208 HTTP-Dienste

Der Apache Webserver ist eines der erfolgreichsten Open-Source-Projekte überhaupt. Aufgrund seiner Sicherheit, Stabilität und Leistungsfähigkeit wurde sein Erfolg bis heute durch keinen anderen Webserver infrage gestellt.

208.1 Grundlegende Apache-Konfiguration

Wichtung: 4

Beschreibung: Die Prüflinge sollten dazu in der Lage sein, einen Webserver zu installieren und einzurichten. Dieses Lernziel beinhaltet die Überwachung der Serverauslastung und -leistung, die Beschränkung von Benutzerzugriffen, die Einrichtung der Unterstützung von Skriptsprachenmodulen und die clientseitige Benutzerauthentifizierung. Des Weiteren ist auch die Konfiguration der Serveroptionen, die der Einschränkung der Ressourcennutzung dienen, mit eingeschlossen. Kandidaten sollten dazu in der Lage sein, virtuelle Hosts auf Webservern zu erstellen und den Dateizugriff anzupassen.

Wichtigste Wissensgebiete:

- Apache 2.4-Konfigurationsdateien, -Begriffe und -Dienstprogramme
- Konfiguration der Apache-Protokolldateien und deren Inhalte
- Methoden und Dateien zur Zugriffsbeschränkung
- Konfiguration von *mod_perl* und PHP
- Client-Benutzerauthentifizierungsdateien und -Dienstprogramme
- Einstellung der maximalen Anzahl an Anfragen sowie der minimalen und maximalen Anzahl an Serverprozessen und Clients
- Apache 2.4-Implementierung von virtuellen Hosts (mit und ohne fest zugeordneten IP-Adressen)
- Einsatz von Redirect-Anweisungen in Apache-Konfigurationsdateien, um Dateizugriffe zu individualisieren

Liste wichtiger Dateien, Verzeichnisse und Anwendungen:

- Zugriffslogdateien (Access Logs) und Fehlerlogdateien (Error Logs)
- *.htaccess*

- *httpd.conf*
- *mod_auth_basic*, *mod_authz_host* und *mod_access_compat*
- `htpasswd`
- *AuthUserFile*, *AuthGroupFile*
- `apachectl`, `apache2ctl`
- *httpd*, *apache2*

Allgemeines

Der *Apache Webserver* ist nach wie vor der erfolgreichste Webserver im Internet. Die Ursprünge des Apache Webservers liegen in einem inzwischen ausgestorbenen Webserver, dem NCSA-Webserver *httpd*. Dieser wurde von der Organisation NCSA im Jahre 1995 eingestellt und nicht weiterentwickelt. Die NCSA ist übrigens dieselbe Organisation, die im Jahre 1993 den damals sensationellen Webbrowser *Mosaic* entwickelte. Das war der erste Browser, der auch von normalen Menschen bedient werden konnte, die nicht als absolute Computerfreaks bekannt waren.

Nachdem die NCSA ihren Webserver in der Version *httpd 1.3* eingestellt hatte, fingen diverse Entwickler, die sich später zur Apache Software Foundation zusammenschlossen, an, Sicherheitslücken in diesem Webserver zu beheben und ihm neue Funktionen hinzuzufügen. Es wurden immer mehr Patches zusammengetragen und den Aussagen von Zeitzeugen nach wurde daraus zunächst die Bezeichnung »A patchy Server«. Es gibt aber auch die Aussage, der Name wäre eine Hommage an die gleichnamigen nordamerikanischen Indianerstammesgruppen. Möglicherweise ist an beiden Geschichten etwas dran.

Installation von Apache

Es gibt verschiedene Möglichkeiten, wie man zu einem Apache Webserver kommen kann. In Abhängigkeit von der verwendeten Linux-Distribution können Sie natürlich `yum` oder `aptitude` verwenden, um den Webserver zu installieren. Damit Sie den Webserver in seiner reinsten Form kennenlernen, ist es im Rahmen der Prüfungsvorbereitung allerdings sinnvoller, Apache aus einem tar-Ball heraus zu installieren. Sie finden die entsprechenden Downloadlinks zu etlichen HTTP- und FTP-Mirror-Servern auf der Webseite *http://www.apache.org*.

Die folgende Beispielinstallation wurde auf einem Server unter CentOS ausgeführt. Führen Sie einfach die folgenden Kommandos aus, um einen funktionierenden Webserver zu installieren:

```
[root@arch-cent ~]# mkdir /usr/src/apache-2.2
[root@arch-cent ~]# cd /usr/src/apache-2.2/
```


Aus dem vorbereiteten Installationsverzeichnis heraus kann man mittels `wget` den tar-Ball gleich an Ort und Stelle herunterladen. Sie können natürlich auch eine andere Quelle und eine andere Version von Apache verwenden, wenn Sie das möchten.

```
[root@arch-cent apache-2.2]# wget \
http://ftp.halifax.rwth-aachen.de/apache/httpd/httpd-2.4.23.tar.gz
```

Packen Sie den tar-Ball aus:

```
[root@arch-cent apache-2.4]# tar xvzf httpd-2.4.23.tar.gz
```

Wechseln Sie in das Verzeichnis, das gerade durch die Extraktion des tar-Balls entstanden ist:

```
[root@arch-cent apache-2.4]# cd httpd-2.4.23
```

Damit Sie den Webserver konfigurieren und kompilieren können, benötigen Sie spätestens jetzt einen Compiler. Ich persönlich bevorzuge den GNU-C- und den GNU-C++-Compiler:

```
[root@arch-cent httpd-2.4.23]# yum install gcc
```

Bei der Konfiguration sollten Sie zumindest das Installationsverzeichnis mit `--prefix` übergeben. Das Verzeichnis */usr/local/apache2* ist ein typisches Ziel für den Webserver. Die anderen Optionen wählen die Funktionalitäten *http*, *https*, *cgi* und das automatische Einbinden von Modulen.

```
[root@arch-cent httpd-2.4.23]# ./configure --prefix=/usr/local/apache2 \
--enable-http --enable-https --enable-so --enable-cgi
```

Es gibt sehr viele Optionen, die Sie für die Konfiguration übergeben können. Sie erhalten eine komplette Liste der Optionen, indem Sie im Installationsverzeichnis das Kommando `./configure --help` ausführen.

Wenn die Konfiguration abgeschlossen ist, können Sie *httpd* mit `make` kompilieren:

```
[root@arch-cent httpd-2.4.23]# make
```

Wie bei der Installation vieler anderer tar-Balls folgt jetzt die eigentliche Installation des Webservers:

```
[root@arch-cent httpd-2.4.23]# make install
```

Wenn alles gut gegangen ist, können Sie den Apache Webserver jetzt zum ersten Mal starten:

```
[root@arch-cent /]# /usr/local/apache2/bin/apachectl start
httpd: Could not reliably determine the server's fully qualified domain name,
using 127.0.0.1 for ServerName
```

Es ist wahrscheinlich, dass Sie beim Start des Servers obige Fehlermeldung erhalten. Der Webserver läuft dann trotzdem, aber Sie sollten diesen Schönheitsfehler beheben, indem Sie der Datei */usr/local/apache2/conf/httpd.conf* (das ist übrigens die Hauptkonfigurationsdatei des Webservers) eine Zeile wie diese hinzufügen:

```
ServerName arch-cent.homelinux.net:80
```

Prüfen Sie nach, ob der Webserver läuft, indem Sie nach Instanzen des Daemons *httpd* suchen. Das ist der Daemon des Apache Webservers:

```
[root@arch-cent /]# ps -A|grep httpd
 9487 ?        00:00:00 httpd
 9488 ?        00:00:00 httpd
 9489 ?        00:00:00 httpd
 9490 ?        00:00:00 httpd
```

Den eigentlichen Test führen Sie dann mit einem Webbrowser aus. Wenn Sie lokal an derselben Maschine arbeiten, die den Webserver ausführt, rufen Sie einfach die URL *http://localhost* auf. Alternativ können Sie natürlich auch von einem anderen PC aus über das Netzwerk zugreifen. Der angezeigte Content befindet sich im Document-Root-Verzeichnis */usr/local/apache2/htdocs*, wenn Sie Apache von Hand installiert haben. Sie können hier Ihre eigenen Inhalte unterbringen.

Konfigurationsdateien

Die Konfigurationsdateien für Apache befinden sich üblicherweise in den Verzeichnissen */usr/local/apache2/conf* (wenn Sie den Server selbst kompiliert haben, befinden Sie sich genau da), */etc/apache*, */etc/apache2* oder */etc/httpd/conf*. In Abhängigkeit von der verwendeten Linux-Distribution können Sie sich aber auch noch in anderen Verzeichnissen befinden (hier hilft z. B. `find /etc -name httpd.conf`). Leider variiert auch der Inhalt des jeweiligen Konfigurationsverzeichnisses. Das liegt daran, dass in den meisten Linux-Distributionen Teile der Hauptkonfigurationsdatei *httpd.conf* in kleinere Dateien ausgelagert wurden, um die Übersichtlichkeit zu erhöhen. In der LPI-Prüfung wird häufig nach den folgenden Dateien gefragt:

- *httpd.conf* ist die Hauptkonfigurationsdatei. In dieser Datei stehen alle Parameter, die für den Betrieb des Webservers notwendig sind.
- *srm.conf* enthielt früher die `ResourceConfig`-Anweisungen. Da diese Anweisungen inzwischen mit in *httpd.conf* integriert worden sind, ist *srm.conf* heutzutage leer oder gar nicht mehr vorhanden.
- *access.conf* enthielt früher die Anweisungen zur Zugriffssteuerung. Auch diese Anweisungen wurden inzwischen mit in die Datei *httpd.conf* aufgenommen.

Wenn Sie Apache selbst konfiguriert und kompiliert haben, enthält dessen Konfigurationsverzeichnis */usr/local/apache2/conf* ein Unterverzeichnis namens *extra*. In diesem Unterverzeichnis finden Sie weitere Konfigurationsdateien, die mittels `include`-Anweisungen in die Datei *httpd.conf* integriert worden sind:

```
httpd-autoindex.conf
httpd-languages.conf
httpd-ssl.conf
httpd-dav.conf
httpd-manual.conf
httpd-userdir.conf
httpd-default.conf
httpd-mpm.conf
httpd-vhosts.conf
httpd-info.conf
httpd-multilang-errordoc.conf
```

Standardmäßig sind die zu diesen Dateien gehörenden `include`-Anweisungen auskommentiert. Wenn Sie eine der Konfigurationsdateien benötigen, müssen Sie lediglich in der Datei *httpd.conf* in der entsprechenden Zeile am Anfang die Raute entfernen. Der Verwendungszweck der einzelnen Dateien ist in der Hauptkonfigurationsdatei *httpd.conf* dokumentiert.

Tipp

Wenn Sie `httpd-manual.conf` aktivieren, indem Sie die vorangestellte Raute entfernen, haben Sie Zugriff auf das Manual von Apache. Sie müssen anschließend lediglich `apachectl restart` ausführen und können daraufhin auf die URL *http://localhost/manual* zugreifen.

Wichtige Einträge in der Datei httpd.conf

Unabhängig von deren tatsächlicher Wichtigkeit werden nun einige Einträge aus der Datei *httpd.conf* aufgezählt, die in der Prüfung häufig auftauchen. Lassen Sie sich nicht von Fragen erschrecken, die Optionen enthalten, die Sie gar nicht kennen. Oft kann man durch Logik und Ausschlussverfahren die richtigen Antworten ermitteln.

- `Port` ändert den Port, an dem Apache für eingehende Webanfragen lauscht. Der Standardport ist 80. Diese Direktive wurde in neueren Apache-Versionen durch `Listen` ersetzt.
- `Listen` ändert den Port und ggf. die IP-Adresse, an der Apache für eingehende Webanfragen lauscht. Standardmäßig ist hier Port 80 ohne IP-Adresse angegeben.

- `MinSpareServers` legt die Anzahl der Instanzen fest, die beim Starten von Apache in den Speicher geladen werden. Ein typischer Wert für kleine bis mittlere Serverauslastungen ist `10`.
- `ServerType` legt fest, ob Apache von `inetd` gestartet wird oder ob er selbstständig läuft. Mögliche Werte sind `standalone` oder `inetd`.
- `ServerRoot` ist das Verzeichnis, in dem Apache seine Protokolle, Serverkonfiguration, CGI-Skripte und Ähnliches findet.
- `DocumentRoot` ist das Hauptverzeichnis für Dokumente, zu dem die Öffentlichkeit Zugang haben soll.

Prüfungstipp

Diese Einträge sollten Sie unbedingt kennen. Achten Sie darauf, dass Sie nicht `Port` und `Listen` miteinander verwechseln. Die Anzahl der `MinSpareServers` für eine kleine bis mittlere Umgebung wird oft und in vielen Varianten gefragt. Merken Sie sich also hierbei unbedingt die Zahl 10!

Starten und stoppen

Es gibt mehrere Methoden, um einen Apache Webserver zu starten, zu beenden oder neu zu starten. Sie sollten die hierfür benötigten Kommandos kennen:

- Wie jeden anderen Daemon können Sie auch *httpd* mit dem Kommando `/etc/init.d/httpd start` starten. Sollten Sie den Server selbst kompiliert haben, gibt es kein Startskript. Sie können dann einfach ein vorhandenes Skript kopieren und entsprechend anpassen.
- ohne `init`-Skript: `/usr/local/apache2/bin/httpd -f /usr/local/apache2/conf/httpd.conf`
- `/usr/local/apache2/bin/apachectl` ist zur Laufzeit allerdings die eleganteste Lösung. Das Skript versteht die folgenden Optionen:
  - `start` startet den Server.
  - `stop` beendet den Server.
  - `restart` startet den Server neu.
  - `graceful` startet den Server neu, aber bestehende Verbindungen bleiben erhalten. So können etwaige Konfigurationsänderungen registriert werden.
  - `configtest` überprüft die Konfigurationsdateien auf Syntaxfehler.

Sie werden auf einigen Systemen `apache2ctl` anstatt `apachectl` antreffen. Die beiden Skripte unterscheiden sich allerdings nicht nennenswert voneinander.

Im Allgemeinen ist `apachectl` nicht sehr gesprächig und gibt lediglich beim `configtest` eine Erfolgsmeldung aus. In allen anderen Fällen sollten Sie sich bei einem Produktionsserver davon überzeugen, dass er die von Ihnen angeforderte Aktion auch wirklich durchgeführt hat.

```
[root@arch-cent bin]# ./apachectl configtest
Syntax OK
```

Die Richtigkeit der Konfigurationsänderungen wurde bestätigt.

Zugriffssteuerung

Zur Steuerung des Zugriffs auf eine Webseite gibt es eine ganze Reihe von Modulen, die entweder standardmäßig im Kern von Apache integriert sind oder optional geladen werden können. Drei ausdrücklich als Prüfungsthema angegebene Module sind:

- *mod_auth_basic* zur Authentifizierung in Klartext
- *mod_authz_host* zur Zugriffskontrolle über Hostnamen oder IP-Adressen (früher *mod_access*)
- *mod_access_compat* ist ein Vorläufer von *mod_authz_host*, der weniger Direktiven beinhaltet.

Diese drei Module sind im Kern des Servers enthalten und müssen nicht explizit geladen werden.

Wenn eine Webseite Informationen enthält, die nicht für die Öffentlichkeit bestimmt sind, diese Seite aber im Internet erreichbar ist, sollten Sie eine Authentifizierung von den Benutzern anfordern. Hier kommt das Modul *mod_auth_basic* zum Zuge. Die erforderlichen Benutzerkonten können mit dem Programm `htpasswd` erstellt und anschließend einer Webseite bzw. einem Webverzeichnis zugeordnet werden.

htpasswd verwenden

Die Befehlsfolge im folgenden Beispiel ist so gewählt, dass sie sich einerseits übersichtlich darstellen lässt und andererseits durch kurze Kommandos leicht nachzustellen ist:

```
[root@arch-cent /]# cd /usr/local/apache2/bin/
```

Das folgende Kommando erstellt eine Passwortdatei und legt im selben Arbeitsschritt den ersten Benutzer samt Passwort an. Die Passwortdatei befindet sich anschließend in *ServerRoot /usr/local/apache2*.

```
[root@arch-cent bin]# ./htpasswd -c ../password.list user1
New password:
Re-type new password:
Adding password for user user1
```

Wenn Sie weitere Benutzer für Ihren Webserver erstellen wollen, verwenden Sie ein ähnliches Kommando. Sie müssen dann lediglich die Option `-c` (das `c` steht für »create«) weglassen.

```
[root@arch-cent bin]# ./htpasswd ../password.list user2
New password:
Re-type new password:
Adding password for user user2
```

Wechseln Sie in eine Verzeichnisebene höher, um den Inhalt der Passwortdatei zu betrachten. Sie werden feststellen, dass die Passwörter verschlüsselt abgespeichert wurden:

```
[root@arch-cent bin]# cd ..
[root@arch-cent apache2]# cat password.list
user1:$apr1$MSADRkjD$RiOkhfPNrxeM3sFWuOvvf/
user2:$apr1$puQJAZbN$sgfv1MEZ2AnimKzqIY22DO
```

Um die Authentifizierung für die standardmäßig vorhandene Webseite anzufordern, müssen Sie die Konfigurationsdatei *httpd.conf* anpassen. Suchen Sie die Sektion `<Directory />` und fügen Sie die unten fett gedruckten Zeilen hinzu:

```
<Directory />
    Options FollowSymLinks
    AllowOverride None
    Order deny,allow
    Deny from all
    AuthName "Authentifizierung erforderlich"
    AuthType Basic
    AuthUserFile password.list
    require valid-user
</Directory>
```

Der Inhalt des Statements `AuthName` ist frei wählbar und wird dem Benutzer im Authentifizierungsdialog angezeigt. Mit `AuthType Basic` wird die Methode der Authentifizierung eingestellt. `AuthUserFile` zeigt auf die soeben erstellte Passwortdatei. Diese Datei befindet sich im vorliegenden Beispiel im Hauptverzeichnis des Webservers, weshalb hier auf eine Pfadangabe verzichtet werden kann. Das Statement `require valid-user` legt fest, dass sich hier nur gültige Benutzer anmelden können. Es ist übrigens auch möglich, konkreten Benutzern exklusiven Zugriff einzuräumen.

Sie können die Authentifizierung nun mithilfe eines Webbrowsers testen und anschließend die soeben vorgenommenen Änderungen aus der Datei *httpd.conf* löschen oder auskommentieren. Diese Maßnahme ist für die nächste Lektion nötig.

.htaccess

Eine andere Methode der Zugriffssteuerung ist die Verwendung einer verborgenen Datei mit der Bezeichnung *.htaccess*. Dieses Verfahren unterscheidet sich kaum von dem bereits beschriebenen Verfahren. Sie können sogar die zuvor erstellte Passwortdatei verwenden.

Damit Sie *.htaccess*-Dateien verwenden können, müssen Sie zuerst in der Datei *httpd.conf* die folgende Einstellung in der gewünschten `<Directory />`-Direktive konfigurieren:

```
AllowOverride All
```

Erstellen Sie anschließend eine Datei mit der Bezeichnung *.htaccess* und folgendem Inhalt im DocumentRoot des Webservers:

```
AuthName "Authentifizierung erforderlich"
AuthType Basic
AuthUserFile password.list
require valid-user
```

Wie Sie sehen, sind Inhalt und Syntax mit den Einträgen in der *httpd.conf*-Datei, die Sie vorher gemacht haben, absolut identisch. Der einzige Vorzug dieser Datei ist, dass sie auch ein gewöhnlicher Benutzer erstellen kann. Dazu benötigt der Benutzer keine Schreibrechte auf die Datei *httpd.conf*, sondern lediglich im jeweiligen Webverzeichnis. Auf diese Art kann die Verwaltung einzelner Webverzeichnisse an unterschiedliche Benutzer delegiert werden.

Testen Sie nun erneut den Zugriff auf den Webserver. Wenn Sie wiederholt Tests durchführen, sollten Sie nach jedem Test den Cache Ihres Browsers leeren. Ansonsten werden Ihre Tests zu unerwarteten Ergebnissen führen, weil Webseiten eventuell ohne Authentifizierung direkt aus dem Browsercache heraus angezeigt werden. Sie können die Datei *.htaccess* ganz einfach immer in die Verzeichnisse kopieren, auf die Sie den Zugriff beschränken wollen. Es sollte bei hochfrequentierten Servern aber berücksichtigt werden, dass die Verwendung dieser Methode die Performance des Servers negativ beeinflusst.

.htgroup

In größeren Umgebungen kann es sinnvoll sein, die Benutzer, die auf Webressourcen zugreifen dürfen, zu gruppieren. Im Prinzip können Sie die Datei zur Gruppierung von Benutzern nennen, wie Sie wollen, aber der Name *.htgroup* hat sich im Laufe der

Jahre eingebürgert. Diese Datei durch einen Punkt am Anfang zu verstecken ist grundsätzlich eine gute Idee, um die Datei vor neugierigen Augen zu verbergen, auch wenn sie keine Passwörter enthält. Um das Verfahren zu testen, erstellen Sie passend zu den bereits erstellten Benutzern in der Datei *password.list* die Datei *.htgroup*, ebenfalls im DocumentRoot-Verzeichnis des Servers. Die Datei *.htgroup* könnte diesen Inhalt haben:

```
all:user1,user2
agents:user1
```

Ändern Sie nun Ihre bestehende *.htaccess*-Datei ab, um die Gruppendatei zu verwenden:

```
AuthName "Zugriff nur für Agents"
AuthType Basic
AuthUserFile password.list
AuthGroupFile .htgroup
require group agents
```

Beachten Sie bitte, dass `AuthUserFile` angegeben werden muss, auch wenn die Zugangsvoraussetzung von der Gruppe abhängig ist. `AuthName` zeigt einem Benutzer schon beim ersten Zugriff, wer auf dieses Webverzeichnis zugreifen darf. Wenn Sie einen Test durchführen (nachdem Sie den Browsercache geleert haben), werden Sie feststellen, dass `user1` auf das Webverzeichnis zugreifen darf und `user2` nicht.

Hinweis

Es gibt wesentlich komplexere Methoden, um den Zugriff auf einen Webserver bzw. auf dessen Verzeichnisse zu steuern, wie Sie es sich wahrscheinlich schon gedacht haben. Für die Prüfung müssen Sie aber lediglich die bisher beschriebenen Mechanismen kennen.

Zugriff über Namen und IP-Adressen steuern

Wenn eine Webseite lediglich im lokalen Netz erreichbar ist, kann man möglicherweise auf die Authentifizierung der Benutzer verzichten. Das erspart sowohl dem Administrator Zeit, weil er nicht pro User den Zugriff einrichten muss, als auch dem Benutzer, weil er sich nicht mehr an der Webseite anmelden muss. In solchen Fällen kann eine Zugriffssteuerung über die IP-Adressen der Clientcomputer oder deren Host- bzw. Domänennamen erfolgen. Zuständig sind hierfür die Module *mod_authz_host* und *mod_access_compat*. Sie müssen diese Module nicht konfigurieren, denn sie werden standardmäßig geladen. Die Erlaubnis zum Zugriff auf eine Webseite erfolgt dann innerhalb deren Konfiguration mithilfe der Direktive `Allow`. Hier ein paar Beispiele:


```
Allow from beispiel.com
```

Erlaubt den Computern der Domäne beispiel.com den Zugriff. Es ist die Angabe kompletter FQDNs wie auch die Angabe von Domänen.

```
Allow from 192.168
Allow from 192.168.0.0/16
Allow from 192.168.0.0/255.255.0.0
```

Die vorangehenden drei Zeilen sind von der Aussage her identisch. In allen drei Fällen wird Clients, die private Adressen der C-Klasse verwenden, der Zugriff gestattet. Selbstverständlich können auch IPv6-Adressen herangezogen werden:

```
Allow from 2001:6F8:1D2D::/48
```

Module integrieren

Funktionen, die von den meisten Anwendern benötigt werden, sind im Kern von Apache fest integriert. Wenn weitere Funktionen benötigt werden, müssen diese als Module eingebunden werden. Die Integration von Perl und insbesondere PHP gehört allerdings immer noch zu den Standardaufgaben bei der Apache-Konfiguration.

Integration von mod_php

PHP ist eine Skriptsprache, mit der es möglich ist, HTML-Code dynamisch zu generieren. Sehr viele der heutigen Webseiten verwenden PHP auch, um Zugriffe auf MySQL-Datenbanken durchzuführen. Die folgende Anleitung wird Ihnen dabei helfen, PHP in Ihren Webserver zu integrieren. Erstellen Sie zunächst ein Installationsverzeichnis und wechseln Sie hinein:

```
[root@arch-cent /]# mkdir /usr/src/php
[root@arch-cent /]# cd /usr/src/php
```

Laden Sie anschließend den entsprechenden `tar`-Ball für PHP vom Webserver *php.net* herunter und packen Sie ihn aus:

```
[root@arch-cent php]# wget http://de.php.net/distributions/php-5.3.6.tar.bz2
[root@arch-cent php]# tar -xvjf php-5.3.6.tar.bz2
```

Beachten Sie bitte, dass es inzwischen eine neuere Version von PHP geben könnte. Besuchen Sie ggf. die *php.net*-Webseite um Näheres zu erfahren, falls der oben genannte Downloadlink nicht mehr funktionieren sollte. Wechseln Sie nun in das entstandene Installationsverzeichnis:

```
[root@arch-cent php]# cd php-5.3.6
```

Die Konfiguration von PHP setzt das Entwicklungspaket *libxml2-devel* voraus. Sie sollten das Paket also spätestens jetzt nachinstallieren:

```
[root@arch-cent php-5.3.6]# yum install libxml2-devel
```

Bei Systemen, die auf Debian basieren, heißt das entsprechende Paket übrigens *libxml2-dev*. Jetzt können Sie die Konfiguration durchführen. Sie müssen bei der Konfiguration zumindest das Verzeichnis angeben, in dem Apache seine Erweiterungen erwartet:

```
[root@arch-cent php-5.3.6]# ./configure --with-apxs2=/usr/local/apache2/bin/apxs
```

Wenn die Konfiguration sauber durchgelaufen ist, können Sie PHP kompilieren, testen und installieren. Der Test ist übrigens optional und dauert relativ lange. Wenn Sie also lediglich ein Testsystem zu Prüfungsvorbereitungszwecken konfigurieren, können Sie den Test getrost überspringen.

```
[root@arch-cent php-5.3.6]# make
[root@arch-cent php-5.3.6]# make test
[root@arch-cent php-5.3.6]# make install
```

Die Installationsroutine hat der Konfigurationsdatei *httpd.conf* eine Zeile hinzugefügt, die dafür sorgt, dass das PHP-Modul beim nächsten Neustart des Webservers geladen wird:

```
LoadModule php5_module        modules/libphp5.so
```

Damit Apache die Dateierweiterungen von PHP selbstständig erkennen kann, sollten Sie allerdings noch die folgende Zeile von Hand in die globale Sektion der Datei *httpd.conf* eintragen:

```
AddType application/x-httpd-php .php .phtml
```

Überprüfen Sie die Konfiguration des Webservers und starten Sie ihn anschließend neu, um die Konfigurationsänderungen zu übernehmen:

```
[root@arch-cent bin]# ./apachectl configtest
[root@arch-cent bin]# ./apachectl restart
```

PHP-Programmierung ist nicht Bestandteil Ihrer Prüfung, aber Sie finden im Internet leicht Beispielskripte um Ihre Konfiguration zu testen, wenn Sie das möchten.

Integration von mod_perl

Perl ist eine Skriptsprache, die ursprünglich nicht für Webseitenprogrammierung geschrieben worden ist. In den letzten Jahren wurde Perl, zumindest was Webseiten

anbelangt, mehr und mehr von PHP verdrängt. Die Installation des Perl-Moduls für Apache funktioniert etwas anders als die Integration von PHP. Konsequenterweise kommt bei der Installation ein Perl-Skript zum Einsatz, was natürlich voraussetzt, dass der Perl-Interpreter schon auf dem System vorhanden ist. Bei den meisten aktuellen Linux-Distributionen ist dieser Interpreter in einer Standardinstallation schon enthalten. Wenn Sie die folgenden Schritte durchführen, sollten Sie eine funktionierende Apache-Konfiguration mit Perl erhalten. Legen Sie zunächst wieder ein Installationsverzeichnis an, und wechseln Sie hinein:

```
[root@arch-cent /]# mkdir /usr/src/mod_perl
[root@arch-cent /]# cd /usr/src/mod_perl
```

Laden Sie anschließend das Perl-Modul als `tar`-Ball von der Apache-Webseite herunter:

```
[root@arch-cent mod_perl]# wget http://perl.apache.org/dist/mod_perl-2.0-
current.tar.gz
```

Es könnte jetzt natürlich schon neuere Versionen des Moduls geben. Passen Sie dann die Kommandos (bzw. hier die URL) entsprechend an. Packen Sie den `tar`-Ball aus:

```
[root@arch-cent mod_perl]# tar -xvzf mod_perl-2.0-current.tar.gz
```

Wechseln Sie anschließend in das gerade entstandene Installationsverzeichnis und führen Sie das Perl-Skript zur Erstellung des Makefiles aus wie angegeben. Die Variable `MP_APXS` gibt hierbei den absoluten Pfad zu den Apache-Erweiterungen an.

```
[root@arch-cent mod_perl]# cd mod_perl-2.0.5
[root@arch-cent mod_perl-2.0.5]# perl Makefile.PL MP_APXS=/usr/local/apache2/
bin/apxs
```

Ab hier unterscheidet sich die Installation nicht mehr von der Installation anderer Programme, die als `tar`-Ball vorliegen:

```
[root@arch-cent mod_perl-2.0.5]# make
[root@arch-cent mod_perl-2.0.5]# make install
```

Sie müssen jetzt nur noch dafür sorgen, dass Apache das Perl-Modul lädt. Fügen Sie der Konfigurationsdatei *httpd.conf* deshalb noch folgende Zeile hinzu:

```
LoadModule perl_module modules/mod_perl.so
```

Starten Sie Apache anschließend einmal neu, damit die Konfigurationsänderungen wirksam werden:

```
[root@arch-cent mod_perl-2.0.5]# /usr/local/apache2/bin/apachectl restart
```

Auch Perl-Programmierung ist nicht Bestandteil der LPI-Prüfungen. Deshalb muss ich auch hier bezüglich Perl-Programmen zu Testzwecken auf das Internet verweisen.

Protokollierungseinstellungen

Was die Protokollierung anbelangt, werden Sie bei den meisten Apache Webservern, die paketbasiert installiert worden sind, feststellen, dass die Protokolle der Webserver im Verzeichnis */var/log/apache2* liegen. Hier befinden sich im Normalfall die Dateien *access.log* und *error.log*. Da diese Dateien üblicherweise von *logrotate* rotiert werden, finden Sie hier auch noch archivierte, komprimierte Versionen dieser beiden Dateien.

In der Datei *access.log* wird jede einzelne URL, die abgerufen wird, protokolliert. Es ist deshalb möglich, sehr genau festzustellen, von welchen Computern aus auf welche Inhalte zugegriffen worden ist. Fehlgeschlagene Zugriffe werden hier ebenfalls protokolliert.

Die Datei *error.log* enthält keine Einträge, die auf Zugriffe durch Benutzer zurückzuführen sind, sondern vielmehr nur schwerwiegende Fehler. Hierbei kann es sich um Fehler durch Fehlkonfiguration des Servers handeln oder Module, die sich aus irgendeinem Grund nicht laden lassen. Sie sollten diese Datei also unbedingt konsultieren, wenn es zu Fehlfunktionen des Servers kommt.

Wenn Sie Ihren Webserver nach der Anleitung in diesem Buch konfiguriert haben, finden Sie die Protokolldateien in einem Unterverzeichnis des Webservers, nämlich in */usr/local/apache2/logs*. Die Dateinamen sind hier *access_log* und *error_log*.

Die Position und die Dateinamen der Protokolldateien werden in der Konfigurationsdatei *httpd.conf* festgelegt. Hierbei sind die Pfadangaben relativ ab dem ServerRoot zu betrachten:

```
CustomLog "logs/access_log" common
ErrorLog "logs/error_log"
```

Leistungseinstellungen

In der Konfigurationsdatei *httpd.conf* finden Sie einige Direktiven, die sich auf die Leistung des Servers auswirken. Diese Direktiven müssen an die Umstände angepasst werden, unter denen der Server läuft. Wenn zu erwarten ist, dass ein Webserver ständig von sehr vielen Benutzern verwendet wird, sollten natürlich auch entsprechend viele Arbeitsprozesse zur Verfügung stehen, um diese Anfragen zu handhaben. Umgekehrt würden zu viele Arbeitsprozesse unnötigerweise die Systemressourcen verschwenden. Entsprechend den Anforderungen können Sie folgende Parameter konfigurieren:

- `StartServers`: Anzahl der Serverprozesse beim Start
- `MinSpareServers`: minimale Anzahl von Serverprozessen, die als Reserve zur Verfügung stehen müssen
- `MaxSpareServers`: maximale Anzahl von Serverprozessen, die als Reserve zur Verfügung stehen dürfen
- `ServerLimit`: maximale Anzahl von Serverprozessen, die zur Laufzeit des Servers ausgeführt werden dürfen
- `MaxClients`: maximale Anzahl von Serverprozessen, die gleichzeitig ausgeführt werden dürfen
- `MaxRequestsPerChild`: maximale Anzahl von Anfragen, die an einen Serverprozess gesendet werden dürfen

Konfiguration virtueller Hosts

Wenn auf einem Webserver mehrere Webseiten gehostet werden sollen, gibt es grundsätzlich drei verschieden Möglichkeiten, um diese Seiten voneinander zu unterscheiden. Sie könnten jeder Webseite eine eigene IP-Adresse zuordnen und die jeweiligen IP-Adressen in den entsprechenden DNS-Zonen eintragen (lassen). Da öffentliche IP-Adressen Geld kosten, ist das allerdings keine sehr gute Wahl. Eine weitere Möglichkeit besteht darin, unterschiedliche TCP-Ports für die verschiedenen Webseiten zu verwenden. Das ist allerdings bei öffentlich zugänglichen Webseiten schon aus kosmetischen Gründen abzulehnen, weil die Besucher der Webseite den entsprechenden Port dann in der URL-Zeile angeben müssten. Das sähe dann beispielsweise so aus: *http://www.beispiel.de:82*, also nicht gerade besonders professionell. Die dritte und im Normalfall beste Lösung ist die Unterscheidung von Webseiten über den Hostnamen. Die URL, die der Benutzer in seinen Browser eingegeben hat, wird innerhalb der Abfrage an den Webserver übermittelt. Diese URL kann der Webserver auswerten und einem virtuellen Host zuordnen.

Unabhängig davon, welche dieser drei Varianten konfiguriert werden soll, müssen in der Datei *httpd.conf* virtuelle Hosts erstellt werden. Hierbei handelt es sich jeweils um Gruppen von Direktiven, die am Anfang durch `<VirtualHost>` und am Ende mit `</VirtualHost>` eingeschlossen werden. Eine komplette Konfiguration für einen virtuellen Host könnte z. B. so aussehen:

```
<VirtualHost www.super-admin.org>
    ServerAdmin webmaster@super-admin.org
    DocumentRoot /var/www/super-admin
    ServerName www.super-admin.org
    ErrorLog logs/super-admin-error_log
    CustomLog logs/super-admin-access_log common
</VirtualHost>
```

Der Inhalt der Direktiven erklärt sich fast von selbst. Der wichtigste Eintrag ist jedoch `DocumentRoot`, weil diese Direktive dem Webserver sagt, in welchem Verzeichnis der zur angegebenen URL passende Content zu finden ist. Die einleitende Zeile fällt unterschiedlich aus, je nachdem, auf welche Art die Webseite identifiziert werden soll. Hier ein paar Beispiele:

- Identifikation durch die IPv4-Adresse:

```
<VirtualHost 24.215.7.162>
```

- Identifikation durch die IPv6-Adresse:

```
<VirtualHost [2a01:198:5dd:7a03:a00:27ff:fe2d:5987]>
```

- Identifikation durch den verwendeten TCP-Port:

```
<VirtualHost *:82>
```

- Identifikation durch den Hostnamen und einen TCP-Port:

```
<VirtualHost www.super-admin.org:82>
```

Beachten Sie, dass die Angabe eines TCP-Ports innerhalb eines virtuellen Hosts nicht verhindert, dass der Server weiterhin zusätzlich an Port 80 lauscht.

Die Redirect-Direktive

Redirects werden z. B. verwendet, wenn der Inhalt eines Webverzeichnisses an eine andere Stelle verschoben wurde. Der Inhalt des entsprechenden Verzeichnisses kann sich hierbei auch auf einem anderen Webserver befinden. In der Redirect-Direktive kann ein Statuscode angegeben werden. Ist das nicht der Fall, wird per Voreinstellung der Wert 302 (found) an den Client gesendet. Das lokale Verzeichnis wird relativ zum DocumentRoot angegeben, während das Ziel absolut und als URL notiert wird.

Im folgenden Beispiel wird das lokale Unterverzeichnis */pdf* an die URL *http://www.super-admins.org/dokumente* umgeleitet. An den Client wird der Statuscode 301 (moved permanently) ausgegeben. Diese Statusinformation ist besonders für Suchmaschinen interessant, weil eine dauerhafte Änderung (im Gegensatz zu einer temporären Änderung) im Index der Suchmaschinen berücksichtigt werden sollte.

```
Redirect permanent /pdf http://www.super-admins.org/dokumente
```

Für den Statuscode eines Redirects können folgende Argumente verwendet werden:

- `permanent`: Der Server sendet den Statuscode 301 (moved permanently), um anzuzeigen, dass die Ressource dauerhaft unter der neuen URL erreichbar ist.
- `temp`: Der Server gibt den Statuscode 302 (found) zurück. Dies ist die Standardeinstellung, wenn Sie kein Statusargument angeben.

- `seeother`: Der Server gibt den Statuscode 303 (see other) zurück. Das bedeutet, dass die ursprüngliche Ressource ersetzt wurde.
- `gone`: Der Server sendet den Statuscode 410 (gone). Die angeforderte Ressource ist auf dem Server dauerhaft nicht mehr erreichbar. Es ist keine neue URL bekannt.

Sie sollten übrigens nicht versuchen, ein Unterverzeichnis des ursprünglichen Webverzeichnisses als neues Ziel in einem Redirect zu verwenden. Diese Methode wird von der Redirect-Direktive nicht unterstützt und führt zu einer Fehlermeldung im Browser des Clients.

208.2 Apache für HTTPS konfigurieren

Wichtung: 3

Beschreibung: Die Prüflinge sollten dazu in der Lage sein, einen Webserver für die Nutzung von HTTPS zu konfigurieren.

Wichtigste Wissensgebiete:

- SSL-Konfigurationsdateien, -Begriffe und -Dienstprogramme
- Fähigkeit, einen privaten Serverschlüssel und einen Zertifikats-Request (CSR) für eine kommerzielle Zertifizierungsstelle zu erstellen
- Fähigkeit, ein selbst signiertes Zertifikat mithilfe einer privaten CA zu erstellen
- Fähigkeit, ein Zertifikat und einen Schlüssel zu installieren
- Konfiguration virtueller Hosts mithilfe von SNI
- Wissen über das Verhalten virtueller Hosts in Verbindung mit SSL
- Sicherheitsbelange bei der SSL-Nutzung, deaktivieren unsicherer Protokolle und Verschlüsselungen

Liste wichtiger Dateien, Verzeichnisse und Anwendungen:

- Apache-Konfigurationsdateien
- */etc/ssl/* , /etc/pki/**
- `openssl`,`CA.pl`
- `SSLEngine`, `SSLCertificateKeyFile`, `SSLCertificateFile`
- `SSLCACertificateFile`, `SSLCACertificatePath`
- `SSLProtocol`, `SSLCipherSuite`, `ServerTokens`, `ServerSignature`, `TraceEnable`

Allgemeines

Auf den letzten Seiten haben Sie eine Menge über die Grundkonfiguration eines Apache Webservers erfahren. In der Praxis werden Sie aber noch weitere Funktionen

eines Webservers benötigen. Webserver, die von Internet Service Providern betrieben werden, hosten normalerweise mehrere Webseiten. Es ist nicht ungewöhnlich, dass ein einziger Apache Webserver die Seiten von fünfzig und mehr Kunden hostet. Zu diesem Zweck werden virtuelle Hosts auf den Servern konfiguriert.

Eine weitere häufig benötigte Funktionalität ist SSL (Secure Socket Layer). Mit SSL wird die Authentizität einer Webseite garantiert, die Integrität gewahrt und der Inhalt während der Übertragung durch Verschlüsselung gesichert. Damit Sie HTTPS verwenden können, muss das entsprechende Modul SSL installiert sein.

Wenn eine Webseite eine Authentifizierung verlangt oder vertrauliche Daten übermittelt, sollte die Verbindung verschlüsselt erfolgen. Es wäre sonst zu befürchten, dass jemand die Verbindung abhorcht und dabei die Authentifizierungsdaten oder den vertraulichen Inhalt der Verbindung abfängt. In einer solchen Situation kommt HTTPS zum Einsatz. Im Grunde genommen handelt es sich hier immer noch um eine HTTP-Kommunikation, aber die Datenpakete werden in einer SSL-Verbindung gesichert übermittelt.

Das gesamte Konstrukt basiert auf einer Infrastruktur für öffentliche Schlüssel (Public Key Infrastructure, PKI). Da hier Zertifikate im Spiel sind, können Sie HTTPS nicht nur für die Verschlüsselung verwenden. Es ist auch eine gegenseitige Authentifizierung von Webserver und Client möglich, wobei in der Praxis meist nur die Authentifizierung des Servers gegenüber dem Client implementiert wird. Schließlich müssen Sie als Kunde wissen, ob Sie einer Webseite trauen können, wenn Sie z. B. Kreditkartentransaktionen durchführen wollen.

Wenn Sie eine mit SSL gesicherte Webseite konfigurieren, auf die vom Internet aus zugegriffen wird, sollten Sie über ein Zertifikat für diese Seite verfügen, das von einer öffentlichen Zertifizierungsstelle (z. B. GlobalSign, Thawte, Verisign) ausgestellt wurde. Ansonsten erhalten Benutzer beim Zugriff auf die Webseite eine Warnmeldung und meiden die Seite möglicherweise. Damit das Zertifikat eines Webservers von einem Client ohne Warnmeldungen akzeptiert wird, müssen drei Kriterien erfüllt sein:

- Das Zertifikat muss von einer vertrauten Zertifizierungsinstitution stammen.
- Das Zertifikat muss gültig sein (nicht abgelaufen oder zurückgezogen).
- Der Antragstellername im Zertifikat muss mit der URL übereinstimmen, die ein Benutzer in den Browser eingibt.

Das bedeutet, dass Sie problemlos zu Testzwecken ein Zertifikat verwenden können, das Sie selbst ausgestellt haben. Wenn Sie SSL verwenden, um so etwas wie eine Verwaltungswebseite (z. B. phpMyAdmin, CUPS, Webmin o. Ä.) abzusichern, wollen Sie wahrscheinlich lediglich sicherstellen, dass die Kommunikation verschlüsselt erfolgt und nicht die Echtheit des Zielservers überprüfen. Auch in solchen Fällen können Sie

ohne weiteres selbst signierte Zertifikate verwenden und entsprechende Warnmeldungen des Browsers ignorieren.

Konfiguration von SSL mittels openssl

Die Konfiguration auf den folgenden Seiten basiert diesmal auf einer Apache-Installation mittels `yum`. Auf diese Art lernen Sie gleichzeitig eine andere Variante von Apache kennen. Der wesentliche Unterschied zum manuell installierten Webserver sind die Positionen der Verzeichnisse des Servers im Dateisystem. Installieren Sie Apache mit:

```
[root@arch-cent ~]# yum install httpd
```

Wenn Sie CentOS verwenden, wird das Modul *mod_ssl* automatisch mitinstalliert. Bei einem System, auf dem Fedora ausgeführt wird, müssen Sie SSL zusätzlich installieren:

```
[root@arch-fc /]# yum install mod_ssl
```

Bei den Red Hat-basierten Systemen finden Sie die Konfigurationsdateien im Verzeichnis */etc/httpd*. Die Datei *httpd.conf* liegt jeweils im Verzeichnis */etc/httpd/conf*. Alle anderen Verzeichnisse sind durch Lesen der Hauptkonfigurationsdatei problemlos zu ermitteln (z. B. `ServerRoot` und `DocumentRoot`).

Wenn Sie mittels `yum` einen Apache Webserver installiert haben, verfügt dieser bereits über ein selbst signiertes Zertifikat. Sie sollten aber natürlich wissen, wie man ein solches Zertifikat mit `openssl` selbst erzeugt und an eine Webseite bindet. Sie finden hier wieder eine Schritt-für-Schritt-Anleitung. Denken Sie aber daran, dass diesem Zertifikat nicht öffentlich vertraut wird. Erstellen Sie zunächst ein Unterverzeichnis für das Zertifikat und den privaten Schlüssel und wechseln Sie anschließend hinein.

```
[root@arch-cent /]# mkdir /etc/httpd/ssl
[root@arch-cent /]# cd /etc/httpd/ssl
```

Verwenden Sie anschließend `openssl`, um das Zertifikat und den Schlüssel zu generieren. Die Benutzereingaben sind wieder fett gedruckt:

```
[root@arch-cent ssl]# openssl req -new -x509 -nodes -out
arch-cent.homelinux.net.crt -keyout arch-cent.homelinux.net.key
Generating a 1024 bit RSA private key
.................++++++
...................++++++
writing new private key to 'arch-cent.homelinux.net.key' ---
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. -----
Country Name (2 letter code) [GB]:DE
State or Province Name (full name) [Berkshire]:Germany
Locality Name (eg, city) [Newbury]:Berlin
Organization Name (eg, company) [My Company Ltd]:Maassen
Organizational Unit Name (eg, section) []:.
Common Name (eg, your name or your server's hostname)
   []:arch-cent.homelinux.net
Email Address []:harald@nwa-net.de
```

Es sollten nun zwei Dateien im aktuellen Verzeichnis liegen. Die Datei mit der Erweiterung *crt* enthält das Zertifikat mit dem öffentlichen Schlüssel, während die Datei mit der Erweiterung *key* den privaten Schlüssel enthält:

```
[root@arch-cent ssl]# ls -l
insgesamt 16
-rw-r--r-- 1 root root 1277 20. Jun 21:50 arch-cent.homelinux.net.crt
-rw-r--r-- 1 root root  887 20. Jun 21:50 arch-cent.homelinux.net.key
```

Damit der Webserver das Zertifikat und den Schlüssel auch verwendet, müssen Sie die entsprechenden Direktiven der Datei *httpd.conf* anpassen. Im Fall von CentOS bzw. Fedora wurde der hierfür zuständige Bereich per `include`-Anweisung in die Datei */etc/httpd/conf.d/ssl.conf* ausgelagert. Öffnen Sie diese Datei mit einem Editor und suchen Sie nach den Einträgen `SSLCertificateFile` und `SSLCertificateKeyFile`. Passen Sie die Dateien an Ihr eigenes Zertifikat und den privaten Schlüssel an:

```
SSLCertificateFile /etc/httpd/ssl/arch-cent.homelinux.net.crt
SSLCertificateKeyFile /etc/httpd/ssl/arch-cent.homelinux.net.key
```

Damit die Konfigurationseinstellungen sofort wirksam werden, sollten Sie Apache neu starten.

```
[root@arch-cent ssl]# apachectl restart
```

Greifen Sie nun mit einem beliebigen Webbrowser auf die abgesicherte Webseite zu, indem Sie das Präfix *https://* angeben. Sie sollten dann eine Sicherheitswarnung erhalten, weil das vom Webserver verwendete Zertifikat nicht von einer vertrauenswürdigen Zertifizierungsstelle ausgestellt worden ist. Die Warnung können Sie in diesem Fall natürlich ignorieren. Nur wenn Sie eine abgesicherte Webseite anderen Benutzern zur Verfügung stellen, sollten Sie ein Zertifikat von einer öffentlichen Zertifizierungsstelle erwerben.

Server Name Indication (SNI)

Vor Apache v2.2.12 mit OpenSSL v0.9.8 konnte pro IP-Adresse nur jeweils eine SSL-Webseite gebunden werden. Das bedeutete für öffentlich zugängliche HTTPS-Webseiten, dass diese nicht preisgünstig gehostet werden konnten, weil immer eine eigene öffentliche IP-Adresse nötig war. Dieses Manko behebt *Server Name Indication (SNI)*. Das Einzige, was Sie tun müssen, ist mehrere virtuelle Hosts zu erstellen, wie Sie es bereits von normalen HTTP-Seiten her kennen. Sie können zu diesem Zweck weitere Konfigurationsdateien erstellen oder eine bestehende Konfiguration modifizieren. Ein Beispiel für zwei HTTPS-Webseiten sehen Sie hier:

```
NameVirtualHost *:443

<VirtualHost *:443>
 ServerName www.beispiel.com
 DocumentRoot /var/www/beispiel
 SSLEngine on
 SSLCertificateFile /etc/ssl/www_beispiel_com.crt
 SSLCertificateKeyFile / etc/ssl/www_beispiel_com.key
 SSLCertificateChainFile / etc/ssl/MyCertCA.crt
</VirtualHost>

<VirtualHost *:443>
 ServerName www.exampel.org
 DocumentRoot /var/www/exampel
 SSLEngine on
 SSLCertificateFile / etc/ssl/www_exampel_org.crt
 SSLCertificateKeyFile / etc/ssl/www_exampel_org.key
 SSLCertificateChainFile / etc/ssl/MyCertCA.crt
</VirtualHost>
```

SSL-Zertifikate mittels CA.pl erstellen

Sie können die Zertifizierungsstelle und die Zertifikate, die Sie benötigen, auch mithilfe des Pearl-Skripts `CA.pl` erstellen. Hierbei handelt es sich um ein Frontend für das Kommando `openssl`. Sie werden sehen, dass bei der Verwendung von `CA.pl` kaum Optionen oder Parameter angegeben werden müssen. Vielmehr handelt es sich um ein interaktives Skript, das alle benötigten Angaben vom Benutzer erfragt und anschließend die daraus resultierenden `openssl`-Kommandos zusammenbaut.

Erstellen der CA-Hierarchie

Wo sich das Skript `CA.pl` befindet, ist von der verwendeten Distribution abhängig. Verwenden Sie `locate` oder `find`, um es zu finden. Sie können innerhalb des Skripts

Änderungen vornehmen, um es an Ihre Bedürfnisse anzupassen. Das gilt z. B. für die Gültigkeitsdauer von ausgestellten Zertifikaten. `CA.pl` arbeitet mit relativen Verzeichnisangaben. Das sollten Sie bei der Erstellung einer neuen CA berücksichtigen. Wenn Sie keine Änderungen an dem Skript vornehmen, erstellt `CA.pl` unterhalb des aktuellen Verzeichnisses eine Verzeichnisstruktur mit dem Namen *demoCA*. Um dieses Verhalten zu ändern, sollten Sie im Skript die folgende Zeile ändern:

von `$CATOP="./demoCA";` nach `$CATOP="/etc/pki";`

Dasselbe gilt für zwei Zeilen in der Konfigurationsdatei */etc/ssl/openssl.cnf*, die denselben Inhalt aufweisen:

aus `dir=./demoCA` wird `dir=/etc/pki`

So wird das Verzeichnis */etc/pki* gleich automatisch zu einem sinnvollen zentralen Speicherort für zertifikatsbezogene Daten.

Zum Erstellen der CA übergeben Sie einfach die Option `-newca`. Im folgenden Beispiel sind wieder der Übersichtlichkeit wegen alle Benutzereingaben fett gedruckt. Aus Platzgründen wurden einige für das Verständnis unwichtige Zeilen entfernt. Es müssen übrigens nicht alle Felder zwingend ausgefüllt werden:

```
root@archangel:/# /usr/lib/ssl/misc/CA.pl -newca
CA certificate filename (or enter to create)
Making CA certificate ...
Generating a 2048 bit RSA private key
writing new private key to './demoCA/private/cakey.pem'
Enter PEM pass phrase: ****** (wird nicht angezeigt)
Verifying - Enter PEM pass phrase: ****** (wird nicht angezeigt)
-----
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.
-----
Country Name (2 letter code) [AU]:DE
State or Province Name (full name) [Some-State]:Germany
Locality Name (eg, city) []:Berlin
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Maassen
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:Harald Maassen
Email Address []:harald@lpic-2.de
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:sichtbares_passwort
An optional company name []:Maassen
Using configuration from /usr/lib/ssl/openssl.cnf
```



```
Enter pass phrase for /etc/pki/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
```

Die Zertifizierungsstelle ist jetzt einsatzbereit. Sehen Sie sich den Inhalt des Verzeichnisses */etc/pki* einmal genauer an. Hier finden Sie z. B. die Datei *cacert.pem*, die Sie benötigen, wenn ein Browser Ihrer Zertifizierungsstelle vertrauen soll. Importieren Sie die Datei in die vertrauenswürdigen Stammzertifizierungsstellen des Browsers.

Der nächste Schritt ist die Erstellung einer Zertifikatsanforderung. Beachten Sie bitte, dass `CA.pl` immer im aktuellen Verzeichnis operiert und auch hier die entsprechenden Dateien generiert. Da Sie diese Dateien jedoch ohnehin später an einen anderen Ort im Dateisystem verschieben werden, ist dieses Verhalten in Ordnung. Führen Sie den Request wie folgt durch:

```
root@archangel:/# /usr/lib/ssl/misc/CA.pl -newreq
Generating a 2048 bit RSA private key
writing new private key to 'newkey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
Country Name (2 letter code) [AU]:DE
State or Province Name (full name) [Some-State]:Germany
Locality Name (eg, city) []:Berlin
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Maassen
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:www.lpic-2.de
Email Address []:harald@lpic-2.de
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:klartext challenge!
An optional company name []:Maassen
Request is in newreq.pem, private key is in newkey.pem
```

Auch diesmal wurden weniger wichtige Zeilen entfernt. Die letzte Zeile zeigt das Ergebnis des Kommandos. Es wurden ein privater Schlüssel (*newkey.pem*) und eine Anforderungsdatei (*newreq.pem*) generiert.

Hinweis

Wenn Sie ein Zertifikat von einer kommerziellen Zertifizierungsstelle anfordern müssen, benötigen Sie ebenfalls die Dateien *newkey.pem* und *newreq.pem*. Die folgenden Schritte würde dann der Betreiber dieser CA auf einem seiner Systeme durchführen und Ihnen das fertige Zertifikat übermitteln.

Lassen Sie die beiden Dateien zunächst da, wo sie jetzt sind, damit Sie den Request signieren können:

```
root@archangel:/# /usr/lib/ssl/misc/CA.pl -signreq
Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for /etc/pki/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number:
            95:1a:27:fb:5c:3c:ce:a5
        Validity
            Not Before: Jul  9 14:13:56 2013 GMT
            Not After : Jul  9 14:13:56 2014 GMT
        Subject:
            countryName               = DE
            stateOrProvinceName       = Germany
            localityName              = Berlin
            organizationName          = Maassen
            commonName                = www.lpic-2.de
            emailAddress              = harald@lpic-2.de
Signed certificate is in newcert.pem
```

Auch hier wurden wieder unwesentliche Zeilen entfernt. Das neue Zertifikat (*newcert.pem*) finden Sie im aktuellen Verzeichnis.

Um das Zertifikat passwortgeschützt zu sichern, können Sie es mit PKCS12 (PKCS = Public Key Cryptography Standards) sichern. Sie können diese Datei später für den Import auch in fremde Betriebssysteme und Programme verwenden. Dabei werden Sie dann zur Eingabe des bei der Erstellung angegebenen Passworts aufgefordert. Erstellen Sie die PKSC12-Datei so:

```
root@archangel:/# /usr/lib/ssl/misc/CA.pl -pkcs12 "WebZerti"
Enter pass phrase for newkey.pem:
Enter Export Password:
Verifying - Enter Export Password:
PKCS #12 file is in newcert.p12
```

Wie Sie sehen, ist die Datei *newcert.p12* erstellt worden.

Die Benutzung des erstellten Zertifikats und des privaten Schlüssels funktioniert genauso, als hätten Sie das Kommando `openssl` zu deren Erstellung angewendet. Eine Wiederholung der entsprechenden Apache-Konfiguration soll deshalb an dieser Stelle nicht erfolgen. Sie sollten natürlich die generierten Dateien jeweils verschieben und aussagekräftig umbenennen, weil das Skript `CA.pl` immer dieselben Dateinamen erstellt.

Direktiven des Moduls mod_ssl und andere Sicherheitseinstellungen

Das Apache-Modul *mod_ssl* bringt viele zusätzliche Direktiven mit. Einige dieser Direktiven haben Sie bereits selbst angewendet, nämlich als Sie den Serverschlüssel und das Zertifikat in die Apache-Konfiguration eingebunden haben. Für die Prüfung sollten Sie aber weitere Direktiven aus diesem Umfeld kennen.

- `SSLEngine` aktiviert oder deaktiviert SSL. Typische Platzierung:

  ```
  <VirtualHost _default_:443>
  SSLEngine on
  ...
  </VirtualHost>
  ```

- `SSLCertificateKeyFile` enthält den vollständigen Pfad zum privaten Serverschlüssel.
- `SSLCertificateFile` enthält den vollständigen Pfad zum Zertifikat des Servers.
- `SSLCertificateChainFile` enthält den Pfad zu einer optionalen Vertrauenskette. Sie können aus mehreren CA-Zertifikaten, von denen sich letztendlich Ihr Serverzertifikat ableitet, eine einzige Datei zusammenstellen und diese hier hinterlegen. Diese Datei enthält natürlich nur die öffentlichen Schlüssel der CAs.
- `SSLCACertificateFile` ähnelt `SSLCertificateChainFile`, enthält aber die Vertrauenskette der CAs, die die Zertifikate für die Clients ausgestellt haben.
- `SSLCACertificatePath` ähnelt `SSLCACertificateFile`, zeigt aber auf ein Verzeichnis, das die Zertifikate der CAs enthält und nicht auf eine Vertrauenskette in Form einer Datei.
- `SSLProtocol` legt fest, welche SSL-Protokolle erlaubt sind. Infrage kommen hier `SSLv2`, `SSLv3`, `TLSv1`, `TLSv1.1`, `TLSv1.2` oder `All`. Die Groß-/Kleinschreibweise der Protokolle muss berücksichtigt werden!
- `SSLCipherSuite` ist eine durch Doppelpunkte getrennte Auflistung der kryptographischen Algorithmen, die der Server verwenden darf. Je »strenger« die Kette ist, desto mehr Rechenzeit wird benötigt.

Die folgenden Direktiven gehören nicht zum Modul *mod_ssl*, enthalten aber wichtige Sicherheitseinstellungen:

- `ServerTokens` legt fest, welche Informationen Apache über sich selbst an den Client sendet. In der strengsten Einstellung (`Prod`) wird lediglich `Server: Apache` übermittelt. Wird diese Direktive mit `All` konfiguriert, informiert Apache den Client über seine genaue Version, das installierte Betriebssystem und die geladenen Module. Eine solche Konfiguration sollten Sie natürlich vermeiden.
- `ServerSignature` erlaubt es Ihnen automatisch, den Fehlermeldungen eine Fußnote hinzuzufügen. Das ist hilfreich, wenn Sie bei verketteten Konfigurationen herausfinden wollen, welcher Server eine Fehlermeldung generiert hat.

- `TraceEnable` sollte aus Sicherheitsgründen mit `off` konfiguriert werden. Dadurch wird unterbunden, dass potenzielle Angreifer zusätzliche Informationen über den Webserver erlangen können, z. B. ob ein Reverse-Proxy zwischengeschaltet ist oder ob der Server direkt antwortet.

[!]

Achtung

Bitte deinstallieren Sie Apache jetzt noch nicht, nur weil dieses Thema nun abgeschlossen ist. Sie können ihn später noch im Zusammenhang mit *nginx* verwenden!

208.3 Implementieren von Squid als Cache-Proxy

Wichtung: 2

Beschreibung: Die Prüflinge sollten dazu in der Lage sein, einen Proxy-Server zu installieren und zu konfigurieren, einschließlich der Zugriffsregeln, der Authentifizierung und der Ressourcennutzung.

Wichtigste Wissensgebiete:

- Squid 3.x-Konfigurationsdateien, -Begriffe und -Dienstprogramme
- Methoden zur Zugriffsbeschränkung
- Methoden zur Client-Benutzerauthentifikation
- Aufbau und Inhalt von ACLs in den Squid-Konfigurationsdateien

Liste wichtiger Dateien, Verzeichnisse und Anwendungen:

- *squid.conf*
- `acl`
- `http_access`

Allgemeines

Ein Proxy-Server wird verwendet, um Webinhalte stellvertretend für Clients anzufordern, zwischenzuspeichern und anschließend an die Client-Computer auszuliefern. Ursprünglich wurden Proxy-Server in Unternehmensnetzwerken hauptsächlich eingesetzt, um die Bandbreite des Internetzugangs optimal nutzen zu können und Client-Anfragen mit zwischengespeicherten Webseiten schneller versorgen zu können. Die Möglichkeit, den Zugriff auf bestimmte Webseiten zu sperren oder den Zugriff auf das Web auf bestimmte Client-Computer zu beschränken, spielte zunächst eine untergeordnete Rolle.

Heutzutage hat sich die Aufgabe eines Proxys eher in Richtung Zugriffssteuerung verlagert. Da Webinhalte mehr und mehr dynamisch generiert und Webseiten immer schneller aktualisiert werden, ist die Zwischenspeicherung oftmals nicht sinnvoll. Aufgrund der inzwischen verfügbaren Bandbreiten heutiger Internetzugänge ist das Caching von Webseiten außerdem nicht mehr so wichtig wie früher. Trotzdem hat ein Proxy immer noch seine Daseinsberechtigung. Schließlich kann er verhindern, dass die Mitarbeiter eines Unternehmens ihre Arbeitszeit verschwenden und dass Kinder jugendgefährdendes Material aus dem Internet herunterladen.

Installation des Squid Proxy Servers

Da die fertigen Squid-Proxys, die als Pakete für die verschiedenen Linux-Distributionen vorliegen, sich nicht wesentlich voneinander unterscheiden, können Sie diesmal auch auf ein solches Paket zurückgreifen. Die Konfiguration eines von Hand installierten Squids ist allerdings übersichtlicher und deshalb für Proxy-Einsteiger leichter zu lesen.

Wenn Sie IPv6-Unterstützung für Internetzugriffe implementieren wollen, sollten Sie mindestens die Squid-Version 3.1 installieren. Die Webseite des Projekts finden Sie unter *http://www.squid-cache.org*. Hier finden Sie auch Informationen über Bezugsquellen der aktuellen Squid-Versionen. Die hier Schritt für Schritt dokumentierte Installation von Squid 3.2.0.9 wurde auf einem Debian 6.0-System durchgeführt.

Die ersten Arbeitsschritte sind Routinearbeiten: Arbeitsverzeichnis erstellen, Quellpaket herunterladen, dekomprimieren und entpacken.

```
root@arch-deb:~# mkdir /usr/src/squid
root@arch-deb:~# cd /usr/src/squid
root@arch-deb:/usr/src/squid# wget ftp://ftp.fu-berlin.de/unix/www/squid/
archive/3.2/squid-3.2.0.9.tar.bz2
root@arch-deb:/usr/src/squid# bunzip2 squid-3.2.0.9.tar.bz2
root@arch-deb:/usr/src/squid# tar -xvf squid-3.2.0.9.tar
root@arch-deb:/usr/src/squid# cd squid-3.2.0.9/
```

Das Konfigurationsskript benötigt für seine Ausführung, zumindest unter Debian, das Paket *build-essential*. Sie sollten dieses Paket also spätestens jetzt installieren:

```
root@arch-deb:/usr/src/squid/squid-3.2.0.9# apt-get install build-essential
```

Übergeben Sie dem Konfigurationsskript das Zielverzeichnis des Programms mit der Option `--prefix`.

```
root@arch-deb:/usr/src/squid/squid-3.2.0.9# ./configure --prefix=/usr/local/
squid
```

Wenn die Konfiguration fertig ist, können Sie wie gewohnt kompilieren und installieren:

```
root@arch-deb:/usr/src/squid/squid-3.2.0.9# make
root@arch-deb:/usr/src/squid/squid-3.2.0.9# make install
```

Nach der Installation finden Sie den kompletten Server inklusive Konfigurationsdateien und Verzeichnis für Logfiles unterhalb von */usr/local/squid*. Wenn Sie zunächst mit dieser rohen Konfiguration arbeiten wollen, müssen Sie das Unterverzeichnis *logs* für Squid beschreibbar machen. Da Squid per default unter dem Sicherheitskontext von nobody läuft, können Sie einfach dem User nobody die Eigentümerschaft an dem Verzeichnis für die Logdateien übertragen:

```
root@arch-deb:/# chown nobody /usr/local/squid/var/logs/ -R
```

Da es sich lediglich um eine Testumgebung handelt, in der Sicherheit keine große Rolle spielt, kann diese unsaubere, aber einfache Konfiguration verwendet werden.

Bevor Sie den Server in Betrieb nehmen können, muss der Cache einmalig initialisiert werden:

```
root@arch-deb:/# /usr/local/squid/sbin/squid -z
2011/06/25 19:25:35 kid1| Creating Swap Directories
```

Beachten Sie bitte, dass in der Standardeinstellung nur im Arbeitsspeicher zwischengespeichert wird. Um einen Festplattencache einzurichten, muss zunächst die Konfigurationsdatei *squid.conf* modifiziert werden. Sie finden auf den nächsten Seiten genaue Informationen über diese Konfigurationsdatei. Wenn der Cache fertig initialisiert worden ist, können Sie den Proxy testweise starten:

```
root@arch-deb:/# /usr/local/squid/sbin/squid
```

Es sind keine weiteren Optionen für den Start erforderlich. Squid läuft standardmäßig als Daemon. Warten Sie ein paar Sekunden und prüfen Sie dann, ob der Server nun läuft (`ps aux|grep squid` oder `pidof squid` sind hier hilfreich). Sollte es wider Erwarten zu Problemen kommen, konsultieren Sie die soeben erstellte Protokolldatei *cache.log*. Sollte diese Datei leer sein, gibt es ein Berechtigungsproblem mit dieser Datei.

Im Lieferumfang von Squid ist ein Testprogramm enthalten, mit dem Sie die Funktionsfähigkeit des Proxys testen können. Mit dem folgenden Kommando greifen Sie über den lokal installierten Proxy (localhost TCP-Port 3128) auf eine Webseite zu. Die Ausgabe des Kommandos sollte der HTML-Code der Webseite sein:

```
root@arch-deb:/# /usr/local/squid/bin/squidclient http://www.lpi.org
HTTP/1.1 200 OK
Date: Sun, 26 Jun 2011 14:08:06 GMT
```



```
Server: Apache/1.3.33 (Debian GNU/Linux) PHP/4.3.10-22 mod_ssl/2.8.22 OpenSSL/
0.9.7e mod_perl/1.29
X-Powered-By: eZ publish
Set-Cookie: eZSESSID=aa69db352a0ec2e3fb9ab1bbe347135b; path=/
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Cache-Control: no-cache, must-revalidate
```

Konfiguration

Damit Sie die Ergebnisse Ihrer Konfigurationsarbeiten überprüfen können, sollten Sie zunächst einen Webbrowser Ihrer Wahl so konfigurieren, dass er durch den Squid-Proxy hindurch auf das Internet zugreift. Wenn Sie Firefox oder Iceweasel unter Linux verwenden, finden Sie die entsprechende Registerkarte unter Bearbeiten • Einstellungen • Erweitert • Netzwerk • Einstellungen. Tragen Sie hier die Adresse des Proxy-Servers und den TCP-Port 3128 ein.

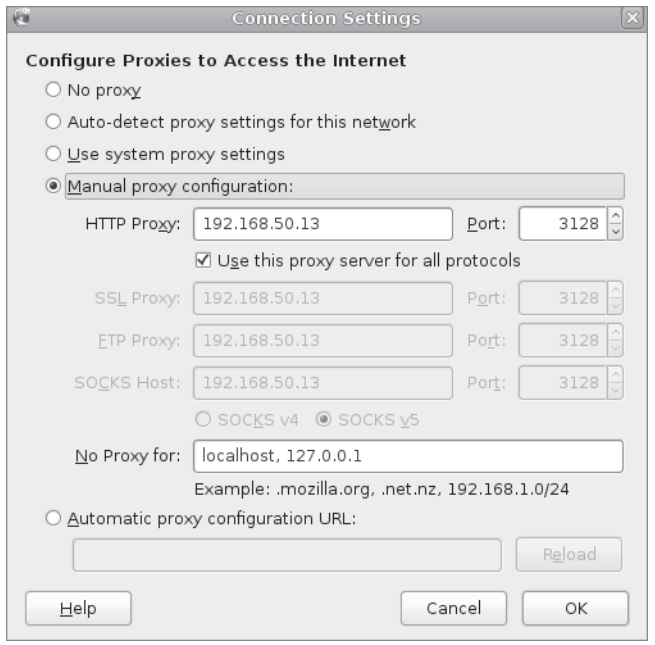

Abbildung 208.1 Tragen Sie die Adresse des Squid-Proxys und die zugehörige Portnummer hier ein.

Zur Fehlervermeidung sollten Sie überprüfen, ob der Squid Proxy Server selbst über eine funktionierende Internetverbindung verfügt. Wenn dennoch Fehler auftreten, lesen Sie bitte unbedingt die entsprechende Fehlermeldung im Browserfenster.

Die Hauptkonfigurationsdatei des Squid-Proxys ist die Datei *squid.conf*. Sie finden diese Konfigurationsdatei normalerweise unter */etc/squid.conf* oder */etc/squid3/*

squid.conf. Wenn Sie Squid aus einem tar-Ball heraus installiert haben, lautet der Dateipfad */usr/local/squid/etc/squid.conf*. Sie sollten bezüglich der Grundkonfiguration zumindest die folgenden Optionen in dieser Datei kennen:

- `http_port` legt fest, an welchem TCP-Port Squid lauschen soll. Der Standardwert ist 3128.
- `cache_mem` gibt an, wie viel Arbeitsspeicher für die Zwischenspeicherung verwendet werden darf (z. B. `cache_mem 512MB`). In der Standardeinstellung werden lediglich 8 MB verwendet.
- `cache_dir` konfiguriert den festplattenseitigen Cache des Proxys. Hierbei werden (in dieser Reihenfolge) der Typ des Caches, der Verzeichnisname, die Cachegröße in Megabytes, die Anzahl der Unterverzeichnisse erster Ebene und die Anzahl der Unterverzeichnisse zweiter Ebene festgelegt. Beispiel:

  `cache_dir ufs /usr/local/squid/var/cache 1024 16 256`

  In diesem Beispiel wird ein Cache vom Typ UFS verwendet. Das Hauptverzeichnis des Caches ist */usr/local/squid/var/cache* und wurde auf 1 GB beschränkt. In diesem Verzeichnis werden 16 Unterverzeichnisse und darin wiederum jeweils 256 Unterverzeichnisse erstellt.
- `reply_body_max_size` limitiert die Größe eines reply-body und hindert Benutzer daran, übergroße Dateien aus dem Internet herunterzuladen.
- `access_log` legt hauptsächlich den Pfad zu der Protokolldatei fest, in der Client-Zugriffe protokolliert werden.

Cache auf Festplatte einrichten

In der Standardkonfiguration verwendet Squid keinen Festplattencache, wenn Sie den Proxy aus einem tar-Ball heraus installiert haben. Damit der Cache initialisiert werden kann, benötigt Squid Schreibrechte auf das Verzeichnis, in dem die Verzeichnishierarchie für den Cache erstellt werden soll. Da in der Testumgebung Sicherheit keine Rolle spielt, können Sie einfach dieses Kommando verwenden:

```
root@arch-deb:/# chown nobody /usr/local/squid/var/cache/ -R
```

In der Datei *squid.conf* sollten Sie die Parameter für `cache_dir` Ihren persönlichen Bedürfnissen anpassen. Besonders der erste numerische Wert, der die Cachegröße in Megabytes festlegt, dürfte hier von Interesse sein. Die anderen Parameter von `cache_dir` wurden bereits im vorangehenden Abschnitt erläutert.

```
cache_dir ufs /usr/local/squid/var/cache 20000 16 256
```

Beenden Sie nun Squid, falls es noch laufen sollte, und führen Sie anschließend das folgende Kommando aus, um den Cache neu zu initialisieren:


```
root@arch-deb:/usr/local/squid/var# /usr/local/squid/sbin/squid -z
root@arch-deb:/usr/local/squid/var# 2011/06/26 15:24:44 kid1|
2011/06/26 15:24:44 kid1| Creating Swap Directories
2011/06/26 15:24:44 kid1| Making directories in /usr/local/squid/var/cache/00
2011/06/26 15:24:44 kid1| Making directories in /usr/local/squid/var/cache/01
2011/06/26 15:24:44 kid1| Making directories in /usr/local/squid/var/cache/02
...
2011/06/26 15:24:44 kid1| Making directories in /usr/local/squid/var/cache/0F
```

Die Initialisierung ist abgeschlossen und Sie können Squid nun wieder normal starten und verwenden.

Zugriffssteuerung mithilfe von ACLs

Die Zugriffssteuerung auf den Squid-Proxy wird über ACLs vorgenommen. In diesen ACLs können Sie zunächst einige Definitionen vornehmen. Im weiteren Verlauf der Konfiguration können Sie dann auf diese Definitionen (ACLs) zurückgreifen, um den Zugriff jeweils zu erlauben oder zu verweigern. Typischerweise enthalten ACLs Gruppen von Quell-IP-Adressen, Ziel-IP-Adressen, URL-Listen oder Ports. Namen von Zugriffssteuerungslisten können mehrfach verwendet werden. Wenn Sie Squid 3.x selbst kompiliert haben, gibt es z. B. drei ACL-Einträge in der Datei *squid.conf*, die für die Definition von privaten IPv4-Netzwerkadressen verwendet werden:

```
acl localnet src 10.0.0.0/8
acl localnet src 172.16.0.0/12
acl localnet src 192.168.0.0/16
```

Bis hierhin handelt es sich lediglich um eine Deklaration, in der alle privaten IPv4-Adressen als mögliche Quelladressen (`src`) der ACL `localnet` hinzugefügt werden. In einem ähnlichen Eintrag wird der lokale Computer selbst definiert:

```
acl localhost src 127.0.0.1/32 ::1
```

Ein paar Zeilen später wird dann für den lokalen Computer und die privaten IPv4-Netzwerke der Zugriff auf den Proxy erlaubt:

```
http_access allow localnet
http_access allow localhost
```

Hinweis

Wenn Sie Squid paketbasiert installieren, wird in den meisten Fällen per Voreinstellung überhaupt kein Zugriff auf den Proxy möglich sein. Sie müssen in diesem Fall die entsprechenden ACLs von Hand erstellen und den Zugriff erlauben.

Sie können den Zugriff auf Webseiten über reguläre Ausdrücke filtern. Dazu benötigen Sie zunächst einen geeigneten ACL-Eintrag in der Datei *squid.conf*:

```
acl sperrliste url_regex "/usr/local/squid/etc/sperrliste"
```

Den zugehörigen Berechtigungseintrag sollten Sie vor anderen Berechtigungseinträgen positionieren, weil vorangehende `allow`-Einträge sonst frühzeitig Zugriff gewähren, sodass der Filter gar nicht erst greift.

```
http_access deny sperrliste
http_access allow localnet
http_access allow localhost
```

Sie müssen jetzt nur noch die Datei */usr/local/squid/etc/sperrliste* anlegen und Schlagwörter (jeweils in einer eigenen Zeile) eingeben, die in den URLs nicht vorkommen dürfen. Nach einem Neustart von Squid können Sie die Konfiguration mit einem Browser testen.

Zum Abschluss der Zugriffssteuerung sollte immer der Zugriff für alle anderen Clients verweigert werden. Ansonsten könnten böswillige Benutzer vom Internet aus Ihren Proxy verwenden, um ihre Herkunft zu verschleiern und unter Ihrer Identität z. B. Webserver attackieren. Die letzte Regel sollte also immer diese sein:

```
http_access deny all
```

Praxistipp

Wenn Sie den Zugriff auf einen Proxy häufig umkonfigurieren müssen oder wenn eine recht komplexe Zugriffskonfiguration existiert, sollten Sie ein Frontend verwenden, um nicht den Überblick zu verlieren. Eine gute Wahl ist hierfür SquidGuard. Dieses Produkt ist aber nicht prüfungsrelevant.

Benutzerauthentifizierung

Sie können den Zugriff auf einen Proxy einschränken, indem Sie eine Authentifizierung konfigurieren. Die Benutzer werden dann beim Zugriff auf Webseiten aufgefordert, einen Benutzernamen und ein Passwort einzugeben. Squid verfügt über mehrere Authentifizierungsmodule, sodass Sie die Anmeldung z. B. über PAM, LDAP, Windows-Domänen oder, im einfachsten Fall, über eine eigene Passwortdatei abwickeln können. Die Verwendung einer Passwortdatei ähnelt der Konfiguration der Basisauthentifizierung von Apache. Führen Sie zunächst die folgenden Schritte durch, um die Authentifizierungskomponenten nachzuinstallieren:


```
root@arch-deb:/# cd /usr/src/squid/squid-3.2.0.9/helpers/basic_auth/NCSA
root@arch-deb:/usr/src/squid/squid-3.2.0.9/helpers/basic_auth/NCSA/# make
root@arch-deb:/usr/src/squid/squid-3.2.0.9/helpers/basic_auth/NCSA/#
 make install
```

Normalerweise kommt es bei der Installation zu keinerlei Komplikationen. Die installierten Authentifizierungsmodule finden Sie jetzt im Verzeichnis */usr/local/squid/libexec*.

Genau wie bei der Apache-Basisauthentifizierung erstellen Sie als Nächstes eine Passwortdatei mit Benutzernamen und Kennwörtern:

```
root@arch-deb:/# cd /usr/local/squid/etc
root@arch-deb:/usr/local/squid/etc# htpasswd -c passwd willi
New password:
Re-type new password:
```

Bei der Erstellung weiterer Benutzer lassen Sie die Option `-c` (create) einfach weg. Es würde sonst noch eine neue Datei erstellt und die bestehenden Benutzerkonten gingen verloren. Beispiel:

```
root@arch-deb:/usr/local/squid/etc# htpasswd passwd susi
New password:
Re-type new password:
```

Die Passwörter werden in der Passwortdatei verschlüsselt abgespeichert. Sie sollten auf einem Produktionssystem trotzdem sicherstellen, dass die Benutzer nicht das Recht haben, diese Datei zu lesen.

```
root@arch-deb:/usr/local/squid/etc# cat passwd
willi:9EvUN84SDrJy6
susi:5FMd5ieiINMf2
```

Hinweis

Sollten Sie Apache in der Zwischenzeit deinstalliert haben, verfügen Sie möglicherweise nicht mehr über das Programm `htpasswd`. Sie können das Programm aber bei Bedarf einzeln unter folgender Adresse aus dem Internet herunterladen:

http://www.squid-cache.org/htpasswd/

Damit die Basisauthentifizierung funktioniert, sind einige Anpassungen an der Datei *squid.conf* erforderlich. Fügen Sie zunächst *in einer einzigen Zeile* den folgenden Eintrag hinzu, um das Authentifizierungsmodul zu laden:

```
auth_param basic program /usr/local/squid/libexec/basic_ncsa_auth /usr/local/
squid/etc/passwd
```

Hierbei zeigt */usr/local/squid/libexec/basic_ncsa_auth* den Pfad zum Authentifizierungsmodul und */usr/local/squid/etc/passwd* die zu verwendende Passwortdatei an. Es folgen ein paar Grundeinstellungen:

```
auth_param basic children 5
auth_param basic realm Authentifizierung erforderlich!
auth_param basic credentialsttl 8 hours
```

Der erste Parameter sorgt dafür, dass fünf Child-Prozesse gestartet werden, wie Sie wahrscheinlich schon selbst vermutet haben. Mit dem zweiten Eintrag können Sie eine Nachricht an den Benutzer übermitteln, wenn er sich anmeldet. Sie sehen das Ergebnis in Abbildung 208.2 weiter hinten. Der letzte Eintrag sorgt dafür, dass ein Benutzer sich innerhalb von acht Stunden nur einmal authentifizieren muss. Nach einem Browserneustart ist allerdings in jedem Fall eine neue Anmeldung fällig.

Als Nächstes müssen Sie der Datei *squid.conf* einen ACL-Eintrag hinzufügen, der die Verwendung einer Authentifizierung voraussetzt:

```
acl users proxy_auth REQUIRED
```

Bei der Positionierung des Statements, das den soeben erstellten ACL-Eintrag verwendet, ist Vorsicht geboten, was die Reihenfolge der Berechtigungsvergaben anbelangt. Wenn der Eintrag zu weit unten in der Konfiguration steht, könnte ein Benutzer schon Zugriff erlangen, bevor die Authentifizierung überhaupt greift. Beispiel:

```
http_access allow users
http_access allow localnet
http_access allow localhost
```

Falsch wäre etwa:

```
http_access allow localnet
http_access allow localhost
http_access allow users
```

Bei der falschen Konfiguration bekäme ein Benutzer bereits Zugriff aufgrund seiner Position in einem lokalen Netz.

Sie können die Authentifizierung jetzt mit einem Webbrowser testen.

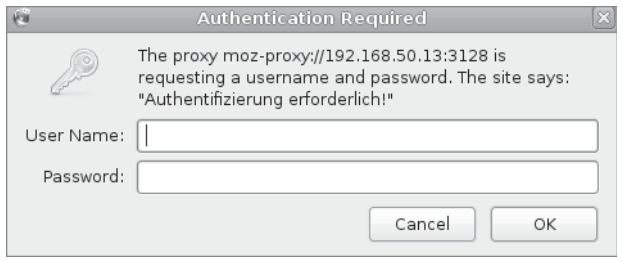

Abbildung 208.2 Authentifizierung bei einem Proxy-Server

Sollte der Authentifizierungsdialog nicht erscheinen und Sie können den Proxy ohne Authentifizierung verwenden, prüfen Sie noch einmal die Reihenfolge der Berechtigungseinträge in der Datei *squid.conf*. Schieben Sie den Eintrag `http_access allow users` im Zweifelsfall weiter nach oben.

208.4 Implementieren von nginx als Webserver und Reverse-Proxy

Wichtung: 2

Beschreibung: Die Kandidaten sollten dazu in der Lage sein, *nginx* als Reverse-Proxy zu konfigurieren. Auch Kenntnisse über die Grundkonfiguration als Webserver sind notwendig.

Wichtigste Wissensgebiete:

- *nginx*
- Reverse-Proxy
- Basiswebserver

Liste Wichtiger Dateien, Verzeichnisse und Anwendungen:

- */etc/nginx*
- *nginx*

Allgemeines

nginx ist eine Kombination aus Webserver, Reverse-Proxy und E-Mail-Proxy für IMAP und POP3. Ausgesprochen wird es übrigens »Engine X«. Für die Prüfung ist die E-Mail-Proxy-Funktionalität nicht von Belang, aber Sie sollten zumindest eine Grundkonfiguration als Reverse-Proxy und als Webserver durchführen können. *nginx* ist plattformunabhängig, und die Windows-Version läuft sogar ohne eine Emulationsschicht direkt auf der Win32-API.

Reverse-Proxy

Da Sie vermutlich noch über einen komplett konfigurierten Apache Webserver verfügen, will ich hier mit der Konfiguration von *nginx* als Reverse-Proxy für einen Apache Server beginnen. In diesem Beispiel wird *nginx* auf derselben Maschine ausgeführt, wie der Apache Webserver selbst.

Zuerst müssen Sie *nginx* natürlich installieren. Unter Debian und seinen Derivaten können Sie das Paket ganz einfach mit `apt-get` installieren. Bei CentOS und Fedora müssen Sie zunächst die Paketquellen selbst nachinstallieren, weshalb hier der Einfachheit halber das Beispiel auf einem Ubuntu-Server präsentiert wird.

Am besten noch vor der Installation von *nginx* sollten Sie Apache auf einen anderen Port als den standardmäßigen Port umkonfigurieren. Je nachdem, wie Ihr Apache derzeit konfiguriert ist, müssen Sie in der Datei *httpd.conf* (bei vielen Distributionen in der Datei *ports.conf*) folgende Änderungen vornehmen:

```
NameVirtualHost *:8000
Listen 8000
```

Sie können natürlich einen anderen Port nehmen als den Port 8000; Hauptsache Sie verwenden keinen Port, der bereits von einem anderen Daemon verwendet wird. Wenn Sie eine Konfigurationsdatei für einen virtuellen Host verwenden, müssen Sie hier ebenfalls eine Änderung vornehmen:

```
<VirtualHost *:8000>
```

Bei einer Standardkonfiguration unter Debian und Ubuntu finden Sie die Konfigurationsdatei im Verzeichnis */etc/apache2/sites-available*. Starten Sie den Apache Webserver neu, damit er den geänderten Port verwendet:

```
root@archangel:/etc/apache2/sites-available# apachectl restart
```

Da Apache jetzt nicht mehr den TCP-Port 80 abhört, können Sie diesen für *nginx* verwenden. Sie können die Installation also jetzt gefahrlos durchführen:

```
root@archangel:/# apt-get install nginx
```

Nach der Installation ist *nginx* grob als Webserver (allerdings ohne existierendes Dokumentenverzeichnis) konfiguriert. Die Konfigurationsdateien befinden sich unterhalb von */etc/nginx*. Sie werden feststellen, dass es hier, genauso wie bei Apache, die Verzeichnisse *sites-available* und *sites-enable* gibt. Diese Verzeichnisse werden von *nginx* auch genauso verwendet wie von Apache. Um den standardmäßig konfigurierten Webserver zu deaktivieren, müssen Sie also lediglich den entsprechenden Link löschen:

```
root@archangel:/# rm /etc/nginx/sites-enabled/default
```


Die Hauptkonfigurationsdatei ist *nginx.conf*. Hier werden einige grundlegende Parameter definiert, wie die Anzahl der Arbeitsprozesse und die maximale Anzahl gleichzeitiger Zugriffe. Die anderen Dateien, die direkt in */etc/nginx* liegen, sind Beispieldateien, die Sie in Ihre eigene Konfiguration bei Bedarf einbinden können. Für eine einfache Reverse-Proxy-Konfiguration erstellen und bearbeiten Sie z. B. diese Datei:

```
root@archangel:/# vi /etc/nginx/sites-available/proxy
```

Erstellen Sie eine Serverdirektive mit mindestens diesem Inhalt:

```
server {
    listen 80 default;
    location / {
        proxy_pass http://127.0.0.1:8000;
    }
}
```

Der Reverse-Proxy lauscht an Port 80 und leitet Anfragen an Port 8000 des Loopback-Adapters weiter. Das ist der Port, auf den Sie vorhin Apache konfiguriert haben. An dieser Stelle bietet es sich optional an, den Inhalt der Beispieldatei */etc/nginx/proxy_params* einzubinden. Sie müssen jetzt nur noch einen Softlink anlegen, damit *nginx* diese Konfiguration auch lädt:

```
root@archangel:/#ln -s  /etc/nginx/sites-available/proxy \
/etc/nginx/sites-enabled/proxy
```

Jetzt ist der richtige Zeitpunkt für einen ersten Start:

```
root@archangel:/# /etc/init.d/nginx start
```

Wenn alles gut gegangen ist, können Sie jetzt mit einem Webbrowser ganz normal auf den TCP-Port 80 dieses Servers zugreifen und erhalten den Inhalt der Webseite, die von Apache gehostet wird.

Sie können ggf. die Syntax der Konfigurationsdateien von *nginx* überprüfen, indem Sie folgendes Kommando verwenden:

```
root@archangel:/# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Es scheint alles in Ordnung zu sein. Sie können *nginx* zur Laufzeit Signale übermitteln, indem Sie die Option `-s` mit einem der folgenden Parameter verwenden: `stop`, `quit`, `reopen`, `reload`. Wenn Sie eine Konfigurationsänderung durchgeführt haben, können Sie diese also mit folgendem Kommando sofort wirksam werden lassen:

```
root@archangel:/# nginx -s reload
```

nginx als Webserver

Wenn Sie einen Apache Webserver konfigurieren können, wird Ihnen die Einrichtung eines *nginx*-Servers auch keine Schwierigkeiten bereiten. Sie können einfach mehrere virtuelle Webserver bereitstellen, indem Sie im Verzeichnis */etc/nginx/sites-available* die jeweiligen Konfigurationsdateien erstellen und bei Bedarf in */etc/nginx/sites-enabled* verlinken. Eine solche Konfigurationsdatei könnte z. B. so aussehen:

```
server {
    listen          80;
    server_name     www.lpic-2.de;
    index           index.html;
    root            /var/www/lpic-2.de;
}
```

Was Sie hier sehen, ist ein Serverblock. Bei Apache würde man dieses Konstrukt als einen virtuellen Server bezeichnen. Die meisten Einstellungen sind selbsterklärend, aber zur Sicherheit sollen diese hier dennoch erläutert werden.

Die Direktive `listen` legt fest, an welchem Port der Server lauschen soll. Mit der Einstellung `server_name` wird der Hostheadername festgelegt, falls auf dem physischen Server mehrere Webseiten unter derselben IP-Adresse und demselben Port erreichbar sein müssen. Diese Webseite wird also nur dann ausgeliefert, wenn Sie in den Browser *www.lpic-2.de* eingeben. Vergessen Sie nicht, die entsprechende DNS-Konfiguration durchzuführen, falls Sie eine Konfiguration mit mehreren Webseiten ausprobieren wollen. Der Eintrag `index` sagt dem Webserver, welche Datei er herausgeben soll, wenn keine Datei ausdrücklich auf der URL angegeben wurde. Die Direktive `root` entspricht dem DocumentRoot bei Apache. Es handelt sich also um das Hauptverzeichnis des virtuellen Servers.

Bei Servern, die mehrere Webseiten hosten, bietet sich als sauberer Abschluss eine Konfiguration an, welche jene Client-Anfragen bedient, bei denen der Hostheaderwert mit keiner gültigen Konfiguration übereinstimmt, oder falls der Benutzer eine IP-Adresse in das URL-Feld des Browsers eingegeben hat:

```
server {
    listen          80 default_server;
     index           index.html;
    root            /var/www/default;
}
```

Dieser Eintrag bedient alle verbleibenden Anfragen an TCP-Port 80 des Servers über all seine erreichbaren IP-Adressen.

Übungsfragen zu LPI 117-202

Die folgenden Fragen sollen Ihnen helfen, sich an die Art der Fragestellung in der wirklichen Prüfung zu gewöhnen. Es hat keinen Zweck, die Fragen einfach auswendig zu lernen, denn es sind keine echten Prüfungsfragen. Sie sollten versuchen, die Antworten zu jeder einzelnen Frage zu verstehen. Deshalb werden sowohl die richtigen als auch die falschen Antworten im Lösungsteil des Buchs detailliert besprochen. Das Üben mit diesen Fragen soll Ihnen auch die Herangehensweise bei eventuell Ihnen unbekannten Themen näher bringen. Ein unbekanntes Kommando in einer Frage ist nämlich noch längst kein Grund, eine Frage einfach nicht zu beantworten. Oft führt ein wenig Logik oder das Ausschlussverfahren dennoch zum Ziel.

Fragen

Frage 1:

Sie verwalten einen Apache Webserver, auf dem vertrauliche Dokumente gespeichert sind. Um die Verwaltung zu vereinfachen, wollen Sie die Benutzer, die auf den Server zugreifen dürfen, in einer Gruppe zusammenfassen. Welche Datei werden Sie erstellen bzw. bearbeiten?

- ☐ A: */etc/passwd*
- ☐ B: */etc/groups*
- ☐ C: *.htaccess*
- ☐ D: *.htgroup*
- ☐ E: *httpd.conf*

Frage 2:

Sie verwenden zu Testzwecken einen Apache Webserver auf einer Workstation. Sie müssen den Zugriff auf den Server über das Netzwerk verhindern. Mit welcher Direktive erreichen Sie das?

- ☐ A: `Port 81`
- ☐ B: `Listen 127.0.0.1:80`
- ☐ C: `Listen 82`
- ☐ D: `ServerType standalone`
- ☐ E: `ServerType inetd`

Frage 3:

Sie stellen bei einem hoch frequentierten Apache Webserver fest, dass der Zugriff auf Webseiten nur verzögert erfolgt. CPU-Ressourcen und Arbeitsspeicher sind jedoch nicht ausgelastet. Wie erhöhen Sie die Serverperformance? (Wählen Sie zwei Antworten.)

- ☐ A: Fügen Sie dem Server weitere IP-Adressen hinzu.
- ☐ B: Erhöhen Sie den Wert für `MinSpareServers`.
- ☐ C: Legen Sie den `ServerType` mit `standalone` fest.
- ☐ D: Legen Sie den `ServerType` mit `inetd` fest.
- ☐ E: Konfigurieren Sie einen anderen Port.

Frage 4:

Sie müssen einen Apache Webserver neu starten. Sie wollen hierfür ein Skript verwenden, das ausdrücklich für solche administrativen Eingriffe gedacht ist. Welches Kommando werden Sie verwenden? (Geben Sie ggf. benötigte Optionen mit an.)

_____

Frage 5:

Sie müssen einer existierenden Passwortdatei eines Apache Webservers einen neuen Benutzer namens `ritchie` hinzufügen. Wie lautet das richtige Kommando, wenn der aktuelle Pfad dem `ServerRoot` entspricht?

- ☐ A: `bin/htpasswd passwortdatei ritchie`
- ☐ B: `bin/htpasswd -c passwortdatei ritchie`
- ☐ C: `useradd ritchie -m`
- ☐ D: `adduser ritchie`
- ☐ E: `echo ritchie >> .htaccess`

Frage 6:

Welche Direktive eines Apache Webservers legt fest, welches Verzeichnis aus der Sicht eines Benutzers das Hauptverzeichnis des Webservers ist?

- ☐ A: `ServerRoot`
- ☐ B: `ApacheRoot`
- ☐ C: `DocumentRoot`
- ☐ D: `Path`
- ☐ E: `httpd-Path`

Frage 7:

Sie müssen Fehlermeldungen eines virtuellen Hosts, der auf einem Apache Webserver läuft, in einer separaten Datei aufzeichnen. Welche Direktive werden Sie konfigurieren?

- ☐ A: `LogFile`
- ☐ B: `syslog.conf`
- ☐ C: `DocumentRoot`
- ☐ D: `CustomLog`
- ☐ E: `ErrorLog`

Frage 8:

Sie wollen mehrere Webseiten auf einem einzigen Apache Webserver hosten. Welche Möglichkeiten der Unterscheidung gibt es beim Zugriff auf die verschiedenen Webseiten? (Wählen Sie drei Antworten.)

- ☐ A: mehrere Verzeichnisse
- ☐ B: mehrere IP-Adressen
- ☐ C: mehrere HTML-Dateien
- ☐ D: mehrere TCP-Ports
- ☐ E: mehrere `ServerName`-Direktiven

Frage 9:

Sie müssen Ihren Benutzern den Zugriff auf Webseiten ermöglichen, die durch OpenSSL-Zertifikate gesichert wurden. Welchen Port müssen Sie auf der Firewall zulassen? (Geben Sie nur den Port ohne Protokoll an.)

_____

Frage 10:

Sie benötigen ein Zertifikat zur Absicherung einer Webseite. Mit welchem Programm können Sie das Zertifikat generieren?

- ☐ A: `httpd`
- ☐ B: `openssh`
- ☐ C: `openssl`
- ☐ D: `openswan`
- ☐ E: `https`

Frage 11:

Sie wollen die Speichermenge, die ein neuer Squid-Proxy auf der Festplatte belegen darf, konfigurieren. Welche Einstellung ändern Sie?

- ☐ A: `cache_mem`
- ☐ B: `cache_dir`
- ☐ C: `reply_body_max_size`
- ☐ D: `disk_usage`
- ☐ E: `access_log`

Frage 12:

Sie haben die Option `cache_dir` in der Datei *squid.conf* eines neuen Servers konfiguriert. Welches Kommando werden Sie anschließend ausführen? Vervollständigen Sie das Kommando!

`/usr/local/squid/sbin/squid_____`

Frage 13:

Sie müssen verhindern, dass Benutzer beim Zugriff auf einen Samba-Server Dateien mit einer bestimmten Dateierweiterung zu sehen bekommen. Wenn ein Benutzer auf eine solche Datei direkt zugreift, soll er diese dennoch öffnen können. Welchen Eintrag werden Sie in der Datei *smb.conf* verwenden?

- ☐ A: `lock directory`
- ☐ B: `security = user`
- ☐ C: `security = share`
- ☐ D: `hide files`
- ☐ E: `veto files`

Frage 14:

Bei einem Samba-Server soll verhindert werden, dass Benutzer auf Dateien mit einer bestimmten Dateierweiterung zugreifen können. Das soll auch dann gelten, wenn User Dateinamen direkt auswählen. Welchen Eintrag benötigen Sie hierfür?

- ☐ A: `lock directory`
- ☐ B: `security = user`
- ☐ C: `security = share`
- ☐ D: `hide files`
- ☐ E: `veto files`

Antworten und Erklärungen zu den Prüfungsfragen

Hier finden Sie die Erläuterungen zu allen Fragen des zweiten Teils. Sie sollten unbedingt auch die Kommentare zu den falschen Antworten lesen. Einige Fakten werden hier nicht zufällig mehrfach erwähnt, sondern weil wesentliche Prüfungsinhalte auf diese Weise besser in Ihrem Gedächtnis haften bleiben.

Frage 1:

D: *.htgroup* ist die richtige Datei für diese Aufgabe. Hier können Sie Benutzer, die zuvor in einer Passwortdatei angelegt wurden, gruppieren. Sie können die Datei auch anders nennen, aber *.htgroup* ist der übliche Name.

zu A: */etc/passwd* enthält normale Benutzerkonten.

zu B: */etc/groups* enthält Benutzergruppen, die aber üblicherweise nicht zur Zugriffssteuerung auf Webserver verwendet werden.

zu C: *.htaccess* wird ebenfalls zur Zugriffssteuerung auf Apache-Servern verwendet. Sie müssen im vorliegenden Fall sogar die *.htgroup*-Datei in der Datei *.htaccess* als `AuthGroupFile` angeben.

zu E: *httpd.conf* ist die Hauptkonfigurationsdatei des Apache-Servers.

Frage 2:

B: `Listen 127.0.0.1:80` bewirkt, dass `httpd` nur noch an der Loopback-Adresse lauscht. Ein Zugriff vom Netzwerk aus ist dann nicht mehr möglich.

zu A: `Port 81` wird bei älteren Apache-Versionen verwendet, um den Server mit einem alternativen Port zu konfigurieren. Aber der Server wäre weiterhin vom Netzwerk aus erreichbar.

zu C: `Listen 82` sorgt dafür, dass Apache an Port 82 lauscht. Hierbei werden aber weiterhin alle Netzwerkschnittstellen verwendet.

zu D: `ServerType standalone` klingt natürlich wie eine sehr gute Antwort, bewirkt aber lediglich, dass Apache alleinstehend, also ohne vorgeschalteten Super-Daemon (`inetd`) läuft. Das ist übrigens die Standardeinstellung.

zu E: `ServerType inetd` teilt dem Webserver mit, dass er durch `inetd` aufgerufen wird. Eine solche Konfiguration sollte man übrigens aus Gründen der schlechteren Performance vermeiden.

Frage 3:

B: »Erhöhen Sie den Wert für `MinSpareServers`.« Diese Maßnahme erhöht die Anzahl der Prozesse, die auf eingehende Verbindungen lauschen. Benutzer können so schneller bedient werden.

C: »Legen Sie den `ServerType` mit `standalone` fest.« Das ist zwar die Standardeinstellung, muss aber gegenüber den anderen Antworten in Erwägung gezogen werden – insbesondere, weil die anderen Antworten entweder keine oder sogar eine negative Auswirkung auf die Serverleistung haben.

zu A: »Fügen Sie dem Server weitere IP-Adressen hinzu.« Das bringt nichts in Bezug auf die Leistung des Servers.

zu D: »Legen Sie den `ServerType` mit `inetd` fest.« Das verzögert sogar die Antwort bei jedem Benutzerzugriff.

zu E: »Konfigurieren Sie einen anderen Port.« Das bringt nichts in Bezug auf die Leistung des Servers.

Frage 4:

`apachectl restart` oder `apache2ctl restart` sind jeweils richtig. Es gibt natürlich auch andere Methoden, Apache neu zu starten, aber hier war ausdrücklich nach einem dafür vorgesehenen Skript gefragt.

Frage 5:

A: `bin/htpasswd passwortdatei ritchie` ist richtig, wenn Sie einen Benutzer einer existierenden Passwortdatei hinzufügen wollen. Da der aktuelle Pfad `ServerRoot` entspricht, muss dem `htpasswd`-Kommando der relative Pfad `bin/` vorangestellt werden.

zu B: `bin/htpasswd -c passwortdatei ritchie` ist hier falsch, weil die Option `-c` nur bei der Erstellung einer neuen Passwortdatei verwendet wird.

zu C und D: `useradd ritchie -m` und `adduser ritchie` sind falsch, weil diese Kommandos zur Erstellung normaler Benutzerkonten verwendet werden und nicht für Apache-Benutzer in Passwortdateien.

zu E: `echo ritchie >> .htaccess` ist völlig falsch, weil in dieser Datei keine Benutzerkonten angelegt werden. Es wird hier lediglich die Passwortdatei angegeben.

Frage 6:

C: `DocumentRoot` ist die Direktive, die das Hauptverzeichnis für den Content des Webservers angibt.

zu A: `ServerRoot` ist das Serverhauptverzeichnis, in dem sich auch Module und Konfigurationsdateien des Servers befinden. Hierauf darf ein Benutzer natürlich nicht zugreifen.

zu B, D und E: `ApacheRoot`, `Path` und `httpd-Path` sind einfach frei erfunden, also falsch.

Frage 7:

E: `ErrorLog` ist die Direktive, die das Logfile für die Fehlerprotokollierung angibt.

zu A: `LogFile` gibt es beim Apache-Webserver nicht.

zu B: `syslog.conf` konfiguriert den `syslogd`, ist aber nicht zur Konfiguration der Apache-Logfiles geeignet.

zu C: `DocumentRoot` ist das Hauptverzeichnis für den Webcontent.

zu D: `CustomLog` enthält nicht nur Fehler, sondern auch Aufzeichnungen über erfolgreiche Zugriffe auf den Webserver.

Frage 8:

B: mehrere IP-Adressen, D: mehrere TCP-Ports oder E: mehrere `ServerName`-Direktiven können hier jeweils verwendet werden. IP-Adressen oder Ports können über die Direktive `LISTEN` für mehrere Webseiten unterschiedlich konfiguriert und zugewiesen werden. Die Direktive `ServerName` kann für mehrere virtuelle Hosts jeweils innerhalb der Direktiven `VirtualHost` unterschiedlich konfiguriert werden.

zu A und C: Mehrere Verzeichnisse oder mehrere HTML-Dateien können zur Unterscheidung von Webseiten aus der Sicht des Netzwerks nicht herangezogen werden.

Frage 9:

443 ist die korrekte Antwort. Es handelt sich hier letztendlich um das ganz normale HTTPS-Protokoll.

Frage 10:

C: `openssl` ist ein Programm, mit dem man unter anderem X.509-Zertifikate ausstellen kann.

zu A: `httpd` ist der eigentliche Webserver-Daemon.

zu B: `openssh` ist ein Paket, das die Secure Shell enthält.

zu D: `openswan` ist ein Paket zur Implementierung von IPSec.

zu E: `https` ist lediglich das Protokoll, das für HTTP über SSL verwendet wird.

Frage 11:

B: `cache_dir` ist die Direktive, in der die Verzeichnisstruktur des Proxy-Caches konfiguriert wird. Hier wird auch die Größe des festplattenseitigen Caches festgelegt.

zu A: `cache_mem` legt fest, wie viel Arbeitsspeicher Squid für das Caching verwenden darf.

zu C: `reply_body_max_size` wird verwendet, um Benutzer am Herunterladen übergroßer Dateien zu hindern.

zu D: `disk_usage` gibt es nicht bei Squid.

zu E: `access_log` legt den Pfad zur Protokolldatei fest.

Frage 12:

`-z` wäre hier die benötigte Ergänzung. Nach der Konfiguration der Direktive `cache_dir` muss die Verzeichnisstruktur des Caches aufgebaut werden. Das wird durch dieses Kommando erreicht:

```
/usr/local/squid/sbin/squid -z
```

Frage 13:

D: `hide files` ist hier die richtige Option. Sie können so auch einzelne, bestimmte Dateien verstecken.

zu A: `lock directory` ist das Verzeichnis, in dem Samba-Lockfiles anlegt, damit überprüft werden kann, ob es bereits laufende Instanzen von Samba gibt.

zu B: `security = user` legt fest, dass Zugriffe nur durch Benutzer mit gültigen Benutzerkonten erfolgen dürfen.

zu C: `security = share` lässt Zugriffe ohne Authentifizierung von Benutzern zu. Diese Konfiguration wird von neueren Samba-Versionen nicht mehr unterstützt.

zu E: `veto files` ähnelt vom Verhalten her der richtigen Antwort. Es wird hier aber auch der Zugriff auf Dateien verweigert, die ein Benutzer konkret angibt.

Frage 14:

E: `veto files` verhindern den Zugriff auf angegebene Dateien oder auch Erweiterungen. Das gilt auch dann, wenn ein Benutzer eine konkrete Datei auswählt.

zu D: `hide files` versteckt die Dateien nur. Ein direkter Zugriff bleibt weiterhin möglich.

zu A, B und C: Für `lock directory`, `security = user` und `security = share` gelten die Erläuterungen zu den Antworten der vorherigen Frage.

Frage 15:

A: `testparm` überprüft den Inhalt der Datei *smb.conf* auf Korrektheit.

zu B: `hdparm` dient der Anzeige und Überprüfung von Festplattenparametern.

zu C: `nmblookup` ist ein Tool, das zur NetBIOS-Namensauflösung verwendet wird.

zu D: `smbstatus` zeigt den Status bestehender Samba-Verbindungen von Clients an.

zu E: `smbcontrol` wird verwendet, um Signale an die Daemons `smbd` oder `nmbd` zu senden.