nftables, far more than %s/ip/nf/g

Éric Leblond

Nefilter Coreteam

September 24, 2013
1. Introduction
2. Netfilter in 2013
3. Iptables limitations
4. Nftables, an Iptables replacement
5. Advantages of the approach
6. An updated user experience
7. Conclusion
Éric Leblond

Hacker and contractor
- Independant Open Source and Security consultant
- Started and developed NuFW, the authenticating firewall
- Core developer of Suricata IDS/IPS

Netfilter Coreteam member
- Work on kernel-userspace interaction
- Kernel hacking
- ulogd2 maintainer
- Port of Openoffice firewall to Libreoffice
ipchains (1997)
- Linux 2.2 firewalling
- stateless
- Developed by Paul ’Rusty’ Russel

iptables (2000)
- Linux 2.4 firewalling
- Stateful tracking and full NAT support
- in-extremis IPv6 support

Netfilter project
- ’Rusty’ Russel developed iptables and funded Netfilter project
- Netfilter coreteam was created to consolidate the community
Features

Filtering and logging

- Filtering
  - on protocol fields
  - on internal state
- Packet mangling
  - Change TOS
  - Change TTL
  - Set mark

Connection tracking

- Stateful filtering
- Helper to support protocol like FTP

Network Address Translation

- Destination Network Address Translation
- Source Network Address Translation
Hooks

- Hooks at different points of network stack
- Verdict can be issued and skb can be modified
- To each hook correspond at least table
- Different families
  - filter
  - raw
  - nat
  - mangle
- Loading a module create the table

Connection tracking tasks

- Maintain a hash table with known flows
- Detect dynamic connection opening for some protocols
Major components

Netfilter filtering
- In charge of accepting, blocking, transforming packets
- Configured by ioctl

Connection tracking
- Analyse traffic and maintain flow table
- Cost in term of performance
- Increase security

iptables
- Configuration tools
- Update ruleset inside kernel
The nfnetlink (r)evolution

Nfnetlink
- First major evolution of Netfilter (Linux 2.6.14, 2005)
- Netfilter dedicated configuration and message passing mechanism

New interactions
- NFLOG: enhanced logging system
- NFQUEUE: improved userspace decision system
- NFCT: get information and update connection tracking entries

Based on Netlink
- datagram-oriented messaging system
- passing messages from kernel to user-space and vice-versa
Netlink

Header format

```
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|   |   |   |   |   |   |   |   |   |   | 32 bits message length |   |   |   | 16 bits type |   |   | 16 bits flags |   |   |   |   | 32 bits sequence number |   |   |   |   |   | 32 bits PID |
```

Figure 2. Layout of a Netlink message header

Payload format

```
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|   |   |   |   |   |   |   |   |   |   | 16 bits type |   |   |   | 16 bits length |   |   |   |   | 32 bits value (eg. 32 bits) |   |   |   |   |   | 16 bits type |   |   |   |   | 16 bits length |   |   |   |   |   | 64-bit value value |
```

Figure 3. An example of a hypothetical Netlink payload in TLV format
Components created following 2.6.14

**conntrack-tools**
- **conntrackd**
  - connection tracking replication daemon
  - provide high availability
  - developed by Pablo Neira Ayuso
- **conntrack**: command line tool to update and query connection tracking

**ulogd2**
- logging daemon
- handle packets and connections logging
Latest changes

**ipset**
- Efficient set handling
- Address list or more complex set
- Reach vanilla kernel in 2011 (Linux 2.6.39)

**nfacct**
- Efficient accounting system
- Appeared in 2012
## Kernel code

### How much code
- 70000 LOC reside in kernelspace
- around 50000 LOC in user-space

### Iptables extensions
- 111 iptables extensions.
- Various tasks:
  - tcp
  - cluster
  - bpf
  - statistic
Adding a rule

The problem

- Atomic replacement of ruleset
  - Sent from kernel to userspace
  - Modified and sent back by userspace
- Huge performance impact

Éric Leblond (Nefilter Coreteam)
Dynamic ruleset

Network gets dynamic

- Firewall can’t be static anymore
  - Cloud
  - IP reputation
- Combinatory explosion: one rule per-server and protocol

Set handling

- Set handling is made via ipset
- Efficient but not as integrated as possible
Code duplication

**Different filtering family**
- Netfilter classic filtering
- Brigde filtering
- Arp filtering
- IPv4 and IPv6

**Matches and target**
- Similar code in numerous Netfilter module
- Nothing is shared
- Manual parsing
Problem due to binary blob usage

 ABI breakage
 - Binary exchange between userspace and kernel
 - No modification possible without touching kernel

 Trusting userspace
 - Kernel is parsing a binary blob
 - Possible to break the internal parser
# Integration via exec

## Frontend and iptables
- No officially available library
- Frontend fork iptables command

## libiptables
- Available inside iptables sources
- Not a public library
- API and ABI breakage are not checked during version upgrade
Lack of flexible table and chains configurations

Module loading is the key
- Chains are created when module init
- Induce a performance cost even without rules

No configuration is possible
- Chains are hardcoded
- FORWARD is created on a server
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
## History

### Introduced in 2008
- Developed and presented by Patrick McHardy at NFWS2008
- Presentation took 3 hours
- Alpha stage in 2008

### Development did stop
- Patrick McHardy did not finish the code alone
- Nobody did join the effort
Video Interlude

The video

http://www.youtube.com/watch?v=DQp1AI1p3f8

Video generation

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

**Should have "Release often release early"?**
- Started by Patrick McHardy only
- Almost complete work presented during NFWS 2008
- Complex to enter the project

**Too early?**
- No user were demanding for that explicitly
- Ipset was available and fixing the set issue
- Solution for dynamic handling was sufficient
Development restarted in 2012

**Funding by Sophos/Astaro**
- Pablo Neira Ayuso get funded by Astaro
- Work restart in 2012

**Gaining momentum**
- Tomasz Bursztyka joined the development team
  - Work on Connman
  - Lack of libs was painful to him
  - Start to hack on nftables
- Google summer of code
  - 3 students
  - Some good results
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
- libnftables: 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
- Deliniearization
- 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
```
### 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.

---

### xtables compatibility

- Possible to use old extensions
- Necessary to provide backward compatibility
### Event based notification
- Each rule update trigger an event
- Event is sent to userspace via nfnetlink

### Userspace usage
- Implemented in libnftables
- Program can update his view on the ruleset without dump
<table>
<thead>
<tr>
<th>A limited in-kernel size</th>
</tr>
</thead>
<tbody>
<tr>
<td>- A limited set of operators and instructions</td>
</tr>
<tr>
<td>- A state machine</td>
</tr>
<tr>
<td>- No code dedicated to each match</td>
</tr>
<tr>
<td>- One match on address use same code as a match on port</td>
</tr>
<tr>
<td>- New matchs are possible without kernel modification</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LOC count</th>
</tr>
</thead>
<tbody>
<tr>
<td>- 50000 LOC in userspace</td>
</tr>
<tr>
<td>- only 7000 LOC in kernel-space</td>
</tr>
</tbody>
</table>
### 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
Example of ICMPv6

```
include/datatype.h |  2 ++
include/payload.h  | 14 ++++++++---
src/parser.y       | 33 +++++++++++++++++++++++++++++++++--
src/payload.c      | 59 ++++++++++++++++++++++++++++++++++++++++
src/scanner.l      |  4 ++
5 files changed, 109 insertions(+), 3 deletions(-)
```
Example of ICMPv6

```c
static const struct datatype icmp6_type_type = {
    .type = TYPE_ICMP6_TYPE,
    .name = "icmpv6_type",
    .desc = "ICMPv6 type",
    .byteorder = BYTEORDER_BIG_ENDIAN,
    .size = BITS_PER_BYTE,
    .basetype = &integer_type,
    .sym_tbl = &icmp6_type_tbl,
    .sym_tbl = &icmp6_type_tbl,
};

#define ICMP6HDR_FIELD(__name, __member) \
    HDR_FIELD(__name, struct icmp6_hdr, __member)
#define ICMP6HDR_TYPE(__name, __type, __member) \
    HDR_TYPE(__name, __type, struct icmp6_hdr, __member)

const struct payload_desc payload_icmp6 = {
    .name = "icmpv6",
    .base = PAYLOAD_BASE_TRANSPORT_HDR,
    .templates = {
        [ICMP6HDR_TYPE] = ICMP6HDR_TYPE("type", &icmp6_type_type, icmp6_type),
        [ICMP6HDR_CODE] = ICMP6HDR_FIELD("code", icmp6_code),
        [ICMP6HDR_CHECKSUM] = ICMP6HDR_FIELD("checksum", icmp6_cksum),
        [ICMP6HDR_PPTR] = ICMP6HDR_FIELD("parameter−problem", icmp6_pptr),
        [ICMP6HDR_MTU] = ICMP6HDR_FIELD("packet−too−big", icmp6_mtu),
        [ICMP6HDR_ID] = ICMP6HDR_FIELD("id", icmp6_id),
        [ICMP6HDR_SEQ] = ICMP6HDR_FIELD("sequence", icmp6_seq),
        [ICMP6HDR_MAXDELAY] = ICMP6HDR_FIELD("max−delay", icmp6_maxdelay),
    },
};
```
Basic utilisation

File mode

nft -f ipv4-filter

Command line mode

```
nft add rule ip filter input tcp dport 80 drop
nft list table filter -a
nft delete rule filter output handle 10
```

CLI mode

```
# nft -i
nft> list table
<cli>:1:12–12: Error: syntax error, unexpected end of file, expecting string
list table
^
nft> list table filter
table filter {
  chain input {
    ip saddr 1.2.3.4 counter packets 8 bytes 273
  }
```

Éric Leblond (Nefilter Coreteam) nftables, far more than %s/ip/nf/g September 24, 2013 41 / 48
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

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

```plaintext
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, \n    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 log accept
    ip daddr @web_servers tcp dport https accept
    counter log drop
}
chain logmetender { 
    log limit 10/minute accept
}
chain logmetrue { 
    counter log accept
}
```
Transition and evolution

A complete iptables compatibility
- iptables-nftables
  - Binary compatible with iptables
  - Using nftables framework
- Same kernel can be used with two systems
- A progressive update

A high level library
- To be used by frontends
- Or by network manager systems
Conclusion

A huge evolution

- Solving iptables problem
- An answer to new usages
  - Set handling
  - Complex matches

Availability for end 2013, beginning 2014

- Finalizing iptables compatibility
- High level library
- Debug and some functionalities
Questions?

Do you have questions?

Thanks to
- Netfilter team
- Astaro/Sophos for financing the development
- Google for GSoC 2013

More information
- Netfilter: http://www.netfilter.org
- Nftables quick & dirty: https://t.co/cM4zogob8t

Contact me
- Mail: eric@regit.org
- Twitter: @Regiteric

Éric Leblond (Netfilter Coreteam) nftables, far more than %s/ip/nf/g September 24, 2013