

# Tech Exploration 4 - Wireshark Filters

Author: Anthony Cavalieri

# About Wireshark

Wireshark is an essential tool in any networking based career, but especially in security. Created in the late 1990's, Wireshark is free and open-source, so that all may use it. By capturing data on the network, and making it human readable, users can easily find, understand, and solve problems on the network. While Wireshark is a very powerful tool in many aspects, filters are arguably the most powerful aspect of the program. This tutorial will cover 4 essential filters, and show the user how to user and/or read them.

## Prerequisites

Wireshark can be used on almost any device, and can be used with either a GUI or through the command line. This tutorial will be using the GUI, but can be preformed on any supported OS.

- Supported OS (Windows, Mac, Linux)
- Installation of Wireshark
- Working NIC

## Learning Objectives

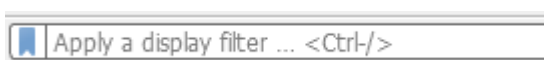
- Improve on using Wireshark so that the learner can be more effective in future positions after graduation.
- Teach classmates functions that they may not know, so that they may be better suited for future use of the tool.
- Be able to apply and understand filters that are applied to a Wireshark session, being able to customize the filters to fit the user's specific needs.

## Instructions

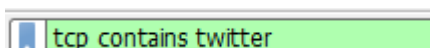
### TCP/UDP Contains

The 'contains' filter is a very simple, yet effective filter, which allows you to search for any plaintext word in a TCP or UDP steam in Wireshark. For example, to search for web traffic related to Twitter, you would do the following:

1. Step 1. In the 'Add a Display Filter' search bar, type in 'tcp contains twitter'. TCP can be replaced with UDP, and twitter can be replaced with anything that you'd like to filter for.



1. Step 2. Hit 'Enter' or the arrow button to the right of the search bar to apply the filter.



1. Step 3. View the results of the filter. If nothing shows up, navigate to the website, and browse for a small amount of time.

tcp contains twitter						
No.	Time	Source	Destination	Protocol	Length	Info
533	20.968791	192.168.1.120	104.244.42.130	TLSv1.2	571	Client Hello
542	21.025741	192.168.1.120	104.244.42.65	TLSv1.2	571	Client Hello
568	21.158633	192.168.1.120	104.244.42.130	TLSv1.2	571	Client Hello
15733	109.517500	192.168.1.120	104.244.42.3	TLSv1	571	Client Hello
15800	109.549815	104.244.42.3	192.168.1.120	TLSv1.2	1514	Server Hello
15940	109.624050	192.168.1.120	104.244.42.3	TLSv1.2	571	Client Hello
15944	109.657943	104.244.42.3	192.168.1.120	TLSv1.2	1514	Server Hello

In the results at the bottom of Wireshark, you will be able to see some of the data within the captured traffic. You can see here that this filter worked, as the data shows 'twitter' in the results.

0030	01 00 38 8c 00 00 16 03	01 02 00 01 00 01 fc 03	..8.....
0040	03 f4 ef 6c 16 ce 6a ca	86 7f 03 48 c4 0b 3a 44	..1..j..H..:D
0050	76 1b 92 42 a5 aa 38 d4	37 79 7e 86 0b f9 0e 07	v..B..8..7y~.....
0060	d4 20 f5 cc 9e 1a 48 dd	77 2e 4b d7 b0 42 b3 ae	..H..w.K..B..
0070	4d 14 69 70 da 01 be 27	0a 1d b3 cf 20 48 f1 8d	M-ip... ' ..H..
0080	1f 1b 00 22 6a 6a 13 01	13 02 13 03 c0 2b c0 2f	..."jj.. ..+.. /
0090	c0 2c c0 30 cc a9 cc a8	c0 13 c0 14 00 9c 00 9d	.,0.....
00a0	00 2f 00 35 00 0a 01 00	01 91 6a 6a 00 00 00 00	./5....jj....
00b0	00 14 00 12 00 00 0f 61	70 69 2e 74 77 69 74 74	.....a pi.twitt
00c0	65 72 2e 63 6f 6d 00 17	00 00 ff 01 00 01 00 00	er.com.. ..
00d0	0a 00 0a 00 08 5a 5a 00	1d 00 17 00 18 00 0b 00	....ZZ. ....
00e0	02 01 00 00 23 00 00 00	10 00 0e 00 0c 02 68 32	....#... ..h2
00f0	08 68 74 74 70 2f 31 2e	31 00 05 00 05 01 00 00	http/1. 1.....
0100	00 00 00 0d 00 14 00 12	04 03 08 04 04 01 05 03	.....
0110	08 05 05 01 08 06 06 01	02 01 00 12 00 00 00 33	.....3
0120	00 2b 00 29 5a 5a 00 01	00 00 1d 00 20 c6 8d 51	+.)ZZ.. ..Q

## !(arp or icmp or dns)

This filter will discard arp, icmp, and dns requests sent on the network. These results are usually the bulk of your results, but are not always helpful, and can just take up space. By applying this filter you will be able to view more important traffic, without having to dig through all of these requests.

1. Step 1. In the 'Add a Display Filter' search bar, type in '!(arp or icmp or dns)'.

Apply a display filter ... <Ctrl-/>

1. Step 2. Hit 'Enter' or the arrow button to the right of the search bar to apply the filter.

!(arp or icmp or dns)

1. Step 3. View the results of the filter. You will likely still have quite a few results, but will not see any of the specified protocols in the Protocols column in the results pane. You can also add more exclusions by adding more to the protocols in the parenthesis.

!(arp or icmp or dns)						
No.	Time	Source	Destination	Protocol	Length	Info
68659	586.696161	104.16.59.37	192.168.1.120	TLSv1.2	131	Application Data
68660	586.737052	192.168.1.120	104.16.59.37	TCP	54	49826 → 443 [ACK] Seq=796 Ack=15855 Win=251 Len=0
68661	587.137120	192.168.1.1	239.255.255.250	IGMPv2	42	Membership Query, specific for group 239.255.255.250
68662	587.232547	104.16.59.37	192.168.1.120	TLSv1.2	151	Application Data
68663	587.273286	192.168.1.120	104.16.59.37	TCP	54	49826 → 443 [ACK] Seq=796 Ack=15952 Win=251 Len=0
68664	587.590789	192.168.1.120	162.254.193.47	UDP	126	58484 → 27018 Len=84

## ip.addr ==

This filter allows you to view traffic for a specific IP address, which can help immensely when trying to view specific traffic on the network.

1. Step 1. In the 'Add a Display Filter' search bar, type in 'ip.addr == x.x.x.x', replacing the x's with the desired IP address.

Apply a display filter ... <Ctrl-/>

1. Step 2. Hit 'Enter' or the arrow button to the right of the search bar to apply the filter.

ip.addr == 127.0.0.1

1. Step 3. View the results of the filter.

ip.addr == 35.24.63.98						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	35.24.63.98	104.244.42.67	TCP	55	51893 → 443 [ACK] Seq=1 Ack=1 Win=252 Len=1 [TCP segment of a reassembled PDU]
2	0.090248	104.244.42.67	35.24.63.98	TCP	66	443 → 51893 [ACK] Seq=1 Ack=2 Win=132 Len=0 SLE=1 SRE=2
3	1.425244	35.24.63.98	162.254.193.6	UDP	126	52280 → 27019 Len=84
4	4.146509	35.24.63.98	172.217.8.14	GQUIC	1392	Client Hello, PKN: 1, CID: 15370318909630539734
5	4.183209	172.217.8.14	35.24.63.98	GQUIC	1392	Payload (Encrypted), PKN: 1
6	4.183210	172.217.8.14	35.24.63.98	GQUIC	73	Payload (Encrypted), PKN: 2
7	4.184297	35.24.63.98	172.217.8.14	GQUIC	81	Payload (Encrypted), PKN: 2, CID: 15370318909630539734
8	4.184582	35.24.63.98	172.217.8.14	GQUIC	70	Payload (Encrypted), PKN: 3, CID: 15370318909630539734
9	4.185083	35.24.63.98	172.217.8.14	GQUIC	1392	Payload (Encrypted), PKN: 4, CID: 15370318909630539734
10	4.185210	35.24.63.98	172.217.8.14	GQUIC	749	Payload (Encrypted), PKN: 5, CID: 15370318909630539734
11	4.212468	172.217.8.14	35.24.63.98	GQUIC	62	Payload (Encrypted), PKN: 3
12	4.237011	172.217.8.14	35.24.63.98	GQUIC	62	Payload (Encrypted), PKN: 4
13	4.285330	172.217.8.14	35.24.63.98	GQUIC	374	Payload (Encrypted), PKN: 5
14	4.285331	172.217.8.14	35.24.63.98	GQUIC	446	Payload (Encrypted), PKN: 6
15	4.285332	172.217.8.14	35.24.63.98	GQUIC	70	Payload (Encrypted), PKN: 7

This filter can be further specified into source and destination IP addresses. You can do this by using the filters 'ip.src == ', and 'ip.dst == ', as shown below.

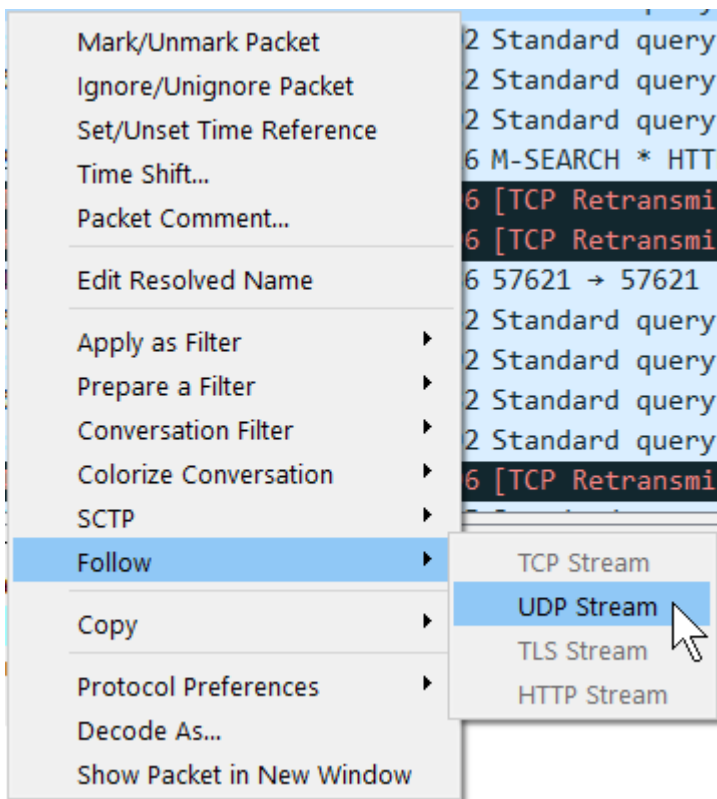
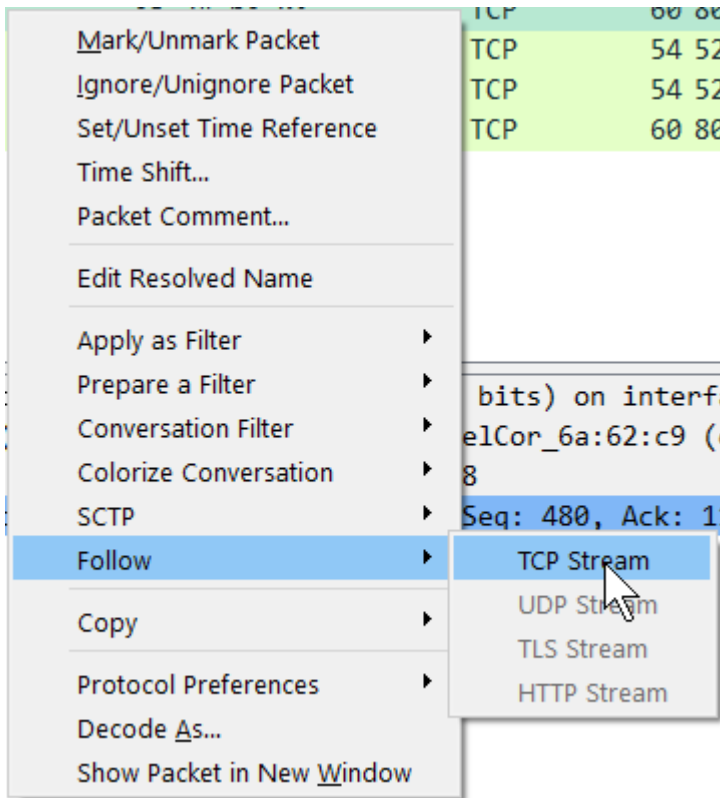
ip.src == 35.24.63.98

ip.dst == 35.24.63.98

## tcp.stream

This filter allows you to piece together a string of packets to view the data that they contain. This will work better with HTTP traffic, as opposed to HTTPS, as the encryption makes the data unreadable.

1. Step 1. This filter can be used with manual filter entry, however, this is much more complicated than just right clicking a packet and selecting 'Follow TCP/UDP Stream'. Note: There are also options for HTTP and TLS streams, but the majority of the packets captured are going to be TCP or UDP.



1. Step 2. Left click on the TCP/UDP stream option.
2. Step 3. View the results of the TCP/UDP conversation.

835	187.116396	35.24.63.98	13.107.4.52	TCP	66 52399 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
836	187.128230	13.107.4.52	35.24.63.98	TCP	66 80 → 52399 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM=1
837	187.128308	35.24.63.98	13.107.4.52	TCP	54 52399 → 80 [ACK] Seq=1 Ack=1 Win=66048 Len=0
838	187.128441	35.24.63.98	13.107.4.52	HTTP	165 GET /connecttest.txt HTTP/1.1
839	187.139445	13.107.4.52	35.24.63.98	TCP	60 80 → 52399 [ACK] Seq=1 Ack=112 Win=262656 Len=0
840	187.143069	13.107.4.52	35.24.63.98	HTTP	533 HTTP/1.1 200 OK (text/plain)
841	187.143070	13.107.4.52	35.24.63.98	TCP	60 80 → 52399 [FIN, ACK] Seq=480 Ack=112 Win=262656 Len=0
842	187.143120	35.24.63.98	13.107.4.52	TCP	54 52399 → 80 [ACK] Seq=112 Ack=481 Win=65536 Len=0
843	187.143162	35.24.63.98	13.107.4.52	TCP	54 52399 → 80 [FIN, ACK] Seq=112 Ack=481 Win=65536 Len=0
844	187.154225	13.107.4.52	35.24.63.98	TCP	60 80 → 52399 [ACK] Seq=481 Ack=113 Win=262656 Len=0

```
GET /connecttest.txt HTTP/1.1
Connection: Close
User-Agent: Microsoft NCSI
Host: www.msftconnecttest.com

HTTP/1.1 200 OK
Cache-Control: no-store
Content-Length: 22
Content-Type: text/plain; charset=utf-8
Last-Modified: Sun, 07 Apr 2019 17:45:22 GMT
Accept-Ranges: bytes
ETag: 0x8D343F9E96C9DAC
Access-Control-Allow-Origin: *
Access-Control-Expose-Headers: X-MSEdge-Ref
Timing-Allow-Origin: *
X-MSEdge-Ref: Ref A: DB23420AFAB44ECCA775B5CBEF6E4A40 Ref B: CHGEDGE0809 Ref C: 2019-04-09T19:36:56Z
Date: Tue, 09 Apr 2019 19:36:56 GMT
Connection: close

Microsoft Connect Test
```

You can see above that the individual packet data gets put into a more human readable form, and is much easier to understand. This TCP conversation was unencrypted, so it is plain text readable.

This *can* be done with HTTPS (encrypted) traffic, but it will almost definitely be unreadable, as shown below.

```
....E...A...0.Nm..
.[K.;V.q7...3...{h....
. ....5./..
...
.....enterprise.nmu.edu.
.....#.....Xh=.....G.....O^.....=\.....1.....~.q...F.....UTHV;3..` j.%.[V;@a4)...9;n-z{"...t.V...@.1.....Z...F....
(G&.T...~...A...86...A...DA...Bh*.z7
...&_...2...P0m.Z..7|.n+f.".....i...*'.]2./..m.eTQ.G
.....1.....=.4Gr@d...e...
L..n.?pw. 0..y.7.....0...8...w*..!N..(?LhxA..A..J..81.9...?....w{.....0..E.....~)...pqJ...di. ....N..1..Z.....G/..W....
_.....JU.:/.....[s...a...u.....P.....k(.X0...v=i.^.....z:W.
.....6+e...j...i...j...1..E8.....I.i..G.8-M.....04i...a...A.....#...A'lg....]..`!c)%..x.HI.....FQXq...^..w.k.....tBuz..)
(...j...f...X.p...c.....X.....11o.&e..Q.-...3...AyJ.^C.&B.6. .Z!V.....%+.6.z...=.....A.t.....0...~.....'~(\ ..Q.....$K.z..
8.y.....`...../:UaQ{$.S.[.H....
...e..1...3...#e..Q..w...!S.....%T.A.....c...{8F.... ..@^.....&...[0..o...J#9....dj...!
```

## Challenge

Go out and try these filters on your own! Use these specific filters, find other ways to use them, or find filters not shown here.

Try this on your home network, see what data you capture, and what you can make of it. You might be sending data you didn't know about.

Go out and try other packet sniffing programs, and compare them to Wireshark:

- Are they better/worse?
- Any features that you can't find in Wireshark?
- Do the same filters for Wireshark work in the other program(s)?

## Disclaimer

Sniffing packets on a network you do not have permission to sniff is illegal, and can get you in serious trouble. This tutorial did not go over promiscuous mode, due to NMU laptop NICs not

supporting it. However, turning on promiscuous mode will allow you to see *all* available traffic on the network. Keep promiscuous mode off, and stay on a home network to ensure you are in compliance with best network practices.

## Reflection

Now that you have learned some useful filters for Wireshark, and have a better understanding of the program, consider the following:

- In what situations could you use Wireshark to find/solve issues on the network?
- What should you do if you find someone using Wireshark (or another packet sniffer) on your network?
- In what situation(s) is turning on promiscuous mode okay?
- What other programs could you use in conjunction with Wireshark to utilize the captured data?