Aufgabe 1 – HTTP/1.1 plain – Browser (2.5 Punkte)

Rufe die URI http://nwa1-h11.sin-lab.at:8080/ mit dem Browser Mozilla Firefox auf!

1a. (0.5 Punkte) Dokumentiere und erkläre HTTP Request (Request Line) und http Response (Status Line) dieses Aufrufs

Request-line:

```
GET / HTTP/1.1
Host: nwa1-h11.sin-lab.at:8080
```

- **GET**
  - Methode, zum Abrufen einer Ressource über eine URI.
  - Parameter (z. B. Formulardaten) werden im Query-Part der URI an den Webserver übertragen
- Host: http://nwa1-h11.sin-lab.at:8080/
  - Host wird verwendet, um eine Ressource auf einem Ursprungsserver oder Gateway zu identifizieren. Standardmäßig wird Port 80 verwendet, wenn man keinen angibt. Wir geben 8080 an, da auf diesem Port http 1.1 plain bereitgestellt wird und wir dies für die Übung verwenden möchten
    [https://www.tutorialspoint.com/http/http_requests.htm]
- Protokoll Version: **HTTP/1.1** – ist auch die meistverwendete Version

[https://sematext.com/glossary/http-requests]
[https://datatracker.ietf.org/doc/html/rfc2616#section-5.1]

Status-line:

```
HTTP/1.1 200 OK
```

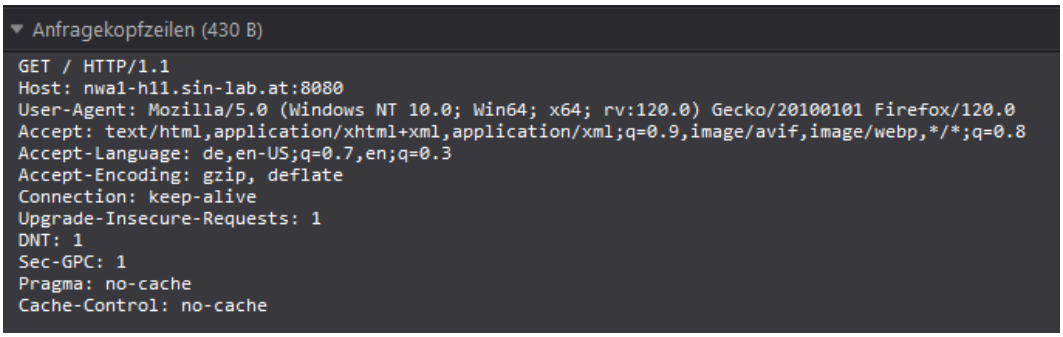
Status-lines sind normalerweise folgendermaßen aufgebaut:

1. Protokoll Version: **HTTP/1.1** – ist auch die meist verwendete version
2. Status code: **200** - daraus lässt sich folgern ob der request erfolgreich war oder nicht
3. Status text: **OK** - Eine textliche Beschreibung des status codes

[https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages]

1b. (1 Punkt) Erläutere die Bedeutung der einzelnen Header-Felder (belege!)!

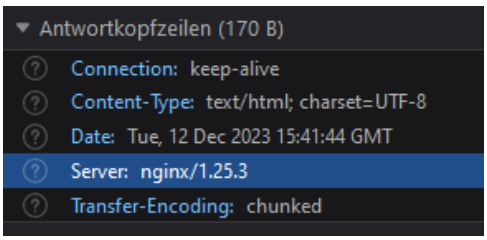
Anfrage Header:



- **User-Agent**: In ihm stehen Informationen über den Client, sodass z. B. ein serverseitiges Skript an verschiedene Browser angepasste Inhalte ausliefern kann
- **Accept**: Welche Inhaltstypen der Client verarbeiten kann z.B. text/html,..
- **Accept-Language**: Welche Sprachen der Client akzeptiert. Falls der Server passend eingerichtet ist und die Sprachversionen vorhanden sind, wird über Content Negotiation die passende Datei ausgeliefert.
- **Accept-Encoding**: Welche komprimierten Formate der Client unterstützt
- **Connection**: Welchen Typ von Verbindung der Client bevorzugt.
- **Upgrade-Insecure-Requests:**
- **DNT**: Befiehlt einer Website, den Nutzer nicht zu tracke
- **Sec-GPC:** gibt an, ob der Benutzer damit einverstanden ist, dass eine Website oder ein Dienst seine persönlichen Daten an Dritte verkauft oder weitergibt
- **Pragma**: enthält Optionen, die möglicherweise nur von einigen Implementationen verstanden werden und sich an alle Glieder in der Frage-Antwort-Kette richten
- **Cache-Control**: Wird genutzt, um Optionen festzulegen, denen durch alle Caching-Mechanismen entlang der Anfrage-/Antwort-Kette Folge geleistet werden muss.

[https://de.wikipedia.org/wiki/Liste_der_HTTP-Headerfelder]
[https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Sec-GPC]
[https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Upgrade-Insecure-Requests]

Antwort Header:



- **Connection:** steuert, ob die Netzwerkverbindung nach Abschluss der aktuellen Transaktion geöffnet bleibt
- **Content-Type:** MIME-Typ des Bodys
- **Date:** Datum und Zeit zum Sendezeitpunkt der Anfrage
- **Server:** Serverkennung (so wie User-Agent für den Client ist, ist Server für die Serversoftware)
- **Transfer-Encoding:** Die Methode, die genutzt wird, den Inhalt sicher zum Nutzer zu bringen. Zurzeit sind folgende Methoden definiert: chunked (aufgeteilt), compress (komprimiert), deflate (komprimiert), gzip (komprimiert), identity.

1c. (0.5 Punkte) Welche Header-Felder sind verpflichtend (belege!)?

Verpflichtende Request Header:

- HTTP 1.1 benötigt das Host Feld
- In http 1.0 sind keine Felder required

[https://serverfault.com/questions/163511/what-is-the-mandatory-information-a-http-request-header-must-contain]

Verpflichtende Response Header:

- Date (außer in Ausnahmefällen)

  Die erste Zeile der Antwort ist ebenfalls verpflichtend und besteht aus dem Protokoll (HTTP/1.1), dem Antwortcode (200) und der Beschreibung (OK).

[https://datatracker.ietf.org/doc/html/rfc2616]
[http://www.steves-internet-guide.com/http-headers/]

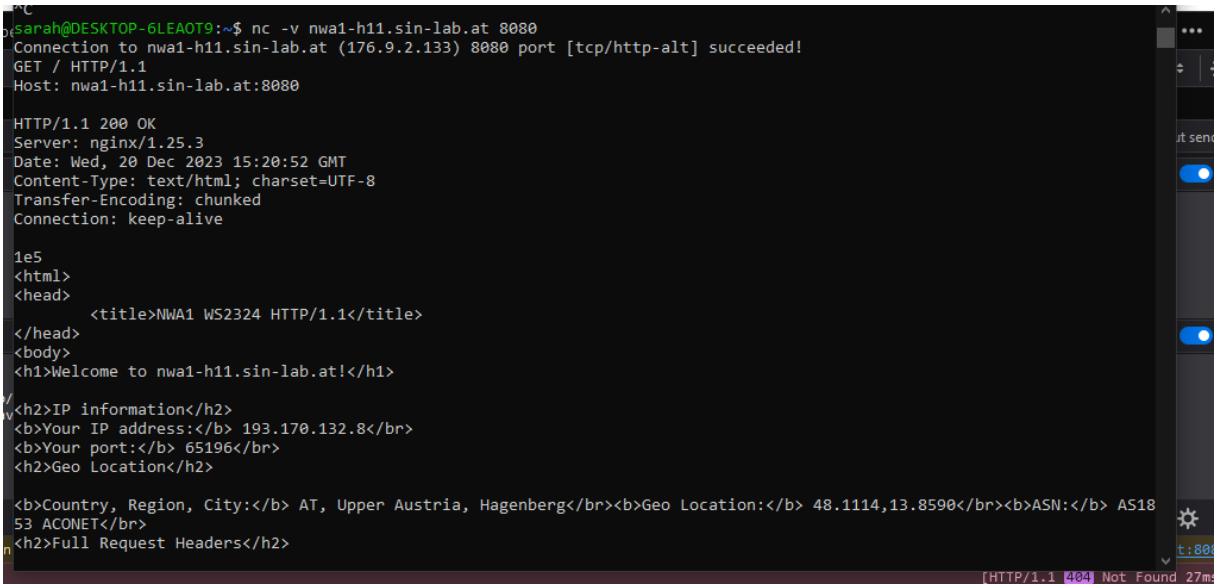
1d. (0.5 Punkte) Speichere alle Dateneinheiten (und nur diese) der für diesen Aufruf notwendigen TCP-Verbindung in einer PCAP-Datei!

Zu finden im File ue3_aufgabe1.pcap

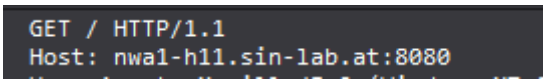
Aufgabe 2 – HTTP/1.1 plain – CLI (2 Punkte)

Rufe die URI http://nwa1-h11.sin-lab.at:8080/ nun mit ncat per HTTP/1.1 auf! Verwende dabei einen minimalen Request, der nur die zwingend notwendigen Informationen enthält!

2a. Dokumentiere und erkläre HTTP Request (Request Line) und http Response (Status Line) dieses Aufrufs!



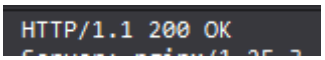
Request-line:



- **GET**
  - Methode, zum Abrufen einer Ressource über eine URI.
  - Parameter (z. B. Formulardaten) werden im Query-Part der URI an den Webserver übertragen
- Host: http://nwa1-h11.sin-lab.at:8080/
  - Host wird verwendet, um eine Ressource auf einem Ursprungsserver oder Gateway zu identifizieren. Standardmäßig wird Port 80 verwendet, wenn man keinen angibt. Wir geben 8080 an, da auf diesem Port http 1.1 plain bereitgestellt wird und wir dies für die Übung verwenden möchten
    [https://www.tutorialspoint.com/http/http_requests.htm]
- Protokoll Version: **HTTP/1.1** – ist auch die meistverwendete Version

[https://sematext.com/glossary/http-requests/]
[https://datatracker.ietf.org/doc/html/rfc2616#section-5.1]

Status-line:



Status-Lines sind normalerweise folgendermaßen aufgebaut:

4. Protokoll Version: **HTTP/1.1** – ist auch die meist verwendete Version
5. Status code: **200** - daraus lässt sich folgern, ob der request erfolgreich war oder nicht
6. Status Text: **OK** - Eine textliche Beschreibung des Status Codes

[https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages]

2b. (0.5 Punkte) Erläutere, welche(r) Header in einem minimalen Request enthalten sein müssen/ muss (belege!)!

Verpflichtende Request Header:

- HTTP 1.1 benötigt das Host Feld
- In http 1.0 sind keine Felder required

[https://serverfault.com/questions/163511/what-is-the-mandatory-information-a-http-request-header-must-contain]

2c. (0.5 Punkte) Speichere alle Dateneinheiten (und nur diese) der für diesen Aufruf notwendigen TCP-Verbindung in einer PCAP-Datei!

Zu finden im File ue3_aufgabe2.pcap

Aufgabe 3 – HTTP/2 plain – curl (2.5 Punkte)

Rufe die URI http://nwa1-h2.sin-lab.at/ nun mit curl per HTTP/2 auf!

3a. (0.5 Punkte) Dokumentiere den Output des curl-Aufrufs bei Verwendung des –verbose Parameters!

```
toor@ubu2310:~$ curl -v --http2-prior-knowledge http://nwa1-h2.sin-lab.at:8080
* Host nwa1-h2.sin-lab.at:8080 was resolved.
* IPv6: (none)
* IPv4: 176.9.2.133
*   Trying 176.9.2.133:8080...
* Connected to nwa1-h2.sin-lab.at (176.9.2.133) port 8080
* [HTTP/2] [1] OPENED stream for http://nwa1-h2.sin-lab.at:8080/
* [HTTP/2] [1] [:method: GET]
* [HTTP/2] [1] [:scheme: http]
* [HTTP/2] [1] [:authority: nwa1-h2.sin-lab.at:8080]
* [HTTP/2] [1] [:path: /]
* [HTTP/2] [1] [user-agent: curl/8.5.0]
* [HTTP/2] [1] [accept: */*]
> GET / HTTP/2
> Host: nwa1-h2.sin-lab.at:8080
> User-Agent: curl/8.5.0
> Accept: */*
>
* Remote peer returned unexpected data while we expected SETTINGS frame.  Perhap
s, peer does not support HTTP/2 properly.
* Closing connection
curl: (56) Remote peer returned unexpected data while we expected SETTINGS frame
.  Perhaps, peer does not support HTTP/2 properly.
toor@ubu2310:~$
```

3b. (1 Punkt) Erkläre HTTP Request und HTTP Response diese Aufrufs!

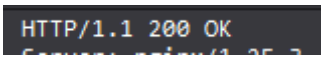
Request-line:

```
> GET / HTTP/2
> Host: nwa1-h2.sin-lab.at:8080
```

- **GET**
  - Methode, zum Abrufen einer Ressource über eine URI.
  - Parameter (z. B. Formulardaten) werden im Query-Part der URI an den Webserver übertragen
- Host: http://nwa1-h11.sin-lab.at:8080/
  - Host wird verwendet, um eine Ressource auf einem Ursprungsserver oder Gateway zu identifizieren. Standardmäßig wird Port 80 verwendet, wenn man keinen angibt. Wir geben 8080 an, da auf diesem Port http 1.1 plain bereitgestellt wird und wir dies für die Übung verwenden möchten [https://www.tutorialspoint.com/http/http_requests.htm]
- Protokoll Version: **HTTP/2** – dafür müssen wir *–http2-prior-knowledge* verwenden

[https://sematext.com/glossary/http-requests/]
[https://datatracker.ietf.org/doc/html/rfc2616#section-5.1]

Status-line:



Status-lines sind normalerweise folgendermaßen aufgebaut:

7. Protokoll Version: **HTTP/1.1** – ist auch die meistverwendete Version
8. Status code: **200** - daraus lässt sich folgern ob der request erfolgreich war oder nicht
9. Status text: **OK** - Eine textliche Beschreibung des status codes

[https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages]

3c. (0.5 Punkte) Erkläre wie die Verwendung von HTTP/2 in diesem Vorgang festgelegt wird (belege!)! (Wie wird realisiert das http 2 verwendet wird)

Beim curl Aufruf wird *--http2-prior-knowledge* verwendet

- Prior Knowledge: Aktivieren der Verwendung von HTTP/2 ohne HTTP/1.1-Upgrade. Wir sagen, dass wir schon wissen, dass http 2 verwendet wird.
- http/1.1-Upgrade: zuerst wird eine http 1.1 Verbindung hergestellt und dann versucht auf http 2 upzugraden. Wenn dies nicht funktioniert, wird auf http 1.1 verblieben.

[https://curl.se/docs/http2.html]
[https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Upgrade]

3d. (0.5 Punkte) Speichere alle Dateneinheiten (und nur diese) der für diesen Aufruf notwendigen TCP-Verbindung in einer PCAP-Datei!

Zu finden im File ue3_aufgabe3.pcap

Aufgabe 4 – HTTP/2 TLS – curl (1.5 Punkte)

Rufe die URI https://nwa1-h2.sin-lab.at/ nun mit curl per HTTP/2 auf!

4a. (0.5 Punkte) Dokumentiere den Output des curl-Aufrufs bei Verwendung des –verbose Parameters!

```
sarah@DESKTOP-6LEAOT9:~$ curl -v https://nwa1-h2.sin-lab.at/
*   Trying 176.9.2.133:443...
* Connected to nwa1-h2.sin-lab.at (176.9.2.133) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
*  CAfile: /etc/ssl/certs/ca-certificates.crt
*  CApath: /etc/ssl/certs
* TLSv1.0 (OUT), TLS header, Certificate Status (22):
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS header, Certificate Status (22):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS header, Finished (20):
* TLSv1.2 (IN), TLS header, Supplemental data (23):
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
* TLSv1.2 (IN), TLS header, Supplemental data (23):
* TLSv1.3 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS header, Supplemental data (23):
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
* TLSv1.2 (IN), TLS header, Supplemental data (23):
* TLSv1.3 (IN), TLS handshake, Finished (20):
* TLSv1.2 (OUT), TLS header, Finished (20):
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (OUT), TLS header, Supplemental data (23):
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA384
* ALPN, server accepted to use h2
* Server certificate:
*  subject: CN=nwa1-h2.sin-lab.at
*  start date: Dec 11 21:35:21 2023 GMT
*  expire date: Mar 10 21:35:20 2024 GMT
*  subjectAltName: host "nwa1-h2.sin-lab.at" matched cert's "nwa1-h2.sin-lab.at"
*  issuer: C=US; O=Let's Encrypt; CN=R3
*  SSL certificate verify ok.
* Using HTTP2, server supports multiplexing
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
* TLSv1.2 (OUT), TLS header, Supplemental data (23):
* TLSv1.2 (OUT), TLS header, Supplemental data (23):
* TLSv1.2 (OUT), TLS header, Supplemental data (23):
* Using Stream ID: 1 (easy handle 0x5652f2827e90)
* TLSv1.2 (OUT), TLS header, Supplemental data (23):
> GET / HTTP/2
> Host: nwa1-h2.sin-lab.at
> user-agent: curl/7.81.0
> accept: */*
```

```
> accept: */*
>
* TLSv1.2 (IN), TLS header, Supplemental data (23):
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* TLSv1.2 (IN), TLS header, Supplemental data (23):
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* old SSL session ID is stale, removing
* TLSv1.2 (IN), TLS header, Supplemental data (23):
* Connection state changed (MAX_CONCURRENT_STREAMS == 128)!
* TLSv1.2 (OUT), TLS header, Supplemental data (23):
* TLSv1.2 (IN), TLS header, Supplemental data (23):
< HTTP/2 200
< server: nginx/1.25.3
< date: Thu, 21 Dec 2023 11:29:40 GMT
< content-type: text/html; charset=UTF-8
<
<html>
<head>
        <title>NWA1 WS2324 HTTP/2</title>
</head>
<body>
<h1>Welcome to nwa1-h2.sin-lab.at!</h1>

<h2>IP information</h2>
<b>Your IP address:</b> 193.170.124.206</br>
<b>Your port:</b> 10000</br>
<h2>Geo Location</h2>

<b>Country, Region, City:</b> AT, Upper Austria, Hagenberg</br><b>Geo Location:</b> 48.1114,13.8590</br><b>ASN:</b> AS18
53 ACONET</br>
<h2>Full Request Headers</h2>

Accept: */*</br>User-Agent: curl/7.81.0</br>Host: nwa1-h2.sin-lab.at</br>Content-Length: </br>Content-Type: </br>
</body>
</html>
* Connection #0 to host nwa1-h2.sin-lab.at left intact
sarah@DESKTOP-6LEAOT9:~$
```

4b. (0.5 Punkte) Erkläre wie die Verwendung von HTTP/2 in diesem Vorgang festgelegt wird (belege!)!

Curl verwendet standardmäßig HTTP/1.1 für HTTP-Server. Wenn man jedoch eine Verbindung zu HTTPS herstellet, versucht er, HTTP/2 automatisch auszuhandeln, oder greift auf 1.1 zurück

[https://everything.curl.dev/http/versions]

4c. (0.5 Punkte) Speichere alle Dateneinheiten (und nur diese) der für diesen Aufruf notwendigen TCP-Verbindung in einer PCAP-Datei!

Zu finden im File ue3_aufgabe4.pcap

Aufgabe 5 – HTTP/3 TLS – curl (1.5 Punkte)

Rufe die URI https://nwa1-h3.sin-lab.at/ nun mit curl per HTTP/3 auf!

5a. (0.5 Punkte) Dokumentiere den Output des curl-Aufrufs bei Verwendung des –verbose Parameters!

```
toor@ubu2310:~$ curl -v --http3 https://nwa1-h3.sin-lab.at
* Host nwa1-h3.sin-lab.at:443 was resolved.
* IPv6: (none)
* IPv4: 176.9.2.133
*   Trying 176.9.2.133:443...
* QUIC cipher selection: TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_CCM_SHA256
*  CAfile: /etc/ssl/certs/ca-certificates.crt
*  CApath: /etc/ssl/certs
*   Trying 176.9.2.133:443...
*  subjectAltName: host "nwa1-h3.sin-lab.at" matched cert's "nwa1-h3.sin-lab.at"
* Verified certificate just fine
* Connected to nwa1-h3.sin-lab.at (176.9.2.133) port 443
* using HTTP/3
* [HTTP/3] [0] OPENED stream for https://nwa1-h3.sin-lab.at/
* [HTTP/3] [0] [:method: GET]
* [HTTP/3] [0] [:scheme: https]
* [HTTP/3] [0] [:authority: nwa1-h3.sin-lab.at]
* [HTTP/3] [0] [:path: /]
* [HTTP/3] [0] [user-agent: curl/8.5.0]
* [HTTP/3] [0] [accept: */*]
> GET / HTTP/3
> Host: nwa1-h3.sin-lab.at
> User-Agent: curl/8.5.0
> Accept: */*
>
```

```
< HTTP/3 200
< server: nginx/1.25.3
< date: Thu, 21 Dec 2023 11:53:40 GMT
< content-type: text/html; charset=UTF-8
<
<html>
<head>
        <title>NWA1 WS2324 HTTP/3</title>
</head>
<body>
<h1>Welcome to nwa1-h3.sin-lab.at!</h1>

<h2>IP information</h2>
<b>Your IP address:</b> 193.170.124.206</br>
<b>Your port:</b> 10003</br>
<h2>Geo Location</h2>

<b>Country, Region, City:</b> AT, Upper Austria, Hagenberg</br><b>Geo Location:</b> 48.1114,13.8590</br><b>ASN:</b> AS1853 ACONET</br>
<h2>Full Request Headers</h2>

Accept: */*</br>User-Agent: curl/8.5.0</br>Content-Length: </br>Content-Type: </br>
</body>
</html>t
* Connection #0 to host nwa1-h3.sin-lab.at left intact
```

5b. (0.5 Punkte) Erkläre wie die Verwendung von HTTP/3 in diesem Vorgang festgelegt wird (belege!)!

Note für mich: Im Allgemeinen wird der Standard-Port für HTTPS (HTTP über TLS/SSL) als Port 443 verwendet. Deswegen muss der Port nicht extra angegeben werden

Als Verknüpfung direkt zu HTTP/3 wird --http3 verwendet, damit Curl versucht, eine QUIC-Verbindung direkt zum angegebenen Hostnamen (und zur angegebenen Portnummer) herzustellen.

Ohne die Option --http3 impliziert eine HTTPS://-URL normalerweise, dass ein Client über TCP (wir haben UDP!) (und TLS) eine Verbindung zur URL herstellen muss.

[https://everything.curl.dev/http/versions/http3]

5c. (0.5 Punkte) Speichere alle Dateneinheiten (und nur diese) der für diesen Aufruf notwendigen QUIC-Kommunikation in einer PCAP-Datei!

Zu finden im File ue3_aufgabe5.pcap

Aufgabe 6 – E-Mail (10 Punkte)

6a. (5 Punkte) Versende eine E-Mail mit Anmeldung per SMTPS (STARTTLS)! Dokumentiere den SMTPS-Dialog und dein Vorgehen (z. B. Tool-Aufruf, SMTP-Kommandos, Authentifzierung, Status-Codes, ...)! Folgende Informationen sind dafür von Bedeutung:

- *Absender (Envelope + From): [S-Nummer]@sin-lab.at (z. B. S2310239123@sin-lab.at)*
- *Empfänger (Envelope + To): tutor nwa1@sin-lab.at*
- *Betreff (Subject): [S-Nummer] no Authentication + secure SMTP (z. B. "S2310239123 no Authentication + secure SMTP")*
- *Datum (Date): aktueller Zeitpunkt des Versendens der E-Mail*
- *Message ID (Message-ID): <YYYYMMDD-HHMMSS@sin-lab.at> auf Basis der Versandzeitpunktes der E-Mail (Date Header).*
- *Inhalt (Body): Eine nette, ev. weihnachtliche Nachricht an den Tutor Simon Titz.*

1. Verbindung zum SMTPS-Server herstellen:
   openssl s_client smtps.sin-lab.at:587 -starttls smtp -ign_eof -crlf
   -**starttls smtp**: Fordert den Server auf, auf eine TLS-verschlüsselte Verbindung zu wechseln, nachdem die Verbindung hergestellt wurde und bevor der SMTP-Dialog beginnt.
   -**ign_eof**: Ignoriert das EOF-Signal (End-of-File) und hält die Verbindung offen. Dies ist oft nützlich, wenn du manuell SMTP-Befehle senden möchtest.
   -**crlf**: Erzwingt die Verwendung von CRLF (Carriage Return + Line Feed) als Zeilenumbruch, was oft in SMTP-Protokollen erforderlich ist.
2. Serverantwort abwarten und EHLO senden (SMTP-Sitzung initiieren):
   EHLO sin-lab.at (=mein domännname)
   Der Server antwortet mit einer Liste der Funktionen, die er unterstützt, sowie einem Statuscode (normalerweise 250 OK, oder z.B. 502 Command not recognized)
3. Authentifizierung durchführen:
   AUTH LOGIN
   <base 64 encoded username>
   <base 64 encoded passwort>
   Der Hauptgrund, warum Base64 hier verwendet wird, liegt in der Struktur von SMTP-Kommandos und -Antworten. SMTP ist ein textbasiertes Protokoll, das nur den Austausch von ASCII-Zeichen erlaubt. Benutzernamen und Passwörter können jedoch binäre Daten enthalten, die nicht direkt in ASCII umgewandelt werden können.

4. Email Struktur eingeben (Siehe Screenshot)

```
toor@ubu2310:~$ openssl s_client -connect smtps.sin-lab.at:587 -starttls smtp -ign_eof -crlf
CONNECTED(00000003)
depth=2 C = US, O = Internet Security Research Group, CN = ISRG Root X1
verify return:1
```

...

```
read R BLOCK
EHLO sin-lab.at
250-mx.sin-lab.at
250-PIPELINING
250-SIZE 52428800
250-ETRN
250-AUTH PLAIN LOGIN
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-DSN
250 CHUNKING
AUTH
501 5.5.4 Syntax: AUTH mechanism
UzIzMTAyMzkwMDhAc2luLWxhYi5hdA==
502 5.5.2 Error: command not recognized
UzIzMTAyMzkwMDhAc2luLWxhYi5hdA==
502 5.5.2 Error: command not recognized
AUTH
501 5.5.4 Syntax: AUTH mechanism
UzIzMTAyMzkwMDhAc2luLWxhYi5hdA==
502 5.5.2 Error: command not recognized
AUTH LOGIN
334 VXNlcm5hbWU6
UzIzMTAyMzkwMDhAc2luLWxhYi5hdA==
334 UGFzc3dvcmQ6
bDBuZ3Bhc3N3b3Jkc2FyZW11Y2hiM3R0ZXJzMWIyMw==
235 2.7.0 Authentication successful
MAIL FROM:<s2310239008@sin-lab.at>
250 2.1.0 Ok
RCPT TO:<tutor_nwa1@sin-lab.at>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
To: tutor_nwa1@sin-lab.at
From: Sarah Feichtinger <s2310239008@sin-lab.at>
```

```
Subject: s2310239008 no Authentication + secure SMTP
Date: Wed, 27 Dec 2023 13:02:00 +0100
Message-ID: <20231227-130200@sin-lab.at>
Hallo Simon,
ich wünsche dir schöne Feiertage.
LG Sarah


.
250 2.0.0 Ok: queued as DACB073
QUIT
221 2.0.0 Bye
closed
toor@ubu2310:~$
```

Base64 Kodierung von Nutzernamen und Passwort:

```
toor@ubu2310:~$ echo -n S2310239008@sin-lab.at | base64
UzIzMTAyMzkwMDhAc2luLWxhYi5hdA==
toor@ubu2310:~$ echo -n l0ngpasswordsaremuchb3tters1b23 | base64
bDBuZ3Bhc3N3b3Jkc2FyZW11Y2hiM3R0ZXJzMWIyMw==
toor@ubu2310:~$
```

6b. (5 Punkte) Rufe Mail Header und Mail Body von zumindest einer E-Mail aus deinem Postfach über IMAPS ab (implizites TLS)! Dokumentiere den IMAPS-Dialog und dein Vorgehen (z. B. Tool-Aufruf, IMAP-Kommandos, Authentifizierung, Status-Codes, ...)!

- *Verwende für die Kommunikation mit dem Mail-Server entweder ncat oder openssl s_client*

1. Verbindung zum IMAPS-Server herstellen:
   openssl s_client -connect imaps.sin-lab.at:993 -crlf
   der s_client baut eine gesicherte Verbindung zu einem IMAPS-Server (sicherer IMAP) auf und einige Informationen wie das Serverzertifikat, dessen Zertifikatskette und die SSL-Handshake Informationen werden angezeigt.
   Hier wird eine Verbindung zum IMAPS-Port 993 hergestellt, der standardmäßig eine verschlüsselte Verbindung (TLS/SSL) erwartet. Daher ist die explizite Verwendung von -starttls nicht erforderlich, da OpenSSL bereits davon ausgeht, dass eine gesicherte Verbindung hergestellt werden soll.
2. Serverantwort abwarten
   ```
   * OK [CAPABILITY IMAP4rev1 SASL-IR LOGIN-REFERRALS ID ENABLE IDLE LITERAL+ AUTH=PLAIN AUTH=LOGIN] Dovecot (Debian) ready.
   ```
3. Authentifizieren (Siehe Screenshots)
4. Postfach abrufen und auslesen

```
toor@ubu2310:~$ openssl s_client -connect imaps.sin-lab.at:993 -crlf
CONNECTED(00000003)
depth=2 C = US, O = Internet Security Research Group, CN = ISRG Root X1
verify return:1
```

....

```
tag LOGIN s2310239008@sin-lab.at l0ngpasswordsaremuchb3tters1b23
tag OK [CAPABILITY IMAP4rev1 SASL-IR LOGIN-REFERRALS ID ENABLE IDLE SORT SORT=DISPLAY THREAD=REFERENCES THREAD=REFS THREAD=ORDEREDSUBJECT MULTIA
PPEND URL-PARTIAL CATENATE UNSELECT CHILDREN NAMESPACE UIDPLUS LIST-EXTENDED I18NLEVEL=1 CONDSTORE QRESYNC ESEARCH ESORT SEARCHRES WITHIN CONTEX
T=SEARCH LIST-STATUS BINARY MOVE SNIPPET=FUZZY LITERAL+ NOTIFY SPECIAL-USE QUOTA] Logged in
? SELECT INBOX
* FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
* OK [PERMANENTFLAGS (\Answered \Flagged \Deleted \Seen \Draft \*)] Flags permitted.
* 0 EXISTS
* 0 RECENT
* OK [UIDVALIDITY 1703671562] UIDs valid
* OK [UIDNEXT 1] Predicted next UID
? OK [READ-WRITE] Select completed (0.001 + 0.000 secs).
? SELECT INBOX
* OK [CLOSED] Previous mailbox closed.
* FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
* OK [PERMANENTFLAGS (\Answered \Flagged \Deleted \Seen \Draft \*)] Flags permitted.
* 1 EXISTS
* 0 RECENT
* OK [UNSEEN 1] First unseen.
* OK [UIDVALIDITY 1703671562] UIDs valid
* OK [UIDNEXT 2] Predicted next UID
? OK [READ-WRITE] Select completed (0.001 + 0.000 secs).
? FETCH 1 (BODY[HEADER])
* 1 FETCH (FLAGS (\Seen) BODY[HEADER] {1409}
Return-Path: <s2310239008@sin-lab.at>
Delivered-To: s2310239008@sin-lab.at
Received: from mx.sin-lab.at
        by mx.sin-lab.at with LMTP
        id v7LvHz4ijGWNawAASwrvIg
        (envelope-from <s2310239008@sin-lab.at>)
        for <s2310239008@sin-lab.at>; Wed, 27 Dec 2023 14:10:22 +0100
Message-ID: <71dca798-de54-4580-a4df-4f698be1eb18@sin-lab.at>
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed; d=sin-lab.at; s=2023;
        t=1703682622;
        h=from:from:reply-to:subject:subject:date:date:message-id:message-id:
         to:to:cc:mime-version:mime-version:content-type:content-type:
         content-transfer-encoding:content-transfer-encoding;
        bh=F1+vowXhprIXljvHYDkGEEkxRnHM/UgHclbXU5buc6Y=;
        b=l/PzFc3z2N44vPQ6lEnD2od2lJqZwsXSYNSpSOPiLeq2BRXRaGnWcURV34IXxYVy1UfRDQ
        AUGnXXF/74qIS288wV1tptHMtatrv2ySF2J3jQcGfMBn9Nqk03tQsXXI4/fy9gi7jivyLy
```

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed; d=sin-lab.at; s=2023;
        t=1703682622;
        h=from:from:reply-to:subject:subject:date:date:message-id:message-id:
         to:to:cc:mime-version:mime-version:content-type:content-type:
         content-transfer-encoding:content-transfer-encoding;
        bh=F1+vowXhprIXljvHYDkGEEkxRnHM/UgHclbXU5buc6Y=;
        b=l/PzFc3z2N44vPQ6lEnD2od2lJqZwsXSYNSpSOPiLeq2BRXRaGnWcURV34IXxYVy1UfRDQ
        AUGnXXF/74qIS288wV1tptHMtatrv2ySF2J3jQcGfMBn9Nqk03tQsXXI4/fy9gi7jivyLy
        kfEP5ImGPqW47Dhca17l1a75S6ton3iu+P2c5XY5Si0RJCVVV+6VVf+TERtw9oHovfjb/Y
        4AnqYPuRUciljDgZEG66t5L4iXSp8XO7ZFATx5PTq9Fud2/ufYB44x5bqWpQ1C/LT1qFd5
        hhBRehIffs3wIqLK0P4HfxxWG1L5qBQSsvPG7EUrV3SwjhRJHSzYU7KAQw1A/w==
Authentication-Results: ORIGINATING;
        auth=pass smtp.auth=S2310239008@sin-lab.at smtp.mailfrom=s2310239008@sin-lab.at
Date: Wed, 27 Dec 2023 14:10:21 +0100
MIME-Version: 1.0
Content-Language: en-US
To: s2310239008@sin-lab.at
From: Sarah F <s2310239008@sin-lab.at>
Subject: test
Content-Type: text/plain; charset=UTF-8; format=flowed
Content-Transfer-Encoding: 7bit

)
? OK Fetch completed (0.002 + 0.000 + 0.001 secs).
```