

How to use IP Tables

By krnlpanic

With the release of the 2.4 kernel, everyone who was used to using ipchains to configure their firewall will now be looking at iptables. The native packet filtering mechanism for the 2.4 kernel series is iptables, but you can still compile in ipchains support at kernel configuration time.

What Are These "Chains"?

Just like its predecessor "ipchains", iptables uses a set of chain rules. The three default chains are named INPUT, OUTPUT, and FORWARD. A chain is just a simple checklist of rules and specifies what to do with each of the packets. The chain rules will either ACCEPT a packet or DROP a packet (note the "drop" syntax rather than "deny"). If the packet doesn't have anymore rules left in the chain, the system will consult the chain policy to decide what to do. Most systems are setup with a policy of deny. So if the packet doesn't match any rules that "allow" it through, then it will "drop" it.

The first decision the kernel has to make upon receipt of a packet is "Where is the destination?" If the destination is for the box itself it will consult the rules for the INPUT chain. If the destination is for another network interface (and you have IP Forwarding enabled), the packet is compared against the FORWARD chain. As long as the packet gets an "ACCEPT" by one of the chain rules the packet will be forwarded on. If the linux box itself needs to send network packets, it will consult the OUTPUT chain and if the packet is ACCEPT-ed by one of the rules it will be sent out to the appropriate interface.

One of the key concepts for people transitioning from ipchains is to realize that the INPUT and OUTPUT chains actually refer to the local machine rather than to all incoming and outgoing packets. Another point to consider is the use of "-o" to specify the interface. We used to use "-i" in ipchains to refer to the interface. In iptables "-i" is only used when referring to the incoming interface (so on the INPUT and FORWARD chains it is ok). So you are pretty much going to want to use "-o" where you used to use the "-i". This will refer to both the FORWARD and OUTPUT interface.

Chain Types and Options

As I mentioned, you have three basic chains:

INPUT
OUTPUT
FORWARD

However, you can create your own chains by using:

`/sbin/iptables -N`

The above will create a new chain. Here are some more options to manipulate your chains:

- N Create a new chain
- X Delete an EMPTY chain
- P Change the Policy for a built-in chain
- L Lists the chain rules
- F Flushes the rules of a chain
- Z Sets the counters to zero on all the rules in a chain

Command Line Examples

Now for some options you can use to configure rules inside your chains:

- A Append a new rule
- I Insert a new rule

How to use IP Tables

By krnlpanic

- R Replace a rule at a certain position
- D Delete a rule at a certain position

For example:

```
/sbin/iptables -A INPUT -p tcp -j ACCEPT
```

This rule would accept all tcp traffic. This is a little too broad isn't it? Let's take a look at how we can specify some other options.

Taking it one step further:

- j Specify the target (--jump)
- i Specify the input interface (--in-interface)
- o Specify the output interface (--out-interface)
- p Specify the protocol (--proto)
- s Specify the source (--source)
- d Specify the destination (--destination)
- ! Specifies an inversion (match addresses NOT equal to)

Now we're talking! Let's try it...

```
/sbin/iptables -A FORWARD -s 192.168.1.0/24 -p tcp -j ACCEPT
```

This rule will allow traffic to be forwarded, as long as the protocol was tcp, and the source was a machine on the 192.168.1.0 subnet

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

This rule, coupled with the one above will allow for MASQUERADE(ing) your internal network traffic, via NAT (Network Address Translation), so that you can share your internet connection with the rest of your network.

Some useful tcp options (these also work for udp):

- sport Filters on the source port
- dport Filters on the destination port

This is handy. Let's try it out!

This rule would allow traffic going to the www port (80) to be forwarded on.

```
/sbin/iptables -A FORWARD -p tcp --dport 80 -j ACCEPT
```

Change destination addresses of web traffic to 5.6.7.8, port 8080.

```
/sbin/iptables -t nat -A PREROUTING -p tcp --dport 80 -i eth0 -j DNAT --to 5.6.7.8:8080
```

--tcp-flags

This allows you to filter on specific TCP flags. The first option after "--tcp-flags" specifies which flags are to be examined, and the second option specifies which flags are to be set.

Here is an example of the --tcp-flags in use:

```
/sbin/iptables -A INPUT -p tcp --tcp-flags ALL SYN -j DENY
```

Note: The list of possible flags is as follows
SYN,ACK,FIN,RST,URG,PSH

How to use IP Tables

By krnlpanic

One of the other nice features is the ability to use the "! --syn" option. This is equivalent to typing:
`--tcp-flags ACK,FIN,RST,URG,PSH`

This would be useful in setting up your firewall to accept only connections that were initiated internally.

More information on using IPTABLES to block entire network blocks - This is useful if you want to block access to your webserver from network blocks that are outside the United States. This has really helped to limit the number of users who spam my message boards, contact forms and other network facilities.

I hope this helps!

-Krnlpanic