Capture – Filter – Dissect

Network Protocol Analysis with Wireshark

Created by: Jon Meyer and Jared Zook
Modified by: Ananth Jillepalli, Robert Breckenridge, and Matthew Holman

July 1st, 2017
Version 3.2

University of Idaho

CS 439/539: 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. To understand a bit of the history and capabilities of Wireshark;
2. To learn how to perform basic packet capture operations, including selecting capture interfaces and setting up basic capture filters;
3. To understand the basics of analyzing captures, including packet dissection and utilizing display filters;
4. To learn how to save selected portions of capture files as new files.

This tutorial is not a complete user's guide to Wireshark. Rather, this tutorial covers the basics of Wireshark. In the following lesson, we will briefly explain, through activities and challenges, the following things about Wireshark:

1. Wireshark, as we know it today, is a product of a transformations from one stage to the next. To understand the capabilities of Wireshark, we must look at its history and background. This is so we can understand 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. Learning the fundamentals of how Wireshark's packet capture operation works, including the selection of capture interfaces and setting up basic capture filters, are all part of this tutorial.

3. Wireshark's packet capture, capturing interfaces, and filters, are complemented by the analysis mechanism. This mechanism includes packet dissection and using display filters to look at filtered packets. Analysis of packet captures is an integral step in Network Forensic analysis. Wireshark provides a great platform to carry out such an analysis.

4. The popularity of Wireshark comes from its versatility and flexibility. Wireshark's arsenal of functions is not just limited to allowing packet captures, filters, and filtered analysis. Additionally, Wireshark can also save select and filtered portions of a capture file as completely new files.

2 Required Background

<>

We assume that the reader has background knowledge in the following areas:

1. A working experience using computers and software applications (i.e. web browsers, and virtualization apps);
2. A basic idea of computer networks and Internet functionality;
3. Knowledge of the fundamentals of networking mechanisms like packet streaming, capturing, etc;
4. An overall idea on issues like data privacy, computer security, etc.

Due to restrictions on time and manpower resources, we are not able to make the tutorial completely self-contained. 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 users of this tutorial to have some previous skills/ knowledge:

1. Practical experience using computers and 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 basic idea on the workings of computer networks and the 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 on networking mechanisms and computer networks. The tutorial expects a user to understand technical concepts like the OSI model of networks and basic network attacks like the man-in-the-middle attack etc.

4. A bit of exposure to logical notations would be useful in understanding the filter strings. A brief idea on general computer-related issues like “data privacy", “computer security", “network security", “network protocol encapsulation" etc., will also 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 4 virtual machines (VMs) smoothly, with no noticeable lag or delay;
2. A virtualization software platform. For example, VMWare or VirtualBox;
3. Any VM with Wireshark installed and 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 4 virtual machines without any noticeable delay or lag. 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 operating systems used for this tutorial are VM 1 - Kali linux 2016.1, VM 2 - SEEDUbuntu 12.04 (See [1]), VM 3 - Windows 7, and VM 4 - Windows 10. This tutorial can also be completed with 1 VM with Wireshark installed if the live capture activity and challenge IV are skipped.

Note: Please go to part **2** of the **Appendix** section for Network Layout Diagram.

4 Hardware and Software Requirements Cont. <>

1. VMs connected over an internal network;
2. FTP server like vsftpd;
3. Updated hosts files.

1. Internal network with each VM having a static IP address. Our setup VM 1 - 10.1.1.3, VM 2 - 10.1.1.2, VM 3 - 10.1.1.5, and VM 4 10.1.1.4.
2. VM 2 - vsftpd configured to allow anonymous and user based access.
3. VM 2 - Text file named song.txt and a picture named mysteryFile.
4. VM 2 - Apache server needs to be running.
5. VM 1,3,4 - hosts files need to be configured with access to www.wtelectronicsstore.com, www.wtshoestore.com, and www.sqllabcollabtive.com. Which are running on VM 2's apache server.
6. VM 1 - Address Resolution Protocal (ARP) spoofing to have VM 1 be the middle man between all the VMs. Also IP forwarding needs to be enabled. Four bash scripts are used to accomplish this and they are named arpspoof1.sh, arpspoof2.sh, arpspoof3.sh, and arpspoof4.sh.

Note: Please go to part **1** of the **Appendix** section for information on resources and website links to access some of the required software for the tutorial.

5 Info: 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 [2] 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. [2]
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. [2]

6 Info: 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` [3] 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.

7 Info: Importance of Wireshark

1. The ability to capture and analyze network packets is important in facilitating:
   (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 for 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.

8 Related 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 [5]; [November 2014];
3. Use Wireshark to secure your home network [6]; [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 [5].
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 [6].

9 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?"

Answer to Q1:
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.

Answer to Q2:
Yes. For example, a user's phone may have an SSID saved called "Ritz Carlton – Honolulu, Hawaii".

Answer to Q3:
Promiscuous mode enables Wireshark to sniff out wireless traffic. When it's turned off, only traffic on the wired network can be captured.

10 Info: 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 [3].
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 [7]. 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.

11 Info: 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. Wireshark provides the capability to filter packets it displays with display filters;
3. The syntax for display filters is:
   (a) 'C'-like logical syntax. Which is very "programmer" friendly.
4. 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 In Wireshark when you type a display 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.

ii 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).

iii 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.

iv 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.

v 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.

vi 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 several 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 the contents of captured packets. It has many 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 Activity: 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 **TutorialFTP** (Wireshark will add a “.pcapng” suffix).
4. For more information on Display filter syntax see the Wireshark article on display filters. [8]

19 Activity: 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.

1. The exchange of “SYN” messages allows the client and server to negotiate the parameters of the TCP session.
2. 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.
3. The client then either accepts the parameters specified by the server or session establishment fails.

20 Activity: 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?

Answer to 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.

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

22 Observations: 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: Find Login Credentials <>

Capture files for these challenges were downloaded from [9].

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

Duration: 5 - 10 min.

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

24 Challenge II: Find the Mail Servers <>

Capture files for these challenges from the sample caps wiki [9].

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?

Duration: 5 - 10 min.

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: Dig Through Web Traffic <>

Capture files for these challenges are from the same wiki [9].

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.

Duration: 5 - 10 min.

26 Observations: Capture Options <>

1. The Capture Options display window has three tabs:
   (a) Input;
   (b) Output;
   (c) Options.

1. The input tab contains the capture pane and some extra settings.
   (a) The Capture pane shows a list of connected interfaces and their settings which includes:
      i. Interface – Name of the interface and its address if known.
      ii. Traffic – Graphical representation of interface traffic.
      iii. Link-layer header – Displays header type.
      iv. Promicuous mode – Shows if the interface should accept packets on the connected network that aren't intended for its MAC address. [10]
      v. Snaplen – Max data size for a packet. If the packet is larger then the set max value the packet will be sliced. [10]
      vi. Buffer – Kernel buffer size used to save packets. [10]
      vii. Monitor mode – Same as promicuous mode except it allows for the capturing of packets outside of one's network. [10]
      viii. Capture filter – Shows the currently set capture filter.
   (b) Below the capture pane you can set a capture filter for the selected interface.
2. Output tab allow for setting the capture files name and size limit before creating a new file.
3. Options tab allows for setting specific display options, names that should be resolved, and setting the capture limit.

27 Activity: Capture Filters <>

1. Capture filters allow us to be more selective about the data we capture;
2. Capture filters can't be changed after a capture has begun;
3. Uses libpcap filter language which is not very "programmer" friendly [11];
4. The syntax for capture filters is:
   (a) [**not**] **primitive** [**and**|**or** [**not**] **primitive** ...] [12].

Capturing all the traffic on one or more interfaces can generate an overwhelming amount of data. For example on my home network, which is not particularly busy, there are sixteen hundred packets per second being captured by Wireshark and this sort of traffic volume is not even particularly high. As you can see the need for capture filters is extremely important. To see this first hand lets start a capture on our current network.

1. The display window should be filling up with Address Resolution Protocol (ARP) requests. We don't need to see all these so we will add a capture filter that will ignore all ARP requests. Following the syntax mentioned above we will use `not arp`. Now rerun the capture and there should be no more ARP request in the display window.
2. Some of the most useful capture filter primitives are:
   (a) `host [IP address]` - Captures all incoming and outgoing packets from the given IP address
   (b) `net [IP address range]` - Capture all incoming and outgoing packets in the given IP address range
      i. IP address range is denoted - `192.168.0.0/25`
   (c) `dst [IP address]` - Captures all incoming packets from the given IP address
   (d) `src [IP address]` - Captures all outgoing packets from the given IP address
   (e) `port [port number]` - Capture all packets on the given port

28 Question: Capture Filters <>

Q7: How would you write a capture filter to ignore ARP requests, get all incoming traffic going to VM 2, and all outgoing traffic from VM 4.

* VM 1 - Kali `10.1.1.3`, VM 2 - Ubuntu `10.1.1.2`,
* VM 3 - Windows 7 `10.1.1.5`, and VM 4 - Windows 10 `10.1.1.4`

Answer to Q7:
not arp and (dst 10.1.1.2 or src 10.1.1.4)

29 Activity: Live Capture

1. Start a capture;
   (a) Use your default network interface (`eth0`);
   (b) Use a capture filter to only capture VM 4's traffic.
2. Open a browser in VM 3 and visit some of the available websites;
3. FTP from VM 4 to VM 2;
   (a) Get `song.txt` and `mysteryFile`.

* VM 1 - Kali `10.1.1.3`, VM 2 - Ubuntu `10.1.1.2`,
* VM 3 - Windows 7 `10.1.1.5`, and VM 4 - Windows 10 `10.1.1.4`.
* Elec Store - `www.wtelectronicsstore.com`;
* Shoe Store - `www.wtshoestore.com`;
* Collabtive - `www.sqllabcollabtive.com`

1. Using a capture filter so only VM 2's traffic is captured will reduce the amount of packets being captured on interface eth0. Which will effectively ignore all packets created by VM 3 in this activity.
2. The available websites for VM 3 are bookmarked in Internet Explorer.
3. If packets from VM 3 show up in Wireshark when browsing the available websites in step 2 retry the capture with a different capture filter.
4. Command to FTP into VM 2 is - ftp [target ip]
5. The login credentials for FTPing into VM 2.
   (a) username: seed and password: dees
6. Command to get a file over FTP - `get [target file name] [local file name]`
7. Command to end FTP - `quit`

30 Activity: Live Capture Continued

<>

1. Open a browser in VM 4 and visit the Elec Store;
2. VM 4 go to Collabtive and login;
3. VM 4 edit and save profile info on Collabtive;
4. VM 4 go to the Shoe Store and click on one item;
5. VM 4 go back to the Elec Store;
6. Stop the capture.

* VM 1 - Kali `10.1.1.3`, VM 2 - Ubuntu `10.1.1.2`,
* VM 3 - Windows 7 `10.1.1.5`, and VM 4 - Windows 10 `10.1.1.4`
* Elec Store - `www.wtelectronicsstore.com`
* Shoe Store - `www.wtshoestore.com`
* Collabtive - `www.sqllabcollabtive.com`

1. Browse the Elec Store and click on some of the TVs to get some more activity in this capture.
2. Login for Collabtive;
   (a) username: `admin` & password: `admin`
3. On Collabtive got to the account profile page. Then edit the user's profile info and save the changes.
4. Visit the Shoe Store and click on view details of one pair of shoes.
5. Return to the Elec Store and browse through some of the TVs.
6. Finally stop the capture.

31 Challenge IV: Find Login Credentials <>

Use the capture created in the last activity (Live Capture);

1. Find the `www.sqllabcollabtive.com` login credentials.

Duration: 5 - 10 min.

Hint: Filter for `HTTP` protocol.

32 Challenge V: Find the Shoes <>

Use the capture created in the last activity (Live Capture);

1. Find the brand and ID of the pair of shoes looked at.

Duration: 5 - 10 min.

Hint: Filter for `HTTP` protocol.

33 Challenge VI: Find FTP Login Credentials <>

Use the capture created in the last activity (Live Capture);

1. Find the FTP login credentials.

Duration: 5 - 10 min.

Hint: Filter for `FTP` protocol.

34 Challenge VII: Find the Files <>

Use the capture created in the last activity (Live Capture)

1. Find what the contents of `song.txt`.
2. **Bonus**: What is the file named `mysterFile`?

Duration: 5 - 10 min.

Hint 1: Filter by ftp-data;
Hint 2: Check the beginning of the contents of the mysteryFile.

35 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.

36 Appendix: Solutions, and Related Resources <

1. Solutions to the challenges:
   (a) Challenge I;
   (b) Challenge II;
   (c) Challenge III;
   (d) Challenge IV;
2. Network Layout Diagram;
3. Tutorial-Related Resources;
4. 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: `smtp1.google.com`; Address: `216.239.57.25`;
B. Name: `smtp2.google.com`; Address: `216.239.37.25`;
C. Name: `smtp3.google.com`; Address: `216.239.57.26`;
D. Name: `smtp4.google.com`; Address: `216.239.37.26`;
E. Name: `smtp5.google.com`; Address: `64.233.167.25`;
F. Name: `smtp6.google.com`; Address: `66.102.9.25`.

(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`.

(d) **Challenge IV:**
Use the capture created during the activity Live Capture.

i. Type in '`http.request.method == POST`' into the *Apply a display filter* field.
ii. Click on each POST packet and check in the packet bytes pane right hand ASCII column for something similar to '`username=admin&pass=admin`'

(e) **Challenge V:**
Use the capture created during the activity Live Capture.

i. Type in '`http.request.method == POST`' into the *Apply a display filter* field.
ii. Click on each POST packet and check in the Packet Details pane. Open up the HTML Form URL Encoded drop down and look at the form items. Should be something like '`hdnMobileName = Puma`' and '`hdnMobileId = s2`'.

(f) **Challenge VI:**
Use the capture created during the activity Live Capture.

i. Type in '`ftp`' into the *Apply a display filter* field.
ii. In the Packet List pane check the info of the FTP packets and you should see '`Request: USER seed`' and another packet with '`Request: PASS dees`'.

(g) **Challenge VII:**
Use the capture created during the activity Live Capture.

i. Part 1:

   A. Type in '`ftp-data`' into the *Apply a display filter* field.

   B. Right click on each packet and click follow tcp stream.

   C. The contents of song.txt begins with '`You defeated the evil that was here`'.

ii. Part 2:

   A. Type in '`ftp-data`' into the *Apply a display filter* field.

   B. Right click on each packet and click follow tcp stream.

   C. The contents of mysteryFile begins with '`CREATOR: gd-jpeg`'.

   D. Change '`Show data as`' to Raw and click '`Save as...`'.

   E. View the saved file and you will see an image of an Ubuntu desktop.

1. **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;

   SEED Ubuntu 12.04 VM - CLICK ME;

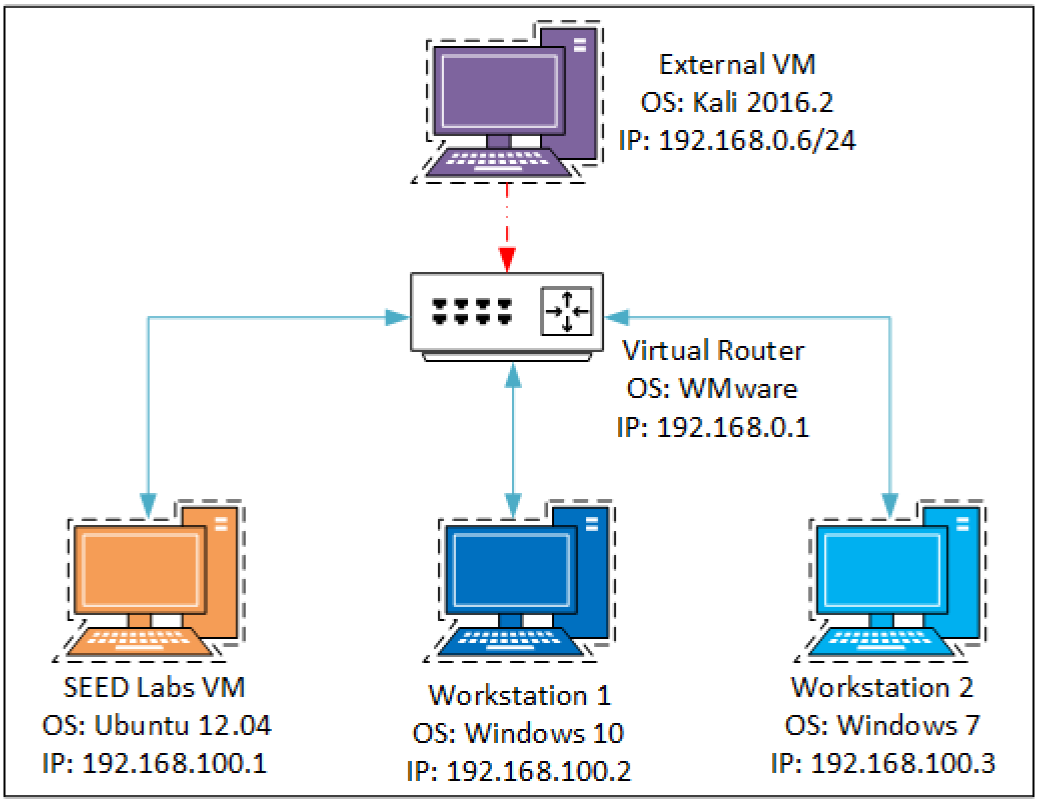
   Instructions vsftpd Part 1 - CLICK ME;

   Instructions vsftpd Part 2 - CLICK ME;

   Details arpspoof - CLICK ME;

   **Wireshark1.pcapng - Included in the “Captures" folder was created by Jon Meyer, Jared Zook and Ananth Jillepalli;

2. **Network Layout Diagram:**



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.
v3	Jan. 29th 2017	Robert Breckenridge	Added Observations: Capture Options, Activity: Capture Filters, Question: Capture Filters, Activity: Live Capture, and Challenge IV-VII. Updated Hardware and Software Requirements and adjusted some formatting throughout the document.
v3.1	Jan. 30th 2017	Matthew Holman	Updated/ edited wording throughout the entire document for increased clarity.
v3.2	July 1st 2017	Ananth Jillpalli	Standardization (network layout diagram edits, consistency, TeX markup cleaning, and more)

References

[1] D. Wenliang. Lab envrionment setup. [Online]. Available: http://www.cis.syr.edu/~wedu/seed/lab_env.html

[2] . The Wireshark Foundation. Wireshark. [Online]. Available: http://www.wireshark.org

[3] . Wikimedia Foundation Inc. tcpdump. [Online]. Available: http://en.wikipedia.org/wiki/Tcpdump

[4] K. Zetter. (2015) How to detect sneaky nsa quantum insert attacks. [Online]. Available: https://www.wired.com/2015/04/researchers-uncover-method-detect-nsa-quantum-insert-hacks/

[5] S. Gallagher. (2014) Where've you been? your smartphone's wi-fi is telling everyone. [Online]. Available: http://arstechnica.com/information-technology/2014/11/where-have-you-been-your-smartphones-wi-fi-is-telling-everyone/

[6] A. Henry. (2014) How to tap your network and see everything that happens on it. [Online]. Available: http://lifehacker.com/how-to-tap-your-network-and-see-everything-that-happens-1649292940

[7] . Wikimedia Foundation Inc. pcap. [Online]. Available: http://en.wikipedia.org/wiki/Pcap

[8] Wireshark.org. Displayfilters. [Online]. Available: https://wiki.wireshark.org/DisplayFilters

[9] ——. Samplecaptures. [Online]. Available: https://wiki.wireshark.org/SampleCaptures

[10] . The Wireshark Foundation. The capture options dialog box. [Online]. Available: https://www.wireshark.org/docs/wsug_html_chunked/ChCapCaptureOptions.html

[11] Wireshark.org. Capturefilters. [Online]. Available: https://wiki.wireshark.org/CaptureFilters

[12] . The Wireshark Foundation. Filtering while capturing. [Online]. Available: https://www.wireshark.org/docs/wsug_html_chunked/ChCapCaptureFilterSection.html