PCAP-filters

✓ Successfully pass the free certification exam at IW Academy and become an Infinet Certified Engineer.

To the certification exam

- Description
- Primitives
- Examples

Description

In the text form, the PCAP filter is an expression which consists of one or more primitives. Primitives in the expression determine whether the filter can accept the packet. Each primitive defines a specific element of the standard protocol packet and its value, compared by the filter with the corresponding element value of the packet. If the primitive value coincides with the packet element value, the filter marks it as true and proceeds to compare the next primitive. In case all expression values coincide with the checked elements values, the filter decides to accept this packet, otherwise the packet is ignored.

Primitives usually consist of an id (name or number) preceded by one or more qualifiers. There are three different kinds of qualifier:

- *"type"* – id name or number type. Possible values: host, net, port and portrange. If there is no type qualifier, host is assumed.
- *"dir"* – transfer direction to and/or from id. Possible directions: *"src"* (source), *"dst"* (destination), *"src and dst"* (source and destination), *"src or dst"* (source or destination). If no qualifier is specified *"src or dst"* is used.
- *"proto"* – protocol type. Possible values: *"ether", "fddi", "ip", "arp", "rarp", "decnet", "lat", "sca", "moprc", "mopdl", "tcp"* and *"udp"*. If there is no proto qualifier, all protocols consistent with the type are assumed.

In addition to the above, there are some special primitive keywords that don't follow the pattern: "*broadcast*", "*less*", "*greater*" and arithmetic expressions. Detailed description is given below.

More complex filter expressions are built up by using the words "*and*", "*or*" and "*not*" to combine primitives. Primitives can be grouped with brackets and logical operations:

- negation ("!" or *"not"*);
- addition ("&&" or "and");
- disjunction ("||" or "or").

Negation has the highest priority. The addition and disjunction have same priority in the expression and are read from left to right.

⚠ **NOTE**

If there are several identical qualifiers in the filter, it is possible not to write them down to shorten the record.

Values *"ip", "arp", "rarp", "atalk", "aarp", "iso", "stp", "ipx", "netbeui"* are abbreviations for "*ether proto p"*, there "*p"* is one of these protocols. *"tcp", "udp", "icmp"* are abbreviations for "*ip proto p"*, there "*p"* - is one of these protocols. "*clnp*", "*esis*", "*isis*" are abbreviations for *"iso proto p"*, there "*p"* - is one of these protocols.

Primitives

Primitive	Description
dst host HOST	True if the IPv4 packet destination field is *"HOST"* (may be either an address or a host name).
src host HOST	True if the IPv4 packet source field is *"HOST"*.
host HOST	True if either the IPv4 source or destination of the packet is *"HOST"*.

⚠

> ⚠️ **NOTE**
>
> Any of the above host expressions can be prefixed with the keywords *"ip", "ip6", "arp", "rarp"*.

ether dst EHOST	True if the ethernet destination MAC address is "*EHOST*". *"EHOST"* must be in numeric format: XX:XX:XX:XX:XX:XX.
ether src EHOST	True if the ethernet source MAC address is "*EHOST*".
ether host EHOST	True if either the ethernet source or destination MAC address is *"EHOST"*.
dst net NET	True if the IPv4 packet destination address has a network number of "*NET*".
src net NET	True if the IPv4 packet source address has a network number of "*NET*".
net NET	True if either the IPv4 source or destination address of the packet has a network number of "*NET*".
net NET mask NETMASK	True if the IPv4 address matches *"NET"* with the specific *"NETMASK"*. May be qualified with "*src*" and "*dst*".
net NET/LEN	True if the IPv4 address matches *"NET"* with a netmask *"LEN"* bits wide. May be qualified with "*src*" and "*dst*".
dst port PORT	True if the packet is UDP or TCP and has a destination port value "*PORT*".
src port PORT	True if the packet has a source port value "*PORT*".
port PORT	True if either the source or destination port of the packet is "*PORT*".
dst portrange PORT1-PORT2	True if the packet is UDP or TCP and has a destination port value is in range "*PORT1-PORT2*".
src portrange PORT1-PORT2	True if the packet has a source port value is in range "*PORT1-PORT2*".
portrange PORT1-PORT2	True if either the source or destination port of the packet is in range "*PORT1-PORT2*".

> ⚠️ **NOTE**
>
> Any of the above "*port*" or "*port range*" expressions can be prefixed with the keywords "*tcp*" or "*udp*", in this case, the filtration will be performed also according to the protocol value.

less LENGTH	True if the packet has a length less than or equal to "*LENGTH*". This is equivalent to: "*len <= length*".
greater LENGTH	True if the packet has a length greater than or equal to *"LENGTH"*. This is equivalent to: "*len >= length*".
ip proto PROTOCOL	True if the packet is IPv4 packet, and contains protocol header with type *"PROTOCOL"*. "*PROTOCOL"* - can be a number or one of the names: *"icmp", "icmp6", "igmp", "igrp", "pim", "ah", "esp", "vrrp", "udp"* or *"tcp"*. Note that the identifiers "*tcp*", "*udp*" and *"icmp"* are also keywords and must be escaped via backslash (\). Note that this primitive does not chase the protocol header chain.

ip protochain PROTOCOL	True if the packet is IPv4 packet, and contains protocol header with type "*PROTOCOL"* in its protocol header chain.
ether broadcast	True if the packet is an Ethernet broadcast packet. The "*ether*" is optional.
ether multicast	True if the packet is an Ethernet multicast (or broadcast) packet. The "*ether"* is optional. This is shorthand for "*ether[0] & 1 != 0*".
ip multicast	True if the packet is an IPv4 multicast (or broadcast) packet.
ether proto PROTOCOL	True if the packet has ether type "*PROTOCOL*". "*PROTOCOL"* can be a number or one of the names: *"icmp", "icmp6", "igmp", "igrp", "pim", "ah", "esp", "vrrp", "udp"* or *"tcp".* Note these identifiers are also keywords and must be escaped via backslash (\).
svlan [vlan _id]	True if the packet is an IEEE 802.1Q Service VLAN packet (ether proto 0x88a8). In the case of Ethernet, WANFleX checks the Ethernet type field for most of those protocols. The exceptions are: • *"iso", "stp"* and *"netbeui"* - WANFLeX checks for an 802.3 frame and then checks the LLC header as it does for FDDI, Token Ring, and 802.11. • *"atalk"* - WANFLeX checks both for the AppleTalk etype in an Ethernet frame and for a SNAP-format packet as it does for FDDI, Token Ring, and 802.11. • *"aarp"* - WANFLeX checks for the AppleTalk ARP etype in either an Ethernet frame or an 802.2 SNAP frame with an OUI of 0x000000. • *"ipx"* - WANFLeX checks for the IPX etype in an Ethernet frame, the IPX DSAP in the LLC header, the 802.3-with-no-LLC-header encapsulation of IPX, and the IPX etype in a SNAP frame.
vlan [vlan_ id]	True if the packet is an IEEE 802.1Q VLAN packet (ether proto 0x8100). If "*[vlan_id]",* is specified, only true if the packet has the specified "*vlan_id".* ⚠ **NOTE** The "*vlan [vlan_id]*" expression may be used more than once, to filter on VLAN hierarchies. Each use of that expression increments the filter offsets by 4.
mpls [label _num]	True if the packet is an MPLS packet. If "*[label_num]",* is specified, only true is the packet has the specified "*label_num".* ⚠ **NOTE** The "*mpls [label_num]"* expression may be used more than once, to filter on MPLS hierarchies. Each use of that expression increments the filter offsets by 4.
pppoed	True if the packet is a PPP-over-Ethernet Discovery packet (Ethernet type 0x8863).
pppoes	True if the packet is a PPP-over-Ethernet Session packet (Ethernet type 0x8864).
iso proto PROTOCOL	True if the packet is an OSI packet of protocol type "*PROTOCOL".* Protocol can be a number or one of the names: "*clnp", "esis", "isis".*

⚠

Title

expr relop expr	True if the relation holds, where "*relop*" is one of ">", "<", ">=", "<=", "=", "!=", and "*expr*" is an arithmetic expression composed of integer constants "+", "-", "*", "/", "&", "|", "<<", ">>". ⚠ **NOTE** Note that all comparisons are unsigned, so that, for example, 0x80000000 and 0xffffffff are > 0. To access data inside the packet, use the following syntax: *"proto* [*expr* : *size*]". • *"proto"* is one of *"ether", "fddi", "tr", "wlan", "ppp", "slip", "link", "ip", "arp", "rarp", "tcp", "udp", "icmp"* and indicates the protocol layer for the index operation. Values *"ether", "fddi", "tr", "wlan", "ppp", "slip", "link"* refer to the link layer. Note that "*tcp", "udp"* and other upper-layer protocol types only apply to IPv4. • "*size"* is optional and indicates the number of bytes in the field of interest; it can be either 1, 2 or 4, by default is 1. The length operator, indicated by the keyword "*len*". Some offsets and field values may be expressed as names rather than as numeric values. The following protocol header field offsets are available: "*icmptype"* (ICMP type field), "*icmpcode"* (ICMP code field) and "*tcpflags"* (TCP flags field): • The following ICMP type field values are available: "*icmp-echoreply", "icmp-unreach", "icmp-sourcequench", "icmp-redirect", "icmp-echo", "icmp-routeradvert", "icmp-routersolicit", "icmp-timxceed", "icmp-paramprob", "icmp-tstamp", "icmp-tstampreply", "icmp-ireq", "icmp-ireqreply", "icmp-maskreq", "icmp-maskreply"*. • The following TCP flags field values are available: *"tcp-fin", "tcp-syn", "tcp-rst", "tcp-push", "tcp-ack", "tcp-urg"*.

Examples

Filtration prohibits the incoming traffic which data belongs to the port 80 ("*udp*" or "*tcp*"). In this example, the full "*ipfw*" command syntax is used, in the following examples, the command parameters will be omitted.

```
ipfw add reject -f "port 80"
```

If the filter has several identical repeating classifiers, they can be specified once, to shorten the record.

```
net 192.168.0.0/24 and (tcp port 21 or tcp port 20 or tcp port 25 or tcp port 80 or tcp port
110)
```

is equal to:

```
net 192.168.0.0/24 and (tcp port 21 or 20 or 25 or 80 or 110)
```

Discards packets that have "*1.1.1.1*" and "*1.1.1.2*" IP-addresses.

```
not (host 1.1.1.1 and host 1.1.1.2)
```

is equal to:

```
not (host 1.1.1.1 and 1.1.1.2)
```

should not be confused with:

```
not host 1.1.1.1 and 1.1.1.2
```

In this case, packets that do not have the first IP-address and have the second one will be skipped.
Following shortening is also not permitted:

```
not (host 1.1.1.1 or 1.1.1.2)
```

In this case, packets with at least one of the specified IP-addresses will be discarded.

Traffic filtration, which has the "*192.168.0.1*" IP-address (source or destination).

```
host 192.168.0.1
```

Traffic filtration, which has the destination IP-address belongs to "*172.16.0.0/16*" network (more precisely, is in range from "*172.16.0.0*" to "*172.16.255.255*").

```
dst net 172.16.0.0/16
```

Traffic filtration, which belongs to "*192.168.0.0/24*" network (source or destination), using TCP protocol and port 21.

```
net 192.168.0.0/24 and tcp port 21
```

Multicast traffic filtration.

```
ether[0] & 1 != 0
```

IPv4 packets filtration.

```
ip[0] & 0xf != 5
```

Catches only unfragmented IPv4 datagrams and discards fragmented IPv4 datagrams. This check is implicitly applied to the tcp and udp index operations. The *"tc p[0]"* always means the first byte of the TCP header, and never means the intervening fragment first byte.

```
ip[6:2] & 0x1fff = 0
```

Filters VLAN 200 encapsulated within Service VLAN 100.

```
svlan 100 && vlan 200
```

Filters IPv4 protocols encapsulated in VLAN 300.

```
vlan 300 && ip
```

Filters all packets encapsulated within Service VLAN 100.

```
svlan 100
```

Filters packets with an outer label 100000 and an inner label 1024.

```
mpls 100000 && mpls 1024
```

Filters packets to or from 192.9.200.1 with an inner label of 1024 and any outer label.

```
mpls && mpls 1024 && host 192.9.200.1
```

Filters IPv4 protocols encapsulated in PPPoE.

```
pppoes && ppp proto 0x21
```