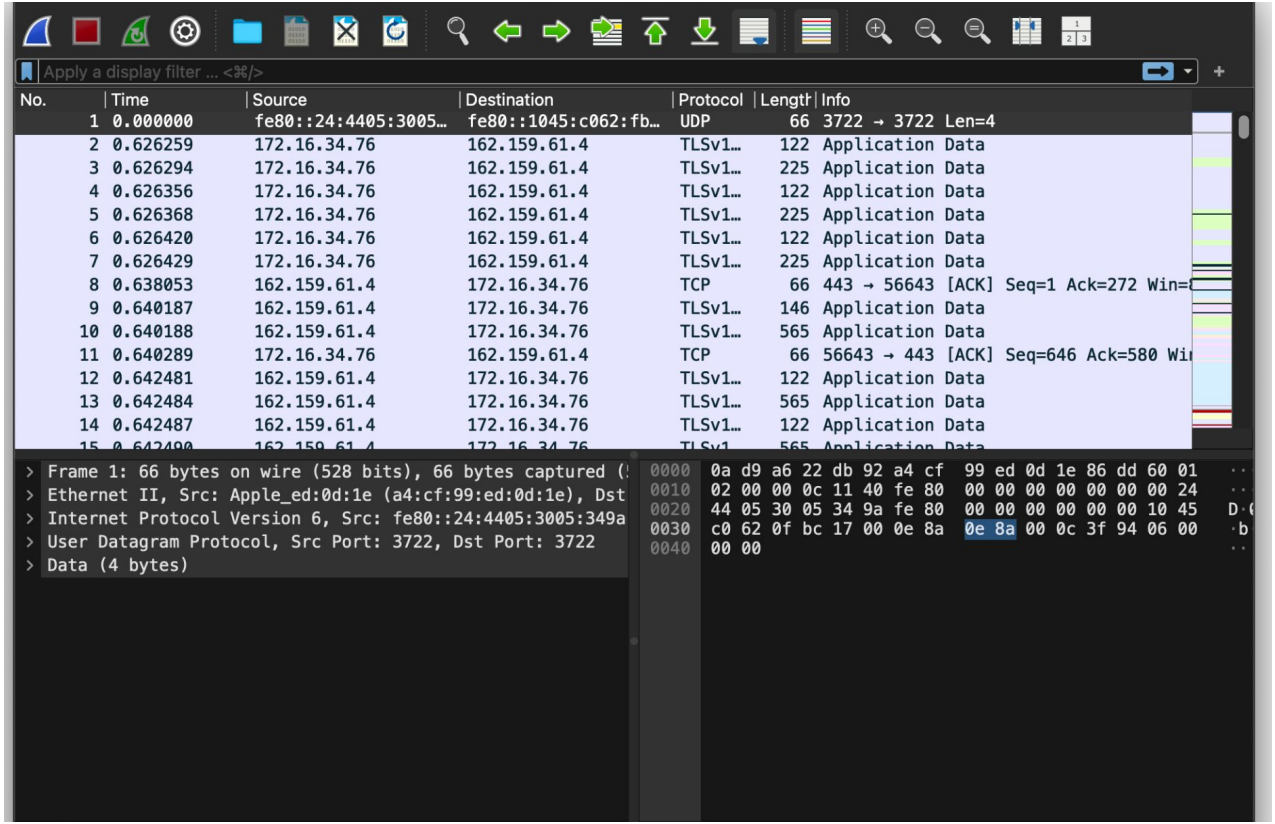
Real Time Network Threat Monitoring

Security Problem - Packet Sniffing

<<0:20-+>>



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	fe80::24:4405:3005…	fe80::1045:c062:fb…	UDP	66	3722 → 3722 Len=4
2	0.626259	172.16.34.76	162.159.61.4	TLSv1…	122	Application Data
3	0.626294	172.16.34.76	162.159.61.4	TLSv1…	225	Application Data
4	0.626356	172.16.34.76	162.159.61.4	TLSv1…	122	Application Data
5	0.626368	172.16.34.76	162.159.61.4	TLSv1…	225	Application Data
6	0.626420	172.16.34.76	162.159.61.4	TLSv1…	122	Application Data
7	0.626429	172.16.34.76	162.159.61.4	TLSv1…	225	Application Data
8	0.638053	162.159.61.4	172.16.34.76	TCP	66	443 → 56643 [ACK] Seq=1 Ack=272 Win=
9	0.640187	162.159.61.4	172.16.34.76	TLSv1…	146	Application Data
10	0.640188	162.159.61.4	172.16.34.76	TLSv1…	565	Application Data
11	0.640289	172.16.34.76	162.159.61.4	TCP	66	56643 → 443 [ACK] Seq=646 Ack=580 Wi
12	0.642481	162.159.61.4	172.16.34.76	TLSv1…	122	Application Data
13	0.642484	162.159.61.4	172.16.34.76	TLSv1…	565	Application Data
14	0.642487	162.159.61.4	172.16.34.76	TLSv1…	122	Application Data

```
> Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (
> Ethernet II, Src: Apple_ed:0d:1e (a4:cf:99:ed:0d:1e), Dst
> Internet Protocol Version 6, Src: fe80::24:4405:3005:349a
> User Datagram Protocol, Src Port: 3722, Dst Port: 3722
> Data (4 bytes)
```

```
0000  0a d9 a6 22 db 92 a4 cf  99 ed 0d 1e 86 dd 60 01   ··
0010  02 00 00 0c 11 40 fe 80  00 00 00 00 00 00 00 24   ··
0020  44 05 30 05 34 9a fe 80  00 00 00 00 00 00 10 45   D·
0030  c0 62 0f bc 17 00 0e 8a  0e 8a 00 0c 3f 94 06 00   ·b
0040  00 00                                              ··
```

PCAP file in WireShark

Security Problem - Common Attacks

<<0:20-+>>

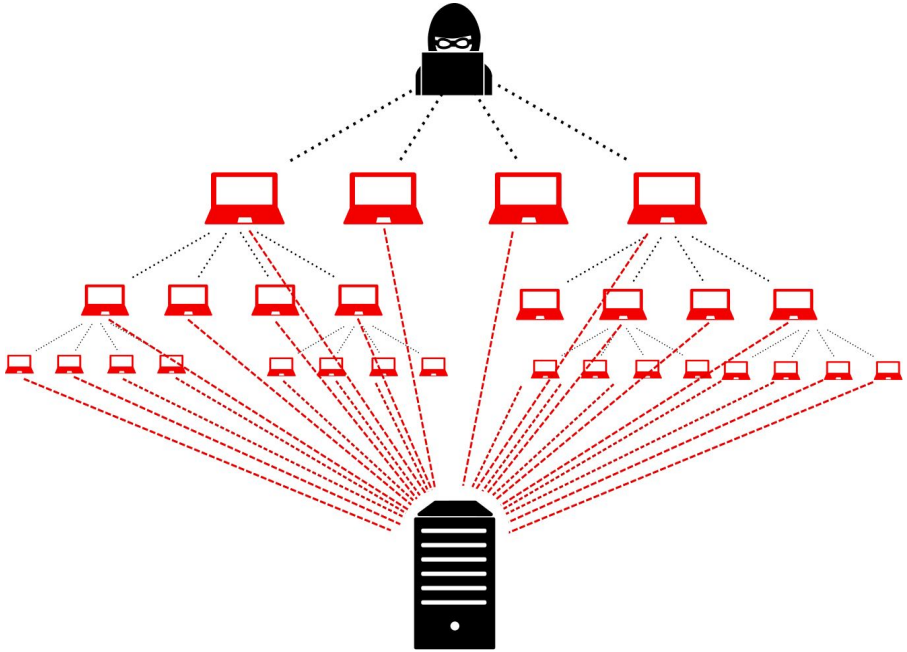
No.	Time	Source	Destination	Protocol	Flags
1	2015-02-15 18:05:01.818908	172.28.98.129	172.31.252.197	TCP	0x0010
2	2015-02-15 18:05:01.819081	172.28.98.129	172.31.252.197	TCP	0x0010
3	2015-02-15 18:05:01.819222	172.28.98.129	172.31.252.197	TCP	0x0010
4	2015-02-15 18:05:01.819361	172.28.98.129	172.31.252.197	TCP	0x0010
5	2015-02-15 18:05:01.819498	172.28.98.129	172.31.252.197	TCP	0x0010
6	2015-02-15 18:05:01.819689	172.28.98.129	172.31.252.197	TCP	0x0010
7	2015-02-15 18:05:01.820734	172.31.252.197	172.28.98.129	TCP	0x0004
8	2015-02-15 18:05:01.820758	172.31.252.197	172.28.98.129	TCP	0x0004
9	2015-02-15 18:05:01.820816	172.31.252.197	172.28.98.129	TCP	0x0004
10	2015-02-15 18:05:01.821038	172.31.252.197	172.28.98.129	TCP	0x0004
11	2015-02-15 18:05:01.821125	172.31.252.197	172.28.98.129	TCP	0x0004
12	2015-02-15 18:05:01.821258	172.31.252.197	172.28.98.129	TCP	0x0004
13	2015-02-15 18:05:01.822284	172.28.98.129	172.31.252.197	TCP	0x0010
14	2015-02-15 18:05:01.822318	172.28.98.129	172.31.252.197	TCP	0x0010
15	2015-02-15 18:05:01.822448	172.28.98.129	172.31.252.197	TCP	0x0010
16	2015-02-15 18:05:01.822648	172.28.98.129	172.31.252.197	TCP	0x0010
17	2015-02-15 18:05:01.822789	172.28.98.129	172.31.252.197	TCP	0x0010
18	2015-02-15 18:05:01.822925	172.28.98.129	172.31.252.197	TCP	0x0010
19	2015-02-15 18:05:01.823060	172.28.98.129	172.31.252.197	TCP	0x0010
20	2015-02-15 18:05:01.823196	172.28.98.129	172.31.252.197	TCP	0x0010
21	2015-02-15 18:05:01.823331	172.28.98.129	172.31.252.197	TCP	0x0010

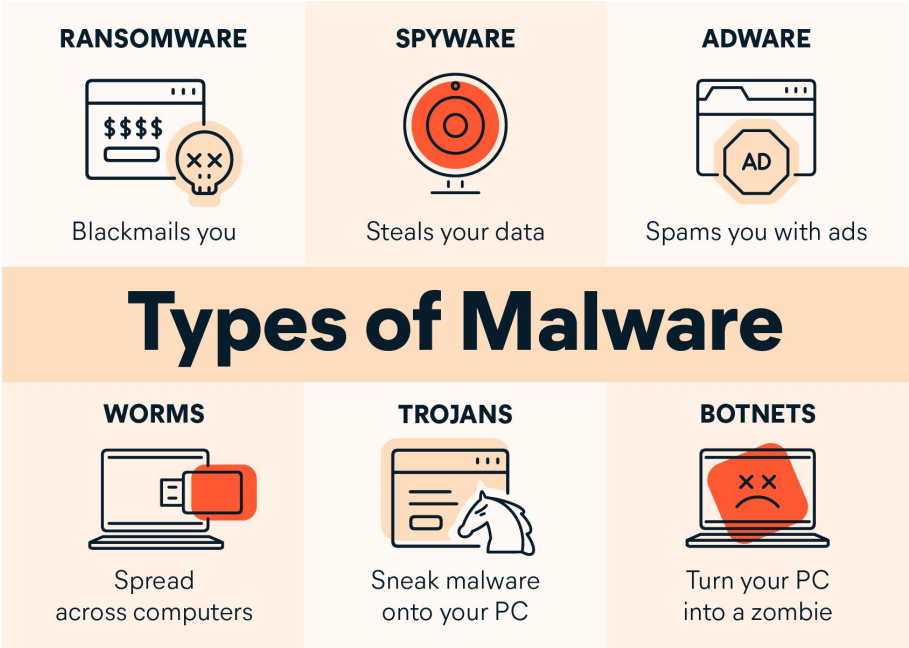
```
    Type: IP (0x0800)
▷ Internet Protocol Version 4, Src: 172.28.98.129 (172.28.98.129), Dst: 172.31.252.197 (172.31.252.197)
▽ Transmission Control Protocol, Src Port: sns-quote (1967), Dst Port: http (80), Seq: 322082625, Ack: 1130405221, Len:
    Source port: sns-quote (1967)
    Destination port: http (80)
    [Stream index: 0]
    Sequence number: 322082625
    Acknowledgment number: 1130405221
    Header length: 20 bytes
  ▷ Flags: 0x010 (ACK)
    Window size value: 512
    [Calculated window size: 512]
    [Window size scaling factor: -1 (unknown)]
  ▷ Checksum: 0x6319 [validation disabled]

0000  00 00 5e 00 01 67 04 01  3f 77 da 01 08 00 45 00   ..^..g.. ?w....E.
0010  00 28 93 fc 00 00 40 06  2f 51 ac 1c 62 81 ac 1f   .(....@. /Q..b...
0020  fc c5 07 af 00 50 13 32  97 41 43 60 9d 65 50 10   .....P.2 .AC`.eP.
0030  02 00 63 19 00 00                                  ..c...
```

DDOS Attack in Wireshark

Importance & Impact

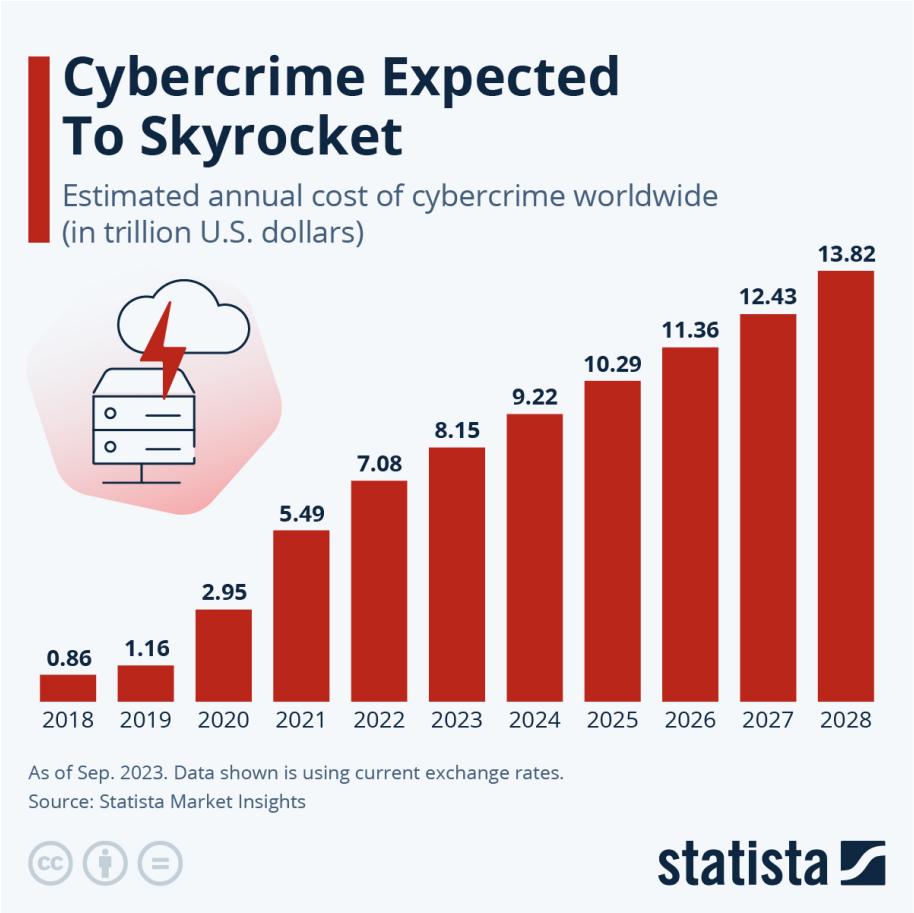
<<0:20-+>>





Importance & Impact

<<0:20-+>>



Importance & Impact

<<0:20-+>>

Solution - Wireshark Filtering

<<0:20-+>>

802.11 Wireshark Filters

Management Frames	wlan.fc.type == 0
Association Request	wlan.fc.type_subtype == 0
Association Response	wlan.fc.type_subtype == 1
Reassociation Request	wlan.fc.type_subtype == 2
Reassociation Response	wlan.fc.type_subtype == 3
Probe Request	wlan.fc.type_subtype == 4
Probe Response	wlan.fc.type_subtype == 5
Beacon	wlan.fc.type_subtype == 8
Disassociation	wlan.fc.type_subtype == 10
Authentication	wlan.fc.type_subtype == 11
Deauthentication	wlan.fc.type_subtype == 12
Action	wlan.fc.type_subtype == 13

Control Frames	wlan.fc.type == 1
Block ACK Request	wlan.fc.type_subtype == 24
Block ACK	wlan.fc.type_subtype == 25
PS-Poll	wlan.fc.type_subtype == 26
Ready To Send (RTS)	wlan.fc.type_subtype == 27
Clear to Send (CTS)	wlan.fc.type_subtype == 28
ACK	wlan.fc.type_subtype == 29

Data Frames	wlan.fc.type == 2
Data	wlan.fc.type_subtype == 32
Null	wlan.fc.type_subtype == 36
QoS Data	wlan.fc.type_subtype == 40
QoS Null	wlan.fc.type_subtype == 44

Display Filter Operators		
Equal	==	eq
Not Equal	!=	ne
And	&&	and
Or	||	or
Xor	^^	xor
Not	!	not
Contains	wlan.xxx contains "xx:xx"	

Addresses	
MAC address	wlan.addr == *MAC_address*
Transmitter Address (TA)	wlan.ta == *MAC_address*
Receiver Address (RA)	wlan.ra == *MAC_address*
Source Address (SA)	wlan.sa == *MAC_address*
Destination Address (DA)	wlan.da == *MAC_address*

Access Points and SSIDs	
BSSID	wlan.bssid == *AP_radio_MAC_address*
SSID	wlan_mgt.ssid == *SSID*

Radio Tap Header	
Specific Channel	radiotap.channel.freq == *frequency*
Specific Data Rate	radiotap.datarate == *rate_in_Mbps*
RSSI	radiotap.dbm_antsignal == *rate_in_dBm*

802.11k,v,r	
802.11v DMS request	wlan.fixed.action_code == 23
802.11v DMS response	wlan.fixed.action_code == 24
802.11k Neighbor request	wlan.rm.action_code == 4
802.11k Neighbor response	wlan.rm.action_code == 5
802.11r FT auth req	(wlan.fc.type_subtype==0) && (wlan.rsn.akms.type == 3)
802.11r FT auth res	(wlan.fc.type_subtype==1) && (wlan.tag.number == 55)
802.11r FT reassoc req	(wlan.fc.type_subtype==2) && (wlan.tag.number == 55)
802.11r FT reassoc res	(wlan.fc.type_subtype==3) && (wlan.tag.number == 55)

Retries	
Retry	wlan.fc.retry==1

Weak Signal and Probes	
Weak Signal	wlan_radio.signal_dbm < -dB
Weak Probe responses	wlan.fc.type_subtype == 5 && wlan_radio.signal_dbm < -dB
Weak Probe requests	wlan.fc.type_subtype == 4 && wlan_radio.signal_dbm < -dB

4-Way Handshake Filter	wlan.addr == MAC && eapol

Solution - Wireshark Filtering

<<0:20-+>>

- SYN Flag set
- No response (ACK)
- Same IP Address
- Large amount of requests

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.0.2	10.128.0.2	TCP	54	3341 → 80 [SYN] Seq=0 Win=512 Len=0
2	0.003987	10.128.0.2	10.0.0.2	TCP	58	80 → 3222 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
3	0.005514	10.128.0.2	10.0.0.2	TCP	58	80 → 3341 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
4	0.008429	10.0.0.2	10.128.0.2	TCP	54	3342 → 80 [SYN] Seq=0 Win=512 Len=0
5	0.010233	10.128.0.2	10.0.0.2	TCP	58	80 → 3220 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
6	0.014072	10.128.0.2	10.0.0.2	TCP	58	80 → 3342 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
7	0.016830	10.0.0.2	10.128.0.2	TCP	54	3343 → 80 [SYN] Seq=0 Win=512 Len=0
8	0.022220	10.128.0.2	10.0.0.2	TCP	58	80 → 3343 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
9	0.023496	10.128.0.2	10.0.0.2	TCP	58	80 → 3219 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
10	0.025243	10.0.0.2	10.128.0.2	TCP	54	3344 → 80 [SYN] Seq=0 Win=512 Len=0
11	0.026672	10.128.0.2	10.0.0.2	TCP	58	80 → 3218 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
12	0.028038	10.128.0.2	10.0.0.2	TCP	58	80 → 3221 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
13	0.030523	10.128.0.2	10.0.0.2	TCP	58	80 → 3344 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
14	0.033714	10.0.0.2	10.128.0.2	TCP	54	3345 → 80 [SYN] Seq=0 Win=512 Len=0
15	0.039322	10.128.0.2	10.0.0.2	TCP	58	80 → 3345 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
16	0.040725	10.128.0.2	10.0.0.2	TCP	58	80 → 3225 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
17	0.041334	10.128.0.2	10.0.0.2	TCP	58	80 → 3224 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
18	0.042165	10.0.0.2	10.128.0.2	TCP	54	3346 → 80 [SYN] Seq=0 Win=512 Len=0
19	0.047510	10.128.0.2	10.0.0.2	TCP	58	80 → 3346 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
20	0.050575	10.0.0.2	10.128.0.2	TCP	54	3347 → 80 [SYN] Seq=0 Win=512 Len=0
21	0.051715	10.128.0.2	10.0.0.2	TCP	58	80 → 3223 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
22	0.055986	10.128.0.2	10.0.0.2	TCP	58	80 → 3347 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
23	0.059015	10.0.0.2	10.128.0.2	TCP	54	3348 → 80 [SYN] Seq=0 Win=512 Len=0
24	0.064603	10.128.0.2	10.0.0.2	TCP	58	80 → 3348 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
25	0.066994	10.128.0.2	10.0.0.2	TCP	58	80 → 3228 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460

Solution - Wireshark Filtering

<<0:20-+>>

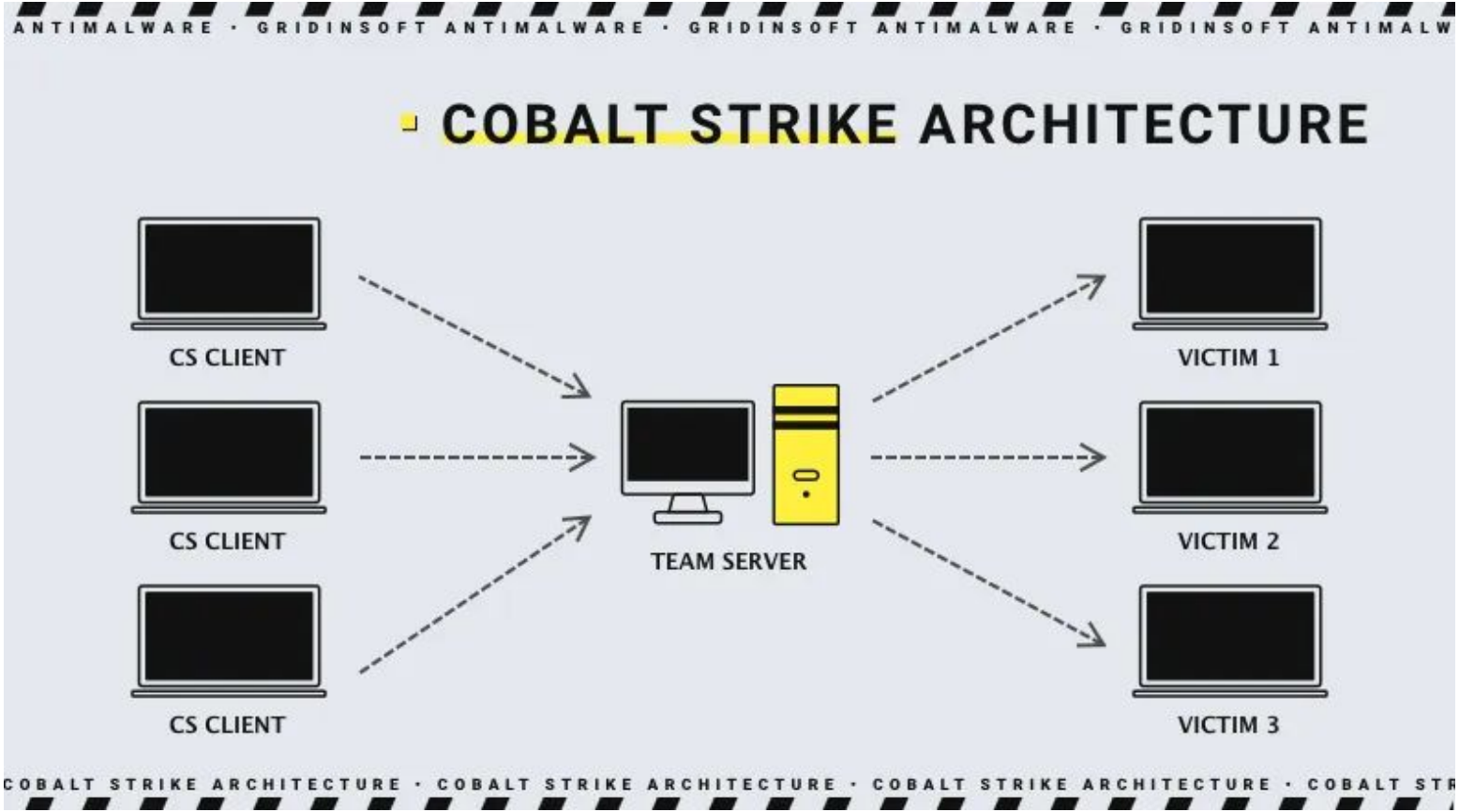
((tcp.flags.syn == 1 && tcp.flags.ack == 0) || (udp && frame.len > 1000) || icmp.type == 8 || icmp.type == 0) || frame.len > 1500

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.0.2	10.128.0.2	TCP	54	3341 → 80 [SYN] Seq=0 Win=512 Len=0
4	0.008429	10.0.0.2	10.128.0.2	TCP	54	3342 → 80 [SYN] Seq=0 Win=512 Len=0
7	0.016830	10.0.0.2	10.128.0.2	TCP	54	3343 → 80 [SYN] Seq=0 Win=512 Len=0

- SYN request without ACK reply
- High frame length
- ICMP request or ICMP response
- Frame length over 1500

Solution - Wireshark Filtering

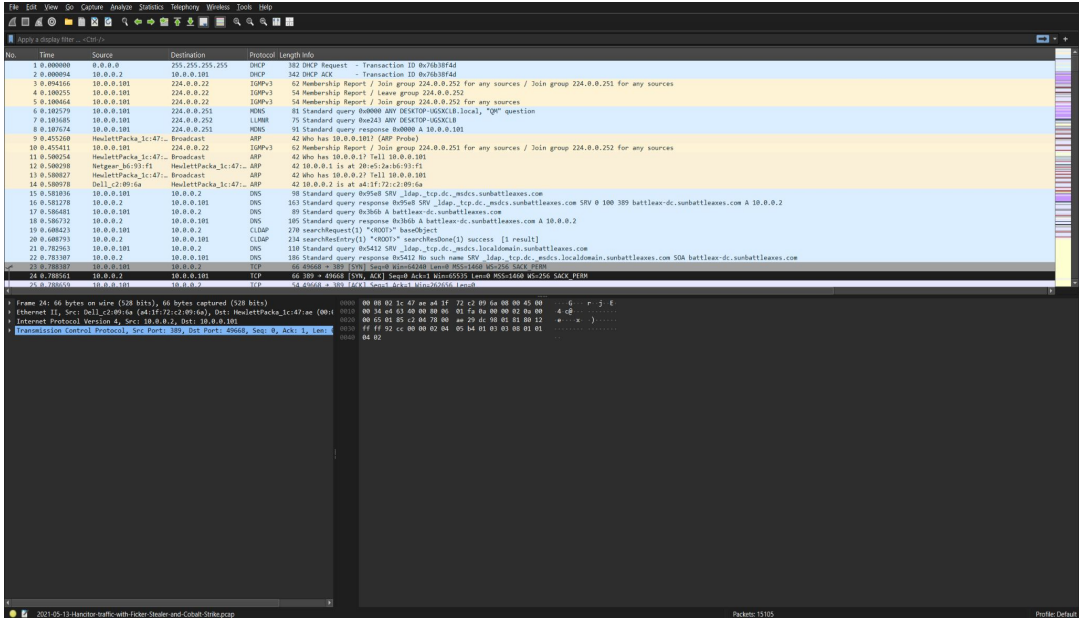
<<0:20-+>>



Solution - Wireshark Filtering

<<0:20-+>>

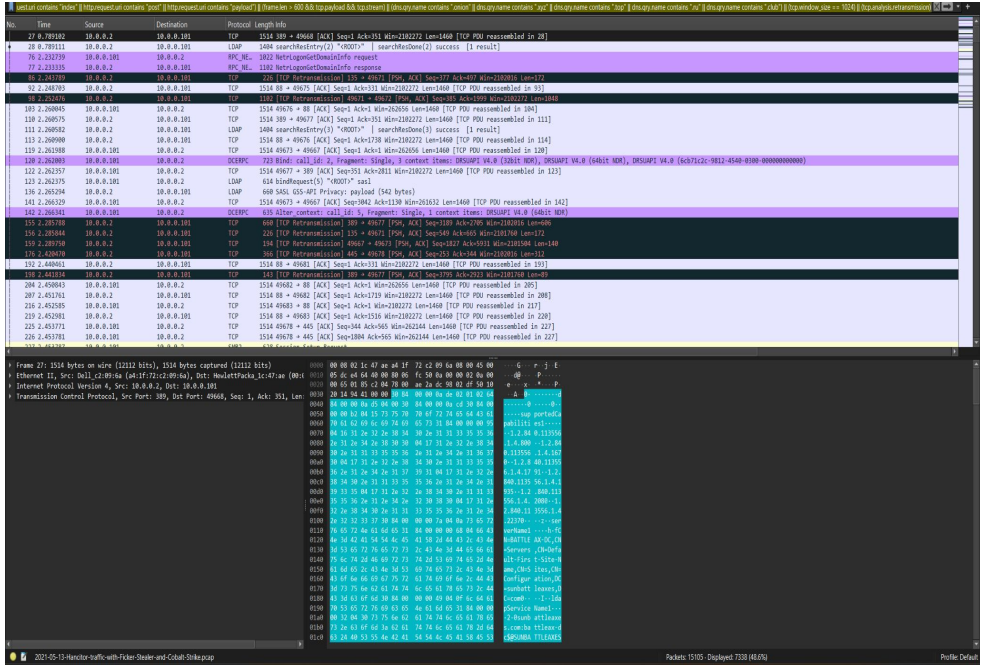
- 15105 Packets
- Large mix of packets from different IPs
- Large mix of protocols (SMB, KRB5, TCP, LDAP, ARP)



Solution - Wireshark Filtering

<<0:20-+>>

- **Filters applied**
  - Ports
  - Time delay
  - Common C2 names searched
  - DNS names
  - TCP window size
  - 7338 packets



Solution - Potential for Automation

<<0:20-+>>

Solution - Potential for Automation

<<0:20-+>>

Python Application - Packages & Libraries

- PyShark
  - Reads PCAP Files
- SYS
  - Takes Arguments

Solution - Potential for Automation

<<0:20-+>>

```python
if __name__ == '__main__':
    if len(argv) != 2:
        print(f'usage: {argv[0]} <pcap file name>')
        exit()

    pcap = argv[1]

    if not os.path.exists(pcap):
        print("File not found")
        exit()

    # Send the pcap file to the functions
    detect_ddos(pcap)
    detect_mal_commands(pcap)
    if not detect_ddos and not detect_mal_commands:
        print("No malicious traffic patterns detected!")
        exit()
```

Script Entry Point

Script Entry Point:

- Input Validation (arguments and file availability)
- Grabs PCAP file
- Sends file to function to detect attacks
- Returns results

Solution - Potential for Automation

<<0:20-+>>

```python
def detect_ddos(pcap):
    syn_flood_filter = 'tcp.flags.syn == 1 && tcp.flags.ack == 0'
    udp_flood_filter = 'udp && frame.len > 1000'
    imcp_flood_filter = 'icmp.type == 8 || icmp.type == 0'
    large_frame_filter = 'frame.len > 1500'

    def extract_ips(cap):
        ips = Counter()
        for packet in cap:
            ip = packet.ip.src
            if ip:
                ips[ip] += 1
        return ips

    syn_flood_cap = pyshark.FileCapture(pcap, display_filter=syn_flood_filter)
    syn_count = len([packet for packet in syn_flood_cap])
    if syn_count > SYN_FLOOD_THRESHOLD:
        print(f"Potential SYN Flood attack detected! (SYN count: {syn_count})")
        syn_ips = extract_ips(syn_flood_cap)
        print("Suspicious IPs involved:")
        for ip, count in syn_ips.most_common(5):
            print(f"  {ip} - {count} packets")
        attack = True

    syn_flood_cap.close()
```

detect_ddos method:

- Applies the DDOS attack filters to the PCAP file for each attack type
- Compares packet counts against a predefined threshold
- Extracts additional information
- Repeats this for the other 3 filters (UDP, IMCP, large frame)

Solution - Potential for Automation

<<0:20-+>>

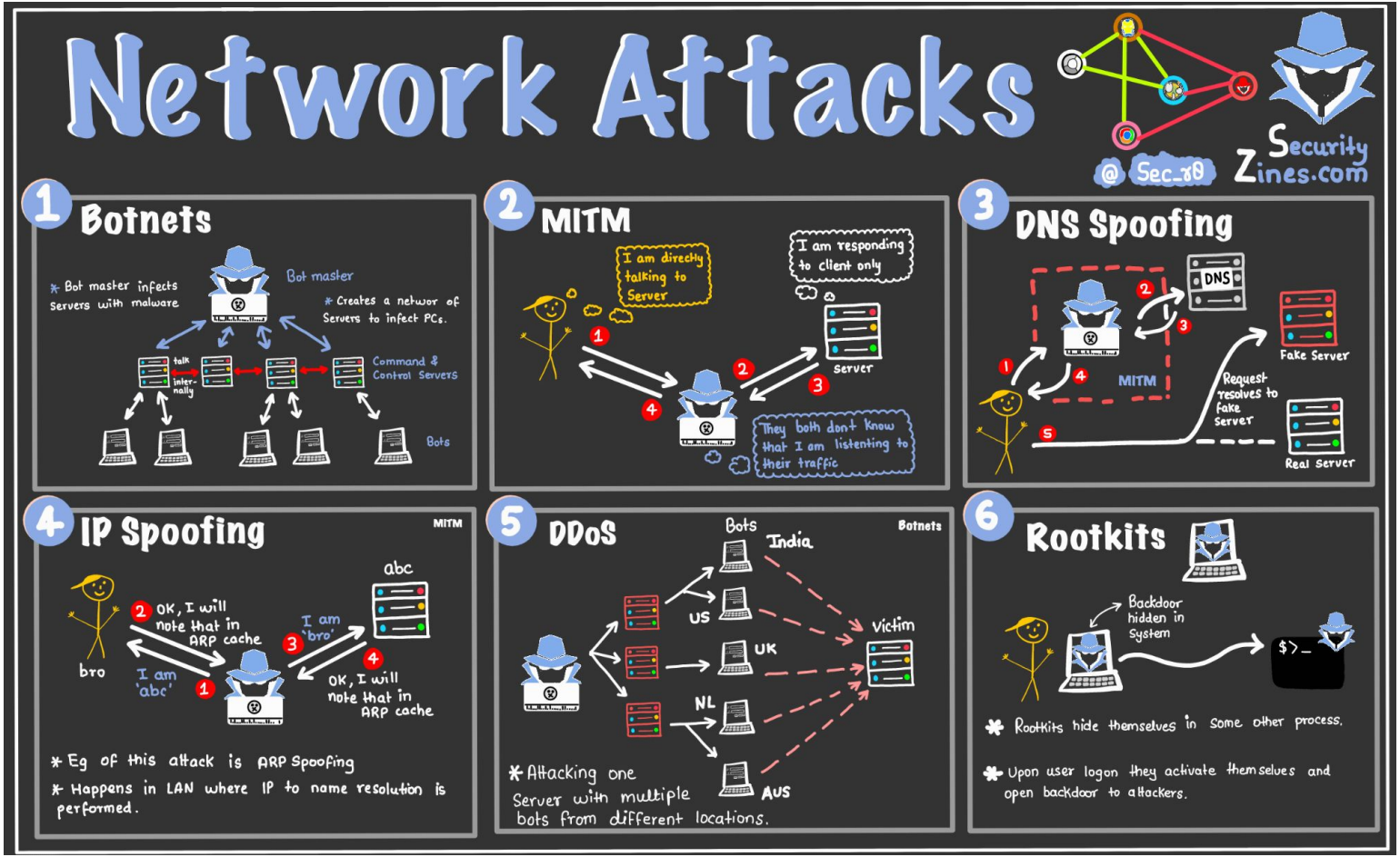
```
def detect_mal_commands(pcap):
    # Fill in the function to detect other attack
    tcp_filter = '((tcp.port == 80 || tcp.port == 443 || tcp.port == 2222 ||
tcp.port == 4444 || tcp.port == 8080 || tcp.port == 8443 || tcp.port > 1024)
&& (frame.time_delta > 45 && frame.time_delta < 75))||(tcp.window_size ==
1024) || tcp.analysis.retransmission || (frame.len > 600 && tcp.payload &&
tcp.stream)'
    cobalt_filter = '(ssl.handshake.extensions_server_name contains "cobalt"
|| ssl.handshake.type == 1 || ssl.record.content_type == 23) ||
(http.user_agent contains "Cobalt" || http.user_agent contains "Mozilla" ||
http.request.uri contains "beacon" || http.request.uri contains "index" ||
http.request.uri contains "post" || http.request.uri contains "payload")'
    malicious_dns_filter = 'dns.qry.name contains ".onion" || dns.qry.name
contains ".xyz" || dns.qry.name contains ".top" || dns.qry.name contains
".ru" || dns.qry.name contains ".club"'

    # Checks for calls over commonly closed TCP ports, or large frame size
    tcp_cap = pyshark.FileCapture(pcap, display_filter=tcp_filter)
    tcp_frame_count = len([packet for packet in tcp_cap])
    if tcp_frame_count > 1:
        print("Potentially malicious activity over TCP")
        tcp_cap.close()
        return True
…
```

detect_mal_commands method:

- Applies the malware attack filters to the PCAP file, broken down by each type of attack
- Compares the packet counts of each filter against a predefined threshold

Solution - Potential for Automation

<<0:20-+>>



Conclusion - Key Takeaways

<<0:20-+>>

No.	Time	Source	Destination	Protocol	Flags
1	2015-02-15 18:05:01.818908	172.28.98.129	172.31.252.197	TCP	0x0010
2	2015-02-15 18:05:01.819081	172.28.98.129	172.31.252.197	TCP	0x0010
3	2015-02-15 18:05:01.819222	172.28.98.129	172.31.252.197	TCP	0x0010
4	2015-02-15 18:05:01.819361	172.28.98.129	172.31.252.197	TCP	0x0010
5	2015-02-15 18:05:01.819498	172.28.98.129	172.31.252.197	TCP	0x0010
6	2015-02-15 18:05:01.819689	172.28.98.129	172.31.252.197	TCP	0x0010
7	2015-02-15 18:05:01.820734	172.31.252.197	172.28.98.129	TCP	0x0004
8	2015-02-15 18:05:01.820758	172.31.252.197	172.28.98.129	TCP	0x0004
9	2015-02-15 18:05:01.820816	172.31.252.197	172.28.98.129	TCP	0x0004
10	2015-02-15 18:05:01.821038	172.31.252.197	172.28.98.129	TCP	0x0004
11	2015-02-15 18:05:01.821125	172.31.252.197	172.28.98.129	TCP	0x0004
12	2015-02-15 18:05:01.821258	172.31.252.197	172.28.98.129	TCP	0x0004
13	2015-02-15 18:05:01.822284	172.28.98.129	172.31.252.197	TCP	0x0010
14	2015-02-15 18:05:01.822318	172.28.98.129	172.31.252.197	TCP	0x0010
15	2015-02-15 18:05:01.822448	172.28.98.129	172.31.252.197	TCP	0x0010
16	2015-02-15 18:05:01.822648	172.28.98.129	172.31.252.197	TCP	0x0010
17	2015-02-15 18:05:01.822789	172.28.98.129	172.31.252.197	TCP	0x0010
18	2015-02-15 18:05:01.822925	172.28.98.129	172.31.252.197	TCP	0x0010
19	2015-02-15 18:05:01.823060	172.28.98.129	172.31.252.197	TCP	0x0010
20	2015-02-15 18:05:01.823196	172.28.98.129	172.31.252.197	TCP	0x0010
21	2015-02-15 18:05:01.823331	172.28.98.129	172.31.252.197	TCP	0x0010

Conclusion - Impact of Our Work

<<0:20-+>>