Übungsblatt 02 – Application Layer

Aufgabe A – HTTP(S) & HTML

1. Aus der letzten Übung haben sie die Datei `shell_tutorial.md`. Navigieren sie mithilfe der Kommandozeile in den entsprechenden Ordner. Benennen sie die Datei in *index.html* um.

2. Finden sie die IP-Adresse ihres Rechners heraus. Nutzen sie folgendes Kommando:

```
ifconfig em0
```

   Hinter dem Schlüsselwort *inet* finden sie ihre IP-Adresse.

3. Python liefert ihnen eine rudimentären Webserver frei Haus. Starten sie im Ordner, wo die `shell_tutorial.md/index.html` Datei liegt folgendes Kommando:

```
python3.8 -m http.server 8000
#Fuer die alte VM:
python3.7 -m http.server 8000
```

   Der Server läuft per Default auf dem Localhost-Adapter und ist extern nicht erreichbar.

4. Mit dem Tool Wireshark können sie den Netzwerkverkehr aufzeichnen lassen. Starten sie Wireshark (im Menü unter Applications/Internet/Wireshark). Wählen sie unter *Capture* als Schnittstelle *loopback lo0* aus und starten sie den Mitschnitt (die blau Haiflosse oben links).

5. Rufen sie die Webseite mit `http://IP_ADDRESS:8000` (alternativ: `localhost:8000`) im Browser auf.
   Hinweis: *IP_ADDRESS* muss durch die IP-Adresse aus der zweiten Aufgabe ersetzt werden.

6. Zeichnen sie diesen Aufruf parallel mit Wireshark auf. Finden sie heraus, welche Befehle der Browser an den Server zum Abruf der HTML-Seite gesendet hat.
   a) Setzen sie den Filter hierfür auf das Protokoll *http* (Capture Filter).
   b) Öffnen sie einen *HTTP*-Request (aufklappen eines Eintrags).
   c) Schauen sie in das *HTT*-Protokoll (Vorletzter Eintrag im Protokollstapel). Dort finden sie den *GET* Aufruf, sowie einige zusätzliche Parameter.

7. Verbinden sie sich nun via Kommandozeilen-Programm *telnet* mit dem selben Server und Port.

```
telnet IP_ADDRESS 8000
```

8. Nachdem die Verbindung hergestellt wurde, tippen sie die Befehle ein, die auch durch den Browser gesendet wurden und schauen sie, ob sie als Antwort ihre `shell_tutorial.md/index.html` Datei erhalten.

9. Welche der vom Browser gesendeten Befehle müssen sie mindestens eingeben, um die Webseite zu sehen?

10. Wenn sie zu einem Server eine Verbindung aufbauen, wird serverseitig ein Timeout gestartet, so das, wenn nicht innerhalb einer gewissen Zeitspanne eine Anfrage kommt, der Server die Verbindung beendet. Wenn sie etwas umfangreichere Befehle an den Server senden müssen, oder das Ganze ohne manuellem Eintippen automatisieren wollen, können Sie das Tool *netcat* nutzen.

11. Mit dem Werkzeug *netcat* können Website direkt über die Kommandozeile aufgerufen werden.

```
1 nc –v IP_ADDRESS 8000
```

    a) Schreiben sie die gleichen HTTP-Befehle zum Abruf der Webseite jetzt in eine lokale Textdatei (alle Zeilenumbrüche beachten!).
    Dise können sie mithilfe der *printf* Funktion beispielsweise lösen:

```
1 printf "GET \ HTTP1.1 ..." > request
```

    Alternativ könne sie dies auch via *vim* erledigen.

    b) Lassen sie sich den Inhalt der Datei auf der Kommandozeile nach Std-Out ausgeben (*cat*).

    c) Leiten sie diese Ausgabe mittels einer *Pipe* (|) als Eingabe in den Befehl *netcat* um. Rufen sie *netcat* dabei mit Parametern so auf, dass es eine Verbindung wieder zum gleichen Webserver und Port wie bisher aufbaut.
    Wenn sie alles richtig gemacht haben, sehen sie wieder die gleiche Ausgabe.

12. Bis jetzt werden ihre Daten im Klartext übertragen, Fremde können also prinzipiell ihre Verkehr mitlesen. Mithilfe eines zusätzlichen Protokolls kann der Verkehr für den Nutzer transparent verschlüsselt werden.
    Ein Programm um beliebige Verbindungen nachträglich mit SSL/TLS zu versehen ist Teil des *OpenSSL*-Toolkits. Mit dem Befehl *openssl s_server* können sie Serveranwendungen, welche kein TLS unterstützen aber über Std-In Befehle entgegennehmen darüber absichern. Mit dem Befehl `openssl s_client` wiederum können sie Client-Verbindungen mit TLS-Unterstützung aufbauen oder auch manuell ausführen.

    a) Zeichnen sie alle Abrufe der Webseite mit mit Wireshark auf und prüfen sie, was sie dort sehen können.
    Achtung: Sie müssen hierfür bei Wireshark den Adapter auf *em0* setzen, da die Verbindungen nicht über den Localhost laufen!

b) Bauen Sie noch einmal testweise eine HTTP-Verbindung mit *telnet* oder *netcat* zum Webserver des Rechenzentrums der HTW (`www.rz.htw-berlin.de`) auf und fragen sie die Startseite an.

c) Nutzen sie nun anstelle von *telnet* das Programm `openssl s_client` um eine Verbindung zum gleichen Webserver aber auf dem *HTTPS* Port aufzubauen (Welcher Port wird für HTTPS genutzt?). Rufen sie nach erfolgreichem Verbindungsaufbau wieder die Startseite ab. Dazu müssen *openssl s_cleint* vie folgt aufrufen:

```
1 openssl s_client -connect HOSTNAME:PORT
```

d) Welche Informationen über den TLS-gesicherten Server bekommen Sie mit `openssl s_client`? Wo sehen sie z.B. die Gültigkeit des Zertifikates? Den Zeitraum der Gültigkeit? Wer hat das Zertifikat ausgestellt?

e) Setzen sie nach erfolgreicher Verbindung ihren *HTTP*-Request ab.

Aufgabe B – Python

1. Starten sie eine interaktive Python-Session.

```
1 python
```

2. Python als Taschenrechner.
   a) Berechnen sie folgende Terme:
      - $2 + 3$
      - $3.5 + 4.5$
      - $3 + 3.5$

      Lassen sie sich mit *type()* den jeweiligen Datentyp ausgeben.
   b) Deklarieren sie folgende Variablen:
      - Variable a mit Wert 2
      - Variable b mit Wert 3
      - Variable c als Summe von a und b
      - Eine Variable $s1$ die den Inhalt: *Netzwerke* und eine Variable $s2$ mit dem Inhalt *verteilte Systeme* enthält.
      - Fügen sie in der Variablen s die beiden Texte $s1, s2$ mit dem + Operator zusammen.
3. Mithilfe der *print()* Funktion können Ausgaben auf der Kommandozeile ausgegeben werden.

a) „printen“ sie die Zeile „Hallo, Welt!“ auf der Kommandozeile.

4. Kontrollstrukturen:

```
if x % 2:
  print("x is even")
else:
  print("x is odd")
```

a) Schreiben und testen sie eine Block der prüft, ob zwei Variablen numerisch identisch sind („==“).

b) Schreiben und testen sie eine Block der eine Ausgabe ausführt, wenn die erste Variable größer als die zweite ist.

5. For-Schleifen:

```
for i in range(X) # X ist hier die Anzahl der Wiederholungen
```

a) Schreiben und testen sie mithilfe eine For-Schleife, die ihnen 10 mal „Hallo, Welt!“ ausgibt.

b) Schreiben und testen sie mithilfe einer For-Schleife die Gaußsche Summenformel. Also die Summe der Zahlen von 0 bis m

$$0 + 1 + 2 + ... + m = \sum_{k=0}^{m} k$$

6. While-Schleifen:

```
while i <= n: # X ist hier die Anzahl der Wiederholungen
```

a) Schreiben sie eine *While*-Schleife, die alle Zahlen kleiner der Variablen c nacheinander ausgibt.

7. Funktionen

```
  def foo(): # Funktion ohne Argument
  def bar("Blub"): #Funktion mit Argument hard coded
  def bar2(m): Funktion mit Argument als Variable
  def square(x): # Funktion mit Argument und Rückgabewert
    return x**2
```

a) Schreiben sie eine Funktion `euclid` die den euklidischen Algorithmus implementiert. Nutzen sie eine Funktiosdeklaration.

Fakultativ: Aufgabe C – E-Mail mit POP3, IMAPv4 & SMTP

Zum Abruf von E-Mails gibt es die beiden Protokolle *POP3* und *IMAPv4*.

1. Bauen Sie nun mit `openssl s_client` eine gesicherte Verbindung zum einem Ihrer Mail-Server (z.B. dem der HTW-Berlin: `mail.rz.htw-berlin.de`) auf und loggen Sie sich auf Ihrem Account ein, um dann Ihre Mails abzurufen.
   Achtung – bitte loggen Sie sich nicht ohne TLS aus dem Labor heraus auf einem Mailserver ein. Andere Studierende werden sicherlich parallel Wireshark laufen lassen und könnten dann Ihre Zugangsdaten sehen!

2. Bauen Sie mit `openssl s_client` eine Verbindung zum POP3-SSL Port auf und loggen Sie sich mit Ihren Nutzerdaten ein. Anschließend rufen Sie erst die Liste aller Nachrichten und dann eine spezielle Nachricht ab, um sie zu lesen. (Eine beispielhafte POP3-Session mit den notwendigen Befehlen finden Sie leicht im Netz oder z.B. bei Wikipedia).

3. Setzen Sie das Gleiche mit *IMAP* um. **Hinweis**: alle Mail-Protokolle unterstützen auch das *STARTTLS* Kommando. Damit kann eine nicht gesicherte Verbindung nachträglich noch mit TLS abgesichert werden. Sie bauen also im Klartext z.B. zum POP3 Server auf dem Standard-Port eine Verbindung auf und senden dann im Klartext das Kommando `STARTTLS`. Daraufhin wird auf diesem Port eine verschlüsselte Verbindung aufgebaut und alle nachfolgenden Befehle können nicht mehr von anderen mitgelesen werden. OpenSSL/LibreSSL unterstützt dies auch für etliche Protokolle.

4. Starten Sie `openssl s_client` und mit dem Parameter `starttls` eine gesicherte Verbindung zum POP3-Standard-Port. Versuchen Sie sich dann mit falschen Login-Daten anzumelden. Beenden Sie die Verbindung.

5. Zeichnen Sie den Verbindungsaufbau parallel mit Wireshark auf und prüfen Sie, was sie davon sehen können. **Hinweis**: Sollten Sie kein E-Mail-Programm griffbereit haben, können Sie das natürlich auch das per Hand erledigen. Das SMTP-Protokoll ist ebenfalls relativ einfach und text-basiert.

6. Bauen Sie mit `openssl s_client` nacheinander eine Verbindung zu allen drei SMTP-Ports auf. Finden Sie heraus, welche Ports direkt mit SSL gesichert sind und welche Ports mit STARTTLS nachträglich gesichert werden müssen.

7. Loggen Sie sich nun auf dem SSL-Port mit openssl ein und versenden Sie eine E-Mail.
   Hinweis: Viele Server nutzen inzwischen beim Versand zur Spambekämpfung *SMTP-AUTH* (SMTP-Authentication) um nur eigenen Nutzern zu erlauben, Mails

an fremde Server zu versenden. An eigene E-Mail-Adressen des SMTP-Server können Sie aber immer senden (d.h. wenn Sie mit dem SMTP-Server der HTW verbunden sind, können sie immer eine E-Mail an eine Empfängeradresse „s0XXXXXX@htw-berlin.de“ senden. Wollen Sie eine E-Mail an z.B. „...@posteo.de“ senden, müssen Sie sich vorher authentifizieren.