Wireshark
Deep packet inspection with Wireshark

Wireshark is a free and open-source packet analyzer. It is commonly used to troubleshoot network issues and analysis. Originally named Ethereal, in May 2006 the project was renamed Wireshark due to trademark issues.

This article attempts to give some detail into how to search through packet dump files or pcap files using Wireshark. I give some useful information on using wireshark & tshark to do deep packet analysis.

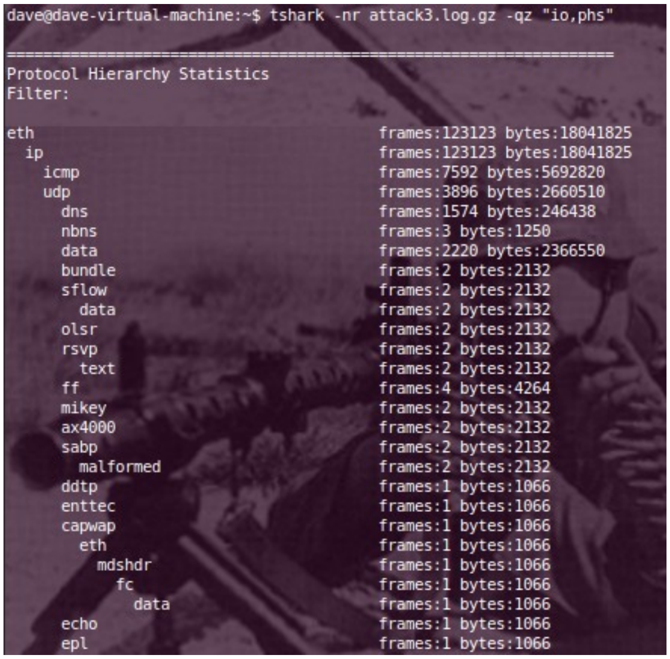
Intrusion detection devices such as Snort use the libpcap C/C++ library for network traffic capture. It is this capture file that we will be using wireshark on.

Wireshark is included in many Linux distros, if it is not; it is available in the package repositories. Wireshark formally known as ethereal is available for download through the project website, which has a number of tutorial and resources.

tshark

The tshark utility allows you to filter the contents of a pcap file from the command line. To view the most significant activity, I use the following command (see Figure #1):

$ tshark –nr attack3.log.gz –qz "io,phs"



```
dave@dave-virtual-machine:~$ tshark -nr attack3.log.gz -qz "io,phs"

================================================================================
Protocol Hierarchy Statistics
Filter:

eth                                      frames:123123 bytes:18041825
  ip                                     frames:123123 bytes:18041825
    icmp                                 frames:7592 bytes:5692820
    udp                                  frames:3896 bytes:2660510
      dns                                frames:1574 bytes:246438
      nbns                               frames:3 bytes:1250
      data                               frames:2220 bytes:2366550
      bundle                             frames:2 bytes:2132
      sflow                              frames:2 bytes:2132
        data                             frames:2 bytes:2132
      olsr                               frames:2 bytes:2132
      rsvp                               frames:2 bytes:2132
        text                             frames:2 bytes:2132
      ff                                 frames:4 bytes:4264
      mikey                              frames:2 bytes:2132
      ax4000                             frames:2 bytes:2132
      sabp                               frames:2 bytes:2132
        malformed                        frames:2 bytes:2132
      ddtp                               frames:1 bytes:1066
      enttec                             frames:1 bytes:1066
      capwap                             frames:1 bytes:1066
        eth                              frames:1 bytes:1066
          mdshdr                         frames:1 bytes:1066
            fc                           frames:1 bytes:1066
              data                       frames:1 bytes:1066
      echo                               frames:1 bytes:1066
      epl                                frames:1 bytes:1066
```

Figure 1: ***Tshark statictics output***

The –n switch disables network object name resolution, -r indicates that packet data is to be read from the input file, in this case attack3.log.gz. The –z allows for statistics to display after it is finished reading the capture file, the –q flag specifies that only the statistics are printed. See Figure 1 for the output of this information. To view a list of help commands used with tshark, type:

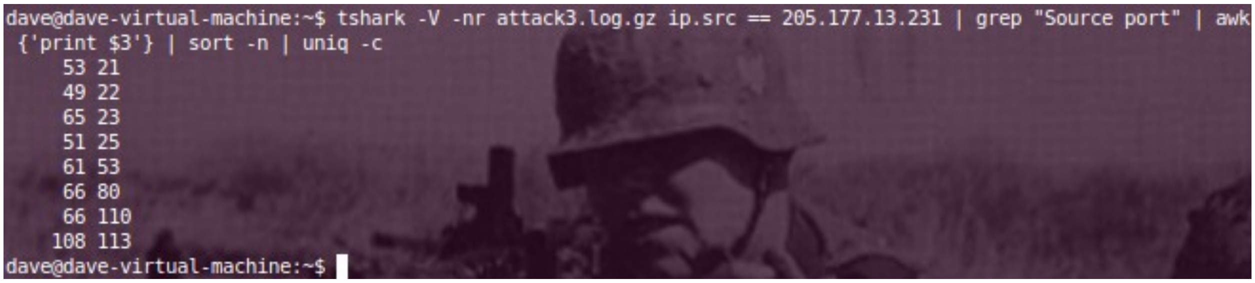
$ tshark –h

For a list of arguments type –z:

$ tshark –z help

If you are looking for a particular IP address [205.177.13.231] that you think may appear in a packet dump and the associated port it is connecting on and the number of times it connected use the following command (See Figure #2):

$ tshark –V –nr attack3.log.gz ip.src == 205.177.13.231 | grep "Source port" | awk {'print $3'} | sort –n | uniq –c



```
dave@dave-virtual-machine:~$ tshark -V -nr attack3.log.gz ip.src == 205.177.13.231 | grep "Source port" | awk
 {'print $3'} | sort -n | uniq -c
     53 21
     49 22
     65 23
     51 25
     61 53
     66 80
     66 110
    108 113
dave@dave-virtual-machine:~$
```

Figure 2: ***List of ports communicating with 205.177.13.231 and the number of times it occurred***

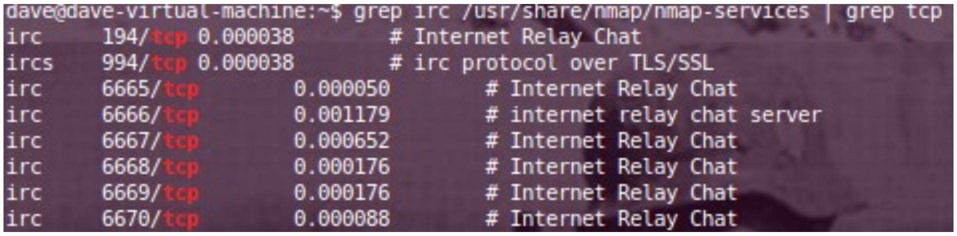
The *–V* causes tshark to print a view of the packet details rather than a one-line summary of the packet. The *grep* command looks for the text string *Source port* in the packet dump, and *awk { 'print $3'}* looks for the third field in the text resulting from the grep and prints it; *sort –n* will sort the results according to string numerical value, and *uniq –c* will take matching lines and merge to the first occurrence and list the number of times that it occurred.

The resulting output shows 205.177.13.231 having connections on ports (21, 22, 23, 25, 53, 80, 110 and 113) along with the number of times each of these occurred.

Let's look to find possible IRC traffic in the packet capture. What are the ports used by IRC traffic? We can issue the following command:

$ grep irc /usr/share/nmap/nmap-services | grep tcp

Figure #3 shows the results of this command.

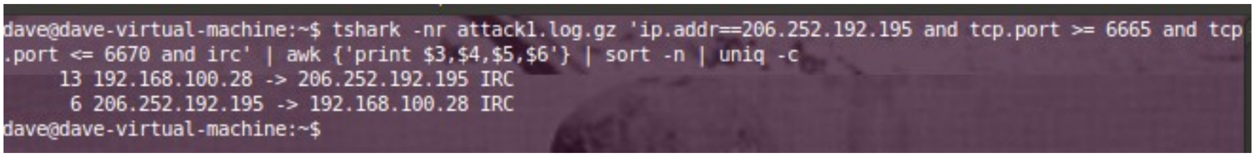


```
dave@dave-virtual-machine:~$ grep irc /usr/share/nmap/nmap-services | grep tcp
irc     194/tcp 0.000038        # Internet Relay Chat
ircs    994/tcp 0.000038        # irc protocol over TLS/SSL
irc     6665/tcp        0.000050        # Internet Relay Chat
irc     6666/tcp        0.001179        # internet relay chat server
irc     6667/tcp        0.000652        # Internet Relay Chat
irc     6668/tcp        0.000176        # Internet Relay Chat
irc     6669/tcp        0.000176        # Internet Relay Chat
irc     6670/tcp        0.000088        # Internet Relay Chat
```

Figure 3: ***Locating IRC port numbers with grep***

When we search the packet dump looking for evidence of IRC traffic to and from IP address 206.252.192.195 we would use the following command (see Figure #4):

$ tshark –nr attack1.log.gz 'ip.addr==206.252.192.195 and tcp.port >= 6665 and tcp.port >= 6670 and irc; | awk {'print $3,$4,$5,$6'} | sort –n | uniq –c



```
dave@dave-virtual-machine:~$ tshark -nr attack1.log.gz 'ip.addr==206.252.192.195 and tcp.port >= 6665 and tcp
.port <= 6670 and irc' | awk {'print $3,$4,$5,$6'} | sort -n | uniq -c
     13 192.168.100.28 -> 206.252.192.195 IRC
      6 206.252.192.195 -> 192.168.100.28 IRC
dave@dave-virtual-machine:~$
```

Figure 4: ***IRC connections found in the packet dump***

Here is the following breakdown of the above command.

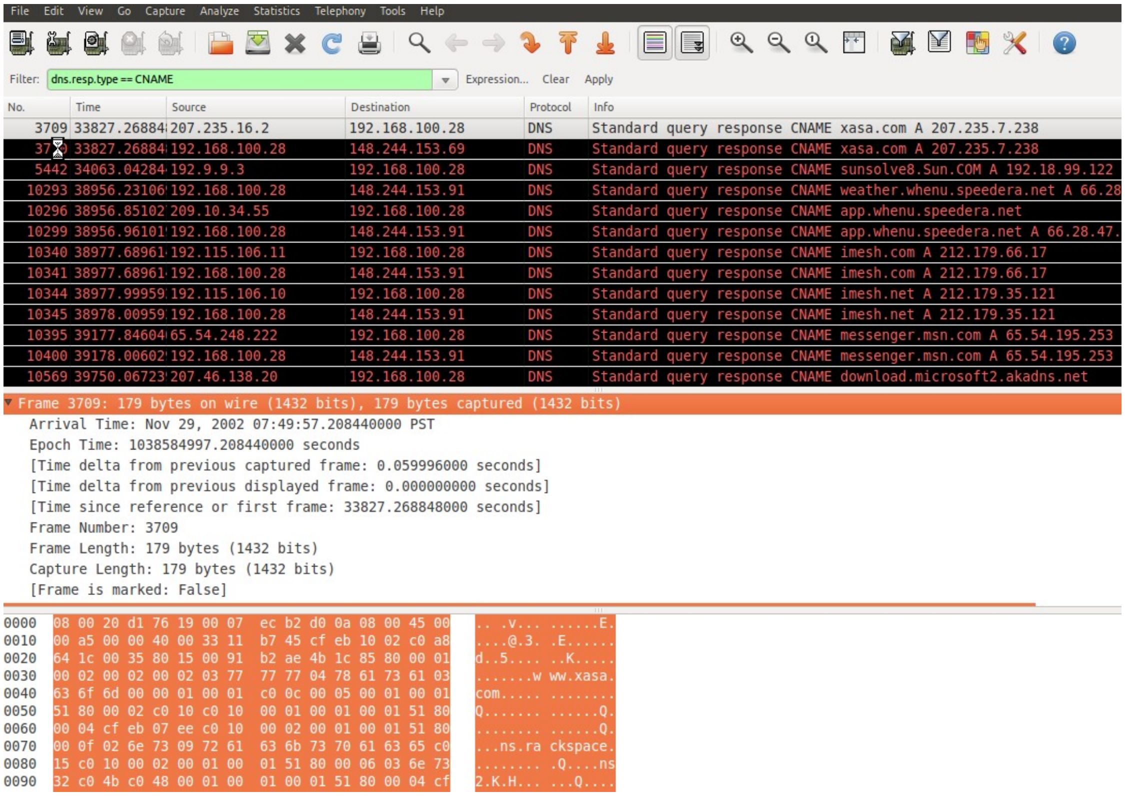
-nr	switch disables network name resolution and packet to be read
'ip.addr==206.252.192.195	This is the IP address that I am looking for
and tcp.port >=6665	Start of the port range
and tcp.port <=6670	End of the port range
and irc'	Search for IRC traffic only
awk {'print $3,$4,$5,$6'}	Prints the third through sixth patterns from each matching line
sort –n	Sorts according to string numerical value
uniq –c	Only prints the number of matches that are unique

Wireshark the GUI

The Wireshark GUI application can be started from the Application menu or from the terminal. To load a capture file from the terminal simply type wireshark filename at the command prompt < *$ wireshark alert1.log.gz*>

The graphical front-end has some integrated sorting and filtering options available. One of them is the Filter box at the top that allows you to enter criteria for the search. To search for all the Canonical Name records within the capture file, type the following filter (see Figure #5):

dns.resp.type == CNAME



Filter: dns.resp.type == CNAME

No.	Time	Source	Destination	Protocol	Info
3709	33827.26884	207.235.16.2	192.168.100.28	DNS	Standard query response CNAME xasa.com A 207.235.7.238
37[illegible]	33827.26884	192.168.100.28	148.244.153.69	DNS	Standard query response CNAME xasa.com A 207.235.7.238
5442	34063.04284	192.9.9.3	192.168.100.28	DNS	Standard query response CNAME sunsolve8.Sun.COM A 192.18.99.122
10293	38956.23106	192.168.100.28	148.244.153.91	DNS	Standard query response CNAME weather.whenu.speedera.net A 66.28
10296	38956.85102	209.10.34.55	192.168.100.28	DNS	Standard query response CNAME app.whenu.speedera.net
10299	38956.96101	192.168.100.28	148.244.153.91	DNS	Standard query response CNAME app.whenu.speedera.net A 66.28.47.
10340	38977.68961	192.115.106.11	192.168.100.28	DNS	Standard query response CNAME imesh.com A 212.179.66.17
10341	38977.68961	192.168.100.28	148.244.153.91	DNS	Standard query response CNAME imesh.com A 212.179.66.17
10344	38977.99959	192.115.106.10	192.168.100.28	DNS	Standard query response CNAME imesh.net A 212.179.35.121
10345	38978.00959	192.168.100.28	148.244.153.91	DNS	Standard query response CNAME imesh.net A 212.179.35.121
10395	39177.84604	65.54.248.222	192.168.100.28	DNS	Standard query response CNAME messenger.msn.com A 65.54.195.253
10400	39178.00602	192.168.100.28	148.244.153.91	DNS	Standard query response CNAME messenger.msn.com A 65.54.195.253
10569	39750.06723	207.46.138.20	192.168.100.28	DNS	Standard query response CNAME download.microsoft2.akadns.net

```
▼ Frame 3709: 179 bytes on wire (1432 bits), 179 bytes captured (1432 bits)
    Arrival Time: Nov 29, 2002 07:49:57.208440000 PST
    Epoch Time: 1038584997.208440000 seconds
    [Time delta from previous captured frame: 0.059996000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 33827.268848000 seconds]
    Frame Number: 3709
    Frame Length: 179 bytes (1432 bits)
    Capture Length: 179 bytes (1432 bits)
    [Frame is marked: False]
```

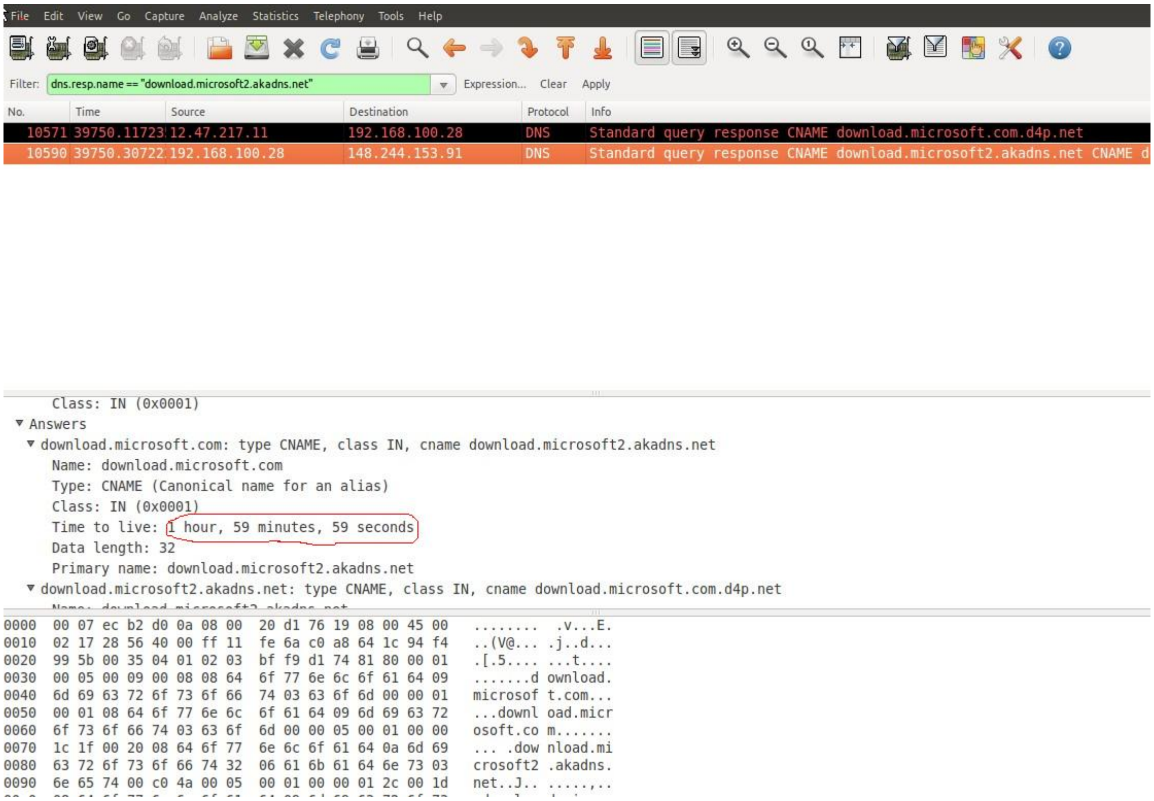
```
0000  08 00 20 d1 76 19 00 07  ec b2 d0 0a 08 00 45 00   .. .v... ......E.
0010  00 a5 00 00 40 00 33 11  b7 45 cf eb 10 02 c0 a8   ....@.3. .E......
0020  64 1c 00 35 80 15 00 91  b2 ae 4b 1c 85 80 00 01   d..5.... ..K.....
0030  00 02 00 02 00 02 03 77  77 77 04 78 61 73 61 03   .......w ww.xasa.
0040  63 6f 6d 00 00 01 00 01  c0 0c 00 05 00 01 00 01   com..... ........
0050  51 80 00 02 c0 10 c0 10  00 01 00 01 00 01 51 80   Q....... ......Q.
0060  00 04 cf eb 07 ee c0 10  00 02 00 01 00 01 51 80   ........ ......Q.
0070  00 0f 02 6e 73 09 72 61  63 6b 73 70 61 63 65 c0   ...ns.ra ckspace.
0080  15 c0 10 00 02 00 01 00  01 51 80 00 06 03 6e 73   ........ .Q....ns
0090  32 c0 4b c0 48 00 01 00  01 00 01 51 80 00 04 cf   2.K.H... ...Q....
```

Figure 5: ***Searching for CNAME records in Wireshark***

After you enter a filter, remember to clear it out before starting a new search.

Now if we wanted to know how long a client resolver cached the IP address associated with the name download.microsoft2.akadns.net (Figure #6), enter the following in the filter:

Dns.resp.name == "download.microsoft2.akadns.net"



Filter: dns.resp.name == "download.microsoft2.akadns.net"

```
No.   Time        Source          Destination     Protocol  Info
10571 39750.11723 12.47.217.11    192.168.100.28  DNS       Standard query response CNAME download.microsoft.com.d4p.net
10590 39750.30722 192.168.100.28  148.244.153.91  DNS       Standard query response CNAME download.microsoft2.akadns.net CNAME d
```

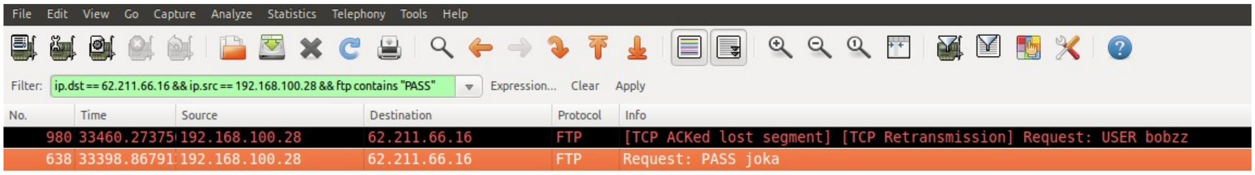
```
      Class: IN (0x0001)
 ▼ Answers
  ▼ download.microsoft.com: type CNAME, class IN, cname download.microsoft2.akadns.net
      Name: download.microsoft.com
      Type: CNAME (Canonical name for an alias)
      Class: IN (0x0001)
      Time to live: 1 hour, 59 minutes, 59 seconds
      Data length: 32
      Primary name: download.microsoft2.akadns.net
  ▼ download.microsoft2.akadns.net: type CNAME, class IN, cname download.microsoft.com.d4p.net
```

```
0000  00 07 ec b2 d0 0a 08 00  20 d1 76 19 08 00 45 00   ........  .v...E.
0010  02 17 28 56 40 00 ff 11  fe 6a c0 a8 64 1c 94 f4   ..(V@... .j..d...
0020  99 5b 00 35 04 01 02 03  bf f9 d1 74 81 80 00 01   .[.5.... ...t....
0030  00 05 00 09 00 08 08 64  6f 77 6e 6c 6f 61 64 09   .......d ownload.
0040  6d 69 63 72 6f 73 6f 66  74 03 63 6f 6d 00 00 01   microsof t.com...
0050  00 01 08 64 6f 77 6e 6c  6f 61 64 09 6d 69 63 72   ...downl oad.micr
0060  6f 73 6f 66 74 03 63 6f  6d 00 00 05 00 01 00 00   osoft.co m.......
0070  1c 1f 00 20 08 64 6f 77  6e 6c 6f 61 64 0a 6d 69   ... .dow nload.mi
0080  63 72 6f 73 6f 66 74 32  06 61 6b 61 64 6e 73 03   crosoft2 .akadns.
0090  6e 65 74 00 c0 4a 00 05  00 01 00 00 01 2c 00 1d   net..J.. .....,..
```

Figure 6: ***Length of time client resolved address cache***

If we wanted to find the user name and password for an FTP account that someone was accessing and we knew that there was a connection somewhere in the packet dump, how would we find it? The information we have is the source and destination [62.211.66.16 & 192.168.100.22]. We would enter in the filter field the following (see Figure #7):

ip.dst == 62.211.66.16 && ip.src == 192.168.100.22 && ftp contains "PASS"



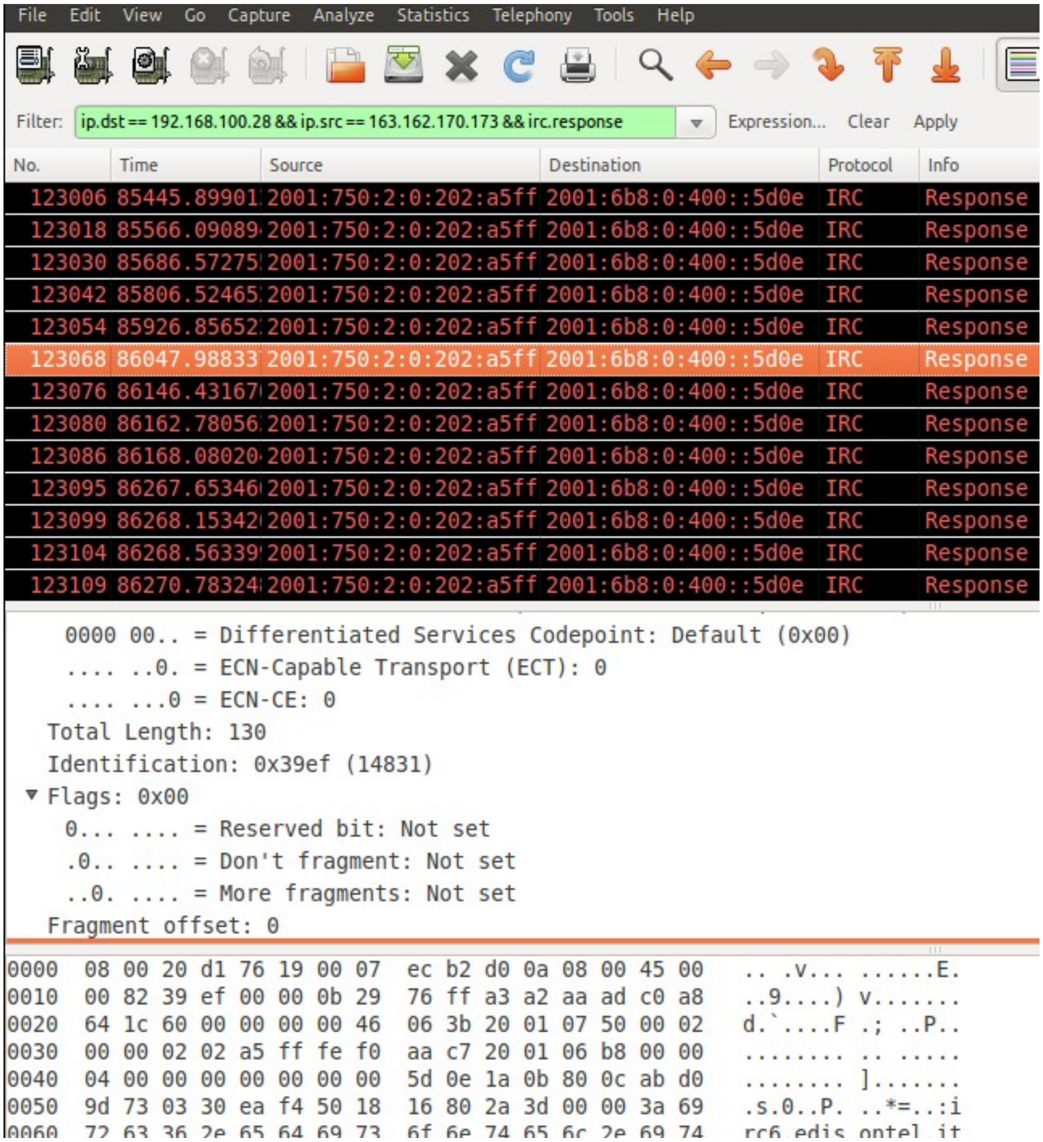
Filter: ip.dst == 62.211.66.16 && ip.src == 192.168.100.28 && ftp contains "PASS"

```
No.  Time        Source          Destination   Protocol  Info
980  33460.27375 192.168.100.28  62.211.66.16  FTP       [TCP ACKed lost segment] [TCP Retransmission] Request: USER bobzz
638  33398.86791 192.168.100.28  62.211.66.16  FTP       Request: PASS joka
```

Figure 7: ***Locating the user name and password for FTP account***

To locate and find the conversation someone had on an IRC chan between source IP 192.168.100.28 and IP destination 163.162.170.173 use the following filter (see Figure #8):

ip.dst == 192.168.100.28 && ip.src == 163.162.170.173 && irc.response


File Edit View Go Capture Analyze Statistics Telephony Tools Help
Expression... Clear Apply

Filter: ip.dst == 192.168.100.28 && ip.src == 163.162.170.173 && irc.response

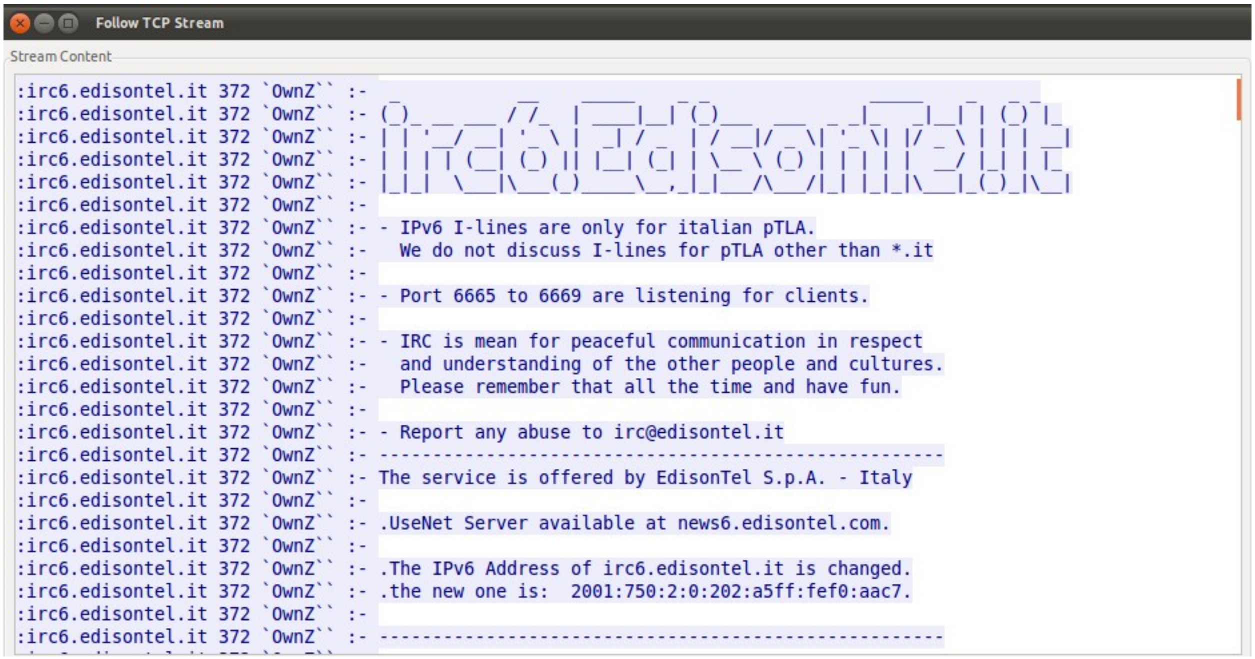
```
No.    Time        Source              Destination          Protocol Info
123006 85445.89901 2001:750:2:0:202:a5ff 2001:6b8:0:400::5d0e IRC   Response
123018 85566.09089 2001:750:2:0:202:a5ff 2001:6b8:0:400::5d0e IRC   Response
123030 85686.57275 2001:750:2:0:202:a5ff 2001:6b8:0:400::5d0e IRC   Response
123042 85806.52465 2001:750:2:0:202:a5ff 2001:6b8:0:400::5d0e IRC   Response
123054 85926.85652 2001:750:2:0:202:a5ff 2001:6b8:0:400::5d0e IRC   Response
123068 86047.98833 2001:750:2:0:202:a5ff 2001:6b8:0:400::5d0e IRC   Response
123076 86146.43167 2001:750:2:0:202:a5ff 2001:6b8:0:400::5d0e IRC   Response
123080 86162.78056 2001:750:2:0:202:a5ff 2001:6b8:0:400::5d0e IRC   Response
123086 86168.08020 2001:750:2:0:202:a5ff 2001:6b8:0:400::5d0e IRC   Response
123095 86267.65346 2001:750:2:0:202:a5ff 2001:6b8:0:400::5d0e IRC   Response
123099 86268.15342 2001:750:2:0:202:a5ff 2001:6b8:0:400::5d0e IRC   Response
123104 86268.56339 2001:750:2:0:202:a5ff 2001:6b8:0:400::5d0e IRC   Response
123109 86270.78324 2001:750:2:0:202:a5ff 2001:6b8:0:400::5d0e IRC   Response
```

```
    0000 00.. = Differentiated Services Codepoint: Default (0x00)
    .... ..0. = ECN-Capable Transport (ECT): 0
    .... ...0 = ECN-CE: 0
  Total Length: 130
  Identification: 0x39ef (14831)
▼ Flags: 0x00
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
  Fragment offset: 0
```

```
0000  08 00 20 d1 76 19 00 07  ec b2 d0 0a 08 00 45 00   .. .v... ......E.
0010  00 82 39 ef 00 00 0b 29  76 ff a3 a2 aa ad c0 a8   ..9....) v.......
0020  64 1c 60 00 00 00 00 46  06 3b 20 01 07 50 00 02   d.`....F .; ..P..
0030  00 00 02 02 a5 ff fe f0  aa c7 20 01 06 b8 00 00   ........ .. .....
0040  04 00 00 00 00 00 00 00  5d 0e 1a 0b 80 0c ab d0   ........ ].......
0050  9d 73 03 30 ea f4 50 18  16 80 2a 3d 00 00 3a 69   .s.0..P. ..*=..:i
0060  72 63 36 2e 65 64 69 73  6f 6e 74 65 6c 2e 69 74   rc6.edis ontel.it
```

Figure 8: ***IRC communication between 192.168.100.28 & 163.162.170.173***

Now pick one of the packets and right click and Follow TCPStream and this will produce the conversation (see Figure #9).



```
:irc6.edisontel.it 372 `OwnZ`` :-  _          __   _____     _ _              _____    _   _ _
:irc6.edisontel.it 372 `OwnZ`` :- (_)_ __ ___/ /_ | ____|__| (_)___  ___  _ _|_   _|__| | (_) |_
:irc6.edisontel.it 372 `OwnZ`` :- | | '__/ __| '_ \|  _| / _` | / __|/ _ \| '_ \| |/ _ \ | | | __|
:irc6.edisontel.it 372 `OwnZ`` :- | | | | (__| (_) | |__| (_| | \__ \ (_) | | | | |  __/ |_| | |_
:irc6.edisontel.it 372 `OwnZ`` :- |_|_|  \___|\___/|_____\__,_|_|___/\___/|_| |_|_|\___|_(_)_|\__|
:irc6.edisontel.it 372 `OwnZ`` :-
:irc6.edisontel.it 372 `OwnZ`` :- - IPv6 I-lines are only for italian pTLA.
:irc6.edisontel.it 372 `OwnZ`` :-   We do not discuss I-lines for pTLA other than *.it
:irc6.edisontel.it 372 `OwnZ`` :-
:irc6.edisontel.it 372 `OwnZ`` :- - Port 6665 to 6669 are listening for clients.
:irc6.edisontel.it 372 `OwnZ`` :-
:irc6.edisontel.it 372 `OwnZ`` :- - IRC is mean for peaceful communication in respect
:irc6.edisontel.it 372 `OwnZ`` :-   and understanding of the other people and cultures.
:irc6.edisontel.it 372 `OwnZ`` :-   Please remember that all the time and have fun.
:irc6.edisontel.it 372 `OwnZ`` :-
:irc6.edisontel.it 372 `OwnZ`` :- - Report any abuse to irc@edisontel.it
:irc6.edisontel.it 372 `OwnZ`` :- -----------------------------------------------------
:irc6.edisontel.it 372 `OwnZ`` :- The service is offered by EdisonTel S.p.A. - Italy
:irc6.edisontel.it 372 `OwnZ`` :-
:irc6.edisontel.it 372 `OwnZ`` :- .UseNet Server available at news6.edisontel.com.
:irc6.edisontel.it 372 `OwnZ`` :-
:irc6.edisontel.it 372 `OwnZ`` :- .The IPv6 Address of irc6.edisontel.it is changed.
:irc6.edisontel.it 372 `OwnZ`` :- .the new one is:  2001:750:2:0:202:a5ff:fef0:aac7.
:irc6.edisontel.it 372 `OwnZ`` :-
:irc6.edisontel.it 372 `OwnZ`` :- -----------------------------------------------------
```

Figure 9: ***IRC conversation between 192.168.100.28 & 163.162.170.173***

Conclusion

Wireshark is a powerful tool used to search through packet dumps to locate clues about nefarious activity.