Lösung 12: DNS, HTTP

Aufgabe 1: Network Address Translation (NAT)

a) Beispiel:
- Home Router, externe bzw. öffentliche IP Adresse: 24.34.112.235
- Home Router, interne Adresse: 192.168.0.1
- Laptop 1: 192.168.0.101
- Laptop 2: 192.168.0.102
- Laptop 3: 192.168.0.103

b) Hierfür gibt es mehrere Möglichkeiten. Wichtig ist, dass jede HTTP Verbindung auf der WAN Seite eine eindeutige Portnummer hat. Für einen bestimmten Laptop darf nicht der gleiche Quell-Port für 2 verschiedene HTTP-Verbindungen gewählt werden. Eine mögliche gültige Belegung der Tabelle (auch wenn sie etwas unwahrscheinlich ist) könnte wie folgt aussehen:

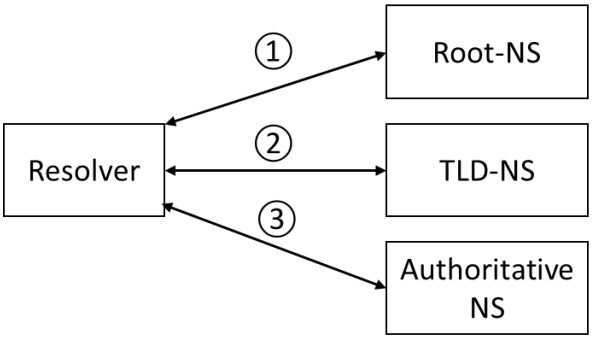
LAN Seite/Heim-Netzwerk		*WAN Seite / Internet*	
IP Adresse	Port	IP Adresse	Port
192.168.0.101	18001	24.34.112.235	12000
192.168.0.101	18002	24.34.112.235	12001
192.168.0.102	53221	24.34.112.235	12002
192.168.0.102	53222	24.34.112.235	12003
192.168.0.103	49111	24.34.112.235	12004
192.168.0.103	49112	24.34.112.235	12005

Aufgabe 2 : Domain Name System (DNS)

a) Neuere Ubuntu-Versionen verwenden einen internen „Stub Resolver“ (weitere Indirektion) unter der IP 127.0.0.53. Dieser reicht DNS Anfragen an den eigentlichen Resolver weiter. Letzteren ermittelt man z.B. per `systemd-resolve -q status`. Konkret werden hier als lokale DNS Server / Resolver 141.60.110.103 und 141.60.120.105 verwendet.

```
Link 2 (enp0s3)
      Current Scopes: DNS
       LLMNR setting: yes
MulticastDNS setting: no
      DNSSEC setting: no
    DNSSEC supported: no
         DNS Servers: 141.60.110.103
                      141.60.120.105
          DNS Domain: dhcp.fh-rosenheim.de
```

b) Der Reihe nach ergeben sich z.B. die folgenden Ergebnisse:



Name: i.root-servers.net
IP: 192.36.148.17

Name: a.nic.de
IP: 194.0.0.53

Name: ns.th-rosenheim.de
IP: 141.60.160.2

Hinweis: Es gibt im Verlauf oft mehrere Nameserver, die weiterhelfen können. Beim letzten DNS Server (`ns.fh-rosenheim.de`) handelt es sich um den *Authoritative DNS Server*, der für den Namen www.th-rosenheim.de maßgeblich ist. Das Ergebnis, dass dieser Server zurückliefert entspricht der IP Adresse von www.th-rosenheim.de.
Man sieht auch, dass es zwei DNS Server gibt, die Hostnamen für IP Adressen der TH Rosenheim auflösen können, nämlich: `ns.fh-rosenheim.de` und `deneb.dfn.de`.

```
dev@OS-dev /etc $ dig @141.60.160.2 www.th-rosenheim.de

; <<>> DiG 9.11.3-1ubuntu1.9-Ubuntu <<>> @141.60.160.2 www.th-rosenheim.de
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43803
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 4

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 1bdd36f2e139bcd54d91ed635dfb98c23c24b0d6b85e2bc6 (good)
;; QUESTION SECTION:
;www.th-rosenheim.de.           IN      A

;; ANSWER SECTION:
www.th-rosenheim.de.    86400   IN      A       141.60.160.196

;; AUTHORITY SECTION:
th-rosenheim.de.        86400   IN      NS      ns.fh-rosenheim.de.
th-rosenheim.de.        86400   IN      NS      dns-3.dfn.de.

;; ADDITIONAL SECTION:
ns.fh-rosenheim.de.     600     IN      A       141.60.160.2
dns-3.dfn.de.           72865   IN      A       193.174.75.58
dns-3.dfn.de.           72865   IN      AAAA    2001:638:d:b103::1

;; Query time: 3 msec
;; SERVER: 141.60.160.2#53(141.60.160.2)
;; WHEN: Thu Dec 19 16:35:30 CET 2019
;; MSG SIZE  rcvd: 206
```

c) `dig MX th-rosenheim.de` liefert zunächst:

```
dev@OS-Dev ~ $ dig MX th-rosenheim.de

; <<>> DiG 9.16.1-Ubuntu <<>> MX th-rosenheim.de
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13809
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;th-rosenheim.de.               IN      MX

;; ANSWER SECTION:
th-rosenheim.de.        7087    IN      MX      90 sophos-app-prim.th-rosenheim.de.
th-rosenheim.de.        7087    IN      MX      90 sophos-app-sec.th-rosenheim.de.

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Thu Jan 13 12:54:50 CET 2022
;; MSG SIZE  rcvd: 107
```

Die MX-Einträge stehen für einen Mailserver. Löst man den Namen sophos-app*** per dig auf, so erhält man: 141.60.160.114 oder 141.60.160.115.

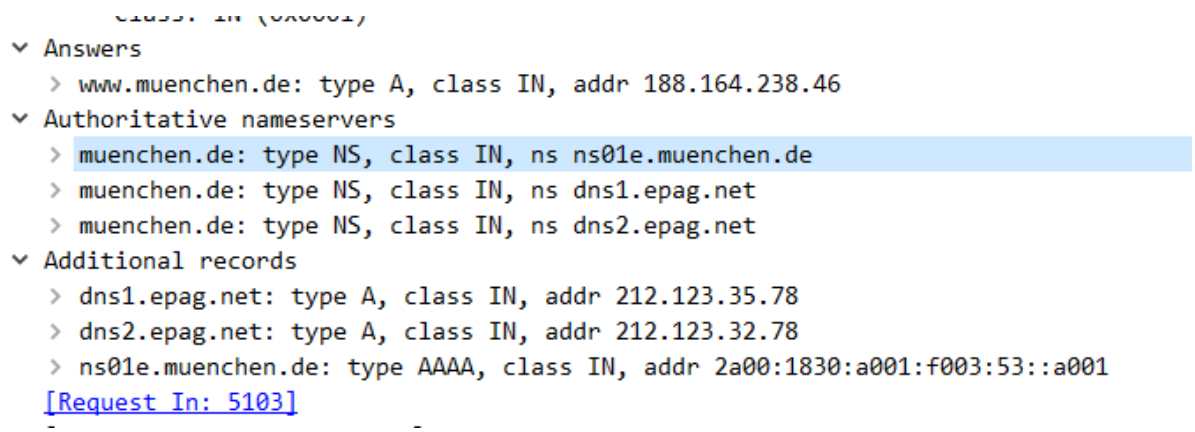
d) Man erkennt, dass für www.berlin.de bereits eine IPv6 Adresse zurückgeliefert wird, aber für www.muenchen.de nicht. Das legt den Schluss nahe, dass die Webseite von www.muenchen.de noch nicht IPv6-fähig ist.

```
;; ANSWER SECTION:
www.berlin.de.          86352   IN      A       212.45.111.17
www.berlin.de.          2774    IN      AAAA    2a00:cd0:1002:1::17

;; ANSWER SECTION:
www.muenchen.de.        3542    IN      A       188.164.238.46
```

e) Es werden mehrere IP Adresse zurückgegeben, z.B. 4 verschieden IP Adressen. Die Reihenfolge der IP Adressen kann sich dabei ändern. In der Regel wird eine Anwendung (z.B. Webrowser) immer die 1. Adresse verwenden. Auf diese Weise erreicht man Load Balancing.
Im konkreten Fall ändert sich die Reihenfolge vermutlich nicht, der Grund ist dass die DNS Daten gecacht sind.
Man erkennt auch schön, dass erhaltene DNS Records nur eine begrenzte Gültigkeit haben.

f) Siehe z.B. Beispiel Trace `dns.pcapng`: Die DNS-Anfrage ist in Paket 5101, die Antwort in 5102. Der Standardport ist 53. Ungefragt erhält man also Antwort z.B. die Nameserver der Domain muenchen.de und deren IP Adresse zurück, siehe Screenshot.



Aufgabe 3: HTTP

a) TCP Client: 192.168.178.25, TCP Server: 195.200.71.187. Es werden die Ports 443 und 19675 verwendet. Das legt nahe, dass es sich um HTTPS Verkehr hält. Unverschlüsseltes HTTP ist heute nur noch selten anzutreffen.

b) Der Grund: Sämtliche Nutzlast der TCP Verbindung ist TLS verschlüsselt. Mit TLS lässt sich alles verschlüsseln, was über TCP läuft. Handelt es sich um HTTP, so spricht man von HTTPS. Auch wenn es verschlüsselt ist: Das HTTP Format bleibt unverändert.

c) Ja, nun sind HTTP-Inhalte sichtbar. Der Grund ist, dass Wireshark durch Bekanntgabe des TLS Schlüssels die Verbindung nun entschlüsseln kann.

d)
- Paket #11
- Ethernet, IPv4, TCP, TLS, http
- Mozilla/5.0, Firefox/70.0
- Die bevorzugte Sprache ist Deutsch, Zweitwahl ist amerikanisches Englisch:
  Mit „de, en-US; q=0.7,en;q=0.3“ wird der Server gebeten, die deutsche Version der Webseite auszuliefern, aber nur, wenn dessen Qualität >=70% des relativen Qualitätsfaktors ist.

e)
- Paket # 31
- Status Code: 200 OK
- Am 21. Dezember 2019 um 11:56:49 Uhr (Date Feld)
- Der HTTP Header benötigt 505 Bytes. Am einfachsten sieht man das, wenn man auf „Hypertext Transfer Protocol“ im mittleren Fenster von Wireshark drückt. Ganz unten in der Statusleiste wird dann 5050 Bytes angezeigt.
- 7 Reassembled TCP Segments.

f)
- Scheinbar wird jede TCP Verbindung nur für eine HTTP Anfrage verwendet. Also kein „Persistent http“! Man sieht das daran, dass jede Anfrage einen neuen TCP Source Port verwendet: 19675, 19679, 19676, usw. Das ist erstaunlich angesichts des Overheads. Schließlich muss man auch jedes Mal einen sogenannten TLS Handshakes durchlaufen.
- network.http.max-persistent-connections-per-server = 6 (im Firefox des Dozenten)

g)
- Ja, in Paket #31 (200 OK) wird ein Cookie gesetzt:
  `Set-Cookie: LB_STK_BAYERN_DE=912108298.20480.0000; path=/; Httponly\r\n`
- Ja, in Paket #4920 wird das Cookie beim erneuten Aufruf vom Webbrowser an den Webserver gesendet:
  `Set-Cookie: LB_STK_BAYERN_DE=912108298.20480.0000; path=/; Httponly\r\n`
- Firefox: *Extras, Einstellungen, Datenschutz & Sicherheit*: Unter „*Cookies und Website-Date*n“ kann man sich die Cookies ansehen.

h) Im GET von Paket #5307 signalisiert der Webbrowser durch ein „If-Modified-Since“, dass der Webserver den Inhalt nur erneut senden muss, wenn es einen aktuelleren Inhalt gibt. Dies ist hier nicht der Fall, weswegen ein Paket #5339 eine leere Response zurückgesendet wird (kein Inhalt, nur HTTP Header)