

6 Informationen per Web-Server verteilen

Immer mehr Einrichtungen nutzen Web-Server, um Informationen im Intranet, Extranet und Internet bereitzustellen. Dies beeinflusst die gesamte Kommunikationskultur erheblich.

Grundlagen des Web sind:

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

Alle marktführenden Linux-Distributionen enthalten einen Web-Server, meist den Apache. Da SuSE ihn bei der Standardinstallation nicht automatisch installiert, holen Sie das am besten schnell nach.

Apache hat 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 Windows NT weiter.

Dieses Kapitel beschreibt die Grundlagen, 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 die Datenübertragung verschlüsseln.

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 (6.4),
- man eine Zugriffssteuerung für geschlossene Nutzergruppen einrichtet (6.5),
- was virtuelle Server sind (6.6),
- wie man gesicherte Zugriffe mit Secure Sockets Layer (SSL) einrichtet (6.7),
- wie man Web-Server-Zugriffe protokollieren (6.8),
- die Protokolldatei des Web-Servers grafisch aufbereiten (6.9) und
- eine eigene Suchmaschine einrichten (6.10) kann.

6.1 Wann brauchen Sie einen eigenen Web-Server?

Einen eigenen Web-Server brauchen Sie eigentlich immer. Statt Informationen auf einem schwarzen Brett in der Kantine auszuhängen oder Kunden per Mailing zu informieren, kann man besser Seiten für den lokalen Web-Server erstellen und dort aktuelle Ankündigungen und Termine hinterlegen. Wichtig ist, Inhalte regelmäßig zu pflegen und zu aktualisieren.

Hierzu verwendet man am besten Content Management Systeme. Ein freies Content Management System ist das Programm Midgard, das SuSE mit ausliefert. Die Web-Site des Midgard-Projektes an der Adresse <http://www.midgard-project.org/> beschreibt das Programm ausführlich.

Beim Entwickeln von Web-Auftritten können jedem leicht Fehler unterlaufen. Peinlicherweise sind diese bei über das Internet zugänglichen Seiten weltweit sichtbar. Den eigenen Auftritt sollte man daher 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, der Web-Browser, eine Anfrage nach einem Dokument an den Server, den http-Dämon. Dieser liefert dem Client den MIME-Typ der angeforderten Datei und die Datei selbst. Aus dem MIME-Typ schließt der Client, 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 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, deutet dies auf ein Problem mit der Datei `/etc/httpd/mime.types`.

Für jede laufende Verbindung ist ein `httpd`-Prozess zuständig. Der WWW-Server startet 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) keinen Speicherplatz für Prozesse, weil alle Kopien des WWW-Servers den Speicher gemeinsam nutzen.

6.3 Web-Server Apache installieren und einrichten

Aktuell gibt es zwei unterschiedliche Versionen des Apache. Die bisherige Version und den vollkommen neu entwickelten Apache2. Da bisher leider noch nicht alle Erweiterungs-Module für den neuen Apache 2.0 angepasst sind, beschreibt diese Auflage die altbewährte Version.

SuSE legt den Apache in die Selektion `Einfacher Webserver` ins Paket `apache`, bzw. auf dem FTP-Server oder CD2 in die entsprechende `rpm`-Datei. Da SuSE den Web-Server in der Standardinstallation nicht installiert, holen Sie dies einfach gemäß der Anleitung im Kapitel 2.5 dieses Buchs nach.

Nach der Installation müssen Sie den Web-Server noch mit dem Befehl

```
rcapache start
```

starten.

Überzeugen Sie sich, dass der Web-Server lauffähig ist, indem Sie von einem Client aus seine URL, hier im Beispiel `http://192.168.1.2`, aufrufen. Der Browser müsste folgende recht leere Startseite anzeigen:

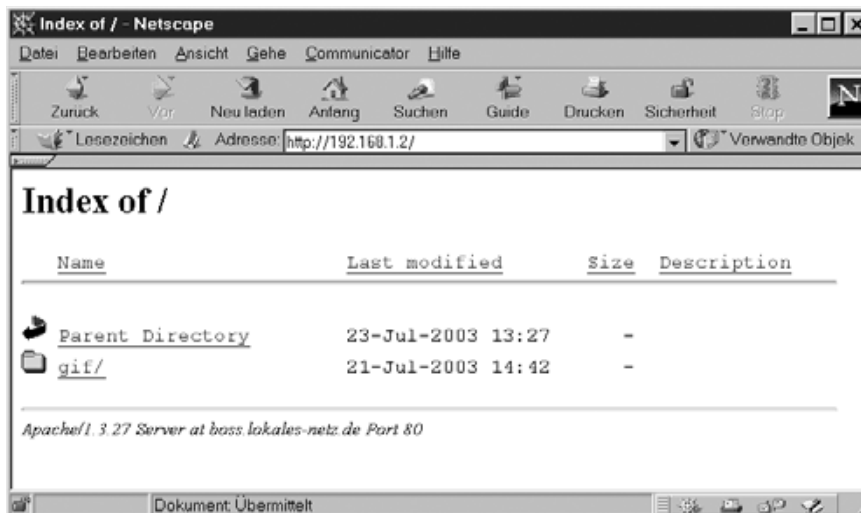


Abbildung 6.1: Standardstartseite im Browser

Um die Beispielseiten der bisherigen Versionen zu nutzen, müssen Sie auch das Paket *apache-example-pages* installieren, das Sie in der Selektion *Einfacher Web-server* oder direkt auf *CD2* finden.

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

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

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

Heute braucht man zum Einrichten des Apache nur noch die Datei `httpd.conf`; und nicht mehr wie früher die Dateien `httpd.conf`, `access.conf` und `srm.conf`, die aus Kompatibilitätsgründen noch als leere Dateien vorhanden sind.

SuSE hat gegenüber den vorherigen Versionen das Wurzelverzeichnis des Web-Servers verschoben. Statt unterhalb von `/usr/local/httpd/` finden Sie die Webdokumente jetzt unterhalb von `/srv/www/`.

Die mehr als 1600 Zeilen lange Konfigurationsdatei `/etc/httpd/httpd.conf` hat SuSE recht gut kommentiert. Der Apache ist bereits ohne Änderungen an der Datei voll funktionsfähig! Der folgende Text erläutert wichtige Abschnitte der Konfigurationsdatei, die für eine normale Nutzung bzw. das grundlegende Verständnis wichtig sind.

Tipp: Bearbeiten Sie die Konfigurationsdatei möglichst nie direkt, sondern nur mit YaST. Individuelle Veränderungen tragen Sie in zusätzliche Konfigurationsdateien ein und binden sie mit YaST per Include-Anweisung ein.

Ein großer Teil der Datei beschäftigt sich mit den ladbaren Modulen. Dies sind Programmteile, die der Apache bei Bedarf nachladen kann. Diese Module, die auch von Programmierern außerhalb des Apache-Teams stammen können, müssen sich an die Spezifikationen der *Apache HTTP Server Group* halten. Diese Offenheit und Erweiterbarkeit hat den enormen Erfolg des Apache Web-Servers mit begründet.

Einen Eindruck von der Vielzahl der Erweiterungsmöglichkeiten bietet der folgende Ausschnitt aus der Konfigurationsdatei.

`/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 http://httpd.apache.org/docs/
# dso.html 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 in which modules are loaded is important.
# Don't change the order below without expert advice.
```

```

# Note:
#
# The file that is included after the LoadModule statements is
# generated by SuSEconfig according to
#
# 1) which modules (ones not included with apache) are installed
# 2) the settings in /etc/sysconfig/apache
#
# SuSEconfig uses the /etc/httpd/modules/
# * files that come with each module
# to determine the necessary directives.
#
# Apache no longer needs to be started with
# '-D <modules>' switches (with
# the exception of mod_ssl, which has a lot of conditional
# statements).

# 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
# mod_userdir will be included below by SuSEconfig if
# HTTPD_SEC_PUBLIC_HTML=yes
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 unique_id_module   /usr/lib/apache/mod_unique_id.so
LoadModule setenvif_module      /usr/lib/apache/mod_setenvif.so
<IfDefine DUMMYSSL>
LoadModule ssl_module           /usr/lib/apache/libssl.so
</IfDefine>

Include /etc/httpd/suse_loadmodule.conf

```

Dieser Teil der Konfigurationsdatei beschäftigt sich mit dem Laden der Module, hierbei muss man dem Apache den Dateinamen des Moduls angeben.

Hervorgehoben ist hier der Abschnitt für das SSL-Modul, das Sie noch kennen lernen werden.

Über die Zeile *Include /etc/httpd/suse_loadmodule.conf* bindet SuSE eine eigene Konfigurationsdatei ein, die weitere Module laden kann. Diese Konfigurationsdatei verwalten Yast bzw. SuSEconfig.

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
# mod_userdir will be included below by SuSEconfig if
# HTTPD_SEC_PUBLIC_HTML=yes
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_unique_id.c
AddModule mod_so.c
AddModule mod_setenvif.c
<IfDefine DUMMYSSL>
AddModule mod_ssl.c
</IfDefine>

# Again, the following file is generated by SuSEconfig for
# modules that actually have been installed

Include /etc/httpd/suse_addmodule.conf

```

Auch hier bindet SuSE eine zusätzliche Konfigurationsdatei ein, die YaST verwaltet.

Die Funktionen des Apache kann man durch Programmteile erweitern, die er nur bei Bedarf lädt. Wie oben schon erwähnt, kann man von Fremd-Programmierern erstellte Module in den Apache einbinden. Dazu muss man solche Programme nicht einmal neu kompilieren, es genügt, das Modul zu laden (LoadModule) und zu aktivieren (AddModule).

Einige Module lädt die Konfiguration nur bedingt:

```
<IfDefine DUMMYSSL>
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 möglichst bald nachinstallieren (6.7), um auch gesicherte Verbindungen anbieten zu können.

Im nächsten Abschnitt legen Sie den Benutzernamen und die Gruppe für den Apache fest.

```
#
# 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 wenigen Rechten verbunden sind. Dadurch verhindern Sie z. B., dass der Web-Server auf fremde Dateien zugreifen kann.

Im nächsten Abschnitt geben Sie die Mailadresse des Administrators an.

```
#
# ServerAdmin: Your address, where problems with the server
# should be e-mailed.
# This address appears on some server-generated # pages, such
# as error documents.
#
# Note: this email address is set by SuSEconfig according to
# the setting of the HTTPD_SEC_SERVERADMIN variable in
# /etc/sysconfig/apache!
ServerAdmin webmaster@boss.lokales-netz.de
```

Diese von YaST erzeugte Einstellung ist sehr allgemein, die Mail an diese Adresse wird aber sicher zugestellt. Passen Sie hier den Namen des Netzwerks an und tragen Sie ggf. Ihre persönliche Mailadresse ein. Da der Apache diese Adresse bei Fehlermeldungen ausgibt, sollte sie einen Bezug zum lokalen System besitzen. Üblich ist eine Angabe wie `webmaster@boss.lokales-netz.de`.

Um die Vorgabe zu ändern, müssen Sie im YaST-Kontrollzentrum unter *System • Editor für /etc/sysconfig-Daten • Network • WWW • Apache* unter `HTTPD_SEC_SERVERADMIN` den gewünschten Wert angeben.

Im Abschnitt Virtuelle Server (6.6) lesen Sie hier, dass der Apache mit mehreren Adressen gleichzeitig arbeiten kann. Daher können Sie ihm angeben, mit welchem Namen er sich gegenüber dem Klienten melden soll. Auch hier hat YaST bereits einen Vorschlag eingetragen:

```
#
# 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.
# 127.0.0.1 is the TCP/IP local loop-back address, often named
# localhost. Your machine always knows itself by this address.
# If you use Apache strictly for local testing and development,
# you may use 127.0.0.1 as the server name.
#
# Note: the host name is set by SuSEconfig according to
# the setting of the FQHOSTNAME variable in /etc/sysconfig/
# network/config!
ServerName boss.lokales-netz.de
```

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`. Wollte man lieber `www.lokales-netz.de` übermitteln, so müsste man das hier direkt ändern. Man darf aber nur Namen benutzen, die der Server auch korrekt auflösen kann. Hinweise zur Namensauflösung finden Sie im Kapitel 15 über den Domain Name Server. Solange auf Ihrem Linux-Server noch kein Nameserver läuft, sollten Sie hier zunächst die Vorgabe belassen.

Sie müssen dem Apache auch mitteilen, wo er seine Webseiten findet.

```
#
# 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 "/srv/www/htdocs"
```

Gegenüber vorherigen Versionen von SuSE Linux ist dies eine Veränderung; bisher lagen die Webseiten unterhalb von `/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 +Multiviews
    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 für Verzeichnisse überschreiben, wenn `AllowOverride All` dies erlaubt.

Einen Teil dieser Einschränkungen überschreiben Sie für das `htdocs`-Verzeichnis gleich wieder.

```
#
# 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 "/srv/www/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

#
# disable WebDAV by default for security reasons.
#
<IfModule mod_dav.c>
    DAV Off
</IfModule>
```

```

        </IfModule>

#
# Enable SSI (Server Side Includes) for the demo index.html
# pages, as some of the content is created dynamically.
# This should be disabled when setting up a productive
# server.
<Files /srv/www/htdocs/index.htm*>
    Options -FollowSymLinks +Includes +MultiViews
</Files>

#
# Protect the php3 test page, so it cannot be viewed from an
# outside system.
#
<Files test.php3>
    Order deny,allow
    deny from all
    allow from localhost
</Files>

</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 auf das Verzeichnis zugreifen dürfen, legt man durch die Reihenfolge von Regeln und Einzel-Regeln fest:

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

Zuerst bestimmt eine Regel die Reihenfolge des Erlaubens und Ablehnens. Hier im ersten Beispiel haben 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 wie hier im zweiten Beispiel möglich mit

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

Sie können URLs verkürzen, wenn Sie Standards für die Namen der Startseite vorgeben. Üblich sind hier u. a. die Angaben `index.html` und `welcome.html`. Um hier etwas flexibler zu werden, können Sie eine Zeile 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 auch Startdateien wie `index.htm` und `welcome.htm` zu berücksichtigen, müssen Sie diese Zeile erweitern. Legen Sie eine Datei `/etc/httpd/linuxbuch.conf` mit folgendem Inhalt an:

```
#
# 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 über den Vorrang. Wenn sowohl eine Datei `index.html`, als auch eine Datei `welcome.htm` existieren, dann überträgt Apache die Datei `index.html`.

Zum Aktivieren dieser Änderung müssen Sie anschließend im YaST-Kontrollzentrum unter *System • Editor für /etc/sysconfig-Daten • Network • WWW • Apache* für die Variable `HTTPD_CONF_INCLUDE_FILES` den Wert `/etc/httpd/linuxbuch.conf` angeben.

Nach einem Neustart des Web-Servers mit

```
ncapache restart
```

sind diese Änderungen wirksam.

In der Standard-Installation des Apache funktioniert der Seitenaufruf `http://192.168.1.2/icons/`. Ein Verzeichnis `icons` gibt es aber nicht unterhalb von `/srv/www/htdocs`.

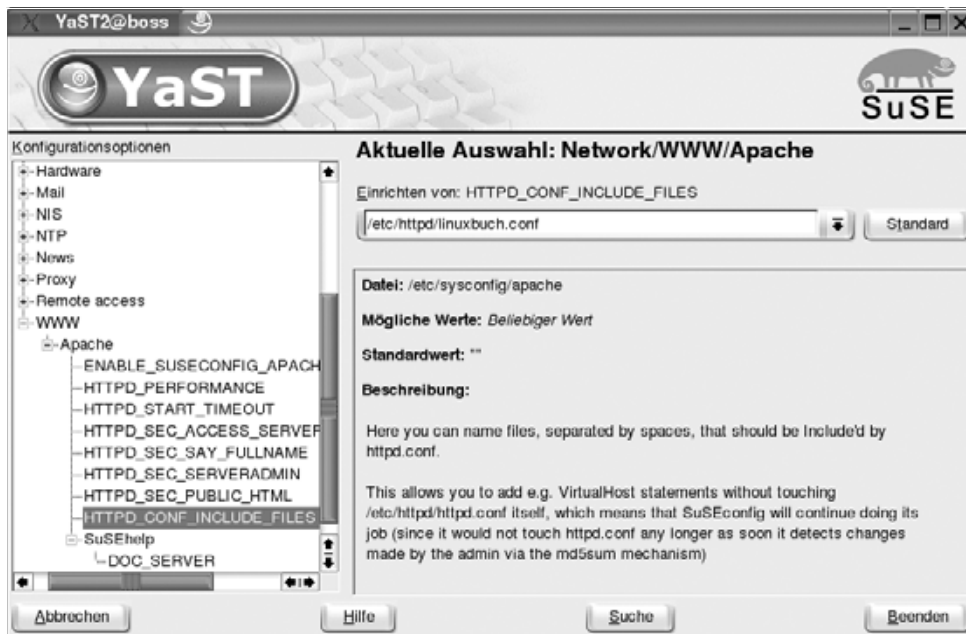


Abbildung 6.2: Eigene Konfigurationsdatei einbinden

Dass der Link trotzdem funktioniert, hängt mit den Einstellungen in der Datei `/etc/httpd/httpd.conf` zusammen.

```
<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/".If the fakename is slash-terminated, then the
# realname must also be slash terminated, and if the
# fakename omits the trailing slash, the realname must
# also omit it.
#
Alias /icons/ "/srv/www/icons/"

<Directory "/srv/www/icons">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

Der Apache ordnet virtuellen Namen, hier `/icons/`, reale Dateien bzw. Verzeichnisse zu, hier `/usr/local/httpd/icons/`. Der virtuelle Name heißt Alias. Der Aufruf von `http://192.168.1.2/icons/` greift also nicht auf `/usr/local/httpd/htdocs/icons/` zu, sondern auf `/usr/local/httpd/htdocs/`. Wie Sie diese praktische Einrichtung selber nutzen, lesen Sie im Abschnitt 6.4.

Ausführbare Programme (z. B. cgi-Skripte) sammelt man üblicherweise in dem speziellen Verzeichnis `/cgi-bin/`. Zur Verbesserung der Systemsicherheit legt man dieses Verzeichnis nicht unterhalb von `htdocs` an. 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 einen speziellen Alias-Befehl:

```
#
# 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/ "/srv/www/cgi-bin/"

<IfModule mod_perl.c>
# Provide two aliases to the same cgi-bin directory,
# to see the effects of the 2 different mod_perl modes.
# for Apache::Registry Mode
ScriptAlias /perl/      "/srv/www/cgi-bin/"
# for Apache::Perlrun Mode
ScriptAlias /cgi-perl/  "/srv/www/cgi-bin/"
</IfModule>
#
# "/srv/www/cgi-bin" should be changed to whatever
# your ScriptAliased CGI directory exists,
# if you have that configured.
#
<Directory "/srv/www/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>
```


Jedes ausführbare Programm in diesem Verzeichnis ist ein Sicherheitsrisiko. 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. Beim Verwalten von Web-Sites kann man in der Praxis drei Systeme beobachten:

- zentralisiert,
- hierarchisch und
- chaotisch.

Bei einer zentralisierten Web-Verwaltung hat im Extremfall nur ein einziger Mitarbeiter, der Webadministrator, Schreibzugriff 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 der Webadministrator das Verzeichnis `/srv/www/htdocs` per FTP (s. u.) bzw. Samba (s. u.) erreichen kann. Beim FTP-Zugriff gestattet man diesem Webadministrator entweder einen Zugriff auf das gesamte System oder man legt sein Home-Verzeichnis nach `/srv/www/htdocs`.

Bei einem hierarchischen System verwaltet ein Webadministrator die Startseite, alle weiteren Rubriken betreuen jeweils andere Mitarbeiter. Mitarbeiter bekommen ein Verzeichnis, dessen Inhalt sie selbst verantworten, z. B. die Benutzerin 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. Hierzu legen Benutzer ein Verzeichnis `html` in ihr Home-Verzeichnis. Der Administrator setzt ein Alias auf dieses Verzeichnis, hier im Beispiel in der Datei `/etc/httpd/linuxbuch.conf`:

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

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

Am aufwändigsten ist die chaotische Verwaltung zu regeln, bei der alle Benutzer vollen Zugriff auf alle Dokumente des Web-Servers haben. 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 `/srv/www/htdocs` anfangen.

6.5 Zugriffssteuerung für geschlossene Nutzergruppen

Auf vielen Web-Servern (nicht nur auf unanständigen) gibt es Bereiche, die man nur 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 muss man in der Datei `/etc/httpd/linuxbuch.conf` eine weitere `Directory`-Direktive einfügen:

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

Die erste Zeile legt den Text fest, den Apache den Benutzern im Eingabefenster für das Passwort anzeigt. Die zweite Zeile bestimmt die Art der Autorisierung. 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 liegt und die letzte Zeile regelt, 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 `authuser-`

File könnte man auch noch ein `authgroupFile` angeben, um gruppenbezogene Zugriffe zu erlauben.

Tip: Die Benutzer, Gruppen und Passwörter haben nichts mit denen des Linux-Systems zu tun. Die Apache-Benutzernamen sollten von Linux-Benutzernamen abweichen, da Benutzernamen unverschlüsselt über das Netz gehen, wenn man nicht mit gesicherten http-Verbindungen arbeitet.

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. Eine neue Passwortdatei mit einer Benutzerin `meyer` erzeugt man mit

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

Hier muss man dann zweimal ihr 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

```
rcapache 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.

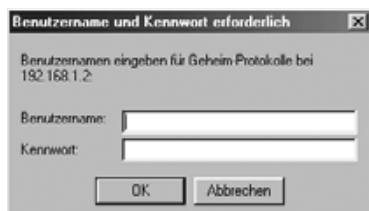


Abbildung 6.3: Authentizierung

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

Nach erfolgreichem Aufruf müssten Sie nun das bisher leere Inhaltsverzeichnis des Ordners sehen. Bei einer Fehlermeldung finden Sie die Fehlerursache in der Datei `/var/log/httpd/error_log` auf dem Server.

Einträge in der Passwortdatei löscht man mit einem Texteditor, nicht mit `htpasswd`, da dieses keine derartige Funktion kennt.

Die Zeile für die Benutzerin meyer, die Sie soeben eingerichtet haben, sieht in der Datei folgendermaßen aus:

```
/etc/httpd/protokolle.pwd  
meyer:gvHI6UCjbEtk6
```

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

Zum Anlegen der Gruppdateien 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 Mitglieder-liste.

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

Wenn aber sehr viele Benutzer auf diese Art den Zugriff auf die gesicherten Webseiten bekommen sollen, wie es z. B. bei den *Linuxbu.ch/Tools* (siehe Kapitel 3) der Fall ist, dann ist der Aufwand hoch.

In der aktuellen Version gibt es dazu ein Paket u. a. mit verschiedenen Modulen zur Authentifikation gegenüber dem Web-Server. Neben Modulen zur Nutzung von Datenbanken wie MySQL ist auch ein Modul zur Nutzung von PAM, dem normalen Anmeldesystem des Servers dabei. Mit diesem Modul können Sie die Benutzerdaten direkt aus den Systemdateien beziehen.

Installieren Sie dazu das Paket *apache-contrib* nach, am besten über YaST, da einige Abhängigkeiten vorliegen. Sie finden das Paket am einfachsten über die Suchfunktion der Software-Installation.

Nach der Installation des Paketes finden Sie im YaST-Kontrollzentrum unter *System • Editor für /etc/sysconfig-Daten • Network • WWW • Apache* mehrere neue Einstellmöglichkeiten.

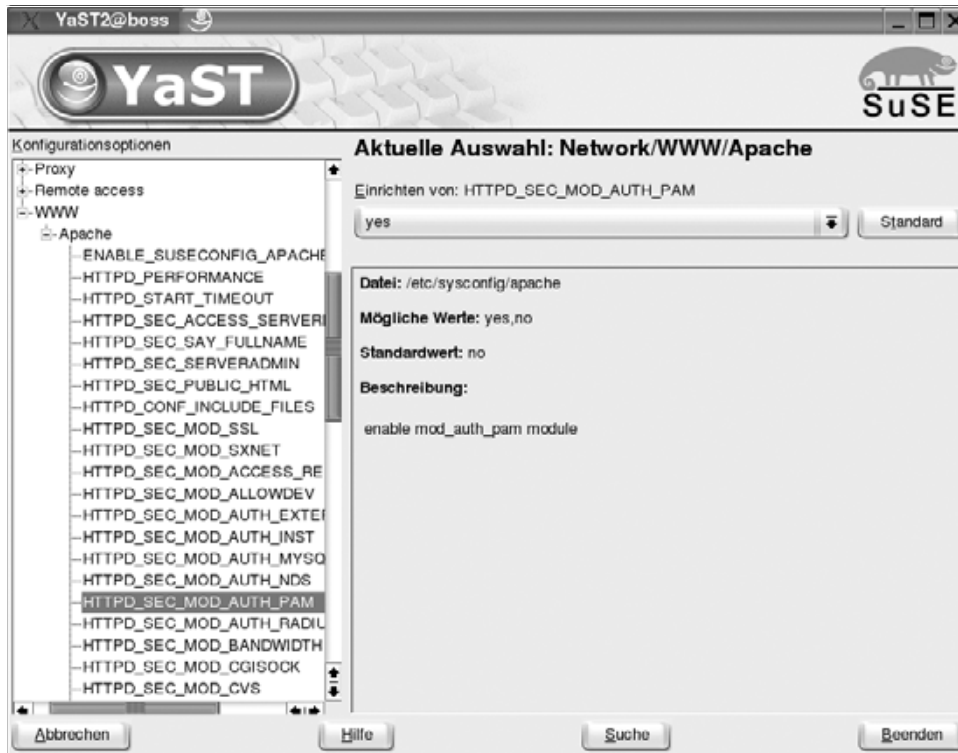


Abbildung 6.4: YaST-Kontrollzentrum

Setzen Sie hier `HTTPD_SEC_MOD_AUTH_PAM` auf `yes` um die Funktion zu aktivieren. Nun müssen Sie dem Web-Server noch den Zugriff auf die Passwörter in der Shadow-Datei erlauben.

```
usermod -G shadow wwwrun
```

nimmt den Benutzer `wwwrun` in die Gruppe `shadow` auf, deren Mitglieder berechtigt sind die Shadow-Datei zu lesen.

Nach einem Neustart des Web-Servers mit

```
rcapache restart
```

sind die neuen Einstellungen aktiv. Bei den Startmeldungen des Apache sollte jetzt das Modul mit aufgeführt werden.

```
Starting httpd [ auth_pam ]
```

Die zugehörigen Einstellungen in der Datei `/etc/httpd/linuxbuch.conf` können Sie nun etwas vereinfachen:

```
<Directory /srv/www/htdocs/protokolle>
  authName Geheim-Protokolle
  authType Basic
  require valid-user
</Directory>
```

Wenn kein *authuserFile* angegeben ist, dann benutzt der Apache das Modul *auth_pam* und entnimmt die Benutzerdaten aus den Systemdateien.

Nach den Änderungen an der *linuxbuch.conf* müssen Sie den Web-Server noch einmal neu starten. Nun hat jeder Ihrer System-Benutzer automatisch Zugriff auf die geschützten Webseiten.

Wenn Sie statt der Angabe

```
require valid-user
```

den Text

```
require group autoren
```

einsetzen, dann bekommen auf diese Art nur die Mitglieder der vorher definierten Gruppe *autoren* Zugriff auf die geschützten Seiten.

6.6 Virtuelle Server

Internet-Provider bieten Homepages für viele Kunden auf dem gleichen Web-Server an. All diese Web-Sites bedient der gleiche Web-Server, der nicht nur auf seine IP-Adresse sondern auch auf viele verschiedene Web-Adressen reagieren muss. Für jede Web-Adresse benutzt der virtuelle Server ein anderes Home-Verzeichnis.

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

Bevor Sie virtuelle Server konfigurieren, müssen Sie einen Nameserver installiert haben (siehe Kapitel 15).

Mehrere virtuelle Web-Server auf dem gleichen System können auch im lokalen Netz sinnvoll sein. Sie können damit inhaltliche Bereiche klar voneinander trennen.

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:

```
#
# Use name-based virtual hosting.
#
#NameVirtualHost *
NameVirtualHost 192.168.1.2
```

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

Benutzer mit dynamischen IP-Adressen konnten bei den früheren Apache-Versionen keine virtuellen Server einrichten, da sie keine feste IP für die Konfigurationsdatei angeben konnten. Bei der aktuellen Apache-Version können Sie statt der IP immer auch das Jokerzeichen `*` angeben, das dann für alle IP-Adressen steht.

```
#
# Use name-based virtual hosting.
#
NameVirtualHost *
```

Damit können Sie auch im Zusammenhang mit dynamischen IP-Adressen virtuelle Server einrichten.

Die folgenden Zeilen finden Sie als Beispiel in der Konfigurationsdatei:

```
#
# VirtualHost example:
# Almost any Apache directive may go into a VirtualHost
# container. The first VirtualHost section is used for
# requests without a known server name.
#
#<VirtualHost *>
#   ServerAdmin webmaster@dummy-host.example.com
#   DocumentRoot /www/docs/dummy-host.example.com
#   ServerName dummy-host.example.com
#   ErrorLog logs/dummy-host.example.com-error_log
#   CustomLog logs/dummy-host.example.com-access_log common
#</VirtualHost>
```

Den neuen virtuellen Server mit dem Wurzelverzeichnis `/srv/www/htdocs/protokolle` definieren Sie, indem Sie die Datei `/etc/httpd/linuxbuch.conf` folgendermaßen erweitern:

```
NameVirtualHost *

<VirtualHost *>
  ServerName www.lokales-netz.de
</VirtualHost>

<VirtualHost *>
  ServerName www2.lokales-netz.de
  DocumentRoot /srv/www/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 Anfragen regeln, die nicht über `www` oder `www2` auf den Server zukommen, sondern, z. B. direkt über die IP-Adresse; auch hierfür muss ein virtueller Host definiert sein. Alle denkbaren Möglichkeiten deckt eine `default`-Definition für den WWW-Port 80 ab:

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

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

Wenn Sie die Konfigurationsdatei verändert haben, müssen Sie Apache neu starten, damit er diese Änderungen übernimmt:

```
ncapache 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.

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

Das zum Nutzen dieses Protokolls benötigte Apache-Modul `mod_ssl` bindet die SuSE-Installation nicht per Voreinstellung ein.

Um die Sicherheit ihres Web-Angebots zu erhöhen, installieren Sie dieses Modul aus dem Paket `mod_ssl` der Selektion `Einfacher Webserver` nach. Nach der Installation dieses Pakets müssen Sie noch im YaST-Kontrollzentrum unter *System* • *Editor für /etc/sysconfig-Daten* • *Network* • *WWW* • *Apache* die Variable `HTTPD_SEC_MOD_SSL` auf `yes` setzen und dann den Apache neu starten, um das SSL-Modul endgültig einzubinden. Beim Start bzw. Neustart sollte der Apache das Modul und eventuell weitere installierte Module aufführen.

```
Starting httpd [ SSL ]
```

Zwei Konfigurationsschritte bleiben noch:

- Man muss die Apache-Konfiguration so erweitern, dass Apache auf dem Port 443 gesicherte Verbindungen aufbaut, und
- 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 1421)
###
### SSL Virtual Host Context
###

<VirtualHost _default_:443>

# General setup for the virtual host
DocumentRoot "/srv/www/htdocs"
ServerName new.host.name
ServerAdmin you@your.address
ErrorLog /var/log/httpd/error_log
TransferLog /var/log/httpd/access_log

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

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

Da Ihnen das SuSE-Startskript schon einen großen Teil der Konfigurationsarbeit abnimmt, müssen Sie nur noch virtuelle Server für den Apache definieren.

Sie definieren für SSL-Verbindungen einen eigenen Server (Virtual Host). Die Einstellung 443 für den Standardport für https, 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 riskant, wenn gesicherter und ungesicherter Server im gleichen Verzeichnis liegen, da das gesicherte Verzeichnis dann auch über den normalen Server erreichbar ist. Das SuSE hier in der `httpd.conf` eine konkrete Vorgabe gemacht hat, tragen Sie in der `httpd.conf` die `DocumentRoot` direkt ein.

Die restlichen Einstellungen überschreiben die Grundeinstellungen für diesen Server. Die Log-Dateien können mit denen für den normalen Server identisch sein; darin besteht kein Sicherheitsrisiko.

```
# General setup for the virtual host
DocumentRoot "/srv/www/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 für die `SSLEngine` ist wichtig. Nur wenn `SSLEngine` auf `on` steht, aktiviert der Apache wirklich SSL. Die Vorgabe von SuSE ist `on`.

```
# 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:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:
↵ +SSLv2:+EXP:+eNULL
...
```

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

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

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

Zum Erzeugen der Zertifikate wechseln Sie in das Verzeichnis `/usr/share/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: Encrypting 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, ob Sie das unbekannte Zertifikat annehmen wollen. Wenn Sie sieben mal weiter geklickt haben, können Sie die Startseite des SSL-Servers sehen.

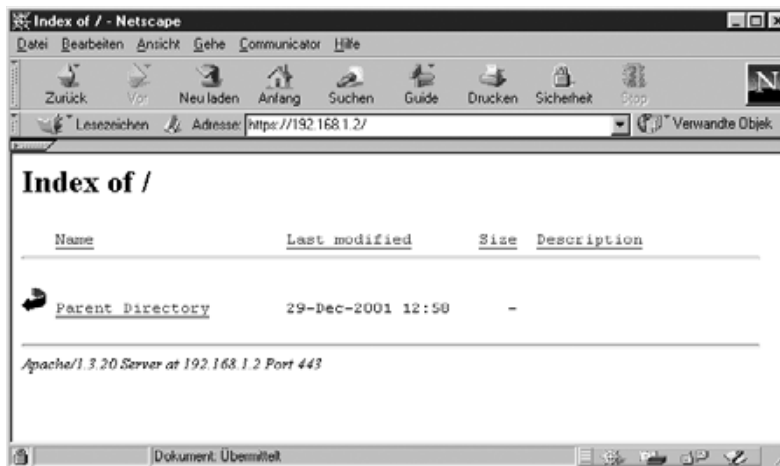


Abbildung 6.5: Sichere Verbindung im Netscape

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 das neue Zertifikat anzunehmen und die Startseite anzuzeigen.

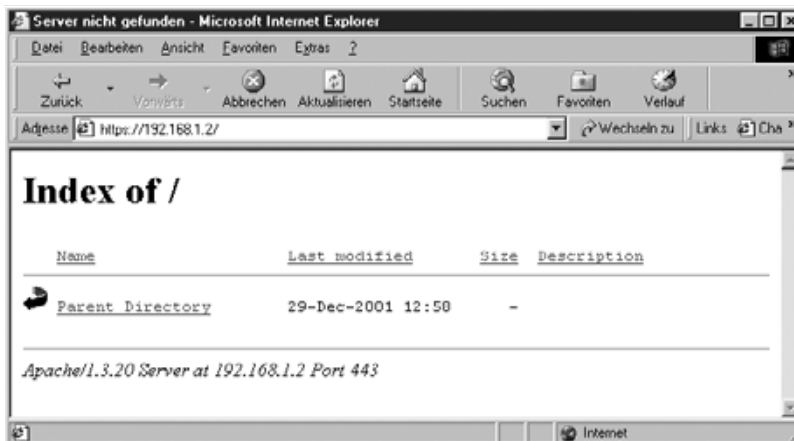


Abbildung 6.6: Sichere Verbindung im Internet Explorer

Das so erstellte Zertifikat ist nur für ein Jahr gültig. Wer mehr über das Zertifikat erfahren möchte, sollte seinen Browser neu starten. Hier im Beispiel ist das Zertifikat vom Netscape Browser 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, kann man Details des Zertifikates sehen.



Abbildung 6.7: as Zertifikat

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

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



Abbildung 6.8: Das Zertifikat im Internet Explorer

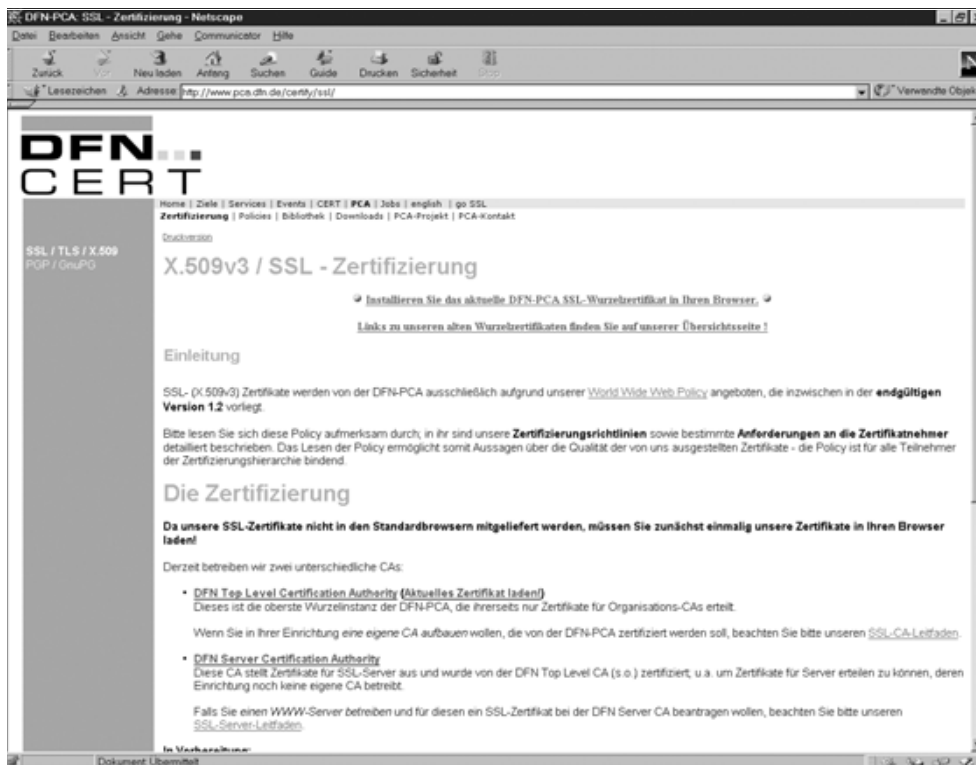


Abbildung 6.9: SSL-Zertifizierung beim DFN

Dieser Teil des Kapitels konnte nur die allerwichtigsten technischen Fragen zu Zertifikaten streifen und Ihnen helfen, ein funktionsfähiges Testsystem einzurichten. Für ein reales System brauchen Sie eine offizielle Zertifizierung. 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

Betreiber von Web-Sites möchten gern wissen, ob ihr Web-Server anständig funktioniert und was die Besucher auf der Web-Site treiben.

Apache protokolliert alle Zugriffe in der Datei `/var/log/httpd/access_log`. Geben Sie im Browser die URL `http://192.168.1.2` ein, hinter der sich eine Pinguin GIF-Datei versteckt, trägt Apache Folgendes in die Log-Datei ein:

```
192.168.1.56 - - [23/Jul/2003:15:58:05 +0200]
┆ "GET / HTTP/1.0" 200 873
192.168.1.56 - - [23/Jul/2003:15:58:06 +0200]
┆ "GET /icons/blank.gif HTTP/1.0" 200 148
192.168.1.56 - - [23/Jul/2003:15:58:06 +0200]
┆ "GET /icons/back.gif HTTP/1.0" 200 216
192.168.1.56 - - [23/Jul/2003:15:58:07 +0200]
┆ "GET /icons/folder.gif HTTP/1.0" 200 225
192.168.1.56 - - [23/Jul/2003:16:08:10 +0200]
┆ "GET / HTTP/1.0" 200 878
```

Die erste Zeile dieser Einträge ist folgendermaßen zu lesen:

<i>Eintrag</i>	<i>Bedeutung</i>
192.168.1.56	IP-Nummer des Client-Rechners, hier ein Rechner aus dem lokalen Netz.
23/Jul/2003:15:58:05 +0200	Datum und Uhrzeit. Da im Juli in Deutschland die Sommerzeit gilt, weicht die Zeit um +2 Stunden von der Standardzeit (GMT) ab.
"GET / HTTP/1.0"	Die Datei <code>/srv/www/htdocs/index.html</code> oder (falls nicht vorhanden) das Inhaltsverzeichnis wird mit dem Protokoll HTTP 1.0 übertragen.
200	Die Datei wurde erfolgreich übertragen.
873	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`:

```
192.168.1.56 - - [23/Jul/2003:15:59:05 +0200] "GET
└ /nichtda.htm HTTP/1.1" 404 294
```

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 gut für statistische Auswertungen nutzbar.

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

```
[Wed Jul 23 15:39:01 2003] [notice] Apache/1.3.27
└ (Linux/SuSE) mod_ssl/2.8.12 OpenSSL/0.9.6i configured
└ -- resuming normal operations
[Wed Jul 23 15:39:01 2003] [notice] suEXEC mechanism enabled
└ (wrapper: /usr/sbin/suexec)
[Wed Jul 23 15:39:01 2003] [notice] Accept mutex: sysvsem
└ (Default: sysvsem)
[23/Jul/2003:15:59:05 +0200] [error] [client 192.168.1.56]
└ 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.

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

6.9 Auswertung mit Webalizer

Wenn Ihnen die manuelle Auswertung der Log-Dateien nicht ausreicht, können Sie mit Analyse-Tools wesentlich übersichtlichere Statistiken erstellen.

Ein sehr weit verbreitetes Analyse-Tool ist das Programm Webalizer, das Sie auf CD4 im gleichnamigen Paket finden. Bei der Installation über YaST ist Webalizer nicht direkt über eine der Selektionen zu finden, benutzen Sie hier die Suchfunktion.

Das Programm liefert eine Übersicht über die Nutzung des Web-Servers in den letzten 12 Monaten.

Die Übersicht vergleicht die Monatsdaten. Die Summen und Durchschnittswerte beziehen sich auf einen einzelnen Tag.

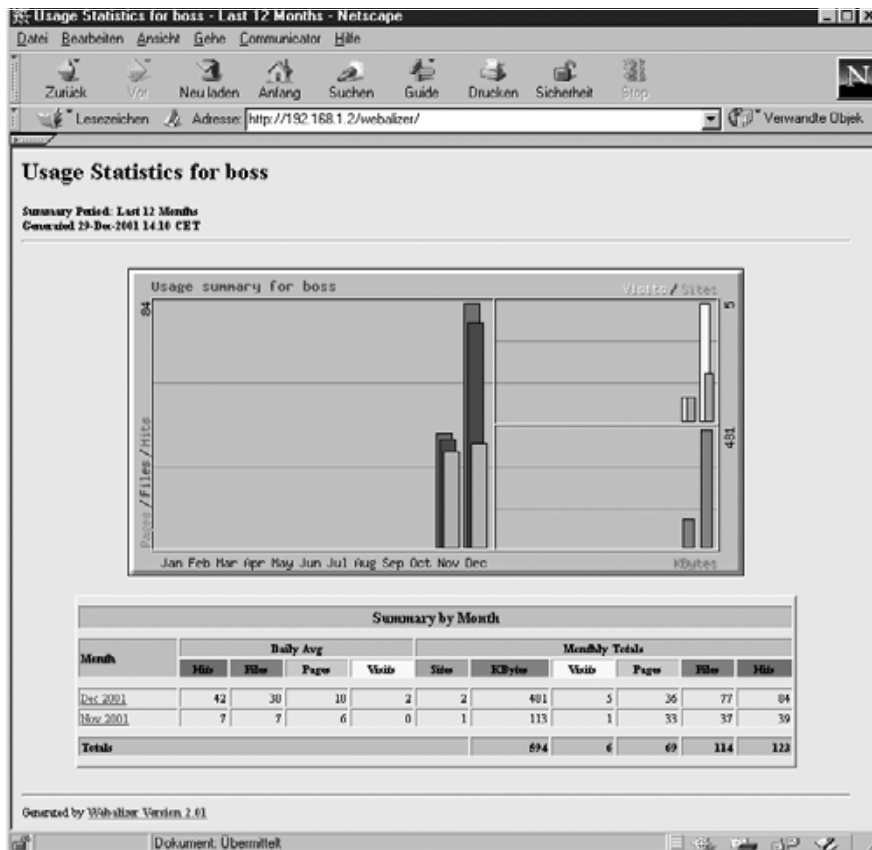


Abbildung 6.10: Webalizer Monatsübersicht

6.9.1 Monatliche Auswertung

Klicken Sie in dieser Übersicht auf einen der Monate, so erhalten Sie eine viel umfangreichere Auswertung für den ausgewählten Monat. In dieser Auswertung finden Sie

- eine Zusammenfassung für den aktuellen Monat,
- die Zugriffsstatistik, aufgeschlüsselt nach den einzelnen Tagen des Monats,
- eine Statistik, aufgeschlüsselt nach Uhrzeiten,
- eine Auswertung der am häufigsten abgerufenen Seiten,
- eine Liste Ihrer häufigsten Einstiegsseiten,

- eine Liste der häufigsten Ausstiegsseiten,
- eine Zusammenstellung, welche Rechner Ihren Server aufgesucht haben,
- eine sehr interessante Liste der Adressen, von denen Ihre Besucher gekommen sind,
- wenn Besucher über Suchmaschinen zu Ihnen gekommen sind, welche Suchbegriffe sie erfolgreich benutzt haben,
- welche Browser die Besucher benutzen und
- aus welchen Ländern die Besucher kommen.

Viele Informationen bereitet der Webalizer sowohl als Tabelle als auch als Grafik auf.

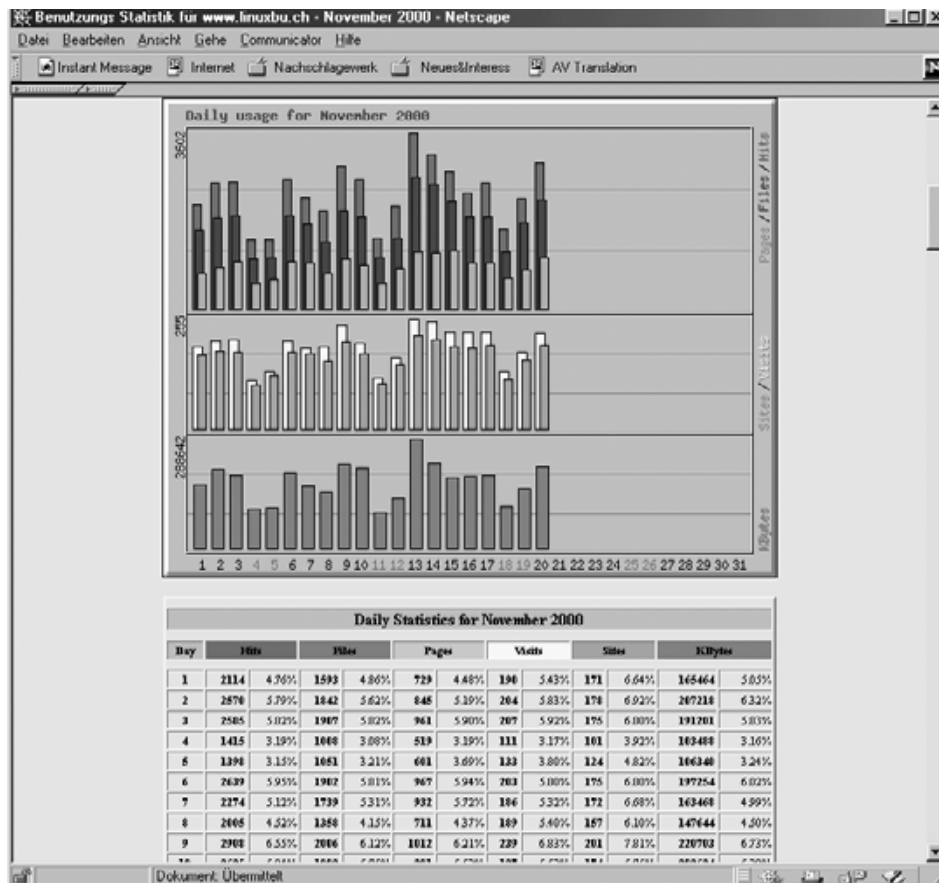


Abbildung 6.11: Tagesstatistik

Die Informationen aus den Auswertungen helfen, gezielt auf die Interessen und Gewohnheiten der Besucher der Web-Site einzugehen.

6.9.2 Konfiguration von Webalizer

Zum Konfigurieren des Webalizer müssen Sie nur die Datei `/etc/webalizer.conf` an Ihre Bedürfnisse anzupassen.

`/etc/webalizer.conf` (Dateianfang)

```
#
# Sample Webalizer configuration file
# Copyright 1997-2000 by Bradford L. Barrett (brad@mrunix.net)
#
# Distributed under the GNU General Public License.  See the
# files "Copyright" and "COPYING" provided with the webalizer
# distribution for additional information.
#
# This is a sample configuration file for the Webalizer
# (ver 2.01)
# Lines starting with pound signs '#' are comment lines and are
# ignored.  Blank lines are skipped as well.  Other lines are
# considered as configuration lines, and have the form
# "ConfigOption  alue" where
# ConfigOption is a valid configuration keyword, and Value is
# the value to assign that configuration option.
# Invalid keyword/values are ignored, with appropriate
# warnings being displayed.  There must be at least
# one space or tab between the keyword and its value.
#
# As of version 0.98, The Webalizer will look for a
# 'default' configuration file named "webalizer.conf"
# in the current directory, and if not found
# there, will look for "/etc/webalizer.conf".

# LogFile defines the web server log file to use.
# If not specified here or on the command line,
# input will default to STDIN.
# If the log filename ends in '.gz' (ie: a gzip compressed
# file), it will be decompressed on the fly as it is being read.

LogFile      /var/log/httpd/access_log

# LogType defines the log type being processed.
# Normally, the Webalizer expects a CLF or Combined
```

```
# web server log as input. Using this option,
# you can process ftp logs as well
# (xferlog as produced by wu-ftp and
# others), or Squid native logs.
# Values can be 'clf', 'ftp' or 'squid',
# with 'clf' the default.

#LogType          clf

# OutputDir is where you want to put the output files.
# This should be a full path name,
# however relative ones might work as well.
# If no output directory is specified, the current
# directory will be used.

OutputDir         /var/lib/webalizer
```

Wichtig ist hier vor allem der Pfad zum Apache-Logfile:

```
LogFile           /var/log/httpd/access_log
```

Wenn Ihre Statistik allgemein zugänglich sein soll, ist es geschickter, statt `/var/lib/webalizer` das ebenfalls von der SuSE-Installation angelegte Verzeichnis `/srv/www/htdocs/webalizer` zu nutzen. Ändern Sie also an dieser Stelle die Konfigurationsdatei.

```
OutputDir         /srv/www/htdocs/webalizer
```

Der Webalizer ist dann ohne weitere Änderung sofort einsatzbereit. Starten Sie das Programm von der Konsole aus, indem Sie

```
webalizer
```

eingeben. Sobald das Programm die Reports erzeugt hat, können Sie in einem beliebigen Browser das Ergebnis unter der URL

```
http://192.168.1.2/webalizer/
```

aufrufen. Bei einem neu installierten System wird die Auswertung noch nicht sehr umfangreich sein, aber das kann sich ja im Laufe der Zeit ändern.

Die Konfigurationsdatei können Sie sehr leicht an Ihre Bedürfnisse anpassen, sie ist sehr gut und ausführlich dokumentiert.

6.9.3 Webalizer automatisieren

Da der Webalizer sehr schnell ist und Ihr System nicht unnötig belastet, können Sie ihn täglich starten. Dazu bietet sich ein Cronjob wie im folgenden Auszug aus der Crontab von root an:

```
PATH=/bin:/usr/bin:/usr/local/bin:/sbin:/root/bin:/root/sbin
mailto=root

50 23 * * * webalizer
```

Rufen Sie den Webalizer täglich kurz vor Mitternacht auf, da bei SuSE-Systemen Cron um Mitternacht einen Job startet, der die Länge von Log-Dateien überwacht und diese gegebenenfalls stutzt. Wenn Sie den Webalizer erst danach starten, fehlen Ihnen die Zugriffe zumindest des letzten Tages, was hässliche Lücken in der Statistik hinterlässt.

Damit der Webalizer seine Auswertungen speichert, sollten Sie unbedingt die `webalizer.conf` bearbeiten.

```
/etc/webalizer.conf (Auszug ab Zeile 54)
# Incremental processing allows multiple partial log files to
# be used instead of one huge one.
# Useful for large sites that have to rotate their log files
# more than once a month.
# The Webalizer will save its internal state before exiting,
# and restore it the next time run, in order to continue
# processing where it left off.
# This mode also causes The Webalizer to scan for and ignore
# duplicate records (records already processed by a previous
# run).
# See the README file for additional information.
# The value may be 'yes' or 'no', with a default of 'no'.
# The file 'webalizer.current' is used to store the current
# state data, and is located in the output directory of the
# program (unless changed with the IncrementalName option
# below).
# Please read at least the section on Incremental processing
# in the README file before you enable this option.

#Incremental      no
```

Damit der Webalizer den Status der bisherigen Auswertungen speichert, ändern Sie die hervorgehobene Zeile in:

```
Incremental      yes
```

Falls dann Cron die Logdateien des Apache verkürzt, bleiben die Informationen über die vergangenen Wochen und Monate trotzdem erhalten. Wenn sie die Voreinstellung belassen, würde Webalizer immer nur die Informationen darstellen, die sich aktuell in der Apache-Logdatei befinden.

Webalizer kann nicht nur die Statistiken des Web-Servers auswerten, sondern auch die des FTP-Servers und des Proxyservers. Sie werden daher in den entsprechenden Kapiteln erneut auf dieses Programm stoßen.

6.10 Eine eigene Suchmaschine mit htdig

Wenn Ihre Web-Site anfängt zu wachsen, taucht schnell der Wunsch nach einer eigenen Suchmaschine auf. Eine Suchmaschine ermöglicht den Nutzern Ihrer Web-Site, Informationen gezielt auf Ihrem Web-Server zu suchen, unabhängig von der vorgegebenen Navigationsstruktur.

Ein sehr leistungsfähiges, aber trotzdem einfach zu konfigurierendes Programm ist *ht://Dig*, dessen aktuellste Version Sie im Web an der Adresse <http://www.htdig.org/> finden. Die aktuelle SuSE Distribution installiert dieses Programm normalerweise mit. Ansonsten können Sie es über die Selektion *Einfacher Web-server* finden.

6.10.1 Konfiguration von ht://Dig

Die Konfigurationsdatei finden Sie unter `/srv/www/htdig/conf/htdig.conf`, hier müssen Sie nur wenig ändern, vor allem Ihre Start-URL:

```
#
# Example config file for ht://Dig.
#
# This configuration file is used by all the programs that
# make up ht://Dig.
# Please refer to the attribute reference manual for more
# details on what can be put into this file.
# (http://www.htdig.org/confindex.html)
# Note that most attributes have very reasonable default values
# so you really only have to add attributes
# here if you want to change the defaults.
#
# What follows are some of the common attributes you might
# want to change.
#
# Specify where the database files need to go.
# Make sure that there is plenty of free disk space
# available for the databases. They can get pretty big.
#
```

```

database_dir:          /var/lib/htdig/db

#
# This specifies the URL where the robot (htdig) will start.
# You can specify multiple URLs here. Just separate them by
# some whitespace. The example here will cause the
# ht://Dig homepage and related pages to be indexed.
# You could also index all the URLs in a file like so:
# start_url:           `${common_dir}/start.url`
#
start_url:             http://www.htdig.org/

```

Mit der vorgegebenen Einstellung würden Sie die Web-Site von www.htdig.org durchsuchen. Um Suche in Ihrem lokalen Web-Server zu erlauben, ändern Sie die URL-Zeile dem Beispiel dieses Buchs folgend in:

```
start_url:             http://192.168.1.2/
```

Etwas später in der Konfigurationsdatei finden Sie den Abschnitt:

```

#
# The string htdig will send in every request to identify
# the robot. Change this to your email address.
#
maintainer:           unconfigured@htdig.searchengine.
└─ maintainer

```

Diese Mailadresse hinterläßt *ht://Dig* in den Logdateien der besuchten Web-Server, von daher sollte sie auf Ihr System verweisen.

```

#
# The string htdig will send in every request to identify
# the robot. Change this to your email address.
#
maintainer:           debacher@boss.lokales-netz.de

```

Damit ist Ihre Suchmaschine bereits einsatzbereit.

Die Arbeit einer Suchmaschine besteht immer aus zwei Teilen:

- Indizieren der Seiten,
- Beantworten von Suchanfragen.

6.10.2 Seiten indizieren

Bevor Sie das Suchen erlauben, müssen Sie zuerst einen Index für Ihre Suchmaschine aufbauen. Dazu rufen Sie an der Konsole:

```
/usr/bin/rundig
```


auf. Dieser Befehl wird Ihren Linux-Server für einige Minuten beschäftigen. Der Zeitbedarf fürs Indizieren hängt von der Leistungsfähigkeit und sonstigen Belastung des Linux-Servers und dem Umfang Ihrer Web-Site ab.

Sowie Ihre Suchmaschine zufrieden stellend funktioniert, sollten Sie das Indizieren der Web-Site über einen Cronjob automatisieren:

```
20 03 * * 7 /usr/bin/rundig
```

Hiermit aktualisieren Sie an jedem Sonntag um 03.20h Ihren Suchindex. Beim Planen dieses Cronjobs sollten Sie bedenken, dass *ht://Dig* beim Indizieren alle Seiten laden und auswerten muss, was den Web-Server belastet.

6.10.3 Beantworten von Suchanfragen

Sowie Sie den Index einmal aufgebaut haben, können Sie auch Suchanfragen starten. Die dafür notwendige Programmkomponente *htsearch* finden Sie im Verzeichnis `/srv/www/cgi-bin/`. Geben Sie in Ihrem Browser die URL

```
http://192.168.1.2/cgi-bin/htsearch
```

ein. Als Antwort sollten Sie folgende Seite erhalten:

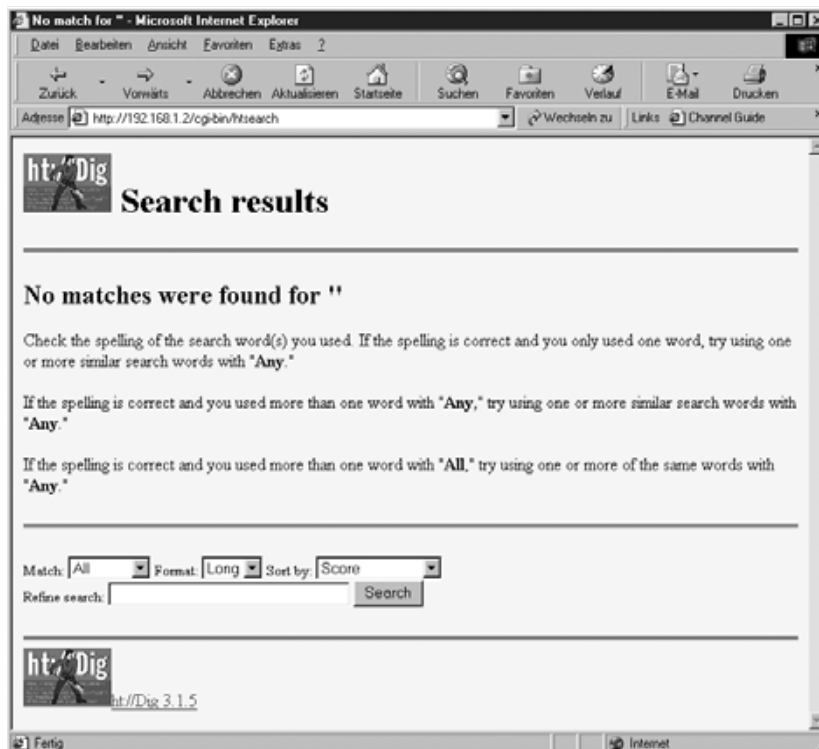


Abbildung 6.12: Erste Suche mit ht://Dig

Die Seite meldet einen Fehler, weil Sie dem Programm *htsearch* keinen Suchbegriff übergeben haben. Sie müssen dazu ein Formular im Stil Ihrer Web-Site erstellen, das den Suchbegriff übergibt. Für einen ersten Versuch können Sie das Eingabefeld auf der Seite mit der Fehlermeldung verwenden.

Das folgende Listing, das Sie im Verzeichnis `/srv/www/htdocs/suche.html` ablegen können, enthält ein Muster für ein eigenes Suchformular :

```
<html><head><title>Suche mit ht://Dig</title></head><body>
<h1>Suche mit ht://Dig</h1><hr noshade size="4">
<p>
<form method="get" action="/cgi-bin/htsearch">
<font size="-1">
Treffer: <select name="method">
<option value="and" selected>All
<option value="or">Any
<option value="boolean">Boolean
</select>

Format: <select name="format">
<option value="builtin-long">Long
<option value="builtin-short">Short
</select>

Sortiert nach: <select name="sort">
<option value="score" selected>Score
<option value="time">Time
<option value="title">Title
<option value="revscore">Reverse Score
<option value="revtime">Reverse Time
<option value="revtitle">Reverse Title
</select>

<br>Suchbegriff:
<input type="text" size="30" name="words" value="">
<input type="submit" value="Search">
</select>
</font>
</form>
</body></html>
```

Falls die Seiten, auf denen *ht://Dig* die Suchergebnisse präsentiert, nicht in Ihr Layout passen, so können Sie diese nahezu beliebig anpassen. Vorschläge für passende HTML-Seiten finden Sie im Verzeichnis `/srv/www/htdig/common/`.

Ein Beispiel für die Anpassung der Suchmaschine an das eigene Layout ist die Suchfunktion auf den Seiten von <http://www.linuxbu.ch>.

Die Suchfunktion fügt sich hier nahtlos in das Gesamtlayout ein.

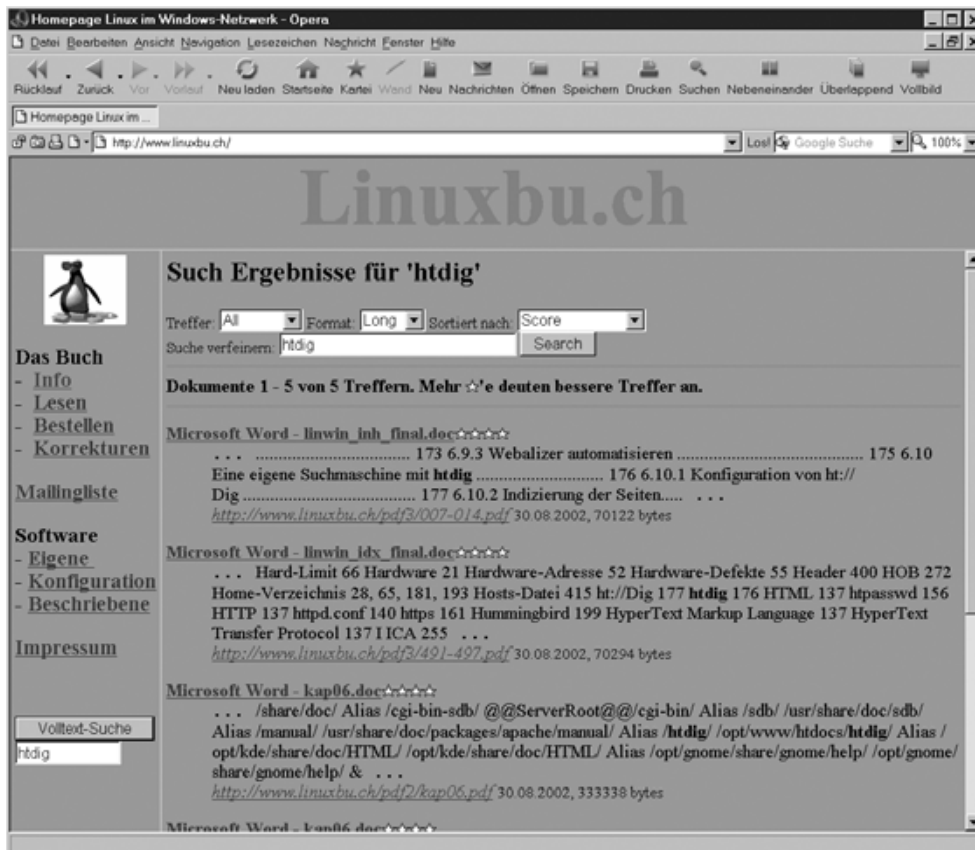


Abbildung 6.13: ht://Dig im eigenen Layout