DisplayFilters

Wireshark uses display filters for general packet filtering while viewing and for its ColoringRules.

The basics and the syntax of the display filters are described in the User's Guide.

The master list of display filter protocol **fields** can be found in the display filter reference.

If you need a display filter for a specific protocol, have a look for it at the ProtocolReference.

Examples

Show only SMTP (port 25) and ICMP traffic:

```
tcp.port eq 25 or icmp
```

Show only traffic in the LAN (192.168.x.x), between workstations and servers -- no Internet:

```
ip.src==192.168.0.0/16 and ip.dst==192.168.0.0/16
```

TCP buffer full -- *Source is instructing Destination to stop sending data*

```
tcp.window_size == 0 && tcp.flags.reset != 1
```

Filter on Windows -- *Filter out noise, while watching Windows Client - DC exchanges*

```
smb || nbns || dcerpc || nbss || dns
```

Sasser worm: --*What sasser really did*--

```
ls_ads.opnum==0x09
```

Match packets containing the (arbitrary) 3-byte sequence 0x81, 0x60, 0x03 at the beginning of the UDP payload, skipping the 8-byte UDP header. Note that the values for the byte sequence implicitly are in hexadecimal only. *(Useful for matching homegrown packet protocols.)*

```
udp[8:3]==81:60:03
```

The "slice" feature is also useful to filter on the vendor identifier part (OUI) of the MAC address, see the Ethernet page for details. Thus you may restrict the display to only packets from a specific device manufacturer. E.g. for DELL machines only:

```
eth.addr[0:3]==00:06:5B
```

It is also possible to search for characters appearing anywhere in a field or protocol by using the matches operator.

Match packets that contains the 3-byte sequence 0x81, 0x60, 0x03 anywhere in the UDP header or payload:

```
udp contains 81:60:03
```

Match packets where SIP To-header contains the string "a1762" anywhere in the header:

```
sip.To contains "a1762"
```

The matches operator makes it possible to search for text in string fields and byte sequences using a regular expression, using Perl regular expression syntax. Note: Wireshark needs to be built with libpcre in order to be able to use the matches operator.

Match HTTP requests where the last characters in the uri are the characters "gl=se":

```
http.request.uri matches "gl=se$"
```

Note: The $ character is a PCRE punctuation character that matches the end of a string, in this case the end of http.request.uri field.

Filter by a protocol (e.g. SIP) and filter out unwanted IPs:

```
ip.src != xxx.xxx.xxx.xxx && ip.dst != xxx.xxx.xxx.xxx && sip
```

[Feel free to contribute more]

Gotchas

Some *filter fields* match against multiple *protocol fields*. For example, "ip.addr" matches against both the IP source and destination addresses in the IP header. The same is true for "tcp.port", "udp.port", "eth.addr", and others. It's important to note that

```
ip.addr == 10.43.54.65
```

is equivalent to

```
ip.src == 10.43.54.65 or ip.dst == 10.43.54.65
```

This can be counterintuitive in some cases. Suppose we want to filter out any traffic to or from 10.43.54.65. We might try the following:

```
ip.addr != 10.43.54.65
```

which is equivalent to

```
ip.src != 10.43.54.65 or ip.dst != 10.43.54.65
```

This translates to "pass all traffic except for traffic with a source IPv4 address of 10.43.54.65 **and** a destination IPv4 address of 10.43.54.65", which isn't what we wanted.

Instead we need to negate the expression, like so:

```
! ( ip.addr == 10.43.54.65 )
```

which is equivalent to

```
! (ip.src == 10.43.54.65 or ip.dst == 10.43.54.65)
```

This translates to "pass any traffic except with a source IPv4 address of 10.43.54.65 **or** a destination IPv4 address of 10.43.54.65", which is what we wanted.

See Also

CaptureFilters

External Links

- Displaying HTTP traffic to debug Apache
- Capture and display filter Cheat sheets

DisplayFilters (2008-10-20 16:35:09由GeraldCombs编辑)