

1 Analysis mit Wireshark

Im Rahmen dieser Sektion wollen wir eine weitere Datei analysieren, welche mit Wireshark aufgenommen wurde, dump_http.pcap.

1.1 Welche Objekte wurden vom Klienten via HTTP angefordert? Mithilfe von

File > Export Objects > HTTP...

kann man direkt einsehen, welche Elemente im Rahmen aller HTTP-Protokollierten Frames, innerhalb des derzeitig geladenen .pcap übertragen wurden. In unserem dump_http.pcap wurden die Elemente text(145 Byte, Frame 6), logo.gif(4445 Byte, Frame 22) und TechnikErleben.png(27 kB, Frame 66) angefordert, von 192.168.1.10.

1.2 Recherchieren Sie die Bedeutung der einzelnen Header-Felder bei den Anfragen bzw. Antworten des Servers. TCP-Header bestimmen eine ganze Reihe an diversen Datensätzen. Über 10 Felder von je 20 bytes & zusätzlichen 40 byte an optionalen, informativen Daten, bestimmt man

◇ Sender TCP Port Nummer (2 Byte)

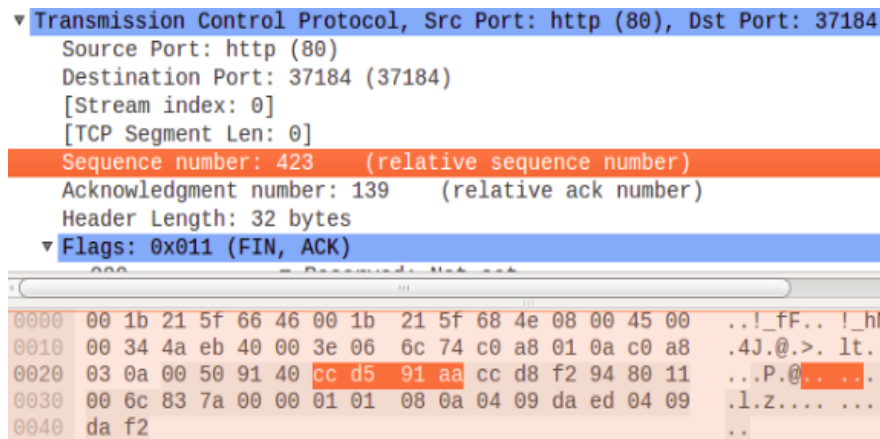
Der spezifizierte Port, von welchem der Sender sendet.

◇ Empfänger TCP Port Nummer (2 Byte)

Der spezifizierte Port, wo der Empfänger die Nachricht erhält.

◇ Sequenznummer (4 Byte)

... sie dient der Information über die Menge bereits gesendeter Data. Sie ist in jedem Datenpaket vorhanden und dient mitsamt der Acknowledge Nummer (next §) dazu, dem Sender den erfolgreichen Transfer zu übermitteln. Zwecks Sicherheitsvorkehrungen startet diese immer mit einem zufälligen Wert zwischen 0 und $(2^{32}) - 1$. Wireshark bietet hinsichtlich der X-hundertsten Zahlenwerten eine relative Sequenznummer an, um die Paketidentifikation in mehreren Hinsichten zu vereinfachen.



◊ Acknowledge number (4 Byte)

Diese dient dazu den Datentransfer zu bestätigen. Prinzipiell gilt, dass die geantwor-tete ACK zur neuen SEQ wird, siehe auch

Time	192.168.3.10		192.168.1.10	Comment
0.000000	37184	→	80	Seq = 0
0.100320	37184	←	80	Seq = 0 Ack = 1
0.100352	37184	→	80	Seq = 1 Ack = 1
0.100399	37184	→	80	Seq = 1 Ack = 1
0.200659	37184	←	80	Seq = 1 Ack = 139
0.200858	37184	←	80	Seq = 1 Ack = 139
0.200863	37184	→	80	Seq = 139 Ack = 423
0.200867	37184	←	80	Seq = 423 Ack = 139
0.201056	37184	→	80	Seq = 139 Ack = 424
0.201532	37185	→	80	Seq = 0
0.301299	37184	←	80	Seq = 424 Ack = 140
0.301798	37185	←	80	Seq = 0 Ack = 1
0.301807	37185	→	80	Seq = 1 Ack = 1
0.301839	37185	→	80	Seq = 1 Ack = 1

... Via Wireshark, lässt sich ein solcher Graph mittels

Statistics > Flow Graph > Flow Type: TCP flow

erstellen.

◊ TCP Data Offset (X words ... 8 Bytes each)

.. dieser dient zur Spezifikation des eigentlichen Dateistarts. Anhand von 'Header Length: xx Byte' lässt sich dies feststellen.

◊ Reserved Data (4 Bit)

Das Reserved Data-Feld ist für zukünftige Verwendungen reserviert. Alle Bits müssen null sein.

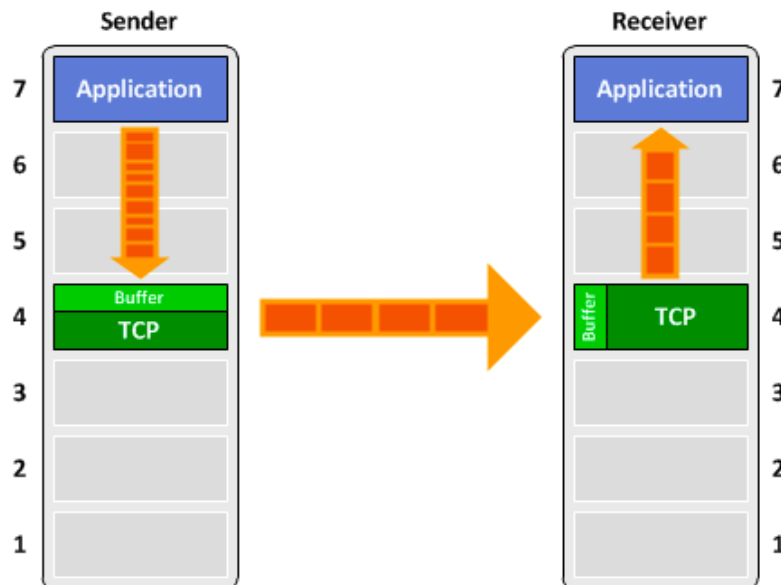
◊ Control Flags (8 Bit)

Beschrieben durch 2 Variablen oder 4 Hexa-Werten. Zu den möglichen Flags gehören:

- ECE - Explicit Congestion Notification (ECN-ECHO): Bei Netzwerküberlastung
- CWR - Congestion Window Reduced: Revertiert ECE
- URG - Urgent: Selten genutzt, oftmals als Interrupt-Signal
- ACK - Acknowledgement: Setzt Gültigkeit für Sequenz/Acknowledge-Values

- PSH - Push: Umgehung lokaler Puffer für schnellere Übertragung
- RST - Reset: Zum Abbruch von Verbindungen
- SYN - Initiieren der Verbindung. Üblicherweise beantwortet mit SYN+ACK oder RST
- FIN - Finish: Schlussflag, dient zur Freigabe der Verbindung & Bestätigt vollständige Übertragung.

Bezüglich PSH: Datenverkehr & Pufferfunktion bei Übertragung



- 1.3 Wie viele TCP-Verbindungen werden insgesamt aufgebaut? Wie unterscheidet sich das von dump_protocols.pcap? Mithilfe des Filters '**tcp && !http**' können wir insgesamt 68 TCP-Frames feststellen.
- 1.4 Bestimmen Sie, wie viele Bytes in jeder Verbindung ausgetauscht werden und wie lange die einzelnen Verbindungen bestehen.