Command Line Review of Wireshark CLI Tools, tshark & more

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

• Basics on Wireshark CLI Tools
• Useful situations for switching to CLI
• Batch Jobbing
• „Data“ / Field extraction
Tshark basics

C:\Users\Landi\> tshark -h

- 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

- 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

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

```bash
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/,/\n/g | sort -nr

80441
64022
52194
50364
49143
[...]
```
Demo Time

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

172.16.31.10,172.16.31.55     800     172.16.31.55,192.168.1.1
```

Src IP from IP header and ICMP quote  MTU  Dst IP from IP header and quote
Demo Time

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    0.0000003000
```

** beware the TCP stream reassembly setting
Demo Time

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 Number</th>
<th>Timestamp</th>
<th>Source IP</th>
<th>Destination IP</th>
<th>Protocol</th>
<th>Seq</th>
<th>Ack</th>
<th>Time Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>0.082276000</td>
<td>87.106.221.182</td>
<td>192.168.0.12</td>
<td>TCP</td>
<td>283</td>
<td>50215</td>
<td>1.0008</td>
</tr>
<tr>
<td>25</td>
<td>0.082277000</td>
<td>87.106.221.182</td>
<td>192.168.0.12</td>
<td>TCP</td>
<td>330</td>
<td>50215</td>
<td>2.0001</td>
</tr>
<tr>
<td>30</td>
<td>0.100018000</td>
<td>192.168.0.12</td>
<td>87.106.221.182</td>
<td>TCP</td>
<td>251</td>
<td>50215</td>
<td>3.0000</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
Open and analyze the trace file “exercise1.pcapng”
Download trace files at
www.packet-foo.com/SF17/08.zip

Check with GUI and/or CLI for the following issues:

1. Web server response times (request to response code packet time delay after RTT)

2. Errors in HTTP communication (return codes 400s, 500s)

3. Give a guess where that trace file was captured to explain what you have seen
!! Thank you for attending !!

Questions?

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