ipset next

József Kadlecsik
<kadlec@blackhole.kfki.hu>
KFKI RMKI

Content

- Problems with present ipset code
- Background questions
- New ipset: kernel part
- New ipset: userspace
- Application on top of ipset

Present ipset

- It's working, so why rewrite?

- Rigid userspace-kernel communication protocol
  - getsockopt/setsockopt is ok
- Missing IPv6 support
- “Strange” restrictions for some set types
- Mistakes in the userspace syntax

Smooth upgrade path

- Netlink over setsockopt/getsockopt :-)
- Goal was: exactly the same interface as netlink (drop in)
- Dumping

Hash functions I.

- Current kernel hash function: jhash2
- New Jenkins hash: jhash3
- Compare:
  - jenkins2
  - jenkins3 by Bob Jenkins:
    http://www.burtleburtle.net/bob/c/lookup3.c
  - murmur2 by Austin Appleby:
    http://sites.google.com/site/murmurhash/
  - superfasthash by Paul Hsieh :
    http://www.azillionmonkeys.com/qed/hash.html

Hash functions II.

- Patrick Schaaf's cttest program, extended:
  - Statistical analysis of the hash functions:
    http://www.kfki.hu/~kadlec/sw/netfilter/ct2/
- http://www.kfki.hu/~kadlec/sw/netfilter/ct3/

Hashing methods I.

- Rusty on hashing methods, assuming 64-bit
  - http://rusty.ozlabs.org/?p=89
  - http://rusty.ozlabs.org/?p=94
- Ipset is different:
  - 32/64-bit machines
  - Small data to store
  - Optimize both for memory and speed

Hashing methods II.

- Collision limit: 12 elements
  - Single-linked list
  - Single-linked list + doubling
  - Flat hash: search by hashing again + doubling
  - Flat hash: linear search + doubling
  - Link four-element blocks + doubling
- http://www.kfki.hu/~kadlec/sw/netfilter/hash/

Ipset protocol over netlink I.

- Message type is the command code: ADD, DEL
- Mandatory attribute: protocol version
- Additional command-specific attributes
- Two containers (nested attributes) for grouping sub-attributes or multiple elements
- Error handling

Ipset protocol over netlink II.

- A set type is identified by
  - Typename
  - Family: INET, INET6, UNSPEC (both)
  - Revision
- A set is identified by
  - Setname
  - Typename
  - Family

Ipset protocol over netlink III.

- IP addresss attribute: simple attribute, not nested
  - We **know** the family
  - Spares memory

Ipset protocol over netlink IV.

- Netlink dump is too rigid: currently no way to initialize dumping
- How to dump then (list/save) a given set only?
  - Netlink patch required

Locking: set types

- Simple linked list of set types
- Register and unregister
  - Serialized by a mutex
  - list_add|del_rcu
- Lookup
  - RCU read-locking

Locking: sets I.

- Fixed array of set pointers
- External set references store the index in the array
  - Referenced sets are protected by a counter
  - Makes swapping easy

Locking: sets II.

- No locking
- Create, destroy a set, rename:
  - Userspace commands only
  - Serialized by the nfnl mutex of nfnetlink
- Swap two sets
  - Userspace command only
  - Serialized by the nfnl mutex of nfnetlink
  - Pointer assignment is atomic

Locking: set content

- Standard rwlock
- Handled by the core
- Could be made more fine-grained
  - Set types which handle content locking
  - For all other set types locking handled by the core

Timeout and gc I.

- Two flavours for all set type
  - Without and with timeout support
- Again, data is small: don't waste memory
  - Normal timer avoided
  - Garbage collection instead
  - Types recognize timed out entries

Timeout and gc II.

- GC: don't run too often and too rarely either
  - Timeout is measured in seconds: run at every timeout/3 seconds, but
  - At most at every second
  - At least at every three minutes

Code generation for compiling

- For the hash types:
  - Same hashing method for every type
  - Every hash type exists in four flavours:
    - IPv4 and IPv6
    - Without timeout and with timeout
- Avoid manual code multiplication: code generation
- Current ipset: ugly, macro-based
- Ipset next: nice(r), token-replacement

Ipset kernel – iptables

- set match and SET target
- Iptables error reporting is insufficient
  - E.g. mistyped setnames **must** be reported
- setsockopt/getsockopt kept
  - “Backward” compatibility
  - No additional library dependency

Ipset, userspace I.

- Rewritten from scratch
- Mini ipset library
  - Intermediate data handling
  - Parsing, printing
  - Type handling and cache
  - Interface to (kernel) communication method
  - Communication session handling
  - Depends on libmnl

Ipset, userspace II.

- Ipset itself:
  - Type definitions
  - User interface
  - Kernel error decoding

Unified syntax

- Set elem: part0[,part1[,part2]]
  - 192.168.1.1,tcp:80,10.10.10.10
- Iptables match/target dir option: dir[,dir[,dir]]
  - dst,dst,src
- Type: method:kind0[,kind1[,kind2]]
  - hash:ip,port,ip
- Backward compatiblity

Command syntax

- create, add, del, test, ...
  - No need for two dash: --create, --add, ...
- Abbreviation, one-letter shortcuts
- Similar to **ip**
- Backward compatibility

New generic option

- create, add, del, restore:
  - `--exist, -!`

Timeout

- Every set type supports it:
  - Option, not part of the element

```
create test hash:ip timeout 10
add test 10.0.0.1 timeout 0
```

bitmap:ip, bitmap:port types

- No change

bitmap:ip,mac type

- Flag matchunset removed
- Entry can be added to the set without MAC
  - MAC filled out at the first matching
  - Timer starts when IP and MAC pair is complete

hash:ip type

- Both IPv4 and IPv6 supported

hash:net type

- Both IPv4 and IPv6 supported
- Both networks and host addresses can be stored
- Linear search in the list of different prefixes
- No builtin overlap checking

hash:ip,port, hash:ip,port,ip and hash:ip,port,net

- Both IPv4 and IPv6 supported
- Actually port **and** protocol
- Supported protocols:
  - TCP, UDP
  - ICMP, ICMPv6
  - Anything else with zero “port”
- Limitation for the IP address from a /16 block removed

hash:net,port

- New set type
- Arbitrary network or host IP address
- Similar to hash:ip,port

iptree and iptreemap types

- Not implemented, but replaced with hash:ip
- Backward compatibility

list:set type

- No change

Testsuite

- Tests for all set types
  - Without and with timeout
  - IPv4 and IPv6
- Tests for the match and target
  - IPv4 and IPv6

Application: essence I.

- Old tool in Perl, rewritten for ipset
- The essence of a firewall:
  - Raw table:
    - Banned hosts and networks
    - IP spoofing protection
  - Filter table
    - Policy

Application: essence II.

- Simple “keyword = value” syntax
- General settings
  - Logging, set parameters, etc
- Zone rules
  - Spoof protection
- Policies
  - Rules for hosts, networks to play the role of servers, clients

Essence: zone

```
zone = intranet
    interface = eth2
    address = 10.10.0.0/16, 192.168.1.0/24
zone = dmz
    interface = eth1
    address = 192.168.2.0/24
zone = internet
    interface = eth0
    address = 10.10.0.0/24, 0/0
```

Essence: policy

```
policy = servername
    ip = 10.10.10.10
    service = http, https
    service = ssh
        allow = 10.12.0.0/24
    client = ping, http
    client = ssh
        deny = 10.100.0.1, 10.200.0.0/24
```

Todo list

- New protocol and netlink instead of sockopt ✓
- IPv6 support ✓
- Remove (hash type) limitations ✓
- Improved userspace syntax ✓
- Submission for kernel inclusion :-)

Thanks!