nftables, bien plus que :%s/ip/nf/g

Éric Leblond

Nefilter Coreteam

5 juin 2013

Éric Leblond

Hacker et consultant

- Consultant indépendant Open Source et Sécurité
- Initiateur du projet de pare-feu authentifiant NuFW
- Core développeur de l'IDS/IPS Suricata

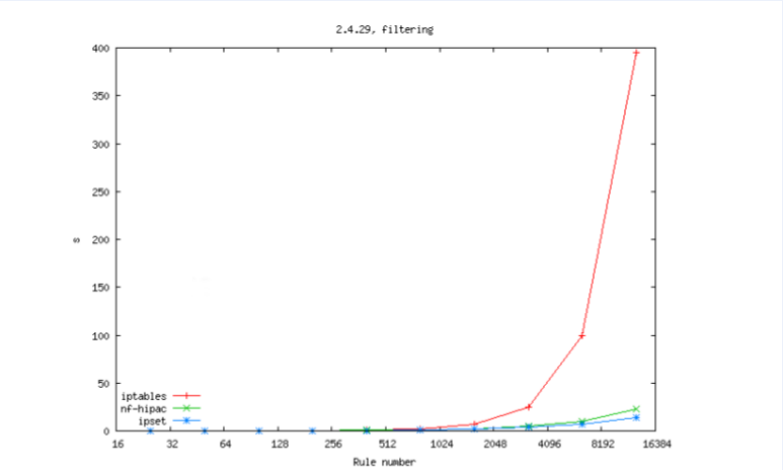
Membre de la coreteam Netfilter

- Travail sur les interactions noyau-espace utilisateur
- Hacking noyau
- Mainteneur de ulogd2
- Port du pare-feu Openoffice vers Libreoffice

Performance

Ajouter une règle

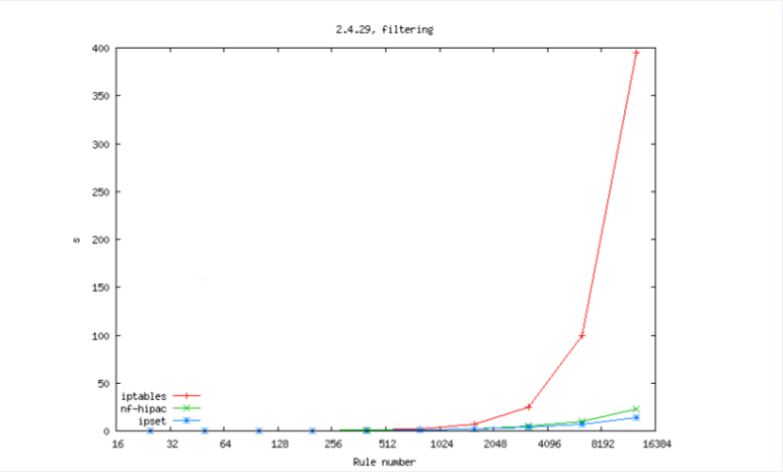
- Remplacement atomique du jeu de règles complet
- Impact en terme de performance



Performance

Ajouter une règle

- Remplacement atomique du jeu de règles complet
- Impact en terme de performance



Gestion des ensembles

- Explosion combinatoire : une règle par serveur et par protocole
- Des jeux de règles de plus en plus dynamiques
- Une gestion des ensembles assurée par ipset

Évolutivité

Duplication de code

- Code similaire pour de nombreux modules Netfilter
- Sans mise en commun
- Parsing manuel

Mise à jour système pénible

- Échange binaire entre espace utilisateur et noyau
- Aucune modification possible sans toucher le noyau

Intégration par exec

- Pas de bibliothèques utilisables
- Les frontends forkent des commandes iptables

Un nouveau système de filtrage

- Développé et présenté en 2008 par Patrick McHardy
- Remplace iptables et l'infrastructure de filtrage
- Pas de changements dans
  - Les hooks
  - Le suivi de connexions
  - Les helpers

Un nouveau système de filtrage

- Développé et présenté en 2008 par Patrick McHardy
- Remplace iptables et l'infrastructure de filtrage
- Pas de changements dans
  - Les hooks
  - Le suivi de connexions
  - Les helpers

Un nouveau langage

- Basé sur une grammaire
- Accessible depuis une bibliothèque

Un nouveau système de filtrage

- Développé et présenté en 2008 par Patrick McHardy
- Remplace iptables et l'infrastructure de filtrage
- Pas de changements dans
  - Les hooks
  - Le suivi de connexions
  - Les helpers

Un nouveau langage

- Basé sur une grammaire
- Accessible depuis une bibliothèque

Une communication basée sur Netlink

- Des modifications atomiques
- Un système de notification

Un filtrage basé sur une pseudo-machine à état

Inspiré de BPF

- 4 registres
- un verdict
- ensemble d'instructions extensibles

Un filtrage basé sur une pseudo-machine à état

Inspiré de BPF

- 4 registres
- un verdict
- ensemble d'instructions extensibles

Add Some Magic ?

```
reg = pkt.payload[offset, len]
reg = cmp(reg1, reg2, EQ)
reg = pkt.meta(mark)
reg = lookup(set, reg1)
reg = ct(reg1, state)
```

Un filtrage basé sur une pseudo-machine à état

Inspiré de BPF

- 4 registres
- un verdict
- ensemble d'instructions extensibles

Add Some Magic ?

```
reg = pkt.payload[offset, len]
reg = cmp(reg1, reg2, EQ)
reg = pkt.meta(mark)
reg = lookup(set, reg1)
reg = ct(reg1, state)
```

Créations d'opérations faciles

```
reg1 = pkt.payload[offset_src_port, len]
reg2 = pkt.payload[offset_dst_port, len]
reg = cmp(reg1, reg2, EQ)
```

Avantages de l'approche

Un sous ensemble noyau réduit

- Un nombre limité d'opérateurs et d'instructions
- et une machine à état
- Pas de codes pour chaque type de vérification
  - Un match sur les adresses codé comme match de port
  - De nouveaux matchs possibles sans modification noyau

Avantages de l'approche

Un sous ensemble noyau réduit

- Un nombre limité d'opérateurs et d'instructions
- et une machine à état
- Pas de codes pour chaque type de vérification
  - Un match sur les adresses codé comme match de port
  - De nouveaux matchs possibles sans modification noyau

Moins de mise à jour noyau

- Un nouveau match ne nécessitera pas forcément un nouveau noyau

Utilisation basique

Mode fichier

```
nft -f ipv4-filter
```

Utilisation basique

Mode fichier

```
nft -f ipv4-filter
```

Mode ligne de commande

```
nft add rule ip filter input tcp dport 80 drop
nft list table filter -a
nft delete rule filter output handle 10
```

Utilisation basique

Mode fichier

```
nft -f ipv4-filter
```

Mode ligne de commande

```
nft add rule ip filter input tcp dport 80 drop
nft list table filter -a
nft delete rule filter output handle 10
```

Mode CLI

```
# nft -i
nft> list table
<cli>:1:12-12: Error: syntax error, unexpected end of file, expecting stri
list table
           ^
nft> list table filter
table filter {
  chain input {
    ip saddr 1.2.3.4 counter packets 8 bytes 273
```

Gestion des ensembles

Intérêt des ensembles

- Une seule règle évaluée
- Clarté du jeu de règles
- Gestion des évolutions

Gestion des ensembles

Intérêt des ensembles

- Une seule règle évaluée
- Clarté du jeu de règles
- Gestion des évolutions

Ensemble anonyme

```
nft add rule ip global filter \
     ip daddr {192.168.0.0/24, 192.168.1.4} \
     tcp dport {22, 443} \
     accept
```

Gestion des ensembles

Intérêt des ensembles

- Une seule règle évaluée
- Clarté du jeu de règles
- Gestion des évolutions

Ensemble anonyme

```
nft add rule ip global filter \
     ip daddr {192.168.0.0/24, 192.168.1.4} \
     tcp dport {22, 443} \
     accept
```

Ensemble nommé

```
nft add set global ipv4_ad { type ipv4_address;}
nft add element global ipv4_ad { 192.168.1.4, 192.168.1.5}
nft delete element global ipv4_ad { 192.168.1.5}
nft add rule ip global filter ip saddr @ipv4_ad drop
```

Mapping

Principe et intérêts

- Dictionnaire associatif liant deux notions
- Un match sur la clé entraine l'utilisation de la valeur
- Utilisant des adresses, des interfaces, des verdicts

Mapping

Principe et intérêts

- Dictionnaire associatif liant deux notions
- Un match sur la clé entraine l'utilisation de la valeur
- Utilisant des adresses, des interfaces, des verdicts

Exemples

- Mapping anonyme :

```
# nft add rule filter output ip daddr vmap \
    {192.168.0.0/24 =\> drop, 192.168.0.1 =\> accept}
```

- Mapping nommé :

```
# nft -i
nft> add map filter verdict_map { type ipv4_address  => verdict; }
nft> add element filter verdict_map { 1.2.3.5 => drop}
nft> add rule filter output ip daddr vmap @verdict_map
```

Exemple de mise en oeuvre

```
set web_servers {
 type ipv4_address
 elements = { 192.168.1.15, 192.168.1.5}
}
map admin_map {
 type ipv4_address => verdict
 elements = { 192.168.0.44 => jump logmetender, \
              192.168.0.42 => jump logmetrue, 192.168.0.33 => accept}
}
chain forward {
 ct state established accept
 ip daddr @web_servers tcp dport ssh ip saddr map @admin_map
 ip daddr @web_servers tcp dport http log accept
 ip daddr @web_servers tcp dport https accept
 counter log drop
}
chain logmetender {
 log limit 10/minute accept
}
chain logmetrue {
 counter log accept
}
}
```

Transition et évolution

Une compatibilité iptables complète

- iptables-nftables
  - Des binaires compatibles iptables
  - Utilisant le framework nftables
- Cohabitation possible
- Une migration progressive

Transition et évolution

Une compatibilité iptables complète

- iptables-nftables
  - Des binaires compatibles iptables
  - Utilisant le framework nftables
- Cohabitation possible
- Une migration progressive

Une bibliothèque de haut niveau

- Pour utilisation dans les frontends
- Pour les mécanismes de contrôle d'accès des gestionnaires de réseaux

Conclusion

Une évolution considérable

- Résolution des problèmes de iptables
- Une réponse aux nouveaux usages
  - Gestion des ensembles
  - Matchs complexes

Une disponibilité prévue fin 2013

- Finalisation de la compatibilité iptables
- Bibliothèque haut niveau
- Debug et diverses fonctionnalités

Questions ?

Avez-vous des questions ?

Merci à

- Toute l'équipe de Netfilter
- Astaro/Sophos pour le financement
- Google pour GSoC 2013

Pour aller plus loin

- Netfilter : `http://www.netfilter.org`
- Nftables quick & dirty : `https://t.co/cM4zogob8t`

Me joindre

- Courriel : `eric@regit.org`
- Twitter : @Regiteric