

Task 5: Network Traffic Capture & Analysis Report

Tool Used: Wireshark

Websites Visited: YouTube, GitHub, Dailymotion

File Submitted: .pcap file + Screenshots

Objective:

To capture live network traffic using Wireshark and analyze packets based on protocol types. This helps understand how different protocols behave and identify them during traffic inspection.

Steps taken:

1. Started Wireshark and selected my active network interface.
2. Began live capture and opened 3 websites in browser:
 - YouTube (streaming/media-heavy site)
 - GitHub (HTTPS-encrypted developer platform)
 - Dailymotion (another video/media-based site)
3. Stopped the capture after ~1 minute.
4. Applied filters one by one:
 - http to view HTTP traffic (non-encrypted web data)

No.	Time	Source	Destination	Protocol	Length	Info
27	11.1815	2404:ba00:1:8089:47...	2606:4700::6812:15d5	HTTP	195	GET /59.cr1 HTTP/1.1
28	11.178835	2606:4700::6812:15d5	2404:ba00:1:8089:47...	PKIX-C...	204	Certificate Revocation List
29498	94.493480	192.168.1.41	192.124.249.22	HTTP	280	GET //MEQwQjBAyMD4wPDAjBguDgMCGgUABBSLwZ6Eh5gdYc9UaSEaaLjjETNtkAQUv1%2B30c7dH4b0W1wS3ncQwg6piOCCazkUhA%3D%3D HTTP/1.1
29501	94.551566	192.124.249.22	192.168.1.41	OCSP	1249	Response
29504	94.569786	192.168.1.41	192.124.249.22	HTTP	278	GET //MEIwQDA%2BMDwwOjAJBgUrDgMCGgUABBBQUwPiEZQ6%2FsvZNPaFToltFxx8ZwqAQUFAwyH6fZiH%2FEFwiJYqihzqsHwycCAQc%3D HTTP/1.1
29507	94.628365	192.124.249.22	192.168.1.41	OCSP	1287	Response

- tcp to view TCP-level connections for all visited sites

No.	Time	Source	Destination	Protocol	Length	Info
27783	91.008934	188.65.124.58	192.168.1.41	TCP	60	443 → 51426 [ACK] Seq=8257 Ack=55387 Win=42496 Len=0
27784	91.009720	188.65.124.58	192.168.1.41	TCP	60	443 → 51426 [ACK] Seq=8257 Ack=59743 Win=42496 Len=0
27785	91.009822	188.65.124.58	192.168.1.41	TCP	60	443 → 51426 [ACK] Seq=8257 Ack=61252 Win=42496 Len=0
27786	91.010278	188.65.124.58	192.168.1.41	TLSv1.3	439	Application Data
27794	91.066467	192.168.1.41	188.65.124.58	TCP	54	51426 → 443 [ACK] Seq=61252 Ack=8642 Win=130816 Len=0
27886	91.171509	54.84.92.154	192.168.1.41	TCP	66	443 → 51554 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1452 SACK_PERM WS=4096
27887	91.172082	192.168.1.41	54.84.92.154	TCP	54	51554 → 443 [ACK] Seq=1 Ack=1 Win=132096 Len=0
27888	91.175635	192.168.1.41	54.84.92.154	TLSv1.3	324	Client Hello (SHA=report2.hb.brainlyads.com)
27891	91.244488	18.66.41.73	192.168.1.41	TCP	66	[TCP Retransmission] 443 → 51413 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1440 SACK_PERM WS=512
27895	91.380888	54.84.92.154	192.168.1.41	TCP	60	443 → 51554 [ACK] Seq=1 Ack=271 Win=69632 Len=0
27896	91.380996	54.84.92.154	192.168.1.41	TLSv1.3	1506	Server Hello, Change Cipher Spec, Application Data
27897	91.380996	54.84.92.154	192.168.1.41	TLSv1.3	1027	Application Data, Application Data, Application Data
27898	91.381077	192.168.1.41	54.84.92.154	TCP	54	51554 → 443 [ACK] Seq=271 Ack=2426 Win=132096 Len=0
27903	91.410281	2404:ba00:1:8089:47...	2606:4700::6812:15d5	TCP	86	51555 → 80 [SYN] Seq=0 Win=64440 Len=0 MSS=1432 WS=256 SACK_PERM
27904	91.411494	2606:4700::6812:15d5	2404:ba00:1:8089:47...	TCP	86	80 → 51555 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1360 SACK_PERM WS=8192
27905	91.411670	2404:ba00:1:8089:47...	2606:4700::6812:15d5	TCP	74	51555 → 80 [ACK] Seq=1 Ack=1 Win=131840 Len=0
27906	91.411815	2404:ba00:1:8089:47...	2606:4700::6812:15d5	HTTP	195	GET /59.cr1 HTTP/1.1
27907	91.413040	2606:4700::6812:15d5	2404:ba00:1:8089:47...	TCP	74	80 → 51555 [ACK] Seq=1 Ack=122 Win=131072 Len=0
27908	91.413090	2606:4700::6812:15d5	2404:ba00:1:8089:47...	TCP	74	[TCP Dup ACK 27907#1] 80 → 51555 [ACK] Seq=1 Ack=122 Win=131072 Len=0
27909	91.446150	2606:4700::6812:15d5	2404:ba00:1:8089:47...	TCP	1434	80 → 51555 [ACK] Seq=1 Ack=122 Win=131072 Len=1360 [TCP PDU reassembled in 28318]
27910	91.446256	2606:4700::6812:15d5	2404:ba00:1:8089:47...	TCP	1434	80 → 51555 [ACK] Seq=1361 Ack=122 Win=131072 Len=1360 [TCP PDU reassembled in 28318]
27911	91.446284	2404:ba00:1:8089:47...	2606:4700::6812:15d5	TCP	74	51555 → 80 [ACK] Seq=122 Ack=2721 Win=131840 Len=0
27912	91.446329	2606:4700::6812:15d5	2404:ba00:1:8089:47...	TCP	1434	80 → 51555 [ACK] Seq=2721 Ack=122 Win=131072 Len=1360 [TCP PDU reassembled in 28318]
27913	91.446429	2606:4700::6812:15d5	2404:ba00:1:8089:47...	TCP	1434	80 → 51555 [ACK] Seq=4081 Ack=122 Win=131072 Len=1360 [TCP PDU reassembled in 28318]
27914	91.446457	2404:ba00:1:8089:47...	2606:4700::6812:15d5	TCP	74	51555 → 80 [ACK] Seq=122 Ack=5441 Win=131840 Len=0

> Frame 27898: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{BABDEEB8-BDF8-4A29-A03C-B371EA0F5332}

Ethernet II, Src: RealtekSemic_d1:18:a6 (00:e0:4c:d1:18:a6), Dst: GOIPGlobalSe_05:48:41 (b8:b7:db:05:48:41)

Internet Protocol Version 4, Src: 192.168.1.41, Dst: 54.84.92.154

Transmission Control Protocol, Src Port: 51554, Dst Port: 443, Seq: 271, Ack: 2426, Len: 0

0000

b8 b7 db 05 48 41 00 e0 4c d1 18 a6 08 00 45 00 ...HA... L.....E

0010

00 28 d2 d5 40 00 80 06 d3 3a c0 a8 01 29 36 54 ...@.....:....)6T

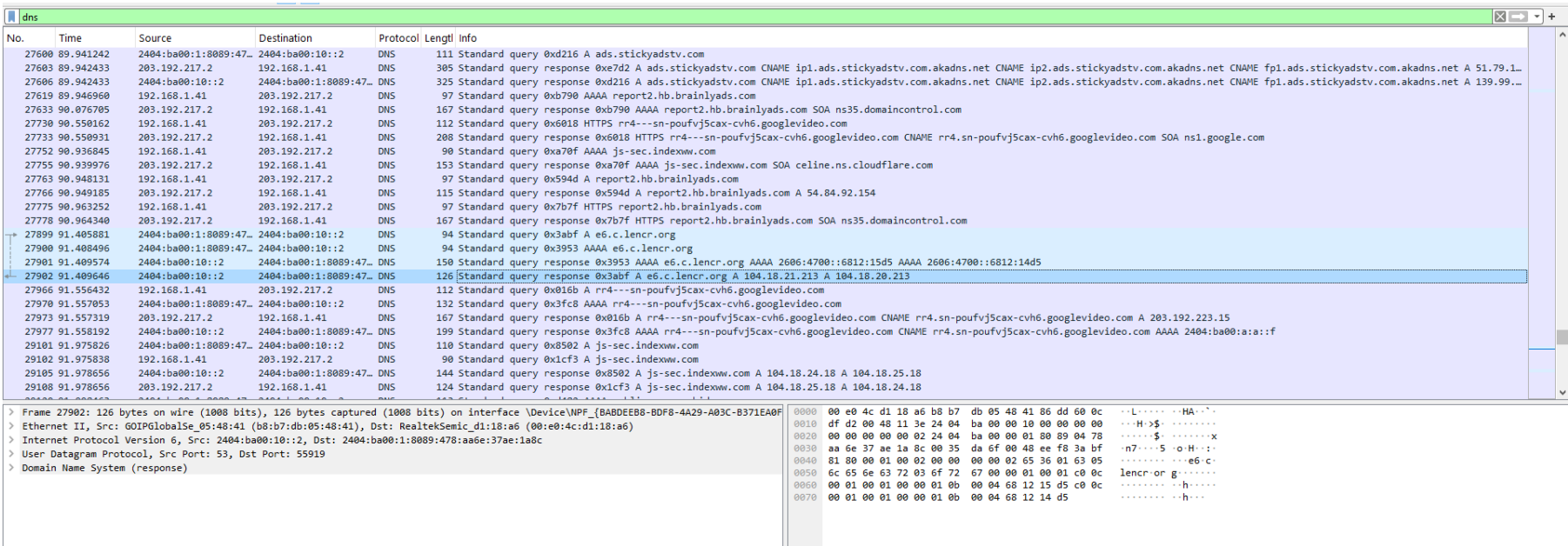
0020

5c 9a c9 62 01 bb 97 7a 31 fd 22 80 02 ac 50 10 \..b...z 1"...P

0030

02 04 9f 4f 00 00 ...O...

- dns to view DNS query/response packets (domain resolution)



5. Saved the capture as .pcap file and took protocol-specific screenshots.

6. Analyzed individual packets for source/destination IPs, flags, and info.

Protocols Identified:

Protocol	Where It Was Found	Purpose
DNS	On all websites	Used to resolve domain names (e.g., youtube.com to IP)
TCP	On all connections	Underlying transport protocol, ensures delivery
HTTP	Seen on GitHub/Dailymotion (some content)	Transmits web content (unencrypted)
HTTPS	Seen for YouTube & GitHub	Secure version of HTTP (was visible as TCP + encrypted)

Observations:

- **YouTube and Dailymotion** triggered lots of TCP packets for video loading (streaming behavior).
- **GitHub** showed secure https traffic but its **DNS queries** and **TCP handshakes** were visible.
- DNS packets clearly showed the domain names I accessed.
- The packet info (source/destination, flags) helped track how devices communicated.

Conclusion:

Wireshark provided real-time insight into how data flows when visiting different websites. I could see how each website uses a mix of DNS, TCP, and HTTP(S) to load content. This task gave me practical experience in filtering, identifying protocols, and understanding how even encrypted traffic leaves metadata behind.