Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE
PREMIER MINISTRE

S . G . D . S . N
Agence nationale de la sécurité des systèmes d'information
CERT-FR

Paris, le 31 juillet 2017
N° CERTFR-2017-ACT-030

Affaire suivie par :
CERT-FR

BULLETIN D'ACTUALITÉ

Objet : Bulletin d'actualité CERTFR-2017-ACT-030

1 - Prise en main de l'outil nftables

Introduction

Nftables est un outil d'interface utilisateur pour le pare-feu Linux Netfilter permettant la manipulation de règles de la même manière qu'iptables. La plupart des concepts étant identiques ou très proches, cela permet aux administrateurs de s'adapter rapidement. Nftables est supporté à partir de Linux 3.13.

Nous présenterons ici, les concepts de nftables et ses différences avec iptables, puis nous finirons par quelques conseils d'utilisation.

Présentation

Nftables a été créé dans le but de :

- proposer dans un seul et même outil de gestion des règles de filtrage ipv4/v6, arp, bridges, etc ;
- changer la façon dont les règles de pare-feu sont interprétées et permettre aux administrateurs d'optimiser l'écriture de règles (ajout de variables, de structures de données complexes, etc.) ;
- regrouper au sein d'un même binaire l'ensemble des fonctionnalités qui était éparpillées dans plusieurs binaires du paquet iptables ;
- donner un accès par API (libnl) aux utilisateurs sans nécessiter un accès en mode noyau.

Nftables est composé d'une interface utilisateur (nft), de modules noyau spécifiques, ainsi que d'une bibliothèque (libnl) permettant la communication avec ledit module.

Voici un schéma explicatif des briques logicielles et de leurs interactions.

```
Espace Mémoire Utilisateur      |   Espace Mémoire Noyau
Interface     |    Bibliothèque |   Modules Noyau
nft     < -------- > libnl < ------ >    netfilter
         netlink API
```

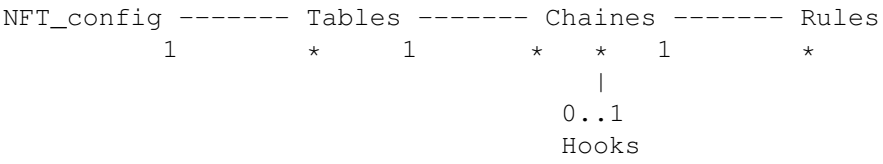
Concepts

La terminologie de nftables est identique à celle d'iptables. Les règles sont toujours listées de manière ordonnée dans des chaînes (input, output, etc.), elles-mêmes regroupées dans des tables. La première chose à retenir de nftables est la totale liberté laissée à l'utilisateur dans le nommage des tables et des chaînes. En effet, iptables impose l'existence des tables filter, nat, mangle, raw et security sans permettre l'ajout ou le renommage.

Dans le cas de nftables, aucune table ou chaîne n'existe de base et l'administrateur est libre de ne pas créer les chaînes dont il n'a pas l'utilité (ce qui permet d'éviter des traitements inutiles). Une table est identifiée par son nom et son type ("ip" pour ipv4, "ip6" pour ipv6, "inet" pour ipv4 et v6, "arp" ou "netdev" pour les protocoles homonymes). Afin de diriger le trafic dans des chaînes (pour en permettre l'analyse et en déduire des actions), il

est nécessaire de l'associer à un point d'ancrage (hooks) offerts par netfilter. Les points d'ancrage de base pour ip, ip6 et inet sont les suivants : input, output, forward, prerouting, postrouting.

Voici un schéma simplifié des concepts de nftables et des relations entre eux :



Si une chaîne n'est associée à aucun point d'ancrage, par défaut, le noyau n'y fera passer aucun trafic. Ces chaînes sont appelées "chaînes régulières" et permettent de regrouper un ensemble de règles qui pourront être utilisées par l'instruction "jump" par la suite.

Les chaînes dites "de base", quant à elles, sont attachées à un point d'ancrage et possèdent un type choisi parmi :

- filter : contiendra des règles pour accepter ou rejeter des paquets ;
- nat : contiendra des règles permettant de modifier les attributs ip (adresses source et destination par exemple) ;
- route : contiendra des règles de routage par politique (policy based routing).

Environnement de base

Dans iptables, la table filter est composée à minima des chaînes input, forward et output. Nftables permet de créer ces chaines en utilisant les commandes ci-dessous.

```
$ nft add table table_filtrante
$ nft add chain table_filtrante chaine_entree { type filter hook input priority 0\;
  policy drop \; }
$ nft add chain table_filtrante chaine_sortie { type filter hook output priority 0\;
  policy drop \; }
```

Explications :

- "type filter" permet de configurer la chaîne qui contiendra des règles de filtrage ;
- "hook input" permet de choisir le point d'ancrage netfilter auquel nous nous accrochons ;
- "priority 0" permet un ordonnancement si plusieurs chaînes venaient à être attachées au point d'ancrage (0 étant la plus forte priorité) ;

La "policy accept" est sélectionnée par défaut. Pour modifier ce comportement la commande suivante peut être utilisée.

```
$ nft chain table_filtrante chaine_entree { policy accept \; }
```

Sans précision, le type de notre table est "ip" (c'est-à-dire ipv4). Nous aurions pu préciser un type lors de sa création comme suit :

```
$ nft add table table_filtrante inet
```

Généralement une configuration nftables se fait dans un fichier de configuration que nous chargerons au démarrage. La commande "nft list ruleset" nous permet de voir l'ensemble des règles, c'est-à-dire toute la configuration nftables active.

```
$ nft list ruleset
table ip table_filtrante {
        chain chaine_entree {
                type filter hook input priority 0; policy drop;
        }
        chain chaine_sortie {
                type filter hook output priority 0; policy drop;
        }
}
```

Nftables permet l'import et l'export de configuration très facilement comme suit :

```
$ nft list ruleset > /etc/nftables.conf
$ nft -f /etc/nftables.conf
```

Ecriture des règles

Les règles de nftables sont évaluées de gauche à droite comme un ensemble de conditions puis d'actions.

```
$ nft add rule table_filtrante chaine_entree tcp dport ssh ip saddr 192.168.0.15
  accept
```

On ajoute la règle à notre chaîne d'entrée avec deux expressions conditionnelles :

1. tcp dport ssh
2. ip saddr 192.168.0.15

Les expressions sont construites relativement au modèle OSI. Dans l'exemple ci-dessus, le port de destination est dans l'en-tête tcp, l'adresse ip source est dans l'en-tête ip. Nftables interrompt l'interprétation de la règle à la 1er condition non validée. Dans notre exemple, l'évaluation de la règle dans le cas d'un paquet udp s'interrompt au 1er test (le paquet n'est pas du tcp).

Dans le cas de paquets issus d'une session ssh provenant d'une ip autre que 192.168.0.15, l'évaluation de la règle s'arrêtera après la 2ème expression. Une fois les conditions validées, les actions sont appliquées dans l'ordre (toujours de gauche à droite). Ici une seule action est utilisée (accept) mais il est possible d'ajouter d'autres actions comme compter les paquets ou journaliser l'utilisation des règles.

Voici la même règle avec un compteur de paquets.

```
$ nft add rule table_filtrante chaine_entree tcp dport ssh ip saddr 192.168.0.15
  counter accept
```

Exemples de configuration

Pour une liste exhaustive des possibilités qu'offre l'écriture de règles vous pouvez consulter les sections "Supported selectors for packet matching" et "Possibles actions on packets" dans la documentation de nftables (lien en fin de bulletin).

États de connexions

L'utilisation de "ct state" (connexion tracker) permet de définir un état de connexion netfilter désiré comme suit.

```
table ip table_filtrante {
        chain chaine_entree {
                type filter hook input priority 0; policy drop;
                ct state { related, established } accept
                tcp dport { http, https} ct state new accept
        }
        chain chaine_sortie {
                type filter hook output priority 0; policy drop;
                ct state { related, established } accept
        }
}
```

Dans cet exemple, les connexions déjà établies sont autorisées en entrée et en sortie. Les connexions nouvelles ne sont acceptées que pour http et https. L'autorisation des connexions dans la chaine de sortie est nécessaire, car nftables ne laisse pas passer implicitement des flux acceptés en 'new'.

Sous-chaînes

L'instruction "jump" permet le saut vers d'autres chaînes de la même table. Une fois la fin de la chaîne atteinte, l'évaluation des règles continue dans la chaîne parente. L'instruction "goto" est identique, à l'exception qu'elle ne retourne pas dans la chaine parente. Les sauts sont principalement utilisés pour mutualiser des règles et/ou structurer une configuration.

En voici un exemple :

```
table ip table_filtrante {
        chain chaine_entree {
```

```
                type filter hook input priority 0; policy drop;
        jump chaine_base
                tcp dport { http, https } ip saddr 192.168.20.0/24 accept
        jump chaine_entree_admin
        }
        chain chaine_sortie {
                type filter hook output priority 0; policy drop;
        jump chaine_base
        tcp dport ntp ip daddr ntp.reseau.local accept
        tcp dport http ip daddr miroir.reseau.local accept
        }
    chain chaine_base {
        iifname lo accept
        ct state { related, established } accept
        ip protocol icmp icmp type { echo-request, echo-reply, time-exceeded, \
                            parameter-problem, destination-unreachable } accept
    }
    chain chaine_entree_admin {
        tcp dport { ssh, microsoft-ds } ip saddr 192.168.10.0/24 accept
        tcp dport 3000 ip saddr puppet.reseau.local accept
    }
}
```

Ici nous avons mutualisé certaines règles de bases dans une chaîne du même nom. Nous avons également déplacé certaines règles d'administration dans une chaîne spécifique aux flux d'administration afin de créer une séparation forte avec les flux métiers.

Journalisation

Nftables est compatible avec les modules noyau xt_LOG et ipt_LOG déjà utilisables avec iptables. Dans cet exemple, nous allons utiliser le nouveau module de nftable nfnetlink_log couplé à l'agent ulogd2 côté espace utilisateur. Pour cela :

- installez le packet ulogd2
- assurez-vous que le module noyau nfnetlink_log est bien chargé

Sa configuration se fait via les fichiers dans /proc/sys/net/netfilter/nf_log/.

Chaque fichier portant un numéro correspond à un protocole différent. Ipv4 correspond au numéro 2 et ipv6 au numéro 10.

Afin d'utiliser le module nfnetlink_log pour journaliser de l'ipv4 voici comment le dire à netfilter :

```
echo -n 'nfnetlink_log' > /proc/sys/net/netfilter/nf_log/2
```

Vous pouvez vérifier la configuration à tout moment comme suit :

```
$ cat /proc/net/netfilter/nf_log
 0 NONE (nfnetlink_log)
 1 NONE (nfnetlink_log)
 2 nfnetlink_log (nfnetlink_log)
 3 NONE (nfnetlink_log)
 4 NONE (nfnetlink_log)
 5 NONE (nfnetlink_log)
 6 NONE (nfnetlink_log)
 7 NONE (nfnetlink_log)
 8 NONE (nfnetlink_log)
 9 NONE (nfnetlink_log)
10 nfnetlink_log (nfnetlink_log)
11 NONE (nfnetlink_log)
12 NONE (nfnetlink_log)
```

ATTENTION : Cette action est volatile et disparaîtra au prochain démarrage. Une fois le module noyau configuré et ulogd2 lancé, nous pouvons commencer à journaliser des événements avec nftables.

Pour journaliser tout le trafic entrant (à placer en haut de la chaîne) :

```
$ nft add rule table_filtrante chaine_entree log
```

Pour journaliser le trafic provenant d'une ip (les lignes de journaux commenceront par le préfixe donné)

```
$ nft add rule table_filtrante chaine_entree ip addr 192.168.1.2 log prefix
  "utilisateur02"
```

Pour journaliser seulement les ouvertures de sessions (ou tentatives infructueuses)

```
$ nft add rule table_filtrante chaine_entree ct state new log prefix "connexion"
```

Structures de données

Dans une table vous pouvez définir des structures de données avancées comme des listes :

```
table ip table_filtrante {
    set services_prod {
            type inet_service
            elements = {ssh, http, https}
    }
        chain chaine_entree {
                type filter hook input priority 0; policy drop;
                tcp dport @services_prod accept
        }
}
```

Un "set" peut être décrit à la volée lors de l'écriture des règles comme nous l'avons fait quelques exemples plus haut :

- "tcp dport http, https " ;
- "ct state related, established ".

Les dictionnaires vous permettent de définir plusieurs actions possibles en fonction d'un critère dans une règle :

```
table ip table_filtrante {
        chain chaine_sortie {
        oifname vmap { "eth0"~: jump chaine_dmz_entree, eth1~: \
        jump chaine_internet_sortie }
        }
    chain chaine_dmz_entree {
    }
    chain chaine_internet_sortie {
    }
}
```

Utiliser un dictionnaire dans cet exemple équivaut à écrire les deux règles comme suit :

1. oifname "eth0" jump chaine_dmz_entree
2. oifname "eth1" jump chaine_internet_sortie

Conseils d'utilisation

Désactiver iptables

Si vous comptez utiliser nftables seulement, vous pouvez désactivez iptables comme suit.

Cette manipulation n'est pas obligatoire, mais conseillée sur le wiki d'archlinux. Sauf cas très spécifique, vous n'aurez pas besoin de garder les deux activés. Ajoutez les modules propres à iptables en liste noire.

```
$ cat /etc/modprobe.d/iptables_desactivation.conf
```

```
blacklist x_tables
blacklist iptable_nat
blacklist iptable_raw
blacklist iptable_mangle
blacklist iptable_filter
blacklist ip_tables
blacklist ipt_MASQUERADE
blacklist ip6table_nat
blacklist ip6table_raw
blacklist ip6table_mangle
blacklist ip6table_filter
blacklist ip6_tables
```

Service systemd

Afin de charger votre configuration à chaque démarrage, créez votre propre fichier de service systemd.

```
$ cat /etc/systemd/system/nftables.service
[Unit]
Description=nftables
Documentation=man:nftables(8)
Wants=network-pre.target
Before=network-pre.target
[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/bin/nft -f /etc/nftables.conf
ExecStop=/usr/bin/nft flush ruleset
[Install]
WantedBy=multi-user.target
```

Activez ensuite votre service :

```
systemctl enable nftables
```

Version utilisée

Nftables est encore un projet jeune et peut être packagé dans une version ancienne en fonction de votre distribution. Assurez-vous d'avoir à minima la version 0.7 soit via les backports, soit directement via les sources du projet.

Conclusion

Nous avons vu ensemble les bases de créations de règles avec nftables. Nous avons pu étudier quels étaient les concepts novateurs par rapport à son prédécesseur iptables et avons illustré cela par des exemples. Nftables est aujourd'hui assez mature (dans sa dernière version) pour être utilisé en production.

Documentation

- Documentation officielle
  https://wiki.nftables.org/wiki-nftables/index.php/Main_Page
- Schémas de propagation des paquets dans netfilter
  https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks
- Explication en profondeur du suivi de connexion dans netfilter (valable pour iptables et nftables)
  http://www.iptables.info/en/connexion-state.html
- Présentation sur la journalisation par un développeur netfilter
  http://moutane.net/RMLL2014/day_1-1620-Eric_Leblond-Netfilter_logging_at_the_nftables_age.pdf
- Tutoriel nftable (de qualité)
  https://people.netfilter.org/pablo/nft-tutorial.pdf

2 - Rappel des avis émis

Dans la période du 24 au 30 juillet 2017, le CERT-FR a émis les publications suivantes :

- CERTFR-2017-AVI-234 : Vulnérabilité dans les produits Citrix
- CERTFR-2017-AVI-235 : Multiples vulnérabilités dans Joomla!
- CERTFR-2017-AVI-236 : Multiples vulnérabilités dans Google Chrome
- CERTFR-2017-AVI-237 : Multiples vulnérabilités dans Fortinet FortiOS et FortiAnalyzer
- CERTFR-2017-AVI-238 : Multiples vulnérabilités dans McAfee Web Gateway
- CERTFR-2017-AVI-239 : Multiples vulnérabilités dans les produits VMware
- CERTFR-2017-AVI-240 : Multiples vulnérabilités dans Fortinet FortiOS
- CERTFR-2017-AVI-241 : Multiples vulnérabilités dans Microsoft Outlook

Gestion détaillée du document

31 juillet 2017 version initiale.