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**
* 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:

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:**

Ein Bild, das Text, Screenshot, Software, Schrift enthält.

Automatisch generierte Beschreibung

* **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:**

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

* **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!

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

Automatisch generierte Beschreibung

**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:

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]

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!

Ein Bild, das Text, Elektronik, Screenshot, Software enthält.

Automatisch generierte Beschreibung

3b. (1 Punkt) Erkläre HTTP Request und HTTP Response diese 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/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:

1. Protokoll Version: **HTTP/1.1** – ist auch die meistverwendete 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]

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!*

Ein Bild, das Text, Screenshot enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

*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!*

Ein Bild, das Text, Screenshot, Software enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

*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*](mailto:S2310239123@sin-lab.at)*)*
* *Empfänger (Envelope + To): tutor* [*nwa1@sin-lab.at*](mailto: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.

1. Serverantwort abwarten und EHLO senden (SMTP-Sitzung initiieren):  
   *EHLO sin-lab.at (=mein domänname)*

Der Server antwortet mit einer Liste der Funktionen, die er unterstützt, sowie einem Statuscode (normalerweise 250 OK, oder z.B. 502 Command not recognized)

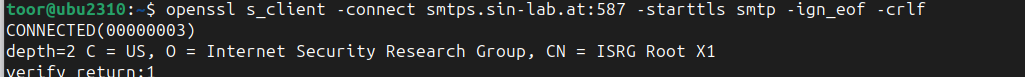
1. Authentifizierung durchführen:

*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.

1. Email Struktur eingeben (Siehe Screenshot)



…

Ein Bild, das Text, Screenshot, Schrift, Karte Menü enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Base64 Kodierung von Nutzernamen und Passwort:

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

*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*

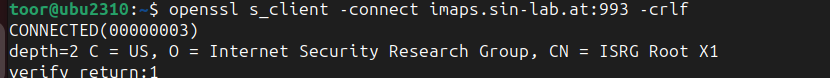
1. Verbindung zum IMAPS-Server herstellen:

*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.

1. Serverantwort abwarten



1. Authentifizieren (Siehe Screenshots)
2. Postfach abrufen und auslesen



….

Ein Bild, das Text, Screenshot enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung