Chapter 10. Iptables matches

In this chapter we'll talk a bit more about matches. I've chosen to narrow down the matches into five different subcategories. First of all we have the *generic matches*, which can be used in all rules. Then we have the *TCP matches* which can only be applied to TCP packets. We have *UDP matches* which can only be applied to UDP packets, and *ICMP matches* which can only be used on ICMP packets. Finally we have special matches, such as the state, owner and limit matches and so on. These final matches have in turn been narrowed down to even more subcategories, even though they might not necessarily be different matches at all. I hope this is a reasonable breakdown and that all people out there can understand it.

As you may already understand if you have read the previous chapters, a match is something that specifies a special condition within the packet that must be true (or false). A single rule can contain several matches of these kinds. For example, we may want to match packets that come from a specific host on a our local area network, and on top of that only from specific ports on that host. We could then use matches to tell the rule to only apply the target - or jump specification - on packets that have a specific source address, that come in on the interface that connects to the LAN and the packets must be one of the specified ports. If any one of these matches fails (e.g., the source address isn't correct, but everything else is true), the whole rule fails and the next rule is tested on the packet. If all matches are true, however, the target specified by the rule is applied.

10.1. Generic matches

This section will deal with *Generic matches*. A generic match is a kind of match that is always available, whatever kind of protocol we are working on, or whatever match extensions we have loaded. No special parameters at all are needed to use these matches; in other words. I have also included the **--protocol** match here, even though it is more specific to protocol matches. For example, if we want to use a *TCP match*, we need to use the **--protocol** match and send TCP as an option to the match. However, **--protocol** is also a match in itself, since it can be used to match specific protocols. The following matches are always available.

Table 10-1. Generic matches

Match	**-p, --protocol**
Kernel	2.3, 2.4, 2.5 and 2.6
Example	**iptables -A INPUT -p tcp**
Explanation	This match is used to check for certain protocols. Examples of protocols are TCP, UDP and ICMP. The protocol must either be one of the internally specified TCP, UDP or ICMP. It may also take a value specified in the */etc/protocols* file, and if it can't find the protocol there it will

	reply with an error. The protocl may also be an integer value. For example, the ICMP protocol is integer value 1, TCP is 6 and UDP is 17. Finally, it may also take the value ALL. *ALL* means that it matches only TCP, UDP and ICMP. If this match is given the integer value of zero (0), it means ALL protocols, which in turn is the default behavior, if the **--protocol** match is not used. This match can also be inversed with the **!** sign, so **--protocol ! tcp** would mean to match UDP and ICMP.
Match	**-s**, **--src**, **--source**
Kernel	2.3, 2.4, 2.5 and 2.6
Example	**iptables -A INPUT -s 192.168.1.1**
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 *192.168.1.1*. It could also be used with a netmask in a CIDR "bit" form, by specifying the number of ones (1's) on the left side of the network mask. This means that we could for example add */24* to use a *255.255.255.0* netmask. We could then match whole IP ranges, such as our local networks or network segments behind the firewall. The line would then look something like *192.168.0.0/24*. This would match all packets in the *192.168.0.x* range. Another way is to do it with a regular netmask in the *255.255.255.255* form (i.e., *192.168.0.0/255.255.255.0*). We could also invert the match with an **!** just as before. If we were, in other words, to use a match in the form of **--source ! 192.168.0.0/24**, we would match all packets with a source address not coming from within the *192.168.0.x* range. The default is to match all IP addresses.
Match	**-d**, **--dst**, **--destination**
Kernel	2.3, 2.4, 2.5 and 2.6
Example	**iptables -A INPUT -d 192.168.1.1**
Explanation	The **--destination** match is used for packets based on their destination address or addresses. It works pretty much the same as the **--source** match and has the same syntax, except that the match is based on where the packets are going to. To match an IP range, we can add a netmask either in the exact netmask form, or in the number of ones (1's) counted from the left side of the netmask bits. Examples are: *192.168.0.0/255.255.255.0* and *192.168.0.0/24*. Both of these are equivalent. We could also invert the whole match with an **!** sign, just as before. **--destination ! 192.168.0.1** would in other words match all packets except those destined to the *192.168.0.1* IP address.
Match	**-i**, **--in-interface**
Kernel	2.3, 2.4, 2.5 and 2.6
Example	**iptables -A INPUT -i eth0**
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. The default behavior of this match, if no particular interface is specified, is to assume a string value of +. The + value is used to match a string of letters and numbers. A single + would, in other words, tell the kernel to match all packets without considering which interface it came in on. The + string can also be appended to the type of interface, so **eth+** would be all Ethernet devices. We can also invert the meaning of this option with the help of the **!** sign. The line would then have a syntax looking something like **-i ! eth0**, which would match all incoming interfaces, except eth0.
Match	**-o**, **--out-interface**
Kernel	2.3, 2.4, 2.5 and 2.6
Example	**iptables -A FORWARD -o eth0**
Explanation	The **--out-interface** 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 **--in-interface** match. Other than this, it works pretty much the same as the **--in-interface** match. The + extension is understood as matching all devices of similar type, so **eth+** would match all eth devices and so on. To invert the meaning of the match, you can use the **!** sign in exactly the same way as for the --in-interface match. If no **--out-interface** is specified, the default behavior for this match is to match all devices, regardless of where the packet is going.
Match	**-f**, **--fragment**
Kernel	2.3, 2.4, 2.5 and 2.6
Example	**iptables -A INPUT -f**
Explanation	This match is used to match the second and third part of a fragmented packet. The reason for this is that in the case of fragmented packets, there is no way to tell the source or destination ports of the fragments, nor ICMP types, among other things. Also, fragmented packets might in rather special cases be used to compound attacks against other computers. Packet fragments like this will not be matched by other rules, and hence this match was created. This option can also be used in conjunction with the **!** sign; however, in this case the **!** sign must precede the match, i.e. **! -f**. When this match is inverted, we match all header fragments and/or unfragmented packets. What this means, is that we match all the first fragments of fragmented packets, and not the second, third, and so on. We also match all packets that have not been fragmented during transfer. Note also that there are really good defragmentation options within the kernel that you can use instead. As a secondary note, if you use connection tracking you will not see any fragmented packets, since they are dealt with before hitting any chain or table in **iptables**.