

The netfilter/iptables framework in Linux 2.4.x

by

Harald Weite <laforge@gnumonks.org>

Contents

- Introduction
- Netfilter hooks in protocol stacks
- Packet selection based on IP Tables
- The Connection Tracking Subsystem
- The NAT Subsystem based on netfilter + iptables
- Packet filtering using the 'filter' table
- Packet mangling using the 'mangle' table
- Advanced netfilter concepts
- Current development and Future

Introduction

Why did we need netfilter/iptables?

Because ipchains....

- has no infrastructure for passing packets to userspace
- makes transparent proxying extremely difficult
- has interface address dependent Packet filter rules
- has Masquerading implemented as part of packet filtering
- code is too complex and intermixed with core ipv4 stack
- is neither modular nor extensible
- only barely supports one special case of NAT (masquerading)
- has only stateless packet filtering

Introduction

Who's behind netfilter/iptables

- **Paul 'Rusty' Russel**
 - ▷ co-author of iptables in Linux 2.2
 - ▷ was paid by Watchguard for about one Year of development
- **James Morris**
 - ▷ userspace queuing (kernel, library and tools)
 - ▷ REJECT target
- **Marc Boucher**
 - ▷ NAT and packet filtering controlled by one command
 - ▷ Mangle table
- **Harald Welte**
 - ▷ Conntrack+NAT helper infrastructure (newnat)
 - ▷ Userspace packet logging (ULOG)
 - ▷ PPTP and IRC conntrack/NAT helpers
- **Jozsef Kadlec**
 - ▷ TCP window tracking
 - ▷ H.323 conntrack + NAT helper
 - ▷ Continued newnat development
- **Non-core team contributors**
 - ▷ <http://www.netfilter.org/scoreboard/>

Netfilter Hooks

What is netfilter?

- System of callback functions within network stack
- Callback function to be called for every packet traversing certain point (hook) within network stack
- Protocol independent framework
- Hooks in layer 3 stacks (IPv4, IPv6, DECnet, ARP)
- Multiple kernel modules can register with each of the hooks
- Asynchronous packet handling in userspace (ip_queue)

Traditional packet filtering, NAT, ... is implemented on top of this framework

Can be used for other stuff interfacing with the core network stack, like DECnet routing daemon.

Netfilter Hooks

Netfilter Hooks

- Any kernel module may register a callback function at any of the hooks

- The module has to return one of the following constants
 - `NF_ACCEPT` continue traversal as normal
 - `NF_DROP` drop the packet, do not continue
 - `NF_STOLEN` I've taken over the packet do not continue
 - `NF_QUEUE` enqueue packet to userspace
 - `NF_REPEAT` call this hook again

IP tables

Packet selection using IP tables

- The kernel provides generic IP tables support
- Each kernel module may create it's own IP table
- The three major parts of 2.4 firewalling subsystem are implemented using IP tables
 - ▷ Packet filtering table 'filter'
 - ▷ NAT table 'nat'
 - ▷ Packet mangling table 'mangle'
- Can potentially be used for other stuff, i.e. IPsec SPDB

IP Tables

Managing chains and tables

- An IP table consists out of multiple chains
- A chain consists out of a list of rules
- Every single rule in a chain consists out of
 - match[es] (rule executed if all matches true)
 - target (what to do if the rule is matched)

matches and targets can either be builtin or implemented as kernel modules

- The userspace tool iptables is used to control IP tables
 - handles all different kinds of IP tables
 - supports a plugin/shlib interface for target/match specific options

IP Tables

Basic iptables commands

- To build a complete iptables command, we must specify
 - which table to work with
 - which chain in this table to use
 - an operation (insert, add, delete, modify)
 - one or more matches (optional)
 - a target

The syntax is

```
iptables -t table -O operation chain -j target match(es)
```

Example:

```
iptables -t filter -A INPUT -j ACCEPT -p tcp --dport smtp
```

IP Tables

Matches

○ Basic matches

- ▷ -p protocol (tcp/udp/icmp/...)
- ▷ -s source address (ip/mask)
- ▷ -d destination address (ip/mask)
- ▷ -i incoming interface
- ▷ -o outgoing interface

○ Match extensions (examples)

- ▷ tcp/udp TCP/udp source/destination port
- ▷ icmp ICMP code/type
- ▷ ah/esp AH/ESP SPIID match
- ▷ mac source MAC address
- ▷ mark nmark
- ▷ length match on length of packet
- ▷ limit rate limiting (n packets per timeframe)
- ▷ owner owner uid of the socket sending the packet
- ▷ tos TOS field of IP header
- ▷ ttl TTL field of IP header

IP Tables

Targets

- very dependent on the particular table.
- Table specific targets will be discussed later
- Generic Targets, always available
 - ACCEPT accept packet within chain
 - DROP silently drop packet
 - QUEUE enqueue packet to userspace
 - LOG log packet via syslog
 - ULOG log packet via ulogd
 - RETURN return to previous (calling) chain
 - foobar jump to user defined chain

Packet Filtering

Overview

- Implemented as 'filter' table
- Registers with three netfilter hooks
 - NF_IP_LOCAL_IN (packets destined for the local host)
 - NF_IP_FORWARD (packets forwarded by local host)
 - NF_IP_LOCAL_OUT (packets from the local host)

Each of the three hooks has attached one chain (INPUT, FORWARD, OUTPUT)

Every packet passes exactly one of the three chains. Note that this is very different compared to the old 2.2.x ipchains behaviour.

Packet Filtering

Targets available within 'filter' table

- Builtin Targets to be used in filter table
 - ACCEPT accept the packet
 - DROP silently drop the packet
 - QUEUE enqueue packet to userspace
 - RETURN return to previous (calling) chain
 - foobar user defined chain

- Targets implemented as loadable modules
 - REJECT drop the packet but inform sender
 - MIRROR change source/destination IP and resend
 - LOG log via syslog
 - ULOG log via userspace

Connection Tracking Subsystem

- Connection tracking...
 - implemented seperately from NAT
 - enables stateful filtering
 - implementation
 - ▷ hooks into NF_IP_PRE_ROUTING to track packets
 - ▷ hooks into NF_IP_POST_ROUTING and NF_IP_LOCAL_IN to see if packet passed filtering rules
 - ▷ protocol modules (currently TCP/UDP/ICMP)
 - ▷ application helpers currently (FTP,IRC,H.323,talk,SNMP)
 - divides packets in the following four categories
 - ▷ NEW - would establish new connection
 - ▷ ESTABLISHED - part of already established connection
 - ▷ RELATED - is related to established connection
 - ▷ INVALID - (multicast, errors...)
 - does **NOT** filter packets itself
 - can be utilized by iptables using the 'state' match
 - is used by NAT Subsystem

Network Address Translation

Overview

- Previous Linux Kernels only implemented one special case of NAT: Masquerading
- Linux 2.4.x can do any kind of NAT.
- NAT subsystem implemented on top of netfilter, iptables and conntrack
- NAT subsystem registers with all five netfilter hooks
- 'nat' Table registers chains PREROUTING, POSTROUTING and OUTPUT
- Following targets available within 'nat' Table
 - ▷ SNAT changes the packet's source while passing NF_IP_POST_ROUTING
 - ▷ DNAT changes the packet's destination while passing NF_IP_PRE_ROUTING
 - ▷ MASQUERADE is a special case of SNAT
 - ▷ REDIRECT is a special case of DNAT

Network Address Translation

□ Source NAT

○ SNAT Example:

```
iptables -t nat -A POSTROUTING -j SNAT --to-source 1.2.3.4 -s 10.0.0.0/8
```

○ MASQUERADE Example:

```
iptables -t nat -A POSTROUTING -j MASQUERADE -o ppp0
```

□ Destination NAT

○ DNAT example

```
iptables -t nat -A PREROUTING -j DNAT --to-destination 1.2.3.4:8080 -p tcp --dport 80 -i eth1
```

○ REDIRECT example

```
iptables -t nat -A PREROUTING -j REDIRECT --to-port 3128 -i eth1 -p tcp --dport 80
```

Packet Mangling

- Purpose of mangle table
 - packet manipulation except address manipulation

- Integration with netfilter
 - 'mangle' table hooks in all five netfilter hooks
 - priority: after conntrack

- Targets specific to the 'mangle' table:
 - DSCP - manipulate DSCP field
 - IPV4OPTSSTRIP - strip IPv4 options
 - MARK - change the nmark field of the skb
 - TCPMSS - set TCP MSS option
 - TOS - manipulate the TOS bits
 - TTL - set / increase / decrease TTL field

Simple example:

```
iptables -t mangle -A PREROUTING -j MARK --set-mark 10 -p tcp --dport 80
```

Advanced Netfilter concepts

- **Userspace logging**
 - flexible replacement for old syslog-based logging
 - packets to userspace via multicast netlink sockets
 - easy-to-use library (libipulog)
 - plugin-extensible userspace logging daemon (ulogd)
 - Can even be used to directly log into MySQL

- **Queuing**
 - reliable asynchronous packet handling
 - packets to userspace via unicast netlink socket
 - easy-to-use library (libipq)
 - provides Perl bindings
 - experimental queue multiplex daemon (ipqmpd)

Current Development and Future

Netfilter (although it proved very stable) is still work in progress.

- Areas of current development
 - infrastructure for conntrack manipulation from userspace
 - failover of stateful firewalls
 - making iptables layer3 independent (pkttables)
 - new userspace library (libiptables) to hide plugins from apps
 - more matches and targets for advanced functions (pool, hashslot)
 - more conntrack and NAT modules (RPC, SNMP, SMB, ...)
 - better IPv6 support (conntrack, more matches / targets)

Thanks

- Thanks to
 - the BBS people, Z-Netz, FIDO, ...
 - ▷ for heavily increasing my computer usage in 1992
 - KNF
 - ▷ for bringing me in touch with the internet as early as 1995
 - ▷ for providing a playground for technical people
 - ▷ for telling me about the existence of Linux!
 - Alan Cox, Alexey Kuznetsov, David Miller, Andi Kleen
 - ▷ for implementing (one of?) the world's best TCP/IP stacks
 - Paul 'Rusty' Russell
 - ▷ for starting the netfilter/iptables project
 - ▷ for trusting me to maintain it today
 - Linux User Group Nuernberg (ALIGN, LUG-N)
 - ▷ for helping me with my initial Linux problems

Availability of slides / Links

The slides and the an according paper of this presentation are available at

□ <http://www.gnumonks.org/>

The netfilter homepage

□ <http://www.netfilter.org/>