

计算机科学与工程系

Department of Computer Science and Engineering

CS 315 Computer Security Course

Lab 1: Packet Sniffing and Wireshark

Introduction

The first part of the lab introduces packet sniffer, Wireshark. Wireshark is a free opensource network protocol analyzer. It is used for network troubleshooting and communication protocol analysis. Wireshark captures network packets in real time and display them in human-readable format. It provides many advanced features including live capture and offline analysis, three-pane packet browser, coloring rules for analysis. This document uses Wireshark for the experiments, and it covers Wireshark installation, packet capturing, and protocol analysis.



Figure 1: Wireshark in Kali Linux



Background

TCP/IP Network Stack



Figure 2: Encapsulation of Data in the TCP/IP Network Stack

In the Introduction to Computer Networking Course, TCP/IP network stack is introduced and studied. This background section briefly explains the concept of TCP/IP network stack to help you better understand the experiments. TCP/IP is the most commonly used network model for Internet services. Because its most important protocols, the Transmission Control Protocol (TCP) and the Internet Protocol (IP) were the first networking protocols defined in this standard, it is named as TCP/IP. However, it contains multiple layers including application layer, transport layer, network layer, and data link layer.

- Application Layer: The application layer includes the protocols used by most applications for providing user services. Examples of application layer protocols are Hypertext Transfer Protocol (HTTP), Secure Shell (SSH), File Transfer Protocol (FTP), and Simple Mail Transfer Protocol (SMTP).



- Transport Layer: The transport layer establishes process-to-process connectivity, and it provides end-to-end services that are independent of underlying user data. To implement the process-to-process communication, the protocol introduces a concept of port. The examples of transport layer protocols are Transport Control Protocol (TCP) and User Datagram Protocol (UDP). The TCP provides flowcontrol, connection establishment, and reliable transmission of data, while the UDP is a connectionless transmission model.
- Internet Layer: The Internet layer is responsible for sending packets to across networks. It has two functions: 1) Host identification by using IP addressing system (IPv4 and IPv6); and 2) packets routing from source to destination. The examples of Internet layer protocols are Internet Protocol (IP), Internet Control Message Protocol (ICMP), and Address Resolution Protocol (ARP).
- *Link Layer*: The link layer defines the networking methods within the scope of the local network link. It is used to move the packets between two hosts on the same link. An common example of link layer protocols is Ethernet.

Packet Sniffer

Packet sniffer is a basic tool for observing network packet exchanges in a computer. As the name suggests, a packet sniffer captures ("sniffs") packets being sent/received from/by your computer; it will also typically store and/or display the contents of the various protocol fields in these captured packets. A packet sniffer itself is passive. It observes messages being sent and received by applications and protocols running on your computer, but never sends packets itself.

Figure 3 shows the structure of a packet sniffer. At the right of **Figure** 3 are the protocols (in this case, Internet protocols) and applications (such as a web browser or ftp client) that normally run on your computer. The packet sniffer, shown within the dashed rectangle in **Figure** 3 is an addition to the usual software in your computer, and consists of two parts. The packet capture library receives a copy of every link-layer frame that is sent from or received by your computer. Messages exchanged by higher layer protocols such as HTTP, FTP, TCP, UDP, DNS, or IP all are eventually encapsulated in link-layer frames that are transmitted over physical media such as an Ethernet cable. In Figure 1, the assumed physical media is an Ethernet, and so all upper-layer protocols are eventually encapsulated within an Ethernet frame. Capturing all link-layer frames thus gives you access to all messages sent/received from/by all protocols and applications executing in your computer.

The second component of a packet sniffer is the packet analyzer, which displays the contents of all fields within a protocol message. In order to do so, the packet analyzer





Figure 3: Packet Sniffer Structure

must "understand" the structure of all messages exchanged by protocols. For example, suppose we are interested in displaying the various fields in messages exchanged by the HTTP protocol in **Figure** 3. The packet analyzer understands the format of Ethernet frames, and so can identify the IP datagram within an Ethernet frame. It also understands the IP datagram format, so that it can extract the TCP segment within the IP datagram. Finally, it understands the TCP segment. Finally, it understands the TCP segment. Finally, it understands the HTTP protocol and so, for example, knows that the first bytes of an HTTP message will contain the string "GET," "POST," or "HEAD".

We will be using the Wireshark packet sniffer [http://www.wireshark.org/] for these labs, allowing us to display the contents of messages being sent/received from/by protocols at different levels of the protocol stack. (Technically speaking, Wireshark is a packet analyzer that uses a packet capture library in your computer). Wireshark is a free network protocol analyzer that runs on Windows, Linux/Unix, and Mac computers.



Getting Wireshark

The Kai Linux has Wireshark installed. You can just launch the Kali Linux VM and open Wireshark there. Wireshark can also be downloaded from here:

https://www.wireshark.org/download.html



Figure 4: Download Page of Wireshark



Starting Wireshark

When you run the Wireshark program, the Wireshark graphic user interface will be shown as **Figure** 5. Currently, the program is not capturing the packets.



Figure 5: Initial Graphic User Interface of Wireshark

Then, you need to choose an interface. If you are running the Wireshark on your laptop, you need to select WiFi interface. If you are at a desktop, you need to select the Ethernet interface being used. Note that there could be multiple interfaces. In general, you can select any interface but that does not mean that traffic will flow through that interface. The



network interfaces (i.e., the physical connections) that your computer has to the network are shown. The attached **Figure** 6 was taken from my computer.

After you select the interface, you can click start to capture the packets as shown in **Figure** 7.

| | | | Wireshark: Capture Inter | aces | | • • • |
|----|----------|------------|--------------------------|----------------|---------|-----------|
| | | Device | Description | IP | Packets | Packets/s |
| | | eth0 | | 172.16.108.151 | 0 | 0 |
| | P | any | | none | 0 | 0 |
| | _ | lo | | 127.0.0.1 | 0 | 0 |
| | 8 | bluetooth0 | | none | 44 | 0 |
| | P | nflog | | none | 0 | 0 |
| | _ | nfqueue | | none | 0 | 0 |
| | | usbmon1 | | none | 0 | 0 |
| | | usbmon2 | | none | 0 | 0 |
| He | elp | | Start | Stop | ions | Close |

Figure 6: Capture Interfaces in Wireshark

| | Capturing from eth0 [Wireshark 1.12.6 (Git Rev Unknown from unknown)] | • • • |
|---|---|--|
| File Edit View Go Capture Analyze St | tatistics Telephony Tools Internals Help | |
| 🖲 🖲 📕 🔬 🤠 🗂 🛪 🕫 | ୧ ቀ ⇒) ∓ ± 📃 🚽 ୧୧୧୯ 🕾 🕍 🕅 🗄 🗉 👔 | |
| Filter: | - Expression Clear Apply Save | |
| No. Time Source 1 0.00000000 172.16.108.151 2 0.000293000 172.16.108.151 3 0.000291000 172.16.108.151 4 0.000390000 172.16.108.151 5 0.001283000 172.16.108.151 • Frame 4: 74 bytes on wire (592 bit | Destination Protocol Length Info 172.16.108.2 DNS 72 Standard query 0x28c5 A www.kali.o 172.16.108.2 DNS 72 Standard query 0x630c AAAA www.kali.o 172.16.108.2 DNS 74 Standard query 0x26d4 A tools.kali 172.16.108.2 DNS 74 Standard query 0x26a0 AAAA tools.kali 172.16.108.2 DNS 74 Standard query 0x3dbb A www.offens 172.16.108.2 DNS 86 Standard query 0x3dbb A www.offens is), 74 bytes captured (592 bits) on interface 0 0 0 | rg i.org ali.org ive-security.com |
| Ethernet II, Src: Vmware_6d:7a:35 Internet Protocol Version 4, Src: User Datagram Protocol, Src Port: Demain Name System (query) | (00:0c:29:6d:7a:35), Dst: Vmware_f0:1a:b5 (00:50:56:f0:1a:b5) 172.16.108.151 (172.16.108.151), Dst: 172.16.108.2 (172.16.108.2) 46328 (46328), Dst Port: 53 (53) | |
| | | |
| 0000 00 50 56 f0 1a b5 00 cc 29 6d 0010 00 3c 37 6f 40 00 40 11 d2 87 0020 6c 02 b4 f8 00 35 00 28 63 35 00 83 53 0030 00 00 00 00 00 57 4 6f 6f 0040 69 03 6f 72 67 00 00 01 00 01 | 7a 35 08 00 45 00 .PV | |
| ◯ 💅 eth0: <live capture="" in="" progress=""> File: /t</live> | Packets: 42 · Displayed: 42 (100.0%) | Profile: Default |







Figure 8: Wireshark Graphical User Interface on Microsoft Windows

The Wireshark interface has five major components:

The **command menus** are standard pulldown menus located at the top of the window. Of interest to us now is the File and Capture menus. The File menu allows you to save captured packet data or open a file containing previously captured packet data, and exit the Wireshark application. The Capture menu allows you to begin packet capture.

The **packet-listing window** displays a one-line summary for each packet captured, including the packet number (assigned by Wireshark; this is not a packet number contained in any protocol's header), the time at which the packet was captured, the packet's source and destination addresses, the protocol type, and protocol-specific information contained in the packet. The packet listing can be sorted according to any of these categories by clicking on a column name. The protocol type field lists the highest-level protocol that sent or received this packet, i.e., the protocol that is the source or ultimate sink for this packet.



The **packet-header details window** provides details about the packet selected (highlighted) in the packet-listing window. (To select a packet in the packet-listing window, place the cursor over the packet's one-line summary in the packet-listing window and click with the left mouse button.). These details include information about the Ethernet frame and IP datagram that contains this packet. The amount of Ethernet and IP-layer detail displayed can be expanded or minimized by clicking on the right-pointing or down-pointing arrowhead to the left of the Ethernet frame or IP datagram line in the packet details window. If the packet has been carried over TCP or UDP, TCP or UDP details will also be displayed, which can similarly be expanded or minimized. Finally, details about the highest-level protocol that sent or received this packet are also provided.

The **packet-contents window** displays the entire contents of the captured frame, in both ASCII and hexadecimal format.

Towards the top of the Wireshark graphical user interface, is the **packet display filter field**, into which a protocol name or other information can be entered in order to filter the information displayed in the packet-listing window (and hence the packet-header and packet-contents windows). In the example below, we'll use the packet-display filter field to have Wireshark hide (not display) packets except those that correspond to HTTP messages.



Capturing Packets

After downloading and installing Wireshark, you can launch it and click the name of an interface under Interface List to start capturing packets on that interface. For example, if you want to capture traffic on the wireless network, click your wireless interface.

Test Run

Do the following steps:

- 1. Start up the Wireshark program (select an interface and press start to capture packets).
- 2. Start up your favorite browser (ceweasel in Kali Linux).
- 3. In your browser, go to Wayne State homepage by typing www.wayne.edu.
- 4. After your browser has displayed the http://www.wayne.edu page, stop Wireshark packet capture by selecting stop in the Wireshark capture window. This will cause the Wireshark capture window to disappear and the main Wireshark window to display all packets captured since you began packet capture see image below:





- 5. Color Coding: You'll probably see packets highlighted in green, blue, and black. Wireshark uses colors to help you identify the types of traffic at a glance. By default, green is TCP traffic, dark blue is DNS traffic, light blue is UDP traffic, and black identifies TCP packets with problems — for example, they could have been delivered out-of-order.
- 6. You now have live packet data that contains all protocol messages exchanged between your computer and other network entities! However, as you will notice the HTTP messages are not clearly shown because there are many other packets included in the packet capture. Even though the only action you took was to open your browser, there are many other programs in your computer that communicate via the network in the background. To filter the connections to the ones we want to focus on, we have to use the filtering functionality of Wireshark by typing "http" in the filtering field as shown below:

| | | | Capturing | rom ethO [Wireshark | 1.12. | 5 (Git Rev Unknown from unknown)] | • | 8 |
|---------|---|--------------------------|--|-----------------------|--------|--|---|---|
| File E | dit View Go C | apture Analyze S | Statistics Telephony Too | ls Internals Help | | | | |
| • |) 🖊 🗖 🔏 | 市 🖹 × の | ०, + + .⊅ ∓ | t 📑 🔹 🔍 Q | ર્ ૦ | 🖺 😹 🛛 📴 🗉 🚺 | | |
| Filter: | http | | - | Expression Clear Appl | ly Sav | e | | |
| No. | Time | Source | Destination | Protocol Le | ength | Info | | |
| 41 | 24 25.63383500 | 172.16.108.152 | 141.217.1.160 | HTTP | 508 | GET /promos/1376/programs-min_1.png HTTP/1.1 | | |
| 41 | 26 25.63399800 | 172.16.108.152 | 141.217.1.160 | HTTP | 513 | GET /promos/1376/apply-students-2015.jpg HTTP/1.1 | | |
| 41 | 50 25.66536200 | 141.217.1.160 | 172.16.108.15 | 2 HTTP | 287 | HTTP/1.1 200 OK (text/javascript) | | |
| 41 | 73 25.67091200 | 172.16.108.152 | 141.217.1.160 | HTTP | 528 | GET /promos/1376/winter-registration2015-3section_1.jpg HTTP/1.1 | | |
| 41 | 3/ 25.6/288800 | 141.21/.1.160 | 1/2.16.108.15 | 2 HITP | 1247 | HTTP/I.I 200 OK (PNG) | | |
| 42 | 28 25.6/323100 | 1/2.10.108.152 | 141.217.1.100 | | 1204 | HTTP/1 1 200 OK (CTEROD) | | |
| 42. | 23 25.68285500 | 141.217.1.100 | 172.10.108.15 | 2 HITP | 402 | HTTP/1.1 200 OK (OLPOSA) | | |
| 42 | 35 25.68306500 | 172.16.108.152 | 141.217.1.160 | нттр | 508 | GET /images/news/van-iones-news.ing HTTP/1.1 | | |
| 42 | 36 25,68308400 | 172.16.108.152 | 141.217.1.160 | HTTP | 518 | GET / resources/images/footer/give-to-wsu.gif HTTP/1.1 | | |
| 42 | 50 25.68478900 | 141.217.1.160 | 172.16.108.15 | 2 HTTP | 1402 | HTTP/1.1 200 OK (PNG) | | |
| Erail | ne 4126 513 h | tes on wire (41 | 04 hits) 513 hytes c | antured (4104 bits) | on in | terface 0 | | |
| > Eth | ernet II. Src: | Vmware 6d:7a:35 | (00:0c:29:6d:7a:35). | Dst: Vmware f0:1a:b | 5 (00 | :50:56:f0:1a:b5) | | |
| Int | ernet Protocol | Version 4, Src: | 172.16.108.152 (172. | | 1.217 | 1.160 (141.217.1.160) | | |
| ▶ Tra | nsmission Contr | ol Protocol, Sr | c Port: 52099 (52099) | , Dst Port: 80 (80), | Seq: | 1, Ack: 1, Len: 459 | | |
| Hype | ertext Transfer | r Protocol | | | | | | |
| | | | | | | | | |
| 0000 | 00 50 56 f0 1a | h5 00 0c 29 6 | 4 70 25 08 00 45 00 | DV)m75 E | | | | _ |
| 0010 | 01 f3 32 ce 40 | 00 40 06 5e 1 | 5 ac 10 6c 98 8d d9 | 2.@.@. ^l | | | | |
| 0020 | 01 a0 cb 83 00 |)50 f1 b7 ba2 | 6 14 6a 7a c6 50 18 | P&.jz.P. | | | | |
| 0030 | 72 10 c3 8a 00 |)004/45542 7362f61707 | 0 21 70 72 61 60 61 0 6c 79 2d 73 74 75 | rGE /promo | | | | |
| 0050 | 64 65 6e 74 73 | 3 2d 32 30 31 3 | 5 2e 6a 70 67 20 48 | dents-20 15.jpg H | | | | |
| 0060 | 54 54 50 2f 31 | 2e 31 Od 0a 4 | 8 6f 73 74 3a 20 77 | TTP/1.1Host: w | | | | |
| 0070 | 61 79 6e 65 2e | 96564750d0 | a 55 73 65 72 2d 41 | ayne.eduUser-A | | | | |
| 0090 | 30 20 28 58 31 | 31 3b 20 4c 6 | 9 6e 75 78 20 69 36 | 0 (X11; Linux i6 | | | | |
| 00a0 | 38 36 3b 20 72 | 2 76 3a 33 31 2 | e 30 29 20 47 65 63 | 86; rv:3 1.0) Gec | | | | |
| 0000 | 66 6f 78 2f 32 30 | 313030313 | 9 63 65 77 65 61 73 | K0/20100 101 Fire | | | | |
| obdo | 65 6c 2f 33 31 | 2e 38 2e 30 0 | d 0a 41 63 63 65 70 | el/31.8. 0Accep | | | | |
| 00e0 | 74 3a 20 69 60 | 61 67 65 2f 7 | 0 6e 67 2c 69 6d 61 | t: image /png,ima | | | | |
| 0010 | 6/ 65 21 2a 3b |)/I30/30/2e3 | 8 2C 2a 2T 2a 3b 71 | ge/*;q=∪ .8,*/*;q | | | | |
|) 🎽 | eth0: <live capture<="" td=""><td>in progress> File: /t</td><td>Packets: 5085 · Displayed:</td><td>60 (1.2%)</td><td></td><td>Profile: Default</td><td></td><td></td></live> | in progress> File: /t | Packets: 5085 · Displayed: | 60 (1.2%) | | Profile: Default | | |

Notice that we now view only the packets that are of protocol HTTP. However, we also still do not have the exact communication we want to focus on because using HTTP as a filter is not descriptive enough to allow us to find our connection to http://www.wayne.edu. We need to be more precise if we want to capture the correct set of packets.



7. To further filter packets in Wireshark, we need to use a more precise filter. By setting the http.host==sustech, we are restricting the view to packets that have as an http host the www.wayne.edu website. Notice that we need two equal signs to perform the match "==" not just one. See the screenshot below:

| Prome Total Control Contr Control Control | ••• | | | | _ Wi-F | Fi:en1 | |
|--|---|---|---|--|--------------------------------------|--|--------------------|
| Internet Transfer Protocol Prace Decination Prace Bource Decination Prace Decination Decination <thdecination< th=""> Decination</thdecination<> | http | | | | | × × ⊥ | European in the |
| Description Description Description Protect Largh Hole 1089 334.64126 192.168.1.100 232.352.552.55 SSDP 217 M=SAR(H + HTP/1.1 1093 35.64126 192.168.1.100 239.255.255.25 SSDP 217 M=SAR(H + HTP/1.1 1093 35.642013 192.168.1.100 239.255.255.25 SSDP 217 M=SAR(H + HTP/1.1 1095 36.642013 192.168.1.100 239.255.255.25 SSDP 217 M=SAR(H + HTP/1.1 1095 36.642013 192.168.1.100 239.255.255.25 SSDP 217 M=SAR(H + HTP/1.1 1095 56.642013 192.168.1.100 239.355.255.25 SSDP 217 M=SAR(H + HTP/1.1 1095 56.77 ATS 192.168.1.100 218.38.133.56 HTTP 985 GET / answer?148-536496601=16.60huurl=https://doi.00m/276rnd=225f938728 1107.168.1.100 219.38.133.56 HTTP 985 GET / answer?148-536496601=16.60huurl=https://doi.00m/276rnd=225f938728 1107.168.1.100 219.108.011.010.512.297.525.55 User Datagram Protocol Version 4, Str.190.512.297.525.55 User Datagram Protocol Version 4, Str.190.512.297.525.55 <td>Wire</td> <td>less controls are n</td> <td></td> <td></td> <td></td> <td></td> <td>802.11 Preferences</td> | Wire | less controls are n | | | | | 802.11 Preferences |
| 1000 33.64031 107.166.1.100 237.255.255.25 SSDP 217 M=SARCH + HTTP.1.1 1001 33.64176 137.166.1.100 239.255.255.25 SSDP 217 M=SARCH + HTTP.1.1 1002 33.64176 137.166.1.100 239.255.255.25 SSDP 217 M=SARCH + HTTP.1.1 1002 35.64176 137.166.1.100 239.255.255.25 SSDP 217 M=SARCH + HTTP.1.1 1002 35.64176 137.166.1.100 239.255.255.25 SSDP 217 M=SARCH + HTTP.1.1 1002 35.64176 137.166.1.100 239.255.255.25 SSDP 217 M=SARCH + HTTP.1.1 1002 35.64176 137.166.1.100 239.255.255.25 SSDP 217 M=SARCH + HTTP.1.1 1003 35.64176 1372.166.1.100 239.255.255.25 HTTP 965 GET / Answer1#530696661=15:0huurl=httph3A:2Px2Pxwx.ifeng.com/2F6rnd=2251938724. Frame 1085: 217 bytes on wire (1726 bits), 217 bytes captured (1726 bits) on interface 9 Ethernet 17, 5rc: 14:83:erio7:28:18 LeitBits: 10.400.100.100.100.100.100.100.100.100.1 | No. | Time | Source | Destination | Protocol | Length Info | |
| <pre>> Frame 1888: 217 bytes on wire (1736 bits), 217 bytes captured (1736 bits) on interface 0 > Ethernet II, 5rc: 34:83:e7:67:28:18], 04:83:e7:67:28:18], Dst: IPv4ecast_7f:ff:fa (01:00:5e:7f:ff:fa) > Internet Protocol Version 4, 5rc: 392.168.1.100, Dst: 292.55.255.256 > User Datagram Protocol, 5rc for 60:60:60:60:60:60:60:60:60:60:60:60:60:6</pre> | 100 100 100 100 543 | 38. 33, 640931 34, 641026 33, 541764 56, 642013 56, 642013 560, 960002 | 192.165.1100 192.165.1.100 192.165.1.100 193.165.1.100 193.165.1.100 192.166.1.100 | 239,255,255,259 239,255,255,259 239,255,255,259 239,255,255,259 239,255,255,259 218,30,103,56 | SSDP SSDP SSDP SSDP HTTP | 217 H-SEACH + HTTP/1.1 217 H-SEACH + HTTP/1.1 217 H-SEACH + HTTP/1.1 217 H-SEACH + HTTP/1.1 905 GET /answer2id=3364966b1=16sohuurl=httph3Ak2F%2Fwaw.ifeng.com%2F6r | nd=225193a720 |
| 0000 01 00 5e 7f ff fa a4 83 c7 67 28 18 08 00 45 00 | ▶ Fran Ethe Inte User ▶ Hype | e 1088: 217 by rnet II, Src: rnet Protocol Datagram Prot rtext Transfer | rtes on wire (1736 bits), 217 bytes a4:83:e7:67:28:18 (a4:83:e7:67:28:1 Version 4, Src: 192,168.1.100, Dst socol, Src Port: 60366 (60366), Dst Protocol | captured (1736 bits) on 80, Dst: IPv4mcast_7f:f 239.255.255.250 Port: 1900 (1900) | interface f:fa (01:0 | 0 0:5e:7f:ff:fa) | |
| 0000 01 05 07 75 12 08 04 50 1 | | | | | | | |
| | 0000 0010 0020 0030 0040 0050 0050 0050 0050 0050 005 | 01 00 5e 7f ff 00 cb 9d 24 00 17 ab 24 00 ce 07 43 48 20 2a 20 40 41 3a 20 41 45 35 43 20 2a 20 30 3a 22 73 73 64 70 3a 20 47 60 44 58 3a 20 46 67 64 69 61 6c 73 36 62 33 10 40 3a 20 47 67 73 66 3a 31 0d 6a 32 20 47 61 67 73 36 2a 20 27 33 34 24 67 67 73 66 53 20 58 24 75 20 58 24 53 20 58 24 | $ \begin{array}{cccccccccccccccccccccccccccccccccccc$ | | | | |

8. Now, we can try another protocol. Let's use Domain Name System (DNS) protocol as an example here.

| | | | | *et | hO [Wireshark 1.1 | 2.6 (Git | Rev Unkno | wn from | unknown)] | | | | • | • • |
|--------------------------|---------------------|----------------------------|--|---------------------------------------|---------------------------------|------------|------------|---------|------------|-----------|-------------------|---------------------|------|---------|
| File E | dit View Go (| apture Analy: | ze Statistics | s Telephony Too | ls Internals Help | | | | | | | | | |
| • | 3 🔳 🦻 | to 🗋 × | n 0, | + + .∂ ∓ | ± 📃 🕃 🔍 | ର୍ଷ୍ | FT 🛃 | . 🗹 🍢 | 3 |) | | | | |
| Filter: | dns | | | e e e e e e e e e e e e e e e e e e e | xpression Clear A | Apply Sav | ve | | | | | | | |
| No. | Time | Source | | Destination | Protocol | Length | Info | | | | | | | |
| 40 | 5 25.43294300 | 172.16.108. | .2 | 172.16.108.152 | 2 DNS | 210 | Standard | query (| response (| 0x6c5a | CNAME whv2prod.c | cc.wayne.edu A 141. | 217. | 1.160 |
| 412 | 28 25.64762900 | 172.16.108. | .152 | 172.16.108.2 | DNS | 77 | Standard | query (| 0x7e2e A | fonts. | gstatic.com | | | |
| 412 | 29 25.64776200 | 172.16.108 | .152 | 172.16.108.2 | DNS | 77 | Standard | query (| Dxa3fa A | AAA font | ts.gstatic.com | | | |
| 413 | 30 25.65439100 | 172.16.108 | .2 | 172.16.108.152 | 2 DNS | 489 | Standard | query i | response | 0x7e2e | CNAME gstaticads | ssl.l.google.com A | 192. | 122.18 |
| 420 | 3 25.67303700 | 172.16.108. | .2 | 172.16.108.152 | 2 DNS | 277 | Standard | query i | response (| 0xa3fa | CNAME gstaticads | ssl.l.google.com AA | AA 2 | 607:fE |
| 480 | 08 26.12153900 | 172.16.108. | .152 | 172.16.108.2 | DNS | 79 | Standard | query (| Dx2bdl A | insight | ts.hotjar.com | | | 1 |
| 480 | 9 26.12165600 |) 1/2.16.108. | .152 | 1/2.16.108.2 | DNS | /9 | Standard | query (| 0xd529 A | AAA 1nsi | ights.hotjar.com | | | |
| 48. | 1 26.12442/00 | 172.16.108. | .2 | 172.16.108.152 | 2 DNS | 219 | Standard | query | response | 0xd529 | CNAME insights. | 1411/36383.eu-west- | 1.et | b.amaz(|
| 48. | 4 20.12044900 | 172.16.108. | 150 | 172.16.108.154 | DNS | 380 | Standard | query i | vesponse v | incidh | CNAME Insignts | 1411/30383.eu-west- | 1.et | b.amaz |
| 48. | 2 26.13119000 | 172.10.108. | 152 | 172.16.108.2 | DNS | 79 | Standard | query (| JXDDe2 A | AAA ipei | ighte hotion com | | | |
| 40, | .5 20.13129000 | . 1/2.10.108. | (1.1.52 | 1/2.10.108.2 | | 15 | Scanuaru | query | JX7972 A | AAA 11151 | rgires.nocjar.com | | | |
| Frank | 1e 4015: 210 b | ytes on wire | (1680 bits (00)50 | s), 210 bytes ca | ptured (1680 bit | s) on in | iterface (|) | | | | | | |
| , Eure | rnet II, Src. | Vension 4 | a.b3 (00.30 | 5.30.10.14.D3), | 109 2) Dot: 172 | 16 100 | 150 (170 | 16 100 | 150) | | | | | |
| > Lico | Detegrem Pro | tocol Src P | ort: 52 (53 | 3.108.2 (1/2.10. 3) Det Port: 40 | 108.2), DSL. 1/2 M30 (M0430) | . 10. 108. | 152 (1/2. | 10.108. | 132) | | | | | |
| Doma | in Name System | m (response) | 010.00(00 | 57, 530 Port. 40 | (40430) | | | | | | | | | |
| , Donie | iiii nanc system | in (response) | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |
| 0000 | 00 0c 29 6d 7 | a 35 00 50 5 | 56 f0 1a b5 | 5 08 00 45 00 |)mz5.P VE | • | | | | | | | | |
| 0010 | 00 c4 bd 93 0 | 1 ee 00 h0 | 4b da ac 10 | 0 6C 02 aC 10 | 1 5 D17 | • | | | | | | | | |
| 0030 | 00 02 00 02 0 | 03 03 77 1 | 77 77 05 77 | 7 61 79 6e 65 | w ww.wayn | e | | | | | | | | |
| 0040 | 03 65 64 75 0 | 0 00 01 00 0 | 01 c0 0c 00 | 0 05 00 01 00 | .edu | | | | | | | | | |
| 0050 | 00 00 05 00 0 | e 08 77 68 | 76 32 70 72 | 2 6f 64 02 63 | wh v2prod. | C | | | | | | | | |
| 0060 | dg 01 a0 c0 1 | 00 01 00 0 | 01 00 00 00 | 0 05 00 04 80 | c+ | • | | | | | | | | |
| 0080 | 6e 73 32 c0 1 | D c0 10 00 (| 02 00 01 00 | 00 00 05 00 | ns2 | | | | | | | | | |
| 0090 | 05 02 6e 73 c | 0 10 c0 67 (| 00 01 00 01 | L 00 00 00 05 | nsg | | | | | | | | | |
| 00a0 | 00 04 8d d9 9a | a a2 c0 55 (| 00 01 00 01 | L 00 00 00 05 | ······ | | | | | | | | | |
| 0000 | 00 04 80 09 9 | aaacu 55 (7 oo oo oo - | fo oe oo or | | U | • | | | | | | | | |
| 00d0 | 00 02 | 25 00 00 | | | | • | | | | | | | | |
| | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |
|) 💅 | File: "/tmp/wiresha | rk_pcapng_ethO_ | _2 Packe | ets: 5487 · Displayed: | 260 (4.7%) · Dropped: (| 0.0%) | | | | | | Profile: Default | | |



9. Let's try now to find out what are those packets contain by following one of the conversations (also called network flows), select one of the packets and press the right mouse button (if you are on a Mac use the command button and click), you should see something similar to the screen below:

| | *ethO [Wireshark | k 1.12.6 (Git Rev Unknown from unknown)] | 8 8 |
|--|--|--|--|
| File Edit View Go Capture Analyze Statistics Tele | ohony Tools Internals Help | | |
| ⊕ ● ▲ ■ △ □ □ × ○ < + + | .J 🛧 🛨 📃 🕎 @ @ | २ ०. 🖭 📓 🕅 🔚 🙃 ? | |
| Filter: dns | - Expression Clear App | ly Save | |
| No. Time Source Desti 4811 26.12442700 172.16.108.2 172. 4814 26.125644900 172.16.108.2 172. 4822 26.1319000 172.16.108.152 172. 4823 26.13129600 172.16.108.152 172. | Ination Protocol Le 16,108.152 DNS DNS DNS 16 Mark Packet (toggle) DNS DNS | ength Info 219 Standard query response 0xd529 CNAME insights-1411736383.eu-west 386 Standard query response 0x2bd1 CNAME insights-1411736383.eu-west itandard query 0x2be2 A insights.hotjar.com itandard query 0x7972 AAAA insights.hotjar.com | -1.elb.amazonaws.com -1.elb.amazonaws.com A 5 |
| 4824 26.13378100 172.16.108.2 172. 4826 26.1533560 172.16.108.2 172. 4832 26.2285200 172.16.108.152 172. 4833 26.22845200 172.16.108.152 172. 4834 26.2285200 172.16.108.152 172. 4834 26.22852000 172.16.108.152 172. 4835 26.228512000 172.16.108.152 172. 4836 26.228512000 172.16.108.152 172. 4836 26.228512000 172.16.108.152 172. | Grover acket (toggle) Set Time Reference (toggle) Time Shift Edit Packet Edit Packet Comment | tandard query response 0x7972 ONAME insights-1411736383.eu-west tandard query response 0xbe2 OXAME insights-1411736383.eu-west tandard query 0x2bfd A login.wayne.edu tandard query 0x4566 A Aparents.wayne.edu tandard query 0x426b A parents.wayne.edu tandard query 0x242b AAAA parents.wayne.edu tandard query 0x242b AAAA parents.wayne.edu | -l.elb.amazonaws.com A 5 |
| Frame 4814: 386 bytes on wire (3088 bits), 38 Ethernet II, Src: Vmware_f0:1a:b5 (00:50:56:fn Internet Protocol Version 4, Src: 172.16.108. User Datagram Protocol, Src Port: 53 (53), Ds Domain Name System (response) | 5 2 2 4 5 5 6 7 7 7 7 7 7 7 7 8 7 8 7 8 7 8 7 8 7 8 | <pre>>rface 0 >c:29:6d:7a:35) > 22 (172.16.108.152) ></pre> | |
| | Conversation Filter Colorize Conversation SCTP Follow TCP Stream Follow UDP Stream | > > > | |
| $\begin{array}{cccccccccccccccccccccccccccccccccccc$ | Follow SSL Stream Copy Protocol Preferences Decode As Print Show Packet in New Window 00 of dns-42, | > > | |
| ○ M File: "/tmp/wireshark_pcapng_eth0_2 Packets: 548" | 7 · Displayed: 260 (4.7%) · Dropped: 0 (0 | .0%) | Profile: Default |

Click on Follow UDP Stream, and then you will see following screen.





10. If we close this window and change the filter back to "http.host==www.wayne.edu" and then follow a packet from the list of packets that match that filter, we should get the something similar to the following screens. Note that we click on **Follow TCP Stream** this time.

| • | •• | | | | 📕 Wi-Fi | : en1 | | |
|-------|---|---|--|---|---|--------|-----------------------|---|
| | | 1 | 💻 🗋 🖹 🙆 🔍 👄 🔶 🚞 | ₮ ± 🗔 🔳 | Θ, Θ | | | |
| , h | ttp.hos | st == sustech | | | | | | Expression + |
| 1 | Vireles | s controls are no | at supported in this version of Wireshark. | | | | | 802.11 Preferences |
| No. | | Time | Source | Destination | Protocol | Length | Info | |
| | 1088 | 33.640931 | 192.168.1.100 | 239.255.255.250 | SSDP | 217 | M-SEARCH * HTTP/1.1 | |
| | 1091 | 34.641026 | 192.168.1.100 | 239.255.255.250 | SSDP | 217 | M-SEARCH * HTTP/1.1 | |
| | 1093 | 35.641764 | 192.168.1.100 | 239.255.255.250 | SSDP | 217 | M-SEARCH * HTTP/1.1 | |
| | 1095 | 36.642013 | 192.168.1.100 | 239.255.255.250 | SSDP | 217 | M-SEARCH * HTTP/1.1 | |
| | 5415 | 60.960002 | 192,168,1.100 | Mark/Unmark Packet Ignore/Unignore Packet Set/Unset Time Reference Time Shift Packet Comment Edit Resolved Name Apply as Filter Prepare a Filter Conversation Filter Colorize Conversation SCTP | 新し 第四 第四 第四 一 第四 一 一 第 日 一 一 第 の 第 の 一 第 の 第 の 一 第 の 第 の 一 第 の の 一 の の 第 の の の の | | GET /answer?id=536496 | 56bi=16sohuurl=http%34%2F%2F%www.ifeng.com%2F6rnd=225f93a720_ |
| E B | Frame | 5415: 905 byt | tes on wire (7240 bits). 905 bytes capt | Follow | | | CP Stream | |
| . ⊩ 8 | Ethern | net II, Src: a | a4:83:e7:67:28:18 (a4:83:e7:67:28:18), | Сору | (54:75: | ► 37 g | SSL Stream | |
| | Intern Transm <mark>Hypert</mark> | net Protocol \ nission Contro cext Transfer | Version 4, Src: 192.168.1.100, Dst: 218 ol Protocol, Src Port: 62072 (62072), D Protocol | Protocol Preferences Decode As Show Packet in New Wir | 1, Acka Idow | Le | n: 839 | |

| 1 | 0000 | 54 | 75 | 95 | 37 | 5b | 39 | a4 | 83 | e7 | 67 | 28 | 18 | 08 | 00 | 45 | 00 | Tu.7[9 | .g(E. |
|---|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|----------|
| | 0010 | 03 | 7b | 00 | 00 | 40 | 00 | 40 | 06 | 34 | 1a | с0 | a8 | 01 | 64 | da | 1e | .{@.@. | 4d |
| | 0020 | 67 | 38 | f2 | 78 | 00 | 50 | c7 | dd | af | dd | e7 | 3a | 58 | a4 | 80 | 18 | g8.x.P | :X |
| | 0030 | 08 | 0a | 94 | Зc | 00 | 00 | 01 | 01 | 08 | 0a | 56 | c4 | 44 | Ød | 66 | 3c | < | V.D.f< |
| | 0040 | fa | 26 | 47 | 45 | 54 | 20 | 2f | 61 | 6e | 73 | 77 | 65 | 72 | 3f | 69 | 64 | .&GET ∕a | nswer?id |
| | 0050 | 3d | 35 | 33 | 36 | 34 | 39 | 36 | 26 | 62 | 69 | 3d | 31 | 26 | 73 | 6f | 68 | =536496& | bi=1&soh |
| | 0060 | 75 | 75 | 72 | 6c | 3d | 68 | 74 | 74 | 70 | 25 | 33 | 41 | 25 | 32 | 46 | 25 | uurl=htt | p%3A%2F% |
| | 0070 | 32 | 46 | 77 | 77 | 77 | 2e | 69 | 66 | 65 | 6e | 67 | 2e | 63 | 6f | 6d | 25 | 2Fwww.if | eng.com% |
| | 0080 | 32 | 46 | 26 | 72 | 6e | 64 | 3d | 32 | 32 | 35 | 66 | 39 | 33 | 61 | 37 | 32 | 2F&rnd=2 | 25f93a72 |
| | 0090 | 30 | 39 | 32 | 32 | 64 | 61 | 32 | 26 | 69 | 66 | 3d | 31 | 26 | 77 | 3d | 33 | 0922da2& | if=1&w=3 |
| | 00a0 | 36 | 30 | 26 | 68 | 3d | 31 | 31 | 32 | 26 | 6a | 73 | 3d | 63 | 26 | 7a | 3d | 60&h=112 | &js=c&z= |
| | 00b0 | 38 | 34 | 37 | 61 | 36 | 32 | 34 | 66 | 34 | 33 | 36 | 62 | 38 | 64 | 61 | 33 | 847a624f | 436b8da3 |
| | 00c0 | 26 | 70 | 74 | 3d | 37 | 36 | 36 | 39 | 36 | 26 | 70 | 73 | 3d | 31 | 35 | 36 | &pt=7669 | 6&ps=156 |
| | 00d0 | 37 | 31 | 34 | 32 | 35 | 38 | 35 | 37 | 38 | 37 | 26 | 69 | 74 | 3d | 30 | 26 | 71425857 | 87⁢=0& |
| | | | | | | | | | | | | | | | | | | | |

| %2F&rnd=225f93a720922da2&if=1&w=360&h=112&js=c&z= left=440⊤=1259&op=100&csp=2560.1417&bcl=365.11 | √ww.ifeng.com =847a624f436b8da3&pt=76696&ps=156714258! 12&pof=366.112&fs=1&total=1 HTTP/1.1 | 5787⁢=0&vs=0&ft=0&vt=0& |
|---|---|-------------------------|
| Host: eff.inte.sogou.com | | |
| Connection: keep-alive | | |
| Lache-Control: Max-age=0 User_Agent: Mozilla/5 0 (Macintosh: Intel Mac OS | X 10 14 6) AppleWebKit/537 36 (KHTML | like Gecko) Chrome/ |
| 76.0.3809.100 Safari/537.36 | x 10_14_0/ Appreneukit/557.50 (kinic, | CIRC GEEROY CHIOMEY |
| Accept: */* | | |
| Referer: http://www.ifeng.com/a_if/190312/weicc/t | testv2.html | |
| Accept-Encoding: gzip, deflate | | |
| Accept-Language: en-US,en;q=0.9 | | |
| Cookie: SUID=5DCDD98D566C860A5627C152000A48A2; wu | Jid=AAG40wKoIQAAAAqGGWw4YQYAIAY=; | |
| CAID=015D/8A945F88808CD4D20330B54FE93; SUV=00B22 | 11140x1waaaaaaaaaaa | |
| | | |
| HTTP/1.1 200 OK | | |
| Server: nginx | | |
| Date: Fri, 30 Aug 2019 05:23:07 GMT | | |
| Content-Type: text/plain; charset=UTF-8 | | |
| Content-Length: 0 | | |
| Lonnection: Keep-alive | | |
| Elast | | |
| Accept-Ranges: bytes | | |
| | | |
| client pkt(s), 1 server pkt(s), 1 turn. | | |
| | Show data as ASCII | Stream 144 |
| Entire conversation (1.096 bytes) | | ou ou ou ou |
| Entire conversation (1,096 bytes) | | |
| Entire conversation (1,096 bytes) | | Eind Next |



Questions for the Lab

- 1. Carefully read the lab instructions and finish all tasks above.
- 2. If a packet is highlighted by black, what does it mean for the packet?
- 3. What is the filter command for listing all outgoing http traffic?
- 4. Why does DNS use Follow UDP Stream while HTTP use Follow TCP Stream?
- 5. Using Wireshark to capture the FTP password.

There is a FTP server installed on the Kali Linux VM. You need to use a terminal to log into the server and use Wireshark to capture the password. The username for the FTP server is csc5991-student, and the password is [WSU-csc5991.] without the brackets. You will user the username and password to login the FTP server while Wireshark is running. Note that the FTP server is installed on the localhost, make sure you select the right interface for the capturing. You need to explain to me how you find the password and a screenshot of the password packet. Have fun!