

Übungsblatt 02 – Application Layer

Manueller Funktionstest

Viele Protokolle im Internet, insbesondere die älteren, sind Text-basiert, d.h. es werden einzelne, auch für Menschen, lesbare Befehle zwischen den Rechnern hin und her geschickt. Entsprechende Server können Sie demzufolge auch „per Hand“ als Client bedienen, um deren Funktion zu testen bzw. auf deren Dienste zuzugreifen. Dazu reicht beispielsweise Programme wie *netcat* oder *telnet* aus.

Andererseits können Sie aber auch einfach Skripte mit beliebige Befehlsfolgen zusammenstellen und automatisiert an den Server senden. Sei es zum automatisieren auf der Kommandozeile oder als Fuzzy-Test mit beliebigen Datenmüll um zu prüfen, wie Fehlertolerant Ihr eigener Server ist.

Aufgabe A – HTTP(S) & HTML

Aufsetzen eines einfachen Webservers:

Im wesentlichen liefert ein Webserver Dateien über einen Socket aus. Das heißt dieser bekommt anfragen, und bei entsprechender Berechtigung, liefert der Webserver die angeforderten Dateien aus.

- 1.) Auf jeder VM ist ein *Apache* Webserver vorinstalliert. Als Hilfestellung für die Konfiguration des Webservers gehen Sie wie folgt vor:

Für das Binding des Webservers muss in der Apache Konfiguration (s. `/etc/apache2/apache2.conf`) die IP-Adresse und optional der Port mit dem Befehl *Listen* gesetzt werden.

```
1 Listen IP:Port
2 # Example:
3 # Listen 10.0.0.1:80
```

Ihre IP-Adresse können Sie mithilfe folgenden Kommando in Erfahrung bringen:

```
1 ip a s enp3s0 | awk '/inet/ {print $2}'
```

Mit dem Befehl *apachectl* kann der Apache angesprochen werden. Entsprechen sollten die Befehle:

```
1 sudo apachectl configtest
2 sudo apachectl start
```

ausgeführt werden. Der erste testet, ob die Konfiguration des Servers korrekt ist. Der zweite Befehl startet den Webserver.

- 2.) Konfigurieren Sie den Apache, sodass die Default-Site auf der Laboradresse der VM ausgeliefert wird.

- 3.) Rufen Sie die Webseite einer anderen VM im Browser auf (mit HTTP ohne TLS-Verschlüsselung).
- 4.) Zeichnen Sie diesen Aufruf parallel mit Wireshark auf und finden Sie heraus, welche Befehle der Browser an den Server zum Abruf der HTML-Seite gesendet hat. Welcher Port wurde dazu verwendet?
- 5.) Verbinden Sie sich nun mit dem Kommandozeilen-Programm *telnet* mit dem selben Server und Port.
- 6.) Nachdem die Verbindung hergestellt wurde, tippen Sie die Befehle ein, die auch durch den Browser gesendet wurden und schauen Sie, ob Sie als Antwort die Startseite des Webserver erhalten.
- 7.) Welche der vom Browser gesendeten Befehle müssen Sie mindestens eingeben, um die Webseite zu sehen?
- 8.) 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.
 - a.) Schreiben Sie dieselben HTTP-Befehle zum Abruf der Webseite jetzt in eine lokale Textdatei (alle Zeilenumbrüche beachten!).
 - 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 Startseite des Webserver.
- 9.) Damit bei Klartext-Protokollen keine Nutzerdaten durch Fremde mitgelesen werden können, werden von vielen Diensten die eigentlich originalen Protokolle in eine TLS-Verbindung verpackt, um die Daten für die Anwendung transparente zu verschlüsseln.

Ein Programm um beliebige Verbindungen nachträglich mit SSL/TLS zu versehen ist Teil des *OpenSSL*-Toolkits.¹ 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

¹Als Sicherheitsbewusste Informatiker sollten Sie jedoch eher auf LibreSSL setzen!

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.

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.

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.

c.) 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?

Aufgabe B – 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.

Aufgabe C – Domain Name System (DNS)

1.) DNS-Requests:

- a.) Fragen Sie mit jedem der Kommando der Hausaufgaben jeweils einmal einen Hostnamen (bspw. www.htw-berlin.de), einen Domainnamen (htw-berlin.de) und eine IP-Adresse (141.45.5.100) ab.
- b.) Schauen Sie sich die Ausgabe von *dig* bei der Abfrage der IP-Adresse genauer an – dort werden Sie in der „Question Section“ sehen, das nach dem A-Resource-Record mit dem Namen 141.45.5.100 gefragt wurde. Wenn Sie den Namen zu dieser IP-Adresse suchen – welchen Resource-Record müssen Sie dann anstelle des A-Records erfragen?
- c.) In welcher Form müssen Sie dann die IP-Adresse angeben? (Test mit `dig -t <record-type> <richtiges-format-ip-adresse>`).
- d.) Denken Sie sich einen Domainnamen aus, den es wahrscheinlich geben könnte, aber den noch niemand vom Netzwerk der HTW-Berlin aus innerhalb der letzten Stunden angefragt hat (z.B. www.uriminzokkiri.com oder www.northkoreatech.org). Erfragen Sie diesen Namen zweimal kurz hintereinander via *dig* und vergleichen Sie die beiden Ausgaben. Worin unterscheiden sich beide Einträge? Begründen Sie diese Unterschiede!

e.) Erfragen Sie mit *dig* und *nslookup* den zuständigen Mail-Server für die Domain htw-berlin.de.

f.) Erzwingen Sie mit *dig* und *nslookup*, dass die Namensauflösung nicht mit dem Standard-Nameserver des Betriebssystems, sondern mit einem öffentlichen Nameserver (bspw.: 9.9.9.9) erfolgt. Testen Sie am Besten zuerst mit *dig*, da dies Ihnen immer sagt, welche Nameserver sie genutzt haben.

2.) DNS-Resolver: Das Listing zeigt die „*resolv.conf*“ eines Servers.

```
1 # Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
2 # DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
3 nameserver 141.45.3.100
4 search f4.htw-berlin.de
```

Was bedeuten die Einträge mit den Schlüsselwörtern: „*nameserver*“ und „*search*“?