

Csomagszűrés, maszkolás

Kadlecsik József

KFKI RMKI

kadlec@sunserv.kfki.hu

Tartalom

- Szoftver-telepítés: kernel, iptables
- Routing
- Stateless és stateful szűrési példák
- NAT
- Szabály-finomítás
- iptables-save, iptables-restore
- nf_conntrack, xt_match, xt_TARGET

Szoftver-telepítés

- Szoftvert csak tiszta forrásból:
 - kernel: <ftp.kernel.org>
 - iptables: www.netfilter.org
 - [patch-o-matic-ng: www.netfilter.org]
- kernel fordítás
- iptables fordítás
- bővítés patch-o-matic-ng-ből

Routing

- shell scriptből:

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

```
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
```

- /etc/sysctl.conf:

```
net/ipv4/ip_forward=1
```

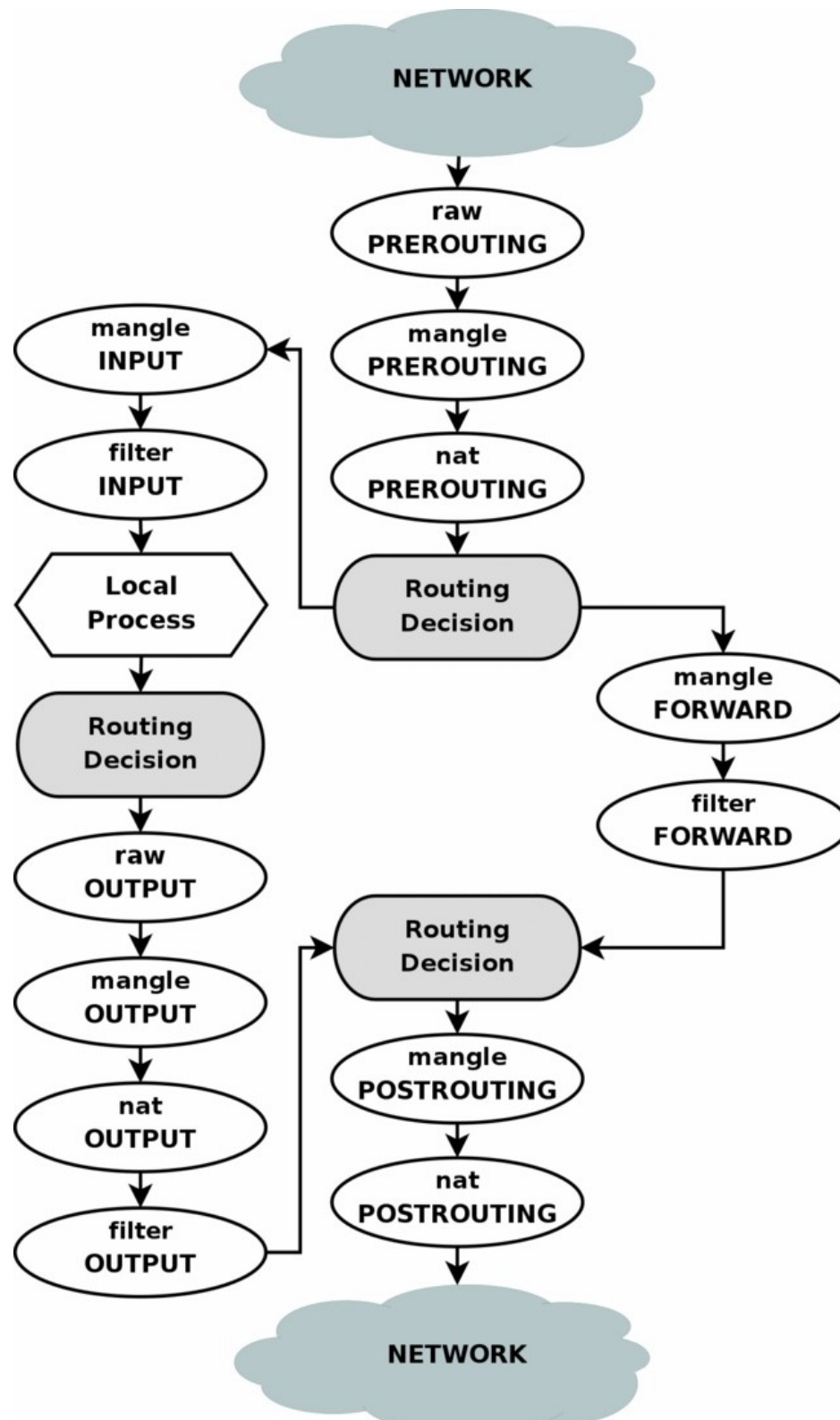
```
net/ipv6/conf/all/forwarding=1
```

- /etc/network/options (*deprecated*):

```
ip_forward=yes
```

rp_filter: reverse-path filtering

- `/etc/network/options:`
`spoofprotect=yes | no`
- `/proc/sys/net/ipv4/conf/all/rp_filter`
- hátrányok:
 - naplózás nélkül eldobja a csomagot
 - komplikált hálózatot nem képes kezelni



Szabályok és láncok törlése

- Táblák függetlenek
- Külön kell a szabályokat **majd** a láncokat törölni

```
for table in filter mangle nat raw; do
```

```
    iptables -t $table -F
```

```
    iptables -t $table -X
```

```
done
```

Szabály-építkezés

- raw table: speciális szűrési szabályok
- mangle table: csomag módosítás
- nat table: címfordítás
- filter table: szűrési szabályok
 - Csak amit explicit megengedünk, az szabad
 - default policy?

Egyszerű kliens-szabály I.

- Állapot-nélküli szűrési szabály, a'la *ipchains*
- Mindkét irányt kezelni kell

```
iptables -A FORWARD -s 192.168.0.0/24 \  
-p tcp --dport 22 -j ACCEPT
```

```
iptables -A FORWARD -d 191.268.0.0/24 \  
-p tcp --sport 22 -j ACCEPT
```

```
iptables -A FORWARD -j DROP
```

Egyszerű kliens-szabály II.

- Állapot-figyelő szűrési szabály
- Mindkét irányt kezelni kell
- A két irány nem szimmetrikus!

```
iptables -A FORWARD -m state --state ESTABLISHED \  
-j ACCEPT
```

```
iptables -A FORWARD -s 192.168.0.0/24 \  
-p tcp --sport 22 -m state --state NEW -j ACCEPT
```

```
iptables -A FORWARD -j DROP
```

Egyszerű kliens-szabály III.

- Állapot-figyelő szűrési szabály
- Mindkét irányt kezelni kell
- A két irány nem szimmetrikus!
- Naplózás nélkül vakok vagyunk.

```
iptables -N accept
```

```
iptables -A accept -j LOG --log-prefix "accept: "
```

```
iptables -A accept -j ACCEPT
```

```
iptables -N drop
```

```
iptables -A drop -j LOG --log-prefix "drop: "
```

```
iptables -A drop -j DROP
```

Egyszerű kliens-szabály IV.

- Állapot-figyelő szűrési szabály
- Mindkét irányt kezelni kell
- A két irány nem szimmetrikus!
- Naplózás nélkül vakok vagyunk.
- Konzolra naplózni általában nem szerencsés (emerg, alert, crit, err, *warning*, notice, info, debug):

```
dmesg -n 4
```

```
klogd -c 4
```

Egyszerű kliens-szabály V.

```
iptables -N accept
```

```
iptables -A accept -j LOG --log-prefix "accept: "
```

```
iptables -A accept -j ACCEPT
```

```
iptables -N drop
```

```
iptables -A drop -j LOG --log-prefix "drop: "
```

```
iptables -A drop -j DROP
```

```
iptables -A FORWARD \
```

```
    -m state --state ESTABLISHED,RELATED -j ACCEPT
```

```
iptables -A FORWARD -s 192.168.0.0/24 \
```

```
    -p tcp --sport 22 -m state --state NEW -j accept
```

```
iptables -A FORWARD -j drop
```

Egyszerű kliens-szabály V.

- *ssh megy, scp nem! Miért??*

Egyszerű kliens-szabály VI.

- Ne szűrjünk minden ICMP hibaüzenetet:
 - ICMP destination unreachable
 - Time exceeded
 - *Parameter problem, Source quench*
- Ha az uplink szűri valahol az ICMP hibaüzeneteket (ICMP dest. unreach/Fragmentation needed), akkor használjuk a TCPMSS targetet

```
iptables -t mangle -A FORWARD -o eth0 \  
-p tcp --tcp-flags SYN,RST SYN \  
-j TCPMSS --clamp-mss-to-pmtu # --set-mss 1460
```

Egyszerű kliens-szabály VII.

- Stateful szabályokkal egyszerűbb az élet:
 - világosabb, egyszerűbb szabály-rendszer
 - ICMP hibákat intelligensen kezelni tudjuk
 - Támogatott protokolloknál segédcsatornákat szintén egyszerűen kezelhetjük:

```
modprobe nf_conntrack_ftp
```

```
modprobe ip_conntrack_irc
```

NAT I.

- Ha lehetséges, ne használjunk NAT-ot:
 - nagy overhead
 - false sense of security
 - csak egy hack, hogy megkerüljük a gyorsan fogyó IPv4 címeket
- Protokoll helperek: `ip_nat_ftp`, `ip_nat_irc`, ...
- SNAT vs. MASQUERADE

```
iptables -t nat -A POSTROUTING \  
-s 192.168.0.0/24 -o eth0 -j SNAT --to-source x.y.z.w  
# iptables -t nat -A POSTROUTING \  
# -s 192.168.0.0/24 -o eth0 -j MASQUERADE
```

NAT II.

- Ha kell használnunk NAT-ot és szervert kell elérhetővé tennünk a NAT-olt hálózatról: DNAT
- REDIRECT használható (transzparens) proxy építéshez

```
iptables -t nat -A PREROUTING \  
-d x.y.z.w -p tcp --dport 80 \  
-j DNAT --to-destination 192.168.0.1
```

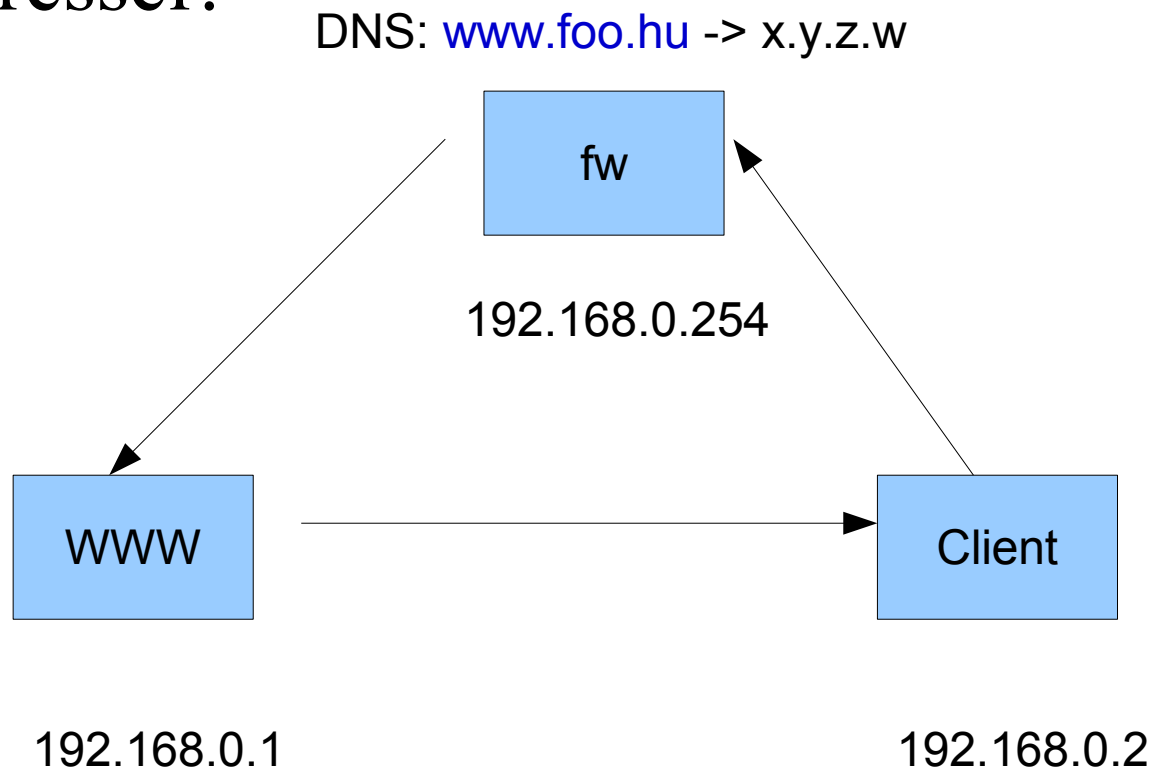
NAT III.

- Nem elég a DNAT: az új szolgáltatásokat engedélyeznünk is kell :-).

```
iptables -A FORWARD \  
-d 192.168.0.1 -p tcp --dport 80 \  
-m state --state NEW -j accept
```

NAT IV.

- Mi van a lokális hálózatról, ugyanazon névvel való hozzáféréssel?



NAT V.

- Lokális hálózatról, ugyanazon névvel való hozzáférés
 - split-DNS: elegánsabb, kisebb terhelés
 - SNAT

```
iptables -t nat -A POSTROUTING \  
-s 192.168.0.0/24 -o eth1 \  
-j NETMAP --to 192.168.1.0/24
```

Szabály-finomítás I.

- Az egyszerű DROP néhány alkalmazásnál hosszú timeout-ot eredményezhet: ident + smtp
 - DROP helyett REJECT target:

```
iptables -A drop \  
-p tcp --dport 113 \  
-j REJECT --reject-with tcp-reset
```

Szabály-finomítás II.

- Naplózás korlátozható a limit match segítségével
 - előny: DoS támadások, portscannelés nem fojt bele minket a logokba
 - hátrány: kevésbé látjuk, pontosan mi is történik
- Használhatjuk feltételesen, pl. a condition match-al kombinálva

Szabály-finomítás II. folyt

```
iptables -N drop
```

```
# /proc/net/ipt_condition/limit
```

```
iptables -A drop -m condition --condition limit \
```

```
    -m limit --limit 2/second --limit-burst 3 \
```

```
    -j LOG --log-prefix "drop (limit): "
```

```
iptables -A drop -m condition --condition ! limit \
```

```
    -j LOG --log-prefix "drop: "
```

```
iptables -A drop -j DROP
```

Szabály-finomítás III.

- Használhatunk dinamikus csapdákat (recent, set/SET) támadókkal szemben (portscan, DoS, spammer, virus) tiltásukra, lassításukra (TARPIT)

```
iptables -t raw -A PREROUTING -m recent \  
  --name attackers --update --seconds 60 -j NOTRACK  
iptables -t raw -A PREROUTING -p tcp --dport 137:139 \  
  -m recent --name attackers --set -j NOTRACK  
iptables -N tarpit  
iptables -A tarpit -p tcp -j TARPIT  
iptables -A tarpit -j DROP  
iptables -A FORWARD -m state --state UNTRACKED \  
  -j tarpit
```

iptables-save

- Dinamikus szabály-módosítás nem hatékony
- iptables-save és iptables-restore

Lineáris szabály-feldolgozás

- Nem csak a szabály-betöltést kell optimalizálni:
szabály-feldolgozás:
 - protokoll-alapú láncok
 - host/network alapú láncok

nf_conntrack

- “protokoll-független” conntrack
 - IPv4 és IPv6 támogatás
 - nf_conntrack_ipv4, nf_conntrack_ipv6
 - IPv4 NAT támogatott
 - IPv6 NAT nem lesz
- x_tables modulok: ipt_ és ip6t_ aliasok

Hasznos linkek

- <http://www.netfilter.org>
- <http://iptables-tutorial.frozentux.net>