T2-5 Advanced Capture and Display Filtering
April 1, 2008

Tony Fortunato
Sr Network Specialist | The Technology Firm

SHARK FEST '08
Foothill College
March 31 - April 2, 2008
About your Presenter

Tony Fortunato, Sr Network Specialist, The Technology Firm

Certified Fluke Networks and Wireshark Instructor

Website: www.thetechfirm.com

A Senior Network Specialist with experience in performance testing, network design, implementation, and troubleshooting LAN/WAN/Wireless networks, desktops and servers since 1989.

Tony has taught at Colleges/Universities, Networld/Interop and many onsite corporate settings to thousands of analysts.

Tony is an authorized and certified Fluke Networks and Wireshark Instructor. His Pine Mountain Group CNA Level I and II certification demonstrates his vendor neutral approach to network design, support and implementations.

Tony has architected, installed and supported various types of Residential Wireless High Speed as well as hundreds of WIFI hotspots. Tony combines custom programs, open source and commercial software to ensure a simple support infrastructure.

Tony works on networks from 2 to 120,000 nodes and specializes in post installation performance/design review. This process involves using various tools (Protocol analyzers, traffic generators and network management) and working on multi-vendor equipment (switches, routers, servers, etc).

Tony works at customer sites within a range of capacities from project management, network design, consulting, troubleshooting, designing customized courses and assisting with installing physical equipment.
Capturing Traffic

Capture Engine

Capture Filters

Winpcap – Airpcap - Libpcap

Network
Options

When capturing with Wireshark, you have 2 options;

- Using the GUI
- Using the command prompt and `tshark`
  - `tshark -D`
  - `tshark -i`
Capture Filters

- Based on the tcpdump format
- Location identified in Help -> About under Folders Tab


- Make sure you use a text editor, or save in a text format
- Ensure you have a blank line at the end of the file
- Good idea to create a header and indent the related filters

```
"MAC FILTERS" HEADER
  Mac Address" ether host 00:11:95:2f:bf:cc
  "Ethernet Source First 3 Bytes" ether.src [6 :3] == 00:11:95
"IP FILTERS" HEADER
  "IP" h
  IP#1 host 192.168.20.32
  Host " host 10.10.1.1
  "Source Host" src host 10.10.1.1
  "Destination Host" dst host 10.10.1.1
"BROADCAST & MULTICAST FILTERS" HEADER
  IP Multicast" ip multicast
  "Ethernet Multicast" ether multicast
  "Broadcast" ether broadcast
"SPECIFIC ETHER TYPE" HEADER
  "ARP" ether proto 0806
  "RARP" ether proto 8035
  "Apple Talk" atalk
  IP" ether proto 0800
  "IP ver 6" ip6
```
# Capture Filter Reference

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ether host</strong> MAC address</td>
<td>Capture all packets to and from a MAC address</td>
</tr>
</tbody>
</table>

## IP Filters

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>host ip address</td>
<td>Capture all packets to and from an ip address</td>
</tr>
<tr>
<td>src host ip address</td>
<td>Capture all packets from an ip address</td>
</tr>
<tr>
<td>dst host ip address</td>
<td>Capture all packets to an ip address</td>
</tr>
</tbody>
</table>

## TCP/UDP Filters

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>port port</td>
<td>Capture all packets to and from a port number</td>
</tr>
<tr>
<td>src port port</td>
<td>Capture all packets from a port number</td>
</tr>
<tr>
<td>dst port port</td>
<td>Capture all packets to a port number</td>
</tr>
</tbody>
</table>

## IP Network Filters

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>net net</td>
<td>Capture all packets to and from a subnet</td>
</tr>
<tr>
<td>src net net</td>
<td>Capture all packets from a subnet</td>
</tr>
<tr>
<td>dst net net</td>
<td>Capture all packets to a subnet</td>
</tr>
</tbody>
</table>
## Capture Filter Examples

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ether host 00:15:c5:37:40:60</strong></td>
<td>Capture all packets to and from MAC 00:15:c5:37:40:60</td>
</tr>
<tr>
<td><strong>host 10.44.10.1</strong></td>
<td>Capture all packets to and from 10.44.10.1</td>
</tr>
<tr>
<td><strong>host <a href="http://www.wireshark.org">www.wireshark.org</a></strong></td>
<td>Capture all packets from <a href="http://www.wireshark.org">www.wireshark.org</a></td>
</tr>
<tr>
<td><strong>TCP/UDP Filters</strong></td>
<td></td>
</tr>
<tr>
<td><strong>port 80</strong></td>
<td>Capture all packets to and from TCP/UDP port number 80</td>
</tr>
<tr>
<td><strong>portrange 67-68</strong></td>
<td>Capture all DHCP bootps/bootpc</td>
</tr>
<tr>
<td><strong>port http</strong></td>
<td>Capture all packets from devices using http</td>
</tr>
<tr>
<td><strong>tcp portrange 1200-2000</strong></td>
<td>Capture all packets with TCP port # 1200-2000</td>
</tr>
<tr>
<td><strong>IP Network Filters</strong></td>
<td></td>
</tr>
<tr>
<td><strong>net 10.44.10</strong></td>
<td>Capture all packets to and from a subnet 10.44.10</td>
</tr>
<tr>
<td><strong>arp</strong></td>
<td>Capture all arp packets</td>
</tr>
<tr>
<td><strong>udp</strong></td>
<td>Capture all udp packets</td>
</tr>
<tr>
<td><strong>tcp</strong></td>
<td>Capture all tcp packets</td>
</tr>
</tbody>
</table>
Supported Capture Protocols

- arp Address Resolution Protocol
- esp Encapsulating Security Payload
- icmp Internet Control Message Protocol
- icmp6 Internet Control Message Protocol, for IPv6
- igmp Internet Group Management Protocol
- igrp Interior Gateway Routing Protocol
- ip Internet Protocol
- ip6 Internet Protocol version 6
- pim Protocol Independent Multicast
- rarp Reverse Address Resolution Protocol
- stp Spanning Tree Protocol
- tcp Transmission Control Protocol
- udp User Datagram Protocol
- vrrp Virtual Router Redundancy Protocol
Data Pattern Offsets

To retrieve a single byte from a packet, use square brackets to indicate the offset of that byte from the beginning of a particular protocol. Offsets start at zero (e.g., tcp[0] gives the first byte in the TCP header and tcp[1] gives the second byte)

<table>
<thead>
<tr>
<th>TCP Header Layout</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</td>
</tr>
<tr>
<td>16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</td>
</tr>
<tr>
<td>16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</td>
</tr>
<tr>
<td>16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</td>
</tr>
<tr>
<td>16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</td>
</tr>
<tr>
<td>16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</td>
</tr>
<tr>
<td>16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</td>
</tr>
<tr>
<td>16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</td>
</tr>
<tr>
<td>16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</td>
</tr>
<tr>
<td>16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32</td>
</tr>
</tbody>
</table>
## HTTP Get Offset Example

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Length</th>
<th>Source IP Address</th>
<th>Destination IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>2932</td>
<td>01:26:19</td>
<td>367</td>
<td>69.181.134.143</td>
<td>63.147.175.35</td>
</tr>
<tr>
<td>2933</td>
<td>01:26:19</td>
<td>373</td>
<td>69.181.134.143</td>
<td>63.147.175.35</td>
</tr>
</tbody>
</table>

**TCP Offset 20**

**4 Bytes of Data**

```
0000 00 01 5c 22 a5 82 08 00 46 f4 3a 09 08 00 45 00
0010 01 62 68 61 40 00 40 06 16 3a 45 b5 86 8f 3f 93
0020 af 3c 99 00 50 f9 98 ce 5c 32 09 46 22 50 18
0030 0f 34 45 00 00 47 45 54 20 01 6f 70 68 74 6f 66 69 73 74 65 72
0040 02 32 30 30 30 30 32 31 31 31 31 34 2f 6e 66 69 73 69 6e 67 75 72
```
Overview of Display Filters

- Can visit http://www.wireshark.org/docs/dfref
Display Filter Syntax

Visit www.wireshark.org for the master list of Display Filter field names, types, descriptions and versions

Display Filter Reference

Wireshark’s most powerful feature is its vast array of display filters (over 51000 as of version 0.99.5). They let you drill down to the exact traffic you want to see and are the basis of many of Wireshark’s other features, such as the coloring rules.

This is a reference. If you need help using display filters, please see the wireshark-filter and the User’s Guide.

Index

2 3 4 5 6 7 8 9 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Display Filter Reference: Transmission Control Protocol

Protocol field name: tcp
Versions: 0.10.0 to 0.99.5
Back to Display Filter Reference

<table>
<thead>
<tr>
<th>Field name</th>
<th>Type</th>
<th>Description</th>
<th>Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcp.ack</td>
<td>32-bit integer</td>
<td>Acknowledgement number</td>
<td>0.10.0 to 0.99.5</td>
</tr>
<tr>
<td>tcp.analysis.ack_lost_segment</td>
<td>None</td>
<td>ACKed Lost Packet</td>
<td>0.10.0 to 0.99.5</td>
</tr>
<tr>
<td>tcp.analysis.ack_rtt</td>
<td>Time duration</td>
<td>The RTT to ACK the segment was</td>
<td>0.10.0 to 0.99.5</td>
</tr>
<tr>
<td>tcp.analysis.acks_frame</td>
<td>Frame number</td>
<td>This is an ACK to the segment in frame</td>
<td>0.10.0 to 0.99.5</td>
</tr>
</tbody>
</table>
Operators and Advanced Filters

Operators

== or eq equal to
|| or or
> or gt greater than
|= or ge greater than or equal to
< or le less than or equal to
! or not not
!= or ne not equal to
contains
matches
Build Filters Based on Captured Packet

Right mouse click on any field and either **Apply** or **Prepare** a filter based on the field and value (with an implied ‘equal to’ operator).
Build Filters Based on Expressions

Field Name
- FractalGeneratorProtocol
- Frame
- FRSAI
- FRSPIC
- FTAM
- FTBP
- FTP
  - ftp.response - Response (TRUE if FTP response)
  - ftp.request - Request (TRUE if FTP request)
  - ftp.request.command - Request command
  - ftp.request.arg - Request arg
  - ftp.response.code - Response code
  - ftp.response.arg - Response arg

Relation
- is present
- !=
- >
- >=
- <
- <=

Value
- 227

Predefined Values:
- File status
- Help message
- NAME system type
- Service ready for new user
- Service closing control connection
- Data connection open; no transfer
- Closing data connection
- Entering Passive Mode
- User logged in, proceed
- Requested file action okay,
Build Filters from Statistics Reports

I use the ‘Prepare a Filter’ to build my filter
Top 10 Useful Filters

IP Address: ip.addr == x.x.x.x
MAC Address: eth.addr == xx:xx:xx:xx:xx:xx
ICMP: icmp
DHCP: bootp
High Delta Time: frame.time_delta > 1
TCP Port: tcp.port == x
UDP Port: udp.port == x
TCP ACK RTT: tcp.analysis.ack_rtt > 1
TCP Length: > x < y tcp.len > x && tcp.len < y

Bonus
Manually Editing the *dfilters* File

```
"Ethernet address 00:08:15:00:08:15" eth.addr =
"Ethernet type 0x0806 (ARP)" eth.type == 0x0806
"Ethernet broadcast" eth.addr == ff:ff:ff:ff:ff:
"No ARP" not arp
"IP only" ip
"IP address 192.168.0.1" ip.addr == 192.168.0.1
"IP address isn't 192.168.0.1, don't use != for
192.168.0.1)
"IPX only" ipx
"TCP only" tcp
"UDP only" udp
"UDP port isn't 53 (not DNS), don't use != for
TCP or UDP port is 80 (HTTP)" tcp.port == 80 |
"HTTP" http
"No ARP and no DNS" not arp and !(udp.port == 53
"Non-HTTP and non-SMTP to/from 192.168.0.1" not
(tcp.port == 25) and ip.addr == 192.168.0.1
"Macof window=512" tcp.window_size == 512
"ICMP type 8 code not 0" (icmp.type == 8) && !(i
"Ping code not 0" (icmp.type == 8) && !icm
```
Display Filters

The easiest way to learn is to look at existing traces and reference the field name you are interested in;

- Hypertext Transfer Protocol
- GET /css/ws-2.css HTTP/1.1

Request Method: GET
Request URL: /css/ws-2.css
Request Version: HTTP/1.1
Host: www.wireshark.org

HTTP Request Method (http.request.method), 3 bytes
When I want to build a filter, but don’t want to type out a long field name and use the Prepare A Filter -> Selected and then modify the filter from there.
Build Filters Based on Packets

Right mouse click on any field and either **Apply** or **Prepare** a filter based on the field and value (with an implied ‘equal to’ operator).
Save Filtered, Marked and Ranges of Packets

Packet Range Selection
- Captured/Displayed
- All packets
- Selected packets
- Marked packets
- First to last marked packet
- Range
Follow the Stream

This feature creates a display filter of the selected Packet’s IP address and port pairs.
One Way stream

You can select the stream data from the client or Server
Saving A Stream

Wireshark: Save Follow Stream As

Name: httpinfo.txt
Save in folder: T2-5 Advanced Capture and Display Filtering apr 19_0010_15

Browse for other folders

placetake 2008
Recently Used

Add Remove

T2-5 Advanced Capture and Display Filtering.pdf
T2-5 Advanced Capture and Display Filtering.ppt

Name
Modified
Filters.pdf 2/15/2008
T2-5 Advanced Capture and Display Filtering.ppt Today at 9:59

GET /lists/ HTTP/1.1
Host: www.wireshark.org
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.12) Gecko/20000201 Firefox/2.0.0.12
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://ca.my.yahoo.com/
Cookie: __utma=87653150.2101756827.1179861028.1203864725.1204161658.10
__utmc=87653150.1204161658.107.89.utmccn=(referral)|utmcsr=ca.
Rebuilding a file

ftp-download-good.pcap

You can rebuild files from the stream
HTTP Filter Example

Common filters for HTTP

http.request.version == "HTTP/1.1"

Will return commands and responses