Praktikum Rechnernetze

Protokoll zu Versuch 2 (Protokollanalyse mit Wireshark) von Gruppe 1

Jakob Waibel Daniel Hiller Elia Wüstner Felix Pojtinger

2021-10-26

Einführung

Mitwirken

Diese Materialien basieren auf Professor Kiefers “Praktikum Rechnernetze”-Vorlesung der HdM Stuttgart.

Sie haben einen Fehler gefunden oder haben einen Verbesserungsvorschlag? Bitte eröffnen Sie ein Issue auf GitHub (github.com/pojntfx/uni-netpractice-notes):

Abbildung 1: QR-Code zum Quelltext auf GitHub

Lizenz

Dieses Dokument und der enthaltene Quelltext ist freie Kultur bzw. freie Software.

Abbildung 2: Badge der AGPL-3.0-Lizenz

SPDX-License-Identifier: AGPL-3.0

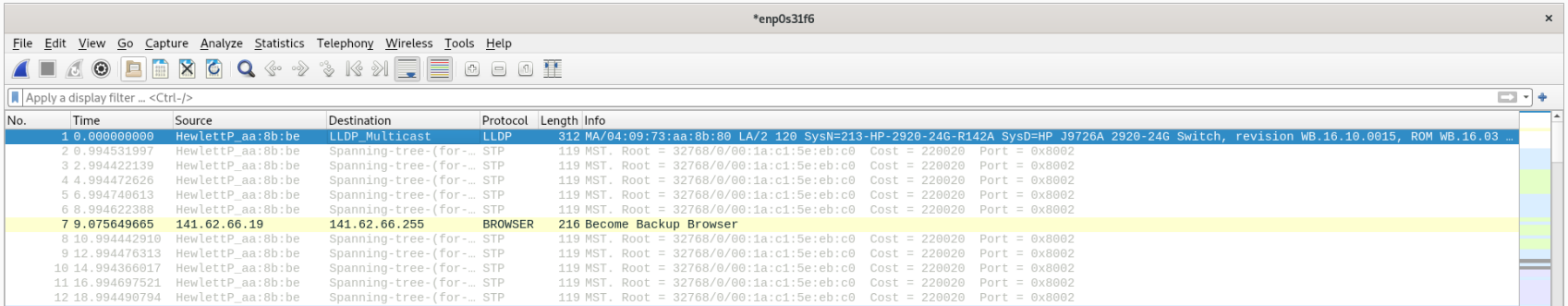
Wireshark

Einführung

An welchem Koppelelement im Systemschrank sollte der Hardware-Analysator/Netzwerk-Sniffer sinnvollerweise angeschlossen werden und warum? Welche grundsätzlichen Möglichkeiten gibt es noch?

- Switch, damit Nachrichten auf Layer 2 auch abgefangen werden können
- Grundsätzlich könnte, vor allem auch in Heimnetzwerken, der Router hierzu verwendet werden, da hier oft Router und Switch zu einem Gerät kombiniert sind.

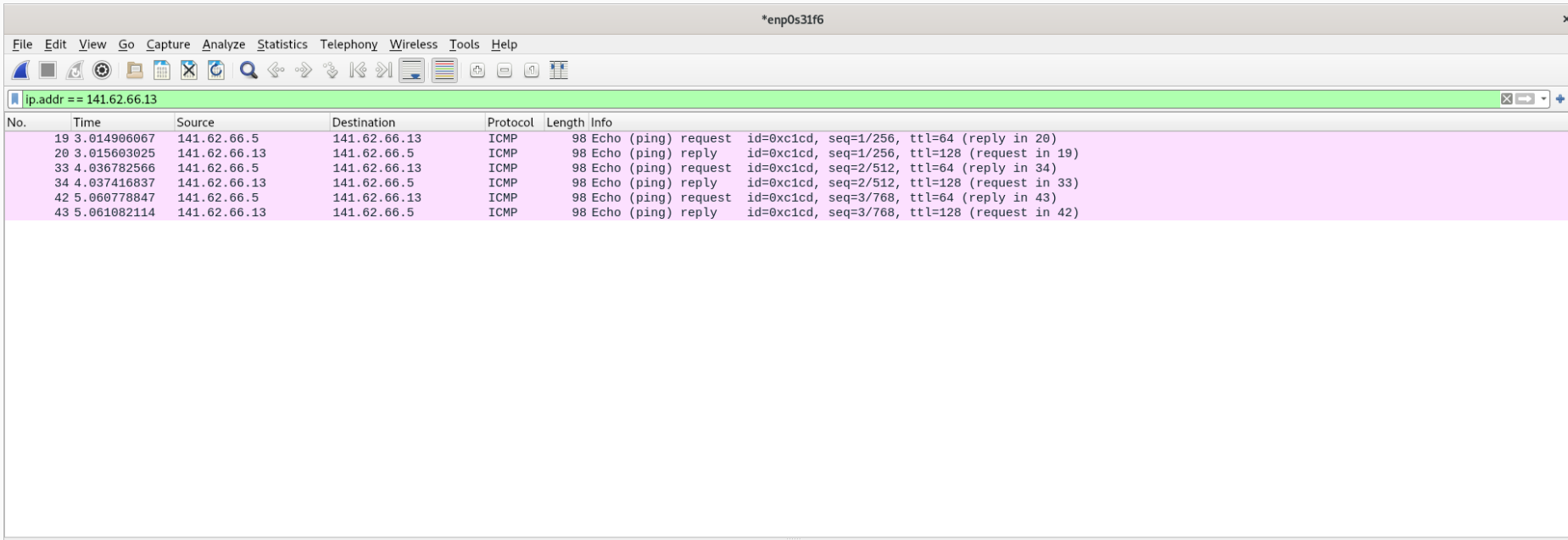
Starten Sie Wireshark und capturern Sie den aktuellen Traffic. Dokumentieren Sie zunächst, was alles auf Wireshark einprasselt.



Ping

Senden Sie einen Ping zu nachfolgenden Empfängern und zeichnen Sie die entsprechenden Protokolle gezielt mit Wireshark auf. Vergleichen Sie die Protokollabläufe: wer sendet welches Protokoll warum an wen? Pingen Sie an ....

Einen Rechner Ihrer Wahl im Labornetz:



DHCP

Analysieren Sie die Abläufe bei DHCP (im Labor installiert). Ihre Teilgruppe am Nachbartisch bootet den PC am Arbeitsplatz, protokollieren Sie die DHCP-Abläufe sowie sonstigen Netzverkehr, den der PC bis zum Erhalt der IP-Adresse erzeugt.

Während des Startens werden drei DHCP-Requests für verschiedene Komponenten abgehandelt.

!ssh && !stp && !tcp && !lldp && !browser && !mdns && !dns && !nbns && frame.number >|43

No.	Time	Source	Destination	Protocol	Length	Info
47	36.248724335	0.0.0.0	255.255.255.255	DHCP	590	DHCP Discover - Transaction ID 0x620e53eb
48	36.249844427	opnsense-router.rnl…	255.255.255.255	DHCP	348	DHCP Offer - Transaction ID 0x620e53eb
55	40.258252423	0.0.0.0	255.255.255.255	DHCP	590	DHCP Request - Transaction ID 0x620e53eb
56	40.259518728	opnsense-router.rnl…	255.255.255.255	DHCP	348	DHCP ACK - Transaction ID 0x620e53eb
57	40.259797973	linux.local	Broadcast	ARP	60	Who has 141.62.66.236? Tell 141.62.66.4
58	40.270416173	linux.local	Broadcast	ARP	60	Who has 141.62.66.250? Tell 141.62.66.4
63	45.478669439	fog.rnlabor.hdm-stu…	linux.local	ARP	60	Who has 141.62.66.4? Tell 141.62.66.236
65	46.502657513	fog.rnlabor.hdm-stu…	linux.local	ARP	60	Who has 141.62.66.4? Tell 141.62.66.236
70	47.526653895	fog.rnlabor.hdm-stu…	linux.local	ARP	60	Who has 141.62.66.4? Tell 141.62.66.236
72	48.497126304	0.0.0.0	255.255.255.255	DHCP	451	DHCP Discover - Transaction ID 0xc1470931
73	48.498452675	opnsense-router.rnl…	255.255.255.255	DHCP	348	DHCP Offer - Transaction ID 0xc1470931
79	50.529353450	0.0.0.0	255.255.255.255	DHCP	463	DHCP Request - Transaction ID 0xc1470931
80	50.531124992	opnsense-router.rnl…	255.255.255.255	DHCP	348	DHCP ACK - Transaction ID 0xc1470931
81	50.531125138	linux.local	Broadcast	ARP	60	ARP Announcement for 141.62.66.4
82	50.584564928	linux.local	Broadcast	ARP	60	Who has 141.62.66.236? Tell 141.62.66.4
85	54.828510780	linux.local	Broadcast	ARP	60	Who has 141.62.66.236? Tell 141.62.66.4
92	66.340215769	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xadc00d50
93	66.342356749	0.0.0.0	255.255.255.255	DHCP	345	DHCP Request - Transaction ID 0xadc00d50
95	66.629416649	linux.local	Broadcast	ARP	60	Who has 141.62.66.250? Tell 141.62.66.4

Abbildung 9: Gesamter Bootprozess

Dokumentieren Sie den Ablauf bei einer DNS-Abfrage

Fall 1: DNS-Server 141.62.66.250:

Mittels folgendem Command wurde eine DNS-Abfrage gemacht:

```
$ dig @141.62.66.250 google.com
google.com.        163 IN A 142.250.186.174
```

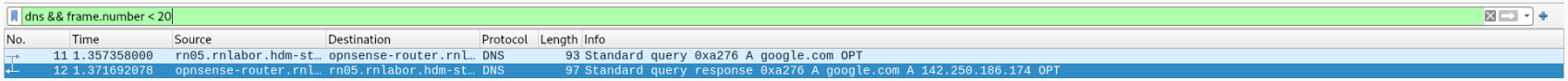


Abbildung 12: Ablauf der Anfrage

Hier nutzten wir den internen DNS Server und machen eine Anfrage auf google.com.

Fall 2: DNS-Server 1.1.1.1 (Cloudflare):

Mittels folgendem Command wurde eine DNS-Abfrage gemacht:

Lösen Sie eine ARP-Anfrage aus und protokollieren Sie die Datenpakete.

Hierzu wurde ein Rechner, welcher zuvor nicht im lokalen ARP-Cache war, neu gestartet.

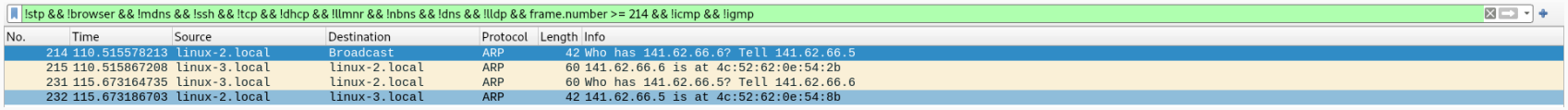


Abbildung 15: Ablauf der Anfrage

Wann wird eine ARP-Anfrage gestartet?

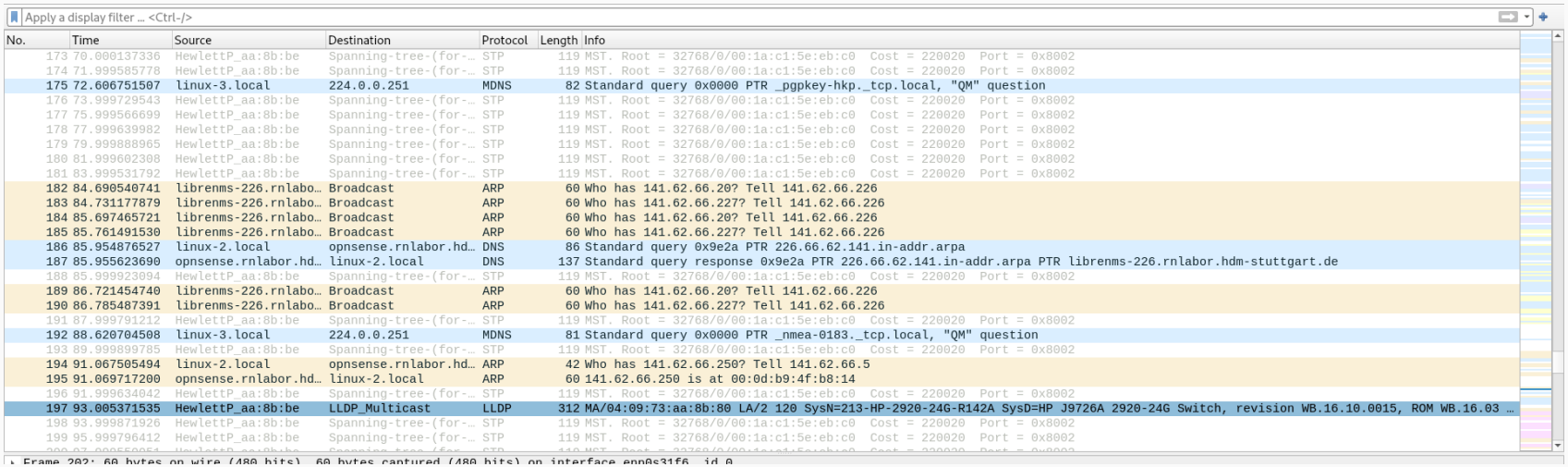
Sobald ein Paket an die Zieladresse (in unserem Fall 141.62.66.6) gesendet werden soll, wird eine ARP-Anfrage in Form eines Broadcasts gestartet, um das Zielgerät im Netzwerk zu ermitteln, sofern sich diese nicht bereits im ARP-Cache befindet. Dieser kann mit ip neigh show ausgelesen werden. Mit ip neigh flush all kann der ARP-Cache geleert werden.

Welcher Rahmentyp wird für die Anfrage verwendet?

Layer-2-Protokolle

Gelegentlich werden vom Analyzer Broadcasts erkannt. Wer sendet sie, warum und in welchen zeitlichen Abständen?

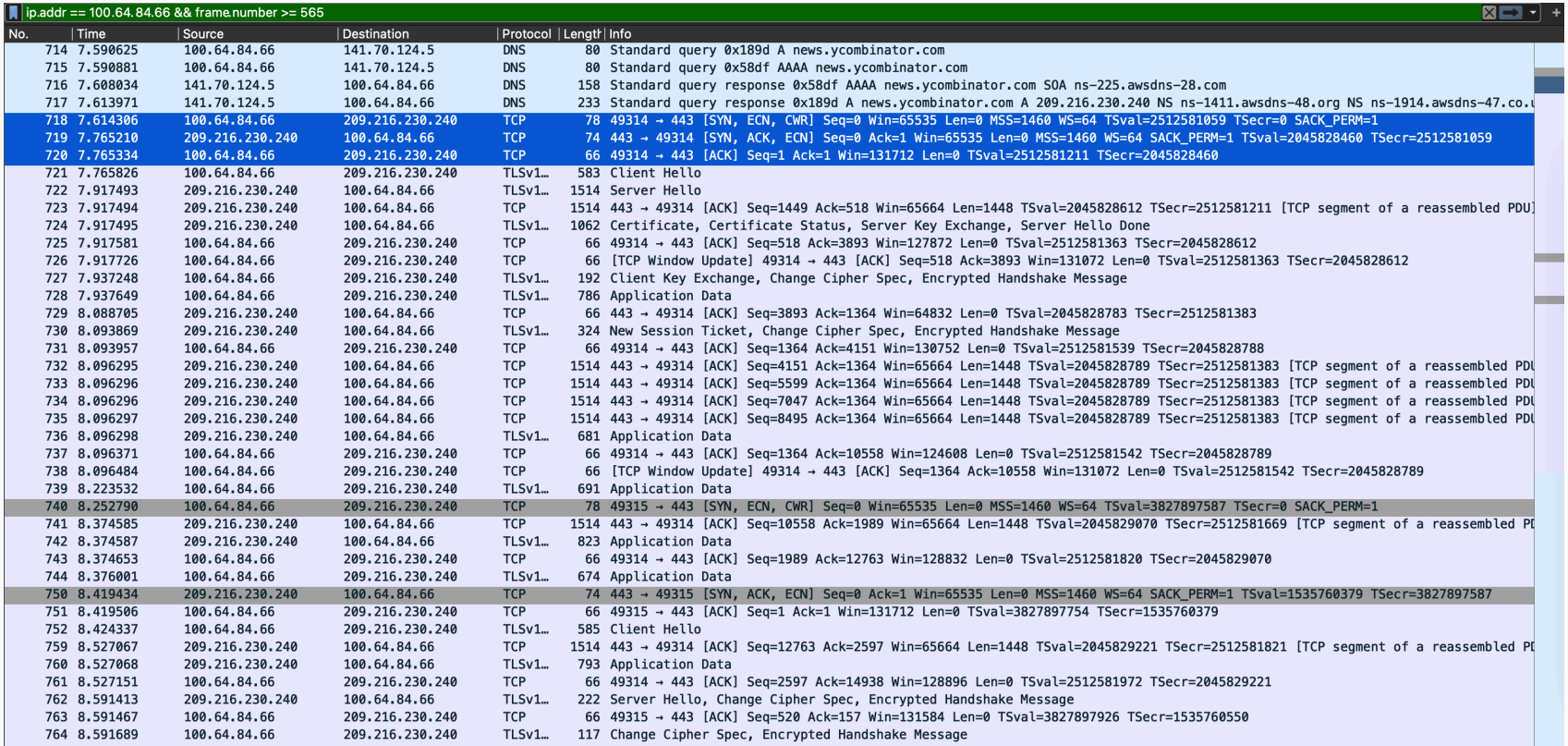
Die Broadcasts sind ARP-Requests. Sie entstehen dadurch, da Geräte versuchen Daten an andere Geräte zu übertragen, für welche sie keinen Eintrag in ihrem ARP-Cache haben, deshalb muss eine ARP-Anfrage in Form eines Broadcasts gesendet werden, da jeder Host potenziell der gesuchte Host sein kann. Dieser besitzt gesuchte IP X und antwortet daraufhin mit seiner Mac.



HTTP und TCP

Initiieren Sie eine HTTP-TCP-Sitzung (beliebige Website) und zeichnen Sie die Protokollabläufe auf

Zuerst wird ein DNS-Request getätigt. Daraufhin folgt der 3-Way-Handshake. Dieser ist an der charakteristischen Abfolge SYN, SYN-ACK, ACK zu erkennen.



ip.addr == 100.64.84.66 && frame.number >= 565

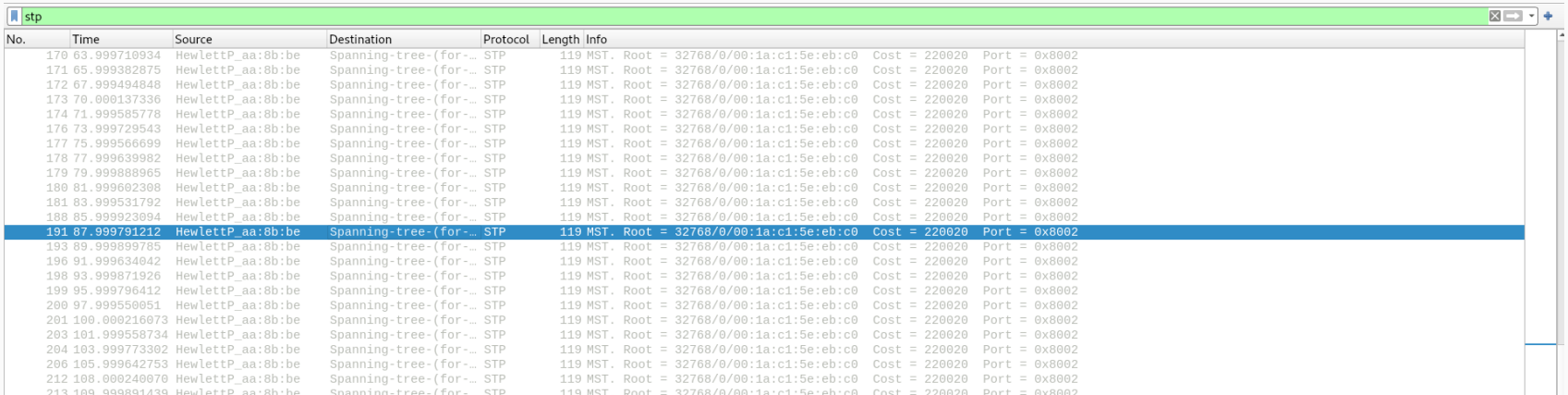
No.	Time	Source	Destination	Protocol	Length	Info
714	7.590625	100.64.84.66	141.70.124.5	DNS	80	Standard query 0x189d A news.ycombinator.com
715	7.590881	100.64.84.66	141.70.124.5	DNS	80	Standard query 0x58df AAAA news.ycombinator.com
716	7.608034	141.70.124.5	100.64.84.66	DNS	158	Standard query response 0x58df AAAA news.ycombinator.com SOA ns-225.awsdns-28.com
717	7.613971	141.70.124.5	100.64.84.66	DNS	233	Standard query response 0x189d A news.ycombinator.com A 209.216.230.240 NS ns-1411.awsdns-48.org NS ns-1914.awsdns-47.co.u
718	7.614306	100.64.84.66	209.216.230.240	TCP	78	49314 → 443 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=2512581059 TSecr=0 SACK_PERM=1
719	7.765210	209.216.230.240	100.64.84.66	TCP	74	443 → 49314 [SYN, ACK, ECN] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 TSval=2045828460 TSecr=2512581059
720	7.765334	100.64.84.66	209.216.230.240	TCP	66	49314 → 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=2512581211 TSecr=2045828460
721	7.765826	100.64.84.66	209.216.230.240	TLSv1…	583	Client Hello
722	7.917493	209.216.230.240	100.64.84.66	TLSv1…	1514	Server Hello
723	7.917494	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=1449 Ack=518 Win=65664 Len=1448 TSval=2045828612 TSecr=2512581211 [TCP segment of a reassembled PDU]
724	7.917495	209.216.230.240	100.64.84.66	TLSv1…	1062	Certificate, Certificate Status, Server Key Exchange, Server Hello Done
725	7.917581	100.64.84.66	209.216.230.240	TCP	66	49314 → 443 [ACK] Seq=518 Ack=3893 Win=127872 Len=0 TSval=2512581363 TSecr=2045828612
726	7.917726	100.64.84.66	209.216.230.240	TCP	66	[TCP Window Update] 49314 → 443 [ACK] Seq=518 Ack=3893 Win=131072 Len=0 TSval=2512581363 TSecr=2045828612
727	7.937248	100.64.84.66	209.216.230.240	TLSv1…	192	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
728	7.937649	100.64.84.66	209.216.230.240	TLSv1…	786	Application Data
729	8.088705	209.216.230.240	100.64.84.66	TCP	66	443 → 49314 [ACK] Seq=3893 Ack=1364 Win=64832 Len=0 TSval=2045828783 TSecr=2512581383
730	8.093869	209.216.230.240	100.64.84.66	TLSv1…	324	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
731	8.093957	100.64.84.66	209.216.230.240	TCP	66	49314 → 443 [ACK] Seq=1364 Ack=4151 Win=130752 Len=0 TSval=2512581539 TSecr=2045828788
732	8.096295	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=4151 Ack=1364 Win=65664 Len=1448 TSval=2045828789 TSecr=2512581383 [TCP segment of a reassembled PDU
733	8.096296	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=5599 Ack=1364 Win=65664 Len=1448 TSval=2045828789 TSecr=2512581383 [TCP segment of a reassembled PDU
734	8.096296	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=7047 Ack=1364 Win=65664 Len=1448 TSval=2045828789 TSecr=2512581383 [TCP segment of a reassembled PDU
735	8.096297	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=8495 Ack=1364 Win=65664 Len=1448 TSval=2045828789 TSecr=2512581383 [TCP segment of a reassembled PDU
736	8.096298	209.216.230.240	100.64.84.66	TLSv1…	681	Application Data
737	8.096371	100.64.84.66	209.216.230.240	TCP	66	49314 → 443 [ACK] Seq=1364 Ack=10558 Win=124608 Len=0 TSval=2512581542 TSecr=2045828789
738	8.096484	100.64.84.66	209.216.230.240	TCP	66	[TCP Window Update] 49314 → 443 [ACK] Seq=1364 Ack=10558 Win=131072 Len=0 TSval=2512581542 TSecr=2045828789
739	8.223532	100.64.84.66	209.216.230.240	TLSv1…	691	Application Data
740	8.252790	100.64.84.66	209.216.230.240	TCP	78	49315 → 443 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=3827897587 TSecr=0 SACK_PERM=1
741	8.374585	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=10558 Ack=1989 Win=65664 Len=1448 TSval=2045829070 TSecr=2512581669 [TCP segment of a reassembled PD
742	8.374587	209.216.230.240	100.64.84.66	TLSv1…	823	Application Data
743	8.374653	100.64.84.66	209.216.230.240	TCP	66	49314 → 443 [ACK] Seq=1989 Ack=12763 Win=128832 Len=0 TSval=2512581820 TSecr=2045829070
744	8.376001	100.64.84.66	209.216.230.240	TLSv1…	674	Application Data
750	8.419434	209.216.230.240	100.64.84.66	TCP	74	443 → 49315 [SYN, ACK, ECN] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 TSval=1535760379 TSecr=3827897587
751	8.419506	100.64.84.66	209.216.230.240	TCP	66	49315 → 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=3827897754 TSecr=1535760379
752	8.424337	100.64.84.66	209.216.230.240	TLSv1…	585	Client Hello
759	8.527067	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=12763 Ack=2597 Win=65664 Len=1448 TSval=2045829221 TSecr=2512581821 [TCP segment of a reassembled PD
760	8.527068	209.216.230.240	100.64.84.66	TLSv1…	793	Application Data
761	8.527151	100.64.84.66	209.216.230.240	TCP	66	49314 → 443 [ACK] Seq=2597 Ack=14938 Win=128896 Len=0 TSval=2512581972 TSecr=2045829221
762	8.591413	209.216.230.240	100.64.84.66	TLSv1…	222	Server Hello, Change Cipher Spec, Encrypted Handshake Message
763	8.591467	100.64.84.66	209.216.230.240	TCP	66	49315 → 443 [ACK] Seq=520 Ack=157 Win=131584 Len=0 TSval=3827897926 TSecr=1535760550
764	8.591689	100.64.84.66	209.216.230.240	TLSv1…	117	Change Cipher Spec, Encrypted Handshake Message

MAC

Wie lauten die MAC-Adressen der im Labor befindlichen Ethernet-Switches? Wie haben Sie die Switches identifizieren können. Welche Möglichkeiten der Identifizierung gibt es?

Beim Spanning-Tree-Protocol lässt sich sehen, dass die Quelle der Nachrichten immer ein HP-Gerät ist. Dieses muss ein fähiges Kopplungselement des Netzwerkes sein, welches das Spanning-Tree-Protocol unterstützt. Daher wird dies mit hoher Wahrscheinlichkeit der Ethernet-Switch sein.

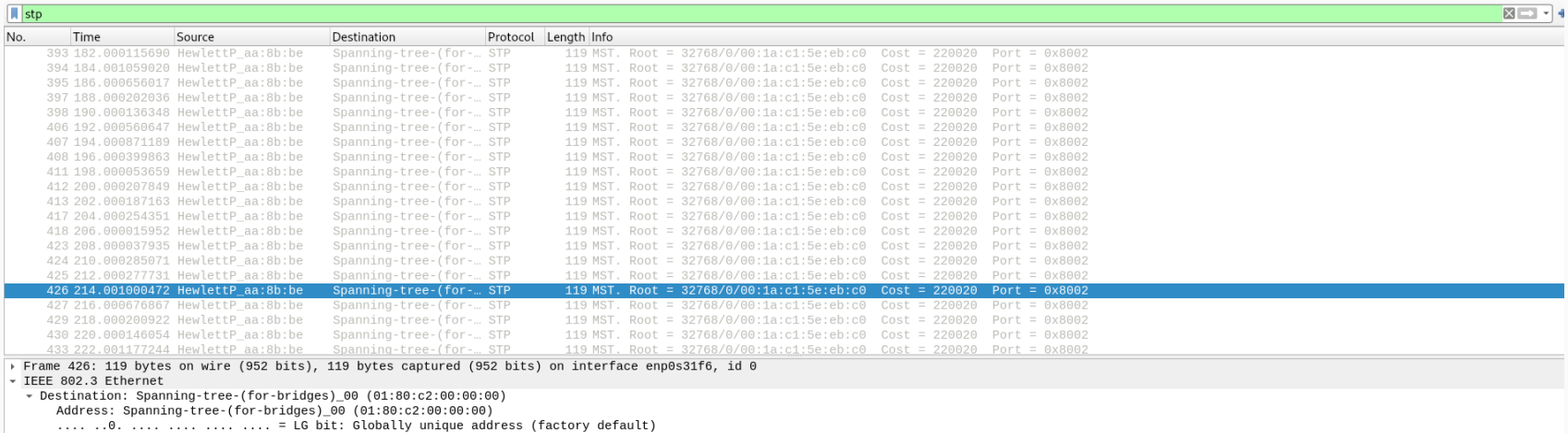
MAC-Adresse: 04:09:73: aa:8b:be


stp

No.	Time	Source	Destination	Protocol	Length	Info
170	63.999710934	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
171	65.999382875	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
172	67.999494848	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
173	70.000137336	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
174	71.999585778	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
176	73.999729543	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
177	75.999566699	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
178	77.999639982	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
179	79.999888965	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
180	81.999602308	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
181	83.999531792	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
188	85.999923094	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
191	87.999791212	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
193	89.999899785	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
196	91.999634042	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
198	93.999871926	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
199	95.999796412	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
200	97.999550051	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
201	100.000216073	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
203	101.999558734	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
204	103.999773302	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
206	105.999642753	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
212	108.000240070	HewlettP_aa:8b:be	Spanning-tree-(for-…	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002

Filtern Sie auf das Protokoll BPDU/STP. Wer sendet es und welchen Sinn hat dieses Protokoll?

Das STP-Protokoll ist das Spanning Tree Protocol. Das STP-Protokoll verhindert Schleifenbildung; dies ist besonders dann von Nutzen, wenn Redundanzen vorhanden sind. Beim STP-Protokoll werden durch alle am Netz beteiligten Switches eine “Root Bridge” gewählt und redundante Links werden deaktiviert. Wie anhand der OUI der MAC-Addresse erkannt werden kann wird dieses hier von einem HP-Switch verwendet.



Auf welchen Komponenten im Netzwerk wird das Protokoll SNMP ausgeführt?

Es konnte kein SNMP-Traffic im Netzwerk gefunden werden. SNMP, das Simple Network Management Protocol, wird jedoch meist zur Wartung von verbundenen Geräte im Network verwendet, woraus sich schließen lässt, dass es auf Komponenten wie Switches, Routern oder Servern zum Einsatz kommen würde.

Streaming and Downloads

Starten Sie einen Download einer größeren Datei aus dem Internet und stoppen Sie ihn während der Übertragung. Dokumentieren Sie, wie der Stop-Befehl innerhalb der Protokolle umgesetzt wird

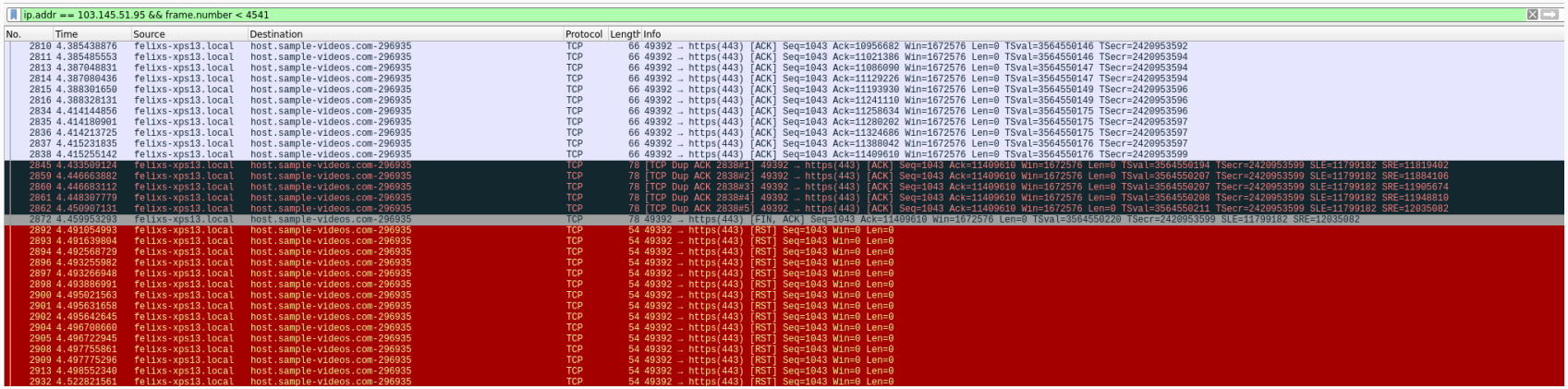


Abbildung 28: Capture beim Canceln des eines Downloads über HTTPS

Da der Download hier via HTTPS durchgeführt wurde, kann erkannt werden, dass die darunterliegende TCP-Verbindung unterbrochen wurde, indem die RST-Flag gesetzt wurde. Auch ein TCP-Segment, in welchem hier die FIN- und ACK-Flags gesetzt wurden, ist dementsprechend zu erkennen.

Protokollieren Sie den Ablauf einer TELNET-Verbindung zur IP-Adresse 141.62.66.207 (login: praktikum; passwd: versuch). Können Sie Passwörter im Wireshark-Trace identifizieren? Wie verhält sich im Vergleich dazu eine SSH-Verbindung zum gleichen Server?

Wie zu erkennen ist, wird für eine Telnet-Verbindung eine TCP-Verbindung aufgebaut. Die Passwörter sind zu erkennen.

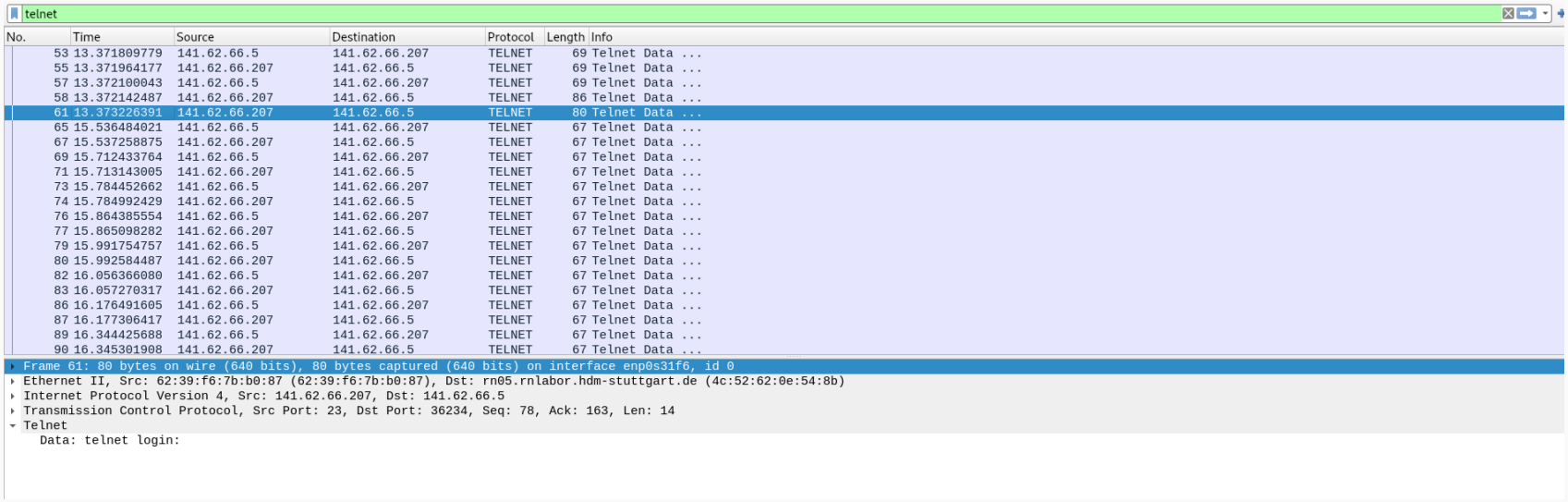
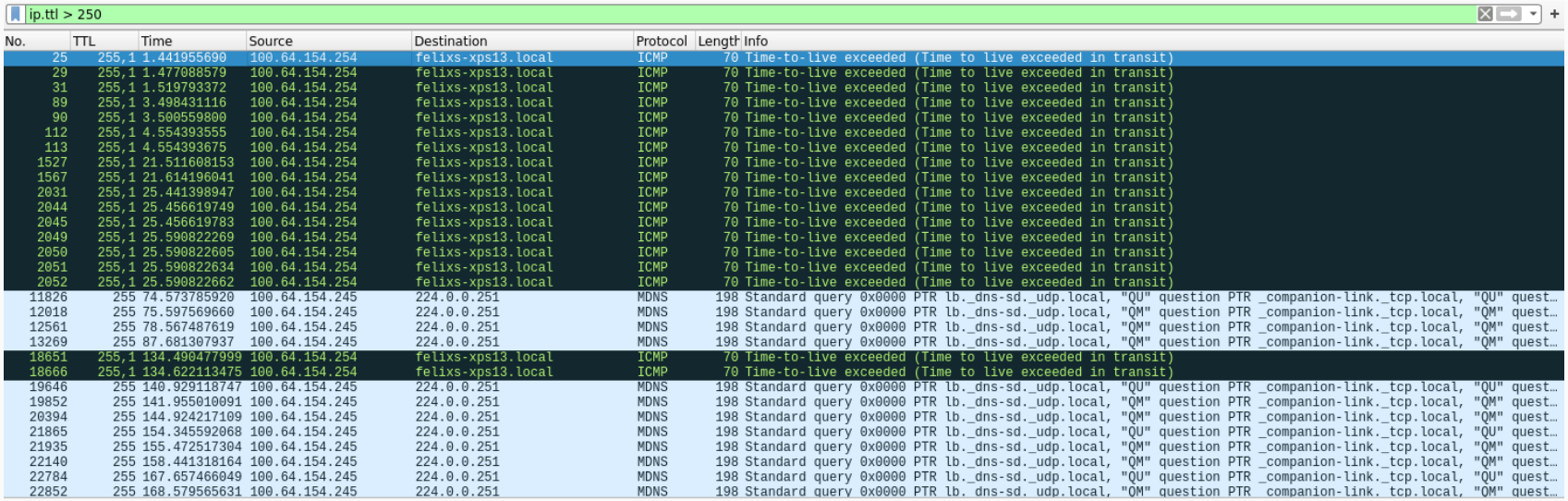


Abbildung 30: Capture des Telnet-Logins

Wireshark-Filter

Entwickeln, testen und dokumentieren Sie Wireshark-Filter zur Lösung folgender Aufgaben:

Nur IP-Pakete, deren TTL größer ist als ein von Ihnen sinnvoll gewählter Referenzwert



ip.ttl > 250

No.	TTL	Time	Source	Destination	Protocol	Length	Info
25	255,1	1.441955690	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
29	255,1	1.477088579	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
31	255,1	1.519793372	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
89	255,1	3.498431116	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
90	255,1	3.500559800	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
112	255,1	4.554393555	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
113	255,1	4.554393675	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
1527	255,1	21.511608153	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
1567	255,1	21.614196041	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2031	255,1	25.441398947	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2044	255,1	25.456619749	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2045	255,1	25.456619783	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2049	255,1	25.590822269	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2050	255,1	25.590822605	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2051	255,1	25.590822634	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2052	255,1	25.590822662	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
11826	255	74.573785920	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QU" question PTR _companion-link._tcp.local, "QU" quest…
12018	255	75.597569660	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QM" question PTR _companion-link._tcp.local, "QM" quest…
12561	255	78.567487619	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QM" question PTR _companion-link._tcp.local, "QM" quest…
13269	255	87.681307937	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QM" question PTR _companion-link._tcp.local, "QM" quest…
18651	255,1	134.490477999	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
18666	255,1	134.622113475	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
19646	255	140.929118747	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QU" question PTR _companion-link._tcp.local, "QU" quest…
19852	255	141.955010091	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QM" question PTR _companion-link._tcp.local, "QM" quest…
20394	255	144.924217109	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QM" question PTR _companion-link._tcp.local, "QM" quest…
21865	255	154.345592068	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QU" question PTR _companion-link._tcp.local, "QU" quest…
21935	255	155.472517304	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QM" question PTR _companion-link._tcp.local, "QM" quest…
22140	255	158.441318164	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QM" question PTR _companion-link._tcp.local, "QM" quest…
22784	255	167.657466049	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QU" question PTR _companion-link._tcp.local, "QU" quest…
22852	255	168.579565631	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QM" question PTR _companion-link._tcp.local, "QM" quest…

Abbildung 34: Capture der TTL-Werte ab 200

Der Linux-Kernel stellt standardmäßig die TTL auf 64; hier wurde ab 200 gefiltert, damit ausschließlich “ungewöhnliche” Pakete wie z.B.