
Praktikum Rechnernetze

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

Jakob Waibel, Daniel Hiller, Elia Wüstner, Felix Pojtinger

2021-10-26

Inhaltsverzeichnis

1 Einführung	2
1.1 Mitwirken	2
1.2 Lizenz	2
2 Wireshark	3
2.1 Einführung	3
2.2 Ping	5
2.3 DHCP	5
2.4 DNS	8
2.5 ARP	9
2.6 Layer-2-Protokolle	11
2.7 HTTP und TCP	13
2.8 MAC	17
2.9 STP	19
2.10 SNMP	19
2.11 Streaming and Downloads	20
2.12 Telnet und SSH	21
2.13 Wireshark-Filter	23

1 Einführung

1.1 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

Wenn Ihnen die Materialien gefallen, würden wir uns über einen GitHub-Stern sehr freuen.

1.2 Lizenz

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



Abbildung 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

2 Wireshark

2.1 Einführung

An welchem Koppellement 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.

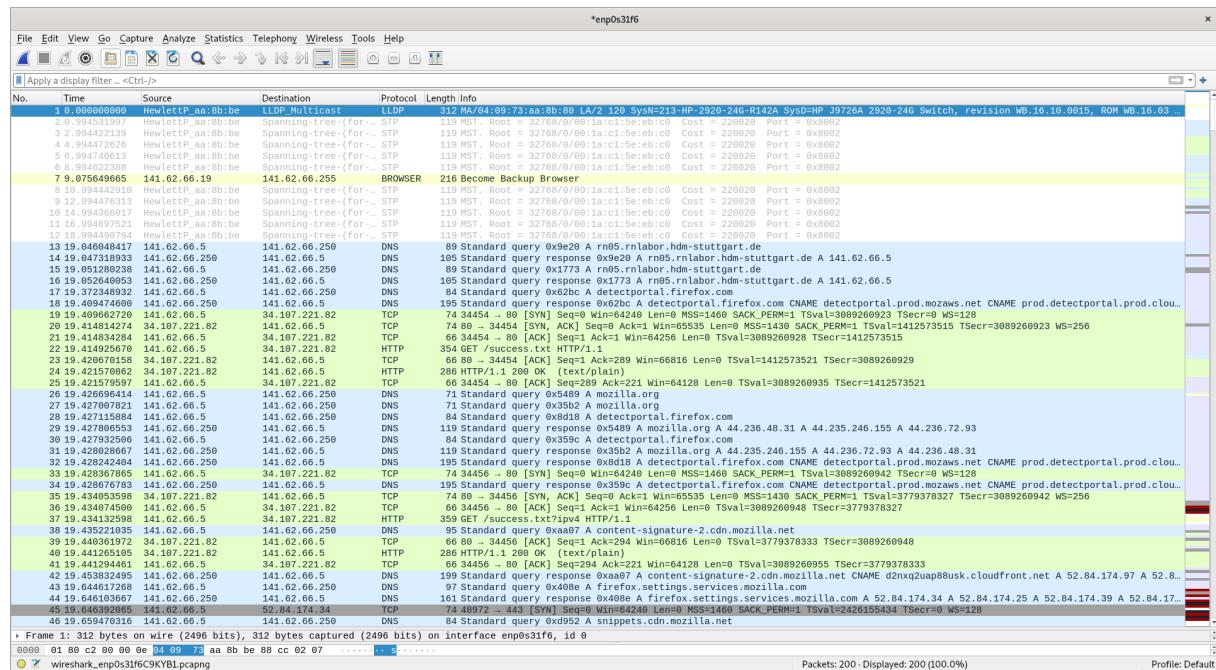


Abbildung 3: Screenshot von Wireshark

Zu erkennen sind Pakete von mehreren Protokollen:

- LLDP
- Spanning-Tree-Protokoll (STP)
- DNS
- TCP
- HTTP

Die letzten beiden Protokolle (TCP, HTTP) lassen sich durch das Öffnen des Browsers erklären.

Wie lautet der Filter, mit dem Sie ihre eigene Verbindung ins Labor ausklammern? Welche Möglichkeiten gibt es?

Hierzu gibt es mehrere Optionen:

```
1 !ip.addr == 141.62.66.5
2 not ip.addr == 141.62.66.5
3 !ip.addr eq 141.62.66.5
```

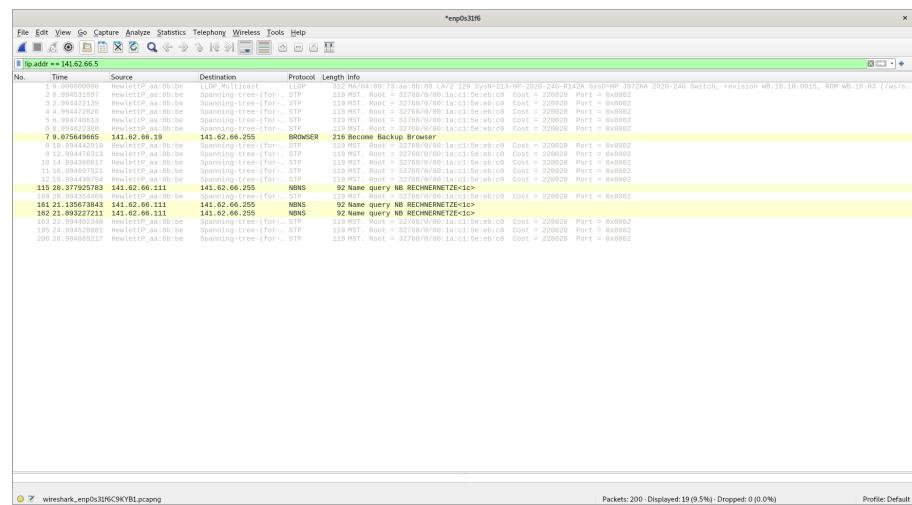


Abbildung 4: Ausklammern der eig. IP, Option 1

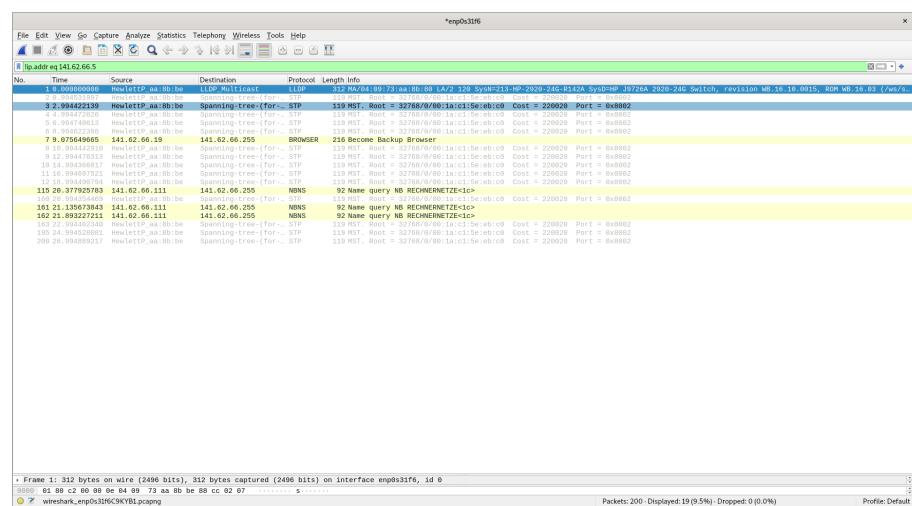


Abbildung 5: Ausklammern der eig. IP, Option 2

2.2 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:

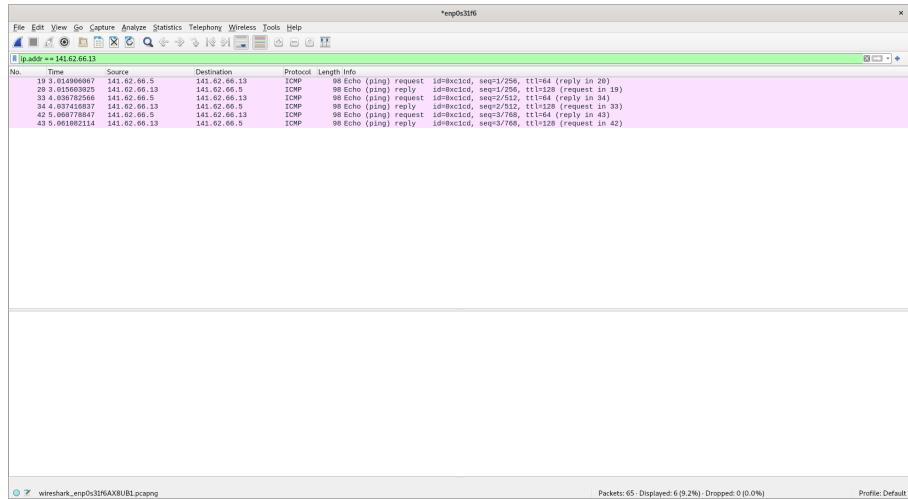


Abbildung 6: Wireshark-Output zu einem Rechner im Labornetz

Einen beliebigen Server im Internet (Google)

Wir haben hierzu die Namensauflösung aktiviert, damit die IPs zur Domain google.com zugeordnet werden können.

Eine beliebige nicht existierenden IP-Adresse

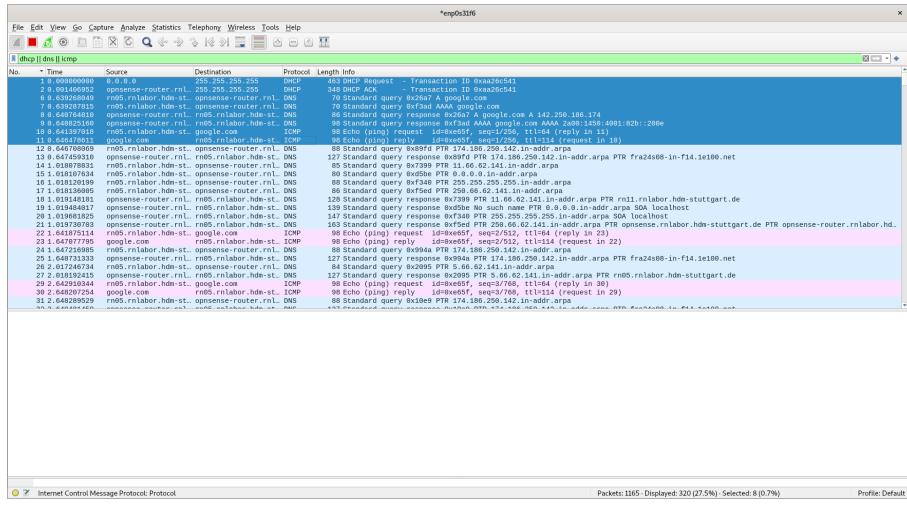
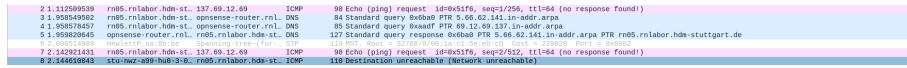
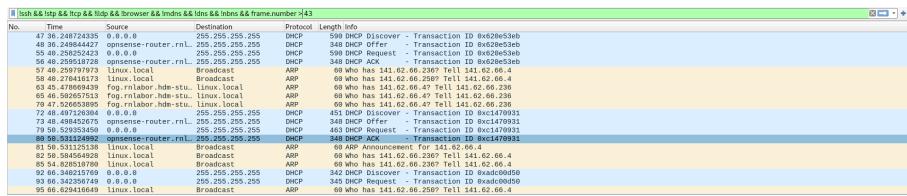
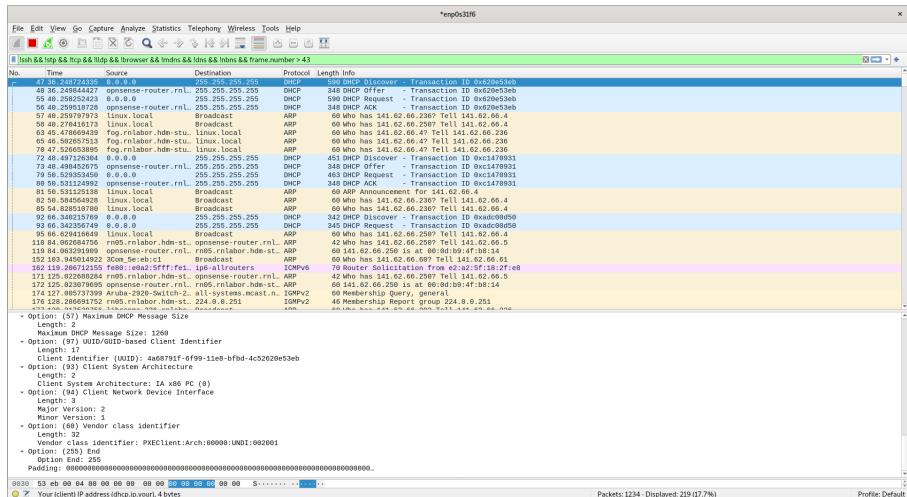
2.3 DHCP

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.

Strukturieren Sie die DHCP-Abläufe und beschreiben Sie, wie DHCP im Detail funktioniert.

Durch Booten des PCs wird dem Rechner mittels DHCP eine IP zugewiesen. Ergänzend kommen noch Standard-Gateway-Adresse und DNS Adresse hinzu. DHCP ermöglicht damit erst, dass verschiedene

**Abbildung 7:** Wireshark-Output zu einem Ping nach google.com**Abbildung 8:** Wireshark-Output zu einem Ping nach 137.69.12.69**Abbildung 9:** Gesamter Bootprozess**Abbildung 10:** Bootprozess: DHCP-Requests des BIOS zum Netzwerkboot, damit der Netzwerkbootloader über i.e. TFTP geladen werden kann

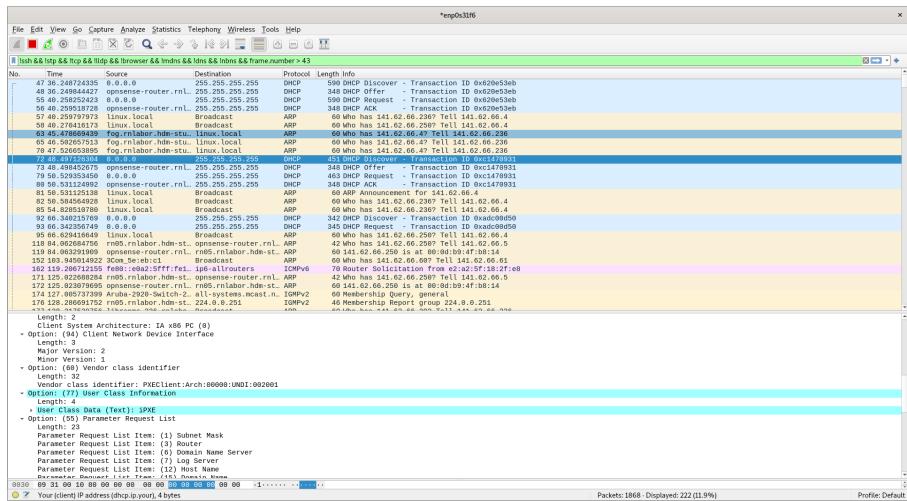


Abbildung 11: Bootprozess: DHCP-Requests des Netzwerbootloaders iPXE

Rechner in einem Netzwerk kommunizieren können, da dafür jeder Computer eine eigene IP benötigt.

Grundlegend funktioniert DHCP mithilfe von vier Nachrichtentypen. Es gibt den DHCP-Discover, welcher den DHCP-Server in erster Linie benachrichtigen will, dass eine neue IP verlangt wird. Der Server antwortet daraufhin mit einer Offer, welche eine IP reserviert und diese dem Client anbietet. Außerdem enthält die Offer die IP des DHCP-Servers, die Subnetzmaske und die Lease-Time. Danach kann der Client mit einer DHCP-Request die angebotene IP anfordern. Wenn das in Ordnung ist, antwortet der DHCP-Server mit einem DHCP-Acknowledge.

Vergleicht den Ablauf, wenn Sie den DHCP-Ablauf per ipconfig /release und ipconfig /renew initialisieren

Mittels der folgenden Commands wurde eine IP-Adresse freigegeben und eine neue angefordert.

```
1 # dhclient -r # Release der IP-Adresse
2 # dhclient # Anfrage einer neuen IP-Adresse
```

No.	Time	Source	Destination	Protocol	Length Info
19	15.392045061	0.0.0.0	255.255.255.255	DHCP	342 DHCP Discover - Transaction ID 0x70ef81d
20	15.393517126	0.0.0.0	255.255.255.255	DHCP	342 DHCP Request - Transaction ID 0x70ef81d
21	15.408801806	linux.local	Broadcast	ARP	69 Who has 141.02.66.250? Tell 141.02.66.4

Dem bereits hochgefahrenen Rechner wird eine neue IP zugeordnet. Wenn wir die IP Zuweisung auf diese weise neu initiieren dann ist der DHCP Ablauf deutlich kürzer, da beim Booten unter der Haube noch deutlich mehr gemacht werden muss (es muss e.g. keine DHCP-Request des BIOS zum Netzwerkboot getätigkt werden).

2.4 DNS

Dokumentieren Sie den Ablauf bei einer DNS-Abfrage

Fall 1: DNS-Server 141.62.66.250:

Mittels folgendem Command wurde eine DNS-Abfrage gemacht:

```
1 $ dig @141.62.66.250 google.com
2 google.com.      163 IN  A  142.250.186.174
```

dns & frame.number < 20						
No.	Time	Source	Destination	Protocol	Length	Info
11	1.357358000	rn05.rnlabor.hdm-st..	one.one.one.one	DNS	93	Standard query 0xa276 A google.com OPT
12	1.371692978	opnsense-router.rnL..	rn05.rnlabor.hdm-st..	DNS	97	Standard query response 0xa276 A google.com A 142.250.180.174 OPT

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:

```
1 $ dig @1.1.1.1 +noall +answer google.com
2 google.com.      231 IN  A  142.250.185.110
```

dns & frame.number < 20						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	rn05.rnlabor.hdm-st..	one.one.one.one	DNS	93	Standard query 0x6247 A google.com OPT
2	0.005982935	one.one.one.one	rn05.rnlabor.hdm-st..	DNS	97	Standard query response 0x6247 A google.com A 142.250.185.110 OPT
4	1.205820780	rn05.rnlabor.hdm-st..	opnsense-router.rnL..	DNS	84	Standard query 0xdab2 PTR 5.66.62.141.in-addr.arpa
5	1.205849397	rn05.rnlabor.hdm-st..	opnsense-router.rnL..	DNS	88	Standard query 0x8083 PTR 1.1.1.1.in-addr.arpa
6	1.207179251	opnsense-router.rnL..	rn05.rnlabor.hdm-st..	DNS	127	Standard query response 0xdab2 PTR 5.66.62.141.in-addr.arpa PTR rn05.rnlabor.hdm-stuttgart.de
7	1.297611398	opnsense-router.rnL..	rn05.rnlabor.hdm-st..	DNS	109	Standard query response 0x8083 PTR 1.1.1.1.in-addr.arpa PTR one.one.one.one

Abbildung 13: Ablauf der Anfrage

Bei der DNS Anfrage über Cloudflare erscheinen weitere DNS-Requests über DNS Reverse-Zones. Dies wird daran liegen, dass wir über den Router mit dem Internet kommunizieren.

Fall 3: DNS-Server 8.8.8.9 (DNS-Dienst ist dort nicht installiert):

Mittels folgendem Command wurde eine DNS-Abfrage gemacht:

```
1 $ dig @8.8.8.9 +noall +answer google.com
2 ;;; connection timed out; no servers could be reached
```

Wie im Bild zu sehen ist, bekommen wir den Response `No such name PTR 9.8.8.8`.

Wie erkennen Sie mit Wireshark, dass “versehentlich” ein falscher DNS-Server eingetragen wurde?

Es gibt eine Antwort, welche auf eine nicht gültige IP-Adresse hinweist (Siehe oben).

No.	Time	Source	Destination	Protocol	Length	Info
3	0.572498372	rn05.rnlabor.hdm-st..	8.8.8.9	DNS	93	Standard query 0x73f9 A google.com OPT
5	1.088465481	rn05.rnlabor.hdm-st..	opnsense.rnlabor.hdm-st..	DNS	81	Standard query 0x74b6 PTR 9.8.8.8.in-addr.arpa
7	1.089965123	opnsense.rnlabor.hdm-st..	rn05.rnlabor.hdm-st..	DNS	89	Standard query 0x74b6 PTR 9.8.8.8.in-addr.arpa
8	1.0998026625	opnsense.rnlabor.hdm-st..	rn05.rnlabor.hdm-st..	DNS	148	Standard query response @0x74b6 No such name PTR 9.8.8.8.in-addr.arpa SOA ns1.google.com
13	2.087996807	rn05.rnlabor.hdm-st..	opnsense.rnlabor.hdm-st..	DNS	81	Standard query 0x4fb6 PTR 259.66.62.141.in-addr.arpa
17	2.089192380	opnsense.rnlabor.hdm-st..	rn05.rnlabor.hdm-st..	DNS	163	Standard query response @0x7fb6 PTR 259.66.62.141.in-addr.arpa PTR opnsense-router.rnlabor.hdm-st..
21	2.089192388	rn05.rnlabor.hdm-st..	opnsense.rnlabor.hdm-st..	DNS	89	Standard query 0x7fb6 PTR 19.75.254.169.in-addr.arpa
23	3.087945863	rn05.rnlabor.hdm-st..	opnsense.rnlabor.hdm-st..	DNS	84	Standard query 0xfc6 PTR 251.0.0.224.in-addr.arpa
24	3.087959318	rn05.rnlabor.hdm-st..	opnsense.rnlabor.hdm-st..	DNS	88	Standard query 0x1f24 PTR 255.255.254.169.in-addr.arpa
25	3.088893145	opnsense.rnlabor.hdm-st..	rn05.rnlabor.hdm-st..	DNS	145	Standard query response @0x59b No such name PTR 19.75.254.169.in-addr.arpa SOA localhost
26	3.089011764	opnsense.rnlabor.hdm-st..	rn05.rnlabor.hdm-st..	DNS	141	Standard query response @0xfc6 No such name PTR 251.0.0.224.in-addr.arpa SOA sns.dns.icann.org
27	3.089125772	opnsense.rnlabor.hdm-st..	rn05.rnlabor.hdm-st..	DNS	147	Standard query response @0x1f24 No such name PTR 255.255.254.169.in-addr.arpa SOA localhost

Abbildung 14: Ablauf der Anfrage

2.5 ARP

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
24	116.6151576313	linux-2.local	Broadcast	ARP	42	Who has 141.62.66.5 Tell 141.62.66.5
243	116.61515867288	linux-3.local	linux-2.local	ARP	69	141.62.66.6 is at 4c:52:62:9e:54:2b
231	115.673184735	linux-3.local	linux-2.local	ARP	69	Who has 141.62.66.5 Tell 141.62.66.6

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?

Als Rahmentyp wird Ethernet II verwendet.

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
219	110.515867288	linux-3.local	linux-2.local	ARP	68	141.62.66.6 is at 4c:52:62:0e:54:2b
231	115.673164735	linux-3.local	linux-2.local	ARP	68	Who has 141.62.66.5? Tell 141.62.66.6
232	115.673186783	linux-3.local	linux-2.local	ARP	42	141.62.66.5 is at 4c:52:62:0e:54:8b


```

Frame 214: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface enp0s31f6, id 0
Ethernet II (ether broadcast) (08:00:22:ff:ff:ff)
  Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  Source: linux-2.local (4c:52:62:0e:54:2b)
  Type: ARP (0x0806)
  Address Resolution Protocol (request)

  Frame 214: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface enp0s31f6, id 0
  Ethernet II (ether broadcast) (08:00:22:ff:ff:ff)
  Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  Source: linux-2.local (4c:52:62:0e:54:2b)
  Type: ARP (0x0806)
  Address Resolution Protocol (request)

```

Abbildung 16: Verwendetes Ethernet-Frame

Beobachten Sie die Veränderung in der ARP-Tabelle Ihres Rechners

Zuvor:

```

1 $ ip neigh show
2 141.62.66.6 dev enp0s31f6 lladdr 4c:52:62:0e:54:2b STALE
3 141.62.66.250 dev enp0s31f6 lladdr 00:0d:b9:4f:b8:14 STALE
4 141.62.66.13 dev enp0s31f6 lladdr 4c:52:62:0e:54:5d STALE
5 141.62.66.236 dev enp0s31f6 lladdr 26:c5:04:8a:fa:eb STALE

```

Danach:

```

1 $ ip neigh show
2 141.62.66.6 dev enp0s31f6 lladdr 4c:52:62:0e:54:2b STALE
3 141.62.66.250 dev enp0s31f6 lladdr 00:0d:b9:4f:b8:14 STALE
4 141.62.66.4 dev enp0s31f6 lladdr 4c:52:62:0e:53:eb STALE
5 141.62.66.13 dev enp0s31f6 lladdr 4c:52:62:0e:54:5d STALE
6 141.62.66.236 dev enp0s31f6 lladdr 26:c5:04:8a:fa:eb STALE

```

2.6 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.

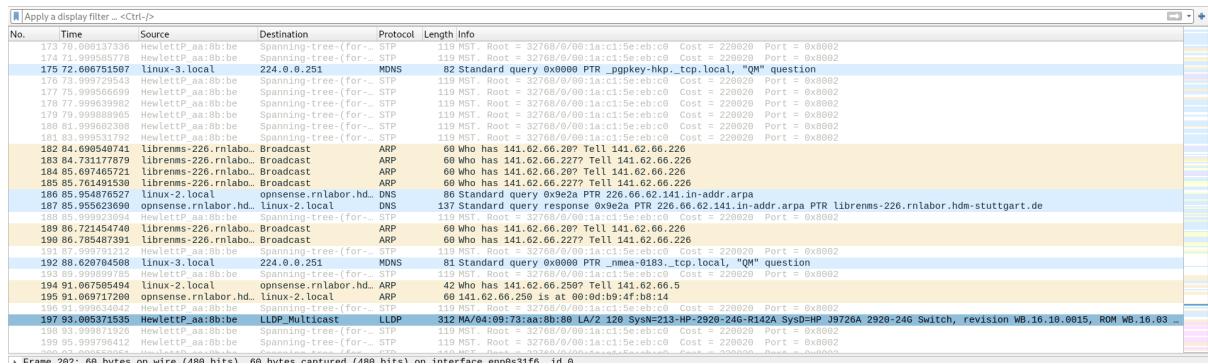


Abbildung 17: Aufzeichnung der ARP-Requests

Haben Sie noch weitere Protokolle “eingefangen”, die offensichtlich im Labor Rechnernetze keinen Sinn machen?

Aus dem Screenshot lässt sich aus der MDNS-Nachricht der `_nmea-0183._tcp.local` Service-String entnehmen. NMEA0183 ist ein Standard, welcher für die Kommunikation zwischen Navigationsgeräten auf Schiffen definiert wurde. Da es mitunter für die Kommunikation zwischen GPS-Empfänger und PCs verwendet wird, macht es in unserem Netzwerk wenig Sinn.

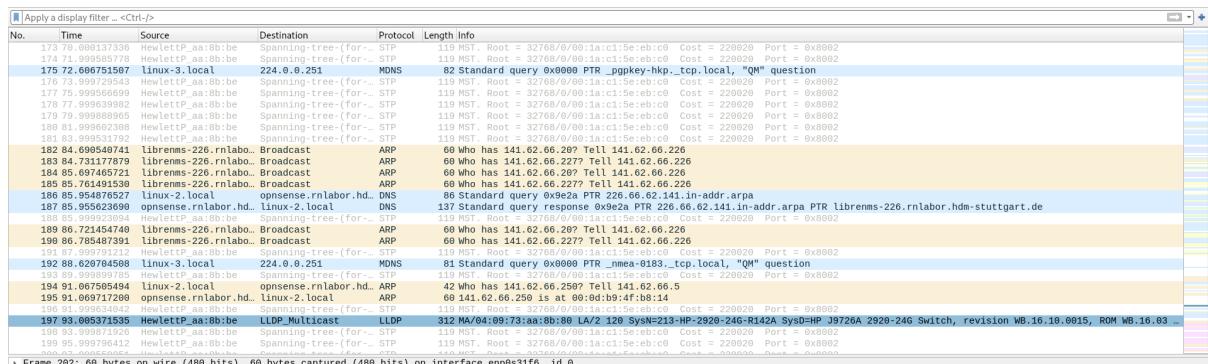


Abbildung 18: Aufzeichnung der ARP-Requests; hier ist das Protokoll zu sehen

Wie sieht es mit UPnP im Labor aus? Auf welchen Maschinen von welchem Hersteller läuft der

Dienst? Mit welchem Wireshark-Filter „fischen“ Sie den Traffic heraus?

Es existiert ein Gerät von AVMAudio im Netzwerk, welches über UPnP angesteuert wird. Dies wird immer von demselben Gerät angesteuert, welches über eine Link-Lokale Adresse verfügt, was dafür sorgt, dass es nur innerhalb des Netzwerkes erreicht werden kann. Diese Adressen werden nicht geroutet, sprich die Geräte müssen durch einen Switch etc. verbunden sein. Es kann über den Display-Filter „herausgefischt werden“, indem man nach [SSDP](#) filtert.

No.	Time	Source	Destination	Protocol	Length Info
826	235.113864599	fe80::5e49:79ff:fe6.. ff02::c	SSDP	365 NOTIFY * HTTP/1.1	
827	235.115078419	fe80::5e49:79ff:fe6.. ff02::c	SSDP	375 NOTIFY * HTTP/1.1	
828	235.115520628	fe80::5e49:79ff:fe6.. ff02::c	SSDP	411 NOTIFY * HTTP/1.1	
829	235.117651013	fe80::5e49:79ff:fe6.. ff02::c	SSDP	411 NOTIFY * HTTP/1.1	
830	240.116104287	fe80::5e49:79ff:fe6.. ff02::c	SSDP	364 NOTIFY * HTTP/1.1	
840	240.116104287	fe80::5e49:79ff:fe6.. ff02::c	SSDP	372 NOTIFY * HTTP/1.1	
841	240.119442125	fe80::5e49:79ff:fe6.. ff02::c	SSDP	435 NOTIFY * HTTP/1.1	
842	240.113785421	fe80::5e49:79ff:fe6.. ff02::c	SSDP	372 NOTIFY * HTTP/1.1	
843	240.114125393	fe80::5e49:79ff:fe6.. ff02::c	SSDP	411 NOTIFY * HTTP/1.1	
844	240.116104287	fe80::5e49:79ff:fe6.. ff02::c	SSDP	372 NOTIFY * HTTP/1.1	
845	240.116104287	fe80::5e49:79ff:fe6.. ff02::c	SSDP	431 NOTIFY * HTTP/1.1	
846	240.120316833	fe80::5e49:79ff:fe6.. ff02::c	SSDP	399 NOTIFY * HTTP/1.1	
847	240.122478594	fe80::5e49:79ff:fe6.. ff02::c	SSDP	443 NOTIFY * HTTP/1.1	
848	240.124712671	fe80::5e49:79ff:fe6.. ff02::c	SSDP	427 NOTIFY * HTTP/1.1	
849	240.126991671	fe80::5e49:79ff:fe6.. ff02::c	SSDP	427 NOTIFY * HTTP/1.1	
850	240.127001479	fe80::5e49:79ff:fe6.. ff02::c	SSDP	439 NOTIFY * HTTP/1.1	
851	241.119212915	fe80::5e49:79ff:fe6.. ff02::c	SSDP	364 NOTIFY * HTTP/1.1	
852	241.119541617	fe80::5e49:79ff:fe6.. ff02::c	SSDP	373 NOTIFY * HTTP/1.1	
853	241.119892237	fe80::5e49:79ff:fe6.. ff02::c	SSDP	436 NOTIFY * HTTP/1.1	
854	241.114269272	fe80::5e49:79ff:fe6.. ff02::c	SSDP	373 NOTIFY * HTTP/1.1	
855	241.114551951	fe80::5e49:79ff:fe6.. ff02::c	SSDP	412 NOTIFY * HTTP/1.1	

From: 192.168.1.10 (eth0) [192.168.1.10] (Wireshark - Network traffic captured (0.000 bits/s) on interface enp0s10, id 0)

- Ethernet II, Src: AVMAudio_8a:a9:78 (5c:49:79:6a:a9:78), Dst: IPv6cast.8c (33:33:00:00:00:80:0c)
- Internet Protocol Version 6, Src: fe80::5e49:79ff:fe6a:a978, Dst: ff02::c
- User Datagram Protocol, Src Port: 19000, Dst Port: 19000
- Simple Service Discovery Protocol

Abbildung 19: Aufzeichnung des SSDP-Protokolls

2.7 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	7.598625	100.64.84.66	141.70.124.5	DNS	80	Standard query 0x189d A news.ycombinator.com
715	7.598981	100.64.84.66	141.70.124.5	DNS	80	Standard query 0x58df AAAA news.ycombinator.com
716	7.608834	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 209.216.230.248 NS ns=1411.awsdns-48.org NS ns=1914.awsdns-47.co
718	7.614386	100.64.84.66	209.216.230.248	TCP	78	49314 -> 443 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSeq=2512581059 TSeq=0 SACK_PERM=1
719	7.765210	209.216.230.248	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 TSeq=2045828460 TSeq=2512581059
720	7.765334	100.64.84.66	209.216.230.248	TCP	66	49314 -> 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSeq=2512581211 TSeq=2045828460
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 TSeq=2045828612 TSeq=2512581211 [TCP segment of a reassembled PDU]
724	7.917495	209.216.230.248	100.64.84.66	TLSv1..	1062	Certificate, Certificate Status, Server Key Exchange, Server Hello Done
725	7.917581	100.64.84.66	209.216.230.248	TCP	66	49314 -> 443 [ACK] Seq=518 Ack=3893 Win=127872 Len=0 TSeq=2512581363 TSeq=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 TSeq=2512581363 TSeq=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..	788	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 TSeq=2045828783 TSeq=2512581383
730	8.093869	209.216.230.248	100.64.84.66	TLSv1..	324	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
731	8.093957	100.64.84.66	209.216.230.248	TCP	66	49314 -> 443 [ACK] Seq=1364 Ack=4151 Win=65664 Len=1448 TSeq=2045828788 TSeq=2512581383
732	8.096295	209.216.230.248	100.64.84.66	TCP	1514	443 -> 49314 [ACK] Seq=4151 Ack=1364 Win=65664 Len=1448 TSeq=2045828789 TSeq=2512581383 [TCP segment of a reassembled PDU]
733	8.096296	209.216.230.248	100.64.84.66	TCP	1514	443 -> 49314 [ACK] Seq=5599 Ack=1364 Win=65664 Len=1448 TSeq=2045828789 TSeq=2512581383 [TCP segment of a reassembled PDU]
734	8.096296	209.216.230.248	100.64.84.66	TCP	1514	443 -> 49314 [ACK] Seq=7047 Ack=1364 Win=65664 Len=1448 TSeq=2045828789 TSeq=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 TSeq=2045828789 TSeq=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=10558 Win=124608 Len=0 TSeq=2512581542 TSeq=2045828789
738	8.096484	100.64.84.66	209.216.230.248	TCP	66	[TCP Window Update] 49314 -> 443 [ACK] Seq=1364 Ack=10558 Win=131072 Len=0 TSeq=2512581542 TSeq=2045828789
739	8.223532	100.64.84.66	209.216.230.248	TLSv1..	691	Application Data
740	8.252798	100.64.84.66	209.216.230.248	TCP	78	49315 -> 443 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSeq=3827897587 TSeq=0 SACK_PERM=1
741	8.374585	209.216.230.248	100.64.84.66	TCP	1514	443 -> 49314 [ACK] Seq=10558 Ack=1989 Win=65664 Len=1448 TSeq=2045829070 TSeq=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=1364 Ack=12763 Win=128822 Len=0 TSeq=2512581820 TSeq=2045829070
744	8.374981	100.64.84.66	209.216.230.248	TLSv1..	674	Application Data
745	8.413591	209.216.230.248	100.64.84.66	TCP	66	49315 -> 443 [SYN, ACK, EON] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 TSeq=1535760379 TSeq=0
751	8.413596	100.64.84.66	209.216.230.248	TCP	66	49315 -> 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSeq=2512581754 TSeq=1535760379
752	8.424337	100.64.84.66	209.216.230.248	TLSv1..	585	Client Hello
759	8.527967	209.216.230.248	100.64.84.66	TCP	1514	443 -> 49314 [ACK] Seq=12763 Ack=2597 Win=65664 Len=1448 TSeq=2045829221 TSeq=2512581821 [TCP segment of a reassembled PDU]
760	8.527968	209.216.230.248	100.64.84.66	TLSv1..	793	Application Data
761	8.527151	100.64.84.66	209.216.230.248	TCP	66	49314 -> 443 [ACK] Seq=2597 Ack=14938 Win=128896 Len=0 TSeq=2512581972 TSeq=2045829221
762	8.591413	209.216.230.248	100.64.84.66	TLSv1..	222	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=520 Ack=131584 Win=0 TSeq=3827897926 TSeq=1535760550
764	8.591689	100.64.84.66	209.216.230.248	TLSv1..	117	Change Cipher Spec, Encrypted Handshake Message
765	8.622083	100.64.84.66	209.216.230.248	TLSv1..	719	Application Data
766	8.772916	209.216.230.248	100.64.84.66	TCP	1514	443 -> 49314 [ACK] Seq=14938 Ack=3258 Win=65664 Len=1448 TSeq=2045829468 TSeq=2512582067 [TCP segment of a reassembled PDU]
767	8.772916	209.216.230.248	100.64.84.66	TCP	66	49315 -> 443 [ACK] Seq=3258 Ack=1448 Win=0 TSeq=2045829468 TSeq=2512582067

Abbildung 20: Initierung einer HTTP-TCP-Sitzung

Können Sie den 3-Way-Handshake erkennen? Markieren Sie ihn in der Dokumentation. Welche TCP-Optionen sind beim Handshake aktiviert und welche Bedeutung haben sie?

No.	Time	Source	Destination	Protocol	Length	Info
714	7.598625	100.64.84.66	141.70.124.5	DNS	80	Standard query 0x189d A news.ycombinator.com
715	7.599881	100.64.84.66	141.70.124.5	DNS	80	Standard query 0x88df AAA news.ycombinator.com SDA ns=225.awsdns-28.com
716	7.608834	141.70.124.66	100.64.84.66	DNS	158	Standard query response 0x58df AAA news.ycombinator.com SDA 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 209.216.230.248 NS ns=1411.awsdns-48.org NS ns=1914.awsdns-47.co
718	7.614386	100.64.84.66	209.216.230.248	TCP	78	49314 - 443 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 Tsvl=2512581059 Tsecr=0 SACK_PERM=1
719	7.615219	209.216.230.248	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 Tsvl=2045828460 Tsecr=2512581059
720	7.615334	100.64.84.66	209.216.230.248	TCP	66	49314 - 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 Tsvl=2512581211 Tsecr=2045828460
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=1448 Tsvl=2045828612 [TCP segment of a reassembled PDU]
724	7.917495	209.216.230.248	100.64.84.66	TLSv1..	1062	Certificate, Certificate Status, Server Key Exchange, Server Hello Done
725	7.917581	100.64.84.66	209.216.230.248	TCP	66	49314 - 443 [ACK] Seq=518 Ack=3893 Win=127872 Len=0 Tsvl=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 Tsvl=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 Tsvl=2045828783 Tsecr=2512581383
730	8.093869	209.216.230.248	100.64.84.66	TLSv1..	324	Session Ticket, Change Cipher Spec, Encrypted Handshake Message
731	8.093957	100.64.84.66	209.216.230.248	TCP	66	49314 - 443 [ACK] Seq=1364 Ack=4151 Win=130752 Len=0 Tsvl=2512581539 Tsecr=2045828788
732	8.096295	209.216.230.248	100.64.84.66	TCP	1514	443 - 49314 [ACK] Seq=4151 Ack=1364 Win=65664 Len=1448 Tsvl=2045828789 Tsecr=2512581383 [TCP segment of a reassembled PDU]
733	8.096296	209.216.230.248	100.64.84.66	TCP	1514	443 - 49314 [ACK] Seq=1364 Win=65664 Len=1448 Tsvl=2045828789 Tsecr=2512581383 [TCP segment of a reassembled PDU]
734	8.096296	209.216.230.248	100.64.84.66	TCP	1514	443 - 49314 [ACK] Seq=7047 Ack=1364 Win=65664 Len=1448 Tsvl=2045828789 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 Tsvl=2045828789 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=10558 Win=124608 Len=0 Tsvl=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=10558 Win=131072 Len=0 Tsvl=2512581542 Tsecr=2045828789
739	8.223532	100.64.84.66	209.216.230.248	TLSv1..	691	Application Data
740	8.252798	100.64.84.66	209.216.230.248	TCP	78	49315 - 443 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 Tsvl=3827897587 Tsecr=0 SACK_PERM=1
741	8.374985	209.216.230.248	100.64.84.66	TCP	1514	443 - 49314 [ACK] Seq=10558 Ack=1989 Win=1448 Tsvl=2045829070 Tsecr=2512581669 [TCP segment of a reassembled PDU]
742	8.374987	209.216.230.248	100.64.84.66	TLSv1..	823	Application Data
743	8.375053	100.64.84.66	209.216.230.248	TCP	66	49314 - 443 [ACK] Seq=1989 Ack=12763 Win=128832 Len=0 Tsvl=2512581820 Tsecr=2045829070
744	8.376901	100.64.84.66	209.216.230.248	TLSv1..	674	Application Data
750	8.419834	209.216.230.248	100.64.84.66	TCP	74	443 - 49315 [SYN, ACK, EON] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 Tsvl=1535760379 Tsecr=3827897587
751	8.419506	100.64.84.66	209.216.230.248	TCP	66	49315 - 443 [ACK] Seq=1364 Ack=1 Win=131712 Len=0 Tsvl=2512581821 Tsecr=2045829070
752	8.424337	100.64.84.66	209.216.230.248	TLSv1..	585	Client Hello
759	8.527697	209.216.230.248	100.64.84.66	TCP	1514	443 - 49314 [ACK] Seq=12763 Ack=2597 Win=65664 Len=1448 Tsvl=2045829221 Tsecr=2512581821 [TCP segment of a reassembled PDU]
760	8.527968	209.216.230.248	100.64.84.66	TLSv1..	793	Application Data
761	8.527151	100.64.84.66	209.216.230.248	TCP	66	49314 - 443 [ACK] Seq=2597 Ack=14938 Win=128896 Len=0 Tsvl=2512581972 Tsecr=2045829221
762	8.591413	209.216.230.248	100.64.84.66	TLSv1..	222	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=528 Ack=157 Win=131584 Len=0 Tsvl=3827897924 Tsecr=1535760550
764	8.591689	100.64.84.66	209.216.230.248	TLSv1..	117	Change Cipher Spec, Encrypted Handshake Message
765	8.622083	100.64.84.66	209.216.230.248	TLSv1..	719	Application Data
766	8.772916	209.216.230.248	100.64.84.66	TCP	1514	443 - 49314 [ACK] Seq=14938 Ack=3258 Win=65664 Len=1448 Tsvl=2045829468 Tsecr=2512582067 [TCP segment of a reassembled PDU]
767	8.772916	209.216.230.248	100.64.84.66	TCP	66	49315 - 443 [ACK] Seq=14938 Ack=3258 Win=65664 Len=1448 Tsvl=2045829468 Tsecr=2512582067 [TCP segment of a reassembled PDU]

Abbildung 21: 3-Way-Handshake.

No.	Time	Source	Destination	Protocol	Length	Info
716	7.608834	141.70.124.5	100.64.84.66	DNS	158	Standard query response 0x58df AAAA news.ycombinator.com SDA 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 209.216.230.248 NS ns=1411.awsdns-48.org NS ns=1914.awsdns-47.co
718	7.614386	100.64.84.66	209.216.230.248	TCP	78	49314 - 443 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 Tsvl=2512581059 Tsecr=0 SACK_PERM=1
719	7.615219	209.216.230.248	100.64.84.66	TCP	74	443 - 49314 [SYN, ACK, ECN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 Tsvl=2045828460 Tsecr=2512581059
720	7.615334	100.64.84.66	209.216.230.248	TCP	66	49314 - 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 Tsvl=2512581211 Tsecr=2045828460
721	7.765826	100.64.84.66	209.216.230.248	TLSv1..	583	Client Hello
Frame Number: 718						
Frame Length: 78 bytes (624 bits)						
Capture Length: 78 bytes (624 bits)						
[Frame is marked: False]						
[Frame is ignored: False]						
[Protocols in frame: eth:ethertype:ip:tcp]						
[Coloring Rule Name: TCP SYN/FIN]						
[Coloring Rule String: tcp.flags & 0x02 tcp.flags.fin == 1]						
> Ethernet II, Src: Apple_44:f3:0e (ad:83:e7:44:f3:0e), Dst: Juniper_N_9a:93:ce (b0:a8:6e:9a:93:ce)						
> Internet Protocol Version 4, Src: 100.64.84.66, Dst: 209.216.230.240						
Transmission Control Protocol, Src Port: 49314, Dst Port: 443, Seq: 0, Len: 0						
Source Port: 49314						
Destination Port: 443						
[Stream index: 10]						
[TCP Segment Len: 0]						
Sequence Number: 0 (relative sequence number)						
Sequence Number (raw): 3747062828						
[Next Sequence Number: 1 (relative sequence number)]						
Acknowledgment Number: 0						
Acknowledgment Number (raw): 0						
1011 = Header Length: 44 bytes (11)						
> IFlags: 0x02 (SYN, EON, CWR)						
Window: 65535						
[calculated window size: 65535]						
Checksum: 0x7f87 [unverified]						
[checksum status: unverified]						
Urgent Pointer: 0						
Options (24 bytes)						
> TCP Option - Maximum segment size: 1460 bytes						
> TCP Option - No-Operation (NOP)						
> TCP Option - Window scale: 6 (multiply by 64)						
> TCP Option - No-Operation (NOP)						
> TCP Option - No-Operation (NOP)						
> TCP Option - Timestamps: Tsvl 2512581059, Tsecr 0						
> TCP Option - SACK permitted						
> TCP Option - End of Option List (EOL)						

Das SYN-Segment enthält die Optionen Maximum Segment Size, Window scale, Timestamps und SACK (Selective Acknowledgement).

No.	Time	Source	Destination	Protocol	Length	Info
716	7.688834	141.70.124.5	100.64.84.66	DNS	158	Standard query response 0x58df AAAA news.ycombinator.com S0A 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.l
718	7.614386	100.64.84.66	209.216.230.240	TCP	78	49314 → 443 [SYN, ECN, CWR] Seq=0 Win=65535 Len=1460 WS=64 Tsva=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 Tsva=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 Tsva=2512581211 Tsecr=2045828460
721	7.765826	100.64.84.66	209.216.230.240	TLSv1_	583	Client Hello

[time delta from previous displayed frame: 0.158904000 seconds]
[time since reference or first frame: 7.765210000 seconds]
Frame Number: 719
Frame Length: 74 bytes (592 bits)
Capture Length: 74 bytes (592 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ether:type:ip:tcp]
[Coloring Rule Name: TCP SYN/FIN]
[Coloring Rule String: tcp.flags & 0x02 || tcp.flags.fin == 1]
> Ethernet II, Src: Juniper_N_9a:93:ce (b0:a8:6e:9a:93:ce), Dst: Apple_44:f3:0e (a4:83:e7:44:f3:0e)
> Internet Protocol Version 4, Src: 209.216.230.240, Dst: 100.64.84.66
▼ Transmission Control Protocol, Src Port: 443, Dst Port: 49314, Seq: 0, Ack: 1, Len: 0
 Source Port: 443
 Destination Port: 49314
 [Stream index: 10]
 [TCP Segment Len: 0]
 Sequence Number: 0 (relative sequence number)
 Sequence Number (raw): 2792502608
 [Next Sequence Number: 1 (relative sequence number)]
 Acknowledgment Number: 1 (relative ack number)
 Acknowledgment number (raw): 3747062829
 1010 = Header Length: 40 bytes (10)
> Flags: 0x052 (SYN, ACK, EON)
 Window: 65535
 [Calculated window size: 65535]
 Checksum: 0xd5db [unverified]
 [Checksum Status: Unverified]
 Urgent Pointer: 0
▼ Options: (28 bytes), Maximum segment size, No-Operation (NOP), Window scale, SACK permitted, Timestamps
 > TCP Option - Maximum segment size: 1460 bytes
 > TCP Option - No-Operation (NOP)
 > TCP Option - Window scale: 6 (multiply by 64)
 > TCP Option - SACK permitted
 > TCP Option - Timestamps: Tsva 2045828460, Tsecr 2512581059
 > [SEQ/ACK analysis]

Das SYN-ACK-Segment verwendet wieder die Optionen Maximum Segment Size, Window scale, SACK und Timestamps.

No.	Time	Source	Destination	Protocol	Length	Info
716	7.688834	141.70.124.5	100.64.84.66	DNS	158	Standard query response 0x58df AAAA news.ycombinator.com S0A 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.l
718	7.614386	100.64.84.66	209.216.230.240	TCP	78	49314 → 443 [SYN, ECN, CWR] Seq=0 Win=65535 Len=1460 WS=64 Tsva=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 Tsva=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 Tsva=2512581211 Tsecr=2045828460
721	7.765826	100.64.84.66	209.216.230.240	TLSv1_	583	Client Hello

[time delta from previous captured frame: 0.000124000 seconds]
[time delta from previous displayed frame: 0.000124000 seconds]
[time since reference or first frame: 7.765334000 seconds]
Frame Number: 720
Frame Length: 66 bytes (528 bits)
Capture Length: 66 bytes (528 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ether:type:ip:tcp]
[Coloring Rule Name: TCP]
[Coloring Rule String: tcp]
> Ethernet II, Src: Apple_44:f3:0e (a4:83:e7:44:f3:0e), Dst: Juniper_N_9a:93:ce (b0:a8:6e:9a:93:ce)
> Internet Protocol Version 4, Src: 100.64.84.66, Dst: 209.216.230.240
▼ Transmission Control Protocol, Src Port: 49314, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
 Source Port: 49314
 Destination Port: 443
 [Stream index: 10]
 [TCP Segment Len: 0]
 Sequence Number: 1 (relative sequence number)
 Sequence Number (raw): 3747062829
 [Next Sequence Number: 1 (relative sequence number)]
 Acknowledgment Number: 1 (relative ack number)
 Acknowledgment number (raw): 2792502609
 1000 = Header Length: 32 bytes (8)
> Flags: 0x010 (ACK)
 Window: 2058
 [Calculated window size: 131712]
 [Window size scaling factor: 64]
 Checksum: 0xfc44 [unverified]
 [Checksum Status: Unverified]
 Urgent Pointer: 0
▼ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
 > TCP Option - No-Operation (NOP)
 > TCP Option - No-Operation (NOP)
 > TCP Option - Timestamps: Tsva 2512581211, Tsecr 2045828460
 > [SEQ/ACK analysis]

Das ACK Segment hat nur die Timestamps-Option gesetzt.

Die Maximum Segment Size gibt die maximale Anzahl an Daten in Bytes an, die pro Segment akzeptiert werden. Der Window scale factor ist dazu da, die zuvor gesetzte maximale window-size über 65535 Bytes zu setzen. Der Timestamp misst die derzeitige Roundtrip time. Dadurch kann man den retransmission-timer jederzeit neu evaluieren. Selective Acknowledgement wird benutzt, um bei verlorenen Segmenten wirklich nur die fehlenden retransmitten zu müssen.

Dokumentieren und erläutern Sie die Verwendung der Portnummern bei der Dienstanfrage und der Beantwortung des Dienstes durch den Server.

Unser Computer sendet von Port 49314 an Port 443, welcher für HTTPS genutzt wird. Unser Port ist dabei arbiträr vom System gewählt, der HTTPS Port ist allerdings fest für HTTPS reserviert. Mit einem Port ist ein Dienst eines Rechners gekennzeichnet. Die Kombination aus Port und IP ergibt einen Socket. Wir senden unsere Nachrichten also an den Socket 209.216.230.240:443.

Klicken Sie auf der Website ein anderes Bild / Link an. Beobachten und dokumentieren Sie: wie verändert sich der TCP-Ablauf?

No.	Time	Source	Destination	Protocol	Length	Info
495	6.142842	2001:7c7:2126:4b00..	update.googleapis..	TLSv1..	150	Change Cipher Spec, Application Data
496	6.147080	update.googleapis..	2001:7c7:2126:4b00..	TLSv1..	694	Application Data, Application Data
497	6.147140	2001:7c7:2126:4b00..	update.googleapis..	TCP	86	49364 - https(443) [ACK] Seq=5293 Win=130432 Len=0 TStamp=132898483 TSect=1399184461
528	7.196215	100.64.84.66	news.ycombinator.c..	TCP	78	49365 - https(443) [SYN, EON, CWR] Seq=0 Win=65535 Len=64 TStamp=1372401487 TSect=0 SACK_PERM=1
529	7.351288	100.64.84.66	news.ycombinator.c..	TCP	74	49365 [SYN, ACK, EON] Seq=1 Win=0 MSS=64 SACK_PERM=1 TStamp=1918799133 TSect=3372401486
530	7.351406	100.64.84.66	news.ycombinator.c..	TCP	66	49365 - https(443) [ACK] Seq=1 Ack=1 Win=131712 Len=0 MSS=64 SACK_PERM=1 TStamp=1918799133 TSect=3372401486
531	7.352161	100.64.84.66	news.ycombinator.c..	TLSv1..	583	Client Hello
532	7.508863	news.ycombinator.c..	100.64.84.66	TLSv1..	1514	Server Hello
533	7.508863	news.ycombinator.c..	100.64.84.66	TCP	1514	https(443) - 49365 [ACK] Seq=1449 Ack=518 Win=65664 Len=1448 TStamp=1918799291 TSect=3372401642 [TCP segment of a reassembly]
534	7.508864	news.ycombinator.c..	100.64.84.66	TLSv1..	1062	Certificate, Certificate Status, Server Key Exchange, Server Hello Done
535	7.508923	100.64.84.66	news.ycombinator.c..	TCP	66	49365 - https(443) [ACK] Seq=518 Ack=3893 Win=127872 Len=0 TStamp=1372401800 TSect=1918799291
536	7.508918	100.64.84.66	news.ycombinator.c..	TCP	66	[TCP Window Update] 49365 - https(443) [ACK] Seq=518 Ack=3893 Win=131072 Len=0 TStamp=1372401800 TSect=1918799291
537	7.514919	100.64.84.66	news.ycombinator.c..	TLSv1..	192	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
538	7.515201	100.64.84.66	news.ycombinator.c..	TLSv1..	786	Application Data
539	7.670131	news.ycombinator.c..	100.64.84.66	TLSv1..	324	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
540	7.670264	100.64.84.66	news.ycombinator.c..	TCP	66	49365 - https(443) [ACK] Seq=1364 Ack=4151 Win=0 TStamp=1372401961 TSect=1918799452
541	7.671518	news.ycombinator.c..	100.64.84.66	TCP	1514	https(443) - 49365 [ACK] Seq=4151 Ack=1364 Win=65664 Len=1448 TStamp=1918799453 TSect=3372401806 [TCP segment of a reassembly]
542	7.671520	news.ycombinator.c..	100.64.84.66	TCP	1514	https(443) - 49365 [ACK] Seq=5599 Ack=1364 Win=1448 TStamp=1918799453 TSect=3372401806 [TCP segment of a reassembly]
543	7.671521	news.ycombinator.c..	100.64.84.66	TCP	1514	https(443) - 49365 [ACK] Seq=7077 Ack=1364 Win=65664 Len=1448 TStamp=1918799453 TSect=3372401806 [TCP segment of a reassembly]
544	7.671522	news.ycombinator.c..	100.64.84.66	TCP	1514	https(443) - 49365 [ACK] Seq=8493 Ack=1364 Win=1448 TStamp=1918799453 TSect=3372401806 [TCP segment of a reassembly]
545	7.671523	news.ycombinator.c..	100.64.84.66	TLSv1..	66	Application Data
546	7.671523	news.ycombinator.c..	100.64.84.66	TCP	66	49365 - https(443) [ACK] Seq=1364 Ack=10559 Win=124680 Len=0 TStamp=1372401962 TSect=1918799453
547	7.671793	100.64.84.66	news.ycombinator.c..	TCP	66	[TCP Window Update] 49365 - https(443) [ACK] Seq=1364 Ack=10559 Win=131072 Len=0 TStamp=1372401962 TSect=1918799453
548	7.844982	f800::2d1:f9cd:6126..	f800::b2a8:6eff:fe..	ICMPv6	86	Neighbor Solicitation for f800::b2a8:6eff:fe:8a93ce from a1:83:e7:44:f4:fe
549	7.845274	f800::2d1:f9cd:6126..	f800::b2a8:6eff:fe..	ICMPv6	78	Neighbor Advertisement for f800::b2a8:6eff:fe:8a93ce (rtr, sol)
550	10.587785	2001:7c7:2126:4b00..	lwn.net	TCP	98	49367 - https(443) [SYN, EON, CWR] Seq=0 Win=65535 Len=64 TStamp=1372401923 TSect=0 SACK_PERM=1
541	10.663242	2001:7c7:2126:4b00..	lwn.net	TCP	98	49367 - https(443) [SYN, EON, CWR] Seq=0 Win=65535 Len=64 TStamp=131163105 TSect=0 SACK_PERM=1
542	10.689637	lwn.net	2001:7c7:2126:4b00..	TCP	94	https(443) - 49367 [SYN, ACK, EON] Seq=0 Ack=0 Win=0 MSS=1448 SACK_PERM=1 TStamp=2318916019 TSect=4280199213 WS=1
543	10.689758	2001:7c7:2126:4b00..	lwn.net	TCP	86	49367 - https(443) [ACK] Seq=1 Ack=1 Win=131328 Len=0 TStamp=1372401923 TSect=1918799453
544	10.690211	2001:7c7:2126:4b00..	lwn.net	TLSv1..	603	Client Hello
545	10.753743	lwn.net	2001:7c7:2126:4b00..	TCP	94	https(443) - 49367 [SYN, ACK, EON] Seq=0 Ack=0 Win=131328 Len=0 TStamp=1372401923 TSect=1918799453
546	10.753857	2001:7c7:2126:4b00..	lwn.net	TCP	86	49368 - https(443) [ACK] Seq=1 Ack=1 Win=131328 Len=0 TStamp=1372401923 TSect=1918799453
547	10.754321	2001:7c7:2126:4b00..	lwn.net	TLSv1..	603	Client Hello
548	10.796688	lwn.net	2001:7c7:2126:4b00..	TCP	86	https(443) - 49367 [ACK] Seq=1 Ack=1 Win=131328 Len=0 TStamp=1372401923 TSect=1918799453
549	10.797292	lwn.net	2001:7c7:2126:4b00..	TLSv1..	1514	Server Hello
550	10.797293	lwn.net	2001:7c7:2126:4b00..	TCP	1514	https(443) - 49367 [ACK] Seq=1429 Ack=518 Win=29696 Len=1428 TStamp=2318916126 TSect=4280199320 [TCP segment of a reassembly]
651	10.797294	lwn.net	2001:7c7:2126:4b00..	TCP	1326	https(443) - 49367 [PSH, ACK] Seq=2857 Ack=518 Win=29696 Len=1240 TStamp=2318916126 TSect=4280199320 [TCP segment of a reassembly]
552	10.797481	2001:7c7:2126:4b00..	lwn.net	TCP	86	49367 - https(443) [ACK] Seq=518 Ack=4097 Win=0 Len=0 TStamp=1372401923 TSect=2318916126
653	10.797669	2001:7c7:2126:4b00..	lwn.net	TCP	86	[TCP Window Update] 49367 - https(443) [ACK] Seq=518 Ack=4097 Win=131072 Len=0 TStamp=1372401923 TSect=2318916126
654	10.798585	lwn.net	2001:7c7:2126:4b00..	TLSv1..	578	Certificate, Server Key Exchange, Server Hello Done

Abbildung 22: Es wird eine TCP-Verbindung zur neuen Seite (lwn.net) aufgebaut. Dies sieht man anhand des wiederholten TCP-Handshakes.

2.8 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
179 63. 999716934		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
171 65. 999536248		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
172 67. 999536248		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
173 70. 999517336		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
174 71. 999557778		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
176 73. 999728543		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
177 75. 999566698		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
177 76. 999566698		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
179 70. 999566698		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
180 81. 899602308		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
181 83. 999513792		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
184 85. 999523098		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
191 87. 999791212		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
192 88. 999566698		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
198 91. 999564392		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
199 93. 999871920		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
199 95. 999764421		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
200 97. 999556095		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
201 100. 000215793		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
203 102. 999566698		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
204 103. 999577382		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
206 105. 999642753		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
212 108. 000240078		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
213 109. 999891439		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
221 111. 999504588		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
223 113. 999504588		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
224 113. 999504588		HewlettP_aa:8b:be	Spanning-tree:(for- STP)	119 MST, Root = 32768/0/0:1a:c1:5e:eb:c0	Cost = 220020	Port = 0x8002
Frame 101: 119 bytes on wire (952 bits), 119 bytes captured (952 bits) on interface emps0s1f0, id 0						
IEEE 802.3 Ethernet						
- Destination: Spanning-tree-(for-brides).00 (01:80:c2:00:00:00)						
Address: Spanning-tree-(for-brides).00 (01:80:c2:00:00:00)						
....0.....0.... = LG bit: Globally unique address (factory default)						
....1.....0.... = IG bit: Group address (multicast/broadcast)						
- Source: HewlettP_aa:8b:be (04:09:73:aa:8b:be)						
Address: HewlettP_aa:8b:be (04:09:73:aa:8b:be)						
....0.....0.... = LG bit: Globally unique address (factory default)						
....0.....0.... = IG bit: Individual address (unicast)						
Length: 105						
Logical-Link Control						
Spanning Tree Protocol						

Abbildung 23: Aufzeichnung des STP-Protokolls

Welche MAC-Adresse hat ihr Nachbarrechner?

Durch einen `ping` konnten wir die MAC-Adresse des Switches herausfinden.

MAC-Adresse: `4c:52:62:0e:54:2b`

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	141.62.66.5	141.62.66.5	ICMP	88	Echo (ping) request id=0xe6cf, seq=1/256, ttl=64 (reply in 216)
2	217.110.51.145:1773	linux-2.local	linux-3.local	ICMP	98	Echo (ping) reply id=0x6c3f, seq=1/256, ttl=64 (request in 216)
3	219.110.51.145:5991	linux-2.local	linux-3.local	ICMP	98	Echo (ping) request id=0x6c3f, seq=2/252, ttl=64 (reply in 228)
4	220.110.51.145:8683	linux-2.local	linux-3.local	ICMP	98	Echo (ping) reply id=0x6c3f, seq=2/252, ttl=64 (request in 219)
5	222.110.51.145:2475	linux-2.local	linux-3.local	ICMP	98	Echo (ping) request id=0x6c3f, seq=3/768, ttl=64 (reply in 223)
6	Frame 2: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp0s31f6, id 0					
7	- Ethernet II Src: linux-2.local (4c:52:62:0e:54:2b), Dst: linux-3.local (4c:52:62:0e:54:2b)					
8	- Destination port: 5249 (Local Multicast Listener Discovery)					
9	- Address: linux-3.local (4c:52:62:0e:54:2b)					
10	-0 = LS bit: Globally unique address (factory default)					
11	-0 = LS bit: Individual address (unicast)					
12	- Source: linux-2.local (4c:52:62:0e:54:2b)					
13	- Address: linux-2.local (4c:52:62:0e:54:2b)					
14	-0 = LS bit: Globally unique address (factory default)					
15	-0 = LS bit: Individual address (unicast)					
16	Type: IPv4 (0x0800)					
17	- Internet Protocol Version 4, Src: linux-2.local (141.62.66.5), Dst: linux-3.local (141.62.66.6)					
18	0100 = Version: 4					
19	- 0101 = Header Length: 20 bytes (5)					
20	- Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)					
21	Total Length: 84					
22	Identification: 0xe063 (57443)					
23	Flags: 0x40 (Don't fragment)					
24	Fragment Offset: 0					
25	Time to Live: 64					
26	Protocol: ICMP (1)					
27	Header Checksum: 0xbbbb [validation disabled]					
28	[Header checksum status: Unverified]					
29	Source Address: linux-2.local (141.62.66.5)					
30	Destination Address: linux-3.local (141.62.66.6)					
31	- Internet Control Message Protocol					

Abbildung 24: MAC-Adresse des Nachbarrechners

Welche MAC-Adresse hat der Labor-Router?

Durch einen `ping` konnten wir die MAC-Adresse des Routers herausfinden.

MAC-Adresse: `00:0d:b9:4f:b8:14`

No.	Time	Source	Destination	Protocol	Length	Info
1	2.0.327898447	rn05.rnlabor.hdm-stuttgart.de	opnsense-router.rnlabor.hdm-stuttgart.de	ICMP	98	Echo (ping) request id=0xe692, seq=4/1024, ttl=64 (reply in 3)
2	3.0.327512124	opnsense-router.rnlabor.hdm-stuttgart.de	rn05.rnlabor.hdm-stuttgart.de	ICMP	98	Echo (ping) reply id=0xe692, seq=4/1024, ttl=64 (request in 2)
3	4.0.935109731	rn05.rnlabor.hdm-stuttgart.de	opnsense-router.rnlabor.hdm-stuttgart.de	DNS	84	Standard query 0xfdffa PTR 5.66.62.141.in-addr.arpa
4	5.0.935109731	opnsense-router.rnlabor.hdm-stuttgart.de	rn05.rnlabor.hdm-stuttgart.de	DNS	84	Standard query response 0x1bb0 PTR 5.66.62.141.in-addr.arpa PTR rn05.rnlabor.hdm-stuttgart.de
5	6.0.936893635	opnsense-router.rnlabor.hdm-stuttgart.de	rn05.rnlabor.hdm-stuttgart.de	DNS	127	Standard query response 0xfdffa PTR 5.66.62.141.in-addr.arpa PTR rn05.rnlabor.hdm-stuttgart.de
6	7.0.936894188	opnsense-router.rnlabor.hdm-stuttgart.de	rn05.rnlabor.hdm-stuttgart.de	DNS	163	Standard query response 0x1bb0 PTR 250.66.62.141.in-addr.arpa PTR opnsense-router.rnlabor.hdm-stuttgart.de
7	8.1.350939100	rn05.rnlabor.hdm-stuttgart.de	opnsense-router.rnlabor.hdm-stuttgart.de	ICMP	98	Echo (ping) request id=0xe692, seq=5/1280, ttl=64 (reply in 9)
8	9.1.351378490	opnsense-router.rnlabor.hdm-stuttgart.de	rn05.rnlabor.hdm-stuttgart.de	ICMP	98	Echo (ping) reply id=0xe692, seq=5/1280, ttl=64 (request in 8)
9	11.2.375018075	rn05.rnlabor.hdm-stuttgart.de	opnsense-router.rnlabor.hdm-stuttgart.de	ICMP	98	Echo (ping) request id=0xe692, seq=6/1936, ttl=64 (reply in 12)
10	12.2.375450812	opnsense-router.rnlabor.hdm-stuttgart.de	rn05.rnlabor.hdm-stuttgart.de	ICMP	98	Echo (ping) reply id=0xe692, seq=6/1936, ttl=64 (request in 11)
11	Frame 2: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp0s31f6, id 8					
12	- Ethernet II Src: rn05.rnlabor.hdm-stuttgart.de (00:0d:b9:4f:b8:14), Dst: opnsense-router.rnlabor.hdm-stuttgart.de (00:0d:b9:4f:b8:14)					
13	- Destination port: 5249 (Local Multicast Listener Discovery)					
14	- Address: opnsense-router.rnlabor.hdm-stuttgart.de (00:0d:b9:4f:b8:14)					
15	-0 = LS bit: Globally unique address (factory default)					
16	-0 = LS bit: Individual address (unicast)					
17	- Source: rn05.rnlabor.hdm-stuttgart.de (4c:52:62:0e:54:2b)					
18	- Address: rn05.rnlabor.hdm-stuttgart.de (4c:52:62:0e:54:2b)					
19	-0 = LS bit: Globally unique address (factory default)					
20	-0 = LS bit: Individual address (unicast)					
21	Type: IPv4 (0x0800)					
22	- Internet Protocol Version 4, Src: rn05.rnlabor.hdm-stuttgart.de (141.62.66.5), Dst: opnsense-router.rnlabor.hdm-stuttgart.de (141.62.66.250)					
23	0100 = Version: 4					
24	- 0101 = Header Length: 20 bytes (5)					
25	- Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)					
26	Total Length: 84					
27	Identification: 0x6807 (26839)					
28	Flags: 0x40 (Don't fragment)					
29	Fragment Offset: 0					
30	Time to Live: 64					
31	Protocol: ICMP (1)					
32	Header Checksum: 0x3256 [validation disabled]					
33	[Header checksum status: Unverified]					
34	Source Address: rn05.rnlabor.hdm-stuttgart.de (141.62.66.5)					
35	Destination Address: opnsense-router.rnlabor.hdm-stuttgart.de (141.62.66.250)					
36	- Internet Control Message Protocol					

Abbildung 25: MAC-Adresse des Labor-Routers

Welche MAC-Adresse hat der Server 141.62.1.5 (außerhalb des Labor-Netzes)?

Da der Rechner außerhalb des Labor-Netzes ist, kann dessen Mac nicht bestimmt werden.

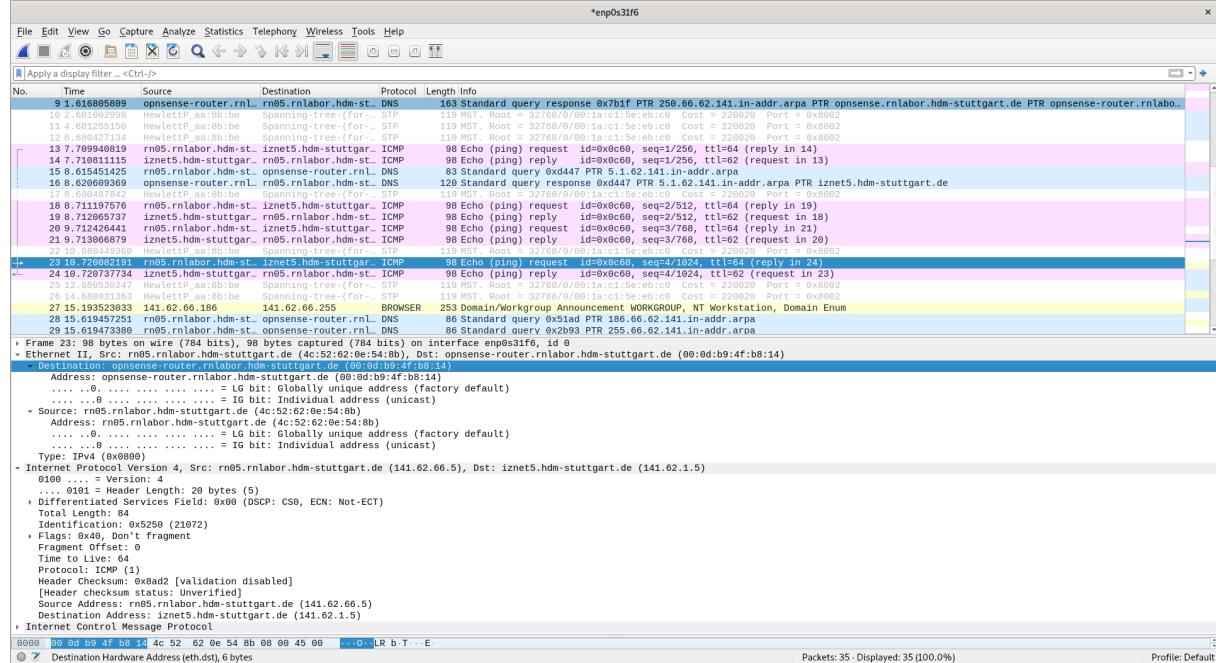


Abbildung 26: MAC-Adresse des externen Rechners

2.9 STP

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.

2.10 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.

Abbildung 27: Capture mit Filter für STP

2.11 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 ein Video-Streaming Ihrer Wahl. Welche TCP-Ports werden wozu benutzt? Filtern Sie alle Rahmen, in denen sich das TCP-Window geändert hat

Hier wurde ein Stream von Twitch konsumiert; wie zu erkennen ist, wird die Window-Size stetig erhöht.

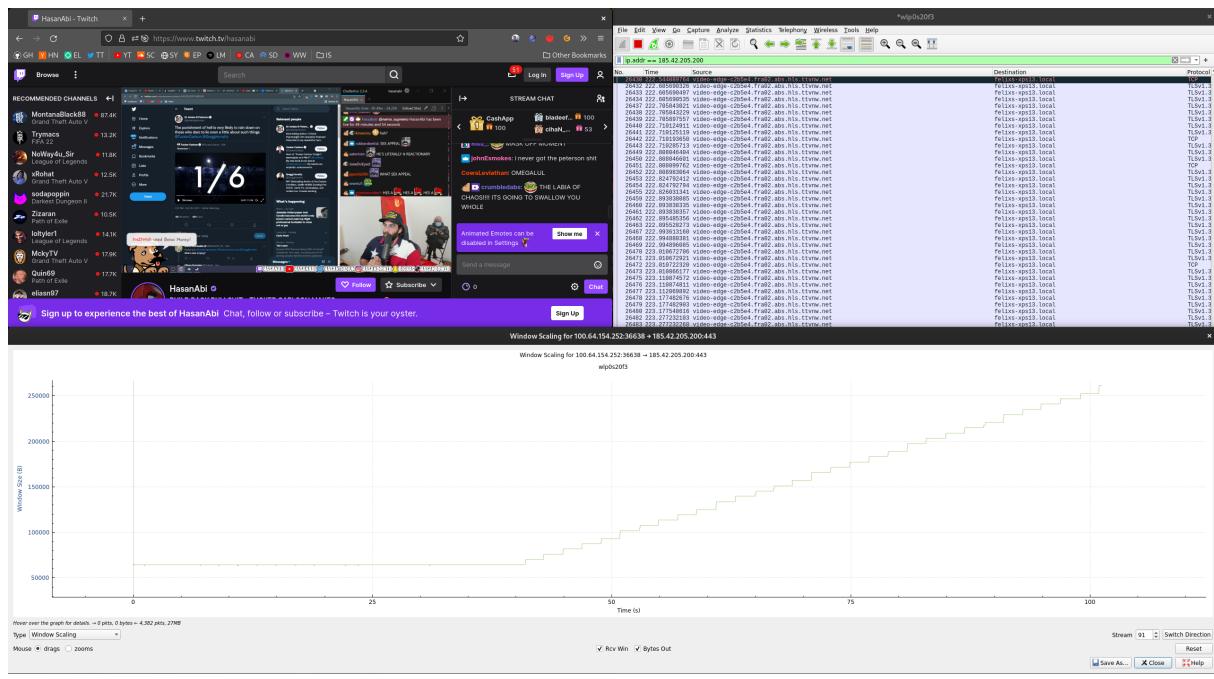


Abbildung 29: Verlauf der TCP-Window-Size beim Streaming von Twitch

Es wird Port 443, der Standard-Port für HTTPS, verwendet. Seitens des Clients wird vom TCP-Stack des Kernels ein temporärer Port zugewiesen.

2.12 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.

Können Sie Passwörter im Wireshark-Trace identifizieren?

Da Telnet unverschlüsselt ist, können Passwörter identifiziert und ausgelesen werden.

Wie verhält sich im Vergleich dazu eine SSH-Verbindung zum gleichen Server?

Die SSH-Verbindung ist verschlüsselt; Passwörter, Logins etc. können hier nicht mitgelesen werden.

No.	Time	Source	Destination	Protocol	Length Info
53	13.371889779	141.62.66.5	141.62.66.207	TELNET	69 Telnet Data ...
55	13.371964177	141.62.66.207	141.62.66.5	TELNET	69 Telnet Data ...
57	13.372108943	141.62.66.5	141.62.66.207	TELNET	69 Telnet Data ...
58	13.372142487	141.62.66.207	141.62.66.5	TELNET	86 Telnet Data ...
60	13.372142487	141.62.66.207	141.62.66.5	TELNET	66 Telnet Data ...
65	15.536484921	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
66	15.537258875	141.62.66.207	141.62.66.5	TELNET	67 Telnet Data ...
67	15.712433765	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
71	15.713143066	141.62.66.207	141.62.66.5	TELNET	67 Telnet Data ...
73	15.784452500	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
74	15.805364250	141.62.66.207	141.62.66.5	TELNET	67 Telnet Data ...
76	15.864385554	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
77	15.865998282	141.62.66.207	141.62.66.5	TELNET	67 Telnet Data ...
79	15.991754757	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
80	15.992584487	141.62.66.207	141.62.66.5	TELNET	67 Telnet Data ...
82	16.056360880	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
84	16.176491695	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
86	16.176491695	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
87	16.177396417	141.62.66.207	141.62.66.5	TELNET	67 Telnet Data ...
89	16.344425688	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
90	16.344425688	141.62.66.207	141.62.66.5	TELNET	67 Telnet Data ...
91	16.528454531	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
93	16.529374164	141.62.66.207	141.62.66.5	TELNET	67 Telnet Data ...
94	16.529374164	141.62.66.5	141.62.66.207	TELNET	68 Telnet Data ...
96	17.193204469	141.62.66.207	141.62.66.5	TELNET	69 Telnet Data ...
98	17.193599561	141.62.66.207	141.62.66.5	TELNET	70 Telnet Data ...
101	19.152498976	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
103	19.344388219	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
105	19.344388219	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
107	19.616478844	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
109	19.688482452	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
111	19.816961616	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
113	19.912439965	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...

Abbildung 30: Capture des Telnet-Logins

No.	Time	Source	Destination	Protocol	Length Info
77	15.865998282	141.62.66.207	141.62.66.5	TELNET	67 Telnet Data ...
79	15.991754757	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
80	15.992584487	141.62.66.207	141.62.66.5	TELNET	67 Telnet Data ...
82	16.056360880	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
83	16.057270317	141.62.66.207	141.62.66.5	TELNET	67 Telnet Data ...
85	16.176491695	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
86	16.177396417	141.62.66.207	141.62.66.5	TELNET	67 Telnet Data ...
88	16.344425688	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
89	16.344425688	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
90	16.528454531	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
92	16.528454531	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
93	16.529374164	141.62.66.207	141.62.66.5	TELNET	67 Telnet Data ...
94	16.529374164	141.62.66.5	141.62.66.207	TELNET	68 Telnet Data ...
96	17.193204469	141.62.66.207	141.62.66.5	TELNET	69 Telnet Data ...
98	17.193599561	141.62.66.207	141.62.66.5	TELNET	70 Telnet Data ...
101	19.152498976	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
103	19.344388219	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
105	19.344388219	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
107	19.616478844	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
109	19.688482452	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
111	19.816961616	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
113	19.912439965	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...

Abbildung 31: Capture des Telnet-Passwords

No.	Time	Source	Destination	Protocol	Length Info
77	15.865998282	141.62.66.207	141.62.66.5	TELNET	67 Telnet Data ...
79	15.991754757	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
80	15.992584487	141.62.66.207	141.62.66.5	TELNET	67 Telnet Data ...
82	16.056360880	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
83	16.057270317	141.62.66.207	141.62.66.5	TELNET	67 Telnet Data ...
86	16.057270317	141.62.66.207	141.62.66.5	TELNET	67 Telnet Data ...
87	16.177396417	141.62.66.207	141.62.66.5	TELNET	67 Telnet Data ...
89	16.344425688	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
90	16.344425688	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
92	16.528454531	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
93	16.529374164	141.62.66.207	141.62.66.5	TELNET	67 Telnet Data ...
95	17.192471399	141.62.66.5	141.62.66.207	TELNET	68 Telnet Data ...
96	17.192471399	141.62.66.207	141.62.66.5	TELNET	68 Telnet Data ...
98	17.193599561	141.62.66.207	141.62.66.5	TELNET	76 Telnet Data ...
101	19.152498976	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
103	19.344388219	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
105	19.344388219	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
107	19.616478844	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
109	19.688482452	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
111	19.816961616	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...
113	19.912439965	141.62.66.5	141.62.66.207	TELNET	67 Telnet Data ...

Abbildung 32: Capture eines Charakters des Telnet-Passwords

No.	Time	Source	Destination	Protocol	Length	Info
282	65.784067321	138.68.70.72	141.62.66.5	SSH	126	Server: Encrypted packet (len=60)
278	65.832318634	138.68.70.72	141.62.66.5	SSH	126	Server: Encrypted packet (len=60)
280	119.032477934	141.62.66.5	138.68.70.72	SSH	126	Client: Encrypted packet (len=36)
438	174.475555778	141.62.66.5	138.68.70.72	SSH	142	Client: Encrypted packet (len=76)
448	174.482595737	138.68.70.72	141.62.66.5	SSH	198	Server: Encrypted packet (len=132)
450	177.2230561045	141.62.66.5	138.68.70.72	SSH	158	Client: Encrypted packet (len=92)
452	177.237049824	141.62.66.5	138.68.70.72	SSH	126	Server: Encrypted packet (len=116)
453	177.237128451	141.62.66.5	138.68.70.72	SSH	126	Client: Encrypted packet (len=68)
454	177.237283147	141.62.66.5	138.68.70.72	SSH	126	Client: Encrypted packet (len=68)
458	177.243289860	138.68.70.72	141.62.66.5	SSH	206	Server: Encrypted packet (len=140)
461	177.250592045	138.68.70.72	141.62.66.5	SSH	111	Client: Encrypted packet (len=52)
463	177.250574112	138.68.70.72	141.62.66.5	SSH	862	Server: Encrypted packet (len=796)
465	177.252448894	141.62.66.5	138.68.70.72	SSH	118	Client: Encrypted packet (len=52)
466	177.25252894	138.68.70.72	141.62.66.5	SSH	134	Server: Encrypted packet (len=68)
468	177.252576374	141.62.66.5	138.68.70.72	SSH	118	Client: Encrypted packet (len=52)
469	177.285539989	141.62.66.5	138.68.70.72	SSH	126	Server: Encrypted packet (len=68)
470	177.285539988	141.62.66.5	138.68.70.72	SSH	118	Client: Encrypted packet (len=52)
Frame 202: 120 bytes captured (1088 bits) on interface enp0s3t6, id 0						
Ethernet II, Src: opnsense.rnrlabor.hdm-stuttgart.de (00:0d:b9:4f:bb:14), Dst: rn05.rnrlabor.hdm-stuttgart.de (4c:52:62:0e:54:b8)						
Internet Control Protocol Version 4, Src: 138.68.70.72, Dst Port: 22, Dst Port: 22, Seq: 1, Ack: 1, Len: 68						
SSH Protocol						
Packet Length (encrypted): 0xbcc15349d502f55030da2cacb0c73e84abebe992378514580fe2c6b2d9dab4f02ad3e..						
[Direction: server-to-client]						

Abbildung 33: Capture eines verschlüsselten SSH-Pakets

2.13 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
28	255	1.1.441995699	100.64.154.254	felix-xpos13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
29	255	1.1.441995759	100.64.154.254	felix-xpos13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
31	255	1.1.519793372	100.64.154.254	felix-xpos13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
89	255	1.3.498431116	100.64.154.254	felix-xpos13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
90	255	1.3.500593900	100.64.154.254	felix-xpos13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
112	255	1.3.583493000	100.64.154.254	felix-xpos13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
113	255	1.4.50033578	100.64.154.254	felix-xpos13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
1527	255	1.21.511980153	100.64.154.254	felix-xpos13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
1567	255	1.21.61419604	100.64.154.254	felix-xpos13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2031	255	1.25.44139947	100.64.154.254	felix-xpos13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2044	255	1.25.4561974	100.64.154.254	felix-xpos13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2045	255	1.29.44139788	100.64.154.254	felix-xpos13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2049	255	1.29.50002269	100.64.154.254	felix-xpos13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2050	255	1.29.50002269	100.64.154.254	felix-xpos13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2051	255	1.29.50002263	100.64.154.254	felix-xpos13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2052	255	1.29.50002266	100.64.154.254	felix-xpos13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
11826	255	74.57378592	100.64.154.254	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local. "QU" question PTR _companion-link._tcp.local, "QM" quest..
2030	255	74.57378592	100.64.154.254	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local. "QU" question PTR _companion-link._tcp.local, "QM" quest..
12561	255	78.567487619	100.64.154.254	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	87.681387937	100.64.154.254	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local. "QM" question PTR _companion-link._tcp.local, "QM" quest..
18051	255	1.134.490477999	100.64.154.254	felix-xpos13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
18666	255	1.134.622113475	100.64.154.254	felix-xpos13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
18646	255	149.44191144	100.64.154.254	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local. "QU" question PTR _companion-link._tcp.local, "QM" quest..
20392	255	149.44191144	100.64.154.254	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local. "QU" question PTR _companion-link._tcp.local, "QM" quest..
28394	255	144.924217309	100.64.154.254	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	154.345592688	100.64.154.254	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local. "QU" question PTR _companion-link._tcp.local, "QM" quest..
21935	255	155.472517304	100.64.154.254	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	158.44131816	100.64.154.254	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	167.65746649	100.64.154.254	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local. "QM" question PTR _companion-link._tcp.local, "QM" quest..
22852	255	168.579565631	100.64.154.254	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. Type: 11 (Time-to-live exceeded)-ICMP-Pakete angezeigt werden.

Nur IP-Pakete, die fragmentiert sind

Mittels eines Filters auf “Must Fragment” konnten in dieser Aufgabe nur fragmentierte Pakete angezeigt werden.

ip.flags.mf ==1 or ip.frag_offset gt 0							
No.	TTL	Time	Source	Destination	Protocol	Length	Info
16357	64	121.27.123.2460	109.64.154.247	255.255.255.255	Tcp	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=9ab0) [Reassembled in #16358]
16358	64	121.27.123.2593	109.64.154.247	255.255.255.255	UDP	392	47099 -> xmsg(1716) Len=1830

Abbildung 35: Capture von fragmentierten IP-Paketen

Beim Login-Versuch auf <ftp.bellevue.de> mit von Ihnen wählbaren Account-Daten nur Rahmen herausfiltern, die das gewählte Passwort im Ethernet-Datenfeld enthalten

Mittels des Filters `ftp.request.command == "PASS"` werden nur Pakete angezeigt, welche das Passwort enthalten.

ftp							
No.	Time	Source	Destination	Protocol	Length	Info	
3657	651.572178872	212.77.241.212	141.62.66.5	FTP	89	Response: 220 0MCnet FTP Daemon	
3704	706.805421263	141.62.66.5	212.77.241.212	FTP	78	Request: USER jakob	
3706	706.843474062	212.77.241.212	141.62.66.5	FTP	99	Response: 331 Password required for jakob	
3713	715.293442858	141.62.66.5	212.77.241.212	FTP	89	Request: PASS passwortfolligio	
3714	715.313954381	212.77.241.212	141.62.66.5	FTP	88	Response: 538 Login incorrect.	
3716	715.313246123	141.62.66.5	212.77.241.212	FTP	72	Request: SYST	
3717	715.331546015	212.77.241.212	141.62.66.5	FTP	85	Response: 215 UNIX Type: L8	

Raw 0x15: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface emp0s3:if0, id 0							
Ethernet II, Src: rns05:rnlabor.hdm-stuttgart.de (4c:52:62:9e:54:b8), Dst: opnsense.rnlabor.hdm-stuttgart.de (00:0d:b9:4f:b8:14)							
Internet Protocol Version 4, Src: 141.62.66.5, Dst: 212.77.241.212							
Transmission Control Protocol, Src Port: 51798, Dst Port: 21, Seq: 13, Ack: 57, Len: 23 [File Transfer Protocol (FTP)] [Current working directory:]							

Abbildung 36: Capture eines FTP-Pakets, welches ein Password enthält

Nur den Port 80-Verkehr zu Ihrer IP-Adresse (ankommend und abgehend)

Mittels eines Filters wurde ausschließlich TCP-Traffic auf Port 80 dargestellt. Mittels `|| udp.port == 80` hätte auch noch UDP-Traffic auf diesem Port dargestellt werden können.

tcp.port == 80							
No.	TTL	Time	Source	Destination	Protocol	Length	Info
6508	64	11.367453746	Felixs-xps13.local	news.ycombinator.com	TCP	74	41206 -> http(80) [SYN] Seq=0 Win=64240 Len=9 MSS=1460 SACK_PERM=1 TSval=366326180 TSecr=3732855280
6644	64	11.609341374	Felixs-xps13.local	news.ycombinator.com	TCP	66	41206 -> http(80) [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=366326422 TSecr=3732855280
6645	64	11.609418667	Felixs-xps13.local	news.ycombinator.com	HTTP	150	GET / HTTP/1.1
6774	64	11.814666182	Felixs-xps13.local	news.ycombinator.com	TCP	66	41206 -> http(80) [ACK] Seq=85 Ack=376 Win=64000 Len=0 TSval=366326627 TSecr=3732855522
6775	64	11.814783601	Felixs-xps13.local	news.ycombinator.com	TCP	66	41206 -> http(80) [FIN, ACK] Seq=85 Ack=376 Win=64128 Len=0 TSval=366326628 TSecr=3732855522
6888	64	12.019239834	Felixs-xps13.local	news.ycombinator.com	TCP	66	41206 -> http(80) [ACK] Seq=86 Ack=377 Win=64128 Len=0 TSval=366326832 TSecr=3732855728

Abbildung 37: Capture aller TCP-Segmente auf Port 80

Nur Pakete mit einer IP-Multicast-Adresse

Mittels eines Filters werden nur IPs > 224.0.0.0 dargestellt, was IP-Multicast-Adressen sind.

No.	TTL	Time	Source	Destination	Protocol	Length	Info
2931	1	3.591566536	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a0d PTR 100.64.154.246.in-addr.arpa, "QM" question
2938	1	3.602016273	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a0e PTR 100.64.154.246.in-addr.arpa, "QM" question
2956	1	3.624785943	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a0f PTR 100.64.154.248.in-addr.arpa, "QM" question
2964	1	3.638125087	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a10 PTR 100.64.154.248.in-addr.arpa, "QM" question
2974	1	3.657613109	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a11 PTR 100.64.154.244.in-addr.arpa, "QM" question
2986	1	3.6676971318	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a12 PTR 100.64.154.244.in-addr.arpa, "QM" question
2998	1	3.699437532	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a13 PTR 100.64.154.251.in-addr.arpa, "QM" question
2195	1	3.709798914	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a14 PTR 100.64.154.251.in-addr.arpa, "QM" question
2128	1	3.723295332	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a15 PTR 100.64.154.254.in-addr.arpa, "QM" question
2128	1	3.733692995	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a16 PTR 100.64.154.254.in-addr.arpa, "QM" question
2142	1	3.756352366	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a17 PTR 100.64.154.252.in-addr.arpa, "QM" question
2147	1	3.766727292	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a18 PTR 100.64.154.252.in-addr.arpa, "QM" question
3641	1	6.182275932	felixs-xps13.local	239.255.255.250	SSDP	213	M-SEARCH * HTTP/1.1
343	1	6.1822759321	felixs-xps13.local	239.255.255.250	SSDP	213	M-SEARCH * HTTP/1.1
4129	1	7.1833032098	felixs-xps13.local	239.255.255.250	SSDP	213	M-SEARCH * HTTP/1.1
4122	1	7.185191345	felixs-xps13.local	239.255.255.250	SSDP	213	M-SEARCH * HTTP/1.1
4783	1	8.184255659	felixs-xps13.local	239.255.255.250	SSDP	213	M-SEARCH * HTTP/1.1
4785	1	8.186126105	felixs-xps13.local	239.255.255.250	SSDP	213	M-SEARCH * HTTP/1.1

Abbildung 38: Capture aller IP-Pakete mit Multicast-Adressen