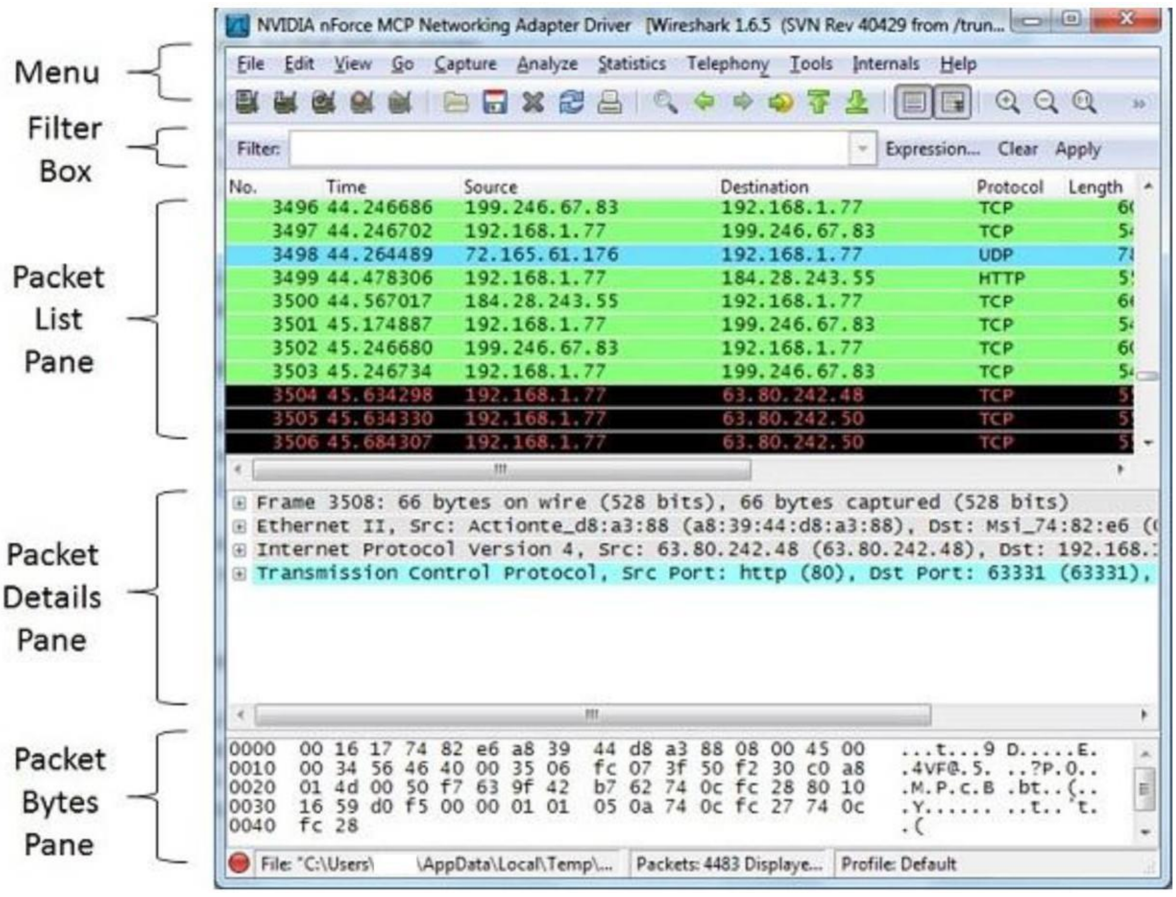
Wireshark is a widely used, open-source packet analyzer. It facilitates the deep inspection of hundreds of protocols. It also allows the user to capture network traffic into packet capture files.

Certain occurrences, regardless of protocol, should be documented. These include:

- Authorization errors
- User credentials passed in the clear
- Authentication errors
- Log on errors
- Missing or “Not found” errors
- Applications using non-standard ports
- Traffic to/from suspicious hosts
- Network reconnaissance processes
- Questionable traffic redirections
- Maliciously malformed frames
- Traffic that matches known keyword attack signatures

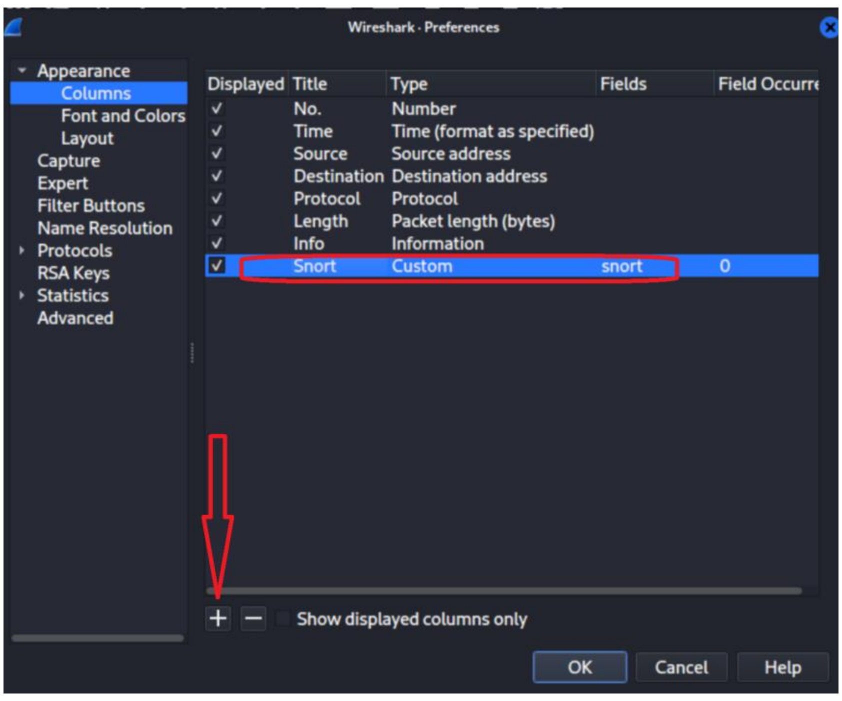
Once captured, packets are then decoded by the packet analyzer and data is displayed in as much detail as possible. Wireshark uses three different panes to display data about each packet. Clicking on a packet will populate all three panes with data for that specific packet.



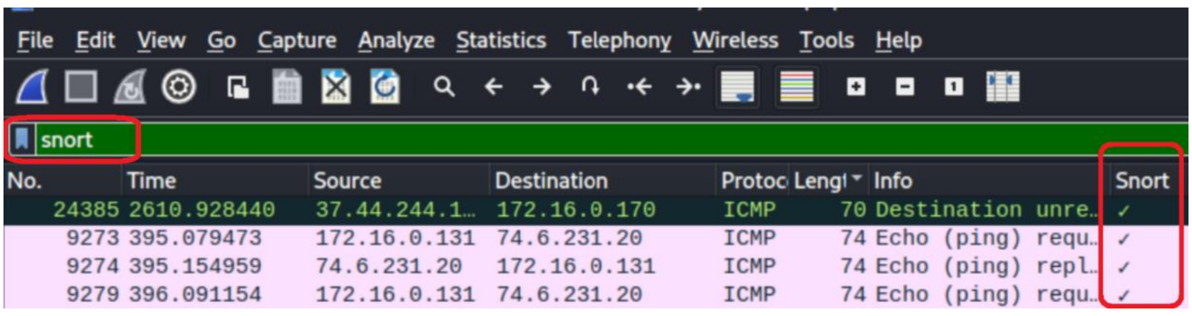
The default columns in the **Packet List Pane** are:

- **No.**: number of packets in the capture.
- **Time**: time stamp of the packet in seconds since start of capture.
- **Source**: address where the packet is coming from.
- **Destination**: address where the packet is going to.
- **Protocol**: protocol name.
- **Length**: length of each packet.
- **Info**: additional information about the packet.

Additional columns can be added by going to Edit->Preferences->Appearance->Columns and clicking the + sign at the bottom. You can also access this pane by right-clicking any existing column. To add a column for Snort rule matches, for example, you input a Title and select ‘snort’ in the Fields column.



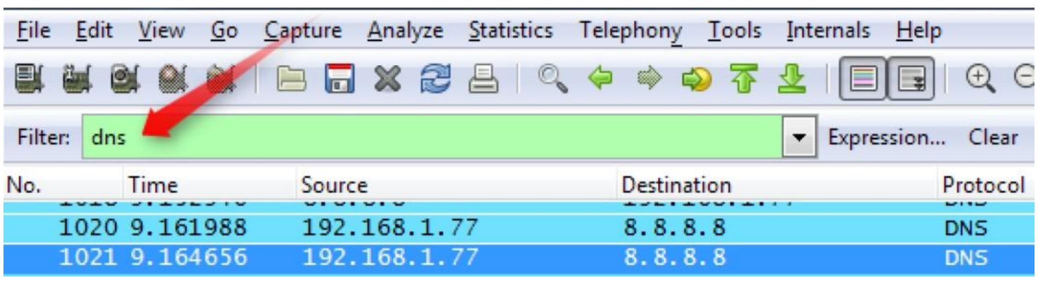
Snort rule matches are indicated by arrows below the Snort column.



The **Packet Details Pane** provides in-depth information about the selected packet. This information is displayed in a tree format, which can be expanded and collapsed. This is where analysts should be conducting their investigation.

The **Packet Bytes Pane** is a hex dump of the packet data. If analysts click on a section in the packet, Wireshark will highlight the corresponding hex section in the packet bytes pane.

Wireshark's **Display Filter** allow analysts to focus on specific packets. Analysts can either filter on traffic they would like to see (inclusion filtering) or hide undesirable traffic (exclusion filtering).



By right clicking and filtering on a particular field in the data of a packet, Wireshark will auto-create a filter based on that data, including the syntax of the filter type. The analyst can then use this syntax to search for whatever data they desire in the given field. This technique can be used to filter on many different fields in many different protocols. It should be used any time a particular field of data is of interest. It is particularly useful when encountering an unfamiliar protocol or proprietary protocols such as most industrial control system protocols.

Right-click -> Apply as filter. Then select one of the following:

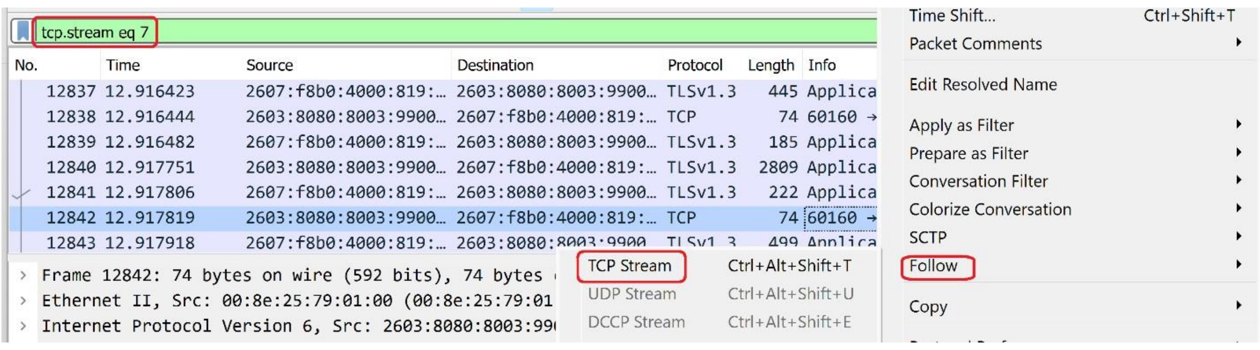
Selected: creates a filter based on the selection
Not Selected: creates an exclusion filter based on the selection
...and Selected: must match the existing filter *and* the selection
...or Selected: must match the existing filter *or* the selection
...and not Selected: must match the existing filter *and not* the selection
...or not Selected: must match the existing filter *or not* the selection

Wireshark has the capability to filter traffic by device addresses—MAC addresses, IPv4 addresses, and IPv6 addresses. Additionally, the filters can be configured to only display traffic found on one side of the conversation—either the source or destination.

Filters can combine multiple different individual filters using Boolean logic to create more focused searches. For example, the analyst may desire to see all traffic going from 192.168.1.1 and 192.168.1.150, and could utilize the "**&&**" syntax between an IP source and an IP destination to create this filter: *ip.src == 192.168.1.1* **&&** *ip.dst == 192.168.1.150*

The analyst can also use logical "OR" statements to display any traffic that matches one or more of the stated conditions. For example, the analyst may wish to see any traffic from a MAC address of 00:00:00:12:34:56 *or* any traffic from a MAC address of 00:00:00:98:76:54. Using the "**||**" syntax for an OR statement, the following filter could be written: *eth.src == 00:00:00:12:34:56* **||** *eth.src == 00:00:00:98:76:54*

Wireshark can identify all TCP segments in a stream, reassemble them using a specific algorithm, and present the results as text. This capability makes it easy to identify the purpose of a conversation and determine whether it is benign, suspicious, or malicious. To tell Wireshark to reassemble a TCP stream, highlight one of the packets in a stream, right-click, and choose Follow TCP Stream.

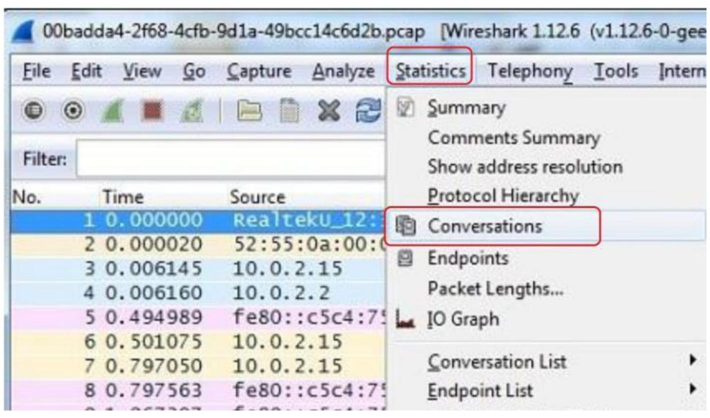


There are several options available after you follow a stream. Click Find to search for a text string. Click Save As to save the conversation as a separate file. The Save As feature is great if you want to export a file that was transported across a conversation. Select Filter Out This Stream to create and apply an exclude display filter for this stream. The ability to filter out conversations after examining them is crucial in narrowing down suspicious traffic on a network.

When investigating connections, analysts should pay particular attention to:

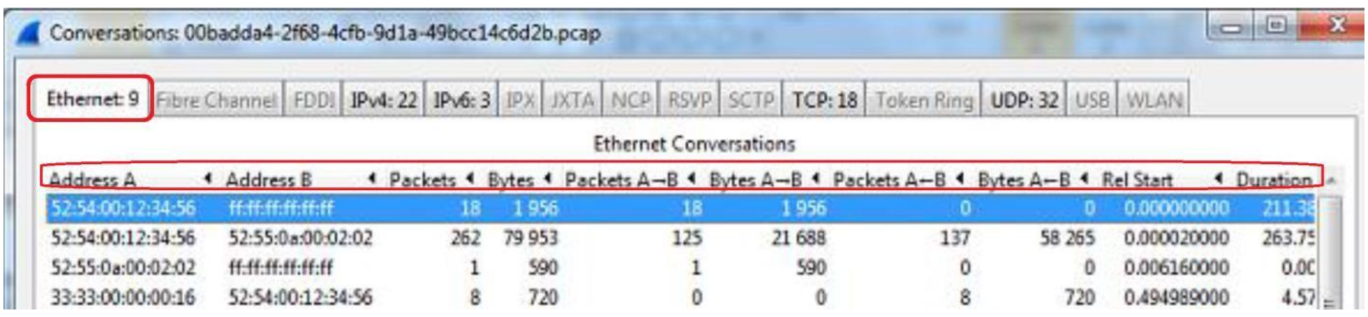
- Broadcasts
- External connections
- Network segmentation
- Devices that have a large variety of connections

The Conversations window in Wireshark allows the user to see all connections between different addresses. Statistics -> Conversations

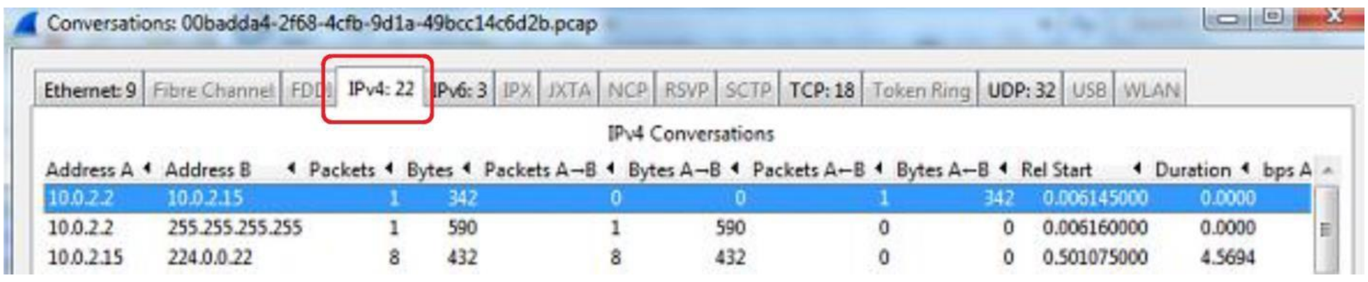


The Conversations window will open in the “Ethernet” tab. Every tab has the same column headings:

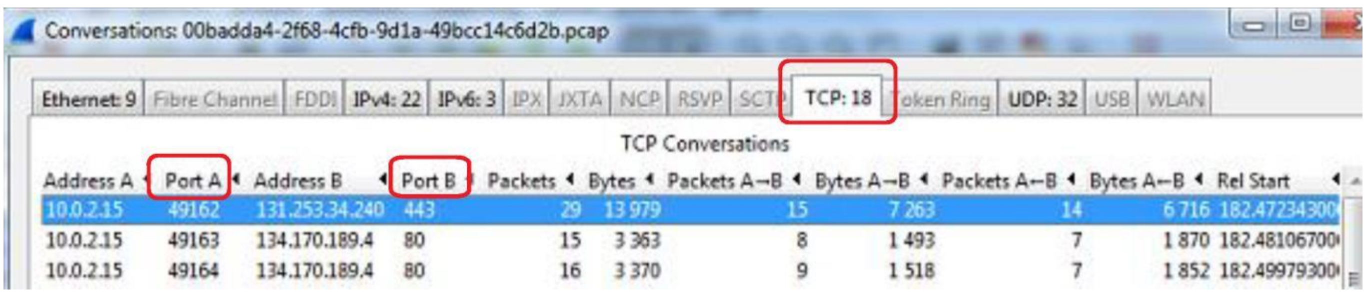
- **Address A:** one side of the conversation
- **Address B:** other side of the conversation
- **Packets:** number of packets in each conversation
- **Bytes:** total size of each conversation
- **Packets A -> B:** number of packets from address A to address B
- **Bytes A -> B:** size of packets sent from address A to address B
- **Packets A <- B:** number of packets from address B to address A
- **Bytes A <- B:** size of packets sent from address B to address A
- **Duration:** how long the conversation lasted



The IPv4 tab will display conversations between IPv4 addresses.



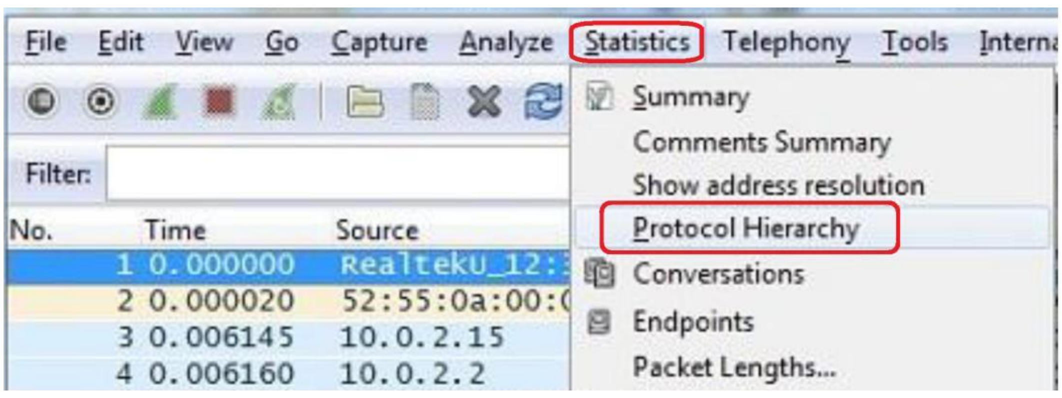
The Transmission Control Protocol (TCP) tab will display any TCP conversations. The addresses will typically be IPv4 addresses, but IPv6 is possible. This tab also has columns for the port of each side of the conversation, which are labeled Port A and Port B. The User Datagram Protocol (UDP) tab will display any UDP conversations.



During a network assessment, it is important to understand what protocols are being communicated over the network, which devices are using them, and in what volume the protocols are seen. Wireshark’s Protocol Hierarchy window provides statistics on protocols in a capture file. Analysts should use the Protocol Hierarchy to:

- Identify which protocols are being used
- Identify the volume at which protocols are being used
- Identify any protocols that are present on the network but should not be

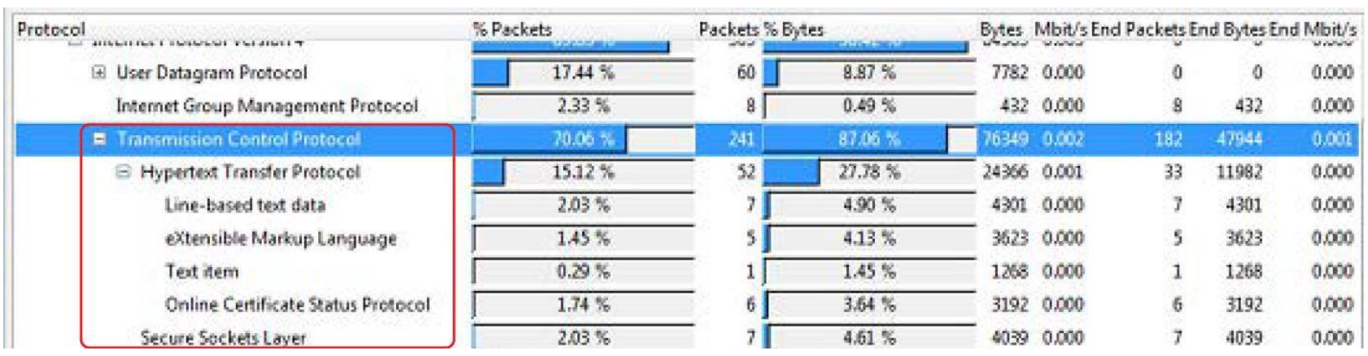
Statistics -> Protocol Hierarchy



The **Protocol Hierarchy** window is a tree of all the protocols found in the capture, with the lowest level protocols at the top and a breakdown of each next level protocol underneath. Each row contains the statistical value of one protocol; the columns contain the following information:

- **Protocol:** protocol name
- **Percent Packets:** percentage of protocol packets in relation to all packets
- **Packets:** total number of packets in the protocol
- **Percent Bytes:** percentage of protocol bytes in relation to the total bytes
- **Bytes:** total number of bytes in the protocol
- **Mbits/s:** bandwidth of the protocol in relation to the capture time (megabits/second)
- **End Packets:** number of packets where the protocol was the highest protocol in the stack
- **End Bytes:** number of bytes where the protocol was the highest protocol in the stack

By clicking the plus (“+”) sign to the left of a protocol, the user can expand the breakdown of higher level protocols under it.



Protocol	% Packets	Packets	% Bytes	Bytes	Mbit/s	End Packets	End Bytes	End Mbit/s
User Datagram Protocol	17.44 %	60	8.87 %	7782	0.000	0	0	0.000
Internet Group Management Protocol	2.33 %	8	0.49 %	432	0.000	8	432	0.000
Transmission Control Protocol	70.06 %	241	87.06 %	76349	0.002	182	47944	0.001
Hypertext Transfer Protocol	15.12 %	52	27.78 %	24366	0.001	33	11982	0.000
Line-based text data	2.03 %	7	4.90 %	4301	0.000	7	4301	0.000
eXtensible Markup Language	1.45 %	5	4.13 %	3623	0.000	5	3623	0.000
Text item	0.29 %	1	1.45 %	1268	0.000	1	1268	0.000
Online Certificate Status Protocol	1.74 %	6	3.64 %	3192	0.000	6	3192	0.000
Secure Sockets Layer	2.03 %	7	4.61 %	4039	0.000	7	4039	0.000

Examining the protocol hierarchy is an excellent way for analysts to characterize traffic. It allows analysts to identify which protocols are being used, as well as the volume of each protocol. Any unusual protocols found should be investigated further.

The **ARP protocol** maps IP addresses to MAC addresses within the local network. ARP can be filtered in Wireshark by using the **arp** filter. ARP traffic is made up of requests and responses. In an ARP request, a host broadcasts a target IP address; essentially, the host is asking what MAC address is associated with that particular IP. The device at the targeted IP will respond with its IP and MAC addresses.

It is important for analysts to note any ARP requests that do not have a corresponding response, as this indicates that a device was connected to the network at one time but has since been improperly removed. Analysts should note which IPs are found in ARP requests and which are found in responses, and then compare the two lists. Any IPs noted in the former but not found in the latter should be documented.

The **DHCP protocol** automatically provides hosts with an IP address and other configuration information, such as subnet mask and default gateway. DHCP is composed of four parts:

- **Discover:** client broadcasts discover messages, which are IP address lease requests.
- **Offer:** server receives a discover message and responds with an IP lease offer.
- **Request:** the client responds to the offer by requesting the IP address offered by the server.
- **Acknowledge:** the server sends an acknowledgement with configuration information

Analysts should look for IP requests that are not a part of the subnet to identify rogue hosts.

Domain Name System (DNS) translates human readable domain names to IP addresses. Unlike traditional IT systems, ICS systems should have very few instances of DNS traffic. Analysts should investigate all DNS queries and associated responses. Most DNS queries request the IP address of a given domain name, although other types of queries exist. The server will send a response back using a DNS reply, typically returning the requested address unless there is an error.

The analyst should be aware of any unusual or out-of-place domain names that are being queried by devices on its network. For instance, if the network has no connection to the Internet, devices should not be sending DNS queries for external websites.

NetBIOS allows applications to communicate within a local area network (LAN). The NetBIOS Name Service (NBNS), like DNS, maps a name to an IP address. The NetBIOS name can give analysts a description of the function or group the host belongs to.

The **Hypertext Transfer Protocol (HTTP)** is the standard protocol for the World Wide Web. Accessing a website, for instance, is accomplished through HTTP. The protocol typically communicates over port 80 on the server side; however, other ports are occasionally used such as 8080.

Most HTTP communications consist of the client sending a **Request** to the server to retrieve or modify a given resource, and the server communicates back a **Response** to the client. HTTP Requests are sent from the client to the server to request data or make a change to the system.

Some common HTTP request methods that analysts should pay particular attention to are:

- **GET:** requests data from a particular source
- **POST:** requests that the server accepts
- **PUT:** requests that the enclosed data be stored
- **DELETE:** request the client-specified resource be deleted
- **PATCH:** request the client-specified resource be partially modified

GET Requests are by far the most common HTTP request, but the analyst should still investigate any GET requests that request potentially suspicious files or resources.

Logins and authorizations are usually accomplished via the POST command but may also use the PUT command. Filtering on POST and PUT requests will show most attempted logins via HTTP. However, for some web software, authorization may also occur in a specific field in HTTP GET requests, so this filter is not guaranteed to capture all logins.

The POST, PUT, DELETE, and PATCH commands all modify data on the server in some way and should be investigated. POST (and more rarely PUT) requests are often necessary for an HTTP server, but DELETE and PATCH typically should not be enabled. Any attempts to send a DELETE or PATCH request should be documented, as well as what Response Code the server sends back.

The HTTP server will respond to requests with an HTTP response. HTTP responses may contain server information of the source. This server information can help enumerate the system and identify potential software that can be analyzed on the host.

HTTP Response Codes are sent from the server to the client in response to the client's HTTP requests. There are multiple codes in ranges of one hundred, corresponding to different categories. Two HTTP Response Codes in the 400 range may indicate failed login or authorization attempts, specifically the **401** and **403** codes. These response codes, as well as the HTTP Requests that caused them, should always be investigated.

HTTPS is the secure version of HTTP, using Secure Socket Layer (SSL) or Transport Layer Security (TLS) for encryption. HTTPS typically communicates over port 443 on the server side. The analyst should note any communications using HTTPS; however, because the payload is encrypted, no further analysis can be accomplished. Request methods, response codes, and any other HTTP-specific information are unreadable due to the encryption of HTTPS. The **ssl** filter can be used to display all HTTPS traffic.

Telnet provides a remote connection to a host and grants the user command line access to that host. It can provide a connection over a Local Area Network (LAN) or over the Internet. It uses port 23. Telnet has no encryption or other built-in security. Login credentials are sent in the clear. All commands sent to the host are also readable in clear text, as are the responses from the host. Because of this, Telnet has serious security concerns and should never be used. Secure Shell (SSH) provides a similar capability but uses encryption, so it is the preferred solution. The **telnet** filter can be used to display all telnet traffic.

Secure Shell provides remote command execution, providing a similar functionality to Telnet. Unlike Telnet, however, SSH uses encryption to secure the communication channel. It uses port 22 over TCP. Its use should be limited since it allows a remote user to execute commands on a host. SSH communications should be accounted for, and any sessions seen in traffic should be documented. The contents of the SSH payload, as with HTTPS, is encrypted and, thus, deeper analysis cannot be conducted. The filter **ssh** will display all SSH traffic.

The **File Transfer Protocol (FTP)** is used to retrieve and store files on a server. It utilizes two ports. Port 21 is the control port, which establishes the FTP connection. Port 20 is the data port, which actually transfers the file's data. The **ftp** filter will display all FTP traffic.

FTP servers use a username and password logins. The user enters their username preceded by the USER command. The filter **ftp.request.command ==** "**USER"** will display any USER commands sent by the client. Following the USER command, the client will have to send a password command to authenticate. The password is sent by the PASS command followed by the user's password. The filter **ftp.request.command ==** "**PASS"** will display any traffic with this command.

Multiple commands may be available to the user. The STOR command allows them to store a file on the server. The RETR command allows them to retrieve a specific file from the server. Any FTP command can be filtered on using the **ftp.request.command == "[Command name]"** filter.

Trivial FTP (TFTP) is used to retrieve and store files, similar to FTP, but with less functionality. TFTP only allows reading and writing files, and does not have commands to list, delete, or rename files. It used port 69 and, unlike FTP, does not have separate control and data ports. TFTP has no mechanism for authentication, so it is inherently less secure than FTP. The **tftp** filter will display all TFTP traffic.

The **Remote Desktop Protocol (RDP)** is a protocol designed by Microsoft to provide a remote connection between a user and a remote machine, as well as provide graphical interface that the user can engage to control the remote system. It uses port 3389, over both TCP and UDP. The **rdp** filter will display all RDP traffic.

Types of Data:

- Full Content → Network Traffic stored as PCAP file
- Extracted Content → Webpages, Files, Images, Media
- Session Data → Timestamps; Source/Destination Ports & IP addresses
- Transaction Data → Application layer request-reply information
- Statistical Data → Traffic size, duration
- Metadata → Geographical location, system owner
- Alert Data → IDS triggered alerts

Boolean Logic Syntax:

==	Equal to
||	Or
&&	And
!	Not
!=	Not equal to
>=	Greater than or equal to
<=	Less than or equal to
contains	Searches for pattern
matches	Searches via regular expression

Filters to detect Host Discovery, Port Scanning & Network Attacks

Technique	Wireshark Filter	Tool Command
ARP scanning	arp.dst.hw_mac==00:00:00:00:00:00	arp-scan -l
IP protocol scan	icmp.type==3 and icmp.code==2	nmap -sO <target>
ICMP ping sweep	icmp.type==8 or icmp.type==0	nmap -sn -PE <subnet>
TCP ping sweeps	tcp.dstport==7	nmap -sn -PS/-PA <subnet>
UDP ping sweeps	udp.dstport==7	nmap -sn -PU <subnet>
TCP SYN scan	tcp.flags.syn==1 and tcp.flags.ack==0 and tcp.window_size<=1024	nmap -sS <target>
TCP Connect() scan	tcp.flags.syn==1 and tcp.flags.ack==0 and tcp.window_size>1024	nmap -sT <target>
TCP Null scan	tcp.flags==0	nmap -sN <target>
TCP FIN scan	tcp.flags==0x001	nmap -sF <target>
TCP Xmass scan	tcp.flags.fin==1 && tcp.flags.push==1 && tcp.flags.urg==1	nmap -sX <target>
UDP port scan	icmp.type==3 and icmp.code==3	nmap -sU <target>
ARP poisoning	arp.duplicate-address-detected or arp.duplicate-address-frame	arpspoof
ICMP flood	icmp and data.len > 48	fping, hping
VLAN hoping	dtp or vlan.too_many_tags	frogger
Client deauthentication	wlan.fc.type_subtype == 12	aireplay-ng
Client disassociation	wlan.fc.type_subtype == 10	mdk3, mdk4
Fake AP beacon flood	wlan.fc.type_subtype == 8	mdk3, mdk4
Authentication DoS	wlan.fc.type_subtype == 11	mdk3, mdk4
Redirections	http.response.code >=300 and http.response.code <400	
Packet loss	tcp.analysis.lost_segment or tcp.analysis.retransmission	
Client-side errors	http.response.code >=400 and http.response.code <500	
Server-side errors	http.response.code >500	
Phone-home attempts	http.host matches "some-domain-name"	
Matches zip files	frame contains "\x50\x4B\x03\x04"	
Matches pdf files	frame contains "\x25\x50\x44\x46"	
Matches keywords	frame matches "(?i)(password|secret)"	
Matches email addresses	smtp matches "[a-zA-Z0-9._%+-]+@[a-zA-Z0-9._%+-]"	

Display Filters:

ip.addr == [IPv4 address]	matches specific source/destination address
ip.src == [IPv4 Address]	matches specific source address
ip.dst== [IPv4 Address]	matches specific destination address
eth.addr == [MAC Address]	matches specific MAC address
eth.src == [MAC Address]	matches specific source MAC address
eth.dst == [MAC Address]	matches specific destination MAC address
arp	matches arp traffic
arp.opcode == 1	matches arp request
arp.opcode == 2	matches arp response
arp.dst.proto_ipv4 == [IPv4 Address]	matches arp request for specific address
bootp	matches all DHCP traffic
bootp.type == 1	matches DHCP requests
bootp.type == 2	matches DHCP replies
bootp.option.type == 50	matches DHCP requests of a specific address
dns	matches all dns traffic
dns.flags.response == 0	matches dns queries
dns.flags.response == 1	matches dns responses
dns.qry.name == “[host name]”	matches dns queries for specific address
nbns	matches NetBIOS name service
netbios	matches all NetBIOS traffic
netbios.nb_name == “[NB_name]”	matches NetBIOS name
http	matches all HTTP traffic
http.request	matches HTTP requests
http.response	matches HTTP responses
http.response.code == [code]	matches specific response code
http.request.method == GET	matches HTTP GET requests
http.request.method == POST	matches HTTP POST requests
http.request.method == DELETE	matches HTTP DELETE request
ftp.request.command == “USER”	matches user commands sent by the client.
ftp.request.command == “PASS”	matches traffic with the password command
ftp.request.command == “RETR”	matches traffic with the file retrieve command
ftp.request.command == “STOR”	matches traffic with the file store command

Common Ports:

80	HTTP	Hypertext Transfer Protocol
23	Telnet	Terminal Network
443	HTTPS	Hypertext Transfer Protocol Secure
21	FTP	File Transfer Protocol
22	SSH	Secure Shell
25	SMTP	Simple Mail Transfer Protocol
3389	RDP	Remote Desktop Protocol
110	POP3	Post Office Protocol
445	SMB	Server Message Block
137/138/139	NetBIOS	Network Basic Input/Output System
143	IMAP	Internet Message Access
53	DNS	Domain Name System
134/135	MSRPC	RPC Endpoint Mapper
3306	MySQL	MySQL Database
8080	HTTP	HTTP Proxy
1723	PPTP	Point to Point Tunneling Protocol
995	POPS3S	Post Office Protocol 3 Secure
993	IMAPS	Internet Message Access Secure
5900	VNC	Virtual Network Computing
631	IPP	Internet Printing Protocol
161/162	SNMP	Simple Network Management
123	NTP	Network Time Protocol
1434	MS-SQL	Microsoft SQL Server
67/68	DHCP	Dynamic Host Configuration
500	ISAKMP	IpSec VPNs
520	RIP	Routing Information Protocol
1900	UPNP	Universal Plug-and-Play
514	Syslog	Unix Log Daemon
179	BGP	Border Gateway Protocol
1293	IPSec	Internet Protocol Security
88	Kerberos	Network Authentication Protocol
1701	L2TP	Layer 2 Tunneling Protocol
389	LDAP	Lightweight Directory Access
636	LDAPS	Lightweight Directory Access Secure
989/990	FTPS	File Transfer Protocol Secure
119	NNTP	Network News Transfer Protocol
1813	RADIUS	Remote Authentication Dial-in User Service
5004/5005	RTP	Real-time Transport Protocol
5060/5061	SIP	Session Initiation Protocol
443	SSL	Secure Sockets Layer
49	TACACS+	AAA Administration
69	TFTP	Trivial File Transfer Protocol