The Next Generation Firewall for Red Hat Enterprise Linux 7 RC

Thomas Graf
Red Hat
Agenda

- FirewallID – Firewall Management as a Service
- Kernel – New Filtering Capabilities
- Nftables – A Look Ahead
FirewallID
Firewall Management as a Service
Existing Packet Filtering Architecture

- **User Land**
  - iptables
  - ip6tables
  - ebtables

- **Kernel**
  - Netfilter
    - IPv4
    - IPv6
    - Bridge

Protocol dependent packet filter and utilities
Firewall Management as a Service

Application Direct Access

Reports

User Interface Graphical CLI

FirewallD

IPv4

IPv6

Bridge
FirewallD – Features

- Unified firewall management as a service
- No service disruptions during rule updates
- Firewall zones
- D-Bus interface
- Runtime & permanent configuration
- Graphical & console user interface
- Direct access
FirewallD – Policy Abstraction

Policy → Zone
FirewallD – Zone Policy

- Default policy
- Enabled services
- Rich rules
- Masquerading
- Port forwarding
- ICMP filter
FirewallD – Graphical User Interface

A firewall zone defines the level of trust for network connections, interfaces and source addresses bound to the zone. The zone combines services, ports, protocols, masquerading, port/packet forwarding, icmp filters and rich rules. The zone can be bound to interfaces and source addresses.

Base Zone Settings

Please configure base zone settings:

- **Name:** internal
- **Version:**
- **Short:** Internal
- **Description:** For use on internal networks. You mostly trust the other computers on the networks to not harm your computer. Only selected incoming
- **Target:** Default Target

Default Zone: public Lockdown: disabled Panic Mode: disabled
FirewallD – Command Line Interface

• Add interface “eth0” to zone “public” permanently:

```
# firewall-cmd --permanent --zone=internal --add-interface=eth0
```

• List enabled services:

```
# firewall-cmd --zone=public --list-services
```
RHEL7 Netfilter Kernel Changes
### Scaling of Legacy Applications (xt_cpu)

```bash
# iptables -t nat -A PREROUTING -p tcp --dport 80 \
  -m cpu --cpu 0 -j REDIRECT --to-port 8080

# iptables -t nat -A PREROUTING -p tcp --dport 80 \
  -m cpu --cpu 1 -j REDIRECT --to-port 8081
```
Connection Tracking target (xt_CT)

- Disable connection tracking per packet
  
  ```
  # iptables -t raw -A PREROUTING -p udp --dport 53 -j CT --notrack
  # iptables -t raw -A OUTPUT -p udp --sport 53 -j CT --notrack
  ```

- Define multiple zones to allow for overlapping identities
  
  ```
  # iptables -t raw -A PREROUTING -i eth0 -j CT --zone 10
  ```

- Replacement for RHEL6 NOTRACK target
Connection Tracking target (xt_CT)

• Attach timeout to connection tracking entry

```bash
# nfct timeout add my-tcp-policy inet tcp \ 
established 100 close 10 close_wait 10
# iptables -t raw -A PREROUTING -p tcp -j CT --timeout my-tcp-policy
```

• Specify connection tracking helper to use for matched packets
IPv6 Connection Tracking & NAT

- Available targets:
  - SNAT, DNAT, MASQUERADE, NETMAP, REDIRECT
- Available Connection Tracking Helpers:
  - SIP, FTP, Amanda

```bash
# ip6tables -t nat -A POSTROUTING -o eth0 -j SNAT --to 2001:aa::1
```
CT Helpers in User Space

• No need to load kernel module to implement connection tracking for new or custom protocols
• Rapid development
• Avoid complex string matching and mangling in kernel
SYNPROXY (xt_SYNPROXY)

- Protection against SYN flood attacks
- Lightweight proxy for TCP three-way handshake

```
# iptables -t raw -A PREROUTING -p tcp --dport 80 --syn -j NOTRACK
# iptables -A INPUT -p tcp --dport 80 -m state UNTRACKED,INVALID \ -j SYNPROXY --sack-perm --timestamp --mss 1480 --wscale 7 –ecn
```
Extended Accounting (xt_nfacct)

- Kernel based meter providing packet and byte statistics
- Avoids need to perform expensive rule set statistics polling

```
# iptables -I INPUT -p tcp --sport 80 \
   -m nfacct --nfacct-name http-traffic

# nfacct-add http-traffic
# nfacct-get
  http-traffic = { pkts = 000000008231, bytes = 000044932916 }
```
Connection Labeling (xt_connlabel)

- Label connection tracking entries with rule:

  # iptables -A INPUT -i eth0 -m helper --helper ftp \ 
  -m connlabel --label customer-bulk-traffic --set
  # iptables -A INPUT -i eth0 -p tcp -dport 22 \ 
  -m connlabel --label customer-interactive --set

- ... then match on labels:

  # iptables -A INPUT -m connlabel --label customer-bulk-traffic \ 
  -m connlimit --connlimit-above 2 -j REJECT
Reverse Path Filtering (xt_rpfiltter)

• Matches if a reply to a packet would be sent via the incoming interface

• Drop packets that failed reverse path filtering:

  ```
  # iptables -t raw -A PREROUTING -m rpfilter --invert -j DROP
  ```

• Identical in functionality as net.ipv4.conf.all.rp_filter = 1
Berkley Packet Filter (xt_bpf)

• Match packets based Berkley Packet Filter (BPF) filters

```
# iptables -A OUTPUT -m bpf \
   --bytecode "8,40 0 0 12,21 1 0 [...]
   -j ACCEPT
```

• Use tcpdump to generate the bytecode:

```
# tcpdump -ddd vlan 20 and dst port 22 | tr '\n','
   26,40 0 0 12,21 1 0 33024,21 0 22 37120 [...]
```
New ipset Features

• Automatic range to subnets translations (IPv4 only)

```bash
# ipset new test hash:net
# ipset add test 10.1.0.0-10.3.49.2
```

• Exceptions in sets:

```bash
# ipset new test hash:net
# [...]
# ipset add test 10.2.0.10/32 nomatch
```
IDLETIMER target (xt_IDLETIMER)

• Provide idle time for a rule via sysfs:

```bash
# iptables -A OUTPUT -o eth0 -j IDLETIMER --timeout 5 --label foo
# cat /sys/class/xt_idletimer/timers/foo
4
[...]
# cat /sys/class/xt_idletimer/timers/foo
0
```

• Used to detect idle interfaces and put them in power safe mode
TEE target (xt_TEE)

• Clone & send packet to local machine for logging

```
# iptables -t mangle -A PREROUTING -i eth0 \
  -j TEE --gateway 2001:db8::1
```
NFQUEUE performance optimizations

• Zero copy Netlink to user space

```
# iptables -A INPUT -j NFQUEUE --queue-num 3
```

• CPU Fanout: CPU # selects queue #:

```
# iptables -A INPUT -i eth0 \ 
   -j NFQUEUE --queue-balance 0:31 --queue-cpu-fanout
```
Generic Address Type Filter (xt_addrtype)

• Match type of source and/or destination address:

```bash
# ip6tables -A INPUT -m addrtype --dst-type MULTICAST -j DROP
# ip6tables -A OUTPUT -m addrtype ! --src-type LOCAL -j REJECT
```
nftables (Tech Preview)
A Look Ahead
nftables – State Machine Based Packet Filtering

• New packet filtering subsystem to replace {ip, ip6, arp, eb} tables
• Byte code execution in kernel pseudo state machine
• Unified interface nft to replace protocol aware utilities
nftables – Features

• Incremental updates
• Byte code can be optimized
• Efficient execution and storage of rules
• Fast lookups through performance data structures
• Heavy code reduction in kernel, less protocol awareness
  • No kernel change required to support new protocols
• Improved error handling
nftables – Want to try it out?

- Included in RHEL7.0 Beta kernel (Tech Preview)
- Userspace packages will be included in future minor release
- Fetch them from upstream to get testing?
  - libmnl, libnfnl, nftables
Thank You
Backup
Explicit Connection Notification (xt_ecn)

- Match ECN bits on IPv4/IPv6 and TCP header (RFC3168):

```shell
# iptables -A INPUT -i eth1 -m ecn ! --ecn-tcp-cwr -j REJECT
```
Compat Support

- Run 32bit iptables on 64bit kernel
Match on IPVS properties

• Combine full NAT functionality with IPVS properties:

```
# iptables -t nat -A POSTROUTING \
    -m ipvs --vaddr 192.168.100.30/32 --vport http \
    -j SNAT [...]  
```