratuit. Il est utilisé compilé "en dur" dans le noyau ou bien en sous forme de modules chargeables à la volée si besoin.

Nous avons opté pour la version modulaire ; les modules sont chargeables avec la commande `modprobe`. Voici ceux dont nous avons besoin dans un premier temps :

```
# modprobe iptables
# modprobe iptable_filter
# modprobe iptable_nat
# modprobe ipt_MASQUERADE
```

Pour vérifier que les modules soient effectivement chargés, on peut utiliser la commande `lsmod`.

Afin de faciliter l'affichage des règles *iptables* que nous allons écrire, nous définissons un alias ("`clip`") dans notre *shell* (l'interpréteur de commandes) :

```
alias clip="clear && iptables -L --line-numbers"
```

Nous pouvons maintenant commencer à définir des règles de filtrage. Tout d'abord, nous allons redéfinir les stratégies par défaut du filtre, puis nous activons la NAT (Network Address Translation, *traduction d'adresses réseau*) pour la DMZ et le réseau interne (le "/23" équivaut au banc d'adresses IP de 192.168.0.1 à 192.168.1.254) :

```
# iptables -P INPUT DROP
# iptables -P OUTPUT DROP
# iptables -P FORWARD DROP

# iptables -A POSTROUTING -t nat -s 192.168.0.0/23 -o eth0 -j MASQUERADE
```

Ensuite, nous autorisons les requêtes DNS en provenance du réseau interne à passer au travers du pare-feu, ainsi que leurs réponses :

```
# iptables -A FORWARD -p udp -s 212.180.1.79 --sport 53 -i eth0 -j ACCEPT
# iptables -A FORWARD -p udp -d 212.180.1.79 --dport 53 -o eth0 -j ACCEPT
# iptables -A FORWARD -p udp -s 212.180.0.137 --sport 53 -i eth0 -j ACCEPT
# iptables -A FORWARD -p udp -d 212.180.0.137 --dport 53 -o eth0 -j ACCEPT
```

Il nous faut aussi laisser passer le trafic HTTP(S) et FTP entre le réseau interne/la DMZ et Internet :

```
# iptables -A FORWARD -p tcp -m multiport -s 192.168.0.0/23 --dport http,https,ftp,ftp-data
-o eth0 -j ACCEPT
# iptables -A FORWARD -p tcp -m state --state RELATED,ESTABLISHED -m multiport -d
192.168.0.0/23 --sport http,https,ftp,ftp-data -i eth0 -j ACCEPT
```

Autoriser l'accès au serveur web de la DMZ :

```
# iptables -A FORWARD -p tcp -d 192.168.1.1/32 --dport http -j ACCEPT
# iptables -A FORWARD -p tcp -m state --state RELATED,ESTABLISHED -s 192.168.1.1/32 --sport http -j ACCEPT
```

On laisse passer les paquets ICMP seulement si un ECHO_REQUEST a été envoyé au préalable ou si ils viennent de l'intérieur ou de la DMZ :

```
# iptables -A FORWARD -p icmp -s 192.168.0.0/23 -o eth0 -j ACCEPT
# iptables -A FORWARD -p icmp -m state --state RELATED,ESTABLISHED -d 192.168.0.0/23 -i eth0 -j ACCEPT
```

Pour la maintenance sur le routeur/pare-feu, on autorise les connexions via SSH sur la machine :

```
# iptables -A INPUT -p tcp --dport ssh -j ACCEPT
# iptables -A OUTPUT -p tcp --sport ssh -j ACCEPT
```

Et enfin, on redirige toute tentative de connexion HTTP sur le pare-feu vers le serveur web de la DMZ :

```
# iptables -t nat -A PREROUTING -p tcp -i eth0 -d 172.16.6.50 --dport 80 -j DNAT --to-destination 192.168.1.1:80
```

Optionnel : logger les paquets refusés pour analyse

Il est possible d'archiver ("*logger*") les paquets rejetés par Netfilter dans des fichiers texte ("*logs*") en redirigeant ces paquets non plus vers la cible DROP ou ACCEPT mais vers la chaîne LOG (requiert le module noyau *ipt_LOG*). De cette manière il sera possible de parcourir ultérieurement les logs pour y déceler des tentatives d'intrusion ou tout autre comportement anormal du trafic réseau.

Habituellement si l'on redirige les paquets refusés vers la cible LOG, les annonces des rejets apparaîtront directement dans les logs du noyau (dans `/var/log/message`). Ceci peut présenter un inconvénient dans la mesure où une forte activité réseau (et donc une forte activité du pare-feu) va surcharger les logs du système et empêcher les administrateurs du système de déceler ses éventuels problèmes.

Il nous faut donc utiliser une autre cible vers laquelle rediriger les paquets non-désirés avant destruction : ULOG. Cette chaîne, liée au module du noyau *ipt_ULOG*, fonctionne selon un principe très similaire à la cible LOG à la différence que couplé au *daemon* *ulogd* elle ne logge que les messages de Netfilter, de manière beaucoup plus paramétrable et surtout dans des fichiers de log dédiés.

La configuration par défaut de *ulogd* étant très convenable, nous ne la modifierons pas dans notre cas. Pour pouvoir logger les paquets grâce à ce logiciel, il suffit de lancer le service au démarrage en même temps que le script du *firewall* (cf. scripts de services en annexe). Les informations quant aux paquets rejetés sont consultables dans le fichier `/var/log/ulogd.syslogemu`.

Par exemple, pour logger tous les paquets ICMP rejetés en entrée du pare-feu :

```
# iptables -A INPUT -p icmp -j ULOG --ulog-prefix="[DROP]"
```

Annexe 1 : rc.firewall, service de pare-feu

Ce script est exécuté au démarrage de l'OS mais peut être arrêté à tout moment par l'administrateur.

```
#!/bin/sh
export IPT="/usr/sbin/iptables"
export INT_EXT="eth0"
export INT_DMZ="eth1"
export INT_INT="eth2"

fw_start() {
  #Préparer le système à router et filtrer
  echo 1 > /proc/sys/net/ipv4/ip_forward
  modprobe ip_tables
  modprobe iptable_filter
  modprobe iptable_nat
```

```

#Purger le filtre par precaution
$IPT -t filter -F
$IPT -t nat -F

#Appliquer les strategies par default : tout refuser
$IPT -P INPUT DROP
$IPT -P OUTPUT DROP
$IPT -P FORWARD DROP

#Activer la translation d'adresses (NAT) pour l'exterieur
$IPT -A POSTROUTING -t nat -s 192.168.0.0/23 -o $INT_EXT -j MASQUERADE

##### Creation des regles de filtrage (par ordre de priorite) #####

#1) Activer le FORWARDING pour la resolution de nom DNS
$IPT -A FORWARD -p udp -s 212.180.1.79 --sport 53 -i $INT_EXT -j ACCEPT
$IPT -A FORWARD -p udp -d 212.180.1.79 --dport 53 -o $INT_EXT -j ACCEPT
$IPT -A FORWARD -p udp -s 212.180.0.137 --sport 53 -i $INT_EXT -j ACCEPT
$IPT -A FORWARD -p udp -d 212.180.0.137 --dport 53 -o $INT_EXT -j ACCEPT

#2) Autoriser les echanges HTTP(S), FTP
$IPT -A FORWARD -p tcp -m state --state RELATED,ESTABLISHED -m multiport -d 192.168.0.0/23 --sport http,https,ftp,ftp-data -i $INT_EXT -j ACCEPT
$IPT -A FORWARD -p tcp -m multiport -s 192.168.0.0/23 --dport http,https,ftp,ftp-data -o $INT_EXT -j ACCEPT
#Reponses HTTP venant du serveur WEB (DMZ)
$IPT -A FORWARD -p tcp -d 192.168.1.1/32 --dport http -j ACCEPT
$IPT -A FORWARD -p tcp -m state --state RELATED,ESTABLISHED -s 192.168.1.1/32 --sport http -j ACCEPT

#3) Autoriser certains echanges ICMP (reponses < exterieur)
$IPT -A FORWARD -p icmp -m state --state RELATED,ESTABLISHED -d 192.168.0.0/23 -i $INT_EXT -j ACCEPT
$IPT -A FORWARD -p icmp -s 192.168.0.0/23 -o $INT_EXT -j ACCEPT

#4) Autoriser les connexions SSH sur le serveur (routeur)
$IPT -A INPUT -p tcp --dport ssh -j ACCEPT
$IPT -A OUTPUT -p tcp --sport ssh -j ACCEPT

#5) Rediriger les entrees sur port 80 sur le serveur WEB (DMZ)
$IPT -t nat -A PREROUTING -p tcp -i $INT_EXT -d 172.16.6.50 --dport 80 -j DNAT --to-destination 192.168.1.1:80
}

fw_stop() {
#Purge le filtre
$IPT -t filter -F
$IPT -t nat -F

#Decharge les modules noyau
modprobe -r ipt_ULOG
modprobe -r ipt_state
modprobe -r ipt_MASQUERADE
modprobe -r iptable_filter
modprobe -r iptable_nat
modprobe -r ip_conntrack
modprobe -r ipt_multiport
modprobe -r ip_tables
}

case "$1" in
'start')
fw_start
;;
'stop')
fw_stop
;;
*)
echo "usage $0 start|stop"
esac

```

Annexe 2 : rc.ulogd, service de logging de paquets rejetés

```
#!/bin/sh
#
#####
# Script de demarrage de ulogd, basé sur le script de demarrage de Privoxy
#
# Créé le: 2003/04/04                Dernière modification le : 2003/04/04
#####

# Source function library.
#. /etc/rc.d/init.d/functions

ULOGD_PRG="ulogd"
ULOGD_BIN="/sbin/$ULOGD_PRG"
ULOGD_CONF="/etc/ulogd.conf"
ULOGD_PID="/var/run/$ULOGD_PRG.pid"
ULOGD_LOCK="/var/lock/subsys/$ULOGD_PRG"
ULOGD="$ULOGD_BIN"

# some checks for us
! [ -x $ULOGD_BIN ] && echo $"Can't find $ULOGD_BIN, exit." && exit 0
! [ -f $ULOGD_CONF ] && echo $"Can't find $ULOGD_CONF, exit." && exit 0

# See how we were called.

start () {
    # start daemon
    echo -n $"Starting $ULOGD_PRG: "
    if [ -f $ULOGD_PID ]; then
        killproc $ULOGD_PRG && rm -f $ULOGD_LOCK $ULOGD_PID
        RETVAL=$?
        [ $RETVAL != 0 ] && return $RETVAL
    fi
    $ULOGD > /dev/null 2>&1 &
    RETVAL=$?
    echo
    [ $RETVAL = 0 ] && touch $ULOGD_LOCK
    return $RETVAL
}

stop () {
    # stop daemon
    echo -n $"Stopping $ULOGD_PRG: "
    pkill $ULOGD_PRG && rm -f $ULOGD_LOCK
    RETVAL=$?
    echo
    return $RETVAL
}

case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart)
        stop
        start
        RETVAL=$?
        ;;
    condrestart)
        # restart only if already running
        if [ -f $ULOGD_PID ]; then
            stop
            start
            RETVAL=$?
        fi
        ;;
    status)
        status $ULOGD_PRG
        RETVAL=$?

```

```
;;
top)
  if [ -f $ULOGD_PID ]; then
    a=""
    for i in `pidof $ULOGD_PRG` ; do
      a="$a -p $i"
    done
    top $a
  fi
;;
*)
  echo $"Usage: $ULOGD_PRG {start|stop|reload|restart|condrestart|status|top}"
  exit 1
esac
exit $RETVAL
```