Wireshark 101

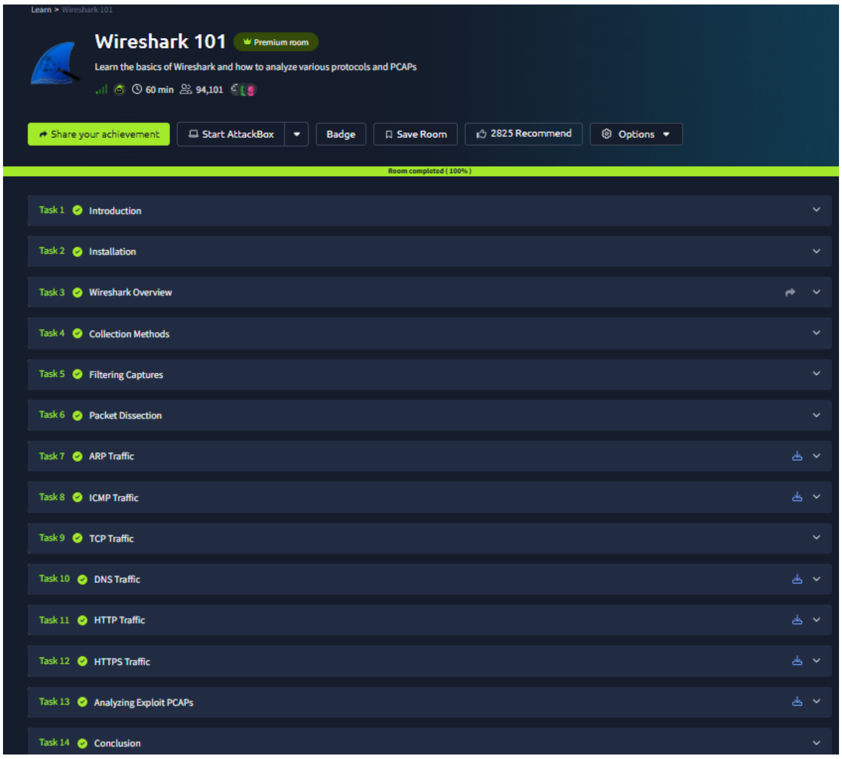
Platform: tryhackme

Frome: Asem Reda

Diploma: Cybersecurity 87

Room link: https://tryhackme.com/room/wireshark

Introduction: This report analyzes and covers the Wireshark 101 ROM on the TryHackMe platform, which consists of 14 tasks explaining the fundamentals of using Wireshark to capture and analyze packet data. The report includes a brief explanation of each task, along with screenshots of the steps involved and answers to frequently asked questions.



Learn > Wireshark 101

Wireshark 101 Premium room

Learn the basics of Wireshark and how to analyze various protocols and PCAPs

60 min 94,101

Room completed (100%)

Task 1 ✓ Introduction

Task 2 ✓ Installation

Task 3 ✓ Wireshark Overview

Task 4 ✓ Collection Methods

Task 5 ✓ Filtering Captures

Task 6 ✓ Packet Dissection

Task 7 ✓ ARP Traffic

Task 8 ✓ ICMP Traffic

Task 9 ✓ TCP Traffic

Task 10 ✓ DNS Traffic

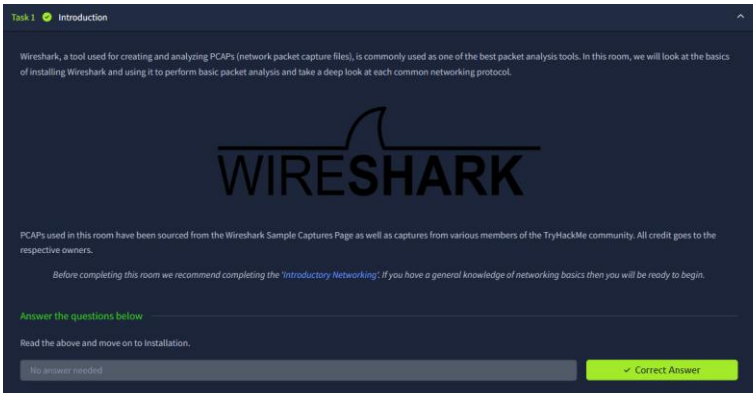
Task 11 ✓ HTTP Traffic

Task 12 ✓ HTTPS Traffic

Task 13 ✓ Analyzing Exploit PCAPs

Task 14 ✓ Conclusion

Task 1: Introduction



Task 1 Introduction

Wireshark, a tool used for creating and analyzing PCAPs (network packet capture files), is commonly used as one of the best packet analysis tools. In this room, we will look at the basics of installing Wireshark and using it to perform basic packet analysis and take a deep look at each common networking protocol.

PCAPs used in this room have been sourced from the Wireshark Sample Captures Page as well as captures from various members of the TryHackMe community. All credit goes to the respective owners.

Before completing this room we recommend completing the 'Introductory Networking'. If you have a general knowledge of networking basics then you will be ready to begin.

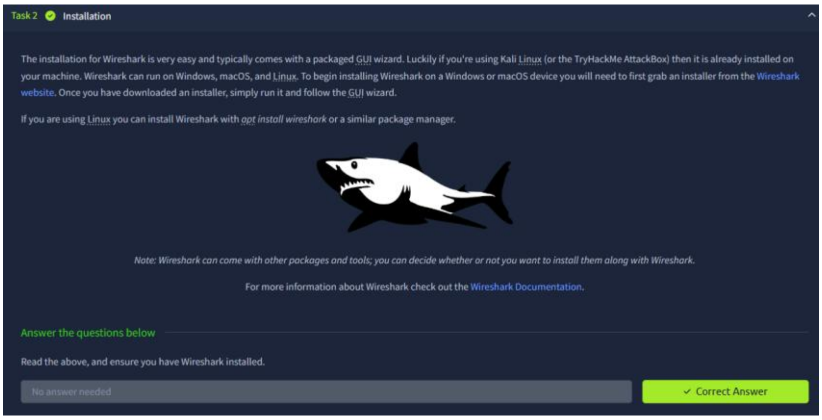
Answer the questions below

Read the above and move on to Installation.

No answer needed

✓ Correct Answer

Task 2: Installation



Task 2 Installation

The installation for Wireshark is very easy and typically comes with a packaged GUI wizard. Luckily if you're using Kali Linux (or the TryHackMe AttackBox) then it is already installed on your machine. Wireshark can run on Windows, macOS, and Linux. To begin installing Wireshark on a Windows or macOS device you will need to first grab an installer from the Wireshark website. Once you have downloaded an installer, simply run it and follow the GUI wizard.

If you are using Linux you can install Wireshark with *apt install wireshark* or a similar package manager.

Note: Wireshark can come with other packages and tools; you can decide whether or not you want to install them along with Wireshark.

For more information about Wireshark check out the Wireshark Documentation.

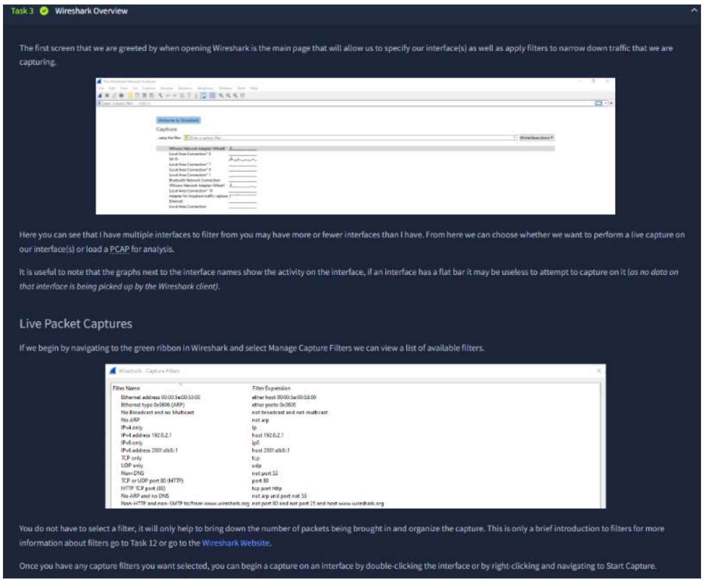
Answer the questions below

Read the above, and ensure you have Wireshark installed.

No answer needed

✓ Correct Answer

Task 3: Wireshark Overview



Task 3 Wireshark Overview

The first screen that we are greeted by when opening Wireshark is the main page that will allow us to specify our interface(s) as well as apply filters to narrow down traffic that we are capturing.

Here you can see that I have multiple interfaces to filter from you may have more or fewer interfaces than I have. From here we can choose whether we want to perform a live capture on our interface(s) or load a PCAP for analysis.

It is useful to note that the graphs next to the interface names show the activity on the interface, if an interface has a flat bar it may be useless to attempt to capture on it (*as no data on that interface is being picked up by the Wireshark client*).

Live Packet Captures

If we begin by navigating to the green ribbon in Wireshark and select Manage Capture Filters we can view a list of available filters.

Wireshark · Capture Filters

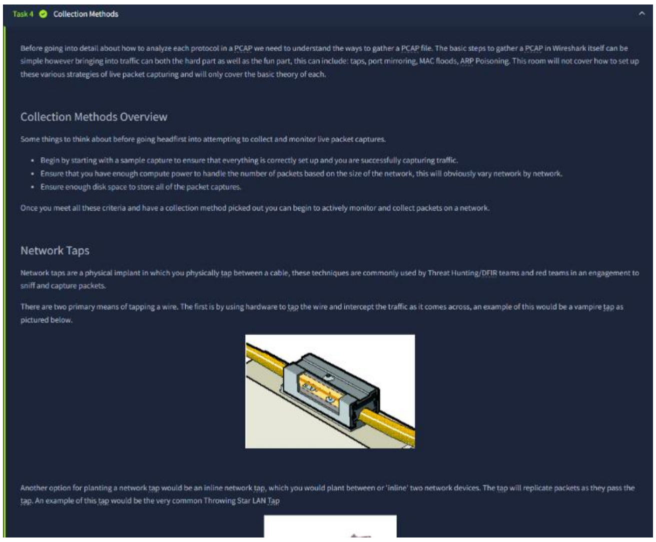
Filter Name	Filter Expression
Ethernet address 00:00:5e:00:53:00	ether host 00:00:5e:00:53:00
Ethernet type 0x0806 (ARP)	ether proto 0x0806
No Broadcast and no Multicast	not broadcast and not multicast
No ARP	not arp
IPv4 only	ip
IPv4 address 192.0.2.1	host 192.0.2.1
IPv6 only	ip6
IPv6 address 2001:db8::1	host 2001:db8::1
TCP only	tcp
UDP only	udp
Non-DNS	not port 53
TCP or UDP port 80 (HTTP)	port 80
HTTP TCP port (80)	tcp port http
No ARP and no DNS	not arp and port not 53
Non-HTTP and non-SMTP to/from www.wireshark.org	not port 80 and not port 25 and host www.wireshark.org

You do not have to select a filter, it will only help to bring down the number of packets being brought in and organize the capture. This is only a brief introduction to filters for more information about filters go to Task 12 or go to the Wireshark Website.

Once you have any capture filters you want selected, you can begin a capture on an interface by double-clicking the interface or by right-clicking and navigating to Start Capture.

When opening **Wireshark**, we see the main page where we can select a network interface or load a **PCAP** file. Capture filters can be used to reduce the number of packets, then we start capturing by clicking on the chosen interface. After collecting the packets, we stop the capture and analyze them. Each packet displays information such as **time, source, destination, protocol, and length**. Wireshark also uses **different colors** to make it easier to detect anomalies and identify protocols quickly.

Task 4: Collection Methods



Task 4 Collection Methods

Before going into detail about how to analyze each protocol in a PCAP we need to understand the ways to gather a PCAP file. The basic steps to gather a PCAP in Wireshark itself can be simple however bringing into traffic can both the hard part as well as the fun part, this can include: taps, port mirroring, MAC floods, ARP Poisoning. This room will not cover how to set up these various strategies of live packet capturing and will only cover the basic theory of each.

Collection Methods Overview

Some things to think about before going headfirst into attempting to collect and monitor live packet captures.

- Begin by starting with a sample capture to ensure that everything is correctly set up and you are successfully capturing traffic.
- Ensure that you have enough compute power to handle the number of packets based on the size of the network, this will obviously vary network by network.
- Ensure enough disk space to store all of the packet captures.

Once you meet all these criteria and have a collection method picked out you can begin to actively monitor and collect packets on a network.

Network Taps

Network taps are a physical implant in which you physically tap between a cable, these techniques are commonly used by Threat Hunting/DFIR teams and red teams in an engagement to sniff and capture packets.

There are two primary means of tapping a wire. The first is by using hardware to tap the wire and intercept the traffic as it comes across, an example of this would be a vampire tap as pictured below.

Another option for planting a network tap would be an inline network tap, which you would plant between or 'inline' two network devices. The tap will replicate packets as they pass the tap. An example of this tap would be the very common Throwing Star LAN Tap

PCAP Collection Methods (Short Summary)

Before analyzing **PCAP** files, it's important to understand the main methods of capturing traffic:

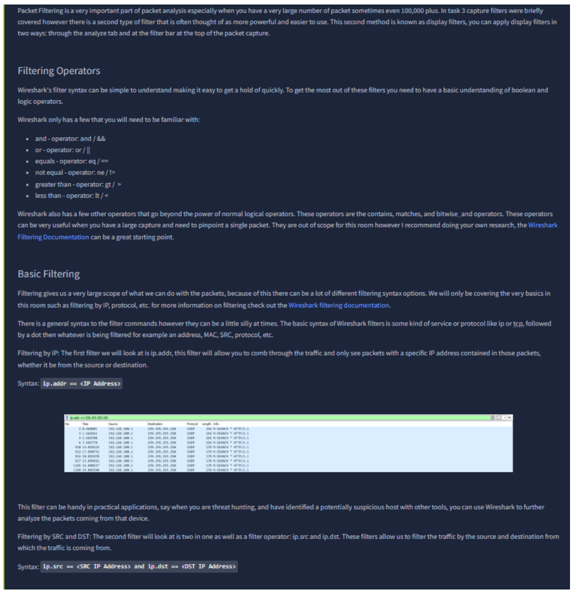
Network Taps: Physical devices placed on network cables to monitor traffic.

MAC Flooding: Overloading the switch’s MAC table so it broadcasts packets to all ports.

ARP Spoofing: Redirecting network traffic to the monitoring device.

Before capturing, ensure you run a test capture, have enough computing power, and sufficient disk space.

Task 5: Filtering Captures



Packet Filtering is a very important part of packet analysis especially when you have a very large number of packet sometimes even 100,000 plus. In task 3 capture filters were briefly covered however there is a second type of filter that is often thought of as more powerful and easier to use. This second method is known as display filters, you can apply display filters in two ways: through the analyze tab and at the filter bar at the top of the packet capture.

Filtering Operators

Wireshark's filter syntax can be simple to understand making it easy to get a hold of quickly. To get the most out of these filters you need to have a basic understanding of boolean and logic operators.

Wireshark only has a few that you will need to be familiar with:

- and - operator: and / &&
- or - operator: or / ||
- equals - operator: eq / ==
- not equal - operator: ne / !=
- greater than - operator: gt / >
- less than - operator: lt / <

Wireshark also has a few other operators that go beyond the power of normal logical operators. These operators are the contains, matches, and bitwise_and operators. These operators can be very useful when you have a large capture and need to pinpoint a single packet. They are out of scope for this room however I recommend doing your own research, the Wireshark Filtering Documentation can be a great starting point.

Basic Filtering

Filtering gives us a very large scope of what we can do with the packets, because of this there can be a lot of different filtering syntax options. We will only be covering the very basics in this room such as filtering by IP, protocol, etc. for more information on filtering check out the Wireshark filtering documentation.

There is a general syntax to the filter commands however they can be a little silly at times. The basic syntax of Wireshark filters is some kind of service or protocol like ip or tcp, followed by a dot then whatever is being filtered for example an address, MAC, SRC, protocol, etc.

Filtering by IP: The first filter we will look at is ip.addr, this filter will allow you to comb through the traffic and only see packets with a specific IP address contained in those packets, whether it be from the source or destination.

Syntax: `ip.addr == <IP Address>`

```
ip.addr == 239.255.255.250
No.  Time       Source         Destination      Protocol Length Info
   2 0.660801   192.168.100.1  239.255.255.250  SSDP     216 M-SEARCH * HTTP/1.1
   3 1.662661   192.168.100.1  239.255.255.250  SSDP     216 M-SEARCH * HTTP/1.1
   4 2.665708   192.168.100.1  239.255.255.250  SSDP     216 M-SEARCH * HTTP/1.1
   6 3.665779   192.168.100.1  239.255.255.250  SSDP     216 M-SEARCH * HTTP/1.1
 910 14.020224  192.168.100.1  239.255.255.250  SSDP     179 M-SEARCH * HTTP/1.1
 912 17.030732  192.168.100.1  239.255.255.250  SSDP     179 M-SEARCH * HTTP/1.1
 914 20.031970  192.168.100.1  239.255.255.250  SSDP     179 M-SEARCH * HTTP/1.1
 917 23.039562  192.168.100.1  239.255.255.250  SSDP     179 M-SEARCH * HTTP/1.1
1165 26.040227  192.168.100.1  239.255.255.250  SSDP     179 M-SEARCH * HTTP/1.1
1168 29.043590  192.168.100.1  239.255.255.250  SSDP     179 M-SEARCH * HTTP/1.1
```

This filter can be handy in practical applications, say when you are threat hunting, and have identified a potentially suspicious host with other tools, you can use Wireshark to further analyze the packets coming from that device.

Filtering by SRC and DST: The second filter will look at is two in one as well as a filter operator: ip.src and ip.dst. These filters allow us to filter the traffic by the source and destination from which the traffic is coming from.

Syntax: `ip.src == <SRC IP Address> and ip.dst == <DST IP Address>`

Filtering Captures (Wireshark) – Summary

Filters help simplify packet analysis, especially with large captures.

Display Filters: Applied after capturing via the filter bar or the Analyze menu.

Basic Operators: and, or, eq, ne, gt, lt

Common Filters:

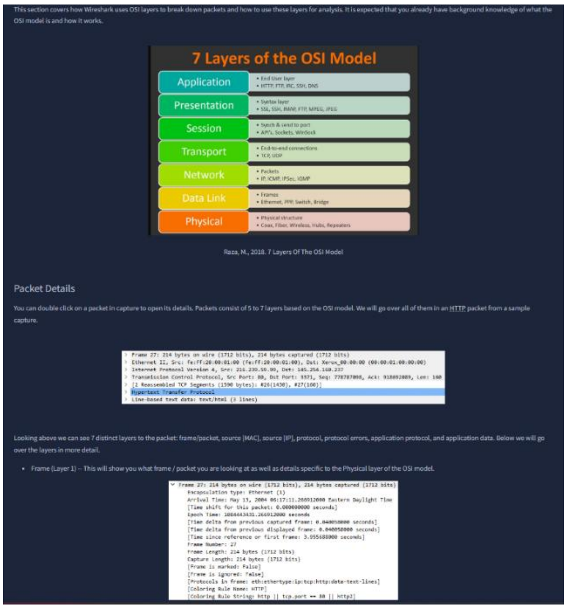
By IP: ip.addr == <IP>

By Source and Destination: ip.src == <SRC> and ip.dst == <DST>

By Protocol or Port: tcp.port eq <Port> / udp.port eq <Port>

Filters allow focusing on important packets and analyzing them efficiently.

Task 6: Packet Dissection



This section covers how Wireshark uses OSI layers to break down packets and how to use these layers for analysis. It is expected that you already have background knowledge of what the OSI model is and how it works.

Raza, M., 2018. 7 Layers Of The OSI Model

Packet Details

You can double click on a packet in capture to open its details. Packets consist of 5 to 7 layers based on the OSI model. We will go over all of them in an HTTP packet from a sample capture.

```
> Frame 27: 214 bytes on wire (1712 bits), 214 bytes captured (1712 bits)
> Ethernet II, Src: fe:ff:20:00:01:00 (fe:ff:20:00:01:00), Dst: Xerox_00:00:00 (00:00:01:00:00:00)
> Internet Protocol Version 4, Src: 216.239.59.99, Dst: 145.254.160.237
> Transmission Control Protocol, Src Port: 80, Dst Port: 3371, Seq: 778787808, Ack: 918692089, Len: 160
> [2 Reassembled TCP Segments (1590 bytes): #26(1430), #27(160)]
> Hypertext Transfer Protocol
> Line-based text data: text/html (3 lines)
```

Looking above we can see 7 distinct layers to the packet: frame/packet, source [MAC], source [IP], protocol, protocol errors, application protocol, and application data. Below we will go over the layers in more detail.

- Frame (Layer 1) -- This will show you what frame / packet you are looking at as well as details specific to the Physical layer of the OSI model.

```
v Frame 27: 214 bytes on wire (1712 bits), 214 bytes captured (1712 bits)
    Encapsulation type: Ethernet (1)
    Arrival Time: May 13, 2004 06:17:11.266912000 Eastern Daylight Time
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1084443431.266912000 seconds
    [Time delta from previous captured frame: 0.040050000 seconds]
    [Time delta from previous displayed frame: 0.040050000 seconds]
    [Time since reference or first frame: 3.955688000 seconds]
    Frame Number: 27
    Frame Length: 214 bytes (1712 bits)
    Capture Length: 214 bytes (1712 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:tcp:http:data-text-lines]
    [Coloring Rule Name: HTTP]
    [Coloring Rule String: http || tcp.port == 80 || http2]
```

Packet Dissection (Wireshark) – Summary

Wireshark uses the **OSI model** to break down packets for analysis. A captured packet can contain **5–7 layers**, typically:

Frame (Layer 1): Shows packet/frame details at the Physical layer.

Source [MAC] (Layer 2): Displays source and destination MAC addresses (Data Link layer).

Source [IP] (Layer 3): Displays source and destination IP addresses (Network layer).

Protocol (Layer 4): Shows transport protocol (TCP/UDP) and source/destination ports (Transport layer).

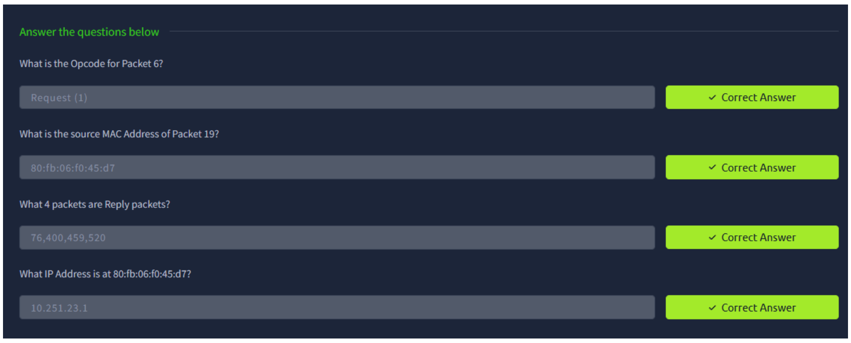
Protocol Errors: Details TCP segments needing reassembly.

Application Protocol (Layer 5): Displays protocol-specific details like HTTP, FTP, SMB (Application layer).

Application Data: Shows application-specific data.

Understanding these layers helps in analyzing packets and the protocols they carry.

Task 7: ARP Traffic



Answer the questions below

What is the Opcode for Packet 6?

Request (1)

What is the source MAC Address of Packet 19?

80:fb:06:f0:45:d7

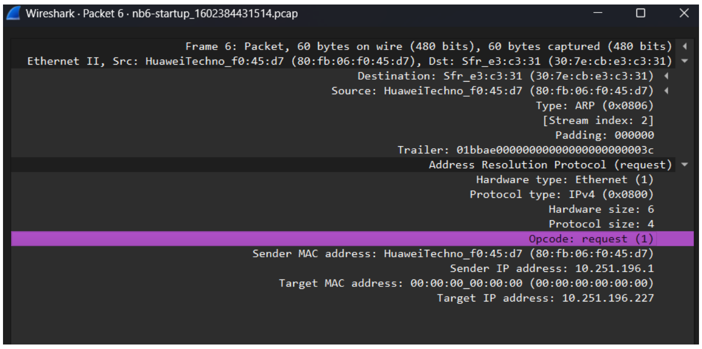
What 4 packets are Reply packets?

76,400,459,520

What IP Address is at 80:fb:06:f0:45:d7?

10.251.23.1

First question:



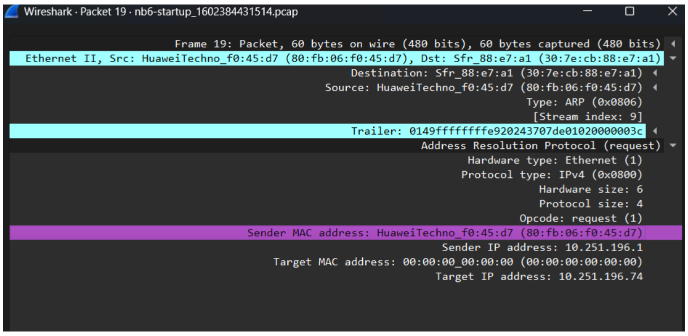
Wireshark · Packet 6 · nb6-startup_1602384431514.pcap

```
                 Frame 6: Packet, 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
 Ethernet II, Src: HuaweiTechno_f0:45:d7 (80:fb:06:f0:45:d7), Dst: Sfr_e3:c3:31 (30:7e:cb:e3:c3:31)
                                          Destination: Sfr_e3:c3:31 (30:7e:cb:e3:c3:31)
                                       Source: HuaweiTechno_f0:45:d7 (80:fb:06:f0:45:d7)
                                                                       Type: ARP (0x0806)
                                                                        [Stream index: 2]
                                                                          Padding: 000000
                                         Trailer: 01bbae00000000000000000000003c
                                                   Address Resolution Protocol (request)
                                                                 Hardware type: Ethernet (1)
                                                                 Protocol type: IPv4 (0x0800)
                                                                         Hardware size: 6
                                                                         Protocol size: 4
                                                                      Opcode: request (1)
                          Sender MAC address: HuaweiTechno_f0:45:d7 (80:fb:06:f0:45:d7)
                                                           Sender IP address: 10.251.196.1
                             Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
                                                          Target IP address: 10.251.196.227
```

Select package number 6

Select the Address Resolution Protocol (request) section

Second question:



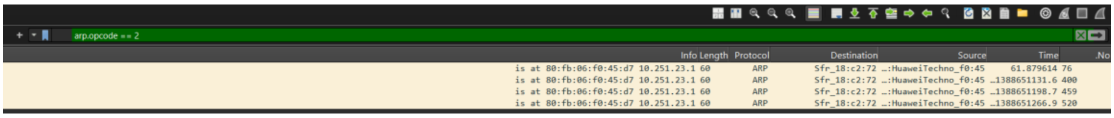
Wireshark · Packet 19 · nb6-startup_1602384431514.pcap

```
                Frame 19: Packet, 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
 Ethernet II, Src: HuaweiTechno_f0:45:d7 (80:fb:06:f0:45:d7), Dst: Sfr_88:e7:a1 (30:7e:cb:88:e7:a1)
                                          Destination: Sfr_88:e7:a1 (30:7e:cb:88:e7:a1)
                                       Source: HuaweiTechno_f0:45:d7 (80:fb:06:f0:45:d7)
                                                                       Type: ARP (0x0806)
                                                                        [Stream index: 9]
                                  Trailer: 0149ffffffffe920243707de01020000003c
                                                   Address Resolution Protocol (request)
                                                                 Hardware type: Ethernet (1)
                                                                 Protocol type: IPv4 (0x0800)
                                                                         Hardware size: 6
                                                                         Protocol size: 4
                                                                      Opcode: request (1)
                          Sender MAC address: HuaweiTechno_f0:45:d7 (80:fb:06:f0:45:d7)
                                                           Sender IP address: 10.251.196.1
                             Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
                                                           Target IP address: 10.251.196.74
```

Select package number 19

Select the Address Resolution Protocol (request) section

Third question:



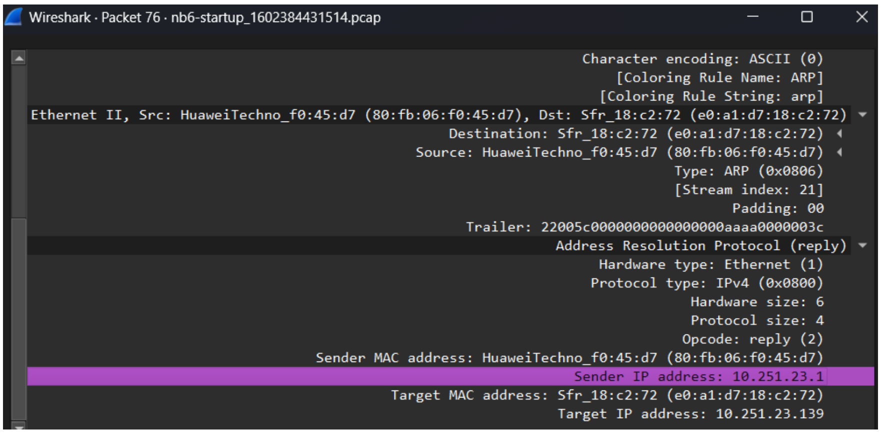
arp.opcode == 2

```
                                  Info Length  Protocol      Destination              Source        Time   .No
is at 80:fb:06:f0:45:d7 10.251.23.1 60   ARP          Sfr_18:c2:72 …:HuaweiTechno_f0:45     61.879614 76
is at 80:fb:06:f0:45:d7 10.251.23.1 60   ARP          Sfr_18:c2:72 …:HuaweiTechno_f0:45 …1388651131.6 400
is at 80:fb:06:f0:45:d7 10.251.23.1 60   ARP          Sfr_18:c2:72 …:HuaweiTechno_f0:45 …1388651198.7 459
is at 80:fb:06:f0:45:d7 10.251.23.1 60   ARP          Sfr_18:c2:72 …:HuaweiTechno_f0:45 …1388651266.9 520
```

Enter the following code in the filter: arp.opcode == 2

Then enter the package numbers that appeared

Fourth question:



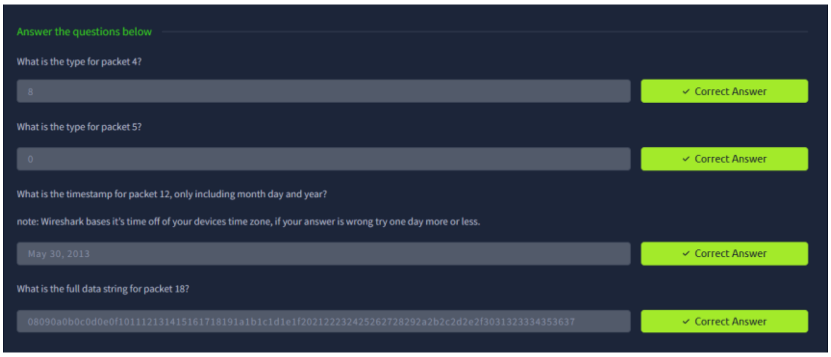
Wireshark · Packet 76 · nb6-startup_1602384431514.pcap

```
                                     Character encoding: ASCII (0)
                                        [Coloring Rule Name: ARP]
                                      [Coloring Rule String: arp]
Ethernet II, Src: HuaweiTechno_f0:45:d7 (80:fb:06:f0:45:d7), Dst: Sfr_18:c2:72 (e0:a1:d7:18:c2:72)
                      Destination: Sfr_18:c2:72 (e0:a1:d7:18:c2:72)
                  Source: HuaweiTechno_f0:45:d7 (80:fb:06:f0:45:d7)
                                               Type: ARP (0x0806)
                                               [Stream index: 21]
                                                      Padding: 00
                   Trailer: 22005c0000000000000000aaaa0000003c
                                 Address Resolution Protocol (reply)
                                          Hardware type: Ethernet (1)
                                         Protocol type: IPv4 (0x0800)
                                                   Hardware size: 6
                                                    Protocol size: 4
                                                   Opcode: reply (2)
     Sender MAC address: HuaweiTechno_f0:45:d7 (80:fb:06:f0:45:d7)
                                     Sender IP address: 10.251.23.1
            Target MAC address: Sfr_18:c2:72 (e0:a1:d7:18:c2:72)
                                    Target IP address: 10.251.23.139
```

In the filter, write eth.addr == 80:fb:06:f0:45:d7

We are looking for packets containing ARP Reply (opcode 2) (Reply)

Task 8: ICMP Traffic



Answer the questions below

What is the type for packet 4?

8

✓ Correct Answer

What is the type for packet 5?

0

✓ Correct Answer

What is the timestamp for packet 12, only including month day and year?

note: Wireshark bases it's time off of your devices time zone, if your answer is wrong try one day more or less.

May 30, 2013

✓ Correct Answer

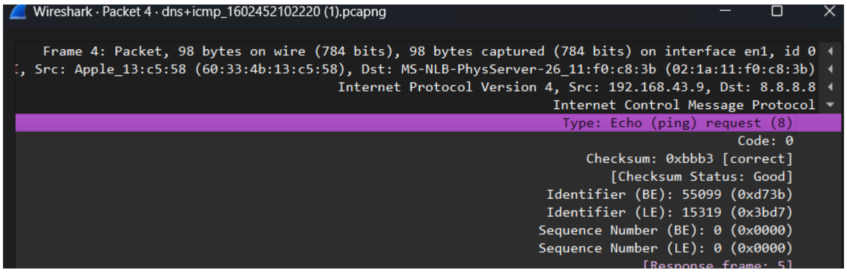
What is the full data string for packet 18?

08090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f3031323334353637

✓ Correct Answer

First question:

Go to package number 4, then go to Internet Control Message Protocol. You will find the package type there

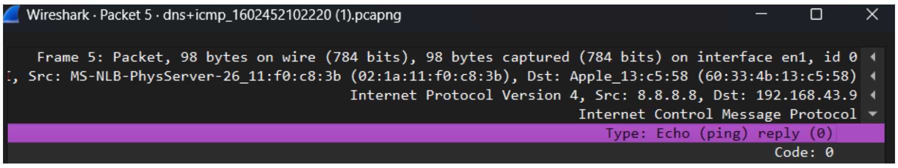


Wireshark · Packet 4 · dns+icmp_1602452102220 (1).pcapng

```
  Frame 4: Packet, 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface en1, id 0
:, Src: Apple_13:c5:58 (60:33:4b:13:c5:58), Dst: MS-NLB-PhysServer-26_11:f0:c8:3b (02:1a:11:f0:c8:3b)
                        Internet Protocol Version 4, Src: 192.168.43.9, Dst: 8.8.8.8
                                                    Internet Control Message Protocol
                                                         Type: Echo (ping) request (8)
                                                                             Code: 0
                                                         Checksum: 0xbbb3 [correct]
                                                             [Checksum Status: Good]
                                                        Identifier (BE): 55099 (0xd73b)
                                                        Identifier (LE): 15319 (0x3bd7)
                                                       Sequence Number (BE): 0 (0x0000)
                                                       Sequence Number (LE): 0 (0x0000)
                                                                  [Response frame: 5]
```

Second question:

Go to package number 5, then go to Internet Control Message Protocol. You will find the package type there

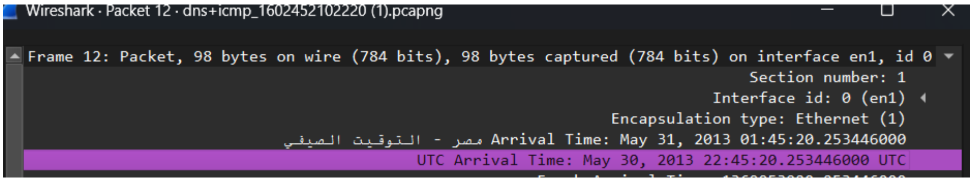


Wireshark · Packet 5 · dns+icmp_1602452102220 (1).pcapng

```
  Frame 5: Packet, 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface en1, id 0
:, Src: MS-NLB-PhysServer-26_11:f0:c8:3b (02:1a:11:f0:c8:3b), Dst: Apple_13:c5:58 (60:33:4b:13:c5:58)
                           Internet Protocol Version 4, Src: 8.8.8.8, Dst: 192.168.43.9
                                                       Internet Control Message Protocol
                                                                Type: Echo (ping) reply (0)
                                                                                    Code: 0
```

Third question:

Access package number 12, then access section one

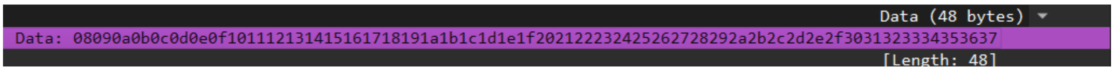


Wireshark · Packet 12 · dns+icmp_1602452102220 (1).pcapng

```
Frame 12: Packet, 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface en1, id 0
                                                                    Section number: 1
                                                                 Interface id: 0 (en1)
                                                          Encapsulation type: Ethernet (1)
                          مصر - التوقيت الصيفي Arrival Time: May 31, 2013 01:45:20.253446000
                                        UTC Arrival Time: May 30, 2013 22:45:20.253446000 UTC
```

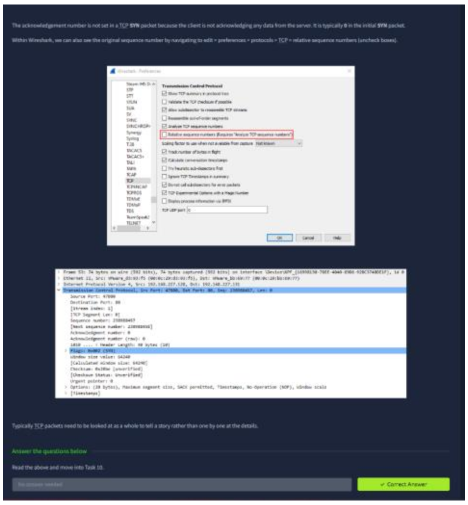
Fourth question:

Go to package number 18, then go to the Data section (48 bytes)



```
                                                                              Data (48 bytes)
Data: 08090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f3031323334353637
                                                                                   [Length: 48]
```

Task 9: TCP Traffic



The acknowledgement number is not set in a TCP SYN packet because the client is not acknowledging any data from the server. It is typically 0 in the initial SYN packet.

Within Wireshark, we can also see the original sequence number by navigating to edit > preferences > protocols > TCP > relative sequence numbers (uncheck boxes).

Transmission Control Protocol

☑ Show TCP summary in protocol tree

☐ Validate the TCP checksum if possible

☑ Allow subdissector to reassemble TCP streams

☐ Reassemble out-of-order segments

☑ Analyze TCP sequence numbers

☐ Relative sequence numbers (Requires "Analyze TCP sequence numbers")

Scaling factor to use when not available from capture Not known

☑ Track number of bytes in flight

☑ Calculate conversation timestamps

☐ Try heuristic sub-dissectors first

☐ Ignore TCP Timestamps in summary

☑ Do not call subdissectors for error packets

☑ TCP Experimental Options with a Magic Number

☐ Display process information via IPFIX

TCP UDP port 0

Typically TCP packets need to be looked at as a whole to tell a story rather than one by one at the details.

Answer the questions below

Read the above and move into Task 10.

No answer needed

✓ Correct Answer

Short Summary

TCP handles reliable delivery, ordering, and error correction.

In Wireshark, TCP behavior helps identify open/closed ports during scans.

If a port is closed, the server responds with RST, ACK.

The key TCP handshake uses three packets:

SYN

SYN-ACK

ACK

Any disruption or an unexpected RST may indicate suspicious activity.

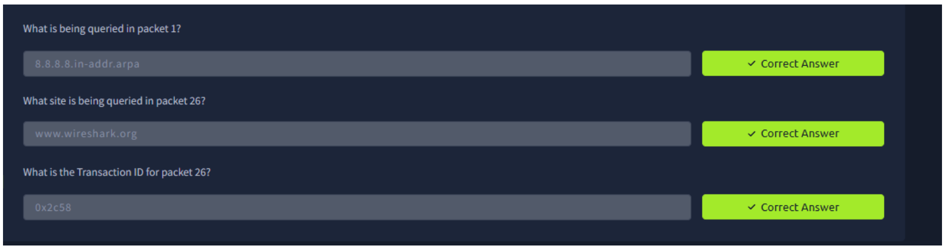
When analyzing TCP, focus on:

Sequence Number

Acknowledgment Number (always 0 in the initial SYN)

Always analyze TCP packets as a sequence, not individually.

Task 10: DNS Traffic



What is being queried in packet 1?

8.8.8.8.in-addr.arpa

What site is being queried in packet 26?

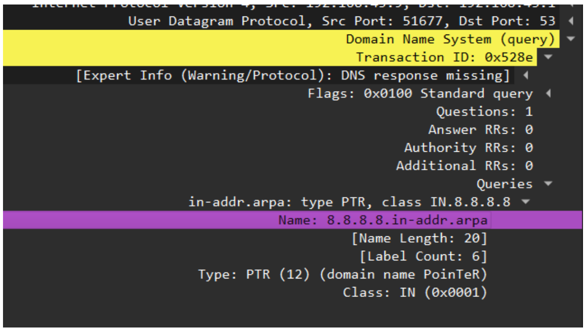
www.wireshark.org

What is the Transaction ID for packet 26?

0x2c58

First question:

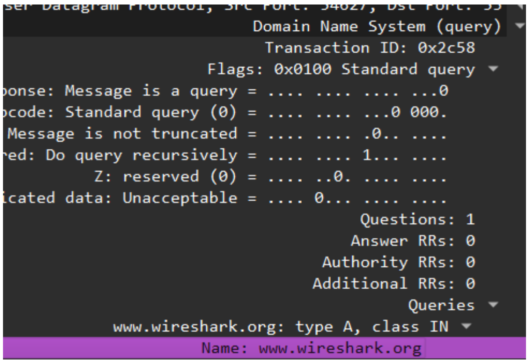
Go to package number 1, then go to section [Expert Info (Warning/Protocol): DNS response missing]. You will then find the query being performed



```
              User Datagram Protocol, Src Port: 51677, Dst Port: 53
                                             Domain Name System (query)
                                             Transaction ID: 0x528e
         [Expert Info (Warning/Protocol): DNS response missing]
                                       Flags: 0x0100 Standard query
                                                       Questions: 1
                                                     Answer RRs: 0
                                                  Authority RRs: 0
                                                 Additional RRs: 0
                                                        Queries
                      in-addr.arpa: type PTR, class IN.8.8.8.8
                                     Name: 8.8.8.8.in-addr.arpa
                                              [Name Length: 20]
                                               [Label Count: 6]
                        Type: PTR (12) (domain name PoinTeR)
                                             Class: IN (0x0001)
```

Second question:

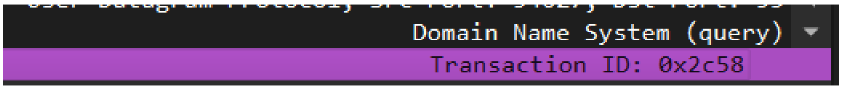
Go to package number 26, then go to the Domain Name System (query) section. You will then find the website being queried



```
                                   Domain Name System (query)
                                       Transaction ID: 0x2c58
                                  Flags: 0x0100 Standard query
ponse: Message is a query = .... .... .... ...0
ocode: Standard query (0) = .... .... ...0 000.
 Message is not truncated = .... .... .0.. ....
red: Do query recursively = .... .... 1... ....
            Z: reserved (0) = .... ..0. .... ....
icated data: Unacceptable = .... 0... .... ....
                                                 Questions: 1
                                                Answer RRs: 0
                                             Authority RRs: 0
                                            Additional RRs: 0
                                                      Queries
           www.wireshark.org: type A, class IN
                       Name: www.wireshark.org
```

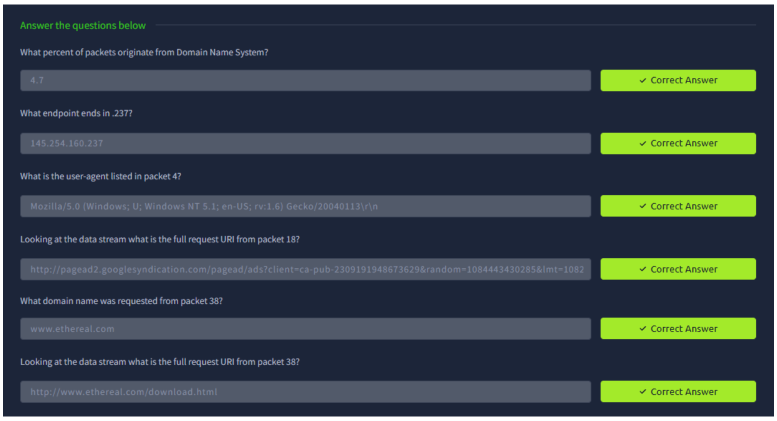
Third question:

Navigate to package number 26, then to the Domain Name System (query) section. You will then find the transaction ID for package 26



```
                     Domain Name System (query)
                        Transaction ID: 0x2c58
```

Task 11: HTTP Traffic



Answer the questions below

What percent of packets originate from Domain Name System?

4.7

✓ Correct Answer

What endpoint ends in .237?

145.254.160.237

✓ Correct Answer

What is the user-agent listed in packet 4?

Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.6) Gecko/20040113\r\n

✓ Correct Answer

Looking at the data stream what is the full request URI from packet 18?

http://pagead2.googlesyndication.com/pagead/ads?client=ca-pub-2309191948673629&random=1084443430285&lmt=1082

✓ Correct Answer

What domain name was requested from packet 38?

www.ethereal.com

✓ Correct Answer

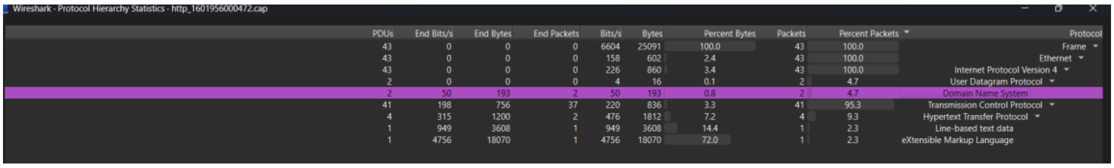
Looking at the data stream what is the full request URI from packet 38?

http://www.ethereal.com/download.html

✓ Correct Answer

First question:

From the top menu, go to Statistics, then Protocol Hierarchy. You will find the percentage of packets originating from the Domain Name System

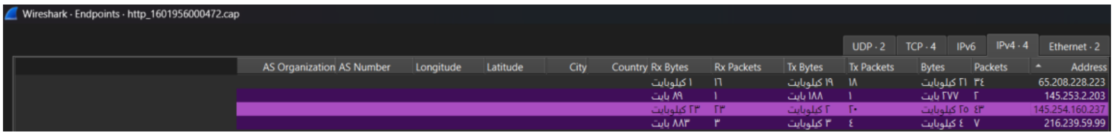


Wireshark · Protocol Hierarchy Statistics · http_1601956000472.cap

PDUs	End Bits/s	End Bytes	End Packets	Bits/s	Bytes	Percent Bytes	Packets	Percent Packets	Protocol
43	0	0	0	6604	25091	100.0	43	100.0	Frame
43	0	0	0	158	602	2.4	43	100.0	Ethernet
43	0	0	0	226	860	3.4	43	100.0	Internet Protocol Version 4
2	0	0	0	4	16	0.1	2	4.7	User Datagram Protocol
2	50	193	2	50	193	0.8	2	4.7	Domain Name System
41	198	756	37	220	836	3.3	41	95.3	Transmission Control Protocol
4	315	1200	2	476	1812	7.2	4	9.3	Hypertext Transfer Protocol
1	949	3608	1	949	3608	14.4	1	2.3	Line-based text data
1	4756	18070	1	4756	18070	72.0	1	2.3	eXtensible Markup Language

Second question:

From the top menu, go to Statistics → Endpoints, select IPv4, and you will find an IP ending in .237



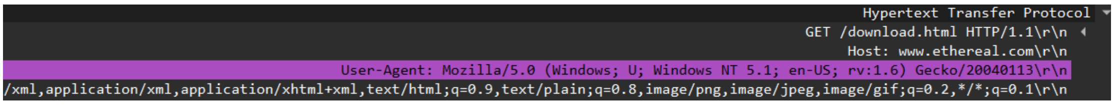
Wireshark · Endpoints · http_1601956000472.cap

AS Organization	AS Number	Longitude	Latitude	City	Country	Rx Bytes	Rx Packets	Tx Bytes	Tx Packets	Bytes	Packets	Address
						١ كيلوبايت	١٦	١٩ كيلوبايت	١٨	٢١ كيلوبايت	٣٤	65.208.228.223
						٨٩ بايت	١	١٨٨ بايت	١	٢٧٧ بايت	٢	145.253.2.203
						٢٣ كيلوبايت	٢٣	٢ كيلوبايت	٢٠	٢٥ كيلوبايت	٤٣	145.254.160.237
						٨٨٣ بايت	٣	٣ كيلوبايت	٤	٤ كيلوبايت	٧	216.239.59.99

Third question:

Select package number 4, then go to Hypertext Transfer Protocol

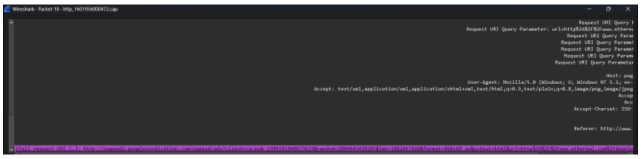
You will then find user-agent



```
                                                              Hypertext Transfer Protocol
                                                        GET /download.html HTTP/1.1\r\n
                                                             Host: www.ethereal.com\r\n
                 User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.6) Gecko/20040113\r\n
/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1\r\n
```

Fourth question:

Go to package number 18 and scroll to the bottom of the page; you will find the full URI for the order

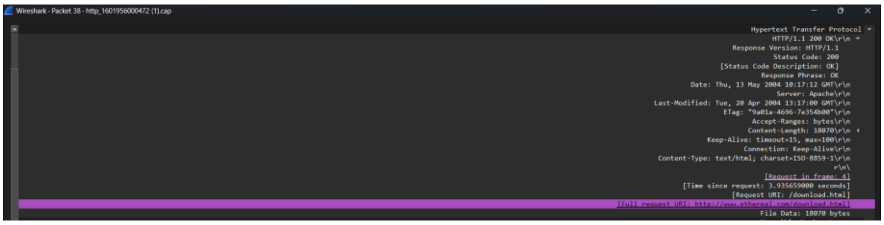


Wireshark · Packet 18 · http_1601956000472.cap

```
                                                                      Request URI Query P
                                        Request URI Query Parameter: url=http%3A%2F%2Fwww.etherea
                                                                    Request URI Query Para
                                                                   Request URI Query Paramet
                                                                   Request URI Query Paramet
                                                                    Request URI Query Param
                                                                  Request URI Query Parameter

                                                                                   Host: pag
                                         User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-
        Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,image/jpeg
                                                                                       Accep
                                                                                         Acc
                                                                        Accept-Charset: ISO-


                                                                        Referer: http://www.

[Full request URI [...]: http://pagead2.googlesyndication.com/pagead/ads?client=ca-pub-2309191948673629&random=1084443430285&lmt=1082467020&format=468x60_as&output=html&url=http%3A%2F%2Fwww.ethereal.com%2Fdownlo
```

Fifth question:

Access package number 38, then Hypertext Transfer Protocol

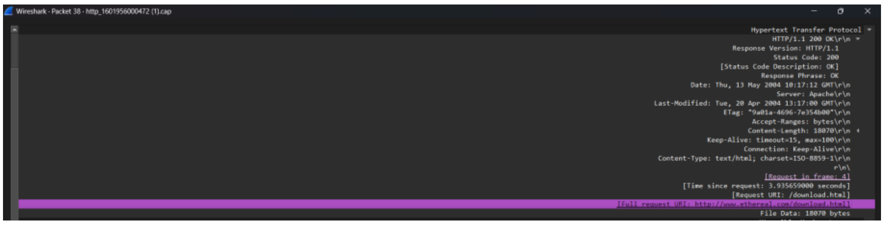


Wireshark - Packet 38 - http_1601956000472 (1).cap

```
                Hypertext Transfer Protocol
                   HTTP/1.1 200 OK\r\n
              Response Version: HTTP/1.1
                       Status Code: 200
            [Status Code Description: OK]
                    Response Phrase: OK
         Date: Thu, 13 May 2004 10:17:12 GMT\r\n
                       Server: Apache\r\n
 Last-Modified: Tue, 20 Apr 2004 13:17:00 GMT\r\n
           ETag: "9a01a-4696-7e354b00"\r\n
                  Accept-Ranges: bytes\r\n
                 Content-Length: 18070\r\n
        Keep-Alive: timeout=15, max=100\r\n
                Connection: Keep-Alive\r\n
  Content-Type: text/html; charset=ISO-8859-1\r\n
                                       r\n\
                     [Request in frame: 4]
      [Time since request: 3.935659000 seconds]
             [Request URI: /download.html]
[Full request URI: http://www.ethereal.com/download.html]
                      File Data: 18070 bytes
```

Sixth question:

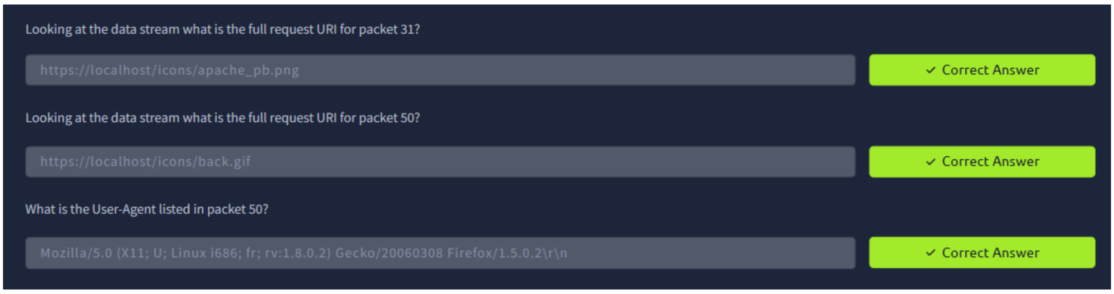
Access package number 38, then Hypertext Transfer Protocol



Wireshark - Packet 38 - http_1601956000472 (1).cap

```
                Hypertext Transfer Protocol
                   HTTP/1.1 200 OK\r\n
              Response Version: HTTP/1.1
                       Status Code: 200
            [Status Code Description: OK]
                    Response Phrase: OK
         Date: Thu, 13 May 2004 10:17:12 GMT\r\n
                       Server: Apache\r\n
 Last-Modified: Tue, 20 Apr 2004 13:17:00 GMT\r\n
           ETag: "9a01a-4696-7e354b00"\r\n
                  Accept-Ranges: bytes\r\n
                 Content-Length: 18070\r\n
        Keep-Alive: timeout=15, max=100\r\n
                Connection: Keep-Alive\r\n
  Content-Type: text/html; charset=ISO-8859-1\r\n
                                       r\n\
                     [Request in frame: 4]
      [Time since request: 3.935659000 seconds]
             [Request URI: /download.html]
[Full request URI: http://www.ethereal.com/download.html]
                      File Data: 18070 bytes
```

Task 12: HTTPS Traffic



Looking at the data stream what is the full request URI for packet 31?

https://localhost/icons/apache_pb.png

Looking at the data stream what is the full request URI for packet 50?

https://localhost/icons/back.gif

What is the User-Agent listed in packet 50?

Mozilla/5.0 (X11; U; Linux i686; fr; rv:1.8.0.2) Gecko/20060308 Firefox/1.5.0.2\r\n

First, we need to decrypt the data using the RSA key within Wireshark

Steps:

Open Wireshark

Go to

Edit > Preferences > Protocols > TLS

Add the key in the RSA section

IP Address: 127.0.0.1

Port: start_tls

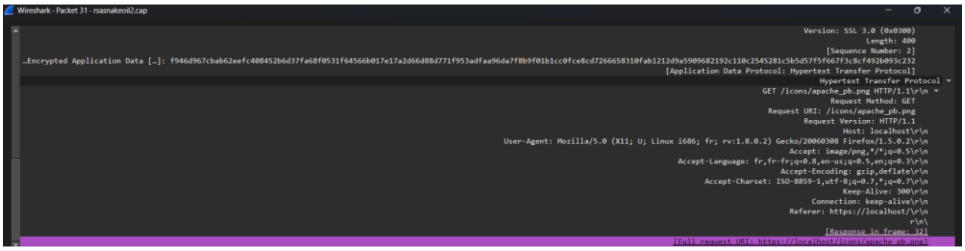
Protocol: http

Keyfile

Enter the path to the RSA file

First question:

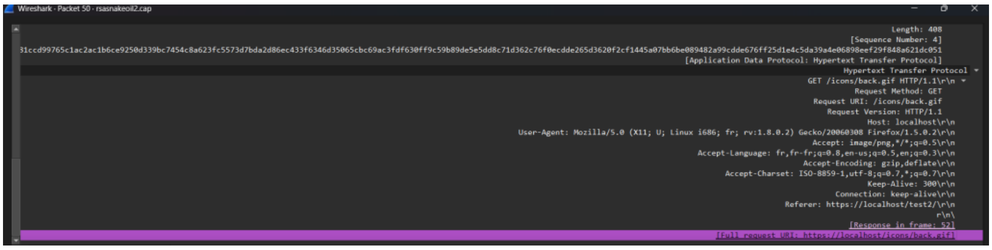
Access package number 31, then access Hypertext Transfer Protocol



```
                                                                 Version: SSL 3.0 (0x0300)
                                                                 Length: 400
                                                                 [Sequence Number: 2]
…Encrypted Application Data […]: f946d967cbab62eefc408452b6d37fa68f0531f64566b017e17a2d66d88d771f953adfaa96da7f8b9f01b1cc0fce8cd7266658310fab1212d9a5909682192c110c2545281c5b5d57f5f667f3c8cf492b093c232
                                                                 [Application Data Protocol: Hypertext Transfer Protocol]
                                                                 Hypertext Transfer Protocol
                                                                 GET /icons/apache_pb.png HTTP/1.1\r\n
                                                                 Request Method: GET
                                                                 Request URI: /icons/apache_pb.png
                                                                 Request Version: HTTP/1.1
                                                                 Host: localhost\r\n
User-Agent: Mozilla/5.0 (X11; U; Linux i686; fr; rv:1.8.0.2) Gecko/20060308 Firefox/1.5.0.2\r\n
                                                                 Accept: image/png,*/*;q=0.5\r\n
                                                                 Accept-Language: fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3\r\n
                                                                 Accept-Encoding: gzip,deflate\r\n
                                                                 Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
                                                                 Keep-Alive: 300\r\n
                                                                 Connection: keep-alive\r\n
                                                                 Referer: https://localhost/\r\n
                                                                 r\n\
                                                                 [Response in frame: 32]
                                                                 [Full request URI: https://localhost/icons/apache_pb.png]
```

Second question:

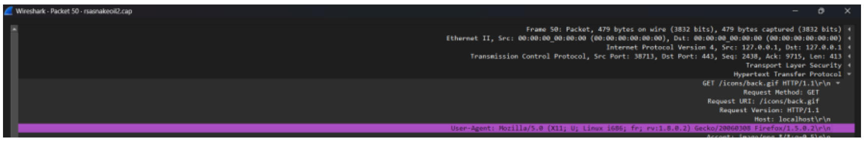
Access package number50, then access Hypertext Transfer Protocol



```
                                                                 Length: 408
                                                                 [Sequence Number: 4]
31ccd99765c1ac2ac1b6ce9250d339bc7454c8a623fc5573d7bda2d86ec433f6346d35065cbc69ac3fdf630ff9c59b89de5e5dd8c71d362c76f0ecdde265d3620f2cf1445a07bb6be089482a99cdde676ff25d1e4c5da39a4e06898eef29f848a621dc051
                                                                 [Application Data Protocol: Hypertext Transfer Protocol]
                                                                 Hypertext Transfer Protocol
                                                                 GET /icons/back.gif HTTP/1.1\r\n
                                                                 Request Method: GET
                                                                 Request URI: /icons/back.gif
                                                                 Request Version: HTTP/1.1
                                                                 Host: localhost\r\n
User-Agent: Mozilla/5.0 (X11; U; Linux i686; fr; rv:1.8.0.2) Gecko/20060308 Firefox/1.5.0.2\r\n
                                                                 Accept: image/png,*/*;q=0.5\r\n
                                                                 Accept-Language: fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3\r\n
                                                                 Accept-Encoding: gzip,deflate\r\n
                                                                 Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
                                                                 Keep-Alive: 300\r\n
                                                                 Connection: keep-alive\r\n
                                                                 Referer: https://localhost/test2/\r\n
                                                                 r\n\
                                                                 [Response in frame: 52]
                                                                 [Full request URI: https://localhost/icons/back.gif]
```

Third question:

Access package number 31, then access Hypertext Transfer Protocol, and you will find User-Agent



```
Frame 50: Packet, 479 bytes on wire (3832 bits), 479 bytes captured (3832 bits)
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 38713, Dst Port: 443, Seq: 2438, Ack: 9715, Len: 413
Transport Layer Security
Hypertext Transfer Protocol
GET /icons/back.gif HTTP/1.1\r\n
Request Method: GET
Request URI: /icons/back.gif
Request Version: HTTP/1.1
Host: localhost\r\n
User-Agent: Mozilla/5.0 (X11; U; Linux i686; fr; rv:1.8.0.2) Gecko/20060308 Firefox/1.5.0.2\r\n
Accept: image/png,*/*;q=0.5\r\n
```

Task 13: Analyzing Exploit PCAPs

Short Summary of the Zerologon Task

The PCAP shows a Zerologon attack against a Domain Controller.

Attacker IP = 192.168.100.128

DC IP = 192.168.100.6

What the PCAP reveals:

Unusual protocols appear (DCERPC, EPM) → start of the attack.

The attacker (192.168.100.128) sends all suspicious requests.

Zerologon behavior is visible through:

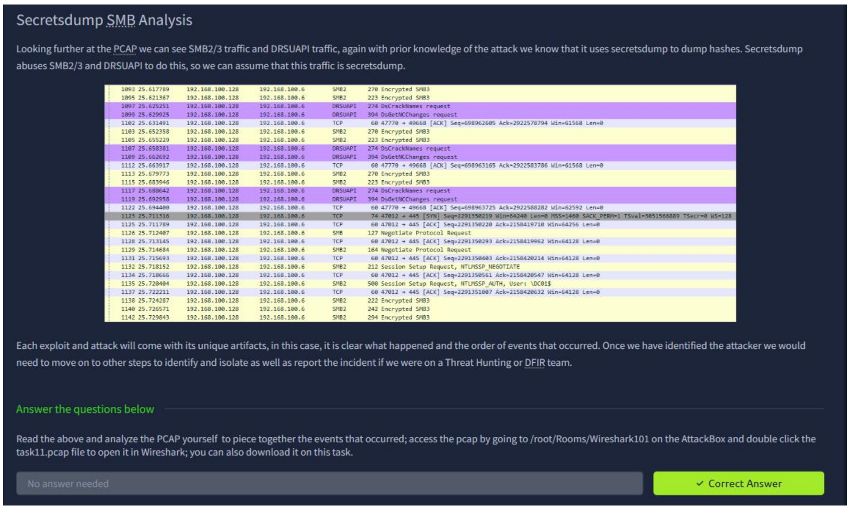
Multiple RPC connections

DCERPC requests to reset the machine account password

Later, SMB2/3 and DRSUAPI traffic appears → indicating secretsdump being used to dump hashes.

Conclusion:

The PCAP clearly shows the attack sequence: Zerologon exploitation → machine password reset → hash dumping.



Secretsdump SMB Analysis

Looking further at the PCAP we can see SMB2/3 traffic and DRSUAPI traffic, again with prior knowledge of the attack we know that it uses secretsdump to dump hashes. Secretsdump abuses SMB2/3 and DRSUAPI to do this, so we can assume that this traffic is secretsdump.

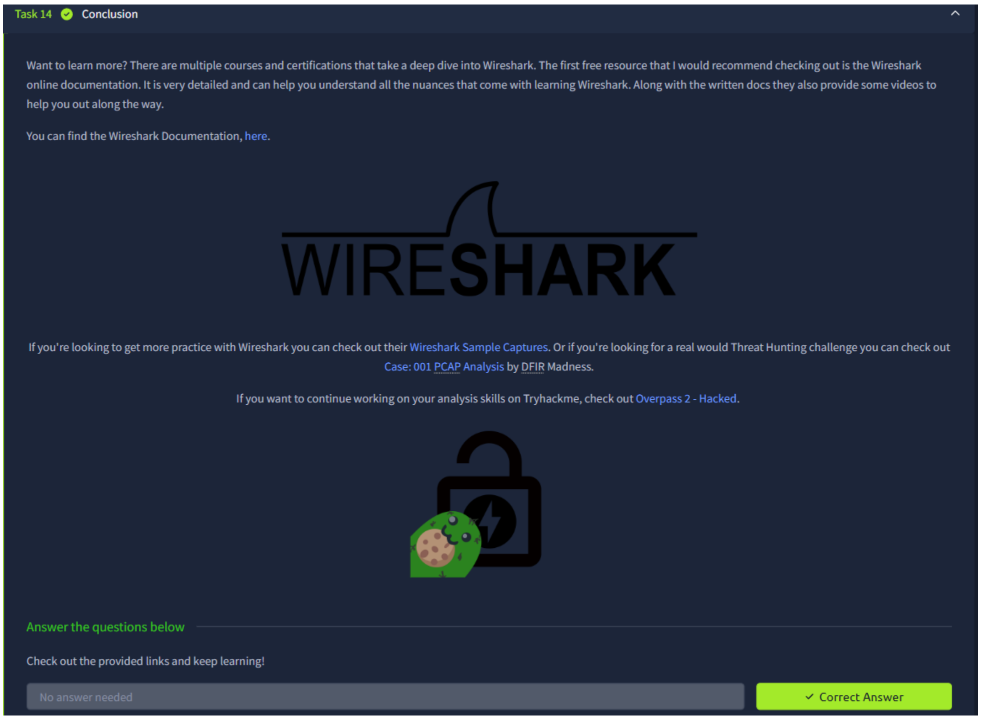
```
1093 25.617789   192.168.100.128   192.168.100.6   SMB2      270 Encrypted SMB3
1095 25.621367   192.168.100.128   192.168.100.6   SMB2      223 Encrypted SMB3
1097 25.625251   192.168.100.128   192.168.100.6   DRSUAPI   274 DsCrackNames request
1099 25.629925   192.168.100.128   192.168.100.6   DRSUAPI   394 DsGetNCChanges request
1102 25.631491   192.168.100.128   192.168.100.6   TCP        60 47770 → 49668 [ACK] Seq=698962605 Ack=2922578794 Win=61568 Len=0
1103 25.652358   192.168.100.128   192.168.100.6   SMB2      270 Encrypted SMB3
1105 25.655229   192.168.100.128   192.168.100.6   SMB2      223 Encrypted SMB3
1107 25.658381   192.168.100.128   192.168.100.6   DRSUAPI   274 DsCrackNames request
1109 25.662692   192.168.100.128   192.168.100.6   DRSUAPI   394 DsGetNCChanges request
1112 25.663917   192.168.100.128   192.168.100.6   TCP        60 47770 → 49668 [ACK] Seq=698963165 Ack=2922583786 Win=61568 Len=0
1113 25.679773   192.168.100.128   192.168.100.6   SMB2      270 Encrypted SMB3
1115 25.683946   192.168.100.128   192.168.100.6   SMB2      223 Encrypted SMB3
1117 25.688642   192.168.100.128   192.168.100.6   DRSUAPI   274 DsCrackNames request
1119 25.692958   192.168.100.128   192.168.100.6   DRSUAPI   394 DsGetNCChanges request
1122 25.694400   192.168.100.128   192.168.100.6   TCP        60 47770 → 49668 [ACK] Seq=698963725 Ack=2922588282 Win=62592 Len=0
1123 25.711316   192.168.100.128   192.168.100.6   TCP        74 47012 → 445 [SYN] Seq=2291350219 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3051566889 TSecr=0 WS=128
1125 25.711789   192.168.100.128   192.168.100.6   TCP        60 47012 → 445 [ACK] Seq=2291350220 Ack=2158419710 Win=64256 Len=0
1126 25.712407   192.168.100.128   192.168.100.6   SMB       127 Negotiate Protocol Request
1128 25.713145   192.168.100.128   192.168.100.6   TCP        60 47012 → 445 [ACK] Seq=2291350293 Ack=2158419962 Win=64128 Len=0
1129 25.714684   192.168.100.128   192.168.100.6   SMB2      164 Negotiate Protocol Request
1131 25.715693   192.168.100.128   192.168.100.6   TCP        60 47012 → 445 [ACK] Seq=2291350403 Ack=2158420214 Win=64128 Len=0
1132 25.718152   192.168.100.128   192.168.100.6   SMB2      212 Session Setup Request, NTLMSSP_NEGOTIATE
1134 25.718666   192.168.100.128   192.168.100.6   TCP        60 47012 → 445 [ACK] Seq=2291350561 Ack=2158420547 Win=64128 Len=0
1135 25.720404   192.168.100.128   192.168.100.6   SMB2      500 Session Setup Request, NTLMSSP_AUTH, User: \DC01$
1137 25.722211   192.168.100.128   192.168.100.6   TCP        60 47012 → 445 [ACK] Seq=2291351007 Ack=2158420632 Win=64128 Len=0
1138 25.724287   192.168.100.128   192.168.100.6   SMB2      222 Encrypted SMB3
1140 25.726571   192.168.100.128   192.168.100.6   SMB2      242 Encrypted SMB3
1142 25.729843   192.168.100.128   192.168.100.6   SMB2      294 Encrypted SMB3
```

Each exploit and attack will come with its unique artifacts, in this case, it is clear what happened and the order of events that occurred. Once we have identified the attacker we would need to move on to other steps to identify and isolate as well as report the incident if we were on a Threat Hunting or DFIR team.

Answer the questions below

Read the above and analyze the PCAP yourself to piece together the events that occurred; access the pcap by going to /root/Rooms/Wireshark101 on the AttackBox and double click the task11.pcap file to open it in Wireshark; you can also download it on this task.

Task 14: Conclusion



Want to learn more? There are multiple courses and certifications that take a deep dive into Wireshark. The first free resource that I would recommend checking out is the Wireshark online documentation. It is very detailed and can help you understand all the nuances that come with learning Wireshark. Along with the written docs they also provide some videos to help you out along the way.

You can find the Wireshark Documentation, here.

If you're looking to get more practice with Wireshark you can check out their Wireshark Sample Captures. Or if you're looking for a real would Threat Hunting challenge you can check out Case: 001 PCAP Analysis by DFIR Madness.

If you want to continue working on your analysis skills on Tryhackme, check out Overpass 2 - Hacked.

Answer the questions below

Check out the provided links and keep learning!