

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):



Figure 1: QR-Code zum Quelltext auf GitHub

Lizenz

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



Figure 2: Badge der AGPL-3.0-Lizenz

Uni Network Practice Notes (c) 2021 Jakob Waibel, Daniel Hiller,
Elia Wüstner, Felix Pojtinger

SPDX-License-Identifier: AGPL-3.0

Wireshark

Einführung

An welchem Koppelement 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.

No.	Time	Source	Destination	Protocol	Length	Info
47	36.2407243935	0.0.0.0	255.255.255.255	DHCP	590	DHCP Discover - Transaction ID 0x620e53eb
48	36.2408444227	ognsense-router.rml...	255.255.255.255	DHCP	348	DHCP Offer - Transaction ID 0x620e53eb
55	40.2502524233	0.0.0.0	255.255.255.255	DHCP	590	DHCP Request - Transaction ID 0x620e53eb
56	40.2505187228	ognsense-router.rml...	255.255.255.255	DHCP	348	DHCP ACK - Transaction ID 0x620e53eb
57	40.250797973	linux.local	Broadcast	ARP	60	Who has 141.62.66.236? Tell 141.62.66.4
58	40.278416173	linux.local	Broadcast	ARP	60	Who has 141.62.66.250? Tell 141.62.66.4
63	45.4786694339	fog.rnrlabor.hdm-stu...	linux.local	ARP	60	Who has 141.62.66.47 Tell 141.62.66.236
65	46.4786694339	fog.rnrlabor.hdm-stu...	1.1.1.1	ARP	60	Who has 141.62.66.47 Tell 1.1.1.1
70	47.5266538893	fog.rnrlabor.hdm-stu...	linux.local	ARP	60	Who has 141.62.66.47 Tell 141.62.66.236
72	49.407126304	0.0.0.9	255.255.255.255	DHCP	451	DHCP Discover - Transaction ID 0x0c1478931
73	49.498452675	ognsense-router.rml...	255.255.255.255	DHCP	348	DHCP Offer - Transaction ID 0x0c1478931
79	50.5293533450	0.0.0.0	255.255.255.255	DHCP	463	DHCP Request - Transaction ID 0x0c1478931
80	50.531124992	ognsense-router.rml...	255.255.255.255	DHCP	348	DHCP ACK - Transaction ID 0x0c1478931
81	50.531125138	linux.local	Broadcast	ARP	60	ARP Announcement for 141.62.66.4
82	50.5845464928	0.0.0.0	Broadcast	ARP	60	Who has 141.62.66.236? Tell 141.62.66.4
85	54.8205154869	linux.local	Broadcast	ARP	60	Who has 141.62.66.236? Tell 141.62.66.4
92	56.342356749	0.0.0.0	255.255.255.255	DHCP	348	DHCP Discover - Transaction ID 0xadic98d59
93	66.342356749	0.0.0.0	255.255.255.255	DHCP	345	DHCP Offer - Transaction ID 0xadic98d59
95	66.6292416640	linux.local	Broadcast	ARP	60	Who has 141.62.66.250? Tell 141.62.66.4

Figure 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
```

dns && frame.number < 20						
No.	Time	Source	Destination	Protocol	Length	Info
11	1.357358668	rn05.rnlab0.hdm-st	opnsense-router.rn1	DNS	93	Standard query 0xa276 A google.com OPT
12	1.371692878	opnsense-router.rn1	rn05.rnlab0.hdm-st	DNS	97	Standard query response 0xa276 A google.com A 142.250.186.174 OPT

Figure 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):

Mit folgendem Command kann die DNS-Aufgabe gelöst werden:

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.

No.	Time	Source	Destination	Protocol	Length	Info
214	110.515578213	Linux-2.local	Broadcast	ARP	42	who has 141.62.66.6 Tell 141.62.66.5
215	110.515867298	Linux-3.local	Linux-2.local	ARP	60	141.62.66.6 is at 4c:52:62:0e:54:2b
231	115.073154795	Linux-3.local	Linux-2.local	ARP	60	who has 141.62.66.57 Tell 141.62.66.6
232	115.073186793	Linux-2.local	Linux-3.local	ARP	42	141.62.66.5 is at 4c:52:62:0e:54:2b

Figure 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

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.

Apply a display filter: <Ctrl>/>						
No.	Time	Source	Destination	Protocol	Length	Info
173	70.088137336	HewlettP...aa:0b:be	Spanning-tree-(For-)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x80002
174	71.09953778	HewlettP...aa:0b:be	Spanning-tree-(For-)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x80002
175	72.086751887	Linux-2.local	224.0.0.255	MDNS	62	Standard query 0x0000 PTR _popkey-hkp._tcp.local. "Qn" question
176	72.086751887	Linux-2.local	224.0.0.255	MDNS	62	Standard query 0x0000 PTR _popkey-hkp._tcp.local. "Qn" question
177	75.099556809	HewlettP...aa:0b:be	Spanning-tree-(For-)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x80002
178	77.099639982	HewlettP...aa:0b:be	Spanning-tree-(For-)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x80002
179	79.099888005	HewlettP...aa:0b:be	Spanning-tree-(For-)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x80002
180	81.099888005	HewlettP...aa:0b:be	Spanning-tree-(For-)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x80002
181	83.099537792	HewlettP...aa:0b:be	Spanning-tree-(For-)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x80002
182	84.099549741	Libremes-226.rnlabor.de	Broadcast	ARP	60	who has 141.02.66.226 Tell 141.02.66.226
183	84.731177879	Libremes-226.rnlabor.de	Broadcast	ARP	60	who has 141.02.66.227 Tell 141.02.66.229
184	85.099549741	Libremes-226.rnlabor.de	Broadcast	ARP	60	who has 141.02.66.227 Tell 141.02.66.229
185	85.761495159	Libremes-226.rnlabor.de	Broadcast	ARP	60	who has 141.02.66.227 Tell 141.02.66.226
186	85.91.084876527	Linux-2.local	opensemse.rnlabor.hdn.de	DNS	66	Standard query 0x0e2a PTR 226.66.62.141.in-addr.arpa
187	85.955623699	opensemse.rnlabor.hdn.de	Linux-2.local	DNS	137	Standard query response 0x0e2a PTR 226.66.62.141.in-addr.arpa PTR libremes-226.rnlabor.hdn-stuttgart.de
188	86.721654749	Libremes-226.rnlabor.de	Spanning-tree-(For-)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x80002
189	86.721654749	Libremes-226.rnlabor.de	Spanning-tree-(For-)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x80002
190	86.785467391	Libremes-226.rnlabor.de	Broadcast	ARP	60	who has 141.02.66.227 Tell 141.02.66.226
191	87.099791211	HewlettP...aa:0b:be	Spanning-tree-(For-)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x80002
192	88.029704508	Linux-3.local	224.0.0.255	MDNS	62	Standard query 0x0000 PTR _www-0193._tcp.local. "Qn" question
193	88.029704508	Linux-3.local	224.0.0.255	MDNS	62	Standard query 0x0000 PTR _www-0193._tcp.local. "Qn" question
194	91.087505484	Linux-2.local	opensemse.rnlabor.hdn.de	ARP	42	who has 141.02.66.259 Tell 141.02.66.5
195	91.089717289	opensemse.rnlabor.hdn.de	Linux-2.local	ARP	60	141.02.66.259 is at 00:0c:09:48:b8:14
196	91.089934042	HewlettP...aa:0b:be	Spanning-tree-(For-)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x80002
197	93.089571935	HewlettP...aa:0b:be	LLDP Multicast	LLDP	312	Ma/0:4:0973:aa:bb/Lv/2/12 SysName:219-WP-2920-240 I4/424 SysID:HP 3972EA 2929-240 Switch, revision W8.10.0015, ROM W8.16.03 -
198	93.090077105	HewlettP...aa:0b:be	Spanning-tree-(For-)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x80002
199	93.090077105	HewlettP...aa:0b:be	Spanning-tree-(For-)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x80002
200	93.090077105	HewlettP...aa:0b:be	Spanning-tree-(For-)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x80002

HTTP und TCP

Initiiieren 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.

No.	Time	Source	Destination	Protocol	Length	Info
714	7.590625	100.64.84.66	141.78.124.5	DNS	88	Standard query 0x189d A news.ycombinator.com
715	7.590641	100.64.84.66	141.78.124.5	DNS	88	Standard query 0x58df AAAA news.ycombinator.com
716	7.600834	141.78.124.5	100.64.84.66	DNS	158	Standard query response 0x58df AAAA news.ycombinator.com 504 ns=225.awdns-28.com
717	7.603971	141.78.124.5	100.64.84.66	DNS	233	Standard query response 0x58df AAAA news.ycombinator.com 209.216.230.248 NS ns=1914.awdns-47.co...
718	7.613186	100.64.84.66	209.216.230.248	TCP	74	49314 - 443 [SYN, ECR, CWR] Seq=0 Win=65535 Len=0 TSval=2045828612 TSecr=2512581059 SACK_PERM=1
719	7.615318	209.216.230.248	100.64.84.66	TCP	74	49314 - 443 [SYN, ACK] Seq=1 Win=65535 Len=0 TSval=2045828612 TSecr=2512581059 SACK_PERM=1
720	7.616334	100.64.84.66	209.216.230.248	TCP	74	49314 - 443 [ACK] Seq=1 Ack=1 Win=31712 Len=0 TSval=2512581711 TSecr=2045828468
721	7.765826	100.64.84.66	209.216.230.248	TLSv1...	583	Client Hello
722	7.917493	209.216.230.248	100.64.84.66	TLSv1...	1514	Server Hello
723	7.917494	209.216.230.248	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	209.216.230.248	100.64.84.66	TLSv1...	1062	Certificate, Certificate, Server Key Exchange, Server Key Exchange, Server Hello Done
725	7.917581	100.64.84.66	209.216.230.248	TCP	66	49314 - 443 [ACK] Seq=518 Ack=3893 Win=127872 Len=0 TSval=2512581363 TSecr=2045828612
726	7.917726	100.64.84.66	209.216.230.248	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	209.216.230.248	TLSv1...	192	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
728	7.937649	100.64.84.66	209.216.230.248	TLSv1...	786	Application Data
729	8.088785	209.216.230.248	100.64.84.66	TCP	66	443 - 49314 [ACK] Seq=3893 Ack=1364 Win=64832 Len=0 TSval=2045828783 TSecr=2512581383
729	8.093869	209.216.230.248	100.64.84.66	TCP	66	49314 - 443 [New Session Ticket, Change Cipher Spec, Encrypted Handshake Message] Seq=3893 Ack=1364 Win=64832 Len=0 TSval=2045828783 TSecr=2045828788
731	8.093957	100.64.84.66	209.216.230.248	TCP	1514	443 - 49314 [ACK] Seq=135131 Ack=1364 Win=64832 Len=0 TSval=2045828788 TSecr=2512581383 [TCP segment of a reassembled PDU]
732	8.094055	209.216.230.248	100.64.84.66	TCP	1514	443 - 49314 [ACK] Seq=5599 Ack=1364 Win=65664 Len=1448 TSval=2045828788 TSecr=2512581383 [TCP segment of a reassembled PDU]
732	8.094696	209.216.230.248	100.64.84.66	TCP	1514	443 - 49314 [ACK] Seq=7847 Ack=1364 Win=65664 Len=1448 TSval=2045828788 TSecr=2512581383 [TCP segment of a reassembled PDU]
734	8.096206	209.216.230.248	100.64.84.66	TCP	1514	443 - 49314 [ACK] Seq=8495 Ack=1364 Win=65664 Len=1448 TSval=2045828788 TSecr=2512581383 [TCP segment of a reassembled PDU]
735	8.096297	209.216.230.248	100.64.84.66	TCP	1514	443 - 49314 [ACK] Seq=8495 Ack=1364 Win=65664 Len=1448 TSval=2045828788 TSecr=2512581383 [TCP segment of a reassembled PDU]
736	8.096298	209.216.230.248	100.64.84.66	TLSv1...	681	Application Data
737	8.096371	100.64.84.66	209.216.230.248	TCP	66	49314 - 443 [ACK] Seq=1364 Ack=18558 Win=124688 Len=0 TSval=2512581542 TSecr=2045828789
738	8.096484	100.64.84.66	209.216.230.248	TCP	66	[TCP Window Update] 49314 - 443 [ACK] Seq=1364 Ack=18558 Win=131072 Len=0 TSval=2512581542 TSecr=2045828789
739	8.223532	100.64.84.66	209.216.230.248	TLSv1...	691	Application Data
740	8.223598	100.64.84.66	209.216.230.248	TCP	78	49315 - 443 [SYN, ECR, CWR] Seq=0 Win=65535 Len=0 MSS=1468 WS=64 TSval=1382795700 TSecr=0 SACK_PERM=1
741	8.374585	209.216.230.248	100.64.84.66	TCP	1514	49314 - 443 [ACK] Seq=18558 Ack=1989 Win=65664 Len=1448 TSval=2045829070 TSecr=2512581669 [TCP segment of a reassembled PDU]
742	8.374587	209.216.230.248	100.64.84.66	TLSv1...	823	Application Data
743	8.374653	100.64.84.66	209.216.230.248	TCP	66	49314 - 443 [ACK] Seq=1989 Ack=17263 Win=0 TSval=2512581820 TSecr=2045829070
744	8.376801	100.64.84.66	209.216.230.248	TLSv1...	674	Application Data
758	8.419434	209.216.230.248	100.64.84.66	TCP	74	443 - 49315 [SYN, ECR, ECR, CWR] Seq=0 Win=65535 Len=0 MSS=1468 WS=64 TSACK_PERM=1 TSval=1535768379 TSecr=382795787
751	8.419586	100.64.84.66	209.216.230.248	TCP	66	49315 - 443 [ACK] Seq=1 Ack=1 Win=31712 Len=0 TSval=3827997399 TSecr=1535768379
752	8.424337	100.64.84.66	209.216.230.248	TLSv1...	585	Client Hello
759	8.527457	209.216.230.248	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 PDU]
760	8.527968	100.64.84.66	209.216.230.248	TLSv1...	793	Application Data
761	8.527151	100.64.84.66	209.216.230.248	TCP	66	49314 - 443 [ACK] Seq=12763 Ack=18939 Win=128996 Len=0 TSval=2512581972 TSecr=2045829221
762	8.528143	209.216.230.248	100.64.84.66	TLSv1...	223	Server Hello, Change Cipher Spec, Encrypted Handshake Message
763	8.591467	100.64.84.66	209.216.230.248	TCP	66	49315 - 443 [ACK] Seq=12844 Ack=15184 Len=0 TSval=3827897926 TSecr=15357680550
764	8.591589	100.64.84.66	209.216.230.248	TLSv1...	132	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

No.	Time	Source	Destination	Protocol	Length Info
178	67.999718932	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
175	67.99938275	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
172	67.999494648	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
173	70.000137306	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
174	70.000137306	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
176	73.999729463	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
177	75.999566099	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
178	77.999939092	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
179	78.999988965	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
180	79.999988965	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
182	83.999931729	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
183	85.999931729	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
184	85.999931729	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
185	85.999923094	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
191	87.999791212	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
193	89.999999785	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
194	91.999999651	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
195	92.999999651	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
199	95.999746432	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
200	97.999850051	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
201	100.999218667	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
203	101.999595734	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
204	102.999595734	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
206	105.999842753	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 Port = 0x8002
212	108.999240977	NewletTCP.an.BB:e	Spanning-tree.(for...).STP	119 MST, Root = 32768/0/0/0:1a:c1:Se:eb:00	Cost = 229020 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-Adresse erkannt werden kann wird dieses hier von einem HP-Switch verwendet.

No.	Time	Source	Destination	Protocol	Length Info
393	182.000315869	HuaweiTP_aa:0b:be	Spanning-tree-(For-	STP	119 MST Root = 32768/0/0/0:1a:c1:5e:eb:09 Cost = 220629 Port = 0x8002
394	184.001058920	HuaweiTP_aa:0b:be	Spanning-tree-(For-	STP	119 MST Root = 32768/0/0/0:1a:c1:5e:eb:09 Cost = 220629 Port = 0x8002
395	186.000287784	HuaweiTP_aa:0b:be	Spanning-tree-(For-	STP	119 MST Root = 32768/0/0/0:1a:c1:5e:eb:09 Cost = 220629 Port = 0x8002
397	188.000282830	HuaweiTP_aa:0b:be	Spanning-tree-(For-	STP	119 MST Root = 32768/0/0/0:1a:c1:5e:eb:09 Cost = 220629 Port = 0x8002
398	190.000313040	HuaweiTP_aa:0b:be	Spanning-tree-(For-	STP	119 MST Root = 32768/0/0/0:1a:c1:5e:eb:09 Cost = 220629 Port = 0x8002
400	192.000560847	HuaweiTP_aa:0b:be	Spanning-tree-(For-	STP	119 MST Root = 32768/0/0/0:1a:c1:5e:eb:09 Cost = 220629 Port = 0x8002
407	194.000287119	HuaweiTP_aa:0b:be	Spanning-tree-(For-	STP	119 MST Root = 32768/0/0/0:1a:c1:5e:eb:09 Cost = 220629 Port = 0x8002
408	196.000287119	HuaweiTP_aa:0b:be	Spanning-tree-(For-	STP	119 MST Root = 32768/0/0/0:1a:c1:5e:eb:09 Cost = 220629 Port = 0x8002
411	198.000053859	HuaweiTP_aa:0b:be	Spanning-tree-(For-	STP	119 MST Root = 32768/0/0/0:1a:c1:5e:eb:09 Cost = 220629 Port = 0x8002
412	200.000287784	HuaweiTP_aa:0b:be	Spanning-tree-(For-	STP	119 MST Root = 32768/0/0/0:1a:c1:5e:eb:09 Cost = 220629 Port = 0x8002
413	202.000287113	HuaweiTP_aa:0b:be	Spanning-tree-(For-	STP	119 MST Root = 32768/0/0/0:1a:c1:5e:eb:09 Cost = 220629 Port = 0x8002
417	204.000287113	HuaweiTP_aa:0b:be	Spanning-tree-(For-	STP	119 MST Root = 32768/0/0/0:1a:c1:5e:eb:09 Cost = 220629 Port = 0x8002
418	206.000285952	HuaweiTP_aa:0b:be	Spanning-tree-(For-	STP	119 MST Root = 32768/0/0/0:1a:c1:5e:eb:09 Cost = 220629 Port = 0x8002
423	208.000053793	HuaweiTP_aa:0b:be	Spanning-tree-(For-	STP	119 MST Root = 32768/0/0/0:1a:c1:5e:eb:09 Cost = 220629 Port = 0x8002
424	210.000285871	HuaweiTP_aa:0b:be	Spanning-tree-(For-	STP	119 MST Root = 32768/0/0/0:1a:c1:5e:eb:09 Cost = 220629 Port = 0x8002
425	212.000287231	HuaweiTP_aa:0b:be	Spanning-tree-(For-	STP	119 MST Root = 32768/0/0/0:1a:c1:5e:eb:09 Cost = 220629 Port = 0x8002
426	214.001058472	HuaweiTP_aa:0b:be	Spanning-tree-(For-	STP	119 MST Root = 32768/0/0/0:1a:c1:5e:eb:09 Cost = 220629 Port = 0x8002
427	216.000287784	HuaweiTP_aa:0b:be	Spanning-tree-(For-	STP	119 MST Root = 32768/0/0/0:1a:c1:5e:eb:09 Cost = 220629 Port = 0x8002
429	218.000280932	HuaweiTP_aa:0b:be	Spanning-tree-(For-	STP	119 MST Root = 32768/0/0/0:1a:c1:5e:eb:09 Cost = 220629 Port = 0x8002
430	220.000514685	HuaweiTP_aa:0b:be	Spanning-tree-(For-	STP	119 MST Root = 32768/0/0/0:1a:c1:5e:eb:09 Cost = 220629 Port = 0x8002

SNMP

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

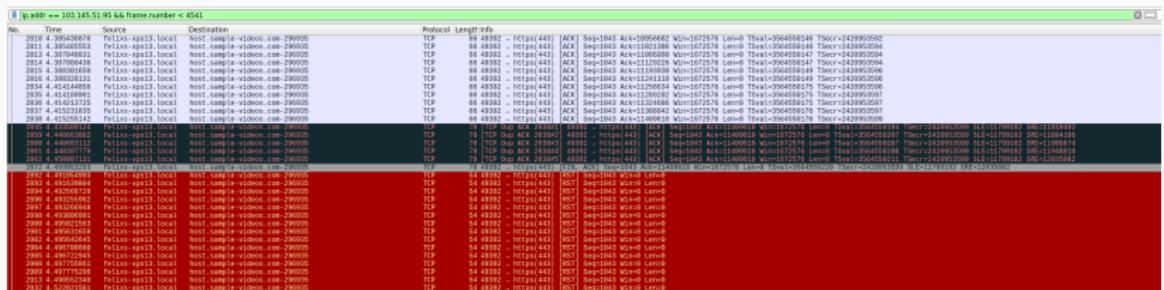


Figure 28: Capture beim Canceln des eines Downloads über HTTPS

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

Telnet und SSH

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.

No.	Time	Source	Destination	Protocol	Length	Info
53	13.3731889779	141.62.66.5	141.62.66.297	TELNET	69	Telnet Data ...
55	13.371964177	141.62.66.287	141.62.66.5	TELNET	69	Telnet Data ...
57	13.3721880643	141.62.66.5	141.62.66.297	TELNET	69	Telnet Data ...
58	13.372142487	141.62.66.287	141.62.66.5	TELNET	66	Telnet Data ...
59	13.372142487	141.62.66.5	141.62.66.297	TELNET	66	Telnet Data ...
60	13.372142487	141.62.66.5	141.62.66.297	TELNET	67	Telnet Data ...
65	15.5364848021	141.62.66.5	141.62.66.297	TELNET	67	Telnet Data ...
67	15.537258875	141.62.66.287	141.62.66.5	TELNET	67	Telnet Data ...
71	15.732433764	141.62.66.5	141.62.66.297	TELNET	67	Telnet Data ...
73	15.784452662	141.62.66.5	141.62.66.297	TELNET	67	Telnet Data ...
74	15.784992429	141.62.66.287	141.62.66.5	TELNET	67	Telnet Data ...
75	15.864385554	141.62.66.5	141.62.66.297	TELNET	67	Telnet Data ...
77	15.902258285	141.62.66.5	141.62.66.297	TELNET	67	Telnet Data ...
79	15.991757577	141.62.66.5	141.62.66.297	TELNET	67	Telnet Data ...
80	15.992584847	141.62.66.287	141.62.66.5	TELNET	67	Telnet Data ...
82	16.056366898	141.62.66.5	141.62.66.297	TELNET	67	Telnet Data ...
83	16.100366898	141.62.66.5	141.62.66.297	TELNET	67	Telnet Data ...
85	16.176491095	141.62.66.5	141.62.66.297	TELNET	67	Telnet Data ...
87	16.177366417	141.62.66.287	141.62.66.5	TELNET	67	Telnet Data ...
88	16.344425688	141.62.66.5	141.62.66.297	TELNET	67	Telnet Data ...
90	16.383339988	141.62.66.5	141.62.66.297	TELNET	67	Telnet Data ...

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

No.	TTL	Time	Source	Destination	Protocol	Length	Info
25	255	1.1.444955690	100.64.104.254	felix-xps13.local	ICMP	68	Time-to-live exceeded (Time to live exceeded in transit)
29	255	1.147708847	100.64.104.254	felix-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
31	255	1.1.819793372	100.64.104.254	felix-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
89	255	1.1.498431131	100.64.104.254	felix-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
90	255	1.1.3.100000000	100.64.104.254	felix-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
112	255	1.1.4.854393550	100.64.104.254	felix-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
113	255	1.1.4.854393675	100.64.104.254	felix-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
1527	255	1.21.511698853	100.64.104.254	felix-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
1597	255	1.21.511698853	100.64.104.254	felix-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2031	255	1.25.441388647	100.64.104.254	felix-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2844	255	1.25.456637949	100.64.104.254	felix-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2845	255	1.25.456619783	100.64.104.254	felix-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2849	255	1.25.598822269	100.64.104.254	felix-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2889	255	1.25.598822605	100.64.104.254	felix-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2891	255	1.25.598822634	100.64.104.254	felix-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2892	255	1.25.598822662	100.64.104.254	felix-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
11826	255	74.573785926	100.64.104.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QDN" question PTR_companion-link._tcp.local, "QDN" quest.
12018	255	75.597596960	100.64.104.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QDN" question PTR_companion-link._tcp.local, "QDN" quest.
12049	255	75.597596785	100.64.104.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QDN" question PTR_companion-link._tcp.local, "QDN" quest.
13269	255	87.123833737	100.64.104.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QDN" question PTR_companion-link._tcp.local, "QDN" quest.
16051	255	1.134.49847999	100.64.104.254	felix-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
16866	255	1.134.622113475	100.64.104.254	felix-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
16946	255	140.929138747	100.64.104.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QDN" question PTR_companion-link._tcp.local, "QDN" quest.
16952	255	145.929138747	100.64.104.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QDN" question PTR_companion-link._tcp.local, "QDN" quest.
23824	255	1.294237189	100.64.104.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QDN" question PTR_companion-link._tcp.local, "QDN" quest.
21865	255	154.345932968	100.64.104.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QDN" question PTR_companion-link._tcp.local, "QDN" quest.
21935	255	158.472539784	100.64.104.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QDN" question PTR_companion-link._tcp.local, "QDN" quest.
22148	255	158.441338164	100.64.104.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QDN" question PTR_companion-link._tcp.local, "QDN" quest.
22784	255	161.657466049	100.64.104.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QDN" question PTR_companion-link._tcp.local, "QDN" quest.
22852	255	161.657466049	100.64.104.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QDN" question PTR_companion-link._tcp.local, "QDN" quest.

Figure 34: Capture der TTL-Werte ab 200

Der Linux-Kernel stellt standardmäßig die TTL auf 64; hier wurde