Network Traffic Analysis and Security Assessment

Objective:

The objective of this study is to investigate and evaluate manual classification techniques utilizing protocol headers for in-depth analysis of network traffic patterns, including potential security threats and attack simulations. This study aims to refine existing methodologies, conduct comprehensive data collection and statistical analysis, and document findings to enhance understanding and proactive management of network security risks. Specifically, the objectives include:

1. Refinement of existing methodologies to focus on using protocol headers for manual classification and analysis of network traffic.
2. Capture live network traffic using Wireshark, targeting a diverse range of protocols and traffic patterns, to ensure representation of both normal network behavior and potential security threats.
3. Manual analysis of captured packets to extract protocol headers and classify each packet based on identified protocol headers into predefined types, including simulations of various cyber attacks such as SYN flood attacks and Denial of Service (DoS) attacks.
4. Aggregation and organization of classified packets into respective traffic types, followed by statistical analysis to determine distribution and characteristics of different protocol types within captured traffic. Additionally, analysis of attack simulations to understand their impact on network traffic.
5. Documentation of the experimental setup, procedures, and findings in a structured manner, preparing comprehensive reports detailing the classification process, results, insights gained from the analysis, and observations from attack simulations.

Experimental Design and Methodology:

1. **Objective Refinement**: Refine the objective to focus on using protocol headers for manual classification and analysis of network traffic.

2. **Data Collection**: Capture live network traffic using Wireshark, targeting a diverse range of protocols and traffic patterns. Collect sufficient data to represent both normal network behavior and potential security threats.

3. **Manual Classification**: Manually analyze captured packets to extract protocol headers (e.g., source and destination IP addresses, port numbers, protocol types). Classify each packet based on the identified protocol headers, categorizing them into predefined types (e.g., HTTP, FTP, DNS, SSH, ICMP).

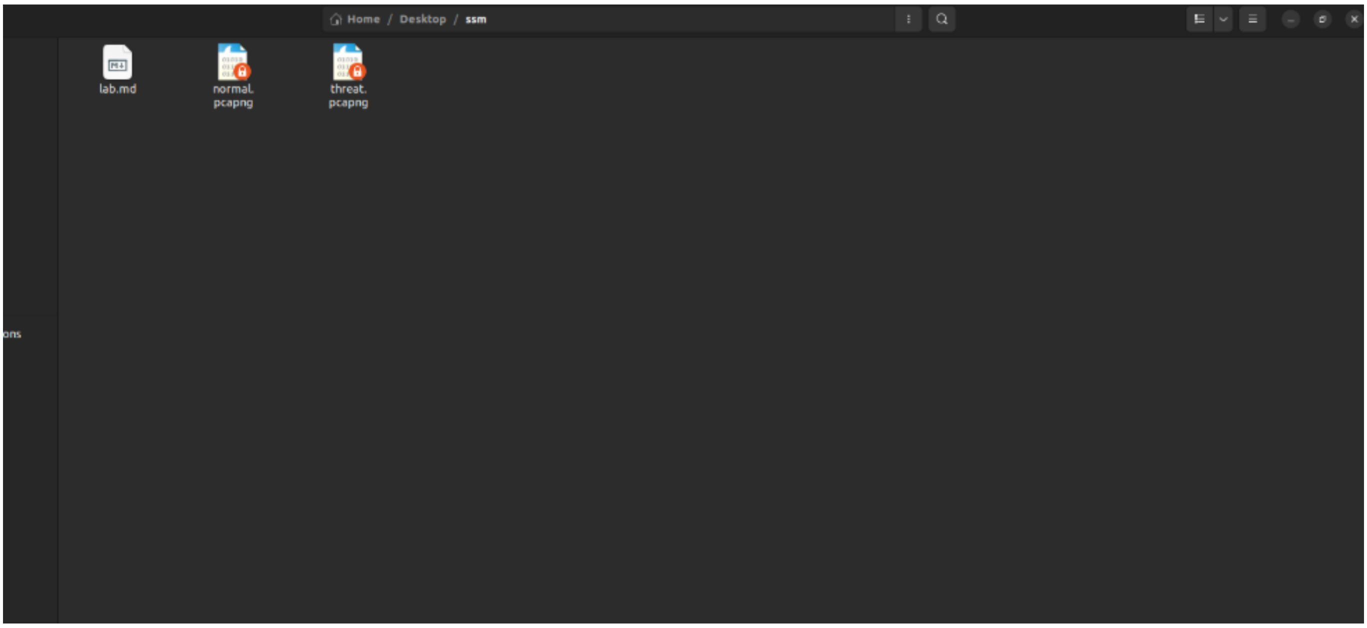
4. **Data Analysis**: Aggregate and organize the classified packets into respective traffic types. Conduct statistical analysis to determine the distribution and characteristics of different protocol types within the captured traffic. Analyze the frequency, volume, and patterns of each traffic type to identify anomalies or potential security threats.

5. **Documentation and Reporting**: Document the experimental setup, procedures, and findings in a structured manner. Prepare comprehensive reports detailing the classification process, results, and insights gained from the analysis.

Data Collection:

For the purpose of this experiment, network traffic data was captured using Wireshark. The following steps were taken to capture network traffic:

1. **Opening Wireshark Application**: The Wireshark application was launched by clicking on the icon, initiating the capture process.
2. **Selecting Network Interface**: From the welcome screen of Wireshark, the appropriate network interface was selected to capture traffic. This selection was based on the network interface through which the desired traffic was flowing.
3. **Starting Packet Capture**: Double-clicking on the selected capture interface initiated the packet capture process. This action began capturing packets flowing through the specified network interface.
4. **Stopping Packet Capture**: The packet capture process was halted by clicking on the red Stop Capture button once sufficient data had been captured for analysis.
5. **Saving Packet Capture**: The captured packet data was saved into a specific file by selecting File > Save As. A location and name for the file were specified to facilitate further analysis.
6. **Repeat Capture for Discrepancy Data**: To capture data with discrepancies or potential anomalies, the above steps were repeated, capturing another set of network traffic data.



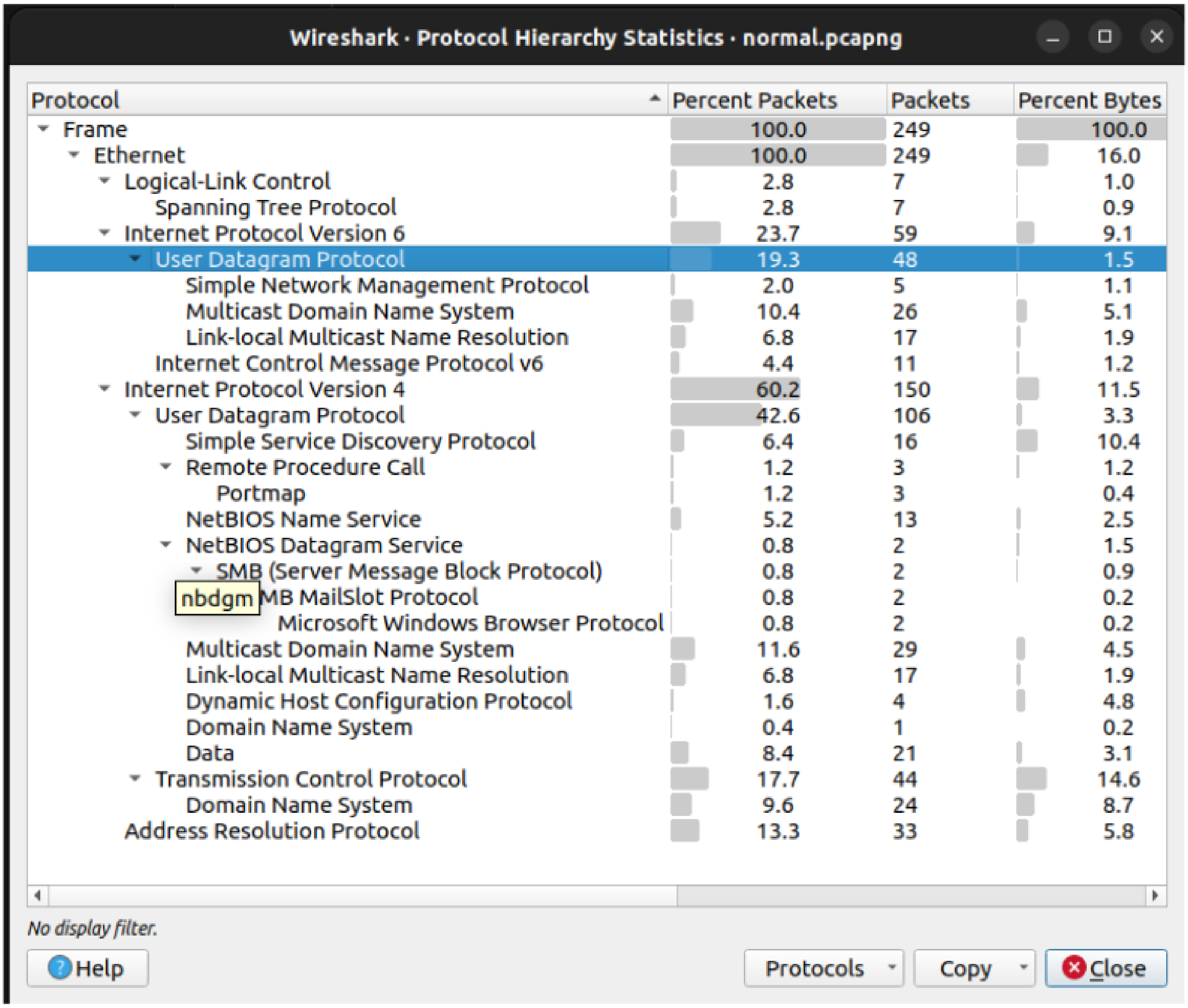
Data Analysis:

Statistical Analysis Record:

Upon capturing network traffic data using Wireshark, statistical analysis was performed to gain insights into the characteristics and patterns of the captured traffic. The following steps were taken:

1. **Accessing Statistical Analysis Tools**: The "Statistics" menu in Wireshark's main menu bar was accessed to explore options for summarizing network traffic.
2. **Protocol Hierarchy Analysis**: The "Protocol Hierarchy" option was selected to identify the protocols used within the captured network traffic. This provided a summary of network protocols used at each TCP/IP stack layer.
3. **Endpoint Examination**: The "Endpoints" option was utilized to identify unique endpoint devices communicating within the network packets. Information such as Ethernet address, IP address, and TCP/UDP port was examined to understand the network topology and communication patterns.
4. **Network Conversations Analysis**: The "Conversations" option was selected to analyze network traffic traveling between endpoints. This facilitated the identification of significant data exchanges between specific IP addresses, aiding in the understanding of network behavior.

By leveraging Wireshark's statistical analysis features, valuable insights were obtained into the distribution, communication patterns, and outliers within the captured network traffic data.

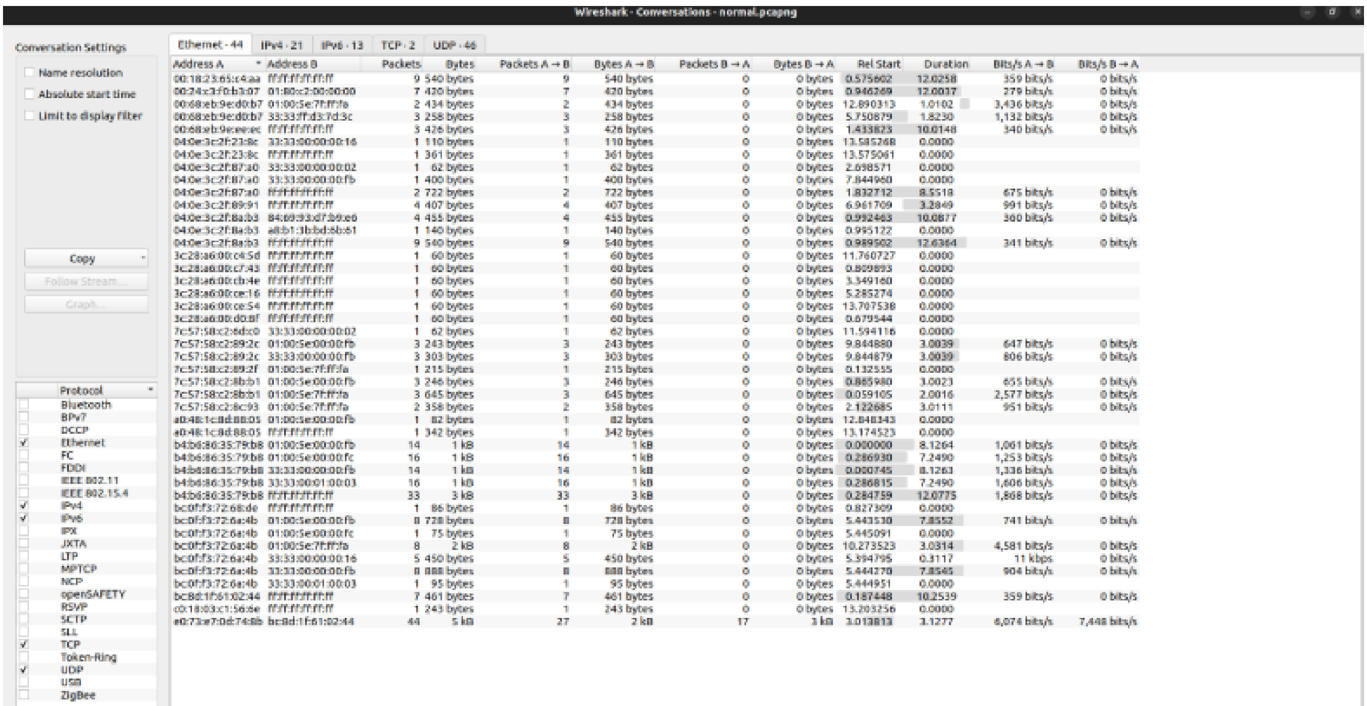


Wireshark · Protocol Hierarchy Statistics · normal.pcapng

Protocol	Percent Packets	Packets	Percent Bytes
Frame	100.0	249	100.0
Ethernet	100.0	249	16.0
Logical-Link Control	2.8	7	1.0
Spanning Tree Protocol	2.8	7	0.9
Internet Protocol Version 6	23.7	59	9.1
User Datagram Protocol	19.3	48	1.5
Simple Network Management Protocol	2.0	5	1.1
Multicast Domain Name System	10.4	26	5.1
Link-local Multicast Name Resolution	6.8	17	1.9
Internet Control Message Protocol v6	4.4	11	1.2
Internet Protocol Version 4	60.2	150	11.5
User Datagram Protocol	42.6	106	3.3
Simple Service Discovery Protocol	6.4	16	10.4
Remote Procedure Call	1.2	3	1.2
Portmap	1.2	3	0.4
NetBIOS Name Service	5.2	13	2.5
NetBIOS Datagram Service	0.8	2	1.5
SMB (Server Message Block Protocol)	0.8	2	0.9
nbdgm MB MailSlot Protocol	0.8	2	0.2
Microsoft Windows Browser Protocol	0.8	2	0.2
Multicast Domain Name System	11.6	29	4.5
Link-local Multicast Name Resolution	6.8	17	1.9
Dynamic Host Configuration Protocol	1.6	4	4.8
Domain Name System	0.4	1	0.2
Data	8.4	21	3.1
Transmission Control Protocol	17.7	44	14.6
Domain Name System	9.6	24	8.7
Address Resolution Protocol	13.3	33	5.8

No display filter.

Help | Protocols | Copy | Close



Wireshark · Conversations · normal.pcapng

Conversation Settings

Name resolution

Absolute start time

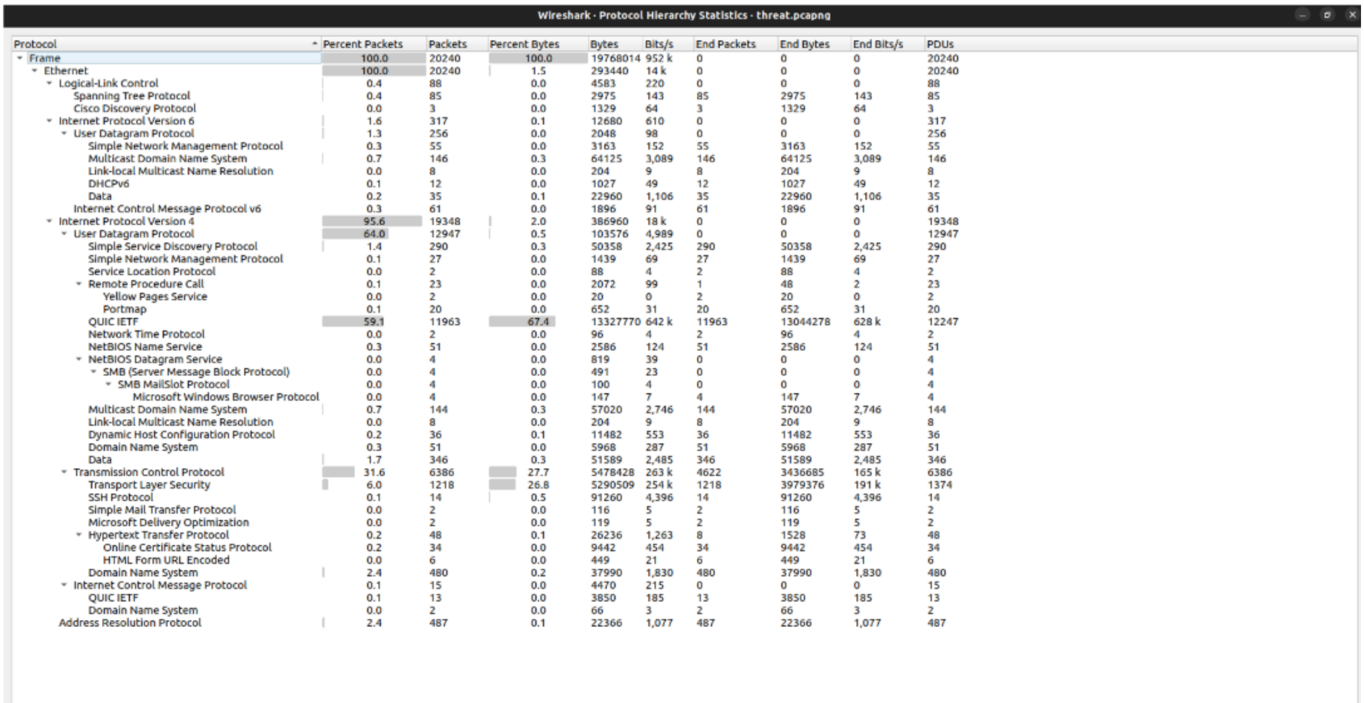
Limit to display filter

Copy | Follow Stream... | Graph...

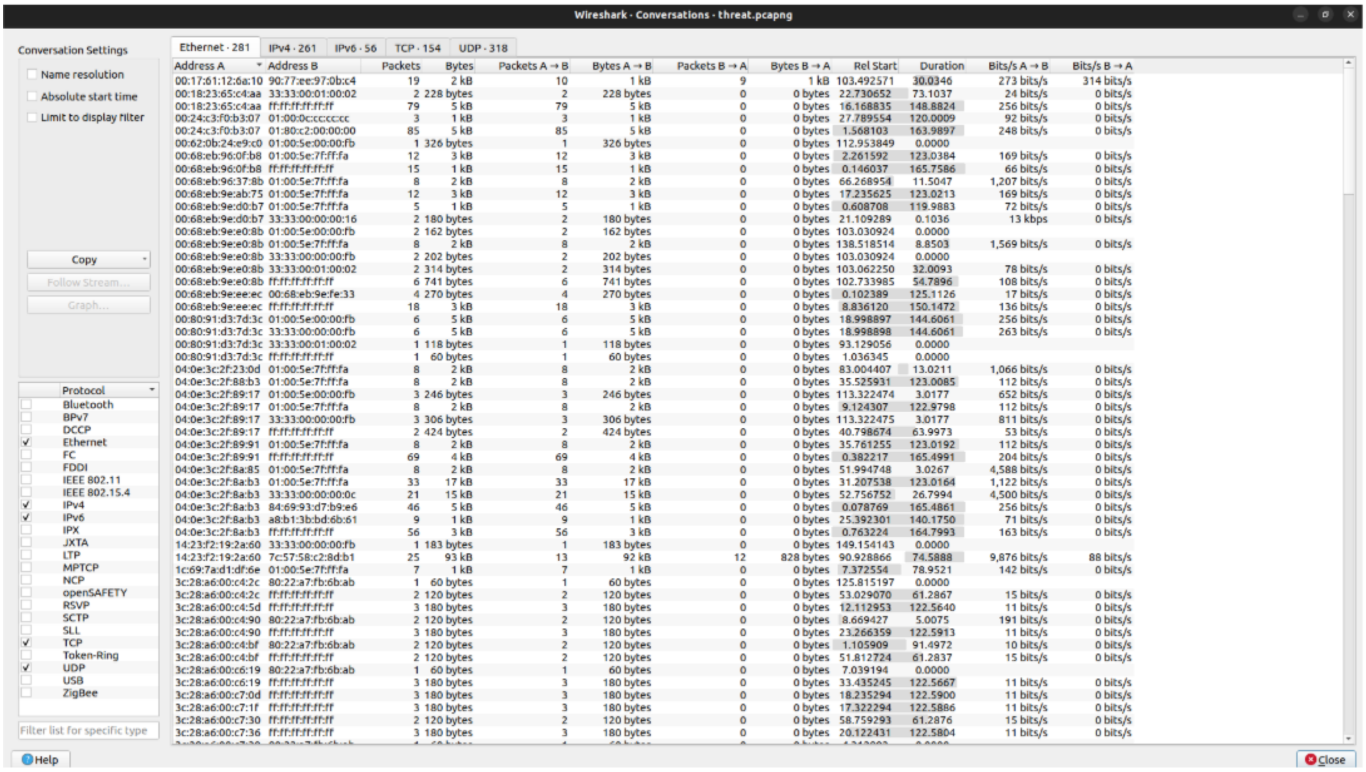
	Protocol
	Bluetooth
	BPv7
	DCCP
✓	Ethernet
	FC
	FDDI
	IEEE 802.11
	IEEE 802.15.4
✓	IPv4
✓	IPv6
	IPX
	JXTA
	LTP
	MPTCP
	NCP
	openSAFETY
	RSVP
	SCTP
	SLL
✓	TCP
	Token-Ring
✓	UDP
	USB
	ZigBee

Ethernet · 44 | IPv4 · 21 | IPv6 · 13 | TCP · 2 | UDP · 46

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
00:18:23:65:c4:aa	ff:ff:ff:ff:ff:ff	9	540 bytes	9	540 bytes	0	0 bytes	0.575602	12.0258	359 bits/s	0 bits/s
00:24:c3:f0:b3:07	01:80:c2:00:00:00	7	420 bytes	7	420 bytes	0	0 bytes	0.946269	12.0037	279 bits/s	0 bits/s
00:68:eb:9e:d0:b7	01:00:5e:7f:ff:fa	2	434 bytes	2	434 bytes	0	0 bytes	12.890313	1.0102	3,436 bits/s	0 bits/s
00:68:eb:9e:d0:b7	33:33:ff:d3:7d:3c	3	258 bytes	3	258 bytes	0	0 bytes	5.750879	1.8230	1,132 bits/s	0 bits/s
00:68:eb:9e:ee:ec	ff:ff:ff:ff:ff:ff	3	426 bytes	3	426 bytes	0	0 bytes	1.433823	10.0148	340 bits/s	0 bits/s
04:0e:3c:2f:23:8c	33:33:00:00:00:16	1	110 bytes	1	110 bytes	0	0 bytes	13.585268	0.0000		
04:0e:3c:2f:23:8c	ff:ff:ff:ff:ff:ff	1	361 bytes	1	361 bytes	0	0 bytes	13.575061	0.0000		
04:0e:3c:2f:87:a0	33:33:00:00:00:02	1	62 bytes	1	62 bytes	0	0 bytes	2.098571	0.0000		
04:0e:3c:2f:87:a0	33:33:00:00:00:fb	1	400 bytes	1	400 bytes	0	0 bytes	7.844960	0.0000		
04:0e:3c:2f:87:a0	ff:ff:ff:ff:ff:ff	2	722 bytes	2	722 bytes	0	0 bytes	1.832712	8.5518	675 bits/s	0 bits/s
04:0e:3c:2f:89:91	ff:ff:ff:ff:ff:ff	4	407 bytes	4	407 bytes	0	0 bytes	6.961709	3.2849	991 bits/s	0 bits/s
04:0e:3c:2f:8a:b3	84:69:93:d7:b9:e6	4	455 bytes	4	455 bytes	0	0 bytes	0.992463	10.0877	360 bits/s	0 bits/s
04:0e:3c:2f:8a:b3	a8:b1:3b:bd:6b:61	1	140 bytes	1	140 bytes	0	0 bytes	0.995122	0.0000		
04:0e:3c:2f:8a:b3	ff:ff:ff:ff:ff:ff	9	540 bytes	9	540 bytes	0	0 bytes	0.989502	12.6364	341 bits/s	0 bits/s
3c:28:a6:00:c4:5d	ff:ff:ff:ff:ff:ff	1	60 bytes	1	60 bytes	0	0 bytes	11.760727	0.0000		
3c:28:a6:00:c7:43	ff:ff:ff:ff:ff:ff	1	60 bytes	1	60 bytes	0	0 bytes	0.809893	0.0000		
3c:28:a6:00:cb:4e	ff:ff:ff:ff:ff:ff	1	60 bytes	1	60 bytes	0	0 bytes	3.349160	0.0000		
3c:28:a6:00:ce:16	ff:ff:ff:ff:ff:ff	1	60 bytes	1	60 bytes	0	0 bytes	5.285274	0.0000		
3c:28:a6:00:ce:54	ff:ff:ff:ff:ff:ff	1	60 bytes	1	60 bytes	0	0 bytes	13.707538	0.0000		
3c:28:a6:00:d0:8f	ff:ff:ff:ff:ff:ff	1	60 bytes	1	60 bytes	0	0 bytes	0.679544	0.0000		
7c:57:58:c2:6d:c0	33:33:00:00:00:02	1	62 bytes	1	62 bytes	0	0 bytes	11.594116	0.0000		
7c:57:58:c2:89:2c	01:00:5e:00:00:fb	3	243 bytes	3	243 bytes	0	0 bytes	9.844880	3.0039	647 bits/s	0 bits/s
7c:57:58:c2:89:2c	33:33:00:00:00:fb	3	303 bytes	3	303 bytes	0	0 bytes	9.844879	3.0039	806 bits/s	0 bits/s
7c:57:58:c2:89:2f	01:00:5e:7f:ff:fa	1	215 bytes	1	215 bytes	0	0 bytes	0.132555	0.0000		
7c:57:58:c2:8b:b1	01:00:5e:00:00:fb	3	246 bytes	3	246 bytes	0	0 bytes	0.885980	3.0023	655 bits/s	0 bits/s
7c:57:58:c2:8b:b1	01:00:5e:7f:ff:fa	3	645 bytes	3	645 bytes	0	0 bytes	0.059105	2.0016	2,577 bits/s	0 bits/s
7c:57:58:c2:8c:93	01:00:5e:7f:ff:fa	2	358 bytes	2	358 bytes	0	0 bytes	2.122685	3.0111	951 bits/s	0 bits/s
a0:48:1c:8d:88:05	01:00:5e:00:00:fb	1	82 bytes	1	82 bytes	0	0 bytes	12.848343	0.0000		
a0:48:1c:8d:88:05	ff:ff:ff:ff:ff:ff	1	342 bytes	1	342 bytes	0	0 bytes	13.174523	0.0000		
b4:b6:86:35:79:b8	01:00:5e:00:00:fb	14	1 kB	14	1 kB	0	0 bytes	0.000000	8.1264	1,061 bits/s	0 bits/s
b4:b6:86:35:79:b8	01:00:5e:00:00:fc	16	1 kB	16	1 kB	0	0 bytes	0.286930	7.2490	1,253 bits/s	0 bits/s
b4:b6:86:35:79:b8	33:33:00:00:00:fb	14	1 kB	14	1 kB	0	0 bytes	0.000745	8.1263	1,336 bits/s	0 bits/s
b4:b6:86:35:79:b8	33:33:00:01:00:03	16	1 kB	16	1 kB	0	0 bytes	0.286815	7.2490	1,606 bits/s	0 bits/s
b4:b6:86:35:79:b8	ff:ff:ff:ff:ff:ff	33	3 kB	33	3 kB	0	0 bytes	0.284759	12.0775	1,868 bits/s	0 bits/s
bc:0f:f3:72:68:de	ff:ff:ff:ff:ff:ff	1	86 bytes	1	86 bytes	0	0 bytes	0.827309	0.0000		
bc:0f:f3:72:6a:4b	01:00:5e:00:00:fb	8	728 bytes	8	728 bytes	0	0 bytes	5.443530	7.8552	741 bits/s	0 bits/s
bc:0f:f3:72:6a:4b	01:00:5e:00:00:fc	1	75 bytes	1	75 bytes	0	0 bytes	5.445091	0.0000		
bc:0f:f3:72:6a:4b	01:00:5e:7f:ff:fa	8	2 kB	8	2 kB	0	0 bytes	10.273523	3.0314	4,581 bits/s	0 bits/s
bc:0f:f3:72:6a:4b	33:33:00:00:00:16	5	450 bytes	5	450 bytes	0	0 bytes	5.394795	0.3117	11 kbps	0 bits/s
bc:0f:f3:72:6a:4b	33:33:00:00:00:fb	8	888 bytes	8	888 bytes	0	0 bytes	5.444270	7.8545	904 bits/s	0 bits/s
bc:0f:f3:72:6a:4b	33:33:00:01:00:03	1	95 bytes	1	95 bytes	0	0 bytes	5.444951	0.0000		
bc:8d:1f:61:02:44	ff:ff:ff:ff:ff:ff	7	461 bytes	7	461 bytes	0	0 bytes	0.187448	10.2539	359 bits/s	0 bits/s
c0:18:03:c1:56:6e	ff:ff:ff:ff:ff:ff	1	243 bytes	1	243 bytes	0	0 bytes	13.203256	0.0000		
e0:73:e7:0d:74:8b	bc:8d:1f:61:02:44	44	5 kB	27	2 kB	17	3 kB	3.013813	3.1277	6,074 bits/s	7,448 bits/s



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	20240	100.0	19768014	952 k	0	0	0	20240
Ethernet	100.0	20240	1.5	293440	14 k	0	0	0	20240
Logical-Link Control	0.4	88	0.0	4583	220	0	0	0	88
Spanning Tree Protocol	0.4	85	0.0	2975	143	85	2975	143	85
Cisco Discovery Protocol	0.0	3	0.0	1329	64	3	1329	64	3
Internet Protocol Version 6	1.6	317	0.1	12680	610	0	0	0	317
User Datagram Protocol	1.3	256	0.0	2048	98	0	0	0	256
Simple Network Management Protocol	0.3	55	0.0	3163	152	55	3163	152	55
Multicast Domain Name System	0.7	146	0.3	64125	3,089	146	64125	3,089	146
Link-local Multicast Name Resolution	0.0	8	0.0	204	9	8	204	9	8
DHCPv6	0.1	12	0.0	1027	49	12	1027	49	12
Data	0.2	35	0.1	22960	1,106	35	22960	1,106	35
Internet Control Message Protocol v6	0.3	61	0.0	1896	91	61	1896	91	61
Internet Protocol Version 4	95.6	19348	2.0	386960	18 k	0	0	0	19348
User Datagram Protocol	64.0	12947	0.5	103576	4,989	0	0	0	12947
Simple Service Discovery Protocol	1.4	290	0.3	50358	2,425	290	50358	2,425	290
Simple Network Management Protocol	0.1	27	0.0	1439	69	27	1439	69	27
Service Location Protocol	0.0	2	0.0	88	4	2	88	4	2
Remote Procedure Call	0.1	23	0.0	2072	99	1	48	2	23
Yellow Pages Service	0.0	2	0.0	20	0	2	20	0	2
Portmap	0.1	20	0.0	652	31	20	652	31	20
QUIC IETF	59.1	11963	67.4	13327770	642 k	11963	13044278	628 k	12247
Network Time Protocol	0.0	2	0.0	96	4	2	96	4	2
NetBIOS Name Service	0.3	51	0.0	2586	124	51	2586	124	51
NetBIOS Datagram Service	0.0	4	0.0	819	39	0	0	0	4
SMB (Server Message Block Protocol)	0.0	4	0.0	491	23	0	0	0	4
SMB MailSlot Protocol	0.0	4	0.0	100	4	0	0	0	4
Microsoft Windows Browser Protocol	0.0	4	0.0	147	7	4	147	7	4
Multicast Domain Name System	0.7	144	0.3	57020	2,746	144	57020	2,746	144
Link-local Multicast Name Resolution	0.0	8	0.0	204	9	8	204	9	8
Dynamic Host Configuration Protocol	0.2	36	0.1	11482	553	36	11482	553	36
Domain Name System	0.3	51	0.0	5968	287	51	5968	287	51
Data	1.7	346	0.3	51589	2,485	346	51589	2,485	346
Transmission Control Protocol	31.6	6386	27.7	5478428	263 k	4622	3436685	165 k	6386
Transport Layer Security	6.0	1218	26.8	5290509	254 k	1218	3979376	191 k	1374
SSH Protocol	0.1	14	0.5	91260	4,396	14	91260	4,396	14
Simple Mail Transfer Protocol	0.0	2	0.0	116	5	2	116	5	2
Microsoft Delivery Optimization	0.0	2	0.0	119	5	2	119	5	2
Hypertext Transfer Protocol	0.2	48	0.1	26236	1,263	8	1528	73	48
Online Certificate Status Protocol	0.2	34	0.0	9442	454	34	9442	454	34
HTML Form URL Encoded	0.0	6	0.0	449	21	6	449	21	6
Domain Name System	2.4	480	0.2	37990	1,830	480	37990	1,830	480
Internet Control Message Protocol	0.1	15	0.0	4470	215	0	0	0	15
QUIC IETF	0.1	13	0.0	3850	185	13	3850	185	13
Domain Name System	0.0	2	0.0	66	3	2	66	3	2
Address Resolution Protocol	2.4	487	0.1	22366	1,077	487	22366	1,077	487



Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
00:17:61:12:6a:10	90:77:ee:97:0b:c4	19	2 kB	10	1 kB	9	1 kB	103.492571	30.0346	273 bits/s	314 bits/s
00:18:23:65:c4:aa	33:33:00:01:00:02	2	228 bytes	2	228 bytes	0	0 bytes	22.730652	73.1037	24 bits/s	0 bits/s
00:18:23:65:c4:aa	ff:ff:ff:ff:ff:ff	79	5 kB	79	5 kB	0	0 bytes	16.168835	148.8824	256 bits/s	0 bits/s
00:24:c3:f0:b3:07	01:00:0c:cc:cc:cc	3	1 kB	3	1 kB	0	0 bytes	27.789554	120.0009	92 bits/s	0 bits/s
00:24:c3:f0:b3:07	01:80:c2:00:00:00	85	5 kB	85	5 kB	0	0 bytes	1.568103	163.9897	248 bits/s	0 bits/s
00:62:0b:24:e9:c0	01:00:5e:00:00:fb	1	326 bytes	1	326 bytes	0	0 bytes	112.953849	0.0000		
00:68:eb:96:0f:b8	01:00:5e:7f:ff:fa	12	3 kB	12	3 kB	0	0 bytes	2.261592	123.0384	169 bits/s	0 bits/s
00:68:eb:96:0f:b8	ff:ff:ff:ff:ff:ff	15	1 kB	15	1 kB	0	0 bytes	0.146037	165.7586	66 bits/s	0 bits/s
00:68:eb:96:37:8b	01:00:5e:7f:ff:fa	8	2 kB	8	2 kB	0	0 bytes	66.268954	11.5047	1,207 bits/s	0 bits/s
00:68:eb:9e:ab:75	01:00:5e:7f:ff:fa	12	3 kB	12	3 kB	0	0 bytes	17.235625	123.0213	169 bits/s	0 bits/s
00:68:eb:9e:d0:b7	01:00:5e:7f:ff:fa	5	1 kB	5	1 kB	0	0 bytes	0.608708	119.9883	72 bits/s	0 bits/s
00:68:eb:9e:d0:b7	33:33:00:00:00:16	2	180 bytes	2	180 bytes	0	0 bytes	21.109289	0.1036	13 kbps	0 bits/s
00:68:eb:9e:e0:8b	01:00:5e:00:00:fb	2	162 bytes	2	162 bytes	0	0 bytes	103.030924	0.0000		
00:68:eb:9e:e0:8b	01:00:5e:7f:ff:fa	8	2 kB	8	2 kB	0	0 bytes	138.518514	8.8503	1,569 bits/s	0 bits/s
00:68:eb:9e:e0:8b	33:33:00:00:00:fb	2	202 bytes	2	202 bytes	0	0 bytes	103.030924	0.0000		
00:68:eb:9e:e0:8b	33:33:00:01:00:02	2	314 bytes	2	314 bytes	0	0 bytes	103.062250	32.0093	78 bits/s	0 bits/s
00:68:eb:9e:e0:8b	ff:ff:ff:ff:ff:ff	6	741 bytes	6	741 bytes	0	0 bytes	102.733985	54.7896	108 bits/s	0 bits/s
00:68:eb:9e:ee:ec	00:68:eb:9e:fe:33	4	270 bytes	4	270 bytes	0	0 bytes	0.102389	125.1126	17 bits/s	0 bits/s
00:68:eb:9e:ee:ec	ff:ff:ff:ff:ff:ff	18	3 kB	18	3 kB	0	0 bytes	8.836120	150.1472	136 bits/s	0 bits/s
00:80:91:d3:7d:3c	01:00:5e:00:00:fb	6	5 kB	6	5 kB	0	0 bytes	18.998897	144.6061	256 bits/s	0 bits/s
00:80:91:d3:7d:3c	33:33:00:00:00:fb	6	5 kB	6	5 kB	0	0 bytes	18.998898	144.6061	263 bits/s	0 bits/s
00:80:91:d3:7d:3c	33:33:00:01:00:02	1	118 bytes	1	118 bytes	0	0 bytes	93.129056	0.0000		
00:80:91:d3:7d:3c	ff:ff:ff:ff:ff:ff	1	60 bytes	1	60 bytes	0	0 bytes	1.036345	0.0000		
04:0e:3c:2f:23:0d	01:00:5e:7f:ff:fa	8	2 kB	8	2 kB	0	0 bytes	83.004407	13.0211	1,066 bits/s	0 bits/s
04:0e:3c:2f:88:b3	01:00:5e:7f:ff:fa	8	2 kB	8	2 kB	0	0 bytes	35.525931	123.0085	112 bits/s	0 bits/s
04:0e:3c:2f:89:17	01:00:5e:00:00:fb	3	246 bytes	3	246 bytes	0	0 bytes	113.322474	3.0177	652 bits/s	0 bits/s
04:0e:3c:2f:89:17	01:00:5e:7f:ff:fa	8	2 kB	8	2 kB	0	0 bytes	9.124307	122.9798	112 bits/s	0 bits/s
04:0e:3c:2f:89:17	33:33:00:00:00:fb	3	306 bytes	3	306 bytes	0	0 bytes	113.322475	3.0177	811 bits/s	0 bits/s
04:0e:3c:2f:89:17	ff:ff:ff:ff:ff:ff	2	424 bytes	2	424 bytes	0	0 bytes	40.798674	63.9973	53 bits/s	0 bits/s
04:0e:3c:2f:89:91	01:00:5e:7f:ff:fa	8	2 kB	8	2 kB	0	0 bytes	35.761255	123.0192	112 bits/s	0 bits/s
04:0e:3c:2f:89:91	ff:ff:ff:ff:ff:ff	69	4 kB	69	4 kB	0	0 bytes	0.382217	165.4991	204 bits/s	0 bits/s
04:0e:3c:2f:8a:85	01:00:5e:7f:ff:fa	8	2 kB	8	2 kB	0	0 bytes	51.994748	3.0267	4,588 bits/s	0 bits/s
04:0e:3c:2f:8a:b3	01:00:5e:7f:ff:fa	33	17 kB	33	17 kB	0	0 bytes	31.207538	123.0164	1,122 bits/s	0 bits/s
04:0e:3c:2f:8a:b3	33:33:00:00:00:0c	21	15 kB	21	15 kB	0	0 bytes	52.756752	26.7994	4,500 bits/s	0 bits/s
04:0e:3c:2f:8a:b3	84:69:93:d7:b9:e6	46	5 kB	46	5 kB	0	0 bytes	0.078769	165.4861	256 bits/s	0 bits/s
04:0e:3c:2f:8a:b3	a8:b1:3b:bd:6b:61	9	1 kB	9	1 kB	0	0 bytes	25.392301	140.1750	71 bits/s	0 bits/s
04:0e:3c:2f:8a:b3	ff:ff:ff:ff:ff:ff	56	3 kB	56	3 kB	0	0 bytes	0.763224	164.7993	163 bits/s	0 bits/s
14:23:f2:19:2a:60	33:33:00:00:00:fb	1	183 bytes	1	183 bytes	0	0 bytes	149.154143	0.0000		
14:23:f2:19:2a:60	7c:57:58:c2:8d:b1	25	93 kB	13	92 kB	12	828 bytes	90.928866	74.5888	9,876 bits/s	88 bits/s
1c:69:7a:d1:df:6e	01:00:5e:7f:ff:fa	7	1 kB	7	1 kB	0	0 bytes	7.372554	78.9521	142 bits/s	0 bits/s
3c:28:a6:00:c4:2c	80:22:a7:fb:6b:ab	1	60 bytes	1	60 bytes	0	0 bytes	125.815197	0.0000		
3c:28:a6:00:c4:2c	ff:ff:ff:ff:ff:ff	2	120 bytes	2	120 bytes	0	0 bytes	53.029070	61.2867	15 bits/s	0 bits/s
3c:28:a6:00:c4:5d	ff:ff:ff:ff:ff:ff	3	180 bytes	3	180 bytes	0	0 bytes	12.112953	122.5640	11 bits/s	0 bits/s
3c:28:a6:00:c4:90	80:22:a7:fb:6b:ab	2	120 bytes	2	120 bytes	0	0 bytes	8.669427	5.0075	191 bits/s	0 bits/s
3c:28:a6:00:c4:90	ff:ff:ff:ff:ff:ff	3	180 bytes	3	180 bytes	0	0 bytes	23.266359	122.5913	11 bits/s	0 bits/s
3c:28:a6:00:c4:bf	80:22:a7:fb:6b:ab	2	120 bytes	2	120 bytes	0	0 bytes	1.105909	91.4972	10 bits/s	0 bits/s
3c:28:a6:00:c4:bf	ff:ff:ff:ff:ff:ff	2	120 bytes	2	120 bytes	0	0 bytes	51.812724	61.2837	15 bits/s	0 bits/s
3c:28:a6:00:c6:19	80:22:a7:fb:6b:ab	1	60 bytes	1	60 bytes	0	0 bytes	7.039194	0.0000		
3c:28:a6:00:c6:19	ff:ff:ff:ff:ff:ff	3	180 bytes	3	180 bytes	0	0 bytes	33.435245	122.5667	11 bits/s	0 bits/s
3c:28:a6:00:c7:0d	ff:ff:ff:ff:ff:ff	3	180 bytes	3	180 bytes	0	0 bytes	18.235294	122.5900	11 bits/s	0 bits/s
3c:28:a6:00:c7:1f	ff:ff:ff:ff:ff:ff	3	180 bytes	3	180 bytes	0	0 bytes	17.322294	122.5886	11 bits/s	0 bits/s
3c:28:a6:00:c7:30	ff:ff:ff:ff:ff:ff	2	120 bytes	2	120 bytes	0	0 bytes	58.759293	61.2876	15 bits/s	0 bits/s
3c:28:a6:00:c7:36	ff:ff:ff:ff:ff:ff	3	180 bytes	3	180 bytes	0	0 bytes	20.122431	122.5804	11 bits/s	0 bits/s

Techniques:

TCP Stream Analysis Record:

For comprehensive network analysis, Wireshark provides the capability to follow TCP streams, enabling visualization of the communication between devices and the data exchanged. The following steps outline the process:

1. **Selecting Packet for TCP Stream Analysis**: Identify a packet within the Wireshark packet list pane that represents the beginning of a TCP conversation.

2. **Following TCP Stream**: Right-click on the selected packet and choose the option "Follow > TCP Stream." This action prompts Wireshark to generate a summary view of the TCP stream in a new window.

3. **Analysis and Visualization**: Wireshark automatically applies a display filter to show packets only from the selected TCP stream, facilitating focused analysis. Additionally, the generated summary view highlights the HTTP messages exchanged between the client and server, providing insights into the nature of the communication.

By following TCP streams in Wireshark, users can effectively troubleshoot network issues, uncover hidden information, and gain a comprehensive understanding of network communication patterns.

In the process of network security monitoring, Wireshark proves instrumental in detecting cyber attacks, including brute force attempts, port scans, and data exfiltration. The following steps outline the identification of an FTP brute force attack using Wireshark:

1. **Initial Filter for FTP Traffic**: Apply the Wireshark display filter ftp.response.code == 530 to isolate FTP traffic indicating authentication failure (FTP response code 530, "Not logged in"). Repeated occurrences of this error with the same username but different passwords suggest a brute force attack targeting FTP authentication.

2. **Examination of Password Attempts**: To assess the variation in password attempts, right-click on a network packet and select "Follow > TCP Stream." This action allows the visualization of multiple password attempts made by the same user during the authentication process.

3. **Analysis of Attack Pattern**: Observation of the TCP stream reveals repeated attempts by the same user to log in using different passwords, indicative of a password spraying attack against the user account "stationx-admin."

4. **Verification of Login Success**: To ascertain the success of any login attempts, employ the display filter ftp.response.code == 230 to identify FTP responses indicating successful authentication ("User logged in, proceed"). In the analyzed scenario, no successful login attempts were detected.

Through the meticulous examination of network traffic using Wireshark and appropriate display filters, the identification of FTP brute force attacks and other malicious activities becomes feasible, empowering blue teams to proactively defend against cyber threats.

ARP Poisoning & Man In The Middle:

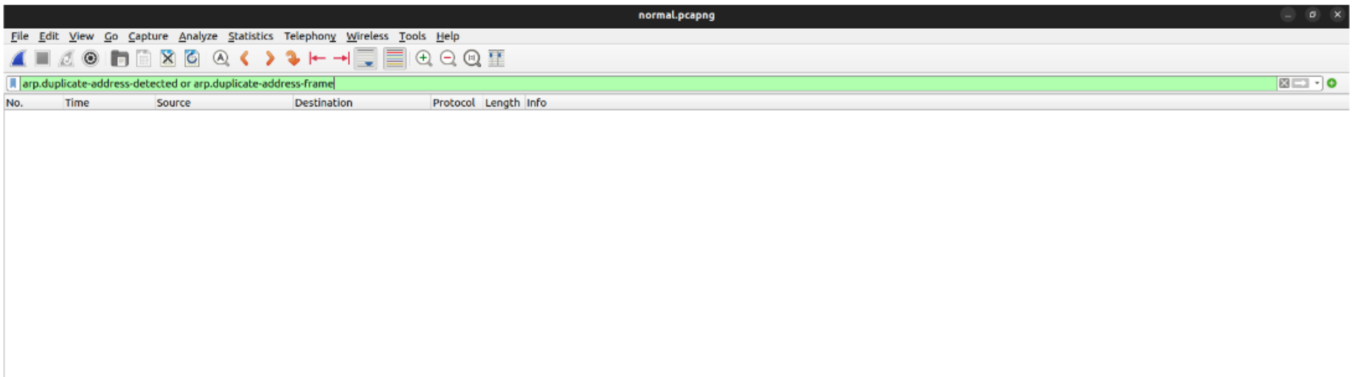
In the investigation of ARP poisoning and man-in-the-middle attacks using my pcap file, the focus is on determining the number of ARP requests crafted by the attacker. The following steps were taken to analyze the network traffic:

1. **Identification of ARP Requests Crafted by Attacker**: The command `arp.duplicate-address-detected` or `arp.duplicate-addressframe` was executed to identify ARP packets related to duplicate address detection. Subsequently, the Sender MAC Address was selected for further analysis.

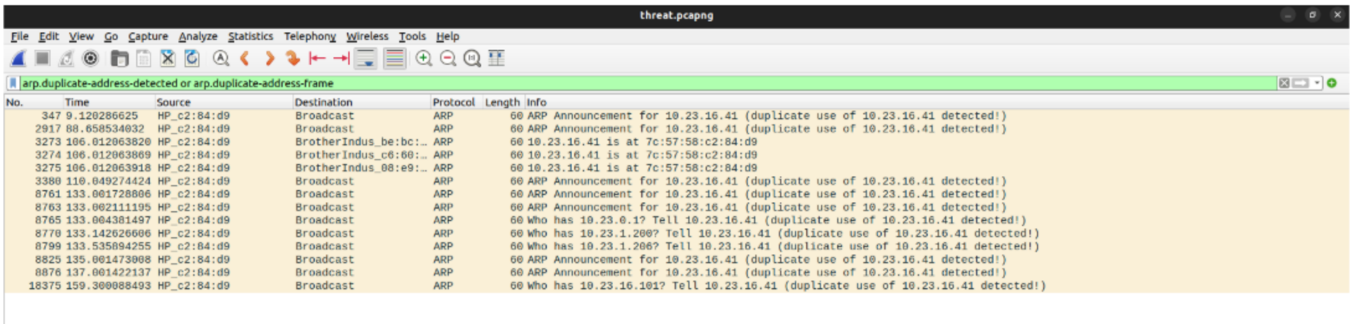
2. **Filtering ARP Requests Crafted by Attacker**: The command `eth.src == sender mac and arp.opcode == 1` was executed to filter ARP requests crafted by the attacker. This filter specifically targets ARP packets with the Sender MAC Address matching and ARP opcode indicating ARP request (opcode 1).

By executing these commands and applying the specified filters, the number of ARP requests crafted by the attacker can be accurately determined, providing crucial insights into the extent of ARP poisoning activity and potential man-in-the-middle attacks within the network.

normal pcap



threat pcap

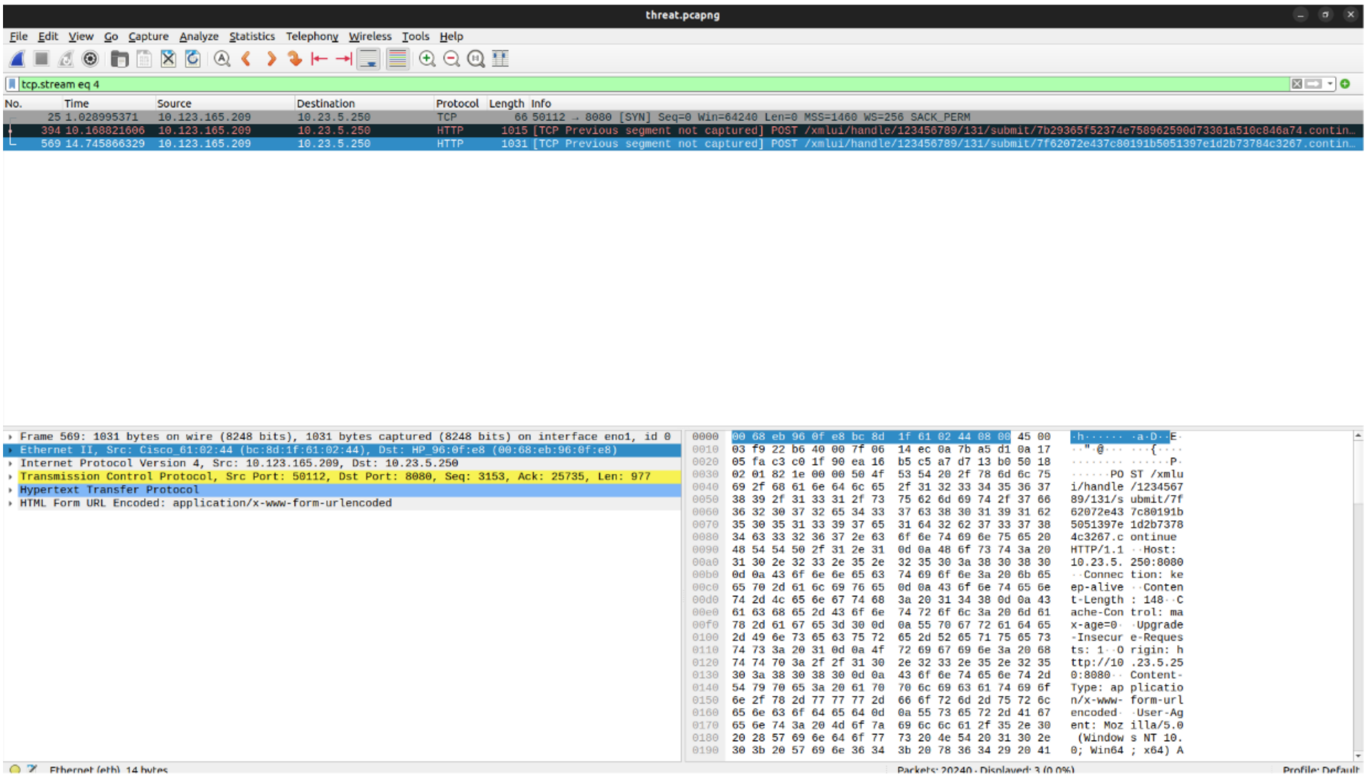


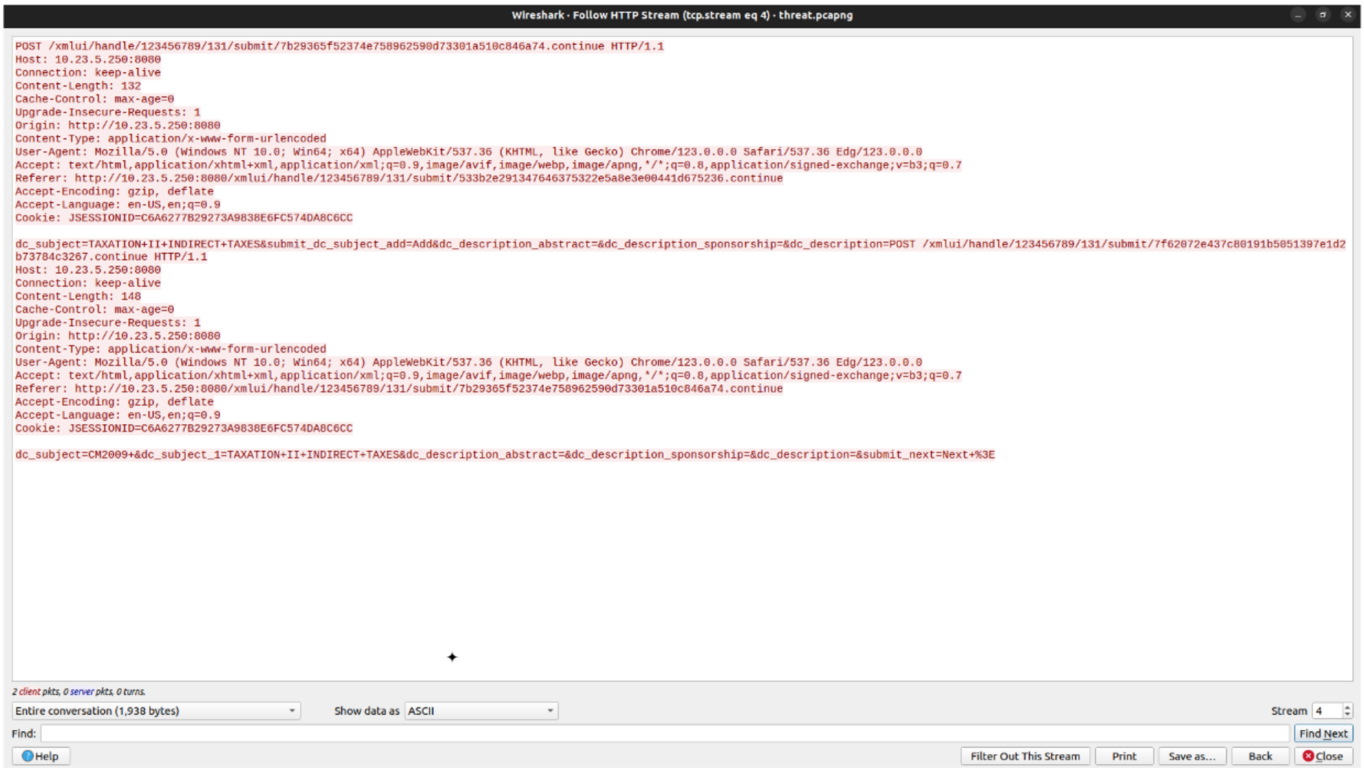
Username & Password Sniffing Analysis:

To determine the number of sniffed username and password entries, the following steps were undertaken:

1. **Selection of TCP Stream**: The command `tcp.stream eq 4` was executed to isolate TCP stream number 4.
2. **Examination of HTTP Requests**: Packet number 1107 was selected, and the option "Follow -> TCP Stream" was chosen to inspect the HTTP stream content.
3. **Identification of HTTP POST Requests**: In the TCP stream window, the command `http.host ==` and `http.request.method == “POST”` was applied to filter HTTP POST requests targeting the specified hostname.
4. **Verification of Passwords**: Each packet within the filtered HTTP POST requests was meticulously examined to identify any transmitted passwords.

By meticulously analyzing the HTTP stream content and verifying the HTTP POST requests targeting the designated hostname, potential instances of username and password sniffing within the network can be identified and further investigated for security implications.


threat.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.stream eq 4
No. Time Source Destination Protocol Length Info
25 1.028995371 10.123.165.209 10.23.5.250 TCP 66 50112 → 8080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
394 10.168821606 10.123.165.209 10.23.5.250 HTTP 1015 [TCP Previous segment not captured] POST /xmlui/handle/123456789/131/submit/7b29365f52374e758962590d73301a510c846a74.contin…
569 14.745866329 10.123.165.209 10.23.5.250 HTTP 1031 [TCP Previous segment not captured] POST /xmlui/handle/123456789/131/submit/7f62072e437c80191b5051397e1d2b73784c3267.contin…
Frame 569: 1031 bytes on wire (8248 bits), 1031 bytes captured (8248 bits) on interface eno1, id 0
Ethernet II, Src: Cisco_61:02:44 (bc:8d:1f:61:02:44), Dst: HP_96:0f:e8 (00:68:eb:96:0f:e8)
Internet Protocol Version 4, Src: 10.123.165.209, Dst: 10.23.5.250
Transmission Control Protocol, Src Port: 50112, Dst Port: 8080, Seq: 3153, Ack: 25735, Len: 977
Hypertext Transfer Protocol
HTML Form URL Encoded: application/x-www-form-urlencoded
Packets: 20240 · Displayed: 3 (0.0%)
Profile: Default


Wireshark · Follow HTTP Stream (tcp.stream eq 4) · threat.pcapng
POST /xmlui/handle/123456789/131/submit/7b29365f52374e758962590d73301a510c846a74.continue HTTP/1.1
Host: 10.23.5.250:8080
Connection: keep-alive
Content-Length: 132
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://10.23.5.250:8080
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36 Edg/123.0.0.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://10.23.5.250:8080/xmlui/handle/123456789/131/submit/533b2e291347646375322e5a8e3e00441d675236.continue
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: JSESSIONID=C6A6277B29273A9838E6FC574DA8C6CC
dc_subject=TAXATION+II+INDIRECT+TAXES&submit_dc_subject_add=Add&dc_description_abstract=&dc_description_sponsorship=&dc_description=POST /xmlui/handle/123456789/131/submit/7f62072e437c80191b5051397e1d2b73784c3267.continue HTTP/1.1
Host: 10.23.5.250:8080
Connection: keep-alive
Content-Length: 148
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://10.23.5.250:8080
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36 Edg/123.0.0.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://10.23.5.250:8080/xmlui/handle/123456789/131/submit/7b29365f52374e758962590d73301a510c846a74.continue
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: JSESSIONID=C6A6277B29273A9838E6FC574DA8C6CC
dc_subject=CM2009+&dc_subject_1=TAXATION+II+INDIRECT+TAXES&dc_description_abstract=&dc_description_sponsorship=&dc_description=&submit_next=Next+%3E
2 client pkts, 0 server pkts, 0 turns.
Entire conversation (1,938 bytes)
Show data as ASCII
Stream 4
Find:
Find Next
Help
Filter Out This Stream
Print
Save as…
Back
Close

Analysis of SYN Flood Attack

A SYN flood attack is a type of DDoS attack where the attacker floods a victim server with a large number of SYN packets, overwhelming its resources and disrupting normal operation. The objective is to exhaust the server's capacity to handle incoming connections, rendering it unable to service legitimate requests.

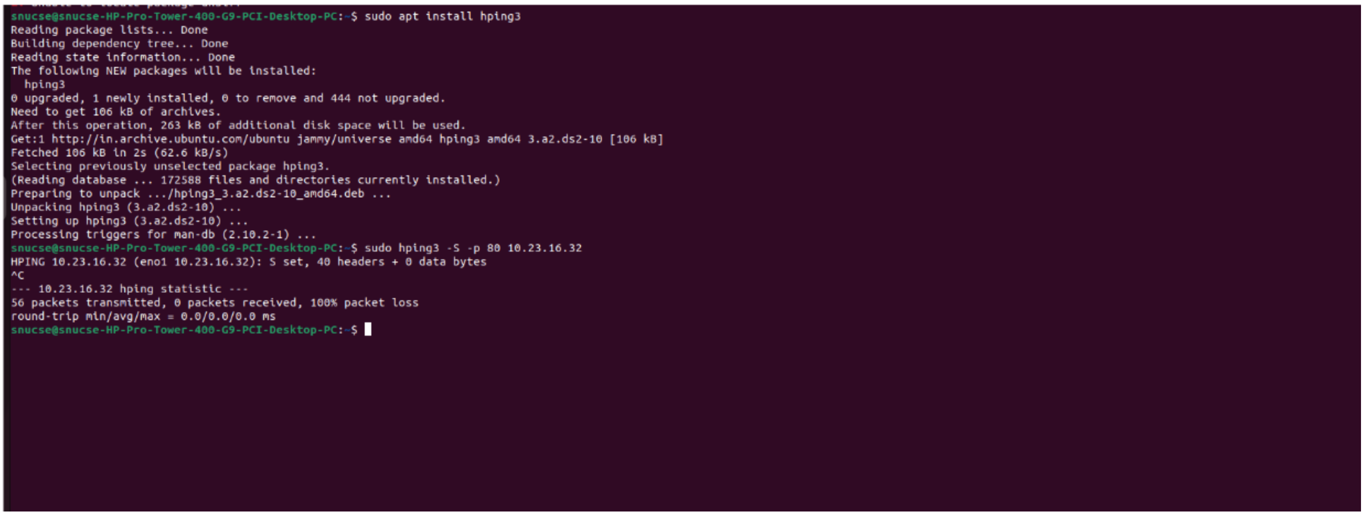
To analyze a SYN flood attack using Wireshark alongside the hping3 tool, the following steps were taken:

1. **Preparation**:
    - The hping3 tool was used to generate a flood of SYN packets directed towards the victim's IP address.
    - Simultaneously, Wireshark was started to capture and analyze the incoming network traffic.
2. **Filtering SYN Packets**:
    - In Wireshark, the filter `tcp.flags.syn == 1` was applied to isolate and focus on SYN packets specifically.
    - This filter allows for the identification of SYN packets within the captured traffic, which are indicative of the initiation phase of the TCP handshake.

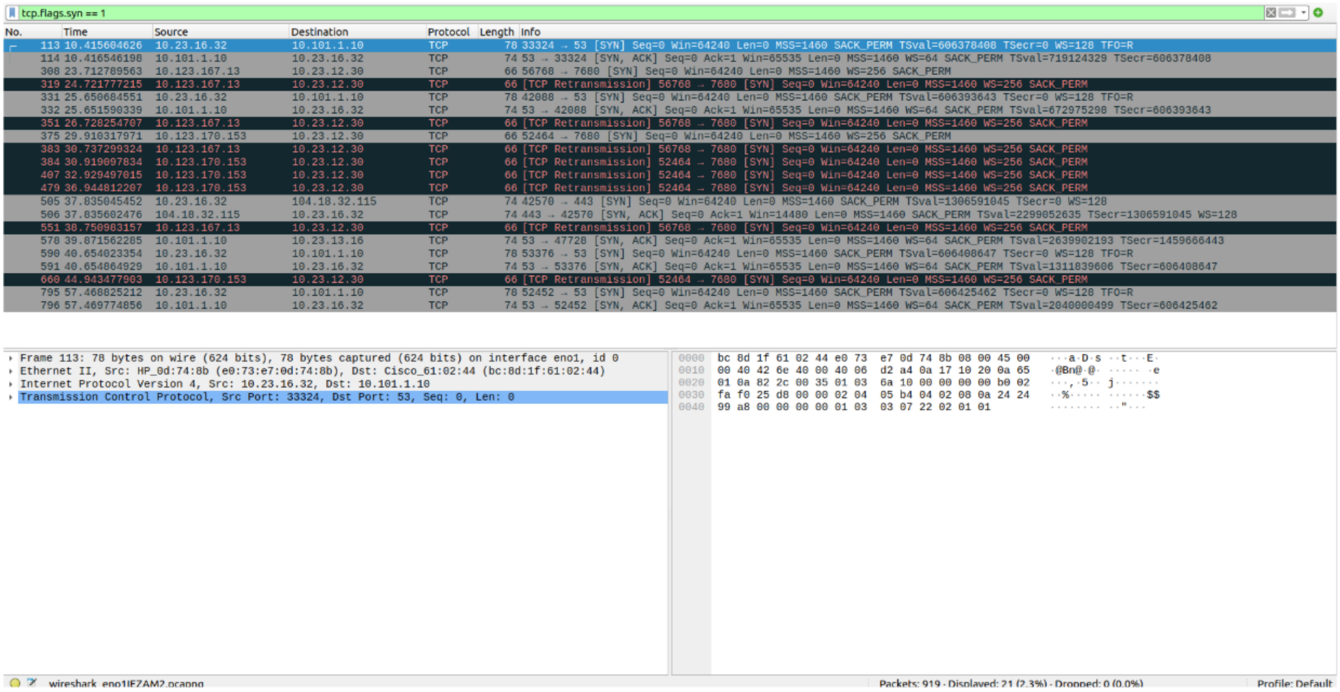
Observations: The captured traffic revealed a significant influx of SYN packets directed towards the victim server. These SYN packets were observed to be of the same size, indicating a consistent pattern in the attack. Additionally, there was a notable absence of lag time between successive SYN packets, suggesting a rapid and continuous flood of SYN requests.

Impact: The SYN flood attack effectively inundates the victim server with a barrage of connection requests, causing it to expend resources on processing incomplete connection attempts. As a result, the victim server's processor and memory usage escalate rapidly, eventually leading to resource exhaustion and service degradation or outage.

By analyzing the characteristics of the captured traffic, including the volume, consistency, and timing of SYN packets, it becomes evident how a SYN flood attack can disrupt network services and compromise the availability of targeted servers. Such insights are crucial for implementing effective mitigation strategies and bolstering network defenses against DDoS attacks.



snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~$ sudo apt install hping3
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
hping3
0 upgraded, 1 newly installed, 0 to remove and 444 not upgraded.
Need to get 106 kB of archives.
After this operation, 263 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 hping3 amd64 3.a2.ds2-10 [106 kB]
Fetched 106 kB in 2s (62.6 kB/s)
Selecting previously unselected package hping3.
(Reading database ... 172588 files and directories currently installed.)
Preparing to unpack .../hping3_3.a2.ds2-10_amd64.deb ...
Unpacking hping3 (3.a2.ds2-10) ...
Setting up hping3 (3.a2.ds2-10) ...
Processing triggers for man-db (2.10.2-1) ...
snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~$ sudo hping3 -S -p 80 10.23.16.32
HPING 10.23.16.32 (eno1 10.23.16.32): S set, 40 headers + 0 data bytes
^C
--- 10.23.16.32 hping statistic ---
56 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~$



tcp.flags.syn == 1
No. Time Source Destination Protocol Length Info
113 10.415604626 10.23.16.32 10.101.1.10 TCP 78 33324 → 53 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=606378408 TSecr=0 WS=128 TFO=R
114 10.416546198 10.101.1.10 10.23.16.32 TCP 74 53 → 33324 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM TSval=719124329 TSecr=606378408
308 23.712789563 10.123.167.13 10.23.12.30 TCP 66 56768 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
319 24.721777215 10.123.167.13 10.23.12.30 TCP 66 [TCP Retransmission] 56768 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
331 25.650684551 10.23.16.32 10.101.1.10 TCP 78 42088 → 53 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=606393643 TSecr=0 WS=128 TFO=R
332 25.651590339 10.101.1.10 10.23.16.32 TCP 74 53 → 42088 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM TSval=672975298 TSecr=606393643
351 26.728254707 10.123.167.13 10.23.12.30 TCP 66 [TCP Retransmission] 56768 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
375 29.910317971 10.123.170.153 10.23.12.30 TCP 66 52464 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
383 30.737299324 10.123.167.13 10.23.12.30 TCP 66 [TCP Retransmission] 56768 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
384 30.919097834 10.123.170.153 10.23.12.30 TCP 66 [TCP Retransmission] 52464 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
407 32.929497015 10.123.170.153 10.23.12.30 TCP 66 [TCP Retransmission] 52464 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
479 36.944812207 10.123.170.153 10.23.12.30 TCP 66 [TCP Retransmission] 52464 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
505 37.835045452 10.23.16.32 104.18.32.115 TCP 74 42570 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1306591045 TSecr=0 WS=128
506 37.835602476 104.18.32.115 10.23.16.32 TCP 74 443 → 42570 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM TSval=2299052635 TSecr=1306591045 WS=128
551 38.750983157 10.123.167.13 10.23.12.30 TCP 66 [TCP Retransmission] 56768 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
578 39.871562285 10.101.1.10 10.23.13.16 TCP 74 53 → 47728 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM TSval=2639902193 TSecr=1459666443
590 40.654023354 10.23.16.32 10.101.1.10 TCP 78 53376 → 53 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=606408647 TSecr=0 WS=128 TFO=R
591 40.654864929 10.101.1.10 10.23.16.32 TCP 74 53 → 53376 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM TSval=1311839606 TSecr=606408647
660 44.943477903 10.123.170.153 10.23.12.30 TCP 66 [TCP Retransmission] 52464 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
795 57.468825212 10.23.16.32 10.101.1.10 TCP 78 52452 → 53 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=606425462 TSecr=0 WS=128 TFO=R
796 57.469774856 10.101.1.10 10.23.16.32 TCP 74 53 → 52452 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM TSval=2040000499 TSecr=606425462
Frame 113: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface eno1, id 0
Ethernet II, Src: HP_0d:74:8b (e0:73:e7:0d:74:8b), Dst: Cisco_61:02:44 (bc:8d:1f:61:02:44)
Internet Protocol Version 4, Src: 10.23.16.32, Dst: 10.101.1.10
Transmission Control Protocol, Src Port: 33324, Dst Port: 53, Seq: 0, Len: 0
0000 bc 8d 1f 61 02 44 e0 73 e7 0d 74 8b 08 00 45 00
0010 00 40 42 6e 40 00 40 06 d2 a4 0a 17 10 20 0a 65
0020 01 0a 82 2c 00 35 01 03 6a 10 00 00 00 00 b0 02
0030 fa f0 25 d8 00 00 02 04 05 b4 04 02 08 0a 24 24
0040 99 a8 00 00 00 00 01 03 03 07 22 02 01 01
wireshark_eno1IEZAM2.pcapng
Packets: 919 · Displayed: 21 (2.3%) · Dropped: 0 (0.0%)
Profile: Default

Analysis of DOS Attack

In the simulated Denial of Service (DoS) attack scenario, the macof tool from the Dsniff suite toolkit was utilized to flood a surrounding device's switch with MAC addresses, generating excessive network traffic. The analysis was performed using Wireshark to inspect the traffic patterns and characteristics.

1. **DoS Attack with Standard Network Traffic**:
    - The observed traffic consists of repeated requests originating from one IP address to another device with consistent data size. This pattern indicates a standard network DoS attack, where the attacker overwhelms the target device with a high volume of legitimate-looking requests, thereby disrupting its normal operations.
2. **DDoS Attack with Fake Source and Destination IP Addresses**:
    - For the DDoS attack simulation, the macof tool was employed again to generate traffic. This time, the traffic includes packets with fake source and destination IP addresses, sending numerous packets with similar data sizes. Such a traffic pattern is characteristic of a Distributed Denial of Service (DDoS) attack, where multiple compromised devices (botnets) inundate the target system with malicious traffic, making it difficult to mitigate the attack and causing severe service disruption.

By analyzing the traffic patterns observed in Wireshark, it becomes evident whether the attack is a traditional DoS attack or a more sophisticated DDoS attack, allowing for appropriate mitigation strategies to be implemented to defend against such malicious activities.

snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC: ~

snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC: ~ × | snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC: ~ × | snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC: ~ × | snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC: ~ ×

```
snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~$ sudo apt-get install dsniff
[sudo] password for snucse:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libnet1 libnids1.21
The following NEW packages will be installed:
  dsniff libnet1 libnids1.21
0 upgraded, 3 newly installed, 0 to remove and 444 not upgraded.
Need to get 175 kB of archives.
After this operation, 678 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu jammy/main amd64 libnet1 amd64 1.1.6+dfsg-3.1build3 [46.9 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 libnids1.21 amd64 1.25-1 [22.0 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 dsniff amd64 2.4b1+debian-30build1 [106 kB]
Fetched 175 kB in 2s (72.2 kB/s)
Selecting previously unselected package libnet1:amd64.
(Reading database ... 172618 files and directories currently installed.)
Preparing to unpack .../libnet1_1.1.6+dfsg-3.1build3_amd64.deb ...
Unpacking libnet1:amd64 (1.1.6+dfsg-3.1build3) ...
Selecting previously unselected package libnids1.21:amd64.
Preparing to unpack .../libnids1.21_1.25-1_amd64.deb ...
Unpacking libnids1.21:amd64 (1.25-1) ...
Selecting previously unselected package dsniff.
Preparing to unpack .../dsniff_2.4b1+debian-30build1_amd64.deb ...
Unpacking dsniff (2.4b1+debian-30build1) ...
Setting up libnet1:amd64 (1.1.6+dfsg-3.1build3) ...
Setting up libnids1.21:amd64 (1.25-1) ...
Setting up dsniff (2.4b1+debian-30build1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
Processing triggers for man-db (2.10.2-1) ...
snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~$ sudo macof -i eno1 -n 100
e5:4e:90:44:4c:49 69:44:3:36:d4:f5 0.0.0.0.25055 > 0.0.0.0.60336: S 651029573:651029573(0) win 512
21:7d:bd:24:42:ed 89:2f:31:27:b4:b8 0.0.0.0.26526 > 0.0.0.0.17384: S 285768584:285768584(0) win 512
4b:c2:73:5b:b4:1f a8:4d:fe:7:9e:25 0.0.0.0.28608 > 0.0.0.0.14485: S 1710614051:1710614051(0) win 512
a7:cb:51:41:8f:5f 52:18:47:36:65:87 0.0.0.0.38689 > 0.0.0.0.46915: S 2108200033:2108200033(0) win 512
56:84:c6:3:e8:33 aa:2:cd:14:d6:de 0.0.0.0.46330 > 0.0.0.0.54330: S 1276197243:1276197243(0) win 512
b6:fa:83:15:4c:23 8b:cc:b4:14:e1:65 0.0.0.0.41725 > 0.0.0.0.14506: S 563252284:563252284(0) win 512
27:4a:d0:73:c7:ba 9:8f:9c:3f:90:8c 0.0.0.0.45801 > 0.0.0.0.55630: S 1507590514:1507590514(0) win 512
29:f4:59:f:6b:e3 a2:22:39:56:fb:1a 0.0.0.0.56479 > 0.0.0.0.63293: S 1095628901:1095628901(0) win 512
d4:7e:f5:39:ee:49 80:a1:f7:63:9:9a 0.0.0.0.3907 > 0.0.0.0.53964: S 1265023910:1265023910(0) win 512
58:9:8b:38:5:45 75:a9:64:4e:8a:10 0.0.0.0.46228 > 0.0.0.0.1151: S 1241565525:1241565525(0) win 512
26:f6:4b:53:d9:16 d4:bd:79:35:4e:42 0.0.0.0.42950 > 0.0.0.0.36467: S 383227383:383227383(0) win 512
4a:c6:55:2a:98:65 85:8b:e2:1f:17:76 0.0.0.0.17530 > 0.0.0.0.36581: S 933983832:933983832(0) win 512
11:77:e5:54:e7:ef 3c:c7:50:49:d0:43 0.0.0.0.32215 > 0.0.0.0.46341: S 1377309834:1377309834(0) win 512
b7:86:30:3e:14:30 c3:b8:43:6f:36:b8 0.0.0.0.28103 > 0.0.0.0.6697: S 1520995750:1520995750(0) win 512
e7:5b:bc:1f:ee:58 cb:e6:62:19:b:27 0.0.0.0.579 > 0.0.0.0.4993: S 574581877:574581877(0) win 512
f0:82:18:5c:ac:5a 76:a4:b7:7b:95:6f 0.0.0.0.53687 > 0.0.0.0.40020: S 1248791349:1248791349(0) win 512
41:31:40:7c:8c:9a ea:98:84:4f:88:55 0.0.0.0.44899 > 0.0.0.0.16522: S 1964605409:1964605409(0) win 512
82:e4:9d:76:9d:f0 b0:38:f0:37:4d:db 0.0.0.0.23325 > 0.0.0.0.39403: S 490386924:490386924(0) win 512
67:96:cb:33:2f:b2 3f:62:3f:62:60:6a 0.0.0.0.2816 > 0.0.0.0.29565: S 1661312073:1661312073(0) win 512
f1:8a:e5:20:d1:2c 89:94:a7:66:45:94 0.0.0.0.61206 > 0.0.0.0.12732: S 284786052:284786052(0) win 512
```

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.stream eq 5

No.	Time	Source	Destination	Protocol	Length	Info
101	7.300639057	10.23.16.32	172.64.149.23	TCP	66	60302 → 80 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=2647731766 TSecr=389680465
102	7.300905559	172.64.149.23	10.23.16.32	TCP	66	[TCP ACKed unseen segment] 80 → 60302 [ACK] Seq=1 Ack=2 Win=122 Len=0 TSval=389681489 TSecr=2647639956
318	17.540689255	10.23.16.32	172.64.149.23	TCP	66	[TCP Dup ACK 101#1] 60302 → 80 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=2647742006 TSecr=389681489
319	17.540930494	172.64.149.23	10.23.16.32	TCP	66	[TCP Dup ACK 102#1] 80 → 60302 [ACK] Seq=1 Ack=2 Win=122 Len=0 TSval=389682513 TSecr=2647639956

Results and Interpretation

Result:

- Utilized Wireshark to perform live packet capture of two pcap files, representing normal and potentially threatening network scenarios.
- No findings of brute source or other attacks were observed, as the search for `ftp.response.code == 530` yielded no results.
- In the analysis of ARP and Man-in-the-Middle attacks, no threats or vulnerabilities were identified in the normal pcap file. However, duplicate requests were detected in the threat pcap file, indicating a potential security concern.
- Attempted to analyze HTTP stream and post content for further insights.
- Conducted a SYN flood attack simulation using the hping3 tool and captured the traffic in Wireshark.
- Simulated a Denial of Service (DoS) attack using the macof tool, observing similar requests with fake source and IP destination addresses.

Interpretation:

- The absence of findings in the normal pcap file suggests that the network was operating within expected parameters, with no evident signs of malicious activity or anomalies.
- Detection of duplicate requests in the threat pcap file during the analysis of ARP and Man-in-the-Middle attacks raises concerns about security threats, possibly indicating ARP poisoning or unauthorized network activity.
- Further analysis of HTTP stream and post content could provide insights into potential data breaches or unauthorized access attempts.
- The SYN flood attack simulation demonstrated the ability to overwhelm a server with a large volume of SYN packets, highlighting the importance of robust network defense mechanisms against DDoS attacks.
- Observation of similar requests with fake source and IP destination addresses in the DoS attack simulation indicates a deliberate attempt to flood the target device with malicious traffic, underscoring the need for proactive security measures to mitigate such threats.
- Overall, the results suggest a proactive approach to network security monitoring and threat detection, emphasizing the importance of continuous monitoring, analysis, and response to safeguard against potential cyber threats and attacks.

Conclusion

- Demonstrated critical thinking skills by refining the project objectives to focus on protocol-based classification techniques for network traffic analysis.
- Successfully collected live network traffic data using Wireshark, showcasing problem-solving abilities in selecting appropriate capture interfaces and managing packet capture processes.
- Applied manual classification methods to extract protocol headers and categorize network packets, indicating analytical thinking in identifying and organizing data for analysis.
- Conducted statistical analysis of network traffic to identify patterns and anomalies, highlighting the ability to interpret data and draw meaningful insights.
- Utilized Wireshark's features effectively, such as protocol hierarchy analysis and TCP stream analysis, demonstrating proficiency in leveraging tools for in-depth network analysis.
- Detected and interpreted potential security threats, such as ARP poisoning and SYN flood attacks, showcasing problem-solving skills in identifying and addressing network vulnerabilities.
- Demonstrated innovation by simulating various types of attacks, including DoS attacks, and analyzing their impact on network traffic, highlighting proactive measures in network security management.