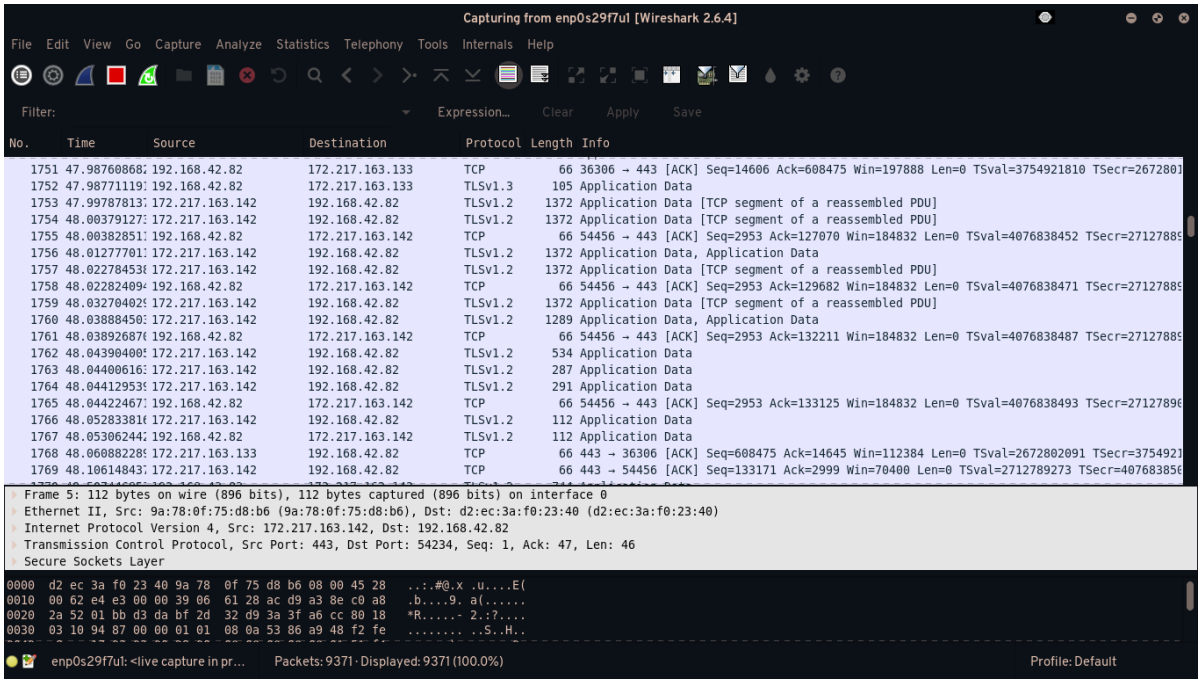
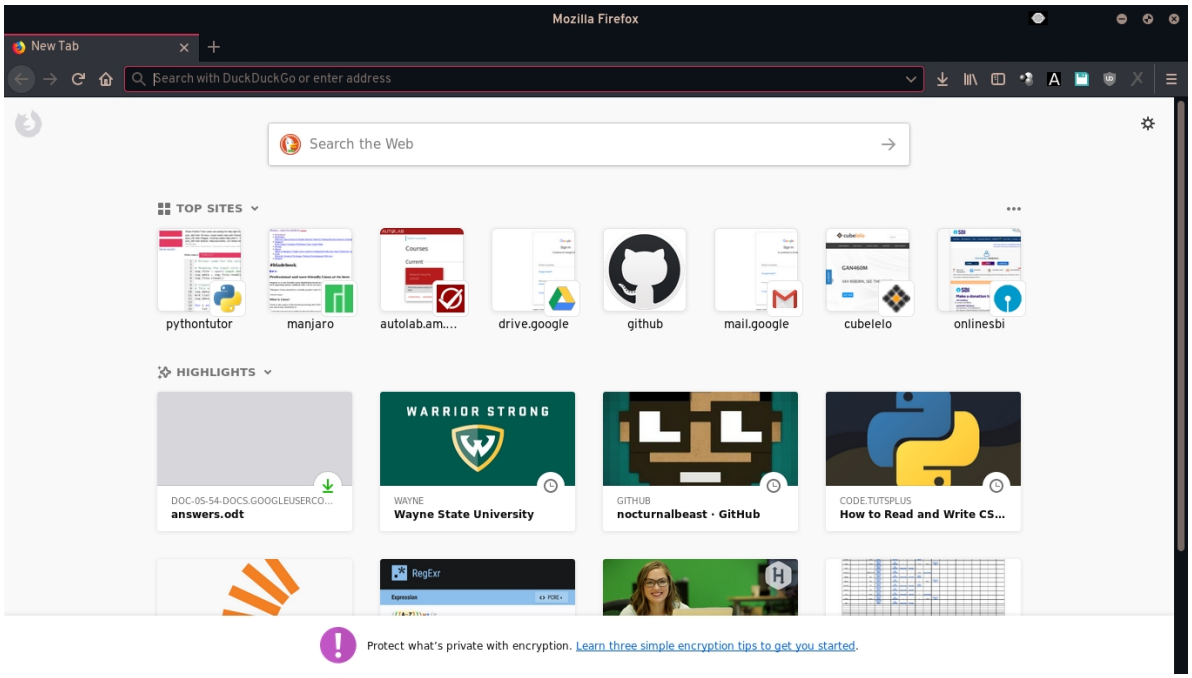
ECS Lab Assignment - Wireshark

1. The screenshots provide the steps that were followed:

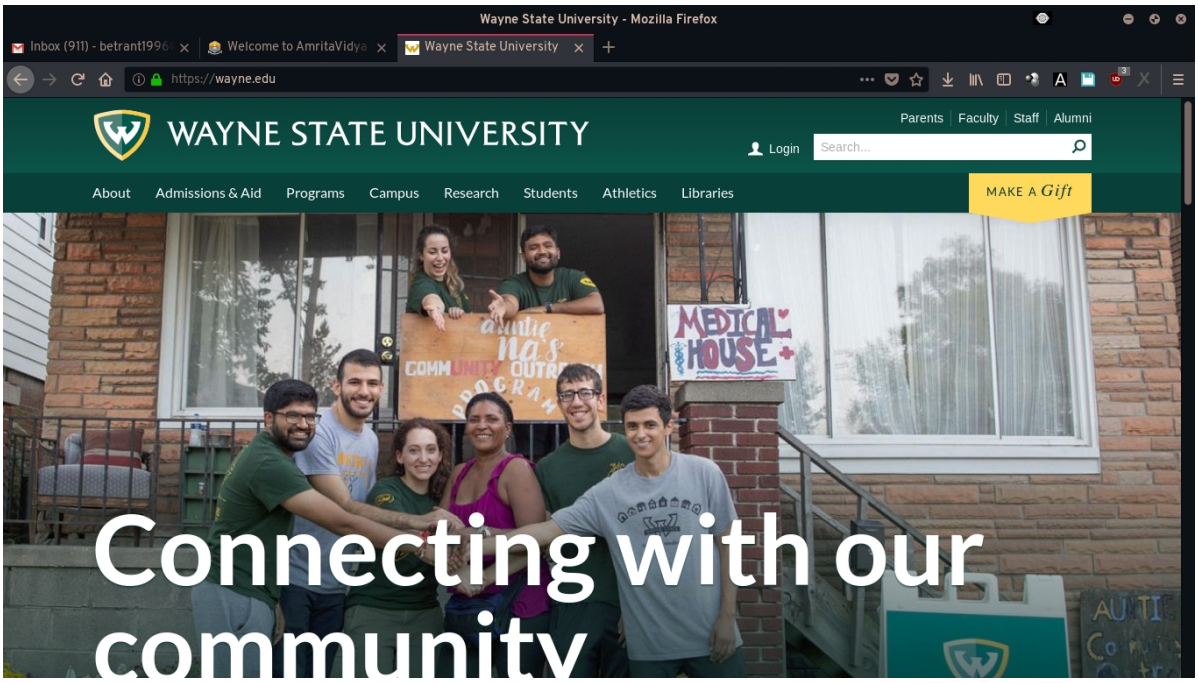
a. Opening Wireshark and starting capture on selected interface.



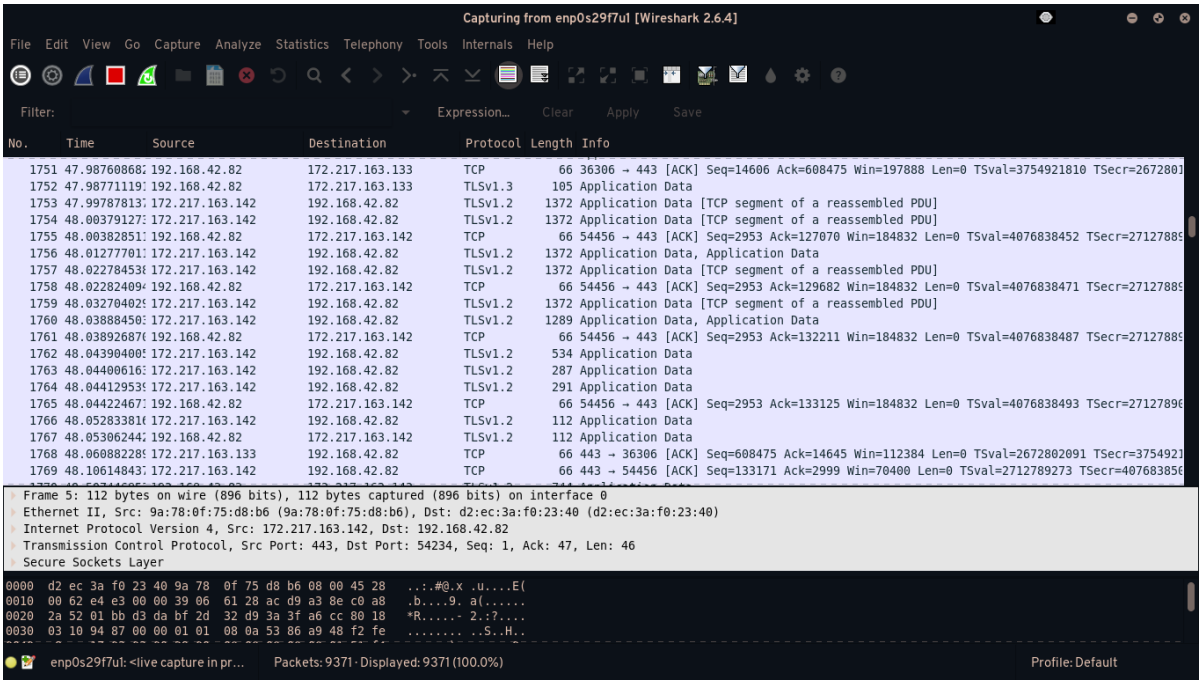
b. Opening preferred web browser.



c. Opening www.wayne.edu in the browser.

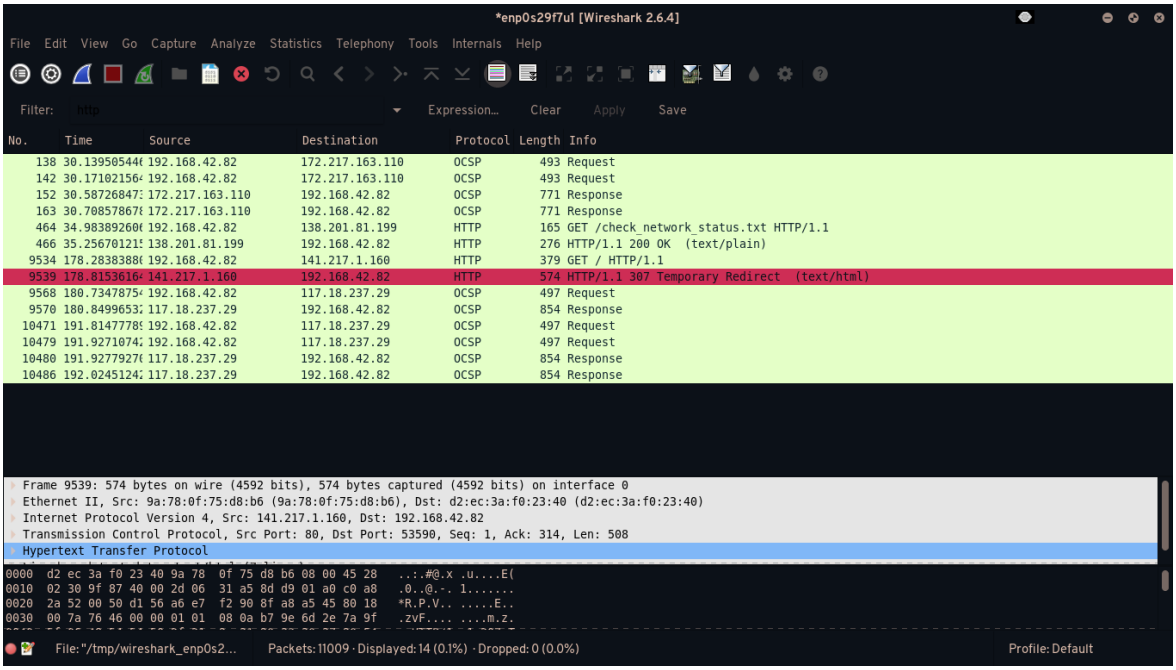


d. Viewing captured packets in Wireshark.

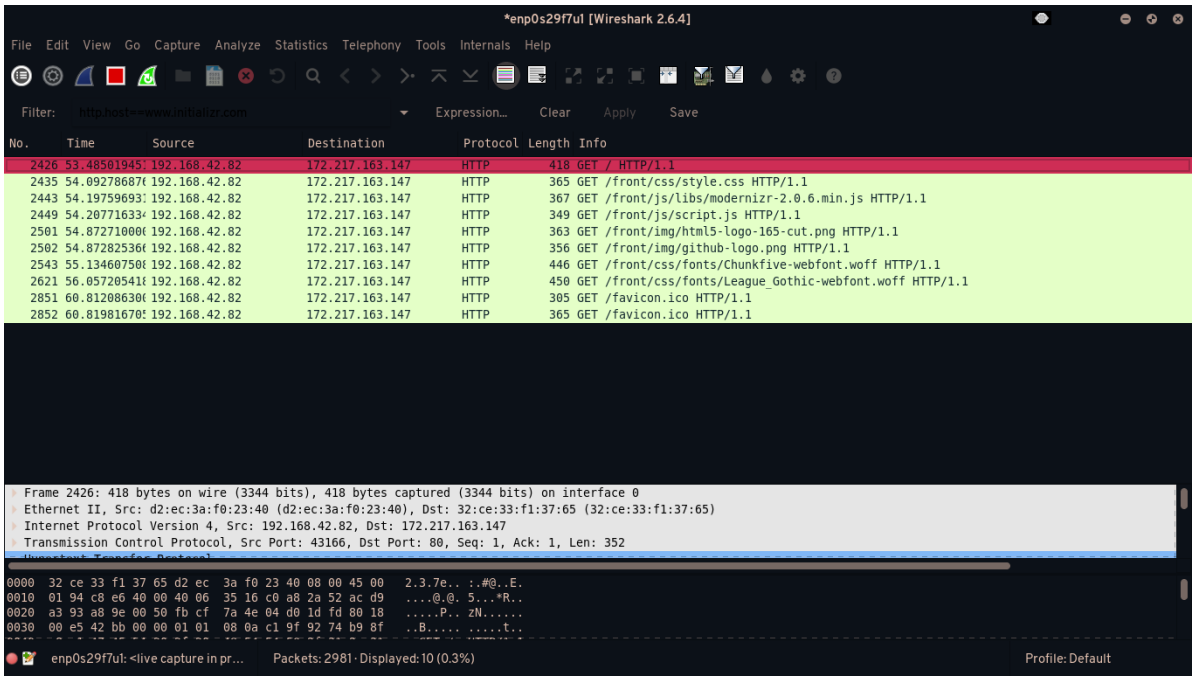


e. Viewing differently colored packets and understanding color rules.

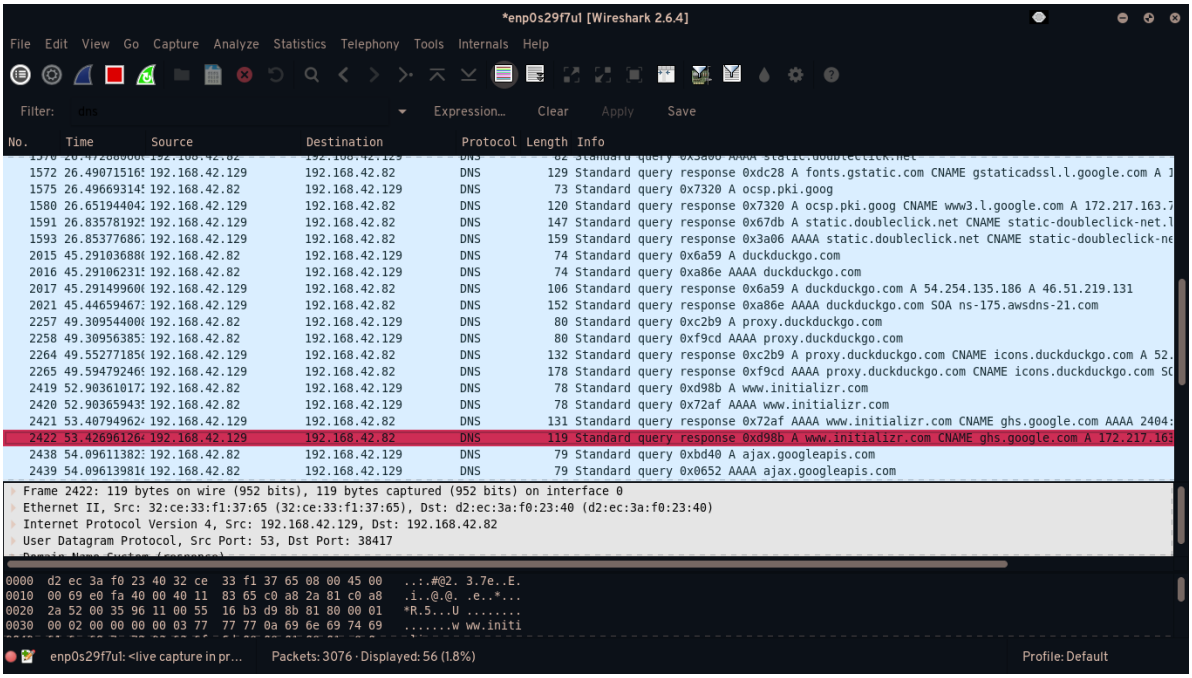
f. Filtering HTTP traffic with the ‘http’ filter command.



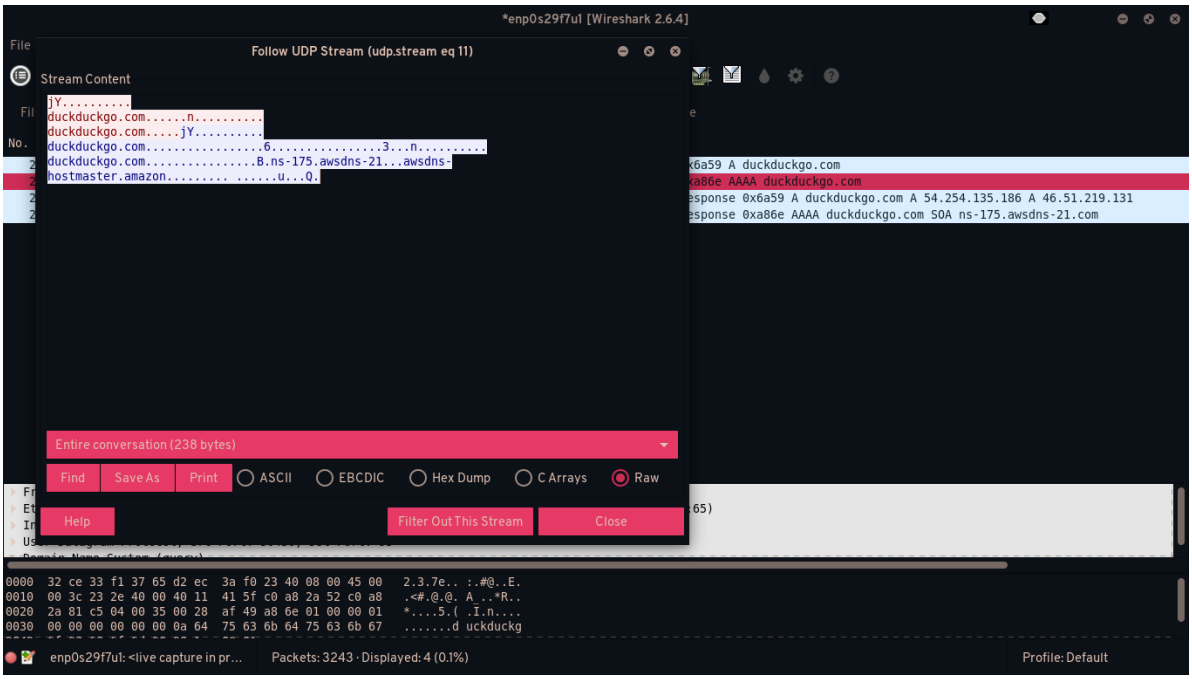
g. Further filtering using ‘http.host==\{Domain name\}’ (Alternate domain used here since wayne.edu migrated to HTTPS)



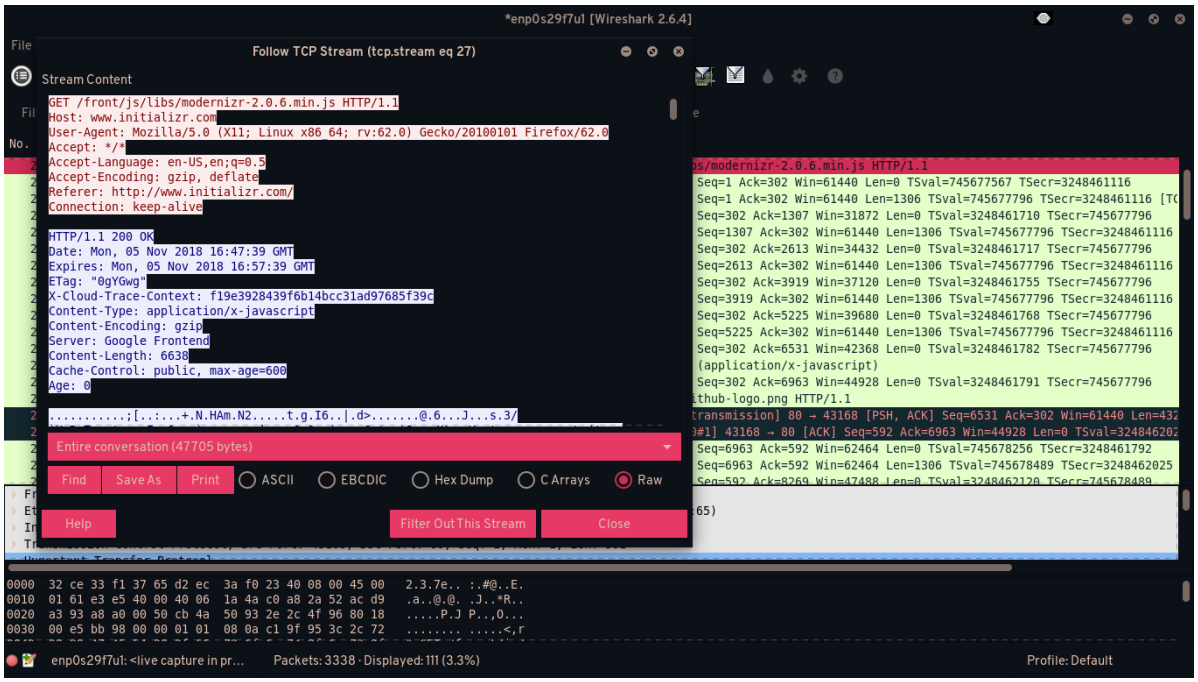
h. Viewing DNS traffic using the ‘dns’ filter.



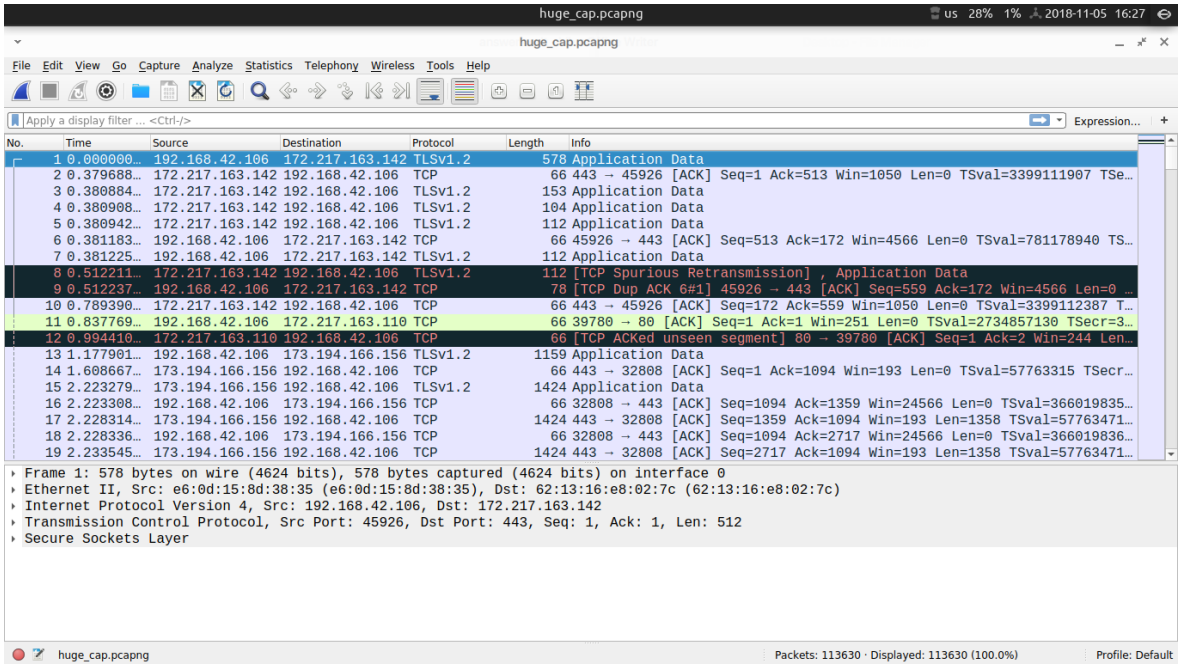
i. Using the ‘Follow UDP stream’ option on selected DNS traffic.

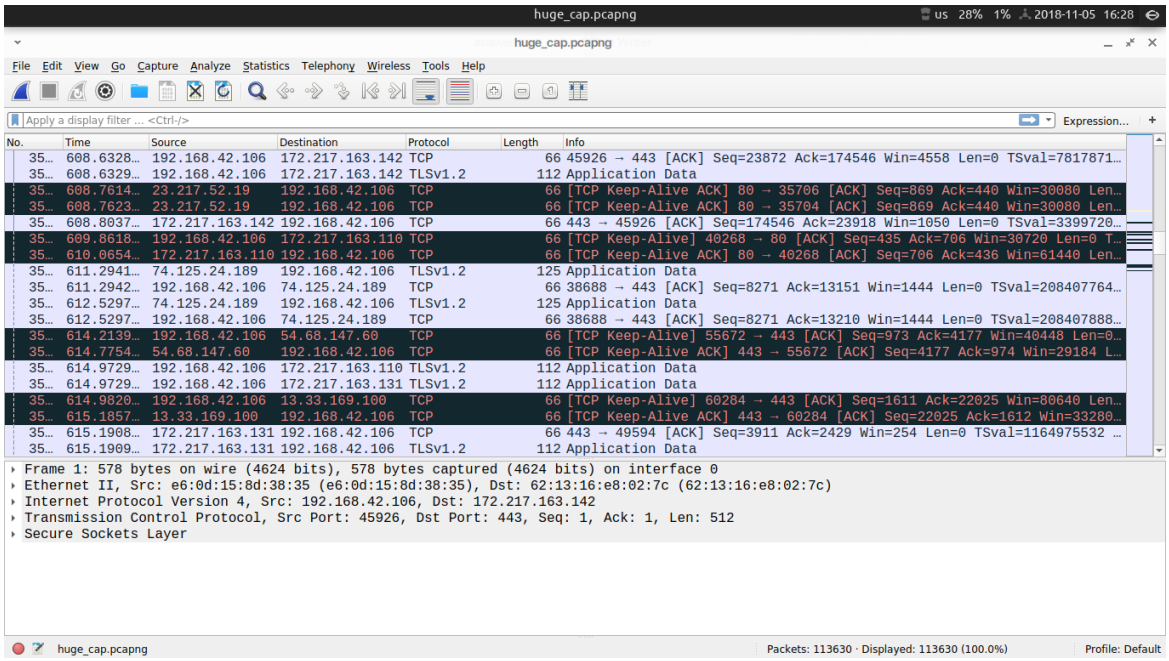


j. Using the ‘Follow TCP stream’ option on selected HTTP traffic.



2. According to the default coloring rules in Wireshark, there are around twelve kinds of colors that signify different kinds of packets. These colors include red, black, light yellow, blue, violet etc. Now the black packet coloring is assigned to certain forms of packets that essentially mean that there is some recoverable error that has happened in the communication.These include packets such as bad TCP packets, topology change packets, ICMP errors and checksum errors. Bad TCP packets include out-of-order packets, duplicate ACK messages and TTL timeouts.

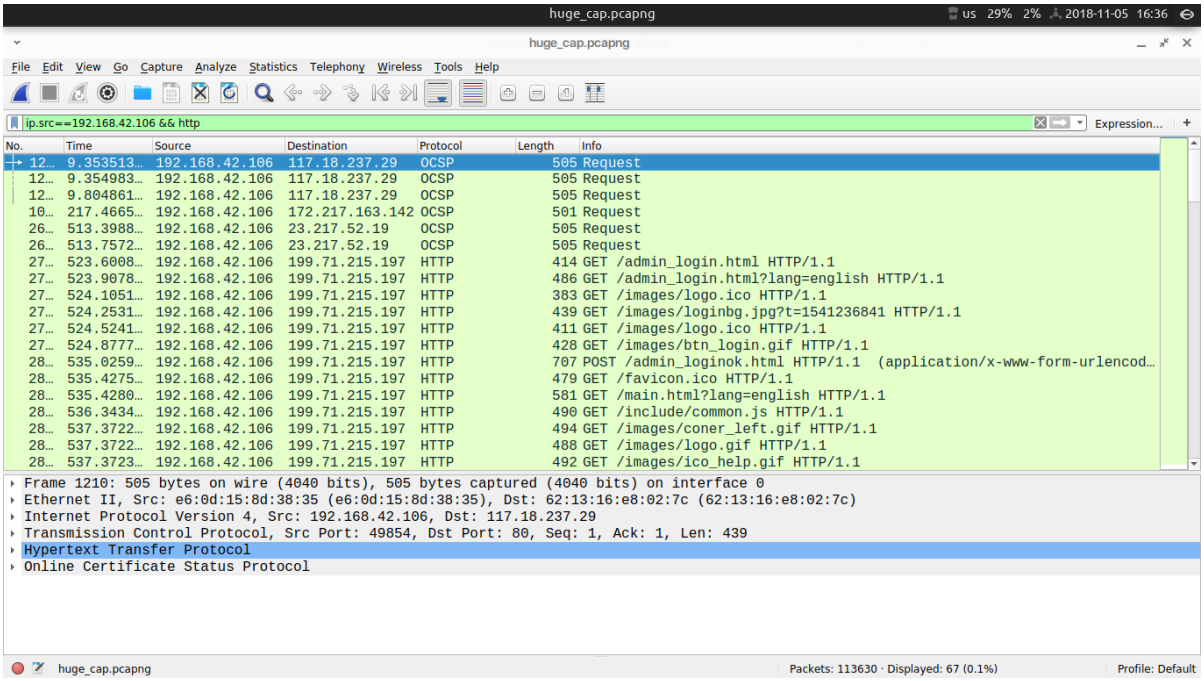



huge_cap.pcapng
us 28% 1% 2018-11-05 16:28
huge_cap.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
No. Time Source Destination Protocol Length Info
35… 608.6328… 192.168.42.106 172.217.163.142 TCP 66 45926 → 443 [ACK] Seq=23872 Ack=174546 Win=4558 Len=0 TSval=7817871…
35… 608.6329… 192.168.42.106 172.217.163.142 TLSv1.2 112 Application Data
35… 608.7614… 23.217.52.19 192.168.42.106 TCP 66 [TCP Keep-Alive ACK] 80 → 35706 [ACK] Seq=869 Ack=440 Win=30080 Len…
35… 608.7623… 23.217.52.19 192.168.42.106 TCP 66 [TCP Keep-Alive ACK] 80 → 35704 [ACK] Seq=869 Ack=440 Win=30080 Len…
35… 608.8037… 172.217.163.142 192.168.42.106 TCP 66 443 → 45926 [ACK] Seq=174546 Ack=23918 Win=1050 Len=0 TSval=3399720…
35… 609.8618… 192.168.42.106 172.217.163.110 TCP 66 [TCP Keep-Alive] 40268 → 80 [ACK] Seq=435 Ack=706 Win=30720 Len=0 T…
35… 610.0654… 172.217.163.110 192.168.42.106 TCP 66 [TCP Keep-Alive ACK] 80 → 40268 [ACK] Seq=706 Ack=436 Win=61440 Len…
35… 611.2941… 74.125.24.189 192.168.42.106 TLSv1.2 125 Application Data
35… 611.2942… 192.168.42.106 74.125.24.189 TCP 66 38688 → 443 [ACK] Seq=8271 Ack=13151 Win=1444 Len=0 TSval=208407764…
35… 612.5297… 74.125.24.189 192.168.42.106 TLSv1.2 125 Application Data
35… 612.5297… 192.168.42.106 74.125.24.189 TCP 66 38688 → 443 [ACK] Seq=8271 Ack=13210 Win=1444 Len=0 TSval=208407888…
35… 614.2139… 192.168.42.106 54.68.147.60 TCP 66 [TCP Keep-Alive] 55672 → 443 [ACK] Seq=973 Ack=4177 Win=40448 Len=0…
35… 614.7754… 54.68.147.60 192.168.42.106 TCP 66 [TCP Keep-Alive ACK] 443 → 55672 [ACK] Seq=4177 Ack=974 Win=29184 L…
35… 614.9729… 192.168.42.106 172.217.163.110 TLSv1.2 112 Application Data
35… 614.9729… 192.168.42.106 172.217.163.131 TLSv1.2 112 Application Data
35… 614.9820… 192.168.42.106 13.33.169.100 TCP 66 [TCP Keep-Alive] 60284 → 443 [ACK] Seq=1611 Ack=22025 Win=80640 Len…
35… 615.1857… 13.33.169.100 192.168.42.106 TCP 66 [TCP Keep-Alive ACK] 443 → 60284 [ACK] Seq=22025 Ack=1612 Win=33280…
35… 615.1908… 172.217.163.131 192.168.42.106 TCP 66 443 → 49594 [ACK] Seq=3911 Ack=2429 Win=254 Len=0 TSval=1164975532 …
35… 615.1909… 172.217.163.131 192.168.42.106 TLSv1.2 112 Application Data
Frame 1: 578 bytes on wire (4624 bits), 578 bytes captured (4624 bits) on interface 0
Ethernet II, Src: e6:0d:15:8d:38:35 (e6:0d:15:8d:38:35), Dst: 62:13:16:e8:02:7c (62:13:16:e8:02:7c)
Internet Protocol Version 4, Src: 192.168.42.106, Dst: 172.217.163.142
Transmission Control Protocol, Src Port: 45926, Dst Port: 443, Seq: 1, Ack: 1, Len: 512
Secure Sockets Layer
huge_cap.pcapng
Packets: 113630 · Displayed: 113630 (100.0%)
Profile: Default

3. The filter to view all outgoing HTTP packets consists of two parts; one wherein the outgoing packets on all protocols are selected with the ‘ip.src’ filter with it being equal to our IP address; and the next wherein only the HTTP packets are filtered out using the ‘http’ filter.

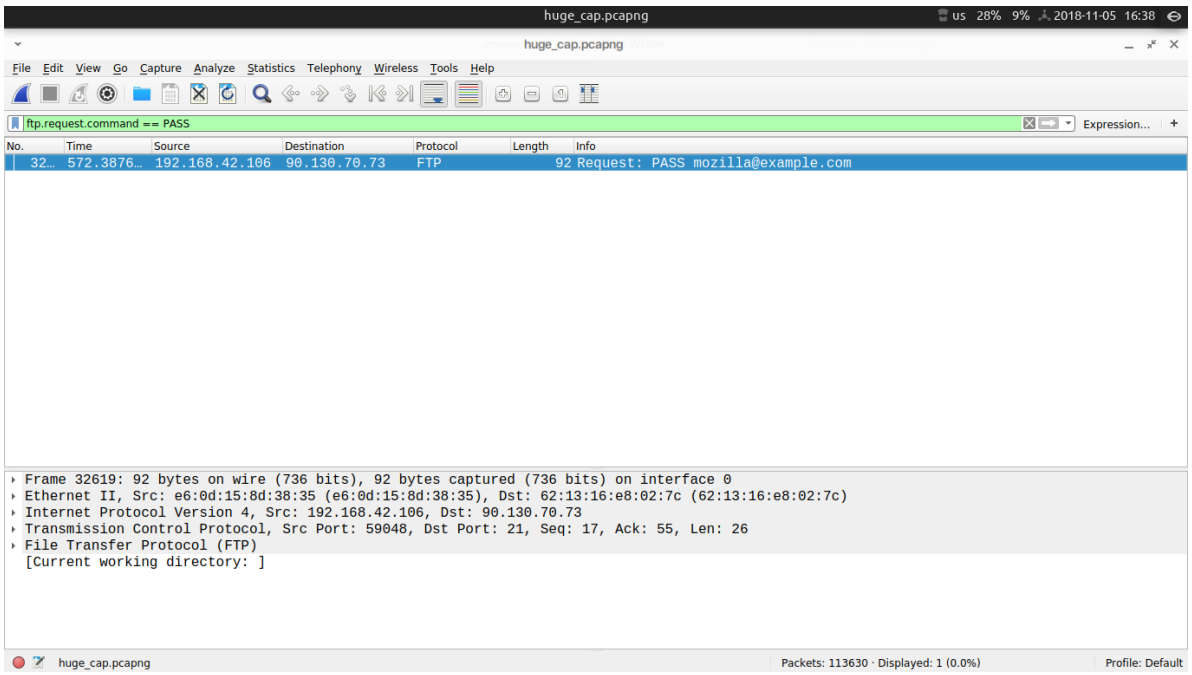
Thus the filter is as such: ‘ip.src==\{IP address\} && http’

Eg: ip.src==192.168.42.106 && http



4. The HTTP protocol is one that relies on the TCP transport layer protocol to work and thus it’s communication is via a TCP stream. Alternatively, the DNS protocol is lightweight and does not require much reliability on its mechanism, and thus uses the UDP transport layer protocol for it to work. Thus Wireshark shows the option “Follow TCP stream” for HTTP traffic and shows the option “Follow UDP stream” for DNS traffic.

5. To filter out the request where the client sends the password to the server, use the following filter:
ftp.request.command == PASS



The password will be obtained by inspecting it’s value.