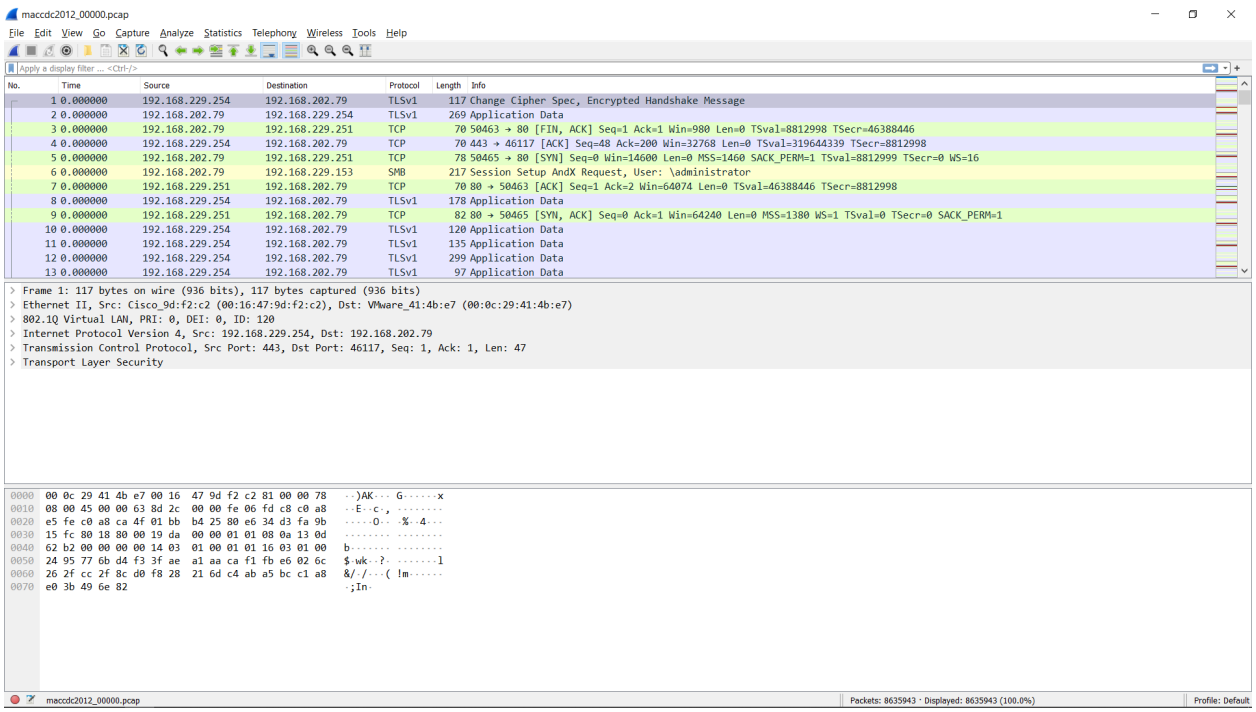
Mikias Berhanu
2021280115
Assignment Submission VI

Analysing Network Traffic Using Wireshark and Python

Wireshark is a free and open source tool which is used for network troubleshooting testing and network traffic analysis. The first name was Ethereal which was later renamed to wireshark.

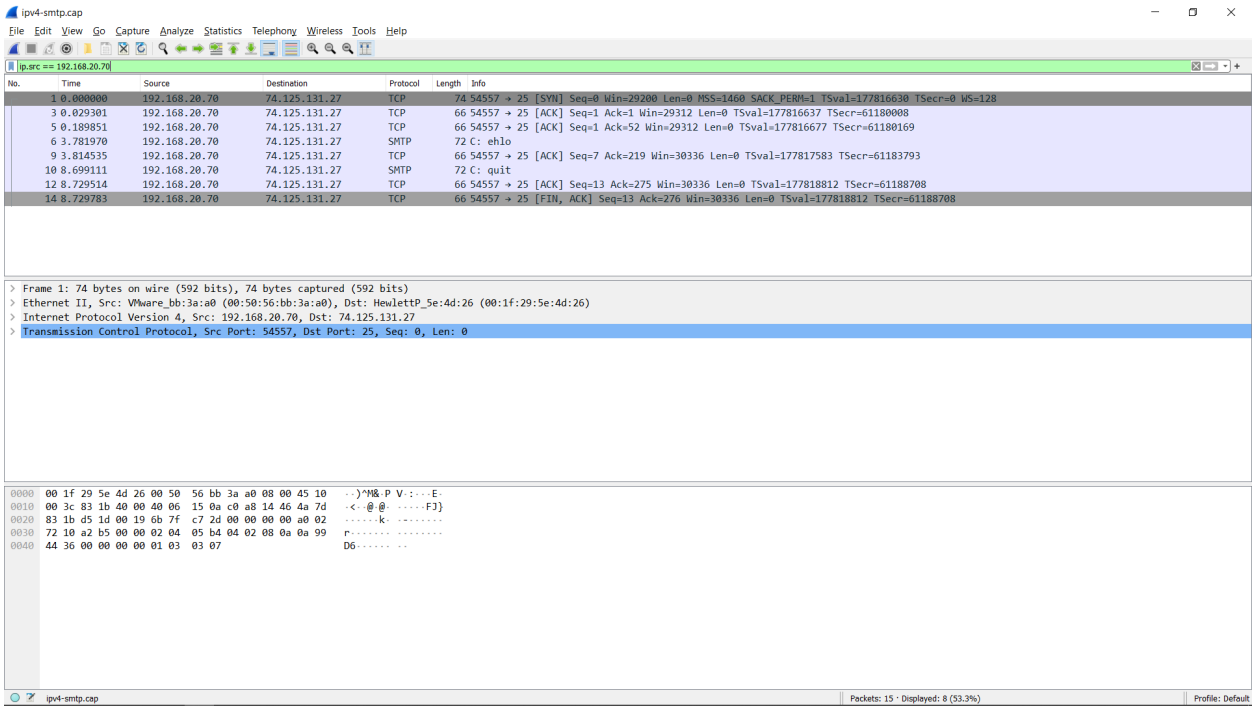
For this assignment I have used pyshark, a python module which allows us to parse packets using wireshark dissectors . github

I download a Pcap file from this website for this assignment here and I have also linked the work for this assignment here

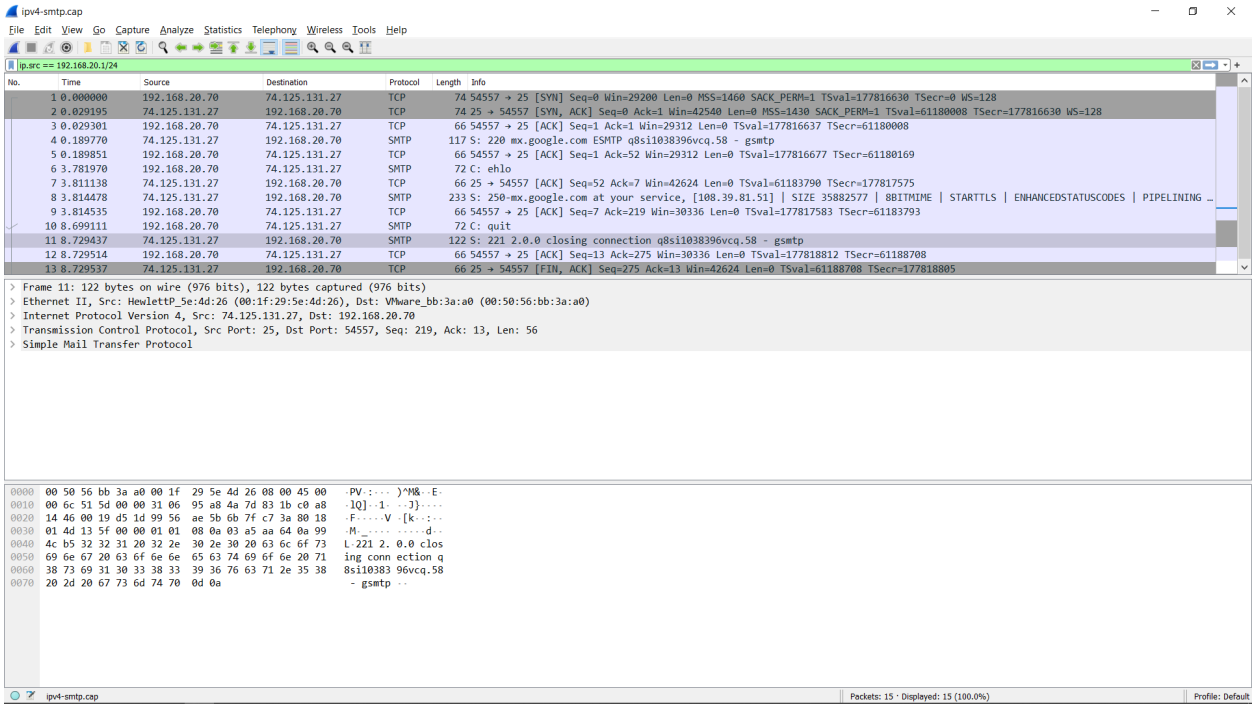


Some simple filter techniques on wireshark

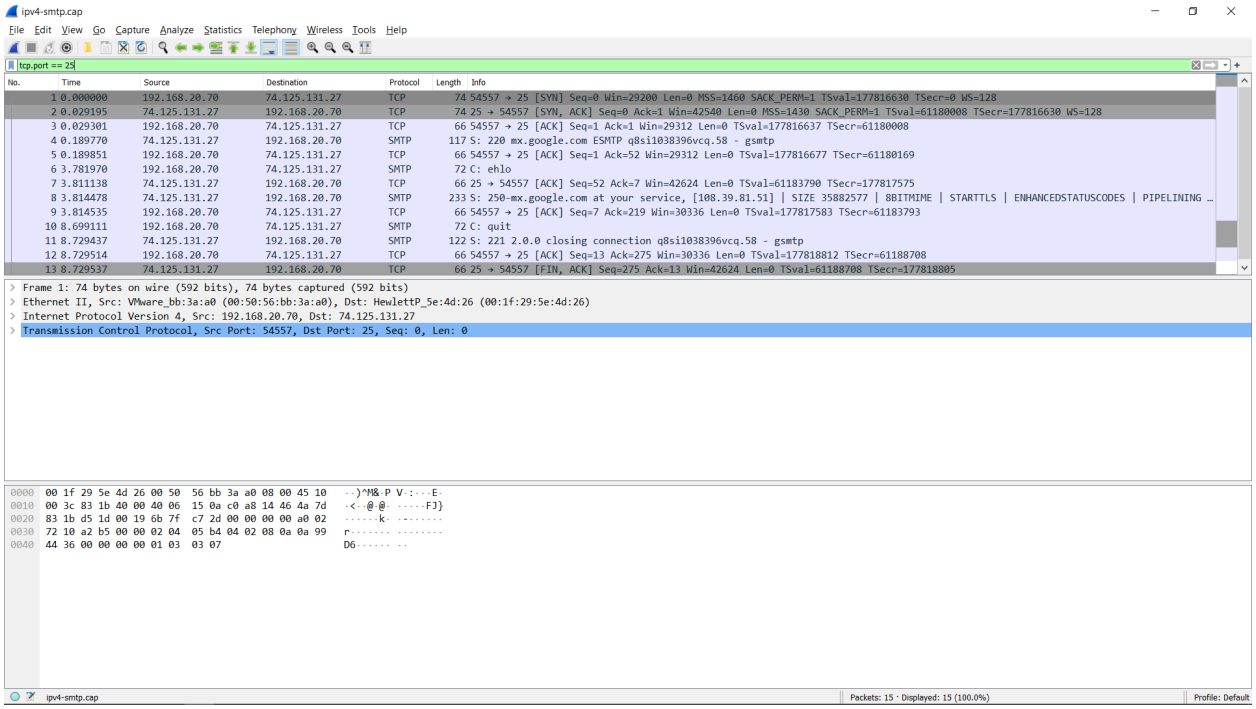
1) If we want to filter out data using ip address source we can use *ip.src== <ip address>*



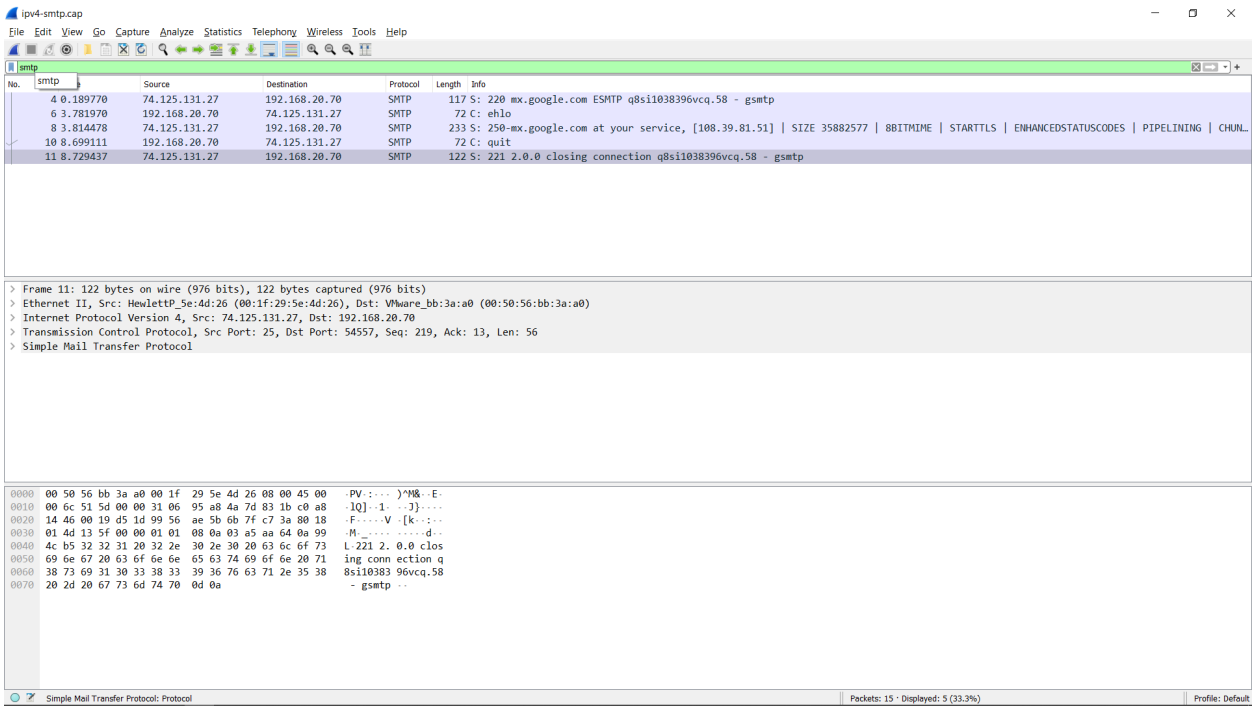
2) If we want to filter out data using ip subnet address we can use *ip.addr=<ip address/subnet>*



3) Filter out using port numbers *tcp.port == <port number>*



4) Get SMTP records just search using keyword *SMTP*



Using pyshark for packet analysis

Note: Most of the values returned by pyshark are string values

---

```
import pyshark

pcap_file = pyshark.FileCapture("nf9-juniper-vmx.pcapng.cap")

# get a single packet
packet = pcap_file[0]

# get packet source ip
print(packet['ip'].src)
# >> 192.168.17.114
print(packet['ip'].dst)
# >> 192.168.16.36

# get packet layers
```

```
print(packet.layers)
# >> [ < ETH Layer > , < IP Layer > , < UDP Layer > , < DATA Layer > ]

# get Ip field names
print(packet.ip.field_names)
#  >>  ['version',  'hdr_len',  'dsfield',  'dsfield_dscp',  'dsfield_ecn',
'len', 'id', 'flags', 'flags_rb', 'flags_df',
#        'flags_mf',    'frag_offset',    'ttl',    'proto',    'checksum',
'checksum_status', 'src', 'addr', 'src_host', 'host', 'dst', 'dst_host']

# get Ip version
print(packet.ip.version)
# > 4

# print out the packet content
print(packet.pretty_print())
"""
Layer ETH:
        Destination: 0a:7c:7c:ec:d6:97
        Address: 0a:7c:7c:ec:d6:97
           .... ..1. .... .... .... .... = LG bit: Locally administered
address (this is NOT the factory default)
            .... ...0 .... .... .... .... = IG bit: Individual address
(unicast)
        Source: 00:05:86:71:82:00
          .... ..0. .... .... .... .... = LG bit: Globally unique address
(factory default)
            .... ...0 .... .... .... .... = IG bit: Individual address
(unicast)
        Type: IPv4 (0x0800)
        Address: 00:05:86:71:82:00
Layer IP:
        0100 .... = Version: 4
        .... 0101 = Header Length: 20 bytes (5)
        Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
        0000 00.. = Differentiated Services Codepoint: Default (0)
           .... ..00 = Explicit Congestion Notification: Not ECN-Capable
Transport (0)
        Total Length: 84
        Identification: 0x0003 (3)
```

```
        Flags: 0x00
        0... .... = Reserved bit: Not set
        .0.. .... = Don't fragment: Not set
        ..0. .... = More fragments: Not set
        ...0 0000 0000 0000 = Fragment Offset: 0
        Time to Live: 250
        Protocol: UDP (17)
        Header Checksum: 0x1daf [validation disabled]
        Header checksum status: Unverified
        Source Address: 192.168.17.114
        Destination Address: 192.168.16.36
Layer UDP:
        Source Port: 47043
        Destination Port: 9995
        Length: 16384 (bogus, payload length 64)
        Expert Info (Error/Malformed): Bad length value 16384 > IP payload
length
        Bad length value 16384 > IP payload length
        Severity level: Error
        Group: Malformed
        Checksum: 0x0000 [zero-value ignored]
        Checksum Status: Not present
        Stream index: 0
        Timestamps
        Time since first frame: 0.000000000 seconds
        Time since previous frame: 0.000000000 seconds
        UDP payload (56 bytes)
DATANone
"""

# get host name
print(packet.http.host)
```
