Capture – Filter – Dissect

Network Protocol Analysis with Wireshark

Jon Meyer, Jared Zook and Ananth Jillepalli

July 13th, 2016
Version 2.1

University of Idaho

CS 539: Applied Security Concepts

Summary

Wireshark is a tool for capturing, analyzing and dissecting network traffic. Originally named Ethereal, Wireshark is built upon the `libpcap` library or equivalent libraries on certain non-Unix based systems. This tutorial provides an overview of Wireshark and an explanation of how it is a powerful tool for network traffic analysis. The process of packet capturing, filtering, and analysis is laid out in a series of demonstrations and walkthroughs. Three challenge questions at the end provide an opportunity to explore Wireshark's functionality on an individual basis.

Contents

1 Objectives of this Tutorial 1

2 Required Background 2

3 Hardware and Software Requirements 3

4 Wireshark Overview 4

5 Problems with Alternatives 5

6 Importance of Wireshark 6

7 Related Recent News: Wireshark 7

8 Questions: Related News 8

9 Wireshark Background 9

10 Intro. to Wireshark: Capture Filtering 10

11 Digression: Well Known Ports 11

12 Activity: Display Filters 12

13 Question: Display Filters 13

14 Observations: Packet & Protocol Analysis 14

15 Observations: Packet Details Pane I 15

16 Observations: Packet Details Pane II 16

17 Observations: Packet Bytes Pane 17

18 Saving Display Filtered Captures 18

19 Session Analysis: Establishing a Socket 19

20 Session Analysis: Establishing the FTP Session 20

21 Questions: Establishing the FTP Session 21

22 Session Analysis: Establishing the FTP Session 22

23 Challenge I 23

24 Challenge II 24

25 Challenge III **25**

26 Conclusion **26**

27 Appendix: Solutions and Change-log **27**

1 Objectives of this Tutorial >

1. Understand a little bit of the history and capabilities of Wireshark.
2. Understand how to perform basic packet capture operations, including selecting capture interfaces and setting up basic capture filters.
3. Understand the basics of analyzing captures, including packet dissection and utilizing display filters.
4. Understand how to save selected portions of capture files as new files.

This tutorial is not a complete user's guide to Wireshark. In contrast, this tutorial covers the basics of Wireshark. In the ensuing tutorial, we will briefly explain, through either activities or challenges, the following specific things about Wireshark:

1. Wireshark, as we know it today, is a product of a serial transformations from one stage to the next stage. To truly understand the capabilities of Wireshark, we must also look at a bit of its' history/background so that we can better get to know the ideology behind Wireshark's development. Wireshark's capabilities are not just restricted to packet analysis. It is a complete suit with host of other functionalities.
2. Fundamentals of Wireshark's packet capture operation works, including selection of capture interfaces and setting up basic capture filters are all part of the content in this tutorial, demonstrated and imparted through activities and challenges.
3. Wireshark's packet capture, capturing interfaces, and filters are complemented by the analysis mechanism, which includes packet dissection, and using display filters to look with the aid of filtered packets. Analysis of packet captures is an integral step in Network Forensic analysis and Wireshark provides a great platform to carry out such an analysis.
4. The popularity of Wireshark and its' distinguished fame comes from its' versatility and flexibility. Wireshark's arsenal of functions are not just limited to allowing packet captures, filters, and filtered analysis. In addition, Wireshark can also save select and filtered portions of a capture file as completely new files.

2 Required Background <>

We assume that the reader of this tutorial has an extent of background knowledge in the following areas:

1. Working experience on usage of computers and software applications, like web browsers, and virtualization apps.
2. Basic overall idea of computer networks and Internet.
3. Fundamentals of networking mechanisms like packet streaming, capturing, etc.,
4. An overall idea on general issues like data privacy, computer security, etc.,

Due to restrictions on time and manpower resources, we are not able to make the ensuing tutorial to be completely self-contained from the perspective of a user. As such, the tutorial is best used when the user already has certain background skills and knowledge. The following are some areas where we expect the users of this tutorial to have some previous skills/knowledge:

1. Practical experience on using computers, installing and using common software applications (particularly web browsers, and virtualization software platforms). The tutorial does not explain how to navigate within the operating system's graphical user interface (GUI). Similarly, the tutorial also does not explain how to browse the Internet, how to install software applications, and how to use/navigate software applications.
2. An overall idea on the working os computer networks and Internet. The tutorial expects a user to understand common computer networking terms and their technical meanings. For example, "packets", "streams", "protocols", and "capture / filter" etc.,
3. Fundamental or very basic idea on networking mechanisms and computer networks. The tutorial expects a user to understand technical concepts like OSI model of networks, basic network attacks like man-in-the-middle attack etc.,
4. Also bit of exposure to logic notations would be very helpful in understanding the filter strings. An overall brief idea on general computer-related issues like "data privacy", "computer security", "network security", "network protocol encapsulation" etc., will help a lot.

3 Hardware and Software Requirements <>

We recommend having at least the following hardware and software specifications for smooth execution of this tutorial's activities and challenges:

1. A computer which can at least boot 1 virtual machine (VM) smoothly, with no noticeable lag and delay
2. A functional virtualization software platform. For example, VMWare or VirtualBox.
3. Any VM with Wireshark installed and efficiently functioning.

For the purpose of getting the best experience out of this tutorial, there are certain minimum hardware and software requirements that we recommend users have at their disposal. However, this tutorial can also be carried out in lower specifications than what is recommended.

1. A computer powerful enough to be able to boot 1 virtual machine without any noticeable delay or log. That would mean at least a quad-core processor, 8 GB RAM, optimally two monitors (can be managed with one), and a functional keyboard and mouse.
2. It is always recommended to use a virtual machine to run tutorials like the one available in this document so as to not destabilize one's own workstation or personal machine's environment and software configurations. To that extent, we recommend having a virtualization software installed on machine, which can be used to generate virtual machines as needed.
3. The present tutorial can be followed with any operating system which supports efficient functioning of Wireshark software application. If the user choses to be consistent with the tutorial, we will be using Ubuntu 16.04 LTS VM.

   Note: Please go to part **2** of the **Appendix** section for information on resources and website links to access some of the required software for the tutorial.

4 Wireshark Overview <>

1. Wireshark:
   (a) Free, open-source network protocol analyzer.
   (b) Supports the decoding / dissecting of hundreds of packet types.
   (c) Can generally access any data available to the capturing system's network interfaces.
   (d) Is able to capture data from suitably configured remote systems.

1. Wireshark [1] is an open source network protocol analyzer. A network protocol analyzer is a tool that collects network packet traffic. If successful, it allows users to interactively analyze detailed reports of the data found within.

2. Wireshark gives the user, power to examine what activities are being carried out on his or her network. This is useful for many administrative tasks, including providing information for network security analysts. [1]

3. Wireshark is free to use under the GNU General Public License version 2 and is available for all mainstream operating systems. It is capable of capturing live data on a wide variety of interfaces including Ethernet topologies, Bluetooth traffic, and Token Ring switches.

4. Captured data may be filtered according to user-defined criteria. After capturing and filtering, users will have extremely detailed protocol analysis at their disposal. [1]

5 Problems with Alternatives <>

1. Prior to Wireshark, there were special purpose "protocol analyzers" which were
   (a) large and heavy (think small to medium sized suitcases filled with books – "luggable" rather than "portable")
   (b) of limited capacity
   (c) expensive
2. Other tools (e.g., tcpdump) existed but typically offered limited protocol analysis and had user interfaces best suited for "expert" users

1. Prior to Wireshark, diagnosing network problems often required special purpose "protocol analyzers".
   (a) These analyzers were large and heavy (think small-to-medium sized suitcases filled with books). They were "luggable" rather than "portable".
   (b) They had a limited capacity, typically being able to only capture a couple of megabytes (or less) of traffic and only understood a few protocols.
   (c) In addition, they were often quite expensive – "cheap" analyzers often cost more than $10,000.
2. Alternative tools, like `tcpdump` [2] existed but offered limited protocol analysis and their user interfaces was best suited for advanced/expert users, making it difficult for novice/beginner users to use such alternative tools.

6 Importance of Wireshark <>

1. The ability to capture and analyze network packets is important to facilitate:
   (a) Diagnosis of network connectivity issues.
   (b) Verification of correct implementation of communications and security protocols.
   (c) Analysis and mitigation of attacks etc.,

1. Advanced abilities of capturing and analyzing network packets are important, because they facilitate:.
   (a) Diagnosis of network connectivity issues. In case a network is facing problems which are not obviously visible (like cable unplugged/damaged), then network capture analysis can reveal additional helpful details.
   (b) Verification and validation of communication and security protocols' correct implementation. Wireshark's detailed analysis helps in the process evaluation.
   (c) Only a rather small portion of network attacks are diagnosable through outer-most analysis. Many network attacks are only detected by deep-packet data analysis. Wireshark not only provides the deep-level sophistication in analyses, but also helpful insights as on how we can mitigate an attack in future.

7 Related Recent News: Wireshark <>

1. Packet sniffing is instrumental to carry out (and detect) NSA malware [4] [April 2015]
2. Packet sniffing of mobile phone location data puts users in a precarious position [3] [November 2014]
3. Use Wireshark to secure your home network [5] [October 2014]

1. A National Security Agency (NSA) attack called "Quantum Insert" was a "man-on-the-side" attack that allowed the agency to silently attach malware to 300 target machines in 2010. To accomplish these injections, the agency used packet sniffing to detect the browsing footprint of their target (e.g. cookies indicating sites that the target visits often). Once HTTP GET requests were sniffed for the common site, high-speed servers placed close to target machine redirected the browser to the malicious site by spoofing a TCP packet. Dutch IT security firm Fox-IT was able to use packet sniffing to detect the attack when simulating it in their own environment. They found that the spoofed TCP packets contained the same sequence number as the legitimate packets that were dropped [4].

2. Smart phones that enable location services and Wi-Fi for location accuracy send out revealing data to nearby packet sniffers. To understand the extent of the data revealed, Ars used Wireshark to passively listen in on Wi-Fi traffic generated by some volunteered smart phones. They filtered the packets they intercepted down to "probe" requests that included the device's MAC address and various SSID names of networks the phones were looking for. By mapping the SSIDs against publicly-accessible, geo-tagged locations of nearby Wi-Fi hotspots, Ars found information about networks used at the users' homes, workplaces, and even travel locations [3].

3. Experts at Lifehacker note that, in addition to sniffing out passwords and cookies, and being considered malicious in general, Wireshark can be used to simply monitor traffic on a network. When using the tool, they recommended operating under "promiscuous mode" to collect all packets traversing the network wirelessly. If any of the packets indicated strange activity, users were directed to use further tools to determine the hostname of the suspected IP address [5].

8 Questions: Related News <>

Q1: Why did the NSA's Quantum Insert require such fast servers to complete its attack? How were researchers able to detect Quantum Insert?

Q2: Can "listening in" on mobile phone location data reveal information about places people have been outside of the geographic region of the sniffing?

Q3: What packets may be collected when Wireshark is NOT in "promiscuous mode?"

A1: Quantum Insert required fast server in close proximity to the target machine so it could race in front of the legitimate site's TCP packet in order to re-direct browsing to the malicious site. Researchers detected Quantum Insert by finding two TCP packets in a row with the same sequence number.

A2: Yes. For example, a user's phone may have an SSID saved called "Ritz Carlton – Honolulu, Hawaii".

A3: Promiscuous mode enables Wireshark to sniff out wireless traffic. When it's turned off, only traffic on the wired network can be captured.

9 Wireshark Background <>

1. Predecessor: tcpdump (WinDump on Windows)
2. Utilizes capabilities of the libpcap library (WinPcap on Windows)
3. Wireshark adds a GUI, the ability to filter displayed data post-capture, and packet decoding for hundreds of protocols and packet types

1. The tcpdump tool was developed in the late 1980s, originally for BSD Unix systems to facilitate capturing and dumping of network traffic to either the command line or a file. It also has the ability to display the contents of a previously-captured file. [2]
2. The libpcap (WinPcap) library was originally developed by Lawrence Berkeley's Network Research Group. It supplies a common API and lower lever functionality, such as capture filtering which is utilized by application packages such as tcpdump and Wireshark to reduce the volume of data captured by the scan [6]. This library is important not just because of its use by these programs but its availability for other programs that capture and potentially manipulate network traffic (e.g., Scapy).
3. Wireshark was originally developed as "Ethereal" in late 1990's by Gerald Combs, a University of Michigan graduate student. In 2006, the name was changed to Wireshark due to trademark issues.

10 Intro. to Wireshark: Capture Filtering <>

1. Capturing all of the traffic on one or more interfaces can generate an overwhelming amount of data.
2. Capture filters allow us to be more selective about the data we capture.
3. Capture filters are "compiled" and registered with the operating system's network stack. Capture filtering typically takes place at this level, not in libpcap or in Wireshark itself.

Wireshark is a tool that facilitates capturing and analyzing network traffic. It has the capacity to analyze hundreds of different communication protocols and dissects network packets according to network levels. It is capable of capturing traffic on one or more network interfaces, presenting the packets as received along with timestamp information.

1. The most recent official Wireshark release, Version 2.0.4 (as of 6th July 2016), contains a graphical representation of the traffic on each of the system's network interfaces. Modern versions of Wireshark are also capable of capturing USB traffic.
2. Note that in the capture from my home network, which is not particularly busy, there are sixteen hundred packets per second being captured. As I said, this sort of traffic volume is not even particularly high. You can see the need for capture filters.
3. In the latest Wireshark, you can type a capture filter right on the opening page. There is also an option to enter it under the "Capture" menu we'll see in a minute. When you type a capture filter, the entry bar will be pinkish red unless you've typed a valid filter. When the filter is valid, the line will have a pale green background.

   One simple filter is just the word 'host' followed by an IP address. This will capture all packets transmitted to or received from a host with that IP address. Using such a simple filter on my network reduced the rate of packet capture to less than 60 per second in this case.

   Capture Filters [9] are based on the syntax of libpcap filters. (See [8]).

11 Digression: Well Known Ports <>

1. TCP and UDP ports are a common component of Wireshark filters, both for display and capture.
2. If one is trying to capture traffic associated with a particular network service, there may well be a standard port associated with that service.
3. A few well-known ports are:
   (a) FTP ports 20 and 21, Telnet port 23
   (b) SMTP port 25, POP3 port 110
   (c) HTTP port 80, HTTPS port 443

Interesting to know:

The Internet Assigned Numbers Authority (IANA) controls the assignment of "well-known ports", also known as *Standard Ports*. These ports range from port 0 to port 1023. In addition, there are a number of "registered" ports that are commonly used for particular purposes, but not reserved in the same way as "well-known ports."

Being familiar with commonly used ports can be useful when trying to construct filters.

12 Activity: Display Filters <>

1. Besides filtering network packets at capture time, Wireshark provides the capability to filter the packets it displays.
2. The syntax for capture and display filters is *different*:
   (a) Capture filter syntax is the syntax used by libpcap.
   (b) Display filter syntax uses 'C'-like logical syntax. It is much more "programmer" friendly.
3. We can easily build display filters "on the fly."

To witness the display filtering capacity of Wireshark, open the file titled **Wireshark1.pcapng** from the folder "Captures" given in the archive of this tutorial. This capture file has almost twenty-five hundred packets. Let's poke around to see if we can see anything interesting.

i One way to filter is on the threshold of "protocol." For the most part, what we mean by "protocol" in this case is the application level service, although we can be talking about something lower level (e.g., TCP or UDP).

ii In this case, we see there are a lot of packets with a protocol of *synergy*. This is an application that allows multiple computers to share a keyboard and mouse. We're not interested in these packets right now, so let's hide them. Type 'not synergy' (without the quotes) into the display filter line.

iii Notice that the filter line goes pale red until you have typed a valid filter. Now click the arrow to the right of the filter line. Notice how all the packets with that protocol have disappeared.

iv There still seems to be a lot of packets we're not interested in, so what else can we get rid of? There are a lot of packets coming with a TCP protocol, but trust me, you don't want to use a "not tcp" filter. That would get rid of a lot of packets we want to keep.

v There seem to be quite a few packets coming from or going to port 24800. Let's assume we are not interested in those for the moment. Let's add "and not tcp.port == 24800" to our display filter. Click the arrow to apply our filter.

13 Question: Display Filters <>

Q4: How would you re-write our display filter in a compilable 'C' logical statement, using a logical disjunction instead of conjunction? You can assume the appropriate variable definitions exist. Try your new filter. It should give the same results as our original.

Reminder:

We don't want any synergy packets or packets from TCP port 24800.

Answer to Q4:
!(synergy || (tcp.port == 24800))

14 Observations: Packet & Protocol Analysis <>

1. The Display window has three main panes:
   (a) Packet List
   (b) Packet Details
   (c) Packet Bytes
   (d) Other details

1. The Packet List pane shows the list of packets from the current capture file that are not currently filtered out. The columns for this pane are configurable, but by default they are:

   (a) Packet Num – The number of the packet relative to the start of the capture.
   (b) Timestamp – The time at which, the packet was received by the network stack. There are a number of useful display options selected by the View/Time Display Format menu. Among the most useful are:
      i. Seconds since 01 Jan 1971 (i.e., the Unix 'epoch').
      ii. Seconds since the previous captured packet.
      iii. Seconds since the previous displayed packet.
   (c) Source Address – who sent the packet
   (d) Destination Address – who is the intended recipient of the packet.
   (e) Protocol – Normally the highest layer protocol Wireshark is able to identify. This may well be several sub-layers into the application level.
   (f) Length – the number of bytes in the packet (or frame).
   (g) Packet specific info – Varies by type of packet. Usually a general description of the packet type along with basic details from the frame contents.

15 Observations: Packet Details Pane I <>

1. Displays a layer-by-layer breakdown of the contents of the packet currently selected in the packet list.
2. Layers displayed include:
   (a) Frame – gives basic information about the frame, including its length and the network interface from which it was captured(*).
   (b) Data Link – often Ethernet or Ethernet II(**).

(*) Expanding the Frame layer also give information about the timestamps, encapsulation, coloring rules, and incorporated protocols.

(**) The Data Link layer includes source and destination MAC addresses. Expanding it gives some information about the natures of the addresses.

16 Observations: Packet Details Pane II <>

1. Other layers displayed may also include:
   (a) Internet Protocol (IP) Layer decoding
   (b) User Datagram Protocol (UDP) Layer decoding
   (c) Transmission Control Protocol (TCP) Layer decoding
   (d) Application specific layers (e.g. HTTP, DNS, FTP)
   (e) Data – a block of data that Wireshark recognizes as legitimate payload for a higher layer, but does not know how to decode / dissect.

Important Information:

1. Wireshark will go as far as it can to dissect and decode contents of captured packets. It has many fairly complex rules for guessing how to interpret the contents of a layer (e.g., the port to which it was addressed). Its guesses are very good, but sometimes you have to correct it or guide it.

2. You can do this by selecting the packet or layer of interest, right clicking on it and choosing "Decode As" You can select how you want decoding to proceed from there.

17 Observations: Packet Bytes Pane <>

1. Displays a hex dump with a right hand column ASCII interpretation of the characters of the packet.
2. Selecting a layer in the Packet details panel highlights the bytes corresponding to that layer in the Packet Bytes Panel.

18 Saving Display Filtered Captures <>

1. Construct and apply the filter you want.
2. Click "File" and then "Export Specified Packets."
3. Type in your desired filename omitting a file extension.
4. Click "Save."

Details:

1. The capture file we have been using has a lot of different message traffic in it, including traffic related to a File Transfer Protocol (FTP) session. Let's create a file with only the traffic related to this session.
2. It would be tempting to simply type "ftp" into the display filter. In some circumstances, that might give us exactly what we want, but in this case we want to save traffic associated with establishing and closing the socket as well, so we need more than that.
3. This is where understanding "well-known ports" is useful. Since we know FTP normally uses ports 20 and 21, we can use a filter like tcp.port == 20 || tcp.port == 21. Let's apply that filter and then **export** the packets to **TutorialTFP** (Wireshark will add a ".pcapng" suffix).
4. For more information on Display filter syntax see the Wireshark article on display filters. [10]

19 Session Analysis: Establishing a Socket <>

1. Session begins with establishing a socket:
   (a) Client sends a "SYN" packet to the correct server address and port.
   (b) Server responds with a "SYN, ACK" packet to the client's address and port from the "SYN" packet.
   (c) Client responds with an "ACK" which completes establishing the socket.

Open the file **TutorialFTP.pcapng**, which was created by Wireshark from the previous slide.

a The exchange of "SYN" messages allows the client and server to negotiate the parameters of the TCP session.

b The client proposes a set of parameters in its "SYN." The server either accepts them by echoing them back in the "SYN, ACK" packet or proposes alternative parameters.

c The client then either accepts the parameters specified by the server or session establishment fails.

20 Session Analysis: Establishing the FTP Session<>

1. FTP Session establishment:
   (a) Server sends an FTP response identifying itself.
   (b) Client sends a "USER" message.
   (c) If nec., the server sends "Password required" response
   (d) If a password was requested, the client sends a PASS request along with the user's password.
   (e) Server either sends a "User logged in" response or a "User cannot log in" response.

Some Pointers:

1. A search can be done for packets containing a desired text string typing Ctrl+F. Change "Packet list" to "Packet details" and "Display Filter" to "String" and type in the desired search string.
2. Clicking "Find" searches for the next occurrence of the string. The search will wrap at the end of the capture.
3. Let's search for the user's password using the "PASS" string.

21 Questions: Establishing the FTP Session <>

Q5: In the capture we see a password requested for user "anonymous" even though that is not an authorized user. Why ask for a password instead of rejecting the login at that point?

Q6: The preceding page demonstrated a significant vulnerability when using protocols such as FTP on an unencrypted connection. What was it?

Q5: If we rejected the login before requesting the password we would be providing a potential attacker with a way to test whether or not guessed user names were valid.

Q6: Since user credentials are exchanged in plain text, they are vulnerable to exposure using sniffers like Wireshark.

22 Session Analysis: Establishing the FTP Session<>

1. At any point the client may issue a "QUIT" request.
2. In response to a "QUIT" or on its own, the server can send a "Goodbye" response, terminating the FTP session.
3. To cleanly terminate the socket the client and server exchange "FIN, ACK" messages. Either side can initiate this.
4. **However:** At any time, either side can *force* the termination of the socket by sending an "RST"(*) or "RST, ACK" message.

* "RST" messages are most often seen when the application connected to the socket is terminated. The system's network stack realizes that there is nothing to handle for a clean shutdown so it forces a shutdown on its own initiative.

23 Challenge I <>

Capture files for these challenges were downloaded from [11]

1. Open capture file telnet-cooked.pcap
2. What are the user's login credentials?

Hint: The telnet server prompts for the user's ID with the string 'login'

24 Challenge II <>

Capture files for these challenges from the sample caps wiki [11]

1. Open capture file dns.cap
2. What are the names of the mail servers associated with the domain google.com?
3. What are IP addresses associated with these names?

Hint: DNS records identifying mail servers are "MX" records. Addresses records for a particular host name are "A" records. Sometimes a helpful name server will return address information as additional data in a response to a request for mail server information.

25 Challenge III <>

Capture files for these challenges are from the same wiki [11]

1. Open capture file http.cap and filter to show only those packets representing web traffic.
2. How many such packets are there?
3. What is the destination port on the first hyper text response.

26 Conclusion <>

1. Wireshark is a powerful tool for capturing and analyzing network traffic.
2. Wireshark can assist with diagnosing and repairing network issues, including network security vulnerabilities. It can also assist those intending to exploit these vulnerabilities.
3. Like most tools, Wireshark is not inherently morally good or bad. It is the user's choice of how to make use of the tool that leads to good or bad.

A final thought:
Wireshark can capture any traffic seen by the network interfaces it accesses. A corollary to that is that it cannot capture traffic not seen by those interfaces. For instance, if your network interface is connected to a switch that only sends traffic for your interface's MAC address to that interface, you'll never see traffic meant for other parts of your network.

27 Appendix: Solutions and Change-log <

1. Solutions to the challenges
   (a) Challenge I
   (b) Challenge II
   (c) Challenge III
2. Tutorial-Related Resources
3. Change-Log

1. **Solutions to the Challenges:**

   (a) **Challenge I:**
   Open the file 'telnet-cooked.pcap' (available in `Captures` folder).
      i. Press "Ctrl + F" key combination.
      ii. In the newly appearing tool-bar, change the left-most drop-down selection to "Packet details".
      iii. Change the right most drop-down selection to "String" value.
      iv. You will taken to packet number 29, which prompts the user for a 'login' name (observable from *Data* sub-section in 'Telnet' section of packet details pane).
      v. In the subsequent series of packets (31 and 38 packet numbers), the reader can observe in the packets' telnet-data sub-section that login and password credentials are "*fake*" (login) and "*user*" (password) respectively.

   (b) **Challenge II:**
   Open the file 'dns.cap' (available in `Captures` folder).
      i. Scan the "Info" coloumn in packet list pane.
      ii. At packet number 3, the "Info" coloumn will show "`MX google.com`" string.
      iii. To that series of queries, response is given in the subsequent packet (packet number 4). In packet details pane, the reader can observe `Additional Records` by expanding `Domain Name System (response)` section in packet details pane.

iv. The `Additional Records` sub-section provides the names of mail servers associated with domain google.com, along with the associated IP-address of mail servers. The names and IP-addresses are:

A. Name: `smtp4.google.com`; Address: 216.239.37.26
B. Name: `smtp4.google.com`; Address: 64.233.167.25
C. Name: `smtp4.google.com`; Address: 66.102.9.25
D. Name: `smtp4.google.com`; Address: 216.239.57.25
E. Name: `smtp4.google.com`; Address: 216.239.37.25
F. Name: `smtp4.google.com`; Address: 216.239.57.26

(c) **Challenge III:**
Open the file 'http.cap' (available in `Captures` folder).

i. Type in '`http`' into the *Apply a display filter* field.
ii. The resultant four packets represent web traffic(packet numbers 4, 18, 27, and 38).
iii. The destination port for response of first hypertext request is available in packet number 5. The destination port is `3372`.

2. **Tutorial-Related Resources:**

   A free virtualization software platform - VirtualBox - CLICK ME

   Wireshark application - Wireshark Foundation - CLICK ME

   A Windows 10 VM with Microsoft Edge - CLICK ME

3. **Change-Log:**

Network Protocol Analysis with Wireshark tutorial			
Ver.	**Date**	**Authors**	**Changes**
v1	Feb. 1st 2016	Jon Meyer and Jared Zook	First draft of tutorial.
v2	July 6th 2016	Ananth Jillepalli	Major content additions and remodeled the structure.
v2.1	July 13th 2016	Ananth Jillepalli	Added the 'Appendix' section and other minor enhancements.

References

[1] The Wireshark Foundation, “Wireshark", http://www.wireshark.org, page accessed January 2016.

[2] Wikipedia, “tcpdump", http://en.wikipedia.org/wiki/Tcpdump, page accessed January 2016.

[3] Sean Gallagher, “Where’ve you been? Your Smartphone’s Wi-Fi is telling everyone", http://arstechnica.com/information-technology/2014/11/where-have-you-been-your-smartphones-wi-fi-is-telling-everyone/, page accessed January 2016.

[4] WIRED, “How to detect sneaky NSA Quantum Insert Attacks", https://www.wired.com/2015/04/researchers-uncover-method-detect-nsa-quantum-insert-hacks/, page accessed January 2016.

[5] Alan Henry, “How to tap your network and see everything that happens on it", http://lifehacker.com/how-to-tap-your-network-and-see-everything-that-happens-1649292940, page accessed January 2016.

[6] Wikipedia, “pcap", http://en.wikipedia.org/wiki/Pcap, page accessed January 2016.

[7] Wikipedia, “List of TCP and UDP port numbers", https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers, page accessed January 2016.

[8] tcpdump.org, “Example libpcap Filters", http://www.tcpdump.org/manpages/pcap-filter.7.html, page accessed January 2016.

[9] Wireshark.org, “Example Capture Filters", https://wiki.wireshark.org/CaptureFilters, page accessed January 2016.

[10] Wireshark.org, “Example Wireshark display filters", https://wiki.wireshark.org/DisplayFilters, page accessed January 2016.

[11] Wireshark.org, “Sample capture files", https://wiki.wireshark.org/SampleCaptures, page accessed January 2016.