

# Introduction to Iptables

<[www.iptables.org](http://www.iptables.org)>

## 1 What is Iptables?

Iptables is the native packet filtering mechanism for the 2.4 kernel series. We can use it to filter packets, implement network address translation and mangle packets. There are three default chains for filtering packets: INPUT, OUTPUT and FORWARD, and two default chains for network address translation: PREROUTING and POSTROUTING, and three tables: filter, nat and mangle. You can add your own chains.

### 1.1 Filter Packet

Filter table is used to filter packets, which looks at the *header* of packets as they pass through, and decides the fate of the entire packet. It might decide to **DROP** the packet, **ACCEPT** the packet, or something more complicated. The iptables tool inserts and deletes rules from the kernel's packet filtering table.

The kernel starts with three lists of rules in the `filter' table; these lists are called **firewall chains** or just **chains**. The three chains are called **INPUT**, **OUTPUT** and **FORWARD**.

1. When a packet comes in (say, through the Ethernet card) the kernel first looks at the destination of the packet: this is called `routing'.
2. If it's destined for this box, the packet passes to the INPUT chain. If it passes this, any processes waiting for that packet will receive it.
3. Otherwise, if the kernel does not have forwarding enabled, or it doesn't know how to forward the packet, the packet is dropped. If forwarding is enabled, and the packet is destined for another network interface (if you have another one), then the packet goes to the FORWARD chain. If it is ACCEPTed, it will be sent out.

4. Finally, a program running on the box can send network packets. These packets pass through the OUTPUT chain immediately: if it says ACCEPT, then the packet continues out to whatever interface it is destined for.

## 1.2 NAT

Normally, packets on a network travel from their source to their destination through many different links. None of these links really alter your packet: they just send it onward.

If one of these links were to do NAT, then they would alter the source or destinations of the packet as it passes through. Usually the link doing NAT will remember how it mangled a packet, and when a reply packet passes through the other way, it will do the reverse mangling on that reply packet, so everything works.

NAT is divided into two different types: **Source NAT** (SNAT) and **Destination NAT** (DNAT).

Source NAT is when you alter the source address of the first packet: i.e. you are changing where the connection is coming from. Source NAT is always done post-routing, just before the packet goes out onto the wire. Destination NAT is when you alter the destination address of the first packet: i.e. you are changing where the connection is going to. Destination NAT is always done before routing, when the packet first comes off the wire. Port forwarding, load sharing, and transparent proxying are all forms of DNAT.

You need to create NAT rules which tell the kernel what connections to change, and how to change them. To do this, we use the iptables tool, and tell it to alter the NAT table by specifying the `-t nat` option.

The table of NAT rules contains two chains, called PREROUTING (for Destination NAT, as packets first come in), and POSTROUTING (for Source NAT, as packets leave).

There is a specialized case of Source NAT called masquerading: it should only be used for dynamically-assigned IP addresses, such as standard dialups (for static IP addresses, use SNAT above).

## 1.3 Mangle Packet

Mangle table is used for mangling packets. There are three targets in this table: TOS, TTL and MARK. The TOS target is used to set and/or change the Type of Service field in the packet. The TTL target is used to change the TTL (Time To Live) field of the packet. The MARK target is used to set special mark values to the packet.

## 2 Syntax of iptables

**iptables** [-t *table*] command [match] [target/jump]

'-t' specifies the table to use.

The 'match' tells the rule the conditions that the packets should satisfy.

The 'target/jumps' tells the rule what to do with a packet that is a perfect match with the match section of the rule.

### 2.1 Tables

'-t' is used to specify the table you use in the rule. It could be mangle table, filter table, and NAT table. Filter table is default.

### 2.2 Commands:

Command	<b>-A, --append</b>
Example	<b>iptables -A INPUT ...</b>
Explanation	This command appends the rule to the end of the chain.
Command	<b>-D, --delete</b>
Example	<b>iptables -D INPUT --dport 80 -j DROP, iptables -D INPUT 1</b>

Explanation	This command deletes a rule in a chain. This could be done in two ways; either by entering the whole rule to match (as in the first example), or by specifying the rule number that you want to match. The rules are numbered from the top of each chain, starting with number 1.
Command	<b>-R, --replace</b>
Example	<b>iptables -R INPUT 1 -s 192.168.0.1 -j DROP</b>
Explanation	This command replaces the old entry at the specified line.
Command	<b>-I, --insert</b>
Example	<b>iptables -I INPUT 1 --dport 80 -j ACCEPT</b>
Explanation	Insert a rule somewhere in a chain.
Command	<b>-L, --list</b>
Example	<b>iptables -L INPUT</b>
Explanation	This command lists all the entries in the specified chain.
Command	<b>-F, --flush</b>
Example	<b>iptables -F INPUT</b>
Explanation	This command flushes all rules from the specified chain.
Command	<b>-Z, --zero</b>
Example	<b>iptables -Z INPUT</b>
Explanation	This command tells the program to zero all counters in a specific chain, or in all chains.
Command	<b>-N, --new-chain</b>
Example	<b>iptables -N allowed</b>
Explanation	This command tells the kernel to create a new chain of the specified name in the specified table.
Command	<b>-X, --delete-chain</b>
Example	<b>iptables -X allowed</b>
Explanation	This command deletes the specified chain from the table.
Command	<b>-P, --policy</b>
Example	<b>iptables -P INPUT DROP</b>
Explanation	This command tells the kernel to set a specified default target, or policy, on a chain. All packets that don't match any rule will then be forced to use the policy of the chain. Legal targets are <b>DROP</b> and <b>ACCEPT</b> (There might be more, mail me if so).

Command	<b>-E, --rename-chain</b>
Example	<b>iptables -E allowed disallowed</b>
Explanation	The <b>-E</b> command tells <b>iptables</b> to change the first name of a chain, to the second name.

## 2.3 Matches

Match	<b>-p, --protocol</b>
Example	<b>iptables -A INPUT -p tcp</b>
Explanation	This match is used to check for certain protocols. The protocol must be TCP, UDP or ICMP.
Match	<b>-s, --src, --source</b>
Example	<b>iptables -A INPUT -s 192.168.1.1</b>
Explanation	This is the source match, which is used to match packets, based on their source IP address. The main form can be used to match single IP addresses, such as <i>192.168.1.1</i> , or whole IP ranges.
Match	<b>-d, --dst, --destination</b>
Example	<b>iptables -A INPUT -d 192.168.1.1</b>
Explanation	The <b>--destination</b> match is used for packets based on their destination address or addresses.
Match	<b>-i, --in-interface</b>
Example	<b>iptables -A INPUT -i eth0</b>
Explanation	This match is used for the interface the packet came in on. Note that this option is only legal in the INPUT, FORWARD and PREROUTING chains and will return an error message when used anywhere else.
Match	<b>-o, --out-interface</b>
Example	<b>iptables -A FORWARD -o eth0</b>
Explanation	The <b>--out-interface</b> match is used for packets on the interface from which they are leaving. Note that this match is only available in the OUTPUT, FORWARD and POSTROUTING chains, the opposite in fact of the <b>--in-interface</b> match.
Match	<b>-f, --fragment</b>
Example	<b>iptables -A INPUT -f</b>
Explanation	This match is used to match the second and third part of a fragmented packet.

### 2.3.1 TCP matches

These matches are protocol specific and are only available when working with TCP packets and streams. To use these matches, you need to specify **--protocol tcp** on the command line before trying to use them.

Match	<b>--sport, --source-port</b>
Example	<b>iptables -A INPUT -p tcp --sport 22</b>
Explanation	The <b>--source-port</b> match is used to match packets based on their source port.
Match	<b>--dport, --destination-port</b>
Example	<b>iptables -A INPUT -p tcp --dport 22</b>
Explanation	This match is used to match TCP packets, according to their destination port.
Match	<b>--tcp-flags</b>
Example	<b>iptables -p tcp --tcp-flags SYN,FIN,ACK SYN</b>
Explanation	This match is used to match on the TCP flags in a packet.
Match	<b>--tcp-option</b>
Example	<b>iptables -p tcp --tcp-option 16</b>
Explanation	This match is used to match packets depending on their TCP options.

### 2.3.2 UDP matches

These matches are protocol specific and are only available when working with UDP packets and streams. To use these matches, you need to specify **--protocol udp** on the command line before trying to use them.

Match	<b>--sport, --source-port</b>
Example	<b>iptables -A INPUT -p udp --sport 53</b>
Explanation	This is used to perform matches on packets based on their source UDP ports.
Match	<b>--dport, --destination-port</b>
Example	<b>iptables -A INPUT -p udp --dport 53</b>
Explanation	This matches packets based on their UDP destination port.

### 2.3.3 ICMP matches

Match	<b>--icmp-type</b>
Example	<b>iptables -A INPUT -p icmp --icmp-type 8</b>

Explanation	This match is used to specify the ICMP type to match.
-------------	---

### 2.3.4 MAC match

Match	<b>--mac-source</b>
Example	<b>iptables -A INPUT -m mac --mac-source 00:00:00:00:00:01</b>
Explanation	This match is used to match packets based on their MAC source address. The MAC address specified must be in the form <i>XX:XX:XX:XX:XX:XX</i> . The <b>MAC</b> match is only valid in the PREROUTING, FORWARD and INPUT chains and nowhere else.

### 2.3.5 Multiport match

Match	<b>--source-port</b>
Example	<b>iptables -A INPUT -p tcp -m multiport --source-port 22,53,80,110</b>
Explanation	This match matches multiple source ports.
Match	<b>--destination-port</b>
Example	<b>iptables -A INPUT -p tcp -m multiport --destination-port 22,53,80,110</b>
Explanation	This match is used to match multiple destination ports.
Match	<b>--port</b>
Example	<b>iptables -A INPUT -p tcp -m multiport --port 22,53,80,110</b>
Explanation	This match extension can be used to match packets based both on their destination port and their source port. Note that the <b>--port</b> match will only match packets coming in from and going to the same port, for example, port 80 to port 80, port 110 to port 110 and so on.

## 2.4 Targets/Jumps

### 2.4.1 ACCEPT target

Syntax: -j ACCEPT

Accept the packet.

### 2.4.2 DROP target

Syntax: -j DROP

Drop the packet

### 2.4.3 REJECT target

Syntax: -j REJECT

Reject the packet.

### 2.4.4 DNAT target

The **DNAT** target is used to do Destination Network Address Translation, which means that it is used to rewrite the Destination IP address of a packet, if the packet is matched. **DNAT** target is only available within the PREROUTING and OUTPUT chains in the nat table.

There is an option for DNAT target.

Option	<b>--to-destination</b>
Example	<b>iptables -t nat -A PREROUTING -p tcp -d 15.45.23.67 --dport 80 -j DNAT --to-destination 192.168.1.1-192.168.1.10</b>
Explanation	The <b>--to-destination</b> option tells the DNAT mechanism which Destination IP to set in the IP header, and where to send packets that are matched.

### 2.4.5 SNAT target

The **SNAT** target is used to do Source Network Address Translation, which means that this target will rewrite the Source IP address in the IP header of the packet. The **SNAT** target is only valid within the nat table, within the POSTROUTING chain.

There is an option for SNAT target:

Option	<b>--to-source</b>
Example	<b>iptables -t nat -A POSTROUTING -p tcp -o eth0 -j SNAT --to-source 194.236.50.155-194.236.50.160:1024-32000</b>
Explanation	The <b>--to-source</b> option is used to specify which source the packet should use.

## 2.4.6 MASQUERADE target

The **MASQUERADE** target is used basically the same as the **SNAT** target, but it does not require any **--to-source** option. The **MASQUERADE** target is only valid within the POSTROUTING chain in the nat table, just as the **SNAT** target.

The **MASQUERADE** target takes one option specified below, which is optional.

Option	<b>--to-ports</b>
Example	<b>iptables -t nat -A POSTROUTING -p TCP -j MASQUERADE --to-ports 1024-31000</b>
Explanation	The <b>--to-ports</b> option is used to set the source port or ports to use on outgoing packets.

## 3 A short tutorial

### 3.1 List Rules

To look at the tables that are currently in effect, run [iptables -L](#) or [iptables --list](#).

By default, you have an INPUT, OUTPUT, and FORWARD chain--all with a policy of accepting packets.

### 3.2 Adding Rules

Without any rules, iptables isn't going to do much, so let's add some rules to the existing chains. If you don't want your machine to respond to pings, for instance, add the following rule to the INPUT chain:

```
iptables -A INPUT -p icmp -j DROP
```

The `-A INPUT` argument tells iptables to append to the INPUT chain. The `-p icmp` argument indicates that this rule applies to the icmp protocol, and the `-j DROP` argument indicates that packets matching this rule should be dropped. If you send a ping to that host, it will simply drop the packets and not reply. Note that you could use either ICMP or icmp to specify the protocol; it's not case-sensitive.

### 3.3 Delete Rules

To reverse this rule and allow the host to respond to pings again, issue this command:

```
iptables -D INPUT 1
```

This tells iptables to drop (-D) the first rule from the INPUT chain. If you have multiple rules, you may drop any one of them without affecting the others.

To clear out all rules from a chain, use this syntax:

```
iptables -F INPUT
```

This tells iptables to flush (-F) all rules from the chain.