Marek Kubica kubica@in.tum.de, Michael Kerscher kerschem@in.tum.de

Punkteverteilung:

27	28	29	Σ

Aufgabe (27)

(a) Anfrage

```
GET / HTTP/1.0
```

Antwort

```
HTTP/1.1 200 OK
Date: Wed, 07 Jul 2010 22:18:57 GMT
Server: Apache/2.2.9 (Debian) mod_auth_kerb/5.3 DAV/2 PHP
   ➥/5.2.6-1+lenny8 with Suhosin-Patch mod_ssl/2.2.9
   ➥OpenSSL/0.9.8g
X-Powered-By: PHP/5.2.6-1+lenny8
Set-Cookie: fe_typo_user=a895257f5b0fa3758bcb7d2165ffa26f
Cache-Control: no-cache
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE html
        PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
        <meta http-equiv="Content-Type" content="text/
           ➥html; charset=iso-8859-1" />

<!--
        This website is powered by TYPO3 - inspiring
           ➥people to share!
        TYPO3 is a free open source Content Management
           ➥Framework initially created by Kasper
           ➥Skaarhoj and licensed under GNU/GPL.
        TYPO3 is copyright 1998-2009 of Kasper Skaarhoj.
           ➥Extensions are copyright of their respective
           ➥owners.
        Information and contribution at http://typo3.com/
           ➥ and http://typo3.org/
-->

        <base href="http://www.net.in.tum.de/" />
        <link rel="stylesheet" type="text/css" href="
           ➥typo3temp/stylesheet_d3922e1b66.css" />
        <title>TUM Info VIII: Startseite</title>
```

(b) Die Source-IP ist unterschiedlich zwischen den Teilnehmern. Zusätzlich wird jede Anfrage über einen “Ephemeral port” abgewickelt, d.h. einen hohen Port der nur für diese Clientverbindung temporär genutzt wird.

(c) Um die Verbindung aufzubauen ist es erstmal nötig die IP-Adresse des Hosts zu wissen. Dazu wird erst der DNS-Server befragt. Im Falle das der Server nicht im lokalen Netzwerk liegt müssen die auf dem Weg befindlichen Router in Anspruch genommen werden um die DNS-Abfrage weiterzuleiten. Sobald die IP-Adresse bekannt ist, kann versucht werden eine TCP-Verbindung zum eigentlichen Server aufzubauen, diese läuft analog zu DNS ab, nur dass die Verbindung nicht zum DNS-Server (ggf. dem des Providers) sondern direkt zum Webserver aufgebaut wird.

(d) Dadurch dass HTTP/1.1 optional Pipelining unterstützt spart man sich den erneuten Verbindungsaufbau bei jedem Request an den gleichen Server. Dadurch spart man den TCP-Verbindungsaufbau-Overhead und die Latenz sinkt. Bei vielen kleinen Dateien von einem Server, wie sie bei HTML üblich sind (HTML, CSS, JS, Bilder) kann dies einen großen Unterschied machen.

(e) Erste Anfrage:

```
GET / HTTP/1.1
Host: www.diadem-firewall.org
```

Nun haben wir eine Seite die per HTTP-Equiv Refresh weiterleitet. Wirken wie aus der Vergangenheit, diese “klicken Sie hier um weitergeleitet zu werden”-Links. Das führt uns dann zur zweiten Anfrage:

```
GET /index.php HTTP/1.1
Host: www.diadem-firewall.org
```

Daraufhin bekommen wir die eigentliche Seite. “The real WTF” ist, warum der Webserver nicht `DirectoryIndex index.php` oder vergleichbar gesetzt hat.

(f) Der Anfang ist auch diesmal der gleiche, lediglich der übertragene Host-Header ist natürlich anders.

```
GET / HTTP/1.1
Host: ilab.net.in.tum.de
```

Wir bekommen ein “302 Found”, was die Browser als Redirect interpretieren, daher wird folgender Request abgeschickt, der den Pfad aus dem Location-Header nutzt:

```
GET /pages/view.php?address=p3&config=2010ss HTTP/1.1
Host: ilab.net.in.tum.de
```

Aufgabe (28)

(a)
```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="lecture">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="name" type="xsd:string"/>
        <xsd:element name="lecturer" type="xsd:string"/>
        <xsd:element name="tutors">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="assistant" maxOccurs="
                 ➥unbounded">
```

```
                <xsd:complexType mixed="true">
                  <xsd:attribute name="type" use="
                     ➥required">
                    <xsd:simpleType>
                      <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="student"/
                           ➥>
                        <xsd:enumeration value="staff"/>
                      </xsd:restriction>
                    </xsd:simpleType>
                  </xsd:attribute>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Um gegen das XML Schema zu validieren wurde ein Validator geschrieben, der die Python XML Library lxml nutzt. Sein Quellcode ist unter `http://gist.github.com/467492` verfügbar.

(b) Damit man feststellen kann, ob die Eingabe die man bekommen hat auch eine gültige Eingabe ist. So kann man auf eine standarisierte Weise auch die gewünschten Eingabe spezifizieren gegen die der Programmautor prüfen kann.

(c) Der XSLT-Code:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.
   ➥org/1999/XSL/Transform">
  <xsl:output method="html"/>
  <xsl:template match="/">
    <!-- DOCTYPE html without XSLT-compat because html5
       ➥validators choke on that -->
    <xsl:text disable-output-escaping="yes">&lt;/
       ➥xsl:text>!DOCTYPE html<xsl:text disable-output-
       ➥escaping="yes">&gt;</xsl:text>
    <html>
      <head>
        <title>Infoseite</title>
        <link rel="stylesheet" type="text/css" href="28.
           ➥css"/>
      </head>
      <body>
        <div>
          <h1>
            <xsl:value-of select="/lecture/name"/>
          </h1>
```

```
          <h2>
            <xsl:value-of select="/lecture/lecturer"/>
          </h2>
          <ul>
            <xsl:for-each select="/lecture/tutors/*">
              <li>
                <xsl:value-of select="."/> (<xsl:value-of
                   ➥ select="@type"/>)
              </li>
            </xsl:for-each>
          </ul>
        </div>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Das Cascading Style Sheet für die generierte HTML-Seite.

```
html {
        font-family: sans-serif;
}

h1 {
        margin-top: 0;
        margin-bottom: 0;
        padding-left: 0.1em;
        background-color: white;
}

h2 {
        margin-top: 0.1em;
        margin-bottom: 0.1em;
        font-style: italic;
        font-weight: normal;
        font-size: 110%;
}

div {
        padding-top: 0.5em;
        padding-left: 0.5em;
        background-color: #ccffcc;
}

ul {
        list-style-type: none;
        margin-top: 0;
        margin-bottom: 0;
        padding-left: 0;
        padding-bottom: 0.5em;
```

```
        font-size: 80%;
}
```

Der XSLT-Code wurde mit `xsltproc` zu HTML kompiliert und rendert zusammen mit dem CSS eine HTML-Seite, die aussieht wie die Vorlage.

(d) Der Vorteil ist, dass die HTML-Seiten vom XSLT-Prozessor des Browsers generiert werden können, somit reicht es, in der XML-Datei die "Rohdaten" zu ändern und schon kann der Browser des Clients diese Rohdaten *sofort* darstellen, da nicht erst HTML als zusätzliche Zwischenschicht generiert werden muss.

Aufgabe (29)

(a) Host A: 10.0.0.2. Die Adresse ist relativ beliebig, sie sollte aber im Subnetz liegen. Da 10.1 eine logische Wahl für den Default-Router (NAT) ist, ist 10.2 gleich die nächste passende Möglichkeit.

(b)
- Zwischen Host A und NAT

  Ausgehend:

Source IP address	10.0.0.2
Source port	33000
Destination IP address	80.80.80.80
Destination port	80
Time-to-Live	64

  Es wird ein "Ephemeral Port" am Client verwendet, sprich der Quellport ist ein hoher Port im Bereich von 32768 bis 61000 (es wird der Einfachkeit halber ausgegangen dass alle Hosts im Beispiel Linux nutzen).

  Eingehend:

Source IP address	80.80.80.80
Source port	35000
Destination IP address	10.0.0.2
Destination port	33000
Time-to-Live	62

- Zwischen NAT und Router A

  Ausgehend:

Source IP address	131.159.24.19
Source port	34000
Destination IP address	80.80.80.80
Destination port	80
Time-to-Live	63

  NAT speichert sich die Zuordnung von Port 33000 von Host A zum Port 34000 seines öffentlichen Interfaces und übersetzt die übermittelten Pakete. Der Port 34000 wurde vom NAT frei gewählt.

  Eingehend:

Source IP address	80.80.80.80
Source port	35000
Destination IP address	131.159.24.19
Destination port	34000
Time-to-Live	63

- Zwischen Router A und Server B

  Eingehend:

Source IP address	131.159.24.19
Source port	34000
Destination IP address	80.80.80.80
Destination port	80
Time-to-Live	62

Ausgehend:

Source IP address	80.80.80.80
Source port	35000
Destination IP address	131.159.24.19
Destination port	34000
Time-to-Live	64

Auch hier wird vom Server ein "Ephemeral Port" gewählt um Port 80 für weitere Clientverbindungen freizuhalten.

(c) Nein, wenn er versucht 10.0.0.10 zu erreichen stellt er fest das es eine für lokale Netzwerke reservierte Netzwerkadresse ist. Eine Möglichkeit 10.10 zu erreichen wäre ein Postforwarding am NAT, das externe Zugriffe auf einen bestimmten Port von 131.159.24.19 auf einen bestimmten Port von 10.10 weiterleitet. In diesem Fall müsste 77.77.77.77 als Ziel-IP 131.158.24.19 verwenden und als Port den weitergeleiteten Port.