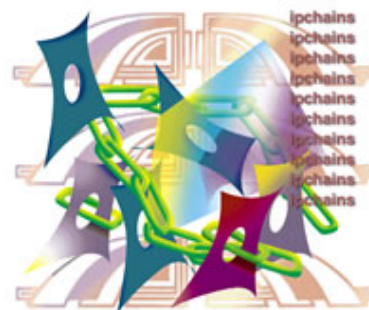


CMP
United Business Media**REGISTER TODAY!**www.sysadminmag.com/events/**Sys Admin.**
the journal for UNIX and Linux
systems administrators**Sys Admin.**
the journal for UNIX and Linux
systems administratorsTry our RSS feed [XML](#)**DEPARTMENTS**Operating Systems
Server Management
Networking
Security
Storage
Databases
Languages
Your Career**SYS ADMIN**Issues
Source Code
Events
Newsletters
Whitepapers
About Us
Contact Us
Editors
Advertise**Current Issue**[Table of contents](#)[Buy this issue.](#)**SUBSCRIBE NOW!**[Sys Admin Magazine](#) > [Archives](#) > [2001](#) > [December 2001](#)[Print-Friendly Version](#)**IPTables/NetFilter — Linux's Next-Generation Stateful Packet Filter***Duncan Napier*

The IPTables/NetFilter application is considered to be the fourth generation of Linux packet filtering implementations. The first generation was Alan Cox's port of BSD UNIX's ipfw to Linux 1.1. Jos Vos and others extended this and added the ipfwadm user tool for manipulating the rules for filtering in the Linux 2.0 kernel. Paul "Rusty" Russell and Michael Neuling made some significant modifications to the 2.2 Linux kernel, and Russell added the user tool ipchains for controlling filtering rules for this kernel. Russell has now implemented a kernel framework called NetFilter.

One of the goals of NetFilter was to provide a single, dedicated packet filter/mangler infrastructure that users and developers could deploy as an add-on built around the Linux kernel. For purposes of this article, packet filtering refers to the redirection of packets (but not modification of packet headers), while mangling refers to packet modification, typically of the source and/or destination IP address. NetFilter was designed to be modular and extensible. IPTables is a module that plugs into the NetFilter framework and allows the user access to kernel filtering/mangling rules and commands. If you are familiar with ipchains, you will notice the similarity between the syntax and format of IPTables and ipchains.

It is also worth noting that NetFilter is outside of the standard Berkeley socket interface and as a result is, at the time of writing, restricted to the Linux OS.

The official NetFilter home page is:

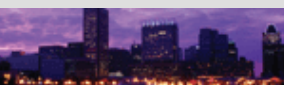
<http://netfilter.samba.org/>

and it provides the latest documentation, information, patches, and releases related to the NetFilter Project. It is critical to stay up to date because IPTables and NetFilter have been undergoing a rapid proliferation of features. Also, if you choose to implement IPTables on a secure production environment, be careful which features you choose to enable, because some features, as noted in the installation, are experimental and have not been fully tested.

Some Features and Enhancements Over ipchains and ipadm

Here are some of the firewalling features of IPTables:

- Stateful Inspection — Perhaps the most talked-about enhancement that NetFilter/

**Sys Admin
Technical
Conference****May 7-8, 2007**

Sheraton Inner Harbor

Baltimore, MD

**Featuring tutorials
by Randal Schwartz
and Hal Pomeranz****Perl, Sendmail,
Linux Security,
Network Incident
Response and
Much More!!!****REGISTER
TODAY!****Sys Admin.**
the journal for UNIX and Linux
systems administrators**CLICK
HERE!**

Newsletter

Subscribe to the
Sys Admin update email
newsletter...

Email Address

First Name

Last Name

CD-ROM

Sys Admin and *The Perl Journal* CD-ROM version 12.0

Version 12.0 delivers every issue of *Sys Admin* from 1992 through 2006 and every issue of *The Perl Journal* from 1996-2002 in one convenient CD-ROM!

[Order now!](#)

IPTables adds to Linux is stateful packet inspection. Stateless firewalls (such as those using Linux ipchains) check and filter each packet individually. Stateless filters, for example, can differentiate between packets that are requesting a connection and those that are already connected in a session by checking whether the SYN flag in the packet header is set (and also whether FIN and ACK flags cleared). A stateless firewall has no recollection of the past history of the connection (or lack of it) and therefore makes filtering choices based on the packet header information presented to it by the packet itself. Stateful firewalls log information related to an entire session, such as source and destination addresses, port numbers, status information from header flags (e.g., SYN, ACK, FIN flags), and TCP sequencing information. This allows stateful systems to make filtering decisions in the context of an entire session rather than that of an individual packet and its header information. For example, if you have clients behind a firewall that make Domain Name Server (DNS) queries to an external DNS server, the client initiates a query by connecting from one of its high-numbered ports to UDP port 53 on the server. The DNS server answers the query from UDP port 53 back to the high-numbered port on the client. On a conventional ipchains firewall, all high-numbered ports above 1023 must be opened to accommodate inbound connections. A stateful firewall can be configured to only accept incoming UDP packets with a source address that matches the destination address of previously outgoing packets. Therefore, the firewall will only accept query responses from DNS servers that match outgoing queries that the firewall has already seen. IPTables/NetFilter stateful inspection can be customized and extended with connection tracker and NAT (network address translation) helper modules.

- **Enhanced Network Address Translation (NAT)** — In addition to Source Network Address Translation (SNAT), which is usually used by two or more machines to share one public IP address using a proxy, IPTables also does Destination NAT (DNAT). DNAT modifies the destination address (as opposed to SNAT, which modifies the source IP address) and allows traffic to be redirected to a proxy. Ipchains did not do port redirection and required the installation of the ipmasqadm utility for such things as forwarding ports into internal hosts. DNAT in IPTables now handles this function. DNAT can also be used as a very basic load-balancing tool to handle traffic flow to a cluster of machines. Additional packet-mangling functions that modify source and destination IP addresses have also been added, including MIRROR, which reverses source and destination addresses for sending packets back to their sources.

- **Enhanced packet inspection** — All six TCP flags can be examined by IPTables, as opposed to just the SYN flag setting in ipchains. This allows a much finer level of control of packets that enter or leave the network. Matches based on a series of source or destination ports can be specified, instead of a single port or single range of ports. This greatly simplifies script maintenance. There is a match extension parameter, which can even filter based on the user id, group id, process id, and session id.

- **MAC address filtering** — Traffic can now be filtered at the MAC (Ethernet hardware or Media Access Control) address level.

- **Enhanced logging** — As of the current release, there are eight levels of logging (debug, info, notice, warning, err, crit, alert, emerg) as well as the ability to prepend error messages with customized strings for unique identification.

- **Rate-Limited matching** — IPTables can limit the rate at which the firewall handles connection requests. This feature can be used not only to protect against Denial-of-Service attacks (such as syn floods) and port scans, but can also limit the rate of logging for repetitive processes.

- **Type of Service (TOS) prioritization** — Traffic can be selectively prioritized into separate queues. The type of service levels currently available are Minimize-Delay, Maximize-Throughput, Maximize-Reliability, Minimize-Cost, and Normal-Service.

Installation and Configuration

IPTables and NetFilter require a Linux kernel version 2.3.15 or greater. A back port to lower versions (2.2.x) of Linux has been suggested by its author, but at the time of writing, this has not been implemented. I have installed and tested NetFilter/IPTables 1.2.1a on a Red Hat Beta 2.4.1 distribution (Wolverine) and a generic 2.4.3 kernel. NetFilter can be compiled into the kernel or run as a loadable module. For purposes of this discussion, I will assume that NetFilter is compiled into the kernel.

The IPTables tool can be downloaded from the Official NetFilter Web site. At the time of writing, given the rapid pace of development of IPTables I recommend using the latest version (in my case the patched version 1.2.1a) containing the latest patches and bug fixes. Following the instructions contained in the distribution, we run the make with the switches to patch the kernel and interactively choose features and tweaks to apply. We then make the package as instructed. Next, we do a standard recompile of the Linux kernel. You can consult the Linux kernel HOWTO (<http://>

linuxdocs.org/HOWTOs/Kernel-HOWTO.html) for more information, but here are the basic steps.

You can manually edit the file `.config` in the kernel source directory tree to include or exclude modules of (usually under `/usr/src/linux` or something similar) or run the kernel `make` interactively. From the command line, first `cd` into the top-level directory of the kernel source, and run the command-line interactive configuration:

```
# cd /usr/src/linux
```

```
# make config
```

(or use `make menuconfig`, which uses `ncurses`, or `make xconfig`, in XWindow) and answer "Y" to `CONFIG_IP_NF` flags that you wish to compile into the kernel.

The IPTables and associated functions can be loaded as modules or compiled into the kernel. To run the basic functional core of NetFilter, the kernel needs to be compiled with the options `CONFIG_NETFILTER=Y` and `CONFIG_IP_NF_IPTABLES=Y`. Otherwise, you will have to load the modules either manually or through a script before accessing them. To access each of the numerous functions of IPTables, make sure that the appropriate features have first been compiled as modules or compiled into the kernel. The compilation of these modules are through the flags `CONFIG_IP_NF_*` in the `.config` file for the kernel source. For example, the flag `CONFIG_IP_NF_NAT` determines whether to enable Network Address Translation (NAT). The flag `CONFIG_IP_NF_MATCH_MAC` determines whether MAC address matching is enabled.

Rather than being implemented through standard `ipfilter` commands, some of the stateful inspection controls are included as separate loadable code that is called a "helper module". For example, `ip_conntrack_ftp` is the helper module that tracks FTP connections tracker for FTP and is controlled by the kernel compilation flag `CONFIG_IP_NF_FTP`. If you choose to track normal or active mode FTP connections, set `CONFIG_IP_NF_FTP=Y`, then the helper module will be statically compiled into the kernel. If you choose to compile the helper module as a loadable module, then set `CONFIG_IP_NF_FTP=M`, compile, run `depmod` on the library object code, then run:

```
# /sbin/modprobe ip_conntrack_ftp
```

to load the connection tracker. Note that statically compiling or loading multiple connection trackers (for example `CONFIG_IP_NF_FTP` and `CONFIG_IP_NF_IRC`, for Internet Relay Chat service) will cause rule and compilation conflicts and should be avoided. Information on writing and extending connection tracking and NAT (Network Address Translation) modules can be found at:

[http://www.gnumonks.org/ftp/pub/doc/ \](http://www.gnumonks.org/ftp/pub/doc/)

conntrack+nat.html

I suggest, for simplicity, statically compiling the modules into the code if you are configuring IPTables/NetFilter for the first time.

Once you have finished setting all the configuration parameters for your new kernel, run:

```
# make dep
```

```
# make bzImage
```

to generate the kernel boot image.

Copy the resulting boot image to the `/boot` partition:

```
# cp /usr/src/linux/arch/i386/boot/bzImage \
```

```
# /boot/vmlinuz-iptables-1.2.1a
```

Edit the boot loader config file `/etc/lilo.conf`:

```
boot=/dev/hda
```

```
map=/boot/map
```

install=/boot/boot.b

prompt

timeout=100

image=/boot/vmlinuz-iptables-1.2.1a

label=linux-iptables

root=/dev/hda1

read-only

image=/boot/vmlinuz-2.4.x.x

label=linux

root=/dev/hda1

read-only

Rerun lilo and you should see:

linux-iptables *

linux

You should now have the option of booting into the kernel "linux-iptables" when you reboot your machine.

Setting Up IPTables Firewalling

Once you reboot your machine into the new kernel, you can try running IPTables commands. This would typically be automated on startup through an rc.d startup script. There are numerous freely available IPTables firewall scripts in the public domain. A search under "IPTables firewall script" in any Internet search engine will list numerous options. I cannot vouch for each and every script that exists in the public domain, but many will provide an excellent starting point for building your firewall.

IPTables syntax is very similar to that of ipchains. Like ipchains, IPTables also contains built-in as well as user-defined lists of rules (the chains), which each packet must first traverse. A match with the rule (the matching packet is called the "target") causes an action (e.g., ACCEPT, DROP) or a jump to a user-defined chain.

Syntax and Use of IPTables

A comprehensive documentation of IPTables/NetFilter is available online (see references below) and additional documentation can be found in the IPTables man pages.

IPTables has the following options to manage whole chains:

- N — Create a new chain.
- X — Delete an empty chain.
- P — Change the policy for a built-in chain.
- L — List the rules in a chain.
- F — Flush the rules out of a chain.
- Z — Zero the packet and byte counters.

The following are ways to manipulate rules inside the chain:

-A — Append a new rule.

-I — Insert a new rule.

-R — Replace a rule.

-D — Delete a rule.

A simple script with comments written by Oskar Andreasson of BoingWorld.com is shown in [Listing 1](#). The script shows the implementation of some simple IP forwarding, masquerading (NAT) spoofing checks of non-routable addresses (RFC1918), and the opening of some ports to allow access. If you are familiar with ipchains, you will probably have little difficulty in following most of the rules.

The command:

iptables -P <chain name> <policy>

sets the default policy for the chain, either ACCEPT or DROP (DENY). Only built-in chains (INPUT, OUTPUT, and FORWARD) have policies. Packet-mangling activities that modify the packets in transit (such as NAT and proxying) use two additional predefined chains, PREROUTE (usually for DNAT), and POSTROUTE (usually for SNAT). The names are descriptive enough and convey the fact that DNAT destination addresses are typically rewritten before other chain rules are applied, and SNAT source addresses are rewritten after they have traversed the rule chains.

Rules are added to chains using the following syntax:

iptables -A <chain name> <match condition> -j <jump>

>The condition is a logical match of, for example, the following:

-s <source IP>

-d <destination IP>

-sport <source port>

-dport <destination port>

-p <protocol tcp, udp, or icmp>

-m <match, e.g., MAC address or TCP state>

-owner <user/group/process/session id>

The above match conditions are meant to be extensible, and numerous other extensions exist. To learn more about a match extension, use the **-h** option. For example, for the ICMP protocol (**-p**), type:

#!/sbin/iptables -p icmp -h

and a list of the three dozen or so **-icmp**-type extensions will be displayed.

The jump (or "judgement") following the **-j** is one of the following:

ACCEPT — Accept the packet.

DROP — Drop the packet.

REJECT — Drop the packet and respond with "port-unreachable" ICMP packet.

NAT — Rewrite the packets source or destination address.

LOG — Log to the kernel logging daemon klogd.

TOS — Impose a type/level of service.

Example Applications of Some Rules

Here are some example rulesets for the features described in the “Enhancements” section:

1. Stateful Inspection — The rule:

```
/sbin/iptables -A FORWARD -m state --state \
ESTABLISHED,RELATED -j ACCEPT
```

forwards packets across the firewall that are part of a pre-existing connection. Besides ESTABLISHED (packet is part of existing connection) and RELATED (packet is related to existing connection and passing in same direction), other defined states are NEW (packet is trying to create a new connection), INVALID (packet doesn't match any existing connection), and RELATED+REPLY (packet is not part of an existing connection, but is related to one (e.g., ftp-data transfer requested following an existing ftp-control session)).

2. DNAT — To redirect traffic from the 10.0.0.0 network to the Web server 10.0.1.1 to the Web server 10.0.1.100, use the rule:

```
# /sbin/iptables -t nat -A POSTROUTING -s 10.0.0.0/24 -d \
10.0.1.1 -p tcp --dport 80 -j DNAT --to 10.0.1.100
```

You can implement a load-balancing solution and redirect all incoming traffic to port 8080 on a group of servers IP 10.0.1.100-10.0.1.102:

```
# /sbin/iptables -t nat -A POSTROUTING -p -s 10.0.0.0/24 \
-d 10.0.1.1 tcp -dport 80 -j DNAT --to 10.0.1.100-10.0.1.102:8080
```

3. Enhanced TCP monitoring — To check all six TCP flags and check that the SYN and ACK flags are set:

```
#!/sbin/iptables -A INPUT -p tcp --tcp-flags ALL SYN,ACK -j DROP
```

To deny outgoing connections to insecure telnet, FTP, and rsh services using a single command:

```
# /sbin/iptables -A output -t DENY -p tcp --destport telnet,ftp,shell
```

Note that in ipchains, each port would have required its own separate rule.

4. Filtering by MAC address — The rule:

```
# /sbin/iptables -A FORWARD -m state --state \
NEW -m mac --mac-source 00:C7:8F:72:14 -j ACCEPT
```

allows only outgoing packets from a known MAC address, given in colon-separated hex notation.

5. Enhanced logging:

```
# /sbin/iptables -A INPUT -s 192.168.0.1 -m limit -limit \
1/second -j LOG
```

limits the rate of writes to the logs to one per second. Specific matches can be labeled. For example, log entries corresponding to connect requests from the Litigation Department's Cisco 1601 router can be labeled:

```
# /sbin/iptables -A INPUT -s 192.168.5.254 \
-j LOG --log-prefix ' ## Litigation Dept Cisco 1601 ## '
```

The logfile entry looks like:

```
Aug 1 14:58:39 mymachine kernel: ## Litigation Dept Cisco 1601 \
## IN=eth0 OUT= MAC=00:f0:28:2c:69:67:00:00:7a:93:5e:62:08:00 \
SRC=192.168.5.254 DST=192.168.1.254 LEN=40 TOS=0x00 PREC=0x00 \
TTL=247 ID=21864 DF PROTO=TCP SPT=42300 DPT=23 WINDOW=8760
RES=0x0
```

```
0 RST URGP=0
```

6. Rate-Limited matching — To protect from syn-flood denial of service, only accept a maximum of 1 per second:

```
## /sbin/iptables -A FORWARD -p tcp -syn -m limit -limit 1/s -j ACCEPT
```

7. Type of Service (TOS) prioritization — To maximize ssh response while maintaining maximum file data transfer over HTTP connections, the following rule can be applied:

```
# /sbin/iptables -A PREROUTING -t mangle -p tcp --sport ssh \
```

```
-j TOS --set-tos Minimize-Delay
```

```
# /sbin/iptables -A PREROUTING -t mangle -p tcp --sport http \
```

```
-j TOS --set-tos Maximize-Throughput
```

Conclusion

This article explains some new firewalling features of IPTables that administrators of ipchains-based Linux firewalls may find useful. My aim was also to provide a starting point for administrators who are familiar with ipchains based-firewalling and are considering a move to IPTables/NetFilter. Much of the hoopla surrounding the Linux 2.4 kernel has revolved around IPTables support for stateful packet inspection. However, I hope that this article has shown that there is more to IPTables than merely stateful inspection. IPTables also can provide firewalling scripts that are cleaner and easier to read, and easier to maintain. It seems that with its many powerful new features, Open Source Linux firewalling has finally come of age.

References

1. NetFilter home page: <http://netfilter.samba.org>
2. NetFilter/IPTables FAQ: <http://netfilter.samba.org/netfilter-faq.html>
3. Linux 2.4 Packet Filtering HOWTO: <http://netfilter.samba.org/unreliable-guides/packet-filtering-HOWTO/index.html>
4. Linux 2.4 NAT HOWTO: <http://netfilter.samba.org/unreliable-guides/NAT-HOWTO/index.html>

Duncan Napier is an avid mountain biker. When not out riding trails, he may be found running Napier Systems Research, an IT networking consultancy based in North Vancouver, B.C., Canada. He can be contacted at: napier@computer.org.

REGISTER TODAY!
www.sysadminmag.com/events/

**Sys
Admin**
the journal for UNIX and Linux
systems administrators

MarketPlace

[We BUY Technology Hardware!](#)

We'll buy your Technology Hardware even if you don't buy ours. Get an offer today!

[We SELL Technology Hardware!](#)

Let Millenium help with your IT Hardware planning and procurement. Get a Quote Today!

[Timesheet + time tracking for payroll and projects](#)

Clockware is the first timesheet and time tracking software that is 100% J2EE-compliant. Clockware's Payroll Timesheet integrates with all major Payroll systems. Clockware also supports Time and Attendance, and Project Timesheets in one system.

[Cross Platform CORBA Tool](#)

Develop distributed systems conforming to open standards like CORBA and Web Services faster with SANKHYA Varadhi - The Digital Bridge.

[Wanna see your ad here?](#)

[Copyright © 2007 CMP Media LLC](#), [Privacy Policy](#), [Your California Privacy rights](#)

Comments about the Web site: webmaster@sysadminmag.com

SDMG Web Sites: [BYTE.com](#), [dotnetjunkies.com](#), [Dr. Dobb's Journal](#), [MSDN Magazine](#), [SD Expo](#), [Sys Admin](#), [sqljunkies.com](#), [UnixReview.com](#),