Agenda

Included
• Create firewall latency charts using tshark & Excel
• Show examples of successful visualizations
• Explain the tcptrace chart and view examples

Excluded
• Review Wireshark’s I/O Graphs
Visualizing Problems Helps An Analyst

• Avoid capturing or mining excessive traffic
• Digest more packets much quicker
• Identify macro patterns and spot anomalies
• Direct (or even avoid) analysis efforts
• Explain the problem to others
• Prove or disprove hypotheses or corrective measures
An Unexpected Visualization
A Surprisingly Obvious Packet

<table>
<thead>
<tr>
<th>Offset (h)</th>
<th>00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>D4 C3 B2 A1 02 00 04 00 00 00 00 00 00 00 00</td>
</tr>
<tr>
<td>00000010</td>
<td>FF FF 00 00 01 00 00 00 00 00 00 00 00 00 00 00</td>
</tr>
<tr>
<td>00000020</td>
<td>40 01 00 00 EA 05 00 00 24 77 03 D9 36 94 88 1F</td>
</tr>
<tr>
<td>00000030</td>
<td>A1 3D 73 CE 08 00 45 20 05 DC BE AC 00 00 73 06</td>
</tr>
<tr>
<td>00000040</td>
<td>06 98 4B 67 6F 25 C0 A8 01 83 17 OD 80 97 DB 9E</td>
</tr>
<tr>
<td>00000050</td>
<td>18 4B DA 86 31 47 50 10 02 01 F0 92 00 00 48 54</td>
</tr>
<tr>
<td>00000060</td>
<td>A9 D3 33 12 34 56 AA DD 00 00 00 16 80 90 45 32</td>
</tr>
<tr>
<td>00000070</td>
<td>00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00</td>
</tr>
<tr>
<td>00000080</td>
<td>00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00</td>
</tr>
<tr>
<td>00000090</td>
<td>00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00</td>
</tr>
<tr>
<td>000000A0</td>
<td>00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00</td>
</tr>
<tr>
<td>000000B0</td>
<td>FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00</td>
</tr>
<tr>
<td>000000C0</td>
<td>FF FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00</td>
</tr>
<tr>
<td>000000D0</td>
<td>FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00 00</td>
</tr>
<tr>
<td>000000E0</td>
<td>FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00</td>
</tr>
<tr>
<td>000000F0</td>
<td>FF FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00</td>
</tr>
<tr>
<td>00000100</td>
<td>FF FF FF FF FF FF 00 00 00 00 00 00 00 00 00 00</td>
</tr>
<tr>
<td>00000110</td>
<td>FF FF FF FF FF FF FF 00 00 00 00 00 00 00 00 00</td>
</tr>
<tr>
<td>00000120</td>
<td>FF FF FF FF FF FF FF FF 00 00 00 00 00 00 00 00</td>
</tr>
<tr>
<td>00000130</td>
<td>FF FF FF FF FF FF FF FF FF 00 00 00 00 00 00 00</td>
</tr>
<tr>
<td>00000140</td>
<td>FF FF FF FF FF FF FF FF FF FF 00 00 00 00 00 00</td>
</tr>
<tr>
<td>00000150</td>
<td>FF FF FF FF FF FF FF FF FF FF FF 00 00 00 00 00</td>
</tr>
<tr>
<td>00000160</td>
<td>FF FF FF FF FF FF FF FF FF FF FF FF 00 00 00 00</td>
</tr>
</tbody>
</table>

Note: The hexadecimal values represent binary data, which may be interpreted differently depending on the context of the packet.
Firewall Latency Charts
The Situation

• App server to database queries slowed to a crawl
• App server and database tiers are in their own VRFs separated by firewalls
• Firewall team reported no recent changes had been applied
#!/usr/bin/env bash

INPUT_FILE='Firewall Latency.pcapng'
OUTPUT_FILE='Firewall Latency.csv'

if [[ ! -f "${OUTPUT_FILE}" ]]; then
    # Output the trace file's packet count and capture duration.
    echo
capinfos -c -u "${INPUT_FILE}"
    echo

    # Process the trace file.
    echo "Processing '${INPUT_FILE}' to '${OUTPUT_FILE}'..."
tshark -r '${INPUT_FILE}'\
    -T fields -E header=yes\
    -e frame.number -e frame.time_relative -e ip.id -e tcp.seq\
    -o tcp.relative_sequence_numbers:FALSE\
    > '${OUTPUT_FILE}'
    echo "done."
    echo
else
    echo
    echo "'${OUTPUT_FILE}' already exists"
    echo
fi
Visualization Accomplishments

✓ Avoid capturing or mining excessive traffic
✓ Digest more packets much quicker
✓ Identify macro patterns and spot anomalies
✓ Direct or avoid analysis efforts
✓ Explain the problem to others
✓ Prove or disprove hypotheses or corrective measures
Web App Load Testing
Performance Problem
The Situation

- Load testing a web application revealed mostly good performance but a small percentage of abysmal response times not specific to any particular operation.
- Web app server logs showed acceptable performance at all times for all operations.
- The infrastructure consisted of two sets of appliances in front of the servers:
HTTP Response Time Through 3 Chained Devices

- Reverse Proxy
- Load Balancer
- Web Server

Response Time (seconds logarithmic) vs Time of Day
#!/usr/bin/env bash

INPUT_FILE='Load Test.pcapng'
OUTPUT_FILE='Load Test.csv'

if [[ ! -f "${OUTPUT_FILE}" ]]; then

  # Process the trace file.
  echo -n "Processing '${INPUT_FILE}' to '${OUTPUT_FILE}'..."
  tshark -r "${INPUT_FILE}"\
    -Y 'http.time'\n    -T fields -E header=yes\n    -e frame.number -e frame.time_relative -e ip.src -e http.time\n    > "${OUTPUT_FILE}"
  echo "done."
  echo

else

  echo
  echo "'${OUTPUT_FILE}' already exists"
  echo
fi
Visualization Accomplishments

- Avoid capturing or mining excessive traffic
- Digest more packets much quicker
- Identify macro patterns and spot anomalies
- Direct or avoid analysis efforts
- Explain the problem to others
- Prove or disprove hypotheses or corrective measures
Check Keying Station
Image Load Delays
The Situation

• For bank checks that aren’t machine readable, operators review check images and manually key in data
• Operators were reporting occasional image load delays that slow down their performance, which in turn impacts their department’s metrics and individuals’ compensation
The Challenges

- Tickets are created when slowdowns are reported, but without helpful analysis information like accurate timings—Packet mining timeline guesswork
- There are two keying station sites geographically separated from the application & image servers, meaning that separate captures must be taken in at least two of the three locations—Multi-point capture correlation with differing timestamps
- Operator-to-app server assignment is non-deterministic, and images are spread across many image servers, which is also non-deterministic—Don’t know what mining filters to provide until after starting to look at packets
- The protocol is proprietary, so no decodes—Looking at packets just got a lot harder
Timelines of Keying Station Packet Drops and User-Reported Slowness

Keying stations in teal had their switch port settings corrected.
Reports of Keying Station Slowness over Time by Data Center

- **Primary DC**
- **Secondary DC**

Time:
- 18:00
- 19:00
- 20:00
- 21:00
- 22:00
- 23:00
- 00:00
- 01:00
- 02:00
- 03:00
- 04:00
- 05:00
Keying stations in teal had their switch port settings corrected.
Visualization Accomplishments

✓ Avoid capturing or mining excessive traffic
✓ Digest more packets much quicker
✓ Identify macro patterns and spot anomalies
✓ Direct or avoid analysis efforts
✓ Explain the problem to others
✓ Prove or disprove hypotheses or corrective measures
File Transfer Tuning Validation
The Situation

• We recommended increasing TCP window sizes to improve file transfer throughput
• Did it work?
Comparison of File Transfer Performance Before and After Tuning

Throughput (Mbps) vs. File Size (MB)

- After Tuning
- Before Tuning
- After Trendline
- Before Trendline
Visualization Accomplishments

- Avoid capturing or mining excessive traffic
- Digest more packets much quicker
- Identify macro patterns and spot anomalies
- Direct or avoid analysis efforts
- Explain the problem to others
- Prove or disprove hypotheses or corrective measures
Introducing tcptrace
tcptrace

• [http://www.tcptrace.org](http://www.tcptrace.org)—“tcptrace is a tool written by Shawn Ostermann at Ohio University, for analysis of [packet capture] files.”
• tcptrace creates a variety of charts, many of which are also implemented in Wireshark’s Statistics | TCP Stream Graphs menu.
• The Time Sequence chart is by far the coolest (IMHO), and is oftentimes termed a tcptrace chart.
TCP Bidirectionality
tcptrace is a Unidirectional Visualization
lawyers.cs.ohiou.edu;ftp-data ==> indigo.cs.ohiou.edu;1038 (time sequence graph)
FTP File Transfer
Realizing Poor Throughput
<table>
<thead>
<tr>
<th>Packet</th>
<th>Absolute Time</th>
<th>Delta Time</th>
<th>Port 1</th>
<th>Port 2</th>
<th>Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7654</td>
<td>15:45:07.929961000</td>
<td>0.000532000</td>
<td>17</td>
<td>1</td>
<td>1500</td>
<td>TCP</td>
</tr>
<tr>
<td>7655</td>
<td>15:45:07.929001000</td>
<td>0.001260000</td>
<td>17</td>
<td>1</td>
<td>1500</td>
<td>TCP</td>
</tr>
<tr>
<td>7656</td>
<td>15:45:07.929488000</td>
<td>0.001490000</td>
<td>17</td>
<td>1</td>
<td>1500</td>
<td>TCP</td>
</tr>
<tr>
<td>7657</td>
<td>15:45:07.929616000</td>
<td>0.001220000</td>
<td>17</td>
<td>1</td>
<td>1500</td>
<td>TCP</td>
</tr>
<tr>
<td>7658</td>
<td>15:45:07.929960000</td>
<td>0.001200000</td>
<td>17</td>
<td>1</td>
<td>1500</td>
<td>TCP</td>
</tr>
<tr>
<td>7659</td>
<td>15:45:07.929172000</td>
<td>0.001230000</td>
<td>17</td>
<td>1</td>
<td>1500</td>
<td>TCP</td>
</tr>
<tr>
<td>7660</td>
<td>15:45:07.929302000</td>
<td>0.001300000</td>
<td>17</td>
<td>1</td>
<td>1500</td>
<td>TCP</td>
</tr>
<tr>
<td>7661</td>
<td>15:45:07.929416000</td>
<td>0.001390000</td>
<td>17</td>
<td>1</td>
<td>1500</td>
<td>TCP</td>
</tr>
<tr>
<td>7662</td>
<td>15:45:07.929537000</td>
<td>0.001400000</td>
<td>17</td>
<td>1</td>
<td>1500</td>
<td>TCP</td>
</tr>
<tr>
<td>7663</td>
<td>15:45:07.929684000</td>
<td>0.001370000</td>
<td>17</td>
<td>1</td>
<td>1500</td>
<td>TCP</td>
</tr>
<tr>
<td>7664</td>
<td>15:45:07.930136000</td>
<td>0.001200000</td>
<td>17</td>
<td>1</td>
<td>1500</td>
<td>TCP</td>
</tr>
<tr>
<td>7665</td>
<td>15:45:07.930248000</td>
<td>0.001300000</td>
<td>17</td>
<td>1</td>
<td>1500</td>
<td>TCP</td>
</tr>
<tr>
<td>7666</td>
<td>15:45:07.930464000</td>
<td>0.001300000</td>
<td>17</td>
<td>1</td>
<td>1500</td>
<td>TCP</td>
</tr>
<tr>
<td>7667</td>
<td>15:45:07.930699000</td>
<td>0.001200000</td>
<td>17</td>
<td>1</td>
<td>1500</td>
<td>TCP</td>
</tr>
<tr>
<td>7668</td>
<td>15:45:07.930826000</td>
<td>0.001200000</td>
<td>17</td>
<td>1</td>
<td>1500</td>
<td>TCP</td>
</tr>
<tr>
<td>7669</td>
<td>15:45:07.931057000</td>
<td>0.001200000</td>
<td>17</td>
<td>1</td>
<td>1500</td>
<td>TCP</td>
</tr>
<tr>
<td>7670</td>
<td>15:45:07.931284000</td>
<td>0.001200000</td>
<td>17</td>
<td>1</td>
<td>1500</td>
<td>TCP</td>
</tr>
<tr>
<td>7671</td>
<td>15:45:07.931546000</td>
<td>0.001230000</td>
<td>17</td>
<td>1</td>
<td>1500</td>
<td>TCP</td>
</tr>
<tr>
<td>7672</td>
<td>15:45:07.932006000</td>
<td>0.001200000</td>
<td>17</td>
<td>1</td>
<td>1500</td>
<td>TCP</td>
</tr>
<tr>
<td>7673</td>
<td>15:45:07.932351000</td>
<td>0.001200000</td>
<td>17</td>
<td>1</td>
<td>52</td>
<td>TCP</td>
</tr>
<tr>
<td>7674</td>
<td>15:45:07.932352200</td>
<td>0.001200000</td>
<td>17</td>
<td>1</td>
<td>52</td>
<td>TCP</td>
</tr>
<tr>
<td>7675</td>
<td>15:45:07.932359000</td>
<td>0.001200000</td>
<td>17</td>
<td>1</td>
<td>52</td>
<td>TCP</td>
</tr>
<tr>
<td>7676</td>
<td>15:45:07.932376000</td>
<td>0.001200000</td>
<td>17</td>
<td>1</td>
<td>52</td>
<td>TCP</td>
</tr>
<tr>
<td>7677</td>
<td>15:45:07.932411000</td>
<td>0.001200000</td>
<td>17</td>
<td>1</td>
<td>52</td>
<td>TCP</td>
</tr>
<tr>
<td>7678</td>
<td>15:45:07.932429000</td>
<td>0.001200000</td>
<td>17</td>
<td>1</td>
<td>52</td>
<td>TCP</td>
</tr>
</tbody>
</table>

0x54 0x65535
Mainframe Sending Segments Out-of-Order
SMB File Transfer
Overrunning a Switch Buffer