T-Shark for the Win

June 14th 2016

Christian Landström

Senior IT Security Consultant | Airbus Defence and Space CyberSecurity
About / Outline

- Basics on T-Shark
- Useful situations for switching to CLI
- Batch Jobbing
- „Data“ / Field extraction
- Demo, Demo, Demo…
Tshark basics

- Tshark is the command line equivalent of Wireshark with access to nearly all features available for everyday use.

- Sticks to the “Default” Profile if no other one is specified.

- Dumps output to CLI which is useful for further processing e.g. using `grep/findstr`, `cut`, `(g)awk`, `sed`.
Tshark basics

- Interface listing useful for local live capture on installed machine

- Specify Capture interface with –i <interface number>

```
tshark -D
1. \\Device\\NPF_{xxx} (Onboard)
2. \\Device\\NPF_{xxx} (VMware Network Adapter VMnet8)
3. \\Device\\NPF_{xxx} (VMware Network Adapter VMnet1)
4. \\Device\\NPF_{xxx} (VPN)
```
Batch Jobbing

- When capturing big amounts of data, ring buffer with multiple files recommended for ease of analysis
- In most cases hundreds of files – each around 50-250 Mbytes
- Need for scripted, automated task offload of common or specific analysis objects for each and every trace file

- Target: Have smaller trace data to be able to load whole selection or time ranges into Wireshark without having too many packets overhead

- Typical example: Selection of all files containing packets from a certain host and filtering for that particular IP address
Batch Jobbing

- tshark -r <infile> -Y <filter> -w <outfile>

- Uses Default Profile -> beware if settings e.g. Reassembly are set
- Profile can be set by using –C <profile> flag
- Recommended: Have a specific “CLI” profile with all unneeded features and dissectors turned off for additional speed e.g. turn off GeoIP lookups if not needed
Batch Jobbing

tshark -r <infile> -Y <displayfilter> -o tcp.relative_sequence_numbers:FALSE

- Can be used to write or overwrite specific values into settings from the profile preferences for the particular tshark run

- E.g. -o tcp.relative_sequence_numbers:FALSE
Batch Jobbing

```
for %a in (*.pcap) DO tshark -r %a
   -Y ip.addr==192.168.0.1 -w filtered\filter1_%a
```

- Used for automated working on multiple capture files for static content filtering e.g. source IP or VLAN filtering
- Remember to set “%%” in front of variable when using Windows .bat files
Field extraction

tshark -r %a -Y ip.addr==192.168.0.1 -Tfields -e ip.src -e ip.dst

- Dump values supplied by the “-e” flags instead of the whole packet list line
- Can be used to access all data which can be described by a display filter
- Can have multiple results per flag e.g. when having inner and outer IP headers or IP addresses within ICMP quotes etc.
Example: Building a DNS domain list from the trace file

# tshark -r "trace.pcap" -Y "dns.flags.response==1 and dns.resp.type==1" -Tfields -e dns.qry.name -e dns.a

OR

# tshark -r "trace.pcap" -q -z hosts
Demo Time

Example: Extracting the TTL values from DNS responses

```bash
# tshark -r "trace.pcap" -Y dns.flags.response==1 -Tfields -e
dns.resp.ttl | sed s/,/\r\n/g | sort -nr
80441
64022
52194
50364
49143
[...]
Example: Extracting information about MTU problems from fragmentation needed packets

```
# tshark -r trace.pcap -Y "icmp.type==3 && icmp.code==4"
   -Tfields -e ip.src -e icmp.mtu -e ip.dst
```

<table>
<thead>
<tr>
<th>Src IP from IP header and ICMP quote</th>
<th>MTU</th>
<th>Dst IP from IP header and quote</th>
</tr>
</thead>
<tbody>
<tr>
<td>172.16.31.10,172.16.31.55</td>
<td>800</td>
<td>172.16.31.55,192.168.1.1</td>
</tr>
</tbody>
</table>
Example: Extracting the HTTP response codes and times** since request

```
# tshark -r ,,trace.pcap" -Y http.response -Tfields -e frame.number -e http.response.code -e http.time
2   200   0.001896000
5   200   0.001051000
8   200   0.001849000
11  200   0.003594000
14  200   0.002530000
17  200   0.003147000
27  302   0.000431000
43  200   0.212918000
48  302   0.000003000
```

** beware the TCP stream reassembly setting
Example: Finding (possible) delays inside encrypted sessions

```
# tshark -r "trace.pcap" -Y "tcp.time_delta > 1" -Tfields -e tcp.stream -e frame.number
```

<table>
<thead>
<tr>
<th>Frame</th>
<th>Sequence Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>1256</td>
</tr>
<tr>
<td>25</td>
<td>2137</td>
</tr>
<tr>
<td>30</td>
<td>3116</td>
</tr>
</tbody>
</table>

Think of the required setting inside the TCP prefs to make delta conv. work.
2-stage batch jobs

for %a in (*.pcap) DO tshark -r %a -Y tcp.analysis.retransmission -Tfields -e tcp.stream > streams_with_retransmissions_%a.txt

- Typically used for conditional filtering of sessions containing a certain marker, due to conditional filtering based on one item not possible within Wireshark
  - e.g. → “Give me all TCP sessions containing packet loss”
- Can be eased by supplying the TCP Session ID (stream number) instead of IP / Port pairs
Example: 2-stage conversation filter containing retransmissions

1st stage: copy to file or attach „> error-streams.txt“
# tshark -r „trace.pcap“ -Y tcp.analysis.retransmission -Tfields -e
   tcp.stream | sort | uniq | sort -rn
154
137
130
126
[…]

2nd stage:
for /F %a in (error-streams.txt) DO
   tshark -r trace.pcap -Y tcp.stream==%a -w filtered\errorstream_%a.pcap
Demo Time

Example: Analyzing TCP Retransmissions with CLI

```bash
# tshark -r packetloss.pcapng -Y tcp.analysis.retransmission -Tfields -e tcp.stream | sort | uniq -ic | sort /R | more

# tshark -r packetloss.pcapng -Y "tcp.stream==0 and tcp.analysis.retransmission" -Tfields -e ip.src | sort | uniq -ic

# tshark -r packetloss_anon.pcapng -Y "tcp.stream==0 and tcp.analysis.retransmission and ip.src==26.0.0.0/8" -Tfields -e tcp.ack

# tshark -r packetloss_anon.pcapng -Y "tcp.stream==0 and tcp.analysis.retransmission and !ip.src==26.0.0.0/8" -Tfields -e tcp.ack
```
!! Thank you for attending !!

Questions?

---------------------------
eMail: landi@packet-foo.com
Web: www.packet-foo.com
Twitter: @0x6C616E6469