What did you do?

1. Visited wireshark.org
2. Clicked on “Get Started”
3. Clicked Windows Intel Installer option
4. Downloaded Wireshark-win64-4.0.7
5. Ran the setup.exe to install and setfup the software
6. Agreed to license agreement
7. Selected components below to install excluding Randpkt; Androiddump; Sshdump, Ciscodump, and Wifidump; UDPdump; Randpktdump; Etwdump
   * Wiresharkf
   * TShark
   * Plugins & Extensions
     + Dissector Plugins
     + Tree Statistics Plugin
     + Mate – Meta Analysis and Tracing Engine
     + TRANSUM – network application performance analysis
     + File type plugins – capture file support
     + Codec plugins
     + Configuration Profiles
     + SNMP MIBs
   * Tools
     + Editcap
     + Text2Pcap
     + Mergecap
     + Reordercap
     + DFTest
     + Capinfos
     + Captype
     + Rawshark
     + MMDBResolve
   * Documentation
8. Created shortcuts in Start Menu and Desktop Icon
9. Marked Associate trace file extensions with Wireshark
10. Chose file location C:\Program Files\Wireshark
11. Already had Npcap 1.75 installed
12. Setup completed successfully
13. Opened Wireshark GUI
14. Clicked on Capture button
15. Saw the only real time captures happening were on Wi-Fi and Adapter for loopback traffic capture
16. Clicked on Wi-Fi and start
17. Immediately began seeing packet information captured
18. Opened Zenmap and scanned 192.168.1.0/24
19. Saw Wireshark capturing flood of packets sent by NMAP
20. Stopped Wireshark capturing once NMAP complete
21. Printed to pdf “Wireshark Capture-Nmap” with as displayed option clicked
22. Realized pdf is 1480 pages long
23. Printed to pdf “Wireshark Capture-Nmap\_Collapsed” with all collapsed option clicked
24. Realized both printed pdf’s are identical
25. Printed to pdf “Nmap Scan GUI-Home Network
26. Used Snipping Tool to capture Home Network Topology
27. Printed to pdf “Nmap Topology -Home Network

Wireshark software captures the bit by bit traffic of packets within the given boundaries input by the user. With a bit by bit analysis, Wireshark is able to tell the type of packet by headers. I immediately realized capturing information on every packet sent and received is overwhelming, hence the filter option at the top of the capture output. The filter input isn’t intuitive, requires the right commands to use it, and other tools to understand it. I found Display Filters under the Analyze tab that provide the following foundation:

|  |  |
| --- | --- |
| Filter Name | Filter Expression |
| Ethernet address 00:00:5e:00:53:00 | eth.addr == 00:00:5e:00:53:00 |
| Ethernet type 0x0806 (ARP) | eth.type == 0x0806 |
| Ethernet broadcast | eth.addr == ff:ff:ff:ff:ff:ff |
| No ARP | not arp |
| IPv4 only | ip |
| IPv4 address 192.0.2.1 | ip.addr == 192.0.2.1 |
| IPv4 address isn’t 192.0.2.1 | ip.addr != 192.0.2.1 |
| IPv6 only | ipv6 |
| IPv6 address 2001:db8::1 | ipv6.addr == 2001:db8::1 |
| TCP only | tcp |
| UDP only | upd |
| Non-DNS port | !(upd.port == 53 II tcp.port == 53) |
| TCP or UDP port is 80 (HTTP) | tcp.port == 80 II udp.port == 80 |
| HTTP | http |
| No ARP and no DNS | not arp and not dns |
| Non-HTTP and non-SMTP to/from 192.0.2.1 | ip.addr == 192.0.2.1 and tcp.port not in {80, 25} |

I found the following filter options to help interpret the packet traffic from Profitap.com.¹

|  |  |
| --- | --- |
| Filter Effect | Filter |
| IP address as source or destination IP | ip.addr == X.X.X.X |
| IP address as source IP | ip.src == X.X.X.X |
| IP address as destination IP | ip.dst == X.X.X.X |
| Conversation between two specific IP addresses | ip.addr == X.X.X.X && ip.addr == X.X.X.X  ip.src == X.X.X.X && ip.dst == X.X.X.X |
| All HTTP ports / All DNS ports | http / dns |
| TCP packet with specific source/destination port | tcp.port==XXX |
| All TCP resets | tcp.flags.reset==1 |
| TCP packets that contain a certain term | tcp contains XXX |
| Follows a TCP stream | tcp.stream eq X |
| TCP Sequence number | tcp.seq == X |
| Detects push events | tcp.flags.push == 1 |
| HTTP get and post requests | http.request |
| Filter out certain protocols | !(arp or icmp or dns) |
| Certain Hex values at any offset | udp contains XX:XX:XX |
| Indicates which DNS requests weren’t resolved | dns.flags.rcode != 0 |

I found the following TCP Flags List from howtouselinux.com.²

|  |  |  |
| --- | --- | --- |
| Flag | Name | Meaning |
| SYN | Synchronize Sequence Number | Packets that are used to initiate a connection. |
| ACK | Acknowledgment | Packets that are used to confirm that the data packets have been received, also used to confirm the initiation request and tear down requests. |
| RST | Reset the Connection | Signify the connection is down or maybe the service is not accepting requests. |
| FIN | Finish Sending Data | Indicate that the connection is being torn down. Both the sender and receiver send the FIN packets to gracefully terminate the connection. |
| PSH | Push Function | Indicate that the incoming data should be passed on directly to the application instead of getting buffered. |
| URG | Urgent Pointer | Indicate that the data that the packet is carrying should be processed immediately by the TCP stack. It can be used to provide out-of-band data transfer, such as signaling that a message is urgent and should be delivered before other data. |

I realized I must find several independent methods of filtering the data to reduce the tsunami of information. Wireshark output can be as macro or micro as you want. The Statistics tab provided the big picture view to shape the data in my mind. Protocol Hierarchy, Conversations, End Points, Packet Lengths, and I/O Graphs gave me an idea of the entire packet pool.

I input “ip.src == 192.168.1.0/24 && ip.dst == 192.168.1.0/24” into the filter field to determine which packets were confined to my home network for the NMAP I ran. This revealed six of the 254 IP addresses available for dynamic assignment on my home network were actively connected. Of note, DHCP gave the Roku device an IP different than last week’s NMAP research report.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 192.168.1.1 | 192.168.1.15 | 192.168.1.129 | 192.168.1.187 | 192.168.1.211 | 192.168.1.227 |
| Gateway Router | PC | Cell Phone | Printer | Roku | Tablet |

In analyzing the data, I identified being able to filter by port/protocol narrowed parameters to see what device was utilizing which port/protocol. Given that Wireshark is a free software, there is no hand holding for a new user. I relied on multiple google searches, reading through various websites covering Wireshark, and several youtube videos before I began to grasp what the software does. I reiterate the begin to grasp because it seems like an iceberg above the water being visible scenario.

What are the results?

The following ports and protocols were filtered to identify the scope of data traffic. TCP is the major protocol used during NMAP (ICMP = 0.4%. TCP = 88.2%, UDP = 6.3%).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | Port | Name | Protocol | Bytes Range | Packets |
| 53 | DNS | Domain Name System | UDP | 70-397 | 171 |
| Dynamic? | GIOP | General Inter-ORB Protocol | TCP | 102 | 2 |
| 7000 | Gryphon | DG Gryphon Protocol | TCP | 66-571 | 2 |
| 80/8000 | HTTP | Hypertext Transfer Protocol | TCP | 60-1064 | 279 |
| 631 | IPP | Internet Printing Protocol | TCP | 212-677 | 2 |
| 5355 | LLMNR | Link-local Multicast Name Resolution | UDP | 75-126 | 88 |
| 515 | LPD | Line Printer Daemon Protocol | TCP | 60-72 | 6 |
| Dynamic? | Portmap | Portmap | TCP | 98 | 6 |
| 9080 | RPC | Remote Procedure Call | TCP | 102-122 | 4 |
| Dynamic? | SIP | Session Initiation Protocol | TCP | 277 | 4 |
| 161 | SNMP | Simple Network Management Protocol | UDP | 88-578 | 42 |
| 1900 | SSDP | Simple Service Discovery Protocol | UDP | 217-322 | 16 |
| 443 | SSLv2  SSLv3  TLSv1  TLSv1.2  TLSv1.3  QUIC | Transport Layer Security | TCP | 60-5094 | 1342 |
| ***Protocol 1*** |  | ***Internet Control Message Protocol*** | ***ICMP*** | ***70-370*** | ***60*** |
| ***Protocol 6*** |  | ***Transport Control Protocol*** | ***TCP*** | ***54-10274*** | ***13187*** |
| ***Protocol 17*** |  | ***User Datagram Protocol*** | ***UDP*** | ***43-1514*** | ***939*** |

These ports are the players I expected to see through Wireshark during an NMAP. Pretty common ones here, hypertext transfer protocols, printing protocols, email, and DNS. The three-way encryption sessions were captured outside of the NMAP scan, but still present on regular network traffic. Here are the details of the packet lengths:

A screenshot of a graph

Description automatically generated

The majority of the packet lengths were in the anticipated range below 639 bits. The packets larger than these could be reviewed to see if there is a data flow bottleneck or issue with a port/protocol. Below are the statistics for conversations.

A screenshot of a computer

Description automatically generated

My Gateway Router IP is 192.168.1.1, so the other five devices on my home network all have the same first three octets with the last changing between devices. I ran nslookup on the remaining IP addresses to see what would populate. I suspect the IP addresses that showed “Non-existent domain” are switches and routers, or devices with firewalls. “Akamai Technologies, Inc. is an American content delivery network, cybersecurity, and cloud service company, providing web and Internet security services. The company operates a network of servers worldwide and rents the capacity of the servers to customers wanting to increase the efficiency of their websites…”³ Akamai Technologies and Amazon Web Services could be doing any number of things with cloud computing.

|  |  |
| --- | --- |
| 3.222.102.97 | Ec2-3-222-102-97.compute-1.amazonaws.com |
| 13.107.5.80 | Non-existent domain |
| 13.107.21.200 | Non-existent domain |
| 13.249.74.122 | Server-13-249-74-122.dfw53.r.cloudfront.net |
| 20.236.93.119 | Non-existent domain |
| 23.35.69.41 | A23-25-69-41.deploy.static.akamaitechnologies.com |
| 23.192.220.18 | A23-192-220-18.deploy.static.akamaitechnologies.com |
| 34.117.228.201 | 201.228.117.34.bc.googleusercontent.com |
| 40.126.29.6 | Non-existent domain |
| 44.206.215.181 | Ec2-44-206-215-181.compute-1.amazonaws.com |
| 51.11.192.50 | Non-existent domain |
| 52.44.199.97 | Ec2-52-44-199-97.compute-1.amazonaws.com |
| 52.96.191.242 | Non-existent domain |
| 54.82.230.177 | Ec2-54-82-230-177.compute-1.amazonaws.com |
| 54.236.236.70 | Ec2-54-236-236-70.compute-1.amazonaws.com |
| 104.18.4.44 | Non-existent domain |
| 104.208.16.90 | Non-existent domain |
| 108.138.168.190 | Server-108-138-168-190.dfw56.r.cloudfront.net |
| 142.250.190.2 | Ord37s32-in-f2.1e100.net |
| 146.75.106.137 | Non-existent domain |
| 162.247.241.14 | Non-existent domain |
| 184.30.42.185 | A184-30-42-185.deploy.static.akamaitechnologies.com |
| 212.102.40.113 | 212-102-40-113.bunnyinfra.net |

What did you learn?

Wireshark is an excellent and versatile tool for troubleshooting packet traffic on networks, and overwhelming at first. If Wireshark isn’t given filter parameters, the data is too robust and detailed. “You can’t see the forest because you’re looking at the trees.” I have found an effective practice when briefing military personnel who do not work in the IT or communications realm. I need to brief by capability and develop a knack for analogies. The more doctrinal description is TPME – Task, Purpose, Method, and Effect.

To describe Wireshark, I’d compare it to using GPS on a trip from San Diego to New York City and only taking surface streets or county roads. You won’t be able to remember or even see all the information on the turns, street names, distances, town locations, gas stations, hotels, and weather conditions. However, using a filter to narrow your parameters or even pinching the screen to zoom in on a section of road will make the information pertinent and digestible.

I was surprised at how many customization options there are within Wireshark. The color coding of certain types of packets, the ability to save filters, display preferences, packet diagrams to see the bit-by-bit construction of packets, additional columns of sortable data, and renaming ability of devices are just a handful of personal touches a user can put onto their GUI.

Wireshark’s website describes the software as a Network Protocol Analyzer, but a more common name from our readings is a packet sniffer. I don’t think this is a deceitful euphemism; I believe it’s an effect of the perspective of the user. White hats, ethical hackers, or network administrators can use Wireshark to root out problems. Black hats, hackers, or disgruntled employee insider threats can use Wireshark to ferret out attack vectors such as identifying the means to be the man in the middle for spoofing attacks. It would be an interesting thought experiment to see how politicians would react to this software when given a snapshot of capability. An argument can be made that other “tools” are regulated based on their potential for harm. Would we need a security clearance to use Wireshark? Would only the government and IPS be able to use the tool? I digress, but I could easily see Wireshark being debated on the House of Representatives floor.

I had two key takeaways from this exposure to Wireshark: 1) understanding of Wireshark grows over time and even Masters will still be learning tips and tricks, 2) finding ways to effectively limit the scope of data is essential for analyzing. The first layer of limitation I discovered was placing filters prior to beginning packet capture. In and of itself, this reduces the amount of data in my feedback. Add multiple filters! Use parenthesis (XXX) and (YYY) to increase filters. With the GUI capturing packets, Wireshark can be used actively and reactively. Meaning a network administrator can be monitoring packets on their network in real time (although I’d have several robust filters before attempting this) as well as analyzing the packet data after the fact. Wireshark allows the human brain to visualize network data streams.

Like NMAP, Wireshark is not an IDS/IPS/IDPS. I can see Wireshark’s utility in troubleshooting network performance issues. You can easily hone into the specifics of delays, dropped packets, etc. To process that data does require an understanding of networks and data flow, or else Wireshark output looks like Greek.

A revelation that was outside my expectations was my PC and Roku had 1208 packets transported within the sniffing window. I’m not familiar with the amount of packets required for streaming video and audio, but this seemed high for a couple minutes of sniffing.

Wireshark is a convergence of software that is absolutely required for network monitoring and troubleshooting. It is a tool and the direction of intent is based on the holder. In providing a practical and necessary capability for network management, Wireshark simultaneously creates the possibility for cyber attack assessment.

1. 14 Powerful Wireshark Filters our Engineers Use

https://insights.profitap.com/14-powerful-wireshark-filters-to-use#:~:text=14%20Powerful%20Wireshark%20Filters%20Our%20Engineers%20Use%201,which%20dns%20requests%20couldn%27t%20be%20correctly%20resolved.%20

1. Understanding TCP Flags SYN ACK RST FIN URG PSH

https://www.howtouselinux.com/post/tcp-flags

1. Wikipedia – Akamai Technologies

https://en.wikipedia.org/wiki/Akamai\_Technologies