

Packet Sniffing with Wireshark and Tcpdump

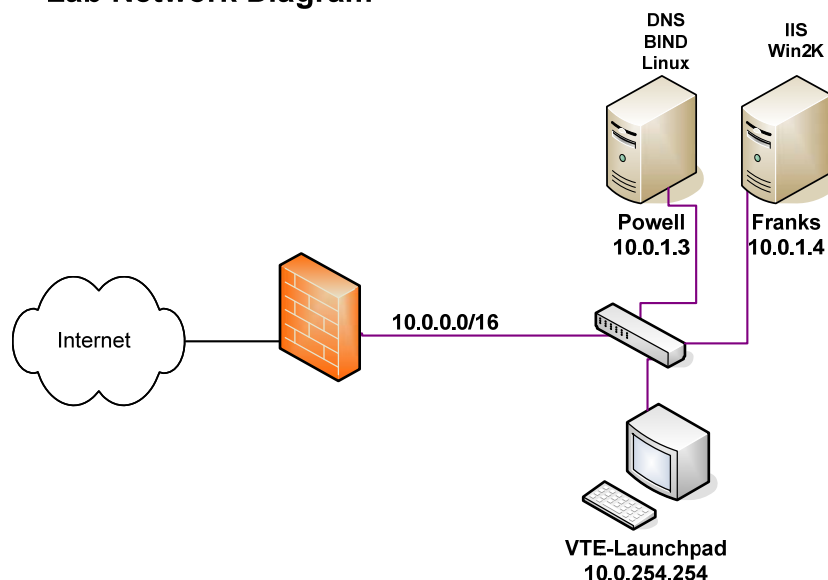
Capturing, or sniffing, network traffic is invaluable for network administrators troubleshooting network problems, security engineers investigating network security issues, developers debugging communication protocol implementations, or anyone trying to learn how their networks work. Because attackers use sniffers for network reconnaissance and to intercept transmitted credentials and data, learning about the capabilities and limitations of packet sniffers is an important facet of understanding the security risks.

In this lab, you will use several popular open-source applications to examine network traffic:

- **Tcpdump** is the most widely used UNIX/Linux tool to record network traffic. It captures packets based on a wide range user-specified criteria, and can save the traffic in different formats. Tcpdump is commonly included in most Linux distributions and can also be obtained from <http://www.tcpdump.org>.
- **Wireshark** is the most widely used graphical application for network monitoring and analysis. It is open-source and runs on most popular computing platforms, including UNIX, Linux, and Windows. It is available for download from <http://www.wireshark.org>.

Your lab environment consists of 3 virtual computer systems:

Lab Network Diagram



1. A Windows 2000 (W2K) web server. This system's hostname is: **Franks** and its IP address is **10.0.1.4**.
2. A Linux system, running Webmin, a web-based Linux administration tool. You will use tcpdump from this host. This system's hostname is: **Powell** and its IP address is **10.0.1.3**.
3. A Windows Server 2003 launchpad system that will allow you to remotely access and analyze the traffic between the servers above. This system's hostname is: **VTE-Launchpad** and its IP address is **10.0.254.254**.

1 Setting up the packet sniffer applications

In determining how to set up a network sniffer, the topography and type of the network are key considerations. In particular, the difference between a switched network and a hub-based network plays a major role in what traffic is visible to the sniffer.

When one host needs to communicate with another system it sends out an Address Resolution Protocol (ARP) broadcast to all hosts on its subnet to determine whether one owns the destination IP address. ARP is an example of *broadcast traffic* that is traffic sent to all hosts on that switch or hub. Only the host with the desired IP address should respond to the ARP request, sending a reply that supplies its network interface's unique MAC address. On a switched network, once communications begin between two hosts, their traffic is isolated by the switch to the physical link between the hosts. However, on a network hub, sometimes termed a repeater hub, the network communications of all systems attached to the hub are copied to each system on the hub. Hubbed networks count on the attached systems ignoring the repeated traffic that isn't addressed to them.

Passive Sniffing places a host's network interface into promiscuous mode, which means it captures everything it sees, including traffic addressed to other hosts. On networks where repeater hubs are used this means capturing all hubbed traffic. Passive sniffing is also possible on some switches that have a SPAN or mirror port; a special port to which all traffic is intentionally copied, by connecting the sniffer to this mirrored port. If the main interest for traffic analysis is traffic entering and exiting the local network, then a passive sniffer positioned in parallel with the network gateway would provide the best insight.

The term *active sniffing* describes alternative tactics to sniff on a switched network. As noted, when a host on network needs to communicate with another, it uses ARP broadcasts to request the correct network address of the destination. By design, only the host with the specified IP address should reply with its MAC address. However, there's no security built into the ARP protocol, allowing another host running an active sniffer application to fake replies to the broadcast, supplying the sniffer's MAC address. An active sniffer using ARP spoofing often tries to saturate the link with its replies to make sure that the real destination's MAC address is discarded or ignored. Using this method, the communications link will be established from the legitimate sender to the host running the sniffing application. After capturing the packets, the active sniffer can then forward them to the legitimate host.

Another active sniffing method is MAC flooding. In this technique, a flooding tool generates a large number of packets with different, spurious MAC addresses. Network switches maintain a table of MAC addresses that map each address to the link that carries its traffic. If a switch is overwhelmed with too many frames to handle at once, or receives so many MAC table entries that they exceed the maximum space allotted to the table, some switches are configured to fail-open. That is, since the switch can no longer reliably determine how to forward packets, it behaves like a repeater hub and passes all traffic to all hosts.

However, because different switches behave differently in response to active sniffing attempts and network performance generally suffers; active sniffing tactics are generally not employed for legitimate network monitoring. In this lab, you will be passively sniffing with freely available Linux and Windows tools. The network you will monitor will behave like a hubbed network.

1.1 Start the Wireshark network analyzer

Wireshark can read capture traffic files from tcpdump, NAI's Sniffer, Sniffer™ Pro, NetXray™, Microsoft's Network Monitor, and many others. It can grab live data over Ethernet, FDDI, PPP, Token-Ring, IEEE 802.11, Classical IP over ATM, and loopback interfaces. Captured network traffic can be reviewed and analyzed via a GUI, or with a text-mode companion program 'tethereal'. Capture files can be programmatically edited or converted, and Wireshark currently knows about and can dissect hundreds of network protocols. Wireshark filters and parses traffic captures and can save output in various formats.

1. From the Desktop of your VTE-Launchpad system, double click the Wireshark icon.
2. On the toolbar menu select 'Capture', and then 'Interfaces'.
 - a. Select the button labeled 'Options' on the same line as the interface with the IP address 10.0.254.254
 - b. In the resulting window, ensure the 'Capture packets in promiscuous mode' option is checked
 - c. In the Name Resolution options, deselect the option to 'Enable MAC name resolution'
 - d. Clear the text box next to the 'Capture Filter:' button.
 - e. Accept all other default options and click the 'Start' button.
 - f. Minimize the capture window and the Wireshark application

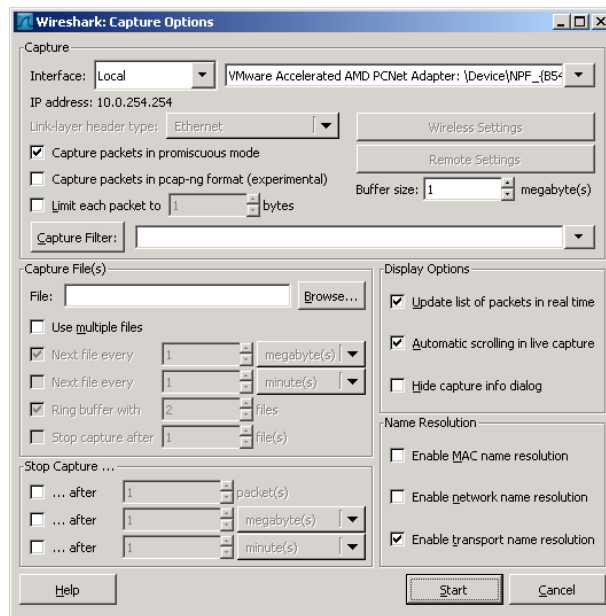


Figure 1: Wireshark Options

1.2 Start capturing with tcpdump

1. From the Desktop of your VTE-Launchpad system, double click the Putty SSH client icon.
2. Enter **10.0.1.3** for the Host IP address, and click the 'Open' button
3. Click 'Yes', on the security alert
4. Log in with username: **root**, password: **tartans**

5. Type `ifconfig` and hit [Enter]. This will display information about your network adapters, including IP address and system name. You will listen on the Ethernet interface, `eth0`.
6. To start `tcpdump` on this interface type:

```
# tcpdump -i eth0 -s 1500 -w capture.txt
```

The target interface is designated by the '-i' option. `Tcpdump` by default sends its output to standard output, printing it to screen. With the '-w' option, you can specify the captured traffic be saved to file instead. The '-s' option indicates to Wireshark to store the first 1500 bytes of each packet, instead of the default first 256 bytes of each packet. For more information about `tcpdump` options and configuration, see either the application's man pages (`man tcpdump`) or, for a summary of command line usage, enter `tcpdump --help`.

7. Minimize the Putty SSH session window.

2 Generating sample network traffic

2.1 Unencrypted web browsing

1. From the start menu of your VTE-Launchpad system, launch Internet Explorer.
2. Enter Franks IP address following URL: <http://10.0.1.4> in the address bar
 - a. Navigate to each of the four linked pages ('News', 'References', 'Leaders', and 'Careers').



Figure 2: Franks webpage

2.2 Encrypted web browsing

1. Access the Webmin application on **Powell** at the following URL: <https://10.0.1.3:10000> (note the 'https', which signifies the SSL protocol)
2. Click 'OK' on the security alert.
3. Click 'Yes' on the security alert for the SSL certificate.
4. Uncheck the 'Continue to prompt when Web site content is blocked' on the Internet Explorer Enhanced Security Configuration, and click 'Close'.
5. Log in with the username: **root** password: **tartans**, click the login button.

6. Decline the auto complete prompt.
7. Browse around several of the Webmin tabs ('System', 'Servers', 'Networking', etc.) and then log out by selecting the log out option in the top right.

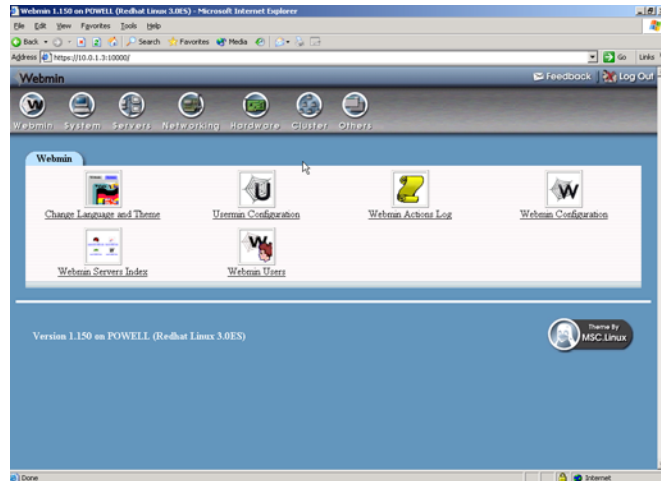


Figure 3: Powell Webmin

8. Close Internet Explorer.

2.3 Telnet session traffic

Attackers often use sniffers to harvest usernames and passwords transmitted without encryption over the network. It's for this reason that authentication for network services should be encrypted, as should remote administration sessions. The risks are well demonstrated by capturing traffic from an unencrypted Telnet session.

1. First, you'll create a new user account for the Telnet session. From the Start menu, select 'Administrative Tools > Computer Management'.
2. In the computer management interface, navigate to the 'Local Users and Groups' section in the left pane and select 'Users'. Right-click in the right pane and select 'New User ...' from the context menu.

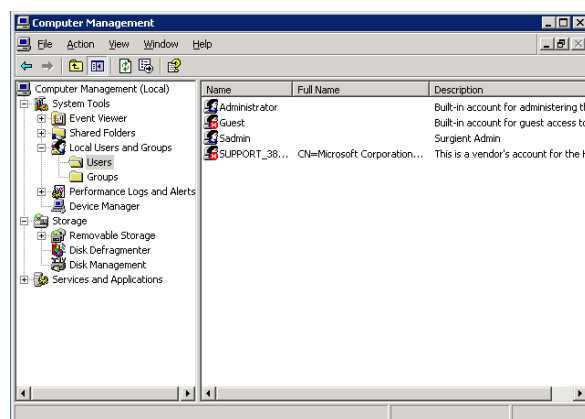


Figure 4: User Management

3. Create a user with the name 'telnet' and password 'tartans'. De-select the box requiring a password change on the next logon. Click 'Create'.
4. Click 'Close' to close the New User dialog.

5. Now you need to make the user a member of the TelnetClients group to permit access to the system via telnet. Once the user is created, double-click the 'telnet' user and, in the 'Properties' box, select the 'Member Of ...' tab. Click the 'Add...' button and type telnetclients into the entry box.
6. Click 'OK' to add the user to the group and click 'OK' again to close the user properties window.
7. Close the 'Computer Management' window.
8. Start the Telnet service on the VTE-Launchpad. From the Start menu, select 'Administrative Tools > Services'.

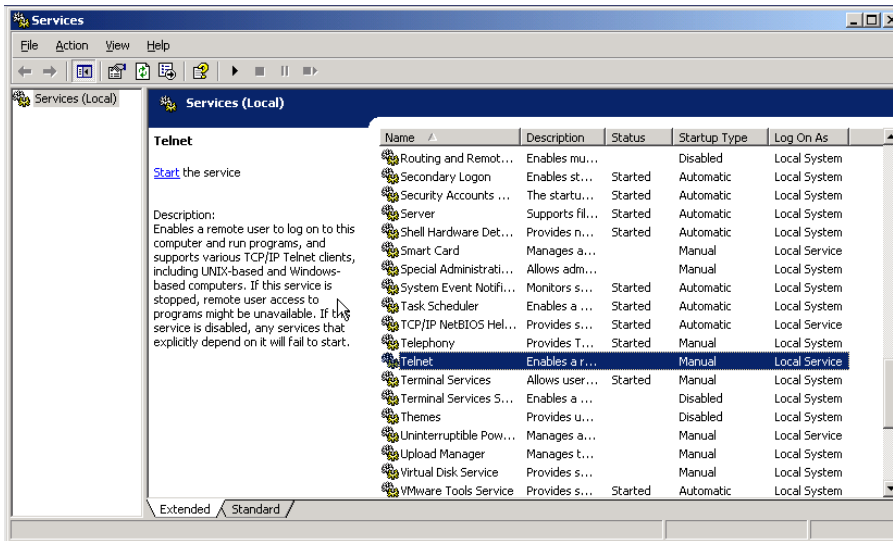


Figure 5: Service Administration

9. Scroll down to the Telnet service entry and double-click it. In the resulting window, select 'Manual' start-up for Telnet and click 'Apply'. Then click the 'Start' button for the service.
10. Click 'OK' and then close the services window, after confirming that the Telnet service is listed as running.

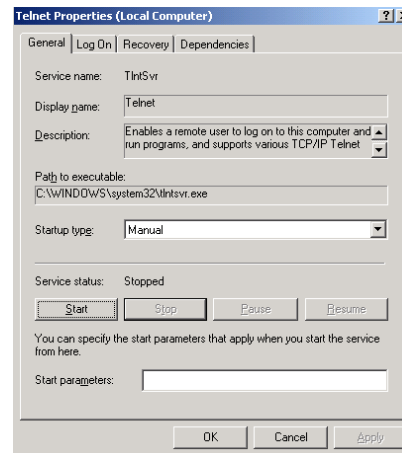


Figure 6: Starting Telnet

11. Click on the 'Putty SSH client' icon to open a new SSH client configuration screen. Do not restore the window of the existing Putty connection to Powell.
12. Enter Powell's IP address, **10.0.1.3** and establish a new SSH session with the Username: **root** and Password: **tartans**.
13. From Powell, telnet back to the VTE-Launchpad.

```
# telnet 10.0.254.254
```

- Wait for the prompt, and then log in with username: **telnet** and password: **tartans**. Then list the contents of the C:\ directory with 'dir C:'

```
# dir C:
```

```

root@POWELL:~
login as: root
Sent username "root"
root@10.0.1.3's password:
Last login: Sat Jul 11 00:52:58 2009 from 10.0.254.254
[root@POWELL root]# telnet 10.0.254.254
Trying 10.0.254.254...
Connected to 10.0.254.254 (10.0.254.254).
Escape character is '^'.
Welcome to Microsoft Telnet Service

login: telnet
password:

*=====
Welcome to Microsoft Telnet Server.
*=====

C:\Documents and Settings\telnet>dir C:
Volume in drive C has no label.
Volume Serial Number is 34A8-599E

Directory of C:\Documents and Settings\telnet

07/11/2009  01:54 AM    <DIR>          .
07/11/2009  01:54 AM    <DIR>          ..
04/01/2005  10:28 AM    <DIR>          Desktop
04/01/2005  10:28 AM    <DIR>          Favorites
04/01/2005  10:28 AM    <DIR>          My Documents
04/01/2005  10:28 AM    <DIR>          Start Menu
04/01/2005  10:29 AM                0 Stl_Trace.log
                1 File(s)              0 bytes
                6 Dir(s)         513,855,488 bytes free

C:\Documents and Settings\telnet>

```

Figure 7: Telnet session

- Quit the telnet session by typing `exit`. At the Powell command prompt, type `exit` again to quit the SSH session.

```
# exit
```

3 Terminate the packet sniffers

3.1 Stop Wireshark

1. Open the Wireshark Capture window and click the 'Stop' button.

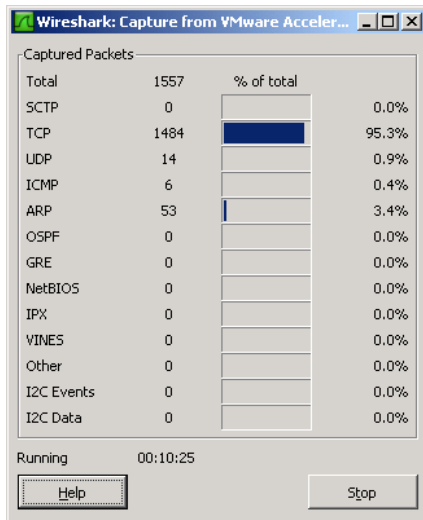


Figure 8: Wireshark capture dialog

3.2 Stop tcpdump

1. Restore the minimized Putty SSH session with Powell, and press [Ctrl-C] to stop the capture. You will see a status window similar to Figure 8.

```
[root@POWELL root]# tcpdump -i eth0 -tx tcp -w capture.txt
tcpdump: listening on eth0

1344 packets received by filter
0 packets dropped by kernel
[root@POWELL root]#
```

Figure 9: Tcpdump packet capture

4 Analysis with tcpdump

With tcpdump, you can use expressions to match packets against a filter. Every field of a packet can be used as a filter parameter, and you can use expressions when starting a capture or when replaying a capture file. Tcpdump expressions consist of one or more of these qualifiers:

- type* Allows you to filter based on the addressing/ports within the packet. **host**, **net**, and **port** are all allowed keywords. Without any keywords, **host** is assumed.
- dir* Allows you to narrow down the above type to the source or destination field in the packet. The allowed options are **src**, **dst**, **src or dst**, and **src and dst**. If no direction is given, **src or dst** is assumed.
- proto* Restricts the capture to a particular protocol. Some of the allowed protocols are **ip**, **ip6** (IPV6), **arp**, **tcp** and **udp**. If no protocol is given, all protocols matching the above type and direction are displayed/captured.

Also, Boolean operators such as **not**, **and** and **or** can be used to further refine a filter. Follow the examples below to see how this works.

4.1 Replay the capture file

1. If you wanted to replay the entire file of captured network traffic, you could use the command line:

```
# tcpdump -r capture.txt
```

2. All of the tcpdump filtering options available while capturing traffic, can be applied to filter traffic replayed from a captured traffic file. Experiment with the following commands (note that you will use the “-tnn” options in each replay of the packet capture file. ‘-t’ means don’t show timestamps, and “nn” means don’t translate IP addresses or port numbers to names):

```
tcpdump -tnn -r capture.txt arp
```

This shows the ARP broadcast traffic from the network’s three hosts.

```
tcpdump -tnn -r capture.txt tcp port 10000
```

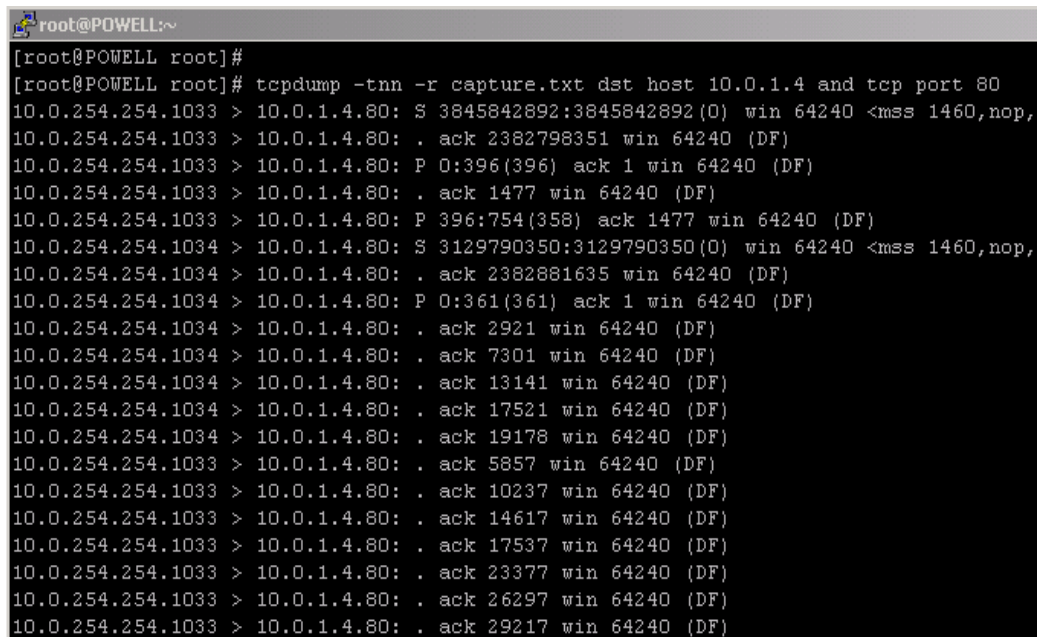
All TCP traffic with a source or destination port of 10000 will be displayed.

```
tcpdump -tnn -r capture.txt dst host 10.0.1.4
```

All packets captured that had a destination address of 10.0.1.4 will be displayed.

```
tcpdump -tnn -r capture.txt dst host 10.0.1.4 and tcp port 80
```

All packets captured that had a destination address of 10.0.1.4 and had TCP port 80 (the HTTP port) specified will be displayed, similar to below:



```

root@POWELL:~
[root@POWELL root]#
[root@POWELL root]# tcpdump -tnn -r capture.txt dst host 10.0.1.4 and tcp port 80
10.0.254.254.1033 > 10.0.1.4.80: S 3845842892:3845842892(0) win 64240 <mss 1460,nop,
10.0.254.254.1033 > 10.0.1.4.80: . ack 2382798351 win 64240 (DF)
10.0.254.254.1033 > 10.0.1.4.80: P 0:396(396) ack 1 win 64240 (DF)
10.0.254.254.1033 > 10.0.1.4.80: . ack 1477 win 64240 (DF)
10.0.254.254.1033 > 10.0.1.4.80: P 396:754(358) ack 1477 win 64240 (DF)
10.0.254.254.1034 > 10.0.1.4.80: S 3129790350:3129790350(0) win 64240 <mss 1460,nop,
10.0.254.254.1034 > 10.0.1.4.80: . ack 2382881635 win 64240 (DF)
10.0.254.254.1034 > 10.0.1.4.80: P 0:361(361) ack 1 win 64240 (DF)
10.0.254.254.1034 > 10.0.1.4.80: . ack 2921 win 64240 (DF)
10.0.254.254.1034 > 10.0.1.4.80: . ack 7301 win 64240 (DF)
10.0.254.254.1034 > 10.0.1.4.80: . ack 13141 win 64240 (DF)
10.0.254.254.1034 > 10.0.1.4.80: . ack 17521 win 64240 (DF)
10.0.254.254.1034 > 10.0.1.4.80: . ack 19178 win 64240 (DF)
10.0.254.254.1033 > 10.0.1.4.80: . ack 5857 win 64240 (DF)
10.0.254.254.1033 > 10.0.1.4.80: . ack 10237 win 64240 (DF)
10.0.254.254.1033 > 10.0.1.4.80: . ack 14617 win 64240 (DF)
10.0.254.254.1033 > 10.0.1.4.80: . ack 17537 win 64240 (DF)
10.0.254.254.1033 > 10.0.1.4.80: . ack 23377 win 64240 (DF)
10.0.254.254.1033 > 10.0.1.4.80: . ack 26297 win 64240 (DF)
10.0.254.254.1033 > 10.0.1.4.80: . ack 29217 win 64240 (DF)

```

Figure 10: tcpdump filtered output

```
tcpdump -tnn -r capture.txt not \(dst host 10.0.1.4 and tcp port 80\)
```

This expression is the opposite of the one before it. All packets **not** having a destination address of 10.0.1.4 and TCP port 80 (the HTTP port) specified will

be displayed. The backslash is required so that the Bash shell will not misinterpret the parentheses characters.

```
tcpdump -tnn -r capture.txt dst host \(10.0.1.3 or 10.0.1.4\)
```

This expression will capture all packets destined for either of two hosts, 10.0.1.3 or 10.0.1.4.

3. By default, tcpdump displays basic information about each packet that it captures. You can increase the information you see with the -v and -vv options. Try this command:

```
tcpdump -vv capture.txt
```

This will print the time to live, identification, total length and options in each packet. If you add the -S option, you can see the output in hex along with ASCII.

4. Note that the traffic displayed consists of header information and packet metadata, as opposed to the full contents of the transmitted packets. By default, Tcpdump captures just the first 68 bytes of the packet. Using the '-s 0' option has tcpdump record the full packet contents.
5. Type `exit` to close the SSH session.

5 Analysis with Wireshark

5.1 Working with capture data

1. Restore Wireshark. To save the current capture file, select 'File', 'Save As'. Type `c:\capture.txt` in the name text box, and click the 'Save' button.

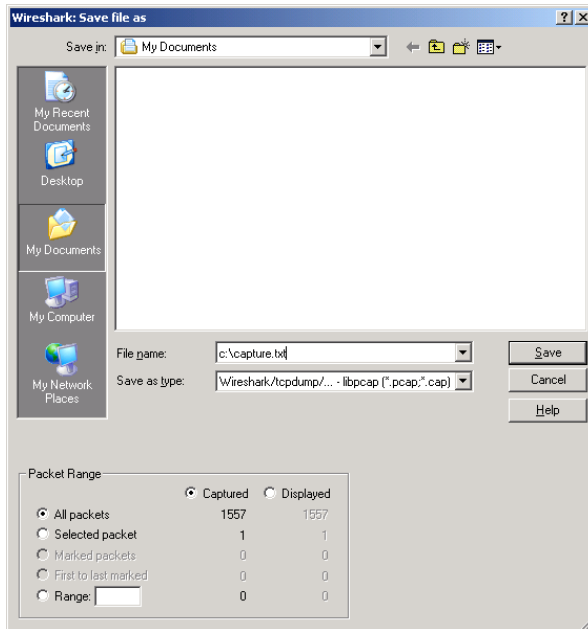


Figure 11: Saving a file in Wireshark

2. The default view in Wireshark splits the window into three panes: a Packet List, Packet Details, and Packet Bytes. This view can be customized from the View menu bar.

- Statistics reports can be generated from the Statistics menu, including summary information and conversation totals.

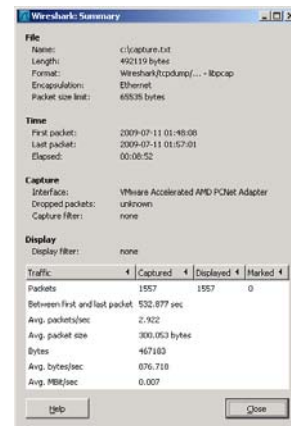


Figure 12: Wireshark summary statistics

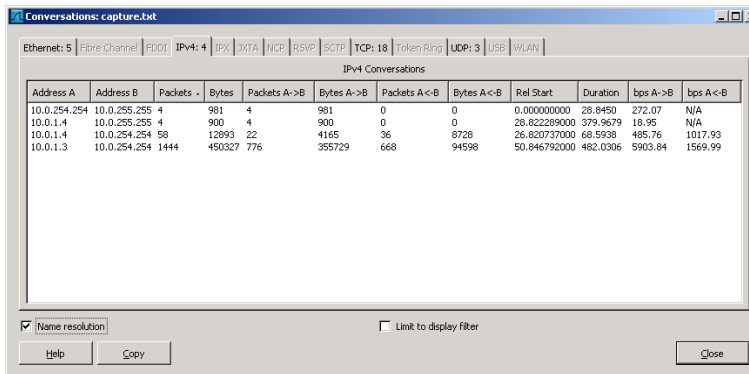


Figure 13: Wireshark conversation statistics

- In the top pane, the capture data can be sorted by selecting any one of the display columns including source, destination, and protocol.

5.2 Filtering and reassembling traffic

- A single TCP communication session can automatically be filtered and reassembled, including its contents, using the 'Follow TCP Stream' option. Find one of the http requests for '10.0.1.4'. Right click on that row and select 'Follow TCP Stream'.

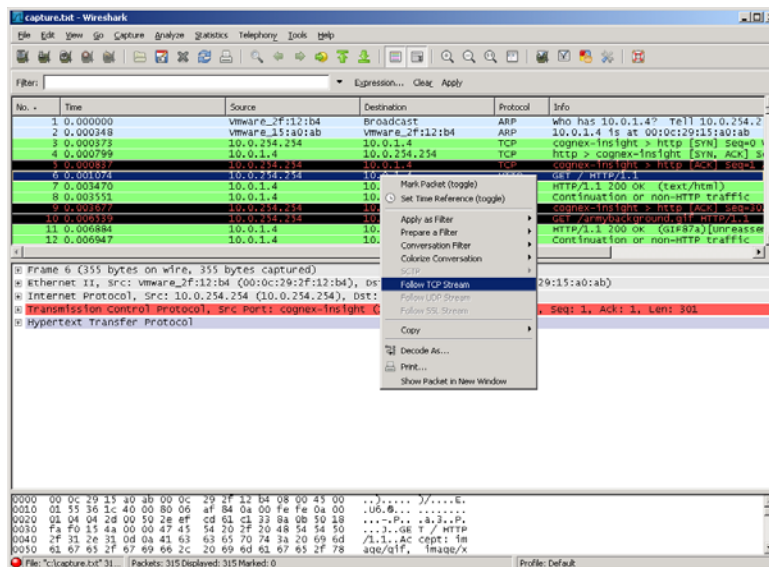


Figure 14: Follow TCP Stream

You will see that the entire http session is displayed, including the http commands, server responses, and even the images transmitted.

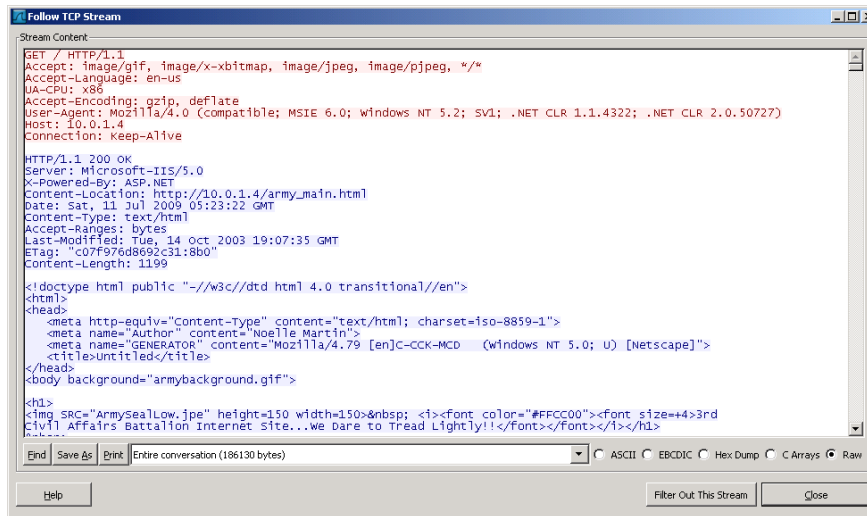


Figure 15: HTTP TCP Stream

2. Close the TCP Follow Stream window, and then clear the filter on the main Wireshark window by clicking the 'Clear' button.

Note: It is necessary to click the clear button each time after you examine the captured data with filters. This ensures that each new filter is then applied to all the packets.

3. Find the SSH traffic and the HTTPS traffic going to 10.0.1.3 port 10000. Perform the same steps to follow the TCP stream. Notice how the encryption of each session protects the packet contents.

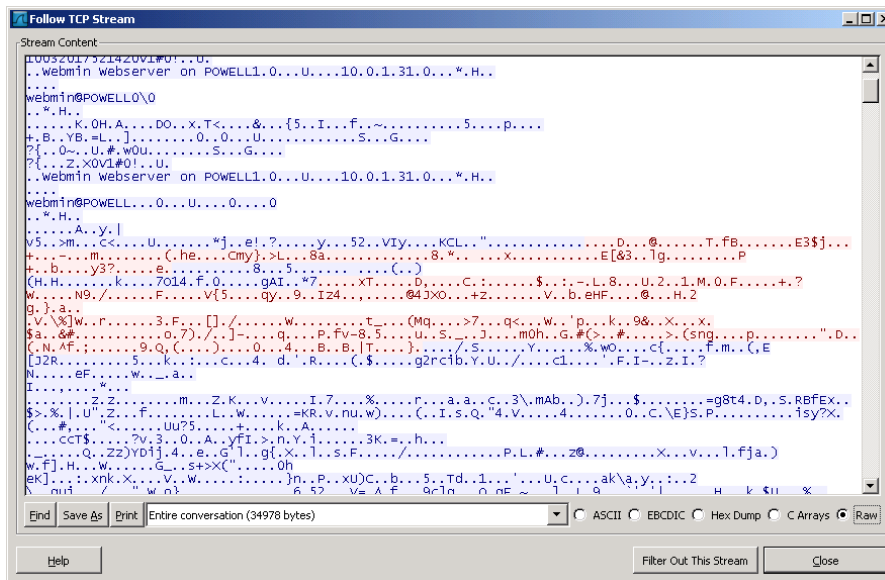


Figure 16: HTTPS TCP Stream

5.3 Building filter expressions

Wireshark implements a powerful and flexible filtering language that allows the user to isolate packets by a range of criteria. In the filter bar, the Expression button allows you build a filtering expression in the GUI interface.

1. To filter out traffic to Franks, click the 'Expression' button. Find the 'IP' variables. Select 'ip.dst', the double equals sign in the relation window (==), and then specify Frank's IP address **10.0.1.4**. Click 'OK'. [Figure 17]

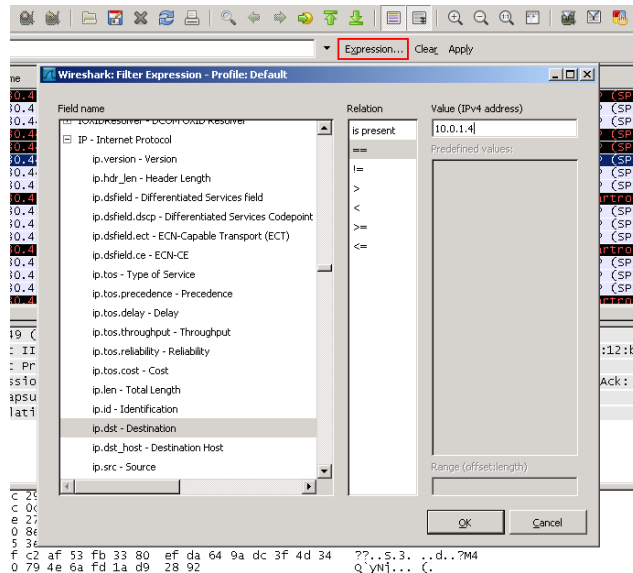


Figure 17: Building a filter expression in Wireshark

2. The expression has now been created for you. To apply the filter, click 'Apply' next to the 'Filter:' expression box. You can add other expressions manually to this box, if desired. Clear the filter when you are done so that you are viewing all the captured traffic.
3. To build an expression that captures the Telnet traffic between Powell and VTE-Launchpad, type the following into the Filter expression box: `ip.addr == 10.0.1.3 and telnet`. Click 'Apply'. This captures any traffic to or from Powell's IP address with the corresponding destination or source port the standard telnet port, tcp port 23.
4. By expanding the telnet protocol dissector in the middle pane, you can review the contents of the telnet session. If you examine the first few packets, you can see the handshaking process as the telnet session was set up.

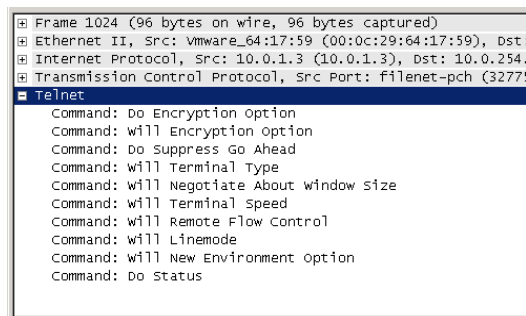


Figure 18: Telnet session set-up

- In the top pane, right-click on a packet in the telnet conversation and select 'Follow TCP Stream'. You will see the full contents of the reassembled clear-text telnet session, including the login credentials for the 'telnet' account, which underscores the value of encrypting the authentication process and other sensitive traffic.
- You can mix and match filter expressions and use different operators, such as '!=' (not equal to). For example, 'ip.dst != 10.0.254.254 and ip.src != 10.0.254.254' would filter out all tcp and udp traffic to or from the VTE-Launchpad, leaving traffic between other hosts. (Mainly broadcast traffic in this lab.)
- Wireshark can filter on its large range of known protocols. For example, specifying the Filter expression 'msnms or aim' will isolate Microsoft Messenger and AOL Instant Messenger traffic, if any were captured.

5.4 Decoding traffic carried over non-standard ports

Wireshark also allows you to parse and analyze traffic carried on non-standard ports for the protocol. For example, Powell's Webmin server uses the SSL protocol to encrypt its traffic, but the server listens on tcp port 10000, rather than the standard HTTPS port of 443. To provide more information about the SSL session, we can ask Wireshark to treat traffic to port 10000 as SSL traffic.

- Enter `tcp.port == 10000` in the Filter expression box to isolate the Webmin traffic. Apply the filter, then right click on a packet in the top pane. In the context menu, select 'Decode as ...' and in the 'Transport' tab specify destination port 10000 and scroll down to the SSL protocol in the selection frame.

- Click 'OK' to configure Wireshark to view this traffic as SSL traffic. Back in the top pane, you now see additional protocol information about the traffic, highlighting handshake sessions for example. (Of course, the session data is still encrypted.) Click on the '+' to expand the 'Secure Sockets Layer' (SSL) protocol dissector in the middle pane for more details about the session.

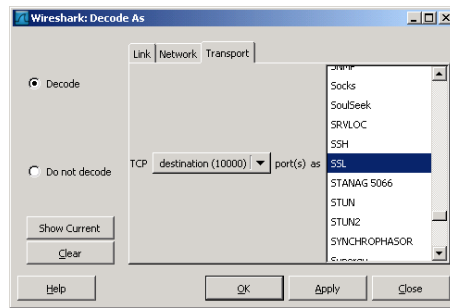


Figure 19: Wireshark Decode selection

- The 'Decode as ...' function is useful for analyzing traffic that has been redirected over non-traditional ports. This might occur, for example, if an attacker uses the HTTP tcp port 80 to carry a telnet or SSH command channel.

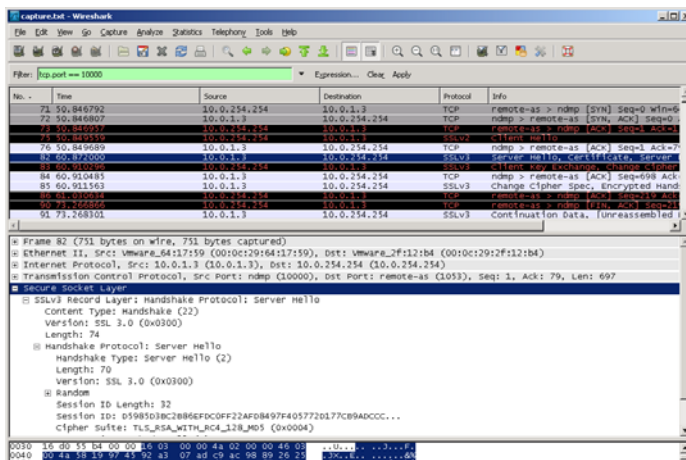


Figure 20: Applying the SSL decode to Webmin traffic

5.5 Extracting captured files using TCP Reassembly

Wireshark can reconstruct transferred files from an HTTP stream (and about 19 other application layer protocols). This can be useful in forensic situations when the client and server systems are not available.

1. In Wireshark, select 'Edit > Preferences'. Expand 'Protocols' in the left pane and scroll down to select TCP. Ensure the 'Allow subdissector to reassemble TCP streams:' is checked.

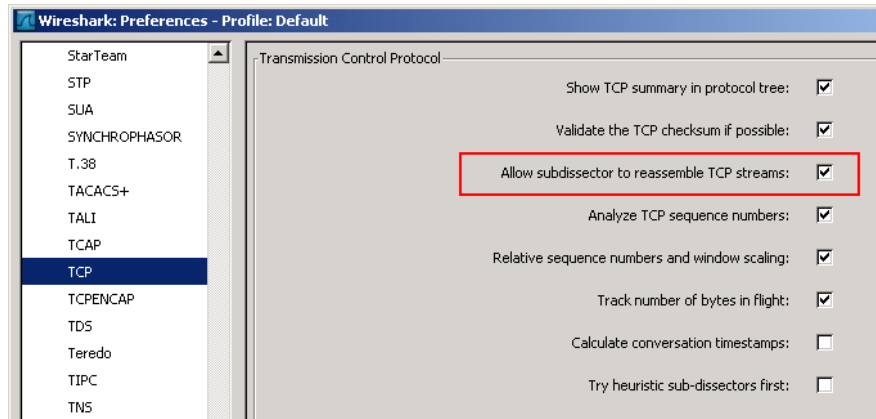


Figure 21: Transmission Control Protocol Preferences

2. Select HTTP in the left pane. Ensure the top two options are checked.

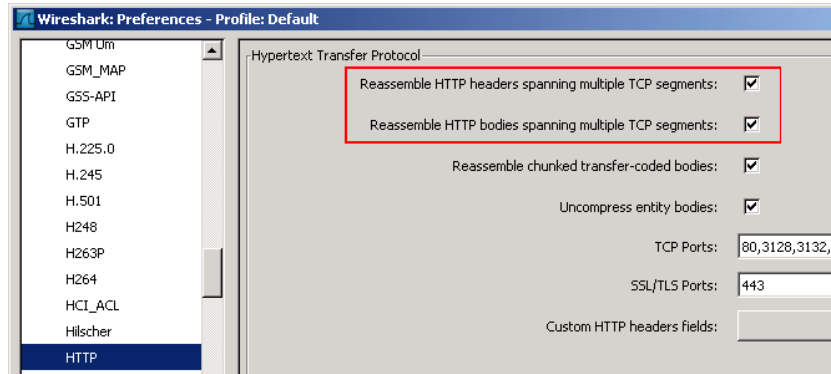


Figure 22: Hypertext Transfer Protocol Preferences

3. Now search for all JPEG files in the current capture session. Type 'http.content_type contains "jpeg"' into the Filter box and click 'Apply'.

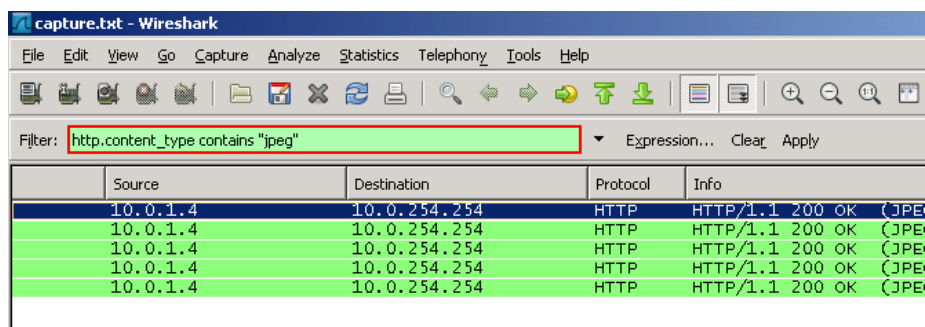


Figure 23: Search for all JPEG HTTP file transfers

4. Select the first packet. Right click on 'JPEG File Interchange Format' in the middle pane and select 'Export Selected Packet Bytes...'

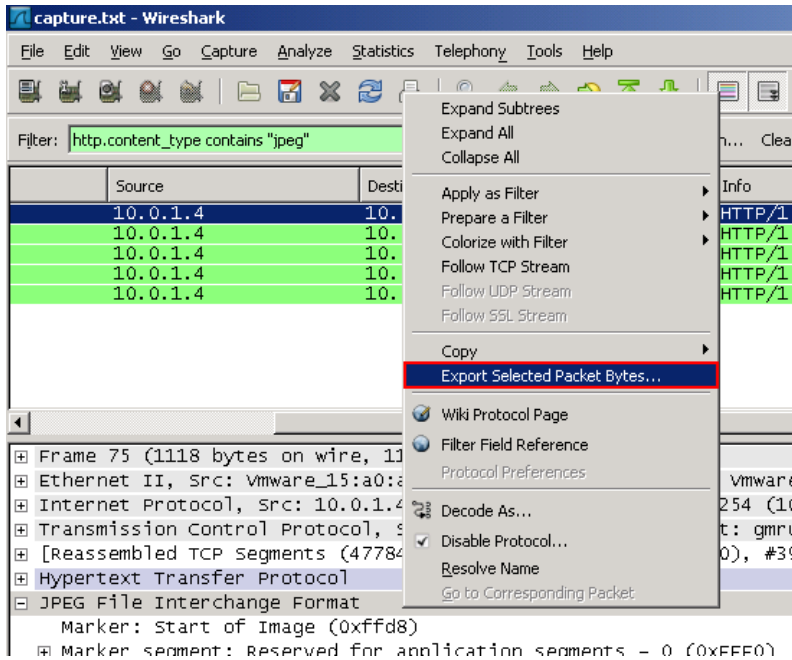


Figure 24: Export JPEG packet bytes to a file

5. Save the file on the Desktop as 'image.jpg' and open it. You should see one of the images from the Franks web server.



6 Introduction to Windump

WinDump is the Windows-platform version of tcpdump. WinDump is fully compatible with tcpdump and can be used to watch and diagnose network traffic according to a flexible set of rules. It can run under Windows 95/98/ME, and under Windows NT/2000/XP. WinDump and the Windows packet capture library on which it depends, WinPcap, are freely downloadable from <http://www.winpcap.org>.