PCAP Analysis with Wireshark and Tshark | Digital Forensics and Incident Response

Pranshu Bajpai
Last updated on Feb 18, 2023 · 6 min read · DFIR

Introduction

PCAPs can greatly aid an investigation after an incident has occurred. However, PCAPs contain massive amounts of data that is difficult to parse and time is valuable, especially during live investigations. How do we then swiftly perform a PCAP analysis that covers maximum ground? This post provides a quick summary of analysis that can be done by Wireshark and its accompanying CLI tool, `tshark`.

We will be using sample pcaps in this post. Grab a sample PCAP file here.

Quick Insights with capinfos

`capinfos` is a CLI tool that ships with Wireshark and can be useful to derive quick insights about the PCAP. It resides in the Wireshark directory, same as `tshark` and `reordercap` – the other CLI tools that ship with Wireshark. In my case (MacOS), `capinfos` was found here:

`/Applications/Wireshark.app/Contents/MacOS/capinfos`

```
 ~/Downloads  /Applications/Wireshark.app/Contents/MacOS/capinfos smallFlows.pcap
File name:           smallFlows.pcap
File type:           Wireshark/tcpdump/... - pcap
File encapsulation:  Ethernet
File timestamp precision:  microseconds (6)
Packet size limit:   file hdr: 65535 bytes
Number of packets:   14k
File size:           9,444kB
Data size:           9,216kB
Capture duration:    298.505344 seconds
First packet time:   2011-01-25 12:52:22.484409
Last packet time:    2011-01-25 12:57:20.989753
Data byte rate:      30kBps
Data bit rate:       247kbps
Average packet size: 646.28 bytes
Average packet rate: 47 packets/s
SHA256:              77d06d3f33f1a95fb9f2610f20ead8ce978449077d6ba22d4945b85bd48f75d1
RIPEMD160:           68e950b2e438c7be011a46646ddfc8105b65b339
SHA1:                06e502dca8dbfe8e6a9cabc61f722d94756f6db9
Strict time order:   True
Number of interfaces in file: 1
```

We therefore immediately see that this packet capture ran for a few minutes, with the first and last packet seen 5 minutes apart. Data byte rate suggests that the network was not under heavy load during the time.

PCAP analysis with Wireshark

Wireshark has become the industry-standard network capture analysis tool, and for good reason. It is powerful, flexible and a great tool to have in your DFIR arsenal.

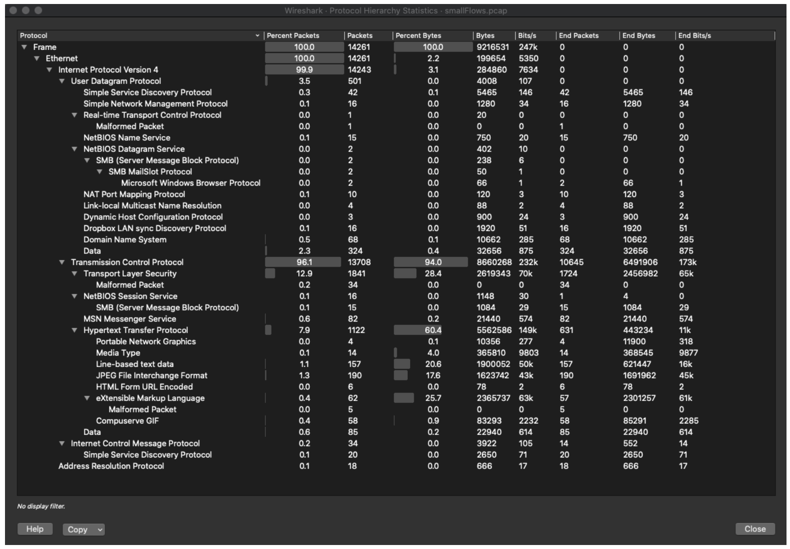
Adjusting timezone

By default, Wireshark will display timestamps in absolute time since the start of the capture. Unless you can read and interpret these, it's best to change these timestamps to human-readable dates and times. I usually change them to UTC for my investigations.

`View -> Time display format -> UTC date and time of day`

One of the first things I like to do after loading a PCAP in Wireshark is to look at the protocol hierarchy to understand the kind of traffic that the PCAP contains.

`Statistics -> Protocol Hierarchy`



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	14261	100.0	9216531	247k	0	0	0
Ethernet	100.0	14261	2.2	199654	5350	0	0	0
Internet Protocol Version 4	99.9	14243	3.1	284860	7634	0	0	0
User Datagram Protocol	3.5	501	0.0	4008	107	0	0	0
Simple Service Discovery Protocol	0.3	42	0.1	5465	146	42	5465	146
Simple Network Management Protocol	0.1	16	0.0	1280	34	16	1280	34
Real-time Transport Control Protocol	0.0	1	0.0	20	0	0	0	0
Malformed Packet	0.0	1	0.0	0	0	1	0	0
NetBIOS Name Service	0.1	15	0.0	750	20	15	750	20
NetBIOS Datagram Service	0.0	2	0.0	402	10	0	0	0
SMB (Server Message Block Protocol)	0.0	2	0.0	238	6	0	0	0
SMB MailSlot Protocol	0.0	2	0.0	50	1	0	0	0
Microsoft Windows Browser Protocol	0.0	2	0.0	66	1	2	66	1
NAT Port Mapping Protocol	0.1	10	0.0	120	3	10	120	3
Link-local Multicast Name Resolution	0.0	4	0.0	88	2	4	88	2
Dynamic Host Configuration Protocol	0.0	3	0.0	900	24	3	900	24
Dropbox LAN sync Discovery Protocol	0.1	16	0.0	1920	51	16	1920	51
Domain Name System	0.5	68	0.1	10662	285	68	10662	285
Data	2.3	324	0.4	32656	875	324	32656	875
Transmission Control Protocol	96.1	13708	94.0	8660268	232k	10645	6491906	173k
Transport Layer Security	12.9	1841	28.4	2619343	70k	1724	2456982	65k
Malformed Packet	0.2	34	0.0	0	0	34	0	0
NetBIOS Session Service	0.1	16	0.0	1148	30	1	4	0
SMB (Server Message Block Protocol)	0.1	15	0.0	1084	29	15	1084	29
MSN Messenger Service	0.6	82	0.2	21440	574	82	21440	574
Hypertext Transfer Protocol	7.9	1122	60.4	5562586	149k	631	443234	11k
Portable Network Graphics	0.0	4	0.1	10356	277	4	11900	318
Media Type	0.1	14	4.0	365810	9803	14	368545	9877
Line-based text data	1.1	157	20.6	1900052	50k	157	621447	16k
JPEG File Interchange Format	1.3	190	17.6	1623742	43k	190	1691962	45k
HTML Form URL Encoded	0.0	6	0.0	78	2	6	78	2
eXtensible Markup Language	0.4	62	25.7	2365737	63k	57	2301257	61k
Malformed Packet	0.0	5	0.0	0	0	5	0	0
Compuserve GIF	0.4	58	0.9	83293	2232	58	85291	2285
Data	0.6	85	0.2	22940	614	85	22940	614
Internet Control Message Protocol	0.2	34	0.0	3922	105	14	552	14
Simple Service Discovery Protocol	0.1	20	0.0	2650	71	20	2650	71
Address Resolution Protocol	0.1	18	0.0	666	17	18	666	17

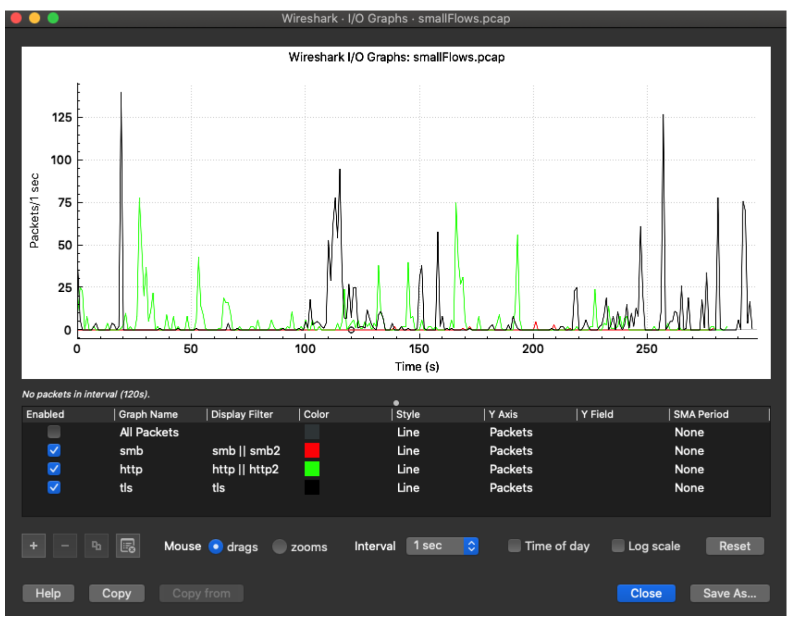
We see that we mostly have TCP traffic in this PCAP (96.1%) with a little bit of UDP (3.5%). Within TCP, we have mostly TLS and HTTP traffic.

IO Graphs

Next, it is good to build a timeline of traffic activity and fortunately Wireshark has I/O graphs for this purpose.

`Statistics -> I/O Graphs`

I usually select `Time of day` within the IO graph to see the accurate date and timestamps on the X-axis.

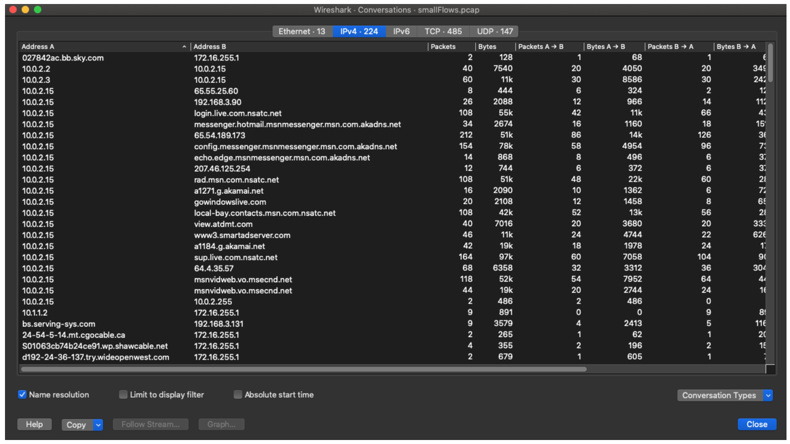


Color the different protocols (or combinations of protocols set with display filters) to improve the visualization. You can display all packets OR just the protocols that you are interested in.

Identify conversations between endpoints

took place and then check to see if they are relevant to the investigation:

`Statistics -> Conversations`



I usually check ‘Name resolution’ which makes it easy to identify domain names. Enable name resolutions in the Wireshark options first:

`Preferences -> Name resolution -> Use an external network name resolver`

Next, Wireshark provides the ability to quickly identify all endpoints involved in conversations:

`Statistics -> Endpoints`

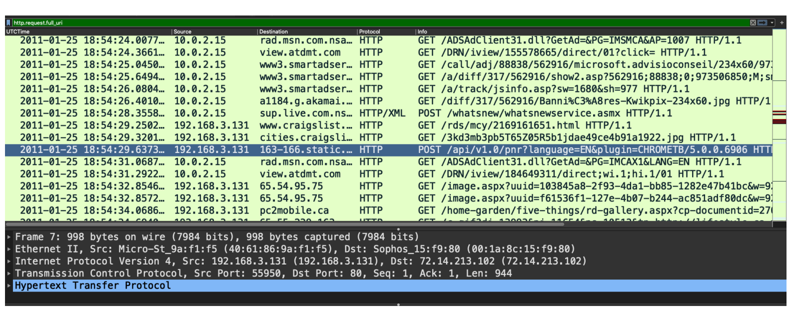
Display Filters

Display filters make it easy to make sense of the vast amounts of information contained within large PCAPs. Without unlimited time and patience, it is infeasible to scroll through the millions of packets that could be contained within the PCAP. Therefore, zeroing in on relevant information by specifying display filters is a good practice.

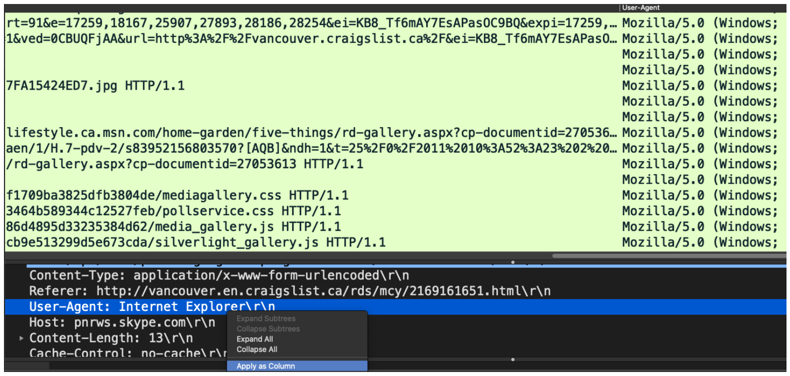
There are vast amounts of details available online on Wireshark display filters which I won’t attempt to replicate. Personally, I frequently use the following display filters during my investigations:

```
Operators:
    ||
    &&
    !
    contains
    ==

Filters:
    smb || smb2
    http || http2
    tls
    ip.addr == <ip>
    http.request.full_uri
    tcp.port == <port>
    tcp.stream eq <stream>
    !(arp or icmp or dns)
    frame contains <searchstring>
    !tcp.analysis.window_update
```



user-agent strings under a column:



Data extraction with tshark

`tshark` is the command-line utility that ships with Wireshark and can provide easy and flexible command-line access to the PCAP analysis data that can then be piped directly to `grep`, `awk` etc. for quick comprehension. `tshark` thus enables quick scripting.

Note: Before beginning analysis with `tshark`, it is advised to reorder packets using Wireshark's `reordercap`. While Wireshark does this transparently for the user, `tshark` will *not* and therefore the incorrect packet ordering will lead to inaccurate timestamps. Accurate timestamps are crucial for building DFIR incident timelines.

`reordercap` resides next to `tshark` in the installation directory of Wireshark. On my Mac, this directory was: `/Applications/Wireshark.app/Contents/MacOS/`

Run `reordercap` to derive a new PCAP with ordered packets:

`reordercap -n orignal.pcap REORDERED.pcap`

Basic tshark query format

I primarily use the following query format when extracting information with `tshark`:

```
tshark -r <pcap> -T fields -e <fieldname> -Y ...

where
    -r  read PCAP
    -T  set the format of output
            (if using 'fields', follow with -e)
    -e  extract specific field
    -Y  display filters
    -E  control printing of fields
            (use with 'separator=,' for CSVs)
```

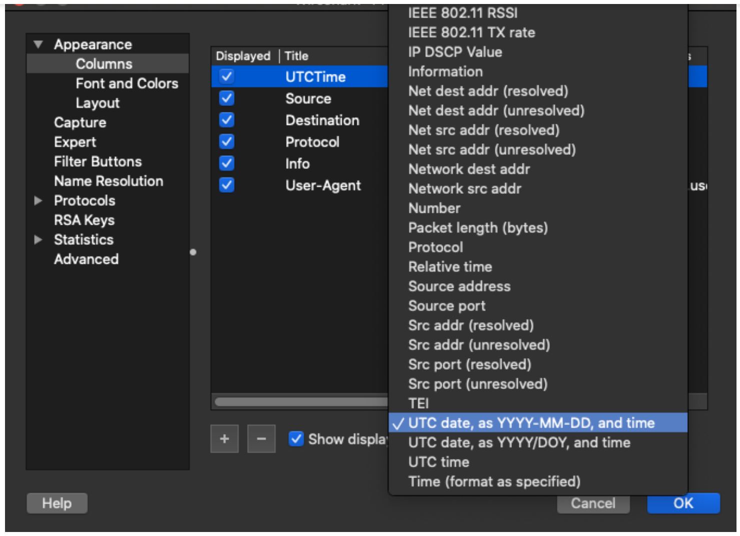
Accessing Wireshark columns via tshark

`tshark` can display data from columns created in Wireshark using `_wc.col.X` where `X` is the column name observed in Wireshark.

I usually create new columns in Wireshark and then extract the data in CLI using `tshark` with `-e _ws.col.<columnname>`

First, create a new column in Wireshark (in this case UTCTime):

`Preferences -> Columns -> Add`



The new display filter can now be used in `tshark`:

Sample tshark extractions

```
tshark -r <pcap> -T fields -e _ws.col.UTCTime -e ip.src ip.dst -Y "http || http2"

tshark -r <pcap> -T fields -e _ws.col.UTCTime -e _ws.col.Destination _ws.col.Info -Y "smb2"

tshark -r <pcap> -T fields -e _ws.col.Info -Y "smb2" || grep -B2 -C2 "FAIL"

tshark -r <pcap> -T fields -e _ws.col.UTCTime -e ip.src -e ip.dst -e dns.qry.name -Y "dns.fl

tshark -r <pcap> -Y smb2 | grep -B4 -C4NTLMSSP_AUTH

tshark -r <pcap> -Y http -T fields -e http.user_agent

tshark -r <pcap> --export-object http,.

tshark -r <pcap> -Y "udp or tcp" -T fields -e _ws.col.Protocol -e _ws.col.SrcPort -e _ws.col

tshark -r <pcap> -Y -e _ws.col.Time ntlmssp.auth.username -T fields -e ip.dst -e ntlmssp.aut

tshark -r <pcap> -T fields -e eth.dst eth.src

tshark -r <pcap> -T fields -e frame.protocols | sort | uniq -c | sort -nr
```

In the examples above, `_wc.col.X` pertains to the columns in Wireshark. Therefore, if you have enabled name resolution in Wireshark, IP addresses will be resolved in the `tshark` output as well.

Conclusion

While Wireshark and `tshark` are great tools for network packet capture analysis, I have found them lacking when PCAPs are sizable. For instance, PCAPs I was recently analyzing were several GBs in size and Wireshark struggled even on my Macbook Pro 16 (2019) with 32 GBs of RAM. To that end, we should be exploring other options when dealing with massive amounts of network capture data.

Additional links

- https://www.wireshark.org/docs/man-pages/tshark.html
- https://www.wireshark.org/docs/man-pages/capinfos.html
- https://www.wireshark.org/docs/man-pages/reordercap.html

Wireshark   Network Security

Pranshu Bajpai
Principal Staff Security Architect
PhD, Michigan State University.

Related

- PCAP Analysis with Zeek | Digital Forensics and Incident Response