nftables, one year later

Éric Leblond

Stamus Networks

September 25, 2014

Éric Leblond

co-founder of Stamus Networks

- Company providing network probe based on Suricata
- Focusing on bringing you the best of Suricata IDS technology

Netfilter Coreteam member

- Work on kernel-userspace interaction
- Kernel hacking
- Ulogd2 maintainer
- Port of Openoffice firewall to Libreoffice

Nftables

A new filtering system

- Replace iptables and the filtering infrastructure
- No changes in
  - Hooks
  - Connection tracking
  - Helpers

Nftables

A new filtering system

- Replace iptables and the filtering infrastructure
- No changes in
  - Hooks
  - Connection tracking
  - Helpers

A new language

- Based on a grammar
- Accessible from a library

Nftables

A new filtering system

- Replace iptables and the filtering infrastructure
- No changes in
  - Hooks
  - Connection tracking
  - Helpers

A new language

- Based on a grammar
- Accessible from a library

Netlink based communication

- Atomic modification
- Notification system

A filtering based on a pseudo-state machine

Inspired by BPF

- 4 registers
- 1 verdict
- A extensive instructions set

A filtering based on a pseudo-state machine

Inspired by BPF

- 4 registers
- 1 verdict
- A extensive instructions set

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)
```

A filtering based on a pseudo-state machine

Inspired by BPF

- 4 registers
- 1 verdict
- A extensive instructions set

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)
```

Easy creation of new matches

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

Architecture

Kernel

- Tables: declared by user and attached to hook
- User interface: nfnetlink socket
  - ADD
  - DELETE
  - DUMP

Userspace

- libmnl: low level netlink interaction
- libnftnl: library handling low-level interaction with nftables Netlink's API
- nftables: command line utility to maintain ruleset

Dynamic chain loading

Chain are created on-demand

- Chain are created via a specific netlink message
- Non-user chain are:
  - Of a specific type
  - Bound to a given hook

Current chain type

- filter: filtering table
- route: old mangle table
- nat: network address translation table

From userspace syntax to kernel

Converting user input

- Operation is made via a netlink message
- The userspace syntax must be converted
  - From a text message following a grammar
  - To a binary Netlink message

Linearize

- Tokenisation
- Parsing
- Evaluation
- Linearization

From kernel to userspace syntax

Kernel send netlink message

- It must be converted back to text

Conversion

- Delinearization
- Postprocessing
- Textify

Example

```
ip filter output 8 7
  [ payload load 4b @ network header + 16 => reg 1 ]
  [ bitwise reg 1 = (reg=1 & 0x00ffffff ) ^ 0x00000000 ]
  [ cmp eq reg 1 0x00500fd9 ]
  [ counter pkts 7 bytes 588 ]
```

is translated to:

```
ip daddr 217.15.80.0/24 counter packets 7 bytes 588 # handle 8
```

Simplified kernel code

A limited in-kernel size

- A limited set of operators and instructions
- A state machine
- No code dedicated to each match
  - One match on address use same code as a match on port
  - New matchs are possible without kernel modification

LOC count

- 50000 LOC in userspace
- only 7000 LOC in kernel-space

Less kernel update

Pseudo state machine instruction

- Current instructions cover need found in previous 10 years
- New instruction require very limited code

Development in userspace

- A new match will not need a new kernel
- ICMPv6 implementation is a single userspace patch

Set handling

Interests of sets

- One single rule evaluation
- Simple and readable ruleset
- Evolution handling

Set handling

Interests of sets

- One single rule evaluation
- Simple and readable ruleset
- Evolution handling

Anonymous set

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

Set handling

Interests of sets

- One single rule evaluation
- Simple and readable ruleset
- Evolution handling

Anonymous set

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

Named set

```
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

Principle and interest

- Associative mapping linking two notions
- A match on the key trigger the use of the value
- Using addresses, interfaces, verdicts

Mapping

Principle and interest

- Associative mapping linking two notions
- A match on the key trigger the use of the value
- Using addresses, interfaces, verdicts

Examples

- Anonymous mapping:

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

- Named mapping:

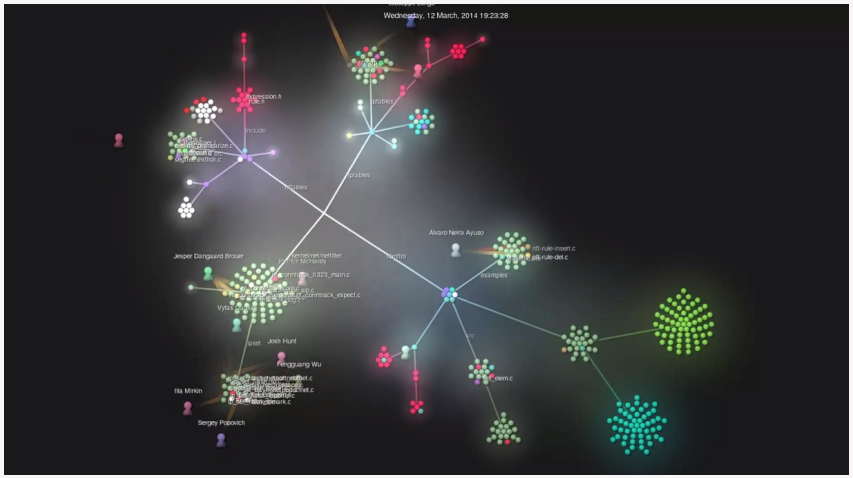
```
# nft -i
nft> add map filter verdict_map { type ipv4_address => verdict; }
nft> add element filter verdict_map { 1.2.3.5 => drop}
nft> add rule filter output ip daddr vmap @verdict_map
```

Usage example

```
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, https} log accept
 counter log drop
}
chain logmetender {
 log limit 10/minute accept
}
chain logmetrue {
 counter log accept
}
}
```

Sexy trailer

The video



`http://youtu.be/fUTgQw75ikA`

Video generation

- Video generated with gource
- Various git history have been merged
- File path has been prefixed with project name

Documentation

Man page

- Complete description of nft
- Man page style:
  - Lot of things
  - Concise description
- A few things missing

A wiki

- Only a howto for now
- `http://wiki.nftables.org/`
- Still incomplete but a good documentation
- Want to contribute: Ask on Netfilter mailing list to get an account

IPv4 and IPv6 filtering

Don't mix the old and the new

- Tables are defined relatively to a IP space
- Must declare a table
  - for each protocol
  - for each chain/hook

Basic filtering chains

```
table filter {
    chain input { type filter hook input priority 0; }
    chain forward { type filter hook forward priority 0; }
    chain output { type filter hook output priority 0; }
}
table ip6 filter {
    chain input { type filter hook input priority 0; }
    chain forward { type filter hook forward priority 0; }
    chain output { type filter hook output priority 0; }
}
```

Handling of IPv4 and IPv6

Inet filtering

Kernel side

- Introduce a new NFPROTO_INET family
- Realize dispatch later based on the effective family
- Activate IPv4 and IPv6 features when needed

Example

```
table inet filter {
   chain input {
    type filter hook input priority 0;
    ct state established,related accept
    iif lo accept
    ct state new iif != lo tcp dport {ssh, 2200} \
        tcp flags == syn counter \
        log prefix "SSH attempt" group 1 \
        accept
    ip saddr 192.168.0.0/24 tcp dport { 9300, 3142} counter accept
    ip6 saddr 2a03:2880:2110:df07:face:b00c:0:1 drop
  }
}
```

Result: easy handling of IPv4 and IPv6

Atomic ruleset update (1/2)

Bring transaction to nftables

- Update ruleset at once
  - Need transaction support
  - All pass or abort
- Need to handle table and rules

Problem of set

- They exists at table level
- They are used at chain level

Atomic ruleset update (2/2)

Atomic ruleset update

- atomically commit a set of rule-set updates incrementally
- based on a generation counter/mask
  - 00 active in the present, will be active in the next generation.
  - 01 active in the present, needs to zero its future, it becomes 00.
  - 10 inactive in the present, delete now.

Batch method

- Method
  - Start transaction
  - Send modifications mixing set and ruleset update
  - Commit transaction
- Interest
  - Limit the number of netlink messages

Dynamic set choice (1/2)

Ipset usage

- Choose set type
- Among the possible choices

The set subsystem

- Various set types are available
  - hash
  - rbtree
- No selector exists

Dynamic set choice (2/2)

Constraint based selection

- Select set based on user constraint
- Memory usage
- Lookup complexity

Syntax

```
nft add set filter set1 { type ipv4_addr ; size 1024 ; }
nft add set filter set1 { type ipv4_addr ; policy memory ; }
nft add set filter set1 { type ipv4_addr ; policy performance ; }
```

Status

- Kernel space is implemented
- Userspace is not yet commited

Warning: NSFC

THE FOLLOWING SLIDE CONTAINS IMAGES THAT MAY HURT THE SENSITIVITY OF SOME CATS.

The young guard

Guiseppe Longo

Arturo Borrero Gonzales
Google Summer of Code

Alvaro Neira Ayuso

Ana Rey
Outreach Program
for Women

Ana Rey: nftables test system

Regression test

- Test nft command and check result
- Most features are tested
- Sponsored by OPW
- Already led to fixes

Example

```
any/queue.t: OK
any/ct.t: WARNING: line: 59: 'nft add rule -nnn ip test-ip4 \
        output ct expiration 30': \
        'ct expiration 30' mismatches 'ct expiration "30s"'
any/ct.t: WARNING: line: 61: 'nft add rule -nnn ip test-ip4 \
        output ct expiration != 233': \
        'ct expiration != 233' mismatches 'ct expiration != "3m53s"'
```

Arturo Borrero: Nftsync (1/2)

Principle

- Distribute ruleset across the network
- Support master/slave
- Deploy ruleset for non gateway systems

Implementation

- Use notification system
- Collect update and distribute them

Nftsync (2/2)

Current state

- Bootstrapped during summer
- Basic mode working
- No encryption yet

Get it, try it, hack it

`http://git.netfilter.org/nft-sync/`

Guiseppe Longo: ebtables compat layer

Provide tools compatibility

- Use old tools with new nftables framework
- Convert old command lines to new internal syntax

Multi layer compatibility

- Bridge level: ebtables
- IP level: iptables

Libnftables

- High level library for third party software
  - Network manager
  - Firewall management interfaces
- It will be based on nftables
  - Using same command line
  - Providing transaction feature

Complete import/export

Exporting ruleset

- Can currently be done via a single nft command
- XML and JSON format

Importing ruleset

- No single command to restore
- `nft -f` is not enough
- `nft import` is needed

Unification with existing BPF

- No real difference
- Different keywords related to Netfilter
  - ct
  - meta
- May be possible to merge

Conclusion

A huge evolution

- Solving iptables problem
- An answer to new usages
  - Set handling
  - Complex matches
  - IPv4 and IPv6 in one table

Already usable

- Main features are here
- Compatibility can be used

Questions ?

Do you have questions ?

Thanks to

- Netfilter team
- Google for GSoC 2014
- Outreach Program for Women

More information

- Netfilter : `http://www.netfilter.org`
- Nftables wiki: `http://wiki.nftables.org`

Contact me

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