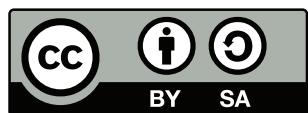


Linux als Web- und FTP-Server

Dateidienste für das Internet



Das tuxcademy-Projekt bietet hochwertige frei verfügbare Schulungsunterlagen zu Linux- und Open-Source-Themen – zum Selbststudium, für Schule, Hochschule, Weiterbildung und Beruf.
Besuchen Sie <https://www.tuxcademy.org/>! Für Fragen und Anregungen stehen wir Ihnen gerne zur Verfügung.

Linux als Web- und FTP-Server Dateidienste für das Internet

Revision: webf:6306e8ce0e93133e:2014-03-31

apw2:b940c2242a09e73f:2014-03-31 10-12
apws:7baeadfdefed6c81:2014-03-31 1-9, B-D
ftpd:6fda4441d0e3d931:2013-02-21 16-18
squi:ba25561140c7b12b:2013-02-21 13-15
webf:BxIUyMRkrcKghdK0YymbVs

© 2015 Linup Front GmbH Darmstadt, Germany
© 2017 tuxcademy (Anselm Lingnau) Darmstadt, Germany
<http://www.tuxcademy.org> · info@tuxcademy.org
Linux-Pinguin »Tux« © Larry Ewing (CC-BY-Lizenz)

Alle in dieser Dokumentation enthaltenen Darstellungen und Informationen wurden nach bestem Wissen erstellt und mit Sorgfalt getestet. Trotzdem sind Fehler nicht völlig auszuschließen. Das tuxcademy-Projekt haftet nach den gesetzlichen Bestimmungen bei Schadensersatzansprüchen, die auf Vorsatz oder grober Fahrlässigkeit beruhen, und, außer bei Vorsatz, nur begrenzt auf den vorhersehbaren, typischerweise eintretenden Schaden. Die Haftung wegen schuldhafter Verletzung des Lebens, des Körpers oder der Gesundheit sowie die zwingende Haftung nach dem Produkthaftungsgesetz bleiben unberührt. Eine Haftung über das Vorgenannte hinaus ist ausgeschlossen.

Die Wiedergabe von Warenbezeichnungen, Gebrauchsnamen, Handelsnamen und Ähnlichem in dieser Dokumentation berechtigt auch ohne deren besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne des Warenzeichen- und Markenschutzrechts frei seien und daher beliebig verwendet werden dürfen. Alle Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt und sind möglicherweise eingetragene Warenzeichen Dritter.



Diese Dokumentation steht unter der »Creative Commons-BY-SA 4.0 International«-Lizenz. Sie dürfen sie vervielfältigen, verbreiten und öffentlich zugänglich machen, solange die folgenden Bedingungen erfüllt sind:

Namensnennung Sie müssen darauf hinweisen, dass es sich bei dieser Dokumentation um ein Produkt des tuxcademy-Projekts handelt.

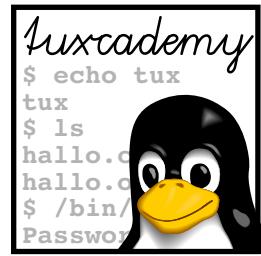
Weitergabe unter gleichen Bedingungen Sie dürfen die Dokumentation bearbeiten, abwandeln, erweitern, übersetzen oder in sonstiger Weise verändern oder darauf aufbauen, solange Sie Ihre Beiträge unter derselben Lizenz zur Verfügung stellen wie das Original.

Mehr Informationen und den rechtsverbindlichen Lizenzvertrag finden Sie unter <http://creativecommons.org/licenses/by-sa/4.0/>

Autoren: Anselm Lingnau, Thomas Erker

Technische Redaktion: Anselm Lingnau (anselm.lingnau@linupfront.de)

Gesetzt in Palatino, Optima und DejaVu Sans Mono



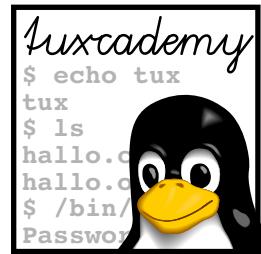
Inhalt

1 Apache-Grundlagen	1
1.1 Eine kurze Geschichte des WWW	2
1.2 URLs, URLs und URNs	3
1.3 Das Hypertext Transfer Protocol	5
1.4 Markup-Sprachen.	9
1.5 Web-Server und -Browser	12
1.6 Das World-Wide-Web-Konsortium.	12
2 Der Web-Server Apache	15
2.1 Die Geschichte von Apache	16
2.2 Installation	17
2.3 Die Konfigurationsdateien	18
2.4 Bearbeitung von Anfragen	19
2.5 Apache-Module	20
2.6 Protokolldateien	21
2.7 Virtuelle Web-Server.	21
2.8 Apache-Betrieb	21
2.9 Apache-Informationsquellen.	22
3 Apache-Konfiguration	25
3.1 Die Datei httpd.conf	26
3.2 Globale Einstellungen	27
3.3 Dynamisch ladbare Erweiterungen	28
3.4 Konfiguration des »Hauptservers«.	28
3.5 Konfiguration für bestimmte Ressourcen	32
4 Organisation einer Web-Präsenz	37
4.1 Verzeichnisstrukturen	38
4.2 Automatisch erzeugte Verzeichnislisten.	38
4.3 Benutzereigene Seiten	43
4.4 Inhaltsaushandlung	46
4.5 Die Alias-Direktive	50
4.6 Weiterleitung	51
4.7 Dynamische Inhalte	52
4.8 Fehlermeldungen	57
5 Zugriffsrechte und Zugriffsschutz	59
5.1 Überblick.	60
5.2 Zugriffskontrolle auf TCP-Ebene	60
5.3 HTTP-basierte Authentisierung.	62
5.4 Apache und SSL	68

6 Protokolldateien	71
6.1 Überblick	72
6.2 Standardformate für Protokolldateien	72
6.3 Selbstdefinierte Formate	73
6.4 Fehlermeldungen	75
6.5 Verwaltung von Protokolldateien	76
6.6 Automatisierte Auswertung von Protokolldateien	78
7 Virtuelle Web-Server	85
7.1 Einführung	86
7.2 IP-basierte virtuelle Server	87
7.3 Namensbasierte virtuelle Server	88
7.4 Tips und Tricks	91
8 Apache-Sicherheit	93
8.1 Was und warum?	94
8.2 Betriebssystem und Netz	96
8.3 Apache-Sicherheitstips	97
9 Apache-Effizienz	101
9.1 Allgemeines	102
9.2 Das Prozessmodell von Apache	102
9.3 Effizienz-Tips	104
9.4 Große und aufwendige Web-Präsenzen	105
10 SSL, TLS, OpenSSL und mod_ssl	107
10.1 SSL	108
10.1.1 Grundlagen	108
10.1.2 Probleme mit SSL und TLS	110
10.2 OpenSSL	114
10.3 mod_ssl	115
11 Zertifikate	119
11.1 Zertifizierung	120
11.2 X.509	121
11.3 Zertifizierungsstellen	123
11.4 Erzeugung von Zertifikaten	125
12 Apache, OpenSSL und mod_ssl	133
12.1 Apache und mod_ssl	134
12.1.1 Grundlagen	134
12.1.2 Globale Konfiguration	134
12.2 Server-Authentisierung	138
12.2.1 Zertifikate installieren	138
12.2.2 Browser und Zertifizierungsstellen	141
12.2.3 Ressourcen-Konfiguration	144
12.2.4 Session Caching	146
12.2.5 Virtuelle Server	149
12.2.6 Apache und TLS-Angriffe	151
12.2.7 Protokollierung	152
12.3 Client-Authentisierung mit Zertifikaten	154
12.3.1 Server-Konfiguration	154
12.3.2 Client-Zertifikate im Browser installieren	156
12.3.3 Zugriffsregeln	156
12.4 Sicherheit für Web-Präsenzen verbessern	161
12.4.1 HTTP Strict Transport Security (HSTS)	161
12.4.2 Public Key Pinning (HPKP)	163
12.4.3 Content Security Policy (CSP)	166

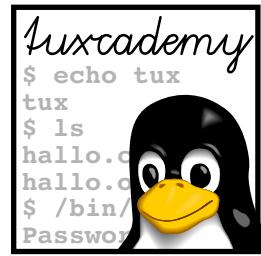
13 Proxy-Grundlagen	171
13.1 Einleitung	172
13.2 Arten von Proxies	172
13.2.1 Generische Proxies	172
13.2.2 HTTP-Proxies	172
13.2.3 Reverse Proxies	174
13.2.4 Proxies und HTTPS	174
13.3 Software	175
13.3.1 Allgemein	175
13.3.2 Squid	175
13.3.3 Apache	176
13.3.4 Andere Programme	176
14 Squid-Konfiguration	179
14.1 Basis-Einrichtung von Squid	180
14.1.1 Vorbereitende Schritte	180
14.1.2 Squid testen	182
14.1.3 Squid steuern	183
14.1.4 Laufzeitverhalten kontrollieren	183
14.2 Squid als clientseitiger Cache	184
14.2.1 Cache-Größe und Speicher-Zuweisung	184
14.2.2 Cache-Statistiken	185
14.2.3 Cache-Hierarchien	186
14.2.4 Auswertung des Zugriffsprotokolls	187
14.3 Zugriffskontrolle	188
14.3.1 Struktur der Zugriffskontrolle	188
14.3.2 Typen von Bedingungen	190
14.3.3 Regeln und Aktionen	191
15 Squid und Firewalls	193
15.1 Squid als Application Level Gateway in Firewall-Systemen	194
15.2 Grundkonfiguration	194
15.3 Authentisierung gegenüber dem Proxy	194
15.3.1 Identifizierungs-Server	194
15.3.2 HTTP-basierte Authentisierung	195
15.4 Benutzer-unspezifische Zugriffsbeschränkungen	197
15.5 Filtern über die Redirector-Schnittstelle	198
15.6 Firewall-Perforierung mit HTTP und HTTPS	199
16 Das File Transfer Protocol (FTP)	201
16.1 Einleitung	202
16.2 FTP-Clients	202
16.2.1 Web-Browser und grafische FTP-Clients	202
16.2.2 Terminalbasierte FTP-Clients	203
16.2.3 Nicht-interaktive FTP-Clients	205
16.2.4 Diagnose-Werkzeuge	205
16.3 Das Protokoll	206
16.3.1 Ursprünge	206
16.3.2 Kontroll- und Daten-Kanal	206
16.3.3 Datenübertragung	207
16.3.4 Authentisierung	208
16.3.5 Weitere FTP-Kommandos	208
16.3.6 Status-Meldungen	209
16.3.7 Alternativen	211
17 Der FTP-Server vsFTPd	213

17.1	Empfehlenswert!	214
17.2	Basis-Konfiguration	214
17.3	FTP-Server mit Benutzer-Authentisierung	215
17.4	Virtuelle Server	216
17.5	Virtuelle Benutzer	217
18	Pure-FTPd	219
18.1	Überblick	220
18.2	Konfiguration	223
18.2.1	Anonyme Downloads	223
18.2.2	Anonyme Downloads und Uploads	224
18.2.3	Zugriff für authentisierte Benutzer	226
18.2.4	Virtuelle Benutzer	231
18.2.5	Nachrichten.	233
18.2.6	Protokolldateien	235
18.2.7	Virtuelle Server	236
18.3	Sicherheit.	237
18.3.1	Verschiedene Sicherheits-Optionen	237
18.3.2	Pure-FTPd und Firewalls	239
18.3.3	FTPS mit Pure-FTPd.	239
A	Musterlösungen	241
B	HTTP-Statuswerte	251
B.1	Information	251
B.2	Erfolg	251
B.3	Umleitung	251
B.4	Client-Fehler	252
B.5	Server-Fehler	252
C	Apache-Direktiven	253
D	Variable für CGI-Skripte und <i>server-side includes</i>	259
D.1	Nicht anfragespezifische Variable	259
D.2	Anfragespezifische Variable	259
D.3	Zusätzliche Variable für <i>server-side includes</i>	260
Index		261



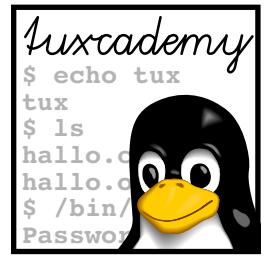
Tabellenverzeichnis

1.1	HTTP/1.1-Methoden	6
1.2	Struktur der HTTP-Statuswerte	6
4.1	Bedingungen in SSI-Ausdrücken	55
6.1	»%«-Sequenzen für Protokollformate	74
6.2	Fehlerstufen für <code>error_log</code>	76
9.1	Leistungsklassen für den Apache-Server bei SuSE-Linux (Stand: SuSE-Linux 8.2/SLES 8)	104
11.1	Symantec-Zertifikate 2013 (Auswahl)	123
12.1	TLS-Umgebungsvariable für CGI-Skripte und Server-Side-Includes	145
12.2	Zusätzliche Umgebungsvariable für Client-Zertifikate	157
12.3	Funktionen in Apache-Ausdrücken	158
16.1	Beispielhafte FTP-Sitzung	204
16.2	Weitere FTP-Kommandos	209
16.3	Beispielhafte FTP-Sitzung (Kontroll-Kanal)	210



Abbildungsverzeichnis

4.1	»Schlichte« Verzeichnisliste (ohne FancyIndexing)	40
4.2	»Aufwändige« Verzeichnisliste (mit Indexoptionen FancyIndexing und FoldersFirst)	41
4.3	Eine einfache Typemap-Datei	46
4.4	Eine Typemap-Datei mit Qualitätsfaktoren	47
5.1	Typische Dialogbox für HTTP-Authentisierung	63
6.1	logrotate-Konfiguration für Apache-Protokolldateien	77
6.2	Jahresstatistik von <i>The Webalizer</i>	79
6.3	Monatsstatistik von <i>The Webalizer</i> (Ausschnitt)	80
6.4	Anfang einer Analog-Statistik	82
6.5	Analog-Statistik: Dateitypen	82
6.6	Analog-Statistik: Domain-Hierarchie	83
11.1	Ein X.509-Zertifikat	121
11.2	Eine Beispiel-Konfigurationsdatei für OpenSSL	127
11.3	Ein selbstsigniertes Zertifikat für eine Zertifizierungsstelle	130
12.1	Zertifizierungsstelle unbekannt (Google Chrome)	142
12.2	TLS-Verbindungsinformationen (Google Chrome)	142
12.3	Zertifikatsverwaltung in Firefox	143
12.4	Bestätigungsdialog für Zertifizierungsstellen (Firefox)	143
12.5	Syntax für SSLRequire-Ausdrücke	157
12.6	Beispiel für HSTS-Konfiguration	163
16.1	Der grafische FTP-Client gftp	203



Vorwort

Dieser Kurs stellt den Web-Server Apache sowie die gängigsten FTP-Server für Linux vor. Er richtet sich an Teilnehmerinnen und Teilnehmer mit Linux- und TCP/IP-Kenntnissen, die einen Apache-Server installieren oder betreuen wollen. Zum Schulungsumfang gehören alle elementaren Kenntnisse, vom Hintergrund des WWW über Installation und Konfiguration des Servers bis zu Einführungen in die Problembereiche „Sicherheit“ und „Effizienz“. Weitere Kapitel beschreiben den Einsatz von SSL zur Server-Authentisierung und den Einsatz des Proxy-Servers Squid. Schließlich werden die Grundzüge des FTP-Protokolls beschrieben und eine Einführung in die Konfiguration der wichtigsten FTP-Server gegeben.

Diese Schulungsunterlage soll den Kurs möglichst effektiv unterstützen, indem das Kursmaterial in geschlossener, ausführlicher Form zum Mitlesen, Nach- oder Vorarbeiten präsentiert wird. Das Material ist in Kapitel eingeteilt, die jeweils für sich genommen einen Teilaspekt umfassend beschreiben; am Anfang jedes Kapitels sind dessen Lernziele und Voraussetzungen kurz zusammengefasst, am Ende finden sich eine Zusammenfassung und (wo sinnvoll) Angaben zu weiterführender Literatur oder WWW-Seiten mit mehr Informationen.

Kapitel Lernziele Voraussetzungen

 Zusätzliches Material oder weitere Hintergrundinformationen sind durch das »Glühbirnen«-Sinnbild am Absatzanfang gekennzeichnet. Zuweilen benutzen diese Absätze Aspekte, die eigentlich erst später in der Schulungsunterlage erklärt werden, und bringen das eigentlich gerade Vorgestellte so in einen breiteren Kontext; solche »Glühbirnen«-Absätze sind möglicherweise erst beim zweiten Durcharbeiten der Schulungsunterlage auf dem Wege der Kursnachbereitung voll verständlich.

 Absätze mit dem »Warnschild« weisen auf mögliche Probleme oder »gefährliche Stellen« hin, bei denen besondere Vorsicht angebracht ist. Achten Sie auf die scharfen Kurven!

 Die meisten Kapitel enthalten auch Übungsaufgaben, die mit dem »Bleistift«-Sinnbild am Absatzanfang gekennzeichnet sind. Die Aufgaben sind nummeriert und Musterlösungen für die wichtigsten befinden sich hinten in dieser Schulungsunterlage. Bei jeder Aufgabe ist in eckigen Klammern der Schwierigkeitsgrad angegeben. Aufgaben, die mit einem Ausrufungszeichen (»!«) gekennzeichnet sind, sind besonders empfehlenswert.

Übungsaufgaben

Auszüge aus Konfigurationsdateien, Kommando Beispiele und Beispiele für die Ausgabe des Rechners erscheinen in Schreibmaschinenschrift. Bei mehrzeiligen Dialogen zwischen Benutzer und Rechner werden die Benutzereingaben in **fetter Schreibmaschinenschrift** angegeben, um Missverständnisse zu vermeiden. Wenn Teile einer Kommandoausgabe ausgelassen wurden, wird das durch »<<<<<< kenntlich gemacht. Manchmal sind aus typografischen Gründen Zeilenumbrüche erforderlich, die in der Vorlage auf dem Rechner nicht stehen; diese werden als »><< dargestellt. Bei Syntaxdarstellungen stehen Wörter in spitzen Klammern (»<Wort>«) für »Variable«, die von Fall zu Fall anders eingesetzt werden können; Material in eckigen Klammern (»<[-f <Datei>]>«) kann entfallen und ein vertikaler Balken trennt Alternativen (»<-a|-b>«).

Wichtige Konzepte
Definitionen

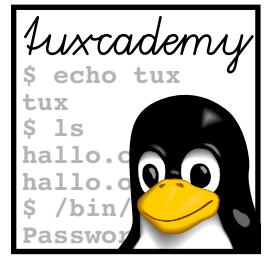
Wichtige Konzepte werden durch »Randnotizen« hervorgehoben; die **Definitionen** wesentlicher Begriffe sind im Text fett gedruckt und erscheinen ebenfalls am Rand.

Verweise auf Literatur und interessante Web-Seiten erscheinen im Text in der Form »[GPL91]« und werden am Ende jedes Kapitels ausführlich angegeben.

Wir sind bemüht, diese Schulungsunterlage möglichst aktuell, vollständig und fehlerfrei zu gestalten. Trotzdem kann es passieren, dass sich Probleme oder Ungenauigkeiten einschleichen. Wenn Sie etwas bemerken, was Sie für verbesslungsfähig halten, dann lassen Sie es uns wissen, etwa indem Sie eine elektronische Nachricht an

info@tuxcademy.org

schicken. (Zur Vereinfachung geben Sie am besten den Titel der Schulungsunterlage, die auf der Rückseite des Titelblatts enthaltene Revisionsnummer sowie die betreffende(n) Seitenzahl(en) an.) Vielen Dank!



1

Apache-Grundlagen

Inhalt

1.1	Eine kurze Geschichte des WWW	2
1.2	URIs, URLs und URNs	3
1.3	Das Hypertext Transfer Protocol	5
1.4	Markup-Sprachen.	9
1.5	Web-Server und -Browser	12
1.6	Das World-Wide-Web-Konsortium.	12

Lernziele

- Die Geschichte des WWW überblicken
- Die Grundlagen des HTTP kennen; simple HTTP-Probleme finden
- Markup-Sprachen verstehen und einordnen können
- Mit den Konzepten von Web-Servern und -Browsern vertraut sein
- Das World-Wide-Web-Konsortium und seine Aufgaben kennen

Vorkenntnisse

- Kenntnisse über das WWW als Benutzer
- Internet- und TCP/IP-Grundlagen

1.1 Eine kurze Geschichte des WWW

Vannevar Bush	Die Urfänge des World Wide Web liegen eine ganze Weile zurück: Oft wird als Anstoß der Artikel »As We May Think« zitiert, den der US-amerikanische Wissenschafts-Manager Vannevar Bush 1945 in der Zeitschrift <i>Atlantic Monthly</i> veröffentlichte [Bus45]. In diesem Artikel vertrat Bush die These, dass die Wissenschaftler sich nach dem Ende des Kriegs darauf konzentrieren sollten, das menschliche Wissen besser zugänglich zu machen, und schlug einen Apparat namens »Memex« (kurz für <i>memory extension</i>) vor, der auf Mikrofiche gespeicherte Informationen untereinander querverbinden und ein Verfolgen der Querverbindungen erlauben sollte. Leider wurde der Memex niemals gebaut.
Ted Nelson: Hypertext	Dann passierte eine Weile lang nichts, bis in die 1960er Jahre. 1965 prägte Ted Nelson den Begriff »Hypertext«. Die ersten Hypertext-Editiersysteme werden entworfen und implementiert, zum Beispiel von Andy van Dam oder Doug Engelbart, dem Erfinder der Maus.
Erste Systeme	Von Juni bis Dezember 1980 hatte Tim Berners-Lee, ein englischer Informatiker, einen Beratungsauftrag am Europäischen Kernforschungszentrum (CERN), wo er ein Hypertextsystem namens <i>Enquire-Within-Upon-Everything</i> entwickelte. Jede Seite in ENQUIRE hatte einen Titel, einen Datentyp und eine Liste von bidirektionalen Links.
ENQUIRE	1989 schrieb Berners-Lee, inzwischen wieder am CERN, einen Antrag, in dem er ein Hypertextsystem vorschlug, um Informationen über die Beschleuniger und Experimente am CERN zu sammeln und zu verwalten. Im Herbst 1990 wurde dieser Antrag genehmigt, und ein NeXT-Rechner für die Softwareentwicklung wurde gekauft; im Dezember lief der erste Browser, ein Programm namens »WorldWideWeb« (Später wurde es in »Nexus« umbenannt, um Verwechslungen zwischen dem Programm und dem Dienst zu vermeiden). Nexus war ein graphisches Programm, das allerdings nur auf dem NeXT lief; andere Plattformen mussten fürs erste mit dem zeilenorientierten Browser vorliebnehmen, den Nicola Pellow ab November 1990 entwickelte. Beide Browser und ein Server waren zu Weihnachten vorführbereit.
Erster WWW-Browser	1991 und 1992 breitete das WWW sich langsam weiter aus. Präsentationen auf Fachtagungen und Zeitschriftenartikel steigerten das Interesse an dem neuen System. Am 12. Dezember 1991 wurde am SLAC (Stanford Linear Accelerator Center) der erste Web-Server außerhalb Europas in Betrieb genommen. Im Frühjahr 1992 kamen die ersten graphischen Browser für Unix-Systeme heraus (»Erwise« aus Finnland und »ViolaWWW«). Im November gab es weltweit 26 stabile Web-Server.
Graphische Browser	1993 ist vielleicht das eigentliche Geburtsjahr des WWW. Im Februar erschien die erste Alpha-Version von Marc Andreessens »Mosaic«-Browser, der im Spätsommer für alle gängigen Plattformen (X, Macintosh und Windows) zur Verfügung stand. Andreessen arbeitete am NCSA, dem <i>National Center for Supercomputer Applications</i> an der Universität von Illinois in Urbana-Champaign. Im Oktober gab es über 200 bekannte Web-Server.
Mosaic	Im März 1994 machten Marc Andreessen und seine NCSA-Kollegen sich selbstständig, um die <i>Mosaic Communications Corp.</i> , später »Netscape«, zu gründen. Im Mai fand die erste World-Wide-Web-Konferenz, später »Woodstock of the Web« genannt, statt. Am 1. Oktober wurde das World-Wide-Web-Konsortium (W3C) gegründet, das als Interessenvereinigung der am World Wide Web beteiligten Firmen und Institutionen dessen Weiterentwicklung voranbringen soll. Das CERN stellte am 16. Dezember 1994 die WWW-Entwicklung aus Geldmangel ein.
Netscape	Irgegdwann in dieser Zeit bemerkte auch Microsoft das World Wide Web und beschloß, sich einen Teil vom Kuchen zu holen. Der bisher unangefochten komfortabelste Browser, Netscape, bekam Konkurrenz durch Microsofts »Internet Explorer«: Die »Browser-Kriege« waren eröffnet, in denen die Programmierer von Netscape und Microsoft sich gegenseitig durch neue Eigenschaften und Leistungsmerkmale ihrer Programme zu übertreffen versuchten. In den jeweiligen 4.x-Versionen begann Microsoft Netscape zu übertrumpfen, nicht zuletzt weil die Firma
W3C	
Microsoft	
Browser-Kriege	

die Möglichkeit hatte, allen Windows-Versionen einen Internet Explorer beizulegen, während die Benutzer Netscape selber installieren mußten. Im Januar 1998 beschloß Netscape, den Quellcode für ihren Browser als »Open-Source« freizugeben (hieraus resultierte das »Mozilla-Projekt«); 1999 wurde die Firma Netscape für rund 4,2 Milliarden Dollar an AOL verkauft.

In den Jahren nach der Jahrtausendwende schien die Dominanz von Microsoft auf dem Browser-Markt so erdrückend, dass die Firma jegliche Innovation im wesentlichen einstellte. Dies sollte sich als strategischer Fehler entpuppen, da einerseits ein starker Drang nach Web-Präsenzen einsetzte, die auf definierten und nicht von einzelnen Implementierungen abhängigen Standards aufbauen, und andererseits solche Standards in den entsprechenden Gremien vorangetrieben wurden. Der Internet Explorer unterstützte diese Entwicklungen aber nicht oder nur halbherzig und mit den üblichen Microsoft-eigenen »Reinterpretationen«, während neue alternative Browser wie Firefox (basierend auf der Netscape-Codebasis), oder der kommerzielle Browser Opera hierbei punkten können und auch beim Bedienungskomfort, der Geschwindigkeit und der Sicherheit Maßstäbe setzen.

Inzwischen ist die Vormachtsstellung von Microsoft im Browser-Bereich längst Geschichte. Außer Firefox und Opera sind auf PCs neue Browser wie Googles Chrome oder Safari von Apple weit verbreitet, die aus dem Lager der »offenen Standards« kommen. Tatsächlich stehen inzwischen leistungsfähige Bausteine für die Implementierung standardkonformer Browser als Open-Source-Software zur Verfügung, die aus diesen Projekten hervorgingen und/oder von ihnen benutzt und weiterentwickelt werden.

Die Kompatibilität zu Microsoft Explorer, falls erforderlich, entwickelt sich bei der Implementierung von Web-Präsenzen immer mehr zu einem Hemmschuh, so dass sogar Microsoft sich entgegen früherer Absichten genötigt sieht, den Internet Explorer mit dem Ziel »Standard-Konformität« weiterzuentwickeln – auch wenn da noch einiges aufzuholen ist. Weitere wichtige Einflußfaktoren auf die Gegenwart und Zukunft des World Wide Web sind die explosionsartige Verbreitung des »mobilen Internet« auf Smartphones und Tablets sowie die Tendenz hin zu aufwendigen im Browser laufenden Anwendungsprogrammen für alle möglichen Zwecke, die neue Herausforderungen für das Design und die Implementierung von Web-Präsenzen mit sich bringen.

 Über die Geschichte der Server-Seite erzählen wir noch ein bisschen was in Abschnitt 2.1.

1.2 URIs, URLs und URNs

Selbst als nicht besonders an Computern und dem Internet interessierte Person kann man sich heutzutage den »Web-Adressen« nicht mehr entziehen: Alle Medien sind voll davon. Aber was ist so ein »URL« eigentlich überhaupt? Die Antwort auf diese Frage steht im RFC 2396 [RFC2396]:

A Uniform Resource Identifier (URI) is a compact string of characters for identifying an abstract or physical resource.

(Eine einheitliche Ressourcen-Bezeichnung ist eine kompakte Zeichenkette, die eine abstrakte oder körperliche Ressource identifiziert.)

(Im Interesse der Klarheit verzichten wir im folgenden auf die Eindeutschung des Begriffs »URI«.)

Damit sind die wichtigsten Eigenschaften von URIs schon erwähnt:

Einheitlichkeit Verschiedene Arten von URIs können im selben Kontext verwendet werden, selbst wenn die für den Zugriff nötigen Mechanismen sich unterscheiden. Ein Browser kann zum Beispiel die Inhalte von HTTP- und FTP-Servern in für den Benutzer sehr ähnlicher Weise zugänglich machen.

Weitere Vorteile sind die einheitliche semantische Interpretation gemeinsamer syntaktischer Konventionen, das leichte Hinzufügen neuer Arten von URIs ohne Änderungen am Gebrauch der alten, und die Wiederverwendbarkeit existierender Bezeichner in anderen Zusammenhängen.

Ressourcen Eine Ressource kann alles sein, was eine Identität hat: ein elektronisches Dokument, ein Bild, ein Dienst (»die Wettervorhersage für morgen für Darmstadt«) oder eine Sammlung anderer Ressourcen (ein Konversationslexikon oder Telefonbuch). Eine Ressource lässt sich nicht notwendigerweise über das Internet abrufen (Menschen, Gebäude oder Bibliotheksbücher sträuben sich dagegen). Wichtig ist auch, dass eine Ressource nicht immer denselben Inhalt haben muss – Sie können sie auch als »Abbildung« eines Konzepts auf einen Inhalt ansehen, die konstant bleibt, selbst wenn der eigentliche Inhalt sich ändert (Stichwort Wettervorhersage).

Bezeichnung Eine Bezeichnung ist ein Objekt, das als Verweis auf etwas dienen kann, das eine Identität hat. Im Falle von URIs ist das Objekt eine Zeichenkette, deren Aufbau gewissen Regeln gehorcht.

Man kann URIs weiter unterteilen in Ortsangaben, Namen und URIs, die beides sind. Einen URI, der eine Ressource über ihre Zugriffsdaten (also ihren »Ort« auf dem Netz) bezeichnet, nennt man auch **Uniform Resource Locator** oder URL. Ein **Uniform Resource Name** oder URN ist eine Bezeichnung, die weltweit eindeutig bleiben muss, auch wenn die Ressource zu existieren aufhört oder unzugänglich wird (Die ISBN zum Beispiel könnte als URN für Bücher verwendet werden, während die Signaturen einer Bibliothek als URLs betrachtet werden können).

URI-Schema Jeder URI gehört zu einem **URI-Schema**, das die Syntax und die Bedeutung des URI weiter einschränken kann. In [RFC2396] werden diejenigen Bestandteile der URI-Syntax definiert, die für alle URI-Schemas gelten müssen oder die von zahlreichen URI-Schemas benutzt werden; mit den darin enthaltenen Informationen können Sie Software, die URI-Verweise analysiert, so schreiben, dass die schemaspezifische Interpretation des URI so lange wie möglich aufgeschoben wird.

Viele URL-Schemas heißen wie Protokolle (zum Beispiel »http«, »ftp«, »telnet«), aber das bedeutet nicht, dass die entsprechenden Ressourcen nur über dieses Protokoll zugänglich sind. Gateways, Proxies und andere »Protokollumsetzer« machen es möglich, auf viele Ressourcen zuzugreifen, egal unter welchem Protokoll sie »ursprünglich« angeboten werden. Zur Auflösung vieler Arten von URLs brauchen Sie auch mehr als ein Protokoll – zum Beispiel ist zum Zugriff auf eine http-Ressource meistens neben HTTP auch noch DNS nötig.

Außerdem kann man noch zwischen **relativen** und **absoluten URIs** unterscheiden. Ein absoluter URI bezieht sich auf eine Ressource ohne Rückgriff auf den Kontext, in dem der URI benutzt wird. Ein relativer URI dagegen beschreibt den Unterschied, der in einem hierarchischen Namensraum zwischen dem aktuellen Kontext und der (absoluten) Bezeichnung der Ressource besteht. Voraussetzung dafür ist, dass das URI-Schema einen hierarchischen Namensraum überhaupt unterstützt – das gilt für http-URIs, aber zum Beispiel nicht für mailto-URIs. Damit ist es möglich, dass eine Ressource, zum Beispiel ein HTML-Dokument, einen Basis-URI vorgibt (etwa `http://www.example.com/blubb/`). Relative URIs in diesem Dokument (etwa `fasel.html#k1`) werden dann an den Basis-URI angehängt, und daraus ergibt sich der endgültige URI (hier `http://www.example.com/blubb/fasel.html#k1`).

 Am Rande bemerkt: Gibt ein HTML-Dokument keinen expliziten Basis-URI vor, dann gilt sein eigener URI als Basis-URI.

Die gängigsten URIs, mit denen Sie es im Web-Umfeld zu tun bekommen, dürfen http-, ftp- und mailto-URIs sein. Hier sind ein paar Beispiele:

```
http://www.example.com/blubb/fasel.html
ftp://ftp.dante.de/tex-archive/tex/latex/base/latex.ltx
mailto:anselm.lingnau@linupfront.de
```

Am Anfang jedes URI steht der Schema-Name, gefolgt von einem Doppelpunkt. Schema-Name
Der Rest des URI wird dann in Abhängigkeit vom Schema-Namen interpretiert; bei mailto-URIs folgt zum Beispiel eine E-Mail-Adresse, bei http-URIs zunächst ein Servername gefolgt von einer genaueren Bezeichnung der Ressource auf dem betreffenden Server.

 Obwohl die Syntax vieler URI-Schemas an die von Unix-Dateinamen erinnert, müssen zum Beispiel http-URIs nicht 1 : 1 mit Dateien auf dem betreffenden Server korrespondieren (auch wenn es oft mehr oder weniger auf so- was heraus läuft). Es wurde eben einfach festgelegt, dass in URIs der Schrägstrich (»/«) als Trennzeichen in hierarchischen Namen verwendet wird – wie ein Server das tatsächlich implementiert, bleibt ihm überlassen.

 Es gibt diverse Einschränkungen dafür, welche Zeichen in URIs erlaubt sind und welche »maskiert« werden müssen. Außerdem dürfen zum Beispiel http-URLs noch diverse Elemente außer Servernamen und URI-»Rest« enthalten. Zeichen in URIs

Übungen

 1.1 [!1] Welche URI-Schemas kennen Sie? Handelt es sich dabei um URLs oder URNs?

 1.2 [2] Wie könnte ein hierarchisches URI-Schema für Telefonanschlüsse aussehen?

1.3 Das Hypertext Transfer Protocol

Grundlage der Kommunikation zwischen Web-Browsern und -Servern ist das **Hypertext Transfer Protocol**, kurz HTTP. Es wurde von Tim Berners-Lee und seinen Kollegen am CERN speziell für das WWW entwickelt und seitdem sukzessive weiterentwickelt. Erste Verbreitung fand das Protokoll in der (für heutige Begriffe extrem primitiven) Version 0.9; die Version 1.0 [RFC1945] enthielt große Mengen neuer Funktionalität, die in der heute gültigen Version 1.1 [RFC2616] noch erweitert und ausgereift wurde.

Hypertext Transfer Protocol

HTTP beruht auf TCP/IP und verwendet normalerweise auf der Serverseite TCP/IP den TCP-Port 80. Das Protokoll ist sehr einfach strukturiert; der Browser öffnet eine TCP-Verbindung zum Server und schickt eine Anfrage (**HTTP-Request**). Das kann eine Bitte um die Übersendung einer HTML-Datei oder Graphik sein, aber auch die Rücksendung eines Formulars oder das Einreichen eines neuen Dokuments zur Veröffentlichung auf dem Server. Der Server beantwortet die Anfrage (**HTTP-Response**) und baut anschließend die Verbindung ab.

TCP/IP

HTTP-Request

HTTP-Response

In HTTP/1.0 wird für jedes zu übertragende Objekt eine eigene TCP-Verbindung auf- und abgebaut. In vielen Fällen (typisch: eine HTML-Seite mit 35 kleinen Graphiken) ist das sehr ineffizient, da das Auf- und Abbauen von TCP-Verbindungen ein relativ »teurer« Vorgang ist, vor allem wenn pro Verbindung nur wenige Nutzdaten transportiert werden. HTTP/1.1 definiert darum einen **Keepalive**-Mechanismus, mit dem mehrere Objekte nacheinander über eine einzige TCP-Verbindung abgerufen werden können. Dies kann benutzt werden, wenn Browser und Server beide HTTP/1.1 unterstützen – was heute in der Regel gegeben ist.

Keepalive

Tabelle 1.1: HTTP/1.1-Methoden

Methode	Bedeutung
OPTIONS	Fragt Informationen über Ressourcen und Kommunikationsparameter ab
GET	Holt die durch den URI bezeichnete Ressource
HEAD	Wie GET, holt aber nur Metainformation
POST	Gibt die Informationen in der Anfrage an die durch den URI bezeichnete Ressource weiter
PUT	Bittet um Speicherung der Informationen in der Anfrage an der durch den URI bezeichneten Stelle
DELETE	Bittet um Löschung der durch den URI bezeichneten Ressource
TRACE	Schickt die Anfrage postwendend zurück; dient zur Fehlerdiagnose
CONNECT	Reservierter Name

Tabelle 1.2: Struktur der HTTP-Statuswerte

Status	Erklärung
1xy	Information – Die Anfrage wurde empfangen und wird bearbeitet
2xy	Erfolg – Die Anfrage wurde erfolgreich empfangen, verstanden und akzeptiert
3xy	Umleitung – Es muss noch etwas getan werden, um die Anfrage vollständig zu bearbeiten
4xy	Client-Fehler – Die Anfrage ist syntaktisch falsch oder kann nicht beantwortet werden
5xy	Server-Fehler – Der Server konnte eine anscheinend gültige Anfrage nicht bearbeiten

Kopfzeilen HTTP-Anfragen und HTTP-Antworten sind sehr ähnlich aufgebaut: Beide beginnen mit einer einführenden Zeile, die die Methode und URI der Anfrage bzw. einen Antwortcode enthält, gefolgt von einer Reihe von **Kopfzeilen** (engl. *headers*). Dann kommt eine Leerzeile und anschließend (falls erforderlich) die »Nutzdaten«.

Die Anfangszeile einer HTTP-Anfrage ist wie folgt aufgebaut:

<Methode> <URI> HTTP/<Version>

Methoden Die *<Methode>* gibt dabei an, was mit dem Rest der Anfrage passieren soll; die gängigste Methode, **GET**, fordert die durch den *<URI>* bezeichnete Ressource an. Andere wichtige Methoden sind **HEAD** (entspricht **GET**, aber nur die Antwortzeile und die Header-Zeilen werden zurückgeschickt, nicht die Nutzdaten) oder **POST** (zur Einreichung von Formularinhalten und ähnlichem). Tabelle 1.1 zeigt die in HTTP/1.1 definierten Methoden, Browser und Server dürfen aber auch eigene, neue Methoden einführen.

Die erste Zeile der HTTP-Antwort sieht so aus:

HTTP/<Version> <Status> <Antworttext>

Status und Antworttext Der *<Status>* ist immer eine dreistellige Zahl, die die Reaktion des Servers auf die Anfrage codiert. Der *<Antworttext>* beschreibt die Reaktion in einer für Menschen verständlichen Weise. So steht zum Beispiel der Status »200« für »OK« (die Anfrage wurde ausgeführt und Daten werden zurückgeschickt) oder »404« für »Not Found« (die durch den URI in der Anfrage beschriebene Ressource konnte nicht gefunden werden). Die Antworttexte können von Implementierung zu Implementierung verschieden sein; nur die Statuswerte und ihre Bedeutung sind

festgeschrieben. Tabelle 1.2 bietet einen Überblick über die Struktur der HTTP-Statuswerte; eine vollständige Liste ist in Anhang B enthalten.

Die Kopfzeilen in HTTP-Anfragen und HTTP-Antworten lassen sich einteilen in **allgemeine Kopfzeilen** (engl. *general headers*), **Objekt-Kopfzeilen** (engl. *entity headers*), **Anfrage-Kopfzeilen** (engl. *request headers*) und **Antwort-Kopfzeilen** (engl. *reply headers*). Anfrage-Kopfzeilen sind nur in HTTP-Anfragen, Antwort-Kopfzeilen nur in HTTP-Antworten erlaubt.

HTTP-Kopfzeile, allgemeine
HTTP-Kopfzeile, Objekt-
HTTP-Kopfzeile, Anfrage-
HTTP-Kopfzeile, Antwort-

Allgemeine Kopfzeilen Allgemeine Kopfzeilen sind für Anfragen und Antworten erklärt und beziehen sich auf die übertragene Nachricht, nicht deren Inhalt. Dazu gehören die Zeitangabe der Erstellung des HTTP-Pakets (Anfrage oder Antwort, *Date*) und die Transfercodierung (Transfer-Encoding), die Steuerung von Verbindungsabbau bzw. *keepalive* (Connection), die Steuerung von Cachespeichern (Cache-Control, *Via*) sowie der Umgang mit Protokollversionen (Upgrade) und diversen Zusätzen (Pragma).

Objekt-Kopfzeilen Objekt-Kopfzeilen enthalten Informationen über die übertragenen Nutzdaten oder, falls keine Nutzdaten übertragen werden, über die durch die Anfrage adressierte Ressource. Dazu gehören die erlaubten Zugriffsmethoden (Allow, siehe Tabelle 1.1), verschiedene Informationen über die übertragenen Daten wie Komprimierung oder andere Formen der Codierung (Content-Encoding), Datentyp und Sprache (Content-Type, Content-Language), Größe in Oktetten (Content-Length), eine Ortsangabe (URI) der mit der Anfrage korrespondierenden Ressource (Content-Location, muss nicht identisch zum URI in der Anfrage sein), eine Prüfsumme (Content-MD5) oder eine Bezeichnung des übertragenen Datenbereichs, wenn es sich nur um einen Ausschnitt einer größeren Ressource handelt (Content-Range). Dazu kommen noch Kopfzeilen, die vor allem für Cachespeicher von Bedeutung sind, wie ein Verfalldatum (Expires) oder den Zeitpunkt, an dem die Ressource zum letzten Mal geändert wurde (Last-Modified).

Anfrage-Kopfzeilen Mit Anfrage-Kopfzeilen kann die Art der Anfrage genauer spezifiziert werden, oder es können Wünsche über das Format der angefragten Ressource geäußert werden. Beispielsweise kann der Browser signalisieren, welche Zeichensätze, Codierungsverfahren und Sprachen der Anwender bevorzugt (Accept-Charset, Accept-Encoding, Accept-Language), und den Server spezifizieren, für den die Anfrage gemeint ist (Host, wichtig im Umgang mit virtuellen Servern – siehe Kapitel 7). Außerdem können die E-Mail-Adresse des Benutzers und Name und Versionsnummer seines Browsers übertragen werden (From, User-Agent), genau wie Authentisierungsinformationen (Authorization, siehe Kapitel 6, und Proxy-Authorization). Die Referer-Zeile gibt Auskunft darüber, in welcher Ressource (zum Beispiel HTML-Datei) der gerade angefragte URI zu finden war. Ferner gibt es verschiedene Möglichkeiten, eine Anfrage nur unter gewissen Bedingungen (komplett) auszuführen, etwa nur, wenn eine Ressource seit einem bestimmten Zeitpunkt verändert wurde (If-Modified-Since), existiert oder nicht existiert (If-Match, If-None-Match). Mit Range kann ein Browser nur einen Teil einer größeren Ressource anfordern, und über If-Range ist es möglich, fehlende Teile nachzuordnen, wobei der Server unaufgefordert die ganze Ressource schickt, wenn sie sich seit einem gegebenen Zeitpunkt geändert hat. Die Max-Forwarders-Zeile dient zu Diagnosezwecken mit der Methode TRACE.

Antwort-Kopfzeilen Diese Kopfzeilen beschreiben die Antwort genauer, die der Server auf eine HTTP-Anfrage hin verschickt. Welche Antwort-Kopfzeilen enthalten sind, hängt von der Art der Antwort ab. Hierzu gehören URI-Angaben für umgeleitete Zugriffe (Location), für Cachespeicher interessante Informationen wie das (geschätzte) Alter der Antwort (Age) oder eine eindeutige, vom URI unabhängige Identifizierung der angesprochenen Ressource inklusive ihres Inhalts (ETag), Informationen für die Au-

thentisierung (WWW-Authenticate und Authentication-Info – siehe Kapitel 5 –, Proxy-Authenticate, Proxy-Authenticate-Info) und die Verhandlung über Datenformate, Sprachen und ähnliches (Vary). Außerdem kann der Server angeben, welches Programm in welcher Version er ist (Server) und in Überlastsituationen o.ä. dem Browser mitteilen, zu welchem Zeitpunkt er eine abgewiesene Anfrage wiederholen soll (Retry-After). Schließlich gibt es noch die Möglichkeit, dass Cachespeicher und Proxies Warnungen an den Browser weiterreichen (Warning).

Die genaue Bedeutung der jeweiligen Kopfzeilen entnehmen Sie am besten [RFC2616].

Im einfachsten Fall besteht eine HTTP-Anfrage nur aus der einleitenden Zeile. Es ist also ohne weiteres möglich, auf einen HTTP-Server testweise über telnet (oder netcat) zuzugreifen:

```
$ telnet localhost http
Trying 127.0.0.1
Connected to test.example.com
Escape character is '^]'
GET /bla.html HTTP/1.0

HTTP/1.1 200 OK
Date: Tue, 13 Oct 2009 11:52:16 GMT
Server: Apache/2.2.13 (Debian)
Last-Modified: Wed, 23 Jan 2008 23:25:04 GMT
```

... und so weiter ...

Cookies

Eine sehr wichtige Feststellung ist, dass HTTP *zustandslos* ist. Das heißt, jede Anfrage steht für sich alleine – das Protokoll hat kein »Gedächtnis«, mit dem es die Eigenschaften oder Ergebnisse früherer Anfragen bei der aktuellen Anfrage berücksichtigen könnte. Das macht das Protokoll sehr einfach, aber natürlich ist es keine Hilfe in Fällen, wo etwas Gedächtnis nützlich wäre (etwa bei den beliebten virtuellen »Einkaufskörben«). Normalerweise zieht man sich aus der Affäre, indem man versucht, die Benutzer eindeutig zu identifizieren (beispielsweise durch sogenannte **Cookies**), und sich außerhalb des HTTP-Servers, etwa in einer Datenbank, die nötigen Informationen über sie merkt. Mit HTTP hat das dann nur noch indirekt etwas zu tun.

Google Mail



Eine relativ neue Entwicklung ist »AJAX« – im Grunde und etwas salopp gesagt die Idee, nicht immer komplett neue Seiten vom HTTP-Server anzufordern, sondern sich bei Bedarf, etwa als Reaktion auf Benutzereingaben, Informationen zu holen und diese dann auf der Client-Seite in die aktuell angezeigte Seite einzubauen, üblicherweise mit der in den meisten Browsern vorhandenen Programmiersprache JavaScript. Die einfachsten Anwendungen von AJAX sind zum Beispiel Texteingabefelder in Formularen, die eine angefangene Benutzereingabe aus einer Datenbank auf dem Server vervollständigen. Im Extremfall gestattet es AJAX, im Browser »Anwendungsprogramme« zu implementieren, die sich in ihrem Bedienungskomfort nicht mehr maßgeblich von »echten« grafischen Programmen auf dem Desktop unterscheiden. Populäre Beispiele sind etwa *Google Mail* oder die anderen officeartigen Dienste von Google. Aktuelle Entwicklungen in HTML, etwa die Unterstützung von Bereichen, die frei programmierbare Grafik enthalten können (Stichwort »Canvas«), unterstützen diesen Ansatz, der in der Zukunft sicher noch weiter an Bedeutung gewinnen wird.

ChromeOS
Firefox OS



Konsequent verfolgt führt diese Philosophie zu Rechnern, auf denen außer einem Browser nichts Wesentliches läuft. Systeme, die einen solchen Ansatz verfolgen, sind zum Beispiel Googles ChromeOS oder (für Smartphones) das Firefox OS von Mozilla. (Diese Systeme basieren »unter der Motorhaube« normalerweise auf Linux, auch wenn der Benutzer davon so wenig wie möglich gezeigt bekommt.)

Übungen



1.3 [!2] Rufen Sie die Indexseite des Web-Servers auf Ihrem Rechner (oder irgendeine andere interessante Web-Seite) über telnet ab. Studieren Sie die zurückgegebenen Antwort-Kopfzeilen. Versuchen Sie auch einmal die Methode HEAD statt GET.

1.4 Markup-Sprachen

HTML – die Sprache des World-Wide Web – ist eine **Markup-Sprache**: Sie dient dazu, Texte mit Informationen über ihre Struktur und ihr Aussehen auf dem Bildschirm »auszuzeichnen« (engl. *to mark up*). Programme wie Browser interpretieren die Auszeichnungen und berücksichtigen sie bei der Ausgabe.

HTML: Markup-Sprache

Grundsätzlich unterscheidet man **logische Auszeichnung** (engl. *logical markup*) und **visuelle Auszeichnung** (engl. *visual markup*). Während bei visueller Auszeichnung das *Aussehen* von Textelementen wie Überschriften beschrieben wird – Schriftart, -größe und -grad –, dient logische Auszeichnung zum Kenntlichmachen ihrer *Bedeutung*. Eine Überschrift würde der Autor beispielsweise nur als »Überschrift« kennzeichnen und keine Vorschriften über ihre tatsächliche Darstellung machen.

logische Auszeichnung

visuelle Auszeichnung

HTML war ursprünglich vor allem als Sprache für logische Auszeichnung konzipiert (mit einigen visuellen Elementen). In der Zwischenzeit wurde HTML allerdings um diverse weitere visuelle Auszeichnungselemente erweitert, wobei die Ausrichtung auf logische Auszeichnung weitgehend auf der Strecke blieb. Dies ist ein Problem vor allem für Anwender mit ungewöhnlichen Anforderungen: So verwenden zum Beispiel Sehbehinderte oft Software, die Web-Seiten vorlesen kann. Ein solches Programm kann eine offensichtliche »Überschrift« ohne weiteres als neuen Abschnitt ankündigen, während es schwieriger ist, aus »Umschalten auf fette 25-Punkt-Times und neue Zeile anfangen« auf einen Abschnittsanfang zurückzuschließen.

Die visuellen Auszeichnungselemente neigen nicht nur dazu, die »Nutzlast« einer HTML-Seite zu überfrachten, sondern unterscheiden sich nach Art und vor allem Wirkung zwischen verschiedenen Browsern. Es kann also sein, dass eine Seite, die mit Browser X sehr gefällig aussieht, von Browser Y furchtbar verhackstükt wird – ein Umstand, an den sich auch die Kunden von Web-Designbüros mühevoll gewöhnen müssen. In letzter Konsequenz führt das zu Entgleisungen wie Seiten, die »optimiert für Browser X bei einer Auflösung von 1024×768 Punkten« sind.

Zum Glück ist dieser Trend inzwischen weitgehend umgekehrt. Die Standardisierung von CSS (cascading style sheets) ermöglicht es, Inhalt und Darstellung wieder voneinander zu entkoppeln. Während im HTML-Dokument nur logisch ausgezeichnet wird, wird im *style sheet* – einer separaten Ressource, die typischerweise ebenfalls über HTTP vom Server geladen wird – eine Formatierung z. B. für Überschriften vorgeschlagen, die der Browser dann umsetzt. Das *style sheet* kann für mehrere HTML-Seiten gelten und räumt einem Designer weitaus mehr Möglichkeiten ein, als er direkt über HTML hätte, etwa wenn es um die Positionierung von Material auf der Seite geht.

CSS



Der sozusagen »klassische« Aufbau einer Webseite besteht zum Beispiel aus einem Kopfbereich oben, einem Navigationsbereich links und dem eigentlichen Inhalt darunter und rechts davon. Früher (vor CSS) realisierte man diese Sorte Layout über geschickt definierte HTML-Tabellen, was diverse Nachteile mit sich brachte. Zum Beispiel mussten Kopfzeile und Navigationsbereich textuell vorne im HTML-Dokument stehen, was wieder zu Schwierigkeiten mit den Vorleseprogrammen für Blinde führte. Heute erlaubt CSS eine flexible Positionierung der verschiedenen Bereiche unabhängig von ihrer Reihenfolge im HTML-Dokument, so dass es ohne weiteres

möglich ist, den Navigationsbereich *hinter* dem Nutzinhalt im Dokument zu haben, auch wenn er vom Browser am üblichen Platz (links) dargestellt wird.



Über CSS ist es möglich, für die Bildschirmdarstellung eines HTML-Dokuments andere Formatierungsvorgaben zu machen als für dessen Präsentation mit einem Beamer oder den Ausdruck auf Papier. Als Anwender hat man außerdem die Möglichkeit, die Vorgaben des Dokuments durch seine eigenen Vorgaben zu erweitern oder zu ersetzen (ebenfalls wieder nützlich für Sehbehinderte).

HTML-Elemente

Zu den in HTML verfügbaren Auszeichnungselementen gehören:

- Textstrukturierungselemente, darunter solche für Überschriften, Absätze, Listen, Zitate
- Strukturierungselemente auf Absatzebene, etwa für Emphase, Programmtexte, Definitionen, Abkürzungen und anderes
- Verweise auf andere Stellen im Dokument oder auf andere Ressourcen
- Tabellen
- Formulare
- Graphiken

Dazu kommen noch verschiedene Elemente zur internen Steuerung, etwa zur Angabe des Dokumenttitels oder zur Definition eines Basis-URI (siehe S. 4).

SGML

Die Wurzeln von HTML liegen in der **SGML** (*Standard Generalized Markup Language*), die während der 1980er standardisiert wurde (ISO 8879:1986). SGML ist – entgegen ihrem Namen – selbst keine Auszeichnungssprache, sondern eher ein Rahmenwerk zur Definition von Auszeichnungssprachen mit ähnlicher Syntax.

DTD

Eine **DTD** (*Document Type Definition*) gibt dabei die verschiedenen Auszeichnungselemente und ihre Beziehungen vor; eine DTD für Bücher könnte beispielsweise Auszeichnungselemente für Kapitel, Abschnitte und Absätze vorsehen und verlangen, dass ein Kapitel eine Folge von einem oder mehreren Abschnitten ist, die ihrerseits aus einem oder mehreren Absätzen bestehen. SGML macht es möglich, Dokumente, die verschiedenen DTDs folgen, mit einheitlichen Werkzeugen wie Editoren oder Formatierungsprogrammen zu bearbeiten.

HTML = SGML-Anwendung

Validierung

HTML ist genaugenommen nicht, wie manchmal behauptet wird, ein »Dialekt« von SGML, sondern eine SGML-»Anwendung«: Die Syntax von HTML ist in einer SGML-DTD definiert. Das macht es möglich, HTML-Dokumente automatisch auf Konformität zur HTML-DTD zu prüfen (zu **validieren**). Damit sind Sie aber auch an den Vorrat von Auszeichnungselementen gebunden, den die HTML-DTD anbietet. Die Idee der strukturellen Auszeichnung stößt dort an ihre Grenzen, weil es keine Möglichkeit gibt, den Elementvorrat von HTML an die Struktur verschiedener Datenbestände anzupassen, die Sie auf dem Web veröffentlichen möchten.

XML

Die Antwort auf dieses Problem heißt **Extensible Markup Language** (XML). XML ist eine »abgespeckte« Form von SGML, die auf diverse problematische Bereiche verzichtet; in SGML gibt es beispielsweise verschiedene Methoden der Eingabevereinfachung, so daß etwa Elemente automatisch beendet werden, wenn das aus dem Kontext bzw. der DTD eindeutig hervorgeht. So sind in HTML (das einige dieser Vereinfachungen zuläßt) die Textausschnitte

```
<table><tr><td>123</td><td>456</td></tr></table>
```

und

```
<table><tr><td>123<td>456</table>
```

im Hinblick auf die Struktur des beschriebenen Dokuments äquivalent. Das macht es natürlich schwieriger, Software zu schreiben, die HTML-Text korrekt liest. In XML müssen alle Elemente explizit geschlossen werden, so dass nur der erste Ausschnitt erlaubt wäre.

Der Hauptvorteil von XML ist aber, dass die Sprache SGML in einem sehr wichtigen Punkt ähnelt: XML erlaubt die Spezifikation von Auszeichnungselementen und ihren Zusammenhängen in DTDs oder den moderneren »XML Schemas«. Außerdem unterstützt XML über ein **Namensraumkonzept** die Kombination von Auszeichnungselementen aus verschiedenen DTDs im selben Dokument. Damit kann zum Beispiel ein Dokument, das in der XML-basierten HTML-Version **XHTML** geschrieben ist, Graphiken im ebenfalls XML-basierten Vektographikformat **SVG** und mathematische Formeln in der XML-basierten Sprache **MathML** enthalten und trotzdem in seiner Gesamtheit mit allen standardkonformen XML-Werkzeugen bearbeitet werden. Selbst wenn ein Werkzeug mit der *Bedeutung* all dieser Auszeichnungselemente nichts anfangen kann, kann es trotzdem mit Dokumenten umgehen, die sie enthält, denn ihre grundlegende *Syntax* ist immer dieselbe.

Namensraumkonzept
XHTML
SVG
MathML

XML hat sich in den letzten Jahren immer weiter ausgebreitet. Neben diversen problemorientierten Auszeichnungssprachen auf XML-Basis (außer den schon erwähnten XHTML, SVG und MathML gibt es Dutzende von anderen) gibt es zum Beispiel XSLT, eine Sprache zur Beschreibung von Umformungsregeln, mit denen Sie XML-Dokumente in andere XML-basierte Formate, in HTML oder auch andere ASCII-artige Textformate umwandeln können. XSL-FO ist die Basis für die »druckreife« Aufbereitung von XML-Dokumenten. Zahlreiche Anwendungsprogramme verwenden XML-basierte Datenformate, beispielsweise StarOffice oder OpenOffice, und viele Web-Browser unterstützen die direkte Anzeige von XML-Inhalten über XSLT-»Stylesheets«.

problemorientierte Auszeichnungssprachen

 Es ist allerdings nicht damit zu rechnen, dass XHTML in der näheren Zukunft HTML ersetzen wird. Als XHTML im Jahr 2000 veröffentlicht wurde, war das zwar der Plan, aber die Trägheit der Browser-Autoren (allen voran Microsoft), das Format mehr als nur rudimentär zu unterstützen, hat dazu geführt, dass dieses Vorhaben erst einmal ad acta gelegt wurde. Die Weiterentwicklung von XHTML kochte wegen des weitverbreiteten Desinteresses auf der Seite der Browser- und Webseiten-(Erstellungssoftware-)Autoren auf kleiner Flamme.

Der letzte Schrei auf diesem Gebiet ist HTML5, das alle möglichen interessanten und nützlichen Erweiterungen bietet. Neben dem schon früher angesprochenen »Canvas« gehören dazu diverse Multimedia-Möglichkeiten, die lokale Speicherung größerer Datenmengen auf dem Client, Erweiterungen beim semantischen Markup und vieles andere mehr. Verabschiedet hat man sich leider von einer durchgängigen Standardisierung der Sprache; statt dessen werden Zug um Zug Teile für offiziell erklärt, was die Innovation fördern soll, aber dazu führt, dass es schwierig ist, konkrete Kompatibilitätsaussagen für Browser zu machen.

HTML5

 HTML5 unterstützt eine »traditionelle« und eine XML-basierte »Darstellung« für Dokumente, aber behält eine sehr weitreichende Kompatibilität zu früheren HTML-Versionen bei.

HTML lernen Sie am einfachsten durch Ausprobieren, indem Sie interessant aussehende HTML-Seiten studieren (Browser haben normalerweise eine Funktion, mit der Sie sich den HTML-Code zu einer beliebigen im Browser angezeigten Seite holen und anschauen können.) Für einen fundierteren Einstieg gibt es neben Dutzenden von Büchern für jeden Geldbeutel die kostenlos beziehbare Online-Einführung »SELFHTML« von Stefan Münz (<http://de.selfhtml.org/>), die nicht mehr ganz taufrisch ist, aber einen Einstieg in HTML und CSS ermöglicht.

HTML lernen
SELFHTML

Übungen



1.4 [3] Entwerfen Sie eine einfache HTML-Seite, die zum Beispiel Ihren Namen und Ihre Adresse enthält. Verwenden Sie einen URI der Form `file:///home/hugo/myhtml.html` (natürlich mit dem vollen Dateinamen Ihrer Datei), um die Datei in einem Web-Browser anzuzeigen.

1.5 Web-Server und -Browser

- Für viele Anwender ist »das Internet« gleichbedeutend mit dem World Wide Web. Das liegt nicht nur daran, dass das bunte und bequeme WWW vermutlich den »zugänglichsten« Teil des weltweiten Rechnernetzes darstellt, sondern auch daran, dass viele Computerbenutzer kein anderes Programm zum Zugriff auf »das Internet« benutzen als ihren Web-Browser. Schließlich unterstützen diese Programme oft nicht nur das Abrufen von Seiten über HTTP, sondern auch den Empfang, das Lesen, Bearbeiten und Verschicken von elektronischer Post (zusammen mit dem Web sicherlich der wichtigste Internet-Dienst für den durchschnittlichen Anwender). Web-Browser stehen heute für jede nennenswerte Rechnerplattform zur Verfügung, angefangen bei Mobiltelefonen und PDAs, und in den letzten Jahren haben nach kleinen tragbaren PCs, den **Netbooks**, auch »Smartphones« (also Computer im Handy-Format, mit denen man auch telefonieren kann) und »Tablets« wie Apples iPad, die über drahtloses LAN oder Mobilfunkverbindungen fast überall Zugriff auf das Internet erlauben, eine Menge Interessenten gefunden. Im weitesten Sinne als Web-Browser könnte man aber auch Geräte bezeichnen wie zum Beispiel *Least-Cost-Router* für Telefonanlagen, die sich automatisch in periodischen Abständen über das Web mit neuen Datenbanken als Arbeitsgrundlage versorgen.
- Dem gegenüber stehen auf der Anbieterseite die Web-Server. Auch hier gibt es die »eierlegenden Wollmilchsäue« wie den in diesem Kurs besprochenen **Apache**, der eine breite Palette von Ressourcen – von Dateien über dynamische Inhalte via CGI-Skripte oder Module bis hin zu ganzen anderswo liegenden Web-Präsenzen als Proxy-Server – zur Verfügung stellen kann. Wobei ein Web-Server nicht daran gebunden ist, Inhalte aus Dateien zu lesen; große Teile des World Wide Web bestehen heutzutage aus Seiten, die aus Datenbanken dynamisch erzeugt werden. Neben Apache gibt es auch noch andere »allgemein taugliche« Webserver, etwa **IIS** von Microsoft oder von Sun (ehemals Netscape), die aber eine untergeordnete Rolle spielen, sowie eine Reihe von spezialisierteren Produkten. Ferner können viele an ein Rechnernetz angeschlossene Peripheriegeräte wie Drucker oder Router über einen Web-Browser konfiguriert werden. Dieser Web-Browser greift dann auf einen Web-Server zu, der Bestandteil des Gerätebetriebssystems ist. Es gibt fertige, kompakte Web-Server zum Einsatz in »eingebetteten Systemen«.

1.6 Das World-Wide-Web-Konsortium

- Die Weiterentwicklung und Standardisierung des WWW liegt seit 1994 in den Händen des **World-Wide-Web-Konsortiums** (W3C) mit Sitz in Boston, Massachusetts. Aufgabe der Organisation ist es, als Diskussionsforum und durch die Förderung von Interoperabilität die Weiterentwicklung des Web als offene Plattform zu sichern. Zu den langfristigen Zielen des W3C gehören:

Universaler Zugriff Das Web soll für alle zugänglich werden, indem Techniken gefördert werden, die kulturelle, sprachliche, Ausbildungs- und Fähigkeitsunterschiede sowie die verschiedenen materiellen Ressourcen und körperlichen Beschränkungen von Benutzern auf der ganzen Welt berücksichtigen.

Semantisches Web Es soll eine Softwareumgebung geben, die es jedem Benutzer erlaubt, die auf dem Web verfügbaren Ressourcen in der bestmöglichen Weise zu nutzen.

Web des Vertrauens Die Entwicklung des Web soll Rücksicht auf die neuen gesetzlichen, wirtschaftlichen und sozialen Themen nehmen, die durch diese Technik aufgebracht werden.

Das W3C unterstützt »Aktivitäten« in fünf verschiedenen Bereichen: Aktivitäten

Architektur Die unterliegenden Techniken des Web

Dokumentformate Formate und Sprachen, die den Benutzern genaue, schöne und steuerbare Informationen zugänglich machen

Interaktion Verbesserung des Benutzerzugriffs auf das Web, insbesondere im Bereich der Inhaltserstellung

Technologie und Gesellschaft Web-Infrastrukturen für soziale und rechtliche Fragen sowie Fragen öffentlicher Politik

Web-Zugreifbarkeits-Initiative (engl. *Web Accessibility Initiative*) Webzugang für Behinderte

Wichtigstes Ergebnis der Arbeit des W3C sind die **Empfehlungen** (engl. *recommendations*). Hierbei handelt es sich um Standarddokumente, die einen Anspruch auf Verbindlichkeit erheben – sie werden nicht »Normen« (engl. *standards*) genannt, da das W3C im Gegensatz zu Organisationen wie der **International Organization for Standardization** (ISO) keinen staatlichen Auftrag hat – und die typischerweise technische Aspekte des Web regeln. Das W3C hat diverse Empfehlungen herausgegeben, darunter die Spezifikationen für HTML (3.2, 4 und XHTML), für CSS (1 und 2), für XML und dazugehörige Techniken wie »XML Namespaces«, XSLT, XPath oder XLink. Dazu kommen weitere von XML abgeleitete Beschreibungssprachen wie SVG, SMIL oder RDF und wichtige Infrastrukturbestandteile wie PNG, PICS, P3P oder die *Web Accessibility Guidelines*.

Empfehlungen

International Organization for Standardization

Das W3C ist eine herstellerunabhängige Organisation, da keine einzelne Firma die Kontrolle über die Zukunft des Web haben soll. Kooperation mit anderen Normungsgremien wie der IETF (Internet Engineering Task Force), dem WAP-Forum oder dem Unicode-Konsortium ist darum sehr wichtig. Das W3C operiert so weit wie möglich nach dem Konsensprinzip und ermöglicht der Öffentlichkeit, seine Arbeit zu kommentieren.

Herstellerunabhängig

Konsensprinzip

Firmen, Universitäten oder Institutionen der öffentlichen Verwaltung können dem W3C beitreten, was allerdings eine Stange Geld kostet, in Abhängigkeit von der Geografie: Der aktuelle Beitrag für deutsche profitorientierte Organisationen mit einem Jahresumsatz von mindestens 51 Millionen Euro beträgt 68.000 Euro im Jahr, andere deutsche Organisationen sind mit 7.800 Euro im Jahr dabei. Für Firmen und Non-Profits, die selber keine Mitglieder aufnehmen, weniger als 2,25 Millionen Euro im Jahr Umsatz machen und noch nie W3C-Mitglied waren, gilt für die ersten beiden Jahre ein Sonderpreis von 1.950 Euro.

Mitglieder

 Das W3C ist nicht wirklich darauf eingerichtet, Einzelpersonen als Mitglieder zu akzeptieren; wenn Sie das wirklich wollen, sollten Sie sich darauf einstellen, den Beitrag für »andere Organisationen« entrichten zu müssen.

 Zum Glück müssen Sie kein Mitglied sein, um an den Aktivitäten des W3C teilnehmen zu können, etwa durch Spenden, Teilnahme an öffentlichen Mailinglisten oder Workshops, Mitarbeit an Open-Source-Software, die das W3C veröffentlicht, oder Mithilfe bei der Übersetzung von W3C-Dokumenten.

 Die regelmäßige Teilnahme an Arbeitsgruppen, die Standards entwickeln, ist eigentlich ein Vorrecht von W3C-Mitgliedern (bzw. deren Beschäftigten), aber es ist für Sie möglich, als »eingeladener Experte« (*invited expert*) dabei zu sein – was voraussetzt, dass Sie einerseits tatsächlich Experte auf dem Gebiet sind, das die Arbeitsgruppe adressiert, und andererseits Ihr Arbeitgeber keine Organisation ist, die ein finanzielles Interesse am Ergebnis der

Arbeitsgruppe hat (in welchem Fall sie dem W3C beitreten sollte). Eingeladene Experten genießen denselben Zugriff auf die W3C-Webseiten wie Mitglieder, aber es wird erwartet, dass sie sich angemessen ins Zeug legen.

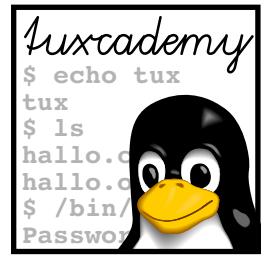
Im Februar 2013 hatte das W3C 385 Mitglieder. – Die Web-Seiten des W3C finden sich unter <http://www.w3.org/>.

Zusammenfassung

- Das World Wide Web wurde 1990 am CERN von Tim Berners-Lee und seinen Kollegen als Kommunikationsmedium für die CERN-Wissenschaftler begonnen
- Seit ca. 1993/94 erregt das Web großes kommerzielles Interesse
- URIs sind Zeichenketten, die Ressourcen identifizieren
- Ein URI, der eine Ressource über ihre Zugriffsdaten identifiziert, heißt URL; ein URI, der eine vom Zugriffsmechanismus unabhängige Identifizierung ermöglicht, heißt URN
- HTTP ist ein einfach strukturiertes Protokoll zur Kommunikation zwischen Web-Browsern und -Servern
- HTML ist die verbreitetste Sprache für Hypertext-Dokumente im WWW
- XML ermöglicht die Definition problemorientierter Auszeichnungssprachen und deren Verarbeitung mit Standardwerkzeugen
- Web-Browser sind Programme, die auf Inhalte im Web zugreifen. Auch andere Programme oder Geräte verwenden das Web
- Neben den gängigen Web-Servern wie Apache enthalten auch viele Geräte oder Softwaresysteme Web-Server, zum Beispiel zur Konfiguration
- Das World-Wide-Web-Konsortium (W3C) koordiniert als herstellerunabhängige Organisation die Weiterentwicklung des Web

Literaturverzeichnis

- BL99** Tim Berners-Lee. *Weaving the Web*. Harper San Francisco, 1999. Deutsch als *Der Web"-Report* (ECON).
<http://www.w3.org/People/Berners-Lee/Weaving/Overview.html>
- Bus45** Vannevar Bush. »As We May Think«. *Atlantic Monthly*, Juli 1945. 176(1):101–108.
<http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm>
- RFC1945** T. Berners-Lee, R. Fielding, H. Frystyk. »Hypertext Transfer Protocol – HTTP/1.0«, Mai 1996. <http://www.ietf.org/rfc/rfc1945.txt>
- RFC2396** T. Berners-Lee, R. Fielding, L. Masinter. »Uniform Resource Identifiers (URI): Generic Syntax«, August 1998. <http://www.ietf.org/rfc/rfc2396.txt>
- RFC2616** R. Fielding, J. Gettys, J. Mogul, et al. »Hypertext Transfer Protocol – HTTP/1.1«, Juni 1999. <http://www.ietf.org/rfc/rfc2616.txt>



2

Der Web-Server Apache

Inhalt

2.1	Die Geschichte von Apache	16
2.2	Installation	17
2.3	Die Konfigurationsdateien	18
2.4	Bearbeitung von Anfragen	19
2.5	Apache-Module	20
2.6	Protokolldateien	21
2.7	Virtuelle Web-Server	21
2.8	Apache-Betrieb	21
2.9	Apache-Informationsquellen	22

Lernziele

- Verstehen, wie die Apache-Entwicklung erfolgt (ist)
- Einen Überblick über die Möglichkeiten und Konfiguration von Apache haben
- Weiterführende Dokumentation zu Apache finden können

Vorkenntnisse

- Überblick über WWW-Begriffe (Kapitel 1)
- Grundkonzepte freier Software

2.1 Die Geschichte von Apache

NCSA-httdp Der Web-Server Apache stammt mittelbar ab vom frei verfügbaren **NCSA-httdp**, den Rob McCool bis Mitte 1994 an der Universität von Illinois in Urbana-Champaign entwickelt hatte. Im Februar 1995 war der NCSA-httdp der beliebteste Web-Server auf dem Internet, aber da er nicht mehr offiziell weiterentwickelt wurde, hatten viele Webmaster ihre privaten Erweiterungen geschrieben. Einige dieser Webmaster beschlossen unter der Ägide von Brian Behlendorf und Cliff Skolnick, sich zusammenzutun und ihre Änderungen zu koordinieren. Ende Februar 1995

Apache Group bestand die ursprüngliche **Apache Group** aus acht Entwicklern und drei Zuarbeitern, die auf der Basis von NCSA-httdp 1.3 alle veröffentlichten Fehlerkorrekturen und sinnvoll wirkenden Erweiterungen zusammenführten, die sie finden konnten. Das Ergebnis wurde nach einigen Tests als erste offizielle Version des Apache-Servers (0.6.2) freigegeben. Zufällig begann auch das NCSA etwa zur selben Zeit mit der Weiterentwicklung des ursprünglichen NCSA-httdp, aber die beiden NCSA-Entwickler schlossen sich als »Ehrenmitglieder« dem Apache-»Stamm« an, um Ideen und Fehlerkorrekturen auszutauschen.

 Für den Namen des Servers gibt es zwei konkurrierende Erklärungen. Die offizielle Lehrmeinung gemäß der Apache-1.3-FAQ [AFAQ] behauptet, dass der Name ausgesucht wurde, um an den Indianerstamm zu erinnern¹. (Brian Behlendorf sagte 2000 in einem Interview, der Name solle etwas andeuten wie »Macht keine Gefangenen, seid ziemlich aggressiv und zeigt's ihnen« [BBLM00] – auch wenn er das inzwischen abstreitet.) Die zweite Interpretation verweist auf den Zustand während der ersten Überarbeitung, als Apache noch »a patchy server« war.

Neue Architektur Im Mai und Juni 1995 begann Robert Thau mit einer völlig neuen, modularen Serverarchitektur, die parallel zur Weiterentwicklung der 0.6-Codebasis implementiert und erprobt und Anfang Dezember 1995 als Apache 1.0 freigegeben wurde. Weniger als ein Jahr nach der Gründung der *Apache Group* überholte Apache den ursprünglichen NCSA-httdp als beliebtester Web-Server, eine Position, die er seitdem hält. Über die Jahre gesehen verwenden meistens mehr Web-Präsenzen auf dem Internet den Apache-Server als alle anderen Web-Server zusammengekommen. Der »Marktanteil« von Apache betrug laut NetCraft im Februar 2013 knapp 55% von 630.795.511 Web-Präsenzen.

 Apache liegt unangefochten an der ersten Stelle, und dann kommt sehr lange nichts. Der nächstpopuläre Web-Server war im Februar 2013 Microsofts IIS mit nicht ganz 17% (Tendenz fallend); der relativ neue Web-Server nginx ist auf Platz 3 mit knapp 13% und legt zu. Betrachtet man nur die Million Web-Präsenzen mit dem meisten Betrieb, ist der Apache-Anteil fast 60%, und IIS und nginx liegen mit jeweils knapp 13% nahezu gleichauf.

Apache Software Foundation 1999 wurde die **Apache Software Foundation** gegründet, die die Weiterentwicklung des Apache-Servers auf eine feste legale und finanzielle Basis stellen sollte. Es gibt eine »Kerngruppe« (engl. *core*) von Entwicklern, die sich um die geschäftlichen Aspekte und gewisse sensitive Dinge wie Sicherheitslücken kümmert; die Hauptentwicklungsarbeit wird von Freiwilligen geleistet, die Codeerweiterungen und -veränderungen vorschlagen, über die die Kerngruppe dann abstimmt.

Apache 2.0 Im Frühjahr 2002 kam Apache 2.0 heraus, eine in zahlreichen Punkten verbesserte Version des Servers. Zum Beispiel ist sie besser skalierbar und enthält eine Infrastruktur zum »Filtern« von generierter Ausgabe. Die nächste Version, 2.2, erschien 2007 und enthält neben verbesserten Cache- und Proxy-Funktionen auch eine flexiblere Programmierschnittstelle für Authentisierung und Autorisierung.

Apache 2.2 Apache 2.4 Heute aktuell ist Apache 2.4, freigegeben Anfang 2012.

¹Unsereiner denkt natürlich gleich an den noblen Winnetou und seine Silberbüchse, aber in den USA, wo die Werke von Karl May nie zu großer Popularität gelangt sind, ist das natürlich kein Faktor.

 Vielleicht fragen Sie sich, was mit Apache 2.1 und Apache 2.3 passiert ist. Die Antwort darauf lautet, dass die Unterversionen mit ungerader Nummer »Entwicklungsversionen« sind, in denen parallel zur »stabilen« Version die nächste stabile Version vorangetrieben wird. Im Moment basteln die Apache-Entwickler also an der Version 2.5 herum, mit Blick auf einen künftigen Apache 2.6. (Wenn Sie schon länger im Geschäft sind, erinnern Sie sich vielleicht noch daran, dass der Linux-Kernel früher eine ähnliche Methodik verfolgte. Seit Linux 2.6 ist das aber nicht mehr so.)

 Die Unterschiede zwischen Apache 2.2 und Apache 2.4 betreffen größtenteils Details (wenn auch mitunter wichtige Details). Wir weisen in dieser Schulungsunterlage bei Bedarf darauf hin.

 Apache 2.2 und Apache 2.0 werden weiter gewartet, auch wenn das Hauptaugenmerk der Entwickler inzwischen auf der Version 2.4 liegt. Wenn Sie also eine Web-Präsenz haben, die auf Apache 2.2 oder gar Apache 2.0 beruht, dann müssen Sie nicht in Panik verfallen. Allerdings sollten Sie mittelfristig über eine Aktualisierung nachdenken.

 Den alten Apache 1.3 sollten Sie nicht mehr verwenden. Zwar erscheinen dafür, falls nötig, noch Sicherheitspatches – zuletzt 2010 –, aber das Programm ist nicht mehr wirklich zeitgemäß.

Der Apache-Server ist **frei verfügbar** und kann beliebig eingesetzt werden, frei verfügbar kommerziell und nichtkommerziell. Er kann sogar als Basis für kommerzielle Softwareentwicklung dienen, wenn gewisse Spielregeln eingehalten werden, die in der *Apache Software License* niedergelegt sind. Dies beschränkt sich im wesentlichen darauf, dass im Namen abgeleiteter Produkte nicht das Wort »Apache« vorkommen darf, und dass in der Dokumentation erwähnt werden muss, dass das Produkt Software der *Apache Software Foundation* enthält. Bei der Weitergabe von Quellcode müssen alle Urhebervermerke erhalten bleiben.

Apache Software License

Die *Apache Software Foundation* kümmert sich inzwischen nicht nur um den HTTP-Server, sondern auch um diverse andere Projekte, beispielsweise **Jakarta**, das sich mit Java-Entwicklungen im Web-Umfeld beschäftigt – Flaggschiff ist **Tomcat**, die offizielle Referenzimplementierung für **Java-Servlets** und **Java Server Pages** – aber auch Apache-Erweiterungen für die Integration von Programmiersprachen wie Perl, Python und PHP und Projekte aus dem XML-Umfeld wie den XSLT-Prozessor **Xalan** oder den Formatierer **FOP**. Außerdem werden regelmäßige Konferenzen für Apache-Entwickler veranstaltet.

Jakarta

Tomcat

Java-Servlets

Java Server Pages

Xalan

FOP

Übungen

 **2.1** [1] Dürfen Sie den Apache-Server in ein von Ihnen entwickeltes Produkt einbauen (beispielsweise einen vorgefertigten »Kommunikationsserver« auf Linux-Basis) und das Produkt anschließend verkaufen, ohne für die Software Lizenzgebühren zahlen zu müssen?

 **2.2** [2] Besuchen Sie die Web-Seiten der *Apache Software Foundation* (siehe Abschnitt 2.9) und machen Sie sich ein Bild von den verschiedenen Projekten, die die Organisation verfolgt.

2.2 Installation

Apache und diverse Zusatzprogramme sind heutzutage fester Bestandteil der meisten Linux-Distributionen. Auf einem Linux-Rechner beschränkt die Installation des Apache-Servers sich also auf das Einspielen der betreffenden Pakete vom Distributionsmedium oder aus dem Internet.

Linux-Distributionen

Quellcode
Unix-Plattformen

Lediglich wenn besondere Eigenschaften gewünscht sind oder wenn eine andere Version als die in der Distribution enthaltene installiert werden soll (was beispielsweise wegen kapitaler Sicherheitslücken in der Distributionsversion notwendig werden kann), ist es erforderlich, Apache aus dem frei verfügbaren Quellcode selbst zu übersetzen und zu installieren. Dasselbe gilt natürlich für Unix-Plattformen, auf denen kein vorübersetzter Apache zur Verfügung steht.

2.3 Die Konfigurationsdateien

Wie die meisten Serverprogramme auf Unix- oder Linux-Systemen wird der Apache-Server über Klartext-Konfigurationsdateien konfiguriert. Diese sind bei »Linux-Distributions-Apaches« normalerweise in einem Verzeichnis unter /etc zu finden. Der genaue Name des Verzeichnisses hängt von der Distribution und der Apache-Version ab. Traditionell enthält eine Datei namens httpd.conf den Hauptanteil der Konfiguration, aber die Distributionen machen das zum Teil erheblich anders.

 Bei den Debian- und Ubuntu-Distributionen steht die Konfiguration für Apache in /etc/apache2 (/etc/apache wurde früher für Apache 1.3 verwendet). Die Datei httpd.conf ist vorhanden, aber leer, und die eigentliche Konfiguration befindet sich einerseits in der Datei apache2.conf und andererseits in weiteren Dateien im Verzeichnis /etc/apache2/conf.d – dies macht es weiteren Softwarepaketen einfacher, sich in die Apache-Konfiguration einzuklinken. Wenn Sie selbst dort Konfigurationsdateien ablegen wollen, dann sorgen Sie dafür, dass ihre Namen mit .local aufhören oder mit local- anfangen.

 Die SUSE-Distributionen platzieren die Apache-Konfigurationsdateien ebenfalls in /etc/apache2. Die wesentliche Konfigurationsdatei ist httpd.conf; große Teile des Rests der Konfiguration sind aber in andere Dateien ausgelagert, vor allem um die Administration von Apache über YaST zu vereinfachen. Der Anfang von httpd.conf enthält einen Kommentar, der die Dateianordnung beschreibt.

 Als Administrator sollten Sie Ihre eigenen Einstellungen in einer Datei (oder Dateien) unter /etc/apache2 unterbringen und den/die Dateinamen in der Variablen APACHE_CONF_INCLUDE_FILES in /etc/sysconfig/apache2 aufzählen.

 Bei Fedora firmiert Apache unter dem Namen httpd (etwas ungeschickt) und legt entsprechend seine Konfigurationdateien unter /etc/httpd ab. Die maßgebliche Konfigurationsdatei ist /etc/httpd/conf/httpd.conf. In /etc/httpd/conf.d befinden sich weitere Konfigurationsdateien, deren Namen auf .conf enden müssen, damit sie beachtet werden. Sie werden in lexikografischer Folge ihrer Namen gelesen.

 Bei einem frisch von Quellcode aus installierten Apache stehen die Konfigurationsdateien, wenn Sie bei der Installation nichts anderes gesagt haben, in /usr/local/apache/conf. Wichtigster Bestandteil der Apache-Konfiguration ist die Datei httpd.conf.

Typische Konfigurationseinstellungen für den Apache umfassen unter anderem:

- Allgemeine (globale) Einstellungen
- Einstellungen für ladbare Module
- Konfiguration des »Hauptservers«
- Konfiguration von virtuellen Servern (optional)

Über diese Datei steht mehr in Abschnitt 3.1.

Manche Apache-Erweiterungsmodule brauchen ihre eigenen Konfigurationsdateien; zum Beispiel verwendet das Modul `mod_mime_magic`, das versucht, Dateitypen aufgrund der Dateiinhalte (anstatt des Dateinamens) zu bestimmen, eine Datei `magic`.

 Wenn Sie einen Server für das SSL-basierte HTTPS-Protokoll verwenden, brauchen Sie ferner noch Dateien mit Schlüsseln, Zertifikaten usw. Mehr Informationen dazu finden Sie in Kapitel 10.

Übungen

 2.3 [1] Wo stehen auf Ihrem Rechner die Konfigurationsdateien des Apache-Servers?

2.4 Bearbeitung von Anfragen

Jede HTTP-Anfrage durchläuft im Apache-Server eine Reihe von separaten Bearbeitungsstufen vom Eingang bis zum Versand der HTTP-Antwort. Diese Stufen sind (in der Reihenfolge der Abarbeitung):

Lesenachbearbeitung (engl. *post-read*): Erste Phase nach der Analyse der Kopfzeilen der Anfrage

URL-Übersetzung (engl. *URL translation*): Der URL der Anfrage wird in einen Dateinamen im Dateisystem des Servers umgesetzt (wobei auch noch andere Sachen passieren können)

Weitergehende Kopfzeilen-Analyse (engl. *header parsing*): Ab dieser Phase können `<Directory>`-Abschnitte (siehe Abschnitt 3.5) benutzt werden

Zugriffskontrolle (engl. *access control*): Wendet Zugriffsvorschriften an, die nicht auf Benutzeridentitäten beruhen (sondern auf IP-Adressen, Rechner- und Domainnamen, ...)

Authentisierung (engl. *authentication*): Prüft die Identität von Benutzern

Autorisierung (engl. *authorization*): Prüft, ob der authentisierte Benutzer auf die Ressource zugreifen darf

MIME-Typprüfung (engl. *MIME type checking*): Bestimmt Attribute der Ressource, etwa den MIME-Dokumenttyp

Reparaturen (engl. *fix-up*): Hier können letzte Anpassungen gemacht werden, bevor die HTTP-Antwort generiert wird

Inhaltserzeugung (engl. *content generation*): Hier wird die HTTP-Antwort für die Anfrage erzeugt

Anfrageprotokollierung (engl. *request logging*): Informationen über die Anfrage und die Antwort werden in die Protokolldatei(en) eingetragen

In Wirklichkeit können Abweichungen von diesem Schema auftreten, wenn Anfragen umgeleitet werden oder ähnliche »ungewöhnliche« Dinge passieren.

2.5 Apache-Module

Module Eine der wichtigsten Eigenschaften des Apache-Servers ist seine Modularität. Fast alle Merkmale des Programms werden durch **Module** realisiert, die sich bei der Übersetzung aus- oder abwählen lassen; die meisten davon können auch zur Laufzeit bei Bedarf nachgeladen werden. Die Apache-Distribution enthält bereits eine Vielzahl von Modulen, die in die folgenden groben Klassen eingeordnet werden können:

Aufbau der Umgebung von Kindprozessen Die Module `mod_setenvif`, `mod_env` und `mod_unique_id` erlauben die Manipulation von Umgebungsvariablen, die für Kindprozesse (etwa CGI-Skripte) sichtbar sind.

URL-Manipulationen Mit `mod_alias` können Sie verschiedene Teile des Server-Dateisystems in die Ressourcenhierarchie des Servers einblenden. `mod_rewrite` schreibt URLs mit regulären Ausdrücken um. Persönliche Verzeichnisse für die Benutzer eines Systems macht `mod_userdir` zugänglich. Mit `mod_speling` werden einfache Tippfehler in URLs automatisch korrigiert.

Zugriffskontrolle Mit dem Modul `mod_authz_host` können Sie den Zugriff auf bestimmte Seiten nur bestimmten Rechnern (identifiziert durch Rechner- oder Domainname oder IP-Adressen) erlauben oder verbieten. Weitere Module (`mod_auth_basic`, & Co.) erlauben die Zugriffskontrolle auf der Basis einer expliziten Benutzerverwaltung, wobei die Benutzerdaten in Textdateien oder verschiedenen Datenbanken abgelegt sein können.

Bestimmung von Datentypen Die Module `mod_mime` und `mod_mime_magic` ordnen Dateien Dateitypen zu, wobei `mod_mime` nur die Dateiendung anschaut, `mod_mime_magic` dagegen den Dateinhalt. (Normalerweise würden Sie `mod_mime_magic` nur für Dateien verwenden, bei denen `mod_mime` nicht funktioniert hat.)

Verzeichnisse Korrespondiert ein URL mit dem Namen eines Verzeichnisses (statt einer Datei), so wird durch das Modul `mod_dir` eine Datei mit einem gegebenen Namen – etwa `index.html` – in diesem Verzeichnis zurückgeliefert. `mod_autoindex` bildet statt dessen eine Liste der Dateien in dem Verzeichnis.

Dynamische Inhalte Das Modul `mod_include` unterstützt *server-side includes*, also HTML-Dateien, bei denen bei der Auslieferung an den Browser gewisse dynamische Elemente ersetzt werden. `mod_cgi` und `mod_actions` dienen zur Ausführung von CGI-Skripten.

Server-Interna `mod_status` und `mod_info` geben dem Systemadministrator Aufschluß über den Zustand des Apache-Servers.

Steuerung von HTTP-Antworten Die Module `mod_headers`, `mod_cern_meta` und `mod_asis` erlauben auf verschiedene Art das Einfügen von beliebigen HTTP-Kopfzeilen in die vom Server zurückgegebenen HTTP-Antworten. Mit `mod_expires` werden automatisch Expires-Zeilen, also Verfalldaten, gemäß verschiedener Kriterien in die Antworten eingebaut.

Protokolldateien `mod_log_config`, und `mod_log_forensic` erlauben die Definition von Protokolldateien in verschiedenen Formaten. Mit `mod_usertrack` können Sie über »Cookies« verfolgen, welcher Benutzer welche Seiten anschaut.

Diverses Mit `mod_imagemap` werden anklickbare Bilder ausgewertet. `mod_proxy` und seine Hilfsmodule geben dem Apache-Server die Möglichkeit, als Proxy (Zwischenspeicher) für Seiten auf anderen Servern zu fungieren oder andere Server transparent in die lokale Dokumentenhierarchie einzubinden. `mod_so` ist notwendig für das dynamische Laden von anderen Modulen zur Laufzeit.

Dazu kommen noch einige andere sehr spezielle Module. Einige Module wurden auch durch neue ersetzt und befinden sich nicht mehr in der Distribution. Außerhalb der Apache-Distribution sind noch weitere Module erhältlich (siehe hierzu Abschnitt 2.9).

Jedes Modul kann Behandlungsroutinen in die verschiedenen Phasen der Anfragebearbeitung einbringen, die der Apache-Server dann nacheinander aufruft. Die meisten Module registrieren nur Routinen für eine oder zwei Phasen.

2.6 Protokolldateien

Im Betrieb schreibt der Apache-Server Dateien, die über die Zugriffe auf die angebotenen Ressourcen Protokoll führen. Üblicherweise ist das zumindest eine Datei namens `access_log`, die die einzelnen Ressourcenzugriffe enthält, sowie eine Datei namens `error_log` für Fehlermeldungen, die bei der Bearbeitung von Anfragen anfallen. Bei »Distributions-Apachen« stehen diese Protokolldateien typischerweise unter `/var/log/apache2` (z. B. Debian GNU/Linux und Ubuntu) oder `/var/log/httpd` (z. B. die SUSE-Linux-Distributionen); bei einem selbst übersetzten Apache können Sie den Ort konfigurieren.

`access_log`
`error_log`

Konfigurieren können Sie auch das Format der Protokolleinträge sowie weitere, spezialisierte Protokolldateien. Es stehen diverse Programme zur Verfügung, die Protokolldateien auswerten und beispielsweise tägliche, wöchentliche oder monatliche Zugriffsstatistiken erstellen.

Protokolldateien und ihre Möglichkeiten werden genauer im Kapitel 6 beschrieben.

2.7 Virtuelle Web-Server

Mit dem Apache-Server ist es möglich, dass dieselbe Apache-Instanz eine Vielzahl von Web-Präsenzen unter verschiedenen Namen verwaltet. Diese **virtuellen Web-Server** können oft sogar dieselbe IP-Adresse benutzen; daneben ist es bei Betriebssystemen wie Linux möglich, dass eine Netzverbindung gleichzeitig für mehrere IP-Adressen zuständig ist.

`virtuelle Web-Server`

Virtuelle Web-Server funktionieren erst mit HTTP 1.1 wirklich gut, was daran liegt, dass erst dort die Host-Kopfzeile eingeführt wurde, die es erlaubt, Anfragen für verschiedene virtuelle Server auf derselben IP-Adresse voneinander zu unterscheiden. Nähere Informationen über virtuelle Web-Server stehen in Kapitel 7.

2.8 Apache-Betrieb

Bestandteil der Apache-Distribution ist ein Programm namens `apachectl`, das dazu dient, den Apache-Server zu steuern. Als »Daemon« hält der Apache-Server ja keine direkte Verbindung zu einem Terminal, wo Sie Kommandos eingeben könnten, sondern operiert losgelöst im Hintergrund.

`apachectl`
`Daemon`

 Bei manchen Distributionen (etwa Debian GNU/Linux oder den SUSE-Distributionen) heißt das Programm `apache2ctl`, um eine Koexistenz verschiedener Apache-Versionen zu ermöglichen.

`apachectl` unterstützt verschiedene Kommandos, die man einzeln oder auch zu mehreren auf der Kommandozeile angeben kann:

start Startet den Apache-Server. Gibt eine Fehlermeldung aus, wenn der Apache-Server schon läuft.

stop Beendet den Apache-Server.

restart Startet den Apache-Server neu; wenn er schon läuft, bekommt er nur ein SIGHUP-Signal geschickt, damit er seine Konfigurationsdateien neu einliest.

graceful Entspricht restart, aber ein laufender Apache-Server bekommt SIGUSR1 statt SIGHUP geschickt. Das bedeutet, dass er die Anfragen, die gerade bearbeitet werden, noch zu Ende bearbeitet, bevor er seine Konfiguration liest. Dies hat Konsequenzen für den Umgang mit Protokolldateien (siehe Abschnitt 6.5).

graceful-stop Entspricht stop, bis darauf, dass gerade bearbeitet werdende Anfragen noch zu Ende geführt werden.

configtest Prüft, ob die Konfigurationsdateien syntaktisch korrekt sind. Damit können Sie natürlich nicht alle Fehler ausschließen, aber dumme Tippfehler in Direktivennamen und ähnliches werden erkannt. Die Kommandos restart und graceful führen zuerst einen configtest aus, damit der Server nicht durch einen trivialen Syntaxfehler angehalten wird, wenn Sie eigentlich wollten, dass er nahtlos mit der neuen Konfiguration weiterläuft.

help Gibt eine kurze Zusammenfassung der möglichen Befehle aus.

status und **fullstatus** geben Statusberichte aus, die Sie sich mit einem Browser anschauen können. Dazu muss das Modul `mod_status` aktiviert sein.

System-V-Init
Init-Skript

Der Apache-Server wird von Linux-Distributionen normalerweise so installiert, dass er beim Systemstart automatisch mit hochgefahren wird. Der von den meisten wesentlichen Distributoren verwendete **System-V-Init**-Mechanismus sieht dafür ein **Init-Skript** vor, das typischerweise in `/etc/init.d/apache` abgelegt wird. Ein systemabhängiger Mechanismus (z. B. `insserv` bei den SUSE-Distributionen, `update-rc.d` bei Debian GNU/Linux) sorgt dafür, dass dieses Init-Skript so in den Systemstart eingegliedert wird, dass zum Beispiel die Netzwerkinfrastruktur beim Apache-Start schon initialisiert ist.

Das Init-Skript kann – ähnlich wie `apachectl` – auch manuell aufgerufen werden. Dabei muss ein Parameter übergeben werden, der angibt, was es tun soll; üblicherweise zur Verfügung stehen »start« und »stop« mit der jeweils naheliegenden Bedeutung, »restart« als Abkürzung für »stop und dann gleich wieder start«, »reload«, um die Konfiguration neu einzulesen, und »status«, um anzufragen, ob der Apache-Server läuft. Die meisten Distributionen stellen außerdem `apachectl` zur Verfügung.

Wenn Sie keinen »Distributions-Apache« verwenden, sondern Apache selber übersetzen wollen, müssen Sie sich selbst um ein Init-Skript kümmern. Schauen Sie nach, ob Sie eine Datei `/etc/init.d/skeleton` haben; hierbei handelt es sich um ein fast komplettes Beispiel-Init-Skript, das Sie nur noch um geeignete `apachectl`-Aufrufe ergänzen müssen.

Übungen



2.4 [1] Starten Sie den Apache-Server auf Ihrem System und vergewissern Sie sich, dass er läuft, etwa indem Sie versuchen, mit einem WWW-Browser auf `http://localhost/` zuzugreifen. Sie können auch mit »`netstat -tl`« schauen, ob ein Dienst den Port 80 (`http`) verwendet.



2.5 [2] Sorgen Sie dafür, dass Apache beim Booten Ihres Systems automatisch gestartet wird. Verwenden Sie dazu, falls angebracht, den entsprechenden Mechanismus Ihrer Distribution.

2.9 Apache-Informationsquellen

Erste Anlaufstelle für Informationen über Apache sind die Seiten der engl. *Apache Software Foundation* unter <http://www.apache.org/>, speziell die Seiten über den Apache-HTTP-Server unter <http://httpd.apache.org>.

Apacheweek Apacheweek (<http://www.apacheweek.com/>) lieferte einst regelmäßig aktuelle Neu-

heiten aus dem Apache-Umfeld. Seit Oktober 2004 wird dieser Dienst nicht mehr weitergeführt, aber die Webseiten sind noch aktiv. Während einiges inzwischen kräftig Moos angesetzt hat, finden sich dort unter anderem gute Listen mit Verweisen auf die zugrundeliegenden RFCs und W3C-Empfehlungen.

Die *Apache Module Registry* (<http://modules.apache.org/>) ist eine Datenbank für Apache-Module, die nicht in der Standarddistribution enthalten sind.

Eine Liste mit weiteren Verweisen auf Apache-Ressourcen auf dem Netz ist http://dir.yahoo.com/Computers_and_Internet/software/internet/world_wide_web/servers/unix/apache/.

Unter <http://www.netcraft.com/survey/> lassen sich die aktuellen »Marktanteile« Marktanteile der verschiedenen Web-Server abrufen.

Übungen



2.6 [2] Wie viele Apache-Module gibt es, die Authentisierungsfunktionen für den Apache über eine Schnittstelle zur Open-Source-SQL-Datenbank MySQL realisieren?



2.7 [1] Welchen »Marktanteil« hatte der Apache-Server in der letzten Netcraft-Erhebung?



2.8 [2] Finden Sie auf den Netcraft-Seiten fünf bekannte Organisationen, die Apache einsetzen.

Kommandos in diesem Kapitel

apache2ctl Bei manchen Distributionen `apachectl` für Apache 2.x

`apache2ctl(8)` 21

apachectl Hilfsprogramm zur Steuerung des Apache-Servers

`apachectl(8)` 21

Zusammenfassung

- Der Apache-Webserver wurde 1995 von einer Gruppe von Web-Administratoren als Weiterentwicklung des NCSA-`httpd` begonnen.
- Heute ist die Weiterentwicklung von Apache die Aufgabe der *Apache Software Foundation*.
- Apache ist frei verfügbar.
- Apache wird über Textdateien konfiguriert, vor allem `httpd.conf` (Name distributionsabhängig).
- Jede HTTP-Anfrage durchläuft im Apache eine Reihe von definierten Phasen der Bearbeitung
- Apache ist modular aufgebaut; die Standarddistribution enthält Module für verschiedene Aufgaben, und weitere Module können über das Internet bezogen werden.
- Apache unterstützt virtuelle Web-Server.

Literaturverzeichnis

AFAQ »Apache Server Frequently Asked Questions«. Gilt für Apache 1.3, den Sie anderweitig nicht mehr benutzen sollten.

[#name](http://httpd.apache.org/docs/1.3/misc/FAQ.html)

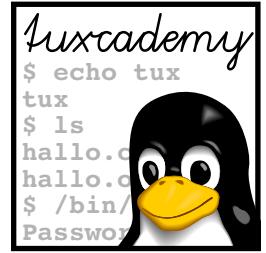
BBLM00 »Apache Power«, April 2000. Ursprünglich ein Artikel im (US-amerikanischen) *Linux Magazine*.

http://web.archive.org/web/20050214074858/http://www.linux-mag.com/2000-04/behlendorf_02.html

ERR02 Lars Eilebrecht, Nikolaus Rath, Thomas Rohde. *Apache Web-Server*. Bonn:

MITP-Verlag GmbH, 2002, 4. Auflage. ISBN 3-8266-0829-1. Enthält eine CD-ROM und eine Referenzkarte.

<http://www.mitp.de/linux/0829/0829.htm>



3

Apache-Konfiguration

Inhalt

3.1	Die Datei <code>httpd.conf</code>	26
3.2	Globale Einstellungen	27
3.3	Dynamisch ladbare Erweiterungen	28
3.4	Konfiguration des »Hauptservers«.	28
3.5	Konfiguration für bestimmte Ressourcen	32

Lernziele

- Einfache Apache-Konfigurationen verstehen und erstellen können
- Syntax der `httpd.conf`-Datei (und ihrer Verwandten) beherrschen

Vorkenntnisse

- WWW- und Apache-Grundbegriffe (Kapitel 1, 2)
- Editieren von Textdateien
- Nützlich: Shellsuchmuster, reguläre Ausdrücke

3.1 Die Datei httpd.conf

Der größte Teil der Apache-Konfiguration erfolgt in der Datei `httpd.conf` (ihr voller Name hängt von Faktoren ab wie der Linux-Distribution oder ob Sie Apache selbst übersetzt haben – bei den SUSE-Distributionen ist es zum Beispiel `/etc/apache2/httpd.conf`, bei Debian GNU/Linux und Ubuntu `/etc/apache2/apache2.conf` und für einen selbst übersetzten Apache meist etwas wie `/usr/local/apache/conf/httpd.conf`). Wir reden ab jetzt von `httpd.conf`, wenn wir »die Datei, die bei Ihrer Distribution die Rolle von `httpd.conf` spielt« meinen.

Informell lässt der Inhalt von `httpd.conf` sich in vier »Abschnitte« untergliedern:

- Globale Einstellungen für den ganzen Apache-Server
- Definitionen für dynamisch ladbare Module
- Konfiguration des »Haupt-Servers«
- Konfiguration etwaiger virtueller Server

 In der Praxis gliedern die meisten Distributionen zumindest Teile davon in untergeordnete Konfigurationsdateien aus.

Um Verwirrung zu vermeiden, steht im folgenden »Apache-Server« für das Programm, das die Inhalte diverser »Web-Server« zur Verfügung stellt. Diese Web-Server sind ein »Haupt-Server« und potentiell viele »virtuelle Server«, wobei weder der eine noch die anderen tatsächlich vorhanden sein müssen – es ist ohne weiteres möglich, auf einem Apache-Server nur virtuelle Server zu installieren, genau wie es möglich ist, ganz auf sie zu verzichten. (Lediglich ein Apache-Server, der weder einen Haupt-Server noch virtuelle Server anbietet, hätte wenig Zweck.) Virtuelle Server sind genauer in Kapitel 7 beschrieben.

Wie in vielen Konfigurationsdateien Unix-basierter Programme ist es möglich,

Kommentarzeilen Kommentarzeilen zu haben, die mit »#« eingeleitet werden. Die standardmäßig mit Apache ausgelieferte `httpd.conf`-Datei macht davon regen Gebrauch. So sind die meisten Einstellungen in dieser Datei sehr umfangreich kommentiert. Leerzeilen werden ignoriert.

Direktiven In der Datei `httpd.conf` stehen Direktiven mit einem oder mehreren Argumenten sowie Blockbegrenzungen, die die Gültigkeit der in den jeweiligen Blöcken enthaltenen Direktiven auf bestimmte Ressourcen beschränken. Die Blockbegrenzungs-Direktiven sind angelehnt an Sprachen wie SGML oder HTML und stehen darum in spitzen Klammern. Details darüber befinden sich in Abschnitt 3.5 und in Kapitel 7.

Blockbegrenzungen Groß- und Kleinschreibung In den Namen von Direktiven ist Groß- und Kleinschreibung irrelevant, allerdings kann es bei ihren Werten darauf ankommen. (Am besten verwenden Sie auch in den Namen die Groß- und Kleinschreibung aus der Dokumentation.) Direktiven dürfen beliebig weit eingerückt werden, um die Konfigurationsdatei zu strukturieren.

Blockbegrenzungs-Direktiven Zeilenstruktur Jede Direktive nimmt genau eine Zeile ein. Reicht eine Zeile nicht aus, so können Sie die nächste Zeile noch dazunehmen, indem Sie ein »\« ans Ende der zu verlängernden Zeile setzen. Zwischen dem Rückstrich und dem tatsächlichen Zeilenende dürfen keine weiteren Zeichen (auch kein Freiplatz) stehen.

Groß- und Kleinschreibung Zeilenstruktur Include Mit der Direktive »`Include <Dateiname>`« können weitere Dateien eingelesen werden – sie werden so interpretiert, als stände ihr Inhalt anstelle der `Include`-Zeile in der Datei.

 Ist `<Dateiname>` in Wirklichkeit der Name eines Verzeichnisses, so liest Apache alle Dateien in diesem Verzeichnis als Konfigurationsdateien ein. Dies macht es zum Beispiel einfacher, große Mengen von virtuellen Servern zu verwalten, indem Sie jeweils die Konfiguration für einen Server in eine kleine Datei in einem solchen Verzeichnis tun (siehe Kapitel 7).

 Seit Apache 2.4 sind Suchmuster erlaubt, à la

```
Include /etc/apache2/local.d/*.conf
```

Dies ist meist eine bessere Idee als das Einlesen kompletter Verzeichnisse über den Verzeichnisnamen. Es könnte nämlich leicht passieren, dass Sie in einem Verzeichnis aus Versehen zusätzliche Dateien herumliegen lassen (etwa Editor-Sicherheitskopien), von denen Ihr Apache Schluckauf bekommt.

 Wenn Sie bei `Include` ein Suchmuster angegeben haben und Apache beim Einlesen der Konfiguration feststellt, dass keine tatsächliche Datei auf das Suchmuster passt, gibt es eine Fehlermeldung. Ebenfalls seit Apache 2.4 gibt es die Direktive `IncludeOptional`, die in diesem Fall einfach weitermacht, ohne sich anzustellen.

`IncludeOptional`

Datei- und Verzeichnisnamen in der Datei `httpd.conf` werden als absolute Namen interpretiert, wenn sie mit einem »/« beginnen. Ist dies nicht der Fall, so wird der Wert der Direktive `ServerRoot` davorgestellt. Der Dateiname `logs/bla.log` würde, wenn `ServerRoot` den Wert `/usr/local/apache` hat, also als `/usr/local/apache/logs/bla.log` interpretiert werden. (Der »/« zwischen `ServerRoot` und dem Dateinamen wird von Apache ergänzt, also sollten Sie keinen Schrägstrich an den Wert von `ServerRoot` anhängen.)

Datei- und Verzeichnisnamen

 Name und Verzeichnis von `httpd.conf` werden bei der Übersetzung von Apache konfiguriert, aber Sie können eine andere Datei verwenden, indem Sie Apache mit der Option »-f `<Datei>`« starten.

Andere Konfigurationsdatei

Änderungen an der `httpd.conf`-Datei werden nur wirksam, wenn Apache neu gestartet wird oder er ein `SIGHUP` oder `SIGUSR1` geschickt bekommt (gegebenenfalls über `apachectl` bzw. `apache2ctl`). Sie können die Syntax einer Konfigurationsdatei prüfen, ohne den Apache wirklich zu starten, indem Sie »`apachectl configtest`« oder Apache mit der Option `-t` aufrufen.

3.2 Globale Einstellungen

Der erste Abschnitt von `httpd.conf` beschäftigt sich mit dem Apache-Server als ganzem (also dem Programm).

Der Wert der Direktive `ServerRoot` dient als »Basis« für relative Datei- und Verzeichnisnamen anderswo in `httpd.conf`. Achten Sie darauf, dass der angegebene Verzeichnisname nicht mit einem »/« endet.

`ServerRoot`

In der in `PidFile` benannten Datei hinterlegt der Apache seine Prozess-ID. Diesen Wert lassen Sie am besten so, wie er ist.

`PidFile`

Die Direktive `ScoreBoardFile` benennt eine Datei, die der Apache zur Kommunikation mit seinen Kindprozessen benutzt. Oft wird diese Datei nicht wirklich benötigt, da die Kommunikation auch in gemeinsamem Speicher stattfinden kann. In jedem Fall sollte jeder auf einem Rechner laufende Apache-Server seine eigene Scoreboard-Datei haben.

`ScoreBoardFile`

Mit `TimeOut` wird die Wartezeit gesetzt, nach der Sende- oder Empfangsverbindungen abgebrochen werden, wenn die Gegenstelle sich nicht meldet. Der Standardwert ist 60 (Sekunden).

`TimeOut`

Die Direktiven `KeepAlive`, `MaxKeepAliveRequests` und `KeepAliveTimeout` regeln den Umgang mit den »Keepalives« von HTTP 1.1. Wenn sie überhaupt unterstützt werden (»`KeepAlive On`«), dann wartet der Server maximal »`KeepAliveTimeout`« Sekunden auf eine weitere Anfrage desselben Browsers auf derselben Verbindung. Pro Verbindung sind maximal `MaxKeepAliveRequests` Anfragen erlaubt, danach wird sie zwangsweise geschlossen. (Der Wert »0« bedeutet »unendlich viele«.)

`KeepAlive`

`MaxKeepAliveRequests`

`KeepAliveTimeout`

Apache unterstützt mehrere Parallelisierungsmodelle, die in Form von *multi-processing modules* oder **MPMs** definiert werden. Diese haben ihre eigenen Direktiven, die in Kapitel 9 beschrieben werden.

`MPM (Apache)`

Übungen



3.1 [!1] Was ist auf Ihrem System der Wert der Konfigurationsdirektive `ServerRoot`?

3.3 Dynamisch ladbare Erweiterungen

Es ist möglich, Apache so zu übersetzen, dass die meisten Module erst beim Start des Programms »dynamisch« eingebunden werden. Das eigentliche Apache-Programm ist dann sehr klein und kompakt und dient nur als »Fundament« für die verschiedenen Module.



Voraussetzung dafür ist, dass das Modul `mod_so static` in das Apache-Programm eingebunden wird – sonst können keine dynamisch ladbaren Erweiterungen gelesen werden, und Sie bekommen ein Henne-Ei-Problem.

LoadModule Am Ende der globalen Einstellungen enthält die Apache-Konfigurationsdatei eine Liste der einzulesenden Module. Die `LoadModule`-Direktive gibt den Modulnamen und die für dieses Modul einzubindende Datei an.



Hier unterscheiden sich die Linux-Distributionen: Während bei den SUSE-Distributionen eine ziemlich große Anzahl von Modulen »ab Werk« aktiviert wird, macht zum Beispiel Debian GNU/Linux nur eine Auswahl tatsächlich verfügbar.

LoadFile Mit der Direktive `LoadFile` ist es möglich, zusätzlichen Code zu laden, der kein Apache-Modul implementiert, aber z. B. von einem solchen benötigt wird.

Übungen



3.2 [!2] Schauen Sie nach, welche Module Ihr Apache-Server einbindet. Welche werden weggelassen? Wie können Sie weitere Module selbst einbinden?



3.3 [2] Welche Vorteile und Nachteile hat der modulare Aufbau des Apache-Servers?

3.4 Konfiguration des »Hauptservers«

Bevor der Apache-Server Anfragen bearbeiten kann, muss er an einem TCP-Port des Serverrechners »lauschen«. Da ein einziger Rechner nicht nur auf den verschiedenen TCP-Ports, sondern auch auf mehreren IP-Adressen (auf verschiedenen oder derselben Netzwerkkarte) Dienste anbieten kann, sieht der Apache-Server hier umfassende Konfigurationsmöglichkeiten vor.

Listen Mit Listen können Sie festlegen, an welcher Adresse und an welchem Port der Server »lauschen« soll. Die Direktive ist »vorgeschrrieben«, muss also in der Konfiguration auftauchen. Wenn Sie nur eine Portnummer angeben (typischerweise 80), dann öffnet Apache diesen Port auf allen Netzchnittstellen des Systems. Listen darf auch mehrmals in der Konfigurationsdatei auftauchen, wenn Sie gezielt bestimmte IP-Adressen oder mehrere verschiedene Portnummern verwenden wollen (Stichwort: Port 443 für HTTPS).



Wenn Sie den Apache auf ungewöhnlichen Ports lauschen lassen wollen, müssen Sie möglicherweise das Protokoll angeben, das er auf dem betreffenden Port erwartet, etwa wie in

Listen 192.168.1.1:10443 https

Die Vorgabe für das Protokoll ist `https` für Port 443 und `http` für alle anderen Ports.

 Wenn Sie mehrere Listen-Direktiven angeben, die sich auf dieselbe Kombination aus IP-Adresse und Port beziehen, gibt es eine Fehlermeldung à la »Adresse schon belegt«. Das gilt auch für Fälle wie

```
Listen *:80
Listen 192.168.1.1:80
```

 Falls Apache sich über eine belegte Adresse beschwert, kann das auch daran liegen, dass ein anderes Programm den Port belegt hat. Sie können mit Kommandos wie

```
# netstat -tlp
```

oder

```
# lsof -i:80
```

den »Schuldigen« überführen.

 Die dritte gängige Fehlerquelle ist, dass Sie eine Adresse angeben, die auf Ihrem Rechner überhaupt nicht konfiguriert ist. Benutzen Sie

```
# ifconfig -a
```

oder

```
# ip addr list
```

um herauszufinden, was es alles gibt.

Die Direktive `ServerName` gibt den Namen an, den der Server in den Umleitungs- URLs verwenden soll, die er generiert. Ist keine `ServerName`-Direktive angegeben, entnimmt der Server seinen Namen dem DNS.

 Sie können beim Servernamen auch ein Protokollschemma und/oder eine Portnummer angeben. Das ist nützlich, wenn Ihr Apache zum Beispiel hinter einem Lastverteiler steht, der SSL verarbeitet und mit Ihrem Apache nur über normales HTTP redet. In diesem Fall können Sie mit etwas wie

```
ServerName https://secure.example.com
```

dafür sorgen, dass trotzdem der korrekte Servername in Umleitungs-URLs auftaucht.

Mit der Direktive `UseCanonicalName` können Sie die Konstruktion von »selbstbezüglichen« URLs steuern, also solchen, die der Apache-Server automatisch erzeugt, um auf sich selbst zu verweisen: Hat die Direktive den Wert »`On`«, werden Rechnername und Port von `ServerName` benutzt. Hat sie den Wert »`Off`« werden der Wert der HTTP-Kopfzeile `Host` und der Port, auf dem die Anfrage einging, benutzt (ansonsten wird auf `ServerName` zurückgegriffen). Schließlich gibt es noch den Wert »`DNS`«; in diesem Fall wird ein Reverse-Lookup für die Zieladresse der HTTP-Anfrage abgesetzt und der daraus resultierende Name zusammen mit dem Port der Anfrage verwendet. Das brauchen Sie aber nur für uralte Clients.

`UseCanonicalName`

 Heutzutage ist es üblich, dass Clients gemäß HTTP 1.1 mit einer Host-Kopfzeile anzeigen, welchen Server sie ansprechen wollen. Allerdings sollten Sie dieser Angabe nicht unbedingt blind vertrauen, da Clients Ihnen prinzipiell das Blaue vom Himmel herunterlügen können. Dies ist vor allem dann ein potentielles Problem, wenn Sie keine virtuellen Server verwenden und ein CGI-Skript o. ä. sich den Wert von `HTTP_HOST` anschaut.

UseCanonicalPhysicalPort

In Analogie zu `UseCanonicalName` gibt es die Direktive `UseCanonicalPhysicalPort`, mit der Sie steuern können, wo Portnummern für selbstbezügliche URLs hergenommen werden. Die Vorgehensweise ist wie folgt: Hat `UseCanonicalName` den Wert »`0n`«, so prüft Apache zunächst, ob in `ServerName` ein Port angegeben wurde, dann übernimmt er, falls `UseCanonicalPhysicalPort` »`0n`« ist, den physikalischen Port aus der Anfrage, sonst den Standardport (80). Hat `UseCanonicalName` den Wert »`0ff`« oder »`DNS`«, wird zuerst auf den Port aus der Host-Kopfzeile zurückgegriffen, dann – falls `UseCanonicalPhysicalPort` »`0n`« ist – auf den physikalischen Port aus der Anfrage, dann auf den Port aus `ServerName` und schließlich, falls nötig, auf den Standardport. Ist `UseCanonicalPhysicalPort` auf »`0ff`« gesetzt, wird der tatsächliche physikalische Port aus der Anfrage nie benutzt.

In automatische Seiten eingebaut wird gerne auch eine Kontaktadresse für den `ServerAdmin`. Diese E-Mail-Adresse kann mit der Direktive `ServerAdmin` angegeben werden.

 Eigentlich können Sie mit `ServerAdmin` einen beliebigen URL angeben. Wenn Apache den Parameter nicht als URL identifizieren kann (weil er nicht mit »`<Schema>`« anfängt oder so), dann wird »`mailto:`« davorgesetzt.

 Tun Sie sich einen Gefallen und verwenden Sie eine »unpersönliche« Rollenadresse à la

```
ServerAdmin webmaster@example.com
```

statt Ihrer eigenen Adresse. Zum einen macht es das einfach, Mail, die den Web-Server betrifft, von anderer Mail zu unterscheiden (die Leute, die Ihnen Mail schicken, sind da vielleicht nicht immer hundertprozentig eindeutig). Zum anderen müssen Sie dann nicht Ihre Apache-Konfiguration anpassen, wenn Sie mal in Urlaub sind und Ihre Kollegen Sie vertreten muss.

ServerSignature

Die Direktive `ServerSignature` steuert, ob der Apache-Server an automatisch generierte Seiten mit Fehlermeldungen eine Zeile mit seiner Adresse und anderen Informationen anhängt; es gibt die Werte »`0n`«, »`0ff`« und »`EMail`«, wobei letzteres die mit `ServerAdmin` angegebene URL (oder Adresse mit `mailto:`) mit in die Meldung aufnimmt.

ServerTokens

Mit `ServerTokens` können Sie beeinflussen, welche Informationen der Server über sich selbst in die HTTP-Kopfzeile `Server` schreibt, die als Teil der HTTP-Antwort verschickt wird. Der Standardwert »`Full`« umfaßt den Servernamen (»Apache«), die Versionsnummer, das aktuelle Betriebssystem sowie Informationen aus Modulen, während »`0S`« nur die Serverinformation und das Betriebssystem aufführt und »`Min`« (oder »`Minimal`«) sich auf den Servernamen und die Versionsnummer beschränkt. »`Minor`« gibt nur die ersten beiden Komponenten der Versionsnummer aus, »`Major`« nur die erste, und »`ProductOnly`« (oder »`Prod`«) liefert sogar nur den Servernamen ganz ohne Versionsnummer.

 Sie sollten sich von »`ServerTokens ProductOnly`« nicht zuviel versprechen. Das Unterdrücken der Versionsnummer erhöht die Sicherheit Ihres Apache-Servers nicht wirklich; Angreifer probieren einfach automatisch alle bekannten Angriffe aus. Das Einzige, was Sie erreichen, indem Sie die Versionsnummer Ihres Apache ausblenden, ist, dass Interoperabilitätsprobleme schwieriger zu diagnostizieren sind.

Ebenfalls wichtig für den Haupt-Server ist die Angabe eines Benutzernamens für den eigentlichen Server-Betrieb. Starten muss der Apache-Server als »root«, um privilegierte TCP-Ports wie den Port 80 öffnen zu können. Allerdings bemüht er sich, die root-Rechte baldmöglichst abzulegen und die mit den Direktiven `User` und `Group` angegebene Identität anzunehmen. Auch die Kindprozesse, die die eigentliche Arbeit erledigen, laufen unter dieser Identität. Beide Direktiven erlauben die Angabe eines Benutzer- bzw. Gruppennamens oder die einer numerischen Benutzer- oder Gruppen-ID in der Form »#123«.

Alle Ressourcen (Dateien, Verzeichnisse, Programme), die der Apache-Server für die Bearbeitung von Anfragen braucht, müssen für den genannten Benutzer bzw. die genannte Gruppe lesbar sein. Andere Systemressourcen, insbesondere wichtige Dateien wie `/etc/passwd`, sollten für den Server nicht lesbar und ganz bestimmt nicht beschreibbar sein.

 Leider ist es unter Linux zur Zeit schwierig bis unmöglich, gezielt einzelne Benutzer vom Zugriff auf Dateien *auszuschließen*. Sie würden eher versuchen, den Zugriff auf Dateien außerhalb der eigentlichen Web-Präsenz auf einem Server über Apache-Zugriffskontrolle zu unterbinden, da es nicht möglich ist, alle Dateien auf dem System so zu konfigurieren, daß der Apache-Benutzer *nicht* auf sie zugreifen kann.

 Wie der Apache-Benutzer genau heißt, hängt von Ihrer Distribution ab.

Üblicherweise bildet der Apache-Server einen Teil der lokalen Dateihierarchie im World Wide Web ab. Ein URL wie `http://www.example.com/test/bla.html` bezieht sich also auf eine Datei namens `test/bla.html` in einem geeigneten Verzeichnis auf dem Server. Dieses Verzeichnis wird über die Direktive `DocumentRoot` benannt. Das angegebene Verzeichnis sollte nicht mit einem Schrägstrich enden.

 Debian GNU/Linux und Ubuntu verwenden das Verzeichnis `/var/www`, um Dateien zu speichern, die mit dem World Wide Web zu tun haben – nicht nur statische HTML-Seiten, sondern auch Hilfsdateien von untergeordneten Softwarepaketen wie Wikis oder Suchmaschinen werden hier abgelegt.

 Seit SUSE-Version 8.1 verwenden die SUSE-Distributionen das Verzeichnis `/srv/www/htdocs` für die statischen HTML-Seiten des Systems. (Der frühere Wert, `/usr/local/httpd/htdocs`, war nicht wirklich FHS-konform.)

 Fedora verwendet dasselbe Verzeichnis wie Debian GNU/Linux und Ubuntu, nämlich `/var/www`.

 Bei einem selbstübersetzten und nicht besonders konfigurierten Apache heißt das Verzeichnis `/usr/local/apache/htdocs`.

 Wenn der angegebene Name kein absoluter Pfadname ist, wird er an den Wert von `ServerRoot` angehängt.

Übungen

 **3.4** [!1] Welche Dateien liefert Ihr Linux-Distributor im `DocumentRoot`-Verzeichnis aus?

 **3.5** [!2] Welchem Benutzer gehören diese Dateien? Warum? Entziehen Sie dem Apache-Benutzer (`wwwrun`, `www-data` oder so ähnlich) das Leserecht auf eine der Dateien und schauen Sie, was passiert, wenn Sie diese Seite abzurufen versuchen.

 **3.6** [2] Konfigurieren Sie Ihren Apache-Server so, dass er bei Fehlern eine Seite ausgibt, die seine Adresse und Ihre E-Mail-Adresse enthält. Testen Sie diese Konfiguration.

3.5 Konfiguration für bestimmte Ressourcen

Oft möchten Sie bestimmte Bereiche einer Web-Präsenz anders konfigurieren als andere. Zum Beispiel könnte ein Teil der Seiten nur für Rechner im lokalen Netz oder nur für bestimmte Benutzer zugänglich sein. Oder die privaten Seiten von Benutzern sollen anders behandelt werden als die »offiziellen« Seiten. Dieser Abschnitt erklärt, welche Möglichkeiten der Apache-Server bietet, um Konfigurationsdirektiven auf bestimmte Ressourcen oder Gruppen von Ressourcen einzuschränken.

Blockbegrenzungs-Direktiven

Grundsätzlich werden dafür sogenannte **Blockbegrenzungs-Direktiven** verwendet, die in ihrem Aussehen vage an HTML oder SGML erinnern. Zum Beispiel:

```
<Directory /var/www/restricted/>
  Options Indexes Includes FollowSymlinks
  AllowOverride None
  Order allow,deny
  Allow from all
</Directory>
```

Dieser Block von Direktiven erstreckt sich vom anfänglichen `<Directory ...>` bis zum nächsten `</Directory>` (Verschachtelungen von `<Directory>`-Blöcken sind nicht erlaubt.) Die in diesem `<Directory ...>`-Block angegebenen Direktiven gelten für das Verzeichnis `/var/www/restricted` und seine Unterverzeichnisse – wobei es möglich ist, dass weitere Blöcke noch andere Restriktionen für die betreffenden Ressourcen angeben oder die hier gezeigten wieder lockern.

Ein `<Directory>`-Block bezieht sich immer auf die Dateien in einem Verzeichnis sowie alle Unterverzeichnisse des Verzeichnisses mit ihren Dateien. Alternativ dazu gibt es den `<File>`-Block, der sich auf Dateinamen statt Verzeichnisnamen bezieht; mit `<File>`-Blöcken sind Konfigurationen für bestimmte Dateien möglich, egal wo sie in der Dateihierarchie zu finden sind. Schließlich gibt es noch `<Location>`-Blöcke, die sich nicht auf das Dateisystem des Servers beziehen, sondern auf die angefragten URLs.

Muster reguläre Ausdrücke

`<Directory>`-, `<File>`- und `<Location>`-Direktiven akzeptieren als Argument Muster, so wie die gängigen Shells sie für die Dateinamenexpansion verwenden. Ein `<Directory /home/*/*public_html>` könnte sich zum Beispiel auf die Verzeichnisse beziehen, in denen Benutzer ihre privaten Web-Seiten ablegen können. Alternativ dazu können auch reguläre Ausdrücke à la grep, awk & Co. verwendet werden, wenn zwischen Blockbegrenzungs-Direktive und Argument (regulärem Ausdruck) eine Tilde (`~`) steht. Die beiden folgenden Direktiven sind äquivalent:

```
<Directory /var/www/[a-c]*>
<Directory ~ /var/www/[a-c].*"$>
```

Zu jeder der drei Blockbegrenzungs-Direktiven gibt es auch eine weitere, die nur reguläre Ausdrücke als Argument akzeptiert: `<DirectoryMatch>`, `<FileMatch>` und `<LocationMatch>`:

```
<DirectoryMatch /var/www/[a-c].*"$>
```

ist äquivalent zu den beiden oben gezeigten Direktiven.

.htaccess-Dateien

Schließlich ist es noch möglich, die in Blöcken gemachten Einschränkungen durch Dateien in den jeweiligen Verzeichnissen zu erweitern. Diese Dateien heißen normalerweise `.htaccess` und können ebenfalls Apache-Direktiven enthalten.

Die Direktiven in Blöcken und verzeichnisspezifischen Dateien werden in der Reihenfolge folgenden Reihenfolge ausgewertet:

1. `<Directory>`-Blöcke ohne reguläre Ausdrücke und verzeichnisspezifische Dateien. Hierbei wird mit dem kürzesten Pfadnamen begonnen und mit dem

längsten aufgehört. Verzeichnisspezifische Dateien überschreiben Einstellungen aus Blöcken.

2. <Directory>-Blöcke mit regulären Ausdrücken und <DirectoryMatch>-Blöcke
3. <Files>- und <FilesMatch>-Blöcke
4. <Location>- und <LocationMatch>-Blöcke

Aufgrund dieser Eigenschaft ist es üblich, das Wurzelverzeichnis des Servers mit einer sehr restriktiven Konfiguration zu versehen. Diese Konfiguration »vererbt« sich dann an alle Dateien im System. Dabei ist es unerheblich, dass das Wurzelverzeichnis (zusammen mit dem größten Teil des Rests des Systems) überhaupt nicht auf dem WWW sichtbar ist:

```
<Directory />
  Options SymLinksIfOwnerMatch
  AllowOverride None
</Directory>
```

In diesem Beispiel sehen Sie gleich zwei der wichtigsten Direktiven für <Directory>-Blöcke: Options und AllowOverride.

Mit der Options-Direktive können Sie verschiedene Optionen des Servers für Options bestimmte Verzeichnisse ein- oder ausschalten. Im einzelnen sind das:

ExecCGI In diesem Verzeichnis ist die Ausführung von Dateien als CGI-Skripten (siehe 4.7) erlaubt.

FollowSymLinks Der Server folgt symbolischen Links aus diesem Verzeichnis in andere. Dabei wird aber der Pfadname, der zum Vergleich mit weiteren noch ausstehenden <Directory>-Blöcken verwendet wird, nicht geändert. *Wichtig:* In <Location>-Blöcken wird diese Direktive ignoriert.

Includes *Server-side includes* (siehe 4.7) sind erlaubt.

IncludesNOEXEC *Server-side includes* sind erlaubt, aber das Kommando #exec und die Ausführung von CGI-Skripten sind verboten. (Es ist trotzdem möglich, CGI-Skripten über #include auszuführen.)

Indexes Wenn ein URL angegeben wird, der auf ein Verzeichnis verweist, und dieses Verzeichnis keine DirectoryIndex-Datei (etwa index.html) enthält, wird eine formatierte Aufstellung des Verzeichnisinhalts generiert und an den Client zurückgegeben.

MultiViews MultiViews (siehe Abschnitt 4.4)(eine Technik zur Auswahl von Ressourcenformaten in Abhängigkeit von den Fähigkeiten des Clients) sind erlaubt.

SymLinksIfOwnerMatch Der Server folgt symbolischen Links, aber nur, wenn die Zielfile oder das Zielverzeichnis demselben Benutzer gehören wie das Link. *Wichtig:* In <Location>-Blöcken wird diese Direktive ignoriert.

All Alles Vorstehende bis auf MultiViews.

None Nichts von alledem.

Ein »+« vor der Option (oder auch gar nichts) schaltet sie ein, ein »-« schaltet sie aus.

Normalerweise gilt ausschließlich die Options-Direktive in dem Verzeichnis, das am besten auf die betrachtete Ressource passt. Steht allerdings vor jeder Option in der Liste entweder ein »+« oder ein »-«, dann werden die Optionen zusammengeworfen: Es werden die gerade gültigen Optionen betrachtet und die »+«-Optionen in der Liste eingeschaltet und die »-«-Optionen ausgeschaltet. Zum Beispiel:

```
<Directory /docs>
  Options Indexes FollowSymLinks
</Directory>
<Directory /docs subdir>
  Options Includes
</Directory>
```

erlaubt im Verzeichnis `/docs/subdir` nur `Includes`. Wenn allerdings die zweite Options-Direktive »+« und »-« enthält,

```
<Directory /docs>
  Options Indexes FollowSymLinks
</Directory>
<Directory /docs subdir>
  Options +Includes -Indexes
</Directory>
```

dann gelten in `/docs/subdir` die Optionen `Includes` und `FollowSymLinks`. (Noch eine Ausnahme: Beide Optionen »-Includes« und »-IncludesNOEXEC« schalten die *server-side includes* komplett ab.)

Der Standardwert, wenn keine Optionen namentlich angegeben wurden, ist `FollowSymLinks`.

AllowOverride Die Direktive `AllowOverride` bestimmt, welche Arten von Direktiven in verzeichnisspezifischen `.htaccess`-Dateien auftauchen können. Die möglichen Direktiven sind dafür in Gruppen zusammengefasst:

AuthConfig Direktiven, die mit Authentisierung und Autorisierung zu tun haben (siehe Kapitel 5): `AuthGroupFile`, `AuthUserFile`, `AuthName`, `AuthType`, ...

FileInfo Direktiven, die mit MIME-Dokumenttypen und -sprachen zu tun haben: `AddEncoding`, `AddLanguage`, `DefaultType`, `ErrorDocument`, Ferner Direktiven, die mit dem Finden von Ressourcen zu tun haben, etwa `Alias` oder `Redirect`, `Action` oder `RewriteEngine`, sowie Direktiven, die sich mit Metadaten von Dokumenten befassen, etwa `SetEnvIf`.

Indexes Direktiven, die mit der automatischen Erzeugung von Dateilisten zu tun haben: `AddDescription`, `AddIcon`, `FancyIndexing`, ...

Limit Direktiven, die mit Zugriffskontrolle zu tun haben: `Allow`, `Deny` und `Order`.

Options Direktiven, die mit Optionen zu tun haben: `Options` und `XBitHack`.



In der Form

```
AllowOverride Options=Indexes,Multiviews
```

können Sie explizit aufzählen, welche Optionen in einer `.htaccess`-Datei mit `Options` gesetzt werden dürfen. Beachten Sie dabei aber, dass etwas wie

```
Options Indexes
```

in einer `.htaccess`-Datei alle anderen möglicherweise in übergeordneten Verzeichnissen gesetzten Optionen neutralisiert. Unter dem Strich heißt das, dass Sie, sobald Sie mit `Options=...` das Setzen zusätzlicher Optionen erlauben, nicht erzwingen können, dass irgendeine spezielle Option gesetzt *bleibt*.

Nonfatal (Seit Apache 2.4.) Mit »`Nonfatal=Override`« können Sie dafür sorgen, dass Apache über die Verwendung per `AllowOverride` verbotener Direktiven hinwegsieht. »`Nonfatal=Unknown`« führt dazu, dass undefinierte Direktiven als

harmlos angesehen haben (dies umfasst außer Tippfehlern auch Direktiven von nicht tatsächlich eingebundenen Modulen). »`Nonfatal=All`« betrachtet beide diese Fehlerkategorien als unproblematisch.



Sie sollten dies mit Vorsicht genießen. Bei etwas wie

`AllowOverride Nonfatal=Override`

werden zum Beispiel Direktiven in `.htaccess`-Dateien, mit denen Zugriffsbeschränkungen auf Inhalte in Kraft gesetzt werden sollen, ohne Fehlermeldung ignoriert. Das kann heißen, dass Benutzer zwar denken, sie hätten ihre Inhalte geschützt, aber das nicht wirklich der Fall ist.

All Alle Direktiven, die überhaupt in `.htaccess`-Dateien vorkommen können.

None Nichts von alledem.

Unser restrikтивes Beispiel auf Seite 33 schaltet die Bearbeitung von `.htaccess`-Dateien also komplett ab (was nicht heißt, dass spezifischere `<Directory>`-Blöcke sie nicht für manche Unterverzeichnisse wieder einschalten könnten).



`AllowOverride` ist nur in `<Directory>`-Blöcken erlaubt, die keine regulären Ausdrücke verwenden, also insbesondere auch nicht in `<DirectoryMatch>`-, `<File>`- oder `<Location>`-Blöcken.

! Bis einschließlich Apache 2.3.8 (also Apache 2.2) war die Voreinstellung `All`; seit Apache 2.3.9 (also Apache 2.4) ist die Voreinstellung `None`.

Es ist eine weitverbreitete Konvention, die verzeichnisspezifischen Konfigurationsdateien `.htaccess` zu nennen, aber streng vorgeschrieben ist es nicht. Der Name dieser Dateien kann mit der Direktive `AccessFileName` festgelegt werden. Sie können `AccessFileName` dort auch mehrere Dateinamen aufzählen, die dann alle gesucht werden.



Normalerweise wird über einen `<Files>`-Block dafür gesorgt, dass niemand die `.htaccess`-Dateien über das Web abrufen kann und so Dinge herausfindet, die ihn eigentlich nichts angehen. Wenn Sie den Namen der `.htaccess`-Datei ändern, sollten Sie auch dafür Sorge tragen, den Zugriff auf die neuen Dateinamen in analoger Form zu unterbinden. Siehe hierzu auch Kapitel 5.

Übungen



3.7 [!3] Testen Sie, ob die Verzeichnisoption `FollowSymLinks` funktioniert: Legen Sie aus dem `DocumentRoot`-Verzeichnis ein symbolisches Link `symlinktest` auf eine geeignete Datei (beispielsweise `/etc/motd`). Schalten Sie für das `DocumentRoot`-Verzeichnis die Option `FollowSymLinks` ein (oder vergewissern Sie sich, dass sie schon eingeschaltet ist) und prüfen Sie, was passiert, wenn Sie auf `http://localhost/symlinktest` zugreifen.



3.8 [2] Wiederholen Sie das Experiment mit der Verzeichnisoption `SymLinksIfOwnerMatch`.

Zusammenfassung

- Die Konfigurationsdatei des Apache-Servers, `httpd.conf`, gliedert sich grob in vier Abschnitte: globale Einstellungen für das Programm, dynamisch ladbare Module, Konfiguration des »Haupt-Servers«, Konfiguration von virtuellen Servern.
- Datei- und Verzeichnisnamen in `httpd.conf` werden, sofern sie nicht absolut sind, an den Wert von `ServerRoot` angehängt.
- Die globale Konfiguration enthält Angaben über verschiedene wichtige Verzeichnisse und Dateien. Ferner gibt es diverse Parameter zur Leistungsfeinabstimmung.
- Viele Aspekte des Apache-Servers sind in ladbare Module ausgegliedert, deren Aktivierung Teil der Konfiguration ist.
- Zur Konfiguration des Hauptservers gehören unter anderem Angaben über die zu verwendenden Identitäten, Adressen für Fehlermeldungen, Netzwerkschnittstellen und -ports und den Ort der zu veröffentlichten Ressourcen.
- `<Directory>`-, `<Files>`- und `<Location>`-Blöcke erlauben es, bestimmte Direktiven nur für Teile der Ressourcenhierarchie in Kraft zu setzen.
- Teile dieser Konfiguration können in verzeichnisspezifische Dateien (typisch `.htaccess`) ausgegliedert werden.

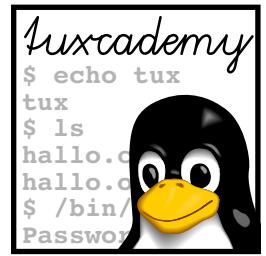
Literaturverzeichnis

For99 Andrew Ford. »Apache Quick Reference Card«, Mai 1999. Verschiedene Formate (PDF, PostScript, DIN A4, US-Letter). Beschreibt Apache 1.3.6.

<http://www.refcards.com/about/apache.html>

For00 Andrew Ford. *Apache Pocket Reference*. Sebastopol, CA: O'Reilly & Associates, 2000. ISBN 1-56592-706-0. Inzwischen nicht mehr ganz aktuell, aber trotzdem sehr nützlich. Deutsch als *Apache – kurz & gut*.

<http://www.oreilly.de/catalog/apachepr>



4

Organisation einer Web-Präsenz

Inhalt

4.1	Verzeichnisstrukturen	38
4.2	Automatisch erzeugte Verzeichnislisten	38
4.3	Benutzereigene Seiten	43
4.4	Inhaltsaushandlung	46
4.5	Die Alias-Direktive	50
4.6	Weiterleitung	51
4.7	Dynamische Inhalte	52
4.8	Fehlermeldungen	57

Lernziele

- Eine Verzeichnisstruktur für eine Web-Präsenz entwerfen können
- Automatische Verzeichnislisten konfigurieren und einsetzen können
- Benutzerspezifische Verzeichnisse freigeben können
- Andere Verzeichnisbäume in eine Web-Präsenz einbinden können
- Ressourcen für verschiedene Sprachen und Datentypen anbieten können
- Zugriffe an andere Seiten oder Server weiterleiten können
- Fehlermeldungen anpassen können

Vorkenntnisse

- Überblick über Apache-Konfiguration (Kapitel 3)
- Unix-Dateisystem
- Nützlich: HTML-Kenntnisse

4.1 Verzeichnisstrukturen

Warum Verzeichnisse?	Die wenigsten Web-Präsenzen kommen mit einem Verzeichnis für alle Seiten aus. In der Regel werden Sie Verzeichnisse einrichten, um <ul style="list-style-type: none"> • die Präsenz überschaubarer und leichter wartbar zu machen • die Struktur der Organisation in der Präsenz abzubilden (etwa: unterschiedliche Ressourcenhierarchien für verschiedene Abteilungen, aber mit gemeinsam genutzten Elementen) • verschiedenen Personen oder Gruppen von Personen unterschiedliche Zugriffsrechte auf verschiedene Teile der Präsenz zu geben, sowohl was das Abrufen von Seiten angeht als auch das Einstellen neuer Inhalte
Verzeichnisse vs. URLs	Verzeichnisse des unterliegenden Dateisystems erscheinen als Pfadelemente in den URLs der Web-Präsenz; ein URL wie <div style="border: 1px solid black; padding: 2px; display: inline-block;"> http://www.example.com/training/anwendungen/apache.html </div> verweist auf die Datei <code>apache.html</code> im Verzeichnis <code>training/anwendungen</code> in dem durch die Konfigurationsdirektive <code>DocumentRoot</code> gegebenen Verzeichnis des Servers <code>www.example.com</code> .
Früh überlegen!	Es empfiehlt sich, sich schon beim Aufbau einer Web-Präsenz Gedanken über eine geeignete Verzeichnisstruktur zu machen. Wenn die URLs beliebter Seiten erst einmal bekanntgemacht oder von Suchmaschinen erfasst wurden, ist es sehr schwierig, die Verzeichnisstruktur noch durchgreifend zu ändern – Sie sind fast gezwungen, die »alte« Struktur über symbolische Links, HTTP-Umleitung oder URL-Umschreiben zu erhalten, wenn Sie die »Kunden« nicht durch ständige Fehlermeldungen verschrecken möchten.
relative Verweise	Verzeichnisse machen es auch möglich, über relative Verweise auf Seitenelemente HTML-Seiten einerseits ein gemeinsames Aussehen zu geben (Stichwort » <i>corporate identity</i> «), etwa dadurch, dass Sie eine allgemeine Vorlage zur Verfügung stellen, und andererseits dasselbe Dokument an verschiedenen Stellen der Präsenz dem jeweiligen Umfeld anzupassen. Ein relativer Verweis auf eine graphische Kopfzeile wie <code></code> kann in verschiedenen Verzeichnissen, die jeweils eine andere Datei <code>titel.gif</code> enthalten, die entsprechend richtige Graphik einbinden.
Sammeln gemeinsamer Elemente	 An dieser Stelle sollten wir erwähnen, dass die Möglichkeiten der automatischen Anpassung durch verschiedene Verzeichnisse doch eher gering sind. Zum Beispiel ist es auf der reinen Ebene von HTML-Dateien nicht möglich, variable Navigationselemente wie Verweise auf die anderen Ressourcen im selben Verzeichnis einzubauen. Das geht nur über »Framesets«, die Teile einer sichtbaren Seite aus anderen, verzeichnisspezifischen HTML-Dateien beziehen, über <i>server-side includes</i> oder über andere Systeme, die die tatsächlichen Antwortseiten aus diversen Teilen dynamisch zusammensetzen.
	Eine weitere sinnvolle Vorgehensweise ist es, beispielsweise Graphikelemente, die auf vielen Seiten in diversen Verzeichnissen auftauchen, in einem speziellen Verzeichnis zu sammeln. Wenn Sie in Ihren Seiten zum Beispiel kleine Bilder von Pfeilen, die in verschiedene Richtungen zeigen, als Navigationselemente einsetzen, dann könnten Sie ein Verzeichnis <code>nav</code> direkt unter dem <code>DocumentRoot</code> -Verzeichnis anlegen und diese Bilder dort platzieren. In Ihren Seiten können Sie sich dann mit relativen URLs der Form <code>/nav/links.gif</code> auf die Bilder beziehen, egal wo in Ihrer Ressourcenhierarchie die bezugnehmenden Seiten tatsächlich stehen.

4.2 Automatisch erzeugte Verzeichnislisten

Wird dem Apache-Server ein URL übergeben, der auf ein Verzeichnis (anstatt einer Datei) verweist, dann prüft er zunächst, ob in dem betreffenden Verzeichnis

eine der Ressourcen existiert, die in einer für dieses Verzeichnis passenden `DirectoryIndex`-Direktive genannt wurden. Typischerweise handelt es sich dabei um relative Dateinamen wie `index.html`, `index.htm` (für Seiten, die von Windows-Rechnern kommen) und ähnliche, wie das folgende Beispiel aus der Apache-Standardkonfiguration zeigt:

```
DirectoryIndex index.html index.htm index.shtml index.cgi
```

Es spricht aber auch nichts dagegen, auf Ressourcen zu verweisen, die *nicht* im betreffenden Verzeichnis liegen und auch nicht als Datei zurückgegeben werden: Beliebige lokale URLs sind erlaubt. Mit

```
DirectoryIndex index.html /cgi-bin/index.pl
```

würde, falls `index.html` nicht existiert, das CGI-Skript `/cgi-bin/index.pl` ausgeführt. (Diese Lösung ist übrigens dem obigen Standardbeispiel weit überlegen, da dort `index.cgi` bedingt, dass Dateien im *aktuellen* Verzeichnis als CGI-Skripte ausgeführt werden können – vom Standpunkt der Systemsicherheit keine gute Idee! Siehe hierzu auch Kapitel 8.) Voraussetzung für das Ganze ist das Modul `mod_dir`.

 Mit

```
DirectoryIndex disabled
```

können Sie die Suchliste komplett leeren.

 Mehrere `DirectoryIndex`-Direktiven im selben Kontext addieren sich auf. Etwa wie

```
<Directory /mysite>
  DirectoryIndex index.html
  DirectoryIndex index.php
</Directory>
```

ist also dasselbe wie

```
<Directory /mysite>
  DirectoryIndex index.html index.php
</Directory>
```

 Rein prinzipiell funktioniert `DirectoryIndex` nur für URLs, die mit einem Schrägstrich aufhören. Wenn Benutzer eine Ressource aufrufen, die auf ein Verzeichnis verweist, aber keinen Schrägstrich am Ende angegeben haben, dann generiert Apache in der Regel eine Umleitung auf den korrekten Namen der Ressource. Das ist grundsätzlich eine gute Idee, weil diese Vorgehensweise unter anderem dazu führt, dass relative URL-Verweise in HTML-Seiten funktionieren. Mit der Direktive `DirectorySlash` können Sie `DirectorySlash` dieses Verhalten steuern: Der Wert »`Off`« unterbindet die Umleitung.



»`DirectorySlash Off`« ist nicht ganz ungefährlich. Betrachten Sie etwas wie

```
<Directory /mydir>
  Options Indexes
  DirectoryIndex index.html
  DirectorySlash Off
</Directory>
```

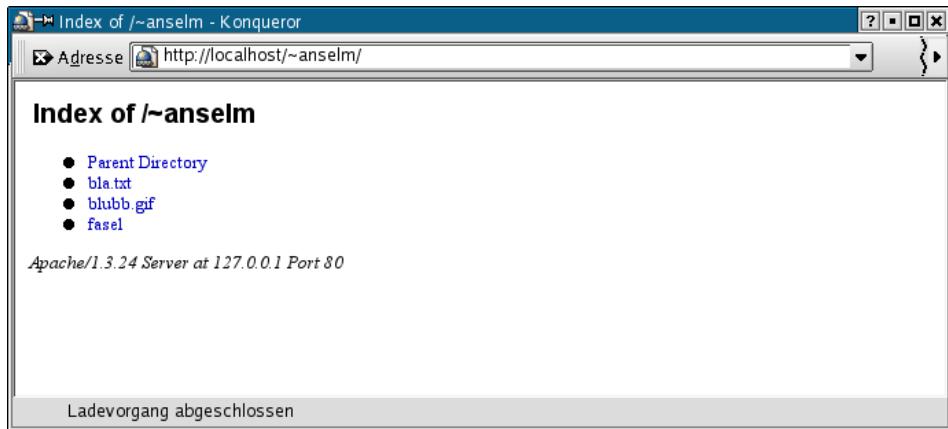


Bild 4.1: »Schlichte« Verzeichnisliste (ohne FancyIndexing)

Eine Anfrage nach `http://www.example.com/mydir/subdir/` sorgt dann dafür, dass nach `http://www.example.com/mydir/subdir/index.html` gesucht wird. Eine Anfrage nach `http://www.example.com/mydir/subdir` (ohne Schrägstrich) dagegen würde den Verzeichnisinhalt auflisten – was in der Regel nicht gewollt ist, wenn es eine `index.html`-Datei gibt.

automatische Verzeichnisliste

Existiert keine der in der `DirectoryIndex`-Direktive genannten Ressourcen und ist die Option `Indexes` für das Verzeichnis eingeschaltet, dann erzeugt Apache eine automatische Verzeichnisliste (Bild 4.1). Zuständig hierfür ist das Modul `mod_autoindex`, das Sie auch leicht ganz weglassen können – in den meisten Fällen sollten Sie auf die automatischen Verzeichnislisten nämlich verzichten, da normalerweise nicht jede Seite in einem Verzeichnis auf dem Web-Server tatsächlich für die Öffentlichkeit gedacht ist (es könnten Sicherheitskopien alter Seiten oder noch nicht offiziell veröffentlichte, halbfertige neue Seiten dort herumliegen, die über eine automatische Liste der Welt zugänglich gemacht würden). Es ist viel besser, auf explizit erzeugte `index.html`-Dateien zu bestehen. – Nützlich sind die automatischen Verzeichnislisten dagegen zum Beispiel, wenn Sie über HTTP Zugang auf den Datenbestand eines anonymen FTP-Servers ermöglichen wollen.

Wie nicht anders zu erwarten gibt es für die automatischen Verzeichnislisten umfangreiche Konfigurationsmöglichkeiten. Zentral ist die `IndexOptions`-Direktive, mit der sich eine ganze Menge Optionen für die Listenerzeugung ein- und ausschalten lassen. Zum Beispiel:

NameWidth=[* | <n>] Erlaubt die Angabe der Breite der Spalte für Dateinamen. Der Wert »*« macht die Spalte gerade so breit, dass der längste Dateiname passt. Andere Werte geben die Breite in Zeichen an.

IconWidth=[<x>] und **IconHeight=[<y>]** Geben die Breite und Höhe der Icons für die Dateien an und liefern diese auch im HTML-``-Tag aus. Dies erlaubt es einem Browser, das Seitenlayout zu berechnen, bevor alle Icons geladen wurden, und beschleunigt damit die Anzeige. Die Standardwerte sind die Breite und die Höhe der mit Apache ausgelieferten Standard-Icons.

FancyIndexing Sorgt für die Erzeugung »vornehmer« Verzeichnislisten, die eine deutlich umfassendere Konfiguration zulassen. Es gibt auch eine eigene `FancyIndexing`-Direktive, die aber nicht mehr benutzt werden sollte.

Die Option `FancyIndexing` macht es möglich, eine ganze Reihe anderer Optionen zu verwenden, zum Beispiel

FoldersFirst Verzeichnisse stehen immer zuerst in der Liste

IconsAreLinks Die Icons werden Teil der anklickbaren Dateinamen

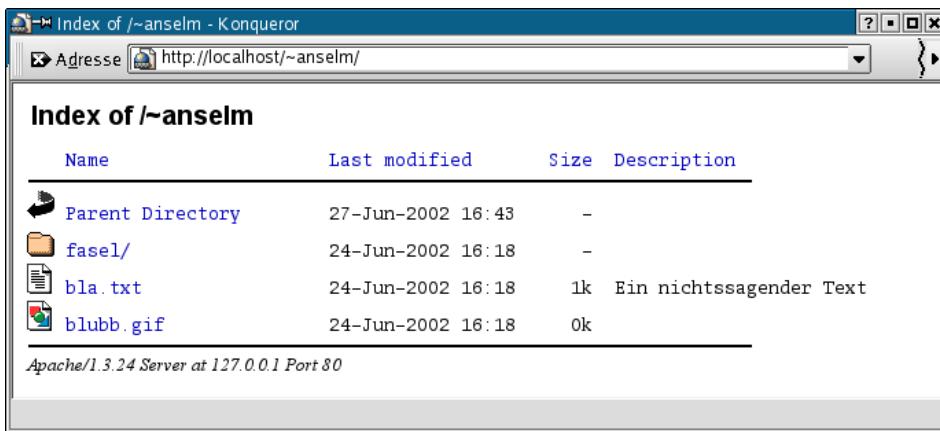


Bild 4.2: »Aufwändige« Verzeichnisliste (mit Indexoptionen FancyIndexing und FoldersFirst)

IgnoreCase Die Dateinamen werden ohne Rücksicht auf Groß- und Kleinschreibung sortiert

SuppressColumnSorting Normalerweise können die Spaltentitel in einer FancyIndexing-Liste angeklickt werden, um die Sortierung gemäß der betreffenden Spalte zu ändern. Diese Option unterdrückt das.

SuppressDescription, SuppressLastModified und SuppressSize Diese Optionen unterdrücken verschiedene Bestandteile einer Verzeichnisliste

Bild 4.2 zeigt eine Verzeichnisliste, für die die Optionen FancyIndexing und FoldersFirst eingeschaltet sind.

Für IndexOptions gilt eine ähnliche Vorgehensweise wie für die Options-Direktive. Wie bei Options können Sie Optionsnamen einfach erwähnen, so wie sie sind, dann wird die betreffende Option eingeschaltet und am Ende gelten alle in der betreffenden IndexOptions-Direktive eingeschalteten Optionen und keine anderen. Alternativ dazu können die Optionen auch aus vorigen Direktiven (die eventuell in der Konfiguration übergeordneter Verzeichnisse stehen) übernommen werden, und Sie können weitere Optionen *inkrementell* setzen oder löschen, indem Sie ein »+« oder »-« vor den Optionsnamen setzen. Ein Optionsname ohne »+« oder »-« löscht alle vorherigen inkrementell angegebenen Optionen; nach

```
IndexOptions +FoldersFirst -IconsAreLinks FancyIndexing
IndexOptions +SuppressSize
```

gilt unter dem Strich also

```
IndexOptions FancyIndexing +SuppressSize
```

 Mehrere IndexOptions-Direktiven für dasselbe Verzeichnis addieren sich auf:

```
<Directory /mydir>
  IndexOptions FancyIndexing
  IndexOptions FoldersFirst
</Directory>
```

ist also dasselbe wie

```
<Directory /mydir>
  IndexOptions FancyIndexing FoldersFirst
</Directory>
```

IndexOrderDefault Die IndexOrderDefault-Direktive bestimmt die anfängliche Sortierung der Dateien in der Liste (FancyIndexing muss aktiv sein). Sortieren können Sie nach dem Dateinamen (Name), dem Datum der letzten Änderung (Date), der Dateigröße (Size) und der Dateibeschreibung (Description), und das jeweils aufsteigend (Ascending) oder absteigend (Descending):

```
IndexOrderDefault Ascending Date
```

sortiert die Dateien aufsteigend nach dem Datum. Ist das Datum – oder allgemein gesagt das primäre Sorterkriterium – bei mehreren Dateien gleich, wird innerhalb dieser Gruppe immer aufsteigend nach dem Namen sortiert. Sie können die anfängliche Sortierung unänderbar festschreiben, indem Sie die Indexoption SuppressColumnSorting aktivieren.

Auswahl von Icons Es gibt diverse Möglichkeiten, die Verzeichnisliste zu verbrämen oder mit mehr Informationen auszustatten, vor allem, wenn FancyIndexing aktiv ist. Mit der Auswahl von Icons (Sinnbildern) für die verschiedenen Dateien beschäftigen sich die Direktiven AddIcon, AddIconByEncoding, AddIconByType und DefaultIcon.

AddIcon Mit AddIcon können Sie Icons für Dateien definieren, deren Namen bestimmte Endungen haben:

```
AddIcon (BILD,/icons/image2.gif) .gif .jpg .png
AddIcon /icons/dir.gif ^^DIRECTORY^^
AddIcon /icons/backup.gif *~
```

Das erste Argument von AddIcon ist entweder ein relativer URL für das Icon oder hat das Format (*alt-Text*,*Icon-URL*), wobei der *alt-Text* als Ersatztext von Browsern ausgegeben wird, die keine Bilder anzeigen können (oder sollen). Das zweite Argument ist ^^DIRECTORY^^ für Verzeichnisse, ^^BLANKICON^^ für leere Zeilen (damit die Formatierung stimmt), eine Dateinamens-Endung, ein Shell-Suchmuster oder ein Dateiname oder ein Stück davon.

AddIconByType Die AddIconByType-Direktive, die AddIcon, wenn möglich, vorzuziehen ist, ordnet bestimmten MIME-Typen Icons zu. Sie hat wie AddIcon zwei Argumente, wobei das zweite ein Suchmuster für MIME-Typen ist:

```
AddIconByType (BILD,/icons/image2.gif) image/*
```

AddIconByEncoding AddIconByEncoding ordnet Dateien Icons auf der Basis ihrer Codierung zu:

```
AddIconByEncoding /icons/compressed.gif x-compress
```

DefaultIcon Mit DefaultIcon können Sie ein Icon für die Dateien vergeben, für die kein anderweitiges Icon vergeben werden kann:

```
DefaultIcon /icons/unknown.xbm
```

(Ein Ersatztext kann hier nicht angegeben werden).

AddAlt In Analogie zu den drei AddIcon...-Direktiven gibt es auch noch AddAlt, AddAltByEncoding und AddAltByType. Seit Sie die Ersatztexte aber direkt bei AddIcon & Co. angeben können, sind diese Direktiven nicht mehr besonders wichtig. Allenfalls könnten Sie damit dafür sorgen, dass zum Beispiel Graphikdateien in verschiedenen Formaten alle denselben Ersatztext bekommen, ohne diesen in jeder AddIcon-Direktive angeben zu müssen.

AddDescription Schließlich ist es möglich, mit AddDescription beschreibende Texte zu verschiedenen Dateien oder Dateitypen anzugeben:

```
AddDescription "Die Pilzköpfe" /web/pics/beatles.gif
```

Auch hier ist das zweite Argument wieder eine Dateiendung, ein (ggf. partieller) Dateiname oder ein Shell-Suchmuster. Das erste Argument ist die Beschreibung, die bis zu 23 Zeichen lang sein kann (Die Index-Option SuppressSize vergrößert den Platz um 7, die Option SuppressLastModified um 19 Zeichen. Die Option DescriptionWidth erlaubt es, die Breite explizit einzustellen.) Was übersteht, wird abgeschnitten. Sie müssen ein bisschen aufpassen, da das erste Argument HTML-Auszeichnungen enthalten darf; bei etwas wie

```
AddDescription "<em>The Artist Formerly Known As Prince</em>" \
/web/pics/tafkap.gif
```

könnte es passieren, dass das schliessende »« abgeschnitten und der Rest des Verzeichnisses dann hervorgehoben ausgegeben wird (brr.)

Mit der Index-Option ScanHTMLTitles können Sie Apache dazu bringen, in Abwesenheit von expliziten AddDescription-Direktiven die Inhalte der <title>-Tags von HTML-Seiten anzuzeigen. Das ist allerdings sehr ressourcenintensiv.

Mit der Direktive IndexStyleSheet können Sie eine CSS-Datei angeben, auf die automatische Verzeichnislisten verweisen. Damit können Sie die Formatierung der Verzeichnislisten sehr detailliert steuern. Die Einzelheiten der verwendeten CSS-Klassen finden Sie in der Apache-Dokumentation.

Die Direktive ReadmeName gibt den Namen (relativ zum betrachteten Verzeichnis) einer Datei an, deren Inhalt an die Verzeichnisliste angehängt wird. Dies ist vor allem nützlich, wenn Sie den Inhalt eines anonymen FTP-Servers über HTTP zugänglich machen möchten:

```
ReadmeName README
```

Analog dazu können Sie mit HeaderName den Namen einer Datei angeben, deren Inhalt am Anfang der Verzeichnisliste eingefügt wird:

```
HeaderName HEADER.html
```

Mit der Direktive IndexIgnore können Sie Dateien oder Gruppen von Dateien aus der Liste ausschließen:

```
IndexIgnore README .htaccess *~
```

Auch hier können komplette oder partielle Dateinamen, Suchmuster oder Dateiendungen angegeben werden.

Übungen

 **4.1** [!2] Legen Sie unterhalb von DocumentRoot ein Verzeichnis idxtest an und kopieren Sie einige Dateien hinein. Schalten Sie für idxtest die Verzeichnisoption Indexes ein und sorgen Sie dafür, dass eine automatische Verzeichnisliste erzeugt wird.

 **4.2** [3] Konfigurieren Sie die Verzeichnisliste von idxtest so, dass das Datum der letzten Änderung unterdrückt wird, Verzeichnisse zuerst in der Liste stehen, die Dateien absteigend nach ihrer Größe sortiert werden und der Inhalt der Datei LIESMICH ans Ende der Liste gestellt wird. Dateien, deren Name mit bla anfängt, sollen nicht in die Liste aufgenommen werden.

4.3 Benutzereigene Seiten

Eine gängige Anwendung des WWW – jedenfalls in Universitäten oder Intranets – besteht darin, Benutzern zu erlauben, dass sie ihre eigenen Seiten veröffentlichen.

Sie könnten nun ein Verzeichnis einrichten, in dem jeder Benutzer sein eigenes Unterverzeichnis hat (ähnlich wie in `/home`) und dort Ressourcen ablegen kann, aber es ist bequemer, einfach ein spezielles Unterverzeichnis im eigenen Heimatverzeichnis jedes Benutzers dafür zu benutzen. Nach Konvention heißt dieses Unterverzeichnis `public_html`, und eine Datei mit einem Namen wie

```
/home/louis14/public_html/l-etat-c-est-moi.html
```

wäre auf dem Web über die URL

```
http://www.versailles.fr/~louis14/l-etat-c-est-moi.html
```

zu erreichen.

`UserDir` Konfiguriert wird das über die Direktive `UserDir`. `UserDir` kann verschiedene Argumente haben:

- Einen relativen Verzeichnisnamen. Dieser bezeichnet ein Verzeichnis im Heimatverzeichnis des Benutzers; der Teil des URL-Pfads hinter `~user` wird an das Verzeichnis angehängt.
- Einen absoluten Verzeichnisnamen. Dieser bezeichnet ein Verzeichnis irgendwo im Dateisystem. Enthält dieser Name einen Stern (`/*`), wird an dieser Stelle der Benutzername eingesetzt, sonst wird der Benutzername ans Ende des Namens angehängt. An das Resultat dieser Ersetzung wird dann der Rest des URL-Pfads angehängt.
- Einen URL. Dies führt dazu, dass Apache eine Umleitung an den betreffenden URL schickt. Zum Beispiel ergibt mit

```
# Konfiguration auf www.example.com
UserDir http://www-ext.example.com/
```

eine Anfrage nach `http://www.example.com/~hugo/blubb.html` eine Weiterleitung nach `http://www-ext.example.com/hugo/blubb.html` (achten Sie auf die Tilde).

- Eine beliebige Mischung aus den vorstehenden drei Möglichkeiten ist auch erlaubt. Einer Definition wie

```
UserDir public_html /srv/www/userdirs http://www.example.net/
```

steht grundsätzlich nichts im Weg.

 Apache hat keine Möglichkeit, herauszufinden, ob eine Umleitung geklappt hat oder nicht. Aus diesem Grund ist eine Umleitung immer das letzte Element in der Liste, das angeschaut wird – wenn dahinter noch andere Elemente kommen, werden diese ignoriert.

- Das Wort `disabled`, möglicherweise gefolgt von einer Liste von (durch Leerzeichen getrennten) Benutzernamen. Damit werden benutzerspezifische Verzeichnisse für alle Benutzer oder, falls Benutzernamen aufgezählt wurden, für die betreffenden Benutzer gesperrt.
- Das Wort `enabled`, möglicherweise gefolgt von einer Liste von (durch Leerzeichen getrennten) Benutzernamen. Dies schaltet benutzerspezifische Verzeichnisse für alle Benutzer (oder die aufgezählten) ein.

Hierfür muss das Modul `mod_userdir` aktiviert sein. Die Direktive hat keinen Standardwert; um die übliche Konvention in Kraft zu setzen, brauchen Sie also

```
UserDir public_html
```

Einige Beispiele: Eine Anfrage nach `http://www.example.com/~tux/a/b.html` wird wie folgt ausgewertet:

```
UserDir public_html => ~tux/public_html/a/b.html
UserDir /var/web    => /var/web/tux/a/b.html
UserDir /home/*/www => /home/tux/www/a/b.html
```

Um einigen Benutzern benutzerspezifische Verzeichnisse zu erlauben, aber sonst niemandem:

```
UserDir disabled
UserDir enabled tux emil hugo
```

Um allen Benutzern benutzerspezifische Verzeichnisse zu erlauben, aber einigen nicht:

```
UserDir enabled
UserDir disabled tux emil hugo
```

Es empfiehlt sich meist, für die benutzerspezifischen Verzeichnisse spezielle Restriktionen einzuführen, etwa so:

```
<Directory /home/*/public_html>
  AllowOverride FileInfo AuthConfig Limit
  Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNOEXEC
</Directory>
```

Dies würde es den Benutzern erlauben, in .htaccess-Dateien in ihrem Verzeichnis (und dessen Unterverzeichnissen) Direktiven zu verwenden, die die zurückgegebenen Dateitypen beeinflussen (FileInfo), Authentisierung auf Benutzerebene erlauben (AuthConfig) und Beschränkungen der gültigen HTTP-Methoden vorsehen (Limit). Die Verzeichnisse selbst unterstützen einfache Inhaltsaushandlung (MultiViews, siehe Abschnitt 4.4), automatische Verzeichnislisten (Indexes, siehe Abschnitt 4.2), symbolische Links, aber nur, wenn die Eigentümer von Link und Ziel übereinstimmen (SymLinksIfOwnerMatch) und *server-side includes*, aber ohne Programmausführung.

Generell ist es eine gute Idee, die Rechte von Benutzern in ihren benutzerspezifischen Verzeichnissen so weit wie möglich einzuschränken, um keine Probleme zu provozieren. So ist zum Beispiel IncludesNOEXEC dem allgemeineren Includes vorzuziehen, solange niemand die Programmausführung benötigt (und selbst dann ist es sinnvoller, sie gezielt für den betreffenden Benutzer freizuschalten). Wenn überhaupt niemand *server-side includes* verwendet, können Sie die Funktionalität auch ganz abschalten.

Rechte von Benutzern

Übungen

 **4.3 [2]** Konfigurieren Sie Ihren Apache-Server so, dass Benutzer eigene Web-Seiten im Verzeichnis `WWW` in ihrem Heimatverzeichnis anlegen können. Achten Sie darauf, dass Sie gegebenenfalls auch einen `<Directory>`-Block anpassen müssen. (SUSE: Betrachten Sie die Datei `suse_public_html.conf`.)

 **4.4 [3]** Nehmen Sie die Identität eines »normalen« Benutzers an und platzieren Sie eine einfache HTML-Seite (oder mehrere) in dessen `$HOME/WWW`-Verzeichnis. Greifen Sie auf diese Seite über einen URL der Form `http://localhost/~<Benutzer>/...` zu. (Einen neuen Benutzer können Sie gegebenenfalls mit »`useradd -m hugo`« anlegen.)

```

URI: bla

URI: bla.en.html
Content-Type: text/html
Content-Language: en

URI: bla.fr.de.html
Content-Type: text/html
Content-Language: fr, de

```

Bild 4.3: Eine einfache Typemap-Datei

4.4 Inhaltsaushandlung

Darstellungen Von einer Ressource kann es mehrere **Darstellungen** (engl. *representations*) geben, etwa in verschiedenen Sprachen oder Datenformaten. **Inhaltsaushandlung** (engl. *content negotiation*) ist ein Prozess, mit dem Browser und Server sich »handelseinig« darüber werden, welche Darstellung als Antwort auf die Frage nach einer Ressource geliefert werden soll.

Eine Darstellung einer Ressource unterscheidet sich von anderen durch Attribute wie den (MIME-)Datentyp, den verwendeten Zeichensatz, die Codierung (komprimiert oder nicht und, wenn ja, wie), die Sprache und so weiter. Jede Ressource kann keine, eine oder mehrere Darstellungen haben. Wenn mehrere Darstellungen existieren, bezeichnet man die Ressource als **verhandelbar** (engl. *negotiable*) und nennt jede der Darstellungen eine **Variante** (engl. *variant*). Die Arten, auf die die Varianten einer verhandelbaren Ressource sich unterscheiden, nennt man die **Dimensionen** der Verhandlung.

Ein Browser kann dem Server über geeignete HTTP-Kopfzeilen signalisieren, in welcher Dimension er welche Werte bevorzugt; Browser machen es dem Benutzer normalerweise möglich, Präferenzen z. B. für die Sprache zu signalisieren, während andere Präferenzen (etwa die unterstützten Graphikformate) meist durch die Fähigkeiten des Browserprogramms selbst vorgegeben sind. Die HTTP-Kopfzeile

Accept-Language: de

würde zum Beispiel bedeuten, dass der Browser mit Vorliebe deutschsprachige Repräsentationen der Ressource geschickt bekommen möchte (wenn es eine Auswahl gibt). Sie können das ganze auch feiner steuern:

Accept-Language: de; q=1.0, en; q=0.5

heißt, dass Deutsch und Englisch akzeptabel sind, aber Deutsch bevorzugt werden soll.

Apache erlaubt Inhaltsaushandlung über das Modul `mod_negotiation`. Dies umfasst insbesondere »servergetriebene Verhandlung« (engl. *server-driven negotiation*) gemäß HTTP/1.1 über die HTTP-Kopfzeilen »Accept«, »Accept-Language«, »Accept-Charset« und »Accept-Encoding«. Apache unterstützt auch »transparente« Inhaltsaushandlung nach RFC 2295 [RFC2295] und RFC 2296 [RFC2296], aber nicht die dort ebenfalls definierte *feature negotiation*. Bei servergetriebener Verhandlung trifft der Server die Entscheidung darüber, welche Variante er liefert; bei transparenter Verhandlung entscheidet der Browser (gegebenenfalls mit »Schützenhilfe« durch den Server).

Bevor eine Inhaltsaushandlung stattfinden kann, muss der Server wissen, welche Variante überhaupt zur Verfügung stehen. Dafür gibt es zwei Möglichkeiten: explizite **Typemap-Dateien** oder **Multiviews**.

Typemap-Dateien

Multiviews

URI: blubb

URI: blubb.jpg

Content-Type: image/jpeg; qs=0.8

URI: blubb.gif

Content-Type: image/gif; qs=0.5

URI: blubb.txt

Content-Type: text/plain; qs=0.01

Bild 4.4: Eine Typemap-Datei mit Qualitätsfaktoren

Eine Typemap-Datei führt für eine gegebene Ressource alle Varianten mit ihren Eigenschaften auf (Bild 4.3). Sie sollte denselben Namen haben wie die Ressource, aber die Endung .var (nach Konvention); vor der Verwendung von Typemap-Dateien muss mit etwas wie

```
AddHandler type-map var
```

dafür gesorgt werden, dass der Apache-Server diese Dateien korrekt verarbeiten kann. Statt auf etwas wie .../bla.html muss der Browser dann auf .../bla.var zugreifen. Die Inhaltsaushandlung sorgt dann dafür, dass der Server die »passendste« Version der Ressource an den Browser schickt.



Es ist natürlich nicht besonders elegant, in der Web-Präsenz nun alle interessanten (d. h. über Inhaltsaushandlung zugänglichen) Dateien über URLs mit der Endung .var ansprechen zu müssen, vor allem wenn die Benutzer etwas wie .html gewöhnt sind. Als Trick können Sie aber den Typemap-Dateien die Endung .html geben (mit einer geeigneten AddHandler-Direktive) – nur müssen Sie dann darauf achten, dass die »echten« HTML-Dateien nicht mehr auf .html enden können. Am besten verwenden Sie für diese dann die Endung .htm (das macht die wenigsten Probleme). Diese Methode ist vielleicht etwas umständlich, aber wirkungsvoll.

Es ist auch möglich, die relative »Qualität« einer Variante auszudrücken (Bild 4.4). Hierzu wird der qs-Parameter in der Content-Type-Zeile benutzt. Er kann Werte zwischen 0.000 und 1.000 annehmen, wobei der Wert 0.000 bedeutet, dass die betreffende Variante nie ausgeliefert wird. Die Qualitätsangabe bezieht sich immer auf die Variante im Vergleich zu allen anderen Varianten; für ein Foto hätte die JPEG-Datei vermutlich die höchste Qualität (höher als GIF und ASCII-Graphik, auf jeden Fall), aber wenn es sich bei der Ressource um eine ursprüngliche ASCII-Graphik handelt, wären GIF- und JPEG-Varianten sicherlich von niedrigerer »Qualität«. Was der Browser kann oder nicht kann, ist für die serverseitigen Typemaps ohne Belang. Varianten ohne Qualitätsangabe haben implizit die Qualität 1.000.

Für jede Variante werden die folgenden Felder unterstützt:

URI Relative (zum Verzeichnis der Typemap-Datei) URI der Datei, in der die Variante steht. Der Client muss auf die Datei direkt zugreifen dürfen (selbst wenn es wegen der Verhandlung nicht nötig ist).

Content-Type Der (MIME-)Datentyp der Variante. Einige Parameter sind erlaubt, etwa charset und level (»Sprachebene«, etwa »level=3« für HTML 3) sowie der qs-Parameter.

Content-Language Die Sprache(n) der Variante, ausgedrückt durch eine Standard-Sprachabkürzung (siehe RFC 3066 [RFC3066]).

Content-Encoding Beschreibt die Codierung (Komprimierung o.ä.) der Datei, falls vorhanden. Apache erkennt nur Codierungen, die vorher mit AddEncoding hinzugefügt wurden, etwa x-gzip. Für Vergleichszwecke wird das »x« ignoriert.

Content-Length Die Länge der Datei. Grundsätzlich könnte Apache auch das Dateisystem danach fragen; wenn die Länge jedoch schon in der Typemap-Datei steht, ist das überflüssig, was das Prozedere beschleunigt.

Description Eine textuelle Beschreibung der Variante. Wenn Apache keine passende Variante findet, dann gibt er eine Liste der Möglichkeiten zurück, damit der Benutzer selbst aussuchen kann. In dieser Liste erscheinen dann auch diese Beschreibungen.

MultiViews Die MultiViews-Option kann pro Verzeichnis eingeschaltet werden (siehe Abschnitt 3.5). Sie macht es möglich, dass der Server unter gewissen Umständen Typemap-Dateien »ratet« kann, die nicht tatsächlich existieren. Ist für das Verzeichnis /bla/blubb MultiViews eingeschaltet und bekommt der Server eine Anfrage nach /bla/blubb/fasel, das nicht existiert, dann sucht er nach Dateien, die auf /bla/blubb/fasel.* passen, und konstruiert daraus eine (interne) Typemap. Die Dateien bekommen dabei die Datentypen und Codierungen, die der Browser übergeben bekommen hätte, wenn er nach ihnen direkt gefragt hätte. Danach wird die Datei ausgesucht, die die Anforderungen des Browsers am besten erfüllt.

Das genaue Verfahren, das Apache zur Feststellung der »besten« passenden Variante verwendet, ist in der Apache-Dokumentation erklärt.

Interessant ist Inhaltsaushandlung vor allem in Verbindung mit verschiedenen Sprachen. Statt den Benutzer, wie man es manchmal sieht, auf der Eingangsseite einer Web-Präsenz die Sprache wählen zu lassen, können Sie von Fall zu Fall die Entscheidung auf der Basis der Sprachpräferenz des Benutzers treffen. Das macht es unnötig, zwei oder mehrere parallele Ressourcen-Bäume aufzubauen; Sie können neue Sprachen leicht sukzessive einfügen, indem Sie nach und nach die Seiten der Präsenz übersetzen (lassen). Bei geeigneter Konfiguration bekommt der Benutzer so automatisch seine Lieblingssprache oder, falls die betreffende Seite noch nicht in der betreffenden Sprache zur Verfügung steht, in einer anderen (akzeptablen) Sprache. Im Notfall wird z. B. die englische Seite verschickt.

Diese Funktionalität lässt sich nutzen, auch ohne dass Sie die Details des Verhandlungs-Algorithmus kennen. Es genügt, die Dateien geeignet zu benennen, etwa index.html.de und index.html.en. Apache erlaubt verschiedene Namenskonventionen, da die Reihenfolge der Endungen meistens irrelevant ist; index.de.html und index.en.html wären ganz genauso erlaubt.

Es ist immer möglich, überhaupt keine Endung anzugeben; index würde entsprechend der Präferenzen des Browsers ergänzt. Der Vorteil dieses Schemas ist, dass Sie den Typ einer Ressource später ändern können, ohne Verweise auf die Ressource ändern zu müssen; wird auf eine Ressource als <http://www.example.com/bla> verwiesen, ist es egal, ob die eigentliche Datei bla.html oder bla.shtml oder bla.txt oder bla.cgi heißt. Wenn Sie die MIME-Endung (etwa .html) in Verweisen beibehalten wollen, muss die Sprach-Endung rechts von der MIME-Endung stehen.

An diesem Punkt stellt sich natürlich die Frage, woher Apache weiß, in welcher Sprache der Inhalt einer Datei ist, wenn Sie es ihm nicht mit einer Typemap explizit verraten haben. Wie in den vorigen Absätzen angedeutet, verwendet er dazu Datei-Endungen. Die Zuordnung zwischen Datei-Endungen und Sprachen wird über die Direktive AddLanguage hergestellt:

```
AddLanguage da .dk
AddLanguage nl .nl
AddLanguage en .en
```

Das erste Argument ist hierbei ein RFC-3066-Sprachencode, das zweite die dafür gewünschte Endung.

 Sie sollten die RFC-3066-Sprachencodes nicht mit den Ländercodes nach ISO 3166 verwechseln; Dänisch hat zum Beispiel den Sprachencode `da`, aber den Ländercode `dk`. Eigentlich sollte das aber keine Überraschung sein, da Sprachen und Länder nicht 1 : 1 zusammenfallen. Die Sprache Deutsch spricht man außer in Deutschland ja auch in Österreich, der Schweiz und an ein paar anderen Plätzen. Umgekehrt gibt es in der Schweiz vier verschiedene offizielle Sprachen.

 Das Dateisuffix muss nicht mit dem Sprachencode identisch sein. Der Sprachencode für Polnisch ist zum Beispiel `pl`, was sich als Dateisuffix aber weniger gut eignet, da die Verwechslungsgefahr mit Perl-Skripten zu groß ist.

Sie können eine Standardsprache für alle nicht anderweitig mit einem akkreditierten Sprachsuffix gekennzeichneten Dateien in einem Verzeichnis festlegen, indem Sie die `DefaultLanguage`-Direktive verwenden:

```
DefaultLanguage fi
```

`DefaultLanguage`

Haben manche Varianten einer Ressource ein Sprachattribut und andere nicht, dann bekommen die Varianten »ohne« für die Zwecke der Inhaltsverhandlung einen impliziten Wert von 0.001. Damit können Sie eine Variante für den Fall aussersehen, dass keine auf die Präferenzen des Browsers passende Variante mit Sprachattribut gefunden wird, damit eine Fehlermeldung vermieden wird.

Angenommen, `MultiViews` ist eingeschaltet und es existieren die folgenden Varianten: `bla.en.html` mit Sprache `en`, `bla.fr.html` mit Sprache `fr` und `bla.html` ohne Sprache. Eine Variante ohne Sprache ist immer akzeptabel. Fragt jemand nach `bla` und die `Accept-Language`-Kopfzeile enthält entweder `fr` oder `en`, wird eine der Dateien `bla.fr.html` oder `bla.en.html` zurückgegeben. Wenn der Browser weder `en` noch `fr` als akzeptabel aufführt, wird `bla.html` zurückgegeben. Fragt der Benutzer direkt nach `bla.html`, findet keine Verhandlung statt; aus diesem Grund nennt man die »sprachlose« Version der Datei manchmal auch `bla.html.html`, um sicherzustellen, dass `MultiViews` richtig funktioniert.

Manche Sprachen verwenden ungewöhnliche Zeichencodierungen, die ebenfalls dem Browser signalisiert werden müssen. Auch diese Zuordnung geschieht über eine Direktive, namentlich `AddCharset`:

```
AddLanguage ja .ja
AddCharset EUC-JP .euc
AddCharset ISO-2022-JP .jis
```

`AddCharset`

Hier wird signalisiert, dass die Datei `bla.ja.euc` japanischen Text in EUC-Codierung enthält, während in der Datei `bla.ja.jis` JIS-codierter Text steht. `bla.euc.ja` und `bla.jis.ja` wären auch akzeptabel.

Übungen

 **4.5 [3]** Eine Audiodatei steht in mehreren Formaten zur Verfügung: `bla.au` ist eine 8-Bit-Mono-Variante mit 8 KHz Abtastrate im μ LAW-Format, `bla.ra` ist eine RealAudio-komprimierte Variante in Mono mit einer Abtastrate von 22,05 KHz, und `bla.mp3` ist eine MPEG-1-Layer-3-komprimierte Variante in Stereo mit einer Abtastrate von 44,1 KHz. Entwerfen Sie eine Typemap-Datei, die diese Varianten mit den korrekten MIME-Typen und angemessenen Qualitätsfaktoren aufführt. (Tip: Eine Liste von MIME-Typen finden Sie in `mime.types` in Ihrem ServerRoot-Verzeichnis.)

 **4.6 [!3]** Verwenden Sie die `Multiviews`-Option, um dieselbe Datei in einer HTML- und einer ASCII-Text-Variante anzubieten. Wenn auf die Datei mit einem URL der Form `http://localhost/bla/fasel` zugegriffen wird, soll die jeweils am besten passende Variante zurückgegeben werden. (Benutzen Sie gegebenenfalls `telnet` mit einer geeigneten HTTP-Kopfzeile, etwa

Accept»Accept: text/plain«, um zu prüfen, ob Sie tatsächlich die ASCII-Variante geschickt bekommen. Am besten schreiben Sie auch in die HTML-Variante irgendwo »HTML« und in die ASCII-Variante »ASCII«, um den Unterschied offensichtlich zu machen.)

4.5 Die Alias-Direktive

Oftmals ist es notwendig, Teile der Web-Präsenz außerhalb des Verzeichnisbaums unterhalb von `DocumentRoot` unterzubringen. Eine Möglichkeit dafür sind symbolische Links (die explizit über die Verzeichnis-Optionen `FollowSymLinks` oder `SymLinksIfOwnerMatch` eingeschaltet werden müssen – siehe Abschnitt 3.5).

Alias Eine andere Möglichkeit besteht darin, mit Hilfe der `Alias`-Direktive beliebige lokale Verzeichnisbäume in den Web-Ressourcenbaum einzubinden. (Das Modul `mod_alias` muss hierfür aktiviert sein.) Mit einer Direktive wie

```
Alias /docs /var/webdocs
```

wird eine URL wie `http://www.example.com/docs/bla/fasel.html` als Zugriff auf die Datei `/var/webdocs/bla/fasel.html` interpretiert.



Vorsicht: Wenn das erste Argument von `Alias` mit einem Schrägstrich (»/«) aufhört, dann müssen Anfragen auch den Schrägstrich enthalten, damit das Alias beachtet wird. Bei

```
Alias /images/ /var/webimages/
```

wird der URL `/images` *nicht* expandiert.



Allfällige `<Directory>`-Blöcke müssen sich auf das *Ziel* des Alias beziehen, da Aliase ausgewertet werden, bevor `<Directory>`-Blöcke überprüft werden.

AliasMatch Eine allgemeinere Form von `Alias` ist `AliasMatch`. Hier wird statt eines einfachen Präfixvergleichs ein Vergleich auf der Basis regulärer Ausdrücke vorgenommen. Auf Teilausdrücke in runden Klammern kann im Alias-Ersatztext über Verweise der Form `$1`, `$2`, ... Bezug genommen werden. So könnten Sie zum Beispiel die Funktionalität von `UserDir` simulieren, ohne die URLs der persönlichen Verzeichnisse durch das `~`-Zeichen zu entstellen:

```
AliasMatch ^/users/([^\/*])/(.*) /home/$1/public_html/$2
```

Hiermit würde ein URL der Form `http://www.example.com/users/hugo/bla/fasel.html` den Inhalt der Datei `/home/hugo/public_html/bla/fasel.html` liefern.

Apache kennt ferner die Direktiven `ScriptAlias` und `ScriptAliasMatch`. Diese funktionieren wie `Alias` und `AliasMatch`, allerdings werden die Dateien im Zielverzeichnis als CGI-Skripte ausgeführt. Hierzu steht mehr im Abschnitt 4.7.

mod_rewrite Mit dem Modul `mod_rewrite` ist eine wesentlich weitreichendere Umsetzung von URLs möglich. Hierbei kann auf Eigenschaften der HTTP-Anfrage, etwa die IP-Adresse des anfragenden Rechners oder den Browsertyp, Bezug genommen werden. Außerdem ist es möglich, externe Programme oder Datenbankdateien in die Umsetzungsentscheidung mit einzubeziehen. Näheres verrät die Apache-Dokumentation.

Übungen



4.7 [2] Auf einem System sollen Benutzer die Möglichkeit haben, ihre WWW-Seiten im Verzeichnis `/www/<Benutzername>` abzulegen (stellen Sie sich z. B. vor, dass die »echten« Heimatverzeichnisse über den Automount eingebunden werden und Sie die Zugriffe auf die benutzereigenen

WWW-Seiten beschleunigen möchten). Konfigurieren Sie Ihren Apache-Server so, dass auf die benutzereigenen Seiten (a) über URLs der Form `http://www.example.com/~<Benutzername>/blubb.html`, und (b) über URLs der Form `http://www.example.com/users/<Benutzername>/blubb.html` zugegriffen werden kann.

4.6 Weiterleitung

Während Alias und AliasMatch dafür sorgen, dass der Apache-Server Daten von einem bestimmten Platz im Dateisystem holt, kann es manchmal wünschenswert sein, dem Browser mitzuteilen, dass der gewünschte Inhalt unter einem anderen URI zu finden ist. Der Browser kann dann automatisch eine neue Anfrage stellen. Das mod_alias-Modul unterstützt auch diese **Weiterleitung**, und zwar mit der Direktive `Redirect`.

`Redirect` bildet eine alte (partielle) URL auf eine neue ab und schickt diese an den Browser, der dann versucht, die gewünschte Ressource unter der neuen URL abzurufen:

```
Redirect /bla http://www.example.com/blubb
```

Wenn der Browser `http://www.example.com/bla/fasel.html` anspricht, wird er auf `http://www.example.com/blubb/fasel.html` verwiesen. Dies ist zum Beispiel nützlich, wenn Sie eine Web-Präsenz nachträglich umorganisieren müssen.

Die Ziel-URL muss nicht auf demselben Server liegen wie dem, wo die ursprüngliche Anfrage eingeht. Mit `Redirect` lässt sich also auch ein Server-Umzug in den Griff bekommen.

 `Redirect` hat Vorrang gegenüber `Alias` und `ScriptAlias`, egal in welcher Reihenfolge die Direktiven in der Konfigurationsdatei stehen.

Es gibt verschiedene Arten von Umleitungen, die über ein weiteres Argument gesteuert werden, das Sie direkt hinter dem Wort `Redirect` einfügen:

permanent Schickt dem Browser den HTTP-Antwortcode 301 als Zeichen dafür, dass die Ressource(n) dauerhaft »umgezogen« ist (sind). Ein Browser kann dann z. B. seine »Bookmarks« oder »Favoriten« entsprechend anpassen. Der Browser sollte auch automatisch die Ressource von ihrem neuen Platz holen, jedenfalls wenn es um eine GET- oder HEAD-Anfrage ging; bei anderen Anfragen sollte das nicht ohne Rückfrage an den Benutzer passieren.

temp Schickt dem Browser den HTTP-Antwortcode 302 (temporäre Weiterleitung). In diesem Fall sollten keine Favoriten angepasst werden, sondern der Browser sollte bei künftigen Anfragen weiter den ursprünglichen URI verwenden. Trotzdem sollte er auch hier (bei GET- oder HEAD-Anfragen) automatisch die Ressource von dem umgeleiteten Ort holen und anzeigen.

seeother Schickt dem Browser den HTTP-Antwortcode 303. Dieser deutet an, dass die Antwort unter einer anderen URI zu finden ist und von dort mit einer GET-Anfrage geholt werden sollte. (Der Unterschied zu `temp` besteht darin, dass bei `seeother` *immer* GET verwendet wird, um auf die Ressource an ihrem neuen Ort zuzugreifen, während bei `temp` dieselbe Methode verwendet werden muss wie bei der ursprünglichen Anfrage. Nicht alle Browser implementieren das tatsächlich so.) `seeother` dient vor allem dazu, den Ort für eine GET-Anfrage über ein durch die HTTP-Methode POST aktiviertes Skript festzulegen. Dieser Statuscode ist neu in HTTP/1.1 und wird daher nicht notwendigerweise von allen Browsern verstanden.

gone Schickt dem Browser den HTTP-Antwortcode 410, um dem Browser mitzuteilen, dass die gewünschte Ressource nicht mehr zur Verfügung steht und

auch kein neuer URI dafür bekannt ist. Es wird angenommen, dass dieser Zustand permanent ist (ansonsten wäre der HTTP-Antwortcode 404, »*not found*«, angemessener); Browser sollten entsprechende »Favoriten« nach Rückfrage aus ihrer Datenbank löschen. Der *gone*-Status dient vor allem dazu, dem Empfänger mitzuteilen, dass eine Ressource absichtlich entfernt wurde und dass Verweise auf sie gelöscht werden sollten – etwa wenn eine Werbeaktion vorbei ist oder ein Benutzer nicht mehr dort arbeitet, wo der Server steht. Bei *gone* muss kein Ziel-URI für die Umleitung angegeben werden.

Wenn kein Statusargument angegeben wurde, wird *temp* angenommen.

 Es ist auch möglich, beliebige andere HTTP-Antwortcodes als Statusargument zu verwenden. Diese müssen dann numerisch angegeben werden. Für Codes zwischen 300 und 399 muss ein Ziel-URI mit aufgeführt werden, sonst ist das nicht erlaubt.

Ein paar Beispiele:

```
Redirect permanent /data http://newserver.example.com/data
Redirect gone /summer_sale
Redirect 303 /bla http://www.example.com/fasel
```

RedirectMatch

Mit der Direktive *RedirectMatch* können (in Analogie zur Direktive *AliasMatch*) aufwendigere Umleitungsaktionen spezifiziert werden. Hier wird kein Präfixvergleich vorgenommen, sondern ein Vergleich mit einem regulären Ausdruck. Wie bei *AliasMatch* werden die Werte von Teilausdrücken in runden Klammern in den Ziel-URI substituiert:

```
RedirectMatch (.*).gif$ http://www2.example.com$1.png
```

RedirectTemp
RedirectPermanent

 Es gibt zwei Direktiven *RedirectTemp* und *RedirectPermanent*, deren Funktionalität inzwischen in *Redirect* übernommen wurde (mit den Statusargumenten *permanent* und *temp*). Sie sollten nicht mehr verwendet werden.

mod_rewrite

Mit *mod_rewrite* können auch Umleitungen spezifiziert werden. Näheres steht in der Apache-Dokumentation; Sie sollten sich jedoch der folgenden Charakterisierung von Brian Moore bewusst sein: »*mod_rewrite is voodoo. Damned cool voodoo, but still voodoo.*«

Übungen

 **4.8** [!2] Eine Firma hat drei Abteilungen, Marketing, Entwicklung und Support, deren Web-Präsenzen bisher unter <http://www.bla.de/marketing/>, <http://www.bla.de/entwicklung/> usw. angesprochen wurden. Nun bekommen die drei Abteilungen jeweils ihre eigenen Web-Server, `marketing.bla.de`, `entwicklung.bla.de` usw. Formulieren Sie eine Weiterleitungsregel, die Zugriffe auf die »alten« URLs »permanent« auf die neuen Server umleitet.

4.7 Dynamische Inhalte

Heutzutage ist es üblich, große Teile einer Web-Präsenz nicht in Form »statischer« HTML-Dateien vorzuhalten, sondern die Inhalte bei Bedarf *dynamisch* zu generieren, etwa aus einer Datenbank. Auf diese Weise kann der Wartungsaufwand für die Inhalte minimiert werden. Beispielsweise können die Artikelseiten in einem

Produktkatalog direkt auch Informationen über Preise, Verfügbarkeit usw. der Artikel enthalten.

Eine der einfachsten Möglichkeiten, den Inhalt von HTML-Seiten zu »dynamisieren«, sind *server-side includes* (SSI). Hierbei handelt es sich um »magische« Direktiven (nicht zu verwechseln mit den Direktiven der Apache-Konfiguration), die Sie in HTML-Seiten einstreuen können. Unmittelbar bevor der Server eine solche Seite an den Browser schickt, werden die SSI-Direktiven ausgewertet. Auf diese Weise ist es möglich, dynamische Elemente in HTML-Seiten einzubauen, ohne die ganze Seite von irgendeinem Programm oder Skript generieren lassen zu müssen.

Damit *server-side includes* funktionieren, müssen sie in der `httpd.conf`-Datei oder in einer passenden `.htaccess`-Datei aktiviert werden:

```
Options +Includes
```

Das reicht aber noch nicht ganz aus; Sie müssen dem Apache auch noch sagen, welche Dateien er nach *server-side includes* durchsuchen soll.

Hierfür gibt es zwei Möglichkeiten: Sie können verfügen, dass alle Dateien mit einer bestimmten Endung bearbeitet werden sollen:

```
AddType text/html .shtml
AddHandler server-parsed .shtml
```

Diese Direktiven legen fest, dass in Dateien mit der Endung `.shtml` HTML-Text steht, und dass auf solche Dateien der *handler* für *server-side includes* angewendet werden soll. Der Nachteil dieses Ansatzes besteht darin, dass Sie den Namen einer Seite und alle Links darauf ändern müssen, wenn Sie SSI darin aufnehmen möchten.

Die andere Möglichkeit verwendet die *XBitHack*-Direktive, die dem Apache signalisiert, dass Dateien, die als ausführbar gekennzeichnet sind (das x-Bit gesetzt haben), nach *server-side includes* durchsucht werden sollen. Dann genügt es, die betreffenden HTML-Dateien mit `chmod` ausführbar zu machen:

```
$ chmod +x seite.html
```



Manchmal finden Sie Leute, die empfehlen, einfach alle HTML-Dateien nach *server-side includes* zu durchsuchen, damit Sie keine Dateien umbenennen müssen. Einerseits gibt es inzwischen den *XBitHack*, so dass das kein Problem mehr darstellt, und andererseits bedeutet diese Idee, dass der Server jede einzelne HTML-Datei vor dem Verschicken lesen und auf SSI-Direktiven abklopfen muss, selbst wenn gar keine darin enthalten sind. Das wird ihn ziemlich verlangsamen.

Die grundlegende Syntax von SSI-Direktiven ist wie folgt:

```
<!--#name attribut=wert attribut=wert ... -->
```

Das heisst, sie sind in HTML-Kommentaren »versteckt«, und wenn der Server nicht für SSI konfiguriert ist, bringen sie so den Browser nicht durcheinander. Wenn der Server korrekt konfiguriert ist, werden die Direktiven restlos durch ihre Ergebnisse ersetzt.

Hier ist eine Zusammenstellung der gängigsten SSI-Direktiven:

Variable Die Direktive `#echo` gibt den Wert einer Variablen aus:

```
<!--#echo var="DATE_LOCAL" -->
```

Es gibt einen ganzen Haufen Variable, die der Apache-Server vordefiniert (darunter alle Variable, die sonst auch an CGI-Skripte übergeben werden).

Mit der Direktive `#set` können Sie eigene Variable setzen:

```
<!--#set var="bla" value="fasel" -->
```

In Zuweisungen und andernorts können Sie auf die Werte von Variablen Bezug nehmen, indem Sie ein Dollarzeichen (`>$<`) davorsetzen (analog zu gängigen Shells):

```
<!--#set var="blubb" value="$bla" -->
```

Dateien Die Direktiven `#fsize` und `#flastmod` geben die Größe und den Zeitpunkt der letzten Änderung der im `file`-Attribut benannten Datei an:

```
<!--#fsize file="bla.html" -->
```

Für den Zeitpunkt der letzten Änderung steht bequemerweise auch die Variable `LAST_MODIFIED` zur Verfügung.

Bezugnahme auf andere Dateien Oft sollen alle Ressourcen einer Web-Präsenz ein einheitliches Erscheinungsbild haben. Mit SSI ist es leicht möglich, zum Beispiel Datei-»Köpfe« und -»enden« standardisiert vorzuhalten:

```
<html>
  <head><title>Meine Datei</title></head>
  <body>
    <!--#include file="header.html" -->
    <h1>Meine Datei</h1> ...
    <!--#include file="footer.html" -->
  </body>
</html>
```

Die Inhalte von `header.html` und `footer.html` werden an den bezeichneten Stellen in diese Datei eingefügt. `#include`-Direktiven dürfen auch verschachtelt werden, und die eingelesenen Dateien dürfen wiederum SSI-Direktiven enthalten. Zum Beispiel kann es nützlich sein, in `footer.html` ein

```
Letzte Änderung: <!--#echo var="LAST_MODIFIED" -->
```

einzbauen; das bezieht sich dann auf die ursprüngliche Datei und nicht auf `footer.html`.

Der Wert von `file` ist ein Pfad *relativ zum aktuellen Verzeichnis*. Absolute Pfade oder die Sequenz `../` sind nicht erlaubt. Oft ist es klüger, statt dem `file`-Attribut das `virtual`-Attribut zu verwenden, das statt eines relativen Pfads einen relativen URL enthalten darf. Dieser darf mit `»/«` anfangen, aber muss auf demselben Server liegen. `virtual` darf sich auch auf CGI-Skripte beziehen.

Programme ausführen Mit der SSI-Direktive `#exec` können Sie Shell-Kommandos auf dem Server ausführen. (Die Verzeichnisoption `IncludesNOEXEC` schaltet diese Direktive komplett ab.) Das Kommando steht im Attribut `cmd` und wird mit `/bin/sh` ausgeführt:

```
<pre>
<!--#exec cmd="ls -l" -->
</pre>
```

Tabelle 4.1: Bedingungen in SSI-Ausdrücken

Ausdruck	Bedeutung
$\langle s \rangle$	wahr, falls $\langle s \rangle$ nicht leer ist
$\langle s1 \rangle = \langle s2 \rangle$	Vergleiche $\langle s1 \rangle$ mit $\langle s2 \rangle$ *
$\langle s1 \rangle != \langle s2 \rangle$	*
$\langle s1 \rangle < \langle s2 \rangle$	
$\langle s1 \rangle <= \langle s2 \rangle$	
$\langle s1 \rangle > \langle s2 \rangle$	
$\langle s1 \rangle >= \langle s2 \rangle$	
$((b))$	Wahr, wenn $\langle b \rangle$ wahr ist, sonst falsch
$! \langle b \rangle$	Wahr, wenn $\langle b \rangle$ falsch ist, sonst falsch
$\langle b1 \rangle \&& \langle b2 \rangle$	Wahr, wenn Bedingung $\langle b1 \rangle$ und Bedingung $\langle b2 \rangle$ beide wahr sind
$\langle b1 \rangle \langle b2 \rangle$	Wahr, wenn Bedingung $\langle b1 \rangle$ oder Bedingung $\langle b2 \rangle$ wahr sind
$\langle s \rangle, \langle s1 \rangle, \langle s2 \rangle$	= Zeichenketten; $\langle b \rangle, \langle b1 \rangle, \langle b2 \rangle$ = Bedingungen.

* = In diesem Vergleich kann $\langle s2 \rangle$ die Form $/\langle s \rangle/$ haben und wird dann als regulärer Ausdruck interpretiert.

Alternativ dazu können Sie im Attribut cgi einen relativen Pfad zu einem CGI-Skript auf dem lokalen Server angeben:

```
<pre>
<!--#exec cgi="/cgi-bin/bla.cgi" -->
</pre>
```

Allerdings ist es in den meisten Fällen besser, #include virtual zu benutzen, um Probleme und Sicherheitslücken zu vermeiden.

Konfiguration Mit der SSI-Direktive #config können Sie die Bearbeitung von *server-side includes* beeinflussen. Sie unterstützt die Attribute errmsg, sizefmt und timefmt. sizefmt kann die Werte bytes oder abbrev annehmen und steuert die Formatierung von Dateigrößen – bytes liefert Ergebnisse in Bytes, während abbrev eine »abgekürzte« Form liefert, bei der in Kibibyte oder Mebibyte gezählt wird. timefmt übernimmt eine Zeichenkette gemäß der Funktion strftime(3) zur Formatierung von Zeitangaben (etwa bei #flastmod). errmsg erlaubt es, eine Fehlermeldung anzugeben, die bei SSI-Problemen an den Browser geschickt wird.

Bedingungen Es ist möglich, gewisse Teile der HTML-Datei nur unter gewissen Bedingungen an den Browser zu schicken (Schleifen sind leider nicht erlaubt). Die grundlegende Struktur ist

```
<!--#if expr="Ausdruck" -->
<!--#elif expr="Ausdruck" -->
<!--#else -->
<!--#endif -->
```

Die #if-Direktive wertet den *Ausdruck* aus. Ist das Ergebnis »wahr«, dann wird der Text bis zum nächsten #elif, #else oder #endif in die Ausgabe übernommen. #elif und #else können wegfallen; sie dienen dazu, Text in die Ausgabe zu übernehmen, wenn der ursprüngliche Ausdruck »falsch« war. #endif ist immer nötig.

Im *Ausdruck* sind diverse gängige Operatoren erlaubt (Tabelle 4.1). Alles, was nicht wie eine Variable oder ein Operator aussieht, wird als Zeichenkette betrachtet. Zeichenketten dürfen in einfachen Anführungszeichen stehen (doppelte sind ja schon anderweitig benutzt), damit sie Freiplatz (Leer- und Tabulatorzeichen) enthalten können. Mehrere Zeichenketten, die in einer Reihe stehen, werden bei der Auswertung durch Leerzeichen getrennt aneinandergehängt.

Ob Sie SSI verwenden oder eine andere Technik, hängt vor allem davon ab, welcher Anteil der Seiten statisch ist und welcher bei jedem Aufruf neu berechnet werden muss. Wenn sich herausstellt, dass der größte Teil einer Seite bei jedem Aufruf neu generiert wird, dann sind SSI vermutlich nicht mehr das Mittel der Wahl.

Common Gateway Interface

Eine sehr weit verbreitete Methode dafür, Web-Inhalte von Programmen erzeugen zu lassen, ist das **Common Gateway Interface**, kurz »CGI«. Schon früh in der Geschichte des WWW kam der Gedanke auf, komplexere Inhalte, etwa solche, die Datenbankabfragen enthalten, zur Verfügung zu stellen, indem der Web-Server ein Programm aufruft, das die gewünschten Inhalte zusammensucht. Die Ausgabe dieses Programms wird dann an den anfragenden Browser geleitet, der sie darstellt. CGI ist eine Konvention dafür, wie ein Web-Server ein solches externes Programm aufrufen soll; »common« ist es deshalb, weil mehrere Web-Server dieses Protokoll unterstützen (heute sind es so ziemlich alle, die der Rede wert sind).

Umgebungsvariable

CGI beruht darauf, dass die externen Programme diverse Parameter aus der HTTP-Anfrage (und andere) über Umgebungsvariable übergeben bekommen. Größere Datenmengen stellt der Server ihnen auf ihrer Standardeingabe zur Verfügung. Die Standardausgabe der externen Programme geht an den Browser, und die Standardfehlerausgabe wird sorgfältig vom Server protokolliert. Das bedeutet, dass Sie CGI-Programme (oder »Skripte«) in jeder Programmiersprache schreiben können, die einerseits auf ihre Prozessumgebung zugreifen und andererseits mit den Standardein- und ausgabekanälen umgehen kann. Unter Linux trifft das auf praktisch jede Programmiersprache zu, angefangen mit der Shell bis hin zu C oder FORTRAN. Trotzdem sind die Programmiersprachen der Wahl für CGI-Programmierung heutzutage interpretierte »Skriptsprachen« wie Perl, Tcl, Python oder PHP.

ScriptAlias

Der Apache-Server bietet mehrere Möglichkeiten an, CGI-Skripte ausführbar zu machen. Die gängigste und einfachste ist, mit der `ScriptAlias`-Direktive zu erklären, dass ein Verzeichnis CGI-Skripte enthält:

```
ScriptAlias /cgi-bin/ /usr/local/apache/cgi-bin/
```

zeigt dem Apache an, dass Anfragen nach URLs wie `http://www.example.com/cgi-bin/test.pl` auf Dateien in `/usr/local/apache/cgi-bin` verweisen, und dass die betreffenden Dateien als CGI-Skripte ausgeführt werden sollen, anstatt textuell an den anfragenden Browser geschickt zu werden. In diesem Beispiel würde der Apache-Server versuchen, `/usr/local/apache/cgi-bin/test.pl` zu starten, und die Ausgabe dieses Programms an den Browser schicken. Natürlich muss die Datei existieren und ausführbar sein, und die Ausgabe muss auch gewissen Anforderungen genügen, sonst gibt es nur eine Fehlermeldung.

ScriptAliasMatch

 Analog zu `Alias` und `AliasMatch` gibt es auch eine Direktive namens `ScriptAliasMatch`, die die Zuordnung zwischen URLs und CGI-Verzeichnissen auf der Basis regulärer Ausdrücke herstellt. So könnten Sie zum Beispiel mit

```
ScriptAliasMatch ^/~([^\/]+)/cgi-bin/ /home/$1/public_cgi
```

Benutzern in Analogie zu ihrem `public_html`-Verzeichnis ein `public_cgi`-Verzeichnis zur Verfügung stellen, in dem sie ihre eigenen CGI-Skripte unterbringen können.

Meist ist es aus sicherheitstechnischen Gründen sinnvoll, CGI-Skripte nur in mit `ScriptAlias` dafür freigegebenen Verzeichnissen zu erlauben. Mit den nötigen Sicherheitsvorkehrungen spricht jedoch nichts dagegen, CGI-Skripte nicht auch in anderen Verzeichnissen auszuführen. Beispielsweise können Sie die Ausführung von CGI-Skripten in einem Verzeichnis erlauben, indem Sie die Verzeichnis-Option `ExecCGI` einschalten (siehe 3.5):

```
<Directory /var/htdocs/bla/blubb>
  Options +ExecCGI
</Directory>
```

Dann weiß der Apache aber immer noch nicht, welche Dateien als CGI-Skripte anzusehen sind. Dies lässt sich durch eine `AddHandler`-Direktive erreichen:

```
AddHandler cgi-script cgi pl
```

erklärt alle Dateien, die auf `cgi` oder `pl` enden, zu CGI-Skripten.

Diese verzeichnisorientierte Freischaltung kann natürlich auch in `.htaccess`-Dateien erfolgen. Vorbedingung dafür ist, dass an geeigneter Stelle in der `httpd.conf`-Datei ein

```
AllowOverride Options
```

untergebracht wurde.

Wenn ein CGI-Skript identifiziert wurde und gestartet werden soll, richtet der Apache-Server eine geeignete Prozessumgebung für das Skript her, die die notwendigen Parameter enthält, und erzeugt einen Kindprozess, der diese Umgebung übergeben bekommt. In diesem Kindprozess wird dann das CGI-Skript gestartet (gegebenenfalls über den dafür zuständigen Interpreter, etwa `/usr/bin/perl`). Auf der Standardeingabe des Skripts stehen, falls die HTTP-Anfrage mit der Methode `POST` gestellt wurde, die im Browser zum Beispiel in ein HTML-Formular eingegebenen Parameter zur Verfügung. Die Standardausgabe des Skripts wird vom Apache-Server gelesen und an den Browser weitergereicht.

Prozessumgebung

Leider kann in dieser Schulungsunterlage nicht detailliert auf das Thema »CGI-Programmierung« eingegangen werden. Die Apache-Dokumentation, diverse Web-Seiten und handelsübliche Literatur aus der Fachbuchhandlung behandeln dieses Thema viel ausführlicher.

4.8 Fehlermeldungen

In vielen Situationen, etwa wenn eine Anfrage nach einer nicht existierenden Ressource gestellt wurde, gibt der Apache-Server eine **Fehlermeldung** zurück. Diese Fehlermeldungen werden automatisch als HTML-Seiten generiert.

Fehlermeldung

Im Interesse der *corporate identity* und der Benutzerfreundlichkeit ist es oft angebracht, die Standard-Fehlerseiten des Apache durch eigene zu ersetzen. Beispielsweise könnten Sie auf einer eigenen Fehlerseite neben dem Firmenlogo und angemessen entschuldigender Prosa eine Suchmöglichkeit oder Links auf die wichtigsten Seiten der Web-Präsenz unterbringen, damit die Benutzer sich unterstützt fühlen statt mit `404 Not Found` vor den Kopf gestoßen.

Apache erlaubt über die `ErrorDocument`-Direktive die Definition eigener Fehlerseiten. Diese Direktive hat zwei Argumente, nämlich einen numerischen HTTP-Statuscode und eine Spezifikation für die Fehlerseite. Letztere kann eine relative oder absolute URL sein oder direkt eine Nachricht, dann muss sie aber mit einem doppelten Anführungszeichen (»"«) beginnen:

ErrorDocument

```
ErrorDocument 500 http://www.example.com/interror.html
ErrorDocument 401 /subscribe_info.html
ErrorDocument 403 "Heute nicht, sorry"
```

Wird eine absolute URL (eine mit »`http://`« am Anfang) angegeben, dann schickt Apache eine Weiterleitung an den Browser, selbst wenn die URL auf denselben Server verweist. Dabei geht der ursprüngliche HTTP-Statuscode verloren (er wird durch den der Weiterleitung ersetzt), was ein Problem sein kann. Insbesondere gilt das für den Statuscode `401`, den ein Browser geschickt bekommt, wenn er den

Benutzer nach einem Benutzernamen und Kennwort fragen soll; wird für diesen Statuscode ein ErrorDocument mit Weiterleitung vereinbart, dann wird der Benutzer nie nach seinen Daten gefragt und kann die »geschützte« Ressource darum nie zu sehen bekommen.

ErrorDocument-Direktiven können Sie in .htaccess-Dateien einbauen, wenn »AllowOverride FileInfo« in Kraft ist.

Übungen



4.9 [!2] Konfigurieren Sie Ihren Apache-Server so, dass Zugriffe auf nicht existierende Seiten mit einer Fehlerseite beantwortet werden, die Verweise auf einige wichtige Seiten auf Ihrem Server enthält (etwa die Hauptseite).

Zusammenfassung

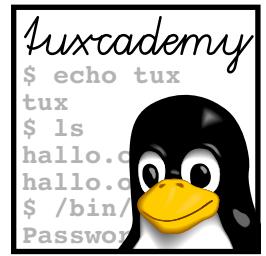
- Eine gut überlegte Verzeichnisstruktur kann eine Web-Präsenz überschau-
barer und wartbarer machen.
- Ist die Option Indexes für ein Verzeichnis aktiv und existiert keine DirectoryIndex-
Datei, so zeigt Apache eine automatisch erzeugte Liste der Dateien im Ver-
zeichnis an. Das Aussehen der Liste kann weitreichend konfiguriert werden.
- Benutzereigene Verzeichnisse können mit UserDir definiert und über URLs
der Form `http://www.example.com/~<Benutzername>/` angesprochen werden.
- Inhaltsaushandlung erlaubt es, eine Ressource in verschiedenen Varianten
zur Verfügung zu stellen, aus denen der Browser die »beste« aussucht.
- Mit den Alias- und AliasMatch-Direktiven können andere Verzeichnisse in die
Ressourcenhierarchie eingebunden werden.
- Die Direktiven Redirect und RedirectMatch erlauben die Weiterleitung von Zu-
griffen an andere URIs.
- *Server-side includes* erlauben die Integration dynamischer Elemente in HTML-
Seiten; sie werden ersetzt, unmittelbar bevor die Seite an den Browser aus-
geliefert wird.
- Das *Common Gateway Interface* (CGI) ist eine Methode, mit der Web-Server
externe Programme aufrufen und deren Ausgabe an den Browser weiterlei-
ten können.
- Mit ErrorDocument können Sie die eingebauten Fehlermeldungen des Apache-
Servers durch eigene ersetzen. Auch Umleitungen sind möglich.

Literaturverzeichnis

RFC2295 K. Holtman, A. Mutz. »Transparent Content Negotiation in HTTP«,
März 1998. <http://www.ietf.org/rfc/rfc2295.txt>

RFC2296 K. Holtman, A. Mutz. »HTTP Remote Variant Selection Algorithm –
RVSA/1.0«, März 1998. <http://www.ietf.org/rfc/rfc2295.txt>

RFC3066 H. Alvestrand. »Tags for the Identification of Languages«, Januar 2001.
<http://www.ietf.org/rfc/rfc3066.txt>



5

Zugriffsrechte und Zugriffsschutz

Inhalt

5.1	Überblick	60
5.2	Zugriffskontrolle auf TCP-Ebene	60
5.3	HTTP-basierte Authentisierung	62
5.4	Apache und SSL	68

Lernziele

- Zugriffskontrolle auf Ressourcen auf TCP-Ebene konfigurieren können
- Ablauf von HTTP-Authentisierung verstehen
- Zugriffsrechte auf Ressourcen über HTTP-Authentisierung konfigurieren können

Vorkenntnisse

- Überblick über Apache-Konfiguration (Kapitel 3)
- TCP/IP-Kenntnisse

5.1 Überblick

Oft möchten Sie gewisse Teilmengen der Ressourcen auf einem Web-Server nur bestimmten Benutzerkreisen zugänglich machen. Zum Beispiel könnte ein Web-Server »öffentliche« Ressourcen enthalten, die für die ganze Welt zu sehen sind, und »private«, die nur für Mitglieder oder nur für lokale Benutzer zugänglich sein sollen. Der Apache-Server bietet verschiedene Möglichkeiten an, dies in die Tat umzusetzen:

- | | |
|----------------------------------|---|
| Zugriffskontrolle | <ul style="list-style-type: none"> • Zugriffskontrolle (engl. <i>access control</i>) auf der TCP-Ebene. Hiermit können Sie bestimmte Rechner oder Netze vom Web-Server ausschließen oder ihnen ausdrücklich den Zugriff erlauben. |
| Authentisierung
Autorisierung | <ul style="list-style-type: none"> • Mit Authentisierung und Autorisierung auf HTTP-Ebene können Sie zwischen verschiedenen Benutzern unterscheiden und ihnen aufgrund ihrer Identität oder Zugehörigkeit zu einer Benutzergruppe verschiedene Rechte geben. |
| SSL | <ul style="list-style-type: none"> • Apache kann Benutzer auch auf der Basis von SSL-Zertifikaten authentisieren. Häufiger verwendet werden allerdings SSL-Zertifikate für Server, damit zum Beispiel die Kunden von »Online-Shops« die Authentizität der Server prüfen und wichtige Daten wie Kreditkartennummern verschlüsselt übertragen können. |
| Unterschied | Der Unterschied zwischen Zugriffskontrolle und Authentisierung besteht darin, dass bei der Authentisierung der Benutzer etwas vorweist, was ihn legitimiert – eine Kombination aus Benutzernamen und Kennwort oder ein Client-Zertifikat. Die Zugriffskontrolle orientiert sich an potentiell nahezu beliebigen Kriterien – neben IP-Adressen und Rechner- und Domainnamen auch weiteren, auf die der Benutzer ähnlich wenig Einfluss hat (die Mondphase). Siehe hierzu auch Übung 5.3. |

5.2 Zugriffskontrolle auf TCP-Ebene

Zugriffskontrolle auf TCP-Ebene dient dazu, Rechnern den Zugriff auf den Server gemäß ihres Rechnernamens oder ihrer IP-Adresse zu erlauben. Die TCP-basierte Zugriffskontrolle des Apache-Servers alleine ist kein besonders starkes Hindernis, da Rechnernamen oder IP-Adressen leicht »gefälscht« werden können. Sie sollten also zumindest dafür sorgen, dass ein geeigneter Firewall oder zumindest ein Paketfilter plumpen Attacken unterbindet und zum Beispiel Pakete mit »internen« Absenderadressen, die aus dem »externen« Netz kommen, verwirft.

Allow TCP-basierte Zugriffskontrolle wird über die Direktiven `Allow`, `Deny` und `Order` konfiguriert. Diese Direktiven können in `<Directory>`-, `<Files>`- oder `<Location>`-Blöcken auftreten; in `.htaccess`-Dateien sind sie erlaubt, wenn »`AllowOverride` `Limit`« aktiv ist. Die allgemeine Syntax ist

```
Allow from <Adresse>
Deny from <Adresse>
```

wobei `<Adresse>` eine der folgenden Möglichkeiten ist:

IP-Adresse • Eine (komplette oder partielle) IP-Adresse. Ein einzelner Rechner kann zum Beispiel mit

```
Deny from 11.22.33.44
```

ausgesperrt werden, ein ganzes Netz mit etwas wie

```
Deny from 11.22
```

- Eine Kombination aus Adresse und Netzmaske, etwa wie

```
Allow from 192.168.14.160/255.255.255.224
Allow from 192.168.14.160/27
```

(die beiden Allows im Beispiel beschreiben genau dieselben Rechner).

- Ein Rechner- oder Domain- oder partieller Domainname, zum Beispiel

```
Allow from example.com
```

Hierbei werden nur komplette Teilnamen betrachtet, das Beispiel passt also auf `tux.example.com`, aber nicht auf `tuxexample.com`. Apache prüft dafür nach, welcher Name zu der IP-Adresse der Anfrage gehört (in der der Name ja nicht verlässlich drinsteht).

- Das Schlüsselwort `all` sperrt alle Rechner aus oder lässt sie zu. In Abhängigkeit von der `Order`-Direktive kann das durchaus Zweck haben.



Es gibt für `Allow` und `Deny` auch »`from env=<Umgebungsvariable>`«. Hierbei wird der Zugriff gestattet bzw. abgewiesen, wenn die benannte `(Umgebungsvariable)` existiert. Mit der Direktive `SetEnvIf` aus dem Modul `mod_setenvif` können Sie Umgebungsvariable auf der Basis von HTTP-Kopfzeilen oder anderen Aspekten der Anfrage setzen.

Die `Order`-Direktive regelt, in welcher Reihenfolge die `Allow`- und `Deny`-Direktiven `Order` ausgewertet werden. Die möglichen Werte sind:

Deny,Allow Hiermit werden die `Deny`-Direktiven vor den `Allow`-Direktiven bearbeitet. Jeder Client, der *nicht* auf eine `Deny`-Direktive passt, oder der auf eine `Allow`-Direktive passt, bekommt Zugang zum Server. Clients, die weder von den `Allow`- noch den `Deny`-Direktiven erfasst werden, bekommen ebenfalls Zugang.

Allow,Deny Hiermit werden die `Allow`-Direktiven vor den `Deny`-Direktiven bearbeitet, und standardmäßig wird der Zugang verweigert. Clients, die *nicht* auf eine `Allow`-Direktive passen, oder die auf eine `Deny`-Direktive passen, werden abgewiesen.

Mutual-failure Lässt nur diejenigen Clients zu, die auf eine `Allow`-Direktive passen und nicht auf eine `Deny`-Direktive. Dies entspricht im wesentlichen `Allow,Deny` und ist deswegen inzwischen verpönt.

Es werden immer alle Direktiven betrachtet, auch wenn bereits passende gefunden wurden.

Zum Beispiel:

```
Order Deny,Allow
Deny from all
Allow from example.com
```

Hier bekommen alle Clients aus der Domain `example.com` Zugang; alle anderen werden abgewiesen. Im folgenden Beispiel,

```
Order Allow,Deny
Allow from example.com
Deny from bla.example.com
```

dürfen alle Clients aus `example.com` auf den Server zugreifen, mit Ausnahme derjenigen, die in der Subdomain `bla.example.com` sind. Alle anderen Rechner werden abgewiesen (wegen `Allow,Deny`). Stünde in diesem Beispiel `Deny,Allow` statt `Allow,Deny`, bekämen alle Clients (egal von wo) Zugriff. In diesem Fall würde nämlich die `Allow`-Direktive nach der `Deny`-Direktive abgearbeitet, und `Allow from example.com` würde `Deny from bla.example.com` widerrufen. Alle Clients außerhalb von `example.com` bekämen Zugriff, weil das bei `Deny,Allow` die Vorgabe ist (sie treten in keiner Direktive auf).

Zwei wichtige Feststellungen sind noch notwendig: Zum einen kann eine einzelne `Order`-Direktive schon die Zugriffsrechte für Teile eines Servers ändern, selbst wenn überhaupt keine `Allow`- oder `Deny`-Direktiven angegeben wurden, weil die Standardvorgabe bei `Deny,Allow` »Zulassen« ist und bei `Allow,Deny` »Abweisen«. Zum anderen beschränkt die von `Order` vorgegebene Auswertungsreihenfolge sich nur auf jede einzelne Phase der Konfigurationsauswertung. Beispielsweise wird eine `Allow`- oder `Deny`-Direktive in einem `<Location>`-Block immer nach einer `Allow`- oder `Deny`-Direktive in einem `<Directory>`-Block oder einer `.htaccess`-Datei ausgewertet, egal auf welchen Wert `Order` gesetzt ist, da `<Location>`-Blöcke insgesamt erst nach `<Directory>`-Blöcken und `.htaccess`-Dateien ausgewertet werden.

Übungen

 **5.1** [!2] Welche Rechner bekommen bei der folgenden Konfiguration Zugriff auf die entsprechenden Inhalte?

```
Order Allow,Deny
Deny from all
Allow from 192.168.5.0/24
Deny from 192.168.5.111
```

 **5.2** [2] Stellen Sie sich einen Apache-Server vor, der konfiguriert ist wie in der vorigen Übung und der selbst die IP-Adresse 192.168.5.1 hat. Sie melden sich auf dem Apache-Server an und wollen lokal mit einem Browser auf die Inhalte zugreifen, etwa über `http://localhost/`. Was passiert und warum?

 **5.3** [4] Statt einer IP-Adresse können Sie in `Allow` und `Deny` auch den Namen einer Umgebungsvariable in der Form `env=<Variable>` angeben. Dann wird der Zugriff zugelassen, wenn die betreffende Umgebungsvariable für den Zugriff gesetzt ist, und sonst abgewiesen. Mit der Direktive `SetEnvIf` (aus dem Modul `mod_setenvif`) können Sie Umgebungsvariable zum Beispiel in Abhängigkeit von HTTP-Kopfzeilen setzen: Die Direktive

```
SetEnvIf Referer ^http://www.example.com/ from_example_com
```

setzt die Umgebungsvariable `from_example_com`, wenn der Zugriff über einen Verweis auf einer Seite erfolgt, die auf dem Server `www.example.com` liegt, und mit einer Direktive wie

```
Deny from env=from_example_com
```

können Sie anschliessend alle Zugriffe über solche Verweise ausschließen. – Geben Sie eine Konfiguration an, die analog hierzu alle Zugriffe ausschließt, die über einen Firefox-Browser erfolgen.

5.3 HTTP-basierte Authentisierung

Basic-Authentisierung (engl. *basic authentication*) ist die einfachste Authentisierungsmethode. Ist eine Ressource über Basic-Authentisierung geschützt, schickt

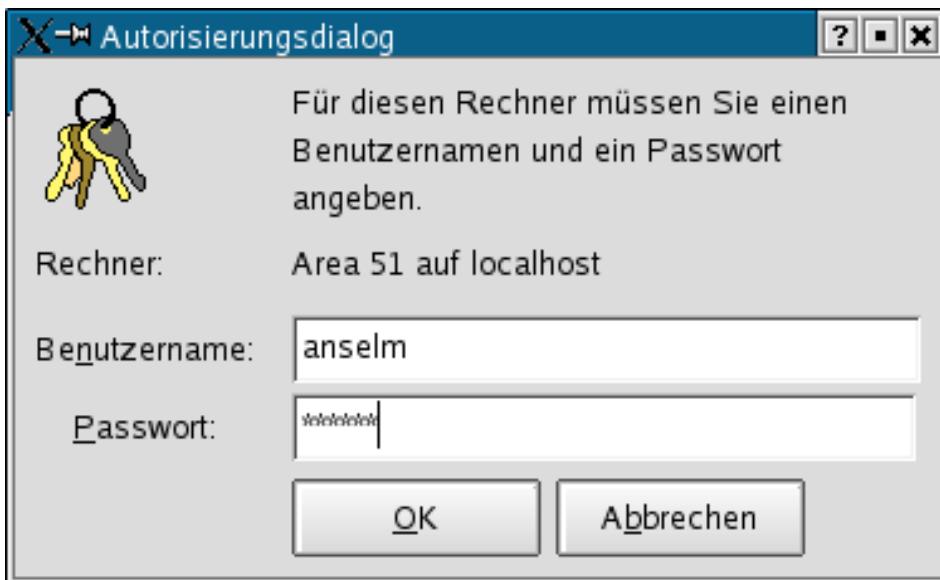


Bild 5.1: Typische Dialogbox für HTTP-Authentisierung

Apache bei einem Zugriffsversuch eine »authentication required«-Antwort an den anfragenden Browser (HTTP-Statuscode 401). Für den Browser ist das ein Signal, den Benutzer nach seinem Benutzernamen und Kennwort zu fragen (Bild 5.1). Diese Informationen werden an den Server zurückgeschickt, und wenn das Kennwort zum Benutzernamen passt und der Benutzer auf der Liste der Befugten steht, wird die Ressource an den Browser ausgeliefert. Dies gilt genauso für jede neue Anfrage, da HTTP zustandslos ist; der Browser versteckt jedoch die Details, da er den eingegebenen Benutzernamen und das Kennwort ungefragt weiterbenutzt. Beenden Sie den Browser und starten ihn neu, so müssen Sie in aller Regel Benutzernamen und Kennwort nochmals eingeben.

Die Authentisierung erfolgt immer für einen **Authentisierungsbereich** (engl. *realm*). Dieser wird über die Direktive `AuthName` angegeben und bei der Anfrage nach Benutzernamen und Kennwort an den Browser weitergereicht (der Browser zeigt ihn normalerweise in der Dialogbox an, wo nach Benutzernamen und Kennwort gefragt wird). Damit ist es möglich, auf demselben Server mehrere unabhängige Authentisierungsbereiche zu unterstützen; der Browser speichert intern ein Tripel

$(\langle \text{Authentisierungsbereich} \rangle, \langle \text{Benutzername} \rangle, \langle \text{Kennwort} \rangle)$

und benutzt es bei Bedarf wieder.

Um Ressourcen mit Basic-Authentisierung zu schützen, sind mehrere Schritte notwendig. Zuerst muss eine **Kennwortdatei** angelegt werden, und anschließend muss der Apache-Server so konfiguriert werden, dass er diese Kennwortdatei tatsächlich benutzt.

Kennwortdateien für Apache enthalten – ähnlich wie die Kennwortdateien des Linux-Systems, `/etc/passwd` und `/etc/shadow` – Benutzernamen und die dazugehörigen verschlüsselten Kennwörter. (Die von Apache verwendeten Verschlüsselungsverfahren ähneln denen, die Linux für seine Kennwörter benutzt.) Sie können irgendwo im Dateisystem liegen, aber sinnvollerweise plazieren Sie sie außerhalb des `DocumentRoot`-Verzeichnisses und seiner Unterverzeichnisse, oder allgemeiner gesagt außerhalb des über den Web-Server zugänglichen Bereichs. Die Kennwörter sind zwar unentschlüsselbar verschlüsselt, aber einerseits gibt es Software, die automatisch versucht, verschlüsselte Kennwörter zu brechen, indem wahrscheinlich aussehende Texte probehalber verschlüsselt werden, und andererseits geben die meisten Benutzer sich keine große Mühe mit Kennwörtern für irgendwelche Web-Ressourcen, so dass Sie einem halbwegs entschlossenen Cracker die Kennwortdatei nicht auf einem silbernen Tablett präsentieren sollten.

Authentisierungsbereich
AuthName

Kennwortdatei

Benutzernamen
Kennwörter

htpasswd Apache enthält ein Hilfsprogramm namens `htpasswd`, mit dem Sie Kennwortdateien anlegen können. Es übernimmt als Parameter den Namen der gewünschten Kennwortdatei und den einzutragenden Benutzernamen. Beim ersten Anlegen der Datei müssen Sie außerdem die Option `-c` übergeben:

```
# htpasswd -c /usr/local/www/passwd hugo
New password: blaFasel
Re-type new password: blaFasel
Adding password for user hugo
```

*hier wird das neue Kennwort eingegeben
und hier nochmal*

Dieses Kommando erzeugt eine Datei `/usr/local/www/passwd` und trägt dort den Benutzer `hugo` mit seinem (verschlüsselten) Kennwort ein. Die Datei sieht dann so aus:

```
hugo:$apr1$0g6zb2eI$4ZXF9l2lHMPLhC0pGcrQu.
```

(das verschlüsselte Kennwort kann variieren).



Das Programm `htpasswd` verwendet standardmäßig eine Apache-spezifische Methode zur Kennwortverschlüsselung, die auf MD5 beruht. MD5 ist als kryptografische Hashfunktion nicht mehr unbedingt Stand der Technik, allerdings sorgt das Apache-Verfahren dafür, dass die gröbsten Probleme nicht zum Tragen kommen. Zum Beispiel wird MD5 nicht nur einmal, sondern tausendmal nacheinander auf verschiedene Kombinationen einer Zufallszahl und des Kennworts angewendet.



`htpasswd` unterstützt auch andere Verschlüsselungsverfahren. Diese Verfahren sind aber durchweg schwächer. Um die traditionelle Unix-Kennwortverschlüsselung (CRYPT) sollten Sie einen weiten Bogen machen, und auch von der SHA1-basierten Verschlüsselung (SHA) ist dringend abzuraten: Hier wird das Kennwort nämlich nur einmal mit SHA1 gehasht, und es wird auch keine zufällige Verwürfelung (»Salz«) mit eingebaut, so dass Angreifer, denen Ihre Kennwortdatei in die Hände fällt, vergleichsweise leichtes Spiel haben, um Kennwörter zu »raten«.



Welches Verschlüsselungsverfahren `htpasswd` verwendet, können Sie pro Kennwort auswählen. In derselben Kennwortdatei können ganz unterschiedlich verschlüsselte Kennwörter stehen.



Apache betrachtet nur die ersten beiden Felder jeder Zeile in der Kennwortdatei. Grundsätzlich könnten Sie also auch gegen `/etc/passwd` bzw. `/etc/shadow` authentisieren, wenn die Zugriffsrechte das zulassen. Empfehlenswert ist das aber nicht wirklich.

Wenn Sie weitere Benutzer eintragen wollen, müssen Sie die `-c`-Option weglassen:

```
# htpasswd /usr/local/www/passwd emil
```

trägt `emil` als weiteren Benutzer in die Datei `/usr/local/www/passwd` ein. Die Option `-c` würde alle schon existierenden Benutzer löschen!

Zugriffsrechte für die Kennwortdatei Es empfiehlt sich, die Zugriffsrechte für die Kennwortdatei so zu setzen, dass nur der Benutzer, mit dessen Identität der Apache-Server läuft (etwa `nobody`, `www-data` oder `wwwrun`, abhängig von Ihrem System) die Kennwortdatei lesen darf.

Konfiguration Wenn die Kennwortdatei angelegt ist, muss der Apache-Server noch so konfiguriert werden, dass er sie auch verwendet. Diese Konfiguration können Sie in einem `<Directory>`-Block (oder einer anderen Sorte von bereichsbegrenzenden Block) in `httpd.conf` unterbringen; wenn für ein Verzeichnis »`AllowOverride AuthConfig`« gesetzt ist, auch in einer `.htaccess`-Datei.

AuthType Zunächst stellen Sie mit der Direktive »`AuthType Basic`« ein, dass Basic-Authen-

tisierung gewünscht wird. Der Authentisierungsbereich bekommt mit AuthName einen Namen zugewiesen, der dann auch im Browser in der Eingabeaufforderung für Benutzernamen und Kennwort erscheint. Mit der AuthUserFile-Direktive geben Sie an, welche Kennwortdatei zur Überprüfung der Zugriffsrechte verwendet wird:

```
AuthType Basic
AuthName "Area 51"
AuthUserFile /usr/local/www/passwd
```

AuthUserFile

 Grundsätzlich zwingt Sie niemand dazu, die Benutzerdaten in Dateien abzulegen. Wenn Sie sehr viele Benutzer haben, kann sich das sogar zu einem Effizienzflaschenhals auswachsen, weil für jeden Zugriff die Datei nach den Informationen des betreffenden Benutzers durchsucht werden muss. In so einem Fall ist es günstiger, auf eine Datenbank auszuweichen, die den direkten Zugriff auf die (oder doch zumindest eine flottere Suche nach den) Benutzerdaten ermöglicht. Mit der Direktive AuthBasicProvider können Sie eine oder mehrere Datenquellen (engl. *authentication providers*) angeben: »file« steht für »Dateien im Dateisystem« gemäß dem Modul mod_authn_file; andere mögliche Werte sind zum Beispiel »dbm« für Benutzerdaten in DBM-Dateien (inklusive typischerweise Berkeley DB) mit dem Modul mod_authn_dbm, »dbd« für Zugriffe auf SQL-Datenbanken (etwa PostgreSQL) mit mod_authn_dbd oder »ldap« für Zugriffe auf LDAP-Verzeichnisse mit mod_authnz_ldap. Für aufwendige Operationen wie die Suche auf entfernten LDAP- oder SQL-Servern kann mit mod_authn_socache ein Cache eingeführt werden.

AuthBasicProvider
engl. *authentication providers*

 Es spricht gegebenenfalls auch nichts dagegen, mehrere Quellen für Benutzerdaten anzugeben (AuthBasicProvider lässt eine Liste zu). Wenn keine Informationen über den gesuchten Benutzer gefunden wurden, weist Apache den Zugriff ab, es sei denn, die Direktive AuthBasicAuthoritative hat den Wert »0ff«. In diesem Fall können andere Authentisierungsmodule zum Zug kommen.

mehrere Quellen

AuthBasicAuthoritative

Wenn die Quelle(n) für die Benutzerdaten definiert sind, können Sie mit der Direktive Require bestimmen, welche Bedingungen ein Benutzer erfüllen muss, um Zugang zu der Ressource zu bekommen. Und zwar:

Require

valid-user Der Benutzer muss einen gültigen (d. h. in einer Quelle gefundenen) Benutzernamen und das dazu passende Kennwort angeben. Welcher Benutzername das ist, ist egal, solange er nur in der Kennwortdatei vorkommt.

user *<benutzer>* ... Wenn user angegeben wurde, gefolgt von einem oder mehreren Benutzernamen, werden nur die benannten Benutzer zugelassen, die natürlich außerdem das jeweils korrekte Kennwort angegeben haben müssen.

group *<gruppe>* ... Es gibt auch die Möglichkeit, mehrere Benutzer in Gruppen zusammenzufassen und den Zugriff zu Ressourcen auf der Basis dieser Gruppen zu gestatten oder zu verweigern. Hierzu muss eine **Gruppendatei** angelegt werden (siehe unten). Der group-Parameter lässt einen Benutzer zu, wenn er sein korrektes Kennwort angegeben hat und außerdem in (mindestens) einer der benannten Gruppen enthalten ist.

Gruppendatei

Den Namen einer Gruppendatei können Sie mit AuthGroupFile festlegen. Die Gruppendatei ist sehr einfach aufgebaut; sie besteht aus Zeilen der Form

AuthGroupFile

```
webadmins: hugo emil
```

das heißt, einem Gruppennamen gefolgt von einem Doppelpunkt und einer Liste der Benutzer, die Mitglieder der Gruppe sind.

 Tatsächlich werden diese `Require`-Bedingungen nicht von Apache selbst implementiert, sondern von den betreffenden Authentisierungsmodulen (etwa `mod_authz_user`). Weitere Authentisierungsmodule können auch noch andere Arten von Bedingungen zur Verfügung stellen.

In Apache 2.4 gibt es auch noch ein paar andere Formen der `Require`-Direktive, namentlich

all granted Alle Zugriffe werden zugelassen.

all denied Alle Zugriffe werden zurückgewiesen.

env `<Variable>` ... Der Zugriff wird zugelassen, wenn mindestens eine der aufgezählten Umgebungsvariablen existiert.

method `<HTTP-Methode>` ... Nur Zugriffe mit den aufgezählten HTTP-Methoden werden zugelassen.

expr `<Ausdruck>` Der Zugriff wird nur zugelassen, wenn `<Ausdruck>` den Wert »wahr« ergibt. Über Ausdrücke können Sie in der Apache-Dokumentation nachlesen; als kleiner Vorgeschmack hier ein Verzeichnis, dessen Ressourcen nur während der »üblichen Bürozeiten« zugänglich sind:

```
<Directory /business>
  Require expr %TIME_HOUR -gt 9 && %TIME_HOUR -lt 17
</Directory>
```

Zugriffsregeln in Blöcken

Ebenfalls neu in Apache 2.4 ist die Möglichkeit, beliebige Zugriffsregeln in Blöcken zusammenzufassen: Bei

```
<RequireAny>
  Require ip 192.168.1.0/24
  Require valid-user
</RequireAny>
```

dürfen Benutzer entweder aus dem IP-Netz 192.168.1.0/24 (Stichwort »Intranet«) oder über Eingabe von Benutzername und Kennwort zugreifen. Mit

```
<RequireAll>
  Require ip 192.168.1.0/24
  Require valid-user
</RequireAll>
```

können Sie verlangen, dass die Benutzer *sowohl* im Intranet sitzen *als auch* einen gültigen Benutzernamen nebst Kennwort angeben müssen.

not  In Apache 2.4 können Sie mit `not` das Ergebnis der Authentisierung »umdrehen«. Etwas wie

```
<RequireAll>
  Require ip 192.168.1.0/24
  Require not user susi
</RequireAll>
```

würde also verlangen, dass der Benutzer nicht `susi` ist. Allerdings kann die `not`-Konstruktion nur dazu führen, dass ein Zugriff abgewiesen wird – der Zugriff muss ein »normales« `Require` erfüllen, um durchgelassen zu werden. Im Beispiel muss ein Benutzer also im Intranet sitzen *und* nicht `susi` sein; Konstruktionen wie

```
<RequireAll>
  Require not ip 192.168.1.0/24
  Require not user susi
</RequireAll>
```

oder

```
<RequireAll>
  Require not user susi
</RequireAll>
```

würden nicht funktionieren.

 In einem `<RequireNone>`-Block darf *keine* der `Require`-Direktiven zutreffen, damit der Zugriff erlaubt wird.

 Sie können die verschiedenen Blöcke für Zugriffsregeln auch verschachteln. Das folgende an die Apache-Dokumentation angelehnte Beispiel illustriert dies:

```
<Directory /srv/www/htdocs>
  <RequireAll>
    <RequireAny>
      Require user superadmin
    <RequireAll>
      Require group admins
      <RequireAny>
        Require group sales
        Require group marketing
      </RequireAny>
    </RequireAll>
    </RequireAny>
    <RequireNone>
      Require group temps
      Require group interns
    </RequireNone>
  </RequireAll>
</Directory>
```

In diesem Beispiel muss ein Benutzer entweder der Benutzer `superadmin` sein oder aber sowohl zur Gruppe `admins` als auch einer der Gruppen `sales` und `marketing` gehören. Außerdem darf er weder Mitglied der Gruppe `temps` noch Mitglied der Gruppe `interns` sein.

Sie sollten sich nicht einbilden, dass Basic-Authentisierung besonders sicher sei. Zum einen werden Benutzernamen und Kennwörter im Klartext übertragen (genau wie die »geschützten« Ressourcen, wenn die Autorisierung geglückt ist), zum anderen müssen sie in jeder Anfrage mit angegeben werden, so dass sie für einen Angreifer relativ leicht zu erhaschen sind – es ist nicht so, dass er einen ganz bestimmten Zeitpunkt abpassen müsste. Basic-Authentisierung ist eher ein elektrischer Weidezaun als eine fünf Meter hohe Mauer mit Stacheldraht und Wachtürmen; Sie setzen sie ein, um Unfug zu vermeiden, aber nicht, weil Sie wirklich etwas schützen wollen. Dafür gibt es SSL.

Eine Alternative zu Basic-Authentisierung, bei der wenigstens keine Kennwörter im Klartext über das Netz geschickt werden, ist **Digest-Authentisierung** nach RFC 2617 [RFC2617] (implementiert im Apache-Modul `mod_auth_digest`). Digest-Authentisierung wird nicht von jedem Browser unterstützt. Auch werden dabei zwar die Kennwörter nicht in die Welt trompetet, die geschützten Inhalte gehen aber nach wie vor im Klartext über das Netz.

Probleme

Digest-Authentisierung

Digest-Authentisierung wird im wesentlichen genauso konfiguriert wie Basic-Authentisierung. Statt `htpasswd` verwenden Sie das Programm `htdigest` (mit ähnlicher Syntax), um eine Kennwortdatei anzulegen; die Apache-Konfiguration benutzt »AuthType Digest« statt »AuthType Basic«, und die Informationen über Benutzer und ihre Kennwörter besorgt Apache sich aus einer Quelle, die Sie mit der Direktive `AuthDigestProvider` bestimmen können.

Viele Benutzer
AuthDigestProvider
Ist der Kreis der zu authentisierenden Benutzer groß, kann es Probleme mit den Kennwort- und Gruppendateien geben: Die Suche nach dem passenden Eintrag in diesen Dateien dauert länger, als der Apache-Server auf »inaktive« Verbindungen zu warten bereit ist. Als Abhilfe können Sie die Benutzerinformationen in einer Datenbank ablegen. Im einfachsten Fall kommen dafür `dbm`- oder Berkeley-DB-Dateien in Frage, aber es ist (mit zusätzlich installierter Software) auch möglich, zum Beispiel eine SQL-Datenbank oder ein LDAP-basiertes Verzeichnis dafür heranzuziehen.

Datenbank
LDAP
Satisfy
💡 Wird eine Ressource sowohl über TCP-basierte Zugriffskontrolle als auch über HTTP-Authentisierung geschützt, so werden die beiden Mechanismen implizit UND-verknüpft, das heißt, eine Anfrage muss erst auf der Basis der Zugriffskontrolle akzeptiert werden, bevor geprüft wird, ob der Benutzer autorisiert ist. Mit der Direktive »`Satisfy any`« ist statt dessen eine ODER-Verknüpfung möglich. Damit können Sie zum Beispiel Anfragen zulassen, die entweder aus dem Intranet kommen oder von einem akkreditierten Benutzer aus dem Internet gestellt werden. (Das Gegenteil ist »`Satisfy all`«.)

💡 Ab Apache 2.4 bieten die weiter vorne diskutierten Blöcke für Authentisierungsbedingungen (Require) mehr Flexibilität.

Übungen

- 📝 5.4 [!2] Legen Sie mit `htpasswd` eine Kennwortdatei `~/public_html/secret/.htpasswd` an, die Einträge für die Benutzer `lotte` und `hugo` enthält.
- 📝 5.5 [!2] (Fortsetzung der vorigen Übung.) Erstellen Sie eine Apache-Konfiguration, die den Zugriff auf die Ressourcen im Verzeichnis `~/public_html/secret` nur (a) den in der Datei `.htpasswd` eingetragenen Benutzern; (b) der Benutzerin `lotte` gestattet.
- 📝 5.6 [3] Konfigurieren Sie den Zugriff auf `secret` so, dass alle Zugriffe über `http://localhost/~hugo/secret/` automatisch zugelassen werden; bei allen Zugriffen von anderen Rechnern aus soll ein Benutzername nebst Kennwort abgefragt werden.
- 📝 5.7 [5] (Benötigt besondere Kenntnisse.) Verwenden Sie `mod_authn_dbd` oder eins der MySQL-basierten Module (`mod_auth_mysql` oder ähnliches), um den Zugriff auf ein Verzeichnis `sqlsecret` gegen eine SQL-Datenbank zu authentisieren.

5.4 Apache und SSL

Secure Socket Layer
SSL oder **Secure Socket Layer** ist ein allgemeines Verfahren zum Verschlüsseln von TCP-Verbindungen und zur Authentisierung der Endpunkte. Mit HTTP-Verbindungen über SSL können Sie sicherstellen, dass Client und Server die sind, die sie zu sein vorgeben, und Sie können die Inhalte der Kommunikation (Kennwörter und Nutzdaten) davor schützen, dass Unbefugte sie anschauen oder gar verändern können. SSL verwendet dazu diverse kryptografische Verfahren.

Mehr über Apache und SSL erfahren Sie in Kapitel 10.

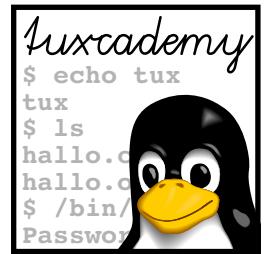
Zusammenfassung

- Die Direktiven `Allow`, `Deny` und `Order` steuern die Zugriffskontrolle auf TCP-Ebene.
- Mit Basic-Authentisierung können Ressourcen über Benutzernamen und Kennwörter geschützt werden.
- Digest-Authentisierung vermeidet den Versand von Kennwörtern im Klartext über das Netz.
- Sowohl Basic- als auch Digest-Authentisierung verhindern nicht, dass die geschützten Ressourcen unverschlüsselt übertragen werden.
- Die Authentisierungsdaten können statt in Textdateien auch in Datenbankdateien, SQL-Datenbanken, LDAP-Verzeichnissen o. ä. gespeichert werden.
- Secure Socket Layer (SSL) ermöglicht die authentisierte und verschlüsselte Übertragung von Ressourcen.

Literaturverzeichnis

RFC2617 J. Franks, P. Hallam-Baker, J. Hostetler, et al. »HTTP Authentication: Basic and Digest Access Authentication«, Juni 1999.

<http://www.ietf.org/rfc/rfc2617.txt>



6

Protokolldateien

Inhalt

6.1	Überblick	72
6.2	Standardformate für Protokolldateien	72
6.3	Selbstdefinierte Formate	73
6.4	Fehlermeldungen	75
6.5	Verwaltung von Protokolldateien	76
6.6	Automatisierte Auswertung von Protokolldateien	78

Lernziele

- Apache-Protokolldateien interpretieren können
- Apache-Protokolldateien konfigurieren können
- Protokolldateien in eigenen Formaten erstellen
- Die Programme *The Webalizer* und *Analog* kennen

Vorkenntnisse

- Überblick über Apache-Konfiguration (Kapitel 3)
- TCP/IP-Kenntnisse

6.1 Überblick

Wie bereits in Abschnitt 2.6 kurz umrissen, vermerkt der Apache-Server Zugriffe und Fehlermeldungen in verschiedenen Protokolldateien (engl. *log files*). Standardmäßig gibt es zumindest eine Datei namens `access_log`, die Details über die vom Server bearbeiteten Zugriffe enthält, und eine Datei `error_log`, die etwaige Fehlermeldungen sowohl des Apache-Servers selbst als auch von vom Apache-Server über CGI aufgerufenen Programmen aufführt. Unter Unix bzw. Linux ist es außerdem möglich, Protokolldaten direkt an ein anderes Programm weiterzurichten.

Auch das Format der Protokolldateien ist konfigurierbar. Damit ist es möglich, eigene zusätzliche Protokolldateien anzulegen, die bestimmte Daten erheben.

6.2 Standardformate für Protokolldateien

Frühere Versionen des Apache-Servers unterstützten eine ganze Reihe von Direktiven zur Definition von Protokolldateien und ihren Formaten. Inzwischen wurden alle diese Direktiven von der `CustomLog`-Direktive abgelöst, die ihre Funktionen in sich vereint. `CustomLog` verwendet eine weitere Direktive, `LogFormat`, mit der das Format von Protokolleinträgen definiert wird. `LogFormat` übernimmt als Formattangabe eine Zeichenkette mit speziellen Einträgen (meist »%<Buchstabe>«), die dann durch die zu protokollierenden Angaben ersetzt werden.

Ein weit verbreitetes Format für die `access_log`-Datei ist das sogenannte *Common Log Format* mit Zeilen der Form

```
127.0.0.1 - hugo [21/Jun/2002:11:05:03 +0200] "GET /bla.html HTTP/1.0" 200 1345
```

»127.0.0.1« ist die IP-Adresse, von der die Anfrage kam. Theoretisch können Sie statt der rohen IP-Adresse auch den dazugehörigen Rechnernamen protokollieren (indem Sie die Direktive `HostNameLookups` auf `on` stellen), aber das kann den Server sehr verlangsamen. Es ist besser, IP-Adressen zu protokollieren und die Rechnernamen gegebenenfalls später nachzuschlagen (die Apache-Distribution enthält dafür ein Programm namens `logresolve`).

Das Minuszeichen (»-«) in der Logdatei ist ein Indiz dafür, dass die betreffende Information nicht zur Verfügung stand. Im besonderen steht es hier für den Benutzernamen des Anfragestellers auf seinem eigenen Rechner, der theoretisch über das IDENT-Protokoll (RFC 1413 [RFC1413]) herauszufinden sein müsste. Diese Information ist aber leicht zu fälschen und extrem wenig vertrauenswürdig, so dass der Apache-Server sich normalerweise nicht einmal die Mühe macht, sie überhaupt zu erheben. (Wenn Sie sie wirklich haben wollen, können Sie `IdentityCheck` auf `on` setzen. Allerdings setzt das voraus, dass auf den Client-Rechnern ein entsprechender Dienst angeboten wird. Eigentlich hat das nur in Intranets wirklich Zweck.)

»hugo« steht für den Benutzernamen des Anfragestellers gemäß der HTTP-Authentisierung (Kapitel 5). Ist die angefragte Ressource nicht über HTTP-Authentisierung gesichert, dann erscheint auch hier ein »-«. Ist der HTTP-Statuscode (siehe unten) 401, dann hat noch keine Authentisierung für die Ressource stattgefunden, und der Wert ist nicht verlässlich.

Die Zeitangabe »[21/Jun/2002:11:05:03 +0200]« gibt den Zeitpunkt an, an dem der Server mit der Bearbeitung der Anfrage fertig geworden ist. Ihr Format ist offensichtlich.

»"GET /bla.html HTTP/1.0"« ist die tatsächliche Anfrage, inklusive der verwendeten Protokollversion. »200« ist der HTTP-Statuscode, den der Server auf die Anfrage hin zurückliefert und der zum Beispiel angibt, ob die Anfrage erfolgreich war (»2« am Anfang) oder wegen eines Fehlers im Browser (»4« am Anfang) oder im Server (»5« am Anfang) fehlschlug. Eine vollständige Liste aller Statuscodes steht im Anhang B.

Der letzte Eintrag, »1345«, gibt die Größe der an den Browser zurückgeschickten Daten an, ausschließlich der Kopfzeilen. Wurden keine Daten zurückgeschickt, erscheint »-«.

Ein anderes häufig verwendetes Protokollformat, das *Combined Log Format*, fügt dem *Common Log Format* noch zwei Felder hinzu:

```
127.0.0.1 - hugo [21/Jun/2002:11:05:03 +0200] "GET /bla.html HTTP/1.0" 200 1345 "http://www.example.com/test.html" Mozilla/4.08 [en] (Linux; I; Nav)
```

(in Wirklichkeit wird wie zuvor pro Anfrage eine Zeile protokolliert, die aus Platzgründen jedoch hier umbrochen werden musste).

Das erste der neuen Felder, »"http://www.example.com/test.html"«, gibt den Inhalt der HTTP-Kopfzeile Referer wieder, also den URI der Seite, von der aus der Benutzer den Zugriff auf die aktuelle Ressource veranlasst hat. Das zweite neue Feld identifiziert den vom Benutzer verwendeten Browser, entspricht also der HTTP-Kopfzeile User-Agent.

Übungen

 **6.1** [1] Beobachten Sie das Ende der `access.log`-Datei – etwa mit »`tail -f`« –, während Sie einige Anfragen an Ihren Apache-Server stellen.

 **6.2** [3] Warum ist es interessant, die Referer-Zeile zu protokollieren?

6.3 Selbstdefinierte Formate

Mit der `LogFormat`-Direktive können Sie eigene Formate für Protokolldateien festlegen. Hier ist die Definition des *Common Log Format*:

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
```

Das erste Argument bestimmt das Format einer Zeile der Protokolldatei, während das zweite Argument diesem Format einen Namen gibt. Auf diesen Namen können Sie dann bei der Definition einer konkreten Protokolldatei Bezug nehmen:

```
CustomLog logs/access_log common
```

definiert, dass die Protokolldatei `access_log` im *Common Log Format* (wie eben definiert) geschrieben werden soll. (Das Verzeichnis `logs` liegt dabei im `ServerRoot`-Verzeichnis).

Wesentlich bei der Definition von Protokollformaten sind die »%«-Sequenzen, die durch die relevanten Informationen ersetzt werden. Tabelle 6.1 enthält die möglichen »%«-Sequenzen.

Es ist auch möglich, bei der `CustomLog`-Direktive direkt eine Formatangabe statt eines Formatnamens anzugeben. Zusätzliche Protokolldateien definieren Sie einfach über weitere `CustomLog`-Direktiven:

```
CustomLog logs/referer_log "%{Referer}i -> %U"
CustomLog logs/agent_log "%{User-Agent}i"
```

Der Apache-Server kann seine Protokollinformationen statt in eine Datei auch in eine Pipe zu einem anderen Programm schreiben. Dazu geben Sie in der `CustomLog`-Direktive statt eines Dateinamens ein Pipe-Symbol (»|«) gefolgt von einem Shellkommando an:

```
CustomLog "|/usr/bin/gzip -c >>/var/log/access_log.gz" common
```

Tabelle 6.1: »%«-Sequenzen für Protokollformate

Sequenz	Bedeutung
%a	IP-Adresse des Anfragestellers
%A	IP-Adresse des Servers
%B	Verschickte Bytes, außer HTTP-Kopf; keine Daten = »0«
%b	Verschickte Bytes, außer HTTP-Kopf; keine Daten = »-«
%c	Status der Verbindung am Ende der Übertragung; »X«: Verbindung wurde vorzeitig abgebrochen, »+«: Verbindung kann offenbleiben, »-«: Verbindung wird geschlossen
{BLA}e	Inhalt der Umgebungsvariablen BLA
%f	Dateiname der angeforderten Ressource
%h	Rechnername des Anfragestellers
%H	Protokoll der Anfrage
{Bla}i	Inhalt der HTTP-Kopfzeile Bla der Anfrage
%l	Entfernter Benutzername (via IDENT), falls vorhanden
%m	Methode der Anfrage
{Bla}n	Inhalt der Notiz Bla aus einem anderen Modul
{Bla}o	Inhalt der HTTP-Kopfzeile Bla der Antwort
%p	Kanonischer TCP-Port des antwortenden Servers
%P	Prozess-ID des Kindprozesses, der die Anfrage beantwortet
%q	Der <i>query string</i> , mit »?« am Anfang; leer, falls nicht vorhanden
%r	Erste Zeile der Anfrage (Methode, Pfad, Protokoll)
%s	Status der ursprünglichen Anfrage
%>s	Status der letzten Anfrage (bei interner Umleitung)
%t	Zeit (im <i>Common-Log-File-Format</i>)
{Format}t	Zeit im angegebenen <i>Format</i> , à la <code>strftime(3)</code>
%T	Zeit, die für die Beantwortung der Anfrage nötig war (in Sekunden)
%u	Benutzername (via HTTP-Authentisierung), möglicherweise fragwürdig, wenn %s 401
%U	Angefragter Pfad (ohne <i>query string</i>)
%v	Kanonischer Servername des antwortenden Servers
%V	Servername gemäß <code>UseCanonicalName</code> -Einstellung

Zwischen % und der eigentlichen Sequenz kann noch eine »Bedingung« stehen. Dies ist eine Liste von HTTP-Statuscodes, vor der ein »!« stehen kann. Zum Beispiel protokolliert »%400,501{User-Agent}i« den Namen des Browsers nur, wenn die Fehler 400 und 501 (»Bad Request« und »Not Implemented«) aufgetreten sind; »%!200,302,304{Referer}i« protokolliert nur, wenn kein halbwegs normaler Status zurückgegeben wird.

Die Zeichenketten, die durch %i, %o und %r im Protokoll erzeugt werden, werden direkt so geschrieben, wie sie in der HTTP-Anfrage stehen. Das heißt, dass möglicherweise »gefährliche« Steuerzeichen in der Protokolldatei stehen, die darum mit der gebotenen Vorsicht zu behandeln ist.

würde zum Beispiel den Inhalt der Protokolldatei sofort komprimieren. Eine Zeile wie

```
CustomLog "|/usr/local/bin/logresolve >/var/log/access_log" common
```

löst IP-Adressen im Protokoll in die zugehörigen Rechnernamen auf, ohne den Apache-Server selbst damit zu belasten.

Wichtig ist, dass die Programme von Apache möglicherweise mit root-Rechten gestartet werden. Es sollte sich dabei also um einfache und sichere Programme handeln.

Die TransferLog-Direktive ist eine vereinfachte Form von CustomLog, die keine direkte Formatangabe zulässt. Das Format einer mit TransferLog geöffneten Datei oder Pipe ist immer das der letzten mit nur einem Argument aufgerufenen LogFormat-Direktive.

```
LogFormat "%{Referer}i -> %U"
TransferLog logs/referer
```

Übungen

 **6.3** [!2] Bringen Sie Ihren Apache dazu, eine Protokolldatei namens `ua.log` zu schreiben, die Zeilen der Form

```
<IP-Adresse> [<Datum und Uhrzeit>] => <User-Agent-Kopfzeile>
```

enthält.

 **6.4** [4] Sorgen Sie dafür, dass jedesmal, wenn Ihr Apache den Fehlercode 500 (engl. *Internal Server Error*) protokolliert, eine E-Mail-Nachricht an `webmaster@example.com` geschickt wird, die die betreffende Anfrage (Methode, URL-Pfad, HTTP-Version) enthält.

6.4 Fehlermeldungen

Fehlermeldungen protokolliert Apache in eine weitere Datei, deren Namen Sie über die ErrorLog-Direktive einstellen können (üblich ist `error_log`). Das Format der ErrorLog Einträge in diese Datei lässt sich nicht beeinflussen; sie sehen meist aus wie

```
[Fri Jun 21 12:18:32 2002] [error] [client 127.0.0.1]
File does not exist: /var/www/bla.html
```

(in einer Zeile). Das heißt, die Zeile beginnt mit Datum und Uhrzeit, gefolgt von der Schwere des Fehlers (außer `error` gibt es unter anderem noch `notice` oder `alert`, siehe Tabelle 6.2) und (gegebenenfalls) der IP-Adresse des Browsers, der für die Fehlermeldung »verantwortlich« ist. Anschließend folgt die eigentliche Fehlermeldung.

Mit der LogLevel-Direktive können Sie festlegen, welche Arten von Apache-Meldungen tatsächlich ins ErrorLog geschrieben werden. Das Argument von LogLevel ist eine Fehlerstufe aus Tabelle 6.2; es werden alle Meldungen der betreffenden Stufe und darüber protokolliert, für »LogLevel info« also auch Meldungen der Stufen `notice`, `warn` und so weiter.

Für jede Fehlermeldung gibt es einen korrespondierenden Eintrag im `access_log`; Sie können also den üblichen Protokolldateimechanismus heranziehen, um sich mehr Informationen über Fehlersituationen zu verschaffen.

Mit der ErrorLog-Direktive können Sie Fehlermeldungen auch an den Unix-syslog-Mechanismus übergeben, indem Sie statt eines Dateinamens das Schlüsselwort »`syslog`« angeben. Normalerweise benutzt Apache die *facility* »`local7`«, aber auch das lässt sich ändern: Mit

Tabelle 6.2: Fehlerstufen für `error_log`

Stufe	Erklärung/Beispiel
emerg	Notfall – System ist unbenutzbar <i>Child cannot open log file. Exiting</i>
alert	Sofortiger Eingriff ist notwendig <i>getpwid: couldn't determine user name from uid</i>
crit	Kritisches Problem <i>socket: Failed to get a socket, exiting child</i>
error	Fehler <i>Premature end of script headers</i>
warn	Warnung <i>child process 1234 did not exit, sending another SIGHUP</i>
notice	Normaler, aber bedeutsamer Umstand <i>httpd: caught SIGBUS, attempting to dump core in ...</i>
info	Interessante Information <i>Server seems busy (you may need to increase StartServer, or Min/MaxSpareServers)</i>
debug	Informationen für die Fehlersuche <i>Opening config file ...</i>

Die Fehlerstufen sind in absteigender Priorität aufgeführt.

ErrorLog syslog:local0

würden die Apache-Fehlermeldungen zum Beispiel in `local0` protokolliert. (Dem gewieften Systemadministrator ist die Ähnlichkeit der `LogLevel`-Werte zu den `syslog`-Prioritäten sicher längst aufgefallen.)

Übungen



6.5 [!2] Beobachten Sie die Datei `error_log`, während Sie versuchen, diverse Fehlermeldungen zu provozieren.

6.5 Verwaltung von Protokolldateien

Die Protokolldateien eines halbwegs beschäftigten Apache-Servers können ziemlich schnell wachsen. Es empfiehlt sich darum sehr, einen automatischen Mechanismus zu installieren, der die Protokolldateien beobachtet und gegebenenfalls kürzt oder löscht. Am besten ist es natürlich, wenn in periodischen Abständen (etwa einmal in der Woche) eine neue Protokolldatei angefangen wird. Die alte Protokolldatei können Sie dann komprimieren und archivieren oder gegebenenfalls löschen.



Datenschutz **Der Umgang mit Protokolldateien hat nicht nur technische, sondern auch rechtliche Konsequenzen.** Nach der einschlägigen Gesetzgebung ist es nicht zulässig, personenbezogene Daten ohne Grund oder beliebig lange zu speichern. Wenn Sie also Informationen protokollieren, die einzelnen Benutzern zugeordnet werden können (und wenn Sie Ressourcen anbieten, die über Basic-Authentisierung zugänglich sind, dann enthalten die Protokolldateien die betreffenden Benutzernamen), dann könnte es sein, dass Sie Ihre Protokolldateien nur eine angemessene Zeit zur Fehlersuche aufbewahren dürfen. Eine ausführliche Würdigung der entsprechenden Gesetzeslage ist an dieser Stelle nicht qualifiziert möglich; bitte konsultieren Sie im Zweifel einen Rechtsbeistand Ihres Vertrauens

```
/var/log/apache/*_log {
    weekly
    missingok
    rotate 52
    compress
    delaycompress
    notifempty
    create 640 root adm
    sharedscripts
    postrotate
        /usr/sbin/apachectl graceful > /dev/null
    endscript
}
```

Bild 6.1: logrotate-Konfiguration für Apache-Protokolldateien

Das gängige Paket zur Verwaltung von Protokolldateien ist `logrotate`, das die meisten Linux-Distributionen heutzutage einsetzen. Mit `logrotate` können Sie detailliert angeben, was mit den verschiedenen Protokolldateien im System passieren soll.

Eine mögliche `logrotate`-Konfiguration für den Apache-Server zeigt Bild 6.1: Die Protokolldateien in `/var/log/apache` werden wöchentlich angeschaut (»`weekly`«); es macht nichts, wenn dort gar keine Protokolldateien zu finden sind (»`missingok`«); maximal 52 (wöchentliche) alte Protokolldateien werden aufgehoben (»`rotate 52`«); alte Protokolldateien werden mit `gzip` komprimiert (»`compress`«), aber nicht gleich (»`delaycompress`«); wenn die Protokolldatei leer ist, passiert nichts (»`notifempty`«); neue Versionen der Protokolldatei werden mit dem Eigentümer `root`, der Gruppe `adm`, Lese- und Schreibrecht für den Eigentümer und Leserecht für die Gruppe angelegt (»`create ...`«); und nach dem Rotieren wird der Apache-Server »sanft« neu gestartet (Kommando zwischen »`postrotate`« und »`endscript`«), und zwar nur einmal, egal wie viele Logdateien angeschaut wurden (»`sharedscripts`«).

Die Option `delaycompress` verdient noch eine genauere Erklärung. Wenn Apache mit »`apachectl graceful`« dazu gebracht wird, seine Konfiguration neu einzulesen, dann bearbeitet er gerade ausstehende Anfragen zu Ende. Da die Protokolleinträge aber am Ende der Anfragenbearbeitung gemacht werden, könnte es sein, dass diese ans Ende der alten Protokolldatei geschrieben werden, obwohl `logrotate` diese schon rotiert und komprimiert hat – ein kleines Chaos wäre die Folge. Aus diesem Grund bewirkt `delaycompress`, dass beim Anlegen einer neuen Protokolldatei die unmittelbar vorhergehende Protokolldatei nicht gleich komprimiert wird, sondern erst bei der nächsten Rotation. Auf diese Weise kann der Apache-Server dort noch ein paar Einträge anhängen, selbst wenn die neue Protokolldatei schon angelegt wurde.



Wenn der Apache-Server die Protokolldatei einmal geöffnet hat, hält er sie offen, auch wenn sie im Dateisystem umbenannt oder gelöscht wird (das ist unter Unix/Linux normal so). Erst mit einem »`apachectl restart`« oder »`graceful`« wird die alte Protokolldatei geschlossen und die neue geöffnet (bei »`graceful`« gegebenenfalls mit Verzögerung).

Übungen



6.6 [1] Welche Vorkehrungen trifft Ihre Distribution, um die Protokolldateien des Apache-Servers zu verwalten?

6.6 Automatisierte Auswertung von Protokolldateien

Das Erheben von Protokollinformationen hat nur dann Sinn, wenn Sie auch etwas mit ihnen anstellen. Beispielsweise könnten Sie sich dafür interessieren, ob überhaupt jemand auf den Web-Server zugreift, welche Seiten am beliebtesten sind, woher Anfragen kommen und vieles andere mehr. Solche Informationen dienen oft auch nicht nur der Befriedigung der eigenen Neugierde, sondern auch handfesten kommerziellen Interessen: Eine vielbesuchte Web-Präsenz ist zum Beispiel für »Werbekunden« viel interessanter als ein obskurer Server, den nie jemand besucht.

Zugriffe vs. Popularität

Bevor Sie mit der Anzahl der Zugriffe protzen, die der eigene Server bearbeitet, sollten Sie jedoch ein bisschen überlegen. Die Anzahl der reinen Zugriffe sagt nämlich nicht allzuviel über die Popularität des Servers aus – es könnte sein, dass eine reine Textseite von viel mehr verschiedenen Benutzern angesteuert wurde als eine sehr graphiklastige Seite mit Hunderten von kleinen Bildchen, während die Zugriffszahl sehr eindeutig zugunsten der Graphikseite ausfällt. Einige Begriffsklärungen sind darum sinnvoll:

Zugriffe (engl. *hits*) Abrufe von Ressourcen egal welches Datentyps: HTML-Seiten, Bilder, ... Bei Web-Präsenzen mit viel Graphik ist die Anzahl der Zugriffe meistens wild überhöht und irreführend.

Seitenzugriffe (engl. *page views*) Abrufe von »inhaltstragenden« Ressourcen wie HTML-Seiten oder CGI-Skripten. Bilder, die »nur« Navigationselemente oder fest in Seiten enthaltene Illustrationen darstellen, zählen nicht mit. Die Anzahl der Seitenzugriffe ist wesentlich aussagekräftiger für den tatsächlichen Verkehr auf einer Seite als die Anzahl der Zugriffe unabhängig vom Datentyp.

Besuche (engl. *visits*) Da HTTP kein An- und Abmelden kennt, ist es sehr schwer, festzulegen, wie lange ein Benutzer tatsächlich auf einer Web-Präsenz »verweilt«. Ein »Besuch« ist darum so definiert, dass bei jedem (Seiten-)Zugriff auf die Präsenz gewissermaßen eine »Stoppuhr« gestartet wird. Erfolgt innerhalb von 30 Minuten ein neuer Seitenzugriff, so läuft die Uhr wieder von vorne los, und beide Seitenzugriffe gelten als Teil desselben Besuchs. (Wenn der Benutzer in den ersten 5 Minuten ein sehr langes Dokument abruft, darin eine gute halbe Stunde liest und sich direkt anschließend ein weiteres sehr langes Dokument von derselben Web-Präsenz holt, gilt das als zwei Besuche. Auf der anderen Seite zählt es nur einfach, wenn ein Benutzer mal fünf Minuten lang herumstöbert, sich dann zwanzig Minuten lang überall sonst im Web umschaut und dann nochmal für zwei Minuten wiederkommt. Auch Proxy-Server bringen die Statistik durcheinander. Die Disraeli'sche Sentenz über »Lügen, verdammte Lügen und Statistiken« lässt grüßen ...)

Es gibt Dutzende von Programmen, die Apache-Protokolldateien auswerten können (siehe [ALA]). Im folgenden werden zwei der beliebtesten, *The Webalizer* und analog, kurz vorgestellt.

Statistiken

The Webalizer wird von Bradford L. Barrett entwickelt und unter der GPL vertrieben [Webalizer]. Es liest Protokolldateien in diversen Formaten (darunter das *common* und das *combined log format*) und generiert Statistiken in Form von HTML-Seiten. Die Statistiken erscheinen sowohl in tabellarischer als auch graphischer Form. Im einzelnen gibt es die folgenden Statistiken:

Zugriffe, Seitenzugriffe und Besuche wie oben besprochen.

Dateizugriffe (engl. *pages*) sind Zugriffe, bei denen Nutzdaten als Antworten zurückgeschickt wurden (und nicht nur Fehlermeldungen oder Weiterleitungsnachrichten).

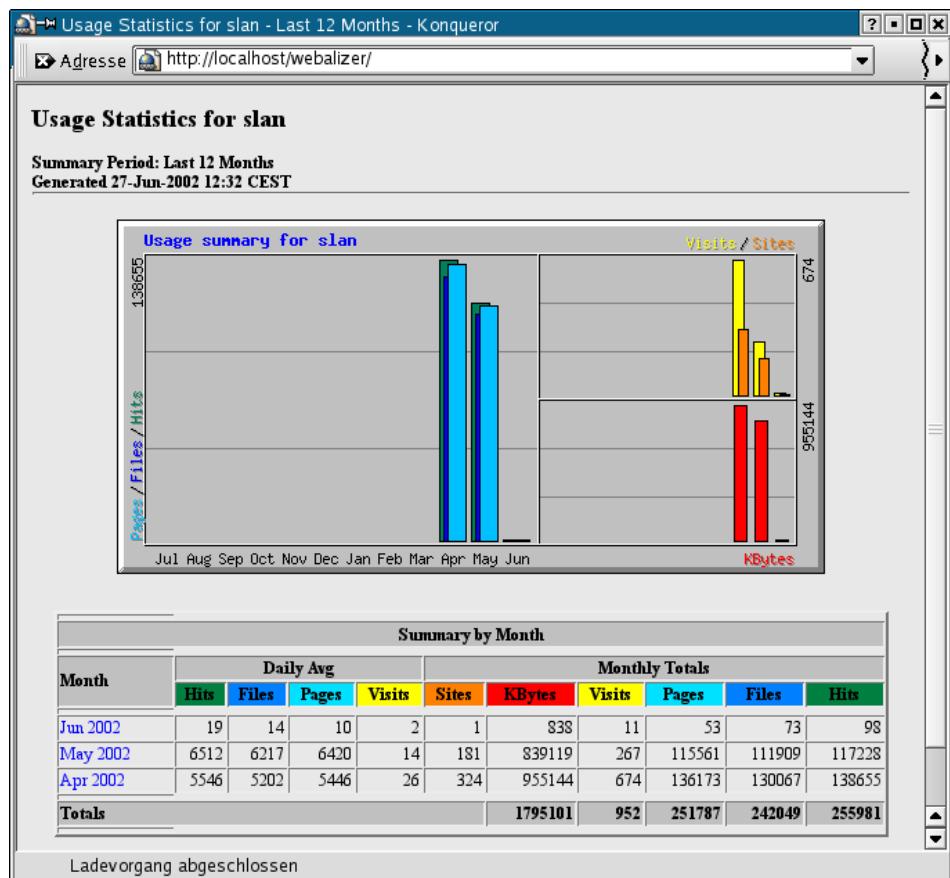


Bild 6.2: Jahresstatistik von *The Webalizer*

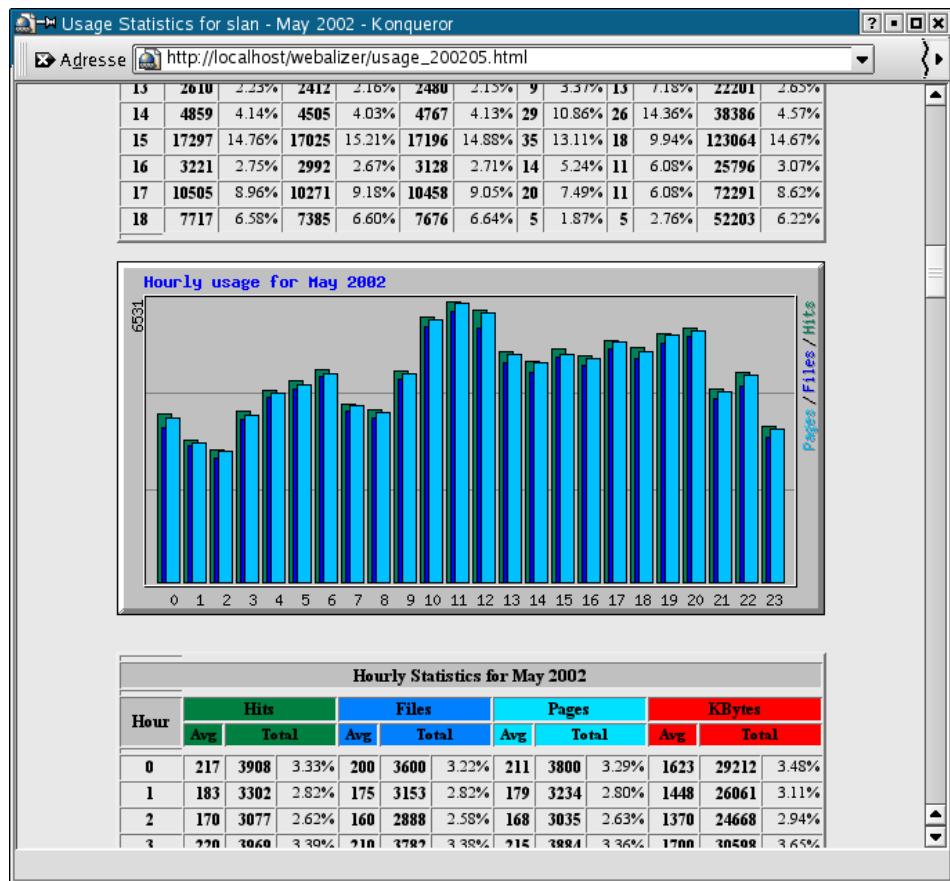


Bild 6.3: Monatsstatistik von *The Webalizer* (Ausschnitt)

Rechner (engl. sites) sind die verschiedenen IP-Adressen, von denen aus zugegriffen wurde. Dies ist die beste Näherung an die Anzahl der tatsächlichen verschiedenen Besucher, die auf der Basis von HTTP möglich ist.

Daten (in KiB), die der Server verschickt hat.

Beliebteste Ein- und Austrittsseiten auf der Basis der Besuche.

Herkunftsadressen und -länder der Anfragen.

Herkunftsseiten der Anfragen, auf der Basis der Referer-Kopfzeilen der Anfragen

Suchbegriffe der gängigsten WWW-Suchmaschinen, über die Benutzer die Seiten gefunden haben, ebenfalls auf der Basis der Referer

Verwendete Browser auf der Basis der User-Agent-Kopfzeilen der Anfragen

Die Statistiken über Zugriffe, Seitenzugriffe, Besuche, Dateizugriffe, Anzahl der verschiedenen Rechner und den ausgehenden Datenverkehr werden monatlich und tageweise zusammengestellt, die ersten drei außerdem stündlich in dem Sinne, dass für einen ganzen Monat berechnet wird, wie viele Zugriffe täglich z. B. zwischen 10 und 11 Uhr morgens erfolgten (Bild 6.2 bzw. 6.3). Damit Sie nicht Protokolldateien im Werte von Jahren aufheben müssen, kann die Statistik auch *inkrementell* erstellt werden; in diesem Fall speichert das Programm frühere Zwischenergebnisse ab und benutzt sie weiter.

Die Ausgabe von *The Webalizer* lässt sich sehr umfangreich konfigurieren. Normalerweise liest das Programm eine Konfigurationsdatei (typischerweise `/etc/webalizer.conf`), die die Namen der zu betrachtenden Protokolldatei sowie eines

Verzeichnisses enthält, in dem die HTML-Seiten und Zwischenergebnisse abgelegt werden. Daneben können Sie die HTML-Ausgabe anders gestalten, verschiedene Statistiken einschalten oder unterdrücken oder genau einstellen, wie sie berechnet werden (beispielsweise können Sie angeben, welche Dateiendungen als »Seiten« im Sinne der Statistik gelten sollen). Sie können auch bestimmte Rechner aus der Statistik herausnehmen, zum Beispiel den Rechner, auf dem der Server selber läuft, oder für die Zwecke der Summenbildung zusammenfassen (etwa alle Rechner, deren Namen auf `.aol.com` enden – AOL betreibt einen großen Zoo von Proxy-Servern). Die Konfigurationsdatei ist sehr ausführlich kommentiert.

Aufgerufen werden kann *The Webalizer* über die Kommandozeile. Das Kommando

```
$ webalizer
```

würde die in der Datei `/etc/webalizer.conf` benannte Protokolldatei – meist etwas wie `/var/log/apache/access_log` – lesen und die Statistiken ins ebenfalls in der Konfigurationsdatei festgelegte Verzeichnis – typischerweise `webalizer` unterhalb der `DocumentRoot` – schreiben. (Wenn dieses Verzeichnis tatsächlich unterhalb der `DocumentRoot` steht, wäre eine Zugriffsregel wie

```
<Directory /usr/local/apache/htdocs/webalizer>
  Order Deny,Allow
  Deny from all
  Allow from 127.0.0.1
</Directory>
```

angebracht, um den Zugriff auf die Statistik z. B. auf den lokalen Rechner oder das lokale Netz zu beschränken.) Natürlich können über die Kommandozeile diverse Optionen oder auch eine andere Protokolldatei angegeben werden:

```
$ webalizer -c mywebalize.conf /var/log/apache/myaccess.log
```

würde beispielsweise die Konfigurationsdatei `mywebalize.conf` im aktuellen Verzeichnis lesen und anschließend die Protokolldatei `/var/log/apache/myaccess.log` betrachten. Im »wirklichen Leben« würden Sie das Programm aber zum Beispiel täglich über `cron` aufrufen, um immer die aktuellen Statistiken zur Verfügung zu haben. Der inkrementelle Bearbeitungsmodus ist in diesem Zusammenhang sehr zu empfehlen.

Angeblich »das populärste Protokolldatei-Analyseprogramm der Welt« ist »Analog« von Stephen Turner [Analog]. Es ist ebenfalls frei verfügbar und übertrifft *The Webalizer* noch in der Anzahl der möglichen Statistiken, auch wenn die Präsentation etwas zurückbleibt (aber es gibt ein Zusatzprogramm namens *Report Magic*, das die Analog-Ausgabe optisch aufpeppt). Analog kann nicht nur monatliche, tägliche oder stündliche Statistiken aufstellen, sondern auch jährliche, vierteljährliche, wöchentliche, viertelstündliche und solche für Intervalle von fünf Minuten (!). Auch die Anzahl der verschiedenen Statistiken ist deutlich größer; sie umfasst neben allen Statistiken, die *The Webalizer* beherrscht, noch beispielsweise Aufstellungen der benutzten Betriebssysteme, der verschiedenen Dateiendungen ausgelieferter Dateien, der verschiedenen virtuellen Server und Umleitungen und vieles mehr. (Vorbedingung für viele dieser Statistiken sind entsprechend konfigurierte Protokolldateien.) Einige Statistiken, beispielsweise die Aufstellung der Domains mit den meisten Zugriffen, haben sogar eine hierarchische Struktur, bei der z. B. die fünf Top-Level-Domains mit den meisten Zugriffen und innerhalb dieser fünf wiederum die Unterdomains mit den meisten Zugriffen gezeigt werden.

Optionen

Übungen

 6.7 [!3] Verwenden Sie *The Webalizer*, um eine Zugriffsstatistik für Ihren Apache-Server aufzustellen, und erlauben Sie den Zugriff auf die resultie-

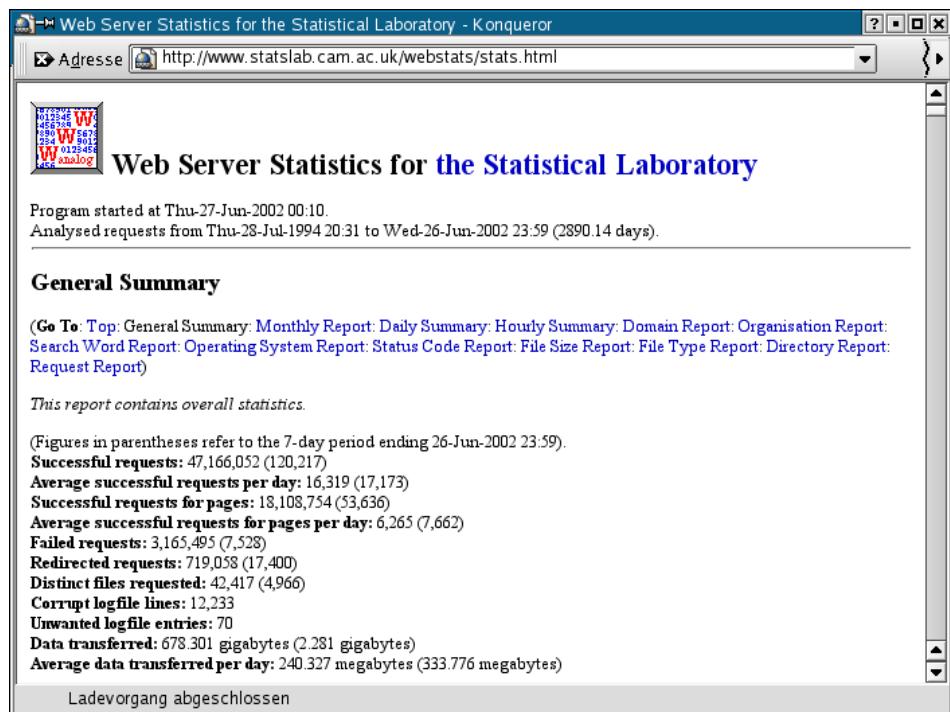


Bild 6.4: Anfang einer Analog-Statistik

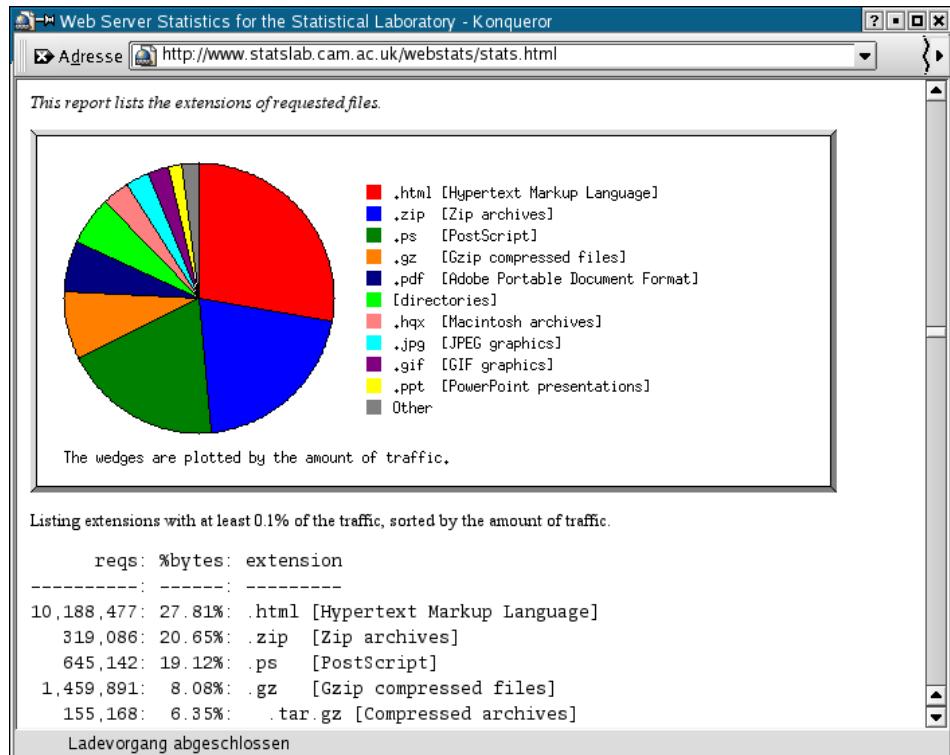


Bild 6.5: Analog-Statistik: Dateitypen

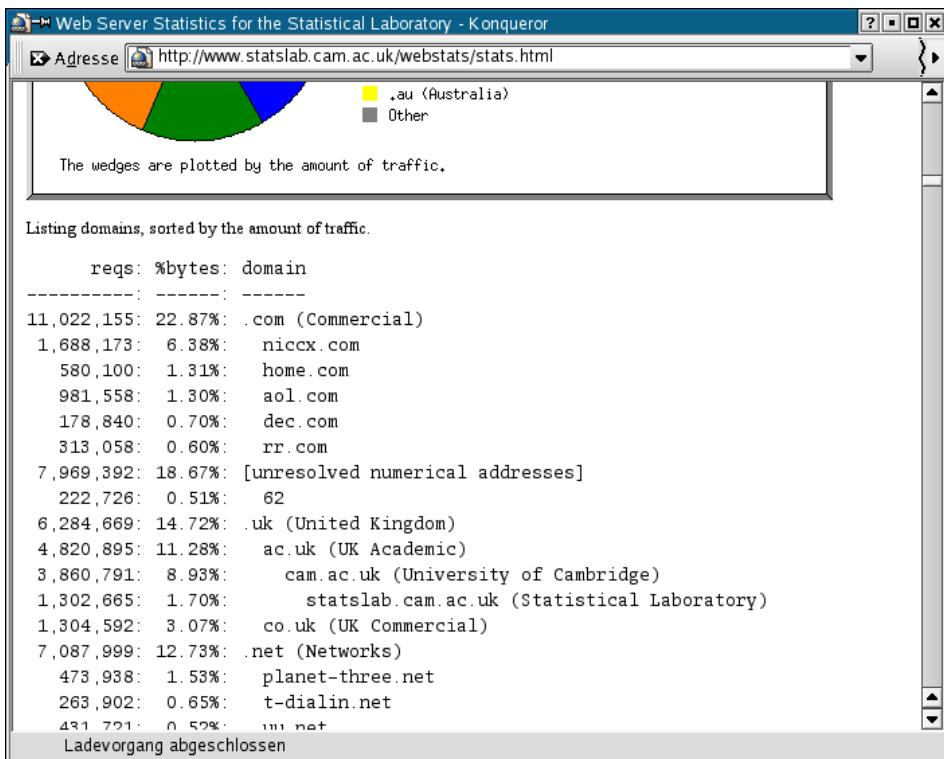


Bild 6.6: Analog-Statistik: Domain-Hierarchie

rende Statistik über Ihren Server für alle Rechner im lokalen Netz.

Zusammenfassung

- Gängige vordefinierte Protokollformate sind das *Common Log Format* und das *Combined Log Format*.
- Mit *CustomLog* und *LogFormat* können Sie das Ausgabeformat für Protokolldateien bestimmen und weitere Protokolldateien definieren.
- Protokolleinträge können nicht nur in Dateien geschrieben, sondern auch direkt an andere Programme weitergereicht werden.
- Fehlermeldungen des Servers und die Ausgabe von CGI-Programmen werden im *error_log* protokolliert.
- Die Direktive *ErrorLog* steuert den Namen dieser Datei, *LogLevel* ihren Inhalt.
- Fehlermeldungen können auch an den *syslog*-Dienst weitergereicht werden.
- Protokolldateien können sehr schnell wachsen. *logrotate* ist ein gutes Werkzeug, um Protokolldateien zu kürzen, zu komprimieren usw.

Literaturverzeichnis

ALA »Access Log Analyzers«. Eine Übersicht über diverse Analyseprogramme für WWW"=Protokolldateien, mit Verweisen.

<http://www.uu.se/Software/Analyzers/Access-analyzers.html>

Analog Stephen Turner. »Analog – The most popular logfile analyser in the world«. Ein freies Analyseprogramm für WWW"=Protokolldateien.

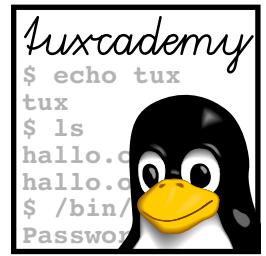
<http://www.analog.cx/>

RFC1413 M. St. Johns. »Identification Protocol«, Februar 1993.

<http://www.ietf.org/rfc/rfc1413.txt>

Webalizer Bradford L. Barrett. »The Webalizer – What is your web server doing today?« Ein freies Analyseprogramm für WWW"=Protokolldateien.

<http://www.webalizer.com/>



7

Virtuelle Web-Server

Inhalt

7.1	Einführung	86
7.2	IP-basierte virtuelle Server	87
7.3	Namensbasierte virtuelle Server	88
7.4	Tips und Tricks	91

Lernziele

- Typen von virtuellen Web-Servern, ihre Einsatzgebiete und Einschränkungen kennen
- IP-basierte virtuelle Web-Server konfigurieren können
- Namensbasierte virtuelle Web-Server konfigurieren können

Vorkenntnisse

- Überblick über Apache-Konfiguration (Kapitel 3)
- TCP/IP-Kenntnisse
- DNS-Überblick

7.1 Einführung

Virtuelle Web-Server sind eine Möglichkeit, mehrere bis viele Web-Präsenzen mit demselben Apache-Server zu bedienen. Das heißt, auf einem Rechner, zum Beispiel `server.example.com`, läuft nur ein Apache-Server, der aber weitgehend unabhängig voneinander diverse virtuelle Web-Server, etwa `www.example.com`, `www.bla.de`, `www.fasel.de` usw. anbietet.

Es gibt zwei verschiedene Möglichkeiten, dies zu organisieren:

IP-basierte virtuelle Server Der Server-Rechner bekommt eine IP-Adresse für jeden virtuellen Server (entweder über separate Netzverbindungen oder über dieselbe Netzverbindung mit IP-Aliasing oder einem ähnlichen Mechanismus). Der Apache-Server lauscht auf jeder dieser IP-Adressen am TCP-Port 80.

Namensbasierte virtuelle Server Alle virtuellen Server teilen sich dieselbe IP-Adresse. Der Apache-Server findet heraus, welcher virtuelle Server gemeint ist, indem er den Inhalt der HTTP-Kopfzeile `Host` betrachtet. (Diese HTTP-Kopfzeile ist Standard in HTTP/1.1, und auch die meisten HTTP/1.0-basierten Browser unterstützen sie als Erweiterung. Nur einige sehr alte Browser können nicht mit namensbasierten virtuellen Servern umgehen.)

Ein dritter Ansatz – »portbasierte« virtuelle Server – ist nicht so interessant, weil er die Anwender dazu zwingt, explizite Portnummern in URLs zu verwenden. Dies kann zu Problemen mit restriktiv konfigurierten anwenderseitigen Firewalls führen. Auf der anderen Seite erfordert er keine besondere DNS-Konfiguration.

Der Vorteil von namensbasierten gegenüber IP-basierten virtuellen Servern besteht vor allem darin, dass es heutzutage schwierig sein kann, jedenfalls zu nicht exorbitanten Preisen genug IP-Adressen für alle gewünschten virtuellen Server zu bekommen. Allerdings funktionieren namensbasierte virtuelle Server prinzipiell nicht mit SSL; das Problem ist, dass für den Aufbau einer verschlüsselten Verbindung das Zertifikat des (virtuellen) Servers notwendig ist, dieser Server aber erst in der HTTP-Kopfzeile `Host` benannt wird – für deren Übertragung die verschlüsselte Verbindung schon stehen muss.

Sie sollten immer namensbasierte virtuelle Server einsetzen anstelle von IP-basierten, wenn kein spezieller Grund dagegen spricht. Der wichtigste Grund, der dagegen sprechen könnte, ist das SSL-Problem.



Erweiterungen in SSL bzw. dem Nachfolgestandard TLS machen es möglich, dass ein Client schon beim Verbindungsaufbau sagen kann, mit welchem virtuellen Webserver er reden möchte. Das Stichwort heißt *Server Name Indication* oder SNI und wird von so gut wie allen aktuellen Browsern und auch Apache unterstützt. Dass SNI sich noch nicht allgemeiner Beliebtheit erfreut, liegt (mal wieder) an Microsoft, namentlich daran, dass Windows XP kein SNI kann und Microsoft auch nicht geruht hat, diese Funktion nachzurüsten (es ist nicht, dass es eine brandneue Idee wäre). Da das eine Restriktion des Betriebssystems ist, hilft es auch nicht, statt Internet Explorer einen aktuellen Google Chrome o. ä. zu benutzen. Und weil noch alle möglichen Leute ihre alten XP-Möhren haben, können Sie es sich noch nicht unbedingt leisten, konsequent auf SNI zu setzen.



Es gibt andere Mittel und Wege, alle irgendwie unbequem, mit denen Sie versuchen können, sich aus der Affäre zu ziehen. Das im Detail zu beleuchten würde hier aber zu weit führen.

Mehrere Apache-Instanzen

Grundsätzlich können Sie mehrere Web-Präsenzen auf demselben Server auch dadurch unterstützen, dass Sie mehr als eine Instanz des Apache-Servers laufen lassen und jeden dieser Server mit unterschiedlichen Werten für `User`, `Group`, `Listen` und `ServerRoot` betreiben. Dies ist allerdings deutlich ineffizienter als die Lösung, die Präsenzen als virtuelle Server zu realisieren. Trotzdem kann es Gründe geben,

die für diesen Ansatz sprechen, etwa wenn Sicherheitserwägungen bestehen, die den Betrieb beider Präsenzen im selben Apache-Server ausschließen.

Virtuelle Server (egal ob namens- oder IP-basiert) werden in der `httpd.conf`-Datei über `<VirtualHost>`-Blöcke konfiguriert. Innerhalb von `<VirtualHost>`-Blöcken können fast alle Apache-Direktiven auftreten; in Anhang C sind die betreffenden Direktiven mit einem »V« kenntlich gemacht.

7.2 IP-basierte virtuelle Server

Angenommen, ein Apache-Server soll die Web-Präsenzen `www.bla.de` und `www.fasel.de` anbieten. Voraussetzung für die Konfiguration der entsprechenden virtuellen Server ist, dass die betreffenden Namen im DNS eingetragen sind und auf die richtigen Adressen weisen, beispielsweise 111.22.33.44 für `www.bla.de` und 111.22.33.55 für `www.fasel.de`. Eine Konfiguration mit IP-basierten virtuellen Servern für diese beiden Präsenzen könnte etwa so aussehen:

```
<VirtualHost 111.22.33.44>
  ServerAdmin webmaster@bla.de
  DocumentRoot /home/bla.de/www
  ServerName www.bla.de
  ErrorLog /home/bla.de/logs/error_log
  CustomLog /home/bla.de/logs/access_log common
</VirtualHost>

<VirtualHost 111.22.33.55>
  ServerAdmin webmaster@fasel.de
  DocumentRoot /home/fasel.de/www
  ServerName www.fasel.de
  ErrorLog /home/fasel.de/logs/error_log
  CustomLog /home/fasel.de/logs/access_log common
</VirtualHost>
```

Wie Sie sehen, werden für die einzelnen virtuellen Server verschiedene `ServerNames`, `ServerAdmins`, `DocumentRoots` und Protokolldateien eingestellt. Für alle praktischen Zwecke handelt es sich um getrennte Web-Server.

Theoretisch ist es erlaubt, in der `<VirtualHost>`-Direktive anstatt auf IP-Adressen auf Rechnernamen zu verweisen, also

Rechnernamen?

```
<VirtualHost www.bla.de>
...
<VirtualHost www.fasel.de>
...
```

Das ist aber nicht empfehlenswert, da Sie sich so diversen DNS-Problemen öffnen. Im einfachsten Fall machen Sie sich davon abhängig, dass DNS läuft, wenn der Apache-Server bootet (denn sonst werden Server, deren Namen nicht aufgelöst werden konnten, nicht oder nicht vollständig initialisiert); im schlimmsten Fall ermöglichen Sie es »Kunden«, die ihren eigenen DNS-Server kontrollieren, Anfragen an andere Web-Seiten zu »stehlen«.

Übungen

 **7.1** [!3] Richten Sie auf Ihrem Rechner einen IP-basierten virtuellen Server ein (verwenden Sie IP-Aliasing über ein Kommando wie »`ifconfig eth0:1 <IP-Adresse> up`«, um die Netzwerkschnittstelle mit einer zweiten IP-Adresse zu versehen). Sie können auf diesen Server über URLs der Form `http://<IP-Adresse>/bla.html` zugreifen; Ihr Trainer wird Ihnen mitteilen, ob es im DNS vordefinierte Namen und Adressen für diese Übung gibt.

7.3 Namensbasierte virtuelle Server

NameVirtualHost

Um namensbasierte virtuelle Server zu verwenden, müssen Sie angeben, auf welcher IP-Adresse Apache auf virtuelle Server lauschen soll. Hierzu dient die NameVirtualHost-Direktive, die eine IP-Adresse (gegebenenfalls mit Port) als Argument übernimmt:

```
NameVirtualHost 111.22.33.44:80
```

Diese Adresse muss eine sein, auf der der Apache-Server bereits lauscht (was Sie gegebenenfalls mit Listen einstellen können). Das spezielle Argument »*« ermöglicht namensbasierte virtuelle Server auf *allen* IP-Adressen des Rechners:

```
NameVirtualHost *
```

Anschließend muss analog zum vorigen Abschnitt ein <VirtualHost>-Block für jeden virtuellen Server eingerichtet werden. Auch hier müssen die Servernamen im DNS enthalten sein (CNAME-Records genügen). Zum Beispiel:

```
NameVirtualHost *

<VirtualHost *>
  ServerName www.bla.de
  DocumentRoot /home/bla.de/www
  ...
</VirtualHost>

<VirtualHost *>
  ServerName www.fasel.de
  DocumentRoot /home/fasel.de/www
  ...
</VirtualHost>
```

(Statt der »*« in den VirtualHost-Blöcken könnten Sie auch jeweils die IP-Adresse aufführen.)

Zusätzliche Namen
ServerAlias

Für Server, die unter mehr als einem Namen gefunden werden sollen, können Sie mit ServerAlias weitere Namen vergeben: Mit

```
ServerAlias bla.de *.bla.de
```

Vorgehensweise

in der Konfiguration des virtuellen Servers für `www.bla.de` beantwortet dieser Server alle Anfragen, die an Rechner in `bla.de` gerichtet sind. Selbstverständlich können Sie nicht beliebig Namen erfinden und in ServerAlias schreiben; sie müssen im DNS eingetragen sein.

Apaches Vorgehensweise bei eingehenden Anfragen ist wie folgt:

1. Zuerst wird geprüft, ob die Kombination aus IP-Adresse und TCP-Port, auf der die Anfrage eingegangen ist, in einer NameVirtualHost-Direktive vorkommt.
2. Wenn ja, dann werden die passenden <VirtualHost>-Blöcke angeschaut, ob in einem davon der ServerName oder ein ServerAlias auf die Host-Kopfzeile in der Anfrage passt. Wenn ja, dann wird die Konfiguration dieses virtuellen Server zur Beantwortung der Anfrage verwendet.
3. Wurde kein passender <VirtualHost>-Block gefunden, benutzt Apache den ersten virtuellen Server in der Konfigurationsdatei, der auf die IP-Adressen/Port-Kombination passt.

Daraus folgt, dass der erste virtuelle Server in der Datei als »Ersatzserver« fungiert. Insbesondere wird die Konfiguration des *Hauptservers* nicht mehr herangezogen, wenn die IP-Adresse und der TCP-Port der Anfrage auf eine *NameVirtualHost*-Direktive passen. Sie müssen den Hauptserver gegebenenfalls auch als virtuellen Server konfigurieren.

Hier ist noch ein Beispiel. Der Server hat eine IP-Adresse, nämlich 111.22.33.44, mit dem Namen `server.example.com`. Außerdem gibt es zwei CNAMEs, `www.bla.de` und `www.fasel.bla.de`, die ebenfalls auf 111.22.33.44 verweisen:

```
Port 80
ServerName server.example.com

NameVirtualHost 111.22.33.44

<VirtualHost 111.22.33.44>
    DocumentRoot /www/bla
    ServerName www.bla.de
    ...
</VirtualHost>

<VirtualHost 111.22.33.44>
    DocumentRoot /www/fasel
    ServerName www.fasel.bla.de
    ...
</VirtualHost>
```

Der »Hauptserver« bedient nur Anfragen an `localhost` (IP-Adresse 127.0.0.1), da Anfragen an 111.22.33.44 von den virtuellen Servern abgedeckt werden. `www.bla.de` ist der »Ersatzserver« für Anfragen, die keinen passende oder keine `Host`-Kopfzeile enthalten.

Im nächsten etwas komplizierteren Szenario hat der Server zwei IP-Adressen, nämlich 111.22.33.44 (für `server1.example.com`) und 111.22.33.55 (für `server2.example.com`). Außerdem gibt es noch `www.example.com`, das ein Aliasname für den Hauptserver ist. Es soll ein virtueller Server für den Aliasnamen `www.bla.de` eingerichtet werden, und ein anderer virtueller Server, `www.sub.example.com` soll alle Anfragen an Servernamen der Form `*.sub.example.com` beantworten. Die virtuellen Server sollen die IP-Adresse 111.22.33.55 benutzen.

Der Hauptserver wird wie folgt konfiguriert:

```
Listen 80
ServerName www.example.com
DocumentRoot /www/example
```

Die virtuellen Server benutzen die andere Adresse:

```
NameVirtualHost 111.22.33.55

<VirtualHost 111.22.33.55>
    ServerName www.bla.de
    DocumentRoot /www/bla
    ...
</VirtualHost>

<VirtualHost 111.22.33.55>
    ServerName www.sub.example.com
    ServerAlias *.sub.example.com
    DocumentRoot /www/subexample
    ...
</VirtualHost>
```

Anfragen an die Adresse 111.22.33.55 gehen an die virtuellen Server, Anfragen an alle anderen Adressen (111.22.33.44 und 127.0.0.1) an den Hauptserver. Anfragen an 111.22.33.55, die keinen oder einen unbekannten Host enthalten, werden von www.bla.de beantwortet.

Zum Schluss: Der Server www.example.com hat zwei IP-Adressen, 192.168.1.1 und 111.22.33.55, und befindet sich zwischen einem Intranet und dem Internet. Auf dem Internet wird der Name www.example.com zu 111.22.33.55 aufgelöst, im Intranet zu 192.168.1.1. Wie das folgende Beispiel zeigt, kann der Server Anfragen aus dem Internet und dem Intranet mit demselben Inhalt beantworten:

```
NameVirtualHost 192.168.1.1
NameVirtualHost 111.22.33.55

<VirtualHost 192.168.1.1 111.22.33.55>
  ServerName www.example.com
  ServerAlias www
  DocumentRoot /www/example
  ...
</VirtualHost>
```

Namens- und IP-basier-
te Server gleichzeitig

Sie können namensbasierte und IP-basierte virtuelle Server auch mischen. Zum Beispiel: Der Server hat drei IP-Adressen, 111.22.33.44, 111.22.33.55 und 111.22.33.66, die respektive zu den Namen server.example.com, www.bla.de und www.fasel.de gehören. Die Adresse 111.22.33.44 wird für namensbasierte virtuelle Server verwendet, die anderen für IP-basierte:

```
Port 80
ServerName server.example.com

NameVirtualHost 111.22.33.44

<VirtualHost 111.22.33.44>
  ServerName www.example.com
  DocumentRoot /www/example
  ...
</VirtualHost>

<VirtualHost 111.22.33.44>
  ServerName www.sub.example.com
  DocumentRoot /www/sub.example
  ...
</VirtualHost>

<VirtualHost 111.22.33.44>
  ServerName www.sub2.example.com
  DocumentRoot /www/sub2.example
  ...
</VirtualHost>

<VirtualHost 111.22.33.55>
  ServerName www.bla.de
  DocumentRoot /www/bla
  ...
</VirtualHost>

<VirtualHost 111.22.33.66>
  ServerName www.fasel.de
  DocumentRoot /www/fasel
  ...
</VirtualHost>
```

```
</VirtualHost>
```

Übungen



7.2 [!3] Richten Sie auf Ihrem Rechner einen namensbasierten virtuellen Server ein, der die Adresse des »Hauptservers« mitbenutzt. (Ihr Trainer wird Ihnen sagen, welchen Namen Sie für den virtuellen Server verwenden können.) Überzeugen Sie sich, dass der Hauptserver über diese Adresse nicht mehr zugänglich ist, und richten Sie ihn als weiteren virtuellen Server ein.

7.4 Tips und Tricks

Sie können mit dem Servernamen »`_default_`« in einer `<VirtualHost>`-Direktive einen virtuellen Server einrichten, der sich um IP-Adressen und Ports kümmert, die von keinem anderen virtuellen Server »beansprucht« werden: Ersatzserver

```
<VirtualHost _default_:*>
  DocumentRoot /www/default
  ...
</VirtualHost>
```

Der `_default_-Server` zusammen mit »`*`« als Port verhindert, dass Anfragen den Hauptserver erreichen.

Ein `_default_-Server` kümmert sich niemals um IP-Adressen/Port-Kombinationen, die von namensbasierten virtuellen Servern bedient werden. Kann für eine Anfrage an eine für namensbasierte virtuelle Server vorgesehene Kombination von IP-Adresse und TCP-Port kein Server identifiziert werden, geht sie, wie oben besprochen, an den ersten konfigurierten virtuellen Server für diese Kombination. Mit `AliasMatch` können Sie beliebige Anfragen, die diesen »Ersatzserver« erreichen, so umschreiben, dass eine Seite mit einer passenden Fehlermeldung zurückgegeben wird.

Wie weiter oben gesehen ist es möglich, jedem virtuellen Server seine eigenen Protokolldateien zu geben. Solange Sie nur ein paar virtuelle Server betreiben, ist das kein Problem; wenn es jedoch viele werden (je nach System ein paar hundert bis ein paar tausend), kann es sein, dass der Apache-Server nicht genug Dateideskriptoren zur Verfügung hat, um alle diese Dateien gleichzeitig offen zu halten. In diesem Fall ist es günstiger, ein Protokollformat zu verwenden wie zum Beispiel Protokolldateien

```
LogFormat "%v %l %u %t \"%r\" %>s %b" vcommon
```

bei dem der virtuelle Server mitprotokolliert wird, der die Anfrage beantwortet hat. Diese Datei können Sie anschließend mit einem anderen Programm in einzelne, serverspezifische Protokolldateien aufteilen. Die Apache-Distribution enthält hierzu ein Perl-Skript namens `split logfile`.

Um eine ganze Reihe von virtuellen Servern einzurichten, die sich in ihrer Konfiguration unterscheiden und wo gelegentlich Server hinzukommen oder entfernt werden, können Sie die Eigenschaft der `Include`-Direktive ausnutzen, dass sie alle Dateien in einem Verzeichnis liest, das als Parameter angegeben wurde. Sie legen ein Verzeichnis an, zum Beispiel `/usr/local/apache/conf/vhosts`, und tut jeweils die Konfiguration für einen virtuellen Server in eine eigene (kleine) Datei in diesem Verzeichnis. In der `httpd.conf`-Datei schreiben Sie dann nur noch

```
# Virtuelle Server konfigurieren (in conf/vhosts/*)
Include conf/vhosts
```

(wenn ServerRoot den Wert /usr/local/apache hat). Für Änderungen genügt es, die passende Datei in conf/vhosts abzulegen oder von dort zu entfernen und den Server (etwa mit apachectl reload) anzustoßen, damit er seine Konfiguration neu einliest.

Massenhosting Echtes »Massenhosting« von virtuellen Web-Servern mit schematischer Konfiguration (nur ServerName, DocumentRoot und möglicherweise ScriptAlias unterscheiden sich) ermöglicht das Apache-Modul `mod_vhost_alias`. Näheres hierzu enthält die Apache-Dokumentation.

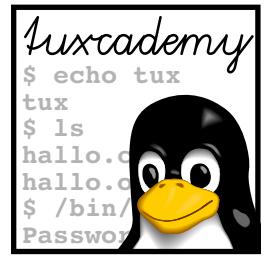
Übungen

 7.3 [3] Richten Sie auf Ihrem Apache-Server einen Ersatzserver ein, der alle Anfragen an nicht existierende IP-basierte virtuelle Server auf eine Webseite umlenkt, die eine entsprechende Fehlermeldung ausgibt.

 7.4 [2] Sorgen Sie dafür, dass diese Fehlermeldung auch erscheint, wenn ein nicht existierender namensbasierter virtueller Server angesprochen wird.

Zusammenfassung

- Durch »virtuelle Server« kann ein einziger Apache-Server mehrere unabhängige Web-Präsenzen anbieten.
- Virtuelle Server werden durch `<VirtualHost>`-Blöcke in der `httpd.conf`-Datei konfiguriert.
- IP-basierte virtuelle Server werden durch ihre IP-Adresse unterschieden.
- Namensbasierte virtuelle Server teilen sich eine IP-Adresse; die Unterscheidung erfolgt durch den Wert der `Host`-Zeile im Kopf eingehender HTTP-Anfragen.
- Mit dem Servernamen `_default_` können Ersatzserver für nicht existierende IP-basierte virtuelle Server definiert werden.
- »Massenhosting« erlaubt das Modul `mod_vhost_alias`.



8

Apache-Sicherheit

Inhalt

8.1	Was und warum?	94
8.2	Betriebssystem und Netz	96
8.3	Apache-Sicherheitstips	97

Lernziele

- Gängige Sicherheitsprobleme erkennen und vermeiden können
- Überblick über den Einsatz von Apache in der DMZ haben

Vorkenntnisse

- Überblick über Apache-Konfiguration (Kapitel 3)
- Apache-Zugriffsrechte und Zugriffsschutz (Kapitel 5)
- TCP/IP-Kenntnisse
- Nützlich: Kenntnisse über Netzwerksicherheit und Firewalls

8.1 Was und warum?

»Sicherheit« ist ein vielzitiertes Schlagwort, auch und gerade wenn es um den Betrieb von Computersystemen geht. Web-Server dienen sehr oft dazu, allen Benutzern des Internet Zugriff zu Ressourcen zu geben, und es ist heute leider wahrscheinlich, dass diese Zugriffsmöglichkeiten auch ausgenutzt werden, um zu schauen, was man mit dem Server noch so alles machen kann – entweder aus Spieltrieb oder sogar mit böser Absicht im Hintergrund.

- Als Administrator eines Web-Servers sollten Sie darum darauf achten, Spaßvögel und Spionen aus dem Netz eine möglichst geringe Angriffsfläche zu bieten. Sie sollten nicht nur den Web-Server so konfigurieren, dass keine »Hintertüren« offenbleiben, durch die jemand vom Netz aus Daten ausspähen kann, die ihn nichts angehen (oder Schlimmeres), sondern die ganze Infrastruktur so ausrichten, dass möglichst wenige Angriffsmöglichkeiten existieren. In der dynamischen Welt der Rechnersicherheit müssen Sie außerdem ständig *up to date* bleiben und sich über die neuesten Entwicklungen unterrichten, sowohl was mögliche Angriffe angeht als auch wie Sie sie gegebenenfalls abwehren. Kein Programm ist sicher und fehlerfrei, und auch in einem verbreiteten und (inzwischen) gut getesteten Programm wie Apache werden immer wieder sicherheitsrelevante Probleme gefunden – und prompt korrigiert, aber Sie müssen als Administrator wissen, dass es ein Problem gibt, dass es eine Korrektur gibt, und Sie müssen die Korrektur auch im eigenen System installieren, bevor Sie ruhig schlafen können.

Informationen über frisch gefundene Sicherheitslücken finden Sie beispielsweise auf den WWW-Seiten der *Apache Software Foundation* (siehe Abschnitt 2.9) oder in einschlägigen Foren auf dem Internet, beispielsweise BUGTRAQ [BUGTRAQ].

- Auch die Linux-Distributionshersteller geben bei Bedarf sogenannte *advisories* heraus, in denen sie die Anwender ihrer Produkte über Probleme unterrichten und auf reparierte Versionen hinweisen. Zum Glück liegen zwischen dem Bekanntwerden eines Problems und der weiten Verfügbarkeit einer Korrektur meist nur Tage, aber in diesen Tagen kann schon einiges passieren; Sie sollten sich also nicht auf Ihren Lorbeeren ausruhen.

Welche Gefahren drohen nun einem Web-Server (und seinen Administratoren)? Hier ist eine Aufstellung der wichtigsten Kategorien [Ste98]:

- Ausspähen vertraulicher Informationen
- Verändern von Inhalten (öffentliche Peinlichkeit)
- Einbruch in den Server-Rechner
- Verwenden des Server-Rechners, um andere Rechner anzugreifen
- Dienstverweigerung (*denial of service*)

- Wer kommt als »Angreifer« in Frage? Hier können Sie zwischen **externen** und **internen Bedrohungen** unterscheiden. Interne Angreifer befinden sich in der eigenen Organisation, externe Angreifer kommen aus dem Internet insgesamt. Zu den externen Angreifern zählen beispielsweise *script kiddies*, also relativ inkompetente und unerfahrene Benutzer, die vorgefertigte Einbruchswerkzeuge (*exploits*) ausnutzen, um aus Spaß fremde Server anzugreifen, aber auch erfahrene und kompetente Cracker, die entweder aus politischen oder weltanschaulichen Gründen oder für Geld Angriffe ausführen. Interne Angreifer können verärgerte oder ehemalige Angestellte sein, die ihre Zugriffsrechte ausnutzen. Neben Fehlern in der Software können auch Fehlkonfigurationen oder der Missbrauch von Zugangsinformationen wie Benutzernamen und Kennwörtern »Vektor« für einen Angriff sein.

Die Sicherung eines Systems beginnt immer mit zwei grundlegenden Schritten:

- Zielvorgabe
- Sie machen eine **Zielvorgabe**, die beschreibt, welcher Grad an Sicherheit gewünscht wird. Perfekte Sicherheit ist nicht erreichbar; der Grad an Sicherheit, der erreicht werden kann, ist eine Funktion des dafür betriebenen

Aufwands, wobei die Abhängigkeit nicht proportional ist, sondern eher logarithmisch: Mit relativ wenig Aufwand können Sie einen ganz anständigen Grad an Sicherheit erreichen, aber Sie kommen irgendwann an einen Punkt, wo sehr großer zusätzlicher Aufwand nur noch relativ wenig zusätzliche Sicherheit bewirkt. Die wenigsten Betreiber von Web-Servern dürften zum Beispiel einen Grad von (physischer) Sicherheit implementiert haben, der ihre Systeme immun gegen einen konzertierten militärischen Angriff macht. (Zum Glück sind auch die wenigsten Betreiber von Web-Servern bei vernünftiger Betrachtung solchen Gefahren ausgesetzt.)

Im Rahmen der Zielvorgabe sollte also eine **Risikoanalyse** stattfinden, die die möglichen Schäden und den Aufwand aufführt, der zu ihrer Behebung notwendig wäre. (Sicherheitsmaßnahmen sollten keinen Aufwand erfordern, der größer ist als der, der zur Behebung der Schäden nötig ist, die auftreten können, wenn die Sicherheitsmaßnahme nicht getroffen wurde.)

Im einzelnen sollten Sie die folgenden Punkte analysieren:

- Wie wird auf den Web-Server zugegriffen? (Art der Netzanbindung, ...)
- Welche Dienste muss der Web-Server anbieten (außer dem HTTP-Dienst vermutlich interaktiven Zugang für die Administratoren und/oder die Möglichkeit, neue Inhalte auf den Server »hochzuladen«)?
- Wie erfolgt der Zugang zu den Inhalten (direkt auf HTML-Dateien, über CGI-Skripte, ein Inhaltsverwaltungssystem, ...)?
- Wie (und von wem) werden die Inhalte geändert?
- Wie wird der Rechner überwacht?
- Wie wird der Rechner physisch gesichert (Klima, USV, Zugang zur Hardware, ...)?
- Was passiert im Falle einer Katastrophe (Hardwaredefekt, Feuer, Einbruch ins System, ...)?

- Zweiter Schritt ist eine **Überprüfung (audit)** des Systems, um festzustellen, inwieweit die Zielvorgabe bereits realisiert ist und welche Maßnahmen noch erforderlich sind, um den geforderten Sicherheitsgrad zu erreichen. Diese Überprüfung muss auch im späteren Betrieb des Systems in periodischen Abständen wiederholt werden.

Zur Überprüfung der Netzinfrastruktur, der Dienste und des Zugangs lassen sich Werkzeuge wie nmap, OpenVAS und ähnliche ausnutzen. Der Zugang aus dem internen Netz und der aus dem Internet sollten getrennt überprüft werden. Es kann sich auch auszählen, die Überprüfung oder zumindest Teile davon an externe Personen zu vergeben.

Für die Zwecke einer Sicherheitsbetrachtung können Sie einen Web-Server in die folgenden Funktionsbereiche einteilen:

- Der Web-Server selbst; die Server-Software (Apache) und alle beteiligten Ressourcen (statische und dynamische, etwa CGI-Skripte)
- Die Betriebssystemplattform, auf der der Web-Server läuft
- Die Netzinfrastruktur, also die Rechnersysteme, die den Server-Rechner mit dem Internet verbinden. Hierzu gehören Router, Firewalls und ähnliche Vorrichtungen (in Hard- und Software).

Alle diese Funktionsbereiche müssen separat gesichert werden.

8.2 Betriebssystem und Netz

Ein grundlegendes Prinzip ist, auf dem Rechner, auf dem der Web-Server läuft, möglichst wenige andere Dienste zu installieren. Im Idealfall läuft auf dem betreffenden Rechner der Web-Server (Apache und allfällige andere Software, die für die Ressourcen notwendig ist) und sonst wenig bis nichts; ein Rechner, der als Web-, Mail-, FTP-, Login-usw.-Server fungiert, ist verwundbar gegen Probleme in jedem einzelnen Dienst, über die dann auch die anderen Dienste in Mitleidenschaft gezogen werden können. Konsequent weitergedacht bedeutet das aber auch, dass im Apache-Server auf diesem Rechner zum Beispiel nur diejenigen Module aktiviert sein sollten, die dieser Apache wirklich benötigt. Ein für einen Web-Server dedizierter Rechner sollte keine Benutzer außer den unbedingt notwendigen unterstützen und Zugang nur z. B. über die *Secure Shell* (ssh) erlauben. Zum Installieren von Inhalten könnten Werkzeuge wie scp, rsync über ssh oder – unter gewissen Umständen und strikter Kontrolle – ein moderner FTP-Server wie etwa der Pure-FTPd verwendet werden.

Vertrauengrenze Zwischen Internet und internem Netz verläuft eine **Vertrauengrenze** – während Sie das interne Netz kontrollieren und ihm deshalb vertrauen können (jedenfalls ziemlich), ist das beim Internet nicht der Fall. Deshalb installieren Sie an dieser Grenze üblicherweise einen **Firewall**, um die beiden Vertrauensbereiche voneinander zu trennen. Hierbei handelt es sich um eine Kombination aus Hard- und Software, typischerweise Router, Paketfilter, Proxy-Server für wichtige Dienste usw. (Je nach Art der Netzanbindung eignen sich abgelegte PCs mit Linux hervorragend zur Implementierung von Firewalls.) Im Idealfall besteht das Firewall-System aus zwei Routern – einem inneren, der dem internen Netz näher steht, und einem äußeren, der dem Internet näher steht. Zwischen den beiden befindet sich die **demilitarisierte Zone** (DMZ). In diesem Pufferbereich werden Server plaziert, die vom Internet aus zu sehen sein müssen; der äußere Firewall lässt Zugriffe auf die Systeme in der DMZ zu, während das interne Netz durch den inneren Firewall vom Internet aus nicht direkt erreichbar ist.

demilitarisierte Zone Firewall-Regeln Firewall-Systeme lassen es normalerweise zu, dass Sie **Regeln** angeben, die beschreiben, welche Pakete durchgelassen und welche abgewiesen werden sollen. Der äußere Firewall sollte z. B. so konfiguriert werden, dass er alle Pakete abweist bis auf die, die an Systeme in der DMZ adressiert sind, beispielsweise den Web-Server. Dienste wie Mail oder WWW für die Benutzer des internen Netzes werden über Proxy-Server in der DMZ geleitet; aus dem internen Netz kann nur der Proxy-Server erreicht werden, nicht das eigentliche Internet, während für die Antworten aus dem Internet ebenfalls nur der Proxy-Server zu erreichen ist (das interne Netz tritt gar nicht in Erscheinung). Die Server in der DMZ sollten auf separaten Rechnern laufen und nicht auf einem der beiden Firewall-Systeme; die Firewall-Systeme sollten nur die allernötigsten Dienste enthalten, da ein Fehler oder eine Sicherheitslücke in der Software des Firewalls das ganze Sicherheitskonzept korrumpern kann. Aus diesem Grund kann es auch sinnvoll sein, für den inneren und den äußeren Firewall nicht dieselbe Sorte Gerät einzusetzen, da sonst die Wahrscheinlichkeit größer ist, dass ein Sicherheitsproblem in beiden Firewalls existiert und einem Angreifer eine direkte Route ins interne Netz ermöglicht.

Das Thema »Konzeption und Installation von Firewall-Systemen« ist umfangreich dokumentiert; in diesem Kurs kann darauf leider nicht näher eingegangen werden. Man sollte allerdings noch einmal darauf hinweisen, dass es in diesem wichtigen Bereich schlüsselfertige Lösungen, die Sie fertig kaufen, installieren und vergessen können, eigentlich nicht gibt. Ein Firewall-System ist immer Teil eines umfassenden Sicherheitskonzepts, das genauer Anpassung und ständiger Pflege bedarf. Im Zweifelsfall ist es immer sinnvoll, professionelle Hilfe heranzuziehen.

8.3 Apache-Sicherheitstips

Für den Apache-Webserver und die darauf laufenden Web-Präsenzen gibt es auch eine Reihe von Punkten, die zu beachten sind, wenn maximale Sicherheit gewünscht wird. Das beginnt damit, dass wie oben erwähnt nur diejenigen Module installiert und aktiviert sein sollten, die wirklich für die angebotenen Ressourcen benötigt werden. Sie sollten die Apache-Konfiguration also genau prüfen und gegebenenfalls nicht gebrauchte Module abschalten. Einige Linux-Distributionen (zum Beispiel die von SUSE) aktivieren praktisch alle standardmäßig mitgelieferten Apache-Module in der Standardkonfiguration.

Einer der elementarsten Punkte ist, dass der Apache-Server normalerweise als `root` gestartet wird. Er nimmt zwar die in der `User`-Direktive angegebene Identität an, um Anfragen zu beantworten, aber trotzdem müssen Sie sicherstellen, dass die Apache-Programmdateien und -verzeichnisse nicht von Unbefugten geändert werden können. Dies betrifft insbesondere das `ServerRoot`-Verzeichnis und seine Unterverzeichnisse, die `root` gehören und kein Schreibrecht für andere Benutzer bieten sollten. Das `DocumentRoot`-Verzeichnis kann – je nachdem – auch für andere Benutzer schreibbar sein, da Apache unter normalen Umständen keine Dateien darin beim Start ausführt. Auf der anderen Seite sollte niemand außer `root` im Verzeichnis mit den Protokolldateien schreiben können, da es sonst möglich wäre, eine Protokolldatei durch ein symbolisches Link auf eine wichtige Systemdatei zu ersetzen und letztere dadurch zu zerstören.

Zur Sicherheit kann es auch nicht schaden, einen `<Directory>`-Block zu installieren, der Zugriffe auf das Dateisystem außerhalb von `DocumentRoot` abweist. Der Apache-Server sollte zwar von sich aus verhindern, dass Benutzer Ressourcen ansprechen, die außerhalb des `DocumentRoot`-Baums liegen, aber sicher ist sicher. Dies lässt sich zum Beispiel durch eine Definition wie

```
<Directory />
  Order Allow,Deny
  Deny from all
  AllowOverride None
</Directory>
```

erreichen. Damit wird gleichzeitig unterbunden, dass Benutzer in eigenen `.htaccess`-Dateien weiterreichende Berechtigungen einstellen können, außer in Verzeichnissen, für die das ausdrücklich freigegeben wurde. Für Verzeichnisse unterhalb von `DocumentRoot` und für etwaige benutzerspezifische Verzeichnisse müssen Sie die `Deny`-Einstellung natürlich wieder rückgängig machen:

```
<Directory /usr/local/apache/htdocs>
  Order Deny,Allow
  Allow from all
</Directory>
```

Sie können auch Überraschungen erleben, wenn URLs auf ungewöhnliche Weise ins Dateisystem abgebildet werden. Manche Unix-Systeme (nicht die gängigen Linux-Distributionen) verwenden das Wurzelverzeichnis »/« als Heimatverzeichnis für `root`; wenn dann noch ein symbolisches Link von »/public_html« nach »/« existiert (auch wenn das weit hergeholt ist), bedeutet das, dass Sie bei der Standardeinstellung von `UserDir` über URLs, die mit `http://www.bla.de/~root/` anfangen, das komplette Dateisystem lesen können. Dasselbe gilt auch für ungewöhnliche `UserDir`-Einstellungen wie »./«. Am besten fügen Sie in der Konfiguration eine Zeile

```
UserDir disabled root
```

um sich vor solchen Überraschungen der unangenehmen Art zu schützen.

Minimale Modulausstattung

Dateisicherheit

Zugriffe außerhalb von `DocumentRoot`

URLs ins Dateisystem

Dynamische Inhalte Server-side includes	Aufpassen sollten Sie auch bei dynamischen Inhalten, zum Beispiel <i>server-side includes</i> oder CGI-Skripten. <i>Server-side includes</i> sollten Sie nur einschalten, wenn Sie sie tatsächlich brauchen, und dann gezielt für die betreffenden Verzeichnisse (mit der Verzeichnisoption <i>Includes</i>). Sie sollten auch nicht jede HTML-Datei blindlings nach <i>server-side includes</i> durchsuchen, sondern auch das über eine gesonderte Dateiendung (.shtml) oder den <i>XBitHack</i> auf diejenigen Dateien beschränken, die tatsächlich entsprechende Direktiven enthalten.
#exec	Vorsicht ist auch geboten bei der <i>#exec</i> -Direktive, mit der Sie beliebige Programme und Shellkommandos aus HTML-Dateien heraus ausführen können. Sie sollten niemals den Kommandoinhalt einer <i>#exec</i> -Direktive von Daten ableiten, die der Benutzer kontrollieren kann (Sachen wie » <i>#exec cmd="\$QUERY_STRING"</i> « sollten Sie sich dringend verkneifen). Es ist möglich und oft sinnvoll, statt <i>Includes</i> die Option <i>IncludesNOEXEC</i> zu verwenden, die die direkte Ausführung von Kommandos über die <i>#exec</i> -Direktive unterbindet. Selbst dann können Benutzer immer noch CGI-Skripte ausführen, indem sie » <i>#include virtual="..."</i> « verwenden, um auf eine Datei zuzugreifen, die in einem <i>ScriptAlias</i> -Verzeichnis steht.
Includes vs. IncludesNOEXEC	CGI-Skripte sollten immer nur in Verzeichnissen untergebracht werden, die außerhalb des <i>DocumentRoot</i> -Baums stehen und mit <i>ScriptAlias</i> eingeblendet werden. Es ist prinzipiell möglich, CGI-Skripte in beliebigen Verzeichnissen auszuführen, aber das hat mehr Nachteile als Vorteile. Das »Apache-Security-Tips«-Dokument [apache-sectips] charakterisiert dies etwas drastisch wie folgt:
CGI nur mit ScriptAlias	»Benutzern sollten Sie nur dann die Ausführung von CGI-Skripten in beliebigen Verzeichnissen erlauben, wenn: <ol style="list-style-type: none"> 1. Sie Ihren Benutzern vertrauen, dass sie keine Skripte schreiben, die Ihr System absichtlich oder versehentlich Angriffen aussetzen, 2. Sie Ihre Sicherheit für so schwach halten, dass es auf ein Loch mehr auch nicht ankommt, 3. Sie keine Benutzer haben und nie jemand Ihren Server besucht.« Selbst wenn Sie nur CGI-Skripte in <i>ScriptAlias</i> -Verzeichnissen erlauben, sollten Sie das Einbringen von CGI-Skripten nur vertrauenswürdigen Benutzern gestatten und/oder CGI-Skripte vor der Freigabe sorgfältig untersuchen.
CGI-Identität suEXEC	Normalerweise werden alle CGI-Skripte mit den Rechten des Apache-Benutzers (nobody o.ä.) ausgeführt. Damit haben CGI-Programmierer zwar fast keine Rechte im System, können sich aber immer noch gegenseitig in die Quere kommen. Apache unterstützt einen Mechanismus namens <i>suEXEC</i> , mit dem es möglich ist, CGI-Skripten statt mit den Rechten des Apache-Benutzers mit den Rechten des Skript-Eigentümers auszuführen. Dieser Mechanismus ist etwas diffizil zu konfigurieren, aber das gilt als gute Idee – auf diese Weise werden Apache-Administratoren dazu gezwungen, sich mit <i>suEXEC</i> zu beschäftigen und so mehr über die damit verbundene Sicherheitsproblematik zu lernen. Die Vorgehensweise ist in [apache-suexec] dokumentiert.

Übungen

-  **8.1** [!2] Welche Dienste bietet Ihr Rechner außer dem Apache-Server an? Können Sie auf diese Dienste (oder manche davon) verzichten?
-  **8.2** [2] Deaktivieren Sie alle Apache-Module, die Sie nicht benötigen. Wie viele bleiben übrig?
-  **8.3** [4] Versuchen Sie, eine HTML-Datei mit *server-side includes* zu schreiben, die die oben skizzierte Schwäche aufweist.

Zusammenfassung

- Systemsicherheit ist eine andauernde Aufgabe.
- Grundlage für ein Sicherheitskonzept sind eine Risikoanalyse und regelmäßige Systemüberprüfungen (*audits*)
- Web-Server, Betriebssystemplattform und Netzinfrastruktur müssen separat gesichert werden.
- Ein Firewall wird an der Grenze zwischen Vertrauensbereichen plaziert und schottet diese gegeneinander ab.
- Ein nach außen sichtbarer Web-Server muss in der demilitarisierten Zone (DMZ) laufen.
- Der Apache-Server sollte mit der minimal notwendigen Ausstattung von Modulen laufen.
- Auf Apache-Dateien und die meisten Verzeichnisse sollte nur `root` schreibenden Zugriff haben.
- Ein `<Directory />`-Block schützt das Dateisystem des Server-Rechners vor unbefugten Zugriffen.
- *Server-side includes* sollten, wenn möglich, vermieden oder zumindest eingeschränkt werden.
- CGI-Skripte sollten immer in `ScriptAlias`-Verzeichnissen stehen.
- Der suEXEC-Mechanismus ermöglicht es, CGI-Skripte mit den Privilegien ihrer Eigentümer (und nicht denen des Apache-Benutzers) auszuführen.

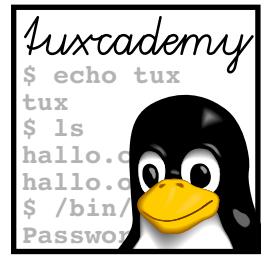
Literaturverzeichnis

apache-sectips »Apache HTTP Server: Security Tips«.
http://httpd.apache.org/docs/misc/security_tips.html

apache-suexec »Apache suEXEC Support«.
<http://httpd.apache.org/docs/suexec.html>

BUGTRAQ Diverse. »BUGTRAQ-Mailingliste«.
<http://www.securityfocus.com/popups/forums/bugtraq/intro.shtml>

Ste98 Lincoln D. Stein. *Web Security: A Step-by-Step Reference Guide*. Addison-Wesley Longman, 1998. ISBN 0-201-63489-9.
<http://www.aw.com/catalog/academic/product/1,4096,0201634899,00.html>



9

Apache-Effizienz

Inhalt

9.1	Allgemeines	102
9.2	Das Prozessmodell von Apache.	102
9.3	Effizienz-Tips	104
9.4	Große und aufwendige Web-Präsenzen	105

Lernziele

- Überblick über die Möglichkeiten der Leistungssteuerung von Apache haben
- Wissen, welche Apache-Direktiven großen Einfluss auf die Leistung haben
- Apache für verschiedene Zugriffsprofile konfigurieren können
- Möglichkeiten zur Leistungssteigerung eines Web-Servers kennen

Vorkenntnisse

- Überblick über Apache-Konfiguration (Kapitel 3)
- TCP/IP-Kenntnisse

9.1 Allgemeines

Geschwindigkeit vs. Korrektheit

Als allgemein verwendbarer HTTP-Server liegt beim Apache das Hauptaugenmerk nicht auf Geschwindigkeit, sondern auf Korrektheit. Erfreulicherweise ist der Apache trotzdem nicht gerade langsam. Wenn es nur um statische HTML-Seiten geht, ist der Apache auf einem Pentium-Rechner aus der Mottenkiste absolut in der Lage, eine 10-MBit/s-Verbindung zu sättigen. Sobald allerdings CGI-Skripte und Datenbanken ins Spiel kommen, kann die Welt schon anders aussehen.

Auf der Hardwareseite ist das Wichtigste, was ein HTTP-Server haben muss, RAM-Speicher. Andere Faktoren wie die CPU, die Platten und die Netzwerkanbindung sind nicht so ausschlaggebend, solange sie »schnell genug« sind (und das ist heute selbst auf billiger Hardware in der Regel der Fall, jedenfalls wenn es nicht um hochkomplexe Web-Präsenzen geht). Sie sollten aber auf jeden Fall vermeiden, dass der HTTP-Server zu swappen anfängt.

9.2 Das Prozessmodell von Apache

Unter Unix gibt es zwei einfache Möglichkeiten, Internet-Server zu implementieren. Die einfachere funktioniert so, dass der Serverprozess an einem Socket auf eingehende Anfragen wartet. Kommt eine Anfrage an, bearbeitet er die Anfrage selbst, schickt das Ergebnis zurück und wartet dann auf die nächste Anfrage. Für viele simple Dienste reicht das aus, aber wenn, wie bei vielen Web-Servern, die Bearbeitung der Anfrage eine Weile dauern kann – etwa weil Datenbankzugriffe und ähnliches im Spiel sind –, dann kann in der Zwischenzeit keine anderen Anfrage beantwortet werden. Die Anfragen stauen sich bis zu einer gewissen Grenze auf, und dann werden sie abgewiesen.

Neuer Prozess pro Anfrage

Die aufwändigeren Methode besteht darin, für jede neue Anfrage, die der Server erhält, einen neuen Prozess zu starten, der die Anfrage bearbeitet. In diesem Moment ist der eigentliche Server bereit für eine weitere Anfrage, und die aktuelle darf so lange laufen, wie sie braucht. Das Problem dabei ist, dass das Erzeugen von Prozessen eine relativ teure Operation ist – und da sehr viele Web-Ressourcen Dinge sind wie kleine Graphiken, möchten Sie teure Operationen wie das Erzeugen von Prozessen oder das Einrichten von TCP-Verbindungen doch eher vermeiden oder zumindest über viele Anfragen hinweg amortisieren.

pre-forking

Apache geht darum traditionell einen Mittelweg, der als *pre-forking* bezeichnet wird: Der Server erzeugt eine gewisse Anzahl von Kindprozessen im Vorhinein. Diese Prozesse bekommen dann eingehende Anfragen übergeben und bearbeiten sie, bevor sie sich für die nächste Anfrage zur Verfügung stellen. Dadurch wird der Aufwand der Prozess-Erzeugung »pro Anfrage« vermieden, aber es ist dennoch möglich, mehrere Anfragen quasi parallel zu verarbeiten. Apache kann in Perioden hoher Last zusätzliche Kindprozesse erzeugen, die dann wieder verschwinden, wenn es ruhiger wird.

Effizienzoptimierung

Ein Bestandteil der Effizienzoptimierung von Apache besteht darin, die Parameter für das *pre-forking* passend zu konfigurieren. Im einzelnen sind das die folgenden:

StartServers

StartServers bestimmt die Anzahl von Kindprozessen, die beim Start des Apache-Programms angelegt werden. Dieser Parameter ist ziemlich irrelevant, da direkt nach dem Start das dynamische Prozess-Erzeugungsmodell von Apache zu greifen beginnt. Lediglich wenn das System allgemein sehr belastet ist, macht der Wert von StartServers möglicherweise einen Unterschied.

MinSpareServers

Mit MinSpareServers können Sie die minimale Zahl von »untätig« Kindprozessen bestimmen. Ein untätig Kindprozess ist ein Kindprozess, der keine Anfrage bearbeitet. Wenn weniger als $\langle n \rangle$ Kindprozesse untätig sind, erzeugt Apache neue Kindprozesse, bis $\langle n \rangle$ erreicht wurde. Dabei werden in der ersten Sekunde ein Kindprozess, in der nächsten zwei, in der darauffolgenden vier usw. erzeugt (bis zu einem Maximum von 32 Kindprozessen pro Sekunde), um Anfragespitzen

abzufangen. Den Wert von `MinSpareServers` lassen Sie am besten so, wie er ist – in jedem Fall sollten Sie ihn nicht auf eine »grosse« Zahl setzen. Hat `MinSpareServers` den Wert m und werden über eine längere Zeit hinweg n Anfragen parallel bearbeitet, dann befinden sich $m + n$ Apache-Prozesse im System.

Als Pendant dazu gibt `MaxSpareServers` die maximale Zahl von untätigen Kindprozessen an. Sind mehr als $\langle n \rangle$ Kindprozesse untätig, so beendet Apache die überzähligen. Auch diesen Wert lassen Sie am besten unverändert; es ist am besten, wenn Sie einige untätige Kindprozesse haben, um plötzliche Lastsituationen abzupolstern, aber untätige Kindprozesse, die nur herumhängen, verbrauchen nutzlos Ressourcen. Zu niedrig sollten Sie den Wert aber auch nicht einstellen, da das Erzeugen von Prozessen einige Rechenzeit kostet.

Jeder Kindprozess bearbeitet bis zu `MaxRequestsPerChild` viele Anfragen, bevor er beendet und ein neuer an seiner Stelle gestartet wird. Der Hauptzweck dieser Direktive besteht darin, Kindprozesse daran zu hindern, dass sie aufgrund von Programmierfehlern Speicher vom Betriebssystem anfordern und nicht wieder freigeben und dem System so langsam der Speicher ausgeht. Außerdem hilft sie dabei, überflüssige Kindprozesse abzubauen, wenn die Serverlast sinkt. Ein Wert von 0 steht für »unbegrenzt viele«.

 Wenn `keepalive` aktiviert ist, dann wird nicht die Anzahl der Anfragen ge- zählt, sondern die Anzahl der TCP-Verbindungen.

`MaxClients` gibt an, wie viele Anfragen gleichzeitig bearbeitet werden können (in anderen Worten, die maximale Anzahl von Kindprozessen). Kommen weitere Anfragen an, werden sie in eine Warteschlange eingereiht, die bis zu `ListenBacklog` viele Anfragen aufnehmen kann. Anfragen aus der Warteschlange werden bearbeitet, sobald wieder ein Kindprozess frei ist. Eine wichtige Aufgabe von `MaxClients` ist, dafür zu sorgen, dass ein »wildgewordener« Apache nicht den ganzen Server-Rechner unbenutzbar machen kann. Sie sollten ihren Wert also nicht so hoch setzen, dass Sie sich nicht mehr auf dem Server-Rechner anmelden können, wenn die Grenze ausgeschöpft ist. Sinnvollerweise behalten Sie den Standardwert bei, solange keine Anfragen abgewiesen werden.

 Der maximale Wert für `MaxClients` ist 256; wenn Sie einen höheren Wert haben möchten, müssen Sie Apache neu übersetzen.

Betrachtenswert für die Leistung des Servers sind auch die Direktiven `KeepAlive` und `KeepAliveTimeout`. `KeepAlive` sollte auf `On` stehen, damit HTTP-Verbindungen besser ausgenutzt werden (Messungen haben gezeigt, dass die Latenzzeit beim Aufbau von HTML-Seiten mit vielen kleinen Graphiken mit `keepalive` nur die Hälfte von dem betragen kann, was sie ohne `keepalive` wäre). `KeepAliveTimeout` sollte so eingestellt werden, dass Verbindungen nicht zu lange offen gehalten werden, nachdem der Benutzer alles bekommen hat, was er braucht – aber auch so, dass Verbindungen nicht zu früh abgebaut werden, solange der Benutzer noch Ressourcen abrufen möchte. Solange ein Prozess auf weitere Anfragen über dieselbe Verbindung wartet, wird er für andere Anfragen blockiert; wenn keine mehr kommen, dann halten Sie sich für die entsprechende Zeit an einer »toten« Verbindung fest, und das ist auch nicht effizient. Im Prinzip verringern Sie mit engl. `keepalive` die benötigte Netzbandbreite und bezahlt dafür mit Server-Ressourcen, die möglicherweise ungenutzt bleiben. Der Standardwert von 15 Sekunden für den `KeepAliveTimeout` ist ganz vernünftig; er sollte auf keinen Fall höher liegen als 60 Sekunden.

Normalerweise konfigurieren Sie diese Parameter direkt in `httpd.conf`. Die SUSE-Distributionen haben einen Mechanismus, bei dem Sie in der Datei `/etc/sysconfig/apache` eine »Leistungsklasse« auswählen können, aus der das SuSEconfig-Programm dann Werte für die Apache-Steuerungsdirektiven ableitet. Tabelle 9.1 enthält eine Übersicht über die jeweiligen Einstellungen.

Heute ist die Bedeutung der Direktiven `StartServers`, `MinSpareServers` und `MaxSpareServers` viel geringer als früher, da die Algorithmen des Apache-Servers zur Prozessverwaltung stark verbessert wurden. Der Server protokolliert in der Datei `error_log`,

Tabelle 9.1: Leistungsklassen für den Apache-Server bei SuSE-Linux (Stand: SuSE-Linux 8.2/SLES 8)

Leistungsklasse	StartServers	MaxSpareServers	MinSpareServers	MaxClients
slim	1	1	1	150
mid	5	10	5	150
thick	50	20	10	150
enterprise	150	150	50	250

wenn mehr als 4 Kindprozesse pro Sekunde erzeugt wurden; wenn viele solche Einträge auftreten, sollten Sie `MinSpareServers` und `MaxSpareServers` erhöhen.

9.3 Effizienz-Tips

Neben den Direktiven für das Prozessmodell haben auch diverse andere Apache-Direktiven Auswirkungen auf die Effizienz. Hier sind einige Dinge, auf die Sie aufpassen sollten:

HostNameLookups

Die Direktive `HostNameLookups` bestimmt, ob IP-Adressen von Anfragen für die Zwecke von Protokolldateien und CGI-Skripten im DNS nachgeschlagen werden. Dies kann den Server sehr verlangsamen, und Sie sollten sich darum gründlich überlegen, ob das nötig ist (siehe auch Abschnitt 6.2). DNS-Anfragen werden auch gemacht, wenn Sie Zugriffskontrolle über Rechner- und Domännamen in der Form

```
<Directory /usr/local/apache/htdocs>
  Order Deny,Allow
  Deny from all
  Allow from bla.de
</Directory>
```

UseCanonicalName

verwenden, da geprüft werden muss, ob die IP-Adresse einer Anfrage zur genannten Domain gehört oder nicht. Dabei werden sogar doppelte Anfragen gestellt, da zunächst versucht wird, den Rechnernamen für die IP-Adresse zu bestimmen, und anschliessend noch einmal geprüft wird, ob der resultierende Rechnernamen wirklich der IP-Adresse zugeordnet ist. Sie sollten diese Art von Abfrage darum sinnvollerweise auf diejenigen Verzeichnisse einschränken, wo sie wirklich gebraucht wird. Entsprechendes gilt für `UseCanonicalName`.

Multiviews
FancyIndexing

Optionen wie `Multiviews` (für Verzeichnisse) oder die aufwendige automatische Indexerstellung (`FancyIndexing`) sind ebenfalls potentielle Zeitfresser, die mit Bedacht eingesetzt werden sollten. Es ist unklug, etwas anzugeben wie

```
DirectoryIndex index
```

da dies für jeden Zugriff auf die »Indexseite« eines Verzeichnisses den Inhaltsaushandlungsmechanismus (Abschnitt 4.4) in Gang setzt. Es ist besser, eine explizite Liste wie

```
DirectoryIndex index.html index.htm index.shtml
```

zu verwenden.

AllowOverride

Wenn `AllowOverride` einen anderen Wert hat als `None`, sucht Apache normalerweise in jedem Verzeichnis entlang des Pfads zu einer Ressource nach `.htaccess`-Dateien. Am besten setzen Sie

```
<Directory />
  AllowOverride None
</Directory>
```

und konfigurieren abweichende Werte ganz gezielt in den »tiefstmöglichen« Verzeichnissen, um die Anzahl der auf `.htaccess` geprüften Verzeichnisse klein zu halten. Ähnliches gilt für symbolische Links; sind Optionen wie `FollowSymLinks` und `SymLinksIfOwnerMatch` aktiv, dann muss der Server für jedes Verzeichnis und die endgültige Datei prüfen, ob es sich um ein symbolisches Link handelt. Im Fall von `SymLinksIfOwnerMatch` kommt auch noch die Überprüfung des Eigentümers dazu; in diesem Fall (wie auch sonst oft) ist die sicherere Option auch die aufwendigere.

Die Direktive `TimeOut` gibt an, wie lange Apache maximal auf eine TCP-Verbindung wartet. Der Vorgabewert von 300 (Sekunden) ist normalerweise *viel* zu hoch – auch wenn der Server 5 Minuten auf ein Bestätigungspaket zu warten bereit ist, dann gilt das mit großer Sicherheit nicht für den Benutzer am anderen Ende der Verbindung. Sie sollten den Wert sukzessive halbieren, bis eine Einstellung erreicht ist, bei der Benutzer keine Fehlermeldungen bekommen und Sie selbst nicht übermäßig lange auf Übertragungen warten, die am anderen Ende niemand mehr abschickt.

TimeOut

9.4 Große und aufwendige Web-Präsenzen

Für Web-Präsenzen mit wenigen Zugriffen (bis Tausende am Tag) reichen die Standardeinstellungen des Apache mit einiger Sicherheit aus. In der »Königsklasse«, wo es um Millionen von Zugriffen täglich geht, ist ein Linux-PC mit Apache allerdings mitunter überfordert, und es müssen weitere Schritte unternommen werden. Als erstes sollten natürlich die Rechnerhardware und die Netzanbindung adäquat ausgelegt werden. Weitere Strategien zur Effizienzsteigerung beruhen darauf, den einzelnen Apache-Server zu entlasten:

- Verwendung eines **Reverse Proxy**, also eines Proxy-Servers, dessen Aufgabe es ist, nicht den eingehenden, sondern den *ausgehenden* Verkehr zu puffern. Wenn der Apache-Server dynamische Inhalte erzeugt, die nicht für jede Anfrage verschieden sind, kann dies den eigentlichen Web-Server stark entlasten. Ein bekanntes freies Programm, das als »Reverse Proxy« fungieren kann, ist `squid`.
- Verwendung von Lastverteilung (engl. *load sharing* bzw. engl. *load balancing*), entweder auf DNS-Ebene oder über spezielle Hardware. Im einfachsten Fall können Sie einen DNS-Server so konfigurieren, dass er auf Anfragen nach Rechnern wie `www.bla.de` zufällig oder im Ringelreihen-Verfahren eine von mehreren IP-Adressen zurückgibt, die dann auf verschiedene Server-Rechner mit identischem Datenvorrat verweisen.

Reverse Proxy

Übungen



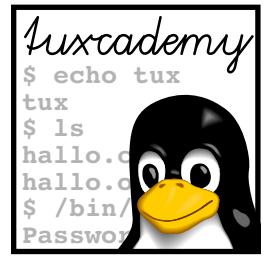
- 9.1** [4] Konfigurieren Sie zwei Rechner als Web-Server und installieren Sie auf einem davon einen DNS-Server, der als Antwort auf einen gemeinsamen Namen wie `www.example.com` die Adressen der beiden Rechner in abwechselnder Reihenfolge liefert. Vergewissern Sie sich, dass beide Apache-Server je in etwa die Hälfte der Anfragen zugesprochen bekommen.

Zusammenfassung

- Die Hardwareauslegung eines Web-Servers hängt von der zu erwartenden Last ab; am wichtigsten ist in jedem Fall genug Hauptspeicher.
- Apache erzeugt Kindprozesse »auf Vorrat«, um Anfragen effizient und parallel behandeln zu können.
- Die Prozess-Erzeugung des Apache-Servers kann mit Direktiven wie StartServers, MinSpareServers, MaxSpareServers und MaxClients gesteuert werden.
- Diverse andere Apache-Direktiven haben auch Einfluss auf die Effizienz des Servers.
- Strategien zur Unterstützung von Web-Präsenzen mit viel Nachfrage sind beispielsweise *reverse proxying* oder Lastverteilung.

Literaturverzeichnis

Kil02 Patrick Killelea. *Web Performance Tuning*. Sebastopol, CA: O'Reilly & Associates, 2002, 2. Auflage. <http://www.oreilly.de/catalog/webpt2/>



10

SSL, TLS, OpenSSL und mod_ssl

Inhalt

10.1 SSL	108
10.1.1 Grundlagen	108
10.1.2 Probleme mit SSL und TLS	110
10.2 OpenSSL	114
10.3 mod_ssl	115

Lernziele

- Die Prinzipien von SSL und TLS kennen
- OpenSSL kennen und installieren können
- mod_ssl kennen und installieren können

Vorkenntnisse

- Grundkenntnisse über Kryptografie
- Kenntnisse über Konfiguration und Betrieb eines Apache-Servers
- Ggf. Kenntnisse über die Übersetzung und Inbetriebnahme von C-Programmen unter Linux

10.1 SSL

10.1.1 Grundlagen

Geschichte SSL, kurz für *Secure Socket Layer*, ist das am weitesten verbreitete Sicherheitsprotokoll für TCP/IP. SSL kann beliebige TCP-Verbindungen verschlüsseln und die Kommunikationspartner authentisieren. Es bildet die Grundlage für **HTTPS**, aber kann auch zur Verschlüsselung vieler anderer Protokolle wie POP3, IMAP, SMTP, FTP, TELNET, ... dienen.

 SSL war ursprünglich eine Erfindung der Firma Netscape. Das Geschäftsmodell von Netscape basierte auf der Idee, einen WWW-Browser weithin verfügbar zu machen, der mit einem WWW-Server (dem von Netscape implementierten) vertraulich kommunizieren konnte. Die Möglichkeit, damit zum Beispiel sicher Waren bestellen zu können, sollte ein Alleinstellungsmerkmal der Netscape-Browser-Server-Kombination werden und dazu führen, dass viele Organisationen den Server kaufen (der Browser wurde zwar – quasi – verschenkt, der Server hingegen gewiss nicht). Leider wurden alsbald Details von SSL publik, die eine frei verfügbare Nachimplementierung zuließen, welche in anderen WWW-Servern (etwa Apache) eingesetzt werden konnte. Dies führte mittelbar zum Versagen des Geschäftsmodells von Netscape und zum Ableben der Firma.

TLS  Wenn wir über SSL reden, meinen wir auch **TLS** (engl. *Transport Layer Security*), das sozusagen die Fortführung von SSL als herstellerunabhängiger Standard unter der Ägide der IETF darstellt. TLS 1.0 [RFC2246] entspricht im wesentlichen SSL 3.0, enthält aber einige signifikante Unterschiede, die eine Interoperabilität von TLS 1.0 und SSL 3.0 ausschließen. Aktuell ist TLS 1.2 [RFC5246].

 TLS 1.0 und der Nachfolgestandard TLS 1.1 [RFC4346] unterscheiden sich vor allem darin, dass TLS 1.1 zusätzlichen Schutz gegen bestimmte Angriffe bietet und das Registrieren von Parametern bei der IANA erlaubt. Neuigkeiten in TLS 1.2 gegenüber TLS 1.1 umfassen unter anderem die Abkehr von MD5 und SHA-1 bei pseudozufälligen Funktionen und digitalen Signaturen, die offizielle Aufnahme von TLS-Erweiterungen und AES in den Standard sowie administrative Vereinfachungen etwa bei der Aushandlung von Hash- und Signaturalgorithmen zwischen TLS-Clients und -Servern.

Einsatz in Anwendungen SSL und TLS setzen üblicherweise auf einem zuverlässigen Transportprotokoll wie TCP auf, während Anwendungsprotokolle wie HTTP, SMTP oder IMAP SSL und TLS als eine Art »erweitertes TCP« zu sehen bekommen. Nachdem eine Verbindung über SSL initialisiert ist, gibt es aus der Sicht eines Programms keinen Unterschied mehr zu einer normalen TCP-Verbindung.

 Ursprünglich war TLS nur für TCP gedacht. Es gibt allerdings einen Standard für TLS über datagrammorientierte Protokolle wie UDP [RFC4347].

Es gibt zwei verschiedene Ansätze dafür, SSL in Anwendungen zu integrieren:

- Man kann ein Anwendungsprotokoll so, wie es ist, über eine mit SSL gesicherte Verbindung abwickeln. Das heißt, die »SSL-Version« eines Dienstes bekommt eine eigene Portnummer zugeordnet, und wenn ein Client sich auf dieser Portnummer mit einem Server verbindet, werden als allererstes Authentisierung und Verschlüsselung organisiert, bevor das eigentliche Anwendungsprotokoll ans Ruder kommt. Diese Vorgehensweise kommt vor allem bei HTTPS zum Einsatz, für das der TCP-Port 443 reserviert ist (»normales« HTTP verwendet den Port 80), aber sie ist auch bei anderen Diensten wie POP3S (TCP-Port 995) oder IMAPS (TCP-Port 993) anzutreffen.

- Man kann ein Anwendungsprotokoll so erweitern, dass eine Verbindung zunächst unverschlüsselt und unauthentisiert über den »normalen« Port aufgebaut und anschließend durch ein Kommando im Protokoll in eine verschlüsselte und authentisierte Verbindung umgewandelt wird. Dieser Ansatz wird zum Beispiel bei SMTP verwendet [RFC3207], wo Server als Erweiterung ein STARTTLS-Kommando anbieten können; bei der Begrüßung des Clients gibt der Server bekannt, dass er TLS unterstützt, und anschließend kann der Client mit STARTTLS die bestehende Verbindung in eine TLS-Verbindung umwandeln.

 Traditionell wird SSL eher mit dem ersten und TLS eher mit dem zweiten Ansatz assoziiert. Das hat aber keine tiefere Bedeutung.

Wenn wir nicht ausdrücklich etwas Gegenteiliges verkünden, verwenden wir ab jetzt »SSL« und »TLS« als Synonyme, statt immer »SSL und TLS« zu sagen.

Protokollablauf Hier ist kurz beschrieben, wie eine SSL-Verbindung aufgebaut wird:

1. Der Client verbindet sich mit dem Server und schickt eine Liste der *cipher suites*, die er unterstützt. Eine *cipher suite* ist eine Kombination aus einer Schlüsselaustauschmethode, einem Kryptosystem für die Datenübertragung und einer kryptografischen Hash-Funktion zur Bestimmung von MACs.
2. Der Server wählt die stärkste *cipher suite* aus, die er auch unterstützt, und teilt dem Client diese Auswahl mit.
3. Anschließend schickt der Server sein Zertifikat an den Client.

 Dieser Schritt ist übrigens optional (!) – wenn die Authentisierung des Servers Ihnen nicht wichtig ist, können Sie SSL auch ohne Zertifikate verwenden. Statt (typischerweise) RSA mit dem öffentlichen Schlüssel im Zertifikat wird dann Diffie-Hellman-Schlüsselaustausch verwendet, für den kein öffentlicher Schlüssel und mithin kein Zertifikat nötig ist.

4. Der Client wählt eine Zufallszahl, verschlüsselt sie mit dem öffentlichen Schlüssel des Servers (aus dem Zertifikat) und schickt sie dem Server zurück.
5. Der Server entschlüsselt die Zufallszahl mit seinem privaten Schlüssel. (An dieser Stelle verabschieden die beiden sich von allen unbefugten Zuhörern, die nicht über den privaten Schlüssel des Servers verfügen.)
6. Beide Seiten verwenden die Zufallszahl zum Erzeugen allfälliger Schlüssel für die *cipher suite*.

Durch diese Aushandlung wird – sofern nichts schiefgegangen ist – eine gesicherte Verbindung etabliert, über die Client und Server kommunizieren können. Beim Verbindungsabbau ist nichts Besonderes zu beachten.

 Nach dem Empfang des Serverzertifikats kann der Client theoretisch prüfen, ob das Zertifikat (noch?) gültig ist, etwa über eine Zertifikats-Rückrufliste oder indem er mit der Zertifizierungsstelle Kontakt aufnimmt. In der Praxis passiert das aber eher selten.

Natürlich gibt es a priori keine Garantie, dass nicht ein Angreifer die Verbindung zum Server abfängt und dem Client sein eigenes Zertifikat unterschiebt. Der Angreifer könnte dann eine eigene Verbindung zum eigentlichen Server aufbauen und die Daten mitlesen. Man spricht von einem *Man-in-the-middle*-Angriff. Um

Man-in-the-middle-Angriff

einen solchen Angriff zu vermeiden, muss der Client nicht nur die Signatur auf dem Zertifikat prüfen, sondern auch sicherstellen, dass die Institution, die das Zertifikat ausgestellt hat, vertrauenswürdig ist. WWW-Browser enthalten in der Regel eine Reihe von Zertifikaten, die der Hersteller des Browsers für vornünftig hält. Der Server kann auch vom Client ein Zertifikat fordern, das wird aber selten gemacht.

Nach der Feststellung der Authentizität einigen Client und Server sich auf die zu verwendenden kryptografischen Verfahren, etablieren entsprechend dem oben erklärten Strickmuster einen Sitzungsschlüssel und können so vertraulich kommunizieren. Die SSL-Verbindung sieht dann aus wie eine gewöhnliche TCP-Verbindung. SSL integriert verschiedene populäre Kryptosysteme und erlaubt es Client und Server, nur bestimmte davon anzubieten. Auf diese Weise ist es möglich, Verfahren, die sich als angreifbar erweisen, kompatibel durch andere zu ersetzen. Zum Beispiel bieten viele WWW-Browser die gängigen Kryptosysteme außer in der üblichen Stärke in »schwachen« Varianten mit 40- oder 56-Bit-Schlüsseln an, um (antiquierten) US-Exportrestriktionen Sorge zu tragen. Da die starken Verfahren inzwischen universell zur Verfügung stehen, können Sie die schwachen Varianten in der Browser- oder Serverkonfiguration ausschließen und laufen so keine Gefahr, irrtümlich doch nicht so vertraulich zu kommunizieren, wie Sie dachten.

US-Exportrestriktionen

 Besonders perfide ist der Umstand, dass SSL zu Fehlersuchzwecken einen Modus unterstützt, bei dem das Protokoll zwar wie üblich abgewickelt wird, aber keine tatsächliche Verschlüsselung stattfindet. Sie sollten darauf achten, dass Sie diesen Modus nicht versehentlich aktivieren.

Übungen

 **10.1** [1] Wie viele Zertifizierungsstellen kennt Ihr Browser? Wie ist es möglich, Zertifizierungsstellen das Vertrauen zu entziehen?

 **10.2** [!2] Welche symmetrischen Verschlüsselungsverfahren unterstützt die Version von SSL bzw. TLS in Ihrem Browser? Welche asymmetrischen? Welche kryptografischen Hash-Verfahren?

 **10.3** [2] Warum verwendet HTTP den Ansatz mit einem separaten Port und nicht etwas wie das STARTTLS-Kommando von SMTP?

10.1.2 Probleme mit SSL und TLS

SSL und TLS sind verbreitet und funktionieren im großen und ganzen gut; dennoch sollten einige Probleme mit dem System nicht unerwähnt bleiben. Die meisten dieser Probleme lassen sich allerdings mit entsprechender Sorgfalt zumindest weitgehend entkräften.

privater Schlüssel

Schlüsselverwaltung Ein SSL-Server braucht den zu seinem Zertifikat passenden privaten Schlüssel. Dieser private Schlüssel muss irgendwo auf dem Server zur Verfügung stehen, und das sinnvollerweise ohne dass er Angreifern zu leicht in die Hände fallen kann. Am besten bringen Sie ihn auf einer Smartcard unter, die unter normalen Umständen nicht ausgelesen werden kann. Wenn das nicht möglich ist, haben Sie im Grunde zwei Möglichkeiten: Den Schlüssel unverschlüsselt auf der Platte liegen lassen, oder ihn verschlüsseln. Die letztere Option ist sicherer, verlangt aber, dass zumindest bei einem Server-Reboot jemand zugegen ist, der das Kennwort für die Entschlüsselung des privaten Schlüssels angeben kann. In jedem Fall kann ein Angreifer, der `root`-Rechte auf Ihrem Server hat, den unverschlüsselten privaten Schlüssel erlangen, indem er z. B. den SSL-Server mit einem Debugger beobachtet.

Smartcard

Reboot

Schlechte Zertifikate Mit dem privaten Schlüssel eines Servers kann ein Angreifer einen Server kompromittieren. Oft muss man sich aber gar nicht solche Mühe geben, wenn es gelingt, die Zertifizierungsstelle dazu zu bringen,

ein Zertifikat zu bestätigen, das unrichtige Informationen enthält. (VeriSign, die größte kommerzielle Zertifizierungsstelle, signierte Anfang 2001 einige Zertifikate, die fälschlich vorgaben, Microsoft zu gehören.) So ein Zertifikat sieht für alle Benutzer gültig aus. Prinzipiell kann eine Zertifizierungsstelle Zertifikate »zurückrufen« und für ungültig erklären, indem sie auf sogenannte Zertifikats-Rückruflisten (engl. *certificate revocation lists*, CRLs) gesetzt werden; in der Praxis kann es einerseits sehr lange dauern, bis ein als falsch bekanntes Zertifikat auf einer CRL landet, und andererseits kümmern Clients sich meistens nicht um CRLs (wann haben Sie zuletzt eine CRL heruntergeladen?). Die zeitnahe und effiziente Verteilung von CRLs ist auch problematisch. SSL unterstützt prinzipiell CRLs, aber da es keinen standardisierten Verteilungsmechanismus gibt, lassen CRLs sich zumeist (noch) nicht sinnvoll einsetzen. Dies ist ein starkes Argument dafür, private Schlüssel möglichst effektiv zu schützen.

 Eine Alternative zu CRLs ist das »Online Certificate Status Protocol«, OCSP [RFC2560]. Die Idee hinter OCSP ist, dass SSL-Clients (etwa Browser) mit der Zertifizierungsstelle oder einem entsprechend beauftragten Dritten Kontakt aufnehmen, um die Gültigkeit eines ihnen vorgelegten Zertifikats »in Echtzeit« zu überprüfen.

 Die gängigen Browser unterstützen heutzutage OCSP, aber ob die Zertifizierungsstellen das tun, steht auf einem ganz anderen Blatt.

 Der Vorteil von OCSP gegenüber CRLs ist, dass eine Zertifizierungsstelle keine (möglicherweise gigantischen) Rückruflisten veröffentlichen muss. Auf der anderen Seite muss die Zertifizierungsstelle bereit sein, (möglicherweise gigantische Mengen von) OCSP-Anfragen sehr schnell zu beantworten. (Stellen Sie sich vor, Sie sind die Zertifizierungsstelle, die das Zertifikat für www.google.com ausgestellt hat.) Es gibt Mittel und Wege, die »Kosten« für OCSP auf die Zertifikatsinhaber abzuwälzen – Stichwort »OCSP Stapling« [RFC4366, Abschnitt 3.6] –, aber das ist noch viel weniger verbreitet als OCSP selbst.

Zertifikatsprüfung Clients müssen Zertifikate genau prüfen, bevor sie sich auf deren öffentliche Schlüssel verlassen. Dies ist zum Beispiel für HTTPS die Aufgabe des WWW-Browsers, und es zeigt sich, dass die in der Software enthaltenen Prüfroutinen oftmals fehlerhaft sind.

 Zum Beispiel wurde 2009 bekannt, dass die meisten SSL-Implementierungen ein Nullbyte in der Mitte eines Servernamens im Zertifikat als Ende des Namens ansehen – was aber nicht der Spezifikation entspricht¹. Das heißt, Sie können als Eigentümer der Domain `example.com` ein Zertifikat für den Server

`www.paypal.com<Nullbyte>.example.com`

beantragen (und bekommen das womöglich auch problemlos ausgestellt) und auf Ihrem Server veröffentlichen. Aus der Sicht der gängigen Browser sieht das aber aus wie ein Zertifikat für `www.paypal.com` – Sie könnten also ohne Weiteres als *man in the middle* aktiv werden [Mar09].

 Es kommt sogar noch schlimmer: Wenn Sie sich ein Zertifikat für

`*<Nullbyte>.example.com`

besorgen, passt(e) das aus der Sicht der SSL-Implementierung von Mozilla (Hersteller von Firefox) auf *jede* Domain.

¹ Nullbytes in der Mitte eines Servernamens kommen im wirklichen Leben nicht vor. Da die offiziellen Standards für Zertifikate – X.509 – aber eine andere Datencodierung verwenden, stören sie auch nicht, sondern rutschen bei Aktionen wie dem Signieren von Zertifikaten in der Zertifizierungsstelle einfach so durch.

Selbst wenn das nicht der Fall ist, neigen viele Anwender dazu, beliebige Zertifikate blindlings zu akzeptieren. Beispielsweise bietet `shop.example.com` die Möglichkeit an, sich seinen »virtuellen Einkaufskorb« randvoll zu laden. Anschließend geht es an die Kasse, natürlich werbewirksam »SSL-gesichert«, doch o Wunder! Das Zertifikat gehört zu einem völlig anderen Server, nämlich `ssl.inkasso.de` (Name erfunden), dem Institut, das die Inhaber von `shop.example.com` der Bequemlichkeit und Rentabilität halber damit betraut haben, die Kreditkartenabwicklung, Bonitätsprüfung usw. für ihre Kunden zu übernehmen. Prinzipiell gibt es keinen Grund zu der Annahme, dass `shop.example.com` und `ssl.inkasso.de` *irgendetwas* miteinander zu tun haben – das Szenario ist von einem klassischen *Man-in-the-middle*-Angriff nicht zu unterscheiden, womit die angebliche »Sicherheit« von SSL völlig unterlaufen wird. Im wirklichen Leben scheint das aber kein Hinderungsgrund zu sein.



In einer kaum zu überbietenden Kombination aus Überheblichkeit und Ignoranz haben die Kreditkartenhersteller sich in der jüngeren Vergangenheit etwas einfallen lassen, das alle Versuche, Benutzer über die Gefahren von *Man-in-the-middle*-Szenarien und die erforderliche Vorsicht aufzuklären, ad absurdum führt. Die Rede ist von »3-D Secure«, besser bekannt als »SecureCode« (für MasterCard) und »Verified by Visa«. Die Idee dahinter ist, dass Kunden, die online etwas mit ihrer Kreditkarte bezahlen sollen, vom *Kartenaussteller* ein Formular präsentiert bekommen, wo sie ein Kennwort eingeben müssen. Eine der größeren Macken dieses Ansatzes ist, dass das entsprechende Formular von einem völlig anderen Server kommt als dem des Anbieters, in der Regel völlig anders aussieht, und daher für den Benutzer völlig unterscheidbar von einem überaus plumpen *Man-in-the-middle*- oder Phishing-Angriff ist. Näheres zu diesem Thema findet sich in [MA10].

Zu guter Letzt sollte man darauf aufmerksam machen, dass viele Browser Dutzende von Zertifizierungsstellen kennen und als vertrauenswürdig akzeptieren, bei denen im Einzelfall nicht ohne weiteres klar ist, ob das auch angebracht ist. Zwar können Sie als Benutzer solche Voreinstellungen modifizieren; ob die große Masse der WWW-Anwender diese Möglichkeiten nutzt oder überhaupt kennt, darf aber getrost als zweifelhaft angesehen werden.

Schlechte Zufallszahlen SSL zieht zur Erzeugung von Sitzungsschlüsseln und anderen Geheimnissen Zufallszahlen heran, die auf dem betreffenden Rechner irgendwie generiert werden. Diese Zufallszahlen müssen »kryptografisch« zufällig sein; die (Pseudo-)Zufallszahlengeneratoren, die in die meisten Programmiersprachen integriert sind, reichen hierfür nicht aus. Betriebssysteme wie Linux sammeln zufällige Daten auf der Basis von physikalischen Prozessen – etwa den sehr genau gemessenen Zeitabständen zwischen Tastendrücken oder Mausbewegungen des Benutzers – und stellen sie Anwendungsprogrammen zur Verfügung (in Linux etwa über die »Geräte« `/dev/random` und `/dev/urandom`). Ein Maß für die »Zufälligkeit« von Zufallszahlen ist deren **Entropie**; wenn ein Bit mit derselben Wahrscheinlichkeit 0 sein kann wie 1, enthält es 1 Bit Entropie. Hat ein Datensatz 3 Bit Entropie, heißt das, dass man ihn mit einer Wahrscheinlichkeit von $1/2^3$ (1 von 8) auf Anhieb richtig erraten kann; nach höchstens 8 Versuchen bekommt man ihn auch so heraus, wobei man im Durchschnitt mit 4 Versuchen rechnen muss. Entsprechend muss man für einen 128-Bit-Schlüssel mit 128 Bit Entropie im Durchschnitt 2^{127} Versuche einkalkulieren (eine neun- und dreißigstellige Dezimalzahl) – ein ziemlich aussichtsloses Unterfangen, nur setzt das voraus, dass wirklich 128 Bit Entropie da sind, um Zufallszahlen zu erzeugen. Beispielsweise konnten Ian Goldberg und David Wagner 1996 Schwächen in der Initialisierung des Zufallszahlengenerators der Netscape-Implementierung von SSL Version 2 nachweisen, die dazu führten,

dass maximal nur 47 Bit Entropie verwendet wurden. Mit einigen Tricks konnten Goldberg und Wagner diese Tatsache ausnutzen, um SSL-Sitzungen innerhalb von weniger als einer halben Minute zu kompromittieren [GW96].

Unsichere Kryptografie SSL und TLS als Verfahren sind inzwischen relativ gut analysiert worden. TLS gilt, sinnvollen Gebrauch vorausgesetzt, als hinreichend sicher. Frühere Versionen bis zur ebenfalls recht verbreiteten Version SSL 2.0 hatten allerdings mehr oder weniger gravierende Sicherheitslücken. Aus diesem Grund sollten Sie SSL 2.0 nicht mehr unterstützen, auch wenn Ihre Software Ihnen die Möglichkeit dazu gibt. Sie sollten außerdem, wie bereits erwähnt, die 40- und 56-Bit-Versionen stärkerer Verfahren ausschließen. Heute akzeptable Kryptosysteme sind RC4, 3DES oder (in neueren TLS-Versionen) AES.

 Ganz ohne Probleme ist auch TLS natürlich nicht. Im August 2009 wurde ein Angriff auf SSL 3.0 und alle TLS-Versionen gefunden, der ungefähr wie folgt aussieht: TLS gibt bei einer *bestehenden* Verbindung den Kommunikationspartnern die Möglichkeit, neue Kryptografieparameter, Schlüssel usw. auszuhandeln. Das findet *innerhalb* der bestehenden TLS-Verbindung statt, das heißt, die Datenpakete für die neue Aushandlungsprozedur werden wie andere Datenpakete verschlüsselt, aber sind nicht anderweitig an die Verbindung gekoppelt. Der Angreifer baut also eine eigene Verbindung zum TLS-Server auf (er kann mit ihm kommunizieren, wie er möchte) und übernimmt dann die Verbindung zwischen Client und Server². Dabei leitet er zunächst die Daten des Clients über seine eigene verschlüsselte Verbindung an den Server weiter. Der Client verhandelt aus seiner Sicht in eigener Sache *im Klartext* mit dem Server, während der Server schon eine verschlüsselte Verbindung sieht und denkt, dass der Client *neu verhandelt*. Sobald die Neuverhandlung abgeschlossen ist, kann der Angreifer nicht mehr mitlesen, aber das macht nichts.

 Als Illustration des vorigen Absatzes: Im wirklichen Leben authentisieren Web-Server ihre Clients typischerweise einmal über Benutzername und Kennwort und setzen dann einen Cookie für die Sitzung, den der Client ab da mit jeder neuen Anfrage mitschickt. Wenn der Angreifer am Anfang einer HTTP-Anfrage beliebige Sachen einbauen kann, könnte er an `pizzadienst.example.com` etwas schicken wie

```
GET /pizza?typ=4-jahreszeiten;adresse=angreifer HTTP/1.1
X-Ignoriere-Dies:
```

(ohne Zeilentrener am Ende). Wenn der Client seinerseits etwas schickt wie

```
GET /pizza?typ=hawaii;adresse=client HTTP/1.1
Cookie: meincookie
```

ergibt sich insgesamt die Anfrage

```
GET /pizza?typ=4-jahreszeiten;adresse=angreifer HTTP/1.1
X-Ignoriere-Dies: GET /pizza?typ=hawaii;adresse=client HTTP/1.1
Cookie: meincookie
```

Die Client-Anfrage ab dem `GET` und die Antwort auf diese Anfrage werden zwischen Client und Server verschlüsselt übertragen und der Angreifer kann sie nicht mehr lesen. Das heißt, im Gegensatz zum

²Etwas weil er einen Router zwischen den beiden kontrolliert. In der Praxis würde er wahrscheinlich *erst* die Verbindung vom Client abfangen und *dann* seine eigene Verbindung zum Server öffnen, aber das ist egal.

klassischen *Man-in-the-middle*-Angriff hat der Angreifer keinen Zugriff auf möglicherweise vertrauliche Daten. Allerdings kann der Angreifer von »Nebeneffekten« profitieren, denn er bekommt seine Pizza auf die Rechnung des Clients. (Die Neuaushandlung der Verbindung findet zwischen dem X-Ignoriere-Dies und dem GET des Clients statt, aber da sie auf der SSL-Ebene passiert, bekommt der Web-Server davon nichts mit.)

Effizienz SSL ist, wie nicht anders zu erwarten, merkbar langsamer als gewöhnliches TCP/IP. Schon beim Verbindungsaufbau müssen durch den Austausch von Zertifikaten und Schlüsseln beträchtliche Datenmengen übertragen werden; außerdem wird hier (langsame) asymmetrische Kryptografie verwendet. Nachdem die Sitzung aufgebaut wurde, ist der Effizienzverlust nicht mehr ganz so krass, aber immer noch merkbar, hauptsächlich, weil mehr Daten übertragen werden müssen. Die symmetrische Kryptografie ist im Vergleich auf heutiger Hardware kein so großes Problem mehr. Die gängigen SSL-Implementierungen versuchen durchaus, auf Effizienzanforderungen Rücksicht zu nehmen, machen aber keine Kompromisse, was die Sicherheit angeht. Ein Verfahren, das behauptet, dasselbe zu tun wie SSL, aber viel schneller zu sein, sollte mit einigem Argwohn betrachtet werden.

Übungen

-  **10.4** [1] Wie können Sie bei Ihrem Lieblingsbrowser Informationen über eine SSL-Verbindung wie die Identitäten des Servers und der Zertifizierungsstelle oder die Art des verwendeten Verschlüsselungsverfahrens abrufen?
-  **10.5** [4] Vergleichen Sie die Geschwindigkeit einer HTTPS-Verbindung mit der einer gewöhnlichen HTTP-Verbindung, indem Sie die Zeit messen, die die Übertragung einer großen Ressource benötigt. (Dazu brauchen Sie wahrscheinlich einen SSL-Server gemäß Kapitel ???. Verwenden Sie ein Programm wie wget zum Anfordern von Ressourcen. Diese Übung eignet sich dazu, bei der Kursnachbereitung in Angriff genommen zu werden.)

10.2 OpenSSL

Bibliothek
Werkzeuge

OpenSSL ist eine Implementierung des SSL-Protokolls als Bibliothek, die in Anwendungsprogramme integriert werden kann. OpenSSL ist von SSLeay abgeleitet, einer SSL-Implementierung, die von Eric A. Young und Tim J. Hudson zwischen 1995 und 1998 erstellt wurde. OpenSSL besteht aus einer Bibliothek, die neben diversen grundlegenden kryptografischen Verfahren alle Versionen des SSL-Protokolls inklusive TLSv1 implementiert, und einer Reihe von Programmen, die den Gebrauch dieser Verfahren erleichtern oder beispielhaft in Anspruch nehmen. Mit OpenSSL können Sie beispielsweise Zertifikate generieren und verwalten, Dokumente ver- und entschlüsseln oder Verbindungen zu SSL-Servern aufbauen.

OpenSSL ist »freie Software«, das heißt, sie steht im Quellcode zur Verfügung und darf von jedem verwendet, angepaßt und weiterverbreitet werden. Insbesondere können Sie sich selbst davon überzeugen, dass die kryptografischen Verfahren korrekt implementiert worden sind und dass keine »Falltüren« existieren, die Unbefugten Zugang zu verschlüsseltem oder zu verschlüsselndem Material gewähren.

Die meisten Distributionen enthalten OpenSSL in vorübersetzter Form, so dass die Installation sich darauf beschränkt, das entsprechende Paket zu installieren. Den Quellcode von OpenSSL können Sie von <http://www.openssl.org/> abrufen, wenn Sie es bevorzugen, das Paket selbst zu übersetzen.

Installation

Zur Installation von OpenSSL brauchen Sie Perl 5 und einen ANSI-C-Compiler,

etwa GCC. Heutige Linux-Distributionen enthalten normalerweise beides; natürlich können Sie auch Perl und GCC von Quellcode installieren, aber das wird hier nicht weiter besprochen. Packen Sie die Distributionsdatei (`openssl.tar.gz` oder so ähnlich) an einer geeigneten Stelle aus und lesen Sie die darin enthaltene Datei `INSTALL`. Zur Installation sind die folgenden Schritte notwendig:

```
$ ./config
$ make
$ make test
$ make install
```

(den letzten Schritt müssen Sie wahrscheinlich als `root` durchführen). Das `config`-Skript passt das OpenSSL-Paket an die lokalen Gegebenheiten an und kann über diverse Optionen gesteuert werden; normalerweise installiert OpenSSL sich in das Verzeichnis `/usr/local/ssl`, aber Sie können das mit der Option `--prefix` ändern. Ferner können Sie einstellen, ob *shared libraries* erzeugt werden sollen oder nicht, ob plattformabhängiger Assembler-Code zur Beschleunigung eingebunden werden soll und ob bestimmte Verschlüsselungsverfahren mit eingebaut oder weggelassen werden sollen. Die Details stehen in der Datei `INSTALL` in der OpenSSL-Distribution.

Übungen

 **10.6** [1] Warum ist es wichtig, dass ein Softwarepaket wie OpenSSL als freie Software im Quellcode zur Verfügung steht?

 **10.7** [25] Finden und dokumentieren Sie eine Sicherheitslücke im aktuellen OpenSSL-Paket.

10.3 mod_ssl

Das Apache-Modul `mod_ssl` stellt eine Verbindung zwischen Apache und OpenSSL her. Mit `mod_ssl` unterstützt Apache **HTTPS**, also HTTP über SSL-Verbindungen; ein Apache-Server kann sich gegenüber Clients durch X.509-Zertifikate authentisieren und seinerseits Client-Authentisierung auf der Basis von clientseitigen X.509-Zertifikaten akzeptieren. `mod_ssl` wurde von Ralf S. Engelschall implementiert.

HTTPS

 Seit Apache 2.0 wird `mod_ssl` mit dem Apache-Server mitgeliefert. Es ist also nicht mehr nötig, dass Sie es separat besorgen und vom Quellcode selbst übersetzen. Wenn Sie eine Linux-Distribution verwenden, sollten Sie trotzdem nachschauen, ob das normale Apache-Paket schon `mod_ssl` enthält oder ob Sie dafür noch etwas Anderes installieren müssen.

Die meisten Linux-Distributionen unterstützen `mod_ssl` direkt; es muss also kein großer Aufwand getrieben werden, um die Software lauffähig zu machen. Ihnen als Endbenutzer bleibt lediglich die Installation geeigneter Zertifikate, die Sie sich vorher verschaffen müssen (siehe hierzu die Folgekapitel). Ferner können Sie das Verhalten des Moduls und damit des Apache-SSL-Servers in relativ weiten Kreisen beeinflussen, indem Sie in der `httpd.conf`-Datei (der tatsächliche Name hängt von Ihrer Linux-Distribution ab) entsprechende Direktiven setzen.

Distributionen

 Wenn Ihre Distribution keine vorkonfigurierten SSL-Direktiven in der `httpd.conf`-Datei mitliefert, dann ist es klug, die SSL-Konfiguration in eine eigene Datei, etwa `httpd-ssl.conf`, auszulagern, die in der eigentlichen `httpd.conf`-Datei mit `Include` eingebunden wird. Dies vereinfacht die Wartung von `httpd.conf`, wenn neue Versionen davon erscheinen.

Direktiven Die wichtigsten Direktiven für `mod_ssl` sind:

SSLEngine on|off Schaltet SSL für den betreffenden Server ein oder aus. Auch für virtuelle Server nützlich.

 Seit Apache 2.1 ist auch der Wert `optional` erlaubt, der es erlaubt, eine bestehende HTTP/1.1-Verbindung nach [RFC2817] zu einer TLS-Verbindung hochzustufen. Allerdings gibt es noch keine Browser, die das unterstützen.

SSLProtocol [+|-]<Protokoll> ... Diese Direktive gibt an, welche SSL-Protokollversio-nen akzeptabel sind. Clients können nur eine der erlaubten Protokollversio-nen benutzen (Groß- und Kleinschreibung egal):

SSLv2 Das ursprüngliche SSL-Protokoll von Netscape. Hat bekannte Sicher-heitslücken und sollte darum am besten ausgeschlossen werden, um zu verhindern, dass Clients versehentlich seinen Gebrauch aushan-deln. In aktuellen Apache-Versionen können Sie `SSLv2` überhaupt nicht mehr angeben.

SSLv3 Die Nachfolgeversion von SSL 2.0 und der Vorgänger von TLS 1.0. Wird von fast allen Browsern unterstützt.

TLSv1 Das Nachfolgeprotokoll von SSL 3.0. Wird heute von den meisten Browsern unterstützt. Neuere TLS-Versionen tauchen auf dem Radar von `mod_ssl` (noch) nicht auf.

TLSv1.1 und TLSv1.2 erlauben in modernen Apache-Versionen, die minde-stens OpenSSL 1.0.1 benutzen, die Verwendung der entsprechen-den TLS-Versionen.

All Alle Versionen. Für aktuelle Apaches heißt das »`+SSLv3 +TLSv1 +TLSv1.1 +TLSv1.2`«, wenn sie OpenSSL 1.0.1 (oder später) benutzen, oder »`+SSLv3 +TLSv1`« mit älteren OpenSSL-Implementierungen.

SSLCipherSuite <Schlüsselliste> Gibt an, welche SSL-Verschlüsselungsverfahren ak-zeptabel sind. Details der Syntax und der möglichen Verfahren finden Sie in der Dokumentation zu `mod_ssl`.

SSLRandomSeed <Kontext> <Quelle> [<Bytes>] OpenSSL benötigt Zufallszahlen von »kryptografischer« Qualität, also solche, die nicht von Angreifern erraten werden können (die typischen »Zufallsgeneratoren« von gängigen Pro-grammiersprachen kommen nicht in Frage). Die Bibliothek enthält einen guten Zufallszahlengenerator, der aber mit hinreichender Entropie versorgt werden muss. Diese Direktive beschreibt die Quellen von Entropie für die erste Initialisierung (<Kontext> ist `startup`) oder für den SSL-Verbindungs-aufbau (<Kontext> ist `connect`). Die möglichen Quellen sind

builtin Die »eingebaute« Quelle ist immer vorhanden, liefert aber gerade beim Start sehr wenig Entropie (es stehen wenige »zufällige« Bits zur Verfügung). Wenn möglich, sollte zumindest für `startup` eine bessere Quelle verwendet werden.

file:<Dateiname> Der benannten Datei werden zufällige Bytes entnommen. Linux verfügt über zwei »Geräte« namens `/dev/random` und `/dev/urandom`, die Entropie aus dem laufenden System sammeln und sich gut als Quelle eignen. Der Unterschied zwischen den beiden ist, dass `/dev/random` nur so viel Entropie liefert, wie tatsächlich zur Verfügung steht: Wenn Sie 256 zufällige Bytes lesen wollen und es sind gerade nur 100 vorhanden, dann blockiert der Lesevorgang, bis genug zufällige Bytes zur Verfügung gestellt werden können. `/dev/urandom` tut das nicht, aber dafür sind die resultierenden zufälligen Bytes möglicherweise nicht ganz so gut wie die von `/dev/random`.

exec:⟨Programmname⟩ **und** **egd:**⟨Socketname⟩ Noch zwei Möglichkeiten für Zufallsquellen: ein externes Programm und der *Entropy-Gathering Daemon*. Diese sind unter Linux nicht so wichtig, weil es die **random*-Geräte gibt. Näheres steht in der mod_ssl-Dokumentation.

Zusammenfassung

- SSL und TLS sind die verbreitetsten Sicherheitsprotokolle und können beliebige TCP-Verbindungen verschlüsseln und authentisieren, insbesondere HTTP.
- Am Anfang einer SSL/TLS-Verbindung authentisiert der Server sich gegenüber dem Client über sein Zertifikat. Anschließend werden die zu benutzenden kryptografischen Verfahren ausgehandelt und Schlüssel ausgetauscht, bevor die Verbindung wie eine gewöhnliche TCP-Verbindung verwendet werden kann.
- Zur Prüfung der Authentizität des Servers prüft der Client die Signatur(en) auf dessen Zertifikat und entscheidet anhand der Vertrauenswürdigkeit der Zertifizierungsstelle.
- OpenSSL ist eine freie Implementierung von SSL und TLS zur Integration in eigene Software.
- mod_ssl stellt die Verbindung zwischen Apache und OpenSSL her. Apache kann damit SSL zur Server- und Client-Authentisierung einsetzen.

Literaturverzeichnis

GW96 Ian Goldberg, David Wagner. »Randomness and the Netscape Browser«.
Dr. Dobb's Journal, Januar 1996.

<http://www.eecs.berkeley.edu/~daw/papers/ddj-netscape.html>

MA10 Steven J. Murdoch, Ross Anderson. »Verified by Visa and MasterCard SecureCode: or, How Not To Design Authentication«, Januar 2010.

<http://www.cl.cam.ac.uk/~rja14/Papers/fc10vbwsecurecode.pdf>

Mar09 Moxie Marlinspike. »Null Prefix Attacks Against SSL/TLS Certificates«, Juli 2009. <http://www.thoughtcrime.org/papers/null-prefix-attacks.pdf>

RFC2246 T. Dierks, C. Allen. »The TLS Protocol Version 1.0«, Januar 1999.

<http://www.ietf.org/rfc/rfc2246.txt>

RFC2560 M. Myers, R. Ankney, A. Malpani, et al. »X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP«, Juni 1999.

<http://www.ietf.org/rfc/rfc2560.txt>

RFC2817 R. Khare, S. Lawrence. »Upgrading to TLS Within HTTP/1.1«, Mai 2000. <http://www.ietf.org/rfc/rfc2817.txt>

RFC3207 P. Hoffman. »SMTP Service Extension for Secure SMTP over Transport Layer Security«, Februar 2002. <http://www.ietf.org/rfc/rfc3207.txt>

RFC4346 T. Dierks, E. Rescorla. »The Transport Layer Security (TLS) Protocol Version 1.1«, April 2006. <http://www.ietf.org/rfc/rfc4346.txt>

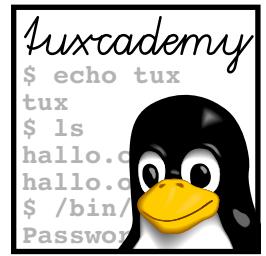
RFC4347 E. Rescorla, N. Modadugu. »Datagram Transport Layer Security«, April 2006. <http://www.ietf.org/rfc/rfc4347.txt>

RFC4366 S. Blake-Wilson, M. Nystrom, D. Hopwood, et al. »Transport Layer Security (TLS) Extensions«, April 2006. <http://www.ietf.org/rfc/rfc4366.txt>

RFC5246 T. Dierks, E. Rescorla. »The Transport Layer Security (TLS) Protocol Version 1.2«, August 2008. <http://www.ietf.org/rfc/rfc5246.txt>

VMC02 John Viega, Matt Messier, Pravir Chandra. *Network Security with OpenSSL*. Sebastopol, CA: O'Reilly & Associates, 2002. ISBN 0-596-00270-X.

<http://www.oreilly.com/catalog/openssl/>



11

Zertifikate

Inhalt

11.1	Zertifizierung	120
11.2	X.509	121
11.3	Zertifizierungsstellen	123
11.4	Erzeugung von Zertifikaten	125

Lernziele

- Struktur und Einsatz von OpenSSL-Zertifikaten kennen
- Aufgaben und Tätigkeiten einer Zertifizierungsstelle (CA) kennen
- Eine eigene Zertifizierungsstelle einrichten können

Vorkenntnisse

- Kenntnisse über Kryptographie
- Kenntnisse über SSL und OpenSSL (Kapitel 10)

11.1 Zertifizierung

public-key infrastructure	In einer Infrastruktur, die auf öffentlichen Schlüsseln beruht (public-key infrastructure, PKI), müssen Teilnehmer in der Lage sein, die öffentlichen Schlüssel ihrer Kommunikationspartner herauszufinden. Ist die Teilnehmergruppe klein genug, so ist das kein Problem (man kennt sich); ein Teilnehmerkreis dagegen, der potentiell das ganze Internet umfasst, braucht eine Methode, mit deren Hilfe sich die Authentizität der öffentlichen Schlüssel beliebiger Parteien sicherstellen lässt – in anderen Worten, die bestätigt, dass der öffentliche Schlüssel, der für eine Person, einen Web-Server oder ein Unternehmen kursiert, tatsächlich zu der betreffenden Person, dem Web-Server oder dem betreffenden Unternehmen gehört. Hierzu verwendet man die Idee der Zertifizierung : Eine »vertrauenswürdige« Instanz wird herangezogen, um die Echtheit eines öffentlichen Schlüssels zu bestätigen.
Vertrauensnetz	Die genaue Ausgestaltung dieser vertrauenswürdigen Instanz ergibt sich nicht zwangsläufig. Das E-Mail-System PGP bedient sich dazu eines Vertrauensnetzes (engl. <i>web of trust</i>) – jeder Teilnehmer zertifiziert die Schlüssel der Teilnehmer, die er unmittelbar kennt, und andere Teilnehmer übernehmen dieses Vertrauen nach gewissen Regeln von ihm. Solange sich eine »Vertrauenskette« zu einer tatsächlichen Bestätigung der Authentizität herstellen lässt, gilt der Schlüssel als authentisch.
Zertifizierungsstellen	Der von (Open)SSL verfolgte Ansatz dagegen beruht auf »zentralen« Zertifizierungsstellen (engl. <i>certification authorities</i> , CAs), die ihr Mandat entweder von Staats wegen zugesprochen bekommen oder aufgrund ihrer Geschäftspraktiken und ihres guten Rufs Vertrauen »verdienen«. So gibt es Firmen wie Symantec (die 2010 das Zertifizierungsstellen-Geschäft von VeriSign übernommen hat), Thawte (die sich in den 1990er Jahren den PKI-Markt zu etwa gleichen Teilen mit Verisign teilte, bis Thawte 1999 von Verisign übernommen wurde; heute ist Thawte ein Tochterunternehmen von Symantec) oder auch die T-Systems Enterprise Services GmbH, die eine Zertifizierung von öffentlichen Schlüsseln als Dienstleistung anbieten. Grundsätzlich kann jeder OpenSSL-Benutzer als Zertifizierungsstelle tätig werden – die notwendige Software ist frei verfügbar –, allerdings gibt es normalerweise keinen besonderen Grund, warum Sie als Anwender einer beliebigen anderen Person auf dem Internet vertrauen sollten, und in den meisten Ländern können an die Tätigkeit als »echte« Zertifizierungsstelle im Sinne der jeweiligen Gesetzgebung mehr oder weniger strenge Auflagen gekoppelt sein.
	 Wie auf dem Gebiet der Standardbetriebssysteme für PCs hat auch bei Zertifizierungsstellen Popularität nichts mit Kompetenz zu tun. Insbesondere die Firma VeriSign hat sich in der Vergangenheit diverse extreme Schnitzer geleistet, was ihrer Beliebtheit jedoch keinen Abbruch zu tun scheint.
Zertifikat	Das Resultat eines Zertifizierungsvorgangs nennt man (naheliegenderweise) ein Zertifikat – das Zertifikat wird von der Zertifizierungsstelle digital signiert, die damit die Verantwortung dafür übernimmt, dass die Zuordnung zwischen Inhaber und Schlüssel korrekt ist. Das Zertifikat benennt den öffentlichen Schlüssel, um den es geht, und die Identität des Inhabers; daneben enthält es die Identität der Zertifizierungsstelle und deren Signatur, die die Authentizität des Schlüssels bestätigt. Dazu kommen noch Verwaltungsinformationen wie beispielsweise das Datum der Zertifikatserzeugung und die maximale Gültigkeitsdauer des Zertifikats. Jeder, der das Zertifikat in die Hände bekommt, kann sich mit dem öffentlichen Schlüssel der Zertifizierungsstelle davon überzeugen, dass die Signatur auf dem Zertifikat stimmt, und sich anschließend entscheiden, ob die Zertifizierungsstelle als Institution so glaubwürdig ist, dass das Zertifikat als echt angesehen werden sollte. In der Regel machen das die WWW-Browser automatisch für einen, ohne im Detail nachzufragen.

```

Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 1 (0x1)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: CN=Test-CA, L=Darmstadt, C=DE/emailAddress=ca@example.com,
             O=Zertifizierungsstelle
    Validity
      Not Before: Oct 27 14:36:58 2003 GMT
      Not After : Oct 26 14:36:58 2005 GMT
    Subject: CN=www.example.com, L=Darmstadt,
             C=DE/emailAddress=webmaster@example.com, O=Beispiel GmbH,
             OU=Web-Server
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (1024 bit)
        Modulus (1024 bit):
          00:d2:d1:a2:6b:22:4f:54:83:ae:60:6a:8f:37:05:
          <<<<<
          68:5b:57:b7:34:6e:f6:05:ba:04:5f:ba:f8:da:e0:
          87:17:33:e0:7e:c6:54:b7:f3
        Exponent: 65537 (0x10001)
    X509v3 extensions:
      X509v3 Basic Constraints:
        CA:FALSE
      Signature Algorithm: md5WithRSAEncryption
        69:c4:fa:fb:85:6b:b4:5d:cd:da:e1:c1:51:10:cc:1c:09:bc:
        <<<<<
        7d:e6:59:f0:37:0f:30:fc:fa:d8:2b:c4:77:91:a9:8e:6f:af:
        d7:cd:dc:64

```

Bild 11.1: Ein X.509-Zertifikat

Übungen

 **11.1** [3] Diskutieren Sie die Vor- und Nachteile des *web of trust* à la PGP gegenüber der zentralisierten Architektur von OpenSSL.

11.2 X.509

Für die Details der Zertifikate richtet SSL sich nach der ITU-T-Empfehlung **X.509**, X.509 die einen Rahmen für Dienstleistungen wie Authentisierung unter der zentralen Idee eines »Verzeichnisses« (engl. *directory*) definiert. Bei X.509 handelt es sich allerdings in der Tat nur um eine Empfehlung, nicht um eine genau definierte Norm, so dass sich die tatsächlichen Umsetzungen von Firmen wie Netscape, Microsoft oder RSA im Detail unterscheiden. Beispielsweise kann ein X.509-Zertifikat, das für ein Netscape-Produkt akzeptabel ist, von einem Microsoft-Produkt aufgrund von verschiedenen Auffassungen über die Definition von X.509 abgewiesen werden. Ferner haben X.509-Zertifikate ein codiertes Format, das für menschliche Benutzer nicht sofort lesbar ist – Sie müssen sich also darauf verlassen, dass die Browser daraus das Richtige machen.

X.509-Zertifikate enthalten die folgenden Informationen:

Versionsnummer Gibt an, welche Version des X.509-Standards für dieses Zertifikat gilt (inzwischen gibt es drei verschiedene). Diese Versionsnummer

bestimmt, welche anderen Informationen im Zertifikat zu finden sind.

Seriensummer Der Erzeuger des Zertifikats versieht das Zertifikat mit einer Seriennummer, um es von anderen selbsterzeugten Zertifikaten zu unterscheiden. Diese Seriennummer wird zum Beispiel benötigt, um das Zertifikat notfalls zurückzurufen.

Bezeichnung des Herausgebers Der Name der Institution, die das Zertifikat signiert hat (normalerweise eine Zertifizierungsstelle). Um das Zertifikat benutzen zu können, müssen Sie *dieser* Institution vertrauen (der Institution, deren Schlüssel zertifiziert wird, nicht notwendigerweise). Zertifizierungsstellen signieren üblicherweise ihre eigenen Zertifikate selbst.

Bezeichnung des Signaturalgorithmus Die Zertifizierungsstelle gibt hier an, welches Verfahren sie verwendet hat, um das Zertifikat zu signieren.

Gültigkeitszeitraum Ein Zertifikat gilt nur für eine begrenzte Zeit, die durch ein Anfangs- und ein Enddatum beschrieben wird. Wie lang diese Zeit ist, hängt vor allem von der Sicherheit des privaten Schlüssels ab, mit dem das Zertifikat signiert wurde – ist dieser kurz, so ist eher damit zu rechnen, dass er gebrochen wird und alle damit signierten Zertifikate ihren Wert verlieren. Der Gültigkeitszeitraum von Zertifikaten kann von Sekunden bis zu Jahrzehnten reichen.

Zu zertifizierender Name Der Name der Institution oder Person, deren öffentlicher Schlüssel zertifiziert wird. Dieser **ausgezeichneter Name** (engl. *distinguished name*, DN), wird im X.500-Format angegeben und ist eindeutig auf dem Internet.

Zu zertifizierender Schlüssel Der öffentliche Schlüssel der benannten Institution oder Person, zusammen mit einer Bezeichnung für das Kryptosystem, in dem der Schlüssel verwendet wird, und allfälligen weiteren Parametern.

Signatur Die kryptographische Signatur der Zertifizierungsstelle, die die Authentizität des Schlüssels bestätigt.

Darstellung Diese Informationen werden gemäß ASN.1/DER (*Abstract Syntax Notation 1/Definite Encoding Rules*) codiert und sind damit systemunabhängig beschrieben. Je nach der verwendeten X.509-Version können die Zertifikate noch weitere Informationen enthalten; die heute aktuelle Version 3 zum Beispiel erlaubt die Angabe von **Erweiterungen**, etwa um den Einsatzbereich von Schlüsseln einzuschränken. Bild 11.1 zeigt ein X.509-Zertifikat in lesbbarer Form (einige Zeilen wurden umbrochen).

 Rein interessehalber: Wenn Sie ein Zertifikat vorliegen haben – etwa in der Datei `zert.pem` –, können Sie eine Ausgabe wie in Bild 11.1 wie folgt erzeugen:

```
$ openssl x509 -in zert.pem -text -noout
```

 »PEM« steht für ein Dateiformat, in dem das nach ASN.1/DER dargestellte Zertifikat Base-64-codiert und mit einer Start- und einer Endzeile versehen wird. Das Ganze sieht dann ungefähr so aus:

```
-----BEGIN CERTIFICATE-----
MIIDgTCCAmmgAwIBAgIBATANBgkqhkiG9w0BAQUFADBgMRcwFQYDVQQDEw5MaW51
cCBGcm9udCBDQTEVMBMGA1UECxMMQ0EgKENsaWVudHMpMRowGAYKCZImiZPyLGQB
<<<<< Diverse ähnlich kryptische Zeilen entfernt
2nqdYG/qQsSpe805ikxDU8RIpkRnIcjodVnHAkDth80wc/tW2B56RzgyrHd72mS6
17t0SIc+J476xE5BWVC78waXaDhdgSzd6vtqqBirA0yV9gbjnQ==
-----END CERTIFICATE-----
```

Tabelle 11.1: Symantec-Zertifikate 2013 (Auswahl)

Name	Typ	Stärke	Zertifizierungspraxis	Preis
Digital ID	Client	—	Benutzer kann an der angegebenen Adresse Mail empfangen	\$22,95/Jahr
Secure Site	Server	40–256 Bit	Organisationsüberprüfung	\$399/Jahr
SSL Wildcard	Server	40–256 Bit	Organisationsüberprüfung	\$1.999/Jahr
Secure Site Pro	Server	128–256 Bit	Organisationsüberprüfung	\$995/Jahr
Secure Site (EV)	Server	40–256 Bit	Erweiterte Validierung	\$995/Jahr
Secure Site Pro (EV)	Server	128–256 Bit	Erweiterte Validierung	\$1.499/Jahr

Bei den Server-Zertifikaten gibt es Rabatt, wenn man sie für mehr als ein Jahr ausstellen lässt. Bei der »Organisationsüberprüfung« prüft Symantec, ob es Ihre Organisation gibt, ob Sie das DNS der Organisation kontrollieren und ob Sie überhaupt die Autorität haben, ein Zertifikat beantragen zu dürfen. Zum Thema »erweiterte Validierung« steht mehr auf Seite 124.

Die »ausgezeichneten Namen« oder DNs in Zertifikaten folgen dem X.500-Standard und sind damit hierarchisch aufgebaut. Ein DN wie

CN=Hugo Schulz, OU=Entwicklung, O=Beispiel GmbH,
L=Irgendwo, ST=Südrhein-Ostfalen, C=DE

beschreibt eine Person namens (CN, *common name*) Hugo Schulz, die Mitglied der Abteilung (OU, *organizational unit*) »Entwicklung« der Organisation (O) »Beispiel GmbH« ist, die am Ort (L, *location*) »Irgendwo« im Bundesland (ST, *state*) »Südrhein-Ostfalen« des Staats (C, *country*) Deutschland (DE) angesiedelt ist. (X.500 ist eine umfangreiche und an vielen Stellen nur vage festgeschriebene Norm, für deren genaue Diskussion hier leider kein Platz ist.)

Übungen

 **11.2** [1] Geben Sie einen ausgezeichneten Namen (DN) an, der Sie auf der Basis Ihrer Organisationszugehörigkeit möglichst eindeutig beschreibt.

 **11.3** [2] Welchen Sinn könnte es haben, dass Zertifikate nicht nur ein End-datum der Gültigkeit enthalten, sondern auch ein Anfangsdatum?

11.3 Zertifizierungsstellen

Eine Zertifizierungsstelle (CA) ist einfach eine Person oder Institution, die Zertifikate ausstellt. Wie schon erwähnt wurde, brauchen Sie dafür keine besonderen Privilegien, eine Kopie z. B. der OpenSSL-Software reicht aus. Es ist der individuellen Zertifizierungsstelle überlassen, sich Standards dafür aufzuerlegen, wann sie welchen Schlüssel zertifiziert; eine Firma, die eine Zertifizierungsstelle betreibt, würde zum Beispiel bestätigen, dass es sich beim Inhaber eines Zertifikats um einen Angestellten der Firma handelt. Eine kommerzielle Zertifizierungsstelle wie zum Beispiel Symantec bietet normalerweise mehrere Stufen der Zertifizierung zu unterschiedlichen Preisen an (Tabelle 11.1).

Sie können diverse Arten von Zertifizierungsstellen unterscheiden [GS02]:

Interne CA Eine Organisation zertifiziert ihre eigenen Mitglieder (z. B. Name, Position, ...). Diese internen Zertifikate könnten für den Zugang zu organisationseigenen Informationsquellen benutzt werden oder der Authentisierung von Kunden gegenüber organisationseigenen Ressourcen dienen. Beispielsweise könnte ein Online-Wertpapierverwaltungssystem verlangen, dass Benutzer ein Zertifikat präsentieren, bevor sie Transaktionen über einem gewissen Geldbetrag durchführen dürfen.

Ausgegliederte CA für Angestellte Eine Organisation, die Zertifikate verwenden, aber selbst nicht als CA tätig werden möchte, könnte diese Dienstleistung ausgliedern.

Ausgegliederte CA für Kunden Analog hierzu könnte eine Organisation die Zertifizierung von Kunden extern vergeben, solange sie der externen CA vertraut.

Vertrauenswürdige Dritt-CA Eine Firma oder Regierung kann eine Zertifizierungsstelle betreiben, die öffentliche Schlüssel mit den gesetzlichen Namen von Personen oder Firmen verknüpft. Zertifikate einer solchen Zertifizierungsstelle können Sie als analog zu offiziellen Identitätsdokumenten wie Personalausweisen ansehen.

Selbstsignierte Zertifikate

Um die von einer bestimmten Zertifizierungsstelle ausgestellten Zertifikate benutzen zu können, brauchen Sie den öffentlichen Schlüssel der Zertifizierungsstelle, damit Sie die Authentizität der digitalen Signatur auf den Zertifikaten überprüfen können. Zertifizierungsstellen verbreiten ihre eigenen öffentlichen Schlüssel über Zertifikate, die sie selbst signiert haben; das ist natürlich nicht der Gipfel der Sicherheit, aber irgendwem müssen Sie irgendwann vertrauen. Tatsächlich enthalten die gängigen WWW-Browser schon »ab Werk« Zertifikate für die wichtigsten CAs. Es ist normalerweise möglich, weitere Zertifikate zu installieren.

Zertifizierungspraxis-Beschreibung

Eine Zertifizierungsstelle muss festlegen, wie sie mit Zertifikaten umgeht. Diese Zertifizierungspraxis-Beschreibung (engl. *certification practices statement*, CPS) gibt an, unter welchen Bedingungen und wie eine Zertifizierungsstelle Zertifikate vergibt oder zurückruft – sie beschreibt die »Bedeutung« eines Zertifikats dieser Zertifizierungsstelle. Beim CPS handelt es sich um ein für Menschen geschriebenes Dokument, das ein wichtiges Kriterium zur Auswahl einer CA darstellt; es könnte neben den Zertifizierungskriterien zum Beispiel auch eine Erklärung enthalten, bis zu einem gewissen Betrag für Zertifizierungsfehler zu haften.



Die neueste Mode auf dem Gebiet der Zertifikate sind sogenannte *Extended-Validation*- oder EV-Zertifikate. Das »CA/Browser Forum« (eine Industrievereinigung der wichtigsten Zertifizierungsstellen und Browser-Hersteller) hat sich darauf geeinigt, bestimmte Zertifikate nur nach besonders strengen Regeln zu vergeben und durch solche Zertifikate ausgewiesene Server dann im Browser besonders kenntlich zu machen (typischerweise durch ein grün hinterlegtes URL-Feld; mit »normalen« Zertifikaten versehene Server zeigen an derselben Stelle zum Beispiel Blau). Zertifizierungsstellen, die EV-Zertifikate ausgeben wollen, müssen sich einer jährlichen Prüfung (engl. *audit*) unterziehen. Das CA/Browser Forum veröffentlicht sowohl die Anforderungen für die Erteilung von EV-Zertifikaten als auch die Anforderungen für die CA-Prüfung.



Aus der Sicht eines Web-Server-Betreibers liegt die Latte für die Erteilung eines EV-Zertifikats nicht nennenswert höher, als sie sowieso liegen sollte: Die Anforderungen sehen im Wesentlichen vor, dass eine CA überprüfen muss, dass die antragstellende Organisation an der behaupteten Adresse existiert, geschäftlich tätig und ordentlich als »aktiv« im Handelsregister oder einem vergleichbaren amtlichen Verzeichnis eingetragen ist. Es gibt Ausnahmeklauseln für Regierungsorganisationen und andere Organisationen, für die die angegebenen Kriterien keinen Sinn ergeben.



Als sichtbaren Unterschied für Benutzer (außer dem freundlichen Grünton) enthalten die Zertifikate neben dem üblichen Namen der zertifizierten Organisation und dem Rechnernamen noch die Adresse des Firmensitzes (nichts Besonders) und die Angabe des Orts, wo die Organisation zum Beispiel ins Handelsregister eingetragen ist. Dazu kommt, falls vorhanden, noch die Nummer des Registereintrags.

 Im Gegensatz zu »normalen« Zertifikaten können Sie EV-Zertifikate nicht mit den üblichen Mitteln (etwa OpenSSL) selber erzeugen. Browser erkennen gültige EV-Zertifikate daran, dass diese einen CA-spezifischen Code enthalten¹, den der Browser mit einer fest eingebauten Liste von CAs und Codes vergleichen kann. (Nicht jeder Browser hat dieselbe Liste, ähnlich wie auch bei CAs.) Damit Sie selber Zertifikate generieren können, die wie EV-Zertifikate behandelt werden, müssen Sie also nicht bloß einen geeigneten Code ins Zertifikat schreiben, sondern auch den Browser-Quellcode anpassen, damit der Browser Ihre Zertifikate als »EV-Zertifikat« erkennt. Das funktioniert natürlich nur, wenn Ihre Benutzer dann auch »Ihren« Browser einsetzen.

 Über den Sinn und Unsinn von EV-Zertifikaten kann man eine Weile philosophieren. Eigentlich sollte es sich von selbst verstehen, dass die CAs die gründlichen Maßstäbe, die für EV-Zertifikate gelten, auf *alle* ihre Zertifikate anwenden. Alles andere ist letzten Endes nichts anderes als ein Eingeständnis ihrer eigenen Bequemlichkeit und/oder ihres Unvermögens, ihre Arbeit vernünftig zu machen. Ein Blick in die Preisliste von Symantec zeigt, dass ein »gewöhnliches« Zertifikat für \$399/Jahr über den Ladentisch geht (Stand: Februar 2013), während für ein entsprechendes EV-Zertifikat schon \$995/Jahr oder mehr zu berappen sind. Gleichzeitig sagen die Zertifizierungsstellen mehr oder weniger durch die Blume, dass nur die teuren EV-Zertifikate die Mühe wert sind, weil »die Benutzer« ja sonst nicht erkennen können, dass der betreffende Web-Server wirklich »sicher« ist. – EV-Zertifikate sind also vor allem eine geschickte Methode der Zertifizierungsstellen, um *de facto* die Preise mal eben auf das Dreifache (oder so) zu erhöhen, denn wer möchte sich als respektabler Online-Händler schon mit einem Zertifikat sehen lassen, über das schon der *Hersteller* sagt, dass es nicht mehr das tut, wofür es mal gedacht war?² Der letztendliche Sicherheitsgewinn dürfte am Ende des Tages minimal sein, da der durchschnittliche Benutzer sowieso nicht weiß, was sich hinter einem Zertifikat (geschweige denn einem EV-Zertifikat) verbirgt und das auch nicht wirklich wissen möchte. Denn um sicherzugehen, müsste er ja einerseits – egal ob für eines normales oder EV-Zertifikat – das CPS der ausstellenden Zertifizierungsstelle finden und lesen und andererseits überprüfen, ob er wirklich deren korrekten öffentlichen Schlüssel hat. Und das macht bekanntlich sowieso niemand.

Übungen

 11.4 [2] Besorgen Sie sich ein CPS einer Zertifizierungsstelle Ihres Vertrauens (?) und studieren Sie es.

11.4 Erzeugung von Zertifikaten

Die OpenSSL-Software macht es möglich, diverse Arten von Zertifikaten zu erzeugen. Neben Zertifikaten zur Identifizierung von WWW-Servern und -clients ist es auch möglich, Zertifikate zu erstellen, die zur Zertifizierung anderer Zertifikate taugen. Ein solches Zertifikat dient als Basis einer eigenen Zertifizierungsstelle. Eine Zertifizierungsstelle bekommt **Zertifizierungsanfragen** (engl. *certificate signing requests*, **CSRs**), prüft diese und erteilt auf ihrer Basis Zertifikate, indem sie die CSRs digital signiert.

Ausgangspunkt für alle weiteren Experimente mit OpenSSL ist die Einrich-

Zertifizierungsanfragen
CSRs

Einrichtung einer eigenen Zertifizierungsstelle

¹Ganz präzise gesagt im *certificate policies extension field*. Eine Liste finden Sie zum Beispiel unter http://en.wikipedia.org/wiki/Extended_Validation_Certificate.

²Stellen Sie sich vor, Ihr Metzger sagt Ihnen etwas wie »Nehmen Sie nicht mein Schweinenackensteak, das ist völlig vergammelt und eklig. Hier habe ich doch dieses ganz hervorragende Rinderfilet für Sie, praktisch geschenkt für das Dreifache!«

tung einer eigenen Zertifizierungsstelle. Theoretisch wäre es möglich, mit einem »offiziellen« Zertifikat von einer Firma wie Symantec anzufangen, aber sinnvollerweise experimentieren Sie erst einmal in eigener Regie und holen die bürokratischen Teile dann bei Bedarf nach. Außerdem ist ein offizielles Zertifikat vielleicht überhaupt nicht nötig. – Eine ernstgemeinte private (etwa unternehmensweite) Zertifizierungsstelle sollten Sie übrigens auf einem Rechner einrichten, der nicht anderweitig verwendet wird. Am Netz muss bzw. sollte er auch nicht sein; am besten benutzen Sie einen Notebook-Rechner, der bei Nichtgebrauch in einem Safe eingeschlossen wird.

Weitere Voraussetzung ist, dass die OpenSSL-Software installiert wurde (siehe Kapitel 10). Sinnvollerweise legen Sie für die Zertifizierungsstelle ein eigenes Verzeichnis – etwa `/usr/local/openssl/testCA` – an. Dieses Verzeichnis sollte Unter- verzeichnisse `certs` und `private` haben, und im wirklichen Leben müssen Sie darauf achten, dass niemand Unbefugtes schreibend auf `certs` zugreifen kann, da es ihm sonst möglich ist, dort »trojanische« Zertifikate zu installieren, die anschließend zum Signieren von anderen Zertifikaten benutzt werden. Das Verzeichnis `private` sollte für überhaupt niemanden außer dem Eigentümer zugänglich sein, da dort die privaten Schlüssel der Zertifizierungsstelle abgelegt werden. Zur Initialisierung der Zertifizierungsstelle bietet sich eine Befehlsfolge wie diese an:

```
$ echo $USER
hugo
$ /bin/su -
Password: xxx123
# mkdir -p /usr/local/openssl/testCA
# chown hugo:users /usr/local/openssl/testCA
# exit
$ cd /usr/local/openssl/testCA
$ mkdir private certs
$ chmod 700 private certs
$ echo 01 >serial
$ touch index.txt
```

Oder was auch immer ...

Zertifikatsverwaltung Die Dateien `serial` und `index.txt` werden zur Zertifikatsverwaltung durch OpenSSL benötigt. `serial` enthält die »Seriennummer« des nächsten zu erzeugenden Zertifikats, und `index.txt` verzeichnet die bisher erzeugten Zertifikate zu Archivzwecken und zur Generierung von CRLs.

Konfigurationsdatei für OpenSSL

Der nächste Schritt ist das Anlegen einer Konfigurationsdatei für OpenSSL. Im Prinzip könnten Sie auch von den Standardvorgaben ausgehen, aber es ist mehr als wahrscheinlich, dass Sie, spätestens wenn Sie selbst zum Beispiel Client-Zertifikate erzeugen wollen, sowieso von diesen Vorgaben abweichen möchten. Also können wir das auch gleich richtig erklären. Die Konfigurationsdatei dient im besonderen dazu, dass die OpenSSL-Kommandozeilenwerkzeuge daraus Informationen beziehen, die sie für die Zertifikatserstellung brauchen. Durch geeignete Konfiguration sparen wir uns hier später einige Tipp- (und Merk-)Arbeit.

Beispiel für die Konfiguration

Bild 11.2 zeigt ein Beispiel für die Konfiguration: Die Datei ist in Abschnitte eingeteilt, die jeweils mit einem Schlüsselwort in eckigen Klammern beginnen. Der erste Abschnitt, `[ca]`, enthält Vorgaben für das OpenSSL-Kommandozeilenprogramm `ca`, das sich (wer hätte es gedacht?) mit den Funktionen einer Zertifizierungsstelle befasst. Die einzige darin vorhandene Direktive, `default_ca`, gibt den Namen eines Abschnitts in der Konfigurationsdatei an, der weitere Details über die Zertifizierungsstelle enthält.



Es ist durchaus möglich, in derselben Konfigurationsdatei Vorgaben für diverse Zertifizierungsstellen zu machen, die dann natürlich in verschiedenen Abschnitten stehen müssen.

Angaben über die Zertifizierungsstelle

Der zweite Abschnitt, `[testCA]`, enthält entsprechend Angaben über die Zertifizierungsstelle

```
[ca]
default_ca = testCA

[testCA]
dir          = /usr/local/openssl/testCA
certificate  = $dir/cacert.pem
database     = $dir/index.txt
new_certs_dir = $dir/certs
private_key   = $dir/private/cakey.pem
serial        = $dir/serial

default_crl_days = 7
default_days     = 730
default_md       = sha1

policy          = testCA_policy
x509_extensions = certificate_extensions

[testCA_policy]
commonName      = supplied
localityName    = supplied
countryName     = supplied
emailAddress    = supplied
organizationName = supplied
organizationalUnitName = optional

[certificate_extensions]
basicConstraints = CA:false

[req]
default_bits    = 2048
default_keyfile = /usr/local/openssl/testCA/private/cakey.pem
default_md      = sha1
prompt          = no
distinguished_name = testCA_dn
x509_extensions = testCA_extensions

[testCA_dn]
commonName      = Test-CA
localityName    = Darmstadt
countryName     = DE
emailAddress    = ca@example.com
organizationName = Zertifizierungsstelle

[testCA_extensions]
basicConstraints = CA:true
```

Bild 11.2: Eine Beispiel-Konfigurationsdatei für OpenSSL

fizierungsstelle. Die Verzeichnisse und Dateien, über die dieser Abschnitt redet, haben wir größtenteils gerade eben angelegt. Die anderen drei Direktiven, `default_crl_days`, `default_days` und `default_md`, geben respektive den Zeitabstand an, in dem CRLs veröffentlicht werden (wird in die CRL geschrieben), die Gültigkeitsdauer des Zertifikats und das kryptographische Hash-Verfahren für digitale Signaturen in Zertifikaten. Alle diese Parameter können beim Aufruf des `openssl`-Programms durch Kommandozeilenoptionen überschrieben werden.

Anforderungen an Zertifikate

Die `policy`-Direktive gibt den Namen des Abschnitts in der Datei an, der die Anforderungen an Zertifikate beschreibt. Hier können Sie bestimmen, welche Felder ein *distinguished name* in einem CSR enthalten sein müssen (oder dürfen), damit er akzeptabel ist. Für jedes Feld (bzw. jede Direktive) gibt es drei erlaubte Werte: `match`, `supplied` oder `optional`. Der Wert `match` bedeutet, dass der Wert des betreffenden Feldes mit dem des gleichnamigen Felds im Zertifikat der Zertifizierungsstelle übereinstimmen muss. Als `supplied` gekennzeichnete Felder müssen im CSR vorhanden sein, während `optional`-Felder nicht notwendigerweise vorhanden sein müssen.

Erweiterungen

X.509-Zertifikate können außerdem auch Erweiterungen enthalten. Die Direktive `x509_extensions` gibt an, in welchem Abschnitt diese näher beschrieben sind (hier `[certificate_extensions]`). Die einzige Erweiterung, die wir hier angeben, ist `CA:false` – sie bedeutet, dass die von unserer Zertifizierungsstelle erzeugten Zertifikate nicht verwendet werden können, um wiederum neue Zertifikate zu zertifizieren.



Für Zertifizierungsstellen ist das ein probates Mittel, um Autorität an »Unterzertifizierungsstellen« zu delegieren. Eine Prüfung des Zertifikats muss dann die Authentizität der Zertifikate bis hin zu einem als vertrauenswürdig bekannten CA-Zertifikat verfolgen – das Server-Zertifikat muss von einer Sub-CA signiert sein, deren Zertifikat wiederum von einer CA signiert ist und so weiter, bis Sie bei einem Zertifikat ankommen, das von der tatsächlichen CA signiert wurde. Anfang 2003 wurde ein Implementierungsfehler in diversen populären Browsern (darunter Konqueror und Internet Explorer) publik, der dazu führte, dass diese Erweiterung nicht ausgewertet, sondern *alle* Zertifikate als vertrauenswürdig zum Signieren weiterer Zertifikate angesehen wurden, so dass jeder Inhaber eines VeriSign-Zertifikats prinzipiell selbst Zertifikate machen konnte, die als von VeriSign signiert galten ...

OPENSSL_CONF

OpenSSL verwendet normalerweise eine systemweite Konfigurationsdatei in `/etc/openssl/openssl.cnf` (oder so ähnlich). Wir können bequem unsere eigene Konfigurationsdatei bekannt machen, indem wir ihren kompletten Pfadnamen in die Umgebungsvariable `OPENSSL_CONF` eintragen:

```
$ export OPENSSL_CONF=/usr/local/openssl/testCA/openssl.cnf
```

Zertifikat für die Zertifizierungsstelle

Das Zertifikat für die Zertifizierungsstelle generieren wir mit dem Kommando `req` des `openssl`-Programms. Da wir es nur hierfür verwenden, können wir die Parameter für den Aufruf fest in der Konfigurationsdatei vorschreiben; das hat nicht nur den Vorteil, dass wir sie nicht auf der Kommandozeile angeben müssen, sondern dient außerdem als Dokumentation, wie das Zertifikat angelegt wurde. Ferner ist dies die einzige Möglichkeit, ein X.509v3-Zertifikat mit Optionen zu erhalten. Die entsprechende Konfiguration befindet sich im `[req]`-Abschnitt: `default_bits` gibt an, dass ein 2048 Bit langer Schlüssel erzeugt werden soll – Standard wären 512 Bit, und das ist heutzutage zu unsicher, erst recht für ein Schlüsselpaar, das einer Zertifizierungsstelle zugrunde liegen soll! Die zusätzliche Sicherheit ist den nötigen Rechenaufwand bestimmt wert. Der Schlüssel wird in die in `default_keyfile` benannte Datei geschrieben (die `$dir`-Variable ist leider nur im `ca`-Abschnitt sichtbar) und eine mit dem in `default_md` angegebenen Hash-Verfahren bestimmte Prüfsumme signiert. Hier sollten Sie »sha1« eintragen; das früher verbreitete Verfahren MD5 gilt heute als unsicher. `prompt` bekommt den Wert

no, damit openssl nicht nach den Parametern des DN fragt; sie sind in dem in distinguished_name benannten Abschnitt testCA_dn angegeben. (Das optionale Feld haben wir uns gespart.) Zum Schluss geben wir per X.509-Option an, dass *dieses* Zertifikat in der Tat zum Signieren anderer Zertifikate zu gebrauchen sein soll.

Das Zertifikat für die Zertifizierungsstelle lässt sich dann leicht mit

```
$ openssl req -x509 -days 1825 -newkey rsa:2048 -out cacert.pem
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to '/usr/local/openssl/testCA/private/cakey.pem'
Enter PEM pass phrase:blafasel
Verifying - Enter PEM pass phrase:blafasel
$ _
```

erzeugen. Auf die resultierende Datei cakey.pem sollten Sie sehr gut aufpassen – geht sie verloren, steht der private Schlüssel für die Zertifizierungsstelle nicht mehr zur Verfügung, und das ist der »Super-GAU«: Sie können keine neuen Zertifikate mehr beglaubigen und die existierenden nicht mehr zurückrufen (was fast schlimmer ist). Wenn Sie ein neues Schlüsselpaar generieren, müssen alle Server und vor allem Browser, die Zertifikate der Zertifizierungsstelle akzeptieren sollen, den neuen öffentlichen Schlüssel mitgeteilt bekommen, und das ist ein größeres Ärgernis.

Die *pass phrase* dient zur Freischaltung des privaten Schlüssels der Zertifizierungsstelle vor der Zertifizierung anderer Schlüssel. Es ist möglich, auf eine *pass phrase* für den Schlüssel zu verzichten, allerdings bedeutet das, dass jeder, der irgendwie an den privaten Schlüssel gekommen ist, neue Zertifikate im Namen der Zertifizierungsstelle beglaubigen kann (noch ein »Super-GAU«). Sie sollten das also tunlichst sein lassen. Wird der private Schlüssel kompromittiert, ist damit die komplette Zertifizierungsstelle kompromittiert; Sie können also weder den in der Vergangenheit ausgestellten noch allfälligen in der Zukunft ausgestellt werdenen Zertifikaten noch trauen.

 Die *-days*-Option gibt an, wie lang das Zertifikat der Zertifizierungsstelle gilt. Da mit dem Ablauf dieses Zertifikats die von der Zertifizierungsstelle ausgestellten Zertifikate nicht mehr verifiziert werden können, sollte die Gültigkeitsdauer mit Bedacht gewählt werden: Eine zu kurze Gültigkeitsdauer zwingt Sie dazu, schon bald alle ausgestellten Zertifikate erneuern zu müssen, während eine zu lange Gültigkeitsdauer im schlimmsten Fall bedeuten kann, dass die Zertifizierungsstelle kompromittiert wird, weil Fortschritte in der Computertechnik es möglich gemacht haben, den öffentlichen Schlüssel zu brechen. Ein Zertifikat mit einem 2048-Bit-Schlüssel sollte allerdings für die nächsten 10 Jahre (oder so) sicher sein.

Das erzeugte Zertifikat können Sie sich mit einem Kommando wie

Zertifikat anschauen

```
$ openssl x509 -in cacert.pem -text -noout
```

anschauen. Die Ausgabe sollte so ähnlich aussehen wie Bild 11.3 (die Schlüssel sind natürlich mit großer Sicherheit anders). Beachten Sie, dass »Issuer« und »Subject« denselben DN enthalten – die Zertifizierungsstelle beteuert die Korrektheit des eigenen öffentlichen Schlüssels. Ferner weisen die »X.509v3 Basic Constraints« dieses Zertifikat als Zertifikat einer Zertifizierungsstelle aus.

Herzlichen Glückwunsch: Wenn Sie bis hierhin mitgearbeitet haben, sind Sie nun stolzer Besitzer einer X.509-Zertifizierungsstelle. Im nächsten Kapitel erklären wir Ihnen, wie Sie mit Ihrer Zertifizierungsstelle Zertifikate zum Beispiel für WWW-Server generieren können.

```
Certificate:
Data:
  Version: 3 (0x2)
  Serial Number:
    a2:48:2a:d9:11:19:29:0c
  Signature Algorithm: sha1WithRSAEncryption
  Issuer: CN=Test-CA, L=Darmstadt, C=DE/emailAddress=ca@example.com,
    O=Zertifizierungsstelle
  Validity
    Not Before: Jan 30 23:42:26 2013 GMT
    Not After : Jan 29 23:42:26 2018 GMT
  Subject: CN=Test-CA, L=Darmstadt, C=DE/emailAddress=ca@example.com,
    O=Zertifizierungsstelle
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (2048 bit)
      Modulus (2048 bit):
        00:ad:8d:7b:25:bf:5d:55:67:70:f1:96:3d:9b:36:
        <<<<<<
        04:2b
      Exponent: 65537 (0x10001)
  X509v3 extensions:
    X509v3 Basic Constraints:
      CA:TRUE
  Signature Algorithm: sha1WithRSAEncryption
  a3:06:1a:03:e4:27:58:e4:a4:12:ff:e3:9b:ae:bf:97:e9:52:
  <<<<<<
  3c:1c:e5:6a
```

Bild 11.3: Ein selbstsigniertes Zertifikat für eine Zertifizierungsstelle

Übungen



11.5 [2] Wie können Sie sich von der Authentizität des selbstsignierten Zertifikats einer Zertifizierungsstelle überzeugen?

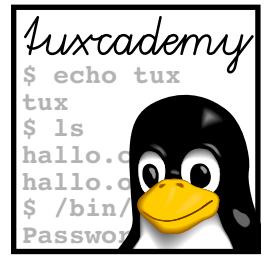
Zusammenfassung

- Ein Zertifikat stellt den Zusammenhang zwischen einer Person, einer Organisation oder einem Server und einem öffentlichen Schlüssel her.
- Zertifikate werden von Zertifizierungsstellen (CA) herausgegeben. Die Vertrauenswürdigkeit eines Zertifikats entspricht der Vertrauenswürdigkeit der Zertifizierungsstelle.
- Zertifikate für OpenSSL folgen dem X.509-Standard. Die Namen von Personen, Organisatoren oder Servern sind als »ausgezeichnete Namen« (*distinguished names*) im Sinne von X.500 angegeben.
- Man kann verschiedene Arten von Zertifizierungsstellen unterscheiden: interne und ausgegliederte CAs sowie vertrauenswürdige Dritt-CAs.
- Zertifizierungsstellen signieren auch Zertifikate mit ihrem eigenen öffentlichen Schlüssel, die dann zur Authentizitätsprüfung anderer Zertifikate verwendet werden. WWW-Browser enthalten oft eine Anzahl solcher Zertifikate für die gängigsten Zertifizierungsstellen.
- Um eine Zertifizierungsstelle einrichten zu können, benötigt man ein entsprechendes selbstsigniertes Zertifikat. Dieses erhält man mit dem Kommando `openssl req`.

Literaturverzeichnis

GS02 Simson Garfinkel, Gene Spafford. *Web Security, Privacy & Commerce*. Sebastopol, CA: O'Reilly & Associates, 2002, 2. Auflage. ISBN 0-596-00045-6.
<http://www.oreilly.com/catalog/websec2/>

VMC02 John Viega, Matt Messier, Pravir Chandra. *Network Security with OpenSSL*. Sebastopol, CA: O'Reilly & Associates, 2002. ISBN 0-596-00270-X.
<http://www.oreilly.com/catalog/openssl/>



12

Apache, OpenSSL und mod_ssl

Inhalt

12.1	Apache und mod_ssl	134
12.1.1	Grundlagen	134
12.1.2	Globale Konfiguration	134
12.2	Server-Authentisierung.	138
12.2.1	Zertifikate installieren	138
12.2.2	Browser und Zertifizierungsstellen	141
12.2.3	Ressourcen-Konfiguration	144
12.2.4	Session Caching	146
12.2.5	Virtuelle Server	149
12.2.6	Apache und TLS-Angriffe.	151
12.2.7	Protokollierung	152
12.3	Client-Authentisierung mit Zertifikaten.	154
12.3.1	Server-Konfiguration	154
12.3.2	Client-Zertifikate im Browser installieren	156
12.3.3	Zugriffsregeln	156
12.4	Sicherheit für Web-Präsenzen verbessern	161
12.4.1	HTTP Strict Transport Security (HSTS)	161
12.4.2	Public Key Pinning (HPKP)	163
12.4.3	Content Security Policy (CSP)	166

Lernziele

- Apache mit mod_ssl konfigurieren können
- Server-Zertifikate zur Authentisierung von Web-Servern (auch virtuellen) installieren und nutzen können
- Client-Zertifikate zur Authentisierung von Clients verteilen, installieren und nutzen können
- Gängige Mechanismen zur besseren Absicherung von Web-Präsenzen kennen und nutzen können

Vorkenntnisse

- Kenntnisse über OpenSSL (Kapitel 10) und Zertifikate (Kapitel 11)
- Kenntnisse über Konfiguration und Betrieb eines Apache-Servers
- Kenntnisse über Bedienung eines WWW-Browsers

12.1 Apache und `mod_ssl`

12.1.1 Grundlagen

Als populärster Web-Server braucht Apache natürlich auch Unterstützung für HTTPS, also authentisierte Web-Präsenzen mit verschlüsseltem Zugriff (und optional der Authentisierung von Clients durch Zertifikate) via TLS. Die gängige Implementierung von HTTPS in Apache beruht auf OpenSSL und verwendet ein Apache-Modul namens `mod_ssl`.

 `mod_ssl` wurde ursprünglich von Ralf S. Engelschall implementiert. In frühen Apache-Versionen mussten Sie es sich separat besorgen und installieren.

 Seit Apache 2.0 wird `mod_ssl` mit dem Apache-Server mitgeliefert. Es ist also prinzipiell nicht mehr nötig, dass Sie es separat besorgen und vom Quellcode selbst übersetzen. Wenn Sie eine Linux-Distribution verwenden, sollten Sie trotzdem nachschauen, ob das normale Apache-Paket schon `mod_ssl` enthält oder ob Sie dafür noch etwas Anderes installieren müssen.

Distributionen Die meisten Linux-Distributionen unterstützen `mod_ssl` direkt; es muss also kein großer Aufwand getrieben werden, um die Software lauffähig zu machen. Ihnen als Endbenutzer bleibt lediglich die Installation geeigneter Zertifikate. Ferner können Sie das Verhalten des Moduls und damit des Apache-TLS-Servers in relativ weiten Kreisen beeinflussen, indem Sie in der `httpd.conf`-Datei (der tatsächliche Name hängt von Ihrer Linux-Distribution ab) entsprechende Direktiven setzen.

 Wenn Ihre Distribution keine vorkonfigurierten SSL-Direktiven in der `httpd.conf`-Datei mitliefert, dann ist es klug, die SSL-Konfiguration in eine eigene Datei, etwa `httpd-ssl.conf`, auszulagern, die in der eigentlichen `httpd.conf`-Datei mit `Include` eingebunden wird. Dies vereinfacht die Wartung von `httpd.conf`, wenn neue Versionen davon erscheinen.

Authentisierung von Servern Mit HTTPS können beide Kommunikationspartner – Client und Server – einer Verbindung authentisiert werden. Die bei weitem wesentlichere Anwendung ist die Authentisierung von Servern: Man möchte den Benutzern das Gefühl geben, dass sie ihre Online-Bestellungen und persönlichen Daten wie Adresse und Kreditkarteninformationen tatsächlich an das Versandhaus X schicken und nicht an den Konkurrenten Y oder irgendeinen kriminellen Z. Die umgekehrte Richtung – Authentisierung von Benutzern gegenüber dem Server – ist für die meisten WWW-Anwendungen viel zu aufwendig: Das finanzielle Risiko, dem einen oder anderen »Irrläufer« aufzusitzen, ist wesentlich eher tragbar als der Umsatzverlust, der daraus entsteht, dass Kunden sich aufwendig TLS-Zertifikate verschaffen und diese in ihrem Browser installieren müssen und dann vielleicht doch entrervt zur Konkurrenz wechseln. Aus diesem Grund setzen Anbieter in der Regel einfachere Authentisierungsmethoden ein (etwa über Benutzernamen und Kennwörter), die für die Kunden mit weniger Umstand verbunden sind, selbst wenn nicht die optimale Sicherheit erreicht wird. Für ein Versandhaus ist das kein fundamentales Problem, denn die herkömmlichen Bestellungen – über (gelbe) Post oder Telefon – sind ja auch nicht besonders authentisiert. Die Infrastruktur, um mit fehlgeleiteten Bestellungen oder säumigen oder zahlungsunwilligen Kunden umzugehen, muss also ohnehin vorhanden sein.

12.1.2 Globale Konfiguration

Im Folgenden beschreiben wir die wichtigsten Direktiven zur Konfiguration von `mod_ssl`. Die meisten davon können entweder in der Konfiguration für den »Hauptserver« oder in `VirtualHost`-Blöcken eingesetzt werden (in welchem Fall sie nur für den betreffenden virtuellen Server gelten).

Damit Apache überhaupt TLS anbietet, müssen Sie das Modul aktivieren. Dazu dient die Direktive `SSLEngine`. Die möglichen Formen sind

<code>SSLEngine on</code>	<i>mod_ssl wird aktiviert</i>
<code>SSLEngine off</code>	<i>mod_ssl wird nicht aktiviert</i>

 Seit Apache 2.1 gibt es auch den Wert `optional`, der es erlaubt, eine bestehende HTTP/1.1-Verbindung nach [RFC2817] zu einer TLS-Verbindung hochzustufen. Allerdings gibt es noch keine Browser, die das unterstützen – der praktische Nutzen hält sich also in engen Grenzen.

Damit HTTPS wirklich funktioniert, müssen Sie Apache auch dazu bringen, auf dem Port 443 auf Verbindung zu lauschen. Sie brauchen also in Ihrer Konfiguration irgendwo eine Zeile wie

`Listen 443`

(es kann gut sein, dass Ihre Distribution dafür sorgt, dass so eine Zeile in Kraft gesetzt wird, wenn Ihr Apache das `mod_ssl`-Modul enthält).

 Apache geht davon aus, dass der Port 443, wenn er überhaupt benutzt wird, für HTTPS benutzt wird. Sie können HTTPS auch über andere Ports laufen lassen, nur müssen Sie das Apache dann mitteilen (ansonsten nimmt er HTTP an):

`Listen 11.22.33.44:10443 https`

Mit `SSLProtocol` können Sie wählen, welche Version(en) des TLS-Protokolls der Server unterstützen soll. Clients können nur eine der freigeschalteten Protokollversionen verwenden (Groß- und Kleinschreibung egal):

SSLv2 Das ursprüngliche SSL-Protokoll von Netscape. Hat bekannte Sicherheitslücken und sollte darum unbedingt ausgeschlossen werden, um zu verhindern, dass Clients versehentlich seinen Gebrauch aushandeln. In Apache 2.4 können Sie `SSLv2` überhaupt nicht mehr angeben.

SSLv3 Die Nachfolgeversion von SSL 2.0 und der Vorgänger von TLS 1.0. Wird von fast allen Browsern unterstützt.

TLSv1 Das Nachfolgeprotokoll von SSL 3.0. Wird heute von praktisch allen Browsern unterstützt. Neuere TLS-Versionen tauchen auf dem Radar von `mod_ssl` (noch) nicht auf.

TLSv1.1 und TLSv1.2 erlauben in modernen Apache-Versionen, die mindestens OpenSSL 1.0.1 benutzen, die Verwendung der entsprechenden TLS-Versionen.

all Alle Versionen. Für aktuelle Apaches heißt das »`+SSLv3 +TLSv1 +TLSv1.1 +TLSv1.2`«, wenn sie OpenSSL 1.0.1 (oder später) benutzen, oder »`+SSLv3 +TLSv1`« mit älteren OpenSSL-Implementierungen.

In Anbetracht des POODLE-Angriffs sollten Sie `SSLv3`, wenn irgend möglich, ausschließen. Etwas wie

<code>SSLProtocol all -SSLv2 -SSLv3</code>	<i>Apache 2.2 (und älter)</i>
<code>SSLProtocol all -SSLv3</code>	<i>Apache 2.4 (und neuer)</i>

ist die empfehlenswerte Einstellung.

 Wenn Sie sich nicht sicher sind, mit welchen Protokollversionen Ihre Benutzer sich mit Ihrem Server verbinden, dann können Sie diese Information protokollieren und statistisch auswerten. Siehe hierzu Abschnitt 12.2.7.

Die akzeptablen *cipher suites* wählen Sie mit `SSLCipherSuite`. Dessen Parameter entspricht einer Auswahlliste für Verschlüsselungsverfahren (siehe `ciphers(1ssl)`). Das heißt, mit etwas wie

```
SSLCipherSuite KEECDH+ECDSA:KEECDH:KEHD:HIGH:+SHA:+RC4:RC4:!aNULL:>
< !eNULL:!LOW:!EXP:!3DES:!MD5:!DSS:!PSK:!SRP:!KECDH:>
< !CAMELLIA:!IDEA:!SEED
```

sind Sie vermutlich auf der sicheren Seite.

 Seit Apache 2.4.7 brauchen Sie »!aNULL;!eNULL;!EXP« nicht mehr zu spezifizieren, da Apache diese Liste automatisch an den Anfang dessen stellt, was Sie selber angeben. Sie können auf Authentisierung und Verschlüsselung also nicht mehr verzichten, selbst wenn Sie wollten.

Sie können natürlich auch eine explizite Liste angeben: etwas wie

```
SSLCipherSuite "ECDHE-ECDSA-AES128-GCM-SHA256 \
ECDHE-ECDSA-AES256-GCM-SHA384 \
ECDHE-ECDSA-AES128-SHA \
ECDHE-ECDSA-AES256-SHA \
ECDHE-ECDSA-AES128-SHA256 \
ECDHE-ECDSA-AES256-SHA384 \
ECDHE-RSA-AES128-GCM-SHA256 \
ECDHE-RSA-AES256-GCM-SHA384 \
ECDHE-RSA-AES128-SHA \
ECDHE-RSA-AES256-SHA \
ECDHE-RSA-AES128-SHA256 \
ECDHE-RSA-AES256-SHA384 \
DHE-RSA-AES128-GCM-SHA256 \
DHE-RSA-AES256-GCM-SHA384 \
DHE-RSA-AES128-SHA \
DHE-RSA-AES256-SHA \
DHE-RSA-AES128-SHA256 \
DHE-RSA-AES256-SHA384 \
EDH-RSA-DES-CBC3-SHA"
```

sollte einen vernünftigen Kompromiss zwischen Sicherheit und Leistung umsetzen, jedenfalls für neuere Apache-Versionen, die Unterstützung für elliptische Kurven anbieten. (Wenn Sie kein ECDSA-Zertifikat haben, können Sie die ersten sechs *cipher suites* weglassen.)

Folgenlosigkeit

 Beachten Sie, dass alle diese *cipher suites* Diffie-Hellman-Verfahren mit vergänglichen Schlüsseln zum Schlüsselaustausch verwenden. Das ist Absicht, denn dadurch wird Folgenlosigkeit sichergestellt – selbst wenn ein Angreifer (etwa die NSA) die komplette verschlüsselte Kommunikation mitschneidet, kann er nicht später an die Sitzungsschlüssel gelangen, um sie zu entschlüsseln. (Bei einem Schlüsselaustausch über RSA wäre das ein Problem, weil die zur Generierung des Sitzungsschlüssels nötigen Informationen über die Verbindung geschickt werden und ein Angreifer, der sich nachträglich den privaten Schlüssel des Servers verschafft, die komplette Kommunikation entschlüsseln kann.)

 Kryptografie mit elliptischen Kurven wird von älteren Apache-2.2-Versionen nicht unterstützt, selbst wenn Ihre Version von OpenSSL eigentlich damit umgehen kann. Sie können Apache 2.2 zwar patchen, aber mit einem Upgrade auf Apache 2.4 sind Sie höchstwahrscheinlich besser bedient.

Standardmäßig betrachtet Apache die vom Client vorgeschlagene Liste von *cipher suites* und wählt die erste in dieser Liste, die auch (irgendwo) in seiner Liste vorkommt. Mit »`SSLHonorCipherOrder on`« können Sie Apache dazu bringen, statt dessen

seine eigene Liste zu betrachten und die erste *cipher suite* in dieser Liste zu wählen, die auch (irgendwo) in der Liste des Clients vorkommt. Damit haben Sie es als Betreiber des Servers in der Hand, die sicherste *cipher suite* auszuwählen, und müssen sich nicht auf den Client verlassen.

 *Cipher suites* wählen Sie am besten auf der globalen Ebene der Apache-Konfiguration aus, weil die Einstellung dann automatisch auch für alle virtuellen Server gilt. In virtuellen Servern sollten Sie sie nur überschreiben, wenn das wirklich nötig ist.

Mit `SSLRandomSeed` können Sie angeben, wie Apache den OpenSSL-Zufallszahlengenerator initialisieren soll. OpenSSL benötigt Zufallszahlen von »kryptografischer« Qualität, also solche, die nicht von Angreifern erraten werden können (die typischen »Zufallsgeneratoren« von gängigen Programmiersprachen kommen nicht in Frage). Die Bibliothek enthält einen guten Zufallszahlengenerator, der aber mit hinreichender Entropie versorgt werden muss. Diese Direktive beschreibt die Quellen von Entropie für die erste Initialisierung (*Kontext* ist `startup`) oder für den SSL-Verbindungsaufbau (*Kontext* ist `connect`). Die möglichen Quellen sind

builtin Die »eingebaute« Quelle ist immer vorhanden, liefert aber gerade beim Start sehr wenig Entropie (es stehen wenige »zufällige« Bits zur Verfügung). Wenn möglich, sollte zumindest für `startup` eine bessere Quelle verwendet werden.

file: `<Dateiname>` Der benannten Datei werden zufällige Bytes entnommen. Linux verfügt über zwei »Geräte« namens `/dev/random` und `/dev/urandom`, die Entropie¹ aus dem laufenden System sammeln und sich gut als Quelle eignen. Der Unterschied zwischen den beiden ist, dass `/dev/random` nur so viel Entropie liefert, wie tatsächlich zur Verfügung steht: Wenn Sie 256 zufällige Bytes lesen wollen und die Entropie reicht gerade für 100, dann blockiert der Lesevorgang, bis genug zufällige Bytes zur Verfügung gestellt werden können. `/dev/urandom` tut das nicht.

 Es wird gerne behauptet, dass die Zufallszahlen, die `/dev/urandom` liefert, nicht so gut seien wie die von `/dev/random`. Das ist aber nicht wirklich so. Die beiden erzeugen Zufallszahlen durch einen »kryptografisch sicheren Pseudozufallszahlen-Generator«, der gelegentlich mit Entropie aus dem System »gefüttert« wird. Das System schätzt ab, wieviel Entropie in den Generator eingegangen ist und wieviel durch die Entnahme von Zufallszahlen wieder »verschwindet«. Der einzige Unterschied zwischen `/dev/random` und `/dev/urandom` besteht darin, dass `/dev/random` blockiert, wenn dieser Saldo auf Null geht, während `/dev/urandom` weiterhin (kryptografisch sichere) Zufallszahlen liefert.

exec: `<Programmname>` **und** **egd:** `<Socketname>` Noch zwei Möglichkeiten für Zufallsquellen: ein externes Programm und der *Entropy-Gathering Daemon* (egd). Diese sind unter Linux nicht so wichtig, weil es `/dev/urandom` gibt. Näheres steht in der `mod_ssl`-Dokumentation.

 Sollten Sie einen überdurchschnittlichen Entropie-Bedarf haben, gibt es verschiedene USB-basierte Geräte, die Halbleitereffekte ausnutzen, um große Mengen zufälliger Bits zu generieren (Radioaktiver Zerfall wäre besser, eignet sich aber nicht so gut für den Einsatz in Computern). Diese Geräte werden gerne über `egd` eingebunden.

 Ein externes Programm als Entropiequelle sollten Sie nur beim Systemstart einsetzen (Kontext `startup`). Für den Verbindungsaufbau ist der Mechanismus zu behäbig.

¹Entropie ist an dieser Stelle ein Maß für den »Informationsgehalt« der Zufallszahlen. Ein (fairer) Münzwurf hat eine Entropie von 1 Bit, weil es zwei mögliche Ergebnisse gibt und der Münzwurf eines der beiden »auswählt«.

 Hier können Sie mit einiger Berechtigung davon ausgehen, dass die Voreinstellung Ihrer Linux-Distribution stimmt. Ansonsten verwenden Sie

```
SSLRandomSeed startup /dev/urandom
SSLRandomSeed connect /dev/urandom
```

 `SSLRandomSeed` können Sie nur auf der globalen Ebene der Apache-Konfiguration benutzen.

12.2 Server-Authentisierung

12.2.1 Zertifikate installieren

Damit Apache als HTTPS-Server funktioniert, brauchen Sie ein Zertifikat und den dazugehörigen privaten Schlüssel. Wo Sie diese herkriegen, ist ausführlich in Kapitel 11 beschrieben.

Für beste Ergebnisse brauchen Sie ein Zertifikat, das von einer wohlbekannten Zertifizierungsstelle beglaubigt ist – »wohlbekannt« in dem Sinne, dass die gängigen Browser das Zertifikat der Zertifizierungsstelle enthalten. Wenn Sie also einen Apache-Server von allgemeinem Interesse ans Netz bringen wollen, kommen Sie kaum daran vorbei, die Dienste einer Firma wie Symantec in Anspruch zu nehmen. (Sie können natürlich auch versuchen, den Benutzern Ihrer Seiten das Zertifikat Ihrer eigenen Zertifizierungsstelle zukommen zu lassen; die Installation eines solchen Zertifikats in den Browser ist jedoch möglicherweise vom durchschnittlichen Anwender etwas viel verlangt – insbesondere wenn Sie diesen Anwendern auch noch einbläuen müssen, das nicht blindlings zu tun, sondern sich vorher bei Ihnen von der Authentizität dieses Zertifikats zu überzeugen.)

 Selbstsignierte Zertifikate funktionieren zwar (zumindest wenn Sie Ihre Benutzer dazu kriegen, etwa in Firefox eine »Zertifikatsausnahme« einzurichten), sind aber sicherheitstechnisch problematisch, da Benutzer in der Regel nicht pingelig genug sind, um auf Zertifikatswarnungen angemessen zu reagieren – die Tendenz ist, einfach oft genug auf »OK« zu klicken, bis die Warnungen verschwunden sind, und ansonsten weiterzumachen wie gehabt. Ein Angreifer könnte so als *man in the middle* Ihr selbstsigniertes Zertifikat durch ein anderes ersetzen, und das würde allermeistens gar nicht groß auffallen.

 Für gewisse Anwendungen – etwa die, wo Sie es nur mit einem geschlossenen, überschaubaren Benutzerkreis zu tun haben –, ist es absolut möglich und sinnvoll, selbst eine Zertifizierungsstelle zu betreiben. Im vorigen Kapitel haben Sie ja gesehen, wie Sie eine einrichten können. Dies gilt insbesondere, wenn Sie auch die Benutzer über Zertifikate authentisieren wollen, da Sie durch *do-it-yourself* eine Vertrauensstufe erreichen können, die Sie über ein Symantec-Zertifikat nie schaffen, und dabei auch noch einen Haufen Geld sparen können. In diesem Fall müssen Sie natürlich dafür sorgen, dass den Benutzern das Zertifikat Ihrer Zertifizierungsstelle zur Verfügung steht.

Zertifikat Wenn Sie sich ein Zertifikat besorgen, sollten Sie darauf achten, dass es mindestens einen 2048-Bit-RSA-Schlüssel hat und mit SHA-256 signiert ist.

 Auf 1024-Bit-RSA-Schlüssel oder gar kürzere, DSA-Schlüssel und Ähnliches sollten Sie sich aus Sicherheitsgründen nicht mehr einlassen, da diese inzwischen wahrscheinlich mit »roher Gewalt« gebrochen werden können.

 Die Hersteller der gängigen Browser haben alle signalisiert, mittelfristig Zertifikate, die mit SHA-1 signiert sind, nicht mehr als »sicher« zu akzeptieren. Es gibt Übergangsfristen, aber spätestens Anfang 2017 können Sie mit

solchen Zertifikaten keinen Blumentopf mehr gewinnen. (Von MD5 fangen wir gar nicht erst an.)

 ECDSA ist eine gute Alternative zu RSA und vermutlich der Zug der Zukunft. Ein 256-Bit-ECDSA-Schlüssel ist sicherer als ein 2048-Bit-RSA-Schlüssel, und schneller ist ECDSA außerdem. Allerdings wird ECDSA nur von modernen Browsern unterstützt; ältere Browser kommen damit nicht zu recht. Sie können gleichzeitig Zertifikate mit RSA- und ECDSA-Schlüsseln anbieten, wenn es Ihnen auf die größtmögliche Leistung mit modernen Browsern ankommt, Sie aber gleichzeitig noch ältere Browser unterstützen müssen.

Es spricht nichts dagegen, domain-Validierte Zertifikate zu verwenden, vor allem weil sie wenig bis nichts kosten. Organisationsvalidierte Zertifikate enthalten zwar Informationen über Ihre Organisation, aber da Browser DV- und OV-Zertifikate gleich behandeln und nur ein verschwindender Prozentsatz Ihrer Benutzer sich die Zertifikatsinformationen im Detail anschauen wird, können Sie sich den zusätzlichen Aufwand in der Regel sparen. EV-Zertifikate sollten Sie nur dann verwenden, wenn alle Ihre Konkurrenten auch welche haben und Sie im Vergleich nicht »unsicher« wirken wollen.

Ein Zertifikat sollte für alle DNS-Namen ausgestellt sein, über die der (virtuelle?) Server zu erreichen ist. Ansonsten gibt es Warnungen, wenn jemand Ihre Webseite unter einem Namen aufruft, der nicht vom Zertifikat abgedeckt wird. DNS-Namen

 Viele Web-Präsenzen sind gleichzeitig unter Namen wie `www.example.com` und `example.com` erreichbar. In so einem Fall sollten Sie sicherstellen, dass beide Namen im Zertifikat vorkommen.

 Aufpassen müssen Sie auch, wenn auf derselben IP-Adresse eine HTTPS-Web-Präsenz (etwa `secure.example.com`) über den Port 443 und eine völlig andere HTTP-Web-Präsenz (etwa `www.example.com`) über den Port 80 zugänglich ist. Wenn jemand auf `https://www.example.com/` zugreift, landet er auf `secure.example.com`, aber mit einem falschen Namen, was zu einer Zertifikats-Warnung führen sollte (von der Überraschung, einen ganz anderen Inhalt zu finden, mal abgesehen). Am besten sorgen Sie dafür, dass solche Überlappungen gar nicht erst existieren.

Denken Sie daran, dass Sie höchstwahrscheinlich außer dem Zertifikat für Ihren eigenen Server noch ein oder mehrere »Zwischen-Zertifikate« brauchen, um die Kette von Ihrem Zertifikat zum selbstsignierten Zertifikat der Zertifizierungsstelle in den Browsern zu komplettieren. (Das selbstsignierte Zertifikat der Zertifizierungsstelle brauchen Sie hingegen nicht.) Zwischen-Zertifikate

 Kurze Ketten sind normalerweise besser, aber die kürzeste Kette ist nicht immer die beste. Eine ganz neue Zertifizierungsstelle hat ihr Zertifikat vielleicht in ganz neuen Browsern, aber für ältere Browser kann es nötig sein, denselben Schlüssel in einem Zertifikat zu haben, das von einer längst etablierten Zertifizierungsstelle beglaubigt wurde. An dieser Stelle ist es besser, die längere Kette zu veröffentlichen, weil die kürzeste nur in den neuen Browsern funktionieren würde.

Damit Apache das Server-Zertifikat, den dazugehörigen Schlüssel und allfällige Zwischen-Zertifikate verwenden kann, müssen Sie sie wie folgt mit den Direktiven `SSLCertificateFile`, `SSLCertificateKeyFile` und `SSLCertificateChainFile` angeben: Apache und Zertifikate

```
SSLCertificateFile /etc/apache2/ssl/server.crt
SSLCertificateKeyFile /etc/apache2/ssl/server.key
SSLCertificateChainFile /etc/apache2/ssl/chain.crt
```

 Die Pfadnamen hängen wie üblich von Ihrer Linux-Distribution ab. Schauen Sie gegebenenfalls nach, was auf Ihrem Rechner verwendet wird.

Seit Apache 2.4.8 ist `SSLCertificateChainFile` verpönt. Statt dessen können Sie Zwischen-Zertifikate an die mit `SSLCertificateFile` angegebene Datei anhängen (fangen Sie mit dem Server-Zertifikat an und arbeiten Sie sich die Kette entlang nach oben vor).

 Der Hintergrund dieser Änderung besteht vermutlich darin, die gleichzeitige Benutzung mehrerer Zertifikate (etwa mit RSA- und ECDSA-Schlüsseln) zu vereinfachen. Mit `SSLCertificateChainFile` müssten alle Zertifikate dieselbe Zertifikatskette verwenden, und das ist eine unangenehme Einschränkung.

 Den privaten Schlüssel, auf den `SSLCertificateKeyFile` verweist, können Sie auch an die Zertifikatsdatei anhängen, deren Name in `SSLCertificateFile` steht.

mehrere Zertifikate Sie können mehrere Zertifikate (etwa mit RSA- und ECDSA-Schlüsseln) parallel verwenden, einfach indem Sie mehrere `SSLCertificateFile/SSLCertificateKeyFile`-Paare angeben. (`SSLCertificateChainFile` können Sie allerdings nur einmal haben.)

 Apache kann die Zertifikatskette(n) auch aus Zertifikaten konstruieren, die in einer Datei gesammelt sind, deren Name mit `SSLCACertificateFile` angegeben wurde, oder in einem Verzeichnis abgelegt wurden, dessen Name in `SSLCACertificatePath` steht. Das erhöht allerdings den Aufwand, der beim Aufbau von TLS-Verbindungen getrieben werden muss, und führt potentiell zu Verwirrung, wenn gleichzeitig Clients über Zertifikate authentisiert werden sollen, da diese Direktiven auch dafür verwendet werden, Zertifikate der Zertifizierungsstellen zu benennen, die die Client-Zertifikate beklagt haben.

Passphrase Wenn Sie die Zertifikatsinformation in die Server-Konfiguration aufgenommen haben, müssen Sie Apache neu starten (Neuladen der Konfiguration reicht leider nicht aus). Während des Neustarts müssen Sie die Passphrase für den verschlüsselten privaten Schlüssel des Zertifikats (oder genauer gesagt die Passphrasen für alle verschlüsselten privaten Schlüssel aller Zertifikate) eingeben.

 Eine SUSE-Eigenheit ist, dass der Apache beim Start nur sehr kurz darauf wartet, dass Sie die Passphrase eingeben – zwei Sekunden müssen gemäß Standardeinstellung reichen. Wenn Sie nicht gerade über die Reaktionszeit eines Jagdfliegers und die Fingerfertigkeit eines Konzertpianisten verfügen, könnte das eine gewisse Herausforderung darstellen. Sie tun besser daran, in der Variablen `APACHE_START_TIMEOUT` in der Datei `/etc/sysconfig/apache2` eine längere Frist festzulegen. Je nach der Komplexität Ihrer Passphrase ist ein Wert von 10 bis 15 Sekunden angemessen.

 Wenn es Sie grundsätzlich stört, dass Sie beim Start Ihres Apache-Servers zugegen sein müssen, um die Passphrase einzugeben, können Sie entweder die Verschlüsselung des privaten Schlüssels entfernen (ein Kommando wie

```
$ openssl rsa -in server.key -out server-np.key
Enter pass phrase for server.key: 123456
writing RSA key
$ _
```

erledigt das problemlos) oder die Passphrase anderweitig »eintippen«. Zum Beispiel könnten Sie die Direktive

```
SSLPassPhraseDialog exec:/usr/local/sbin/apache-passphrase
```

definieren und in `/usr/local/sbin/apache-passphrase` ein Skript wie

```
#!/bin/sh
echo 123456
```

hinterlegen. Sie sollten dann dafür sorgen, dass dieses Skript nur für den Benutzer les- und ausführbar ist, der für den Apache verwendet wird (etwa `wwwrun` bei der SUSE oder `www-data` bei Debian GNU/Linux). Die Sicherheit Ihres Systems wird davon selbstverständlich nicht gesteigert (im Gegenteil), aber Sie müssen dies abwägen gegen die Unbequemlichkeit, eventuell kurzfristig tief in der Nacht im Serverkeller Passphrases eintippen zu müssen.

 Natürlich können Sie auch tief in die Tasche greifen und Ihren privaten Schlüssel auf einer Smartcard oder in einem Hardware-Sicherheitsmodul (HSM) ablegen. Das hat den gravierenden Vorteil, dass Angreifer den privaten Schlüssel nicht stehlen können. Die Details gehen allerdings über das hinaus, was im Rahmen dieser Unterlage erklärt werden kann.

Überzeugen Sie sich, dass Ihr Apache auf dem Port 443 lauscht und auf eine Anfrage an `https://www.example.com/` mit der üblichen Indexseite antwortet – in gängigen Browsern sollte an einem zugeklappten Vorhängeschloss oder einem ganzen (nicht zerbrochenen) Schlüssel zu erkennen sein, dass die Seite über eine HTTPS-Verbindung geholt wurde. Test

 Es ist damit zu rechnen, dass Ihr Browser die HTTPS-Verbindung nicht widerstandslos aufbaut – er sollte zumindest annehmen, dass er das Zertifikat des Servers nicht akzeptiert, weil er das Zertifikat der Zertifizierungsstelle nicht zur Verfügung hat und darum die Authentizität des Server-Zertifikats nicht prüfen kann. Normalerweise können Sie als Benutzer da Ausnahmen veranlassen; ansonsten erklärt der nächste Abschnitt, wie Sie Ihrem Browser das Zertifikat Ihrer Zertifizierungsstelle beibringen können.

Übungen

 **12.1** [!2] Besorgen Sie sich ein Server-Zertifikat für `www.example.com` (aus den Beispieldateien für diese Schulungsunterlage oder von Ihrem Trainer) sowie ggf. die anderen Zertifikate für die Zertifikatskette und installieren Sie sie in Ihrem Server. Vergewissern Sie sich, dass Ihr Server auf Zugriffe auf `https://www.example.com/` mit einer verschlüsselten und authentisierten Testseite antwortet.

 **12.2** [2] Vergewissern Sie sich, dass Apache auch funktioniert, wenn das Zertifikat, der private Schlüssel und die anderen Zertifikate der Kette im `SSLCertificateFile` stehen und die Direktiven `SSLCertificateKeyFile` und `SSLCertificateChainFile` gar nicht benutzt werden.

 **12.3** [2] Versuchen Sie, die Eingabe der Passphrase wie im Text gezeigt über `SSLPassPhraseDialog` und ein Shellskript zu automatisieren. Halten Sie das für eine gute Idee?

12.2.2 Browser und Zertifizierungsstellen

Wie eben angedeutet, akzeptieren Browser anstandslos nur solche Zertifikate, die von Zertifizierungsstellen signiert wurden, deren Zertifikate dem Browser zur Verfügung stehen. Das ist auch ganz richtig so, denn durch eine geeignete Auswahl von Zertifizierungsstellen kann der Anwender des Browsers steuern, wem er (blind) vertrauen möchte und wem nicht. Ist an einem Zertifikat, das ein Web-Server präsentiert, etwas auszusetzen, zeigt der Browser eine Fehlermeldung wie in Bild 12.1 an. Diese ist an sich noch nicht besonders aufschlussreich, aber die meisten Browser erlauben Ihnen, den Dingen etwas weiter auf den Grund zu gehen (in Google Chrome zum Beispiel können Sie auf das Schloss-Symbol im URL-

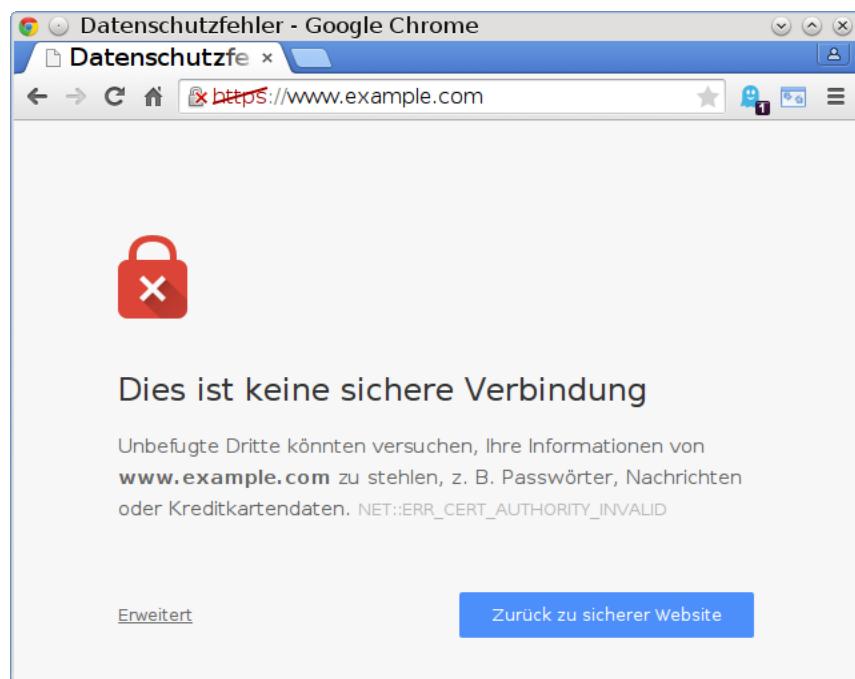


Bild 12.1: Zertifizierungsstelle unbekannt (Google Chrome)

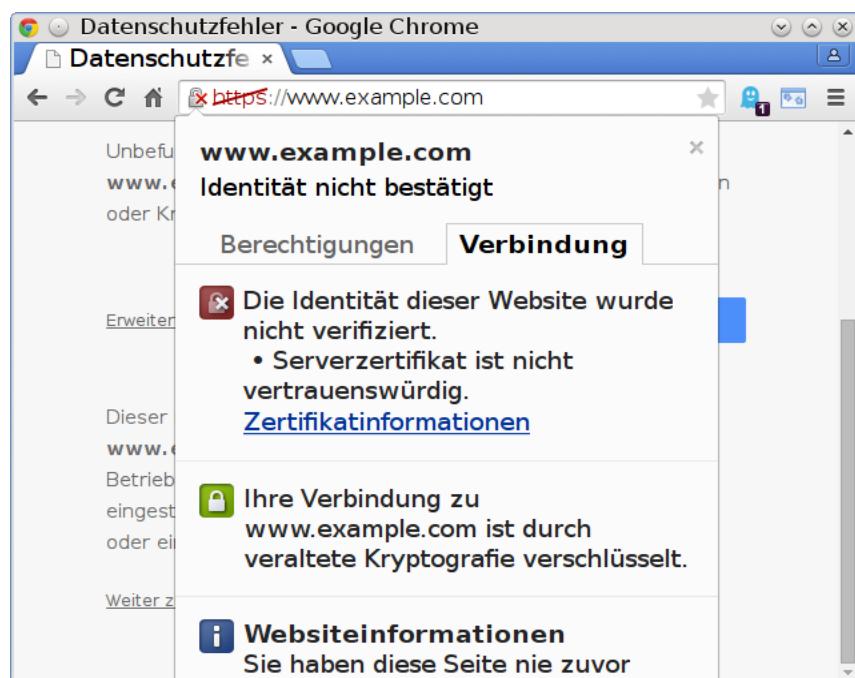


Bild 12.2: TLS-Verbindungsinformationen (Google Chrome)

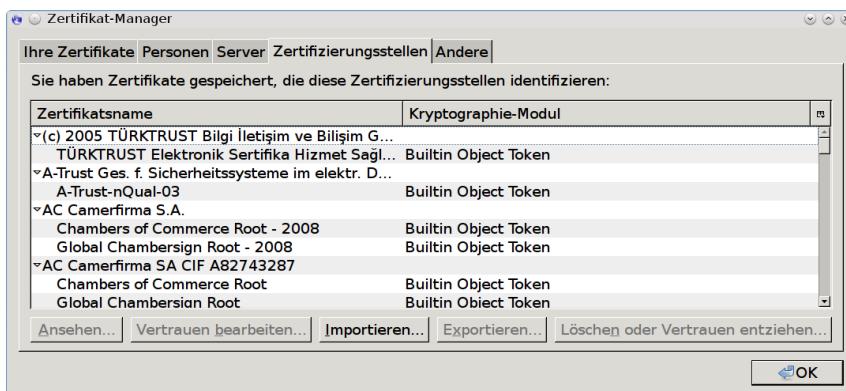


Bild 12.3: Zertifikatsverwaltung in Firefox

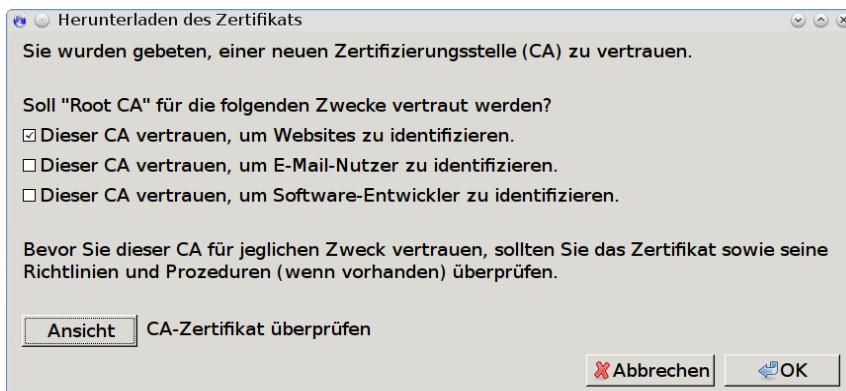


Bild 12.4: Bestätigungsdialog für Zertifizierungsstellen (Firefox)

Fenster klicken und bekommen dann etwas wie Bild 12.2. Mit »Zertifikatinformationen« können Sie dann die Details des Zertifikats inspizieren).

 Sie sollten als Server-Betreiber nicht der absurden Wahnvorstellung anheimfallen, dass »normale« Benutzer sich das Zertifikat Ihres Servers anschauen würden, um festzustellen, wo das Problem ist. Normale Benutzer klicken in dem Fenster mit der Fehlermeldung auf »Erweitert« und dann weiter unten auf »Weiter zu www.example.com (unsicher)«.

 Die Informationen über die TLS-Verbindung stehen Ihnen selbstredend auch dann zur Verfügung, wenn keine Probleme aufgetreten sind.

Damit Ihr Browser Ihr selbstgebasteltes Server-Zertifikat ohne Murren akzeptiert, müssen Sie ihm das selbstsignierte Zertifikat Ihrer Zertifizierungsstelle mitteilen. Die meisten Browser haben einen Konfigurationsdialog für die Zertifikatsverwaltung (Bild 12.3 zeigt den von Firefox – Sie finden ihn bei aktuellen Versionen im »Hamburger-Menü« unter »Einstellungen« und da nicht etwa unter »Sicherheit«, sondern unter »Erweitert«, im Tab »Zertifikate« unter »Zertifikate anzeigen«). Dort können Sie das Zertifikat »importieren«, indem Sie die betreffende Datei auswählen, und es so Ihrem Browser bekanntmachen.

Zertifikatsverwaltung

 Bevor Firefox ein neues Zertifikat einer Zertifizierungsstelle akzeptiert, fragt er erst noch einmal nach, wofür das Zertifikat zu gebrauchen sein soll (Bild 12.4). Insbesondere können Sie hier angeben, welche Arten von Zertifikaten diese Zertifizierungsstelle ausstellen kann. »Websites identifizieren« klingt ungefähr richtig. (Andere Browser machen das ganz ähnlich.)

Wenn Sie das Zertifikat der Zertifizierungsstelle installiert haben, sollte eigentlich alles ohne Probleme funktionieren.

 Als Administrator in einer Firma mit einer eigenen Zertifizierungsstelle stehen Sie natürlich vor der Herausforderung, dafür zu sorgen, dass Ihre Benutzer das Zertifikat Ihrer Zertifizierungsstelle nicht selber installieren müssen (wobei Sie als fauler Mensch das auch nicht persönlich für jeden PC und Browser mit der Hand erledigen wollen). Die geschickteste Methode besteht vermutlich darin, ein lokales Paket für Ihre Linux-Distribution zu erstellen, das das gewünschte Zertifikat enthält und im Trust-Store des Betriebssystems platziert. Dieses Paket können Sie in die (automatische?) Grundinstallation aller Clients aufnehmen, und auch Aktualisierungen lassen sich bequem über Paket-Updates ausrollen.

 Bei Debian-artigen Distributionen genügt es, das betreffende Zertifikat in `/usr/local/share/ca-certificates` abzulegen und (als `root`) das Kommando `update-ca-certificates` auszuführen.

Übungen

 **12.4** [!1] Importieren Sie das selbstsignierte Zertifikat Ihrer Zertifizierungsstelle in Ihren Browser. Können Sie es auch wieder löschen?

 **12.5** [2] (Rechercheaufgabe.) Wo speichert Ihre Linux-Distribution die Zertifikate von Zertifizierungsstellen ab? Wie viele Zertifikate stehen dort? Wo kommt ggf. ein Grundstock an Zertifikaten her, und wie können Software-pakete eigene Zertifikate hinzufügen?

12.2.3 Ressourcen-Konfiguration

Wenn Sie Ihren Apache-Server durch die Installation eines Zertifikats grundsätzlich für HTTPS fit gemacht haben, gibt es noch einige Einstellungen, die Sie machen können, um den Zugriff auf Ihre Inhalte oder Web-Anwendungen zu kontrollieren. Außerdem kommuniziert Apache auf Wunsch diverse wichtige Eigenschaften von HTTP-Anfragen und der Umgebung an CGI-Skripte und andere Ausführungsumgebungen für dynamische Inhalte.

SSLOptions Gesteuert wird dies über die Direktive `SSLOptions`, mit der Sie verschiedene Optionen ein- oder ausschalten können. `SSLOptions` können Sie auf der globalen Ebene oder in `<VirtualHost>`-Blöcken verwenden, aber auch in `<Directory>`-Blöcken oder `.htaccess`-Dateien. Die Direktive wird behandelt wie die `Options`-Direktive: Tauen mehrere `SSLOptions`-Direktiven auf, dann gilt immer diejenige, die am »nächsten« am betreffenden Verzeichnis ist, und diese überschreibt allfällige »weiter entfernte« Direktiven komplett. Nur wenn *jede* angegebene Option ein »+« oder »-« hat, werden so benannten Optionen zu den gerade gültigen Optionen hinzugefügt oder aus dieser Menge entfernt.

Einige der wichtigsten Optionen sind:

StdEnvVars Sorgt dafür, dass die SSL-Umgebungsvariablen für CGI-Skripte und *server-side includes* erzeugt werden. Dies ist eine relativ aufwendige Operation, standardmäßig ausgeschaltet, und sollte daher nur für echte CGI- oder SSI-Anfragen explizit eingeschaltet werden (etwa in einem mit `ScriptAlias` angemeldeten CGI-Verzeichnis). Eine Übersicht über die Variablen finden Sie in Tabelle 12.1.

ExportCertData Stellt das komplette Server-Zertifikat PEM-kodiert in der Umgebungsvariablen `SSL_SERVER_CERT` zur Verfügung. Dies können Sie verwenden, wenn ein CGI-Skript ein Zertifikat genauer unter die Lupe nehmen soll, als die üblichen Umgebungsvariablen das ermöglichen. Da diese Option die Umgebung aufbläht, ist sie standardmäßig ausgeschaltet.

 Es ist nicht ganz offensichtlich, weshalb Sie als Server-Betreiber das Server-Zertifikat genau überprüfen wollen sollten (aber man weiß ja nie). Wichtiger ist, dass `ExportCertData` gegebenenfalls auch ein Client-

Tabelle 12.1: TLS-Umgebungsvariable für CGI-Skripte und Server-Side-Includes

HTTPS	±	Wahr, wenn HTTPS-Anfrage, sonst falsch
SSL_PROTOCOL	\$	Die verwendete Protokollversion (SSLv3, TLSv1, TLSv1.1, TLSv1.2)
SSL_SESSION_ID	\$	Die SSL-Sitzungs-ID (hexadezimal)
SSL_CIPHER	\$	Die verwendete <i>cipher suite</i>
SSL_CIPHER_EXPORT	\$	Wahr, wenn die verwendeten Kryptoverfahren für den Export aus den USA geeignet sind (bzw. waren)
SSL_CIPHER_USEKEYSIZE	0	Anzahl von tatsächlich benutzten Bits im (symmetrischen) Schlüssel
SSL_CIPHER_ALGKEYSIZE	0	Anzahl von prinzipiell möglichen Bits im (symmetrischen) Schlüssel
SSL_COMPRESS_METHOD	\$	Ausgehandelte TLS-Komprimierungsmethode
SSL_VERSION_INTERFACE	\$	Versionsnummer von mod_ssl
SSL_VERSION_LIBRARY	\$	Versionsnummer von OpenSSL
SSL_SERVER_S_DN	\$	Subjekt-DN im Server-Zertifikat
SSL_SERVER_SAN_Email_n	\$	rfc822Name-Einträge in der subjectAltName-Erweiterung des Server-Zertifikats
SSL_SERVER_SAN_DNS_n	\$	dnsName-Einträge in der subjectAltName-Erweiterung des Server-Zertifikats
SSL_SERVER_S_DN_{x509}	\$	Komponente des Subjekt-DN im Server-Zertifikat
SSL_SERVER_I_DN	\$	Aussteller-DN im Server-Zertifikat
SSL_SERVER_I_DN_{x509}	\$	Komponente des Aussteller-DN im Server-Zertifikat
SSL_SERVER_V_START	\$	Gültigkeitsbeginn des Server-Zertifikats
SSL_SERVER_V_END	\$	Gültigkeitsende des Server-Zertifikats
SSL_SERVER_A_SIG	\$	Für die Signatur auf dem Server-Zertifikat verwendete kryptografische Hash-Funktion
SSL_SERVER_A_KEY	\$	Kryptoverfahren, für das der öffentliche Schlüssel im Server-Zertifikat gedacht ist
SSL_SERVER_CERT	\$	Das komplette PEM-kodierte Server-Zertifikat (wenn ExportCertData aktiv ist)
SSL_TLS_SNI	\$	Inhalt der TLS-SNI-Erweiterung (falls vom Client angegeben)

Datentypen: »±« = Boolescher Wert, »\$« = Zeichenkette, »0« = Ganzzahl.

{x509} ist der Name einer X.509-DN-Komponente (C, ST, L, O, OU, CN, T, I, G, S, D, UID, Email).

Seit Apache 2.1 kann {x509} auch ein numerisches Suffix (_n, n ≥ 0) enthalten. Falls die betreffende DN-Komponente mehr als einmal auftritt, können Sie so die einzelnen Werte adressieren (wenn der DN etwa ».../OU=Bla/OU=Fasel/...« enthält, hätte SSL_SERVER_S_DN_OU_0 den Wert Bla und SSL_SERVER_S_DN_OU_1 den Wert Fasel). Ein Name wie SSL_SERVER_S_DN_OU ist äquivalent zu SSL_SERVER_S_DN_OU_0.

Zertifikat (in `SSL_CLIENT_CERT`) sowie die Zertifikatskette des Client-Zertifikats (in `SSL_CLIENT_CERT_n`, für $n \geq 0$) verfügbar macht.

StrictRequire Wenn diese Option aktiv ist, werden Zugriffsbeschränkungen von `mod_ssl` auf jeden Fall umgesetzt. Das heißt, wenn `mod_ssl` entschieden hat, dass eine Ressource nicht zugänglich sein soll (hierzu gleich mehr), dann führt das zur Ablehnung des Zugriffs, selbst wenn mit etwas wie »`Satisfy any`« und einer erfüllten anderen Zugriffsregel der Zugriff ansonsten erlaubt wäre.

OptRenegotiate Es kann passieren, dass für den Zugriff auf bestimmte Ressourcen neue Verschlüsselungsparameter ausgehandelt werden müssen. Zum Beispiel könnte die Ressourcenhierarchie Ihres HTTPS-Servers prinzipiell öffentlich zugänglich sein, während ein Teil eine Client-Authentisierung über ein Zertifikat erfordert. Beim Verbindungsaufbau weiß Apache noch nicht, ob ein Client-Zertifikat angefordert werden muss, also wird zunächst eine gewöhnliche TLS-Verbindung ausgehandelt. Stellt sich später heraus, dass die Anfrage einer Ressource im geschützten Teil gilt, muss Apache eine Neuauhandlung der Verbindungsparameter forcieren und dabei ein Client-Zertifikat anfordern.

Standardmäßig führt jedes Auftreten von `mod_ssl`-Direktiven im Verzeichniskontext (`<Directory>`-Block oder `.htaccess`-Datei) zu einem neuen TLS-Handshake. Mit der Option `OptRenegotiate` können Sie versuchen, die Anzahl dieser Handshakes zu minimieren (wobei die Sicherheit nicht kompromittiert wird). Allerdings kann das die Benutzer verwirren und sollte darum gezielt eingesetzt werden.

LegacyDNStringFormat Bis zur Version 2.3.10 benutzte Apache ein »traditionelles« Format zur Darstellung von DNs in Variablen wie `SSL_SERVER_S_DN`: Die Komponenten wurden durch Schrägstriche getrennt, `C` stand ganz links, und Sonderzeichen wurden nur inkonsistent behandelt. Seit Apache 2.3.11 wird eine Notation verwendet, die sich an [RFC2253] anlehnt: Als Trennzeichen wird ein Komma benutzt, Sonderzeichen werden in UTF-8 dargestellt und/oder mit Backslashes maskiert, und `C` steht ganz rechts. Mit `LegacyDNStringFormat` können Sie aber das Prä-2.3.11-Format erzwingen.

SSLRequireSSL

Wenn Sie in Ihrem Apache-Server HTTPS-Zugriff aktivieren, dann steht der komplette Ressourcenbaum des Servers zunächst sowohl über (gewöhnliches) HTTP als auch über HTTPS zur Verfügung. Sie können aber bestimmte Teile der Ressourcen-Hierarchie nur über HTTPS (und nicht über HTTP) zugänglich machen. Hierzu dient die Direktive `SSLRequireSSL`. Wenn sie in einem `<Directory>`-Block oder einer `.htaccess`-Datei auftaucht, dann sind die Inhalte in und unter dem betreffenden Verzeichnis nur über HTTPS erreichbar, nicht mehr über HTTP.



Trotz `SSLRequireSSL` kann es immer noch möglich sein, ohne HTTPS Zugriff auf eine Ressource zu bekommen, nämlich wenn für die betreffende Ressource »`Satisfy Any`« in Kraft ist. Um das auszuschließen, müssen Sie außer `SSLRequireSSL` noch die SSL-Option `StrictRequire` setzen (siehe oben).



In Apache 2.4 können Sie statt `SSLRequireSSL` auch

`Require ssl`

verwenden. Dies fügt sich besser in die Infrastruktur von `mod_authz_core` ein.

12.2.4 Session Caching

Apache nutzt den TLS-Sitzungsmechanismus, um bei wiederholten Verbindungen zum selben Client die (zeitaufwendige) Aushandlung von Verschlüsselungsparametern abzukürzen. Damit das funktioniert, muss Apache Informationen

über die gerade aktiven TLS-Sitzungen speichern – hauptsächlich weil Browser heutzutage mehrere parallele Verbindungen zum selben Server aufbauen, um zusätzliche Ressourcen abzurufen, und diese Verbindungen im Allgemeinen von verschiedenen Kindprozessen des Servers abgearbeitet werden. Wenn alle Kindprozesse Zugriff auf dieselben TLS-Sitzungsinformationen haben, werden überflüssige TLS-Handshakes vermieden.

Mit der Direktive `SSLSessionCache` können Sie angeben, wie und wo die Sitzungsinformationen abgelegt werden. Es gibt die folgenden Einstellungsmöglichkeiten:

none Keine Speicherung von Sitzungsinformationen. Diese Einstellung ist nicht empfehlenswert, weil sie zumindest zu einem deutlichen Geschwindigkeitseinbruch führt und mit manchen Browsern Probleme macht.

nonenotnull Wie `none`, bis darauf, dass Clients eine nichtleere Sitzungs-ID geschickt bekommen (manche fehlerhaften Clients bekommen sonst Schluckauf).

dbm:<Pfad zu einer DBM-Datei> Speichert Sitzungsinformationen in einer DBM-Datei ab (die allen Apache-Kindprozesse sehen können). Diese Einstellung ist auch nicht empfehlenswert, weil es unter hoher Last zu Schwierigkeiten kommen kann.

shmcb:<Pfad zu einer Datei> Speichert Sitzungsinformationen in einem Shared-Memory-Segment (unter dem angegebenen Dateinamen). Am Ende des Dateinamens kann (in runden Klammern) eine Größe für die »Datei« stehen, etwa wie in

<code>shmcb:/run/apache2/ssl_scache(1024000)</code>

Dies ist die empfohlene Methode. Damit sie funktioniert, muss das Apache-Modul `mod_socache_shmcb` aktiviert sein.

 Laut [Ris14] reicht mit Apache 2.4 ein Megabyte Cache für ungefähr 4000 Sitzungen.

dc:<DC-Spezifikation> Verwendet den `distcache`-Mechanismus, um Sitzungsinformationen so zu speichern, dass Apache-Server auf verschiedenen Rechnern auf sie zugreifen können. Die `<DC-Spezifikation>` ist typischerweise etwas wie `UNIX:/run/apache2/dcsocket` (für ein Unix-Domain-Socket) oder `IP:dc.example.com:9001` (für einen Verweis auf einen entfernten Server). Setzt das Apache-Modul `mod_socache_dc` voraus.

 Dies ist eine von den Direktiven, wo Distributionen in der Regel halbwegs vernünftige Voreinstellungen machen (es läuft meist auf `shmcb`-Irgendwas hinaus).

Die Direktive `SSLSessionCacheTimeout` gibt an (in Sekunden), wie lange Informationen über TLS-Sitzungen gespeichert werden. Der gängige Standardwert 300 (5 Minuten) ist extrem konservativ, und es spricht grundsätzlich nichts dagegen, ihn großzügig zu erhöhen:

<code>SSLSessionCacheTimeout 21600</code>	<i>Sechs Stunden</i>
---	----------------------

 Der Sitzungs-Timeout wird nur während des TLS-Handshake geprüft. Insbesondere werden Informationen über abgelaufene Sitzungen erst aus dem Session Cache verdrängt, wenn Sitzungs-IDs angeschaut werden, nicht wenn die Sitzungen selbst ablaufen.

 Eine zu lange Lebensdauer für TLS-Sitzungen ist auch nicht gesund, da ein Angreifer, der den Session Cache stiehlt, daraus Schlüsselinformationen zurückgewinnen kann. Bei Servern mit wenig Verkehr sollten Sie die Cache-Größe also kleiner halten.



Der Session Cache von Apache wird global für den kompletten Server festgelegt. Dies ist potentiell ein Problem, wenn unterschiedliche Anwendungen auf dem Server laufen, weil ein Angreifer – vor allem bei Sitzungen, die über Client-Zertifikate authentisiert werden – möglicherweise die Zertifikatsprüfung unterlaufen kann, weil die Wiederaufnahme einer Sitzung einen verkürzten Handshake ohne Zertifikatsprüfungen verwendet. Die einzige Abhilfe dagegen sind getrennte Apache-Instanzen.

Sie können das Session Caching von Apache mit »`openssl s_client`« testen, indem Sie die Option `-reconnect` angeben. »`openssl s_client`« baut dann sechsmal hintereinander eine Verbindung zum Server auf. Dabei ist nur die erste Verbindung wirklich neu, die anderen fünf versuchen, die durch die erste Verbindung etablierte Sitzung zu benutzen:

```
$ openssl s_client -connect www.example.com:443 -reconnect
```

Ob das klappt, müssen Sie allerdings aus den üblichen Bergen von Ausgabe herausfriemeln, die das Programm produziert. Wenn Sie nur eine schnelle Antwort wollen, können Sie nach den interessanten Zeilen suchen:

```
$ openssl s_client -connect www.example.com:443 -reconnect \
> </dev/null 2>/dev/null | egrep '^(\New|\Reused)'

New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Reused, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES256-GCM-SHA384
```

TLS Session Tickets



[RFC5077] spezifiziert »TLS Session Tickets«, einen Mechanismus, der die Wiederaufnahme von TLS-Verbindungen ermöglicht, ohne dass Daten auf dem Server zwischengespeichert werden müssen. Das ist zum Beispiel vorteilhaft für Web-Präsenzen, die sehr viele Transaktionen von verschiedenen Benutzern bearbeiten oder Lastausgleich über verschiedene physikalische Server vornehmen müssen. Auch »Embedded«-Server mit sehr wenig Speicher profitieren davon.

Grundidee



Die Grundidee besteht darin, dass ein Client im TLS-Handshake signalisieren kann, dass er Session-Tickets unterstützt; der Server schickt ihm dann ein Paket mit Daten wie der gewählten *cipher suite* und den dazugehörigen Schlüsselinformationen, namentlich das Session-Ticket. Das Session-Ticket ist so verschlüsselt, dass nur der Server es wieder entschlüsseln kann (für den Client ist es nur ein Batzen Bytes). Bei der nächsten Verbindung präsentiert der Client das Session-Ticket, und der Server überprüft dessen Authentizität und übernimmt gegebenenfalls den Inhalt.

Apache und Session-Tickets



Apache unterstützt Session-Tickets (vorausgesetzt, Sie benutzen OpenSSL 0.9.8f oder neuer, was Sie dringend sollten), und das Merkmal ist standardmäßig eingeschaltet (was Sie der Ausgabe von »`openssl s_client`« entnehmen können – sie enthält unter der Überschrift »TLS session ticket« eine hexadezimale Fassung des Tickets). Seit Version 2.4.11 können Sie es mit `SSLSessionTickets` steuern:

```
SSLSessionTickets off
```

oder `on`

(Vorher hätten Sie Apache im Quellcode ändern und neu übersetzen müssen, um Session-Tickets loszuwerden.)

 Mit der Direktive `SSLSessionTicketKeyFile` können Sie einen festen Schlüssel vorgeben, mit dem die Session-Tickets verschlüsselt werden. (Die Direktive enthält genaugenommen den Namen der Datei, in der der Schlüssel steht.) Das ist sinnvoll, wenn Sie parallel mehrere Apache-Instanzen auf verschiedenen Servern betreiben und alle mit den Sitzungsinformationen umgehen können müssen – wenn Sie einen einzelnen Server haben, dann sollten Sie keinen festen Schlüssel verwenden, sondern sich auf die zufälligen Schlüssel verlassen, die `mod_ssl` von sich aus erzeugt.

 Apache erzeugt zufällige Schlüssel für Session-Tickets immer beim Start. Sie sollten Apache also mindestens einmal am Tag neu starten, damit ein neuer zufälliger Schlüssel erzeugt und dadurch die Folgenlosigkeit gesichert wird. Ein »`apache2ctl graceful`« reicht aus.

 Eine Datei, die Sie mit `SSLSessionTicketKeyFile` benutzen wollen, muss 48 zufällige Bytes enthalten, die Sie sich am besten mit einem Kommando wie

```
$ dd if=/dev/urandom of=/run/apache2/server.tkey bs=1 count=48
```

oder (mit OpenSSL)

```
$ openssl rand -out /run/apache2/server.tkey 48
```

besorgen. Auf eine solche Datei sollten Sie so gründlich aufpassen wie auf den privaten Schlüssel für ein Zertifikat. Außerdem sollten Sie sie oft (etwa einmal täglich) neu anlegen, weil nur dadurch alte Session-Tickets ungültig gemacht werden können (im Moment unterstützt OpenSSL keine Lebensdauer für Session-Tickets), und danach den Server neu starten.

 Wenn Ihr Server(-Cluster) unterschiedliche Web-Präsenzen unterstützt, ist es sinnvoll, für jede davon einen eigenen Schlüssel zu vergeben, damit Angreifer nicht versuchen können, mit einem Session-Ticket von Anwendung X auf Anwendung Y zuzugreifen. Sie können `SSLSessionTicketKeyFile` in einem `<VirtualHost>`-Block verwenden.

12.2.5 Virtuelle Server

Apache kann mehrere virtuelle Server gleichzeitig verwalten, die HTTPS benutzen. Allerdings sind dabei einige Besonderheiten zu beachten, die aus dem Umstand herrühren, dass Apache schon beim TLS-Verbindungsaufbau das korrekte Zertifikat für den virtuellen Server liefern muss. Bei namensbasierten virtuellen Servern kann Apache aber erst am Wert der `Host`-Kopfzeile erkennen, welcher virtuelle Server tatsächlich gemeint ist – aber da muss die Verbindung längst stehen (sonst könnte der Browser ja nichts schicken, auch keine `Host`-Zeile). Als Server-Betreiber haben Sie die folgenden Möglichkeiten:

- Am einfachsten ist es, jedem virtuellen Server eine eigene IP-Adresse zuzuordnen. Dann ist von Anfang an klar, welches Zertifikat gebraucht wird. Allerdings ist es (jedenfalls bis zur flächendeckenden Einführung von IPv6) möglicherweise schwierig, IP-Adressen in der geforderten Anzahl zu bekommen – wenn es nur um eine Handvoll geht, dann sollte Ihr Provider allerdings mit sich reden lassen. Dieser Ansatz funktioniert immer. IP-Adresse
- Solange alle Server in derselben Domain sind, können Sie ein »Wildcard-Zertifikat« verwenden, also eins, das für etwas wie `*.example.com` ausgestellt ist. Allerdings ist unklar, wie so ein Zertifikat genau interpretiert werden soll, und es ist auch möglich, dass es zur Verifikation von mehr Servern als erwünscht verwendet werden kann. Wir raten von diesem Ansatz ab. Wildcard-Zertifikat

subjectAltName

- Sie können Zertifikate erzeugen, die für mehr als einen Servernamen gelten. Gemäß [RFC2459] ist es möglich, weitere Servernamen in der `subjectAltName`-Erweiterung eines Zertifikats zu hinterlegen. Dazu muss die OpenSSL-Konfiguration für die Erzeugung des CSR etwas enthalten wie

```
[req]
req_extensions = v3_req
[v3_req]
subjectAltName = DNS:foo.example.com, DNS:bar.example.net
```

Sie sollten außerdem einen dieser Namen als `CN` für das Zertifikat angeben, damit den Formalien Genüge getan ist. Dieser Ansatz funktioniert jedenfalls für Apache mit OpenSSL und allen gängigen Browsern. Allerdings ist es lästig, dass alle Namen sich dasselbe Zertifikat teilen und darum ein neues Zertifikat ausgestellt werden muss, wenn sich ein Name ändert, wegfällt oder ein neuer dazukommt.

Server Name Indication

- In [RFC4366, Abschnitt 3.1] wird eine TLS-Erweiterung namens »Server Name Indication« (SNI) standardisiert, mit der ein Client schon beim Aufbau einer Verbindung bekanntgeben kann, mit welchem (virtuellen) Server er reden möchte – sozusagen eine vorgezogene `Host`-Zeile. Der Server kann dann sofort das richtige Zertifikat liefern. Die Unterstützung dafür ist auf der Browser-Seite relativ gut (die aktuellen Versionen von Firefox, Opera, Internet Explorer, Google Chrome und Safari spielen alle mit – unter Windows allerdings nur mit Vista und späteren Versionen von Windows 7, nicht Windows XP). Auf der Server-Seite kann Apache SNI seit der Version 2.2.12. Dies ist eindeutig der Zug der Zukunft; ob Sie SNI in der Praxis einsetzen wollen oder können, hängt davon ab, wie sehr Sie Rücksicht auf Seitenbesucher nehmen wollen, die immer noch Windows XP benutzen².



Ein potentielles Problem von SNI ist, dass der gewünschte Servername während des TLS-Handshakes im Klartext übertragen wird. Ein Angreifer könnte daraus gewisse Schlüsse ziehen, selbst wenn er den Rest der Kommunikation nicht mehr verfolgen kann. TLS 1.3 wird das vermutlich reparieren.

In der Apache-Konfiguration können Sie wie üblich mehrere `<VirtualHost>`-Blöcke vorsehen. Bei mehreren IP-basierten virtuellen Servern oder SNI bekommt jeder Server eine eigene Zertifikat/Schlüssel-Kombination mit `SSLCertificateFile`- und `SSLCertificateKeyFile`-Direktiven; bei einem Zertifikat mit `subjectAltName` können Sie entweder in mehreren `<VirtualHost>`-Blöcken auf das Zertifikat (und den Schlüssel) verweisen oder einen `<VirtualHost>`-Block mit `ServerAlias` einsetzen. Ansonsten entspricht die Konfiguration der für virtuelle Server üblichen Vorgehensweise.



Wenn Sie SNI verwenden und sichergehen wollen, dass Clients, die kein SNI unterstützen, abgewiesen werden, dann können Sie die Direktive `SSLStrictSNIVHostCheck` verwenden. Wenn Sie im Standard-`<VirtualHost>`-Block (also dem ersten in der Konfiguration)

```
SSLStrictSNIVHostCheck on
```

setzen, dann werden Clients, die kein SNI können, daran gehindert, mit *irgendeinem* virtuellen Server auf dieser IP-Adressen-Port-Kombination Verbindung aufzunehmen. Sie können die Direktive statt dessen auch in einzelnen `<VirtualHost>`-Blöcken setzen und damit nicht SNI-fähige Clients gezielt von diesen virtuellen Servern ausschließen.

²Microsoft hätte zweifellos dafür sorgen können, dass auch Windows XP nachträglich noch SNI unterstützen lernt. Auf der anderen Seite wäre Microsoft das gute alte XP sicher lieber früher als später losgeworden, so dass an solchen lebenserhaltenden Maßnahmen in Redmond, Washington, kein übermäßiges Interesse bestanden haben dürfte. Je unbequemer man den XP-Benutzern das Leben machen kann, desto größer die Chance, dass sie doch mal auf etwas wie Windows 7 hochrüsten.

 Sie sollten sicherstellen, dass Besucher, die Ihren Server über Fantasie-Namen ansprechen (mit /etc/hosts oder einem eigenen DNS-Server kein Problem) nicht an »echte« Web-Präsenzen herankommen, da deren Browser die auf diese Weise gefundenen Ressourcen dem Fantasie-Namen zuordnen und das möglicherweise für Angriffe auszunutzen ist. Am besten definieren Sie immer den ersten namensbasierten virtuellen Server für eine gegebene Kombination aus IP-Adresse und Portnummer so, dass für alle Server SNI erzwungen wird (siehe oben) und alle Zugriffe auf eine Fehlerseite mit dem Code 404 (Ressource nicht gefunden) umgeleitet werden. (Windows-XP-Benutzer schauen dann mitunter in die Röhre, aber die verdienen es nicht besser – Linux existiert.)

12.2.6 Apache und TLS-Angriffe

Einige gängige Angriffe auf TLS haben in den letzten Jahren Schlagzeilen gemacht. Hier beschreiben wir kurz, wie Apache damit umgeht bzw. was Sie unternehmen können oder sollten.

Unsichere Neuverhandlung Das Problem, bei dem ein Angreifer eine TLS-Verbindung zu einem Server anfangen und damit beliebiges Material vor eine Client-Anfrage stellen kann, etwa um die Anfrage selbst zu verfälschen oder Authentisierungsdaten zu stehlen, ist größtenteils gelöst. Seit Version 2.2.15 unterstützt Apache sichere Neuverhandlung nach [RFC5746] und geht auf die vorige clientseitige Neuverhandlung nicht mehr ein. Apache 2.4 war überhaupt nie anfällig.

 Es kann immer noch Probleme mit Clients geben, die keine sichere Neuverhandlung gemäß [RFC5746] unterstützen: Ein Angreifer könnte eine Verbindung zum Server aufbauen und dann vom Server aus eine (unsichere) Neuverhandlung auslösen, um den Client zu kompromittieren.

 Sie können in Apache herausfinden, ob der Client sichere Neuverhandlung unterstützt, indem Sie die Umgebungsvariable `SSL_SECURE_RENEG` inspizieren.

BEAST Gegen BEAST – den Angriff auf vorhersagbare IVs bei blockorientierten Verschlüsselungsverfahren (wie AES) im CBC-Modus – können Sie auf der Server-Seite nicht wirklich etwas unternehmen, da der Angriff voraussetzt, dass der Angreifer den Klartext kontrollieren kann, und das geht nur auf der Clientseite (im Browser).

 Natürlich könnten Sie als Serverbetreiber die Verwendung von RC4 erzwingen (jedenfalls solange der Browser mitspielt), das als stromorientiertes Verfahren gegen BEAST immun ist, aber das wollen Sie nicht, weil RC4 höchstwahrscheinlich unsicher ist.

 BEAST wird heutzutage so weit wie möglich clientseitig unterbunden. Sie können als Serverbetreiber helfen, indem Sie TLS 1.2 unterstützen, das gegen BEAST nicht anfällig ist.

CRIME & Co. Der CRIME-Angriff nutzt die transparente Komprimierung von Daten aus, die TLS vornimmt, um Klartext (etwa Cookies) zu erraten. Auch CRIME ist ein clientseitiger Angriff, den Sie entschärfen können, indem Sie TLS-Komprimierung ausschalten – im Kontext eines Web-Servers können Sie ohnehin Komprimierung gezielt einsetzen, etwa indem Sie HTML-, JavaScript- und CSS-Ressourcen komprimieren, aber JPEG-Bilder nicht (denn die sind sowieso schon so sehr komprimiert, dass Apache oder TLS da nicht mehr arg viel herausholen können, wenn überhaupt), so dass der Verlust der TLS-Komprimierung zu verschmerzen ist.

Mit

```
SSLCompression off
```

können Sie Komprimierung auf TLS-Ebene explizit ausschalten, jedenfalls solange Sie Version 2.4.3 (oder später) von Apache verwenden. In Apache 2.4.3 war die Komprimierung standardmäßig eingeschaltet, seit Version 2.4.4 ist sie standardmäßig aus (so dass Sie eigentlich nichts mehr machen müssen).

 Ältere Versionen von Apache haben TLS-Komprimierung standardmäßig entweder unausschaltbar aktiviert (bis Version 2.2.23 bzw. 2.4.2) oder sind vom Distributor gepatcht worden, um sie zu deaktivieren.

 Selbst mit »`SSLCompression off`« wird die Komprimierung nur deaktiviert, wenn Sie mindestens OpenSSL 0.9.8 verwenden. Bei älteren Versionen von OpenSSL ist nicht garantiert, dass die Komprimierung sich überhaupt abschalten lässt.

 Sie können `SSLCompression` in `<VirtualHost>`-Blöcken benutzen, sofern Sie OpenSSL 1.0.0 (oder später) einsetzen. Allerdings sollten Sie es auf der globalen Ebene ausschalten und dann nicht wieder einschalten.

POODLE POODLE ist ein Angriff auf SSL 3.0, der einem Angreifer die byteweise Rückgewinnung von Klartext gestattet (lohnendes Ziel sind da wieder mal Authentisierungsinformationen wie Cookies). Im Zeitalter von TLS sollte das eigentlich alte Geschichtete sein, bis darauf, dass viele Web-Server von TLS auf SSL 3.0 zurückfallen, wenn mit dem Client keine TLS-Verbindung ausgehandelt werden kann. Sie sollten, wenn Sie es nicht zwingend mit alten möhrigen Browsern zu tun haben, die Unterstützung von SSL 3 in Ihrem Server komplett verhindern, indem Sie das Protokoll ausschalten:

```
SSLProtocol all -SSLv3
```

FREAK Der FREAK-Angriff wurde im März 2015 dokumentiert und erlaubt einem Angreifer, einen verwundbaren Browser und Server dazu zu zwingen, schwache Verschlüsselung zu verwenden, die der Angreifer dann realistisch mit »brutaler Gewalt« brechen kann [FREAK]. Sie brauchen sich über FREAK auf Ihrem Server keine Sorgen zu machen, solange Sie unsere Empfehlungen über die Auswahl von *cipher suites* beachten (siehe Abschnitt 12.1.2) und die für »Export« geeigneten *cipher suites* von vornherein sperren.

Clients  Sicherheitshalber sollten Sie aber dafür sorgen, dass auch Ihre Clients nicht anfällig gegen FREAK sind. Alle modernen Versionen von Chrome und Firefox sollten OK sein (jedenfalls auf PCs), für Internet Explorer und diverse mobile Browser gibt es Patches. Über Seiten wie <https://freakattack.com/clienttest.html> können Sie herausfinden, ob Sie verwundbar sind.

 Wenn Sie denken, Sie verwenden einen sicheren Browser, aber der Test trotzdem das Gegenteil behauptet, dann kann das daran liegen, dass ein verwundbares Anti-Viren-Programm oder ein Proxy-Server sich zwischen Sie und den Server drängelt. In diesem Fall braucht jene Software eine Fehlerkorrektur.

12.2.7 Protokollierung

Über die normalen Zugriffs- und Fehlerprotokolle von Apache hinaus kann es sich lohnen, Protokolle zu erstellen, die den TLS-Zugang betreffen. Daraus können Sie wertvolle Schlüsse für den Serverbetrieb ziehen:

- Wenn Sie die verwendeten Protokollversionen und *cipher suites* erheben, können Sie Aussagen darüber machen, welche Standards Ihre Clients unterstützen. Das hilft Ihnen dabei, veraltete Protokollversionen und Verschlüsselungsverfahren außer Dienst zu stellen.

 Wir hatten Ihnen zum Beispiel empfohlen, SSL 3.0 und RC4 zu deaktivieren; wenn Sie sich nicht so recht trauen, weil Sie befürchten, damit Clients vor den Kopf zu stoßen, die nichts Anderes können, dann können Sie die Zugriffe eine Weile beobachten und prüfen, ob das in der Praxis überhaupt ein Problem darstellt.

- Sie können aus einem entsprechenden Protokoll auch ersehen, ob Mechanismen wie das Session Caching von Apache funktionieren wie gewünscht, und gegebenenfalls die Größe des Session Cache optimieren.

Sie können ein SSL-Protokoll erstellen, indem Sie mit `CustomLog` ein geeignetes Protokollformat definieren:

<pre>SSLOptions +StdEnvVars CustomLog /var/log/apache2/ssl-access.log \ "%t %h %k %X %{SSL_SESSION_RESUMED}e %{SSL_SESSION_ID}e\ %{SSL_PROTOCOL}e %{SSL_CIPHER}e"</pre>	<i>Definiere SSL-Umgebungsvariable</i>
---	--

`%t` ist dabei der Zeitpunkt des Zugriffs und `%h` der Rechnername des Clients (oder, sinnvollerweise aus Effizienzgründen, dessen IP-Adresse – setzen Sie `HostnameLookups` auf `off`). `%k` liefert die Anzahl der Anfragen, die über diese Verbindung abgehandelt wurden – neue Verbindungen haben hier eine Null, aber in der Regel werden nur für neue Verbindungen TLS-Parameter ausgehandelt. `%X` ist »+« für Verbindungen, die nach dem Senden der Antwort offen gehalten werden, bei einem »-« wird die Verbindung geschlossen und ein »x« steht für einen Verbindungsabbruch, bevor die Antwort komplett geschickt werden konnte.

 Wenn, wie üblich, etliche Anfragen und Antworten über dieselbe TLS-Verbindung gehen, bekommen Sie eine Zeile im Protokoll pro Anfrage, nicht pro Verbindung. Das bläht das Protokoll auf, aber so sehen Sie wenigstens, dass Ihre Verbindungen mehrfach benutzt werden (und in welchem Umfang).

Ansonsten protokolliert dieser Vorschlag noch die Meldung `Initial` oder `Resumed` für neue bzw. wiederaufgenommene Sitzungen, die TLS-Sitzungs-ID (allerdings nur bei wiederaufgenommenen Sitzungen), die TLS-Protokollversion und die *cipher suite*.

 Die Variable `SSL_SESSION_RESUMED` steht nur in Apache 2.4 zur Verfügung, nicht in älteren Versionen.

 `mod_ssl` definiert für jede Anfrage zwei »Anmerkungen« (*request notes*), die mit »%{<Name>}« protokolliert werden können:

`ssl-access-forbidden` hat den Wert 1, wenn der Zugriff über eine `SSLRequire`- oder `SSLRequireSSL`-Direktive verhindert wurde, sonst 0.

`ssl-secure-reneg` hat den Wert 1, wenn für die aktuelle Verbindung TLS verwendet wurde und sowohl die von Apache verwendete OpenSSL-Version als auch der Client sichere Neuverhandlung gemäß [RFC5746] unterstützen. Wenn der Client keine sichere Neuverhandlung unterstützt, hat die Anmerkung den Wert 0; wenn die OpenSSL-Version von Apache keine sichere Neuverhandlung unterstützt oder für die aktuelle Verbindung TLS gar nicht verwendet wurde, wird die Anmerkung überhaupt nicht gesetzt.

 Zur Vereinfachung definiert `mod_ssl` den Formatschlüssel »%{...}x« für Protokollnachrichten, der beliebige von irgendwelchen Modulen definierte Variable expandiert.

 Alte Versionen von `mod_ssl` kannten spezielle Protokollfunktionen der Form »%{<Variable>}c« (wie *cryptography*). Diese werden aus Kompatibilitätsgründen noch unterstützt, aber die moderne Methode mit den Umgebungsvariablen ist günstiger.

12.3 Client-Authentisierung mit Zertifikaten

12.3.1 Server-Konfiguration

Apache und `mod_ssl` erlauben nicht nur die Authentisierung des Servers, sondern auch die Authentisierung von Clients mit X.509-Zertifikaten. Der Ansatz ist sicherer als das gängige Verfahren über Benutzernamen (oder E-Mail-Adressen) und Kennwörter, da Benutzer zuerst ein Zertifikat (nebst dem dazugehörigen privaten Schlüssel) in ihrem Browser installieren müssen. Allerdings ist dies mit einem erheblichen zusätzlichen Aufwand verbunden, weswegen sich das Verfahren nicht auf breiter Front durchgesetzt hat.

 Einer der Haupthaken an der Sache ist, dass die Benutzer strenggenommen ihre eigenen CSRs erzeugen müssen, um die Vertraulichkeit ihrer privaten Schlüssel sicherzustellen.

 Alternativ können Sie Smartcards verteilen, die die Zertifikate (und privaten Schlüssel) bereits enthalten. Sie sind dann von außen nicht zugänglich und (vermutlich) sicher.

Der Server muss in der Lage sein, die Gültigkeit von Client-Zertifikaten zu überprüfen. Dazu benötigt er das Zertifikat der Zertifizierungsstelle, die die Client-Zertifikate beglaubigt hat.

 Wenn Sie eine der heute üblichen mehrstufigen Zertifizierungsstellen verwenden, dann müssen Sie ihm alle Zertifikate der Kette von den Client-Zertifikaten bis zum selbstsignierten Zertifikat der Zertifizierungsstelle zur Verfügung stellen.

 Sie brauchen die Zertifikate von Zertifizierungsstellen aus der Kette, aber Sie brauchen *nicht* alle Client-Zertifikate. (Das sollten Sie als beruhigend empfinden.)

Die Zertifikate können Sie entweder in einer Datei aneinanderhängen und dem Apache-Server den Namen dieser Datei mit der Direktive `SSLCACertificateFile` bekanntgeben, oder Sie kopieren die Zertifikate in ein Verzeichnis, dessen Namen Sie in einer `SSLCACertificatePath`-Direktive unterbringen.

 Beachten Sie, dass Sie in dem betreffenden Verzeichnis das Kommando

```
$ c_rehash
```

ausführen müssen, um die »gehaschten« Dateinamen anzulegen, die OpenSSL braucht, um die Zertifikate zu finden. (Sie können das Verzeichnis auch als Parameter übergeben.)

 Weder die `SSLCACertificateFile`- noch die `SSLCACertificatePath`-Methode hat klare Vorteile gegenüber der jeweils anderen. Das Sammeln aller Zertifikate in einer großen Datei macht die Konfiguration übersichtlicher, aber das Inkraftsetzen neuer Zertifikate oder das Streichen nicht mehr gebrauchter

sind relativ mühselig. Beim Zertifikatsverzeichnis ist das wiederum einfacher – Sie müssen nur Dateien umkopieren oder löschen und `make` aufrufen –, aber dafür sorgen die `hash filenames` für eine gewisse Unordnung, und sie müssen ja auch aktuell sein. Im übrigen spricht nichts dagegen, beide Ansätze gleichzeitig zu benutzen und manche Zertifikate in einer großen Datei und andere in einem Verzeichnis unterzubringen.

Damit der Server beim Browser tatsächlich ein Client-Zertifikat anfordert (und überprüft), müssen Sie die `SSLVerifyClient`-Direktive verwenden. Die möglichen Werte von `SSLVerifyClient` sind:

none Es wird kein Client-Zertifikat verlangt.

optional Ein Client-Zertifikat ist nicht nötig, wird aber angenommen, wenn der Browser eins schickt.

require Der Client *muss* ein Zertifikat vorlegen

optional_no_ca Es ist kein Client-Zertifikat nötig, und wenn der Browser eins schickt, muss es nicht verifizierbar sein.

Wirklich sinnvoll sind nur `none` und `required`, ab und zu auch `optional`.



Wenn Sie »`SSLVerifyClient optional`« benutzen, dann können Sie mit

`Require ssl-verify-client`

den Zugriff auf bestimmte Ressourcen nur dann zulassen, wenn der Client sich tatsächlich über ein Zertifikat authentisiert hat.

Die `SSLVerifyClient`-Direktive kann in einem serverweiten Kontext (global oder in einem `<VirtualHost>`-Block) oder in der Konfiguration für ein Verzeichnis auftreten; im letzteren Fall wird nach dem Lesen der HTTP-Anfrage eine TLS-Neuverhandlung erzwungen.



Erinnern Sie sich an »`SSLOptions OptRenegotiate`«, wenn Sie die TLS-Handshakes optimieren wollen.

Außerdem wichtig ist die Direktive `SSLVerifyDepth`. Damit können Sie bestimmen, wie lang eine Zertifikats-Kette jenseits des zu prüfenden Client-Zertifikats maximal sein darf, damit Client-Zertifikate noch als authentisch gelten. Die Voreinstellung ist 1; Client-Zertifikate müssen also direkt von der Zertifizierungsstelle beglaubigt worden sein. Mit »`SSLVerifyDepth 0`« wären nur selbstsignierte Client-Zertifikate zulässig (ziemlicher Unfug), während mit »`SSLVerifyDepth 3`« die Zertifizierungsstelle ein Zertifikat für eine untergeordnete Zertifizierungsstelle ausstellen könnte, diese untergeordnete Zertifizierungsstelle könnte wiederum eine Unter-Unter-Zertifizierungsstelle einrichten, und von dieser ausgestellte Client-Zertifikate würden noch als authentisch gelten.



Kürzer als verlangt darf die Zertifikatskette natürlich sein, nur halt nicht länger.

Betrachten Sie als Beispiel den folgenden Konfigurations-Ausschnitt:

```
SSLVerifyClient require
SSLVerifyDepth 2
SSLCACertificatePath /usr/local/etc/ca-certs
```

(wir nehmen an, dass in `/usr/local/etc/ca-certs` die Zertifikate aus der Zertifikatskette für Client-Zertifikate stehen). Zum Testen können Sie zum Beispiel mit »`openssl s_client`« eine Verbindung zum Server aufbauen:

```
$ openssl s_client -connect www.example.com:443 \
> -cert client.crt -key client.key
```

Dabei sind `client.crt` und `client.key` das Client-Zertifikat und der dazugehörige private Schlüssel. Ohne das Client-Zertifikat sollte der Server die Verbindung ablehnen.

 Von Rechts wegen sollte der Client auch das Zertifikat des Servers prüfen (dafür braucht er das selbstsignierte Zertifikat der Zertifizierungsstelle). Das haben wir hier aus Gründen der Übersichtlichkeit weggelassen.

12.3.2 Client-Zertifikate im Browser installieren

Damit HTTPS mit Client-Zertifikaten richtig Spaß macht, wollen Sie wahrscheinlich einen Browser benutzen und nicht »`openssl s_client`«. Unsere nächste Herausforderung besteht also darin, das Client-Zertifikat in den Browser zu integrieren.

PKCS#12 Die meisten Browser erwarten Client-Zertifikate im **PKCS#12**-Format, das Sie mit OpenSSL wie folgt erzeugen können, wenn Sie über das Zertifikat und den privaten Schlüssel im PEM-Format verfügen:

```
$ openssl pkcs12 -export -in client.crt -inkey client.key \
> -out client.p12
Enter pass phrase for client.key:foobar
Enter Export Password:abc123
Verifying - Enter Export Password:abc123
```

Exportkennwort Hierbei werden Zertifikat und privater Schlüssel zusammengefügt und gemeinsam mit einem »Exportkennwort« verschlüsselt. Das Exportkennwort können Sie frei wählen; es muss nichts mit dem schon für den privaten Schlüssel vergebenen Kennwort zu tun haben.

 Der Browser braucht Zugriff nicht nur auf das Zertifikat, sondern auch auf den dazugehörigen privaten Schlüssel – das Zertifikat könnte ja jeder, dem es irgendwann mal in die Hände gefallen ist, einem Server präsentieren. Der Server muss sich also vergewissern, dass am anderen Ende der Verbindung wirklich der rechtmäßige Inhaber des Zertifikats sitzt, und das geht über ein *challenge-response*-Verfahren, bei dem der Browser etwas mit dem öffentlichen Schlüssel Verschlüsseltes entschlüsseln muss. Und dafür ist bekanntlich der private Schlüssel nötig.

 Die Endung `.p12` für Dateien mit PKCS#12-Paketen ist nicht vorgeschrieben, aber üblich. Es empfiehlt sich, sie zu benutzen, weil die Browser beim Zertifikatsimport davon ausgehen, dass die Dateien so heißen.

Das PKCS#12-Paket mit dem Zertifikat und dem privaten Schlüssel können Sie über den Zertifikatsmanager Ihres Browsers importieren.

 In Firefox wählen Sie dazu die Karte »Ihre Zertifikate« und klicken auf die Schaltfläche »Importieren ...«. Danach bekommen Sie die Gelegenheit, die `.p12`-Datei auszuwählen und werden nach dem Exportkennwort gefragt, das Sie beim Anlegen des PKCS#12-Pakets angegeben haben. In anderen Browsern funktioniert das ähnlich.

12.3.3 Zugriffsregeln

Wenn ein Browser ein gültiges Zertifikat geschickt hat und der Anwender so authentisiert wurde, kann der Server ihm Zugriffsrechte auf bestimmte Ressourcen einräumen oder versagen. Diese Zugriffsrechte werden – in Analogie zur

```

⟨Ausdruck⟩ ← true | false
| ! ⟨Ausdruck⟩                                Logische Negation
| ⟨Ausdruck⟩ && ⟨Ausdruck⟩                  Logische Und-Verknüpfung
| ⟨Ausdruck⟩ || ⟨Ausdruck⟩                  Logische Oder-Verknüpfung
| ( ⟨Ausdruck⟩ )
| ⟨Vergleich⟩

⟨Vergleich⟩ ← ⟨Wort⟩ == ⟨Wort⟩ | ⟨Wort⟩ eq ⟨Wort⟩
| ⟨Wort⟩ != ⟨Wort⟩ | ⟨Wort⟩ ne ⟨Wort⟩
| ⟨Wort⟩ < ⟨Wort⟩ | ⟨Wort⟩ lt ⟨Wort⟩
| ⟨Wort⟩ <= ⟨Wort⟩ | ⟨Wort⟩ le ⟨Wort⟩
| ⟨Wort⟩ > ⟨Wort⟩ | ⟨Wort⟩ gt ⟨Wort⟩
| ⟨Wort⟩ >= ⟨Wort⟩ | ⟨Wort⟩ ge ⟨Wort⟩
| ⟨Wort⟩ in { ⟨Wortliste⟩ }
| ⟨Wort⟩ in PeerExtList( ⟨Wort⟩ )
| ⟨Wort⟩ =~ ⟨RegEx⟩ | ⟨Wort⟩ !~ ⟨RegEx⟩

⟨Wortliste⟩ ← ⟨Wort⟩ | ⟨Wortliste⟩ , ⟨Wort⟩

⟨Wort⟩ ← ⟨Zahl⟩                                Folge von 0...9
| ⟨Zeichenkette⟩                            Etwas in "..."
| ⟨Variable⟩ | ⟨Funktion⟩

⟨Variable⟩ ← %{ ⟨VarName⟩ }

⟨Funktion⟩ ← ⟨FName⟩ ( ⟨Argumente⟩ )

```

Der ⟨VarName⟩ benennt eine der gültigen Umgebungsvariablen, die der Apache für Anfragen erzeugt (Kapitel D) oder eine der zusätzlichen TLS-Variablen in den Tabellen 12.1 oder 12.2. – Die gültigen Funktionsnamen, die Sie für ⟨FName⟩ angeben können, stehen in Tabelle 12.3.

Bild 12.5: Syntax für SSLRequire-Ausdrücke

Tabelle 12.2: Zusätzliche Umgebungsvariable für Client-Zertifikate

Name	Typ	Bedeutung
SSL_CLIENT_M_VERSION	\$	Versionsnummer des Client-Zertifikats
SSL_CLIENT_M_SERIAL	\$	Seriennummer des Client-Zertifikats
SSL_CLIENT_S_DN	\$	DN des Inhabers des Client-Zertifikats
SSL_CLIENT_S_DN_⟨x509⟩	\$	Komponente des SSL_CLIENT_S_DN
SSL_CLIENT_I_DN	\$	DN der Zertifizierungsstelle, die das Client-Zertifikat ausgestellt hat
SSL_CLIENT_I_DN_⟨x509⟩	\$	Komponente des SSL_CLIENT_I_DN
SSL_CLIENT_V_START	\$	Beginn des Gültigkeitszeitraums des Client-Zertifikats
SSL_CLIENT_V_END	\$	Ende des Gültigkeitszeitraums des Client-Zertifikats
SSL_CLIENT_A_SIG	\$	Zum Signieren des Client-Zertifikats benutzter Algorithmus
SSL_CLIENT_A_KEY	\$	Für den öffentlichen Schlüssel des Client-Zertifikats benutzer Algorithmus
SSL_CLIENT_CERT	\$	Das Client-Zertifikat (PEM-codiert)
SSL_CLIENT_CERT_CHAIN r	\$	Das r -te Zertifikat in der Zertifizierungskette des Client-Zertifikats (PEM-codiert)
SSL_CLIENT_VERIFY	\$	Ergebnis der Überprüfung des Client-Zertifikats: NONE, SUCCESS, GENEROUS oder FAILED:⟨Grund⟩

Siehe Tabelle 12.1 für weitere Erklärungen.

Tabelle 12.3: Funktionen in Apache-Ausdrücken

Name	Ausdruck	Priv
<code>req, http</code>	Holt HTTP-Kopfzeile aus der Anfrage; Name der Kopfzeile kann zu <code>Vary</code> -Kopfzeile hinzugefügt werden	–
<code>req_novary</code>	Holt HTTP-Kopfzeile aus der Anfrage; Name wird nicht zu <code>Vary</code> -Kopfzeile hinzugefügt	–
<code>resp</code>	Holt HTTP-Kopfzeile aus der Antwort	–
<code>reqenv</code>	(oder <code>v</code>) Holt Anfrage-Umgebungsvariable	–
<code>osenv</code>	Holt Betriebssystem-Umgebungsvariable	–
<code>note</code>	Holt Anfrage-Anmerkung (<i>request note</i>)	–
<code>env</code>	Holt ersten Treffer aus <code>note, reqenv, osenv</code>	–
<code>tolower</code>	Konvertiert Zeichenkette in Kleinbuchstaben	–
<code>toupper</code>	Konvertiert Zeichenkette in Großbuchstaben	–
<code>escape</code>	Konvertiert Sonderzeichen in %-Hexadezimal-Darstellung	–
<code>unescape</code>	Macht <code>escape</code> rückgängig	–
<code>base64</code>	Kodiert Zeichenkette gemäß Base-64	–
<code>unbase64</code>	Macht <code>base64</code> rückgängig	–
<code>md5</code>	Liefert MD5-Prüfsumme der Zeichenkette (hexadezimal kodiert)	–
<code>sha1</code>	Liefert SHA-1-Prüfsumme der Zeichenkette (hexadezimal kodiert)	–
<code>file</code>	Liefert Inhalt der benannten Datei (ggf. inklusive Zeilenenden)	ja
<code>filesize</code>	Liefert Größe der benannten Datei (oder 0, falls die Datei nicht existiert oder keine einfache Datei ist)	ja

Die Funktionen mit »ja« in der »Priv«-Spalte sind in `mod_ssl` zugänglich, aber nicht notwendigerweise in anderen Modulen.

SSLRequire Require-Direktive bei der Basic-Authentisierung – durch die `SSLRequire`-Direktive auf Verzeichnis- oder Dateiebene festgelegt. `SSLRequire` gestattet es, ausgefeilte Boolesche Ausdrücke anzugeben, die für jeden Zugriff ausgewertet werden. Die Syntax dieser Ausdrücke erinnert an eine Mischung der Programmiersprachen C und Perl und erlaubt den Rückgriff auf diverse SSL-Parameter sowie weitere Kriterien (Bild 12.5, Tabelle 12.2), die wir an ein paar Konfigurationsbeispielen illustrieren.

Beispiele Das folgende Beispiel erlaubt Benutzern den Zugriff zu einer Ressource, die entweder in der Marketing- oder der Entwicklungsabteilung der »Beispiel GmbH« arbeiten:

```
SSLRequire %{SSL_CLIENT_S_DN_0} eq "Beispiel GmbH" \
&& %{SSL_CLIENT_S_DN_OU} in { "Marketing", "Entwicklung" }
```

Das nächste Beispiel – aus [MODSSL-DOC] – greift das Problem aus dem vorigen Abschnitt auf, wo bestimmte durch Zertifikate authentisierte Benutzer Zugriff auf einen besonderen Bereich der Web-Präsenz bekommen sollen, der Rest der Präsenz aber auch den anderen Benutzern zur Verfügung stehen soll:

```
SSLVerifyClient none
<Directory /srv/www/htdocs/members>
  SSLVerifyClient      require
  SSLVerifyDepth       2
  SSLCACertificateFile /etc/httpd/ssl.crt/ca.crt
  SSLRequireSSL
  SSLRequire %{SSL_CLIENT_S_DN_0} eq "Beispiel GmbH" \
    && %{SSL_CLIENT_S_DN_OU} in { "Marketing", "Entwicklung" }
</Directory>
```

Dieses Beispiel funktioniert am besten, wenn Sie sicher sein können, dass die Client-Zertifikate eine bestimmte gemeinsame Struktur aufweisen (etwa weil Sie sie selber ausgestellt haben). Wenn das nicht gegeben ist, müssen Sie tricksen:

```
SSLVerifyClient none
<Directory /srv/www/htdocs/members>
  SSLVerifyClient      require
  SSLVerifyDepth       2
  SSLCACertificateFile /etc/httpd/ssl.crt/ca.crt
  SSLOptions           +FakeBasicAuth
  SSLRequireSSL
  AuthName             "Beispiel-Mitgliederseiten"
  AuthType              Basic
  AuthUserFile          /etc/httpd/httpd.passwd
  Require               valid-user
</Directory>
```

Hier verwenden wir die `FakeBasicAuth`-Option, um aus dem Subjekt-DN des Client-Zertifikats einen Benutzernamen für HTTP-Basic-Authentisierung zu machen. Damit können Sie die Apache-Standardmechanismen zur Authentisierung ausnutzen.

 Was konkret als Benutzername verwendet wird, können Sie mit dem Kommando

```
$ openssl x509 -in client.crt -noout -subject
```

herausfinden – typischerweise etwas wie

```
/C=DE/O=Example/OU=Clients/CN=Hugo Schulz
```

Der »gefakete« Benutzername muss in der Kennwortdatenbank (in unserem Beispiel `httpd.passwd` mit dem Kennwort »xxj31ZMTZzkVA« assoziiert werden (das ist die CRYPT-Verschlüsselung von »password«); wenn Sie lieber MD5-Verschlüsselung verwenden möchten (nicht dass es hier viel zu schützen gäbe), dann lautet das äquivalente Kennwort »\$1\$0XLyS...\$0wx8s2/m9/gfkcRVXzgoE/«. Mit diesem Ansatz können die Client-Zertifikate beliebige DNs haben.

Zum Schluss ein komplizierteres Beispiel für den Zugriff auf einen Intranet-Server. Dieser Zugriff ist entweder über HTTP aus dem Intranet selbst oder – für einen bestimmten Bereich – über HTTPS mit starker Verschlüsselung und alternativ Client-Zertifikaten oder Basic-Authentisierung von beliebigen Rechnern auf dem Internet aus erlaubt. Wir nehmen an, dass das Intranet Adressen aus dem Netz 192.168.0.0/24 benutzt und dass der extern sichtbare Bereich auf dem Server URLs hat, die mit `/ext/` anfangen. Definieren Sie die folgende Konfiguration *außerhalb* Ihres virtuellen Servers für HTTPS, damit sie für HTTP *und* HTTPS gilt.

Intranet-Server

Außerhalb des `/ext/-Bereichs` auf dem Server ist Zugriff nur aus dem Intranet erlaubt:

```
SSLCACertificateFile /etc/httpd/ssl.crt/ca.crt

<Directory /srv/www/htdocs>
  Order deny,allow
  Deny from all
  Allow from 192.168.0.0/24
</Directory>
```

Innerhalb von `/ext/` erlauben wir beliebige Zugriffe aus dem Intranet, aus dem Internet aber nur HTTPS mit starker Verschlüsselung und einem Kennwort oder einem Client-Zertifikat:

```

<Directory /srv/www/htdocs/ext>
  SSLVerifyClient optional
  SSLVerifyDepth 2
  SSLOptions      +FakeBasicAuth +StrictRequire
  SSLRequire      %{SSL_CIPHER_USEKEYSIZE} >= 128

  Satisfy any

  # Intranet ist OK
  Order deny,allow
  Deny  from all
  Allow from 192.168.0.0/24

```

Hier wird wieder die Authentisierungsmethode herangezogen, bei der die Zertifikatsinhaber-DNs als Basic-Authentisierungs-Benutzernamen interpretiert und in einer Kennwortdatei nachgeschlagen werden:

```

AuthType      Basic
AuthName      "Geschützter Intranet-Bereich"
AuthUserFile  /etc/httpd/intra.passwd
Require       valid-user

```

`mod_rewrite` Clients aus dem Internet werden gezwungen, HTTPS zu benutzen. `mod_rewrite` haben wir nicht weiter besprochen; die Idee ist, dass Sie mit `RewriteCond` eine Reihe von Bedingungen aufstellen, die zutreffen müssen, damit über die in `RewriteRule` beschriebene Umsetzung der gerade angefragte URL »umgeschrieben« wird. Das sehr simple Beispiel hier beantwortet jede Anfrage, auf die die `RewriteCond`-Bedingungen passen, mit dem HTTP-Rückgabewert `Forbidden`:

```

RewriteEngine on
RewriteCond  %{REMOTE_ADDR} !^192\.168\.0\.[0-9]+$
RewriteCond  %{HTTPS} != on
RewriteRule  .* - [F]

```

Die `SSLRequireSSL`-Direktive können wir hier nicht benutzen, da aus dem Intranet heraus auch normale HTTP-Zugriffe erlaubt sein sollen.

In Apache 2.4 ist `SSLRequire` verpönt. Statt dessen sollten Sie einfach `Require` benutzen, das ebenfalls logische Ausdrücke akzeptiert.

 Die Ausdrücke von `Require` sind eine Obermenge der bei `SSLRequire` (siehe Bild 12.5) erlaubten Ausdrücke. Der einzige Unterschied besteht darin, dass in `Require` die Vergleichsoperatoren `<`, `<=`, ... lexikografische Vergleiche für Zeichenketten machen, während die Vergleichsoperatoren `lt`, `le`, ... ganzzählige Vergleiche vornehmen. (In `Require` können Sie diese auch als `-lt`, `-le`, ... hinschreiben.)

Sofern Sie die Option `FakeBasicAuth` nicht verwenden, können Sie mit `SSLUserName` bestimmen, was Apache als »Name« des Benutzers ansehen soll. Sie können irgendeine der TLS-Umgebungsvariablen angeben:

```
SSLUserName SSL_CLIENT_S_DN_CN
```

(Dieser Benutzername taucht zum Beispiel als Wert der Umgebungsvariablen `REMOTE_USER` auf.)

Übungen

 **12.6** [2!] Installieren Sie das Zertifikat (oder die Zertifikate) Ihrer Zertifizierungsstelle in Ihrem Web-Server so, dass Sie damit die Authentizität von Client-Zertifikaten prüfen können, die (angeblich) von dieser Zertifizierungsstelle beglaubigt wurden.

 **12.7** [2!] Führen Sie auf Ihrem Web-Server einen Bereich ein, der nur nach erfolgreicher Authentisierung durch ein Client-Zertifikat angeschaut werden kann. Testen Sie dies.

 **12.8** [3] (Für Programmierer.) Schreiben Sie ein Shellskript, das, als CGI-Skript installiert, die SSL-Parameter – also die Umgebungsvariablen, die mit `SSL_` anfangen – einer HTTP-Anfrage ausgibt.

 **12.9** [2] Geben Sie `SSLRequire`-Regeln an, die die folgenden Sachverhalte ausdrücken:

1. Der Inhaber des Zertifikats arbeitet in der Entwicklungsabteilung der »Beispiel GmbH«.
2. Das Zertifikat wurde von der Zertifizierungsstelle der »Beispiel GmbH« ausgestellt.
3. Der Zugriff erfolgt von einem Rechner im Netz 10.11.12.13/16 aus.
4. Der Zugriff erfolgt in der Adventszeit (1.–24.12.)

12.4 Sicherheit für Web-Präsenzen verbessern

12.4.1 HTTP Strict Transport Security (HSTS)

HTTPS ist eine gute Sache, aber rund um die Verschlüsselung und Authentisierung gibt es noch einige Macken. Zum Beispiel würde man gerne herausfinden können, ob eine Web-Präsenz HTTPS unterstützt oder nicht – dem reinen Servernamen sieht man es jedenfalls nicht an, und die Browser probieren, wenn sie nichts Anderes gesagt bekommen, immer zuerst HTTP. Wenn HTTPS tatsächlich unterstützt wird, dann gibt es gerne mal Probleme mit Zertifikaten, wo die Benutzer allfällige Warnungen geflissentlich ignorieren. Viele Web-Anwendungen verwenden Cookies, um wichtige Informationen wie Authentisierungsdaten zu speichern, aber die Programmierer denken nicht unbedingt daran, diese Cookies als »sicher« zu kennzeichnen, so dass sie Angreifern in die Hände fallen können.

 Ein HTTPS-Server kann einen Cookie als »sicher« kennzeichnen. Dies signalisiert dem Browser, dass der Cookie nur über HTTPS-Verbindungen geschickt werden soll. Wenn ein HTTPS-Server einen Cookie setzt, der nicht als »sicher« gekennzeichnet ist, dann sendet der Browser ihn auch über HTTP-Verbindungen, wo ein Angreifer ihn aufpicken und missbrauchen kann.

»HTTP Strict Transport Security«, kurz HSTS ([RFC6797]) beschreibt einen Mechanismus, mit dem Web-Präsenzen bekanntgeben können, dass sie über HTTPS erreichbar sind, und der erzwingt, dass Browser auf solche Web-Präsenzen nur über HTTPS zugreifen, auch wenn der Benutzer HTTP angibt. Ferner führen alle TLS-Verbindungsprobleme zum Verbindungsabbruch, ohne dass »weiche« Warnungen ausgegeben werden, die den Benutzer zum »Durchklicken« animieren.

Um HSTS für eine Web-Präsenz zu aktivieren, müssen Sie Apache dazu bringen, bei verschlüsselten Verbindungen eine `Strict-Transport-Security`-Kopfzeile zu versenden. Diese könnte zum Beispiel aussehen wie

```
Strict-Transport-Security: max-age=31536000; includeSubDomains
```

Der `max-age`-Parameter postuliert, dass die Web-Präsenz für ein Jahr (31,536,000 Sekunden) nur über HTTPS angesprochen werden soll. `includeSubDomains` verfügt, dass das nicht nur für den Server gelten soll, der die Kopfzeile verschickt hat, sondern auch für alle Subdomains.

 HSTS ist ein *Trust-on-first-use*- oder »TOFU«-Mechanismus. Das heißt, der erste Zugriff auf einen Server kann nicht über HSTS abgesichert werden, da der Browser die HSTS-Informationen erst während dieses ersten Zugriffs bekommt. Spätere Zugriffe können dann aber HSTS nutzen.

 Gängige Browser enthalten »ab Werk« HSTS-Informationen für populäre Web-Präsenzen. Google Chrome zum Beispiel kommt bereits mit den HSTS-Daten für die Google-Webseiten (und einige mehr, etwa Paypal, Twitter usw.), so dass auch der erste Zugriff auf diese mit HSTS gesichert ist.

 Damit `includeSubDomains` wirklich funktioniert, müssen Sie dafür sorgen, dass alle Server auf Subdomains korrekt für HTTPS konfiguriert sind. Auch Faktoren wie der TLS-Session-Cache müssen bedacht werden.

 `includeSubDomains` bezieht sich auf den tatsächlichen Namen des Servers (inklusive »www.«, falls vorhanden). Das Hauptproblem, das `includeSubDomains` lösen soll, ist die Verwendung von »Domain-Cookies«, bei denen `www.example.com` einen Cookie für `.example.com` setzt, der dann auch an andere Server unterhalb von `example.com` weitergegeben wird. (Siehe Übung 12.12.)

 Wenn Sie die Namen `www.example.com` und `example.com` (mit `includeSubDomains`) benutzen, dann stellen Sie sicher, dass jede Seite auf `www.example.com` zumindest eine Ressource von `example.com` aufruft, damit der Browser auf jeden Fall die Strict-Transport-Security-Kopfzeile für die komplette Domain zu sehen bekommt.

Die Strict-Transport-Security-Kopfzeile soll laut RFC nur über verschlüsselte Verbindungen geschickt werden; Browser, die solche Kopfzeilen auf unverschlüsselten Verbindungen oder Verbindungen mit fehlerhaften Zertifikaten geschickt bekommen, sind gehalten, sie zu ignorieren. Wenn Sie HSTS einsetzen, ist es am besten, alle über HTTP eingehenden Anfragen auf HTTPS umzuleiten.

 Sie können HSTS wieder rückgängig machen, indem Sie eine Kopfzeile wie

`Strict-Transport-Security: max-age=0`

verschicken. Allerdings muss ein Browser sich mit Ihrer Web-Präsenz verbinden, um sie zu sehen zu kriegen; vorher bekommt er davon nichts mit.

HSTS aktivieren Sie am besten im Web-Server, so dass alle Anfragen für die betreffende (HTTPS-)Web-Präsenz die Kopfzeile geschickt bekommen. Verwenden Sie `mod_headers` und eine Direktive wie

`Header always set Strict-Transport-Security \n"max-age=31536000; includeSubDomain"`

Bild 12.6 zeigt ein etwas ausführlicheres Beispiel, das sich auch um HTTP-Zugriffe kümmert.

Übungen

 **12.10** [!2] Konfigurieren Sie Ihren Web-Server (in Anlehnung an Bild 12.6) so, dass er HSTS unterstützt. Stellen Sie sicher, dass HTTP-Anfragen an den Server korrekt auf HTTPS umgesetzt werden. (Verwenden Sie Wireshark oder die Debugging-Funktionen Ihres Browsers, um zu prüfen, ob der Browser HTTP-Anfragen direkt als HTTPS absetzt oder ob der Server die Anfragen explizit umleitet. Unter welchen Umständen passiert was?)

```

<VirtualHost *:80>
  ServerName www.example.com
  ServerAlias example.com

  # Alle Zugriffe auf die HTTPS-Version der Seite umleiten
  Redirect permanent / https://www.example.com/
</VirtualHost>

<VirtualHost *:443>
  ServerName www.example.com
  ServerAlias example.com

  SSLEngine on
  ...
  Header always set Strict-Transport-Security \
    "max-age=31536000; includeSubDomain"
</VirtualHost>

```

Bild 12.6: Beispiel für HSTS-Konfiguration

 **12.11** [2] Was passiert, wenn die in der Strict-Transport-Security-Kopfzeile angegebene Frist abgelaufen ist und der Browser eine HTTP-Anfrage an den Server stellt? Probieren Sie das (mit einem hinreichend kleinen Wert für max-age) aus.

 **12.12** [3] Angenommen, example.com ist der Domainname für eine Web-Präsenz, die HSTS benutzt, ohne includeSubDomains zu setzen. Ferner angenommen, die Web-Präsenz setzt einen »sicheren« Cookie für die komplette Domain example.com. Wie kann ein Angreifer an diesen Cookie herankommen?

12.4.2 Public Key Pinning (HPKP)

Ein großes Problem mit der aktuellen Public-Key-Infrastruktur ist, dass im Grunde jede Zertifizierungsstelle der Welt ein Zertifikat für jeden beliebigen Server auf der Welt ausstellen kann – die Zustimmung des Server-Betreibers ist dafür nicht erforderlich. Das heißt, dass ein Angreifer, der eine Zertifizierungsstelle kontrolliert, MITM-Angriffe durchführen kann, die naiven Benutzern (und das bedeutet allen, die weder die Zertifikatsdetails prüfen noch wissen, welche Zertifizierungsstelle die »richtige« für den Server ist, also 99,99% der Benutzergemeinde) nicht auffallen werden.

 Diese Sorte Angriff ist kommerziell verfügbar in Gestalt von »Deep Packet Inspection«-Firewalls, die HTTPS-Verbindungen abfangen und selber nach Bedarf Zertifikate für die eigentlich angesprochenen Server ausstellen. Firmen setzen so etwas mitunter ein, um den Internetgebrauch ihrer Angestellten zu kontrollieren, und sind natürlich in der Lage, das dazugehörige selbstsignierte Zertifikat in deren Browser unterzubringen.

 Unappetitlich wird es da, wo Nationalstaaten den kompletten Internet-Verkehr ihrer Bürger über eine solche Infrastruktur laufen lassen. Die dadurch geschaffenen Einblicksmöglichkeiten in deren Privatsphäre gehen weit über das hinaus, was man zumindest nach den Standards moderner Demokratien erwarten würde.

Public Key Pinning (HPKP) nach [RFC7469] ist ein Mechanismus, mit dem der Betreiber eines Web-Servers Aussagen darüber machen kann, welche öffentlichen Schlüssel (genau genommen *subject public key info*-Strukturen oder SPKIs) in der Zertifikatskette für den Web-Server zu erwarten sind. HPKP-kompatible Browser SPKIs

können dann Alarm schlagen, wenn sich Abweichungen ergeben. HPKP ist dafür gedacht, zusammen mit HSTS (Abschnitt 12.4.1) eingesetzt zu werden, aber funktioniert auch ohne. Ähnlich wie bei HSTS muss ein Browser sich einmal mit dem Server verbinden, um Informationen über die zulässigen Schlüssel zu bekommen; Folgezugriffe werden dann abgesichert, aber dem ersten Zugriff muss der Browser vertrauen.

 Ähnlich wie bei HSTS kommen moderne Browser »ab Werk« mit Pinning-Informationen für populäre Webseiten. Sie können also ziemlich sicher sein, dass da, wo Google draufsteht, tatsächlich auch Google drin ist, selbst wenn Sie sich in einem »feindlichen« Netz befinden.

 HPKP ist mit dem Risiko verbunden, dass Sie Ihren Web-Server selbst unverfügbar machen, indem Sie SPKIs »pinnen«, die aus irgendwelchen Gründen ungültig werden.

 HPKP beschäftigt sich mit SPKIs, nicht mit Zertifikaten, weil das Server-Betreibern die Möglichkeit gibt, neue Zertifikate für dieselben öffentlichen Schlüssel auszustellen. SPKIs umfassen nicht nur den öffentlichen Schlüssel, sondern auch einige weitere Parameter, die dessen Missbrauch ausschließen sollen (ansonsten könnte ein Angreifer zum Beispiel einen Diffie-Hellman-Schlüssel »wiederverwenden«, indem er dem Client vormacht, er müsste in einer anderen multiplikativen Gruppe interpretiert werden).

Was pinnen? Die naheliegendste Idee ist, die SPKI Ihres Web-Servers selbst zu »pinnen«. Das ist aber taktisch unklug, da es Ihnen keinen Raum für den Fall lässt, dass Ihr privater Schlüssel kompromittiert wird oder sonst etwas unappetitliches Lokales passiert, das Sie dazu nötigt, einen neuen Schlüssel zu generieren. Es ist geschickter, statt dessen die SPKI aus dem Zwischen-Zertifikat Ihrer Zertifizierungsstelle zu pinnen, mit dem Ihr Zertifikat signiert wurde.

 Es kann immer noch passieren, dass Ihre Zertifizierungsstelle Ihnen ein neues Zertifikat mit einem anderen Schlüssel signiert. Dagegen können Sie nicht viel tun, außer sich von Anfang an eine Zertifizierungsstelle zu suchen, die weiß, was HPKP ist und wie man damit umgehen muss.

 Es hindert Sie niemand daran, »Pins« für mehrere SPKIs zu veröffentlichen. Sie können also immer ein Not-Zertifikat in der Hinterhand haben, das von einer anderen Zertifizierungsstelle beglaubigt wurde.

 Am besten bedient sind Sie, wenn Sie eine eigene private Zertifizierungsstelle haben, deren Zertifikat von einer gut vernetzten öffentlichen Zertifizierungsstelle beglaubigt wurde. Sie können dann die SPKI Ihres eigenen Zertifikats pinnen, mit dem Sie Ihre Server-Zertifikate beglaubigen, und notfalls ein neues Zertifikat für den gepinnten Schlüssel von einer anderen Zertifizierungsstelle ausstellen lassen.

In der Praxis können Sie SPKIs pinnen, indem Sie in Ihre HTTPS-Antworten **Public-Key-Pins** eine Public-Key-Pins-Kopfzeile einbauen, die ein oder mehrere »Pins« enthält. Das kann ungefähr so aussehen:

```
Public-Key-Pins: max-age=2592000;
pin-sha256="k8frmHk+4kJaLJ6XwR2BWcmyctRxYsgXE2Y5Aem96f0=";
pin-sha256="RGA98iPkynS2ZiQCVY06HyUveQbH0C0je1a/V+OLU5g="
```

Dieses Beispiel enthält zwei Pins, die mit SHA-256 gehasht und dann Base-64-kodiert wurden. (SHA-256 ist im Moment die einzige erlaubte Hash-Funktion für PINs.) Der erste Pin bezieht sich auf die SPKI des untergeordneten Zertifikats unserer Zertifizierungsstelle, mit der das Server-Zertifikat beglaubigt wurde. Mit dem zweiten Pin können wir notfalls schnell ein »Reserve-Zertifikat« in Stellung bringen, das mit einer anderen SPKI signiert wurde. Dieses Zertifikat ist im Normalfall allerdings »offline« und nicht in der tatsächlichen Zertifikatkette zu finden.

 Diese Vorgehensweise ist offiziell vorgeschrieben – Browser dürfen keine Public-Key-Pins-Kopfzeilen akzeptieren, die nicht mindestens einen »Reserve-Pin« enthalten.

Browser, die eine Public-Key-Pins-Kopfzeile geschickt bekommen, legen die darin enthaltenen Informationen für den betreffenden Server in einer Datenbank ab und prüfen ab da bei neuen Zugriffen, ob die gepinnten SPKIs in der Zertifikatskette vorkommen (zumindest ein Treffer muss sein). Ist das nicht der Fall, wird die Verbindung abgebrochen.

 Mit `includeSubDomains` können Sie dafür sorgen, dass die Pins nicht nur für den Namen des Rechners selbst durchgesetzt werden, sondern auch für untergeordnete Namen. Es gelten dieselben Warnungen wie bei HSTS.

Wahrscheinlich fragen Sie sich jetzt, wie Sie an Hashes für die Pins kommen. Hashes für die Pins Die Antwort darauf heißt, wie üblich, »mit OpenSSL«:

```
$ openssl x509 -noout -in sub/sub-ca.crt -pubkey \
> | openssl asn1parse -noout -inform pem -out sub-ca.pubkey
$ openssl dgst -sha256 -binary sub-ca.pubkey \
> | openssl enc -base64
```

Das erste Kommando extrahiert die SPKI-Datenstruktur aus dem Zertifikat der untergeordneten Zertifizierungsstelle und schreibt sie in (binärer) DER-Kodierung in die Datei `sub-ca.pubkey`. Das zweite Kommando bestimmt den SHA-256-Hashwert des SPKI und Base-64-kodiert ihn.

Ähnlich wie bei HSTS gibt die `max-age`-Klausel in der Public-Key-Pins-Kopfzeile an, wie lange die Pins gültig sind (in Sekunden). Während Sie bei HSTS aber relativ großzügig mit den Fristen umgehen können, gibt es bei HPKP keine Möglichkeit, Pins mit `max-age=0` zu widerrufen. Sie sollten also am besten erst mal relativ tief stapeln, bis Sie sicher sind, dass Ihre Pins stimmen. Ein (schließlich erreichtes) maximales Alter von 60 Tagen (oder so) ist wahrscheinlich ein halbwegs tragfähiger Kompromiss.

 HPKP hat im Gegensatz zu HSTS einen Mechanismus, mit dem Browser Sie über Verstöße gegen Ihre Pins benachrichtigen können. Sie können in der Public-Key-Pins-Kopfzeile mit einer `report-uri`-Klausel einen URI angeben, an den JSON-strukturierte Daten geschickt werden (mit HTTP-POST), die das Problem beschreiben: Benachrichtigung

```
Public-Key-Pins: max-age=2592000;
pin-sha256="k8frmHk+4kJalJ6XwR2BWcmcyctRxYsgXE2Y5Aem96f0=";
pin-sha256="RGA98iPkynS2ZiQCVY06HyUveQbH0C0je1a/V+0LU5g=";
report-uri="http://www.example.com/pkp-report"
```

Die Details stehen in [RFC7469, Abschnitt 3]. (Siehe auch Übung 12.15.)

 Sie können Pins unverbindlich ausprobieren, indem Sie statt `Public-Key-Pins` eine `Public-Key-Pins-Report-Only`-Kopfzeile verwenden. In diesem Fall führen Pin-Probleme nicht zum Verbindungsabbruch, sondern werden lediglich gemeldet. Das heißt aber auch, dass Sie eine `report-uri`-Klausel angeben müssen, sonst ist der Browser frei, Ihre unverbindlichen Pins zu ignorieren. Pins ausprobieren

 Sie können `Public-Key-Pins` und `Public-Key-Pins-Report-Only` auch gleichzeitig benutzen. In diesem Fall werden die in `Public-Key-Pins` angegebenen Pins durchgesetzt und die in `Public-Key-Pins-Report-Only` angegebenen Pins zwar angeschaut und Verstöße gemeldet, aber nicht durchgesetzt.

 Wenn Sie für eine Web-Präsenz HPKP ausrollen wollen, sollten Sie vorsichtig anfangen und zunächst `Public-Key-Pins-Report-Only` verwenden. Wenn Sie denken, dass alles funktioniert, stellen Sie auf `Public-Key-Pins` um. Arbeiten Sie sich dann von kurzen Fristen (im Minutenbereich) zu längeren hoch; das sinnvolle Maximum ist, wie gesagt, irgendwo in der Gegend von 60 Tagen. HPKP ausrollen

Übungen



12.13 [!1] Konstruieren Sie wie angegeben die Base64-kodierte SHA-256-Prüfsumme für die SPKI in einem X.509-Zertifikat.



12.14 [3] Vergewissern Sie sich, dass HPKP funktioniert, indem Sie Ihren Server so konfigurieren, dass er die SPKI aus dem Zertifikat Ihrer untergeordneten Zertifizierungsstelle pinnt. Ersetzen Sie dann Ihr Server-Zertifikat durch ein Zertifikat, das von einer anderen Zertifizierungsstelle auf denselben DN ausgestellt wurde, und beobachten Sie, was bei einem erneuten Zugriff auf Ihren Server passiert. (*Pro-Tip:* Tauschen Sie in einem Präsenzkurs Zertifikate mit Ihrer Sitznachbarin/Ihrem Sitznachbar.)



12.15 [2] Der Server `www.example.com` beantwortet HTTPS-Zugriffe (unter anderem) mit der Kopfzeile

```
Public-Key-Pins: max-age=2592000;
pin-sha256="k8frmHk+4kJalJ6XwR2BWcmyclRxYsgXE2Y5Aem96f0=";
pin-sha256="RGA98iPkynS2ZiQCVY06HyUveQbH0C0je1a/V+0LU5g=";
report-uri="http://www.example.com/pkp-report"
```

Warum verwendet `report-uri` kein HTTPS?

12.4.3 Content Security Policy (CSP)

Während HSTS dafür gedacht ist, dass Web-Präsenzen konsequent über HTTPS angesprochen werden, und HPKP Fisimatenten mit der Zertifikatskette aufdecken kann, soll der »Content Security Policy«-Mechanismus Angriffe über *cross-site scripting* (XSS) erschweren.



Bei einem XSS-Angriff schleust ein Angreifer Schadcode (typischerweise in JavaScript) in eine HTML-Seite ein und kann auf diese Weise zum Beispiel Authentisierungs-Cookies stehlen und später für seine Zwecke missbrauchen.

CSP ist eine Erfindung des World-Wide-Web-Konsortiums und steht aktuell als W3C *Candidate Recommendation* zur Verfügung [W3C-CSP15].

CSP ist ein Mechanismus, der Serverbetreibern mehr Kontrolle darüber gibt, welche Ressourcen von wo geholt werden können. Sie können zum Beispiel festlegen, von wo externer Code geholt werden darf, und Sie können die Ausführung von Inline-JavaScript komplett verhindern. Ähnlich wie bei HSTS wird das über Kopfzeilen in den HTTP-Antworten geregelt. Das kann ungefähr so aussehen:

```
Content-Security-Policy: default-src 'self';
                        img-src *;
                        script-src code.example.com;
                        object-src *.example.com
```

In diesem Beispiel kann die betreffende HTML-Seite Bilder von irgendwo im Netz und externen JavaScript-Code von `code.example.com` holen. Daten für Plugins, die mit dem HTML-`<object>`-Tag eingebunden werden, können von irgendeinem Server in der Domain `example.com` kommen. Alles andere muss vom selben Server kommen wie die HTML-Seite selbst.



Neben den schon gezeigten Klauseln `default-src`, `script-src` und `object-src` gibt es noch einige mehr. `font-src` kontrolliert zum Beispiel den Zugriff auf Schriften, `style-src` den Zugriff auf CSS-Stylesheets, `media-src` den Zugriff auf HTML5-Audio- und -Video-Dateien, und `connect-src` regelt, welche Ziele für »ausgehende« Verbindungen wie AJAX-Aufrufe mit XMLHttpRequest oder WebSockets in Frage kommen.

Ein typisches Problem in diesem Zusammenhang ist »gemischter Inhalt«, also HTML-Seiten, die über HTTPS geladen werden, aber auf Ressourcen zugreifen, die nur über HTTP zur Verfügung stehen. Browser sind von so etwas nicht begeistert und meckern es an, aber verboten ist es in vielen Fällen nicht (etwa bei Bildern). Gefährlich wird es allerdings bei aktiven Inhalten, also etwa JavaScript-Dateien, die, wenn sie über HTTP geholt werden, von einem Angreifer mit schädlichen Zusätzen verbrämt werden können. Mit CSP können Sie das verhindern, indem Sie zumindest festlegen, dass externe Inhalte über HTTPS geladen werden müssen:

```
Content-Security-Policy: default-src https: ; connect-src https:
```

Inline-JavaScript, also JavaScript"-Code, der in der HTML-Seite selber steht, wird von CSP standardmäßig verboten. Das ist natürlich eine ziemlich heftige Einschränkung, und Sie können sie mit

```
Content-Security-Policy: default-src https: 'unsafe-inline'
```

aufheben. Ebenfalls standardmäßig verboten ist die Ausführung von dynamisch erzeugtem Code; wenn Sie das auch erlauben wollen, müssen Sie noch das Schlüsselwort 'unsafe-eval' anhängen.



Wenn Sie Inline-Skript-Code nicht ganz loswerden können, können Sie Angreifern das Leben schwerer machen, indem Sie denNonce-Mechanismus von CSP ausnutzen. Wenn Sie etwas angeben wie

```
Content-Security-Policy: script-src 'self' 'nonce-⟨Nonce⟩'
```

(wobei ⟨Nonce⟩ für eine idealerweise 128 Bit lange hexadezimal kodierte Zufallszahl steht), dann werden nur solche Inline-`<script>`-Elemente ausgeführt, die denselben ⟨Nonce⟩-Wert enthalten. Das heißt, zu

```
Content-Security-Policy: script-src 'self' 'nonce-cc2ced20393403db'
```

passt nur

```
<script nonce="cc2ced20393403db">
  <!--
  -->
</script>
<script nonce="cc2ced20393403db"
  src="https://code.example.com/script.js"></script>
```

Der Witz besteht darin, dass Sie die ⟨Nonce⟩ bei jedem Seitenabruf neu vergeben. Für einen Angreifer ist es extrem schwer, die ⟨Nonce⟩ korrekt vorherzusagen.

Wenn Sie eine CSP-Konfiguration nicht gleich »scharf schalten«, sondern erst einmal testen wollen, können Sie statt Content-Security-Policy die Kopfzeile Content-Security-Policy-Report-Only verwenden. Dann werden Ressourcen, die gegen die CSP-Konfiguration verstößen, zwar geholt, aber die Verstöße werden gemeldet.



»Gemeldet« werden können Verstöße auf zwei Arten: Einerseits wird dem Document-Objekt der betreffenden Ressource ein JavaScript-Event signalisiert, der Details über den Verstoß enthält. Andererseits kann ähnlich wie bei HPKP (Abschnitt 12.4.2) die Content-Security-Policy-Report-Only-Kopfzeile eine report-uri-Klausel enthalten, die einen URL enthält, an den ein JSON-Objekt mit einer Beschreibung des Verstoßes geschickt wird (mit HTTP-POST). Das kann zum Beispiel aussehen wie

```
{
  "csp-report": {
    "document-uri": "http://www.example.com",
    "referrer": "http://cracker.example.org/crack.html",
    "blocked-uri": "http://cracker.example.org/l33t.png",
    "violated-directive": "default-src 'self'",
    "original-policy": "default-src 'self'; report-uri >
      < http://www.example.org/csp.cgi"
  }
}
```

Die genauen Details stehen in [W3C-CSP15, Abschnitt 4.4].

 Den Melde-Mechanismus können Sie auch für aktive CSP-Konfigurationen verwenden, wenn Sie eine `report-uri`-Klausel zur Kopfzeile hinzufügen.

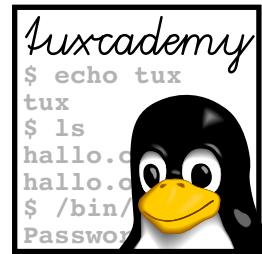
 Sie können durchaus eine CSP-Konfiguration testen, während eine andere CSP-Konfiguration aktiv ist und durchgesetzt wird. Ganz allgemein können sogar mehrere CSP-Konfigurationen gleichzeitig aktiv sein, wobei Resourcenzugriffe von allen Konfigurationen erlaubt werden müssen, damit sie insgesamt erlaubt sind.

Zusammenfassung

- `mod_ssl` stellt die Verbindung zwischen Apache und OpenSSL her. Apache kann damit SSL zur Server- und Client-Authentisierung einsetzen.
- Mit `SSLOptions` können diverse Optionen gesetzt werden.
- Die Direktiven `SSLRequireSSL` und `Require ssl` machen es möglich, Zugriffe auf Ressourcen nur über HTTPS zu erlauben.
- Ein Apache-Server kann mehrere Web-Präsenzen mit SSL verwalten, solange diese entweder als IP-basierte virtuelle Server implementiert sind, ein Zertifikat mit `subjectAltName`-Erweiterung verwendet wird oder Client und Server SNI unterstützen.
- Für den Import in einen Browser müssen Sie Client-Zertifikate (und die dazugehörigen privaten Schlüssel) ins PKCS#12-Format umwandeln.
- Zur Authentisierung von Zugriffen über Client-Zertifikate prüft Apache zunächst die Authentizität des Zertifikats und ordnet dem Zugriff dann anhand der Identität des Zertifikatsinhabers und anderer Kriterien Rechte zu.
- Mit HSTS können Sie erzwingen, dass Zugriffe auf eine Web-Präsenz mit HTTPS erfolgen.
- CSP ist ein Mechanismus, mit dem Angreifer daran gehindert werden sollen, schädlichen Code (oder allgemein schädliche Ressourcen) in eine Webseite einzuschleusen.

Literaturverzeichnis

- FREAK** »Tracking the FREAK Attack«. <https://freakattack.com/>
- MODSSL-DOC** Ralf S. Engelschall. »Apache SSL/TLS Encryption«. <http://httpd.apache.org/docs/2.2/ssl/>
- RFC2253** M. Wahl, S. Kille, T. Howes. »Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names«, Dezember 1997. <http://www.ietf.org/rfc/rfc2253.txt>
- RFC2459** R. Housley, W. Ford, W. Polk, et al. »Internet X.509 Public Key Infrastructure Certificate and CRL Profile«, Januar 1999. <http://www.ietf.org/rfc/rfc2459.txt>
- RFC2817** R. Khare, S. Lawrence. »Upgrading to TLS Within HTTP/1.1«, Mai 2000. <http://www.ietf.org/rfc/rfc2817.txt>
- RFC4366** S. Blake-Wilson, M. Nystrom, D. Hopwood, et al. »Transport Layer Security (TLS) Extensions«, April 2006. <http://www.ietf.org/rfc/rfc4366.txt>
- RFC5077** J. Salowey, H. Zhou, P. Eronen, et al. »Transport Layer Security (TLS) Session Resumption without Server-Side State«, Januar 2008. <http://www.ietf.org/rfc/rfc5077.txt>
- RFC5746** E. Rescorla, M. Ray, S. Dispensa, et al. »Transport Layer Security (TLS) Renegotiation Indication Extension«, Februar 2010. <http://www.ietf.org/rfc/rfc5746.txt>
- RFC6797** J. Hodges, C. Jackson, A. Barth. »HTTP Strict Transport Security (HSTS)«, November 2012. <http://www.ietf.org/rfc/rfc6797.txt>
- RFC7469** C. Evans, C. Palmer, R. Sleevi. »Public Key Pinning Extension for HTTP«, April 2015. <http://www.ietf.org/rfc/rfc7469.txt>
- Ris14** Ivan Ristić. *Bulletproof SSL and TLS*. London: Feisty Duck Ltd., 2014. ISBN 978-1-907117-04-6. <https://www.feistyduck.com/books/bulletproof-ssl-and-tls/>
- W3C-CSP15** Mike West, Adam Barth, Dan Veditz. »Content Security Policy Level 2«. W3C Candidate Recommendation, Februar 2015. <http://www.w3.org/TR/CSP/>



13

Proxy-Grundlagen

Inhalt

13.1 Einleitung	172
13.2 Arten von Proxies	172
13.2.1 Generische Proxies	172
13.2.2 HTTP-Proxies	172
13.2.3 Reverse Proxies	174
13.2.4 Proxies und HTTPS	174
13.3 Software	175
13.3.1 Allgemein	175
13.3.2 Squid	175
13.3.3 Apache	176
13.3.4 Andere Programme	176

Lernziele

- Die Arbeitsweise von HTTP-Proxys verstehen
- Den Proxy-Server Squid konfigurieren können

Vorkenntnisse

- Kenntnisse der Administration von Linux-Systemen
- Elementares Verständnis von HTTP

13.1 Einleitung

Viele Internet-Protokolle, etwa SMTP und DNS, sind darauf ausgerichtet, dass Clients nicht direkt mit dem zuständigen Server Kontakt aufnehmen, sondern Proxy dass sie die Arbeit an Stellvertreter (engl. *proxy*) abgeben. Dazu verbindet sich der Client mit dem Proxy-Server, also beispielsweise mit dem Mailrelais seines Providers. Der Proxy verbindet sich dann entweder direkt mit dem Ziel-Server oder mit einem weiteren Proxy. (Bei Mail und DNS ist die Stellvertreterfunktionalität so fest eingebaut, dass es unüblich ist, hier explizit von Proxies zu sprechen.)

Da Proxies als zentrale Knoten für das jeweilige Protokoll dienen, erlauben sie es, Zusatzfunktionalitäten zu implementieren. Hierzu zählen das Filtern und Protokollieren der Verbindungen (auf Applikationsebene) im Rahmen einer Firewall-Implementierung sowie das Speichern von Objekten (Web-Ressourcen, Dateien, Cache DNS-Datensätze usw.), das Caching. Erneute Anfragen nach einem gespeicherten Objekt, die über den Proxy gehen, werden dann direkt von ihm beantwortet, ohne dass der ursprüngliche Server noch einmal gefragt werden muss, womit sich die Performanz mitunter drastisch erhöhen lässt. (Beispiel: DNS – nicht jedoch SMTP, da sich Mail üblicherweise nicht wiederholt.) Häufig ist der Grund für die Einrichtung eines Proxies gerade die Möglichkeit des Cachings.

13.2 Arten von Proxies

13.2.1 Generische Proxies

Die Vorteile von Proxies bei Performanz und Sicherheit lassen sich leider nicht auf beliebige Protokolle übertragen: Damit ein Daten-Objekt zur weiteren Verwendung gespeichert werden kann, muss natürlich bekannt sein, was die Daten-Objekte des Protokolls sind. Außerdem muss zu einem Objekt seine Verfallszeit ermittelt werden können: Ab wann sollen die Daten des Caches als überholt gelten und durch eine erneute Anfrage beim Ursprungsserver aufgefrischt werden. (Routing-Informationen sind sehr kurzlebig, DNS-Antworten leben etwas länger, Web-Dokumente schließlich können sehr alt werden oder auch vom Caching ganz ausgeschlossen werden.)

Aber selbst wenn man auf das Caching verzichtet und sich auf das Filtern und Loggen auf Applikationsebene beschränken möchte, so ist dies nicht bei jedem generischer Proxy Protokoll möglich. Zwar existieren generische Proxies wie das Programm *rinetd*, die über keinerlei protokollspezifische »Intelligenz« verfügen. Für einen solchen generischen Proxy reduzieren sich aber dann die verfügbaren Informationen auf IP-Adressen und Ports. Da die Ziel-IP-Adresse die des Proxies selbst ist, kann er also nur aufgrund des Ports (und allenfalls der Client-IP-Adresse) die Verbindung aufbauen. Dies bedeutet, dass pro Dienst (Port) nur immer derselbe, fest vorgegebene Server benutzt werden kann.

SOCKS Einen anderen Weg geht der generische Proxy SOCKS. Um an die Verbindungsdaten zu kommen, handeln Client und SOCKS-Server die Verbindungsdaten über ein eigenes Protokoll aus, d. h. die Applikation auf dem Client-Rechner muss modifiziert werden, damit sie beim SOCKS-Server ihren Verbindungs-wunsch anmelden kann.

Zusammenfassend folgt daraus: Ein Proxy-Server, der Verbindungen zu beliebigen Servern stellvertretend aufnehmen können soll, ohne bestehende Protokolle zu verändern oder zu ergänzen, muss die Information, mit welchem Server er sich verbinden soll, aus dem Protokoll selbst entnehmen können. Daher sind nur einige Protokolle in diesem allgemeinen Sinne proxyfähig.

13.2.2 HTTP-Proxies

Eine übliche Anfrage eines Clients an einen Web-Server sieht z. B. so aus:

```
GET / HTTP/1.0
User-Agent: w3m/0.3.2.2
Accept: text/*, image/*, application/*, video/*, audio/*
Accept-Encoding: gzip, compress, bzip, bzip2, deflate
Accept-Language: en;q=1.0
Host: www.example.org
```

Hiervon ist bei HTTP/1.0 die Angabe der Methode (hier: GET; plus Argumente) HTTP-Methode verpflichtend, die Hostangabe (letzte Zeile) ist optional. Erst mit HTTP/1.1 ist auch die Hostangabe zwingend vorgeschrieben; nur damit sind namensbasierte virtuelle Web-Server möglich.

 Bei namensbasierten virtuellen Web-Servern teilen sich mehrere bis viele Web-Präsenzen mit verschiedenen Servernamen dieselbe IP-Adresse. Auf einem Server mit der Adresse 11.22.33.44 könnten zum Beispiel unabhängig voneinander die Web-Präsenzen `www.example.com` und `www.linupfront.de` verwaltet werden. Wenn ein Client mit dem HTTP-Port von 11.22.33.44 Kontakt aufnimmt und eine Anfrage der Form

```
GET /index.html HTTP/1.0
```

stellt, muss der Web-Server entscheiden, ob das `index.html` von `www.example.com` und `www.linupfront.de` gemeint ist. Da er das an der ersten Zeile der HTTP-Anfrage nicht sehen kann (sie sähe in beiden Fällen gleich aus), zieht er dazu, falls vorhanden, die `Host`-Zeile heran, die den Namen der gewünschten Präsenz enthält. Der Browser des Clients ist dafür verantwortlich, aus

```
http://www.example.com/index.html
```

die Anfrage

```
GET /index.html HTTP/1.0
<<<<<
Host: www.example.com
<<<<
```

zu machen.

Die dem Server dargebotenen Informationen, also die absolute Pfadangabe des gewünschten URI, sind für ihn zwar ausreichend, einem Proxy können sie aber nicht genügen. Deswegen muss *der Client* seine Anfrage an einen Proxy anders gestalten. Dazu folgt nach GET der absolute URI statt nur der absoluten Pfadangabe, im Beispiel also GET `http://www.example.org/` statt nur GET `/`. (Weder HTTP/1.0 noch HTTP/1.1 erfordern, dass Clients auch bei direkten Anfragen an einen Web-Server den absoluten URI benutzen. Allerdings sollen Server absolute URIs akzeptieren, um einen generellen Übergang in späteren HTTP-Versionen zu ermöglichen.)

 Sie könnten sich mit einiger Berechtigung fragen, warum man nicht von Anfang an verlangt hat, dass HTTP-Anfragen den kompletten (absoluten) URI der gewünschten Ressource enthalten. Rückblickend wäre das wahrscheinlich vernünftiger gewesen, aber es ist halt nicht so passiert.

Bei HTTP muss also der Client wissen, dass er einen Proxy benutzen soll, um

1. die Anfrage an den Proxy umzuleiten (Manipulation von IP-Adresse und Port)
2. die Anfrage an den Proxy zu erweitern (Manipulation der Anfrage selbst).

 Einige Programme wie wget, w3m und SUSEs Online-Update-Tool you werten die Umgebungsvariablen http_proxy, ftp_proxy u. a. aus, um ihren Proxy zu konfigurieren. Auch KDEs konqueror kann so konfiguriert werden, dass er diese Variablen benutzt.

Außer dieser »expliziten« Methode gibt es auch noch die Möglichkeit des so genannten *interception proxying*, auch *transparent proxying* genannt. Hierbei wird auf den Clients kein Proxy konfiguriert, sondern alle HTTP-Anfragen auf der TCP- bzw. IP-Ebene an den Proxy weitergeleitet. Über Interception Proxying ist es leichter möglich, redundante und damit zuverlässige Proxy-Infrastrukturen aufzubauen, aber das Konzept hat auch verschiedene Nachteile, die es für manche Anwendungen unbrauchbar machen.

13.2.3 Reverse Proxies

Ein weiteres wichtiges Einsatzgebiet von Proxies ist auf der Server-Seite. Oft ist es sinnvoll, einen Proxy *vor* einen öffentlich zugänglichen Web-Server zu setzen, etwa aus den folgenden Gründen:

Sicherheit Der eigentliche Web-Server soll nicht für beliebige Pakete aus dem Internet zu erreichen sein. Ein Proxy nimmt alle Anfragen von außen entgegen und leitet sie an den Web-Server weiter. Dessen Antworten schickt der Proxy dann an den ursprünglichen Client zurück. Ein Spezialfall dieser Anwendung besteht darin, einen Proxy das SSL-Protokoll abhandeln zu lassen, falls der Web-Server das nicht selber kann. Der Proxy akzeptiert dann HTTPS-Anfragen, reicht diese per HTTP an den eigentlichen Web-Server durch und verschlüsselt die Antwort zur Rücksendung an den Client wieder mit SSL. Diese Vorgehensweise ist unbedenklich, da Proxy und Web-Server nicht über das öffentlich zugängliche Internet kommunizieren (häufig laufen sie auf demselben Rechner) und die Kommunikation zwischen ihnen nicht leicht zu kompromittieren ist, selbst wenn sie unverschlüsselt stattfindet. Ein Programm, das sich für diese Anwendung anbietet, ist Pound (Abschnitt 13.3.4).

Geschwindigkeit Ein cachender Reverse-Proxy kann einen Web-Server, der dynamische Inhalte liefert, von wiederholten Anfragen freihalten, die dieselbe Antwort bekommen müssen. Der Web-Server bekommt nur »neue« Anfragen zu sehen, während der Reverse-Proxy die zweite, dritte, ... Anfrage nach derselben Ressource aus seinem Cache beantworten kann.

Fehlertoleranz Ein Reverse-Proxy kann eine gemeinsame »Fassade« für eine Reihe von Web-Servern darstellen, die alle dieselben Inhalte anbieten. Anfragen von außen gehen an den Reverse-Proxy, der sie an einen der nachgeschalteten Server weiterleitet (Lastverteilung oder *load balancing*). Dabei kann auf der Basis verschiedener Kriterien entschieden werden, welcher Server das sein soll. Es ist in einer solchen Konfiguration auch nicht schlimm, wenn einer der nachgeschalteten Server ausfällt, da die übrigen Server dann transparent einspringen können; der Lastverteiler leitet an den ausgefallenen Server einfach keine Jobs mehr weiter.

13.2.4 Proxies und HTTPS

Die Idee hinter HTTPS, also per SSL bzw. TLS verschlüsselte Verbindungen zu einem Web-Server, lässt sich für unsere Zwecke hinreichend dahingehend zusammenfassen, dass HTTPS wirksam genau das verhindert will, was einen Proxy-Server interessant macht: Lesen (Logging), Manipulation (Filtern) und Identifizieren von Objekten (Caching), sowie das Vorspiegeln einer falschen Identität (*Man-in-the-middle*-Angriff), und damit im Grunde jeden Proxy selbst.

Ein »HTTPS-Proxy« muss sich also darauf beschränken, die Verbindung zum HTTP-Tunnel *unmanipuliert* durchzutunnellen. Um dieses Verhalten des Proxies zu

erreichen, benutzt der Client nicht GET sondern die Methode CONNECT, gefolgt von Hostangabe und Port (bei HTTPS standardmäßig 443):

```
CONNECT www.example.org:443 HTTP/1.0
```

Übungen

 **13.1** [1] Finden Sie heraus, wo und wie sich bei Ihrem Lieblingsbrowser der Betrieb mit Proxies konfigurieren lässt.

 **13.2** [3] Richten Sie durch netcat¹ zwei Pseudo-Server ein:

```
# netcat -l -p 80
```

für einen Pseudo-Webserver und

```
# netcat -l -p 8080
```

für einen Pseudo-Proxy.

Rufen Sie nun in einem Browser (bei abgeschaltetem Proxy) den URL `http://localhost` auf. Konfigurieren Sie in Ihrem Browser `http://localhost:8080` als Proxy und rufen Sie den URL `http://www.example.org` auf.

Vergleichen Sie beide Anfragen, d. h. die jeweilige Ausgabe von netcat. (Da netcat kein Web-Server ist, schickt es auch keine Antwort an den Browser zurück, es sei denn, Sie tippen sie selbst. Brechen Sie den Holvgang des Browsers ab oder warten Sie auf den Timeout.)

 **13.3** [3] Führen Sie Übung 13.2 für HTTPS durch: ersetzen Sie dazu Port 80 durch Port 443 und bei Anfragen »`http:`« durch »`https:`«.

13.3 Software

13.3.1 Allgemein

Als Betreiber eines Rechnernetzes können Sie unter diversen Proxy-Servern wählen. Einige sind frei verfügbar, andere kosten Geld; unter den kommerziellen Anbietern sind beispielsweise Firmen wie Microsoft oder Novell. Wir konzentrieren uns in dieser Unterlage auf die frei verfügbaren Angebote – zum einen ist es am wahrscheinlichsten, dass Sie diese zum Herumexperimentieren zur Verfügung haben, und zum anderen sind die freien Angebote so gut, dass vernünftigerweise eigentlich niemand auch nur einen müden Cent für einen proprietären Proxy-Server ausgeben müßte. Aber was zählt schon Vernunft ...

13.3.2 Squid

Squid ist ein cachender Proxy-Server, der neben HTTP auch die Protokolle FTP und gopher (Vorläufer von HTTP/HTML) unterstützt. Darüber hinaus kann er HTTPS und beliebige andere Protokolle über HTTP tunneln.

 Wenn es heißt »Squid unterstützt FTP und Gopher«, dann ist damit gemeint, dass er Ressourcen mit URLs à la `ftp://...` und `gopher://...` holen kann (viel Spaß dabei, letztere irgendwo noch zu finden). Clients sprechen mit Squid immer HTTP, niemals FTP oder Gopher.

¹Bei netcat (oder auch: nc) handelt es sich um ein stark bereinigtes »Telnet«-Programm, das anders als Telnet auch die Server-Seite übernehmen kann. Die genaue Beschreibung findet sich als README beim Paket, z. B. unter `/usr/share/doc/packages/netcat/README`; eine Kurzbeschreibung erhält man mit `netcat -h`.

Hervorgegangen ist Squid aus dem ARPA-finanzierten Harvest Project; er ist für alle gängigen Unix-Varianten verfügbar, einschließlich Cygwin (Unix-Home-Sweet-Home unter Microsoft Windows). Squid steht unter der GPL frei zur Verfügung, Sie können ihn also ohne Lizenzgebühren einsetzen und auch ändern und in originaler oder geänderter Form weiterverteilen, solange Sie den Empfängern dieselben Rechte einräumen. In der Praxis erhalten Sie Squid als Bestandteil einer Linux-Distribution oder können den Quellcode von der Squid-Homepage <http://www.squid-cache.org> herunterladen.

Neben der Einsatzmöglichkeit als cachender Proxy-Server, d. h. zwischen Client und dem Internet sitzend, kann Squid auch als HTTP-Beschleuniger (*HTTP accelerator*), manchmal auch *reverse proxy* genannt eingesetzt werden. Dabei sitzt Squid zwischen dem Internet und dem eigentlichen Web-Server, gibt sich aber nach außen als der Web-Server aus, um diesen zu entlasten und den HTTP-Zugriff dadurch zu beschleunigen.

Squid liegt zur Zeit in der stabilen Version 2.6.4 vor, die Nachfolgeversion 3.0 ist noch im Beta-Stadium. Für die hier besprochenen Konfigurationen ist aber allenfalls mit geringen Änderungen zu rechnen. Die Liste der Neuerungen findet sich auf der Squid-Homepage <http://www.squid-cache.org>.

13.3.3 Apache

Apache ist mit großem Abstand der populärste HTTP-Server im Internet und enthält auch Proxy- und Cache-Funktionalität für HTTP, FTP und einige andere Protokolle. Die aktuelle Version 2.2 bietet verschiedene Möglichkeiten zum Einsatz als Proxy und Reverse Proxy, mit oder ohne Cache und auch mit der Möglichkeit, als Lastverteiler aufzutreten.

13.3.4 Andere Programme

Hier im Telegrammstil noch ein paar andere Proxy-Programme:

BFilter Ein HTTP-Proxy, der Anzeigenbanner, Popups und Webbugs aus Web-Inhalten filtert. Er verwendet einen heuristischen Algorithmus zur Erkennung von zu blockierenden Inhalten; außerdem kann eine explizite Liste von URLs vorgegeben werden. BFilter kann Anfragen an einen anderen Proxy (etwa Squid) weiterreichen. Näheres siehe <http://bfilter.sourceforge.net>.

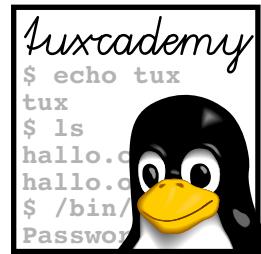
Pound Ein Proxy, der vor allem als Lastverteiler und SSL-Prozessor gedacht ist; Sie können damit Web-Server SSL-fähig machen, die das nicht von Haus aus sind, indem ein vorgeschalteter Pound-Server sich um die SSL-Behandlung kümmert und die Anfragen dann per HTTP an den eigentlichen Server weiterreicht. Die Antworten gehen dann über HTTPS an den ursprünglichen Client zurück. Siehe <http://www.apsis.ch/pound>.

Privoxy Ähnlich wie BFilter dient Privoxy dazu, die Privatsphäre und das Nervenkostüm der Anwender zu schonen. Er filtert kompromittierende Daten aus den HTTP-Anfragen und lästige Beigaben aus den Ressourcen heraus, kann Cookies verwalten und viele andere nützliche Dinge tun. Siehe <http://www.privoxy.org>.

Tinyproxy Ein extrem leichtgewichtiger, nicht cachender HTTP-Proxy, der sich besonders für Firewall-Systeme und ähnliche Situationen anbietet, wo große und komplexe Programm wie Squid oder Apache möglicherweise unerwünscht sind. Siehe <http://tinyproxy.sourceforge.net>.

Zusammenfassung

- Clients benutzen absolute URIs, wenn sie sich mit einem HTTP-Proxy verbinden, und absolute Pfadangaben sonst.
- HTTPS-Verbindungen werden vom Client beim Proxy durch CONNECT eingeleitet. Der Proxy tunneln solche Anfragen unverändert durch.



14

Squid-Konfiguration

Inhalt

14.1	Basis-Einrichtung von Squid	180
14.1.1	Vorbereitende Schritte	180
14.1.2	Squid testen.	182
14.1.3	Squid steuern	183
14.1.4	Laufzeitverhalten kontrollieren.	183
14.2	Squid als clientseitiger Cache	184
14.2.1	Cache-Größe und Speicher-Zuweisung	184
14.2.2	Cache-Statistiken	185
14.2.3	Cache-Hierarchien	186
14.2.4	Auswertung des Zugriffsprotokolls	187
14.3	Zugriffskontrolle	188
14.3.1	Struktur der Zugriffskontrolle	188
14.3.2	Typen von Bedingungen	190
14.3.3	Regeln und Aktionen	191

Lernziele

- Den Proxy-Server Squid konfigurieren können

Vorkenntnisse

- Elementares Verständnis von HTTP
- Grundkenntnisse über Proxies (Kapitel 13)

14.1 Basis-Einrichtung von Squid

squid.conf Squid ist ein recht pflegeleichtes Produkt. Er wird nur durch eine einzige Datei konfiguriert: `squid.conf`, die sich je nach Distribution in `/etc` oder `/etc/squid` findet. Zwar bietet sie eine Fülle an Konfigurationsmöglichkeiten, diese dienen aber der Feinabstimmung und Spezialanwendungen. Squid ist normalerweise sofort lauffähig, lediglich ein paar Handgriffe sind erforderlich.

 Die Syntax-Hervorhebung von `vim` unterstützt neben der Konfigurationsdatei von Apache auch die von Squid. Sie wird in `/etc/vimrc` oder `~/.vimrc` durch die Zeile »`syntax on`« aktiviert.

14.1.1 Vorbereitende Schritte

Im Folgenden werden die Schritte, die zum Betrieb von Squid nötig sind, an Hand eines selbstkompilierten Squid durchgegangen. Bei Ihrer Distribution wird wahrscheinlich ein Squid dabei sein, und einige der Schritte wurden Ihnen schon abgenommen. Dennoch sollten Sie überprüfen, ob die Einstellungen Ihren Anforderungen entsprechen.

Squid kompiliert problemlos¹ und installiert sich unterhalb von `/usr/local/squid`. Alle Dateien finden sich unterhalb dieses Verzeichnisses, inklusive seiner Cache-Dateien, d. h. `/usr/local` muss bei Eigenbau-Squids beschreibbar sein. Im Folgenden werden aber der Einfachheit halber Pfadangaben benutzt, wie sie sich (wahrscheinlich) bei einem vorkonfektioniertem Squid finden.

Effektive Benutzer-ID Aus Sicherheitsgründen läuft Squid nicht mit `root`-Rechten, selbst wenn er von `root` gestartet wurde (da typische Proxy-Ports unprivilegiert sind, ist nicht einmal das nötig). Voreingestellt ist hier der Benutzer `nobody`, besser ist es allerding einen für Squid reservierten Benutzer mit eigener Gruppe zu wählen:

```
# groupadd -r squid
# useradd -r -g squid -d /var/cache/squid -s /bin/false -p '*' squid
```

Hiermit wird die neue Systemgruppe `squid` angelegt, sowie der neue Systemaccount `squid`, der nur Mitglied der Gruppe `squid` ist und aus Sicherheitsgründen eine ungültige Shell (`/bin/false`) sowie das ungültige Kennwort (*) erhält. Als Heimatverzeichnis des Squid-Benutzers wurde hier sein Cache-Verzeichnis (s. u.), also `/var/cache/squid`, gewählt.

Jedes Cache-Verzeichnis muss angelegt worden sein, bevor Squid startet, da er einem dies nicht abnimmt, und es muss für Squid beschreibbar sein:

```
# mkdir /var/cache/squid
# chown squid:squid /var/cache/squid
# chmod 600 /var/cache/squid
```

Die Einschränkung der Rechte für normale Benutzer verhindert den direkten Zugriff auf den Cache, beispielsweise um das Surf-Verhalten anderer Benutzer auszuspionieren.

Entsprechendes gilt für das Verzeichnis, in dem Squid seine Protokolldateien ablegt: es muss existieren und für Squid beschreibbar sein und es sollte nicht weltweit lesbar sein. Den Pfad der einzelnen Logdateien entnimmt man der vordidlich kommentierten Konfigurationsdatei `squid.conf`; die relevanten Direktiven heißen `cache_access_log`, `cache_log` und `cache_store_log`.

Schließlich müssen Sie Squid noch mitteilen, dass er die neue Identität auch benutzen soll. Dazu setzen Sie in der Konfigurationsdatei die Direktive `cache_effective_user` auf `squid`:

¹Mit dem üblichen Kompilierungs-Dreisprung `./configure`, `make` und `make install`; die getestete Version war 2.5STABLE4.

```
cache_effective_user squid
```

(Die Angabe der Gruppe durch `cache_effective_group` ist nicht zwingend nötig; `cache_effective_group` Squid nimmt die primäre Gruppe des gewählten Accounts.)

Cache-Verzeichnisse Da eine der Hauptaufgaben von Squid darin besteht, Anfragen aus seinem Cache zu bedienen, sollte das Cache-Verzeichniss unbedingt auf einer schnellen Festplatte liegen. Es können auch mehrere Verzeichnisse (natürlich auf verschiedenen Festplatten) angegeben werden, um durch Parallelzugriff die Performanz zu erhöhen.

Festgelegt werden die Cache-Verzeichnisse in der Konfigurationsdatei durch die `cache_dir`-Direktive, die als erstes Argument den Cache-Typ verlangt. Der in `cache_dir` Squid eingebaute Typ ist `ufs`, es sind aber auch andere Typen möglich, die unter Umständen externe Programme benötigen und einen entsprechend kompilierten Squid voraussetzen.

Bei Cache-Verzeichnissen vom Typ `ufs` folgt als weitere Argumente der absolute Pfad zum Verzeichnis, die maximale Größe des Caches in Megabyte sowie die Anzahl der Unter- und Unterunter-Verzeichnisse im Cache. Zusammen sieht das Ganze für einen 100 MB großen Cache dann z. B. so aus:

```
cache_dir ufs /var/cache/squid 100 16 256
```

Die Strukturierung in 16 Unter-Verzeichnisse mit jeweils 256 Unterunter-Verzeichnissen kann unverändert bleiben. Sie ist erforderlich, damit sich nicht zu viele Dateien in einem Verzeichnis sammeln, was manche Dateisystemtypen extrem ausbremsen kann.

 Wenn Sie in der Position sind, dem Squid-Cache ein eigenes Dateisystem zu spendieren (LVM ist immer eine gute Idee), dann sollten Sie überlegen, dieses Dateisystem als ReiserFS zu betreiben. ReiserFS ist verhältnismäßig effizient im Umgang mit kleinen Dateien, und die Anzahl der Dateien pro Verzeichnis ist im wesentlichen egal.

Vor der ersten Benutzung eines Cache-Verzeichnisses und nach jeder Änderung der `cache_dir`-Einträge müssen die Cache-Verzeichnisse durch den Aufruf von

```
$ squid -z
```

initialisiert werden.

 Die Größenangabe des Cache-Verzeichnisses bezieht sich auf die Größe der eingelagerten Objekte, nicht auf den real verbrauchten Plattenplatz. Insbesondere wenn Squid eine Festplatte für sich alleine erhält, sollte sie deutlich unter der nominellen Plattenkapazität liegen; empfohlen wird 80% der Plattenkapazität.

Interfaces und Ports Ohne Änderung lauscht Squid auf Port 3128 auf allen Interfaces. Bevorzugt man beispielsweise den ebenfalls gebräuchlichen Port 8080, so kann dies durch die Direktive `http_port` geändert werden. Durch das Voranstellen eines Rechnernamens oder einer IP-Adresse wird Squid veranlasst, nur auf der entsprechenden Schnittstelle zu lauschen. Beispielsweise bietet er seine Dienste mit

```
http_port localhost:8080
```

nur rechnerintern auf Port 8080 an. Die Direktive `http_port` darf mehrfach angegeben werden; Squid lauscht dann auf jedem der angegebenen Ports (und Schnittstellen).

 Manche Leute schwören darauf, ihren Proxy auf dem HTTP-Standardport 80 zu betreiben. Das hat entweder historische Gründe – in der Anfangszeit des WWW diente der angesagte Web-Server, der vom CERN (wo das WWW erfunden wurde), auch als Proxy – oder liegt daran, dass diese Leute die Proxy-Implementierung von Microsoft verwenden, die auf IIS aufbaut und darum Port 80 verwendet. Im allgemeinen ist das keine besonders schlaue Idee, da es Sie darauf festlegt, nur Web-Server zu benutzen, die auch als Proxy agieren können (oder umgekehrt nur Proxy-Server, die auch als Web-Server agieren können) – die Alternative wäre, alle Clients umzukonfigurieren, und das ist mitunter ärgerlich. Wenn Ihr Proxy auf einem anderen Port läuft als Ihr Web-Server, dann können Sie sich für jede Anwendung das Programm aussuchen, das Ihnen am besten gefällt, ohne dafür die andere Anwendung zu präjudizieren.

 Notfalls können Sie Squid dazu bringen, auf Port 80 zu laufen und alle Anfragen, die nicht aussehen wie Proxy-Anfragen (also keine absoluten URIs enthalten) an einen anderen Web-Server weiterzuleiten (der zum Beispiel auf demselben Rechner auf einem anderen Port laufen kann). Dies ist ein Spezialfall eines *reverse proxy*.

Die ebenfalls vorhandene Direktive `https_port` muss *nicht* gesetzt werden, selbst wenn der Proxy HTTPS bearbeiten können soll: Wir hatten gesehen, dass der Client seinem Wunsche nach HTTPS durch Benutzung der Methode `CONNECT` Ausdruck verleiht und nicht durch Kontaktieren eines anderen Ports. Diese Direktive spielt lediglich eine Rolle, wenn der Proxy als HTTP-Beschleuniger eingesetzt wird.

Zugriffskontrolle Ein frisch kompilierter Squid erlaubt niemandem den Zugriff. Sofern Ihre Distribution das nicht schon für Sie erledigt hat, muss also irgendeine Form von Zugriff erlaubt werden. Beispielsweise können direkt vor der Zeile

```
http_access deny all
```

in der Konfigurationsdatei die Zeilen

```
acl our_networks src 192.168.1.0/24
http_access allow our_networks
```

eingefügt werden. (Das setzt natürlich voraus, dass Ihr Netz gerade 192.168.1.0/24 ist.) Zugriffskontrollen werden ausführlich im Abschnitt 14.3 behandelt.

 Man sollte keinesfalls den Zugriff mehr als nötig freigeben, insbesondere nicht weltweit. Es besteht sonst die Gefahr, dass der Proxy missbraucht wird: sei es um auf Ihre Kosten Bandbreite zu schonen, sei es um die eigene Identität für Angriffe zu verschleiern.

14.1.2 Squid testen

Wenn Sie (oder Ihre Distribution) Squid soweit vorbereitet haben, können Sie ihn testen. Dazu rufen Sie ihn am besten durch

```
$ squid -N -d 1 -D
```

auf. Dabei verhindert die Option `-N`, dass sich Squid von der Konsole löst und `»-d 1«` gibt Debugging-Meldungen aus, die sich auch im Protokoll (Direktive: `cache_log`) finden. Normalerweise überprüft Squid beim Start, ob die Namensauflösung funktioniert. Er ist zufrieden, falls er wenigstens eine der durch die Direktive `dns_testnames` angegebenen Domänen auflösen kann. Die Option `-D` schaltet diesen Test ab. Alle Optionen von Squid stehen mit Kommentaren in einer Datei namens `squid.conf.default`; wo Ihre Distribution diese Datei unterbringt (sie tut es hoffentlich), müssen Sie selbst herausfinden. Bei Debian GNU/Linux heißt sie zum Beispiel `/usr/share/doc/squid/examples/squid.conf`.

14.1.3 Squid steuern

Um Squid dauerhaft einzurichten, also ihn etwa automatisch beim Booten zu starten, sollte Ihre Distribution ein Skript mitgeliefert haben. Mit diesem Skript können Sie Squid wahrscheinlich auch weitgehend steuern. Alternativ kann eine laufende Squid-Instanz durch den Aufruf eines weiteren Squid gesteuert werden. Die Option `-k` gefolgt von einem Schlüsselwort bewirkt, dass Squid an die laufende Squid-Instanz ein Signal sendet. Beispielsweise wird durch

```
$ squid -k reconfigure
```

der laufende Squid dazu gebracht, seine Konfigurationsdatei neu einzulesen. Gültige Schlüsselwörter sind:

reconfigure Neueinlesen der Konfigurationsdatei. Äquivalent zum Senden des Signals `SIGHUP` (mittels `kill`).

rotate Rotieren der Log-Dateien. Ist unnötig, wenn die Distribution eigene Mittel bereitstellt, z. B. `logrotate`.

shutdown Herunterfahren von Squid. Bestehende Verbindungen werden in einer Schonfrist noch abgearbeitet. Äquivalent zum Senden von `SIGTERM`.

interrupt Runterfahren von Squid. Bestehende Verbindungen werden sofort gekappt. Äquivalent zum Senden von `SIGINT`.

kill Äquivalent zum Senden von `SIGKILL`.

debug Schaltet Debugging-Meldungen an oder aus.

check Überprüft, ob Squid läuft.

parse Sendet kein Signal an Squid, sondern überprüft die Konfigurationsdatei auf syntaktische Fehler.

14.1.4 Laufzeitverhalten kontrollieren

Wie bei jedem anderen Server auch stellt die Log-Datei die erste Anlaufstelle dar, wenn es beim Betrieb von Squid zu Problemen kommt. Ihr Pfad wird durch die Direktive `cache_log` festgelegt. Reichen die dort gefundenen Meldungen nicht aus, so kann mit `debug_options` die Geschwätzigkeit erhöht werden. Voreingestellt ist

`cache_log`
`debug_options`

```
debug_options ALL,1
```

wobei damit für alle Abschnitte der Level auf 1 gesetzt wird. Der maximale Level ist 9. (Jeder Datei des Quellcodes von Squid ist eine Abschnittsnummer zugeordnet, unter der dort auftretende Fehler protokolliert werden. Beispielsweise ist dies 9 für `ftp.c`, und so kann mit

```
debug_options ALL,1 9,2
```

der Level nur für FTP hochgesetzt werden. Wir verzichten darauf, hier alle 80 Abschnittsnummern aufzulisten; sie stehen in den Quellcode-Dateien immer ganz am Anfang.)

Übungen



14.1 [!1] Kontrollieren Sie wie bei Ihrer Distribution die Basis-Einstellungen ausgeführt wurden: wo liegt die Konfigurationsdatei, was sind Cache- und Log-Verzeichnisse, unter welchem Benutzer läuft Squid usw.



14.2 [2] Ändern Sie die Strukturierung des Cache-Verzeichnisses und initialisieren Sie es danach neu.



14.3 [!2] Starten Sie Squid von Hand und überprüfen Sie sein korrektes Starten durch `ps` und `netstat`. Steuern Sie ihn durch einen anderen Squid mittels der Option `-k` und überprüfen Sie die Reaktion. Testen Sie nun den Proxy durch `wget` oder Ihren Lieblingsbrowser.



14.4 [3] Schauen Sie sich das Skript ihrer Distribution (`/etc/init.d/squid` o. ä.) an, mit dem Squid gestartet wird. Mit welchen Optionen wird Squid aufgerufen und was bedeuten sie?



14.5 [3] Setzen Sie den Debug-Level in der Konfigurationsdatei hoch und starten Sie Squid neu. Provozieren Sie fehlerhafte Anfragen u. ä. und verfolgen Sie Squids Äußerungen in der Log-Datei.

14.2 Squid als clientseitiger Cache

Das Ziel eines clientseitigen Caches ist es, die Arbeit der Clients zu beschleunigen. Statt sich jedesmal die Objekte (Webseiten, Dateien) vom Web- oder FTP-Server zu laden, wird der Proxy-Server kontaktiert, der dann hoffentlich den Client aus seinem Cache bedienen kann oder sich andernfalls selbst im Netz auf die Suche macht. Diese Konstellation ist natürlich nur sinnvoll, wenn Proxy und Client durch eine schnelle Leitung verbunden sind.

An der Basis-Konfiguration von Squid ist nichts zu ändern, da dieses Szenario gerade durch die Kernfunktionalität von Squid abdeckt wird. Bei allen folgenden Einstellungen handelt es sich nur um Fein-Tuning, also um Maßnahmen, deren Ziel es ist, die Treffer-Quote in die Höhe zu treiben, soll heißen, den Anteil der Anfragen, die Squid selbst beantworten kann.

14.2.1 Cache-Größe und Speicher-Zuweisung

Die Größe des Cache-Verzeichnisses hatten wir bereits bei der Basis-Konfiguration durch die `cache_dir`-Direktive festgelegt. Wenn der Plattenplatz knapp wird, d. h.

`cache_swap_low` wenn er den durch `cache_swap_low` angegebenen Prozentsatz überschreitet, dann fängt Squid an, Objekte von der Festplatte zu löschen. Dabei geht Squid um so aggressiver vor, je näher sich die Auslastung dem durch die Direktive `cache_swap_high` angegebenen Prozentsatz nähert. Welche Objekte gelöscht werden, hängt vom gewählten Algorithmus ab, der durch die Direktive `cache_replacement_policy` festgelegt wird. Dadurch können häufig nachgefragte, kleine oder große Objekte bevorzugt werden. Die folgenden Werte für `cache_replacement_policy` sind zulässig:

lru Die *least-recently-used*-Strategie verdrängt mit Vorliebe Objekte aus dem Cache, die lange nicht nachgefragt wurden.

heap LRU LRU-Strategie auf der Basis eines Heaps statt der traditionellen Liste.

heap GDSF Mit der *greedy-dual-size-frequency*-Strategie werden kleine häufig nachgefragte Objekte behalten. Hier kann es passieren, dass große häufig nachgefragte Objekte gelöscht werden, so dass die für den Betrieb benötigte Bandbreite steigt – es kostet mehr, ein großes populäres Objekt neu zu holen als ein kleines.

heap LFUDA Mit der *least-frequently-used-with-dynamic-aging*-Strategie werden populäre Objekte ungeachtet ihrer Größe im Cache belassen. Dies erhöht die Trefferwahrscheinlichkeit, aber die Anwesenheit eines großen populären Objekts kann dafür sorgen, dass für viele kleine populäre Objekte kein Platz mehr da ist. Wenn Sie LFUDA verwenden, sollten Sie überlegen, die Obergrenze für die Größe von Objekten, die überhaupt im Cache abgelegt werden (`maximum_object_size`), gegenüber dem voreingestellten Wert von 4 Mebibyte anzuheben.

Mehr Informationen über die verfügbaren Algorithmen und weitergehende Verweise finden sich in der kommentierten Konfigurationsdatei.

Besser noch als viel Festplattenplatz für den Cache ist viel RAM. Denn Squid versucht häufig nachgefragte Objekte im RAM zu halten; dadurch wird die Reaktionszeit drastisch beschleunigt. Die Größe des hierfür benutzten RAM-Bereichs wird durch `cache_mem` festgelegt. Bei Überschreitung werden Objekte aus dem RAM entfernt, wobei der benutzte Algorithmus durch `memory_replacement_policy` angegeben wird. Die Auswahlmöglichkeiten sind die gleichen wie bei `cache_replacement_policy`.

 Der durch `cache_mem` festgesetzte RAM-Bereich ist *nicht* die Gesamtgröße des Squid-Prozesses. Hinzu kommen vielmehr noch im RAM gehaltene Caches für DNS-Anfragen u. a. Außerdem schlägt jedes Objekt im (Festplatten-)Cache mit 76 Byte (bei 64-Bit-Prozessoren 104 Byte) im RAM für Verwaltungsdaten zu Buche.

Es ist daher nicht möglich, Squid zwar mit viel Plattenplatz auszustatten, aber beim RAM zu geizen.

 Im Detail benutzt Squid Cache-RAM für drei Sorten von Objekten: Objekte, die gerade geholt und weitergereicht werden (engl. *in-transit objects*); viel gefragte Objekte; Objekte, von denen bekannt ist, dass es sie nicht gibt, und wo man sich sparen möchte, das immer wieder aufs Neue herauszufinden (der »negative Cache«). Die *in-transit objects* haben Vorrang gegenüber den anderen Sorten von Objekten; letztere werden aus dem RAM-Speicher geworfen, wenn ihr Platz für erstere gebraucht wird. Squid behält sich vor, den mit `cache_mem` angegebenen Speicher zu überschreiten, wenn der insgesamt für *in-transit objects* benötigte Platz größer ist als der Vorgabewert, wird aber den übermäßig beanspruchten Platz baldmöglichst wieder freigeben.

 Squid holt sich Cache-RAM in Happen von 4 KiB; `cache_mem` gibt also in Vielfachen von 4 KiB an, wie viel RAM Squid im langfristigen Mittel belegen soll. Squid versucht, diesen Speicher nicht brachliegen zu lassen; Platz, der nicht für *in-transit objects* gebraucht wird, wird mit viel gefragten Objekten und dem »negativen Cache« aufgefüllt. Sie sollten also nicht damit rechnen, dass Squid weniger Speicher schluckt als in `cache_mem` angegeben, sondern – wie im vorigen Absatz erwähnt – tendenziell eher mehr.

14.2.2 Cache-Statistiken

Sie können die Treffer-Quote Ihres Caches nur optimieren, wenn Sie auch die Auslastung des Caches kennen. Squid stellt zu diesem Zweck eine eigene Schnittstelle zur Verfügung: URIs, die statt mit `http:` mit `cache_object:` anfangen, werden von Squid als Anfragen nach dem Zustand des Caches interpretiert. Probefahrer können Sie diese Schnittstelle selbst anzapfen:

```
$ telnet localhost 3128
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^].
GET cache_object://localhost/info HTTP/1.0
```

Dabei müssen Sie die letzte Zeile wie bei jeder HTTP-Anfrage mit *zwei* Zeileinschaltungen abschließen. Squid antwortet mit einer umfangreichen Statistik.

 In der Squid-Distribution ist das CGI-Skript `cachemgr.cgi` enthalten. Von einem Web-Server aufgerufen erlaubt es in komfortabler Weise, Squids Statistiken auszulesen. Um es benutzen zu können, muss man es nur in das `cgi-bin`-Verzeichnis seines Web-Servers kopieren.

14.2.3 Cache-Hierarchien

Um nicht (potenziell) das ganze Internet bei sich im Cache haben zu müssen, können sich mehrere Proxies die Arbeit teilen. Dazu werden Proxies in Hierarchien eingefügt. Kann ein Proxy eine Anfrage nicht direkt beantworten, so fragt er erst seine Geschwister- und Eltern-Proxies, ob sie das gewünschte Objekt nicht in ihrem Cache haben. Erst wenn ihm seine Verwandten nicht aushelfen können, versucht er sein Glück direkt beim ursprünglichen Web-Server. Durch dieses Vorgehen werden unter dem Strich die Caches aller beteiligten Proxies zusammengefasst, ohne dass alle Objekte in alle Caches kopiert werden müssten. Dadurch wird der verfügbare Platz besser ausgenutzt.

 In der Anfangszeit des World-Wide Web versprach man sich einiges von umfangreichen Cache-Hierarchien. In der Praxis ist der Ausbau der Netzinfrastruktur so rapide vorangeschritten, dass der Aufwand zur Einrichtung und Wartung großer Cache-Systeme sich nicht wirklich automatisch lohnt. Allerdings kann es in großen Organisationen wie Firmen und Universitäten durchaus sinnvoll sein, existierende Caches zusammenzuschalten.

Direktiven Für den Aufbau von Cache-Hierarchien unterstützt Squid zwei Arten von Direktiven. Die `cache_peer`-Direktive dient zur Beschreibung der Caches in der Hierarchie (aus der Sicht des lokalen Squid). Weitere Direktiven steuern die Weiterleitung von Anfragen an verschiedene andere Caches in der Hierarchie.

cache_peer Betrachten wir zunächst die `cache_peer`-Direktive an einem Beispiel:

```
cache_peer parent.example.com parent 3128 3130 proxy-only
cache_peer sibling.example.com sibling 3128 3130 proxy-only
```

Cache-Name Das erste Feld nach dem `cache_peer` gibt den Namen des Rechners an, auf den die Direktive sich bezieht. Das zweite nennt die Rolle, die der benannte Rechner bezogen auf den lokalen Squid spielt – `parent` (»Elter«) oder `sibling` (»Geschwist«).

 Der Unterschied zwischen `parent`- und `sibling`-Caches besteht darin, dass Squid zuerst die `siblings` nach einem Objekt fragt. Erst wenn klar ist, dass kein `sibling` über das gesuchte Objekt verfügt, geht die Frage an die `parents` weiter. Ein `parent` wird im Gegensatz zu einem `sibling` auch gefragt, wenn er die Ressource *nicht* hat, und muss sie dann eben besorgen.

 Sie haben sich im vorletzten Satz nicht verlesen: Niemand hält Sie davon ab, mehr als einen anderen Cache als `parent` zu kennzeichnen. Squid sucht sich dann für jede Anfrage einen der beiden aus, typischerweise den, der schneller geantwortet hat oder, falls nur einer die Ressource vorrätig hat, natürlich diesen.

Proxy-Port Das dritte Feld gibt den Port an, auf dem der entfernte Proxy auf HTTP-Anfragen (à la Proxy) reagiert. Mit dem vierten Feld können Sie einen Port für das *Inter-Cache Protocol* (ICP) definieren. Dies ist ein UDP-basiertes Protokoll, das Caches wie Squid untereinander sprechen, um sehr schnell herausfinden zu können, ob ein entfernter Cache eine Kopie einer gesuchten Ressource vorrätig ist. Der übliche Port für ICP ist 3130.

 Wenn Sie Anfragen an einen Cache weiterleiten wollen, der ICP nicht unterstützt, können Sie als Portnummer »7« angeben und dafür sorgen, dass auf dem entfernten Rechner das UDP-ECHO-Protokoll aktiviert ist. Das ist allerdings ein übler Hack, und solche Caches kommen auch nur als `parent` in Frage.

Optionen Nach dem ICP-Port können Sie noch Optionen angeben, die die Beziehung zwischen dem lokalen Squid und dem entfernten Cache näher beschreiben. »proxy-only« zum Beispiel bewirkt, dass Ressourcen, die von dem entfernten Cache

geholt wurden, nicht noch einmal lokal abgespeichert werden (das ist einer der Gründe, warum man überhaupt eine Cache-Hierarchie haben möchte). Eine komplette Liste der möglichen Optionen mit Erklärungen findet sich zum Beispiel in der kommentierten Konfigurationsdatei.

Squid unterstützt einige weitere Direktiven, um die Interaktion der Caches zu steuern. Hier eine Auswahl:

cache_peer_domain Diese Option gibt an, dass ein bestimmter Cache nur nach bestimmten Domains gefragt werden soll. Zum Beispiel sorgt

```
cache_peer_domain d-a-ch.example.com .de .at .ch
```

dafür, dass der Cache von `d-a-ch.example.com` nur nach Ressourcen in den Domains `.de`, `.at` und `.ch` gefragt wird (per ICP). Caches ohne Domain-Einschränkung werden nach allen Ressourcen gefragt. Vor einer Domain darf auch ein `»!«` stehen, dann wird der Cache nach allen Ressourcen gefragt, die *nicht* in dieser Domain sind.

neighbor_type_domain Hiermit können Sie die Rolle eines Caches in Abhängigkeit von der gesuchten Domain ändern. Stellen Sie sich zum Beispiel vor, Ihre Firma hat zwei Büros (das Stammhaus und eine Filiale) in Deutschland und eins in der Schweiz und verwendet zwischen diesen Standorten. In jedem Büro existiert ein Squid-Cache. Der Squid in der Filiale Deutschland könnte nun WWW-Seiten von Schweizer Servern über den Squid in der Schweiz holen (weil der näher am Schweizer Internet ist) und andere WWW-Seiten über den Squid im Stammhaus (der größer und dicker ist). Die Konfiguration für dieses zugegebenermaßen etwas konstruierte Beispiel könnte so aussehen:

```
cache_peer      de-main.example.com parent 3128 3130 proxy-only
cache_peer      ch.example.com sibling 3128 3130 proxy-only
neighbor_type_domain ch.example.com parent .ch
```

Das heißt, für Ressourcen in `.ch` wird der Cache von `ch.example.com`, der normalerweise als `sibling` angesprochen wird, zum `parent` erklärt.

dead_peer_timeout Gibt eine Anzahl von Sekunden an, die Squid auf eine ICP-Nachricht von einem anderen Cache wartet, bevor er ihn für tot erklärt. Ein für tot erklärter Cache bekommt weiterhin ICP-Nachrichten geschickt, aber Squid rechnet nicht mehr mit Antworten. Wenn der Cache irgendwann aufersteht, wird er vom lokalen Squid einfach wieder in die Reihen der Lebenden aufgenommen.

14.2.4 Auswertung des Zugriffsprotokolls

Die Auswertung der Anfragen an den Proxy kann aus mehreren Gründen interessant sein: Zum einen können Sie damit clientseitige Probleme (Dinge wie »Zugriff verweigert«) besser eingrenzen, zum anderen können Sie die Cache-Leistung besser optimieren, wenn Sie das Spektrum der Anfragen kennen.

Squid legt das Zugriffsprotokoll in der Datei abgelegt, die durch die Direktive `cache_access_log` bestimmt wird. Wollen Sie die Log-Datei mit Werkzeugen auswerten, wie sie für Web-Server gebräuchlich sind (beispielsweise Webalizer), so müssen Sie das Format durch `emulate_httpd_log` umstellen. Dabei gehen aber Informationen verloren, die ansonsten in Squids eigenem Dateiformat enthalten sind. Den vollen Nutzen aus der Log-Datei ziehen Sie daher nur, wenn Sie Analyzse-Werkzeuge benutzen, die auf Squid zugeschnitten sind. Ein solches Werkzeug ist `calamaris`. Mit dem Aufruf

`cache_access_log`

`emulate_httpd_log`

```
# cd /var/log/squid
# calamaris -a -F html <access.log>calamaris.html
```

wird aus Squids Log-Datei ein vollständiger (-a) Report in HTML (-F html) erzeugt.

Übungen

-  **14.6** [2] Fragen Sie die Statistiken eines laufenden Squids wie im Abschnitt 14.2.2 beschrieben ab. Treiben Sie die Treffer-Rate in die Höhe, indem Sie immer wieder die gleiche Seite abrufen. Kontrollieren Sie das Ergebnis durch eine erneute Statistik-Abfrage.
-  **14.7** [3] Richten Sie einen Apache-Web-Server so ein, dass Sie das Skript `cachemgr.cgi` benutzen können und testen Sie es.
-  **14.8** [2] Erzeugen Sie mit Calamaris einen Report über die Zugriffe auf den Proxy. Verleihen Sie das Ergebnis mit dem von `cachemgr.cgi` erzeugten Report.
-  **14.9** [1] Suchen Sie in Calamarisens Manpage `calamaris(1)` nach allen Optionen, die mit einer Warnung versehen sind und machen Sie sich deren datenschutzrechtliche Relevanz klar.

14.3 Zugriffskontrolle

Squid erlaubt eine sehr fein abgestimmte Zugriffskontrolle. Die Möglichkeiten gehen dabei über die von Paketfiltern oder TCP-Wrappern hinaus, weil Squid eben HTTP beherrscht. Damit empfiehlt es sich, Squid als *Application Level Gateway* einzusetzen (Kapitel 15). Aber auch für den »Normalbetrieb« sind einige grundlegenden Zugriffskontrollen nötig. In diesem Abschnitt erklären wir den allgemeinen Aufbau von Squids Mechanismus zur Zugriffskontrolle.

14.3.1 Struktur der Zugriffskontrolle

Squids Zugriffskontrolle gewährt oder verweigert nicht nur den Zugriff auf Web-Ressourcen, sondern auch auf andere Interna des Proxies. Deswegen ist es besser, über »Aktionen« zu reden, die über die Zugriffskontrolle erlaubt oder verboten werden. Eine davon ist der Zugriff auf eine Web-Ressource, aber Sie werden auch noch andere kennenlernen.

Die Kontrolle selbst ist in vier Schichten aufgebaut: Bedingungs-Typen, Bedingungen, Regeln und Regel-Listen.² Damit die Diskussion aber nicht zu abstrakt gerät, gehen wir zunächst das folgende Beispiel durch: *Der Zugriff auf den Proxy soll für das eigene Netz 192.168.1.0/24 erlaubt sein, für alle anderen verboten.* Als (hypothetische) Konfiguration für den TCP-Wrapper sähe das wie folgt aus:

```
## aus /etc/hosts.allow, nicht funktionsfähig!
squid: 192.168.1.0/255.255.255.0: ALLOW
squid: ALL: DENY
```

Bedingungen und Bedingungs-Typen Da Squids Zugriffskontrolle mehr Möglichkeiten bietet, müssen Sie hier auch mehr arbeiten. Zunächst müssen Sie zwei Bedingungen definieren:

```
acl our_networks src 192.168.1.0/24
acl all src 0.0.0.0/0
```

Eine Bedingung im Sinne von Squid sieht immer so aus:

²Leider ist die Terminologie keinesfalls einheitlich, nicht einmal in den verschiedenen Quellen zur Squid-Dokumentation.

```
acl <Name> <Typ> <Wert> ...
```

<Name> ist der (eindeutige) Name der Bedingung, unter dem sie später verwendet werden kann. Die Bedingung trifft immer genau dann zu, wenn das, was durch <Typ> näher bestimmt wird, einen der Werte <Wert>... annimmt. In unserem Beispiel ist der Bedingungs-Typ `src`, und das ist gerade die IP-Adresse des anfragenden Clients. Bedingungs-Typ

 Jeder Bedingungsname darf eigentlich nur einmal benutzt werden. Kommen zwei Definitionen gleichen Namens vor, so müssen sie vom gleichen Typ sein, und sie werden dann von Squid als eine aufgefasst:

```
acl our_networks src 192.168.1.0/24
acl our_networks src 192.168.2.0/24
```

ist äquivalent zu

```
acl our_networks src 192.168.1.0/24 192.168.2.0/24
```

Mit anderen Worten werden die möglichen Werte der Bedingung ODER-verknüpft – in diesem Beispiel muss die IP-Adresse des Clients entweder im Netz 192.168.1.0/24 oder aber im Netz 192.168.2.0/24 liegen.

Sobald eine Bedingung definiert wurde, kann sie in Zugriffsregeln benutzt werden.

Regeln und Regel-Listen Jede Regel entscheidet darüber, ob eine Aktion ausgeführt oder unterlassen wird. Sie wird deklariert durch eine Zeile der Form Regeln

```
<Aktion> allow <Bedingung> ...
```

oder

```
<Aktion> deny <Bedingung> ...
```

Die möglichen Aktionen werden dabei durch Squid-Direktiven vorgegeben; die Aktion `http_access` regelt zum Beispiel den Zugriff auf Web-Ressourcen über HTTP. Soll die Aktion erlaubt werden, so folgt `allow`, soll sie verweigert werden ein `deny`. Die Regel ist aber nur anwendbar, wenn die Anfrage alle aufgeführten Bedingungen gleichzeitig erfüllt (UND-Verknüpfung). Im vorliegenden Beispiel sieht das so aus:

```
http_access allow our_networks
http_access deny all
```

Wichtig ist hierbei die Reihenfolge: Alle Regeln zu einer Aktion bilden die für diese Aktion relevante Regel-Liste. Die *erste* Regel, deren Bedingungen auf die betrachtete Anfrage passen, entscheidet, ob die Aktion durchgeführt wird. Also ist

```
http_access deny all
http_access allow our_networks
```

gleichbedeutend mit

```
http_access deny all
```

denn die Bedingung »all« passt – wie oben angegeben – auf alle Anfragen mit irgendeiner beliebigen IP-Adresse; die »allow our_networks«-Bedingung wird überhaupt nicht mehr erreicht.

In einer Regel können Sie auch Bedingungen unterschiedlichen Typs verknüpfen. Mit

```
acl SSL_ports port 443 563
acl CONNECT method CONNECT
```

werden zwei Bedingungen definiert. Die erste Bedingung trifft zu, wenn der angefragte Port entweder gleich 443 oder gleich 563 ist (die üblichen Ports für HTTPS und NNTPS). Die zweite Bedingung trifft zu, wenn der Client einen TCP-Tunnel aufbauen, also als Methode CONNECT und nicht GET & Co. verwenden möchte. Diese beiden Bedingungen können Sie dann zur Regel

```
http_access deny CONNECT !SSL_ports
```

kombinieren, die den Zugriff verweigert, wenn der Client einen TCP-Tunnel aufbauen möchte, der nicht zu einem sicheren Web-Server (Port 443) oder sicheren News-Server (Port 563) führt. (Beachten Sie das der zweiten Bedingung vorangestellte !, das die Bedingung logisch negiert – die Regel passt also, wenn die Anfrage die erste Bedingung (Methode CONNECT) erfüllt und die zweite Bedingung (Portnummer 443 oder 563) nicht.)

 Die Regel

```
http_access deny CONNECT !SSL_ports
```

impliziert nicht, dass eine CONNECT-Anfrage für die Ports 443 oder 563 zugelassen werden, sondern nur, dass sie nicht abgewiesen wird – die Regel passt nicht und wird deshalb komplett ignoriert. Ob die Anfrage letzten Endes wirklich zugelassen wird, kommt auf die eventuell noch folgenden weiteren Regeln oder das Standardverhalten (siehe gleich) an.

Wenn Squid für eine Anfrage die komplette Regel-Liste einer Aktion abgearbeitet hat, ohne dass eine Regel passt, nimmt es das Gegenteil der letzten Regel als Standardwert an. War die letzte Regel eine deny-Regel, ist der Standardwert allow und umgekehrt.

 Machen Sie es sich zur Angewohnheit, eine Regelkette immer mit einem expliziten »http_access deny all« (oder was auch immer) abzuschließen, um keine unangenehmen Überraschungen zu erleben. Die Bedingung all ist üblicherweise vordefiniert; falls nicht, können Sie das mit

```
acl all src 0.0.0.0/0.0.0.0
```

nachholen.

Sind für eine Aktion überhaupt keine Regeln definiert, wird die betreffende Aktion immer verweigert.

14.3.2 Typen von Bedingungen

Es folgen einige Typen von Bedingungen, d. h. das, was nach einer acl-Direktive direkt nach dem Namen der Bedingung kommt. Die vollständige Liste inklusive Erläuterungen findet sich in der kommentierten Konfigurationsdatei.

`external_acl_type` Bei Bedarf können durch die Direktive `external_acl_type` neue Typen definiert werden, die externe Programme benutzen.

src Die IP-Adresse(n) des anfragenden Clients. Die Argumente sind von der Form »<IP-Adresse>/<Netzmaske>«, wobei Sie die Netzmaske als Bitanzahl (CIDR-Format) oder durch vier Oktette angeben können:

```
acl localhost src 127.0.0.1/255.255.255.255
acl our_networks src 192.168.1.0/24 192.168.2.0/24
```

dst IP-Adresse(n) des Ziel-Servers. Argumente wie bei **src**.

port Port des Ziel-Servers. Sie können auch Bereiche angeben:

```
acl Safe_ports port 443 563      # https, snews
acl Safe_ports port 1025-65535 # unregistered ports
```

proto Das verwendete Protokoll, also das, was im URI vor »://« steht. Beispielsweise wird der Zugriff auf die Cache-Statistiken durch folgende vier Zeilen beschränkt:

```
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
http_access allow manager localhost
http_access deny manager
```

method Verwendete HTTP-Methode (GET, HEAD, CONNECT,...). Beispiel:

```
acl CONNECT method CONNECT
```

url_regex, urlpath_regex Regulärer Ausdruck, der auf den angeforderten URL bzw. auf den URL-Pfad, d. h. den URL ohne Protokoll und Servername, passen muss. Reguläre Ausdrücke werden in `regex(7)` erklärt. Durch die Option `-i` wird Groß- und Kleinschreibung nicht unterschieden, was zumindest für den Servernamen immer empfehlenswert ist. Beispiele:

```
acl QUERY urlpath_regex cgi-bin \?
acl linux url_regex -i www\..*(tux|linux).*\.(org|com|de)$
```

Weitere Bedingungstypen besprechen wir im Zusammenhang mit Squid als Application Level Gateway (Abschnitt 15.3 und 15.4).

14.3.3 Regeln und Aktionen

Wie bereits erwähnt, entscheidet für jede Aktion die erste auf eine Anfrage passende Regel darüber, ob die Aktion ausgeführt wird. Regeln werden durch Direktiven deklariert, die den Aktionen entsprechen. Einige Aktionen/Regel-Direktiven sind:

http_access Regelt den HTTP-Zugriff auf Web-Ressourcen. Dies ist sicherlich die meistgenutzte Aktion.

http_reply_access Schränkt den Zugriff auf Server-Antworten ein; dient dazu, nach Kriterien zu filtern, die erst mit der Antwort feststehen.

cache Hierauf passende Anfragen werden nicht aus dem Cache bedient und die Ergebnisse auch nicht in den Cache übernommen. Durch die Zeilen

```
acl QUERY urlpath_regex cgi-bin \?
cache deny QUERY
```

werden Anfragen vom Cache ausgeschlossen, wenn sie CGI-Skripte aufrufen oder Argumente (durch »?« vom eigentlichen URI getrennt) enthalten. Hier ist nur die Aktion `deny` erlaubt; Squid speichert normalerweise alles.

redactor_access Der URL der Anfrage wird durch die Redactor-Schnittstelle von Squid geleitet. Vgl. Abschnitt 15.5.

cache_peer_access Die Anfrage wird an einen anderen Proxy umgeleitet, wenn dieser Squid sie nicht selbst bedienen kann. Dazu muss Squid in eine Cache-Hierarchie eingebunden sein.

Weitere Aktionen finden sich in der Squid-FAQ und in der kommentierten Konfigurationsdatei.

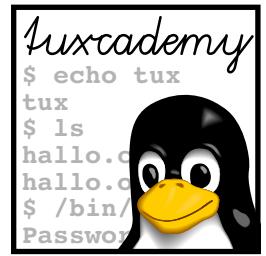
Übungen



14.10 [!2] Beschränken Sie den Squid-Zugriff auf Ihr eigenes Netz.

Zusammenfassung

- Squid wird durch die Datei `/etc/squid/squid.conf` konfiguriert; an ihr sind nur wenige Veränderungen nötig.
- Durch Werkzeuge wie Calamaris können die Zugriffs-Logs von Squid ausgewertet werden.
- Squid kann den Zugriff auf Ressourcen durch Regeln beschränken, die aus einzelnen Bedingungen aufgebaut werden.



15

Squid und Firewalls

Inhalt

15.1	Squid als Application Level Gateway in Firewall-Systemen	194
15.2	Grundkonfiguration	194
15.3	Authentisierung gegenüber dem Proxy	194
15.3.1	Identifizierungs-Server	194
15.3.2	HTTP-basierte Authentisierung	195
15.4	Benutzer-unspezifische Zugriffsbeschränkungen	197
15.5	Filtern über die Redirector-Schnittstelle	198
15.6	Firewall-Perforierung mit HTTP und HTTPS	199

Lernziele

- Squid als Application Level Gateway konfigurieren und einsetzen können
- Methoden zur Proxy-Authentisierung verstehen und anwenden können
- Benutzerunspezifische Zugriffskontrollmechanismen kennen
- Die Redirector-Schnittstelle kennen

Vorkenntnisse

- Kenntnisse über Squid-Konfiguration (Kapitel 14)
- Grundlegendes Verständnis über die Sicherheitsprobleme in Firewall-Konfigurationen

15.1 Squid als Application Level Gateway in Firewall-Systemen

Lag bisher der Schwerpunkt auf den Cache-Fähigkeiten von Squid, so wechselt der Fokus nun auf die Proxy-Möglichkeiten. Ein Application Level Gateway verbindet zwei Netze auf Anwendungsebene, lässt aber keinen direkten IP-Verkehr durch. Anders als Paketfilter, die nur unliebsamen Datenverkehr aussortieren, muss ein Application Level Gateway wie ein Fährmann den Datenverkehr aktiv übersetzen. Für HTTP, HTTPS und FTP bietet sich Squid als dieser Fährmann an.

Durch die Redirector-Schnittstelle können Sperrlisten realisiert werden. Squids Logging-Möglichkeiten erlauben die Kontrolle auf Anwendungsebene an zentraler Stelle.

15.2 Grundkonfiguration

Soll Squid als Application Level Gateway zum Einsatz kommen, so müssen Sie natürlich dafür sorgen, dass beide beteiligten Netze getrennt sind: weder der Rechner, auf dem Squid läuft, noch ein anderer darf Pakete zwischen diesen Netzen vermitteln, da es sonst möglich wäre, den Squid-Proxy zu unterlaufen.

Insbesondere im Rahmen von Firewall-Systemen sollten Sie die Angriffsfläche minimieren, aber auch sonst ist das natürlich eine gute Idee. Mögliche Einstellungen sind dann z. B.:

```
http_port 192.168.0.80:3128
```

lässt Squid nur Anfragen aus dem internen Netz beantworten, und die Einstellungen

```
icp_port 0
htcp_port 0
```

deaktivieren ICP und HTCP, die beide für die Kommunikation zwischen Proxies in einer Cache-Hierarchie benutzt werden. Um den Rechner, der ein Teil des Firewall-Systems ist, nicht unnötig zu beanspruchen, kann es darüberhinaus wünschenswert sein, auf Caching ganz zu verzichten. Dies kann etwa durch

```
acl all src 0.0.0.0/0.0.0.0
no_cache deny all
```

erreicht werden.

15.3 Authentisierung gegenüber dem Proxy

Prinzipiell unterstützt Squid zwei unterschiedliche Methoden der Authentisierung: Anfragen an einen Identifizierungs-Server (`identd`) nach RFC 1413 und Authentisierung des Clients über HTTP.

15.3.1 Identifizierungs-Server

Bei Identifizierung nach [RFC1413]¹ muss auf *jedem* beteiligten Client ein Identifizierungs-Server laufen; unter Unix heißt dieser üblicherweise `identd`. Es gibt auch Implementierungen für Betriebssysteme aus dem Hause Microsoft und andere.

Verbindet sich ein Client mit einem Proxy, bei dem diese Form der Identifizierung aktiviert ist, so stellt der Proxy eine Anfrage an den `identd` (Port 113/tcp) des

¹RFC 1413 ist die überarbeitete Fassung von RFC 931. Spricht letzterer noch von *Authentication Server*, nennt RFC 1413 dies bescheidener *Identification Protocol*.

Clients, wem diese Verbindung gehört.² Der Identifizierungs-Server antwortet mit der Angabe des Betriebssystems und dem Benutzernamen oder einer Fehlermeldung.

In Squid kann diese Anfrage durch die Bedingungs-Typen `ident` und `ident_regex` ausgelöst werden:

ident Werte sind Benutzernamen, wie sie von einer Anfrage an einen Identifizierungs-Server nach RFC 1413 geliefert werden. Der spezielle Benutzername REQUIRED steht für jede erfolgreiche Anfrage.

ident_regex Wie `ident`, nur mit regulären Ausdrücken als Suchmustern. Durch die Option `-i` werden Groß-/Kleinschreibung ignoriert.

So lässt sich beispielsweise der Zugriff für die beiden Benutzer `harry` und `ron` durch

```
acl gryffindor ident harry ron
http_access allow gryffindor
```

freischalten.

 Die Vertrauenswürdigkeit von Antworten eines Identifizierungs-Servers nach RFC 1413 ist äußerst gering:

The Identification Protocol is not intended as an authorization or access control protocol. At best, it provides some additional auditing information with respect to TCP connections. [RFC1413, S. 7]

Zum einen sind die Antworten des Identifizierungs-Servers unsigniert, können also leicht manipuliert werden, zum anderen kann jeder mit lokalen `root`-Rechten jede beliebige Identität annehmen.

Übungen



15.1 [2] Installieren und starten Sie auf dem Client-Rechner (Rechner A) einen `identd` als Identifizierungs-Server. Starten Sie auf dem Server-Rechner (Rechner B) einen Pseudo-Server, wie in Übung 13.2 beschrieben, und verbinden Sie sich von Rechner A aus mit dem Pseudo-Server. Fragen Sie nun auf Rechner B mit `netstat` die Verbindung ab; wichtig ist hierbei der Port, von dem aus von Rechner A die Verbindung zu Rechner B initiiert wurde.
– Befragen Sie den `identd` von Rechner A, indem Sie sich von Rechner B aus mit ihm verbinden (`telnet` an Port 113 auf A). Zum Abfragen müssen Sie nur eine einzige Zeile schicken:

```
33215,3128
```

vorausgesetzt natürlich, Ihr Pseudo-Server lauscht auf Port 3128 und der Ausgangsport des Clients ist 33215.



15.2 [2] Richten Sie Zugriffskontrolle mit Authentisierungs-Servern ein. Lassen Sie nur einen ausgewählten Benutzer auf Squid zugreifen. Testen Sie die Konfiguration.

15.3.2 HTTP-basierte Authentisierung

Die HTTP-basierte Authentisierung gegenüber einem Proxy funktioniert genauso wie die gegenüber einem Web-Server. Wenn ein Browser sich gegenüber einem Server authentisieren kann, dann kann er es auch gegenüber einem Proxy. Dabei ist zu beachten, dass es zwei grundlegende Methoden der Authentisierung gibt: Basic- und Digest-Authentisierung (siehe Kapitel 5)

Squid beherrscht beide Formen der Authentisierung; sie werden über die Bedingungs-Typen `proxy_auth` und `proxy_auth_regex` angesprochen, die analog zu `proxy_auth`

`proxy_auth_regex` ident und `ident_regex` benutzt werden. Damit diese beiden Bedingungs-Typen jedoch verwendet werden dürfen, muss mindestens eine Authentisierungs-Methode aktiviert worden sein. Dies erfolgt durch die Direktive `auth_param`, die Details zu dieser Form der Authentisierung festlegt. Sind mehrere Authentisierungs-Methoden vorhanden, so werden sie den Clients in der Reihenfolge angeboten, in der sie in der Konfigurationsdatei auftreten.

Basic-Authentisierung Um diese Form der Authentisierung durchführen zu können, benötigt Squid ein externes Programm, das auf jede Eingabezeile

`<Benutzer> <Kennwort>`

mit »OK« oder »ERR« als Ausgabe antwortet. Die Squid-Distribution enthält einige solcher Programme, aufgrund der bescheidenen Anforderungen können aber leicht weitere erstellt werden, beispielsweise um eine SQL-Datenbank oder einen LDAP-Server befragen zu können. Wir betrachten hier zunächst nur das bei Squid enthaltene Programm `ncsa_auth`, das als Argument eine Datei benötigt, die Zeilen der Form

`<Benutzer>:<verschlüsseltes Kennwort>`

enthält.

 In der Datenbank von `ncsa_auth` werden weitere, durch : getrennte Einträge ignoriert, so dass theoretisch auch `/etc/shadow` benutzt werden könnte. Davor sollten Sie aber unbedingt die Finger lassen: Web/Proxy-Kennwörter sollten *nie* gültige Login-Kennwörter sein.

Welche Hash-Algorithmen zur Verschlüsselung der Kennwörter verwendet werden können, hängt von den Fähigkeiten der lokalen `crypt(3)`-Implementierung ab. Unter Linux sind dies normalerweise zumindest DES-basierte (`crypt`) und MD5-Hashes.

Ist die Datei `/etc/squid/passwd` entsprechend vorbereitet, können Sie diese Form der Authentisierung in Squids Konfigurationsdatei durch die Zeilen

```
auth_param basic program /usr/sbin/ncsa_auth /etc/squid/passwd
auth_param basic children 5
auth_param basic realm Squid proxy-caching web server
auth_param basic credentialsttl 2 hours
```

aktivieren. Dadurch wird Squid mitgeteilt, welches Hilfsprogramm (mit welchen Argumenten) zum Einsatz kommt, wieviele Instanzen davon gleichzeitig aktiv sind (hier 5), wie der Authentisierungsbereich heißt (engl. *realm*; wird benutzt, um verschiedene Authentisierungen unterscheiden zu können) und wann Squid beim Hilfsprogramm erneut nachfragt (hier nach 2 Stunden). Schließlich müssen Sie noch angeben werden, für wen die Zugriffsbeschränkung gelten soll:

```
acl trusted_users proxy_auth REQUIRED
http_access allow trusted_users
```

 Um DES- und MD5-Hashes zu erzeugen, können Sie das Programm `mkpasswd` benutzen.³ MD5-Hashes erzeugt außerdem das in der GRUB-Shell `grub` aufzurufende Kommando `md5crypt`. Oder Sie legen einen Strohmann-Benutzer an, dem Sie mit `passwd` zu einem Kennwort verhelfen, das Kennwort kopieren und den Benutzer dann wieder löschen.

²Unter Linux können die Besitzer von Netzverbindungen lokal mit dem Kommando »lsof -Pni« ermittelt werden.

³Bei den SUSE-Distributionen oder bei Debian GNU/Linux, Ubuntu & Co. findet es sich im Paket `whois`.

Digest-Authentisierung Auch für diese Methode wird ein externes Programm benötigt, etwa `digest_pw_auth`. Als Eingabe erwartet es Zeilen der Form

```
"<Benutzer>": "<realm>"
```

(die Anführungszeichen müssen mit übergeben werden!) und antwortet jeweils mit einem Hash-Wert, der aus Benutzername, Authentisierungsbereich (*realm*) und Kennwort gebildet wird. Da Squid anders als Apache nur einen Authentisierungsbereich unterstützt, müssen Sie diesen nicht in der Kennwort-Datenbank ablegen. Daher kann `digest_pw_auth` dieselbe Datei wie `ncsa_auth` benutzen. Eingerichtet wird es durch die Zeilen

```
auth_param digest program /usr/sbin/digest_pw_auth /etc/squid/passwd
auth_param digest children 5
auth_param digest realm Squid proxy-caching web server
auth_param digest nonce_garbage_interval 5 minutes
auth_param digest nonce_max_duration 30 minutes
auth_param digest nonce_max_count 50
```

wobei die letzten drei Einträge (und einige weitere) die Lebensdauer von Authentisierungs-Antworten begrenzen, um die Möglichkeit von Nachspiel-Attacken einzuschränken.

 Authentisierung über die Methode *basic* ist unsicher, da hier Klartext-Passwörter ausgetauscht werden. Wenn möglich, sollten Sie daher auf *basic* verzichten: Alle modernen Browser unterstützen inzwischen *digest*. Bei *digest*-Authentisierung wird nie das Kennwort übertragen, allerdings ist auch diese Methode nicht immun gegen Angriffe.

Übungen

 **15.3 [2]** Legen Sie für einige Benutzer eine Squid-Passwort-Datei `/etc/squid/passwd` an und testen Sie sie mit:

```
echo <Benutzer> <Passwort> | ncsa_auth /etc/squid/passwd
```

 **15.4 [1]** Ersetzen Sie die Zugriffskontrolle aus Übung 15.2 durch eine mit *basic*-Authentisierung.

 **15.5 [3]** Schneiden Sie mit `tcpdump(1)` oder `wireshark(1)` die *basic*-Authentisierung gegenüber dem Proxy-Server mit. Finden Sie im Mitschnitt die Kennwortübergabe und dekodieren Sie das (Base64-kodierte) Kennwort. Dazu können Sie `openssl` benutzen:

```
echo <Kodiertes Passwort> | openssl base64 -d
```

15.4 Benutzer-unspezifische Zugriffsbeschränkungen

Sie können verschiedene ACL-Typen verwenden, um bestimmte Gebrauchsmuster des Proxies zu unterbinden oder zu gestatten:

time Bestimmt ein Zeitfenster; das erste Argument bestimmt den oder die Wochentag(e), das zweite die Uhrzeit. Beispielsweise wird durch

```
acl workinghours time MTWTF 8:00-16:30
```

das Zeitfenster »werktags, zur Bürozeit« definiert. Dabei stehen die Großbuchstaben für die englischen Wochentage: `Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday`. Entweder die Angabe der Tage oder die der Uhrzeit kann entfallen.

req_mime_type Das Argument ist ein regulärer Ausdruck, der auf den MIME-Typ der Client-Anfrage passt. Dient zum Filtern von Uploads vom Client auf den Server:

```
acl fileupload req_mime_type -i ^multipart/form-data$
```

rep_mime_type Regulärer Ausdruck, der auf den MIME-Typ der Antwort passt. Beispiel:

```
acl javascript rep_mime_type -i ^application/x-javascript$
```

Bedingungen diese Typs haben keinen Effekt in `http_access`-Regeln, da diese Regeln sich auf die Anfrage beziehen. Die HTTP-Antwort kann erst in `http_reply_access`-Regeln betrachtet werden.

url_regex Kann benutzt werden, um Sperr-Listen zu implementieren. Insbesondere durch die Variante, in der aus einer Datei eingelesen wird, beispielsweise:

```
acl blocklist url_regex "/etc/squid.blocklist"
```

Eine Alternative zu dieser Vorgehensweise wird im Abschnitt 15.5 behandelt.

Gebrauchsspezifische Zugriffsbeschränkungen können natürlich mit benutzerspezifischen kombiniert werden.

15.5 Filtern über die Redirector-Schnittstelle

Deutlich flexibler als mit `url_regex` kann Squid den Zugriff auf einzelne Seiten über die Redirector-Schnittstelle begrenzen. Dazu brauchen Sie ein externes Programm, das auf Eingabezeilen der Form

```
<URL> <Client> <Benutzer> <Methode>
```

mit dem neuen URL antwortet (oder einer Leerzeile, um den ursprünglichen URL gutzuheißen). Dabei sind `<Client>` die IP-Adresse oder der Hostname des anfragenden Clients, `<Benutzer>` der Benutzername, wie ihn der Identifizierungs-Server liefert, und `<Methode>` die HTTP-Methode.

Die Squid-Distribution stellt selbst keinen Redirector bereit, es ist aber eine Vielzahl davon verfügbar, z. B. SquidGuard. Aktiviert wird der Redirector durch die beiden Zeilen

```
redirect_program <Pfad zum Redirector>
redirect_children 5
```

Zum Testen genügt schon ein kleines Skript als Redirector:

```
#!/usr/bin/perl

$|=1;
while (<>) {
    s,http://fromhost.com/,http://tohost.org/,;
    print;
}
```

Durch dieses Skript wird jeder URL auf `http://fromhost.com/` in `http://tohost.org/` umgeschrieben.

Übungen

 **15.6 [4]** Schreiben Sie ein Redirector-Skript, das 5 URLs Ihrer Wahl auf eine andere Seite umleitet. Aktivieren Sie Ihren Redirector in Squid und testen Sie ihn.

Versuchen Sie Ihren Redirector durch Variation der URL auszutricksen.

 **15.7 [4]** Installieren Sie SquidGuard und konfigurieren Sie ihn.

15.6 Firewall-Perforierung mit HTTP und HTTPS

Die Hauptgefahr bei Gateways für HTTP/HTTPS ist, dass HTTP als Vehikel benutzt wird, um (von innen nach außen) jedes Protokoll zu tunneln und damit die Firewall zumindest teilweise absurdum zu führen. Aus diesem Grunde werden in der Squid-Distribution vorab die folgenden Zugriffsbeschränkungen durchgeführt:

```
acl SSL_ports port 443 563
acl Safe_ports port 80          # http
acl Safe_ports port 21          # ftp
acl Safe_ports port 443 563    # https, snews
acl Safe_ports port 70          # gopher
acl Safe_ports port 210         # wais
acl Safe_ports port 1025-65535 # unregistered ports
acl Safe_ports port 280         # http-mgmt
acl Safe_ports port 488         # gss-http
acl Safe_ports port 591         # filemaker
acl Safe_ports port 777         # multiling http
acl CONNECT method CONNECT
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
```

Sie sollen verhindern, dass HTTP-Verbindungen zu beliebigen privilegierten Ports aufgebaut werden können. Beispielsweise ist es so nicht möglich, sich über den Proxy mit Port 25 (SMTP) zu verbinden und damit den Proxy als Mailrelais zu missbrauchen. HTTPS schliesslich (Methode CONNECT) ist nur zu zwei Ports erlaubt.

Je nach Anforderungsprofil sollten Sie die erlaubten Ports noch weiter einschränken.

 Werkzeuge wie Transconnect erlauben es, beliebige TCP-Verbindungen nach »draußen« weiterzuleiten, indem der Proxy über CONNECT dazu veranlaßt wird. Aus diesem Grund lässt Squid in der Standard-Konfiguration nur Verbindungen zu den Ports für sicheres Web und sichere News zu.

Aber selbst wenn Sie CONNECT auf HTTPS beschränken, ist es immer noch möglich, über eine SSL-Verbindung unkontrolliert Daten auszutauschen: Einen entsprechend eingerichteten »Web-Server« vorausgesetzt, kann immer noch aller Verkehr von innen nach außen getunnelt werden. Je nach den erforderlichen Sicherheitsanforderungen müssen Sie gegebenenfalls auf HTTPS ganz verzichten.⁴

⁴Die gleichen Bedenken treffen SSH. Erlauben Sie SSH-Verbindungen, so riskieren Sie, dass sich Benutzer ihr eigenes »privates VPN« aufbauen.

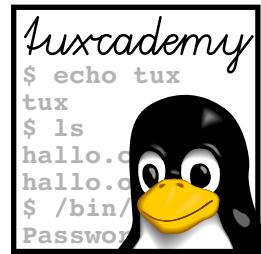
Zusammenfassung

- Squid kann als clientseitiger Application Level Gateway in Firewallkonfigurationen zum Einsatz kommen. Dabei sollte er wahrscheinlich nicht als Cache fungieren.
- Clients können sich gegenüber dem Proxy durch Identifizierungs-Server und über HTTP-Basic- und -Digest-Authentisierung ausweisen.
- Die Redirector-Schnittstelle erlaubt das Einbinden externer Programme zum beliebigen Umschreiben von URLs.
- Unter Umständen kann Squid ausgenutzt werden, um unerwünschten Verkehr durch Firewallsysteme hindurchzuleiten und so ein Sicherheitskonzept zu unterlaufen. Hier ist sorgfältige Konfiguration notwendig.

Literaturverzeichnis

RFC1413 M. St. Johns. »Identification Protocol«, Februar 1993.

<http://www.ietf.org/rfc/rfc1413.txt>



16

Das File Transfer Protocol (FTP)

Inhalt

16.1	Einleitung	202
16.2	FTP-Clients	202
16.2.1	Web-Browser und grafische FTP-Clients	202
16.2.2	Terminalbasierte FTP-Clients	203
16.2.3	Nicht-interaktive FTP-Clients	205
16.2.4	Diagnose-Werkzeuge	205
16.3	Das Protokoll	206
16.3.1	Ursprünge	206
16.3.2	Kontroll- und Daten-Kanal	206
16.3.3	Datenübertragung	207
16.3.4	Authentisierung	208
16.3.5	Weitere FTP-Kommandos	208
16.3.6	Status-Meldungen	209
16.3.7	Alternativen	211

Lernziele

- Grundzüge von FTP verstehen
- Interaktion zwischen FTP-Server und -Client verstehen
- Sicherheitsprobleme von FTP verstehen

Vorkenntnisse

- Verständnis von TCP/IP
- Dienste direkt und über einen Meta-Dämon einrichten können

16.1 Einleitung

Das *File Transfer Protocol (FTP)* ist eines der ältesten Protokolle im Internet. Aber während sein Zeitgenosse TELNET im Internet praktisch ausgestorben ist – lediglich für die Administration von *Embedded Devices* fristet es noch ein Nischendasein im hoffentlich abgesicherten LAN – erfreut sich FTP immer noch reger Verbreitung. Das mag daran liegen, dass sich zwar TELNET 1-zu-1 durch SSH ersetzen lässt, ein solcher Ersatz existiert für FTP jedoch nicht.

Denn obwohl FTP und HTTP (nicht HTML!) in den meisten Fällen wechselseitig in einander überführt werden können, ist diese Transformation in der Praxis mit viel Aufwand verbunden. – Die notwendigen Erweiterungen von FTP bzw. HTTP sind zwar spezifiziert, haben aber (immer) noch nicht die nötige Verbreitung bei Clients wie Servern gefunden. Kurz gesagt: FTP hat Mängel im Bereich Sicherheit und HTTP ist schwach beim *Upload* von Dateien – im Prinzip ist aber beides behebbar.

Die Entscheidung für oder gegen FTP (und gegen oder für HTTP o. Ä.) ist somit weniger eine Frage, ob FTP prinzipiell ein Problem lösen kann, als vielmehr, wieviel Aufwand Sie dazu treiben müssen.

Im Folgenden werden wir als erstes die Client-Seite von FTP beleuchten (Abschnitt 16.2). Danach behandeln wir das Protokoll so weit, wie es für die Konfiguration von FTP-Servern und Firewalls notwendig ist, was uns auch erlauben wird, zumindest kurz FTP gegenüber Alternativen abzugrenzen (Abschnitt 16.3). Schließlich wenden wir uns der Konfiguration eines FTP-Servers zu. Anders als bei HTTP mit Apache gibt es jedoch nicht mehr *den* FTP-Server. Wenn Sie heutzutage einen FTP-Server einrichten wollen, so empfehlen sich Pure-FTPd oder vsFTPD (Kapitel 17), beide sicher und brauchbar. Der populäre WU-FTPD kann das nicht unbedingt behaupten, aber er wird bei der Prüfung für LPIC-2 abgefragt. Die Autoren dieser Schulungsunterlage *raten dringend von der Benutzung von WU-FTPD ab*; er wird hier nur behandelt, um Ihnen die Vorbereitung auf die Prüfung zu ermöglichen. Einen Überblick über die verschiedenen FTP-Server-Programme bietet http://www.linuxmafia.com/faq/Network_Other/ftp-daemons.html.

16.2 FTP-Clients

16.2.1 Web-Browser und grafische FTP-Clients

Da Web- und FTP-Server einen großen Überschneidungsbereich haben – beide sind im Kern Datei-Server –, liegt es nahe, einen FTP-Client in den Web-Browser zu integrieren. Dies ist insbesondere dann angenehm, wenn von einer Web-Site per Hyperlink auf ein Download-Angebot verwiesen wird, das jedoch von einem FTP-Server bereitgestellt wird: Ein Wechsel zu einer anderen Anwendung ist nicht nötig, so dass Sie oft erst nachträglich feststellen, dass Sie sich nicht mehr auf einem Web-Server befinden.

Aber das World Wide Web und FTP entstammen einfach verschiedenen Generationen. Während Web-Browser schon früh grafisch orientiert waren – textbasierte Browser wie lynx oder w3m sind heute Randerscheinungen, und zumindest letzterer bietet Grafik- und Mausunterstützung – sind FTP-Clients primär textbasiert. Das ist weniger eine Frage des verwendeten Protokolls als vielmehr eine Sache der »Gepflogenheiten«, wie ein FTP-Server sich verhält und welche Informationen er preisgibt und wie der Client diese darstellt.

Web-Browser bieten Ihnen daher in der Regel nur eine eingeschränkte Sicht auf den FTP-Server. Für den »Normalbetrieb« wird das ausreichend sein; zum Austesten Ihres FTP-Servers reichen Browser aber nicht. Ähnlich verhält es sich mit grafischen FTP-Clients, wenngleich diese in der Regel besser abschneiden als Web-Browser. Ein positives Beispiel liefert hier gftp (Bild 16.1), der am unteren Fensterrand das Protokoll der Sitzung anzeigt.

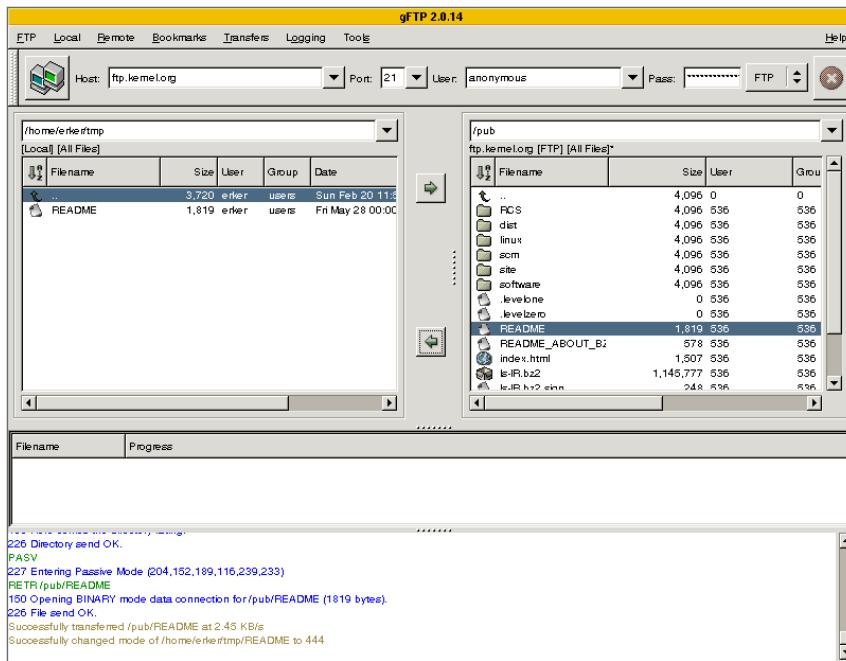


Bild 16.1: Der grafische FTP-Client gftp

16.2.2 Terminalbasierte FTP-Clients

Der FTP-Client für die Kommandozeile heißt traditionell einfach `ftp`, wobei sich dahinter jedoch verschiedene Versionen verstecken können. Empfehlenswert ist hier z. B. `tnftp` von Luke Mewburn, der bei manchen Distributionen noch unter dem alten Namen `lukemftp` vertrieben wird. `tnftp` bietet die von der Bash bekannte Kommandozeilen-History und -Vervollständigung.

Eine beispielhafte FTP-Sitzung zeigt Tabelle 16.1, bei der die Datei `/pub/README` von `ftp.kernel.org` bezogen wird (oder kurz: `ftp://ftp.kernel.org/pub/README`). Dabei wird die Antwort des Servers wie bei SMTP immer mit dem Status-Code eingeleitet und ein »-« nach dem Status-Code besagt, dass die Zeile fortgesetzt wird (bis zu einer Zeile mit demselben Status-Code gefolgt von einem Leerzeichen). Das Ergebnis des `ls`-Kommandos ist nicht direkt zu sehen, da der Server Daten über einen anderen Kanal schickt; mehr dazu in Abschnitt 16.3.

Es folgt eine Aufstellung der wichtigsten Kommandos von `ftp`, die auch auf andere FTP-Clients übertragbar sein sollten.

open, close, user Eine laufende Sitzung kann mit `close` beendet werden, ohne `ftp` zu verlassen; `open` öffnet darauf eine neue Sitzung. Sollte die Anmeldung fehlgeschlagen (falscher Benutzer, falsches Kennwort), so kann sie mit `user` wiederholt werden.

cd, pwd, ls, mkdir, rmdir, chmod, delete, rename Diese Kommandos entsprechen den Unix-Kommandos `cd`, `pwd`, `ls`, `mkdir`, `rmdir`, `chmod`, `rm` bzw. `mv`. Wie weitgehend diese Entsprechung ist, hängt neben dem Client-Programm vom Server-Programm und ggf. auch vom Server-System ab, da manche FTP-Server auf `/bin/ls` usw. zurückgreifen.

get, mget Kopieren eine Datei bzw. mehrere Dateien vom Server.

put, mput Kopieren eine Datei bzw. mehrere Dateien auf den Server.

ascii, binary, system Durch `ascii` schalten Sie die automatische Konvertierung von Zeilenend-Konventionen (Unix: »\n«, DOS: »\r\n«, Mac: »\r«) ein, `binary` unterbindet diese Konvertierung. Die Konvertierung ist nur notwendig

```
$ ftp ftp.kernel.org
Connected to zeus-pub.kernel.org.
220 Welcome to ftp.kernel.org.
Name (ftp.kernel.org:thomas): anonymous
331 Please specify the password.
Password: thomas.erker@linupfront.de
230-                               Welcome to the
230-
230-                               LINUX KERNEL ARCHIVES
230-                               ftp.kernel.org
230-
230-                               "Much more than just kernels"
<*****>
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
227 Entering Passive Mode (204,152,189,116,39,15)
150 Here comes the directory listing.
drwxr-s---  2 536      528          4096 May 21  2001 for_mirrors_only
drwx-----  2 0        0            16384 Mar 18  2003 lost+found
drwxrwsr-x  8 536      536          4096 Dec 26 20:25 pub
226 Directory send OK.
ftp> cd pub
250 Directory successfully changed.
ftp> ls
227 Entering Passive Mode (204,152,189,116,84,77)
150 Here comes the directory listing.
drwxrws---  2 536      536          4096 May 28  2004 RCS
-r-----r--  1 536      536          1819 May 28  2004 README
<*****>
drwxrwsr-x  11 536      536          4096 Jan  1  2002 software
226 Directory send OK.
ftp> get README
local: README remote: README
227 Entering Passive Mode (204,152,189,116,183,41)
150 Opening BINARY mode data connection for README (1819 bytes).
226 File send OK.
1819 bytes received in 00:00 (4.65 KB/s)
ftp> quit
221 Goodbye.
```

Tabelle 16.1: Beispielhafte FTP-Sitzung

bei Textdateien wie z. B. Skripten und nur wenn Client und Server unterschiedlichen Systemtypen angehören. Den Systemtyp des Servers erfahren Sie durch `system`.

passive Schaltet zwischen aktivem und passivem Modus der Datenübertragung um. Server sollten beide Modi unterstützen, es kann aber sein, dass eine Firewall aktive oder passive Datenübertragung unterbindet.

debug, quote Schaltet Debugging an oder aus. Bei eingeschaltetem Debugging gibt der Client auch die Kommandos aus, die er an den Server sendet. Ein vorangestelltes »--->« macht diese Kommandos kenntlich. Mit `quote` können Sie eine Zeichenfolge direkt an den Server senden.

page (tnftp) Anzeige einer Datei mittels `more`, statt sie abzuspeichern.

quit Beendet die Sitzung und `ftp`.

16.2.3 Nicht-interaktive FTP-Clients

Wie auch bei HTTP kann es manchmal nützlich sein, einen Datei-Download oder -Upload skriptgesteuert, und das bedeutet vor allem nicht-interaktiv durchzuführen. Aus der Vielzahl solcher Programme seien hier nur `wget` und `curl` hervorgehoben. Beide verstehen sowohl HTTP als auch FTP, aber während das erste auch als Hilfsprogramm für andere Anwendungen weit verbreitet ist, beherrscht `curl` auch HTTPS und FTPS.

Beide Programme erwarten als Quell-Angabe einen URL; die Ausgabe erfolgt bei `wget` in eine Datei, bei `curl` auf die Standard-Ausgabe:

```
$ curl ftp://ftp.kernel.org/pub/README > README
```

16.2.4 Diagnose-Werkzeuge

Die beste Kontrolle – und damit das stärkste Diagnose-Werkzeug – haben Sie, wenn Sie sich direkt ohne Umwege mit dem Server unterhalten. Allerdings bietet diese Methode, das sei der Fairness halber angemerkt, Null Komfort. Unmittelbar mit dem Server verbinden Sie sich per `netcat` (auch: `nc`):

```
$ netcat 127.0.0.1 21
220 (vsFTPd 2.0.1)
```

Wenn Sie kein `netcat` zur Verfügung haben, dann können Sie zur Not auch `telnet` nehmen, beispielsweise auf Microsoft-Systemen. `netcat` hat aber den Vorteil, dass Sie damit auch in einen lauschenden Modus wechseln können. Mit

```
$ netcat -l -p 40000
```

lauschen Sie auf Port 40000 bis sich jemand mit Ihnen verbindet. Dies wird insbesondere nützlich sein, um FTP-Server mit Datenübertragung im Aktiv-Modus zu testen.

Übungen

 **16.1** [1] Was ist der Unterschied zwischend den URL `http://www.example.org`, `http://ftp.example.org` und `ftp://ftp.example.org`?

 **16.2** [1] Welche FTP-Clients liefert Ihre Distribution mit? Welches Programm(-Paket) verbirgt sich bei Ihrer Distribution hinter `ftp`?

 **16.3** [2] Besorgen Sie sich mit dem Programm `ftp` die Datei `ftp://ftp.kernel.org/pub/README`.



16.4 [!2] Starten Sie `ftp`, ohne einen FTP-Server anzugeben. Schalten Sie mit `debug` Debugging an und laden Sie dann `ftp://ftp.kernel.org/pub/README` herunter. Achten Sie insbesondere auf das, was Ihr Client-Programm an den Server sendet.



16.5 [2] Laden Sie eine Datei von einem FTP-Server sowohl mit `wget` als auch mit `curl` herunter.



16.6 [3] Verbinden Sie sich mit verschiedenen FTP-Servern und untersuchen Sie, wie diese auf verschiedene Optionen von `ls` reagieren. Beispiele:

<code>ftp.kernel.org</code>	(vsFTPd)
<code>ftp.pureftpd.org</code> (213.41.131.17)	(Pure-FTPd)
<code>ftp.rfc-editor.org</code>	(NcFTPd)
<code>ftp.apple.com</code>	(ProFTPD)
<code>ftp.wuftp.org</code>	(WU-FTPd)
<code>ftp.microsoft.com</code>	(Microsoft FTP Service)

Optionen, die Sie ausprobieren sollten, umfassen `-a`, `-F`, `-i` und `--color=yes`.



16.7 [4] Versuchen Sie aufgrund der Charakteristik des `ls`-Kommandos zu erraten, welches FTP-Server-Programm für `ftp.redhat.com` verantwortlich ist.

16.3 Das Protokoll

16.3.1 Ursprünge

FTP ist eines der ältesten noch im allgemeinen Betrieb befindlichen Protokolle. Seine Wurzeln reichen bis in die Anfänge der 1970er Jahre und damit in eine Zeit, als das ARPANET noch nicht TCP benutzte und Bytes nicht unbedingt 8 Bit hatten. Aus diesem Grund ist im Protokoll noch geregelt, wie beispielsweise zwei Rechner kommunizieren können, wenn der eine eine Wortgröße von 32 Bit und der andere eine von 36 Bit hat. Dies macht FTP aufwändig zu implementieren, und viele Server-Entwickler drücken sich davor, den vollständigen Befehlssatz zu implementieren. Das ist in der Regel nicht tragisch, weil diese Befehle heute nicht mehr benötigt werden (Wann haben Sie das letzte Mal Dateien direkt zwischen zwei Rechnern mit EBCDIC-Zeichensatz transferiert?) oder sich aus Sicherheits- oder Performanz-Gründen ohnehin verbieten. Die aktuelle Fassung von FTP beschreibt [RFC0959], mit Korrekturen und Ergänzungen in [RFC1123, RFC1579, RFC2228, RFC2389] u. a.

16.3.2 Kontroll- und Daten-Kanal

Eine Besonderheit von FTP besteht darin, dass die Kommunikation zwischen Client und Server über *zwei* Verbindungen vonstatten geht: einem permanenten Kontroll-Kanal und einem temporären Daten-Kanal. Eine FTP-Sitzung wird dadurch begonnen, dass der Client sich mit dem TCP-Port 21 (`ftp`) des Servers verbindet und damit den **Kontroll-Kanal** etabliert. Auf dem Kontroll-Kanal wird eine stark abgespeckte Form von TELNET gesprochen; deswegen können Sie für den Client auch `telnet` oder `netcat` nehmen. Auf dem Kontroll-Kanal sendet der Client seine Kommandos (im Folgenden immer groß geschrieben, das ist aber nicht notwendig) und der Server antwortet mit einer Status-Meldung (vgl. Abschnitt 16.3.6).

Kontroll-Kanal

Sobald Daten übertragen werden sollen, wird ein **Daten-Kanal** geöffnet. Theoretisch erlaubt FTP die mehrfache Verwendung des Daten-Kanals, aber beim üblicherweise benutzten Transfermodus »stream« wird das Dateiende durch Schließen des Daten-Kanals angezeigt; jede zu übertragende Datei bekommt damit ihren eigenen Daten-Kanal.

Daten-Kanal

stream

Handeln Client und Server nichts anderes aus, so wird der Daten-Kanal *vom Server* initiiert, der sich mit dem gleichen Client-Port des Kontroll-Kanals verbindet, aber der Server-Port ist der Port direkt vor dem des Kontroll-Kanals, also normalerweise `ftp-data = 20/tcp`. Manche Server-Implementierungen erlauben es, einen anderen Ausgangsport zu wählen, verlassen Sie sich also nicht darauf. Man spricht hier von **aktivem FTP**, wenn der Server den Daten-Kanal zum Client aufbaut. aktives FTP

Die Benutzung desselben Ports beim Client ist jedoch ungünstig, da Eigenheiten von TCP die direkte Wiederbenutzung verhindern. Daher kann der Client durch das Kommando `PORT` vorher dem Server einen anderen Port mitteilen. Die Form ist

`PORT <h1>,<h2>,<h3>,<h4>,<p1>,<p2>`

was der IP-Adresse $\langle h1 \rangle.\langle h2 \rangle.\langle h3 \rangle.\langle h4 \rangle$ und dem Port $256 \cdot \langle p1 \rangle + \langle p2 \rangle$ entspricht. Dabei ist die angegebene IP-Adresse normalerweise die des Clients, das ist aber nicht zwingend. Dadurch und wegen der Trennung von Daten- und Kontroll-Kanal können Sie Daten auch direkt zwischen zwei FTP-Servern austauschen, ohne diese über den Client zu kopieren: Verbinden Sie sich über einen *zweiten* Kontroll-Kanal mit einem zweiten FTP-Server, so können Sie ihn durch das Kommando `PASV` veranlassen *passiv* einen Daten-Kanal zu öffnen, d. h. er lauscht auf einem Port, den er Ihnen zusammen mit seiner IP-Adresse im Format wie bei `PORT` zurück gibt. Wenn Sie diese Daten über `PORT` an den ersten Server übermitteln, wird er den nächsten Daten-Kanal zum zweiten Server und nicht zu Ihnen aufbauen. Diese Form der Server-zu-Server-Datenübertragung wird auch als **FXP** bezeichnet; es handelt sich aber nicht um eine besondere Funktionalität oder ein gesondertes Protokoll. FXP

Auch wenn kein zweiter Server beteiligt ist, so kann der FTP-Client durch `PASV` den FTP-Server veranlassen, den Daten-Kanal passiv zu öffnen, und der Client initiiert den Daten-Kanal. Diese Form nennt man auch **passives FTP**. Da diese Form für Firewalls leichter zu bearbeiten ist (Verbindungen werden immer vom Client initiiert), empfiehlt [RFC1579] passives FTP als Standard-Einstellung für Clients und [RFC1123] verlangt, dass alle FTP-Server es unterstützen. Trotzdem hat sich das noch nicht zu allen Clients und Servern herumgesprochen (Beispiel: `wget`). passives FTP



Die Eigenheiten des Daten-Kanals machen FTP zu einem der ungemütlichsten Protokolle, wenn es um NAT und Firewalls geht. Da die Koordinaten des Daten-Kanals erst mitten in der Sitzung ausgehandelt werden, muss eine Firewall oder ein Gateway permanent den Kontroll-Kanal beobachten, um entscheiden zu können, welche Ports in der Firewall durchgelassen werden bzw. wie mit NAT verfahren werden muss – außerdem müssen die Daten im Kontrollkanal umgeschrieben werden. Damit muss die Firewall bzw. das Gateway FTP beherrschen; für andere Protokolle reicht TCP/IP aus.



Die Möglichkeit, den Daten-Kanal auf einen anderen Rechner umzulenken, erlaubt vielfältige Angriffsmöglichkeiten, vgl. [RFC2577]. Erkundigen Sie sich, wie Sie dies bei Ihrem FTP-Server unterbinden oder zumindest einschränken können, damit Ihr Server nicht zum Erfüllungsgehilfen für Cracker wird.

16.3.3 Datenübertragung

Die Bemühungen von FTP, Datentransport zwischen verschiedenen Architekturen zu ermöglichen, spiegeln sich wider in den aus Unix-Sicht exotischen Parametern: Übertragungsmodus, Dateistruktur und Zeichenkodierung.

Der **Übertragungsmodus** wird durch das Kommando `MODE` gesetzt und ist heute fast durchweg *stream* (`MODE S`). Bei diesem Modus wird das Ende der Datei durch das Schließen des Daten-Kanals signalisiert, da dieser Modus keine `E0F`-Kennung kennt. Übertragungsmodus

Dateistuktur	Die Dateistuktur ist <i>file</i> , es sei denn Ihr (Mainframe-)Betriebssystem legt Daten nicht als Dateien, sondern Datensätze (engl. <i>records</i>) ab. Das zugehörige Kommando ist STRU , somit also STRU F .
Zeichenkodierung	Lediglich die Zeichenkodierung (TYPE) ist auch heute noch interessant. FTP kennt neben einem unspezifizierten Bit-Strom (TYPE I für <i>image</i>) noch ASCII (TYPE A oder TYPE AN), EBCDIC und andere. Moderne Clients schalten üblicherweise immer auf Bit-Strom um, da dies die Datei unverändert lässt; nur für direkte Textausgabe (ls usw.) schalten Sie auf ASCII. Die Unterstützung für ASCII hält sich neuerdings in Grenzen, denn der Server muss die ganze Datei durchgehen und die lokale Zeilenend-Konvention in »\r\n« umwandeln. Zusammen mit der Wiederaufnahme eines unterbrochenen Dateitransfers ergibt dies einen schönen DOS-Angriff.

16.3.4 Authentisierung

Um den Dateizugriff auf dem Server zu regeln, müssen Sie sich direkt nach dem Aufbau des Kontroll-Kanals authentisieren. Dies erfolgt durch das Kommando **USER** mit dem Benutzernamen, dem direkt das Kommando **PASS** (plus Kennwort) folgen muss. Eine Sonderrolle spielen dabei die Benutzernamen **anonymous** und **ftp** (manchmal auch noch: **guest**); sie erlauben einen **anonymen Zugang**, d. h. ohne dass Sie ein gültiges Kennwort vorweisen müssen.

Dass Kennwort bei einem anonymen Zugang kann entweder leer sein (»PASS«), auf »guest« lauten (»PASS guest«) oder der Server möchte Ihre E-Mail-Adresse haben. Da der Server aber allenfalls testen kann, ob die angegebene Adresse formalen Kriterien genügt (ist ein »@« enthalten?), müssen Sie hier keine gültige Adresse angeben, wenn Sie das nicht wünschen.

16.3.5 Weitere FTP-Kommandos

FTP kennt eine Vielzahl von Kommandos, so dass wir hier nur kurz die wichtigsten aufführen. Die Kommandos entsprechen in den meisten Fällen direkt den in Abschnitt 16.2.2 behandelten Kommandos, lediglich etwas anders benannt. Tabelle 16.2 gibt eine Gegenüberstellung von FTP- und FTP-Client-Kommandos, zusammen mit der ungefähren Unix-Entsprechung.

Jedes FTP-Kommando hat normalerweise maximal ein Argument, weswegen das Umbenennen einer Datei in zwei Kommandos (RNFR und RNTO) zerlegt wird, die direkt auf einander folgen müssen. Aus diesem Grund gibt es auch kein Äquivalent zu **mget** und **mput**; diese werden auf der Client-Seite in mehrere RETR- bzw. STOR-Aufrufe zerlegt.

Manche Dateioperationen sind nicht unter jedem Betriebssystem möglich, weswegen sie nicht direkt von FTP unterstützt werden. Dazu gehören beispielsweise das Ändern der Dateirechte. Um dennoch die Zugriffsrechte verändern zu können, verfügt FTP mit dem **SITE**-Kommando über die Möglichkeit, **lokale Kommandos** aufzurufen, die der Server bereitstellt, ohne dass sie allgemein spezifiziert wären. Durch **HELP** verrät der Server welche Kommandos er überhaupt unterstützt, und **SITE HELP** liefert ihnen die Liste der lokalen Erweiterungen. Das Ändern der Dateirechte kann dann so aussehen (Ausschnitt; **debug=on**):

```
ftp> ls
<<<<<
150 Here comes the directory listing.
----- 1 test      users      15184 Feb 24 17:30 datei
226 Directory send OK.
ftp> site chmod 644 datei
---> SITE chmod 644 datei
200 SITE CHMOD command ok.
ftp> ls
<<<<<
150 Here comes the directory listing.
```

```
-rw-r--r-- 1 test users 15184 Feb 24 17:30 datei
226 Directory send OK.
```

Die beispielhafte Sitzung aus Tabelle 16.1 stellt sich jetzt so dar, wie es in Tabelle 16.3 zu sehen ist.

16.3.6 Status-Meldungen

Wie bei HTTP oder SMTP antwortet der Server bei FTP mit dreistelligen Statusmeldungen, gefolgt von einer kurzen Erklärung. Die Kodierung der Fehler ist allerdings nicht identisch: existiert eine Datei nicht, so liefert ein HTTP-Server den Status 404, ein FTP-Server jedoch 550. Die Systematik sieht wie folgt aus:

- 1nn Positive ankündigende Antwort, beispielsweise beim Beginn einer Datenübertragung. Weitere Meldungen folgen.
- 2nn Positive endgültige Antwort: Datenübertragung erfolgreich abgeschlossen usw.
- 3nn Positive zwischenzeitliche Antwort auf einen Teil einer Kommando-Sequenz (USER und PASS; RNFR und RNT0; usw.).
- 4nn Negative vorübergehende Antwort; eine spätere Wiederholung ist nicht ausgeschlossen. Beispiel: 452 »kein Plattenplatz vorhanden«.
- 5nn Negative dauerhafte Antwort, wie 503 »Falsche Kommando-Folge«.
- n0n Syntax. Beispiele: 500 »unbekanntes Kommando«, 202 »Kommando (hier) überflüssig«
- n1n Information, wie beispielsweise 215 (Antwort auf SYST).
- n2n Verbindung (Kontroll- und Datenverbindung).
- n3n Authentisierung und Accounting: 331 (»Benutzer OK, aber Kennwort erforderlich«.)
- n4n Reserviert.
- n5n Dateisystem. Beispiel: 257 (»Datei angelegt«).

FTP	Beschreibung	FTP-Client	Unix
CWD <i><Pfad></i>	Change Working Directory	cd	cd
CDUP	Change to Parent Directory	cd ..	cd ..
PWD	Print Working Directory	pwd	pwd
MKD <i><Verzeichnis></i>	Make Directory	mkdir	mkdir
RMD <i><Verzeichnis></i>	Remove Directory	rmdir	rmdir
DELE <i><Datei></i>	Delete	delete	rm
QUIT	Logout	close	logout
RETR <i><Datei></i>	Retrive	get	
STOR <i><Datei></i>	Store	put	
APPE <i><Datei></i>	Append	append	cat >>
ABOR	Abort		 + C
LIST <i><Datei oder Optionen></i>	List	ls	ls -l
NLST <i><Verzeichnis></i>	Name list		ls
RNFR <i><Alter Dateiname></i>	Rename from		
RNT0 <i><Neuer Dateiname></i>	Rename to		
SYST	System	system	uname -o
SITE	Site specific	site	
HELP	Help	rhelp	

Tabelle 16.2: Weitere FTP-Kommandos

```
$ netcat ftp.kernel.org ftp
220 Welcome to ftp.kernel.org.
USER anonymous
331 Please specify the password.
PASS thomas.erker@linupfront.de
230-                               Welcome to the
230-
230-                               LINUX KERNEL ARCHIVES
230-                               ftp.kernel.org
230-
230-                               "Much more than just kernels"
<*****>
230 Login successful.
SYST
215 UNIX Type: L8
TYPE I
200 Switching to Binary mode.
PASV
227 Entering Passive Mode (204,152,189,116,235,157)
LIST
150 Here comes the directory listing.
netcat 204.152.189.116 $((235*256 + 157)) in anderem Terminal
226 Directory send OK.
CWD pub
250 Directory successfully changed.
PASV
227 Entering Passive Mode (204,152,189,116,56,181)
LIST
150 Here comes the directory listing.
netcat 204.152.189.116 $((56*256 + 181)) in anderem Terminal
226 Directory send OK.
PASV
227 Entering Passive Mode (204,152,189,116,83,191)
RETR README
150 Opening BINARY mode data connection for README (1819 bytes).
netcat 204.152.189.116 $((83*256 + 191)) in anderem Terminal
226 File send OK.
QUIT
221 Goodbye.
```

Tabelle 16.3: Beispielhafte FTP-Sitzung (Kontroll-Kanal)

16.3.7 Alternativen

HTTP Für anonymen Download ist ein Webserver sicherlich eine bedenkenswerte Alternative, wenngleich ein moderner FTP-Server leichter zu konfigurieren ist als Apache – aber es gibt ja auch leichtgewichtigere HTTP-Server. Sowohl auf dem Server als auf dem Client macht HTTP weniger Probleme, was die Firewall betrifft.

Für den Datei-Upload ist blankes HTTP allerdings nicht gut geeignet, es ist z. B. eine WebDAV-Erweiterung auf dem Server notwendig (als Apache-Modul) und die Clients müssen WebDAV verstehen. Wegen der fehlenden Breitendurchdringung von WebDAV ist z. Z. HTTP+WebDAV *kein* adäquater Ersatz von FTP, wenngleich die Möglichkeit der einfachen Verschlüsselung mit SSL/TLS es sehr attraktiv erscheinen lässt.

SSH SSH, die *Secure Shell*, liefert sichere Verschlüsselung der Dateienübertragung und ist der offensichtliche, aber leider oft ignorierte Ersatz für TELNET. Durch einen SSH-Tunnel lassen sich auch Dateien kopieren, wofür OpenSSH die Programme `scp` und `sftp` bereitstellt. Dabei hat `sftp` *nichts* mit FTP zu tun, es »fühlt sich nur so an« wie ein FTP-Client.

Der Nachteil von SSH ist, dass damit jeder Benutzer einen vollwertigen Shell-Zugriff auf den Server erhält. Zwar gibt es Möglichkeiten, die Benutzer in einen Chroot-Käfig zu sperren (vgl. `chroot(1)` und `chroot(2)`), dies wird aber z. Z. von OpenSSH nicht direkt unterstützt. Alternativ kann mit dem Paket SCPonly der Zugriff auf den Server auf die Möglichkeiten von `scp` beschränkt werden, aber dieses Paket hat noch keine weite Verbreitung gefunden.

FTPS FTPS, also die Absicherung von FTP durch Verschlüsselung mit SSL/TLS, ist immer noch im Fluss, obgleich diese Technik für HTTP und SMTP längst etabliert ist. Den letzten Stand dokumentieren [RFC2228] und [DRAFT-FTPS05], worin geplant ist, mit dem Kommando AUTH TLS nachträglich Verschlüsselung auszuhandeln. Dies entspricht dem ESMTP-Kommando STARTTLS; der Weg von HTTPS erst einen SSL/TLS-Tunnel (über einen gesonderten Port) aufzubauen, durch den das eigentliche Protokoll übertragen wird, wird nicht mehr verfolgt.

Nur relativ wenige Clients und Server unterstützen FTPS, und nicht alle verschlüsseln Kontroll- *und* Daten-Kanal. Einen Überblick können Sie sich unter <http://www.ford-hutchinson.com/~fh-1-pfh/ftps-ext.html> verschaffen.

TFTP Das *Trivial File Transfer Protocol* (TFTP), wie es in [RFC1350] beschrieben wird, hat gegenüber FTP einen stark reduzierten Funktionsumfang. Insbesondere kennt es keine Benutzer-Authentisierung. TFTP setzt auf UDP statt auf TCP auf; es findet u. a. Verwendung beim Booten von *Diskless Clients*.

Übungen

 **16.8** [1] Klassifizieren Sie die Statusmeldungen 150, 214, 426, 504 und 553 nach den in Abschnitt 16.3.6 angegebenen Kategorien.

 **16.9** [!3] Besorgen Sie sich die Datei `ftp://ftp.kernel.org/pub/README`, wobei Sie dazu nur `netcat` benutzen dürfen. Tabelle 16.3 zeigt, wie es geht.

 **16.10** [4] Wie Übung 16.9, aber benutzen Sie diesmal aktives FTP zur Datenübertragung.

Kommandos in diesem Kapitel

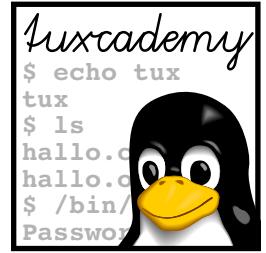
curl	Überträgt Daten von oder an einen Server, mit HTTP, FTP, ...	
		curl(1) 205
ftp	Einfacher FTP-Client für die Kommandozeile	ftp(1) 202, 203
nc	Anderer Name für netcat	nc(1) 205
netcat	Sehr allgemeines Netzwerk-Client-Programm	nc(8) 205
scp	Sicheres Dateikopierprogramm auf SSH-Basis	scp(1) 211
sftp	Sicheres FTP-artiges Programm auf SSH-Basis	sftp(1) 211
telnet	Erlaubt Verbindungen zu beliebigen TCP-Diensten, insbesondere TELNET (Fernzugriff)	telnet(1) 205
wget	Kommandozeilenorientiertes WWW-Clientprogramm	wget(1) 205

Zusammenfassung

- Web-Browser können oft auch als komfortable FTP-Clients benutzt werden. Für Diagnose-Zwecke sind aber Konsolen-Clients (**ftp**) oder gar **netcat** geeigneter.
- FTP ist ein Protokoll aus der Früh- und Vorgeschichte des Internet.
- FTP trennt zwischen Kontroll- und Daten-Kanal.
- Verbindet sich beim Daten-Kanal der Server mit dem lauschenden Client, so spricht man von *aktivem FTP*. Geht die Verbindung vom Client aus, von *passivem FTP*.
- Zeilenend-Konventionen werden im ASCII-Modus automatisch angepasst; ansonsten werden übertragene Daten nicht verändert.
- FTP erfordert immer eine Anmeldung; die Benutzer **anonymous** und **ftp** sind per Konvention für Anmeldung ohne Kennwort reserviert.
- Mit dem **SITE**-Kommando ist FTP lokal erweiterbar.
- Bei Statusmeldungen des FTP-Servers gibt die erste Stelle den Charakter der Meldung an, die zweite Stelle das Gebiet.
- HTTP und SSH stellen nicht in jedem Fall einen vollwertigen Ersatz für FTP dar; FTPS hat nicht die wünschenswerte Verbreitung.

Literaturverzeichnis

- DRAFT-FTPS05** Paul Ford-Hutchinson. »Securing FTP with TLS«, Februar 2005.
<http://www.ietf.org/internet-drafts/draft-murray-auth-ftp-ssl-16.txt>
- RFC0959** J. Postel, J. K. Reynolds. »File Transfer Protocol«, Oktober 1985.
<http://www.ietf.org/rfc/rfc0959.txt>
- RFC1123** »Requirements for Internet Hosts – Application and Support«, Oktober 1989.
<http://www.ietf.org/rfc/rfc1123.txt>
- RFC1350** K. Sollins. »The TFTP Protocol (Revision 2)«, Juli 1992.
<http://www.ietf.org/rfc/rfc1350.txt>
- RFC1579** S. Bellovin. »Firewall-Friendly FTP«, Februar 1994.
<http://www.ietf.org/rfc/rfc1579.txt>
- RFC2228** M. Horowitz, S. Lunt. »FTP Security Extensions«, Oktober 1997.
<http://www.ietf.org/rfc/rfc2228.txt>
- RFC2389** P. Hethmon, R. Elz. »Feature negotiation mechanism for the File Transfer Protocol«, August 1998.
<http://www.ietf.org/rfc/rfc2389.txt>
- RFC2577** M. Allman, S. Ostermann. »FTP Security Considerations«, Mai 1999.
<http://www.ietf.org/rfc/rfc2577.txt>



17

Der FTP-Server vsFTPD

Inhalt

17.1 Empfehlenswert!	214
17.2 Basis-Konfiguration	214
17.3 FTP-Server mit Benutzer-Authentisierung	215
17.4 Virtuelle Server	216
17.5 Virtuelle Benutzer	217

Lernziele

- Den vsFTPD konfigurieren können

Vorkenntnisse

- Grundlagen des FTP-Protokolls (Kapitel 16)
- Konfiguration von Linux-Systemdiensten

17.1 Empfehlenswert!

Können Sie auf die leichte Verwaltung sehr vieler virtueller Benutzer verzichten, beispielsweise weil Sie ohnehin vorhaben, einen Server für anonymes FTP zu betreiben? Spielen virtuelle Server für Sie nur eine untergeordnete Rolle? Dann sollten Sie sich unbedingt vsFTPD näher anschauen.

Der *Very Secure FTP Daemon*, so der offizielle Name, wurde von Anfang an auf Sicherheit hin konzipiert. Auf eine barocke Fülle an Einstellungsmöglichkeiten wurde bewusst verzichtet – durch die Anbindung an PAM und xinetd lassen sich aber viele Szenarien trotzdem gut abbilden.

Durch die Beschränkung auf das Wesentliche ist vsftpd ein stabiler und schneller FTP-Server; viele Sites benutzen ihn, beispielsweise ftp.kernel.org, der FTP-Server für die Verteilung des Linux-Kernels: "May 26, 2004: vsftpd is now serving ftp ... We should have done this sooner ...".

17.2 Basis-Konfiguration

Die mitgelieferte Konfiguration (`/etc/vsftpd.conf`) bzw. bei deren Fehlen die interne Voreinstellung liefert einen sicheren Server für anonymes FTP ohne lokale Benutzer.

Um ihn mit einem Meta-Dämon betreiben zu können, benötigen Sie noch eine entsprechende Konfiguration, die im Falle von xinetd in `/etc/xinetd.d/` abgelegt wird:

```
# Begin /etc/xinetd.d/vsftpd

service ftp
{
    disable          = no
    socket_type     = stream
    wait             = no
    user             = root
    server           = /usr/sbin/vsftpd
    per_source       = 5
    instances        = 200
    banner_fail      = /etc/vsftpd.busy_banner
    log_on_success   += PID HOST DURATION
    log_on_failure   += HOST
}

# End /etc/xinetd.d/vsftpd
```

(Die Datei `/etc/vsftpd.busy_banner` wird von xinetd angezeigt, wenn mehr als 200 Benutzer gleichzeitig auf den Server zugreifen wollen.) Wollen Sie den Server jedoch direkt ohne Meta-Dämon starten, so fügen Sie in `/etc/vsftpd.conf` einfach die Zeilen

```
listen=yes
background=yes
```

ein (keine Leerzeichen vor oder nach dem Gleichheitszeichen!). Durch

```
listen_address=192.0.2.1
```

können Sie ihn außerdem an eine einzelne Adresse binden. Gestartet wird er ohne weitere Optionen.

System-Konto `ftp` Anonym angemeldete Benutzer greifen auf Dateien mit den Rechten eines speziellen System-Benutzers zu, traditionell heißt dieser Benutzer einfach `ftp`. Natürlich können Sie wildfremden Benutzern Ihres FTP-Servers keinen Vollzugriff auf

das System gestatten. Deswegen sperrt vsftpd anonyme Benutzer immer durch einen Chroot-Aufruf (vgl. chroot(2) und chroot(1)) in das Heimatverzeichnis von ftp Chroot-Käfig ein, je nach System /srv/ftp/, /var/lib/ftpd/ o. Ä.

Durch die Direktive `anon_root` können Sie dafür sorgen, dass *nach dem Einsperren* das Verzeichnis gewechselt wird, beispielsweise

```
anon_root=/srv/ftp/pub
```

 Die mitgelieferte Konfigurationsdatei enthält leider nicht alle Optionen. Sie können sich aber eine gut kommentierte Vorlage aus der Handbuchseite `vsftpd.conf(5)` generieren:

```
# mv /etc/vsftpd.conf /etc/vsftpd.conf.bak
# export GROFF_NO_SGR=yes
# man vsftpd.conf | ▷ sed -e $'s/\b//g' -e 's/^## /' >/etc/vsftpd.conf
```

17.3 FTP-Server mit Benutzer-Authentisierung

Von den Voreinstellungen her unterstützt vsFTPD keine lokalen Benutzer, also solche, zu denen ein echtes Benutzer-Konto mit echtem Kennwort gehört. Sie müssen sie daher in `/etc/vsftpd.conf` diese erst freischalten: reale Benutzer

```
local_enable=yes
anonymous_enable=no
```

Bei der Gelegenheit wird anonymer Zugang zudem verweigert. Sollen Ihre Benutzer auch Dateien auf dem Server ablegen können, so müssen Sie dies durch

```
write_enable=yes
chmod_enable=yes
```

freischalten; dabei ist `chmod_enable` (Dateirechte dürfen verändert werden) nicht zwingend erforderlich.

Wollen Sie den Vollzugriff der Benutzer auf Ihr System unterbinden, so können Sie sie in Ihre Heimatverzeichnisse durch Chroot-Käfig

```
chroot_local_user=yes
```

einsperren. Geben Sie zusätzlich

```
passwd_chroot_enable=yes
```

an, so können Sie in `/etc/passwd` durch `./` im Pfad des Heimatverzeichnisses die Stelle markieren, an der der Chroot-Käfig ansetzt. Mit dem Heimatverzeichnis `/home/./tux` hat der Benutzer tux nach oben etwas Luft: ein `cd /` versetzt ihn nach `/home/` und nicht nach `/home/tux/`.

 Aus Sicherheitsgründen sollte das oberste Verzeichnis eines Chroot-Gefängnis nicht beschreibbar sein. Insofern ist eine Trennung von Chroot-Verzeichnis und Heimatverzeichnis durch Einstellungen wie `/home/./tux` sinnvoll, weil der Benutzer tux für `/home/` keine Schreibrechte hat.

 Wenn Sie einen FTP-Server mit Benutzer-Authentisierung betreiben, dann gehen notgedrungen Kennwörter im Klartext über die Leitung. Am besten sichern Sie in einem solchen Fall Kontrollkanal und Datenübertragung durch einen TLS-Tunnel (vsftpd unterstützt FTPS!).

17.4 Virtuelle Server

Virtuelle Server, also das Bedienen von Anfragen an `ftp.foo.example.com` und `ftp.bar.example.net` durch den selben Rechner, werden von vsFTPD nicht direkt unterstützt. Sie können lediglich mehrere Instanzen mit unterschiedlichen Konfigurationen starten:

```
# vsftpd /etc/vsftpd.conf-ftp.foo.example.com
# vsftpd /etc/vsftpd.conf-ftp.bar.example.net
```

Die Konfigurationsdateien können natürlich sehr ähnlich aussehen, Sie sollten aber darauf achten, dass sich die Server wenigstens in den folgenden Direktiven unterscheiden:

ftp_username Diese Direktive gibt den System-Benutzer (normalerweise `ftp`) an, unter dessen Flagge anonym angemeldete Benutzer segeln. Da der öffentlich zugängliche Teil des Servers gerade das Heimatverzeichnis dieses Benutzers ist, benötigen Sie für jeden Server einen eigenen System-Benutzer.

ftp_banner oder banner_file Hier wird die Begrüßungsmeldung hinterlegt.

pam_service_name Wichtig für virtuelle Benutzer; vgl. Abschnitt 17.5.

vsftpd_log_file und xferlog_file Pfade für die Protokolldateien.

Neben diesen »inneren Werten« müssen sich die virtuellen Server natürlich nach außen durch die IP-Adresse unterscheiden, was Sie durch die `listen_address`-Direktive erreichen. Also beispielsweise durch

```
listen_address=192.0.2.1
```

in `/etc/vsftpd.conf-ftp.foo.example.com` und

```
listen_address=192.0.2.2
```

in `/etc/vsftpd.conf-ftp.bar.example.net`.

Um nicht unnötig Prozesse zu starten, empfiehlt es sich hier, die vsFTPD-Instanzen indirekt durch den Meta-Dämon `xinetd` anzusprechen (die `listen_address`-Einträge sind dann überflüssig). Dies ist möglich, weil `xinetd` es erlaubt, seine angebotenen Dienste nicht nur nach Ports, sondern auch nach der lokalen Adresse zu differenzieren: durch

```
# Begin /etc/xinetd.d/vsftpd-ftp.foo.example.com

service ftp
{
    disable                  = no
    socket_type              = stream
    wait                     = no
    user                     = root
    server                   = /usr/sbin/vsftpd
    server_args              = /etc/vsftpd-ftp.foo.example.com
    bind                     = 192.0.2.1
    per_source                = 5
    instances                = 200
    banner_fail              = /etc/vsftpd.busy_banner
    log_on_success            += PID HOST DURATION
    log_on_failure             += HOST
}

# End /etc/xinetd.d/vsftpd-ftp.foo.example.com
```

in der Konfiguration von xinetd lauscht der entsprechende Server nur auf einer Adresse. Der Meta-Dämon inedt ist für diesen Trick allerdings nicht zu gebrauchen, da er nicht die Bindung an einzelne Adressen unterstützt; auch der TCP-Wrapper tcpd hilft hier nicht weiter.

17.5 Virtuelle Benutzer

Die Unterstützung für virtuelle Benutzer ist lediglich eingeschränkt vorhanden: Da vsFTPD die Anbindung an PAM mitbringt, können Sie über PAM auf eine andere Quelle für Benutzer/Kennwörter umschalten. Mit dem Standard-PAM-Paket ist die Benutzung von Datenbank-Dateien (Berkeley-DB) möglich; Anbindungen an SQL-Datenbanken oder LDAP-Server werden durch weitere PAM-Module realisiert, die im Netz erhältlich sind oder Ihrer Distribution beiliegen. Eine beispielhafte PAM-Konfiguration (für Berkeley-DB) könnte so aussehen:

```
# Begin /etc/pam.d/vsftpd

auth    required      pam_userdb.so  db=/etc/vsftpd_login
account required      pam_userdb.so  db=/etc/vsftpd_login

# End /etc/pam.d/vsftpd
```

Wie Sie die Benutzerdatenbank (hier: /etc/vsftpd_login) einrichten, hängt natürlich vom verwendeten PAM-Modul ab. Aus Sicht von vsftpd ist nur interessant, dass Sie PAM benutzen wollen (das ist eine Option beim Übersetzen des Programms) und wie. Für letzteres stehen die Direktiven

```
pam_service_name=vsftpd
session_support=no
```

in /etc/vsftpd.conf zur Verfügung. Mit pam_service_name geben Sie den PAM-Namen an (d. h. den Dateinamen in /etc/pam.d/); durch verschiedene Werte können verschiedene Instanzen ihre jeweils eigene Benutzerverwaltung erhalten. Wollen Sie, dass angemeldete FTP-Benutzer wie lokal oder per SSH angemeldete Benutzer auch mit Programmen wie who sichtbar sind (technisch: /var/log/wtmp und /var/run/utmp werden gepflegt), müssen Sie session_support auf yes setzen. Dies setzt natürlich eine geeignete PAM-Konfiguration voraus (siehe dort).

Da virtuelle Benutzer keinem realen Benutzer auf dem System entsprechen, müssen Sie außerdem durch Benutzer-Transformation

```
guest_enable=yes
guest_username=virtual
```

in /etc/vsftpd.conf alle virtuelle Benutzer auf einen realen Benutzer (hier: virtual) umsetzen.

Übungen

 **17.1** [2] Installieren Sie vsftpd und starten Sie ihn sowohl direkt als auch indirekt über xinetd.

Überprüfen Sie die korrekte Funktion mit einem FTP-Client.

 **17.2** [2] Konfigurieren Sie Ihren vsFTPD so, dass sich Benutzer anmelden und Dateien auf dem Server ablegen können.

 **17.3** [2] Warum benötigen virtuelle FTP-Server zwingend unterschiedliche IP-Adressen? Würden verschiedene DNS-Einträge, die auf die gleiche IP-Adresse verweisen, nicht ausreichen wie bei HTTP-Servern?



17.4 [3] Versehen Sie Ihren Rechner nach Angabe des Trainers mit einer zweiten IP-Adresse. (Alternativ: benutzen Sie im Folgenden die externe IP-Adresse Ihres Rechners und 127.0.0.1.)

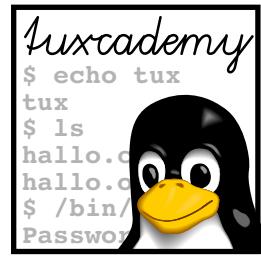
Konfigurieren Sie zwei virtuelle FTP-Server mit unterschiedlichen Begrüßungsmeldungen und Verzeichnissen. Führen Sie diese Übung sowohl mit direkt startenden als auch mit indirekt gestarteten Servern aus.

Kommandos in diesem Kapitel

vsftpd	Einfacher, aber leistungsfähiger FTP-Server	vsftpd(8)	214
xinetd	Verbesserter Internet-Superserver, überwacht Ports und startet ggf. Dienste	xinetd(8)	214

Zusammenfassung

- Der vsFTPd stellt einen ausgesprochen schlanken und sicheren FTP-Server dar.
- Mit der mitgelieferte Konfigurationsdatei ist vsftpd direkt einsatzbereit als anonymer FTP-Server.
- Zugang für reale Benutzer (mit Kennwort) muss bei vsftpd erst freigeschaltet werden.
- Virtuelle Server können mit vsftpd nur über mehrere getrennte Instanzen realisiert werden. Indirektes Starten über xinetd bietet sich hierbei an.
- Über PAM-Module kann vsftpd auch virtuelle Benutzer unterstützen.



18

Pure-FTPd

Inhalt

18.1	Überblick	220
18.2	Konfiguration	223
18.2.1	Anonyme Downloads	223
18.2.2	Anonyme Downloads und Uploads	224
18.2.3	Zugriff für authentisierte Benutzer	226
18.2.4	Virtuelle Benutzer	231
18.2.5	Nachrichten	233
18.2.6	Protokolldateien	235
18.2.7	Virtuelle Server	236
18.3	Sicherheit	237
18.3.1	Verschiedene Sicherheits-Optionen	237
18.3.2	Pure-FTPd und Firewalls	239
18.3.3	FTPS mit Pure-FTPd	239

Lernziele

- Einen Überblick über die Eigenschaften von Pure-FTPd gewinnen
- Pure-FTPd installieren und für gängige Einsatzszenarios konfigurieren können
- Pure-FTPd gegen die wichtigsten Angriffe absichern können

Vorkenntnisse

- Kenntnisse über Linux-System- und Netzadministration
- Grundlagen von FTP (siehe Kapitel 16)

18.1 Überblick

Hintergrund Pure-FTPd (<http://www.pureftpd.org/>) ist ein frei verfügbarer, standardkonformer und effizienter FTP-Server mit vielen nützlichen Eigenschaften. Er geht zurück auf »Troll-FTPd« von Arnt Gulbrandsen, wurde aber in erheblichem Umfang weiterentwickelt.



Während Gulbrandsen in der zweiten Hälfte der 1990er Jahre Troll-FTPd entwickelte, arbeitete er für die norwegische Firma Trolltech, die durch die Entwicklung des Qt-Toolkits bekannt wurde. Daher der Name. (Trolltech wurde 2008 von Nokia aufgekauft; die Qt-Entwicklung ist heute eine Sparte des finnischen Unternehmens Digia, das sie 2012 von Nokia erwarb.)

Installation

Sie können Pure-FTPd entweder im Quellcode von der Webseite des Projekts herunterladen und selber zum Laufen bringen, oder vorgefertigte Pakete verwenden, die Ihre Distribution möglicherweise anbietet.



Die Standardversion von Pure-FTPd findet sich bei Debian GNU/Linux im Paket `pure-ftpd`. Außerdem wird das Paket `pure-ftpd-common` gebraucht, das bei der Installation von `pure-ftpd` über `apt-get` oder `aptitude` automatisch mitinstalliert wird. Ferner gibt es die Pakete `pure-ftpd-ldap`, `pure-ftpd-mysql` und `pure-ftpd-postgresql`, die es ermöglichen, FTP-Benutzer gegen ein LDAP-Verzeichnis oder eine SQL-Datenbank zu authentisieren.



Für Ubuntu gilt im Wesentlichen das, was wir für Debian GNU/Linux gesagt haben. Die Pure-FTPd-Pakete sind Bestandteil der »Universe«-Paketquellen.



Pure-FTPd ist bei den SUSE-Distributionen nicht Bestandteil der Standard-Installationsmedien, kann aber aus alternativen Paketquellen geholt werden. Immerhin kommt YaST mit der Konfiguration zurecht.



Pure-FTPd ist Bestandteil der Fedora-Distribution (Paket `pure-ftpd`). Diese Version unterstützt direkt LDAP, MySQL und PostgreSQL. Eine Version von Pure-FTPd für Red Hat Enterprise Linux (RHEL) finden Sie im erweiterten Paketvorrat von EPEL (siehe <http://fedoraproject.org/wiki/EPEL>).

Start

Im einfachsten Fall (und zum Testen) starten Sie das Programm `pure-ftpd` einfach auf der Kommandozeile:

```
# /usr/sbin/pure-ftpd &
```

Dies erlaubt Systembenutzern über ihren Benutzernamen und ihr Kennwort Zugriff auf das Dateisystem des Rechners. Anonymer FTP-Zugriff ist nur dann erlaubt, wenn es einen Benutzer `ftp` gibt, und in diesem Fall auf dessen Heimatverzeichnis beschränkt. Allerdings dürfen anonyme Benutzer Dateien nicht nur herunter-, sondern auch hochladen.



Bei den gängigen Linux-Distributionen ist Pure-FTPd selbstverständlich passend ins Init-System integriert, so dass Sie Kommandos wie

```
# service pure-ftpd start
```

oder

```
# /etc/init.d/pure-ftpd restart
```

geben können.

Im Gegensatz zu den meisten anderen Programmen seiner Art unterstützt Pure-FTPd standardmäßig keine Konfigurationsdatei, sondern wird ausschließlich über die Kommandozeile konfiguriert. Alle Einstellungen geschehen über Optionen.

Normalerweise lauscht Pure-FTPd zum Beispiel auf dem Port 21 auf allen Netzwerkschnittstellen und IP-Adressen, die der Rechner anbietet (inklusive IPv6). Sie können einen anderen Port definieren oder Pure-FTPd auf eine einzige IP-Adresse festlegen, indem Sie die Option `-S` verwenden:

<pre># /usr/sbin/pure-ftpd -S 2121 # /usr/sbin/pure-ftpd -S 11.12.13.14, # /usr/sbin/pure-ftpd -S 11.12.13.14,2121</pre>	<i>Alternativer Port</i> <i>Eine IP-Adresse</i> <i>Beides</i>
--	---

(Achten Sie im zweiten Fall auf das Komma am Ende der IP-Adresse!) Statt `-S` können Sie übrigens die sprechendere Option `--bind` verwenden:

<pre># /usr/sbin/pure-ftpd --bind 2121</pre>
--

 Diese Methode mag Ihnen auf den ersten Blick vorsintflutlich vorkommen, aber die Logik der Pure-FTPd-Autoren ist unangreifbar: Auf diese Weise muss der Server keinen Code enthalten, der Konfigurationsdateien liest, und Code, den es nicht gibt, ist zu 100% sicher vor Fehlern und Sicherheitslücken und braucht auch keine Ressourcen.

Nichtsdestotrotz gibt es verschiedene Möglichkeiten, Pure-FTPd mit einer Konfigurationsdatei zu verwenden. Die Pure-FTPd-Distribution enthält ein Perl-Programm namens `pure-config.pl`, das eine Konfigurationsdatei liest, die den dort vorgenommenen Einstellungen entsprechenden Kommandooptionen bestimmt und Pure-FTPd mit diesen Optionen aufruft. Wenn in der Datei `/etc/pure-ftpd/pure-ftpd.conf` zum Beispiel die Zeilen

`pure-config.pl`

<pre># Lausche auf der Adresse 11.12.13.14, Port 2121 Bind 11.12.13.14,2121</pre>

stehen, sorgt `pure-config.pl` dafür, dass Pure-FTPd mit der Option `--S 11.12.13.14,2121` aufgerufen wird.

 Wenn Sie eine Abneigung gegen Perl haben: Das gleiche Programm gibt es auch noch in einer Python-Version.

 Es bestehen gute Chancen, dass Ihre Distribution im Init-System eines dieser Programme benutzt, so dass Sie die Konfigurationsdatei verwenden können, anstatt ein Init-Skript o. ä. ändern zu müssen. Die distributions-spezifische Dokumentation Ihres Pure-FTPd-Pakets sollte Ihnen darüber Aufschluss geben – oder Sie schauen einfach im Init-System nach ...

 Debian GNU/Linux verfolgt einen etwas anderen Ansatz für die Konfiguration von Pure-FTPd: Statt einer einzelnen Konfigurationsdatei gibt es hier eine Reihe von Dateien unter `/etc/pure-ftpd/conf`, eine pro Konfigurationsparameter. Um die IP-Adresse und/oder den Port zu ändern, sollten Sie also etwas wie

<pre># echo 11.12.13.14,2121 >/etc/pure-ftpd/conf/Bind</pre>

machen (ein Texteditor tut es natürlich auch). Das Programm, das diese Konfigurationsdaten vorverarbeitet, heißt `pure-ftpd-wrapper` und wird vom Init-System automatisch aufgerufen. Sie finden mehr Informationen hierüber in der Datei `/usr/share/doc/pure-ftpd/README.Debian`.

Statt als freistehenden Hintergrunddienst können Sie Pure-FTPd auch über inetd oder xinetd den inetd oder xinetd starten.



Beim inetd müssen Sie in die Datei /etc/inetd.conf eine Zeile der Form

```
ftp stream tcp nowait root /usr/sbin/pure-ftpd pure-ftpd
```

eintragen. Natürlich können Sie auch den TCP-Wrapper benutzen; dann heißt die Zeile

```
ftp stream tcp nowait root /usr/sbin/tcpd pure-ftpd
```

Anschließend müssen Sie den inetd mit einem SIGHUP über die Änderung benachrichtigen.



Eine mögliche Konfiguration für den xinetd sieht so aus:

```
service ftp
{
    socket_type = stream
    protocol = tcp
    user = root
    server = /usr/sbin/pure-ftpd
    wait = no
    disable = no
}
```

Sie kann entweder in /etc/xinetd.conf stehen oder in einer eigenen Datei, etwa /etc/xinetd.d/pure-ftpd.

Der Betrieb als freistehender Dienst wird allerdings empfohlen.

pure-ftpwho Das Programm pure-ftpwho erlaubt Ihnen, Pure-FTPd bei der Arbeit auf die Finger zu schauen. Es gibt eine Liste der gerade aktiven Verbindungen aus (inklusive Benutzer und Name der Gegenstelle) und gibt für jede Verbindung an, was dort gerade passiert.

Übungen



18.1 [!2] Installieren Sie Pure-FTPd – aus den Paketen für Ihre Distribution oder notfalls vom Quellcode aus.



18.2 [!2] Starten Sie Pure-FTPd mit dem Kommando

```
# /usr/sbin/pure-ftpd &
```

(oder so ähnlich) und überzeugen Sie sich mit einem geeigneten FTP-Client – notfalls einem Web-Browser –, dass Sie mit Ihrem Benutzernamen und Kennwort auf das Dateisystem Ihres Rechners zugreifen können. Ist anonymous Zugriff (mit dem Benutzernamen `ftp`) möglich? (Tipp: Es könnte sein, dass Pure-FTPd nach der Installation automatisch über das Init-System Ihrer Distribution gestartet wird. In diesem Fall sollten Sie den Dienst vor dem Bearbeiten dieser Aufgabe manuell stoppen, etwa mit

```
# service pure-ftpd stop
```

oder einem äquivalenten für Ihr System passenden Kommando. Sie können mit `ps` oder `netstat` herausfinden, ob Pure-FTPd läuft.)



18.3 [2] Konfigurieren Sie Pure-FTPd so, dass er statt als freistehender Hintergrunddienst über den inetd oder xinetd (je nachdem, was auf Ihrem System installiert ist) aufgerufen wird.



18.4 [2] Konfigurieren Sie Pure-FTPd so, dass er nur auf der IP-Adresse 127.0.0.1 auf Verbindungen lauscht. Berücksichtigen Sie dabei, dass Ihre Distribution unter Umständen einen Mechanismus für dateibasierte Konfiguration benutzt. Prüfen Sie die Konfiguration mit netstat und vergewissern Sie sich mit einem FTP-Client, dass der Server zwar über 127.0.0.1, aber nicht über die externe Adresse Ihres Rechners erreichbar ist.

18.2 Konfiguration

18.2.1 Anonyme Downloads

Die einfachste und am wenigsten fehler- und sicherheitslückenanfällige Art, einen FTP-Server zu betreiben, besteht darin, nur anonyme Downloads zuzulassen. Früher war das die Methode der Wahl, um Dokumente und Software auf dem Internet zur Verfügung zu stellen, während Sie dafür heute tendenziell wohl eher einen Web-Server einsetzen würden. Trotzdem kann es immer noch sinnvoll sein, Ressourcen über FTP anzubieten, etwa wenn Sie Benutzern die Möglichkeit geben wollen, komfortable Merkmale von FTP-Clients zu verwenden (zum Beispiel, um mit einem Kommando wie `mget` alle Dateien in einem bestimmten Verzeichnis herunterzuladen, was mit einem Web-Browser nicht so bequem ist).



Natürlich hindert Sie niemand daran, dieselben Ressourcen parallel über HTTP und FTP zur Verfügung zu stellen.

Pure-FTPd unterstützt in seiner Standardeinstellung anonymes FTP, wenn auf dem System ein Benutzer namens `ftp` definiert ist. Wenn Sie als Benutzer beim Anmelden den Benutzernamen `ftp` angeben, werden Sie nicht nach einem Kennwort gefragt, sondern direkt mit einer anonymen Sitzung angemeldet:

Benutzer `ftp`

```
$ ftp 192.168.56.101
Connected to 192.168.56.101.
220----- Welcome to Pure-FTPd [privsep] [TLS] -----
220-You are user number 1 of 50 allowed.
220-Local time is now 20:51. Server port: 21.
220-IPv6 connections are also welcome on this server.
220 You will be disconnected after 15 minutes of inactivity.
Name (192.168.56.101:anselm): ftp
230 Anonymous user logged in
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> _
```



Statt `ftp` können Sie als Benutzername beim Anmelden theoretisch auch `anonymous` verwenden. Aus unerfindlichen Gründen passiert das in der Praxis aber eher selten.



Wenn Sie noch keinen Benutzer `ftp` haben, können Sie ihn bequem wie folgt anlegen:

```
# groupadd ftp
# useradd -r -d /srv/ftp -m -k /dev/null -g ftp ftp
```

Wir legen hier auch gleich eine Gruppe `ftp` an, die uns später noch nützlich sein wird.



Wenn Sie Debian GNU/Linux benutzen, sagen Sie statt dessen

```
# adduser --system --group --home /srv/ftp ftp
```

(Sie könnten auch useradd und groupadd wie im vorigen Absatz verwenden, aber adduser ist Debian-mäßiger.)

/srv/ftp Laut FHS ist /srv/ftp ein gutes Verzeichnis für Inhalte, die per FTP freigegeben werden sollen. Sie können sich natürlich ein beliebiges anderes Verzeichnis aus suchen, solange Sie dafür sorgen, dass es als Heimatverzeichnis des Benutzers ftp eingetragen ist.

Nur anonym Um einen FTP-Server zu erhalten, der *keine* nicht-anonymen Zugriffe erlaubt und anonymen Benutzern auch das Hochladen von Dateien verbietet, müssen Sie beim Start von Pure-FTPd ein paar Kommandooptionen hinzufügen:

```
# /usr/sbin/pure-ftpd -e -i &
```

oder, mit langen Optionsnamen,

```
# /usr/sbin/pure-ftpd --anonymousonly --anonymouscantupload &
```

Die Option -e (--anonymousonly) beschränkt dabei den Zugriff auf anonyme Sitzungen, während -i (--anonymouscantupload) anonyme Benutzer am Hochladen von Dateien hindert. Die korrespondierenden Parameter in einer Konfigurationsdatei heißen AnonymousOnly und AnonymousCantUpload.



Achten Sie auf die doppelte Verneinung: »AnonymousCantUpload no« bedeutet, dass anonyme Benutzer Dateien hochladen *dürfen*.

chroot Anonyme Benutzer sind durch den chroot-Mechanismus im Heimatverzeichnis des Benutzers ftp »eingesperrt«: cd-Kommandos, die sie aus diesem Teilbaum der Verzeichnishierarchie hinausführen würden, werden nicht ausgeführt.

»Punkt-Dateien« Normalerweise dürfen anonyme Benutzer auch nicht auf Dateien oder Verzeichnisse zugreifen, deren Namen mit einem Punkt anfangen (etwa .ssh). Dies ist prinzipiell eine gute Idee, insbesondere wenn Sie ein Verzeichnis sowohl per HTTP als auch per FTP zugänglich machen und ausschließen wollen, dass FTP-Benutzer sich Dateien wie .htaccess oder .htpasswd holen können. Sollten Sie anonymen Benutzern doch den Zugriff auf solche Dateien erlauben wollen (Sie haben sich das hoffentlich gut überlegt), können Sie das mit der Option -z (--allowdotfiles bzw. AllowDotFiles) erreichen.

Übungen



18.5 [!2] Testen Sie den anonymen Zugriff auf Ihren gemäß diesem Abschnitt konfigurierten Pure-FTPd. Was passiert, wenn Sie einen anderen Benutzernamen (außer ftp und anonymous) eingeben?



18.6 [1] Vergewissern Sie sich, dass Sie bei anonymen Zugriffen das Heimatverzeichnis des Benutzers ftp nicht verlassen können.

18.2.2 Anonyme Downloads und Uploads

Mitunter möchten Sie anonymen Benutzern nicht nur das Herunterladen von Dateien erlauben, sondern auch das Hochladen. Stellen Sie sich zum Beispiel vor, Sie betreiben eine Webseite für ein Open-Source-Projekt und wollen die Benutzergemeinde einladen, interessante Beispiele für die Anwendung der Software, Erweiterungen und ähnliches der Allgemeinheit zur Verfügung zu stellen.

Im Prinzip müssen Sie dafür bei Pure-FTPd nichts machen außer die im vorigen Abschnitt erklärte Option -i weglassen. Damit bekommen anonyme Benutzer nicht nur Lese-, sondern auch Schreibzugriff. Im wirklichen Leben wäre das aber wahrscheinlich keine besonders pralle Idee, aus den folgenden Gründen:

- Es besteht ein Unterschied zwischen dem Recht, neues Material von Interesse hochladen zu dürfen, und dem Recht, beliebig im bereits veröffentlichten Material Dateien überschreiben zu dürfen. In der Praxis sorgen Sie also mit Hilfe geeigneter Dateizugriffsrechte dafür, dass neues Material nur in einem bestimmten Verzeichnis (nach Konvention *incoming*) abgelegt werden darf.
- Selbst wenn neues Material nur in *incoming* geschrieben werden darf, ist es normalerweise vernünftig, anderen Benutzern nicht sofort und ohne Weiteres zu erlauben, solches Material wieder herunterzuladen. Dies wäre eine Einladung dazu, Ihren FTP-Server als Lagerort für *waReZ*, also illegale Kopien von Softwarepaketen, Kinofilmen und Musik zu missbrauchen, und das wollen Sie gewiss nicht.

Um den ersten Punkt umzusetzen, legen Sie außer dem Benutzer *ftp* (den Sie brauchen, damit Pure-FTPd überhaupt anonymen Zugriff unterstützt) am besten noch einen weiteren Benutzer, etwa *ftpadmin*, an. Dieser Benutzer fungiert als Eigentümer der zum Herunterladen freigegebenen Ressourcen wie auch als Eigentümer der Verzeichnisse im FTP-Ressourcenbaum – diese sollten aus Sicherheitsgründen nicht dem Benutzer *ftp* gehören.

Verwenden Sie eine Kommandosequenz wie die folgende, um die Verzeichnis-Verzeichnishierarchie für anonymes Hoch- und Herunterladen einzurichten:

```
# groupadd ftpadmin
# useradd -g ftpadmin -d /srv/ftp ftpadmin
# mkdir /srv/ftp/{incoming,pub}
# chown -R ftpadmin:ftpadmin /srv/ftp
# chmod -R 755 /srv/ftp
# chgrp ftp /srv/ftp/incoming
# chmod -R g+w /srv/ftp/incoming
```

Debian-Anwender ersetzen die ersten beiden Kommandos durch

```
# adduser --system --group --home /srv/ftp >
  --shell /bin/bash ftpadmin
```

Dies verschafft Ihnen unterhalb von */srv/ftp* ein Verzeichnis *pub* (für veröffentlichte Inhalte) und ein Verzeichnis *incoming* (für neue von anonymen Benutzern beigebrachte Inhalte). Der wesentliche Unterschied zwischen den beiden ist, dass der Benutzer *ftp* in *incoming* Daten ablegen darf und in *pub* nicht.

Anschließend können Sie Pure-FTPd wie folgt aufrufen:

```
# /usr/sbin/pure-ftpd -e -s &
```

Die Option *-s* (kurz für *--antiwarez*) macht illegalen Software-Tauschern das Leben schwerer, indem sie den Download von Dateien verhindert, die dem Benutzer *ftp* gehören – also neuen von anonymen Benutzern hochgeladenen Dateien. Damit eine Datei »herunterladbar« wird, muss ein Administrator sie mit *chown* einem anderen Benutzer zusprechen, hoffentlich nicht ohne vorher zu prüfen, ob die Datei legal angeboten werden kann. Ansonsten sehen anonyme Benutzer nur etwas wie

```
ftp> get blockbuster.mp4
local: blockbuster.mp4 remote: blockbuster.mp4
200 PORT command successful
550-This file has been uploaded by an anonymous user. It has not
550 yet been approved for downloading by the site administrators.
ftp> _
```

Ein FTP-Administrator würde eine hochgeladene Datei, falls sie akzeptabel ist, aus dem *incoming*- ins *pub*-Verzeichnis (oder ein geeignetes Unterverzeichnis) verschieben und den Eigentümer ändern:

```
# cd /srv/ftp
# mv incoming/foobar-1.0.tar.gz pub/foobar
# chown ftpadmin:ftpadmin pub/foobar/foobar-1.0.tar.gz
```

Dateisystem voll?

 Ebenfalls nützlich ist die Option `-k` (`--maxdiskusagepct`). Sie hat einen Parameter, der angibt, zu wieviel Prozent das Dateisystem maximal belegt sein darf, damit ein Hochladevorgang zugelassen wird. Mit

```
# /usr/sbin/pure-ftpd -e -s -k 90 &
```

zum Beispiel können Dateien nur hochgeladen werden, wenn mindestens 10% der Dateisystemkapazität noch frei sind.



Beachten Sie, dass Pure-FTPd nicht hellsehen kann: Die verfügbare Kapazität des Dateisystems wird am Anfang des Hochladens geprüft, und selbst wenn in diesem Moment noch genug Platz frei zu sein scheint, können Sie das komplette Dateisystem vollschreiben, indem Sie etwas sehr Langes hochladen.

Übungen

 **18.7** [!2] Probieren Sie Pure-FTPds Unterstützung für anonyme Hochladevorgänge aus. Welche Rechte und welche Eigentümer- und Gruppenzuordnung bekommen hochgeladene Dateien?

 **18.8** [2] Testen Sie die *waReZ protection* von Pure-FTPd: Aktivieren Sie die Option `-s` und laden Sie anonym eine Datei hoch. Können Sie die Datei gleich wieder herunterladen?

 **18.9** [2] Prüfen Sie den Belegungstest vor dem Hochladen: Schauen Sie mit `df` nach, zu wieviel Prozent das Dateisystem belegt ist, auf dem Ihr FTP-Verzeichnis liegt. Setzen Sie die Grenze in Pure-FTPd auf einen etwas höheren Prozentwert. Laden Sie eine große Datei hoch (was funktionieren sollte) und dann eine andere große Datei (was nicht mehr funktionieren sollte, wenn die erste Datei das Dateisystem über die Grenze hinweg aufgefüllt hat).

18.2.3 Zugriff für authentisierte Benutzer

Neben anonymen Sitzungen unterstützt Pure-FTPd, wie sich das für einen FTP-Server gehört, auch Sitzungen, bei denen Benutzer sich über einen persönlichen Benutzernamen und das dazugehörige Kennwort authentisieren.



Denken Sie an unsere grundlegenden Warnungen im Zusammenhang mit FTP: Benutzernamen und Kennwörter werden im Klartext übertragen und sind deshalb für jeden sichtbar, der die Möglichkeit hat, den Datenverkehr im Netz zu belauschen. Sie sollten also die gebotene Vorsicht walten lassen, möglicherweise nicht dieselben Kennwörter für FTP und interaktiven Shellzugang verwenden und/oder TLS-Verschlüsselung oder ein VPN einzusetzen, um die Latte zumindest höher zu legen.



Über per TLS verschlüsseltes FTP (FTPS) reden wir im Abschnitt 18.3.3.

PAM

Standardmäßig verwendet Pure-FTPd die PAM-Infrastruktur, um auf Benutzernamen und Kennwörter zuzugreifen. Unter dem Strich wird das, sofern Sie nicht ausdrücklich etwas Anderes sagen, darauf hinauslaufen, dass Benutzerdaten in der Datei `/etc/passwd` nachgeschlagen werden.

 Um »etwas Anderes zu sagen«, müssen Sie die Datei `/etc/pam.d/pure-ftpd` anpassen. Die Details sind (leider) distributionsabhängig.

 Strenggenommen erlaubt Pure-FTPd sowohl den direkten Zugriff auf `/etc/passwd` (vor allem für Systeme, wo die Segnungen von PAM nicht zur Verfügung stehen) als auch Authentisierung via PAM. Ersteres heißt im Pure-FTPd-Jargon auch »Unix-Authentisierung«, letzteres »PAM-Authentisierung«. Sie können auf der Kommandozeile wählen, welche Methode verwendet wird, indem Sie `-l unix` oder `-l pam` angeben. Auf einem Linux-System gibt es allerdings keinen speziellen Grund, `-l unix` zu benutzen; `-l pam` ist auf der ganzen Linie besser.

 Damit Sie sich beim Pure-FTPd anmelden können, muss Ihre Login-Shell in `/etc/shells` aufgeführt sein. Die Unix-Authentisierung prüft das explizit, während die PAM-Authentisierung das PAM-Modul `pam_shells.so` dafür heranzieht. Die Unix-Authentisierung lässt als Ausnahme zu, dass Sie `ftp` als Login-Shell in Ihrem Benutzereintrag haben können, wenn nur FTP-Zugriff erlaubt sein soll und sonst nichts; bei PAM gibt es keine solche Regelung.

 Mit PAM beachtet Pure-FTPd dafür normalerweise die Datei `/etc/ftpusers`, in der diejenigen Benutzernamen aufgezählt sind, die *keinen* FTP-Zugriff bekommen sollen – typischerweise `root` und Konsorten. (Dass die Datei wohl besser `/etc/ftpnonusers` heißen sollte, steht auf einem anderen Blatt.)

Wer sich mit Benutzername und Kennwort angemeldet hat, bekommt anschließend freien Zugriff auf das Dateisystem des Rechners. Im wirklichen Leben ist das oft nicht wünschenswert – stellen Sie sich vor, Sie sind ein Webspace-Anbieter und verwenden FTP, damit Ihre Kunden ihre Inhalte hochladen können. In so einem Fall möchten Sie natürlich nicht, dass Kunde A im Verzeichnis von Kunde B herumschnüffeln kann.

Die Antwort auf diese Herausforderung lautet natürlich wieder mal »chroot«. Mit der Option `-A (--chrooteveryone)` werden nicht nur anonyme Sitzungen, sondern alle Sitzungen im Heimatverzeichnis des betreffenden Benutzers »eingesperrt«. Das heißt, der Benutzer `hugo` kann per FTP-Kommando `cd` sein Heimatverzeichnis (zum Beispiel `/home/hugo`) nicht verlassen:

```
<<<<<
230 OK. Current restricted directory is /
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd ..
250 OK. Current directory is /
```

Auch Kommandos wie »`get ../../etc/passwd`« schlagen fehl. `root` ist hiervon ausgenommen (aber normalerweise anderweitig vom FTP-Zugriff ausgeschlossen).

 Alternativ können Sie die Option `-a <Gruppen-ID>` benutzen. Diese erspart Mitgliedern der Gruppe mit der numerischen ID `<Gruppen-ID>` das Eingesperrtsein:

```
# usermod -a -G ftpadmin hugo
# GID=$(getent group ftpadmin | cut -d: -f3)
# /usr/sbin/pure-ftpd -a $GID &
```

gibt `hugo` freien Zugriff als `ftpadmin`-Mitglied; andere Benutzer bleiben jedoch in ihrem Heimatverzeichnis gefangen:

```
$ ftp 192.168.56.101
Connected to 192.168.56.101.
```

```

220----- Welcome to Pure-FTPd [privsep] [TLS] -----
<<<<<
Name (192.168.56.101:hugo): hugo
331 User hugo OK. Password required
Password: hugo123
230 OK. Current directory is /home/hugo
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> close
ftp> open 192.168.56.101
Connected to 192.168.56.101.
220----- Welcome to Pure-FTPd [privsep] [TLS] -----
<<<<<
Name (192.168.56.101:hugo): susi Nicht restricted
331 User susi OK. Password required
Password: susi123
230 OK. Current restricted directory is / Eingesperrt!
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> _

```

Die Optionen **-a** und **-A** schließen sich gegenseitig aus.

chroot für einzelne Benutzer



Sie können auch ganz ohne die Optionen **-a** oder **-A** einzelne Benutzer in bestimmten Verzeichnissen einsperren, indem Sie im Heimatverzeichnis-Eintrag für den betreffenden Benutzer in der Kennwortdatei die Zeichenfolge **./** hinter das gewünschte Verzeichnis setzen: Ein Eintrag wie

```
susi:x:1001:1001:Susi Sorglos:/home/susi/./:/bin/sh
```

sperrt susi in ihr Heimatverzeichnis ein. Das **./** darf irgendwo im Verzeichnisnamen auftauchen; etwas wie

```
/home/susi./html
```

sorgt dafür, dass susi nach dem Anmelden im Verzeichnis **/home/susi/html** landet, aber keinen Zugriff auf Dateien oder Verzeichnisse außerhalb von **/home/susi** bekommt. (Das ist besonders nützlich, wenn Sie Webspace-Anbieter sind, siehe Übung 18.14.)

Hier sind noch ein paar andere nützliche Pure-FTPd-Optionen, die vor allem für authentisierte Zugriffe von Interesse sind:

Umask Mit der Option **-U** (**--umask**) können Sie die »Umask« für neu angelegte Dateien und Verzeichnisse bestimmen und so deren Zugriffsrechte beeinflussen. Die Umask wird oktal angegeben wie in der Shell, das heißt, jede Oktalziffer steht für einen Benutzerkreis (Eigentümer, Gruppe, Rest der Welt) und jedes Eins-Bit für ein Zugriffsrecht, das der betreffende Benutzerkreis *nicht* haben soll. Die Option übernimmt durch einen Doppelpunkt getrennt eine Umask für Dateien und eine für Verzeichnisse, etwa so:

```
# /usr/sbin/pure-ftpd -U 177:022
```

(um neue Dateien nur für ihren Eigentümer sichtbar zu machen). Standard, wenn nichts Anderes eingestellt wird, ist 133:022.

»Atomares« Überschreiben von Dateien Die Option **-0** (Null! **--notruncate**) sorgt dafür, dass eine Datei, die unter einem schon existierenden Dateinamen hochgeladen wird, die existierende Datei nicht sukzessive überschreibt.

Statt dessen wird die Datei unter einem anderen Namen hochgeladen und dann »atomar« umbenannt (wobei die alte Version entfernt wird). Dies ist sinnvoll im Umgang mit Webseiten – vor allem solchen, die Programmcode wie PHP enthalten –, weil dadurch sichergestellt ist, dass ein unglücklicher Browser-Zugriff mitten während des Hochladevorgangs keine unvollständige Datei zurückliefern kann.

Idiotensicherheit Mit der Option `-Z (--customerproof)` können Sie es als Web-Hoster Ihren Kunden erschweren, sich in den Fuß zu schießen. Aktuell sorgt die Option dafür, dass Benutzer nicht die Möglichkeit bekommen, sich über das FTP-Kommando `CHMOD` alle Zugriffsrechte auf eine Datei oder ein Verzeichnis zu entziehen. In der Zukunft können noch weitere Vorsichtsmaßnahmen dazukommen.

»Punkt-Dateien« Normalerweise können authentisierte Benutzer auf Dateien zugreifen, deren Name mit einem Punkt beginnt (anonyme Benutzer dürfen das aus Sicherheitsgründen standardmäßig nicht). Mit der Option `-x (--prohibitdotfileswrite)` können Sie authentisierten Benutzern das Schreiben von solchen Dateien verbieten; die Option `-X (--prohibitdotfilesread)` verbietet auch das Lesen (und das Betreten von Verzeichnissen, deren Namen mit ».« anfangen).

 Mitglieder der mit `-a` angegebenen "vertrauenswürdigen" Gruppe werden von diesen Einstellungen nicht beeinflusst; sie haben freien Zugriff auf »Punkt-Dateien«.

Benutzer können Dateien hochladen, bis das Dateisystem voll ist. Außerdem können einzelne Benutzer den Plattenplatz monopolisieren und andere Benutzer dadurch behindern. Beides ist nicht wünschenswert und könnte zum Beispiel durch Kontingentierung (*quotas*) vermieden werden. Der Linux-Kernel unterstützt das, aber es ist für die Zwecke eines FTP-Servers doch ziemlich aufwendig zu installieren und zu administrieren. Pure-FTPd unterstützt darum »virtuelle Kontingente«, die im FTP-Server selbst verwaltet werden und keine spezielle Systemkonfiguration voraussetzen. Dazu reicht es, Pure-FTPd mit der Option `-n (--quotas)` aufzurufen. Deren Parameter besteht aus zwei durch einen Doppelpunkt getrennten Zahlen; die erste ist die maximale Anzahl von Dateien, die ein Benutzer haben darf, und die zweite der maximale Platz in Mebibyte, den diese Dateien insgesamt belegen dürfen:

virtuelle Kontingente

```
# /usr/sbin/pure-ftpd -n 100:10 &
```

zum Beispiel beschränkt Benutzer auf maximal 100 Dateien im Gesamtwert von maximal 10 MiB. Das Ganze funktioniert nur, wenn Benutzer per »chroot« in ihre Heimatverzeichnisse eingesperrt werden und »Punkt-Dateien« nicht manipulieren dürfen.

 Pure-FTPd verwendet Dateien namens `.ftpquota` im Heimatverzeichnis jedes Benutzers, um nachzuverfolgen, wieviele Dateien der Benutzer aktuell besitzt und wie groß diese sind.

 Wenn Sie virtuelle Kontingentierung in Kraft setzen, muss die Verzeichnis-hierarchie des FTP-Servers entweder leer sein oder Sie müssen dafür sorgen, dass die `.ftpquota`-Dateien korrekt angelegt werden. Dazu dient das Programm `pure-quotacheck`:

```
# pure-quotacheck -u hugo -d /home/hugo
```

legt die `/home/hugo/.ftpquota` mit dem aktuell gültigen Inhalt an.

 Die Verwaltung von `.ftpquota` durch den Pure-FTPd ist nicht hundertprozentig bombensicher. Es ist darum eine gute Idee, periodisch `pure-quotacheck` für alle Benutzer per Cron auszuführen.

 Die Einstellung mit der Option `-n` gilt für alle Benutzer (außer solchen, die Mitglieder einer mit `-a` angegebenen vertrauenswürdigen Gruppe sind). Sie können aber benutzerspezifische Kontingente vergeben, wenn Sie »virtuelle Benutzer« verwenden. Siehe hierzu Abschnitt 18.2.4.

Übungen

 **18.10** [!2] Konfigurieren Sie Ihren Pure-FTPd so, dass nur authentisierte Zugriffe möglich sind. Anonyme Sitzungen sollen nicht erlaubt sein. Vergewissern Sie sich, dass Ihr Ansatz funktioniert. (*Tipp:* Suchen Sie in der Dokumentation nach der Option `-E`.)

 **18.11** [1] Sorgen Sie dafür, dass der Benutzer `hugo` nicht auf den FTP-Server zugreifen kann.

 **18.12** [*3] Verwenden Sie PAM, um etwas Interessantes mit der Authentisierung von Pure-FTPd zu machen. Zum Beispiel könnten Sie Einmalkennwörter über OTPW (<http://www.cl.cam.ac.uk/~mgk25/otpw.html>) ausprobieren.

 **18.13** [2] Stellen Sie sich vor, Sie sind Anbieter von Webspace und benutzen Pure-FTPd, um Ihren Kunden Zugriff auf ihre Web-Präsenzen zu geben. Sie verwenden eine Verzeichnisstruktur wie die folgende:

<code>/home/hugo</code>	<i>Hugos Heimatverzeichnis</i>
<code>/home/hugo/html</code>	<i>Statische Inhalte für Hugos Webseite</i>
<code>/home/hugo/cgi-bin</code>	<i>Hugos CGI-Skripte</i>
<code>/home/hugo/log</code>	<i>Hugos Protokolldateien</i>
<code>/home/susi</code>	<i>Susis Heimatverzeichnis</i>
<code>/home/susi/html</code>	<i>Statische Inhalte für Susis Webseite</i>
<code>/home/susi/cgi-bin</code>	<i>Susis CGI-Skripte</i>
<code>/home/susi/log</code>	<i>Susis Protokolldateien</i>
~~~~~	

Wie können Sie erreichen, dass jeder Benutzer nur Zugriff auf seine eigenen Dateien hat und die Dateien anderer Benutzer nicht anschauen kann?

 **18.14** [2] (Fortsetzung der vorigen Aufgabe.) Sie möchten es Ihren Benutzern besonders bequem machen und dafür sorgen, dass sie am Anfang einer FTP-Sitzung in ihrem `html`-Verzeichnis landen (damit sie gleich Inhalte hochladen können). Trotzdem sollen sie natürlich einerseits auf ihre eigenen `cgi-bin`- und `log`-Verzeichnisse zugreifen können, aber keine Dateien außerhalb ihres Heimatverzeichnisses zu sehen bekommen. Wie erreichen Sie das?

 **18.15** [2] Überzeugen Sie sich, dass virtuelle Kontingente funktionieren, indem Sie einen neuen Benutzer mit einem leeren Heimatverzeichnis anlegen. Starten Sie Pure-FTPd mit (praktischerweise relativ knappen) Kontingenzen und prüfen Sie, ob der neue Benutzer an die Maxima im Kontingent (Dateianzahl und Gesamtgröße) gebunden ist. Schauen Sie auch mal in die `.ftpquota`-Datei.

 **18.16** [3] Geben Sie ein Shellskript an, das Sie (etwa über das Verzeichnis `/etc/cron.daily`) ausführen können, um für alle Benutzer das Kommando `pure-quotacheck` aufzurufen.

### 18.2.4 Virtuelle Benutzer

Wenn Sie für den FTP-Zugang nicht Ihre normale Benutzerdatenbank nehmen wollen (um andere Kennwörter zu verwenden oder um nicht jeden FTP-Benutzer auch als Systembenutzer eintragen zu müssen, etwa wenn es nur um das Hochladen von Webseiten geht), können Sie Benutzerinformationen auch anderswo ablegen. Die simpelste Möglichkeit besteht darin, die Pure-FTPd-eigene Benutzerdatenbank (oder »puredb«) einzusetzen.

puredb

 Die Pure-FTPd-Benutzerdatenbank erlaubt nicht nur das Speichern von Benutzernamen und Kennwörtern, sondern auch von anderen für den FTP-Server relevanten Informationen wie Kontingentgrößen, Hochlade/Herunterlade-Verhältnissen, Bandbreitenbeschränkungen und so weiter.

 Pure-FTPd kann auch auf Benutzerdaten zugreifen, die in einem LDAP-Verzeichnis oder in einer SQL-Datenbank (MySQL oder PostgreSQL) abgelegt sind. Das macht es einfach, Pure-FTPd in existierende Authentisierungsinfrastrukturen zu integrieren. Aus Platzgründen können wir das hier aber nicht weiter ausführen; ziehen Sie gegebenenfalls die Originaldokumentation zu Rate.

LDAP  
SQL

 Für die ganz harten Fälle erlaubt Pure-FTPd auch den Einsatz eines speziell geschriebenen »Authentisierungsdienstes«, der die nötigen Schritte unternehmen kann, wenn PAM, LDAP, MySQL & Co. nicht ausreichen. Das passt hier aber endgültig nicht mehr rein; suchen Sie in der Originaldokumentation nach »extauth«.

extauth

Um die Pure-FTPd-eigene Benutzerdatenbank zu verwenden, brauchen Sie zumindest einen Systembenutzer, der seine Identität für die »virtuellen« Benutzer zur Verfügung stellt (irgendwem müssen die Verzeichnisse und Dateien ja tatsächlich gehören). Sie könnten mit etwas wie

```
# groupadd ftpgroup
# useradd -g ftpgroup -d /dev/null ftpuser
```

einen Benutzer namens `ftpuser` anlegen, der zur Gruppe `ftpgroup` gehört.

Die Datenbank wird mit dem Kommando `pure-pw` verwaltet (oder Sie editieren sie mit der Hand). Ähnlich wie die Datei `/etc/passwd` besteht eine `puredb`-Datenbankdatei aus durch Doppelpunkte getrennten Feldern; die ersten sechs Felder (bis einschließlich zum Heimatverzeichnis) entsprechen denen von `/etc/passwd`.

pure-pw

 Der Rest der Zeile kann diverse Zusatzinformationen enthalten, die wir hier aus Platzgründen ignorieren; lesen Sie die Datei `README.Virtual-Users` aus der Pure-FTPd-Distribution, um alles ganz genau zu erfahren.

 Wenn Sie nichts Anderes sagen, steht die Pure-FTPd-Benutzerdatei in `/etc/pureftpd.passwd` (genauer gesagt, im Konfigurationsverzeichnis; es könnte auch `/etc/pure-ftpd/pureftpd.passwd` sein). Sie können eine andere Datei verwenden, indem Sie `pure-pw` mit der Option `-f` aufrufen:

```
# pure-pw show susi -f /srv/ftp-users
```

(Dabei müssen Sie natürlich konsequent bleiben.) Sie können den Namen der Datei auch in die Umgebungsvariable `PURE_PASSWDFILE` schreiben.

Pure-FTPd liest nicht die Textdatei, sondern eine Binärversion der Datenbank, die Sie mit dem Kommando

Binärversion

```
# pure-pw mkdb
```

erzeugen. Wenn Sie die Textdatei ändern, müssen Sie die Binärversion aktualisieren, damit die Änderungen wirksam werden.

 Der konventionelle Name dieser Datei ist `/etc/pureftpd.pdb` (aber auch das können Sie nach Belieben ändern). Verwenden Sie gegebenenfalls die Umgebungsvariable `PURE_DBFILE`.

 Die meisten `pure-pw`-Unterkommandos können Sie mit der Option `-m` aufrufen, die automatisch ein `»pure-pw mkdb«` ausführt.

**Neue Benutzer** Einen neuen Benutzer legen Sie mit `»pure-pw useradd«` an:

```
# pure-pw useradd fritz -u ftpuser -d /home/ftpuser/fritz
```

erzeugt einen Benutzer namens `fritz`, der dem Systembenutzer `ftpuser` zugeordnet ist (das heißt, die Dateien, die `fritz` hochlädt, gehören aus der Sicht des Linux-Systems dem Benutzer `ftpuser`). Das Heimatverzeichnis von `fritz` ist `/home/ftpuser/fritz`.

 Wenn Sie das Heimatverzeichnis mit der Option `-d` angeben, wird der Benutzer in das Verzeichnis eingesperrt; Sie können auch `-D` benutzen, dann hat er Zugriff auf das komplette Dateisystem.

 Sie müssen selber dafür sorgen, dass das Heimatverzeichnis des Benutzers existiert, es sei denn, Sie starten Pure-FTPd mit der Option `-j` (–createhome). In diesem Fall wird das Heimatverzeichnis angelegt, wenn der Benutzer sich zum ersten Mal anmeldet.

Denken Sie an `»pure-pw mkdb«` oder die Option `»-m«!`

**Benutzer ändern** Nachträglich ändern können Sie den Eintrag für einen Benutzer mit `»pure-pw usermod«`:

```
# pure-pw usermod fritz -n 100 -N 10
```

beschränkt `fritz` zum Beispiel auf 100 Dateien im Gesamtwert von 10 Mebibytes (sofern Sie `»virtuelle Kontingente«` aktiviert haben). Das Kennwort eines Benutzers ändern Sie mit `»pure-pw passwd«`:

```
# pure-pw passwd fritz
```

**Benutzer löschen** Um einen Benutzer aus der Datenbank zu löschen, verwenden Sie `»pure-pw userdel«`:

```
# pure-pw userdel fritz
```

Die Dateien des Benutzers werden von diesem Kommando nicht entfernt; Sie müssen sie gegebenenfalls manuell löschen.

**Pure-FTPd und die Benutzerdatenbank** Damit Pure-FTPd die Benutzerdatenbank tatsächlich verwendet, müssen ihn mit der Option `-l` starten und angeben, wo die (Binär-)Datei zu finden ist:

```
# /usr/sbin/pure-ftpd -lpuredb:/etc/pureftpd.pdb -j &
```

(erinnern Sie sich: `-j` legt Heimatverzeichnisse automatisch an).

 Wenn Sie alle Systembenutzer zu Pure-FTPd-Benutzern machen wollen (etwa weil Sie für den FTP-Zugang andere Kennwörter verteilen möchten), können Sie mit

```
# pure-pwconvert >/etc/pureftpd.passwd
```

eine erste Näherung für die Pure-FTPd-Kennwortdatei erzeugen. (Wenn Sie das Kommando als root aufrufen, werden die verschlüsselten Kennwörter aus /etc/shadow übernommen; wenn Sie es als unprivilegierter Benutzer tun, sind die Kennwörter leer.)

## Übungen

 **18.17** [2] Legen Sie eine Pure-FTPd-Benutzerdatenbank an und überzeugen Sie sich, dass Pure-FTPd diese benutzt, wenn er mit »-l puredb:...« aufgerufen wird.

 **18.18** [2] Fügen Sie einen Benutzer `limited` hinzu, der nur maximal 5 Dateien haben darf, die gemeinsam nur maximal 1 Mebibyte groß sein dürfen. Vergewissern Sie sich, dass diese Grenzen eingehalten werden, aber andere Benutzer nicht daran gebunden sind.

 **18.19** [W]ie können Sie einen Benutzer erst in der Kennwortdatei des Systems suchen, bevor Sie in der PureFTPd-Benutzerdatenbank nachsehen? (Tipp: Ist es erlaubt, die Option `-l` mehr als einmal anzugeben?)

### 18.2.5 Nachrichten

Mitunter möchten Sie Benutzern Informationen zukommen lassen, wenn sie sich an Ihrem Server angemeldet haben oder bestimmte Verzeichnisse betreten. Hier erklären wir Ihnen, wie das geht:

**Informationen für ein Verzeichnis** Wenn ein Verzeichnis eine Datei namens `.message` enthält, bekommen Benutzer deren Inhalt angezeigt, wenn sie das betreffende Verzeichnis betreten:

```
# cat /srv/ftp/foobar/.message
Dies ist das offizielle Verzeichnis für das foobar-Paket.
Holt Euch die neueste Version hier!
```

liefert zum Beispiel

```
ftp> cd foobar
250-Dies ist das offizielle Verzeichnis für das foobar-Paket.
250-Holt Euch die neueste Version hier!
250 OK. Current directory is /foobar
ftp> _
```

**Banner** Ein »Banner« ist eine Nachricht, die nach dem Anmelden am Server erscheint. Tun Sie die Nachricht in eine Datei `names .banner` im Heimatverzeichnis des betreffenden Benutzers (ftp für anonyme Zugriffe):

```
# echo "Hurra, Hurra, Pure-FTPd ist da ..." >~ftp/.banner
```

Die Nachricht erscheint dann während des Anmeldedialogs:

```
$ ftp 192.168.56.101
Connected to 192.168.56.101.
<=====>
Name (192.168.56.101:anselm): ftp
230-Hurra, Hurra, Pure-FTPd ist da ...
230 Anonymous user logged in
```

```
Remote system type is UNIX.
Using binary mode to transfer files.
```

**Begrüßungsmeldung vor dem Anmelden** Unter Umständen möchten Sie eine bestimmte Meldung ausgeben, noch bevor ein Benutzer sich angemeldet hat. Dies kann zum Beispiel aus rechtlichen Gründen nötig sein, um klarzustellen, dass ein System nur für ausdrücklich befugte Benutzer gedacht ist (um dem Cracker-Argument »Ich wußte ja gar nicht, dass der Rechner nicht für die Allgemeinheit gedacht war« den Wind aus den Segeln zu nehmen).

Pure-FTPd hat hierfür keine direkte Vorkehrung, allerdings können Sie den Mechanismus zweckentfremden, der sonst dazu dient, in der Anmeldeaufforderung launige Sprüche auszugeben: Schreiben Sie Ihre Nachricht zum Beispiel nach `/etc/pure-ftpd/issue` und rufen Sie Pure-FTPd mit der Option `-F (--fortunesfile)` auf:

```
# /usr/sbin/pure-ftpd -F /etc/pure-ftpd/issue
```

(Die Datei muss den Zugriffsmodus 0644 haben, also für alle Benutzer lesbar sein, sonst wird sie ignoriert.) Diese Nachricht ersetzt dann das übliche *Welcome to Pure-FTPd*:

```
$ ftp 192.168.56.101
Connected to 192.168.56.101.
220-Dieses System ist nur für Mitarbeiter von Example Inc.
220 gedacht. Alle anderen bleiben bitte weg!
Name (192.168.56.101:anselm): _
```

 Wenn Sie in der Datei mehrere Nachrichten haben, die durch Zeilen mit nur einem Prozentzeichen voneinander getrennt sind, dann wählt Pure-FTPd zufällig eine der Nachrichten aus:

```
Hallo Seemann!
%
Salaam aleikum!
%
Grüezi miteinand!
```

(Natürlich können Sie auch eine `fortune`-Datei angeben, falls Sie das betreffende Paket mit mehr oder weniger weisen oder witzigen Sprüchen installiert haben.)

## Übungen



**18.20** [!1] Testen Sie die `.message`-Funktionalität von Pure-FTPd.



**18.21** [4] Geben Sie ein Shellskript an, das ausgehend vom Heimatverzeichnis des Benutzers `ftp` alle Unterverzeichnisse betrachtet. In jedem Unterverzeichnis soll in der neuesten (nach Dateidatum) `.tar.gz`-Datei nach einer Datei namens `README` gesucht und deren Inhalt in die Datei `.message` in diesem Verzeichnis kopiert werden.



**18.22** [2] Wie würden Sie dafür sorgen, dass Pure-FTPd seine Benutzer nach dem Anmelden zwischen Mitternacht und 9 Uhr morgens mit »Guten Morgen!« begrüßt, von 9 Uhr morgens bis 14 Uhr mit »Mahlzeit!«, von 14 bis 18 Uhr mit »Guten Tag!« und zwischen 18 Uhr und Mitternacht mit »Guten Abend!«?

### 18.2.6 Protokolldateien

Nachrichten über den Programmablauf schickt Pure-FTPd an den üblichen Systemprotokoldienst (syslogd). Dafür benutzt es die Kategorie `ftp` (naheliegenderweise). Sie können also in Ihrer Konfigurationsdatei (`/etc/syslog.conf` oder `/etc/rsyslog.conf`) etwas schreiben wie

```
ftp.* -/var/log/pure-ftp.log
```

um die Meldungen von Pure-FTPd in eine separate Datei zu schreiben. (Vergessen Sie gegebenenfalls `logrotate` nicht.)

 Wenn Sie eine andere Kategorie benutzen wollen, können Sie diese mit der Option `-f (--syslogfacility)` angeben.

Außerdem kann Pure-FTPd Statistikdaten für Dateizugriffe produzieren. Mit Statistikdaten einem Aufruf wie

```
# /usr/sbin/pure-ftp -0 clf:/var/log/pure-ftp-access.log
```

verwendet Pure-FTPd ein Format, das der Standardeinstellung des Apache-Servers entspricht.

 Der Hauptvorteil dieser Methode ist, dass Sie dieselben Werkzeuge zur Auswertung heranziehen können, die Sie auch für Apache benutzen. Pakete wie *The Webalizer* drängen sich auf.

Alternativ schreibt ein Aufruf wie

```
# /usr/sbin/pure-ftp -0 stats:/var/log/pure-ftp-stats.log
```

eine Datei, die ungefähr so aussieht:

```
1384472751 528560a0.4105 ftp client.example.com>
< D 45691 0 /srv/ftp/foobar/foobar-1.0.tar.gz
```

(Zeilenumbruch aus Platzgründen.) Dabei repräsentiert der erste Eintrag die aktuelle Zeit (in Sekunden seit dem 1.1.1970) und der zweite die aktuelle Sitzung. `ftp` ist der Benutzername (also »anonymer Zugriff«) und `client.example.com` der Name des Rechners, von dem aus der Zugriff erfolgt ist. »`D`« steht für »Download« (Alternative wäre »`U`«), 45691 ist die Dateigröße und 0 die Dauer des Vorgangs in Sekunden. Die Zeile endet mit dem Namen der heruntergeladenen Datei.

 Zu Pure-FTPd gehört ein Programm namens `pure-statsdecode`, das den Zeitstempel in eine lesbare Form bringt.

 Auf einem sehr populären FTP-Server können die DNS-Abfragen für die Clientrechner-Namen lästig werden, weil sie die Bearbeitung von Anfragen verlangsamen. In diesem Fall können Sie mit der Option `-H (--dontresolve)` dafür sorgen, dass nur IP-Adressen protokolliert werden, und die Rückwärtsauflösung gegebenenfalls später »offline« vornehmen. Auf die Rechnernamen im Protokoll sollten Sie sich ohnehin nicht zu sehr verlassen, da Pure-FTPd aus Effizienzgründen keine »Gegenprobe« macht, ob dem betreffenden Namen tatsächlich die aktuelle Client-IP-Adresse zugeordnet ist.

 Pure-FTPd unterstützt außerdem noch die Formate `xferlog` und `w3c`. `xferlog` ist das Statistikformat des alten WU-FTPd und nur von historischem Interesse. `w3c` ist ein Format, das viele Analysewerkzeuge für Webserver (etwa IIS) verstehen.

## Übungen

-  **18.23** [!2] Konfigurieren Sie Ihren Systemprotokolldienst so, dass die Meldungen von Pure-FTPd in der Datei `/var/log/pure-ftpd.log` erscheinen. Überzeugen Sie sich davon, dass das tatsächlich der Fall ist.
-  **18.24** [3] Verwenden Sie `awk`, `Perl` oder eine andere Programmiersprache Ihrer Wahl, um ein Skript zu schreiben, das ausgehend von einer Datei im `stats`-Format bestimmt, wieviel Datenverkehr (Downloads und Uploads) jeder Benutzer verursacht hat, und das Ergebnis (mit den größten Verbrauchern zuerst) ausgibt.

### 18.2.7 Virtuelle Server

Ähnlich wie Apache (oder andere Web-Server) unterstützt auch Pure-FTPd das Konzept sogenannter "virtueller Server", kann also auf demselben Rechner mehrere unabhängige "FTP-Präsenzen" verwalten. Vorbedingung ist allerdings, dass für jede dieser FTP-Präsenzen eine eigene IP-Adresse konfiguriert ist; das Protokoll hat kein Äquivalent zur `Host`-Kopfzeile von HTTP, mit der einer von mehreren "namensbasierten" Servern auf derselben IP-Adresse adressiert werden kann. Pure-FTPd muss also die IP-Adresse selbst als Unterscheidungsmerkmal heranziehen, um zu wissen, welche FTP-Präsenz gemeint ist.

Im einfachsten Fall braucht Pure-FTPd gar keine besondere Konfiguration, um einen virtuellen FTP-Server auf einer bestimmten IP-Adresse einzurichten. Es genügt, im Verzeichnis `/etc/pure-ftpd` ein symbolisches Link unter dem Namen der IP-Adresse anzulegen, das auf das Verzeichnis zeigt, wo die tatsächlichen FTP-Inhalte zu finden sind:

```
# mkdir /srv/ftp-example.com
# ln -s /etc/pure-ftpd/192.168.56.102 /srv/ftp-example.com
```

 Da Debian GNU/Linux `/etc/pure-ftpd` statt `/etc` als Konfigurationsverzeichnis verwendet, ist der Verzeichnisname für die symbolischen Links `/etc/pure-ftpd/pure-ftpd`.

 Wenn Sie sich nicht sicher sind, wo die symbolischen Links stehen sollten, können Sie nachschauen: In der Ausgabe von

```
# strings /usr/sbin/pure-ftpd | grep ^/etc
/etc/pure-ftpd/pure-ftpd/%s
/etc/pure-ftpd/pureftpd-dir-aliases
```

deutet die erste Zeile darauf hin, dass `/etc/pure-ftpd/pure-ftpd` das richtige Verzeichnis ist. (Es geht doch nichts über ein bisschen angewandte Magie.)

Im Gegensatz zu Apache erlaubt Pure-FTPd es nicht, für die einzelnen virtuellen Server unterschiedliche Konfigurationseinstellungen vorzunehmen. Nur die angebotenen Inhalte unterscheiden sich, und diese auch nur für anonyme Zugriffe – authentisierte Zugriffe landen wie üblich im Heimatverzeichnis des betreffenden Benutzers, egal über welche IP-Adresse der Zugriff erfolgt.

»vertrauenswürdige« IP-Adresse

Mit der Option `-V` (–trustedip) können Sie eine »vertrauenswürdige« IP-Adresse angeben. Anschließend sind nur noch auf dieser IP-Adresse authentisierte Zugriffe möglich; alle anderen IP-Adressen erlauben nur noch anonymen Zugang. Angenommen, Sie sind Web-Hoster und haben eine Reihe von Kunden, die jeder eine eigene IP-Adresse haben. Jeder Ihrer Kunden kann so seinen eigenen anonymen FTP-Server einrichten, hat aber trotzdem über die vertrauenswürdige IP-Adresse authentisierten (und schreibenden) Zugriff auf seine Daten.

 Sie sollten darauf achten, dass Ihre Kunden keine allgemein schreibbaren incoming-Verzeichnisse einrichten, damit Ihr Rechner nicht zur waReZ-Schleuder wird. Im Zweifel können Sie anonymen Benutzern mit der Option `-i` (--) anonymouscantupload das Hochladen komplett verbieten. (Die authentisierten Benutzer, also Ihre Kunden, dürfen das natürlich immer noch über die vertrauenswürdige Adresse.)

## Übungen

 **18.25** [3] Aktivieren Sie auf Ihrem Rechner eine weitere IP-Adresse, etwa mit

```
# ifconfig eth0:1 up 192.168.56.102 netmask 255.255.255.0
```

oder

```
# ip addr add dev eth0 192.168.56.102
```

(Netzwerkgerät und IP-Adresse sollten natürlich in etwa zum Rest Ihrer Systemkonfiguration passen.) Legen Sie dann ein FTP-Verzeichnis an (zum Beispiel `/srv/ftp-102`), kopieren Sie ein paar Dateien hinein (nur zur Verdeutlichung) und erzeugen Sie ein symbolisches Link wie oben erklärt:

```
# ln -s /srv/ftp-102 /etc/pure-ftpd/192.168.56.102
```

(achten Sie auf den richtigen Verzeichnisnamen für das symbolische Link). Überzeugen Sie sich, dass Sie bei einem anonymen Zugriff auf die neue Adresse im neuen FTP-Verzeichnis landen.

 **18.26** [2] (Fortsetzung der vorigen Aufgabe.) Stellen Sie sicher, dass auf die neue Adresse nur anonyme Zugriffe möglich sind; authentisierte Zugriffe müssen über die schon vorher vorhandene Adresse gehen.

## 18.3 Sicherheit

### 18.3.1 Verschiedene Sicherheits-Optionen

Auch wenn wir die Wörter »FTP« und »Sicherheit« nicht gerne im selben Satz verwenden, muss man den Pure-FTPD-Entwicklern doch zugute halten, dass sie versuchen, aus einer nicht gerade aussichtsreichen Situation noch das Beste zu machen. Die bekannten Probleme sind im Rahmen des normalen FTP nicht wirklich zu beheben, aber es gibt doch ein paar Möglichkeiten, Benutzer von groben Ungeschicklichkeiten abzuhalten.

**Erlaubte Benutzer** Wir hatten bereits gesehen, wie Pure-FTPD die Datei `/etc/ftpusers` (direkt oder via PAM) verwendet, um bestimmte Benutzer vom FTP-Zugang ausschließt. Linux-Distributionen liefern gerne eine Version von `/etc/ftpusers` mit, in der die gängigen Systembenutzer wie `root` aufgelistet sind. Den gleichen Effekt können Sie aber auch mit der Option `-u` (--) minuid erzielen, mit der Sie die minimale numerische Benutzer-ID angeben können, die auf den FTP-Server Zugriff haben soll. Gemäß der üblichen Politik, dass Benutzer-IDs unter 1000 für Systembenutzer reserviert sind, sorgt ein Aufruf wie

```
# /usr/sbin/pure-ftpd -u 1000
```

dafür, dass nur »echte« Benutzer auf den Server zugreifen können.

**Schindluder** In dem Moment, wo Benutzer über FTP auf den Server schreiben dürfen – vor allem, wenn es um ein gemeinsames incoming-Verzeichnis auf einem anonymen FTP-Server geht –, ist unlauteren Machenschaften Tür und Tor geöffnet:

- Ein Benutzer, der Zugriff auf incoming hat, kann prinzipiell nachschauen, welche Dateien dort liegen, und gleichnamige Dateien hochladen, um die bereits existierenden zu überschreiben. Unterbinden können Sie das mit der Option `-r` (–autorename), die dafür sorgt, dass weitere gleichnamige »Versionen« einer Datei unter anderen Namen abgelegt werden. Wenn zum Beispiel eine Datei `a.txt` bereits existiert, wird die nächste Datei, die unter dem Namen `a.txt` hochgeladen werden soll, tatsächlich unter dem Namen `a.txt.1` abgespeichert und so weiter.
- Die Option `-G` (–norename) verbietet das Umbenennen von Dateien, während `-R` (–nochmod) das Ändern von Zugriffsrechten unterbindet.
- Mit der Option `-K` (–keepallfiles) können Benutzer Dateien zwar hochladen, aber nicht löschen oder umbenennen. Das hindert Benutzer daran, die Dateien aus incoming zu entfernen, die andere Benutzer hochgeladen haben (Dateizugriffsrechte helfen da leider nicht, da alle anonymen Sitzungen sich denselben Systembenutzer teilen).

**Denial of Service** Sie können Obergrenzen für die gleichzeitige Anzahl von Client-Verbindungen (`-c`, `--maxclientsnumber`, Standardwert 50) und die gleichzeitige Anzahl von Client-Verbindungen von derselben IP-Adresse (`-C`, `--maxclientsperip`) setzen. Dies macht es einzelnen Benutzern schwerer, den Server komplett zu monopolisieren. Dies ist eine sehr sinnvolle Eigenschaft.

Um Probleme mit übermäßigem CPU-Verbrauch oder plumpen Angriffen zu entschärfen, liefert Pure-FTPd in der Ausgabe eines `ls`-Kommandos niemals mehr als 10.000 Dateien zurück. Ebenso betrachtet ein »`ls -R`« Unterverzeichnisse höchstens fünf Stufen tief. Sie können diese Grenzen mit der Option `-L` (–limitrecursion) anpassen.

Mit der Option `-m` (–maxload) sind Downloads nur zulässig, wenn die Systemlast (*load*) höchstens dem Parameter der `-m` entspricht. Mit

```
# /usr/sbin/pure-ftpd -m 5
```

zum Beispiel werden bei einer Systemlast von 5 oder mehr nur noch Uploads verarbeitet.



Die Systemlast ist nur ein sehr grobes Maß für die Auslastung eines Rechners. Beachten Sie insbesondere, dass bei einem Rechner mit  $n$  Prozessorkernen eine Systemlast von  $n$  oder weniger in der Regel völlig unbedenklich ist, auch wenn die absolute Zahl möglicherweise unangenehm groß aussieht.

## Übungen



**18.27** [2] Vergewissern Sie sich, dass die Option `-u` wie angegeben funktioniert, indem Sie z. B. 1000 als Untergrenze für gültige Benutzer-IDs festlegen und als Benutzer mit einer niedrigeren Benutzer-ID auf den Server zuzugreifen versuchen.



**18.28** [2] Probieren Sie aus, ob die Option `-c` funktioniert, indem Sie sie auf einen niedrigen Wert setzen und mehrere Verbindungen zum FTP-Server aufzubauen.

### 18.3.2 Pure-FTPd und Firewalls

FTP und Firewalls sind zwei Themen, die nicht wirklich gut zusammenpassen. Das traditionelle »aktive FTP«, bei dem der Server zur Datenübertragung eine Verbindung zum Client aufbaut, wird nur von solchen Firewalls unterstützt, die in der Lage sind, das Protokoll mitzulesen, und so seelisch auf die entsprechende Anfrage von (aus der Sicht des Clients) außen vorbereitet sind. Bei »passivem FTP« baut der Client die Datenverbindung zum Server auf, was aus der Sicht eines clientseitigen Firewalls deutlich unproblematischer ist – aber dadurch verlagert sich das Problem nur auf die Serverseite.

Auf dem Server müssen Sie auf jeden Fall den Port 21 für Clients zugänglich machen, da sonst kein Client eine Steuerverbindung zum Server aufbauen kann. Für aktives FTP muss der Port 20 für ausgehende Verbindungen freigeschaltet werden.

Um eine passende Firewall-Konfiguration für passives FTP vornehmen zu können, muss der Server kontrollieren können, welche Ports er dem Client für Datenverbindungen vorschlägt. Am besten wählen Sie dafür einen Portbereich oberhalb von 1023, der mindestens doppelt so viele Ports enthält, wie Sie gleichzeitige Client-Verbindungen zu unterstützen gedenken. Wenn Sie mit 100 parallelen Sitzungen rechnen, dann könnten Sie zum Beispiel den Portbereich von 60000 bis 60200 ins Auge fassen. Diesen Bereich übergeben Sie dann mit der Option `-p` (--)passiveportrange) an Pure-FTPd:

```
# /usr/sbin/pure-ftpd -p 60000:60200 -c 100
```

(die Option `-c` (--)maxclientsnumber) begrenzt die Anzahl der gleichzeitigen Verbindungen auf 100).

Natürlich müssen Sie die entsprechenden Ports auch an Ihrer Firewall für ankommende Verbindungen freischalten. Grundsätzlich gilt die (etwas konterintuitive) Regel, dass es für passives FTP sicherer ist, mehr Ports freizuschalten als weniger, weil die Wahrscheinlichkeit so geringer ist, dass jemand über *FTP data hijacking* unbefugt Daten abgreift.



*FTP data hijacking* ist ein Angriff gegen passive FTP-Server, bei dem ein Angreifer eine Portnummer rät und eine Verbindung aufbaut, in der Hoffnung, dort einem Client zuvorzukommen, der gerade Daten abrufen möchte. (Das klingt zunächst plump, aber mit etwas Heuristik über benutzte Portnummern und die Rate, mit der Datenverbindungen aufgebaut werden, ist es nicht völlig aussichtslos.) Gute FTP-Server prüfen darum die IP-Adresse einer ankommenden Datenverbindung, um herauszufinden, ob sie die des Clients ist, der die Datenverbindung angefordert hat.

## Übungen



**18.29** [3] Verwenden Sie `iptables`, um auf Ihrem Rechner alle TCP-Ports von 1024 bis 59999 und 60201 bis 65535 für eingehende Verbindungen zu sperren. (Sie können das stichprobenartig oder mit `nmap` testen.) Starten Sie dann Pure-FTPd mit dem Portbereich `60000:60200`. Testen Sie passive FTP-Verbindungen und beobachten Sie die verwendeten Portnummern (im Client oder mit z. B. Wireshark).

### 18.3.3 FTPS mit Pure-FTPd

Neben »regulärem« FTP unterstützt Pure-FTPd auch FTP über TLS, kurz FTPS. Um FTPS zu verwenden, brauchen Sie ein X.509-Zertifikat für Ihren Server und den dazu passenden privaten Schlüssel.



Wir können an dieser Stelle nicht im Detail beschreiben, wie Sie ein X.509-Zertifikat erzeugen.

 Wie immer ist es wichtig, dass der *Common Name* (CN) im Zertifikat genau dem DNS-Namen des Servers entspricht, unter dem er später angesprochen wird.

Damit Pure-FTPd den privaten Schlüssel und das Zertifikat finden kann, müssen die beiden in einer Datei namens `/etc/ssl/private/pure-ftpd.pem` stehen. Sie können den Schlüssel und das Zertifikat einfach hintereinanderhängen und in die Datei schreiben.

 Wenn Sie den Dateinamen ändern wollen, müssen Sie Pure-FTPd neu übersetzen.

Anschließend können Sie Pure-FTPd mit der Option `-Y (--tls)` starten. `-Y` übernimmt einen numerischen Parameter mit der folgenden Bedeutung:

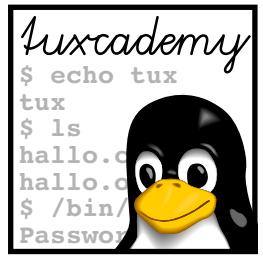
- »-Y 0« schaltet die TLS-Unterstützung komplett aus. Das ist die Voreinstellung, aber höchstwahrscheinlich genau das, was Sie nicht haben wollen, wenn Sie diesen Abschnitt lesen.
- Mit »-Y 1« sind sowohl traditionelle unverschlüsselte wie auch verschlüsselte Verbindungsanfragen möglich. Für viele Fälle ist das vermutlich der geschickteste Weg, wenn Sie es sich nicht mit Ihren Benutzern verscherzen wollen.
- »-Y 2« besteht darauf, dass Steuerverbindungen verschlüsselt sind. Clients, die eine Klartextverbindung aufnehmen wollen, werden zurückgewiesen.
- »-Y 3« schließlich erzwingt Verschlüsselung nicht nur für die Steuerverbindung, sondern auch für die Datenverbindungen. Vom Standpunkt der Datensicherheit her ist dies die optimale Einstellung.

## Übungen

 **18.30** [!3] Besorgen Sie sich ein X.509-Zertifikat für Ihren Server (mit dem dazugehörigen privaten Schlüsseln) und überzeugen Sie sich, dass FTPS mit Pure-FTPd funktioniert.

## Zusammenfassung

- Pure-FTPd ist ein frei verfügbarer, standardkonformer und effizienter FTP-Server.
- Pure-FTPd wird rein über die Kommandozeile konfiguriert (aber es gibt Erweiterungen, die (eine) Konfigurationsdatei(en) ermöglichen).
- Pure-FTPd kann als freistehender Hintergrunddienst oder über `inetd/xinetd` gestartet werden.
- Pure-FTPd lässt sich bequem für verschiedene Einsatzszenarios mit anonymen oder authentisierten Benutzern konfigurieren.
- Benutzerdaten bezieht Pure-FTPd nicht nur vom Betriebssystem, sondern auch aus seiner eigenen Benutzerdatenbank oder anderen Quellen wie LDAP oder SQL-Datenbanken.
- Pure-FTPd unterstützt den Systemprotokolldienst und kann in verschiedenen Formaten Statistikdaten über Uploads und Downloads schreiben.
- Auf Rechnern mit mehreren IP-Adressen kann Pure-FTPd mehrere "virtuelle FTP-Präsenzen" verwalten.
- Pure-FTPd unterstützt diverse Optionen zur Erhöhung der Sicherheit, etwa zur Vermeidung schädlicher Dateioperationen auf dem Server und zur Abmilderung von Denial-of-Service-Angriffen.
- Für passives FTP unterstützt Pure-FTPd die Spezifikation eines Portbereichs für Datenverbindungen
- Pure-FTPd unterstützt FTPS.



# A

## Musterlösungen

Dieser Anhang enthält Musterlösungen für ausgewählte Aufgaben.

**1.1** Die gängigsten URI-Schemas sind zweifellos `http`, `ftp` und `mailto`. Web-Browser enthalten oft mehr oder weniger ausgebauten Unterstützung für `gopher` (einen weiteren Vorläufer des WWW), `news` und `nntp` oder `telnet`. Manche Browser erlauben die Definition »eigener« URI-Schemas, die dann über Textersetzung in »echte« URLs umgesetzt werden. Sie könnten sich zum Beispiel vorstellen, etwas wie `rfc:2396` als `http://www.ietf.org/rfc/rfc2396.txt` zu interpretieren.

**1.2** Eine Möglichkeit wäre etwas wie `pots://49/6151/9067-0`, d. h., das URL-Schema heißt `pots` (wie »*plain old telephone system*«), und die Pfadkomponenten sind die internationale Vorwahl, die nationale Vorwahl (ohne die »0« am Anfang), die Nummer im Ortsnetz und eine lokale Durchwahlnummer. Sie könnten »relative« URIs wie `69/673060` oder `-101` erlauben und müssten dafür nur festlegen, dass sie als »Ende« des URI gemeint sind (wie das bei HTTP usw. ja auch gemacht wird). `69/673060` wäre also eine Nummer im nationalen Netz mit der (nationalen) Vorwahl `069` und keine Nummer auf einer pazifischen Insel mit einer sehr langen nationalen Vorwahl.

**2.1** Ja, solange Sie die Bestimmungen der Apache-Softwarelizenz einhalten. Sie dürfen Ihr Produkt zum Beispiel nicht »Apache-Wigwam« o. ä. nennen und müssen in der Dokumentation darauf aufmerksam machen, dass es Code der *Apache Software Foundation* enthält.

**2.6** Bei der letzten Prüfung (November 2009) waren es 10 für Apache 1.3 und 11 für Apache 2.x

**2.7** Im Februar 2013 waren es 54,68%.

**3.2** Eigene Module können Sie einbinden, indem Sie an geeigneter Stelle eine `LoadModule`-Direktive in die `httpd.conf`-Datei einfügen – jedenfalls solange es sich um ein dynamisch ladbares Modul handelt. Für statisch zu ladende Module müssen Sie den Apache neu übersetzen und dabei die Module in die Konfiguration aufnehmen.

**3.3** Der modulare Aufbau macht den Quellcode des Programms übersichtlicher und leichter zu verstehen, zu warten und so weiter. Es ist einfacher, Funktionalität ein- oder auszugliedern, ohne dass Sie den ganzen Quellcode kennen müssen.

Zur Laufzeit gibt es wohl keinen grossen Unterschied zwischen dem modular aufgebauten Apache und einer hypothetischen »unmodularen« Version.

Zu den Vorteilen des dynamischen Ladens gehört, dass ein größtenteils auf dynamisch ladbaren Modulen aufbauender Apache leichter zu installieren und zu warten ist (vor allem die Hersteller von Linux-Distributionen schätzen das). Vor allem können Sie sehr weitgehend ohne den Quellcode auskommen. Einige Nachteile sind, dass ein viele Module dynamisch ladender Apache länger zum Starten braucht (aber der Server soll ja laufen und nicht gestartet werden) und auf manchen Hardwareplattformen, etwa Intel-x86-Prozessoren, geringfügig langsamer läuft. Theoretisch setzen Sie sich auch der Gefahr aus, dass Cracker versuchen, Ihnen geänderte Module unterzuschieben.

### 3.6 Ein möglicher Ansatz wäre

```
ServerSignature EMail
ServerAdmin webmaster@example.com
```

(Sinnvollerweise verwenden Sie als E-Mail-Adresse ein Alias, das Sie bei Änderungen nur einmal, nämlich in /etc/aliases, umbiegen müssen.)

### 4.2 Die nötigen Direktiven sind

```
Options Indexes
IndexOptions FancyIndexing SuppressLastModified FoldersFirst
IndexOrderDefault Descending Size
ReadmeName LIESMICH
IndexIgnore bla*
```

### 4.5 Hier ist ein mögliches Beispiel:

```
URI: bla

URI: bla.mp3
Content-Type: audio/mpeg; qs=0.8

URI: bla.ra
Content-Type: audio/x-realaudio; qs=0.5

URI: bla.au
Content-Type: audio/basic; qs=0.1
```

### 4.7 Teil (a) realisieren Sie am bequemsten über UserDir:

```
UserDir /www
```

Für Teil (b) ist AliasMatch nötig, weil die Tilde unterdrückt werden soll:

```
AliasMatch ^/users/(.*) /www/$1
```

### 4.8 Mit RedirectMatch ginge es wie folgt:

```
RedirectMatch permanent ^/(marketing|entwicklung|support)/ \
http://$1.bla.de/
```

(Denken Sie daran, dass ein nicht angegebener Weiterleitungsstatus als temp interpretiert wird.)

**4.9** Ihre Umleitungsrichtive sollte ungefähr aussehen wie

```
ErrorDocument 404 /notfound.html
```

und Sie brauchen noch etwas Passendes in /notfound.html.

**5.1** Es dürfen nur die Rechner zugreifen, deren IP-Adressen im Netz 192.168.5.0/24 liegen, mit Ausnahme des Rechners 192.168.5.111. Alle anderen Rechner werden abgewiesen.**5.2** Ihr Zugriff wird abgewiesen, da die Adresse localhost (IP-Adresse 127.0.0.1) nicht in der Liste der erlaubten Rechner steht.**5.3** Probieren Sie etwas wie

```
SetEnvIf User-Agent Firefox ff_browser
<Directory /srv/www/htdocs>
  Order Allow,Deny
  Allow from all
  Deny from env=ff_browser
</Directory>
```

(Die feine englische Art ist sowas natürlich nicht.)

**5.4** Etwa so:

```
$ mkdir ~/public_html/secret
$ htpasswd -c ~/public_html/secret/.htpasswd lotte
New password: 123456
Re-type new password: 123456
Adding password for user lotte
$ htpasswd ~/public_html/secret/.htpasswd hugo
New password: abcdef
Re-type new password: abcdef
Adding password for user hugo
$ _
```

Beachten Sie, dass beim zweiten htpasswd-Aufruf die Option -c weggelassen werden muss!

**5.5** In jedem Fall brauchen Sie etwas wie

```
AuthType Basic
AuthName "Geheim"
AuthUserFile /home/hugo/public_html/secret/.htpasswd
```

Dazu kommt für Teil (a) die Zeile

```
Require valid-user
```

und für Teil (b) die Zeile

```
Require user lotte
```

Sie können diese Zeilen entweder in einem <Directory>-Block in der httpd.conf-Datei unterbringen oder (vermutlich sinnvoller) in einer .htaccess-Datei im zu schützenden Verzeichnis. In letzterem Fall müssen Sie darauf achten, dass für dieses Verzeichnis ein »AllowOverride AuthConfig« in Kraft ist.

## 5.6 Die Konfiguration hierfür ist etwas wie

```
Order Allow,Deny
Deny from all
Allow from 127.0.0.1
AuthType Basic
AuthName "Geheim"
AuthUserFile /home/hugo/public_html/secret/.htpasswd
Require valid-user
Satisfy any
```

Die wesentliche Zeile ist die letzte – sie erlaubt den Zugriff, wenn die IP-basierte Zugriffskontrolle *oder* die HTTP-basierte Authentisierung erfolgreich verläuft.

## 6.2 Einige Ideen:

1. An den Referer-Zeilen können Sie zum Beispiel sehen, über welche Suchbegriffe bei Suchmaschinen wie Google Benutzer zu Ihren Seiten gefunden haben. Die URLs von Google-Ergebnisseiten sehen zum Beispiel immer etwa so aus wie

```
http://www.google.com/search?q=<Suchbegriffe>&<Anderes>
```

Sie können also leicht feststellen, welche Begriffe jemand in Google eingegeben hat, um eine Ergebnisseite zu bekommen, auf der ein Verweis auf Ihre Seite(n) steht – denn den URL der Google-Ergebnisseite in der hier gezeigten Form bekommt Ihr Server als Referer-Kopfzeile übergeben.

2. Firmen, die Werbung im Web als Dienstleistung anbieten, verwenden Referer-Informationen, um Werbebanner zu optimieren. Werbebanner werden als Grafiken auf dem Web-Server der Firma – nennen wir sie mal »TripleClick« – realisiert, auf die die Seiten, wo die Banner stehen, per HTML--Link verweisen. Der TripleClick-Server bekommt als Referer den URL der Seite zu sehen, die Sie sich gerade anschauen, und kann so mit der Zeit Aussagen darüber machen, welche Seiten besonders populär sind und welche weniger (aus diesen Informationen lernt die Firma dann, welche Seiten sich zur Plazierung von Werbebanner lohnen – das Vergnügen kostet TripleClick ja Geld – und welche nicht; auf einer Seite, die im Vergleich selten bis nie angeschaut wird, muss man auch keine Werbung schalten). Aber es kommt noch besser: Wenn Sie ein Banner vom TripleClick-Server abrufen, dann versucht der Server Ihnen ein *Cookie* zu schicken – eine lange zufällige Zeichenkette, die Ihr Browser dem TripleClick-Server bei jeder künftigen Werbebanner-Anforderung (egal auf welcher Seite das Banner stehen soll) wieder zurückschickt und an der die Firma Sie wiedererkennen kann. Mit der Zeit kann TripleClick also ein »Profil« Ihrer Web-Nutzung aufbauen (jedenfalls soweit es die Seiten mit TripleClick-Werbung angeht – die einzigen, die die Firma interessieren) und dieses Profil anschließend benutzen, um Werbebanner auszusuchen, die Sie möglicherweise ansprechen. Wenn TripleClick also mitbekommt – aufgrund der Referer-Kopfzeilen –, dass Sie des öfteren auf mit TripleClick-Werbung versehene Seiten wie Autoportale, Routenplaner, Hotelführer für die Côte d'Azur und ähnliches zugreifen, wird man dort schließen, dass Sie sich für schnelle Autos und das Mittelmeer interessieren und die Werbebanner, die TripleClick Ihnen speziell zeigt (egal auf welcher Seite) entsprechend aussuchen, anstatt Sie mit Werbung zu langweilen, die Sie vielleicht sowieso nicht ansprechen würde. Wirklich interessant wird es in dem Moment, wo man Sie dazu kriegt, sich an einer Verlosung o.ä. zu beteiligen und dafür Ihre E-Mail-Adresse anzugeben – denn diese Adresse ist nicht nur als funktionierend bekannt, sondern auch noch mit detaillierten Konsumenteninformationen

korrelierbar, mithin also gewinnbringend zu verhökern. (Die deutsche Datenschutzgesetzgebung lässt solche Scherzchen nicht zu, jedenfalls nicht ohne Ihre explizite Zustimmung – aber andere Staaten, insbesondere die USA, sind da bei weitem nicht so kleinlich.)

### 6.3 Probieren Sie etwas wie

```
CustomLog /var/log/apache/ua.log "%h %t => %{User-Agent}i"
```

(Den Verzeichnisnamen für die Protokolldatei müssen Sie gegebenenfalls anpassen.)

### 6.4 Am besten definieren Sie sich ein kleines Shellskript wie:

```
#!/bin/sh

while read code request; do
    if [ "$code" = "500" ]; then
        mail -s "Error 500" webmaster@example.com <<ENDE
Error 500 ist aufgetreten in:

$request

ENDE
    fi
done
```

Dann können Sie mit etwas wie

```
CustomLog "|/usr/local/bin/code-500-mail" "%>s %r"
```

dafür sorgen, dass Ihr Shellskript das Protokoll mitlesen kann.

### 7.3 Eine mögliche Konfiguration hierfür wäre

```
<VirtualHost _default_:*>
    AliasMatch /(.*) /no-such-server.html
</VirtualHost>
```

**7.4** Damit das funktioniert, muss ein entsprechender namensbasierter virtueller Server als textuell erster virtueller Server für die betreffende Adresse in der Datei definiert sein.

**10.3** HTTP ist so konstruiert, dass auf eine Anfrage immer genau eine Antwort kommt. Für ein STARTTLS-Kommando ist in diesem Konzept nicht wirklich Platz. (Eigentlich stimmt das nicht: HTTP sieht durchaus die Möglichkeit vor, etwas wie STARTTLS zu integrieren. Allerdings würde das bedeuten, dass zum Abrufen von Ressourcen von einem HTTPS-Server für jede Ressource zwei »Rundreisen« vom Client zum Server gemacht werden müssten und nicht eine – unabhängig vom Überhang der SSL-Aushandlung. Das ist natürlich ineffizient.) (Tatsächlich unterstützen gute Web-Server wie zum Beispiel Apache einen »SSL-Sitzungscache«, um die Wiederverwendung von mühsam ausgehandelten SSL-Verbindungen zu ermöglichen. Das hat aber mit dem Protokoll HTTP als solchem nur am Rande zu tun.)

**10.6** Die Verfügbarkeit von OpenSSL macht es möglich, die Software von unabhängigen Experten prüfen zu lassen. So lässt sich leichter sicherstellen, dass die kryptografischen Verfahren und ihre Implementierung korrekt sind. Fehler lassen sich schneller finden und reparieren. Außerdem ist SSL ein wichtiger Bestandteil der Infrastruktur und sollte nicht von einer einzigen Firma kontrolliert werden. Die Freiheit der Software garantiert ihr Fortbestehen, solange sie für die Nutzergemeinde interessant ist.

**11.5** Das Zertifikat bekommen Sie normalerweise mit Ihrem Browser geliefert. Sie können die Authentizität aber prüfen, indem Sie den »Fingerabdruck« (engl. *fingerprint*) des Zertifikats bilden und mit dem »offiziellen« vergleichen. Den Fingerabdruck eines Zertifikats erhalten Sie mit

```
$ openssl x509 -in cacert.pem -noout -fingerprint
```

Nach dem offiziellen Fingerabdruck müssen Sie die Zertifizierungsstelle fragen – am besten per Telefon.

**12.1** Sie könnten zum Beispiel die folgende Testseite als `index.html` im DocumentRoot-Verzeichnis Ihres Servers verwenden:

```
<html>
<head><title>Testseite</title></head>
<body>
  <h1>HTTP<!--#if expr="%{HTTPS} == 'on'" -->S<!--#endif --></h1>
  <!--#if expr="%{HTTPS} == 'on'" -->
  <p>Dieser Inhalt ist mit HTTPS gesichert.</p>
  <p>Schl&uuml;ssel: <!--#echo var="SSL_CIPHER" --></p>
  <p>Benutzer: <!--#echo var="SSL_CLIENT_S_DN" -->
    (<!--#echo var="REMOTE_USER" -->)</p>
  <!--#else -->
  <p>Dieser Inhalt ist nicht mit HTTPS gesichert (WTF!?).</p>
  <!--#endif -->
</body>
```

Damit diese Seite funktioniert, müssen Sie natürlich *server-side includes* aktivieren, etwa mit

```
DocumentRoot /srv/www/apw2-htdocs
<Directory /srv/www/apw2-htdocs>
  Require all granted
  SSLOptions +StdEnvVars
  Options +Includes
  AddOutputFilter INCLUDES .html
  DirectoryIndex index.html
</Directory>
```

(für Apache 2.4). Wenn Ihr Browser zickt, können Sie auch »`openssl s_client`« verwenden, um den Server zu testen; versuchen Sie nach dem Verbindungsauftakt eine HTTP-Anfrage wie

```
GET / HTTP/1.1
Host: www.example.com
<Leerzeile>
```

**12.6** Benutzen Sie die Optionen `SSLCACertificateFile` oder `SSLCACertificatePath`. Im letzteren Fall müssen Sie noch die korrekten *hash filenames* anlegen.

## 12.8 Im einfachsten Fall etwas wie

```
#!/bin/sh
echo 'Content-Type: text/plain'
echo ''
printenv | grep '^SSL_' | sort
```

Achten Sie darauf, dass dieses Skript keine Programmaufrufe von Sachen abhängig macht, die der Client kontrolliert. Ein Aufruf wie

```
echo 'SSL_CLIENT_I_DN: '$SSL_CLIENT_I_DN
```

ist potentiell gefährlich.

## 12.9 Zum Beispiel:

- Der Inhaber des Zertifikats arbeitet in der Entwicklungsabteilung der »Beispiel GmbH«:

```
SSLRequire %{SSL_CLIENT_S_DN_0} eq "Beispiel GmbH" \
&& %{SSL_CLIENT_S_DN_OU} eq "Entwicklung"
```

- Das Zertifikat wurde von der Zertifizierungsstelle der »Beispiel GmbH« ausgestellt:

```
SSLRequire %{SSL_CLIENT_I_DN_0} eq "Beispiel GmbH" \
&& %{SSL_CLIENT_I_DN_OU} eq "Zertifizierungsstelle"
```

- Der Zugriff erfolgt von einem Rechner im Netz 10.11.12.0/24 aus:

```
SSLRequire %{REMOTE_ADDR} =~ m/^10\.11\.12\.[0-9]+$/
```

- Der Zugriff erfolgt in der Adventszeit (1.–24.12.):

```
SSLRequire %{TIME_DAY} <= 24 && %{TIME_MONTH} == 12
```

## 12.12 Der Angreifer wählt einen Namen innerhalb von `example.com`, der höchstwahrscheinlich nicht anderweitig benutzt wird (etwa `quxbarfelprartz.example.com`) und arrangiert, dass er im DNS registriert wird und auf einen Web-Server zeigt, den der Angreifer kontrolliert. Das hat die folgenden Konsequenzen:

- Der Browser des Opfers hat höchstwahrscheinlich keine HSTS-Einstellung für `quxbarfelprartz.example.com`.
- Eine Anfrage, die an `https://quxbarfelprartz.example.com` geschickt wird, enthält den gewünschten Cookie.
- Wenn `quxbarfelprartz.example.com` beim Verbindungsaufbau ein Zertifikat schickt und der Benutzer (wie üblich) durch die entsprechenden Warnungen »durchklickt«, dann kann der Angreifer den »sicheren« Cookie abgreifen.

Mit `includeSubDomains` führt das Zertifikatsproblem in Schritt 3 zum Verbindungsabbruch. (Siehe auch [RFC6797, Abschnitt 14.4].)

## 12.15 Ein HPKP-kompatibler Browser muss bei Problemen mit den Pins die Verbindung zu `www.example.com` abbrechen. Wenn ein Pin-Problem mit `https://www.example.com` aber an `https://www.example.com` gemeldet werden soll, ist nur damit zu rechnen, dass dasselbe Problem dann wieder auftritt und so keine Meldung erfolgen kann.

**13.2** Der wesentliche Unterschied zwischen der direkten Anfrage und der Proxy-Anfrage sollte in der HTTP-Anfragezeile zu sehen sein:

GET / HTTP/1.1	direkte Anfrage
GET http://www.example.org/ HTTP/1.1	Anfrage an den Proxy

Welche Unterschiede sehen Sie sonst noch?

**13.3** Hier wird der Browser bei der direkten Anfrage versuchen, eine SSL-Verbindung auszuhandeln. Ohne die passenden Antworten geht das natürlich schief (aber Sie können statt netcat ja mal »openssl s_server« mit den entsprechenden Zertifikaten ausprobieren – mehr über SSL erfahren Sie zum Beispiel in der Line-up-Front-Schulungsunterlage *Sichere Webserver mit Apache und SSL*). Im Proxy-Fall versucht der Browser, über eine HTTP-CONNECT-Anfrage einen Tunnel zu `www.example.com`, Port 443, aufzubauen; die SSL-Verbindung würde dann mit jenem Rechner ausgehandelt.

**14.1** Hierfür können wir natürlich keine allgemeingültige Musterlösung angeben. Auf dem Debian-GNU/Linux-System der Redaktion findet die Konfiguration sich in `/etc/squid/squid.conf`, der Cache in `/var/spool/squid` mit 16 Unterverzeichnissen in der ersten und jeweils 256 in der zweiten Ebene. Squid läuft als Benutzer »proxy« mit der primären Gruppe »proxy«, und die Protokolldateien stehen in `/var/log/squid`. Bei anderen Debian-artigen Distributionen (Ubuntu) dürfte es so ähnlich sein, und vermutlich werden auch Red Hat und SUSE heutzutage nicht allzu weit hiervon abweichen ...

**14.2** Hierzu müssen Sie die `cache_dir`-Direktive ändern, insbesondere die beiden Zahlen hinter der Größenangabe. Versuchen Sie mal »32 32«, und vergessen Sie nicht das »`squid -z`« hinterher. Im wirklichen Leben gibt es keinen vernünftigen Grund, diese Parameter zu ändern.

**14.3** Mit `netstat` (versuchen Sie die Optionenkombination »`-tulp`«) sollten Sie sehen, dass Squid auf dem designierten Port auf Verbindungen lauscht. Je nach der Konfiguration kann es sein, dass er auch andere Ports offen hat, typischerweise den ICP-Port 3130/udp.

**14.10** Eigentlich steht die Lösung für diese Aufgabe schon im Haupttext. Auf Basis der IP-Adressen (gehen wir mal davon aus, dass Ihr Netz die Adresse `10.11.0.0/16` hat) könnte das so aussehen:

```
acl mynetwork src 10.11.0.0/16
http_access allow mynetwork
http_access deny all
```

(wobei wir `all` als »`src 0.0.0.0/0.0.0.0`« voraussetzen).

**16.8** Klassifikation, sowie Text nach RFC 959.

**150** Positiv ankündigende Meldung zum Dateisystem – *File status okay; about to open data connection.*

**214** Positiv abschließende informative Meldung – *Help message.*

**426** Vorübergehend negative Meldung zur Verbindung – *Connection closed; transfer aborted.*

**504** Dauerhaft negative Meldung zur Syntax – *Command not implemented for that parameter.*

**553** Dauerhaft negative Meldung zum Dateisystem – *Requested action not taken. File name not allowed.*

**17.3** Der Server muss Anfragen den Servern zuordnen können. HTTP stellt dafür die Host-Kopfzeile zur Verfügung – etwas, was FTP nicht kennt.

**18.5** Wenn Pure-FTPd nur anonyme Zugriffe zuläßt, ist der Benutzername, den Sie eingeben, irrelevant – Sie landen immer im Verzeichnis mit den "anonymen" Inhalten.

**18.10** Die Standardeinstellung (keine Optionen) erlaubt sowohl anonyme wie auch authentisierte Zugriffe. Um anonyme Zugriffe auszuschließen, müssen Sie die Option **-E** (**--noanonymous**) angeben.

**18.11** Am einfachsten geht das – jedenfalls solange Ihr Pure-FTPd Unix- oder PAM-Authentisierung benutzt –, indem Sie **hugo** in die Datei **/etc/ftpusers** eintragen. Erinnern Sie sich: In dieser irreführend benannten Datei stehen genau diejenigen Benutzer, die FTP *nicht* benutzen dürfen.

**18.13** Einsperren im Heimatverzeichnis mit den PureFTPd-Optionen **-a** oder **-A**.

**18.14** Spezifizieren Sie das Heimatverzeichnis für **hugo** in **/etc/passwd** wie folgt:

```
/home/hugo/.html
```

(sinngemäß für die anderen Benutzer). Die Optionen **-a** oder **-A** müssen Sie dann nicht mehr angeben.

**18.19** Sie können etwas benutzen wie

```
# /usr/sbin/pure-ftpd -lpam -lpuredb:/etc/pureftpd.pdb &
```

Bei mehreren **-l**-Optionen arbeitet Pure-FTPd sie von links nach rechts ab.

**18.22** Schreiben Sie etwas wie

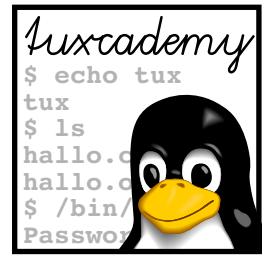
```
* 0 * * * root echo >/srv/ftp/.banner Guten Morgen!
* 9 * * * root echo >/srv/ftp/.banner Mahlzeit!
* 14 * * * root echo >/srv/ftp/.banner Guten Tag!
* 18 * * * root echo >/srv/ftp/.banner Guten Abend!
```

nach **/etc/cron.d/pure-ftpd**.

**18.23** Sie können mit etwas wie »**logger -p ftp.warn ...**« auf Ihrem FTP-Server testen, ob die Konfiguration des Systemprotokolldiensts funktioniert, auch ohne dass Sie mit einem FTP-Client herummachen müssen. Wenn das klappt, sollten Sie es natürlich auch nochmal mit Pure-FTPd versuchen.

**18.26** Hierfür brauchen Sie die Option **-V**, um eine vertrauenswürdige Adresse spezifizieren zu können.





# B

## HTTP-Statuswerte

### B.1 Information

100 Continue  
101 Switching Protocols

### B.2 Erfolg

200 OK  
201 Created  
202 Accepted  
203 Non-Authoritative Information  
204 No Content  
205 Reset Content  
206 Partial Content

### B.3 Umleitung

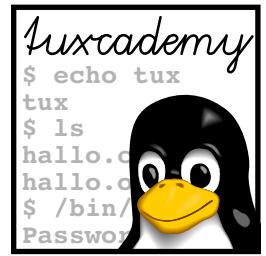
300 Multiple Choices  
301 Moved Permanently  
302 Found  
303 See Other  
304 Not Modified  
305 Use Proxy  
306  
307 Temporary Redirect

## B.4 Client-Fehler

400 Bad Request  
401 Unauthorized  
402 Payment Required  
403 Forbidden  
404 Not Found  
405 Method Not Allowed  
406 Not Acceptable  
407 Proxy Authentication Required  
408 Request Timeout  
409 Conflict  
410 Gone  
411 Length Required  
412 Precondition Failed  
413 Request Entity Too Large  
414 Request-URI Too Long  
415 Unsupported Media Type  
416 Request Range Not Satisfiable  
417 Expectation Failed

## B.5 Server-Fehler

500 Internal Server Error  
501 Not Implemented  
502 Bad Gateway  
503 Service Unavailable  
504 Gateway Timeout  
505 HTTP Version Not Supported



# C

## Apache-Direktiven

Hier sind die in dieser Schulungsunterlage besprochenen Apache-Direktiven kurz zusammengefasst. Für jede Direktive gibt es eine Informationszeile und eine kurze Erklärung. Die Informationszeile gibt die Syntax der Direktive und (in Klammern) einen etwaigen Standardwert wieder; außerdem führt sie das Modul auf, das die Direktive definiert (sofern sie nicht Standardbestandteil des Apache ist) und gibt den Kontext an, in dem die Direktive auftauchen darf. Hierbei steht »G« für die »globale« Ebene der Konfigurationsdatei, »V« für <VirtualHost>-Blöcke, »B« für <Directory>-, <Files>- und <Location>-Blöcke und »D« für .htaccess-Dateien. Am Ende der Erklärung steht ein Verweis auf die Seite in der Schulungsunterlage, die die ausführliche Erläuterung der Direktive enthält.

AccessFileName <i>&lt;Datei&gt;</i> ...	GV
Gibt den (oder die) Namen von verzeichnisspezifischen Konfigurationsdateien an (meist .htaccess)	35
AddAlt <i>&lt;Text&gt;</i> <i>&lt;Name&gt;</i> ...	mod_autoindex GVBD
Definiert <i>&lt;Text&gt;</i> als Bild-Ersatztext für die Dateien mit der Endung (den Endungen) <i>&lt;Name&gt;</i> in automatischen Verzeichnislisten	42
AddAltByEncoding <i>&lt;Text&gt;</i> <i>&lt; MIME-Codierung &gt;</i> ...	mod_autoindex GVBD
Definiert <i>&lt;Text&gt;</i> als Bild-Ersatztext für die Dateien mit der MIME-Codierung (den Codierungen) <i>&lt; MIME-Codierung &gt;</i> in automatischen Verzeichnislisten	42
AddAltByType <i>&lt;Text&gt;</i> <i>&lt; MIME-Typ &gt;</i> ...	mod_autoindex GVBD
Definiert <i>&lt;Text&gt;</i> als Bild-Ersatztext für die Dateien mit dem MIME-Typ (den MIME-Typen) <i>&lt; MIME-Typ &gt;</i> in automatischen Verzeichnislisten	42
AddCharset <i>&lt;Zeichencodierung&gt;</i> <i>&lt;Endung&gt;</i> ...	mod_mime GVBD
Dateien mit Endung(en) <i>&lt;Endung&gt;</i> ... sind nach <i>&lt;Zeichencodierung&gt;</i> codiert	49
AddDescription <i>&lt;Zeichenkette&gt;</i> <i>&lt;Dateiname&gt;</i> ...	mod_autoindex GVBD
Definiert <i>&lt;Zeichenkette&gt;</i> als Erklärung für <i>&lt;Dateiname&gt;</i> in automatischen Verzeichnislisten.	42
AddIcon <i>&lt;Bild&gt;</i> <i>&lt;Name&gt;</i> ...	mod_autoindex GVBD
Definiert <i>&lt;Bild&gt;</i> als Sinnbild für die Dateien mit der Endung (den Endungen) <i>&lt;Name&gt;</i> in automatischen Verzeichnislisten	42
AddIconByEncoding <i>&lt;Bild&gt;</i> <i>&lt; MIME-Codierung &gt;</i> ...	mod_autoindex GVBD
Definiert <i>&lt;Bild&gt;</i> als Sinnbild für die Dateien mit der MIME-Codierung (den Codierungen) <i>&lt; MIME-Codierung &gt;</i> in automatischen Verzeichnislisten	42
AddIconByType <i>&lt;Bild&gt;</i> <i>&lt; MIME-Typ &gt;</i> ...	mod_autoindex GVBD
Definiert <i>&lt;Bild&gt;</i> als Sinnbild für die Dateien mit dem MIME-Typ (den MIME-	

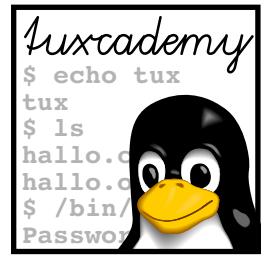
Typen) < MIME-Typ > in automatischen Verzeichnislisten	42
AddLanguage < MIME-Sprache > < Endung > ...	mod_mime GVBD
Dateien mit Endung(en) < Endung > ... sind in Sprache < MIME-Sprache >	48
Alias < URL-Pfad > < Datei > < Verzeichnis >	mod_alias GV
Blendet < Datei > bzw. < Verzeichnis > an der Stelle < URL-Pfad > in die Ressourcen- hierarchie ein	50
AliasMatch < RegAusdruck > < Datei > < Verzeichnis >	mod_alias GV
Falls der aktuelle URL-Pfad auf < RegAusdruck > passt: Gehe über zu < Datei > bzw. < Verzeichnis > (mit Ersetzung)	50
Allow from all < Rechner > env=< UVar >	mod_authz_host BD*
Lässt Zugriffe von allen Rechnern/denjenigen, die auf < Rechner > passen/ denen mit der Umgebungsvariable < UVar > zu	60
AllowOverride All None < Typ > ... (None)	D
Gibt an, welche Arten von Direktiven in .htaccess-Dateien auftreten können	34
AuthBasicAuthoritative On Off (On)	mod_auth_basic BD
Ob mod_auth_basic Anfragen an andere Authentisierungsmodule delegiert, wenn es keine Benutzerdaten finden kann	65
AuthBasicProvider < Quelle > ... (file)	mod_auth_basic BD
Gibt an, wo Benutzerdaten für Basic-Authentisierung gesucht werden	65
AuthDigestProvider < Name > [...] file	mod_auth_digest BD
Bestimmt den oder die »Lieferanten« für Authentisierungdaten für Digest- Authentisierung.	67
AuthGroupFile < Datei >	mod_authz_groupfile BD
Bestimmt den Namen der Gruppendatei für Basic-Authentisierung	65
AuthName < AuthBereich >	BD
Bestimmt den Namen des Authentisierungsbereichs (wird im Eingabedialog des Browsers angezeigt)	63
AuthType Basic Digest	BD
Bestimmt die Art der Authentisierung	64
AuthUserFile < Datei >	mod_authn_file BD
Bestimmt den Namen der Kennwortdatei für Basic-Authentisierung	64
CustomLog < Ziel > < Format > [env=< UVar >]	mod_log_config GV
Beschreibt Ort und Format einer Protokolldatei	72
DefaultIcon < Bild >	mod_autoindex GVBD
Definiert < Bild > als Sinnbild für Dateien ohne anderes Sinnbild in automati- schen Verzeichnislisten	42
DefaultLanguage < MIME-Sprache >	mod_mime GVBD
Legt eine Standardsprache für alle nicht explizit gekennzeichneten Dateien fest	49
Deny from all < Rechner > env=< UVar >	mod_authz_host BD*
Weist Zugriffe von allen Rechnern/denjenigen, die auf < Rechner > passen/ denen mit der Umgebungsvariable < UVar > ab	60
DirectoryIndex disabled < lokaler URL > ... (index.html)	mod_dir GVBD
Gibt an, welche Ressourcen als Verzeichnisinhaltsliste geliefert werden (falls sie existieren)	38
DirectorySlash On Off (On)	mod_dir GVBD
Ob Anfragen nach Verzeichnis-URLs ohne Schrägstrich am Ende umgeleitet werden sollen	39
DocumentRoot < Verzeichnis > (/usr/local/apache/htdocs)	GV
Verzeichnis, in dem die WWW-Ressourcenhierarchie steht. URL-Pfade wer- den an dieses Verzeichnis angehängt	31

ErrorDocument <Statuscode> <Dokument>	GVBD
Definiert eine eigene Fehlerseite für HTTP-Status <Statuscode>	57
ErrorLog <Datei> syslog[:<facility>] (logs/error_log)	GV
Protokolliert Fehlermeldungen nach <Datei> oder syslog	75
Group <Gruppenname> #(GID) (#-1)	GV
Gibt die Unix-Gruppe an, mit deren Identität Anfragen bearbeitet werden	30
HeaderName <Datei>	mod_autoindex GVBD
Kopiert den Inhalt von <Datei> vor automatische Verzeichnislisten	43
HostNameLookups on off double (off)	GBV
Bestimmt, ob IP-Adressen in DNS-Namen aufgelöst werden	72
IncludeOptional <Datei> (<Verzeichnis>)\(<Suchmuster>)	GVD
(Seit Apache 2.4.) Funktioniert wie Include, aber produziert keine Fehlermeldung, wenn auf ein Suchmuster keine Dateien oder Verzeichnisse passen.	27
IndexIgnore <Datei> ...	mod_autoindex GVBD
Ignoriert Dateien, deren Namen auf <Datei> ... passen, für automatische Verzeichnislisten	43
IndexOptions [+ -]<Option> ...	mod_autoindex GVBD
Setzt Optionen für automatische Verzeichnislisten	40
IndexOrderDefault <Richtung> <Feld>	mod_autoindex GVBD
Bestimmt die anfängliche Sortierung einer automatischen Verzeichnisliste	41
IndexStyleSheet <URL-Pfad>	mod_autoindex GBVD
Definiert ein CSS-Stylesheet für automatische Verzeichnislisten.	43
KeepAlive On Off (On)	GV
Ob mehr als ein HTTP-Anfrage/Antwortpaar über dieselbe TCP-Verbindung geschickt werden darf	27
KeepAliveTimeout <Zahl> (5)	GV
Zeit, die Apache abwartet, ob über eine aktive TCP-Verbindung noch eine weitere Anfrage geschickt wird	27
Listen [<IP-Adresse>:<Port> [<Protokoll>]	G*
Gibt eine Adresse (optional) und einen Port an, wo Apache lauschen soll	28
ListenBacklog <n> (511)	G
Die maximale Größe der Warteschlange von auf TCP-Ebene ausstehenden Anfragen.	103
LoadFile <Datei> ...	mod_so G
Lädt die Datei(en) <Datei> (die ausführbaren Code enthalten müssen)	28
LoadModule <Modul> <Datei>	mod_so G
Lädt die <Datei> und aktiviert sie als Modul <Modul>	28
LogFormat <Format> (<Formatname> [<Formatname>])	mod_log_config GV
Definiert Protokollformat <Formatname> zu <Format>, oder setzt <Format> bzw. <Formatname> als Format für folgende TransferLog-Direktiven	72
LogLevel <Fehlerstufe> (warn)	GV
Bestimmt, Meldungen ab welcher Stufe ins ErrorLog eingetragen werden	75
MaxClients <n> (256)	G
Die Maximalzahl von gleichzeitigen Anfragen, die bearbeitet werden können.	103
MaxKeepAliveRequests <Zahl> (100)	GV
Maximale Anzahl der aufeinanderfolgenden HTTP-Anfragen/Antworten auf derselben TCP-Verbindung (0 = unbegrenzt viele)	27
MaxRequestsPerChild <n> (0)	G
Gibt die Maximalzahl von Anfragen an, die ein Kindprozess bearbeitet, bevor er beendet wird (0 = unbegrenzt viele).	103

MaxSpareServers <i>&lt;n&gt;</i> (10)	G
Gibt die Maximalzahl von Kindprozessen an, die nichts zu tun haben.	103
MinSpareServers <i>&lt;n&gt;</i> (5)	G
Gibt die Minimalzahl von Kindprozessen an, die nichts zu tun haben.	102
NameVirtualHost <i>&lt;IP-Adresse&gt;[:&lt;TCP-Port&gt;]</i>	G
Schaltet die Kombination <i>&lt;IP-Adresse&gt;:&lt;TCP-Port&gt;</i> für namensbasierte virtuelle Server frei. <i>&lt;TCP-Port&gt;</i> ist 80, wenn nichts anderes angegeben wurde	87
Options [+ -]<Option> ... (FollowSymLinks)	GVBD
Bestimmt, welche Serveroptionen in einem bestimmten Verzeichnis (und seinen Unterverzeichnissen) gelten	33
Order Allow,Deny Deny,Allow Mutual-failure (Deny,Allow)	mod_authz_host BD
Bestimmt die Reihenfolge der Auswertung von Allow- und Deny-Direktiven	61
PidFile <i>&lt;Datei&gt;</i> (logs/httpd.pid)	G
Name der Datei, wo Apache seine PID hinterlegt	27
ReadmeName <i>&lt;Datei&gt;</i>	mod_autoindex GVBD
Hängt den Inhalt von <i>&lt;Datei&gt;</i> an automatische Verzeichnislisten an	43
Redirect [<Status>] <URL-Pfad> <URL>	mod_alias GVBD
Leitet Zugriffe auf <URL-Pfad> weiter an <URL>; <Status> spezifiziert Art der Weiterleitung	51
RedirectMatch [<Status>] <RegAusdruck> <URL>	mod_alias GVBD
Leitet Zugriffe auf URL-Pfade, die auf <RegAusdruck> passen, weiter an <URL> (mit Ersetzung); <Status> spezifiziert Art der Weiterleitung	52
RedirectPermanent <URL-Pfad> <URL>	mod_alias GVBD
Leitet Zugriffe auf <URL-Pfad> weiter an <URL>. Veraltet; Redirect benutzen.	52
RedirectTemp <URL-Pfad> <URL>	mod_alias GVBD
Leitet Zugriffe auf <URL-Pfad> »temporär« weiter an <URL>. Veraltet; Redirect benutzen.	52
Require valid-user user <Benutzer> ... group <Gruppe> ...	BD
Bestimmt Anforderungen für eine erfolgreiche HTTP-basierte Authentisierung	65
SSLCACertificateFile <Dateiname>	mod_ssl GV
Die benannte Datei enthält die Zertifikate von Zertifizierungsstellen zur Prüfung von Client-Zertifikaten.	154
SSLCACertificatePath <Verzeichnisname>	mod_ssl GV
Das benannte Verzeichnis enthält die Zertifikate von Zertifizierungsstellen zur Prüfung von Client-Zertifikaten. Die Zertifikate werden über <i>hash filenames</i> mit der Endung .N gefunden.	154
SSLCertificateChainFile <Datei>	mod_ssl GV
In dieser Datei steht die komplette »Kette« des Zertifikats des Servers.	139
SSLCertificateFile <Datei>	mod_ssl GV
In dieser Datei steht das Zertifikat des Servers.	139
SSLCertificateKeyFile <Datei>	mod_ssl GV
In dieser Datei steht der private Schlüssel des Servers.	139
SSLCipherSuite <Schlüsselliste> (Siehe unten)	mod_ssl GVBD
Gibt die akzeptablen TLS-Verschlüsselungsverfahren an. Details stehen in der Dokumentation zu mod_ssl. Die Voreinstellung hängt von OpenSSL ab.	135
SSLEngine on off optional off	mod_ssl GV
Schaltet SSL für den betreffenden (globalen oder virtuellen) Server ein oder aus.	134
SSLProtocol [+ -]<Protokoll> ... (all)	mod_ssl GV
Gibt an, welche SSL-Protokollversionen akzeptabel sind.	135

SSLRandomSeed <Kontext> <Quelle> [<Bytes>]	mod_ssl	G
Gibt die Quelle(n) für Zufallszahlen an.		137
SSLRequire <Bedingung>	mod_ssl	BD
Stellt Bedingungen für Client-Zertifikate auf. (Verpönt in Apache 2.4.)	156	
SSLRequireSSL	mod_ssl	BD
Lehnt Zugriffe ab, die nicht über TLS erfolgen.		146
SSLVerifyClient <Stufe> none	mod_ssl	GVBD
Gibt an, ob und mit welchem Nachdruck Apache ein Client-Zertifikat anfordert.		155
SSLVerifyDepth <Tiefe> 1	mod_ssl	GVBD
Gibt die maximale akzeptable Tiefe der Zertifikatskettenprüfung für Client-Zertifikate an.		155
Satisfy any all (all)		BD
Entscheidet, ob TCP-basierte Zugriffskontrolle und HTTP-Authentisierung UND- oder ODER-verknüpft werden		68
ScoreBoardFile <Datei> (logs/apache_status)		G
Auf manchen Architekturen der Name der Datei, die Apache zur Kommunikation mit seinen Kindprozessen benutzt		27
ScriptAlias <URL-Pfad> <Datei> <Verzeichnis>	mod_alias	GV
Blendet <Datei> bzw. <Verzeichnis> an der Stelle <URL-Pfad> in die Ressourcenhierarchie ein und betrachtet darin enthaltene Dateien als CGI-Skripte		56
ScriptAliasMatch <RegAusdruck> <Datei> <Verzeichnis>	mod_alias	GV
Falls der aktuelle URL-Pfad auf <RegAusdruck> passt: Gehe über zu <Datei> bzw. <Verzeichnis> (mit Ersetzung); dort vorhandene Dateien gelten als CGI-Skripte		56
ServerAdmin <E-Mail-Adresse> <URL>		GV
E-Mail-Adresse (eines Server-Administrators), die bei Fehlermeldungen usw. eingesetzt wird		30
ServerAlias <Rechnername> ..		V
Setzt zusätzliche Namen für einen namensbasierten virtuellen Server		88
ServerName [<Schema>://]<FQDN>[:<Port>]		GV
Rechnername des Servers für Umleitungs-URLs; Identifizierung von <VirtualHost>-Blöcken für namensbasierte virtuelle Server		29
ServerRoot <Verzeichnis> (/usr/local/apache)		G
Gibt das Hauptverzeichnis für den Apache-Server an; der Wert dieser Direktive wird als Präfix für andere (relative) Namen benutzt		27
ServerSignature On Off EMail (0ff)		GVBD
Konfiguriert die »Fußzeile« von automatisch generierten Seiten		30
ServerTokens Major Minor Minimal ProductOnly OS Full (Full)		G
Konfiguriert den Inhalt der HTTP-Kopfzeile Server		30
SetEnvIf <Attribut> <RegAusdruck> <Variable>[=<Wert>] ...	mod_setenvif	GVBD
Setzt Umgebungsvariable in Abhängigkeit von Attributen einer Anfrage und HTTP-Kopfzeilen.		62
StartServers <n> (5)		G
Gibt an, wieviele Kindprozesse beim Start des Servers angelegt werden.		102
TimeOut <Zahl> (60)		GV
Wartezeit für den Abbruch von TCP-Verbindungen (in Sekunden)		27
TransferLog <Datei> <Pipe>	mod_log_config	GV
Definiert eine Protokolldatei mit dem Format, das bei der letzten LogFormat-Direktive mit nur einem Argument angegeben wurde, ersatzweise <i>Common Log Format</i>		75
UseCanonicalName On Off DNS (On)		GV
Bestimmt, welcher Rechnername in selbstbezüglichen URLs benutzt wird		29

UseCanonicalPhysicalPort On Off (Off)	GV ^B
Bestimmt, welche Portnummer in selbstbezüglichen URLs benutzt wird	29
User <Benutzername> #<UID> (#-1)	GV
Gibt den Unix-Benutzer an, mit dessen Identität Anfragen bearbeitet werden	
	30
UserDir <Verzeichnis> ...	mod_userdir GV [*]
Gibt an, ob benutzereigene Verzeichnisse unterstützt werden und wenn ja, wo	44
XBitHack on off full (off)	mod_include GVBD
Gibt an, ob HTML-Dateien mit gesetztem x-Bit nach <i>server-side includes</i> durch- sucht werden	53



# D

## Variable für CGI-Skripte und *server-side includes*

### D.1 Nicht anfragespezifische Variable

Diese Umgebungsvariablen werden für jede Anfrage gesetzt.

**SERVER_SOFTWARE** Name und Versionsnummer des HTTP-Servers im Format »<Name>/<Version>«

**SERVER_NAME** Der Name des Servers, so wie er in selbstbezüglichen URLs auftauchen würde

**GATEWAY_INTERFACE** Die Version der CGI-Spezifikation, die dieser Server einhält, im Format »CGI/<Version>«

### D.2 Anfragespezifische Variable

Diese Umgebungsvariablen werden auf der Basis der zu bearbeitenden HTTP-Anfrage gesetzt.

**SERVER_PROTOCOL** Name und Versionsnummer des Protokolls, über das die Anfrage eingereicht wurde, im Format »<Name>/<Version>«

**SERVER_PORT** Portnummer, an die die Anfrage geschickt wurde

**REQUEST_METHOD** Methode, mit der die Anfrage geschickt wurde (für HTTP ist das GET, POST, ...)

**PATH_INFO** Der Teil des URL-Pfads hinter dem Namen des CGI-Skripts, so wie der Browser ihn angegeben hat, in ggf. vom Server dekodierter Form

**PATH_TRANSLATED** Eine ins Dateisystem des Servers übersetzte Version von PATH_INFO

**SCRIPT_NAME** Der URL-Pfad des ausgeführten CGI-Skripts, für selbstbezügliche URLs

**QUERY_STRING** Der Anteil der URL nach einem Fragezeichen (»?«), undekodiert.

**REMOTE_HOST** Der DNS-Name des Rechners, von dem die Anfrage kommt. Ist dieser nicht bekannt, sollte diese Variable nicht gesetzt werden

**REMOTE_ADDR** Die IP-Adresse des Rechners, von dem die Anfrage kommt

**AUTH_TYPE** Wenn das Skript mit HTTP-Authentisierung geschützt ist: Der Name des Authentisierungsverfahrens (z. B. Basic)

**REMOTE_USER** Wenn das Skript mit HTTP-Authentisierung geschützt ist: Der authentisierte Benutzername aus der Anfrage

**REMOTE_IDENT** Der lokale Name des Benutzers auf dem anfragenden Rechner, über das IDENT-Protokoll (nur für Protokollzwecke zu verwenden!)

**CONTENT_TYPE** MIME-Datentyp der eventuell mit einer Anfrage verbundenen Nutzdaten

**CONTENT_LENGTH** Die Länge dieser Nutzdaten in Bytes, gemäß den Angaben des Browsers

Außerdem werden die HTTP-Kopfzeilen aus der Anfrage in die Umgebung gestellt. Die Namen der korrespondierenden Umgebungsvariablen ergeben sich aus den Namen der Kopfzeilen wie folgt:

- Das Präfix `HTTP_` wird davorgesetzt
- Alle »-«-Zeichen werden zu »_«-Zeichen gemacht

Der Server darf Kopfzeilen weglassen, die er schon selbst bearbeitet hat (etwa `Authorization`) oder die vom System vorgegebene Grenzen für die Prozessumgebung sprengen würden.

Mit Modulen wie `mod_setenvif` kann der Server weitere Umgebungsvariable setzen.

### D.3 Zusätzliche Variable für *server-side includes*

Die folgenden Variablen stehen in HTML-Dateien mit *server-side includes* für Direktiven wie `#echo` oder `#if` zur Verfügung:

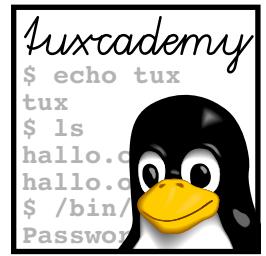
**DATE_GMT** Die aktuelle Zeit (mit Datum) in Weltzeit

**DATE_LOCAL** Die aktuelle Zeit (mit Datum) in der lokalen Zeitzone

**DOCUMENT_NAME** Der Name (ohne Verzeichnisse) des Dokuments

**DOCUMENT_URI** Der (%-dekodierte) URL-Pfad des Dokuments (bei `#include` nicht der Name des *aktuell* betrachteten Dokuments)

**LAST_MODIFIED** Der Zeitpunkt der letzten Veränderung des Dokuments



# Index

Dieser Index verweist auf die wichtigsten Stichwörter in der Schulungsunterlage. Besonders wichtige Stellen für die einzelnen Stichwörter sind durch **fette** Seitenzahlen gekennzeichnet. Sortiert wird nur nach den Buchstaben im Indexeintrag; „`~/.bashrc`“ wird also unter „B“ eingeordnet.

- 3DES, 113
- ABOR, 209
- Accept (HTTP-Kopfzeile), 46
- Accept-Charset (HTTP-Kopfzeile), 7, 46
- Accept-Encoding (HTTP-Kopfzeile), 7, 46
- Accept-Language (HTTP-Kopfzeile), 7, 46, 49
- access_log, 21, 72–73, 75
- AccessFileName (`httpd.conf`), 35
- acl (`squid.conf`), 190
- Action (`httpd.conf`), 34
- AddAlt (`httpd.conf`), 42
- AddAltByEncoding (`httpd.conf`), 42
- AddAltByType (`httpd.conf`), 42
- AddCharset (`httpd.conf`), 49
- AddDescription (`httpd.conf`), 34, 42–43
- AddEncoding (`httpd.conf`), 34, 47
- AddHandler (`httpd.conf`), 47, 53, 57
- AddIcon (`httpd.conf`), 34, 42
- AddIconByEncoding (`httpd.conf`), 42
- AddIconByType (`httpd.conf`), 42
- AddLanguage (`httpd.conf`), 34, 48
- AddType (`httpd.conf`), 53
- adduser, 224
- AES, 113
- Age (HTTP-Kopfzeile), 7
- aktives FTP, **206**
- Alias (`httpd.conf`), 34, 50–51, 56
- AliasMatch (`httpd.conf`), 50–52, 56, 91
- Allow (HTTP-Kopfzeile), 7
- Allow (`httpd.conf`), 34, 60–62
- AllowOverride (`httpd.conf`), 34–35, 64, 104
- analog, 78
- Andreessen, Marc, 2
- anonymes FTP, **208**
- Apache, **12**
- apache
  - `-f <Datei>` (Option), 27
  - `-t` (Option), 27
- Apache Group, **16**
- Apache Software Foundation*, **16**
- Apache Software License*, **17**
- Apache-Modul
  - `mod_actions`, 20
  - `mod_alias`, 20, 50–51
  - `mod_asis`, 20
  - `mod_auth_basic`, 20, 254
  - `mod_auth_digest`, 67
  - `mod_auth_mysql`, 68
  - `mod_authn_dbd`, 65, 68
  - `mod_authn_dbm`, 65
  - `mod_authn_file`, 65
  - `mod_authn_socache`, 65
  - `mod_authnz_ldap`, 65
  - `mod_authz_core`, 146
  - `mod_authz_host`, 20
  - `mod_authz_user`, 65
  - `mod_autoindex`, 20, 39
  - `mod_cern_meta`, 20
  - `mod_cgi`, 20
  - `mod_dir`, 20, 39
  - `mod_env`, 19
  - `mod_expires`, 20
  - `mod_headers`, 20, 162
  - `mod_imagemap`, 20
  - `mod_include`, 20
  - `mod_info`, 20
  - `mod_log_config`, 20
  - `mod_log_forensic`, 20
  - `mod_mime`, 20
  - `mod_mime_magic`, 18, 20
  - `mod_negotiation`, 46
  - `mod_proxy`, 20
  - `mod_rewrite`, 20, 50, 52, 160
  - `mod_setenvif`, 19, 61–62, 260
  - `mod_so`, 20, 28
  - `mod_socache_dc`, 147
  - `mod_socache_shmcb`, 147
  - `mod_speling`, 20
  - `mod_ssl`, 107, 115–116, 133–135, 137, 144, 146, 148, 153–154, 156,

256  
 mod_status, 20, 22  
 mod_unique_id, 19  
 mod_userdir, 20, 44  
 mod_usertrack, 20  
 mod_vhost_alias, 92  
 apache2.conf, 18  
 apache2ctl, 21, 27  
 apachectl, 21–23, 27, 77  
 APPE, 209  
 apt-get, 220  
 aptitude, 220  
 ASCII, 208  
 AUTH TLS, 211  
 auth_param (squid.conf), 195  
 AuthBasicAuthoritative (httpd.conf), 65  
 AuthBasicProvider (httpd.conf), 65  
 AuthDigestProvider (httpd.conf), 67  
 Authentication-Info (HTTP-Kopfzeile), 7  
**Authentisierung, 60**  
 Authentisierungsbereich, 63  
 AuthGroupFile (httpd.conf), 34, 65  
 AuthName (httpd.conf), 34, 63–64  
 Authorization (HTTP-Kopfzeile), 7  
 AuthType (httpd.conf), 34, 64, 67  
 AuthUserFile (httpd.conf), 34, 64  
**Autorisierung, 60**  
 awk, 32, 235  
 Barrett, Bradford L., 78  
 Behlendorf, Brian, 16  
 Berners-Lee, Tim, 2, 5  
 /bin/sh, 54  
 Blockbegrenzungs-Direktiven, 32  
 Bush, Vannevar, 2  
 c_rehash, 154  
**CA, 123**  
 cache (squid.conf), 191  
 Cache-Control (HTTP-Kopfzeile), 7  
 cache_access_log (squid.conf), 180, 187  
 cache_dir (squid.conf), 181, 184, 248  
 cache_effective_group (squid.conf), 181  
 cache_effective_user (squid.conf), 180  
 cache_log (squid.conf), 180, 182–183  
 cache_mem (squid.conf), 184–185  
 cache_peer (squid.conf), 186  
 cache_peer_access (squid.conf), 192  
 cache_peer_domain (squid.conf), 187  
 cache_replacement_policy (squid.conf), 184  
 cache_store_log (squid.conf), 180  
 cache_swap_high (squid.conf), 184  
 cache_swap_low (squid.conf), 184  
 cachemgr.cgi, 188  
 cakey.pem, 129  
 calamaris, 187  
   -a (Option), 187  
   -F html (Option), 187  
 Canvas, 8  
 cd, 203, 209  
 CDUP, 209  
 chmod, 53, 203  
 chown, 225  
 Common Gateway Interface, 56  
 Connection (HTTP-Kopfzeile), 7  
 Content-Encoding (HTTP-Kopfzeile), 7  
 Content-Language (HTTP-Kopfzeile), 7  
 Content-Length (HTTP-Kopfzeile), 7  
 Content-Location (HTTP-Kopfzeile), 7  
 Content-MD5 (HTTP-Kopfzeile), 7  
 Content-Range (HTTP-Kopfzeile), 7  
 Content-Security-Policy (HTTP-Kopfzeile), 167  
 Content-Security-Policy-Report-Only (HTTP-Kopfzeile), 167  
 Content-Type (HTTP-Kopfzeile), 7  
 Cookies, 8  
 CRL, 110  
 cron, 81  
 CSRs, 125  
 CSS, 9  
 curl, 205–206  
 CustomLog (httpd.conf), 72–73, 75, 153  
 CWD, 209  
 Daemon, 21  
 Darstellungen, 45  
 Date (HTTP-Kopfzeile), 7  
 Dateistruktur, 207  
 Daten-Kanal, 206  
 dead_peer_timeout (squid.conf), 187  
 debug_options (squid.conf), 183  
 default_bits (openssl.cnf), 128  
 default_ca (openssl.cnf), 126  
 default_crl_days (openssl.cnf), 126  
 default_days (openssl.cnf), 126  
 default_keyfile (openssl.cnf), 128  
 default_md (openssl.cnf), 126, 128  
 DefaultIcon (httpd.conf), 42  
 DefaultLanguage (httpd.conf), 49  
 DefaultType (httpd.conf), 34  
 Definitionen, xi  
 delaycompress (logrotate), 77  
 DELE, 209  
 demilitarisierte Zone, 96  
 Deny (httpd.conf), 34, 60–62, 97  
 /dev/random, 112, 116, 137  
 /dev/urandom, 112, 116, 137  
 df, 226  
 Digest-Authentisierung, 67  
 digest_pw_auth, 196–197  
 Digia, 220  
 Dimension, 46  
 DirectoryIndex (httpd.conf), 33, 38–39  
 DirectorySlash (httpd.conf), 39  
 Disraeli, Benjamin, 78  
 distinguished_name (openssl.cnf), 128

dns_testnames (squid.conf), 182  
DocumentRoot (httpd.conf), 31, 35, 38, 43, 50, 63, 81, 87, 92, 97–98  
DTD, 10  
egd, 137  
Empfehlungen, 13  
emulate_httpd_log on (squid.conf), 187  
Engelbart, Doug, 2  
Engelschall, Ralf S., 115, 134  
Entropie, 112  
error_log, 21, 72, 75–76, 103  
ErrorDocument (httpd.conf), 34, 57–58  
ErrorLog (httpd.conf), 75  
ETag (HTTP-Kopfzeile), 7  
/etc/aliases, 242  
/etc/apache2/apache2.conf, 26  
/etc/apache2/httpd.conf, 26  
/etc/ftpsusers, 227, 249  
/etc/hosts, 150  
/etc/inetd.conf, 221  
/etc/init.d/apache, 22  
/etc/init.d/skeleton, 22  
/etc/init.d/squid, 184  
/etc/motd, 35  
/etc/openssl/openssl.cnf, 128  
/etc/pam.d/pure-ftp, 226  
/etc/passwd, 31, 63–64, 226, 231, 249  
/etc/pure-ftp, 236  
/etc/pure-ftp/pure-ftp.conf, 221  
/etc/pureftpd.passwd, 231  
/etc/pureftpd.pdb, 231  
/etc/shadow, 63–64, 196, 232  
/etc/shells, 227  
/etc/squid/passwd, 196  
/etc/sysconfig/apache, 103  
/etc/sysconfig/apache2, 18, 140  
/etc/webalizer.conf, 80–81  
/etc/xinetd.conf, 222  
/etc/xinetd.d/pure-ftp, 222  
Expires (HTTP-Kopfzeile), 7, 20  
external_acl_type (squid.conf), 190  
externe Bedrohung, 94  
FancyIndexing (httpd.conf), 34, 40–42  
Fehlermeldung, 57  
Firewall, 96  
Firewall-Regeln, 96  
Folgenlosigkeit, 136  
FOP, 17  
fortune, 234  
frei verfügbar, 17  
Freie Software, 114  
From (HTTP-Kopfzeile), 7  
ftp, 202–203, 205, 212  
FXP, 207  
gftp, 202  
Goldberg, Ian, 112  
GPL, 78  
grep, 32  
Group (httpd.conf), 30, 86  
groupadd, 224  
grub, 196  
Gruppendatei, 65  
Gulbrandsen, Arnt, 220  
gzip, 77  
HeaderName (httpd.conf), 43  
HELP, 208–209  
/home/, 215  
Host (HTTP-Kopfzeile), 7, 21, 29, 86, 88–89, 92, 149–150  
HostNameLookups (httpd.conf), 72, 104  
HostnameLookups (httpd.conf), 153  
HSTS, 161  
.htaccess, 32, 34–35, 45, 53, 57–58, 60, 62, 64, 97, 104, 144, 146, 243, 253–254  
htdigest, 67  
HTML: Markup-Sprache, 9  
htpasswd, 63–64, 68, 243  
-c (Option), 63–64, 243  
HTTP, 5  
HTTP 1.1, 21  
HTTP-Kopfzeile  
    Accept, 46  
    Accept-Charset, 7, 46  
    Accept-Encoding, 7, 46  
    Accept-Language, 7, 46, 49  
    Age, 7  
    Allow, 7  
    Authentication-Info, 7  
    Authorization, 7  
    Cache-Control, 7  
    Connection, 7  
    Content-Encoding, 7  
    Content-Language, 7  
    Content-Length, 7  
    Content-Location, 7  
    Content-MD5, 7  
    Content-Range, 7  
    Content-Security-Policy, 167  
    Content-Security-Policy-Report-Only, 167  
    Content-Type, 7  
    Date, 7  
    ETag, 7  
    Expires, 7, 20  
    From, 7  
    Host, 7, 21, 29, 86, 88–89, 92, 149–150  
    If-Match, 7  
    If-Modified-Since, 7  
    If-None-Match, 7  
    If-Range, 7  
    Last-Modified, 7  
    Location, 7

Max-Forwarders, 7  
Pragma, 7  
Proxy-Authenticate, 7  
Proxy-Authenticate-Info, 7  
Proxy-Authorization, 7  
Public-Key-Pins, 164–165  
Public-Key-Pins-Report-Only, 165  
Range, 7  
Referer, 7, 73, 80, 244  
Retry-After, 7  
Server, 7, 30, 257  
Strict-Transport-Security, 161–162  
Transfer-Encoding, 7  
Upgrade, 7  
User-Agent, 7, 73, 75, 80  
Vary, 7, 156  
Via, 7  
Warning, 7  
WWW-Authenticate, 7  
HTTP-Kopfzeile, allgemeine, 7  
HTTP-Kopfzeile, Anfrage-, 7  
HTTP-Kopfzeile, Antwort-, 7  
HTTP-Kopfzeile, Objekt-, 7  
HTTP-Request, 5  
HTTP-Response, 5  
http_access (squid.conf), 189, 191, 198  
HTTP_HOST (Umgebungsvariable), 29  
http_port (squid.conf), 181  
http_reply_access (squid.conf), 191, 198  
httpd-ssl.conf, 115, 134  
httpd.conf, 18, 23, 25–27, 35, 53, 57, 64, 86, 91, 103, 115, 134, 241, 243  
AccessFileName, 35  
Action, 34  
AddAlt, 42  
AddAltByEncoding, 42  
AddAltByType, 42  
AddCharset, 49  
AddDescription, 34, 42–43  
AddEncoding, 34, 47  
AddHandler, 47, 53, 57  
AddIcon, 34, 42  
AddIconByEncoding, 42  
AddIconByType, 42  
AddLanguage, 34, 48  
AddType, 53  
Alias, 34, 50–51, 56  
AliasMatch, 50–52, 56, 91  
Allow, 34, 60–62  
AllowOverride, 34–35, 64, 104  
AuthBasicAuthoritative, 65  
AuthBasicProvider, 65  
AuthDigestProvider, 67  
AuthGroupFile, 34, 65  
AuthName, 34, 63–64  
AuthType, 34, 64, 67  
AuthUserFile, 34, 64  
CustomLog, 72–73, 75, 153  
DefaultIcon, 42  
DefaultLanguage, 49  
DefaultType, 34  
Deny, 34, 60–62, 97  
DirectoryIndex, 33, 38–39  
DirectorySlash, 39  
DocumentRoot, 31, 35, 38, 43, 50, 63, 81, 87, 92, 97–98  
ErrorDocument, 34, 57–58  
ErrorLog, 75  
FancyIndexing, 34, 40–42  
Group, 30, 86  
HeaderName, 43  
HostNameLookups, 72, 104  
HostnameLookups, 153  
IdentityCheck, 72  
Include, 26–27, 91, 115, 134  
IncludeOptional, 27  
IndexIgnore, 43  
IndexOptions, 40–41  
IndexOrderDefault, 41  
IndexStyleSheet, 43  
KeepAlive, 27, 103  
KeepAliveTimeout, 27, 103  
Listen, 28, 86, 88  
ListenBacklog, 103  
LoadFile, 28  
LoadModule, 28  
LogFormat, 72–73, 75  
LogLevel, 75–76  
MaxClients, 103  
MaxKeepAliveRequests, 27  
MaxRequestsPerChild, 103  
MaxSpareServers, 103  
MinSpareServers, 102–103  
NameVirtualHost, 87–88  
Options, 33–34, 41, 144  
Order, 34, 60–62  
PidFile, 27  
ReadmeName, 43  
Redirect, 34, 51–52  
RedirectMatch, 52  
RedirectPermanent, 52  
RedirectTemp, 52  
Require, 65–68, 156, 160  
RewriteCond, 160  
RewriteEngine, 34  
RewriteRule, 160  
Satisfy, 68, 146  
ScoreBoardFile, 27  
ScriptAlias, 50–51, 56, 92, 98, 144  
ScriptAliasMatch, 50, 56  
ServerAdmin, 30, 87  
ServerAlias, 88, 150  
ServerName, 29, 87–88, 92  
ServerRoot, 27, 31, 49, 73, 86, 91, 97  
ServerSignature, 30  
ServerTokens, 30

SetEnvIf, 34, 61–62  
SSLCACertificateFile, 140, 154  
SSLCACertificatePath, 140, 154  
SSLCertificateChainFile, 139–141  
SSLCertificateFile, 139–141, 150  
SSLCertificateKeyFile, 139–141, 150  
SSLCipherSuite, 116, 135  
SSLCompression, 151–152  
SSLEngine, 115, 134  
SSLHonorCipherOrder on, 136  
SSLOptions, 144, 155  
SSLPassPhraseDialog, 141  
SSLProtocol, 116, 135, 152  
SSLRandomSeed, 116, 137–138  
SSLRequire, 153, 156, 160–161  
SSLRequireSSL, 146, 153, 160  
SSLSessionCache, 147  
SSLSessionCacheTimeout, 147  
SSLSessionTicketKeyFile, 148–149  
SSLSessionTickets, 148  
SSLStrictSNIVHostCheck, 150  
SSLUserName, 160  
SSLVerifyClient, 155  
SSLVerifyDepth, 155  
StartServers, 102–103  
TimeOut, 27, 105  
TransferLog, 75  
UseCanonicalName, 29, 73, 104  
UseCanonicalPhysicalPort, 29  
User, 30, 86, 97  
UserDir, 44, 50, 97  
XBitHack, 34, 53, 97  
**HTTPS**, 108, 111, 115  
`https_port (squid.conf)`, 182  
Hudson, Tim J., 114  
Hypertext Transfer Protocol, 5  
  
Icons, 42  
ICP, 186  
identd, 194–195  
`IdentityCheck (httpd.conf)`, 72  
`If-Match (HTTP-Kopfzeile)`, 7  
`If-Modified-Since (HTTP-Kopfzeile)`, 7  
`If-None-Match (HTTP-Kopfzeile)`, 7  
`If-Range (HTTP-Kopfzeile)`, 7  
`Include (httpd.conf)`, 26–27, 91, 115, 134  
`IncludeOptional (httpd.conf)`, 27  
`index.htm`, 38  
`index.html`, 20, 33, 38  
`index.txt`, 126  
`IndexIgnore (httpd.conf)`, 43  
`IndexOptions (httpd.conf)`, 40–41  
`IndexOrderDefault (httpd.conf)`, 41  
`IndexStyleSheet (httpd.conf)`, 43  
inetd, 216, 221–222  
Inhaltsaushandlung, 45  
Init-Skript, 22  
insserv, 22  
  
International Organization for Standardization, 13  
interne Bedrohung, 94  
iptables, 239  
ISO 3166, 48  
  
Jakarta, 17  
Java Server Pages, 17  
Java-Servlets, 17  
JavaScript, 8  
  
KeepAlive (`httpd.conf`), 27, 103  
KeepAliveTimeout (`httpd.conf`), 27, 103  
Kennwortdatei, 63  
kill, 183  
konqueror, 173  
Kontroll-Kanal, 206  
Kopfzeilen, 5  
  
Ländercodes, 48  
Last-Modified (HTTP-Kopfzeile), 7  
LIST, 209  
Listen (`httpd.conf`), 28, 86, 88  
ListenBacklog (`httpd.conf`), 103  
LoadFile (`httpd.conf`), 28  
LoadModule (`httpd.conf`), 28  
Location (HTTP-Kopfzeile), 7  
LogFormat (`httpd.conf`), 72–73, 75  
logger, 249  
logische Auszeichnung, 9  
LogLevel (`httpd.conf`), 75–76  
logout, 209  
logresolve, 72  
logrotate  
    delaycompress, 77  
logrotate, 76–77, 183, 235  
lokale Kommandos, 208  
ls, 203, 209  
lynx, 202  
  
magic, 18  
make, 154, 180  
*Man-in-the-middle*-Angriff, 109, 174  
MathML, 11  
Max-Forwarders (HTTP-Kopfzeile), 7  
MaxClients (`httpd.conf`), 103  
`maximum_object_size (squid.conf)`, 184  
MaxKeepAliveRequests (`httpd.conf`), 27  
MaxRequestsPerChild (`httpd.conf`), 103  
MaxSpareServers (`httpd.conf`), 103  
May, Karl, 16  
McCool, Rob, 16  
MD5, 128  
`memory_replacement_policy (squid.conf)`, 184  
Mewburn, Luke, 202  
`mime.types`, 49  
`MinSpareServers (httpd.conf)`, 102–103  
MKD, 209

**mkdir**, 203, 209  
**mkpasswd**, 196  
**mod_actions** (Apache-Modul), 20  
**mod_alias** (Apache-Modul), 20, 50–51  
**mod_asis** (Apache-Modul), 20  
**mod_auth_basic** (Apache-Modul), 20, 254  
**mod_auth_digest** (Apache-Modul), 67  
**mod_auth_mysql** (Apache-Modul), 68  
**mod_authn_dbd** (Apache-Modul), 65, 68  
**mod_authn_dbm** (Apache-Modul), 65  
**mod_authn_file** (Apache-Modul), 65  
**mod_authn_socache** (Apache-Modul), 65  
**mod_authnz_ldap** (Apache-Modul), 65  
**mod_authz_core** (Apache-Modul), 146  
**mod_authz_host** (Apache-Modul), 20  
**mod_authz_user** (Apache-Modul), 65  
**mod_autoindex** (Apache-Modul), 20, 39  
**mod_cern_meta** (Apache-Modul), 20  
**mod_cgi** (Apache-Modul), 20  
**mod_dir** (Apache-Modul), 20, 39  
**mod_env** (Apache-Modul), 19  
**mod_expires** (Apache-Modul), 20  
**mod_headers** (Apache-Modul), 20, 162  
**mod_imagemap** (Apache-Modul), 20  
**mod_include** (Apache-Modul), 20  
**mod_info** (Apache-Modul), 20  
**mod_log_config** (Apache-Modul), 20  
**mod_log_forensic** (Apache-Modul), 20  
**mod_mime** (Apache-Modul), 20  
**mod_mime_magic** (Apache-Modul), 18, 20  
**mod_negotiation** (Apache-Modul), 46  
**mod_proxy** (Apache-Modul), 20  
**mod_rewrite** (Apache-Modul), 20, 50, 52, 160  
**mod_setenvif** (Apache-Modul), 19, 61–62, 260  
**mod_so** (Apache-Modul), 20, 28  
**mod_socache_dc** (Apache-Modul), 147  
**mod_socache_shmcb** (Apache-Modul), 147  
**mod_speling** (Apache-Modul), 20  
**mod_ssl** (Apache-Modul), 107, 115–116, 133–135, 137, 144, 146, 148, 153–154, 156, 256  
**mod_status** (Apache-Modul), 20, 22  
**mod_unique_id** (Apache-Modul), 19  
**mod_userdir** (Apache-Modul), 20, 44  
**mod_usertrack** (Apache-Modul), 20  
**mod_vhost_alias** (Apache-Modul), 92  
**MODE**, 207  
**Module**, 19  
Moore, Brian, 52  
**more**, 205  
**MPM** (Apache), 27  
**Multiviews**, 46  
**mv**, 203  
Name, ausgezeichneter, 122  
Namensraumkonzept, 11  
**NameVirtualHost** (httpd.conf), 87–88  
**nc**, 175, 205  
**NCSA-httdp**, 16  
**ncsa_auth**, 196–197  
**neighbor_type_domain** (squid.conf), 187  
Nelson, Ted, 2  
**Netbooks**, 12  
**netcat**, 8, 175, 205–206, 211–212, 248  
**netcat -h**, 175  
**Netscape**, 108  
**netstat**, 22, 183, 195, 222, 248  
**NLST**, 209  
**nmap**, 95, 239  
**Nokia**, 220  
**NSA**, 136  
**OpenSSL**, 114  
    **req**  
        **-days** (Option), 129  
**openssl**, 126, 128, 197  
    **ca** (Option), 126  
    **req** (Option), 128  
**openssl s_client**  
    **-reconnect** (Option), 148  
**openssl.cnf**  
    **default_bits**, 128  
    **default_ca**, 126  
    **default_crl_days**, 126  
    **default_days**, 126  
    **default_keyfile**, 128  
    **default_md**, 126, 128  
    **distinguished_name**, 128  
    **policy**, 128  
    **prompt**, 128  
    **x509_extensions**, 128  
**openssl.tar.gz**, 114  
**OPENSSL_CONF** (Umgebungsvariable), 128  
**Options** (httpd.conf), 33–34, 41, 144  
**Order** (httpd.conf), 34, 60–62  
  
**PASS**, 208  
**passives** FTP, 207  
**passwd**, 196  
**PASV**, 207  
Pellow, Nicola, 2  
**PGP**, 120  
**PidFile** (httpd.conf), 27  
**PKCS#12**, 156  
**PKI**, 120  
**policy** (openssl.cnf), 128  
**PORT**, 207  
**Pragma** (HTTP-Kopfzeile), 7  
**prompt** (openssl.cnf), 128  
**Proxy-Authenticate** (HTTP-Kopfzeile), 7  
**Proxy-Authenticate-Info** (HTTP-Kopfzeile), 7  
**Proxy-Authorization** (HTTP-Kopfzeile), 7  
**Proxy-Server**, 78  
**ps**, 183, 222  
**public-key infrastructure**, 120

Public-Key-Pins (HTTP-Kopfzeile), 164–165  
Public-Key-Pins-Report-Only (HTTP-Kopfzeile), 165  
public_html, 43  
pure-config.pl, 221  
Pure-FTPd, 95  
pure-ftpd, 220  
-0 (Option), 228  
-A (Option), 227–228, 249  
-a (Option), 227–229, 249  
--allowdotfiles (Option), 224  
--anonymouscantupload (Option), 224, 236  
--anonymousonly (Option), 224  
--antiwarez (Option), 225  
--autorename (Option), 238  
--bind (Option), 221  
-C (Option), 238  
-c (Option), 238–239  
--chrooteveryone (Option), 227  
--createhome (Option), 232  
--customerproof (Option), 229  
--dontresolve (Option), 235  
-E (Option), 230, 249  
-e (Option), 224  
-F (Option), 234  
-f (Option), 235  
--fortunesfile (Option), 234  
-G (Option), 238  
-H (Option), 235  
-i (Option), 224, 236  
-j (Option), 232  
-K (Option), 238  
-k (Option), 226  
--keepallfiles (Option), 238  
-L (Option), 238  
-l (Option), 226, 232–233, 249  
--limitrecursion (Option), 238  
-m (Option), 238  
--maxclientsnumber (Option), 238–239  
--maxclientsperip (Option), 238  
--maxdiskusagepct (Option), 226  
--maxload (Option), 238  
--minuid (Option), 237  
-n (Option), 229  
--noanonymous (Option), 249  
--nochmod (Option), 238  
--norename (Option), 238  
--nottruncate (Option), 228  
-p (Option), 239  
--passiveportrange (Option), 239  
--prohibitdotfilesread (Option), 229  
--prohibitdotfileswrite (Option), 229  
--quotas (Option), 229  
-R (Option), 238  
-r (Option), 238  
-S (Option), 220–221  
-s (Option), 225–226  
--syslogfacility (Option), 235  
--tls (Option), 240  
--trustedip (Option), 236  
-U (Option), 228  
-u (Option), 237–238  
--umask (Option), 228  
-V (Option), 236, 249  
-x (Option), 229  
-x (Option), 229  
-Y (Option), 240  
-Z (Option), 229  
-z (Option), 224  
pure-ftpd-wrapper, 221  
pure-ftpwho, 222  
pure-pw, 231–232  
-D (Option), 232  
-d (Option), 232  
-f (Option), 231  
-m (Option), 232  
pure-quotacheck, 229–230  
pure-statsdecode, 235  
PURE_DBFILE (Umgebungsvariable), 231  
PURE_PASSWDFILE (Umgebungsvariable), 231  
PWD, 209  
pwd, 203, 209  
QUIT, 209  
Range (HTTP-Kopfzeile), 7  
RC4, 113  
ReadmeName (httpd.conf), 43  
Redirect (httpd.conf), 34, 51–52  
RedirectMatch (httpd.conf), 52  
redirector_access (squid.conf), 191  
RedirectPermanent (httpd.conf), 52  
RedirectTemp (httpd.conf), 52  
Referer (HTTP-Kopfzeile), 7, 73, 80, 244  
REMOTE_USER (Umgebungsvariable), 160  
Require (httpd.conf), 65–68, 156, 160  
RETR, 208–209  
Retry-After (HTTP-Kopfzeile), 7  
Reverse Proxy, 105  
RewriteCond (httpd.conf), 160  
RewriteEngine (httpd.conf), 34  
RewriteRule (httpd.conf), 160  
rinetd, 172  
Risikoanalyse, 95  
rm, 203, 209  
RMD, 209  
rmdir, 203, 209  
RNFR, 208–209  
RNT0, 209  
RNTP, 208  
rsync, 95

Satisfy (httpd.conf), 68, 146  
 ScoreBoardFile (httpd.conf), 27  
 scp, 95, 211  
 ScriptAlias (httpd.conf), 50–51, 56, 92, 98, 144  
 ScriptAliasMatch (httpd.conf), 50, 56  
 Secure Socket Layer, 68  
 serial, 126  
 Server (HTTP-Kopfzeile), 7, 30, 257  
 Server Name Indication, 150  
*server-side includes*, 52  
 ServerAdmin (httpd.conf), 30, 87  
 ServerAlias (httpd.conf), 88, 150  
 ServerName (httpd.conf), 29, 87–88, 92  
 ServerRoot (httpd.conf), 27, 31, 49, 73, 86, 91, 97  
 ServerSignature (httpd.conf), 30  
 ServerTokens (httpd.conf), 30  
 SetEnvIf (httpd.conf), 34, 61–62  
 sftp, 211  
 SGML, 10  
 sha1, 128  
 SIGHUP, 21  
 SIGUSR1, 21  
 SITE, 208–209  
 SITE HELP, 208  
 Skolnick, Cliff, 16  
 SNI, 150  
 split-logfile, 91  
 Sprachencodes, 48  
 sqlsecret, 68  
 squid, 105
 

- D (Option), 182
- d (Option), 182
- k (Option), 182–183
- N (Option), 182

 squid.conf, 180
 

- acl, 190
- auth_param, 195
- cache, 191
- cache_access_log, 180, 187
- cache_dir, 181, 184, 248
- cache_effective_group, 181
- cache_effective_user, 180
- cache_log, 180, 182–183
- cache_mem, 184–185
- cache_peer, 186
- cache_peer_access, 192
- cache_peer_domain, 187
- cache_replacement_policy, 184
- cache_store_log, 180
- cache_swap_high, 184
- cache_swap_low, 184
- dead_peer_timeout, 187
- debug_options, 183
- dns_testnames, 182
- emulate_httpd_log on, 187
- external_acl_type, 190

 http_access, 189, 191, 198  
 http_port, 181  
 http_reply_access, 191, 198  
 https_port, 182  
 maximum_object_size, 184  
 memory_replacement_policy, 184  
 neighbor_type_domain, 187  
 redirector_access, 191  
 squid.conf.default, 182  
 /srv/ftp, 224–225  
 /srv/ftp/, 214  
 /srv/www/htdocs, 31  
 ssh, 95  
 SSL, 60, 108, 110–111
 

- Probleme, 110

 SSLeay, 114  
 SSL_CLIENT_CERT (Umgebungsvariable), 144  
 SSL_CLIENT_CERT_ (Umgebungsvariable), 144  
 SSL_SECURE_RENEG (Umgebungsvariable), 151  
 SSL_SERVER_CERT (Umgebungsvariable), 144  
 SSL_SERVER_S_DN (Umgebungsvariable), 146  
 SSL_SESSION_RESUMED (Umgebungsvariable), 153  
 SSLCACertificateFile (httpd.conf), 140, 154  
 SSLCACertificatePath (httpd.conf), 140, 154  
 SSLCertificateChainFile (httpd.conf), 139–141  
 SSLCertificateFile (httpd.conf), 139–141, 150  
 SSLCertificateKeyFile (httpd.conf), 139–141, 150  
 SSLCipherSuite (httpd.conf), 116, 135  
 SSLCompression (httpd.conf), 151–152  
 SSLEngine (httpd.conf), 115, 134  
 SSLHonorCipherOrder on (httpd.conf), 136  
 SSLOptions (httpd.conf), 144, 155  
 SSLPassPhraseDialog (httpd.conf), 141  
 SSLProtocol (httpd.conf), 116, 135, 152  
 SSLRandomSeed (httpd.conf), 116, 137–138  
 SSLRequire (httpd.conf), 153, 156, 160–161  
 SSLRequireSSL (httpd.conf), 146, 153, 160  
 SSLSessionCache (httpd.conf), 147  
 SSLSessionCacheTimeout (httpd.conf), 147  
 SSLSessionTicketKeyFile (httpd.conf), 148–149  
 SSLSessionTickets (httpd.conf), 148  
 SSLStrictSNIVHostCheck (httpd.conf), 150  
 SSLUserName (httpd.conf), 160  
 SSLVerifyClient (httpd.conf), 155  
 SSLVerifyDepth (httpd.conf), 155

- StartServers (`httpd.conf`), 102–103  
STARTTLS, 211  
STOR, 208–209  
Strict-Transport-Security (HTTP-Kopfzeile), 161–162  
STRU, 207  
suEXEC, 98  
SuSEconfig, 103  
SVG, 11  
syslogd, 234  
SYST, 209  
System-V-Init, 22  
  
tail, 73  
  -f (Option), 73  
tcpd, 216  
TELNET, 206  
telnet, 8, 49, 195, 205–206  
Thau, Robert, 16  
TimeOut (`httpd.conf`), 27, 105  
TLS, 108, 110  
TLS Session Tickets, 148  
tnftp, 202, 205  
Tomcat, 17  
Transfer-Encoding (HTTP-Kopfzeile), 7  
TransferLog (`httpd.conf`), 75  
Trolltech, 220  
Turner, Stephen, 81  
TYPE, 208  
Typemap-Dateien, 46  
  
Überprüfung, 95  
Übertragungsmodus, 207  
Umgebungsvariable  
  HTTP_HOST, 29  
  OPENSSL_CONF, 128  
  PURE_DBFILE, 231  
  PURE_PASSWDFILE, 231  
  REMOTE_USER, 160  
  SSL_CLIENT_CERT, 144  
  SSL_CLIENT_CERT_, 144  
  SSL_SECURE_RENEG, 151  
  SSL_SERVER_CERT, 144  
  SSL_SERVER_S_DN, 146  
  SSL_SESSION_RESUMED, 153  
uname, 209  
Uniform Resource Locator, 4  
Uniform Resource Name, 4  
update-rc.d, 22  
Upgrade (HTTP-Kopfzeile), 7  
URI, absolute, 4  
URI, relative, 4  
URI-Schema, 4  
URL, 4  
URN, 4  
UseCanonicalName (`httpd.conf`), 29, 73, 104  
UseCanonicalPhysicalPort (`httpd.conf`), 29  
USER, 208  
User (`httpd.conf`), 30, 86, 97  
  
User-Agent (HTTP-Kopfzeile), 7, 73, 75,  
  80  
useradd, 224  
UserDir (`httpd.conf`), 44, 50, 97  
/usr/local/apache/cgi-bin/test.pl, 56  
/usr/local/apache/conf, 18  
/usr/local/apache/conf/httpd.conf, 26  
/usr/local/apache/htdocs, 31  
/usr/local/httpd/htdocs, 31  
/usr/local/ssl, 115  
  
Validierung, 10  
van Dam, Andy, 2  
/var/lib/ftpd/, 214  
/var/log/apache/access_log, 81  
/var/log/apache2, 21  
/var/log/httpd, 21  
/var/www, 31  
Variante, 46  
Vary (HTTP-Kopfzeile), 7, 156  
verhandelbar, 46  
Vertrauengrenze, 96  
Vertrauensnetz, 120  
Via (HTTP-Kopfzeile), 7  
vim, 180  
virtuelle Web-Server, 21  
visuelle Auszeichnung, 9  
vsftpd, 214–215, 217–218  
vsftpd.conf, 214  
  
W3C, 12  
w3m, 173, 202  
Wagner, David, 112  
Warning (HTTP-Kopfzeile), 7  
WebDAV, 211  
Weiterleitung, 51  
wget, 114, 173, 183, 205–207  
who, 217  
Winnetou, 16  
WWW-Authenticate (HTTP-Kopfzeile), 7  
  
X.509, 121  
X.509-Erweiterungen, 122  
x509_extensions (`openssl.cnf`), 128  
Xalan, 17  
XBitHack (`httpd.conf`), 34, 53, 97  
XHTML, 11  
xinetd, 214, 216–218, 221–222  
XML, 10  
XSS, 166  
  
you, 173  
Young, Eric A., 114  
  
Zeichenkodierung, 208  
Zertifikat, 120  
Zertifikats-Rückruflisten, 110  
Zertifizierung, 120  
Zertifizierungsanfragen, 125

Zertifizierungsstellen, **120**  
Zielvorgabe, **94**  
Zugriffskontrolle, **60**