ipset
a tool for faster, more efficient firewalling with iptables

József Kadlecsik
<kadlec@blackhole.kfki.hu>
MTA Wigner FK

Challenges to firewalling with iptables

- Large number of rules
  - Rule evaluation is linear
- Often changed rules
  - iptables must handle the whole table

Rules

- Focus on filtering
  - Exotic matches, targets are not common
- Typical rules
  - Allow/deny a service at a given server, optionally limited to given clients
  - Allow/deny a service for a client machine, optionally limited to given servers

Ippool

- 2000: Joakim Axelsson: bitmap type
- 2001-2002: Joakim Axelsson, Patrick Schaaf and Martin Josefsson: modular, bitmap and macipmap types

Ippool II.

- 2003-2004: patches from me
- 2004: Patrick Schaaf:

  Regarding backwards compatibility, my vote would be not to care, and name the new thing with a new name. Proposal: ipset
- 2011: ipset 6.x

Ipset I.

- Data sets which can store given combinations of data types
  - IP(v4/v6) address, netblock
  - MAC address
  - Protocol and port number/type
  - Interface name
  - Mark value
  - Set name
- Kernel API

Ipset II.

- Different storage methods:
  - Bitmap
  - Hash
  - List
- Set element extensions:
  - Timeout
  - Counters
  - Comment
  - Skbinfo

- Set dimension
  - bitmap:ip
  - hash:ip,port
  - hash:ip,port,ip

Userspace tool

- `ipset` :-)
- Minimal dependency
  - libmnl
- Commandline syntax similar to `ip`
  - Backward compatibility kept with older ipset syntax

Command keywords

- Whole set:
  - create, destroy, list, save, restore, flush, rename, swap
- Set element:
  - add, del, test
- Single letter equivalents

Create and add, del, test syntax

- Create a set: method, data types must be specified
  - method:data_type[,data_type[,data_type]

```
# ipset create test hash:ip,port,ip
```

- Add/delete/test element: components in the given order must be specified

```
# ipset add test 192.168.1.1,udp:53,8.8.8.8
# ipset test test 192.168.1.1,udp:53,8.8.8.8
```

Bitmap method

- Continuous bit vector where every bit represents one address from a range of addresses:

  IPv4 address = Base IPv4 address + bit position

- Can be generalized to support to store
  - Same size IPv4 netblocks
  - IPv4 + MAC address pairs – MAC addresses stored in another data vector
  - TCP or UDP port numbers
- Limited to 65536 elements (/16)

bitmap:ip

- Store IPv4 addresses from a range

```
ipset n set1 bitmap:ip range 10.0.0.0-10.0.0.255
ipset a set1 10.0.0.1
ipset a set 10.0.0.5-10.0.0.15
```

- Store same size IPv4 netblocks

```
ipset c set2 bitmap:ip 0.0.0.0/0 netmask 16
ipset a set2 10.1.0.0        # 10.1.0.0/16
ipset a set2 10.7.0.0        # 10.7.0.0/16
```

bitmap:ip,mac

- Store IPv4 and MAC address pairs
  - Source MAC addresses only
  - Can be added without MAC address, first match will fill out MAC

```
ipset c set3 bitmap:ip,mac 192.168.0.0/16

ipset a set3 192.168.1.1,00:01:23:45:67:89

ipset a set3 192.168.1.2
```

Hashing

- Map data space into a fixed data space, where the algorithm must be
  - Deterministics
  - Uniform
- Linux kernel
  - jhash
- Collision handling
  - Typically linked lists

Hash method

- Hash size is forced to power of two, for speed
- Collided elements are stored in arrays instead of linked lists
  - 4-12 x elem size
  - 12 x elem size array full: grow hash

hash:ip

- Store random IP addresses

```
ipset n set4 hash:ip hashsize 1024
ipset a set4 10.1.1.1
ipset a set4 192.168.168.168
```

- Also, can store same size netblocks

```
ipset n set5 hash:ip family inet6 netmask 64
ipset a set5 2001:2001:2001::
ipset a set5 2001:2001:abcd::
```

hash:net

- Store different sized netblocks
  - /0 not supported
  - „nomatch" keyword to exclude subnets
  - Speed is proportional to the number of different sized netblocks in the set

```
ipset n set6 hash:net

ipset a set6 192.168.0.0/24

ipset a set6 10.1.0.0/16

ipset a set6 10.1.2.0/24 nomatch
```

Hash method and port

- Hash method can store data doubles, triples with „port” kind of sub-data:
- Means protocol and port number together
  - TCP, UDP, SCTP, UDPLite, ICMP, ICMPv6
  - Default is TCP
  - For ICMP and ICMPv6: type/code instead of port number

Hash method: single, double, triple

- hash:ip
- hash:ip,mark
- hash:ip,port
- hash:ip,port,ip
- hash:ip,port,net

- hash:mac

- hash:net
- hash:net,net
- hash:net,port
- hash:net,port,net

hash:ip,port example

- The public services available for everyone

```
ipset n services hash:ip,port
ipset a services 192.168.1.1,icmp:ping
ipset a services 192.168.1.1,udp:53
ipset a services 192.168.1.1,tcp:53
ipset a services 192.168.1.4,25
ipset a services 192.168.1.4,587
```

hash:net,iface

- Special type to store netblock, interface name pairs
  - /0 supported

```
ipset n zones hash:net,iface

ipset a zones 192.168.0.0/16,tenant1

ipset a zones 192.168.1.0/24,tenant2

ipset a zones 0/0,wan
```

List method

- list:set
  - Simple linked list to store sets in sets
  - First match win

```
ipset n sets list:set

ipset a sets set1

ipset a sets set2
```

Timeout extension

- Elements times out automatically
  - Garbage collector
  - Create with the timeout keyword and default value

```
# ipset create test hash:ip timeout 600
```

  - Add elements with specific timeout value

```
# ipset add test 10.0.0.1 timeout 1200
# ipset add test 10.0.0.2 timeout 0
# ipset add test 10.0.0.3
```

Comment extension

- Elements may have comments, max 255 chars
  - Create the set with the comment keyword

```
# ipset create test hash:net comment
```

  - Add elements with the comment value

```
# ipset add test 10.0.0.0/8 \
         comment ”Private A block”
# ipset add test 192.168.0.0/16 \
         comment ”Private B block”
```

Counters extension

- Elements have counters which are updated at every match in the kernel
  - Create the set with the counters keyword

```
# ipset create test hash:ip,port counters
```

  - Add elements with predefined counter values

```
# ipset add test 10.0.0.1:80 \
         packets 8 bytes 1024
```

Skbinfo extension

- Meta informations can be stored and attached to the matching packets
  - skbmark: mark value or mark/mask
  - skbprio: tc class in major:minor format
  - skbqueue: hardware queue number

```
# ipset create test hash:net skbinfo
# ipset add test 10.0.0.0/24 \
    skbmark 0x1 skbprio 1:10 skbqueue 10
```

Ipset and iptables

- Iptables has no idea what kind of set we use
  - Name
  - What direction of a given parameter should be fetched from the packet when constructing the element to lookup
  - The direction parameters must be at least as many as the dimension of the set

```
ipset n services hash:ip,port
iptables -A FORWARD -m set \
    --match-set services dst,dst -j accept
```

ipset and iptables cont.

- The additional direction parameters are ignored

```
ipset n public-services hash:ip,port
ipset n restricted-services hash:ip,port,ip
ipset n services list:set
ipset a services restricted-services
ipset a services public-services
iptables -A FORWARD -m set \
    --match-set services dst,dst,src -j accept
```

SET target

- We can dinamically add/delete elements to sets
- Ideal to block scanners, with timeout combined

```
ipset n scanners hash:ip timeout 1800

iptables -N deny

iptables -A deny -j SET --add-set scanners src

iptables -A deny -j NFLOG --nflog-prefix...

iptables -A deny -j DROP
```

Swap sets

- Atomic operation from iptables point of view

```
ipset n hash:ip main

iptables -A FORWARD -m set --match-set main..

ipset n hash:ip main-tmp

…

ipset swap main main-tmp

ipset destroy main-tmp
```

Ipset and sets in nftables I.

- ipset
  - IPv4 address
  - IPv6 address
  - IPv4 netblock
  - IPv6 netblock
  - MAC address
  - Protocol, port
  - Mark
  - Interface name
  - Set names
  - Fixed combinations

- nftables
  - IPv4 address
  - IPv6 address
  - MAC address (ether)
  - Protocol, port
  - Mark
  - Arbitrary combinations

Ipset and sets in nftables II.

- ipset
  - Named sets
  - Extensions
    - Timeout
    - Comment
    - Counters
    - skbinfo

- nftables
  - Named sets
  - Anonymous sets
    - Timeout
    - Comment
  - Maps
  - Dictionaries

Ipset and sets in nftables III.

- ipset
  - Bitmap
  - Hash
  - List

  - Hash:
    - Arrays
    - Grow only

- nftables
  - Hash

  - Hash:
    - Linked lists
    - Grow-shrink

Performance, iptables test

- Jesper Dangaard Brouer

```
# Simple drop in raw table, single match rule
iptables -t raw -N simple
iptables -t raw -I simple -s 198.18.0.0/15 -j DROP
iptables -t raw -I PREROUTING -j simple
```

Performance, ipset test

```
# Dropping via ipset, 65k IP addresses
echo ″create test hash:ip hashsize 65536″ > test.set
for x in `seq 0 255`; do
  for y in `seq 0 255; do
    echo ″add test 192.168.$x.$y″ >> test.set
done; done
ipset restore < test.set
iptables -t raw -N net198
iptables -t raw -I net198 \
    -m set --match-set test src  -j DROP
iptables -t raw -I PREROUTING -j net198
```

Performance results

Generator sending	12.2Mpps
Iptables, single matching IP rule	11.3Mpps
Single matching ipset rule, 65k elements, before v 6.24	8.0Mpps
Single matching ipset rule, 65k elements, with v 6.24	11.3Mpps

Thank you!

http://ipset.netfilter.org