10.2. Implicit matches

This section will describe the matches that are loaded implicitly. *Implicit matches* are implied, taken for granted, automatic. For example when we match on **--protocol tcp** without any further criteria. There are currently three types of implicit matches for three different protocols. These are *TCP matches*, *UDP matches* and *ICMP matches*. The TCP based matches contain a set of unique criteria that are available only for TCP packets. UDP based matches contain another set of criteria that are available only for UDP packets. And the same thing for ICMP packets. On the other hand, there can be explicit matches that are loaded explicitly. *Explicit matches* are not implied or automatic, you have to specify them specifically. For these you use the **-m** or **--match** option, which we will discuss in the next section.

10.2.1. 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. Note that the **--protocol tcp** match must be to the left of the protocol specific matches. These matches are loaded implicitly in a sense, just as the *UDP* and *ICMP matches* are loaded implicitly. The other matches will be looked over in the continuation of this section, after the *TCP match* section.

Table 10-2. TCP matches

Match	**--sport**, **--source-port**
Kernel	2.3, 2.4, 2.5 and 2.6
Example	**iptables -A INPUT -p tcp --sport 22**
Explanation	The **--source-port** match is used to match packets based on their source port. Without it, we imply all source ports. This match can either take a service name or a port number. If you specify a service name, the service name must be in the */etc/services* file, since **iptables** uses this file in which to find. If you specify the port by its number, the rule will load slightly faster, since **iptables** don't have to check up the service name. However, the match might be a little bit harder to read than if you use the service name. If you are writing a rule-set consisting of a 200 rules or more, you should definitely use port numbers, since the difference is really noticeable. (On a slow box, this could make as much as 10 seconds' difference, if you have configured a large rule-set containing 1000 rules or so). You can also use the **--source-port** match to match any range of ports, **--source-port 22:80** for example. This example would match all source ports between 22 and 80. If you omit specifying the first port, port 0 is assumed (is implicit). **--source-port :80** would then match port 0 through 80. And if the last port specification is omitted, port 65535 is assumed. If you were to write **--source-port 22:**, you would have specified a match for all ports from port 22 through port 65535. If you invert the port range, iptables automatically reverses your inversion. If you write **--source-port 80:22**, it is simply interpreted as **--source-port 22:80**. You can

	also invert a match by adding a **!** sign. For example, **--source-port ! 22** means that you want to match all ports but port 22. The inversion could also be used together with a port range and would then look like **--source-port ! 22:80**, which in turn would mean that you want to match all ports but ports 22 through 80. Note that this match does not handle multiple separated ports and port ranges. For more information about those, look at the multiport match extension.
Match	**--dport**, **--destination-port**
Kernel	2.3, 2.4, 2.5 and 2.6
Example	**iptables -A INPUT -p tcp --dport 22**
Explanation	This match is used to match TCP packets, according to their destination port. It uses exactly the same syntax as the **--source-port** match. It understands port and port range specifications, as well as inversions. It also reverses high and low ports in port range specifications, as above. The match will also assume values of 0 and 65535 if the high or low port is left out in a port range specification. In other words, exactly the same as the **--source-port** syntax. Note that this match does not handle multiple separated ports and port ranges. For more information about those, look at the multiport match extension.
Match	**--tcp-flags**
Kernel	2.3, 2.4, 2.5 and 2.6
Example	**iptables -p tcp --tcp-flags SYN,FIN,ACK SYN**
Explanation	This match is used to match on the TCP flags in a packet. First of all, the match takes a list of flags to compare (a mask) and secondly it takes list of flags that should be set to 1, or turned on. Both lists should be comma-delimited. The match knows about the SYN, ACK, FIN, RST, URG, PSH flags, and it also recognizes the words ALL and NONE. ALL and NONE is pretty much self describing: ALL means to use all flags and NONE means to use no flags for the option. **--tcp-flags ALL NONE** would in other words mean to check all of the TCP flags and match if none of the flags are set. This option can also be inverted with the **!** sign. For example, if we specify **! SYN,FIN,ACK SYN**, we would get a match that would match packets that had the ACK and FIN bits set, but not the SYN bit. Also note that the comma delimitation should not include spaces. You can see the correct syntax in the example above.
Match	**--syn**
Kernel	2.3, 2.4, 2.5 and 2.6
Example	**iptables -p tcp --syn**
Explanation	The **--syn** match is more or less an old relic from the ipchains days and is still there for backward compatibility and for and to make transition one to the other easier. It is used to match packets if they have the SYN bit set and the ACK and RST bits unset. This command would in other words be exactly the same as the **--tcp-flags SYN,RST,ACK SYN** match. Such packets are mainly used to request new TCP connections from a server. If you block these packets, you should have effectively blocked all incoming connection attempts. However, you will not have blocked the outgoing connections, which a lot of exploits today use (for example, hacking a legitimate service and then installing a program or suchlike that enables initiating an existing connection to your host, instead of opening up a new port on it). This match can also be inverted with the **!** sign in this, **! --syn**, way. This would match all packets with the RST or the ACK bits set, in other words packets in an already established connection.
Match	**--tcp-option**
Kernel	2.3, 2.4, 2.5 and 2.6
Example	**iptables -p tcp --tcp-option 16**

Explanation	This match is used to match packets depending on their TCP options. A TCP Option is a specific part of the header. This part consists of 3 different fields. The first one is 8 bits long and tells us which Options are used in this stream, the second one is also 8 bits long and tells us how long the options field is. The reason for this length field is that TCP options are, well, optional. To be compliant with the standards, we do not need to implement all options, but instead we can just look at what kind of option it is, and if we do not support it, we just look at the length field and can then jump over this data. This match is used to match different TCP options depending on their decimal values. It may also be inverted with the **!** flag, so that the match matches all TCP options but the option given to the match. For a complete list of all options, take a closer look at the *Internet Engineering Task Force* who maintains a list of all the standard numbers used on the Internet.

10.2.2. UDP matches

This section describes matches that will only work together with UDP packets. These matches are implicitly loaded when you specify the **--protocol UDP** match and will be available after this specification. Note that UDP packets are not connection oriented, and hence there is no such thing as different flags to set in the packet to give data on what the datagram is supposed to do, such as open or closing a connection, or if they are just simply supposed to send data. UDP packets do not require any kind of acknowledgment either. If they are lost, they are simply lost (Not taking ICMP error messaging etc into account). This means that there are quite a lot less matches to work with on a UDP packet than there is on TCP packets. Note that the state machine will work on all kinds of packets even though UDP or ICMP packets are counted as connectionless protocols. The state machine works pretty much the same on UDP packets as on TCP packets.

Table 10-3. UDP matches

Match	**--sport**, **--source-port**
Kernel	2.3, 2.4, 2.5 and 2.6
Example	**iptables -A INPUT -p udp --sport 53**
Explanation	This match works exactly the same as its TCP counterpart. It is used to perform matches on packets based on their source UDP ports. It has support for port ranges, single ports and port inversions with the same syntax. To specify a UDP port range, you could use 22:80 which would match UDP ports 22 through 80. If the first value is omitted, port 0 is assumed. If the last port is omitted, port 65535 is assumed. If the high port comes before the low port, the ports switch place with each other automatically. Single UDP port matches look as in the example above. To invert the port match, add a **!** sign, **--source-port ! 53**. This would match all ports but port 53. The match can understand service names, as long as they are available in the */etc/services* file. Note that this match does not handle multiple separated ports and port ranges. For more information about this, look at the multiport match extension.
Match	**--dport**, **--destination-port**
Kernel	2.3, 2.4, 2.5 and 2.6
Example	**iptables -A INPUT -p udp --dport 53**
Explanation	The same goes for this match as for **--source-port** above. It is exactly the same as for the equivalent TCP match, but here it applies to UDP packets. It matches packets based on their UDP destination port. The match handles port ranges, single ports and inversions. To match a single port you use, for example, **--destination-port 53**, to invert this you would use **--destination-port ! 53**. The first would match all UDP packets going to

	port 53 while the second would match packets but those going to the destination port 53. To specify a port range, you would, for example, use **--destination-port 9:19**. This example would match all packets destined for UDP port 9 through 19. If the first port is omitted, port 0 is assumed. If the second port is omitted, port 65535 is assumed. If the high port is placed before the low port, they automatically switch place, so the low port winds up before the high port. Note that this match does not handle multiple ports and port ranges. For more information about this, look at the multiport match extension.

10.2.3. ICMP matches

These are the *ICMP matches*. These packets are even more ephemeral, that is to say short lived, than UDP packets, in the sense that they are connectionless. The ICMP protocol is mainly used for error reporting and for connection controlling and suchlike. ICMP is not a protocol subordinated to the IP protocol, but more of a protocol that augments the IP protocol and helps in handling errors. The headers of ICMP packets are very similar to those of the IP headers, but differ in a number of ways. The main feature of this protocol is the type header, that tells us what the packet is for. One example is, if we try to access an unaccessible IP address, we would normally get an `ICMP host unreachable` in return. For a complete listing of ICMP types, see the *ICMP types* appendix. There is only one ICMP specific match available for ICMP packets, and hopefully this should suffice. This match is implicitly loaded when we use the **--protocol ICMP** match and we get access to it automatically. Note that all the generic matches can also be used, so that among other things we can match on the source and destination addresses.

Table 10-4. ICMP matches

Match	**--icmp-type**
Kernel	2.3, 2.4, 2.5 and 2.6
Example	**iptables -A INPUT -p icmp --icmp-type 8**
Explanation	This match is used to specify the ICMP type to match. ICMP types can be specified either by their numeric values or by their names. Numerical values are specified in RFC 792. To find a complete listing of the ICMP name values, do an **iptables --protocol icmp --help**, or check the *ICMP types* appendix. This match can also be inverted with the **!** sign in this, **--icmp-type ! 8**, fashion. Note that some ICMP types are obsolete, and others again may be "dangerous" for an unprotected host since they may, among other things, redirect packets to the wrong places. The type and code may also be specified by their typename, numeric type, and type/code as well. For example **--icmp-type network-redirect**, **--icmp-type 8** or **--icmp-type 8/0**. For a complete listing of the names, type **iptables -p icmp --help**. Please note that netfilter uses ICMP type 255 to match all ICMP types. If you try to match this ICMP type, you will wind up with matching all ICMP types.