Web-Server im echten Leben

Bevor wir uns das Verhalten eines echten Web-Servers anschauen, noch ein paar Bemerkungen:

- **IP-Adresse vs Name**

Bisher haben wir immer eine IP-Adresse verwendet, ab hier müssen wir aber einen Namen nehmen, hier `schulung.qgelm.de`. Auf dem Erprobungswebserver werden nämlich noch mehr Webseiten bedient, z.B. www.qgelm.de, also verschiedene Subdomänen zu der Domäne qgelm.de. Die werden in meinem Fall alle der selben IP-Adresse zugewiesen, also von einem Computer über eine Netzwerkschnittstelle bedient. Der Web-Server kann sie daher nur über die Header-Zeile `Host: schulung.qgelm.de` auseinander halten.

- **Welcher TCP-Port?**

Traditionell verwenden Web-Server immer den Port 80 zum Lauschen auf Verbindungen eines TCP-Client. Das ist aber nicht in Stein gemeißelt. Sofern der Nutzer keine Angaben zum Port macht, nimmt der Browser aber immer stillschweigend den Port 80 an. Übrigens unterscheiden sich HTTP und HTTPS in der Port-Nummer: HTTPS verwendet konventionell Port 443. Auch das kann explizit anders benutzt werden. HTTPS ist normalerweise die gleiche Server-Anwendung bezüglich der HTTP-Logik. Das HTTP-Protokoll samt aller Request/Response Zeilen und Inhalte werden aber durch eine zwischengeschaltete TLS-Software verschlüsselt übertragen. TLS verwendet zum Aufbau einer Krypto-Beziehung ein eigenes Protokoll zwischen Client und Server. TLS packt das HTTP-Protokoll somit in das eigene Protokoll ein.

Nach diesen Vorbemerkungen nun zur Arbeitsweise eines ausgewachsenen statischen HTTP-Servers:

Ich verwende hier als Server-Software den *nginx*. Das ist neben *apache* und einigen anderen ein ziemlich weit verbreitetes Programm. Die Konfiguration des *nginx* für unser recht simples Beispiel (statischer, unverschlüsselter Webserver unter der Adresse `http://schulung.qgelm.de`) sieht so aus:

```
http {
  server {
      listen 80;
      listen  [::]:80;
      server_name schulung.qgelm.de;
      root       /home/thomas/Schulung/html/;
      index      index.html;

      # This will deny access to any hidden file (beginning with a .period)
      location ~ /\. { deny  all; }
  }
}
```

Das meiste sollte nach dem bereits Erlernten recht selbst erklärend sein: Der http-Server, der auf den Namen `schulung` hört und auf dem für http gebräuchlichen TCP Port 80 lauscht, liefert angefraget Ressourcen unterhalb des *root* Verzeichnisses aus. Wenn die Ressource nur / lauten sollte, dann wird nach einer Datei `index.html` in dem jeweiligen Verzeichnis geschaut. Und als kleines Sicherheitsfeature werden Dateien, die mit einem „.“ beginnen vom Server ignoriert, also nicht

ausgeliefert.

In dem *root* Verzeichnis sind übrigens auch die Dateien, die wir bei unserer letzten Aktion verwendet haben.

Ein Aufruf

http://schulung.qgelm.de/test.txt

in einem beliebigem Browser oder mittels `curl` sollte das erwartete Ergebnis liefern. Spannender wird jetzt das Live-Experiment mit der im letzten Abschnitt bereits vorgestellten HTML-Datei, die im *root* Verzeichnis als *index.html* hinterlegt ist:

http://schulung.qgelm.de

(`/index.html` habe ich jetzt schon einfach weggelassen, s.o.)

> Tipp: Die tatsächliche Kommunikation auf Ebene von TCP/IP kann man sich auch direkt anschauen. Dann allerdings wieder in einem Terminal auf dem Server:
>
> ```
> tcpdump -i any port 80
> ```

Genug mit statischen Webseiten, was bedeutet in dem Zusammenhang dynamisch?