6 Informationen verteilen per Web-Server

Immer mehr Einrichtungen nutzen Web-Server, um Informationen innerbetrieblich im Intranet und für Externe im Internet bereit zu stellen.

Dies bleibt nicht ohne Einfluss auf die innerbetriebliche Kommunikationskultur.

Grundlagen des Web sind:

- HyperText Markup Language (HTML), die Sprache der Web-Seiten.
- HyperText Transfer Protocol (HTTP), das die Seitenanforderungen und Übertragungen regelt und
- Uniform Resource Locator (URL), die eindeutige Adresse für eine Information im Internet.

Alle marktführenden Linux-Distributionen enthalten einen Web-Server, meist den Apache. SuSE installiert ihn automatisch.

Apache hat übrigens nichts mit dem Indianerstamm zu tun, sondern verballhornt *a patchy server*. Die Wurzeln des Apache liegen in Anpassungen (Patches) des NCSA Web-Servers. Inzwischen entwickelt eine Gruppe von etwa 20 Programmierern, die *Apache HTTP Server Group*, Apache eigenständig für Linux und NT weiter.

Dieses Kapitel beschreibt die Grundlagen, die Sie benötigen, um Apache sinnvoll im lokalen Netz einzusetzen.

Der Web-Server Apache erlaubt, Seiten nur für geschlossene Benutzergruppen zugänglich zu machen. Nur wer über einen geeigneten Benutzernamen und das zugehörige Passwort verfügt, kann dann auf die geschützten Seiten zugreifen.

Da Browser Benutzernamen und Passwort normalerweise unverschlüsselt an den Web-Server übertragen, was ein unnötiges Sicherheitsrisiko darstellt, sollten Sie sich auch damit beschäftigen, wie man die Datenübertragung verschlüsseln kann.

Lesen Sie in diesem Kapitel, wie

- Web-Server arbeiten (6.2),
- man Apache installiert und einrichtet (6.3),
- man das Einrichten und Pflegen von Web-Inhalten organisatorisch löst,
- man eine Zugriffssteuerung für geschlossene Nutzergruppen einrichtet,
- was virtuelle Server sind,
- wie man gesicherte Zugriffe mit Secure Sockets Layer realisiert und
- wie man Web-Server-Zugriffe auswerten kann.

6.1 Wann brauchen Sie einen eigenen Web-Server?

Eigentlich immer. Statt Informationen auf einem schwarzen Brett in der Kantine auszuhängen, kann man auch Seiten auf dem lokalen Web-Server erstellen und dort aktuelle Ankündigungen und Termine hinterlegen. Wichtig ist nur, sich regelmäßig um Pflege und Aktualisierung zu kümmern.

Den eigenen Internetauftritt sollte man zuerst im lokalen Netz entwickeln und testen, um sich Blamagen zu ersparen.

6.2 So arbeiten Web-Server

Beim HyperText Transfer Protocol (http), sendet der Client (meist Browser genannt) eine Anfrage nach einem Dokument an den Server (den http-Dämon). Dieser liefert ihm dann einerseits den MIME-Typ der angeforderten Datei und die Datei selber. Aus dem MIME-Typ schliesst der Client nun, was er mit den empfangenen Daten anfangen soll.

Die häufigsten MIME-Typen zeigt er so an:

- text/html als HTML-Dokument,
- text/plain als normalen ASCII-Text und
- image/gif als GIF-Grafik.

Daneben gibt es noch eine viele weitere Typen. Auf dem Linux-Server enthält die Datei `/etc/httpd/mime.types` über 100 Einträge der Form:

```
text/html          html htm
text/plain         asc txt c h
image/gif          gif
```

Der Web-Server übermittelt Dateien mit der Endung `.html` oder `.htm` als Typ `text/html`. Zeigt der Browser HTML-Dateien im Quellcode an, gibt es meist ein Problem mit `/etc/httpd/mime.types`.

Für jede laufende Verbindung ist ein httpd-Prozess zuständig. Der WWW-Server startet also bei Bedarf Kopien seiner selbst, die dann die zusätzlichen Verbindungen bedienen und beendet diese dann wieder. Wie viele derartige Prozesse laufen dürfen, läßt sich über die Konfigurationsdatei einstellen.

Trotz vieler Prozesse verschwendet Apache dank Linux (oder des jeweils verwendeten Systems) kein Speicherplatz, weil alle Kopien des WWW-Servers den Speicher gemeinsam nutzen.

6.3 Web-Server Apache installieren und einrichten

SuSE legt den Apache in die Serie `n` im Paket `apache` und installiert ihn eigentlich immer. Überzeugen Sie Sich, dass er sofort lauffähig ist, indem Sie von einem Client aus seine URL `http://192.168.1.2` aufrufen. Der Browser müßte folgende Startseite anzeigen:

Abbildung 6.1: Standardstartseite im Browser

SuSE publiziert einen Teil der auf dem Server installierten Dokumentation über den Web-Server.

Folgende Dateien sind für die Konfiguration des Web-Servers Apache wichtig:

Datei	*Bedeutung*
`/usr/sbin/httpd`	Das Binärprogramm des Apache
`/etc/httpd/`	Verzeichnis für die Konfigurationsdateien
`/etc/httpd/httpd.conf`	Hauptkonfigurationsdatei
`/etc/httpd/mime.types`	Datei mit den bekannten Dateitypen
`/etc/httpd/access.conf`	Nicht mehr notwendige Konfigurationsdatei
`/etc/httpd/srm.conf`	Nicht mehr notwendige Konfigurationsdatei
`/usr/local/httpd/`	Wurzelverzeichnis des Web-Servers
`/usr/local/httpd/htdocs/`	Verzeichnis für normale Webdokumente
`/usr/local/httpd/cgi-bin/`	Verzeichnis für ausführbare Programme (CGI)
`.htaccess`	Konfigurationsdatei im jeweiligen Web-Verzeichnis

Tabelle 6.1: Dateien und ihre Bedeutung für die Konfiguration des Apache

Ursprünglich brauchte man zum Einrichten des Apache die Dateien `httpd.conf`, `access.conf` und `srm.conf`, heute jedoch nur noch die Datei `httpd.conf`; die beiden anderen Dateien sind ohne Inhalt und nur noch aus Kompatibilitätsgründen vorhanden.

Die mehr als 1000 Zeilen lange Konfigurationsdatei `/etc/httpd/httpd.conf` hat SuSE recht gut kommentiert. Der folgende Text erläutert wichtige Abschnitte der Konfigurationsdatei, die für eine normale Nutzung bzw. das grundlegende Verständnis wichtig sind. Der Apache ist aber auch ohne Änderungen an der Datei voll funktionsfähig!

Ein großer Teil der Datei beschäftigt sich mit den ladbaren Modulen. Module sind Programmteile, die der Apache bei Bedarf nachladen kann. Diese Module können auch von anderen Programmierern stammen, sie müssen sich nur an die Spezifikationen halten, die die *Apache HTTP Server Group* dafür veröffentlicht hat. Diese Offenheit und Erweiterbarkeit ist Grundlage für den enormen Erfolg des Apache Web-Servers.

`/etc/httpd/httpd.conf` (Auszug: Laden von Modulen):

```
#
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which
# was built as a DSO you
```

```
# have to place corresponding 'LoadModule'
# lines at this location so the
# directives contained in it are actually
# available _before_ they are used.
# Please read the file README.DSO in the
# Apache 1.3 distribution for more
# details about the DSO mechanism and run
# 'httpd -l' for the list of already
# built-in (statically linked and thus always
# available) modules in your httpd
# binary.
#
# Note: The order is which modules are loaded
# is important.  Don't change
# the order below without expert advice.
#
# Example:
# LoadModule foo_module libexec/mod_foo.so
LoadModule mmap_static_module
➥ /usr/lib/apache/mod_mmap_static.so
LoadModule vhost_alias_module
➥ /usr/lib/apache/mod_vhost_alias.so
LoadModule env_module         /usr/lib/apache/mod_env.so
LoadModule define_module      /usr/lib/apache/mod_define.so
LoadModule config_log_module
➥ /usr/lib/apache/mod_log_config.so
LoadModule agent_log_module   /usr/lib/apache/mod_log_agent.so
LoadModule referer_log_module
➥ /usr/lib/apache/mod_log_referer.so
LoadModule mime_magic_module
➥ /usr/lib/apache/mod_mime_magic.so
LoadModule mime_module        /usr/lib/apache/mod_mime.so
LoadModule negotiation_module
➥ /usr/lib/apache/mod_negotiation.so
LoadModule status_module      /usr/lib/apache/mod_status.so
LoadModule info_module        /usr/lib/apache/mod_info.so
LoadModule includes_module    /usr/lib/apache/mod_include.so
LoadModule autoindex_module   /usr/lib/apache/mod_autoindex.so
LoadModule dir_module         /usr/lib/apache/mod_dir.so
LoadModule cgi_module         /usr/lib/apache/mod_cgi.so
LoadModule asis_module        /usr/lib/apache/mod_asis.so
LoadModule imap_module        /usr/lib/apache/mod_imap.so
```

```
LoadModule action_module      /usr/lib/apache/mod_actions.so
LoadModule speling_module     /usr/lib/apache/mod_speling.so
LoadModule userdir_module     /usr/lib/apache/mod_userdir.so
LoadModule alias_module       /usr/lib/apache/mod_alias.so
LoadModule rewrite_module     /usr/lib/apache/mod_rewrite.so
LoadModule access_module      /usr/lib/apache/mod_access.so
LoadModule auth_module        /usr/lib/apache/mod_auth.so
LoadModule anon_auth_module   /usr/lib/apache/mod_auth_anon.so
LoadModule dbm_auth_module    /usr/lib/apache/mod_auth_dbm.so
LoadModule db_auth_module     /usr/lib/apache/mod_auth_db.so
LoadModule digest_module      /usr/lib/apache/mod_digest.so
LoadModule proxy_module       /usr/lib/apache/libproxy.so
LoadModule cern_meta_module   /usr/lib/apache/mod_cern_meta.so
LoadModule expires_module     /usr/lib/apache/mod_expires.so
LoadModule headers_module     /usr/lib/apache/mod_headers.so
LoadModule usertrack_module   /usr/lib/apache/mod_usertrack.so
LoadModule example_module     /usr/lib/apache/mod_example.so
LoadModule unique_id_module   /usr/lib/apache/mod_unique_id.so
LoadModule setenvif_module    /usr/lib/apache/mod_setenvif.so
<IfDefine SSL>
LoadModule ssl_module         /usr/lib/apache/libssl.so
</IfDefine>
```

Dieser Teil der Konfigurationsdatei beschäftigt sich mit dem Laden der Module, hierbei muss dem Apache auch der Dateiname angegeben werden. Hervorgehoben ist hier der Abschnitt für das SSL-Modul, das Sie noch kennenlernen werden.

Der nächste Abschnitt der Konfigurationsdatei aktiviert die bereits geladenen Module.

`/etc/httpd/httpd.conf` (Auszug: Aktivieren von Modulen):

```
#  Reconstruction of the complete module
#  list from all available modules
#  (static and shared ones) to achieve correct
#  module execution order.
#  [WHENEVER YOU CHANGE THE LOADMODULE
#  SECTION ABOVE UPDATE THIS, TOO]
ClearModuleList
AddModule mod_mmap_static.c
AddModule mod_vhost_alias.c
AddModule mod_env.c
AddModule mod_define.c
```

```
AddModule mod_log_config.c
AddModule mod_log_agent.c
AddModule mod_log_referer.c
AddModule mod_mime_magic.c
AddModule mod_mime.c
AddModule mod_negotiation.c
AddModule mod_status.c
AddModule mod_info.c
AddModule mod_include.c
AddModule mod_autoindex.c
AddModule mod_dir.c
AddModule mod_cgi.c
AddModule mod_asis.c
AddModule mod_imap.c
AddModule mod_actions.c
AddModule mod_speling.c
AddModule mod_userdir.c
AddModule mod_alias.c
AddModule mod_rewrite.c
AddModule mod_access.c
AddModule mod_auth.c
AddModule mod_auth_anon.c
AddModule mod_auth_dbm.c
AddModule mod_auth_db.c
AddModule mod_digest.c
AddModule mod_proxy.c
AddModule mod_cern_meta.c
AddModule mod_expires.c
AddModule mod_headers.c
AddModule mod_usertrack.c
AddModule mod_example.c
AddModule mod_unique_id.c
AddModule mod_so.c
AddModule mod_setenvif.c
<IfDefine SSL>
AddModule mod_ssl.c
</IfDefine>
```

Die Funktionen des Apache kann man durch Programmteile erweitern, die er nur bei Bedarf lädt. Module können auch von anderen Programmierern erstellt und in den Apache eingebunden werden. Dazu muss man das Programm nicht einmal neu kompilieren, es genügt, das Modul zu laden (`LoadModule`) und zu aktivieren (`AddModule`).

Einige Module werden nur bedingt geladen:

```
<IfDefine SSL>
LoadModule ssl_module         /usr/lib/apache/libssl.so
</IfDefine>
```

bewirkt, dass Apache das Modul `ssl_module` nur dann lädt, wenn dies ein Startparameter verlangt. Das für die verschlüsselte Übertragung zuständige Modul `ssl_module` fehlt in der Standardinstallation; Sie sollten es nachinstallieren.

```
#
# If you wish httpd to run as a different
# user or group, you must run httpd
# as root initially and it will switch.
#
# User/Group: The name (or #number) of the
# user/group to run httpd as.
#  . On SCO (ODT 3) use "User nouser" and "Group nogroup".
#  . On HPUX you may not be able to use shared
#    memory as     nobody, and the
#    suggested workaround is to create a
#    user www and use that user.
#  NOTE that some kernels refuse to
#  setgid(Group) or semctl(IPC_SET)
#  when the value of (unsigned)Group is above 60000;
#  don't use Group nogroup on these systems!
#
User wwwrun
Group nogroup
```

Um den Linux-Server, auf dem der Web-Server läuft, zu schützen, verwendet der Web-Server den Benutzernamen `wwwrun` und die Gruppe `nogroup`, die beide mit wenig Rechten verbunden sind.

```
#
# ServerAdmin: Your address, where problems
# with the server should be
# e-mailed.  This address appears on
# some server-generated pages, such
# as error documents.
#
ServerAdmin root@localhost
```

Diese Einstellung ist sehr allgemein, die Mail an diese Adresse wird aber sicher zugestellt. Wer möchte, kann hier seine eigene Mailadresse angeben. Da

der Apache diese Adresse bei Fehlermeldungen ausgibt, sollte die Adresse einen Bezug zum lokalen System besitzen. Üblich ist eine Angabe wie `webmaster@lokales-netz.de`.

```
#
# ServerName allows you to set a host name
# which is sent back to clients for
# your server if it's different than the
# one the program would get (i.e., use
# "www" instead of the host's real name).
#
# Note: You cannot just invent host names and
# hope they work. The name you
# define here must be a valid DNS name
# for your host. If you don't understand
# this, ask your network administrator.
# If your host doesn't have a registered
# DNS name, enter its IP address here.
# You will have to access it by its address
# (e.g., http://123.45.67.89/)
# anyway, and this will make redirections
# work in a sensible way.
#
#ServerName besson.suse.de
ServerName 192.168.1.2
```

Gibt man keinen Namen an, benutzt Apache den lokalen Rechnernamen, wenn der Server Fehlermeldungen an den Browser übermittelt, hier im Beispiel also `boss.lokales-netz.de`. Will man lieber `www.lokales-netz.de` übermitteln, so muss man das hier angeben. Man darf aber nur Namen benutzen, die der Server auch korrekt auflösen kann. Hinweise zur Namensauflösung finden sich im Kapitel über den Domain Name Server. Solange noch kein Name-Server läuft, sollten Sie hier erst einmal die IP des Linux-Servers eintragen, auf dem der Web-Server läuft.

```
#
# DocumentRoot: The directory out of which you
# will serve your documents. By default, all requests
# are taken from this directory, but
# symbolic links and aliases may be
# used to point to other locations.
#
DocumentRoot "/usr/local/httpd/htdocs"
```

Normalerweise braucht man diese Einstellung nicht zu ändern. Im angegebenen Verzeichnis befinden sich die Seiten, die der Web-Server anbieten kann.

Für jedes über das Web zugängliche Verzeichnis kann man Parameter einstellen. Diese vererbt Apache an Unterverzeichnisse, sofern es für diese Unterverzeichnisse nicht neue Angaben gibt.

```
#
# Each directory to which Apache has access,
# can be configured with respect
# to which services and features are
# allowed and/or disabled in that
# directory (and its subdirectories).
#
# First, we configure the "default"
# to be a very restrictive set of
# permissions.
#
<Directory />
    AuthUserFile  /etc/httpd/passwd
    AuthGroupFile /etc/httpd/group

    Options -FollowSymLinks
    AllowOverride None
</Directory>
```

Da dies das höchste Verzeichnis ist, beschränkt man hier massiv Rechte. Die Einschränkungen kann man in den einzelne Unterverzeichnissen wieder aufheben. Die Option `-FollowSymLinks` verbietet dem Apache, symbolischen Links zu folgen. Symbolische Links würden sonst auch einen Zugriff auf das gesamte Dateisystem ermöglichen. Die Zeile `AllowOverride None` bewirkt, dass Benutzer die Einstellungen nicht durch Angaben in einer Datei `.htaccess` im jeweiligen Verzeichnis ändern dürfen. In einer derartigen Datei kann man alle Optionen, die für Verzeichnisse angegeben werden können, überschreiben, wenn dies durch `AllowOverride All` erlaubt wird.

Ein Teil dieser Einschränkungen wird für das `htdocs`-Verzeichnis gleich wieder überschrieben.

```
#
# Note that from this point forward
# you must specifically allow
# particular features to be
# enabled - so if something's not working as
```

```
# you might expect, make sure that
# you have specifically enabled it
# below.
#

#
# This should be changed to whatever you set DocumentRoot to.
#
<Directory "/usr/local/httpd/htdocs">

#
# This may also be "None", "All",
# or any combination of "Indexes", "Includes",
# "FollowSymLinks", "ExecCGI", or "MultiViews".
#
# Note that "MultiViews" must be
# named *explicitly* --- "Options All"
# doesn't give it to you.
#
    Options Indexes -FollowSymLinks +Includes MultiViews

#
# This controls which options the
# .htaccess files in directories can
# override. Can also be "All", or any combination
# of "Options", "FileInfo",
# "AuthConfig", and "Limit"
#
    AllowOverride None

#
# Controls who can get stuff from this server.
#
    Order allow,deny
    Allow from all


        #
        # don´t use DAV without access control !!
        #
        <IfDefine DAV>
```

```
        DAV On
        </IfDefine>

</Directory>
```

Die Option `Options Indexes -FollowSymLinks +Includes MultiViews` bewirkt, dass Apache für Ordner ohne Standard-Datei (z.B. `index.htm` s.u.) ein Inhaltsverzeichnis erzeugt. Symbolische Links sind immer noch verboten, erlaubt sind aber die *Server Side Includes* (SSI), spezielle Programmbefehle, die man in HTML-Seiten integrieren kann.

Welche Rechner Zugriff auf das Verzeichnis haben, legt man folgendermaßen fest.

```
Order allow,deny
Allow from all
```

Zuerst bestimmt eine Regel die Reihenfolge des Erlaubens und Ablehnens. Konkret haben hier Regeln der Art `allow` Vorrang vor Regeln der Art `deny`. Als einzige Regel folgt dann eine `allow`-Regel, die den Zugriff für alle Rechner freigibt. Wollte man nur den Rechnern der eigenen Domäne einen Zugriff erlauben, so wäre das hier im Beispiel möglich mit

```
Order deny,allow
Deny from all
Allow from .lokales-netz.de
```

Eine Zeile sollte man in der Konfiguration noch erweitern. In der Vorlage steht:

```
#
# DirectoryIndex: Name of the file or files
# to use as a pre-written HTML directory index.
# Separate multiple entries with spaces.
#
<IfModule mod_dir.c>
    DirectoryIndex index.html
</IfModule>
```

Dies bewirkt, dass man bei Startseiten den Dateinamen weglassen darf. Die Eingabe der URL `http://192.168.1.2/` ist gleichbedeutend mit `http://192.168.1.2/index.html`. Um alle üblichen Standardfälle zu berücksichtigen, sollte man diese Zeile erweitern zu:

```
#
# DirectoryIndex: Name of the file or files
# to use as a pre-written HTML directory index.
# Separate multiple entries with spaces.
```

```
#
<IfModule mod_dir.c>
    DirectoryIndex index.html index.htm welcome.html
    ➥ welcome.htm
</IfModule>
```

Die Reihenfolge dieser Aufzählung entscheidet auch über den Vorrang. Wenn sowohl eine Datei `index.html`, als auch eine Datei `welcome.htm` existieren, dann überträgt Apache die Datei `index.html`.

In der Standard-Startseite der Apache-Installation findet man z.B. als ersten Link `http://192.168.1.2/hilfe/`. Ein Verzeichnis `hilfe` gibt es aber nicht unterhalb von `/usr/local/httpd/htdocs`. Dass der Link trotzdem funktioniert, hängt mit dem nächsten Abschnitt zusammen, in dem Verweise auf Verzeichnisse vorgenommen werden, die außerhalb des Apache-Verzeichnisses liegen können.

```
Alias /howto/       /usr/doc/howto/
Alias /hilfe/       /usr/doc/susehilf/
Alias /doc/         /usr/doc/
Alias /cgi-bin-sdb/ /usr/local/httpd/cgi-bin/
Alias /sdb/         /usr/doc/sdb/
Alias /manual/      /usr/doc/packages/apache/manual/
Alias /htdig/       /opt/www/htdocs/htdig/
Alias /opt/kde/share/doc/HTML/     /opt/kde/share/doc/HTML/
Alias /opt/gnome/share/gnome/help/
/opt/gnome/share/gnome/help/

<Directory /usr/doc>
    Options FollowSymLinks Indexes +Includes
    AllowOverride None
</Directory>

<Directory /opt/kde/share/doc/HTML/>
    Options FollowSymLinks Indexes
    AllowOverride None
</Directory>

<Directory /opt/gnome/share/gnome/help/>
    Options FollowSymLinks Indexes
    AllowOverride None
</Directory>
#
```

```
# Aliases: Add here as many aliases as you need
# (with no limit). The format is
# Alias fakename realname
#
<IfModule mod_alias.c>

    #
    # Note that if you include a trailing / on
    # fakename then the server will
    # require it to be present in the URL.
    # So "/icons" isn't aliased in this
    # example, only "/icons/"..
    #
    Alias /icons/ "/usr/local/httpd/icons/"

    <Directory "/usr/local/httpd/icons">
        Options Indexes MultiViews
        AllowOverride None
        Order allow,deny
        Allow from all
    </Directory>
```

Der Apache ordnet virtuellen Namen, hier `/hilfe/` reale Dateien bzw. Verzeichnisse zu, hier `/usr/doc/susehilf/`. Der virtuelle Name heißt `Alias`. Der Aufruf von `http://192.168.1.2/hilfe/` greift also nicht auf `/usr/local/httpd/htdocs/hilfe/` zu, sondern auf `/usr/doc/susehilf/`. Von dieser praktischen Einrichtung macht SuSE intensiv Gebrauch, wie Sie an obigem Auszug der Konfigurationsdatei erkennen können.

Mit

```
<Directory /usr/doc>
    Options FollowSymLinks Indexes +Includes
    AllowOverride None
</Directory>
```

erreicht SuSE, dass der Apache auch symbolischen Links innerhalb der Dokumentation folgt.

Ausführbare Programme (z.B. cgi-Scripte) sammelt man üblicherweise in dem speziellen Verzeichnis `/cgi-bin/`. Dieses Verzeichnis legt man nicht unterhalb von `htdocs` an, was einen Sicherheitsgewinn bringt. Benutzern, die nur Webseiten erstellen dürfen, kann man beispielsweise per FTP oder Samba einen Zugriff auf das `htdocs`-Verzeichnis erlauben, ohne dass sie Programme im `cgi-bin`-Verzeichnis ablegen können.

Für Verzeichnisse mit ausführbaren Programmen gibt es eine spezielle Alias-Direktive:

```
#
    # ScriptAlias: This controls which
    # directories contain server scripts.
    # ScriptAliases are essentially the
    # same as Aliases, except that
    # documents in the realname directory
    # are treated as applications and
    # run by the server when requested rather
    # than as documents sent to the client.
    # The same rules about trailing "/" apply
    # to ScriptAlias directives as to
    # Alias.
    #
    ScriptAlias /cgi-bin/ "/usr/local/httpd/cgi-bin/"

    #
    # "/usr/local/httpd/cgi-bin" should be
    # changed to whatever your ScriptAliased
    # CGI directory exists, if you have that configured.
    #
    <Directory "/usr/local/httpd/cgi-bin">
        AllowOverride None
        Options None
        Order allow,deny
        Allow from all
    </Directory>
```

Jedes ausführbare Programm in diesem Verzeichnis stellt ein potenzielles Sicherheitsrisiko dar. Sie sollten die Zugriffsberechtigung für das `cgi-bin`-Verzeichnis daher nur sehr zurückhaltend vergeben.

6.4 Web-Dokumente ordnen und aufspielen

Die Vorgehensweise für das Ordnen und Aufspielen von Webdokumenten hängt sehr von den individuellen Arbeits- und Organisationsformen ab. Bei der Verwaltung von Websites muss man drei Systeme unterscheiden:

- Zentralisiert
- Hierarchisch
- Chaotisch

Bei einer zentralisierten Web-Verwaltung hat im Extremfall nur ein einziger Mitarbeiter, der Webadministrator, schreibenden Zugriff auf die Seiten. Alle anderen Mitarbeiter müssen ihm Seiten zukommen lassen, er überprüft sie und bindet sie in das Gesamtangebot ein. Hier genügt es, wenn dieser Mitarbeiter das Verzeichnis `/usr/local/httpd/htdocs` per FTP (s.u.) bzw. Samba (s.u.) erreichen kann. Beim FTP-Zugriff gestattet man diesem Mitarbeiter entweder einen Zugriff auf das gesamte System, oder man legt sein Home-Verzeichnis nach `/usr/Local/httpd/htdocs`.

Bei einem hierarchischen System verwaltet ein Webadministrator die Startseite, alle weiteren Rubriken betreut jeweils ein anderer Mitarbeiter. Jeder Mitarbeiter bekommt dann ein Verzeichnis, dessen Inhalt er selbst zu verantworten hat, z.B. der Benutzer Meyer das Verzeichnis `speiseplan`. Der Webadministrator muss dann nur die Verweise auf die Startseiten dieser Verzeichnisse anlegen.

Für die Zugriffe auf diese individuellen Verzeichnisse benutzt man beispielsweise das Alias-System des Apache. Jeder Benutzer legt ein Verzeichnis `html` in sein Home-Verzeichnis, worauf der Administrator ein Alias setzt. Für unser Beispiel also:

```
Alias /speiseplan/ /home/meyer/html/
```

Der Zugriff auf die URL `http://192.168.1.2/speiseplan/` landet dann im Home-Verzeichnis des Benutzers Meyer. Auf dieses Verzeichnis hat er bei den hier im Buch beschriebenen Installationen von FTP und Samba vollen Zugriff.

Am aufwendigsten ist die chaotische Verwaltung zu regeln, da hier jeder Benutzer vollen Zugriff auf alle Dokumente des Web-Servers hat. Dazu muss das gesamte `htdocs` Verzeichnis per FTP oder Samba erreichbar sein.

Für Samba ist eine spezielle Freigabe `www` auf dieses Verzeichnis die einfachste Lösung. Beim FTP-Zugriff verzichtet man entweder auf die sicherere *Changed-Root-Umgebung* (siehe FTP, Kapitel 7), oder man legt das `htdocs`-Verzeichnis einfach unterhalb von `/home` an, indem man den Eintrag `DocumentRoot` in der Apache-Konfigurationsdatei verschiebt:

```
DocumentRoot "/home/wwwhome/htdocs"
```

Dies ist ein auf vielen Web-Servern übliches Verfahren. Man muss bei der Veränderung etwas aufpassen, da man alle Pfade in der `/etc/httpd/httpd.conf` anpassen muss, die bisher mit `/usr/local/httpd/htdocs` anfangen.

6.5 Zugriffssteuerung für geschlossene Nutzergruppen

Auf vielen Web-Servern (nicht nur auf unanständigen) gibt es Bereiche, die man nur dann betreten kann, wenn man über einen dafür gültigen Benutzernamen und ein Passwort verfügt.

Wenn man z.B. unterhalb der URL `http://192.168.1.2/protokolle/` vertrauliche Protokolle ablegen will, muss man dem Apache mitteilen, dass er die Berechtigung für Zugriffe auf dieses Verzeichnis überprüfen soll.

Dazu fügen Sie eine weitere Directory-Direktive ein:

```
<Directory /usr/local/httpd/htdocs/protokolle>
 authName Geheim-Protokolle
 authType Basic
 authuserFile /etc/httpd/protokolle.pwd
 require valid-user
</Directory>
```

Die erste Zeile legt einen String fest, den Apache den Benutzern im Eingabefenster für das Passwort anzeigt. Die zweite Zeile legt die Art der Autorisation fest. Üblich ist hier der Typ `Basic`, da nicht alle Browser den Typ `Digest` unterstützen, der die Benutzerdaten verschlüsselt zwischen Client und Server überträgt. Die dritte Zeile legt fest, wo die Datei mit den Benutzernamen und Passwörtern liegen soll. Die letzten Zeile gibt an, dass alle Benutzer, die sich anmelden können, einen Zugriff bekommen. Die möglichen Einstellungen hier sind `user`, `group` und `valid-user`. Würde man hier im Beispiel angeben:

```
require user meyer
```

so bekämen andere Benutzer keinen Zugriff, auch wenn sich ihr Benutzername und Passwort in der angegebenen Passwortdatei wiederfindet. Neben dem `authuserFile` könnte man auch noch ein `authgroupFile` angeben, um gruppenbezogene Zugriffe zu erlauben.

Tipp: Die Benutzer, Gruppen und Passwörter haben nichts mit denen des Linux-Systems zu tun. Der Apache-Benutzername kann und sollte vom Linux-Benutzernamen abweichen.

Bevor Sie die neue Konfiguration testen können, müssen Sie noch die in der Konfiguration angegebene Passwortdatei erzeugen und mindestens einen Benutzer einrichten.

Das Programm `/usr/bin/htpasswd` erzeugt und verändert die Passwortdatei. Man erzeugt mit

```
/usr/bin/htpasswd -c /etc/httpd/protokolle.pwd meyer
```

eine neue Passwortdatei mit einem Benutzer `meyer` und muss dann zweimal sein Passwort angeben. Der Schalter `-c` (für *create*) erzeugt die Datei beim ersten Aufruf und muss bei weiteren Eingaben entfallen.

Nach einem Neustart des Apache mit

```
/sbin/init.d/apache restart
```

können Sie einen ersten Zugriff auf den Ordner ausprobieren, indem Sie die URL `http://192.168.1.2/protokolle/` in einen Browser eingeben. In einem Fenster sehen Sie dann einen Dialog zur Eingabe von Benutzernamen und Passwort.

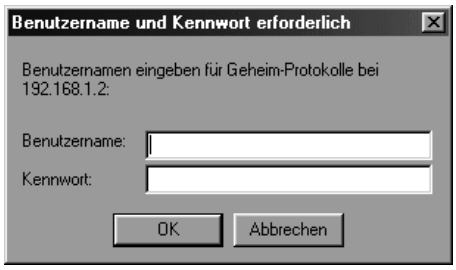

Abbildung 6.2: Authentizierung

Das genaue Aussehen dieses Fensters hängt vom Client-Betriebssystem und dem Browser ab.

Nach einem erfolgreichen Aufruf müssten Sie nun das leere Inhaltsverzeichnis des Verzeichnisses sehen. Bei einer Fehlermeldung finden Sie die Ursache in der Datei `/var/log/httpd.error_log` auf dem Server.

Einträge in der Passwortdatei löscht man mit einem Texteditor, nicht mit `htpasswd`. Die Zeile für den Benutzer `meyer`, den Sie eben eingerichtet haben, sieht in der Datei folgendermaßen aus:

`/etc/httpd/protokolle.pwd`

```
meyer:gvHI6UCjbEtk6
```

In der ersten Spalte, vor dem Doppelpunkt, steht der Benutzername, danach das verschlüsselte Passwort. Löschen Sie diese Zeile, so nehmen Sie dem Benutzer die Zugriffsrechte auf den Ordner wieder weg.

Zum Anlegen der Gruppendateien benötigt man ebenfalls einen Texteditor.

`/etc/httpd/protokolle.grp`

```
autoren: adams, tikart, meyer
koerner: roggen, gerste, hirse
```

Links vom Doppelpunkt steht der Name der Gruppe, rechts davon die Mitgliederliste.

Mit der Gruppenzugehörigkeit und der Möglichkeit, unabhängige Passwort- und Gruppendateien für jedes Verzeichnis anzulegen, kann man die Zugriffsrechte sehr genau regeln.

6.6 Virtuelle Server

Provider wie Strato bieten Homepages für tausende von Kunden auf dem gleichen Server an. All diese Homepages werden vom gleichen Web-Server bedient, der dazu neben seiner IP-Adresse auch die vielen verschiedenen Web-Adressen kennen muss, auf die er reagiert. Für jede Web-Adresse benutzt der virtuelle Server ein anderes Home-Verzeichnis.

Der Apache bietet dieses Feature unter der Bezeichnung `VirtualHosts`, *virtuelle Rechner* an.

Praktisch umsetzen können Sie die Konfiguration aus diesem Abschnitt nur, wenn Sie bereits einen Name-Server installiert haben.

Betreiben Sie neben dem normalen Web-Server `http://www.lokales-netz.de` einen Server `http://www2.lokales-netz.de`, so können Sie diesen so konfigurieren, dass er das Unterverzeichnis `Protokolle` aus dem vorangegangenen Beispiel als Home-Verzeichnis anzeigt. Dazu müssen Sie die Konfigurationsdatei wie folgt ändern, wobei es z.T. schon Vorgaben von SuSE gibt:

```
#
# If you want to use name-based virtual
# hosts you need to define at
# least one IP address (and port number) for them.
#
#NameVirtualHost 12.34.56.78:80
NameVirtualHost 192.168.1.2
```

Beim Arbeiten mit virtuellen Hosts möchte der Apache die zugehörige IP wissen, da es auch möglich wäre, die Hosts auf verschiedene Adressen reagieren zu lassen.

Die folgenden Zeilen finden Sie als Beispiel in der Konfigurationsdatei:

```
#
# VirtualHost example:
# Almost any Apache directive may go into a
# VirtualHost container.
#
#<VirtualHost ip.address.of.host.some_domain.com>
#    ServerAdmin webmaster@host.some_domain.com
#    DocumentRoot /www/docs/host.some_domain.com
#    ServerName host.some_domain.com
#    ErrorLog logs/host.some_domain.com-error_log
#    CustomLog logs/host.some_domain.com-access_log common
#</VirtualHost>
```

Den neuen virtuellen Server mit dem Wurzelverzeichnis `/usr/local/httpd/htdocs/protokolle` definieren Sie folgendermaßen:

```
<VirtualHost www2>
  ServerName www2.lokales-netz.de
  DocumentRoot /usr/local/httpd/htdocs/protokolle
</VirtualHost>
```

Den bisherigen Standardserver muss man jetzt auch noch einmal definieren. Auch dieser ist jetzt nur noch ein virtueller Host. Zusätzlich muss man auch für den Fall, dass eine Anfrage nicht über `www` oder `www2` auf den Server zukommt regeln. Das betrifft z.B. Anfragen direkt über die IP; auch hierfür muss ein virtueller Host definiert sein. Machen Sie das gleich für alle Möglichkeiten, indem Sie eine `default`-Definition für den WWW-Port 80 angeben:

```
<VirtualHost _default_:80>
</VirtualHost>
```

Über virtuelle Hosts kann man das eigene Webangebot benutzerspezifisch strukturieren, oder für mehrere Firmen bzw. Abteilungen Angebote auf einem einzigen Server hosten. Je nach angesprochenem Web-Server bietet der Apache verschiedene Zugänge an.

Wenn Sie Veränderungen an der Konfigurationsdatei vorgenommen haben, dann müssen Sie den Apache neu starten, damit er diese Änderungen übernimmt:

```
/sbin/init.d/apache restart
```

6.7 Gesicherte Zugriffe mit Secure Sockets Layer (SSL)

Beim bisher besprochenen Zugriffsschutz mit Benutzernamen und Passwort schickt der Browser die Daten unverschlüsselt über das Netz.

Zum Übermitteln vertraulicher Informationen, sollte man Daten verschlüsselt übertragen. Das von Netscape entwickelte System basiert auf dem SSL-Protokoll, das auch für andere Dienste, z.B. Telnet verwendbar ist.

Zum Nutzen dieses Protokolls benötigt man das Apache-Modul `mod_ssl`, das SuSE-Install nicht standardmäßig einbindet.

Installieren Sie dieses Modul einfach nach, es findet sich im Paket `mod_ssl` der Serie `sec`. Nach der Installation dieses Paketes müssen Sie den Apache neu starten, um dieses Modul mit einzubinden; SuSE hat ansonsten schon alles für Sie vorbereitet.

Zwei Konfigurationsschritte bleiben noch:

1. Man muss die Apache-Konfiguration so erweitern, dass er auf dem Port 443 gesicherte Verbindungen aufbaut und
2. ein Zertifikat erzeugen, mit dem sich der Linux-Server gegenüber dem Browser ausweist.

Da SuSE schon ziemlich viel vorbereitet hat, braucht man die Einstellungen nur an die eigenen Bedingungen anzupassen und zu aktivieren. Sie finden folgende Einstellungen vor:

`/etc/httpd/httpd.conf` (Auszug ab Zeile 1370)

```
<IfDefine SSL>

##
## SSL Virtual Host Context
##

<VirtualHost _default_:443>

#  General setup for the virtual host
DocumentRoot "/usr/local/httpd/htdocs"
ServerName besson.suse.de
ServerAdmin root@besson.suse.de
ErrorLog /var/log/httpd/error_log
TransferLog /var/log/httpd/access_log

#   SSL Engine Switch:
#   Enable/Disable SSL for this virtual host.
```

```
SSLEngine off

#   SSL Cipher Suite:
#   List the ciphers that the client is
#   permitted to negotiate.
#   See the mod_ssl documentation for a complete list.
#SSLCipherSuite
```

Diesen Abschnitt wertet Apache nur dann aus, wenn er mit dem Parameter zum Einbinden des SSL-Modules startet. Den notwendigen Parameter übergibt das Startscript `/sbin/init.d/apache` automatisch, wenn es das Modul auf der Festplatte vorfindet.

Sie definieren für SSL-Verbindungen einen eigenen Server (Virtual Host). Der Standardport für https ist 443, diese Einstellung sollte man nicht verändern.

```
##
## SSL Virtual Host Context
##

<VirtualHost _default_:443>
```

Sie sollten für diesen Server einen eigenen Verzeichnisbaum aufbauen, hier `ssldocs`. Die Vorlage von SuSE legt den Server auch in den Verzeichnisbaum `htdocs`. Es ist jedoch ungünstig, dass gesicherter und ungesicherter Server dann im gleichen Verzeichnis liegen. Die restlichen Einstellungen überschreiben die Grundeinstellungen für diesen Server. Die Log-Dateien können identisch sein mit denen für den normalen Server; darin besteht kein Sicherheitsrisiko.

```
#  General setup for the virtual host
  DocumentRoot "/usr/local/httpd/ssldocs"
  ServerName 192.168.1.2
  ServerAdmin root@192.168.1.2
  ErrorLog /var/log/httpd/error_log
  TransferLog /var/log/httpd/access_log
```

Die Einstellung ist wichtig. Nur wenn `SSLEngine` auf `on` steht, aktiviert der Apache SSL wirklich. Die Vorgabe von SuSE ist `off`.

```
#   SSL Engine Switch:
#   Enable/Disable SSL for this virtual host.
SSLEngine on
```

Nun folgen bis zum Dateiende noch ein paar Einstellungen und Pfade für SSL, die man nicht zu ändern braucht.

```
#   SSL Cipher Suite:
#   List the ciphers that the client is
#   permitted to negotiate.
#   See the mod_ssl documentation for a complete list.
#SSLCipherSuite
   ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL
...
```

SSL übertragt dann Login und Daten verschlüsselt. Mit dem Schlüssel stellt der Browser sicher, dass er mit dem echten Server verbunden ist und nicht etwa mit einem Rechner, der sich nur für den echten Rechner ausgibt. Dazu muss man auf dem Server ein Schlüsselzertifikat erzeugen, dass man wiederum von einer anerkannten Zertifizierungsstelle (Certification Authority, CA) signieren lassen sollte.

Browser erkennen einige bekannte Zertifizierungsstellen automatisch an. Da das Signieren eines Zertifikates meistens mit Kosten verbunden ist, lesen Sie hier eine Gratis-Lösung für eine Testinstallation:

Benutzen Sie für Tests als Zertifizierungsinstanz die fiktive Firma *Snake Oil;* die notwendigen Daten dieser Firma gehören zum SSL-Modul dazu. Ein Nachteil besteht darin, dass Browser die Zertifikate dieser Firma nicht automatisch anerkennen.

Zum Erzeugen der Zertifikate wechseln Sie in das Verzeichnis `/usr/doc/packages/mod_ssl` und starten das Programm

```
 ./certificate.sh
```

das dann die notwendigen Angaben erfragt. Eigene Eingaben sind hier fett hervorgehoben.

```
SSL Certificate Generation Utility (mkcert.sh)
Copyright (c) 1998 Ralf S. Engelschall, All Rights Reserved.

Generating test certificate signed by Snake Oil CA [TEST]
WARNING: Do not use this for real-life/production systems
_____________________________________________________________________

STEP 0: Decide the signature algorithm used for certificate
The generated X.509 CA certificate can contain either
RSA or DSA based ingredients. Select the one you want to use.
Signature Algorithm ((R)SA or (D)SA) [R]:R
_____________________________________________________________________
```

```
STEP 1: Generating RSA private key (1024 bit) [server.key]
488077 semi-random bytes loaded
Generating RSA private key, 1024 bit long modulus
 ......................+++++
........+++++
e is 65537 (0x10001)
```

```
STEP 2: Generating X.509 certificate signing request
  [server.csr]
Using configuration from .mkcert.cfg
You are about to be asked to enter information that
➥ will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished
  Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
1. Country Name             (2 letter code) [XY]:DE
2. State or Province Name   (full name)     [Snake
   ➥ Desert]:Germany
3. Locality Name            (eg, city)      [Snake
   ➥ Town]:Hamburg
4. Organization Name        (eg, company)   [Snake Oil,
   ➥ Ltd]:lokales-netz
5. Organizational Unit Name (eg, section)   [Webserver
   ➥ Team]:Webteam
6. Common Name              (eg, FQDN)
   ➥ [www.snakeoil.dom]:www.lokales-netz.de
7. Email Address  (eg, name@FQDN)
   ➥ [www@snakeoil.dom]:root@lokales-netz.de
```

```
STEP 3: Generating X.509 certificate signed by Snake Oil CA
        ➥ [server.crt]
Certificate Version (1 or 3) [3]:3
Signature ok
subject=/C=DE/ST=Germany/L=Hamburg/O=lokales-
➥ netz/OU=Webteam/CN=www.lokales-netz/Email=root@lokales-
➥ netz.de
```

```
Getting CA Private Key
Verify: matching certificate & key modulus
read RSA private key
Verify: matching certificate signature
/etc/httpd/ssl.crt/server.crt: OK
_____________________________________________________________________

STEP 4: Enrypting RSA private key with a
➥ pass phrase for security [server.key]
The contents of the server.key file
➥ (the generated private key) has to be
kept secret. So we strongly recommend you to encrypt the
➥ server.key file
with a Triple-DES cipher and a Pass Phrase.
Encrypt the private key now? [Y/n]: n


Warning, you're using an unencrypted RSA private key.
Please notice this fact and do this on your own risk.
_____________________________________________________________________


RESULT: Server Certification Files

o  conf/ssl.key/server.key
   The PEM-encoded RSA private key file which you configure
   with the 'SSLCertificateKeyFile' directive
   ➥ (automatically done
   when you install via APACI). KEEP THIS FILE PRIVATE!

o  conf/ssl.crt/server.crt
   The PEM-encoded X.509 certificate file which you configure
   with the 'SSLCertificateFile' directive (automatically done
   when you install via APACI).

o  conf/ssl.csr/server.csr
   The PEM-encoded X.509 certificate signing request
   ➥ file which
   you can send to an official Certificate Authority
   ➥ (CA) in order
   to request a real server certificate (signed by
   ➥ this CA instead
```

```
   of our demonstration-only Snake Oil CA) which later
   ➥ can replace
   the conf/ssl.crt/server.crt file.

WARNING: Do not use this for real-life/production systems
```

Dies erzeugt ein Serverzertifikat, das die fiktive *Snake Oil CA* zertifiziert. Nach einem Neustart von Apache können Sie den Zugriff testen.

Bei einem Aufruf von `https://192.168.1.2` fragt der Netscape Communicator nach, ob Sie das unbekannte Zertifikat annehmen wollen. Wenn Sie sieben mal `weiter` geklickt haben, dann können Sie die Startseite des SSL-Servers sehen.

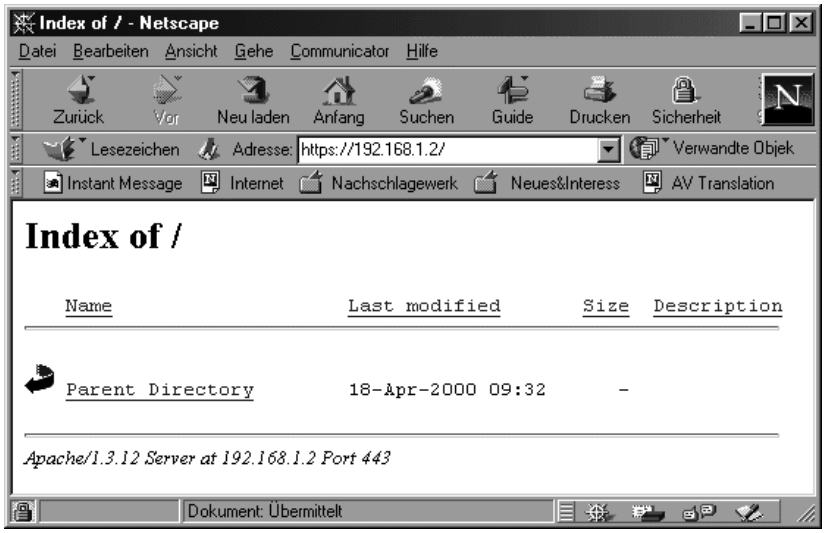

Abbildung 6.3: Sichere Verbindung im Netscape

Anzeichen für eine gesicherte Verbindung sind die beiden hervorgehobenen Schlösser, das eine in der linken unteren Ecke, das andere in der Iconleiste neben dem Drucker.

Beim Internet Explorer muss man nur dreimal auf *Ja* klicken, um die entsprechende Seite anzuzeigen.

Das so erstellte Zertifikat ist nur für ein Jahr gültig. Wer mehr über das Zertifikat erfahren möchte, sollte seinen Netscape-Browser neu starten. Hier im Beispiel haben Sie das Zertifikat bisher nur für die aktuelle Sitzung angenommen. Auf der zweiten Seite, beim Akzeptieren des Zertifikates gibt es einen

Knopf *Mehr Info* ... Klickt man diesen an, so kann man Details des Zertifikates sehen.

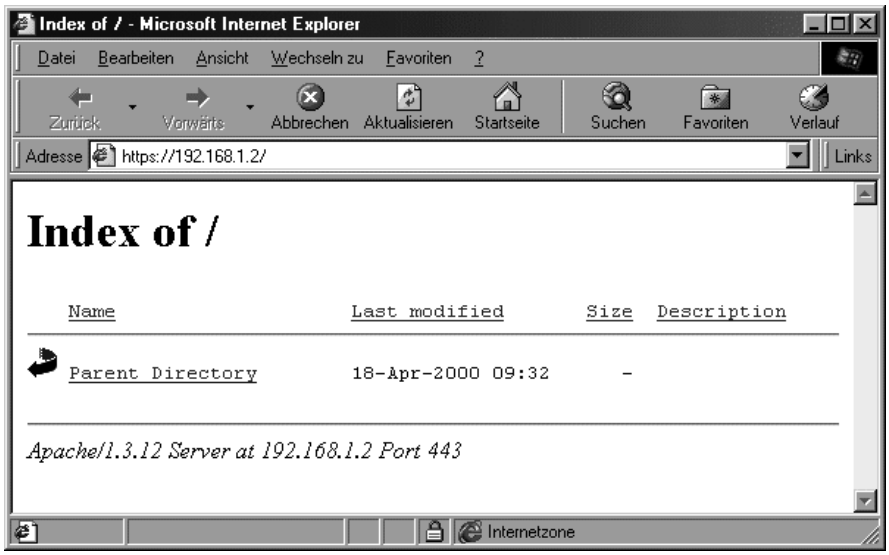

Abbildung 6.4: Sichere Verbindung im Internet Explorer

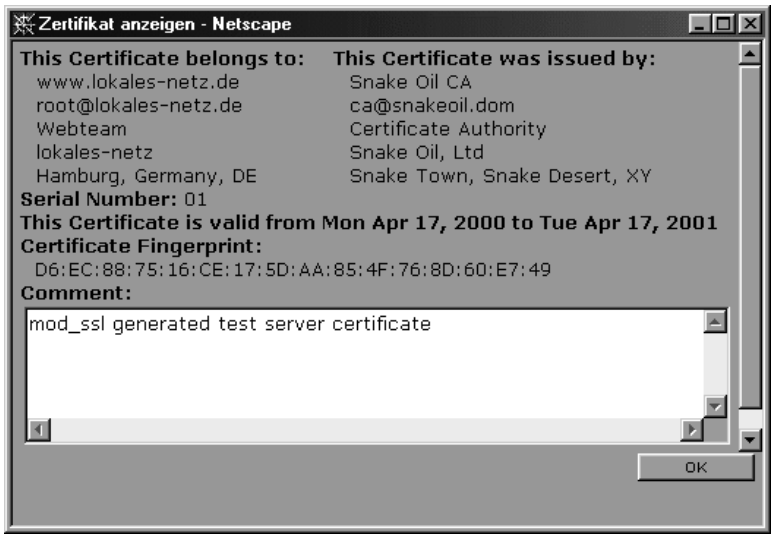

Abbildung 6.5: Das Zertifikat

Auch im Internet Explorer kann man sich Details über das Zertifikat ansehen, bevor man es annimmt.

Abbildung 6.6: Das Zertifikat im Internet Explorer

Auf der dritten Dialogseite kann man anwählen, wie lange der Communicator das Zertifikat akzeptieren soll. In der Voreinstellung ist das Zertifikat nur für die aktuelle Sitzung gültig. Wenn man mit dem erzeugten Zertifikat zufrieden ist, dann kann man es ruhig auch unbefristet annehmen. Dann erscheint der Dialog erst nach einem Jahr wieder, wenn Sie das Zertifikat erneuert haben.

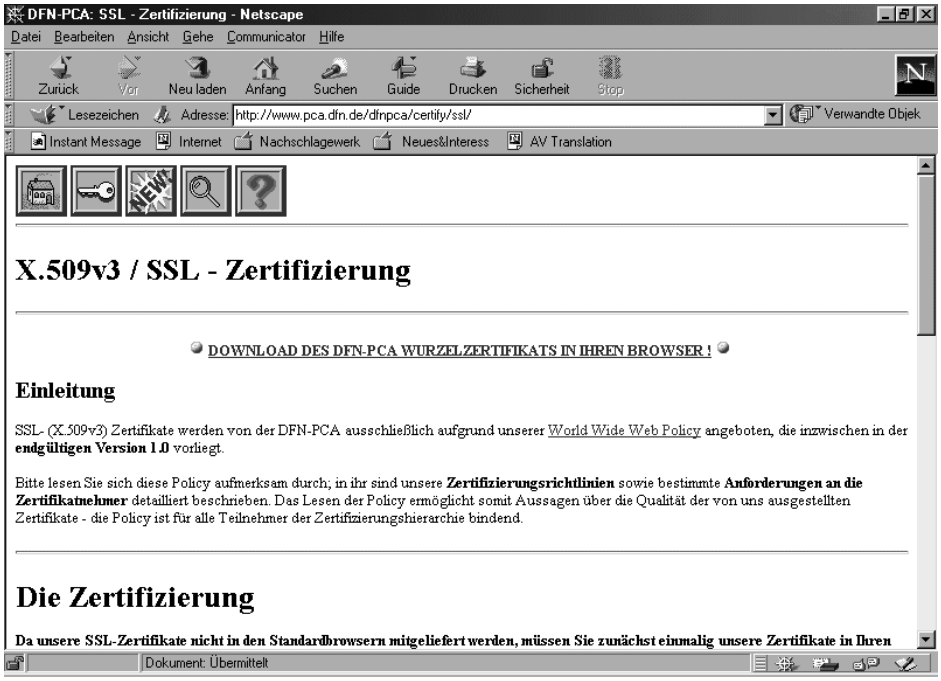

Abbildung 6.7: SSL-Zertifizierung beim DFN

Speziell dieser Teil des Kapitels erhebt keinen Anspruch auf Vollständigkeit. Das Thema Zertifikate ist sehr umfangreich; immerhin sind hier staatliche und wirtschaftliche Interessen berührt. Die beschriebene Konfiguration führt nur zu einem funktionsfähigen Test-System.

Deutsche Zertifizierungsstellen für SSL sind im Aufbau, zu den bereits aktiven Organisationen gehört der DFN-Verein, dessen SSL-Informationen Sie unter `http://www.pca.dfn.de/dfnpca/certify/ssl/` finden.

6.8 Zugriffe protokollieren und auswerten

Apache protokolliert alle Zugriffe in der Datei `/var/log/httpd/access_log`. Geben Sie im Browser die URL `http://192.168.1.2` ein, so trägt er folgendes in die Log-Datei ein:

```
192.168.1.40 - - [18/Apr/2000:09:58:32 +0200] "GET
➥ /gif/suse_150.gif HTTP/1.1" 200 2874
192.168.1.40 - - [18/Apr/2000:09:58:32 +0200] "GET
➥ /gif/penguin.gif HTTP/1.1" 200 51793
192.168.1.40 - - [18/Apr/2000:09:58:33 +0200] "GET / HTTP/1.1"
➥ 200 4745
192.168.1.40 - - [18/Apr/2000:09:58:33 +0200] "GET
➥ /gif/suse_button.gif HTTP/1.1" 200 2101
192.168.1.40 - - [18/Apr/2000:09:58:33 +0200] "GET
➥ /gif/apache_pb.gif HTTP/1.1" 200 2326
```

Die erste Zeile dieser Einträge ist folgendermaßen zu lesen:

Eintrag	*Bedeutung*
`192.168.1.40`	IP-Nummer des Client-Rechners, hier ein Rechner aus dem lokalen Netz.
`18/Apr/200:09:58.32 +0200`	Datum und Uhrzeit. Da im April in Deutschland die Sommerzeit gilt, weicht die Zeit um +2 Stunden von der Standardzeit (GMT) ab.
`"GET /gif/suse_150.gif HTTP/1.1"`	Die Datei `/usr/local/httpd/htdocs/gif/suse_150.gif` wird mit dem Protokoll `HTTP 1.1` übertragen.
`200`	Die Datei wurde erfolgreich übertragen.
`2874`	Größe der übertragenen Datei in Bytes.

Tabelle 6.2: Erklärung der Einträge in der Datei /var/log/httpd/access_log

Bei einer fehlerhaften Anfrage wie `http://192.168.1.2/nichtda.htm` schreibt der Apache folgende Meldung in die `access_log`:

```
192.168.1.40 - - [18/Apr/2000:10:07:12 +0200] "GET
➥ /nichtda.htm HTTP/1.1" 404 285
```

Statt des Codes `200` für eine erfolgreiche Datenübertragung taucht hier `404` für `File does not exist` auf.

Der Inhalt der Logdatei ist sehr aussagekräftig und auch für statistische Auswertungen zu nutzen.

Fehler protokolliert der Apache zusätzlich in der Datei `/var/log/httpd/error_log`. Nach der fehlerhaften Anfrage hat sie folgenden Inhalt:

```
[Tue Apr 18 09:53:39 2000] [notice] Apache/1.3.12 (Unix)
➥ (SuSE/Linux) PHP/3.0.15 mod_ssl/2.6.2 OpenSSL/0.9.5
➥ configured -- resuming normal operations
[Tue Apr 18 09:53:39 2000] [notice] suEXEC mechanism enabled
➥ (wrapper: /usr/sbin/suexec)
[Tue Apr 18 10:07:12 2000] [error] [client 192.168.1.40] File
➥ does not exist: /usr/local/httpd/htdocs/nichtda.htm
```

Die ersten Zeilen hat der Apache beim Start erstellt. Hier können Sie u.a. erkennen, dass das SSL-Modul erfolgreich geladen wurde.

In der letzten Zeile finden Sie die Fehlermeldung als Folge einer fehlerhaften Anforderung.

Tipp: Wenn Sie beginnen, eigene CGI-Programme zu erstellen, sollten Sie dieser Datei gebührende Beachtung schenken, da nur hier die Fehlermeldungen Ihrer Programme auftauchen.