

Ipset

József Kadlecsik

[<kadlec@blackhole.kfki.hu>](mailto:kadlec@blackhole.kfki.hu)

KFKI Research Institute for Particle and Nuclear
Physics
Budapest, Hungary

Content

- Why and where do you might need it
- ipset
- ipset and iptables

Special firewalls

- High number of rules
 - Fast matching algorithms
- Often changed rules
 - Storage structures which can be changed fast
- Low RAM machines
 - Memory optimized storage structures

Special firewalls: iptables?

- iptables: <http://www.netfilter.org/>
- High number of rules: slow
 - Linear evaluation
- Often changed rules: slow
 - Between kernel-userspace all rules are passed back and forth at adding/deleting a single rule. Rules are stored in a blob.
 - iptables-restore
- Medium RAM requirements

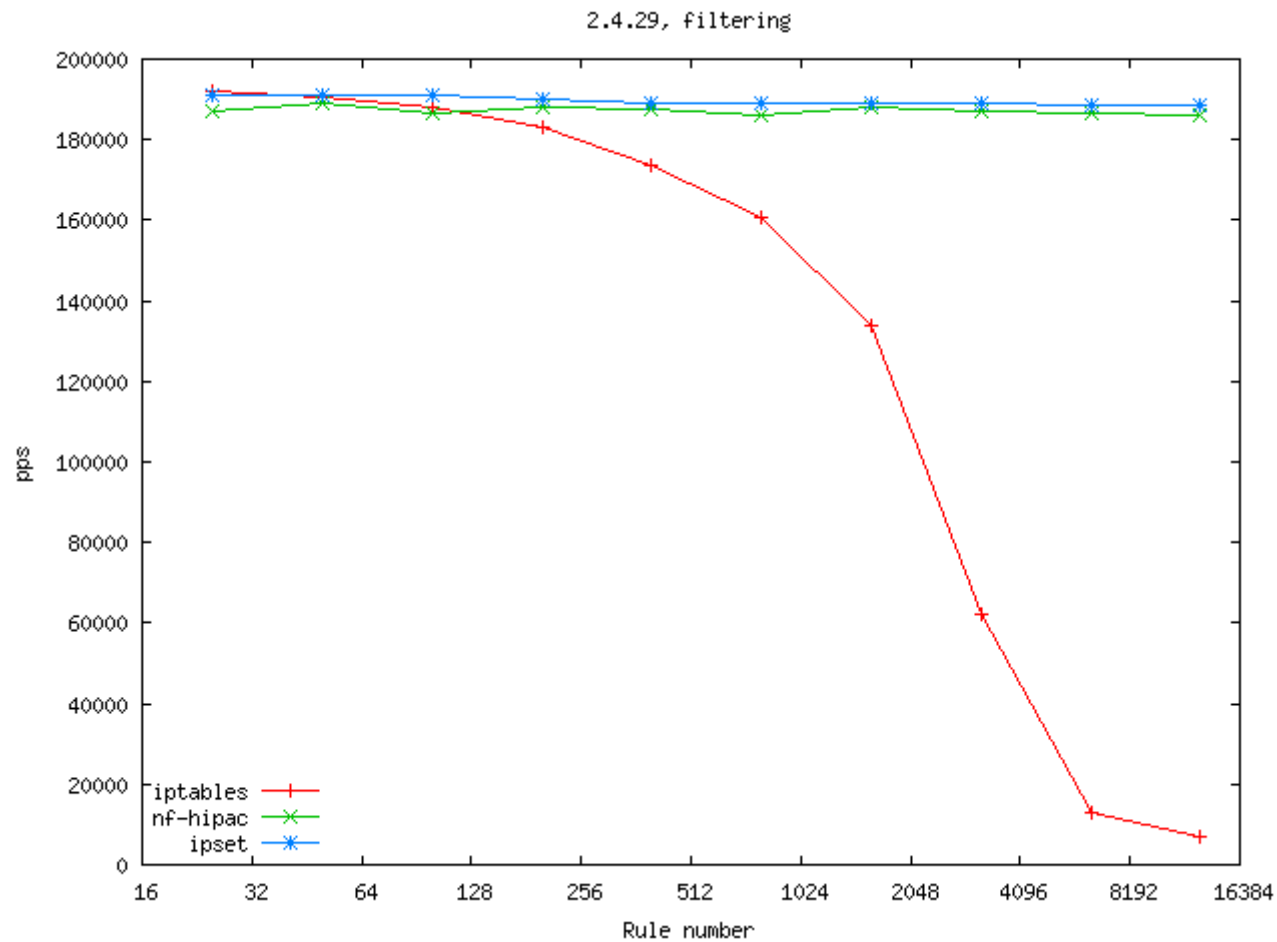
Special firewalls: nf-hipac?

- nf-hipac: <http://www.hipac.org/>
- High number of rules: fast
 - Complex matching algorithms
- Often changed rules: fast
 - Just the new/to be deleted rule passed; hashes, trees.
- Memory requirements?

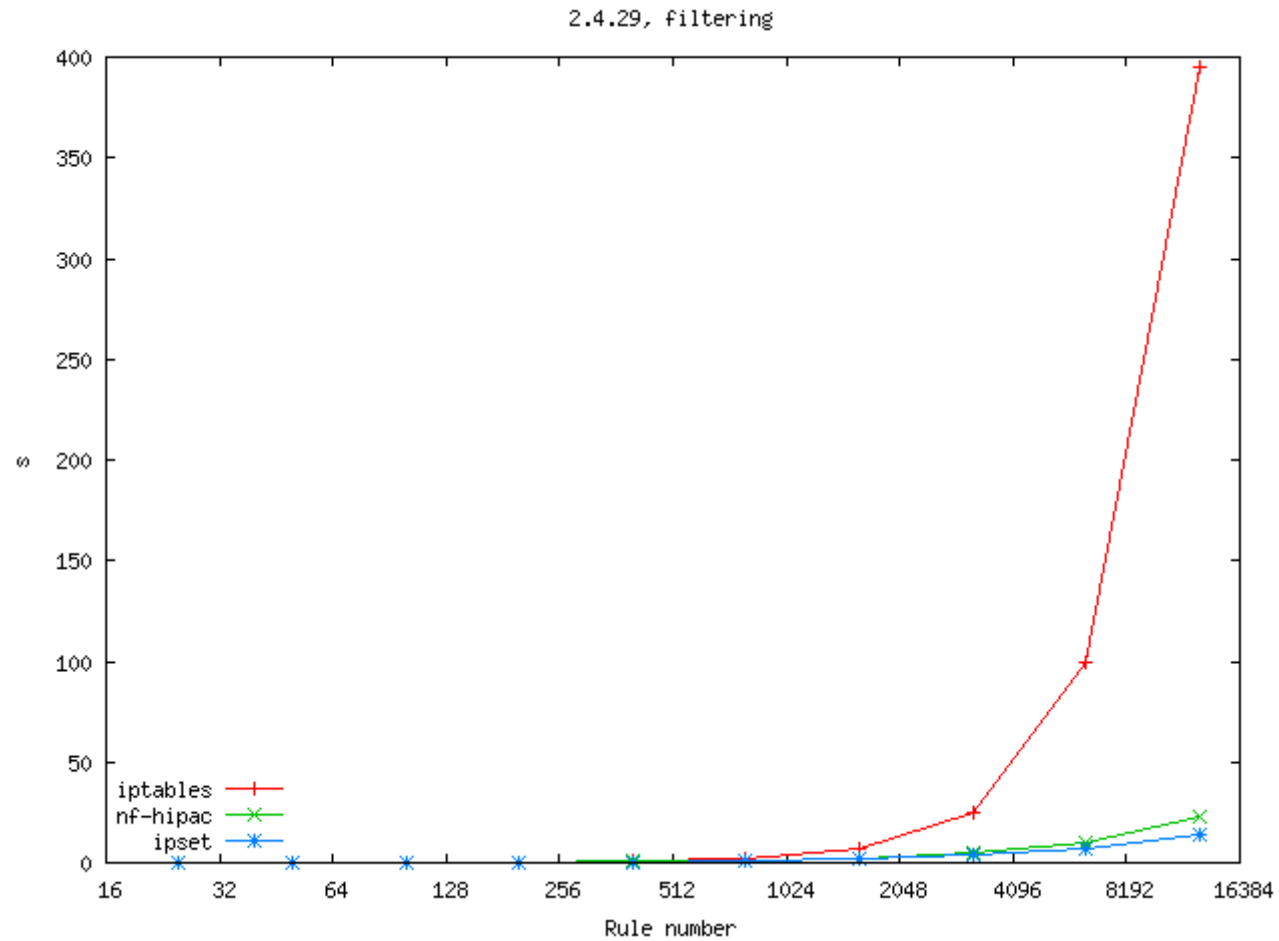
Special firewalls: ipset?

- ipset: <http://ipset.netfilter.org/>
- High number of rules: fast
 - Simple algorithms
- Often changed rules: fast
 - Just the new/to be deleted rule passed; arrays, hashes, trees.
- Memory requirement can be low

iptables, nf-hipac and ipset I.



iptables, nf-hipac and ipset II.



ipset and ippool

- Joakim Axelsson: ippool, bitmap type
- Joakim Axelsson, Patrick Schaaf és Martin Josefsson: modular ippool, bitmap and macipmap types
- ipset: partly rewritten ippool, couple of new set types

ipset

- In-kernel data sets, in which you can store IP addresses, port numbers, MAC addresses or combinations
- Different storage types: (bit)map, hash, tree
- Userspace program to handle the sets: `ipset`
- You can refer to the sets by iptables: `set match` and `SET target`
- [The element of the sets can be bound to another set: `binding`]

ipmap (bitmap) type

- Store IP addresses from a range or netblock
 - An IP address is stored in one **bit**
- Max 65536 element
- Typical usage: general allow/deny policies for individual machines or netblocks

ipmap examples

- Store IP addresses:

```
ipset -N set1 ipmap --from 192.168.0.0 --to 192.168.255.255
```

```
[ipset -N set1 ipmap --network 192.168.0.0/16]
```

```
ipset -A set1 192.168.0.1
```

```
ipset -A set1 192.168.0.5
```

- Store same-size netblocks:

```
ipset -N set2 ipmap --network 0.0.0.0/0 --netmask 16
```

```
ipset -A set2 10.1.0.0
```

```
ipset -A set2 10.7.0.1
```

macipmap type

- Store IP and MAC address pairs (IP addresses must come from a range or netblock)
 - 8 byte
- Only source MAC address
- At first matching MAC addresses can automatically filled out

```
ipset -N macipset1 macipmap --from 192.168.0.0 \  
--to 192.168.255.255
```

```
ipset -A macipset1 192.168.1.1,00:01:23:45:67:89
```

```
ipset -A macipset1 192.168.2.3
```

portmap type

- Store port numbers
 - Every port number is stored in a bit

```
ipset -N portset1 portmap --from 0 --to 1024
```

```
ipset -A portset1 22
```

iphash type

- Store random IP addresses/same size netblock in a fix sized (`hashsize`) hash
 - Rehashing in the case of clash (`probes`)
 - Grow hash if it's “full” (`resize`)
- Tune for speed:
 - large `hashsize`, small `probes` and large `resize`
- Tune for low memory:
 - small `hashsize`, large `probes` and small `resize`

iphash examples

- IP addresses

```
ipset -N myhash1 iphash --hashsize 1024 --probes 2 -resize 50
```

```
ipset -A myhash1 10.1.1.1
```

```
ipset -A myhash1 192.168.52.3
```

- Same size netblocks:

```
ipset -N myhash2 iphash --hashsize 64 --probes 8 -resize 0 \  
--netmask 24
```

```
ipset -A myhash2 10.1.1.0
```

```
ipset -A myhash2 192.168.52.0
```

nethash type

- Similar to iphash, but different size netblocks of size between /1 and /31 can be stored in nethash
 - Network address and netmask together stored in 32 bits

```
ipset -N hash3 nethash --hashsize 1024 --probes 2 --resize 50
```

```
ipset -A hash3 192.168.1.0/24
```

```
ipset -A hash3 10.1.8.0/21
```

Which type for netblocks?

- Same size netblocks from a network: ipmap

```
ipset -N map1 ipmap --network 192.168.0.0/16 --netmask 24
```

```
ipset -A map1 192.168.1.0/24
```

```
ipset -A map1 192.168.2.0/24
```

- Same size random netblocks: iphash

```
ipset -N hash1 iphash --netmask 24
```

```
ipset -A hash1 10.1.1.0/24
```

```
ipset -A hash1 192.168.2.0/24
```

- Different size netblocks: nethash

```
ipset -N hash2 nethash
```

```
ipset -A hash2 192.168.1.0/24
```

```
ipset -A hash2 10.1.8.0/21
```

ipporthash type

- Store IP addresses from a max /16 netblock and port number pairs
 - One IP address + port pair stored in 32 bits
- Store server,service pairs

```
ipset -N porthash ipporthash --network 192.168.0.0/16 \  
      --hashsize 1024 --probes 2 --resize 5
```

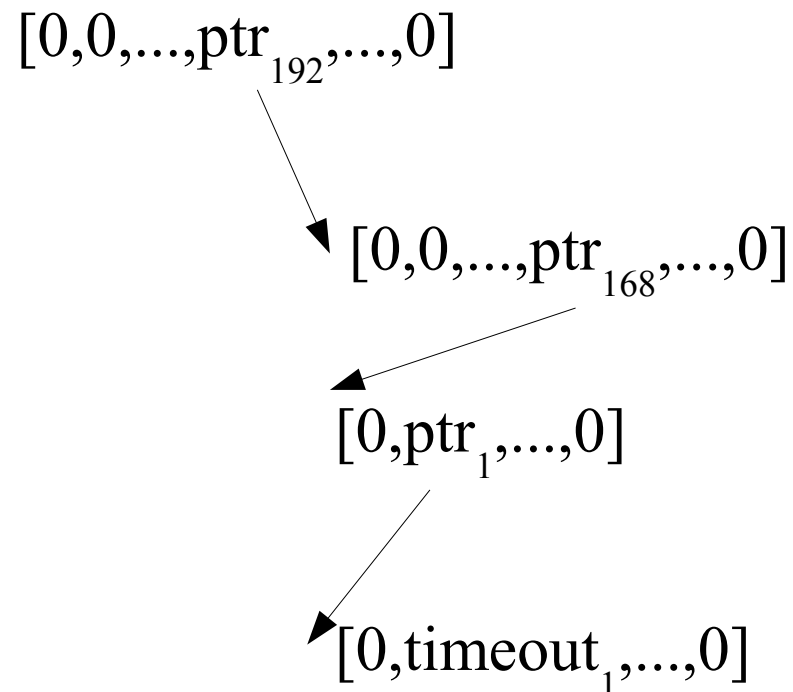
```
ipset -A porthash 192.168.1.1,22
```

```
ipset -A porthash 192.168.1.1,80
```

```
ipset -A porthash 192.168.1.4,25
```

iptree type

- Store IP addresses in a tree like structure with automatic timeout support: címek 192.168.1.1



iptree example

```
ipset -N tree1 iptree --timeout 600
```

```
ipset -A tree1 192.168.1.1,300
```

```
ipset -A tree1 192.168.2.8,0
```

iptreemap type

- Similar to iptree, last octet stored in a bitmap
- You can add/delete IP addresses, ranges or netblocks in one command:

```
ipset -N tree2 iptreemap --timeout 600
```

```
ipset -A tree2 192.168.1.1
```

```
ipset -A tree2 192.168.2.0/24
```

```
ipset -A tree2 192.168.3.1-192.168.3.18
```

set match

- Match elements in a set from netfilter/iptables by the set match
- `-m set <setname> src|dst[,src|dst]`

```
ipset -N servers ipporthash --network 192.168.0.0/24
```

```
ipset -A servers ipporthash 192.168.0.1,25
```

```
ipset -A servers ipporthash 192.168.0.2,80
```

```
iptables -A FORWARD -m set --match-set servers dst,dst \  
-m state --state NEW -j log-accept
```

SET target

- Add/delete elements to/from a set from netfilter/iptables by the SET target
- `-j SET --add-set|--del-set <setname> src|dst[,...]`

```
ipset -N spammers iptree --timeout $((60*60*24*7))
```

```
iptables -A FORWARD -d <honeypot> -p tcp --dport 25 \  
-j SET --add-set spammers src
```

```
iptables -A FORWARD -p tcp --dport 25 \  
-m set --set spammers src -j TARPIT
```

ipset save and restore

- Plain text format (similar to iptables-save) with a strict ordering of the commands

```
ipset -S > ipset.rules
```

```
ipset -R < ipset.rules
```

Swap sets

- Sets referred by iptables cannot be deleted: but can be swapped to another (existing) set:

```
ipset -N main-set ...
```

```
iptables -A ... -m set --set main-set ...
```

```
ipset -N main-set-new ...
```

```
ipset -W main-set main-set-new
```

Firewall with ipset I.

```
# All internal machines which are allowed to access the Internet
# (could be a macipmap type if all is on the LAN of the firewall):
ipset -N clients ipmap --network 192.168.0.0/16
ipset -A clients 192.168.10.1
...
# All internal server with public services
ipset -N servers ipporthash --network 192.168.0.0/16
ipset -A servers 192.168.0.1,22
ipset -A servers 192.168.0.2,25
...
```

Firewall with ipset II.

```
# Stateful firewall:
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -J ACCEPT
# Chain definitions for logging
...
# Server rule
iptables -A FORWARD -m set --set servers dst,dst \
          -m state --state NEW -J log-accept
# Client rule
iptables -A FORWARD -m set --set clients src \
          -m state --state NEW -J log-accept
# Otherwise we log and drop
iptables -A FORWARD -j log-drop
```

TODO

- New types:
 - ipportip triple
 - Union of sets
- IPv6 support
- Submit for kernel inclusion :-)

Thank you!