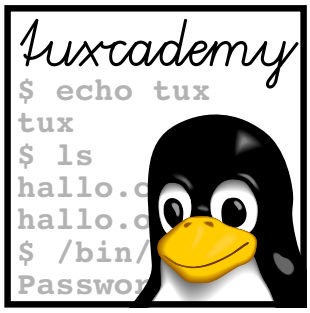
Linux-Netzadministration

Linux im LAN und Internet



tuxcademy – Linux- und Open-Source-Lernunterlagen für alle
www.tuxcademy.org · info@tuxcademy.org

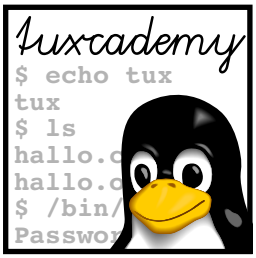
Das tuxcademy-Projekt bietet hochwertige frei verfügbare Schulungsunterlagen zu Linux- und Open-Source-Themen – zum Selbststudium, für Schule, Hochschule, Weiterbildung und Beruf.
Besuchen Sie https://www.tuxcademy.org/! Für Fragen und Anregungen stehen wir Ihnen gerne zur Verfügung.

Linux-Netzadministration Linux im LAN und Internet

Revision: nadm:3087b9b3d75e9857:2014-04-03
dnsx:72672857de912950:2012-10-15 5–11
nadm:34ccb7a5ca5eb94a:2014-04-03 1–4, B
nadm:DfNbjwihcLSshYGYreWGiU

http://www.tuxcademy.org · info@tuxcademy.org

Autor: Anselm Lingnau
Technische Redaktion: Anselm Lingnau ⟨*anselm.lingnau@linupfront.de*⟩
Gesetzt in Palatino, Optima und DejaVu Sans Mono



Inhalt

1 TCP/IP-Schnellüberblick **13**
1.1 Internet-Grundlagen . . . 14
1.1.1 Überblick . . . 14
1.1.2 Rechnernetze . . . 14
1.1.3 Protokolle . . . 14
1.1.4 TCP/IP . . . 15
1.1.5 Adressen . . . 17
1.1.6 Wegleitung . . . 18
1.1.7 Subnetting . . . 19
1.2 Linux-Netzkonfiguration . . . 19
1.2.1 Netzschnittstellen . . . 19
1.2.2 Wegleitung . . . 20
1.2.3 Netzkonfiguration mit `ip` . . . 22
1.2.4 Dauerhafte Netzkonfiguration . . . 23
1.2.5 DHCP . . . 25

2 Benutzer benachrichtigen **27**
2.1 Einleitung . . . 28
2.2 Vorwarnung vor dem Anmelden: `/etc/issue` und `/etc/issue.net` . . . 28
2.3 Begrüßung nach dem Anmelden: `/etc/motd` . . . 29
2.4 Alle Benutzer benachrichtigen: `wall` . . . 29
2.5 Das Ende der Welt ist nahe: `shutdown` . . . 30

3 Netzwerkdiagnose mit `tcpdump` und `wireshark` **31**
3.1 Einführung . . . 32
3.2 `tcpdump` . . . 32
3.2.1 Grundlagen . . . 32
3.2.2 Filter . . . 33
3.2.3 Ausgabe . . . 36
3.3 `wireshark` . . . 41
3.3.1 Grundlagen . . . 41
3.3.2 Netzwerkdaten protokollieren . . . 43
3.3.3 Das Hauptfenster . . . 44
3.3.4 Tipps, Tricks und Techniken . . . 48

4 Linux und WLAN **55**
4.1 Einleitung . . . 56
4.2 WLAN-Grundkonfiguration . . . 56
4.3 WLAN-Verschlüsselung . . . 58

5 DNS: Grundlagen **61**
5.1 Einführung in die Namensauflösung . . . 62
5.2 Das Domain Name System . . . 63
5.2.1 Eine kurze Geschichte . . . 63
5.2.2 Aufbau eines DNS-Namens . . . 64
5.2.3 Namensverwaltung . . . 67

5.2.4 Domains und Zonen . . . 68
5.2.5 Das Protokoll . . . 69
5.3 Linux als DNS-Client . . . 70
5.3.1 Die C-Laufzeitbibliothek . . . 70
5.3.2 Der Resolver . . . 71
5.3.3 Verhalten bei Zeitüberschreitungen . . . 72
5.4 DNS testen . . . 73
5.4.1 Einleitung . . . 73
5.4.2 Das Kommando `dig` . . . 73
5.4.3 Das Kommando `nslookup` . . . 75
5.4.4 Das Kommando `host` . . . 77

6 Der DNS-Server BIND 79
6.1 Überblick . . . 80
6.2 BIND-Grundlagen . . . 81
6.2.1 Serverprogramm und Dateien . . . 81
6.2.2 Die Konfigurationsdatei `named.conf` . . . 82
6.3 Ein *caching-only* DNS-Server . . . 84
6.4 BIND steuern mit `ndc` und `rndc` . . . 85

7 Zonendateien 91
7.1 Zonendateien und Ressourcendatensätze . . . 92
7.2 `SOA`-Records . . . 93
7.3 `NS`-Records . . . 95
7.4 `A`-Records . . . 95
7.5 `CNAME`-Records . . . 96
7.6 Rückwärts-Auflösung von Namen und `PTR`-Records . . . 97
7.7 `MX`-Records . . . 98
7.8 `SRV`-Records . . . 99
7.9 Andere Typen von Ressourcendatensätzen . . . 100

8 Primäre und sekundäre DNS-Server 103
8.1 Redundanz im DNS . . . 104
8.2 Konfiguration . . . 104
8.3 Zonentransfer: Grundlagen . . . 108
8.4 Automatische Benachrichtigung bei Änderungen . . . 109
8.5 Zonentransfer absichern mit TSIG . . . 110

9 Subdomains und Delegation 115
9.1 Einfache Subdomains . . . 116
9.2 Delegation an andere DNS-Server . . . 117
9.3 Delegation für Rückwärts-Zonen . . . 119
9.4 Stub-Zonen . . . 121

10 DNS-Tipps und -Tricks 123
10.1 Adressensuchlisten . . . 124
10.2 Weiterleitung an andere Server . . . 125
10.3 Weiterleitungs-Zonen . . . 127

11 DNS-Sicherheit: Grundlagen 129
11.1 DNS und Sicherheit . . . 130
11.2 BIND ohne `root` . . . 132
11.3 BIND und `chroot` . . . 133
11.4 Getrennte rekursive und autoritative Server . . . 135
11.4.1 Warum trennen? . . . 135
11.4.2 Umsetzung . . . 136

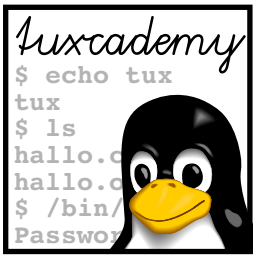
A Musterlösungen 143

B X.509-Crashkurs **151**
B.1 Einleitung: Kryptografie, Zertifikate und X.509 151
B.2 Eine Zertifizierungsstelle generieren 153
B.3 Server-Zertifikate generieren 156

C LPIC-2-Zertifizierung **159**
C.1 Überblick 159
C.2 Prüfung LPI-201 160
C.3 LPI-Prüfungsziele in dieser Schulungsunterlage 160

D Kommando-Index **165**

Index **167**



Tabellenverzeichnis

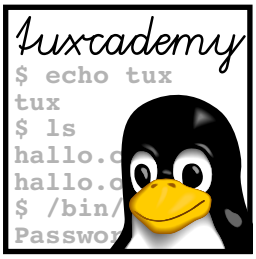
1.1 Private IP-Adressbereiche nach RFC 1918 18

3.1 Suchkriterien in `tcpdump` (Auszug) 34

5.1 Die generischen Top-Level-Domains 66
5.2 Optionen innerhalb `/etc/resolv.conf` 71

6.1 BIND und Signale 89

8.1 `SOA`-Parameter gemäß DENIC 106



Abbildungsverzeichnis

1.1 ISO/OSI-Referenzmodell 15
1.2 Die Datei `/etc/services` (Auszug) 18

3.1 Das Programm `wireshark`: Startbildschirm 42
3.2 Das Programm `wireshark`: Optionen für das Protokollieren 43
3.3 Das Programm `wireshark`: Protokollierung 45
3.4 Das Programm `wireshark`: Hauptfenster 46
3.5 Verfolgung von TCP-Verbindungen mit `wireshark` 50

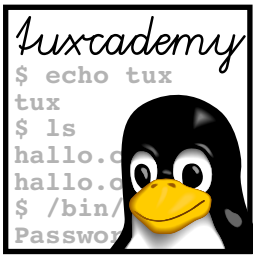
5.1 Die Datei `/etc/hosts` (SUSE) 62
5.2 Beispiel für `/etc/resolv.conf` 72

6.1 Die Datei `named.conf` (Beispiel) 82
6.2 Konfiguration für einen *caching-only* DNS-Server 84

7.1 Beispiel für eine Zonendatei 92
7.2 Beispiel für eine Rückwärts-Zonendatei 97

11.1 Abgesicherte BIND-Konfiguration 137
11.2 Getrennte DNS-Server: Der autoritative Server 138
11.3 Getrennte DNS-Server: Der rekursive Server 138
11.4 Getrennte DNS-Server: Konfiguration mit Views 140

B.1 Konfigurationsdatei für eine OpenSSL-basierte CA 155



Vorwort

Diese Schulungsunterlage vermittelt einführendes Wissen für Konfiguration und Betrieb von Linux-Arbeitsplatzrechnern und -Servern in einem (existierenden) lokalen Netz.

Sie wendet sich an fortgeschrittene Linux-Administratoren und setzt Kenntnisse voraus, wie sie etwa in der LPIC-1-Prüfung abgefragt werden. Dazu gehört eine solide Beherrschung der Shell, eines Texteditors und der grundlegenden Linux-Kommandos sowie der Grundzüge der Linux-Administration. Kenntnisse über TCP/IP und die Konfiguration von Linux-Rechnern als Clients in lokalen TCP/IP-Netzen, wie in der Linup-Front-Unterlage *Linux-Administration 2* beschrieben, werden ebenfalls vorausgesetzt, genau wie ein grundlegendes Verständnis von Shellprogrammierung, Werkzeugen wie `sed` und `awk` sowie Diensten wie `cron` und `at`.

Nach einer Einführung in die Grundlagen von TCP/IP und der Linux-Netzkonfiguration widmet diese Schulungsunterlage sich ausführlich Themen wie dem Gebrauch der Netzwerkanalysewerkzeuge `tcpdump` und `wireshark` sowie der Konfiguration von Linux als WLAN-Client. Anschließend geben wir eine gründliche Einführung in die Grundlagen des DNS und erklären Konfiguration und Betrieb des wichtigsten DNS-Servers, BIND 9.

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 numeriert 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, Kommandobeispiele 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

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

Definitionen

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

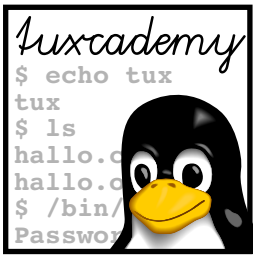
Wir sind bemüht, diese Schulungsunterlage möglichst aktuell, vollständig und fehlerfrei zu gestalten. Trotzdem kann es passieren, dass sich Probleme oder Ungenauigkeiten einschleichen. Wenn Sie etwas bemerken, was Sie für verbesserungsfähig halten, dann lassen Sie es uns wissen, etwa indem Sie eine elektronische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

LPIC-1-Zertifizierung

Diese Unterlage ist Teil eines Kurskonzepts zur Vorbereitung auf die LPIC-1-Zertifizierung. Anhang C enthält hierzu mehr Informationen.



1

TCP/IP-Schnellüberblick

Inhalt

1.1 Internet-Grundlagen 14
1.1.1 Überblick 14
1.1.2 Rechnernetze 14
1.1.3 Protokolle 14
1.1.4 TCP/IP 15
1.1.5 Adressen 17
1.1.6 Wegleitung 18
1.1.7 Subnetting 19
1.2 Linux-Netzkonfiguration 19
1.2.1 Netzschnittstellen 19
1.2.2 Wegleitung 20
1.2.3 Netzkonfiguration mit `ip` 22
1.2.4 Dauerhafte Netzkonfiguration 23
1.2.5 DHCP 25

Lernziele

- Einen schnellen Überblick über TCP/IP und Linux-Netzkonfiguration gewinnen

Vorkenntnisse

- Solide Kenntnisse der Linux-Systemadministration
- Grundlegende Kenntnisse von Rechnernetzen und TCP/IP-Diensten als Anwender sind hilfreich

Dieses Kapitel wiederholt im Wesentlichen Inhalte aus der Linup-Front-Schulungsunterlage *Linux-Administration 2* und richtet sich an diejenigen, die sie nicht schon dort gelesen haben. Wenn Sie jene Unterlage kennen und/oder ein LPIC-1-Zertifikat besitzen, können Sie getrost mit dem nächsten Kapitel fortfahren.

1.1 Internet-Grundlagen

1.1.1 Überblick

»Das Internet«, so wie wir es kennen, also eine mehr oder weniger weltumspannende Zusammenschaltung diverser lokaler und Weitverkehrs-Rechnernetze auf der Basis der Protokollfamilie TCP/IP, basiert auf Überlegungen der 1960er Jahre, wurde in den 1970er und frühen 1980er Jahren spezifiziert (mit ein paar späteren Verfeinerungen) und wuchs ab den 1990er Jahren bedingt durch das allgemeine Interesse an Diensten wie E-Mail und dem World Wide Web zu seiner heutigen Größe an (ein Ende ist nicht abzusehen).

Einige Begriffe aus dem vorstehenden Satz müssen wir näher erläutern.

1.1.2 Rechnernetze

Es gibt *lokale* und *Weitverkehrsnetze* (vulgo »LANs« und »WANs«). Hier sind ein paar der wesentlichen Unterschiede:

- LANs verbinden eine relativ kleine Anzahl von Stationen in einem räumlich begrenzten Areal, WANs eine potentiell große Anzahl von Stationen in einem räumlich sehr großen Areal.
- LANs werden in der Regel von einer einzigen Firma, Organisation oder Familie betrieben und benutzt, während WANs eine Vielzahl von Benutzern verbinden, die typischerweise *nicht* Eigentümer des WAN sind.
- LANs erlauben hohe Bandbreiten und ihre Nutzung ist im wesentlichen kostenlos, während WANs im Vergleich langsamer und ihre Benutzung kostenpflichtig sind.

LANs und WANs verwenden mitunter völlig verschiedene Netzwerktechniken – von drahtlosen Verbindungen über sehr kurze Strecken (Bluetooth) über typische LAN-Technik wie Ethernet bis zu Glasfaserverbindungen mit ATM für WANs. Allerdings verschwimmen die technischen Unterschiede zusehends; Gigabit-Ethernet (eigentlich historisch ein LAN-Protokoll) wird heute zum Beispiel von der Deutschen Telekom auch im Weitverkehr eingesetzt.

1.1.3 Protokolle

Ein »Protokoll« ist eine Vereinbarung dafür, wie zwei (oder mehr) Stationen in einem Netz sich unterhalten. Protokolle lassen sich grob in drei Klassen einteilen:

Übertragungsprotokolle (oft auch »Zugriffsverfahren« genannt) regeln die Datenübertragung grob gesagt auf der Ebene von Netzwerkkarten und Leitungen. Ihre Ausgestaltung hängt von den elektro- und nachrichtentechnischen Eigenschaften und Einschränkungen ab, die aus der Realisierung in »Hardware« folgen.

Heute gängige Zugriffsverfahren sind zum Beispiel Ethernet (für »verkabelte« Netze) oder verschiedene Geschmacksrichtungen von IEEE 802.11 für WLAN.

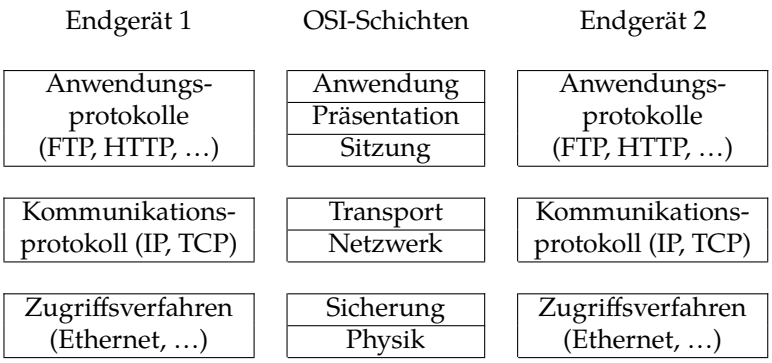


Bild 1.1: ISO/OSI-Referenzmodell

Kommunikationsprotokolle dienen dazu, die Kommunikation zwischen Rechnern in verschiedenen Netzen zu regeln, ohne dass dafür eine genaue Kenntnis der verwendeten Zugriffsverfahren notwendig ist. Sie abstrahieren von den Zugriffsverfahren und erlauben eine einheitliche Adressierung und Wegleitung.

Die Kommunikationsprotokolle, die uns interessieren, sind natürlich IP, TCP und UDP. Dazu kommt noch ICMP als »Infrastrukturprotokoll«, das zur Diagnose, Steuerung und Fehlermeldung dient.

Anwendungsprotokolle realisieren auf der Basis der Kommunikationsprotokolle tatsächliche Dienste wie elektronische Post, Dateiübertragung oder Internet-Telefonie. Kommunikationsprotokolle dienen zum Austausch von Bits und Bytes; Anwendungsprotokolle beschreiben, was diese Bits und Bytes *bedeuten* sollen.

Typische Anwendungsprotokolle, mit denen Sie als Linux-Administrator konfrontiert werden könnten, sind zum Beispiel SMTP, FTP, SSH, DNS, HTTP, POP3 oder IMAP (und ggf. deren »sichere« Ableger).

Die Daten, die über ein Protokoll ausgetauscht werden, nennen wir abstrakt Protokolldateneinheiten – je nach Protokoll können sie spezifischere Namen haben, etwa »Pakete«, »Datagramme«, »Segmente« oder »Frames«.

Protokolldateneinheiten

Aus der Tatsache, dass Kommunikationsprotokolle dafür gedacht sind, die Details der Übertragungsprotokolle zu verstecken, und Anwendungsprotokolle wiederum dazu dienen, die Details der Kommunikationsprotokolle zu verstecken, kann man ein »Schichtenmodell« konstruieren, bei dem die Übertragungsprotokolle die unterste und die Anwendungsprotokolle die oberste Ebene einnehmen. (Daher kommt auch der Begriff »Protokollstapel«.) Jede Schicht auf der Absenderseite empfängt Daten »von oben« und gibt sie »nach unten« weiter; auf der Empfängerseite werden sie »von unten« empfangen und »nach oben« weitergegeben.

Schichtenmodell

Das bekannteste Schichtenmodell ist das »ISO/OSI-Referenzmodell« (Bild 1.1), fast das einzige Überbleibsel einer sehr elaboraten Protokollfamilie, die in der Praxis völlig von TCP/IP verdrängt wurde.

ISO/OSI-Referenzmodell

1.1.4 TCP/IP

TCP/IP steht für *Transmission Control Protocol/Internet Protocol* und ist die heute am häufigsten eingesetzte Methode für den Datentransfer in Rechnernetzen, seien es nur zwei Rechner in einem lokalen Netz oder aber das ganze Internet. Bei TCP/IP handelt es sich nicht nur um ein einzelnes Protokoll, sondern um eine Fülle unterschiedlicher aufeinander aufbauender Protokolle mit teils sehr unterschiedlichen Aufgaben. Man spricht von einer »Protokollfamilie«.

Die Protokolle der TCP/IP-Protokollfamilie lassen sich zumindest ungefähr in das ISO/OSI-Schichtenmodell aus Bild 1.1 einordnen. Die wichtigsten sind hier kurz aufgelistet:

Netzzugangsschicht Ethernet, IEEE 802.11, PPP (dies sind strenggenommen keine TCP/IP-Protokolle)

Internetschicht IP, ICMP, ARP

Transportschicht TCP, UDP, …

Anwendungsschicht HTTP, DNS, FTP, SSH, NIS, NFS, LDAP, …

IP Das »Internet Protocol« [RFC0791] stellt die Verbindung zwischen zwei Systemen her. Es ist als Protokoll der ISO/OSI-Schicht 3 dafür verantwortlich, dass die Daten durch das Internet den Weg vom Sender zum Empfänger finden. Der Haken an der Sache ist, dass dieser Weg weite Strecken umfassen kann, die aus diversen unabhängigen Abschnitten mit deutlich verschiedener Netzwerktechnik bestehen und die deutlich unterschiedliche Kommunikationsparameter aufweisen.

Adressraum

Eine der Leistungen von IP besteht darin, einen »globalen« Adressraum zur Verfügung zu stellen, der jedem am Internet beteiligten System eine eindeutige

Wegleitung

Adresse gibt, über die es identifiziert werden kann. Ferner sorgt es für die Wegleitung (engl. *routing*) von einem System zum anderen ohne Berücksichtigung der tatsächlich verwendeten Netzwerktechnik.

verbindungsloses Protokoll

IP ist ein **verbindungsloses Protokoll**, das heißt, im Gegensatz z. B. zum traditionellen Telefonnetz wird keine feste Verbindung (»Draht«) für die Kommunikation zweier Systeme zur Verfügung gestellt[1], sondern die zu übertragenden

Datagramme

Daten werden in Häppchen, sogenannte **Datagramme**, eingeteilt, die unabhängig voneinander adressiert und zugestellt werden. Prinzipiell kann jedes Datagramm einen anderen Weg zum Empfänger nehmen als das vorige; dies macht IP unempfindlich gegen Ausfälle von Leitungen oder Vermittlungsrechnern, solange sich noch irgendein Weg vom Quell- zum Zielsystem finden läßt. IP gibt keine Garantie, dass alle abgeschickten Daten auch tatsächlich beim Zielsystem ankommen, und genausowenig wird garantiert, dass die Daten, die tatsächlich ankommen, in derselben Reihenfolge ankommen, in der sie abgeschickt wurden. Es ist die Aufgabe »höhergelegener« Protokolle, hier für Ordnung zu sorgen, falls die Anwendung das erfordert.

Fragmentierung

Außerdem kümmert IP sich um **Fragmentierung**. IP-Datagramme dürfen bis zu 65535 Bytes lang sein, aber bei den meisten Transportprotokollen sind nur wesentlich kürzere Protokolldateneinheiten erlaubt – bei Ethernet beispielsweise nur bis zu 1500 Bytes. Längere Datagramme müssen darum »fragmentiert« übertragen werden – am Anfang einer entsprechenden Teilstrecke wird das Datagramm auseinandergenommen, in nummerierte Fragmente aufgeteilt und später wieder zusammengesetzt. IP sorgt dafür, dass nur solche Datagramme als offiziell empfangen gelten, in denen kein Fragment fehlt.

TCP Das »Transmission Control Protocol« (TCP) ist ein zuverlässiges, verbindungsorientiertes Protokoll, das u. a. in [RFC0793] definiert ist. Im Gegensatz zum verbindungslosen IP kennt TCP Operationen zum Verbindungsauf- und -abbau, mit denen zumindest eine »virtuelle« Verbindung zwischen Quell- und Zielsystem geschaltet wird – da TCP-Daten wie alle anderen Daten auch über IP übertragen werden, erfolgt die tatsächliche Datenübertragung nach wie vor verbindungslos und unzuverlässig. TCP erreicht Zuverlässigkeit, indem die Gegenstelle die Ankunft jedes Pakets (im TCP-Jargon »Segment«) bestätigt. Jede der beiden

Folgenummer

kommunizierenden Stationen versieht ihre Segmente mit Folgenummern (engl.

[1] Auch das Telefonnetz – im Fachjargon POTS (für *plain old telephone system*) genannt – funktioniert längst nicht mehr so.

sequence numbers), die die Gegenstelle in einem ihrer nächsten Segmente als »angekommen« quittiert. Kommt innerhalb einer gewissen definierten Zeitspanne keine solche Quittung, versucht die sendende Station, das Segment erneut zu schicken, um es vielleicht diesmal bestätigt zu bekommen. Jedes System unterstützt viele unabhängige, gleichzeitige TCP-Verbindungen, zwischen denen anhand von **Portnummern** unterschieden wird.

Portnummern

Die Kombination aus einer IP-Adresse und einer Portnummer zusammen mit der IP-Adresse und Portnummer der Gegenstelle bezeichnet man auch als *socket*. (Derselbe TCP-Port auf einer Station darf gleichzeitig an mehreren TCP-Verbindungen mit unterschiedlichen Gegenstellen – definiert durch IP-Adresse und Portnummer – beteiligt sein.)

Der Aufbau einer (virtuellen) TCP-Verbindung erfolgt über den sogenannten **Drei-Wege-Handshake** (engl. *three-way handshake*. Im Drei-Wege-Handshake einigen die Kommunikationspartner sich über die zu verwendenden Folgenummern.

Drei-Wege-Handshake

UDP Im Gegensatz zu TCP ist das »User Datagram Protocol« (UDP) [RFC0768] ein verbindungsloses und unzuverlässiges Protokoll. Tatsächlich ist es nicht viel mehr als »IP mit Ports«, denn wie bei TCP können auf einer Station maximal 65535 Kommunikationsendpunkte unterschieden werden (UDP und TCP können dieselbe Portnummer gleichzeitig für unterschiedliche Zwecke verwenden). Bei UDP entfällt der Verbindungsaufbau von TCP genau wie die Bestätigungen, so dass das Protokoll viel »schneller« ist – allerdings um den Preis, dass wie bei IP Daten verloren gehen oder durcheinander geraten können.

Ports TCP und UDP unterstützen das Konzept von Ports, über die ein System mehr als eine Netzwerkverbindung gleichzeitig verwalten kann (gut, bei UDP gibt es keine »Verbindungen«, aber trotzdem …). Getrennt für TCP und UDP gibt es jeweils 65536 Ports, die allerdings nicht alle sinnvoll benutzt werden können. Die Ports 0 bis 1023 sind als *well-known ports* und die von 1024 bis 49151 als *registered ports* potentiell fest bestimmten Diensten zugeordnet. Die Ports 49152 bis 65535 stehen als *dynamic and/or private ports* für die Client-Seite von Verbindungen oder die Implementierung privater Dienste zur Verfügung.

Ports

Auf einem Linux-System steht eine Zuordnungstabelle in der Datei `/etc/services` (Bild 1.2). Diese Zuordnung wird zum Beispiel vom Internet-Daemon (`inetd` oder `xinetd`) verwendet, um zu einem gegebenen Dienstnamen den passenden Port zu finden.

1.1.5 Adressen

IP-Adressen sind 32 Bit lang und werden normalerweise als *dotted quads* notiert – man betrachtet sie als Folge von vier 8-Bit-Zahlen, die man dezimal als Werte zwischen 0 und 255 hinschreibt, etwa als »`203.177.8.4`«. Jede IP-Adresse wird weltweit eindeutig vergeben und bezeichnet eine Station in einem bestimmten Teilnetz des Internet. Dazu werden IP-Adressen in einen Netzwerk- und einen Stationsanteil aufgeteilt.

IP-Adressen

Der Netzwerk- und Stationsanteil ist variabel und kann der Anzahl der in einem Netz benötigten Stationsadressen angepasst werden. Wenn der Stationsanteil n Bit beträgt, bleiben für den Netzwerkanteil 32 – n Bit. Die Verteilung dokumentiert die **Netzmaske**, die für jedes Bit der IP-Adresse, das zum Netzwerkanteil gehört, eine binäre 1 und für jedes Bit des Stationsanteils eine binäre 0 enthält. Die Netzmaske wird entweder als *dotted quad* oder – heutzutage oft – einfach als Anzahl der Einsen notiert. »`203.177.8.4/24`« wäre also eine Adresse in einem Netz mit der Netzmaske »`255.255.255.0`«.

Netzmaske

Die erste und die letzte IP-Adresse in einem Netzwerk werden verabredungsgemäß für spezielle Zwecke reserviert: Die erste Adresse (Stationsanteil nur binäre Nullen) ist die **Netzwerkadresse**, die letzte Nummer (Stationsanteil nur bi-

Netzwerkadresse

```
# Network services, Internet style

echo       7/tcp
echo       7/udp
discard    9/tcp    sink null
discard    9/udp    sink null
systat     11/tcp   users
daytime    13/tcp
daytime    13/udp
netstat    15/tcp
qotd       17/tcp   quote
chargen    19/tcp   ttytst source
chargen    19/udp   ttytst source
ftp-data   20/tcp
ftp        21/tcp
fsp        21/udp   fspd
ssh        22/tcp                  # SSH Remote Login Protocol
ssh        22/udp                  # SSH Remote Login Protocol
telnet     23/tcp
smtp       25/tcp   mail
◁◁◁◁◁
```

Bild 1.2: Die Datei `/etc/services` (Auszug)

Tabelle 1.1: Private IP-Adressbereiche nach RFC 1918

Adressraum	von	bis
Class A	10.0.0.0	10.255.255.255
Class B	172.16.0.0	172.31.255.255
Class C	192.168.0.0	192.168.255.255

Broadcast-Adresse

näre Einsen) die **Broadcast-Adresse**. Im obigen Beispiel wäre also `203.177.8.0` die Netzwerkadresse und `203.177.8.31` die Broadcast-Adresse. Für die Stationen stehen dann die Nummern von 1 bis 30 zur Verfügung.

Private IP-Adressen

Für Netzwerke, die nicht direkt an das Internet angeschlossen sind, sind besondere Adressbereiche, die privaten IP-Adressen gemäß [RFC1918], vorgesehen, die im Internet nicht geroutet werden (Tabelle 1.1). Diese Adressen können Sie ungeniert in Ihren lokalen Netzen verwenden.

1.1.6 Wegleitung

Wegleitung (engl. *routing*) dient dazu, IP-Datagramme, die nicht direkt im lokalen Netz zugestellt werden können, an die richtige Adresse zu schicken. Sie greift da, wo der Empfänger eines IP-Datagramms nicht im selben Netz zu finden ist wie der Absender. Feststellen kann die sendende Station das (natürlich) anhand der IP-Adresse der gewünschten Zielstation, indem sie den Teil der Zieladresse betrachtet, der von ihrer eigenen Netzmaske »abgedeckt« wird, und überprüft, ob er mit ihrer eigenen Netzwerkadresse übereinstimmt. Wenn das der Fall ist, ist der Empfänger »lokal« und kann direkt erreicht werden

Routing-Tabelle

Kann der Empfänger nicht direkt erreicht werden, konsultiert die Station (jedenfalls, wenn sie ein Linux-Rechner ist) eine Routing-Tabelle, die zumindest ein *default gateway* ausweisen sollte, also eine Station, die sich um die Weiterleitung nicht direkt zustellbarer Datagramme kümmert. (Diese Station muss in aller Regel selber direkt erreichbar sein.) Eine solche Station heißt »Router« und ist entweder selbst ein Computer oder ein spezielles für diese Funktion ausgelegtes Gerät. Eine wichtige Beobachtung ist, dass eine Station (PC oder Router) normalerweise nur

über den direkt nächsten Schritt der Wegleitung (man sagt auch »Hop«) entscheidet, anstatt den kompletten Weg vom ursprünglichen Absender des Datagramms bis zum Empfänger vorzugeben.

1.1.7 Subnetting

Es ist möglich, ein Netz (gegeben durch Netzadresse und Netzmaske) durch »Subnetting« weiter zu unterteilen. Dies ist zum Beispiel nötig, wenn die Stationen auf mehrere physikalische LANs aufgeteilt werden sollen. In diesem Fall müssen Sie die Netzmaske der »Subnetze« verlängern (so dass in den Subnetzen entsprechend weniger Adressen für Stationen zur Verfügung stehen).

Zum Beispiel könnten Sie das Netz `203.177.8.0/24` in ein Subnetz für maximal 126 Stationen (etwa `203.177.8.0/25` mit den Stationsadressen `203.177.8.1` bis `203.177.8.126` und der Broadcast-Adresse `203.177.8.127`) und zwei Subnetze mit je 62 Stationsadressen (etwa `203.177.8.128/26` und `203.177.8.192/26` mit den respektiven Stationsadressen `203.177.8.129` bis `203.177.8.190` und `203.177.8.193` bis `203.177.8.254` sowie den Broadcast-Adressen `203.177.8.191` und `203.177.8.255`) aufteilen.

Wenn Sie sowas machen, sind Sie natürlich auch für die Wegleitung selber verantwortlich.

1.2 Linux-Netzkonfiguration

1.2.1 Netzschnittstellen

Je nach verwendeter Technik und Zugangsverfahren sprechen Linux-Rechner das Netz über Modems, ISDN-Karten, Ethernet- oder WLAN-Adapter und ähnliches an. Die folgenden Abschnitte beschäftigen esich hauptsächlich mit der Einrichtung von Ethernetkarten.

Eine Netzwerkkarte wird unter Linux wie andere Hardware auch vom Kernel angesteuert – heute normalerweise über modulare Treiber, die bei Bedarf dynamisch geladen werden. Anders als zum Beispiel Festplattenpartitionen oder Drucker erscheinen Netzwerkkarten aber nicht als Gerätedateien in `/dev`, sondern werden über Schnittstellen (engl. *interfaces*) angesprochen. Diese Schnittstellen haben Namen; ein typischer Name für eine Ethernet-Karte wäre zum Beispiel `eth0`.

Schnittstellen

Netzwerkkarten werden heutzutage beim Start des Systems vom Kernel erkannt, der anhand der PCI-ID das richtige Treibermodul identifizieren kann. Es obliegt der `udev`-Infrastruktur, der Netzwerkkarte einen Namen zu geben und den Treiber tatsächlich zu laden. Bei Rechnern mit nur einer Netzwerkkarte ist das in aller Regel unproblematisch; gibt es mehrere, können Sie Überraschungen erleben. Hierzu später mehr.

Bevor Sie eine Schnittstelle zum Zugriff auf das Netz verwenden können, müssen Sie ihr eine IP-Adresse, eine Netzmaske und so weiter zuweisen. Manuell geht das traditionellerweise mit dem Kommando `ifconfig`:

```
# ifconfig eth0 192.168.0.75 up
# ifconfig eth0
eth0 Link encap:Ethernet  HWaddr 00:A0:24:56:E3:73
     inet addr:192.168.0.75 Bcast:192.168.0.255 Mask:255.255.255.0
     inet6 addr: fe80::2a0:24ff:fe56:e373/64 Scope:Link
     UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
     RX packets:0 errors:0 dropped:0 overruns:0 frame:0
     TX packets:6 errors:0 dropped:0 overruns:0 carrier:6
     collisions:0 txqueuelen:100
     RX bytes:0 (0.0 b)  TX bytes:460 (460.0 b)
     Interrupt:5 Base address:0xd800
```

Nach der Zuweisung einer IP-Adresse können Sie durch Aufruf des gleichen Kommandos ohne Angabe einer IP-Adresse den Status der Schnittstelle auslesen. Hier werden nicht nur die aktuelle IP-Adresse, sondern auch der Hardwaretyp, die MAC- (oder »Hardware-«)Adresse, die Broadcast-Adresse, die Netzmaske, die IPv6-Adresse und viele weitere Daten angezeigt.

Loopback-Interface

Das Loopback-Interface hat nach Konvention die IP-Adresse `127.0.0.1` und wird automatisch konfiguriert. Sollte das aus irgendwelchen Gründen einmal nicht klappen oder die Konfiguration verlorengehen, dann können Sie das über

```
# ifconfig lo 127.0.0.1 up
```

nachholen.

Aliasnamen

Zu Testzwecken oder für besondere Anforderungen kann es sinnvoll sein, einer Schnittstelle einen Aliasnamen mit einer abweichenden IP-Adresse, Netzmaske usw. zu geben. Das ist mit `ifconfig` kein Problem:

```
# ifconfig eth0:0 192.168.0.111
# ifconfig eth0:0
eth0:0 Link encap:Ethernet  HWaddr 00:A0:24:56:E3:72
     inet addr:192.168.0.111 Bcast:192.168.0.255 Mask:255.255.255.0
     UP BROADCAST MULTICAST  MTU:1500  Metric:1
     Interrupt:5 Base address:0xd800
```

Der Aliasname wird aus dem Interfacenamen gebildet, indem getrennt durch einen Doppelpunkt eine Namenserweiterung angehängt wird. Wie diese Namenserweiterung aussieht, ist gleichgültig (es spräche nichts gegen `eth0:Mr.X`), nach Konvention numeriert man die Aliasnamen aber fortlaufend durch: `eth0:0`, `eth0:1`, …

1.2.2 Wegleitung

Routing-Tabelle

Der Kernel unterhält eine Routing-Tabelle, die die aktuelle Konfiguration der Wegleitung zusammenfasst. Sie enthält Regeln (die Routen), die beschreiben, wohin welche Datagramme geschickt werden müssen. Maßgeblich dafür ist deren Zieladresse. Sie können die Routing-Tabelle mit dem Kommando `route` abrufen:

```
# ifconfig eth0 192.168.0.75
# route
Kernel IP routing table
Destination  Gateway Genmask       Flags Metric Ref Use Iface
192.168.0.0  *       255.255.255.0 U     0      0   0   eth0
```

Die Spalten der Tabelle haben folgende Bedeutung:

Default-Route

- Die erste Spalte enthält die Zieladresse. Als Adresse kommen Netz- und Stationsadressen sowie der Eintrag (`default`) für die sogenannte Default-Route in Frage. Die Default-Route legt das Ziel für alle Datagramme fest, für die keine der übrigen Routen gilt.
- Die zweite Spalte definiert als Ziel für die Datagramme einen Router, an den die Pakete weitergegeben werden. Gültige Einträge an dieser Stelle sind Stationsadressen sowie der Eintrag »*« wenn die Pakete nicht an einen anderen Rechner gehen sollen.
- Die dritte Spalte enthält die zur Zieladresse passende Netzmaske. Handelt es sich bei der Zieladresse um eine Station, dann steht hier die Netzmaske `255.255.255.255`. Die Default-Route hat die Netzmaske `0.0.0.0`.

- Die vierte Spalte enthält Flags, die die Route näher beschreiben. Folgende Werte können u. a. vorkommen:

  `U` die Route ist aktiv (up)

  `G` die Route ist eine »Gateway-Route«, das heisst, als Ziel ist ein Router (und kein direkt angeschlossenes Netz wie mit "*") angegeben.

  `H` die Route ist eine »Host-Route«, das heisst, die Zieladdresse bezeichnet einen einzelnen Rechner. `G` und `H` schliessen sich natürlich nicht aus und tauchen manchmal zusammen auf.

- Die fünfte und sechste Spalte enthalten Angaben, die bei dynamischem Routing eine Rolle spielen: Die »Metrik« in der fünften Spalte gibt die Anzahl der »Hops« zum Ziel an; sie wird vom Linux-Kern nicht ausgewertet, sondern ist vor allem für Programme wie den `gated` interessant. Der Wert in der sechsten Spalte wird in Linux nicht verwendet.

- Die siebte Spalte gibt an, wie oft die Route schon verwendet wurde.

- Die achte Spalte schließlich enthält optional die Schnittstelle, über die die Datagramme weitergeleitet werden sollen. Das kommt insbesondere bei Routern vor, die mehrere Interfaces besitzen, etwa Ethernet-Schnittstellen in verschiedenen Netzsegmenten oder eine Ethernet-Schnittstelle und eine Schnittstelle zum ISDN.

Am Beispiel wird deutlich, dass der Kernel beim Setzen der IP-Adresse mit `ifconfig` nicht nur eigenständig Netzmaske und Broadcast-Adresse setzt, sondern auch mindestens eine Route – diejenige nämlich, die alle Datagramme, deren Zieladressen im direkt an die Schnittstelle angeschlossenen Netz liegen, auf dieses Netz leitet.

Das Kommando `route` dient nicht nur zum Abrufen, sondern auch zur Manipulation der Routing-Tabelle. Für das oben gezeigte Beispiel (drei lokale Ethernet-Segmente und die PPP-Verbindung) würde die Routing-Tabelle etwa folgendermassen aufgebaut:

```
# route add -net 192.168.0.0 netmask 255.255.255.0 dev eth0
# route add -net 192.168.2.0 netmask 255.255.255.0 dev eth1
# route add -net 10.10.3.0 netmask 255.255.255.0 gw 192.168.0.1
# route add -host 112.22.3.4 dev ppp0
# route add default dev ppp0
```

Die ersten beiden Zeilen im Beispiel sind eigentlich nicht nötig, da die entsprechenden Routen automatisch in Kraft gesetzt werden, wenn die Schnittstellen ihre Adressen bekommen.

Löschen könnten Sie die Routen zum Beispiel so: Routen löschen

```
# route del -net 192.168.0.0 netmask 255.255.255.0
# route del -net 192.168.2.0 netmask 255.255.255.0
# route del -net 10.0.3.0 netmask 255.255.255.0
# route del -host 112.22.3.4
# route del default
```

Zum Löschen einer Route müssen Sie die gleichen Angaben machen wie zum Hinzufügen einer Route. Lediglich die Angabe des Gateways bzw. des Interfaces können Sie weglassen.

Soll ein Rechner wie im Beispiel als Gateway zwischen zwei Netzen dienen, so sollte der Kernel IP-Pakete, die nicht für den Rechner selbst bestimmt sind, entsprechend der Routing-Tabelle weitergeben. Dieses sogenannte **IP-** IP-Forwarding

Forwarding ist standardmäßig ausgeschaltet. Die Einstellung und Anzeige des IP-Forwarding findet über die (Pseudo!-)Datei `/proc/sys/net/ipv4/ip_forward` statt. In ihr »steht« nur ein Zeichen – entweder eine Null (abgeschaltet) oder eine Eins (aktiviert). »Geschrieben« wird die Datei üblicherweise mit `echo`:

```
# cat /proc/sys/net/ipv4/ip_forward
0
# echo 1 > /proc/sys/net/ipv4/ip_forward
# cat /proc/sys/net/ipv4/ip_forward
1
```

Routen und die Einstellung von `ip_forward` gehen beim Herunterfahren des Rechners verloren. Je nach Distribution gibt es Möglichkeiten, diese Informationen dauerhaft wirksam zu machen (Abschnitt 1.2.4).

1.2.3 Netzkonfiguration mit `ip`

Mit dem Kommando `ip` können sowohl die Schnittstelle als auch Routen konfiguriert werden. Dessen Syntax lautet allgemein wie folgt:

`ip` [⟨*Option*⟩] ⟨*Objekt*⟩ [⟨*Kommando*⟩ [⟨*Parameter*⟩]]

Als ⟨*Objekt*⟩ kommen unter anderem `link` (Parameter der Schnittstelle), `addr` (IP-Adresse und andere Adressen der Schnittstelle) und `route` (Auslesen, Setzen und Löschen von Routen) in Frage. Für jedes Objekt stehen spezifische Kommandos zur Verfügung.

Wird kein Kommando angegeben, werden die momentanen Einstellungen entsprechend des Kommandos `list` bzw. `show` angezeigt. Weitere typische Kommandos sind `set` für das Objekt `link` sowie `add` und `del` für die Objekte `addr` und `route`.

Die meisten Kommandos erfordern noch weitere Parameter, schließlich müssen Sie, möchten Sie mit »`ip addr add`« eine IP-Adresse zuweisen, diese IP-Adresse auch angeben.

Die entsprechende Syntax können Sie dem Befehl `ip` mit dem Kommando `help` entlocken. So zeigt »`ip help`« alle möglichen Objekte an und »`ip link help`« alle zum Objekt `link` gehörenden Parameter inklusive Syntax. Leider ist die Syntax nicht immer ganz einfach zu durchschauen.

Als Beispiel: Möchten Sie einer Netzwerkkarte eine IP-Adresse zuweisen, können Sie dafür folgendes Kommando verwenden:

```
# ip addr add local 192.168.2.1/24 dev eth0 brd +
```

Die Aktivierung erfolgt gesondert:

```
# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo-fast qlen 100
        link/ether 00:a0:24:56:e3:72 brd ff:ff:ff:ff:ff:ff
        inet 192.168.2.1/24 brd 192.168.2.255 scope global eth0
# ip link set up dev eth0
# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo-fast qlen 100
        link/ether 00:a0:24:56:e3:72 brd ff:ff:ff:ff:ff:ff
        inet 192.168.2.1/24 brd 192.168.2.255 scope global eth0
        inet6 fe80::2a0:24ff:fe56:e372/64 scope link
```

Das Setzen und Löschen von Routen ist einfacher als mit `route`:

```
# ip route add 192.168.2.1 via 192.168.0.254
# ip route del 192.168.2.1
```

1.2.4 Dauerhafte Netzkonfiguration

Eins ist sicher: Wenn Sie einmal die richtige Netzkonfiguration für Ihr System herausbekommen haben, werden Sie diese nicht immer wieder von neuem einstellen wollen. Leider vergisst der Linux-Kernel sie aber beim Herunterfahren.

Die verschiedenen Linux-Distributionen haben dieses Problem auf unterschiedliche Weise gelöst:

Bei Debian GNU/Linux und den davon abgeleiteten Distributionen steht die Netzkonfiguration in der Datei `/etc/network/interfaces`. Diese Datei ist mehr oder weniger selbsterklärend:

```
# cat /etc/network/interfaces
auto lo eth0

iface lo inet loopback

iface eth0 inet static                                  oder „... inet dhcp“
  address 192.168.0.2
  netmask 255.255.255.0
  network 192.168.0.0
  broadcast 192.168.0.255
  up route add -net 10.10.3.0/24 gw 192.168.0.1
  down route del -net 10.10.3.0/24 gw 192.168.0.1
```

In der Datei gibt es einen Eintrag für jede Schnittstelle. Die Schnittstellen können mit den Kommandos `ifup` und `ifdown` einzeln oder (mit der Option `-a`) kollektiv aktiviert bzw. deaktiviert werden; beim Systemstart kümmert sich das Skript `/etc/init.d/networking` um die Initialisierung der Schnittstellen. Mehr Beispiele für die fremdartigen und wundervollen Dinge, die mit dem Debian-Netzkonfigurationsmechanismus möglich sind, finden Sie in `interfaces`(5) und der Datei `/usr/share/doc/ifupdown/examples/network-interfaces.gz`.

Mit YaST vorgenommene Netzeinstellungen werden im Verzeichnis `/etc/sysconfig/network` abgelegt. Hier befindet sich für jede Schnittstelle eine Datei namens `ifcfg-`⟨*Schnittstelle*⟩ (also zum Beispiel `ifcfg-eth0`), die die Einstellungen für die betreffende Schnittstelle enthält. Das kann ungefähr so aussehen:

```
BOOTPROTO='static'                                  oder dhcp (unter anderem)
BROADCAST='192.168.0.255'
ETHTOOL_OPTIONS=''
IPADDR='192.168.0.2'
MTU=''
NAME='79c970 [PCnet32 LANCE]'                              Name im YaST
                                                    (VMware läßt grüßen)
NETMASK='255.255.255.0'                             Oder PREFIXLEN=24
NETWORK='192.168.0.0'
REMOTE_IPADDR=''                                           Gegenstelle bei PPP
STARTMODE='auto'                                    oder manual, hotplug, ...
USERCONTROL='no'
```

(An diesen Dateien dürfen Sie prinzipiell auch mit der Hand herumbasteln. Eine ausführliche Erklärung steht in `ifcfg`(5).) Allgemeine Einstellungen für die Netzkonfiguration stehen in `/etc/sysconfig/network/config`. – Auch die SUSE-Distributionen unterstützen Kommandos namens `ifup` und `ifdown`, die allerdings subtil anders funktionieren als die von Debian GNU/Linux. Ferner können Sie die komplette Netzanbindung über Kommandos wie »`rcnetwork start`« steuern.

Routen können Sie bei den Novell/SUSE-Distributionen über die Datei `/etc/sysconfig/network/routes` konfigurieren. Der Inhalt der Datei (hier passend zum oben verwendeten Beispiel) ähnelt der Darstellung des Befehls `route`:

```
# cat /etc/sysconfig/network/routes
10.10.3.0       192.168.0.1     255.255.255.0   eth0
112.22.3.4      0.0.0.0         255.255.255.255 ppp0
default         112.22.3.4      -               -
```

Soll kein Gateway verwendet werden, lautet der Eintrag »`0.0.0.0`«, nicht gesetzte Netzmasken oder Schnittstellennamen werden durch ein »`-`« dargestellt. Auch die Routen werden durch Aufruf von »`rcnetwork restart`« gesetzt.

Wie bei SUSE existieren bei den Red-Hat-Distributionen Dateien in der Art von `ifcfg-eth0` für die Konfiguration jeder Schnittstelle, nur dass sie sich in einem Verzeichnis namens `/etc/sysconfig/network-scripts` befinden. Die SUSE-Dateien sind aber nicht 1 : 1 übertragbar, da sie sich von den Red-Hat-Dateien im internen Aufbau unterscheiden. Bei Fedora könnten Sie unsere Beispielkonfiguration für `eth0` etwa wie folgt realisieren: In `/etc/sysconfig/network-scripts/ifcfg-eth0` steht

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
NETWORK=192.168.0.0
NETMASK=255.255.255.0
IPADDR=192.168.0.2
USERCTL=no
```

Die `ifup`- und `ifdown`-Kommandos gibt es auch bei Fedora, aber wie bei der SUSE können Sie immer nur eine Schnittstelle auf einmal starten oder anhalten.

Statische Routen können Sie bei Fedora in eine Datei in `/etc/sysconfig/network-scripts` tun, die `route-`⟨*Schnittstelle*⟩ heißt (also zum Beispiel `route-eth0`). In diesem Fall ist das Format wie folgt:

```
ADDRESS0=10.10.3.0
NETMASK0=255.255.255.0
GATEWAY0=192.168.0.1
```

(zusätzliche Routen verwenden entsprechend `ADDRESS1`, `NETMASK1`, …, `ADDRESS2` und so weiter). Es gibt auch noch ein älteres Dateiformat, in dem einfach jede Zeile der Datei an »`ip route add`« angehängt wird. Dabei bieten sich also Zeilen an wie

```
10.10.3.0/24 via 192.168.0.1
```

Zu allem Überfluss können Sie auch noch in `/etc/sysconfig/static-routes` statische Routen angeben, ohne sich auf einzelne Schnittstellen beziehen zu müssen. Die Zeilen in dieser Datei werden nur beachtet, wenn sie mit dem Schlüsselwort `any` anfangen; der Rest der Zeile wird an »`route add -`« angehängt (Konsistenz? Wer braucht Konsistenz?), so dass aus einer Zeile wie

```
any net 10.10.3.0 netmask 255.255.255.0 gw 192.168.0.1
```

das Kommando

```
route add -net 10.10.3.0 netmask 255.255.255.0 gw 192.168.0.1
```

resultiert.

1.2.5 DHCP

DHCP, das »Dynamic Host Configuration Protocol«, dient dazu, Ihnen als Administrator die Mühe abzunehmen, die passenden Netzparameter auf jeder einzelnen Station im Netz konfigurieren zu müssen. Statt dessen holt ein Linux-Rechner sich die Netzparameter – neben der IP-Adresse mit Zubehör typischerweise die Adresse eines Default-Routers und eines oder mehrerer DNS-Server – von einem entfernten DHCP-Server, wenn die Netzwerkkarte gestartet wird.

Um DHCP zur Konfiguration zu verwenden, müssen Sie die Konfiguration der gängigen Linux-Distributionen nur geringfügig anpassen:

Setzen Sie bei Debian GNU/Linux oder Ubuntu einfach in `/etc/network/interfaces` statt

```
iface eth0 inet static
```

und den darauffolgenden Zeilen mit den Adressen- und Routeninformationen die Zeile

```
iface eth0 inet dhcp
```

ein. Adresse, Netzmaske und Default-Route werden dann vom DHCP-Server bezogen. Sie können natürlich weiterhin mit `up` und `down` Kommandos ausführen, wenn die Verbindung steht oder bevor sie abgebaut wird.

Bei den Novell/SUSE-Distributionen setzen Sie in der Datei mit der Konfiguration für die betreffende Schnittstelle (`ifcfg-eth0` oder so) statt

```
BOOTPROTO='static'
```

den Parameter

```
BOOTPROTO='dhcp'
```

Die Felder `BROADCAST`, `IPADDR`, `NETMASK` und `NETWORK` lassen Sie einfach leer.

Bei Fedora und den anderen Red-Hat-Distributionen setzen Sie zur Verwendung von DHCP in der Konfigurationsdatei für die Schnittstelle statt

```
BOOTPROTO=none
```

den Parameter

```
BOOTPROTO=dhcp
```

Die Adressenparameter können Sie dann einfach weglassen.

Kommandos in diesem Kapitel

ifconfig	Konfiguriert Netzwerk-Schnittstellen	ifconfig(8)	19
ifdown	Schaltet eine Netzwerk-Schnittstelle aus (Debian)	ifdown(8)	23
ifup	Schaltet eine Netzwerk-Schnittstelle ein (Debian)	ifup(8)	23
inetd	Internet-Superserver, überwacht Ports und startet ggf. Dienste	inetd(8)	17
ip	Verwaltet Netzwerkschnittstellen und Routing	ip(8)	22
route	Verwaltet die statische Routing-Tabelle im Linux-Kern	route(8)	20
xinetd	Verbesserter Internet-Superserver, überwacht Ports und startet ggf. Dienste	xinetd(8)	17

Zusammenfassung

- Das Internet wurzelt in den 1960er Jahren, wurde in den frühen 1980er Jahren auf seine heutige technische Basis gestellt und nahm in den 1980er und 1990er Jahren einen ungeahnten Aufschwung.
- Das ISO/OSI-Referenzmodell dient zur Begriffsbildung über die Struktur von Rechnerkommunikation.
- TCP/IP ist die heute populärste Protokollfamilie für den Datentransfer in Rechnernetzen.
- IP-Adressen identifizieren Stationen weltweit. Sie sind 32 Bit lang und bestehen aus einem Netzwerk- und einem Stationsteil, zwischen denen über die Netzmaske unterschieden wird.
- IP-Netze können weiter in Subnetze unterteilt werden, indem man die Netzmaske anpasst.
- Das Kommando `ifconfig` dient zur Konfiguration von Interface-Parametern auf niedriger Ebene. Sie können damit auch das Loopback-Interface konfigurieren und Aliasnamen für Interfaces vergeben.
- Routen geben an, wie IP-Datagramme an ihren Empfänger geleitet werden.
- Zur Konfiguration von Routen dient das Kommando `route`.
- Das Kommando `ip` ist ein komfortabler Ersatz für `ifconfig` und `route`.
- Die verschiedenen Linux-Distributionen bieten unterschiedliche Methoden zur dauerhaften Netzkonfiguration an.
- Mit DHCP können Linux-Rechner Netzparameter dynamisch von einem zentralen Server beziehen.

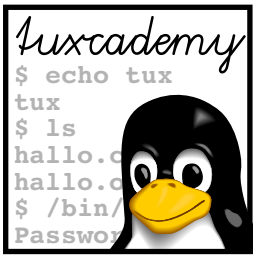
Literaturverzeichnis

RFC0768 J. Postel. »User Datagram Protocol«, August 1980. http://www.ietf.org/rfc/rfc0768.txt

RFC0791 Information Sciences Institute. »Internet Protocol«, September 1981. http://www.ietf.org/rfc/rfc0791.txt

RFC0793 Information Sciences Institute. »Transmission Control Protocol«, September 1981. http://www.ietf.org/rfc/rfc0793.txt

RFC1918 Y. Rekhter, B. Moskowitz, D. Karrenberg, et al. »Address Allocation for Private Internets«, Februar 1996. http://www.ietf.org/rfc/rfc1918.txt



2

Benutzer benachrichtigen

Inhalt

2.1 Einleitung 28
2.2 Vorwarnung vor dem Anmelden: /etc/issue und /etc/issue.net . . . 28
2.3 Begrüßung nach dem Anmelden: /etc/motd. 29
2.4 Alle Benutzer benachrichtigen: wall 29
2.5 Das Ende der Welt ist nahe: shutdown 30

Lernziele

- Die Bedeutung der Dateien /etc/motd, /etc/issue usw. kennen
- Alle angemeldeten Benutzer benachrichtigen können

Vorkenntnisse

- Kenntnisse über Linux-Werkzeuge und die Shell

2.1 Einleitung

Hin und wieder möchten Sie als Systemverwalter Ihren Benutzern etwas mitteilen: Der Plattenplatz wird knapp, morgen um 12 Uhr wird der Dateiserver für einen Upgrade heruntergefahren, der Inhaber des improvisierten mikrobiologischen Labors im Bürokühlschrank möge dieses entsorgen und so weiter. Im 21. Jahrhundert ist wahrscheinlich ein Dienst wie »Twitter« das Mittel der Wahl für so etwas, aber Linux bietet auch einige etwas weniger spektakuläre Ansätze.

In diesem Kapitel erklären wir Ihnen, wie Sie entweder allen Benutzern beim Einloggen eine Nachricht zeigen oder alle aktuell angemeldeten Benutzer über etwas benachrichtigen können.

Ein Wort der Warnung: Die Methoden in diesem Kapitel sind weniger nützlich, als sie im ersten Moment scheinen, aus verschiedenen Gründen, die wir an geeigneter Stelle noch diskutieren werden.

2.2 Vorwarnung vor dem Anmelden: `/etc/issue` und `/etc/issue.net`

`/etc/issue`

Bevor ein Benutzer sich (auf einer Textkonsole) anmeldet, bekommt er den Inhalt der Datei `/etc/issue` angezeigt. Darum kümmert sich der für die Konsole zuständige `getty`-Prozess. Nützlich ist das wahrscheinlich für Fälle, wo Sie etwas ausgeben wollen wie

```
Dieses Rechnersystem ist nur für autorisierte Benutzer gedacht.
Wenn Sie kein autorisierter Benutzer sind, dann lassen Sie
gefälligst die Finger davon. Wir behalten uns rechtliche Schritte
vor, wenn wir Sie trotzdem auf diesem System erwischen. Vielen
Dank für Ihr Verständnis.
```

damit klar ist, dass allfällige Cracker auf Ihrem Rechner unerwünscht sind, und diese sich nicht später damit herausreden können, sie hätten nicht gewusst, dass es sich nicht um einen öffentlichen Rechner handelt. (Nicht lachen – sowas ist schon vorgekommen.)

Escape-Sequenzen

`/etc/issue` wird etwas dadurch aufgepeppt, dass Sie »Escape-Sequenzen« verwenden können, um Sachen wie den Namen des aktuellen Rechners, die Anzahl der angemeldeten Benutzer und ähnliches auszugeben (wie das genau geht, hängt von Ihrem `getty` ab). Allerdings ist das sehr »80er Jahre«; auf einer VAX mit 50 Terminals hat es wahrscheinlich einigen Sinn, aber auf den Ein-Personen-Arbeitsplatzrechnern des 21. Jahrhunderts möglicherweise nicht mehr unbedingt.

`/etc/issue.net`

Die Datei `/etc/issue.net` entspricht `/etc/issue`, aber wird nicht von `getty` ausgegeben, sondern vom TELNET-Daemon. (Wenn Sie sich jetzt erinnern, dass Sie den TELNET-Daemon nicht benutzen sollen, dann sind Sie gut.) Allerdings greifen auch andere Dienste sie zumindest wahlweise auf.

Der Haken an der ganzen Sache ist, dass `/etc/issue` zwar nett ist, sofern Leute sich auf der Textkonsole anmelden – aber wer macht das heute noch so? Es gibt natürlich Mittel und Wege, den Inhalt von `/etc/issue` auf den X11-Anmeldebildschirm zu praktizieren, aber durchgesetzt hat sich das bisher noch nicht.

Übungen

2.1 [!1] Schreiben Sie etwas Interessantes in die Datei /etc/issue und überzeugen Sie sich, dass es auch angezeigt wird.

2.2 [3] Wie würden Sie dafür sorgen, dass der Inhalt von /etc/issue auf dem X11-Anmeldebildschirm erscheint? Probieren Sie Ihren Lösungsansatz aus.

2.3 Begrüßung nach dem Anmelden: /etc/motd

Auch nach dem Anmelden können Sie Ihren Benutzern eine Nachricht zukommen lassen, indem Sie sie in die Datei /etc/motd schreiben. Ausgegeben wird diese Datei vom login-Prozess (nicht von getty), und auch Dienste wie die Secure Shell folgen dieser Konvention (zumindest auf Wunsch; beim sshd ist das der Konfigurationsparameter PrintMotd). /etc/motd

Escape-Sequenzen oder solche Sachen gibt es in /etc/motd nicht; die Datei wird einfach so ausgegeben, wie sie ist.

Das Problem mit /etc/motd ist ähnlich dem von /etc/issue; die Datei setzt mehr oder weniger voraus, dass Sie sich auf einer Textkonsole anmelden, sonst bekommen Sie sie nicht zu sehen. Auch hier ist es natürlich möglich, im Rahmen des Sitzungsaufbaus ein Fenster mit dem Inhalt der Datei anzuzeigen, aber das wird von den gängigen grafischen Arbeitsumgebungen genausowenig umgesetzt. Immerhin haben Sie größere Chancen, die /etc/motd zu sehen, wenn Sie sich mit der ssh irgendwo anmelden oder so.

Der Name /etc/motd ist übrigens eine Abkürzung für *message of the day*.

Übungen

2.3 [!1] Schreiben Sie etwas Interessantes in die Datei /etc/motd und überzeugen Sie sich, dass es auch angezeigt wird.

2.4 Alle Benutzer benachrichtigen: wall

Wenn Sie allen angemeldeten Benutzern eine Nachricht »in Echtzeit« schicken möchten, können Sie das mit dem Kommando wall tun. Übergeben Sie entweder einen Dateinamen oder schreiben Sie Ihre Nachricht direkt auf die Standardeingabe, etwa so:

```
$ wall
Im Raum 337 gibt es Kuchen!
Strg+d
```

Bei den angemeldeten Benutzern erscheint dann etwas wie

```
Broadcast Message from hugo@red
        (/dev/pts/3) at 13:33

Im Raum 337 gibt es Kuchen!
```

Wenn ein Benutzer gleichzeitig auf mehreren Terminals angemeldet ist (im Zeitalter grafischer Oberflächen kein Kunststück mehr), erscheint die Nachricht auf allen. Sie wird gnadenlos ins Terminal geschrieben, egal was da sonst schon steht, so dass Sie mitunter den Bildschirm neu zeichnen lassen müssen, um wieder aufzuräumen (in vi und emacs geht das mit der Tastenkombination Strg+l (Ell). Mehrere Terminals

mesg

Sie können wall-mäßige Nachrichten für ein Terminal abbestellen, indem Sie das Kommando »mesg n« geben. In diesem Moment kann nur noch root Ihnen Nachrichten auf dieses Terminal schicken. Mit »mesg y« schalten Sie das Terminal wieder für Nachrichten frei.

wall ist keine Vorahnung auf Facebook, sondern die Abkürzung von *write all*. Es gibt nämlich auch ein Kommando write, mit dem Sie gezielt einem anderen Benutzer eine ähnliche Nachricht schicken können, sozusagen Instant Messaging für Arme.

Übungen

2.4 [!1] Schicken Sie mit wall eine Nachricht an »alle Benutzer« auf Ihrem Rechner.

2.5 [2] Prüfen Sie nach, ob »mesg n« das tut, was es soll. (Dazu müssen Sie sich ggf. mit einem Kollegen zusammentun oder eine zusätzliche Benutzeridentität anlegen.)

2.6 [2] Wie funktioniert das Kommando mesg? (*Tipp:* Schauen Sie sich die Ausgabe von »ls -l `tty`« an, vor und nach einem »mesg n« bzw. »mesg y«.)

2.5 Das Ende der Welt ist nahe: shutdown

Das Programm shutdown verwendet wall, um die angemeldeten Benutzer von einem bevorstehenden Systemhalt oder -neustart zu unterrichten. Sie können eine Nachricht angeben, die dann – je nachdem, wie weit die Aktion von shutdown in der Zukunft liegt – in periodischen Abständen an die Benutzer geschickt wird.

/etc/nologin

shutdown benutzt außerdem noch die Datei /etc/nologin, um unmittelbar vor dem Herunterfahren des Systems (genauer gesagt 5 Minuten davor) die Benutzer daran zu hindern, sich neu anzumelden. Wenn diese Datei existiert, dann werden – jedenfalls auf der Textkonsole – Anmeldeversuche von allen Benutzern außer root zurückgewiesen, wobei der Inhalt der Datei als Erklärung ausgegeben wird.

Übungen

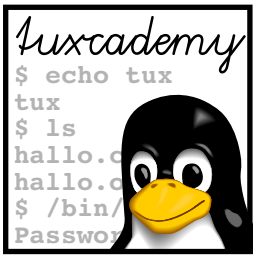
2.7 [2] Nachdem /etc/nologin angelegt wurde, kann sich niemand mehr anmelden. Wie und von welchem Programm wird diese Datei wieder entfernt?

Kommandos in diesem Kapitel

mesg Schaltet wall-Nachrichten für ein Terminal ein oder aus mesg(1) 29
wall Schreibt eine Nachricht auf die Terminals aller angemeldeten Benutzer wall(1) 29

Zusammenfassung

- Die Datei /etc/issue wird vor einem Anmeldevorgang auf der Textkonsole angezeigt.
- Die Datei /etc/motd wird nach einem erfolgreichen Anmeldevorgang auf der Textkonsole angezeigt.
- Das Kommando wall erlaubt es, Nachrichten an alle angemeldeten Benutzer zu schicken.
- shutdown verwendet wall und /etc/nologin, um Benutzer über einen bevorstehenden Systemhalt zu unterrichten.



3

Netzwerkdiagnose mit `tcpdump` und `wireshark`

Inhalt

3.1 Einführung 32
3.2 `tcpdump` 32
3.2.1 Grundlagen 32
3.2.2 Filter 33
3.2.3 Ausgabe 36
3.3 `wireshark` 41
3.3.1 Grundlagen 41
3.3.2 Netzwerkdaten protokollieren 43
3.3.3 Das Hauptfenster 44
3.3.4 Tipps, Tricks und Techniken 48

Lernziele

- Die Paketsniffer `tcpdump` und `wireshark` kennen und einsetzen können

Vorkenntnisse

- Kenntnisse über Linux-Systemadministration
- Kenntnisse über TCP/IP-Protokolle (Kapitel 1)

3.1 Einführung

Mit Werkzeugen wie `ping` und `traceroute` können Sie grundlegende Probleme mit der Netzanbindung Ihres Rechners (oder Ihrer Rechner) ausloten. Allerdings kommt es vor, dass auf den ersten Blick alles zu funktionieren scheint, aber bei bestimmten Protokollen Probleme auftauchen. Oder Ihr Netz scheint aus irgendwelchen Gründen nur im Schneckentempo zu funktionieren. An dieser Stelle kann es dann nötig werden, den Vorgängen im Netz genauer auf den Grund zu gehen. Hierzu dienen Paketsniffer wie `tcpdump` oder `wireshark`, die den gesamten Verkehr auf einem (physikalischen) Netz beobachten und analysieren können.

Paketsniffer

Wichtige Warnung: Die in diesem Kapitel beschriebenen Werkzeuge sind für den Netzwerkadministrator sehr hilfreich, könnten aber auch als (in Staaten wie Deutschland verbotene) »Cracker-Tools« missverstanden werden. Betrachten Sie sie wie ein Skalpell, das in der Hand eines Chirurgen Leben retten, aber in der Hand eines Serienmörders eine gefährliche Waffe sein kann, und verwenden Sie sie entsprechend. Sorgen Sie in Ihrem eigenen Interesse insbesondere dafür, dass Sie sie nur dort einsetzen, wo Sie das definitiv dürfen – also auf Rechnern, die Ihnen gehören oder deren Eigentümer Ihnen das ausdrücklich (am besten schriftlich) erlaubt haben. Die Gesetzgebung ist in diesem Bereich extrem vage, und selbst wenn Sie nichts Ungehöriges im Schild führen, könnte jemand Ihnen daraus einen Strick drehen.

3.2 `tcpdump`

3.2.1 Grundlagen

Normalerweise »interessiert« ein Rechner sich nur für Ethernet-Frames, die entweder an ihn selbst adressiert sind oder als »Broadcast« an alle Stationen auf dem Netz(segment) gehen. Alle anderen Frames werden schon von der Netzwerkkarte ignoriert. Die allermeisten Netzwerkkarten unterstützen aber einen *promiscuous mode*[1], in dem sie *alle* eingehenden Ethernet-Frames an den Linux-Kern weiterreichen. Ein »Paketsniffer« wie `tcpdump` kann diese Daten dann analysieren und ein umfassendes Bild des Netzverkehrs zeichnen, so wie er von dieser Netzwerkschnittstelle gesehen wird.

promiscuous mode

Es ist noch lange nicht gesagt, dass das wirklich der *komplette* Netzverkehr ist. Heutzutage ist es üblich, Switches zu verwenden, die den Verkehr intern »vorsortieren« und nur solche Daten auf das Ethernet-Kabel schicken, von denen sie wissen (oder zu wissen glauben), dass sich ein Rechner am anderen Ende des Kabels für sie interessieren sollte. In diesem Fall zeigt auch `tcpdump` nur den Teil des Netzverkehrs, der den betreffenden Rechner adressiert – entweder ausdrücklich oder über Broadcast.[2]

Was nicht heißt, dass an dieser Stelle alles vorbei ist: Bessere (soll heißen, steuerbare oder neudeutsch *managed*) Switches unterstützen oft die Möglichkeit, den kompletten Netzverkehr, der durch den Switch läuft, in Kopie an eine bestimmte Netzwerkbuchse zu schicken. Das geht gut, solange der Switch nicht so ausgelastet ist, dass er insgesamt – was bessere Switches können – mehr Verkehr vermittelt als sich an eine Buchse schicken läßt. Zum Beispiel könnte ein guter Switch mit 8 100-MBit/s-Ports gleichzeitig mit jeweils der vollen Geschwindigkeit Daten zwischen den Ports 2 und 5, den Ports 7 und 3 und den Ports 4 und 6 vermitteln. In diesem Fall ist es natürlich

[1] Das Wort *promiscuous* ist in diesem Kontext leider unübersetzbar, darum bleiben wir, um Verwirrung zu vermeiden, beim englischen Begriff

[2] Es gibt hinterhältige Methoden, wie Sie trotzdem an Datenverkehr kommen, der nicht für Ihren Rechner gemeint war – Stichwort »ARP-Spoofing«. Wir vertiefen das an dieser Stelle nicht.

nicht möglich, Daten im Wert von 300 MBit/s an den Management-Port 0 zu schicken, wenn dieser auch nur 100 MBit pro Sekunde transportieren kann.

Im einfachsten Fall können Sie mit einem Kommando wie

Einfacher Aufruf

```
# tcpdump -ni eth0
```

den Datenverkehr auf der Schnittstelle `eth0` mitlesen und auf der Standardausgabe von `tcpdump` protokollieren. (Damit Sie `eth0` in den *promiscuous mode* versetzen können, müssen Sie übrigens `root` sein.) Mit der Option »`-i eth0`« wird die Schnittstelle ausgewählt, und die Option `-n` unterdrückt die Auflösung von IP-Adressen zu den dazugehörigen Namen (was mitunter lange dauern kann).

Alternativ können Sie die Netzwerkdaten auch in einer Datei ablegen, statt sie direkt auszugeben, was `tcpdump` wesentlich schneller macht. Dazu müssen Sie es mit der Option `-w` aufrufen:

Protokollieren in eine Datei

```
# tcpdump -i eth0 -w eth0.pcap
```

Die Option `-n` bringt hier nichts, da die Namensauflösung bei der Ausgabe gemacht wird.

`tcpdump` bedient sich intern einer Bibliothek namens `libpcap` (kurz für *packet capture*). Da diverse Programme die Ausgabedateien von `tcpdump` (beziehungsweise `libpcap`) verarbeiten können, gibt man ihnen am besten Namen, die auf `.pcap` enden, damit klar ist, was in ihnen für Daten stehen.

libpcap

Anzeigen können Sie so eine Datei später mit etwas wie

```
$ tcpdump -r eth0.pcap
```

(Hierfür brauchen Sie dann keine `root`-Rechte mehr, wenn Sie die Datei als normaler Benutzer lesen dürfen.)

Eine typische Ausgabe von `tcpdump` könnte ungefähr so aussehen:

```
# tcpdump -ni eth0
tcpdump: listening on eth0
14:26:37.292993 arp who-has 192.168.0.100 tell 192.168.0.1
14:26:37.293281 arp reply 192.168.0.100 is-at 00:A0:24:56:E3:75
14:26:37.293311 192.168.0.1.35993 > 192.168.0.100.21: S ▷
 ◁ 140265170:140265170(0) ...
14:26:37.293617 192.168.0.100.21 > 192.168.0.1.35993: S ▷
 ◁ 135130228:135130228(0) ack 140265171 ...
14:26:37.293722 192.168.0.1.35993 > 192.168.0.100.21: . ack 1 ...
Strg + C                                          Abbruch des Programms
5 packets received by filter
0 packets dropped by kernel
```

Sie sehen hier zum Beispiel die IP-Adressen der beteiligten Rechner und die dazugehörigen Ports. Im weiteren Verlauf dieses Kapitels werden wir einige der Ausgabeformate von `tcpdump` noch genauer betrachten.

3.2.2 Filter

`tcpdump` ist gut darin, eine wahre Datenflut zu produzieren, obwohl Sie sich eigentlich vielleicht nur für eine ganz bestimmte Art von Netzverkehr interessieren – etwa Verbindungen zu Ihrem Web-Server. Zum Glück macht `tcpdump` es möglich, die Daten, die es protokollieren soll, detailliert über »Filter« zu beschreiben, die in einer Steuersprache auf der Kommandozeile angegeben werden können. Zum Beispiel beschränkt ein Aufruf wie

Tabelle 3.1: Suchkriterien in tcpdump (Auszug)

Element	Bedeutung
`host` ⟨*Rechner*⟩	Paket involviert die Station ⟨*Rechner*⟩ (Name oder IP-Adresse)
`src host` ⟨*Rechner*⟩	… als Absender
`dst host` ⟨*Rechner*⟩	… als Empfänger
`ether host` ⟨*Rechner*⟩	… Station gegeben durch MAC-Adresse oder Eintrag in `/etc/ethers`
`ether src` ⟨*Rechner*⟩	… als Absender (entsprechend: `ether dst`)
`gateway` ⟨*Name*⟩	Paket verwendet Station ⟨*Name*⟩ als Gateway (Ethernet-Frame ist an ⟨*Name*⟩ adressiert, aber IP-Absender- und Empfängeradresse gehören anderen Stationen)
`net` ⟨*Netz*⟩	Paket involviert Netz ⟨*Netz*⟩ (aus `/etc/networks` oder als CIDR-Adresse)
`src net` ⟨*Netz*⟩	… als Absender (entsprechend: `dst net`)
`port` ⟨*Dienst*⟩	Paket involviert Dienst ⟨*Dienst*⟩ (aus `/etc/services` oder als Portnummer)
`tcp port` ⟨*Dienst*⟩	… nur für TCP (entsprechend: `udp`)
`src port` ⟨*Dienst*⟩	… nur als Absender (entsprechend: `dst port`, `tcp src port`, …)
`ip`, `ip6`, `arp`, `rarp`, `stp`	Paket gehört zu einer der benannten Protokollfamilien
`ip proto` ⟨*Protokoll*⟩	Paket enthält Daten gemäß ⟨*Protokoll*⟩ – eins von (u. a.) `icmp`, `icmp6`, `esp`, `udp` oder `tcp`. (`icmp`, `tcp` und `udp` müssen als »`\icmp`« usw. geschrieben werden.)
`ip broadcast`	Paket ist ein IP-Broadcast-Datagramm
`ip multicast`	Paket ist ein IP-Multicast-Datagramm
`vlan` [⟨*VLAN-ID*⟩]	Paket gehört zu einem VLAN (bzw. VLAN ⟨*VLAN-ID*⟩)

```
# tcpdump -i eth0 host 192.168.0.1 and tcp port 80
```

die Ausgabe von `tcpdump` auf Daten, die an den TCP-Port 80 auf der Adresse `192.168.0.1` gerichtet sind oder von diesem ausgehen.

Suchkriterien

Tabelle 3.1 gibt einen Überblick über die wichtigsten Suchkriterien in der Steuersprache von `tcpdump`. Sie können solche Kriterien (wie oben angedeutet) mit `and` (oder `&&`) und `or` (oder `||`) logisch verknüpfen oder mit `not` (oder `!`) logisch umkehren.

Die Negation hat Vorrang vor den anderen beiden Operatoren. `and` und `or` haben den gleichen Vorrang (!) und werden von links nach rechts abgearbeitet. Außerdem dürfen Sie Klammern verwenden, um expliziten Vorrang zu erreichen.

Die textuellen Operatoren `and`, `or` und `not` haben gegenüber ihren aus der Programmiersprache C geliehenen Äquivalenten `&&`, `||` und `!` den Vorteil, dass Sie sie nicht vor der Shell verstecken müssen (die letzteren drei sind ja auch Operatoren in der Shell). Auf jeden Fall verstecken (mit »\« oder Anführungszeichen) müssen Sie die Klammern.

Wenn Sie einen Namen, eine Adresse oder ähnliches verwenden, ohne ein Schlüsselwort für ein Suchkriterium davorzusetzen, wird das zuletzt davor angegebene Schlüsselwort weiter benutzt. Das heißt, die Konstruktion

```
not host red and blue
```

ist äquivalent zu

```
not host red and host blue
```

Achtung: Das `not` bezieht sich hier nur auf `host red` und nicht auf `host read and host blue`. Das heißt, den de-Morgan'schen Regeln zum Trotz ist

```
not host red and host blue
```

nicht dasselbe wie

```
not \( host red or host blue \)
```

oder (kürzer)

```
not \( host red or blue \)
```

Außerdem sind logische Ausdrücke erlaubt, die auch etwas Rechnerei enthalten können. Unterstützt werden die vier Grundrechenarten (»+«, »-«, »*«, »/«) und bitweises Und und Oder (»&« und »|«, nicht zu verwechseln mit »&&« und »||«), ferner die gängigen Vergleichsoperatoren (»=«, »!=«, »<«, »<=«, »>«, »>=«) und ganze Zahlen in der Syntax der Programmiersprache C, also

logische Ausdrücke

```
6699
015053                                  Dasselbe in Oktal (Basis 8)
0x1a2b                                  Dasselbe in Hexadezimal (Basis 16)
```

Um die Sache überhaupt interessant zu machen, können Sie auch auf Paketinhalte zugreifen. Dazu müssen Sie einen Protokollnamen (etwa `ether`, `ip`, `tcp` oder `udp` – die komplette Liste steht in `pcap-filter`(7)) mit einem Versatz (in Oktetten) in eckigen Klammern angeben. Dieser Versatz bezieht sich immer auf den Anfang des entsprechenden Kopfes in der Protokolldateneinheit – bei einem Ethernet-Frame, das ein IP-Datagramm enthält, das wiederum ein TCP-Segment transportiert, ist `tcp[0]` also das erste Oktett des TCP-Segments, `ip[1]` das zweite Oktett des IP-Datagramms und `ether[2]` das zweite Oktett des Ethernet-Frames. Außer dem Versatz können Sie noch eine Länge angeben (`1`, `2` oder `4`), die vom Versatz mit einem Doppelpunkt getrennt werden muss; fehlt sie, wird die Länge `1` angenommen. Zum Beispiel:

Paketinhalte

```
ether[0] & 1 = 1                        Ethernet-Broadcast-Frame
ip[16] >= 224                           IP-Multicast-Datagramm
ip[2:2] <= 512                          IP-Datagramm, maximal 512 Oktette lang
tcp[8:4]                                Bestätigungsnummer eines TCP-Segments
```

Für einige Paketinhalte gibt es auch symbolische Namen, zum Beispiel für den ICMP-Datagrammtyp und seine Werte und für die TCP-Flags. Sie können zum Beispiel `ping`-Pakete mit etwas wie

symbolische Namen

```
icmp[icmptype] = icmp-echo or icmp[icmptype] = icmp-echoreply
```

identifizieren. Die Details stehen in `pcap-filter`(7).

Hier sind noch ein paar Beispiele:

- Der Datenverkehr zwischen dem Rechner `red` und entweder `blue` oder `green`:

  ```
  host red and \( blue or green \)
  ```

- Aller DNS- und HTTP-Datenverkehr, der über das Gateway `router` fließt:

  ```
  gateway router and (port domain or tcp port http)
  ```

  (Die Dienstnamen kommen aus `/etc/services`.)

- Alle ICMP-Pakete außer denen von `ping`:

```
icmp[icmptype] != icmp-echo ▷
 ◁ and icmp[icmptype] != icmp-echoreply
```

(Der Umstand, dass Sie mit `icmp` auf Paketinhalte zugreifen, impliziert schon, dass das Paket den passenden Typ haben muss. Eine Bedingung wie

```
icmp[icmptype] = icmp-echo and arp[0] = 123
```

könnte niemals zutreffen, weil dasselbe Paket nicht gleichzeitig ein ICMP- und ein ARP-Datagramm sein kann.)

3.2.3 Ausgabe

Standardfall

Die Ausgabe von `tcpdump` läßt sich auch in vielerlei Hinsicht beeinflussen. Aber betrachten wir zunächst den Standardfall. Hier ist die erste Zeile des Protokolls einer einfachen HTTP-Anfrage, mit nachträglich eingebauten Zeilenumbrüchen zur besseren Erklärung. Im Beispiel kommen die Rechner `red.example.com` und `www.example.com` vor. `red.example.com` (der »Client«) ruft eine Seite vom Web-Server `www.example.com` (dem »Server«) ab:

```
14:46:15.124660 IP                                          Zeitstempel
  red.example.com.43633 > www.example.com.www:              Stationen
  Flags [S],                                                TCP-Flags
  seq 3628560156, win 5840,                                 TCP-Folgenummer, Fenstergröße
  options [mss 1460,sackOK,                                 Optionen
           TS val 4563811 ecr 0,
           nop,wscale 5],
  length 0
```

Der Zeitstempel gibt an, wann das Paket gelesen wurde (dies ist nicht Teil des Pakets). Die mit »Stationen« markierte Zeile enthält den Absender und den Empfänger des Pakets, jeweils als FQDN mit der Portnummer (43633 auf der Absenderseite, `www` oder 80 auf der Empfängerseite. »`Flags [S]`« steht für das gesetzte `SYN`-Flag und kennzeichnet das Paket so als erstes Segment eines TCP-Drei-Wege-Handshakes. »`seq 3628560156`« ist die anfängliche Folgenummer für diese Verbindung, und »`win 5840`« weist auf die TCP-Fenstergröße hin.

Der Client schickt eine Fenstergröße von 5840 und weist damit darauf hin, dass er bereit ist, bis zu 5840 Oktetten Daten zu empfangen.

Die TCP-Optionen in `options` sind für unsere Zwecke nicht so wichtig; »`mss 1460`« gibt zum Beispiel an, dass der Absender TCP-Segmente bis zu einer Maximalgröße von 1460 Oktetten empfangen kann. (Siehe hierzu auch Übung 3.1.) »`length 0`« wiederum ist die Länge der »Nutzlast« des Segments, hier 0 (wir fangen ja gerade erst an).

Das zweite Paket dieser Konversation, geschickt vom Server, sieht so aus:

```
14:46:15.124846 IP                                          Zeitstempel
  www.example.com.www > red.example.com.43633:              Stationen
  Flags [S.],                                               TCP-Flags
  seq 1610138470, ack 3628560157, win 14480,
  options [mss 1460,sackOK,
           TS val 8839719 ecr 4563811,
           nop,wscale 7], length 0
```

Die ersten Zeilen entsprechen sinngemäß denen aus dem vorigen Beispiel (mit vertauschten Rollen). In »`Flags [S.]`« steht der Punkt dafür, dass das `ACK`-Flag gesetzt war; entsprechend enthält die Ausgabe auch »`ack 3628560157`« als Bestätigung der »`syn`«-Folgenummer des Clients.


Die Folgenummer des Clients war 3628560156, aber die Bestätigung ist immer um 1 höher. Das hat also seine Richtigkeit.

Mit dem dritten Paket ist die Verbindung komplett aufgebaut:

```
14:46:15.125042 IP
  red.example.com.43633 > www.example.com.www:
  Flags [.],
  ack 1, win 183,
  options [nop,nop,TS val 4563811 ecr 8839719],
  length 0
```

In der Ausgabe von tcpdump ergeben sich dabei einige Veränderungen:

- Die Folgenummern bei seq und ack werden nur noch relativ zu den anfänglich ausgetauschten Folgenummern angegeben. Das »ack 1« in der Ausgabe von tcpdump entspricht also dem Wert 1610138471 im tatsächlichen TCP-Segment. Folgenummern
- Die Fenstergröße 183 sieht verdächtig klein aus. Das ist allerdings völlig in Ordnung so, da die beiden Kommunikationspartner *window scaling* gemäß [RFC1323] verwenden (was man an der wscale-Option bei den SYN-Segmenten sehen kann). Der Client hatte ursprünglich ein »wscale 5«, die tatsächliche Fenstergröße ist also $183 \cdot 2^5 = 5856$, und das ist komfortabel in der Nähe des ursprünglichen Werts 5840. Fenstergröße

Im nächsten Schritt sendet der Client die tatsächliche HTTP-Anfrage:

```
14:46:15.125262 IP red.example.com.43633 > www.example.com.www:
  Flags [P.],
  seq 1:114, ack 1, win 183,
  options [nop,nop,TS val 4563811 ecr 8839719], length 113
```

Hier ist das P-Flag (kurz für PUSH) gesetzt. Die Folgenummer »seq 1:114« ist als »Oktett 1 bis unmittelbar vor Oktett 114« zu lesen, was mit einer Nutzlast-Länge von 113 Oktetten konsistent ist. Neue Daten vom Server, die zu bestätigen wären, sind keine eingegangen, darum haben wir immer noch »ack 1«.


Das PUSH-Flag bedeutet, dass die übertragenen Daten vom Empfänger möglichst direkt an den verarbeitenden Prozess weitergereicht werden sollen. Normalerweise setzt die TCP-Implementierung des Absenders es automatisch, wenn der Puffer mit zu versendenden Daten komplett geleert wird. Auf der Empfängerseite wird das PUSH-Flag von Linux ignoriert, weil die Daten sowieso in jedem Fall so schnell wie möglich an den verarbeiteten Prozess geschickt werden. PUSH-Flag

Ausgabeformat Bisher haben wir uns nur mit den Daten aus dem TCP- und dem IP-Kopf des Pakets befasst, und zwar in der von tcpdump aufbereiteten Form. Es gibt aber diverse Möglichkeiten, das Ausgabeformat von tcpdump zu ändern. Wenn es Ihnen nur auf den schnellen Überblick ankommt, können Sie zum Beispiel die Option »-q« verwenden, die kurze Zeilen mit den allerwichtigsten Informationen liefert: Überblick

```
$ tcpdump -q -r http.pcap
reading from file http.pcap, link-type EN10MB (Ethernet)
14:46:15.124660 IP red.example.com.43633 > www.example.com.www: tcp 0
14:46:15.124846 IP www.example.com.www > red.example.com.43633: tcp 0
14:46:15.125042 IP red.example.com.43633 > www.example.com.www: tcp 0
14:46:15.125262 IP red.example.com.43633 > www.example.com.www: tcp 113
14:46:15.125383 IP www.example.com.www > red.example.com.43633: tcp 0
```

```
14:46:15.125953 IP www.example.com.www > red.example.com.43633: tcp 491
14:46:15.125987 IP red.example.com.43633 > www.example.com.www: tcp 0
14:46:15.126779 IP red.example.com.43633 > www.example.com.www: tcp 0
14:46:15.127007 IP www.example.com.www > red.example.com.43633: tcp 0
14:46:15.127028 IP red.example.com.43633 > www.example.com.www: tcp 0
```

Mit der Option »-e« gibt tcpdump auch noch die MAC-Adressen und andere Informationen aus dem Ethernet-Frame aus:

```
$ tcpdump -e -r http.pcap
reading from file http.pcap, link-type EN10MB (Ethernet)
14:46:15.124660 08:00:27:a6:4b:3b (oui Unknown) > 0a:00:27:00:00:00▷
 ◁ (oui Unknown), ethertype IPv4 (0x0800), length 74:▷
 ◁ red.example.com.43633 > www.example.com.www: Flags [S], ▷
 ◁ seq 3628560156, win 5840, options [mss 1460,sackOK,TS ▷
 ◁ val 4563811 ecr 0,nop,wscale 5], length 0
14:46:15.124846 0a:00:27:00:00:00 (oui Unknown) > 08:00:27:a6:4b:3b▷
 ◁ (oui Unknown), ethertype IPv4 (0x0800), length 74:▷
 ◁ www.example.com.www > red.example.com.43633: Flags [S.], ▷
 ◁ seq 1610138470, ack 3628560157, win 14480, options ▷
 ◁ [mss 1460,sackOK,TS val 8839719 ecr 4563811,nop,wscale 7], length 0
◁◁◁◁◁
```

Daten

Die Option »-x« liefert die Daten in hexadezimaler Form:

```
$ tcpdump -x -r http.pcap
reading from file http.pcap, link-type EN10MB (Ethernet)
14:46:15.124660 IP red.example.com.43633 > www.example.com.www:▷
 ◁ Flags [S], seq 3628560156, win 5840, options [mss 1460,sackOK,▷
 ◁ TS val 4563811 ecr 0,nop,wscale 5], length 0
        0x0000:  4500 003c cc7f 4000 4006 7c85 c0a8 3865
        0x0010:  c0a8 3801 aa71 0050 d847 6f1c 0000 0000
        0x0020:  a002 16d0 a9ac 0000 0204 05b4 0402 080a
        0x0030:  0045 a363 0000 0000 0103 0305
14:46:15.124846 IP www.example.com.www > red.example.com.43633:▷
 ◁ Flags [S.], seq 1610138470, ack 3628560157, win 14480,▷
 ◁ options [mss 1460,sackOK,TS val 8839719 ecr 4563811,nop,wscale 7],▷
 ◁ length 0
        0x0000:  4500 003c 0000 4000 4006 4905 c0a8 3801
        0x0010:  c0a8 3865 0050 aa71 5ff8 c366 d847 6f1d
        0x0020:  a012 3890 81cc 0000 0204 05b4 0402 080a
        0x0030:  0086 e227 0045 a363 0103 0307
◁◁◁◁◁
```

Mit etwas Kopfrechnen und Wissen über die Formate von IP-Datagrammen und TCP-Segmenten können Sie in den hexadezimalen Daten die von tcpdump dekodierten Informationen wiederfinden. Zum Beispiel sind die letzten vier Oktette in der ersten Zeile die IP-Absenderadresse und die ersten vier in der zweiten Zeile die Adresse des Empfängers.

ASCII-Version

Schließlich können Sie mit »-X« außer der hexadezimalen Darstellung auch noch eine ASCII-Version anzeigen. Hier ist zur Abwechslung mal ein Paket aus der Mitte der Konversation:

```
◁◁◁◁◁
14:46:15.125953 IP www.example.com.www > red.example.com.43633:▷
 ◁ Flags [P.], seq 1:492, ack 114, win 114,▷
 ◁ options [nop,nop,TS val 8839720 ecr 4563811], length 491
    0x0000:  4500 021f 2150 4000 4006 25d2 c0a8 3801  E...!P@.@.%...8.
```

```
    0x0010:  c0a8 3865 0050 aa71 5ff8 c367 d847 6f8e  ..8e.P.q_..g.Go.
    0x0020:  8018 0072 c786 0000 0101 080a 0086 e228  ...r...........(
    0x0030:  0045 a363 4854 5450 2f31 2e31 2032 3030  .E.cHTTP/1.1.200
    0x0040:  204f 4b0d 0a44 6174 653a 2054 6875 2c20  .OK..Date:.Thu,.
    0x0050:  3035 204d 6179 2032 3031 3120 3132 3a34  05.May.2011.12:4
    0x0060:  373a 3339 2047 4d54 0d0a 5365 7276 6572  7:39.GMT..Server
    0x0070:  3a20 4170 6163 6865 2f32 2e32 2e31 3720  :.Apache/2.2.17.
    0x0080:  2844 6562 6961 6e29 0d0a 4c61 7374 2d4d  (Debian)..Last-M
    0x0090:  6f64 6966 6965 643a 2046 7269 2c20 3136  odified:.Fri,.16
    0x00a0:  2041 7072 2032 3031 3020 3230 3a35 333a  .Apr.2010.20:53:
⊲⊲⊲⊲⊲
```

Dekodierung Es gibt auch ein paar Optionen, die die Dekodierung der Daten durch tcpdump beeinflussen. Mit der Option »-n« können Sie die Namensauflösung unterdrücken. Das heißt, Stationen erscheinen als IP-Adressen statt Namen, und statt Dienstnamen werden Portnummern eingesetzt:

Namensauflösung unterdrücken

```
$ tcpdump -n -r http.pcap
reading from file http.pcap, link-type EN10MB (Ethernet)
14:46:15.124660 IP 192.168.56.101.43633 > 192.168.56.1.80:▷
 ◁ Flags [S], seq 3628560156, win 5840, options [mss 1460,sackOK,▷
 ◁ TS val 4563811 ecr 0,nop,wscale 5], length 0
14:46:15.124846 IP 192.168.56.1.80 > 192.168.56.101.43633:▷
 ◁ Flags [S.], seq 1610138470, ack 3628560157, win 14480, options ▷
 ◁ [mss 1460,sackOK,TS val 8839719 ecr 4563811,nop,wscale 7], length 0
```

Die Namensauflösung kann tcpdump ziemlich aufhalten. Wenn Sie eine .pcap-Datei analysieren, ist das egal, aber wenn Sie tcpdump den Verkehr »live« dekodieren und das Resultat in eine Datei schreiben lassen, kann das durchaus ein bremsender ein Faktor sein.

Außerdem produziert die Rückwärtsauflösung von IP-Adressen in Namen möglicherweise DNS-Datenverkehr, der dann auch dekodiert wird und die Ausgabe von tcpdump zumüllt. (Ein weiteres Argument für das Anlegen von .pcap-Dateien.)

Die Option »-N« unterdrückt Domainnamen in der Ausgabe:

Domainnamen

```
$ tcpdump -N -r http.pcap
reading from file http.pcap, link-type EN10MB (Ethernet)
14:46:15.124660 IP red.43633 > www.www: Flags [S], seq 3628560156,▷
 ◁ win 5840, options [mss 1460,sackOK,TS val 4563811 ecr 0,▷
 ◁nop,wscale 5], length 0
14:46:15.124846 IP www.www > red.43633: Flags [S.], seq 1610138470,▷
 ◁ ack 3628560157, win 14480, options [mss 1460,sackOK,TS val 8839719▷
 ◁ ecr 4563811,nop,wscale 7], length 0
```

Die Optionen »-v«, »-vv« und »-vvv« bestimmen den Umfang der Dekodierung. Mit »-v« werden zum Beispiel auch Informationen aus dem IP-Kopf wie die TTL und die IP-Optionen angezeigt und die IP-Prüfsumme überprüft:

Umfang der Dekodierung

```
$ tcpdump -Nv -r http.pcap
reading from file http.pcap, link-type EN10MB (Ethernet)
14:46:15.124660 IP (tos 0x0, ttl 64, id 52351, offset 0, flags [DF],▷
 ◁ proto TCP (6), length 60)
  red.43633 > www.www: Flags [S], cksum 0xa9ac (correct),▷
 ◁   seq 3628560156, win 5840, options [mss 1460,sackOK,▷
 ◁   TS val 4563811 ecr 0,nop,wscale 5], length 0
```

Je mehr »v« Sie angeben, um so umfangreicher wird die Ausgabe. Mit »-vv« liefert tcpdump zum Beispiel eine ausführliche Dekodierung von SMB-Paketen. Details hierzu stehen in tcpdump(8).


Das Programm wireshark (siehe Abschnitt 3.3) kann noch viel mehr Protokolle dekodieren.

Zeitstempel Zum Schluss noch ein Hinweis auf die verschiedenen Möglichkeiten, die Ausgabezeilen von tcpdump mit Zeitstempeln zu versehen. Standardmäßig gibt tcpdump die Uhrzeit aus, zu der das Paket protokolliert wurde, aber es gibt die folgenden Optionen:

-t Gar keinen Zeitstempel ausgeben

-tt Die Zeit in Sekunden seit dem 1.1.1970, 0 Uhr UTC ausgeben (plus Kleinkram)

-ttt Den Zeitunterschied zwischen der aktuellen und der jeweils vorigen Zeile ausgeben

-tttt Das Standardformat (Uhrzeit), aber mit dem Datum am Zeilenanfang

-ttttt Den Zeitunterschied zwischen der aktuellen und der allerersten Zeile ausgeben

Hier zur Veranschaulichung jeweils die ersten drei Zeilen der Ausgabe in den verschiedenen Formaten:

```
$ tcpdump -Nt -c3 -r http.pcap                                    Kein Zeitstempel
IP red.43633 > www.www: Flags [S], seq 3628560156, win 58…
IP www.www > red.43633: Flags [S.], seq 1610138470, ack 3…
IP red.43633 > www.www: Flags [.], ack 1, win 183, option…
$ tcpdump -Ntt -c3 -r http.pcap                                   Unix-Zeit
1304599575.124660 IP red.43633 > www.www: Flags [S], seq …
1304599575.124846 IP www.www > red.43633: Flags [S.], seq…
1304599575.125042 IP red.43633 > www.www: Flags [.], ack …
$ tcpdump -Nttt -c3 -r http.pcap                                  Delta
00:00:00.000000 IP red.43633 > www.www: Flags [S], seq 36…
00:00:00.000186 IP www.www > red.43633: Flags [S.], seq 1…
00:00:00.000196 IP red.43633 > www.www: Flags [.], ack 1,…
$ tcpdump -Ntttt -c3 -r http.pcap                                 Datum und Zeit
2011-05-05 14:46:15.124660 IP red.43633 > www.www: Flags …
2011-05-05 14:46:15.124846 IP www.www > red.43633: Flags …
2011-05-05 14:46:15.125042 IP red.43633 > www.www: Flags …
$ tcpdump -Nttttt -c3 -r http.pcap                                Delta vom Anfang
00:00:00.000000 IP red.43633 > www.www: Flags [S], seq 36…
00:00:00.000186 IP www.www > red.43633: Flags [S.], seq 1…
00:00:00.000382 IP red.43633 > www.www: Flags [.], ack 1,…
```

Übungen


3.1 [2] Warum ist 1460 eine naheliegende MSS für TCP? Welcher Zusammenhang besteht zwischen der MSS und der maximalen Fenstergröße?


3.2 [2] Angenommen, Sie sind vom Rechner red.example.com aus über die SSH auf dem Rechner www.example.com angemeldet und möchten den Netzverkehr auf www.example.com protokollieren. Wie erreichen Sie, dass die Daten Ihrer SSH-Verbindung nicht mit protokolliert werden? Geben Sie ein passendes tcpdump-Kommando an.


3.3 [2] Wie können Sie von allen von Ihrem Rechner ausgehenden TCP-Verbindungen das erste Paket protokollieren? Geben Sie ein passendes tcpdump-Kommando an.

3.4 [2] Verwenden Sie `tcpdump`, um eine FTP-Sitzung zu protokollieren (wenn Sie keinen FTP-Server zur Hand haben oder installieren können, dann verwenden Sie einen geeigneten öffentlich zugänglichen anonymen FTP-Server). Lassen Sie sich dabei auch die Nutzdaten (nicht nur die IP- und TCP-Kopfdaten) ausgeben, etwa mit der Option »`-X`«. Überzeugen Sie sich, dass Informationen wie Benutzername und Kennwort im Klartext übertragen werden.

3.3 wireshark

3.3.1 Grundlagen

Wie `tcpdump` ist auch `wireshark` ein Paketsniffer. Im Gegensatz zu `tcpdump` verfügt `wireshark` aber über eine grafische Oberfläche und einen deutlich größeren Funktionsumfang.

Das Programm `wireshark` hieß früher `ethereal` und ist auf älteren Rechnern möglicherweise noch unter diesem Namen zu finden. Auch manche Webseiten beziehen sich noch auf den alten Namen.

Für Situationen, wo Sie keinen Grafikbildschirm zur Verfügung haben, gibt es unter dem Namen `tshark` auch eine konsolenbasierte Version von `wireshark`.

Nach dem Start von `wireshark` bekommen Sie als Erstes einen Übersichtsbildschirm gezeigt (Bild 3.1), der Ihnen die verschiedenen Netzwerkschnittstellen des Systems zur Beobachtung anbietet und außerdem auf diverse Informationsquellen verweist.

Ob und wie Sie tatsächlich auf die Schnittstellen im *promiscuous mode* zugreifen können, hängt davon ab, wie `wireshark` bei Ihnen installiert ist. Grundsätzlich verwendet es ein Programm namens `dumpcap`, um mit den Netzwerkschnittstellen zu reden. Dieses Programm kann mit den nötigen Rechten für den *promiscuous mode* aufgerufen werden, selbst wenn `wireshark` selbst mit den Rechten eines normalen Benutzers (wie Ihnen) läuft – typischerweise über den Linux-Capability-Mechanismus, aber grundsätzlich auch per Set-UID.

Widerstehen Sie der Versuchung, `wireshark` selbst als `root` aufzurufen. Zum Einen funktioniert das sowieso nicht ohne Weiteres (wenn Sie eine X11-Sitzung als normaler Benutzer laufen haben, brauchen Sie Tricks, damit als `root` gestartete Programme auf die Sitzung zugreifen können), und zum Anderen taucht `wireshark` immer wieder in CERT-Advisories und den Sicherheits-Update-Listen der Distributionen auf, weil Sicherheitslücken gefunden werden. Zum größten Teil stecken die zwar in den Zerpflück-Routinen für diverse mehr oder weniger obskure Paketformate, aber man weiß ja nie …

Dass Sie nie auf die Idee kommen würden, eine Sitzung auf der grafischen Oberfläche als `root` zu starten, nur damit Sie `wireshark` aufrufen können, setzen wir hier mal voraus.

Bei Debian GNU/Linux können Sie während der Installation des `wireshark`-Pakets wählen, ob Sie den Zugriff auf das `dumpcap`-Programm nur `root` oder auch den Mitgliedern der Gruppe `wireshark` erlauben wollen. Letzteres ist sinnvoll, wenn `wireshark` den Datenverkehr gleichzeitig lesen und anzeigen soll – ansonsten wird empfohlen, `dumpcap` als `root` laufen zu lassen und die dabei entstehenden Protokolle später mit einem unter den Rechten eines gewöhnlichen Benutzers laufenden `wireshark` zu analysieren. Das Verhalten können Sie jederzeit mit

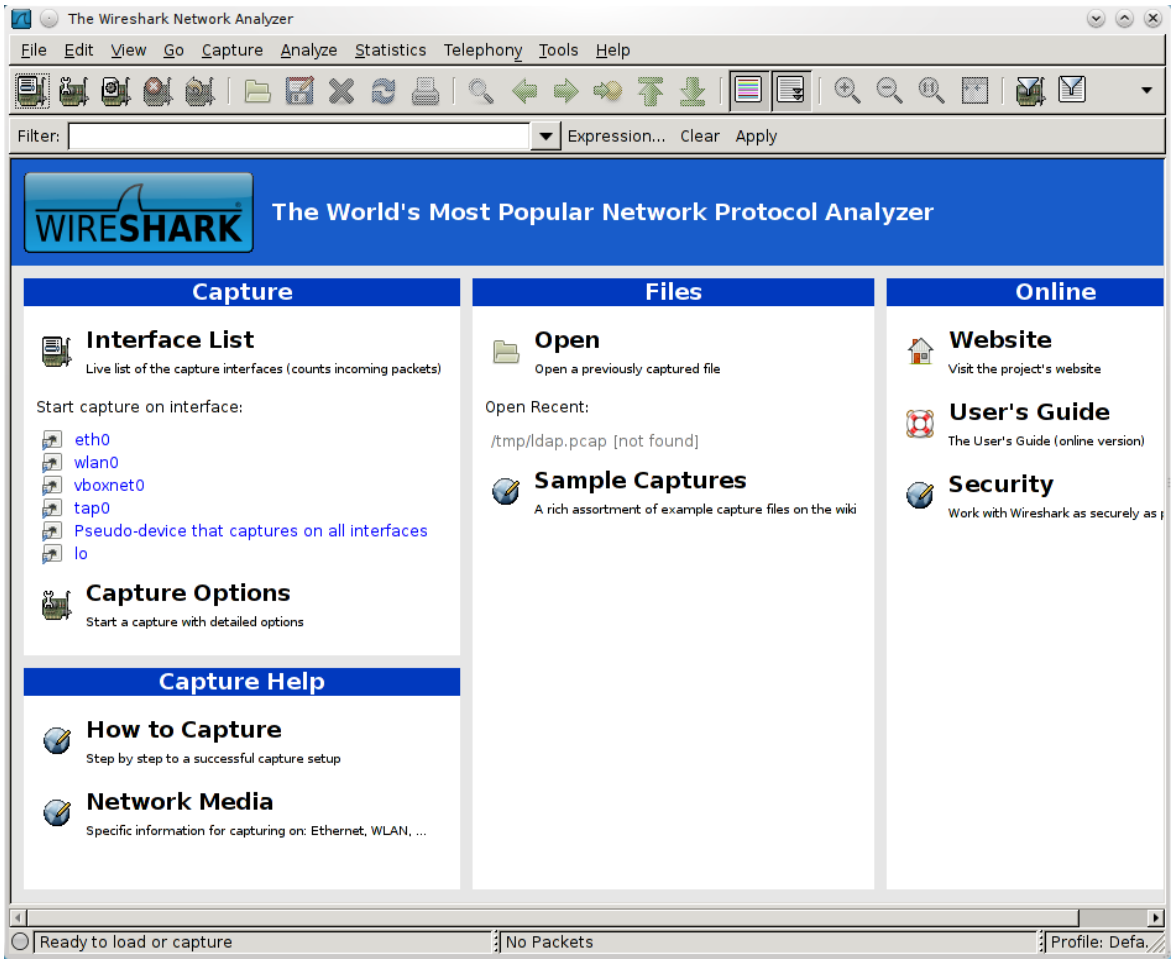


Bild 3.1: Das Programm wireshark: Startbildschirm

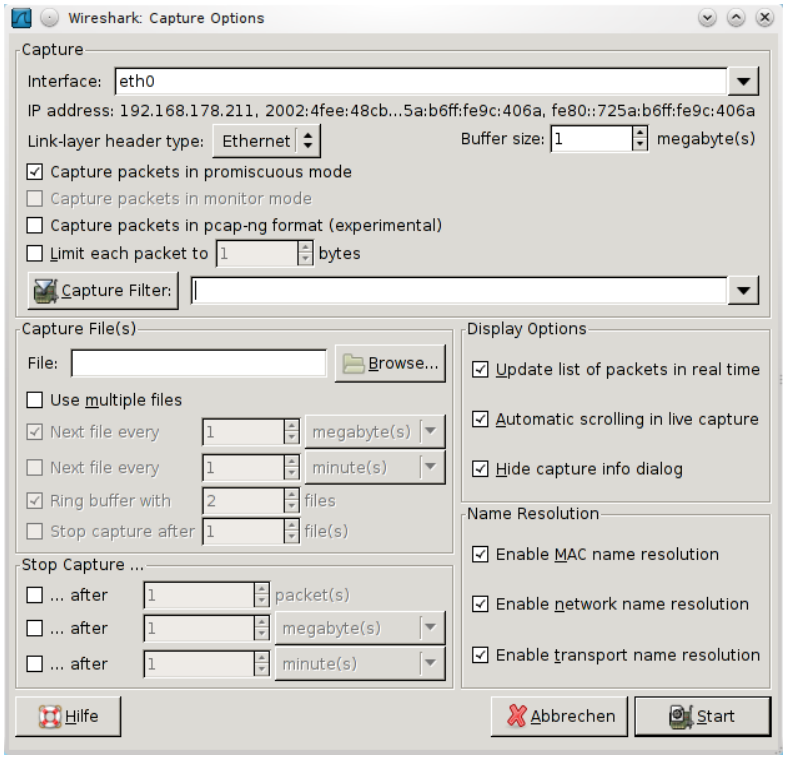


Bild 3.2: Das Programm `wireshark`: Optionen für das Protokollieren

```
# dpkg-reconfigure wireshark-common
```

neu konfigurieren.

3.3.2 Netzwerkdaten protokollieren

Vom Startbildschirm aus können Sie entweder direkt eine Netzwerkschnittstelle beobachten (indem Sie auf deren Namen klicken) oder den detaillierten Dialog zur Konfiguration des Protokollierens aufrufen (Bild 3.2). Hier einige Anmerkungen zu diesem Dialog: Dialog

- Im oberen Teil des Dialogs können Sie eine Netzwerkschnittstelle wählen. Der Inhalt des entsprechenden Menüs ist identisch mit der auf dem Startbildschirm angebotenen Liste. Unter dem Menü sehen Sie die IP-Adresse(n), die für die betreffende Schnittstelle konfiguriert sind, was die Auswahl der richtigen Schnittstelle erleichtern sollte.

  Die weiteren Parameter in diesem Teil des Dialogs sind relativ vernünftig voreingestellt. Erwähnenswert ist das Feld »Capture Filter«, über das Sie festlegen können, welche Pakete überhaupt protokolliert werden. Damit können Sie eine Vorauswahl treffen und gegebenenfalls riesige unnötige Datenmengen vermeiden. Capture Filter

  
  »Capture Filter« in `wireshark` werden mit derselben Syntax definiert wie die Filterausdrücke von `tcpdump`. Alles, was wir in Abschnitt 3.2.2 gesagt haben, ist also auch hier anwendbar.

  
  Netterweise erlaubt `wireshark` das Abspeichern und Wiederbenutzen eines einmal konfigurierten »Capture Filter« unter einem beschreibenden Namen. Auf diese Weise können Sie sich mit der Zeit einen bequemen »Werkzeugkasten« anlegen.

Datei für das Protokoll

- Links unterhalb des Auswahlbereichs für die Netzwerkschnittstelle können Sie eine Datei für das Protokoll festlegen. Dabei ist es auch möglich, mehrere Dateien nacheinander zu verwenden (sogar als »Ringpuffer«, bei dem immer die älteste noch übrige Datei mit neuen Daten überschrieben wird) und dafür festzulegen, ob eine neue Datei nach einer bestimmten Zeit oder einer bestimmten protokollierten Datenmenge angefangen wird.

Beenden des Protokolls

- Unter diesem Bereich können Sie wireshark sagen, dass die Protokollierung selbst nach einer gewissen Anzahl von Paketen, einem gewissen Datenvolumen oder einer bestimmten Zeitspanne beendet werden soll.

Live-Protokoll

- Im rechten unteren Teil des Dialogs können Sie unter anderem festlegen, ob wireshark neue eingehende Pakete sofort (»live«) anzeigen soll und ob eine sofortige Auflösung von MAC-Adressen (*MAC name resolution*), IP-Adressen (*network name resolution*) oder TCP/UDP-Portnummern (*transport name resolution*) stattfinden soll.

  Genau wie bei tcpdump gilt, dass eine sofortige Auflösung von IP-Adressen bei der »Live«-Protokollierung zu merklichen Verzögerungen führen kann. Um diesen Effekt abzumildern, erlaubt wireshark eine *concurrent DNS name resolution*, bei der das Programm gleichzeitig mehrere Anfragen ans DNS stellt, ohne jeweils auf die korrespondierende Antwort zu warten. Statt dessen fährt es mit seiner eigentlichen Arbeit vor, und sobald Antworten vom DNS eingehen, werden diese verarbeitet. Diese Vorgehensweise ist standardmäßig eingeschaltet; kontrollieren können Sie sie über die »Name Resolution«-Karte des »Preferences«-Dialogs, den Sie über den untersten Eintrag im »Edit«-Menü aufrufen können.

- Die Schaltfläche »Start« beginnt mit der Protokollierung.

Sammelvorgang anhalten

Wenn Sie nicht »live« protokollieren, sehen Sie erst einmal nichts außer der Meldung, dass wireshark jetzt Daten sammelt – Sie müssen den Sammelvorgang entweder explizit anhalten (über »Stop« im Menü »Capture« oder das korrespondierende Icon in der Werkzeugleiste) oder, falls Sie eine Zeit- oder Datenmengengrenze angegeben haben, warten, bis diese erreicht ist.

Wenn Sie im Dialog mit den Protokollierungs-Optionen das Häkchen bei »Hide capture info dialog« entfernt haben, bekommen Sie während des Protokollierens ein Fenster angezeigt, das so ähnlich aussehen sollte wie Bild 3.3 (soviel zum Thema »Doppelte Verneinung«). Damit können Sie den Fortgang des Protokollierens im Groben beobachten und (mit »Stop«) die Protokollierung auch anhalten.

3.3.3 Das Hauptfenster

Nach (oder während, wenn »live«) der Protokollierung zeigt wireshark Ihnen in seinem Hauptfenster (Bild 3.4) die gewonnenen Daten an. Das Hauptfenster besteht – wenn wir die Menüleiste am oberen Rand und die Werkzeugleiste direkt darunter mal als gegeben hinnehmen – aus drei Teilen:

- Direkt unter der Werkzeugliste steht eine Liste der protokollierten Pakete. Hier tauchen alle Pakete auf, die von der beobachteten Netzwerkschnittstelle gelesen wurden (und gegebenenfalls den »Capture Filter« passiert haben). Sie können einzelne Pakete genauer anschauen, indem Sie mit der linken Maustaste auf die betreffende Zeile klicken.

  Wenn Sie auf den Titel einer Spalte klicken, können Sie die Liste nach dem Inhalt der betreffenden Spalte sortieren. Ob das in jedem Fall Sinn ergibt, ist natürlich eine andere Frage.

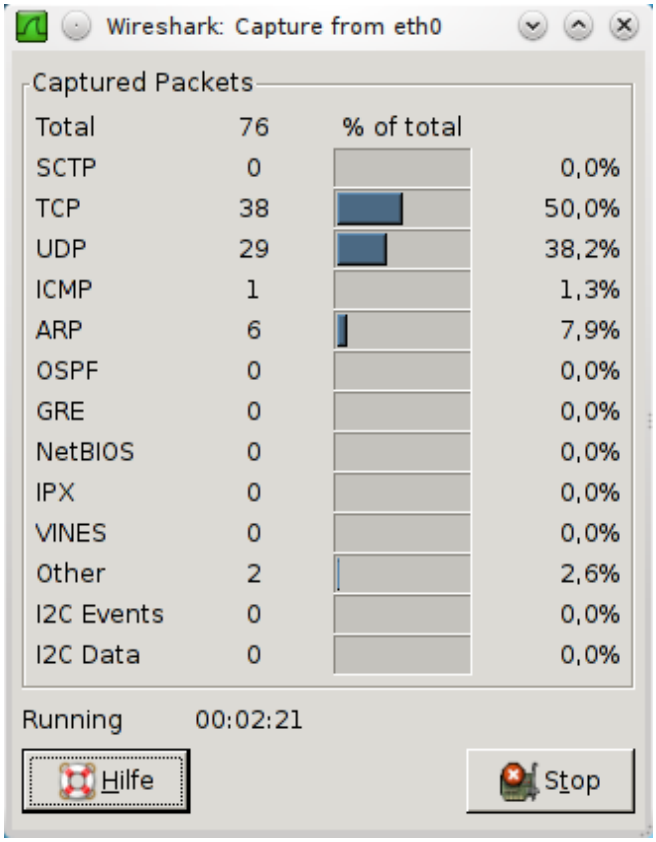


Bild 3.3: Das Programm wireshark: Protokollierung

Wenn Sie mit der rechten Maustaste auf den Titel einer Spalte klicken, dann bekommen Sie ein Menü mit verschiedenen Aktionen.

- Unter der Paketliste erscheint ein Fenster, das eine sehr detaillierte Darstellung eines ausgewählten Pakets bietet. Dabei wird die Protokollhierarchie verdeutlicht.

  Ursprünglich zeigt wireshark eine sehr kompakte Darstellung, bei der jede Protokollebene eine Zeile einnimmt. Details über die jeweiligen Protokolldateneinheiten im Paket erhalten Sie, indem Sie auf das Dreieck am linken Rand der Zeile klicken und damit zusätzliche Informationen »ausklappen«. Manchmal gibt es sogar noch mehr Details, wobei immer die Regel gilt, dass in der ersten (aufklappbaren) Zeile eine kompakte Zusammenfassung steht.

  Hier ein Beispiel: Jedes empfangene Paket (jedenfalls auf einem Ethernet) gilt erst einmal als »Frame«, dann als »Ethernet II«:

```
▷ Frame 12: 180 bytes on wire (1440 bits), 180 bytes captured (…)
▽ Ethernet II, Src: 192.168.61.2 (08:00:27:c4:a9:f8), Dst: 192.168.61.1 (…)
  ▽ Destination: blue.example.com (08:00:27:c9:a4:66)
      Address: blue.example.com (0a:00:27:00:00:00)
      .... ...0 .... .... .... .... = IG bit: Individual address
      .... ..0. .... .... .... .... = LG bit: Globally unique address
  ▽ Source: 192.168.61.2 (08:00:27:c4:a9:f8)
      Address: 192.168.61.2 (08:00:27:c4:a9:f8)
      .... ...0 .... .... .... .... = IG bit: Individual address
      .... ..0. .... .... .... .... = LG bit: Globally unique address
    Type: IP (0x0800)
```

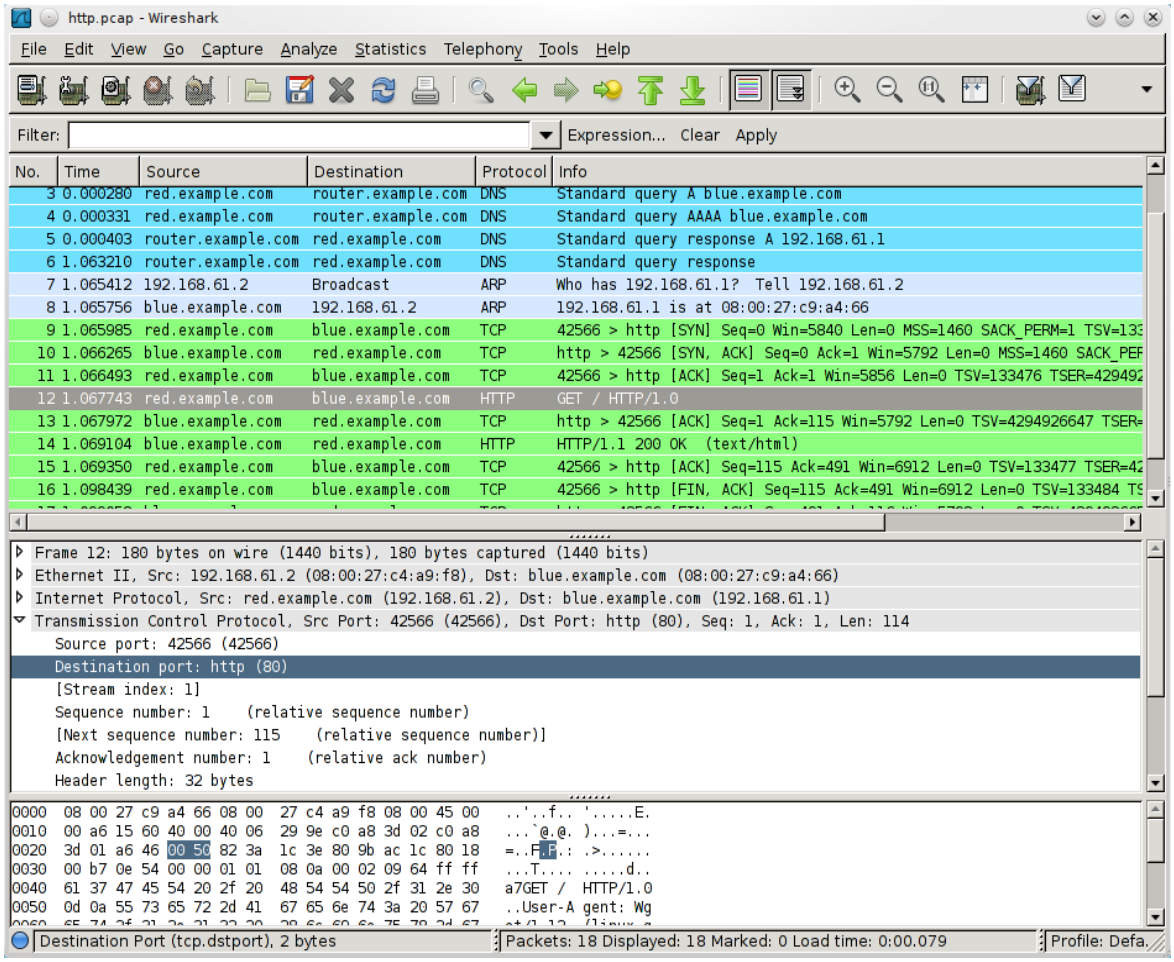


Bild 3.4: Das Programm wireshark: Hauptfenster

Das Ethernet-Frame enthält dann ein IP-Datagramm:

```
▽ Internet Protocol, Src: red.example.com (192.168.61.2),▷
                          ◁ Dst: blue.example.com (192.168.61.1)
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x00: Default;▷
                          ◁ ECN: 0x00)
    Total Length: 166
    Identification: 0x1560 (5472)
  ▽ Flags: 0x02 (Don't Fragment)
      0... .... = Reserved bit: Not set
      .1.. .... = Don't fragment: Set
      ..0. .... = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
  ▷ Header checksum: 0x299e [correct]
    Source: red.example.com (192.168.61.2)
    Destination: blue.example.com (192.168.61.1)
```

Und dies wiederum ein TCP-Segment:

```
▽ Transmission Control Protocol, Src Port: 42566 (42566),▷
                 ◁ Dst Port: http (80) Seq: 1, Ack: 1, Len: 114
    Source port: 42566 (42566)
    Destination port: http (80)
    [Stream index: 1]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 115    (relative sequence number)]
    Acknowledgement number: 1    (relative ack number)
    Header length: 32 bytes
  ▽ Flags: 0x18 (PSH, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgement: Set
      .... .... 1... = Push: Set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
    Window size: 5856 (scaled)
  ▷ Checksum: 0x0e54 [validation disabled]
  ▷ Options: (12 bytes)
  ▷ [SEQ/ACK analysis]
```

(Bemerken Sie, dass wireshark genau wie tcpdump die TCP-Folge- und -Bestätigungsnummern relativ zu den beim Verbindungsaufbau ausgehandelten Startwerten angibt.)

Schließlich zeigt wireshark sogar an, dass es sich um HTTP handelt:

```
▽ Hypertext Transfer Protocol
  ▽ GET / HTTP/1.0\r\n
    ▷ [Expert Info (Chat/Sequence): GET / HTTP/1.0\r\n
        Request Method: GET
        Request URI: /
```

```
        Request Version: HTTP/1.0
    User-Agent: Wget/1.12 (linux-gnu)\r\n
    Accept: */*\r\n
    Host: blue.example.com\r\n
    Connection: Keep-Alive\r\n
    \r\n
```

wireshark kennt die Paketformate einiger hundert Protokolle, und die Chancen sind gering, dass Sie es im wirklichen Leben mit etwas zu tun bekommen, womit wireshark nichts anfangen kann. Mit »Enabled Protocols…« im Menü »Analyze« können Sie eine Liste der unterstützten Protokolle aufrufen und einzelne Protokolle gezielt von der Analyse ausnehmen (wobei das dann auch immer für die übergeordneten Protokolle gilt – wenn Sie zum Beispiel TCP abwählen, dann deaktiviert das implizit auch die Dekodierung aller anderen Protokolle, die auf TCP aufbauen, wie HTTP, SMTP, LDAP und so weiter).

- Im Bereich darunter wiederum sind die tatsächlichen Paketdaten in hexadezimaler Form und ASCII zu sehen. Wenn Sie auf eine Zeile in der detaillierten Darstellung des Pakets im Bereich darüber klicken, werden genau die entsprechenden Oktette hervorgehoben, im Beispiel in Bild 3.4 zum Beispiel die HTTP-Zielportnummer: »`00 50`« entspricht der dezimalen Zahl 80.

Sie können die einzelnen Bereiche vergrößern und verkleinern, indem Sie die »Trennlinien« mit der Maus verschieben. Die Trennlinien erkennen Sie an dem Punktmuster in der Mitte.

Wenn Ihnen die dreiteilige vertikale Anordnung der Bereiche nicht zusagt, können Sie sich mit der »User Interface/Layout«-Karte im »Preferences«-Dialog etwas Anderes aussuchen. Dies ist möglicherweise auf den bei Notebooks heute üblichen sehr breiten, aber nicht besonders hohen Displays nützlich.

3.3.4 Tipps, Tricks und Techniken

Hier noch ein paar Tipps für den Umgang mit wireshark:

»Filter«

Bestimmte Protokolle zeigen oder unterdrücken Oberhalb der Paketliste ist ein mit »Filter« gekennzeichnetes Feld. Hier können Sie Ausdrücke eingeben, die bestimmen, welche Pakete tatsächlich in der Paketliste auftauchen.

Bitte nicht mit dem »Capture Filter« aus dem Dialog für die Protokollierung verwechseln! Pakete, die nicht auf den »Capture Filter« passen, werden gar nicht erst protokolliert und können darum später auch nicht angeschaut werden. Der »Filter« im Hauptfenster dagegen betrifft nur die Anzeige, das heißt, die nicht passenden Pakete bleiben erhalten und werden nur nicht dargestellt.

Gemeinerweise verwendet der »Filter« im Hauptfenster eine völlig andere Syntax als der »Capture Filter«. Die beiden sind also nicht gegenseitig austauschbar.

Angenommen, Sie möchten sich nicht von lästigen ARP-Paketen verwirren lassen. In diesem Fall können Sie in das »Filter«-Feld den Text

```
not arp
```

eintragen und mit ↩ (oder einem Klick auf das danebenstehende »Apply«) bestätigen. Damit werden alle ARP-Pakete ausgeblendet.

Umgekehrt könnten Sie zum Beispiel

```
dns
```

eintragen und bekämen dann *nur* die DNS-Pakete gezeigt.

Bei der Eingabe von Filterausdrücken findet eine automatische Syntaxprüfung statt. Wenn der Inhalt des Feldes ein gültiger Filterausdruck ist, erscheint dessen Hintergrund grün, sonst rot. Bei einem gelben Hintergrund können »unerwartete Ergebnisse« auftauchen.

Syntaxprüfung

Wenn Sie weder ARP- noch DNS-Verkehr sehen möchten, hilft ein entschlossenes

```
not arp and not dns
```

oder auch (nach De Morgan und um damit zu protzen, dass es geht)

```
not (arp or dns)
```

Kompliziertere Filter Für Filterausdrücke kommen nicht nur Protokollnamen in Frage, sondern alle möglichen Eigenschaften von Paketen, inklusive einzelne Felder von Protokolldateneinheiten diverser Protokolle. Sie können zum Beispiel nur Pakete zeigen, die eine bestimmte IP-Adresse (oder den dazugehörigen Rechnernamen) involvieren:

Eigenschaften von Paketen

```
ip.addr == 192.168.61.254
ip.host == router.example.com
```

(beachten Sie das doppelte Gleichheitszeichen, ausgeliehen von der Programmiersprache C).

Passen Sie auf: Wenn Sie Pakete sehen wollen, die eine bestimmte Adresse *nicht* involvieren, dann liefert der offensichtliche Ansatz

```
ip.addr != 192.168.61.254                    Funktioniert nicht!
```

nicht das gewünschte Ergebnis – Pakete mit der betreffenden Adresse können trotzdem durchrutschen. Das liegt daran, dass jedes IP-Datagramm *zwei* Adressen enthält. Der Filterausdruck steht aber für »Dieses Paket enthält mindestens eine Adresse, die nicht `192.168.61.254` ist«, und das gilt auch für Pakete, wo die *andere* Adresse *doch* `192.168.61.254` ist. (Siehe hierzu Übung 3.5.)

Natürlich geht das auch gezielter:

```
ip.src == 192.168.61.1                       Diese Adresse als Absender
ip.dst_host == red.example.com               Dieser Name als Empfänger
```

Oder Sie suchen nur nach Paketen mit mehr als 100 Oktetten Länge:

```
ip.len > 100
```

Als kleine Gedächtnisstütze werden Ihnen beim Eintippen von Ausdrücken wie »`ip.dst_host`« die möglichen Vervollständigungen angezeigt. Sie können die dann auch direkt mit der Maus auswählen.

Ein Klick auf »Clear« rechts vom Eingabefeld löscht dessen Inhalt und schafft Platz für neue Kreationen.

Filter löschen

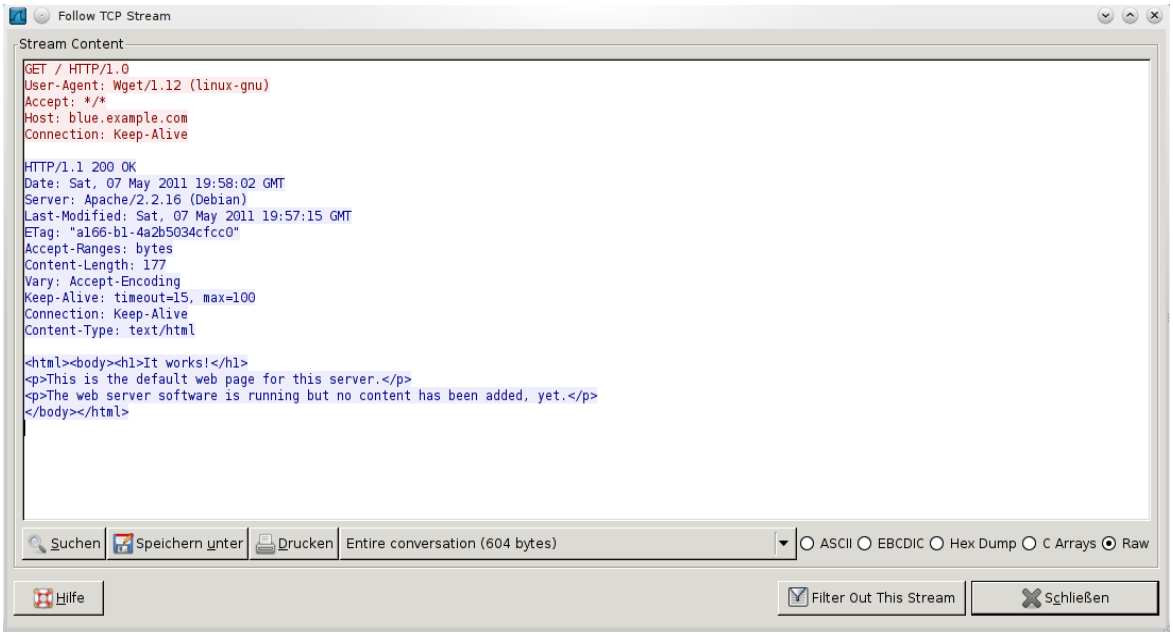


Bild 3.5: Verfolgung von TCP-Verbindungen mit wireshark

Filter interaktiv

Mit »Expression…« direkt rechts vom Eingabefeld können Sie sich Ihre Ausdrücke auch interaktiv mit der Maus zusammenklicken.

Filter aus Paket

Wenn Sie in der Paketliste ein Paket mit der rechten Maustaste anklicken, bekommen Sie (unter anderem) die Möglichkeit, direkt Eigenschaften dieses Pakets zur Filterung heranzuziehen. Mit »Apply as Filter« wird zum Beispiel die Absender- oder Empfängeradresse des Pakets als Filter übernommen (welche, hängt davon ab, wo genau der Mauszeiger steht, wenn Sie die rechte Taste drücken). Sie können die Adresse so auch einem bestehenden Filter hinzufügen.

Auch Anzeigefilter können Sie wie die »Capture Filter« mit Namen versehen und abspeichern. Klicken Sie dazu auf das Wort »Filter« links vom Eingabefeld – es öffnet sich ein Dialog, in dem Sie dem Filterausdruck einen Namen geben können. Aus diesem Dialog können Sie auch früher benutzte, benannte Filter auswählen und erneut verwenden. – Denselben Dialog erhalten Sie auch über den Eintrag »Display Filters…« im Menü »Analyze«.

Einzelne TCP-Verbindungen verfolgen Wenn Sie das Gefühl haben, vor lauter Bäumen den Wald nicht zu sehen (oder umgekehrt?), können Sie sich einzelne TCP-Verbindungen herauspicken. Klicken Sie dazu ein an der Verbindung beteiligtes Paket an und wählen Sie im Menü »Analyze« die Option »Follow TCP Stream«. Dadurch wird die Paketliste auf diejenigen Pakete beschränkt, die zu der betreffenden Verbindung gehören. Außerdem öffnet wireshark ein Fenster, das die Konversation im ASCII-Format zeigt (andere Formate sind wählbar). In diesem Fenster sind die Daten, die der Client[3] schickt, rosa und die, die der Server[4] schickt, blau unterlegt. Dies ist extrem nützlich für alle textorientierten Protokolle, etwa HTTP, SMTP oder POP3.

SSL-Verbindungen

Mit wireshark können Sie sogar SSL-Verbindungen sinnvoll (also nicht nur

[3] Strenggenommen der Rechner, der die Verbindung aktiv aufgebaut, also das SYN-Segment im Drei-Wege-Handshake geschickt hat – im Gegensatz zu dem Rechner, der die Verbindung passiv entgegengenommen, also das SYN/ACK-Segment geschickt hat

[4] Sie wissen schon …

als bis zur Unkenntlichkeit verschlüsselte Daten) protokollieren. Vorbedingung dafür ist, dass Sie über die privaten Schlüssel der beteiligten Zertifikate verfügen (aber wenn Sie der Administrator des betreffenden Servers sind, sollte das kein Problem sein) und dass die privaten Schlüssel nicht mit einer Passphrase versehen sind – besorgen Sie sich unverschlüsselte Kopien der Schlüsseldateien und werfen Sie sie weg, wenn Sie sie nicht mehr brauchen. Um wireshark die Entschlüsselung des Datenverkehrs zu ermöglichen, müssen Sie im »Preferences«-Dialog über »Protocols« zu »SSL« navigieren und können dann im Feld »RSA keys list« auf die beteiligten Schlüssel verweisen. Dies geschieht in der Form

⟨*Server-IP*⟩,⟨*Server-Port*⟩,⟨*Protokoll*⟩,⟨*Schlüsseldatei*⟩

also zum Beispiel

```
192.168.65.1,443,http,/home/hugo/ssl/blue-key.pem
```

(setzen Sie die Rechte auf /home/hugo/ssl so, dass nur Sie die Dateien im Verzeichnis lesen dürfen). Sie dürfen auch mehrere solche Definitionen angeben, wenn Sie sie mit Semikolons trennen. Wenn Sie in »SSL debug file« einen Dateinamen eintragen, schreibt wireshark dort Informationen hinein, die beim Debuggen der wireshark-Konfiguration nützlich sein können.

Experten-Infos Mit der Funktion »Expert Info« im Menü »Analyze« bekommen Sie eine Übersicht über mögliche Anomalien und Fehler, die wireshark in einem Protokoll gefunden hat. wireshark teilt die Einträge in diesem Fenster in vier Prioritäten ein: Prioritäten

Chat Reine Information

Note Bemerkenswerte, aber nicht unbedingt bedenkliche Dinge, etwa HTTP-Fehlermeldungen

Warn Bedenkliche Dinge, etwa unerreichbare Gegenstellen (bei vielen Protokollen)

Error Echte Probleme, etwa syntaktisch falsche Pakete

Der »Severity Filter« rechts oben im Ergebnisfenster erlaubt es, die Ausgabe auf Pakete einer bestimmten Priorität (und darüber) zu beschränken. Zur weiteren Veranschaulichung sind die Einträge noch verschiedenen Gruppen zugeordnet, unter anderem Gruppen

Checksum Eine Prüfsumme war ungültig

Malformed Das Paket ist syntaktisch falsch und wird nicht weiter dekodiert

Protocol Anforderungen eines Protokolls (etwa Größen für Felder oder ähnliches) wurden verletzt, aber die Dekodierung kann weitergehen

Reassemble Ein fragmentiertes Paket konnte nicht wieder zusammengesetzt werden

Request Code Anfrage einer Anwendung (normalerweise unbedenklich und nur informativ)

Response Code Problem mit der Antwort eines Servers (etwa HTTP-Code 404)

Sequence Im Protokollablauf ist etwas nicht in Ordnung

Undecoded Paket konnte nicht dekodiert werden

Daneben kann es auch noch (kurze) Erklärungstexte geben.

Wenn Sie im »Expert Infos«-Fenster auf eine Nachricht klicken, wird im Hauptfenster das dazugehörige Paket angezeigt. Das macht es einfach, die Stelle im Protokoll zu finden, die mit der Experten-Info korrespondiert.

wireshark ist ein mächtiges, aber auch komplexes Werkzeug. Wir können hier bei weitem nicht alle seine Möglichkeiten illustrieren, aber haben hoffentlich einen kleinen und für die Praxis nützlichen Einblick gegeben.

Für wireshark gibt es ein umfangreiches Handbuch, das Sie unter http://www.wireshark.org/docs/ in verschiedenen Formaten (HTML, PDF, …) finden können.

Ohne jetzt das LPI miesmachen zu wollen: Sie können auch eine Zertifizierung nur für wireshark erwerben und sich zum *Wireshark Certified Network Analyst* aufschwingen. Das organisiert die *Wireshark University* (keine echte Universität!) unter http://www.wiresharktraining.com/.

Übungen

3.5 [!1] Wir hatten Ihnen erklärt, dass der wireshark-Filterausdruck

```
ip.addr != 192.168.61.254
```

nicht dafür geeignet ist, nur solche Pakete anzuzeigen, die die Adresse 192.168.61.254 *nicht* involvieren. Geben Sie einen Filterausdruck an, der das Gewünschte leistet.

3.6 [2] Verwenden Sie wireshark, um Ihren Browser dabei zu beobachten, wie er eine komplexe Web-Seite abruft (etwas wie http://www.tagesschau.de/ könnte sich anbieten). Wie ist der allgemeine Ablauf? Wie viele HTTP-Anfragen werden gestellt? Wie viele TCP-Verbindungen werden aufgebaut? Wie viele verschiedene Web-Server sind beteiligt? Welche Hilfsmittel gibt wireshark Ihnen, um diese Fragen zu beantworten?

Kommandos in diesem Kapitel

tcpdump Netzwerk-Sniffer, protokolliert und analysiert Netzwerkverkehr
tcpdump(1) 32

wireshark Paket-Sniffer, liest und analysiert Netzwerkverkehr (Ex-ethereal)
wireshark(1) 41

Zusammenfassung

- Paketsniffer wie tcpdump und wireshark können den gesamten Verkehr auf einem Netzsegment beobachten und analysieren.
- tcpdump ist ein einfaches textorientiertes und universell einsetzbares Werkzeug.
- wireshark ist grafisch orientiert und sehr umfangreich und leistungsfähig.

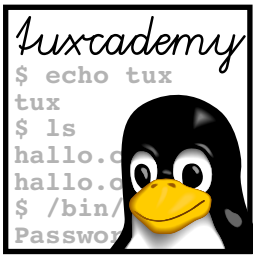
Literaturverzeichnis

Cha10 Laura Chappell. *Wireshark Network Analysis – The Official Wireshark Certified Network Analyst Study Guide*. Protocol Analysis Institute, 2010. ISBN 978-1893939-99-8. http://www.wiresharktraining.com/book.html

RFC1323 V. Jacobson, R. Braden, D. Borman. »TCP Extensions for High Performance«, Mai 1992. http://www.ietf.org/rfc/rfc1323.txt

San07 Chris Sanders. *Practical Packet Analysis: Using Wireshark to Solve Real-World Network Problems*. No Starch Press (Dpunkt Verlag), 2007. ISBN 978-159327149-7.

Ste94 W. Richard Stevens. *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley Professional Computing Series. Boston etc.: Addison-Wesley, 1994.



4

Linux und WLAN

Inhalt

4.1 Einleitung 56
4.2 WLAN-Grundkonfiguration 56
4.3 WLAN-Verschlüsselung 58

Lernziele

- Einen Linux-Rechner als Client in ein WLAN integrieren können
- Gängige WLAN-Verschlüsselungsverfahren konfigurieren können

Vorkenntnisse

- Linux-Systemverwaltung
- Linux-Client-Konfiguration für TCP/IP

4.1 Einleitung

Wie Sie bereits gesehen haben, ist es nicht schwer, einen Linux-Rechner in ein kabelgebundenes Netz zu integrieren. Nachdem Sie das Kabel in die dafür vorgesehene Buchse gesteckt haben, müssen Sie nur noch die IP-Adresse, die Netzmaske und ein paar andere Parameter an der richtigen Stelle in die Konfiguration eintragen und schon sind Sie im Geschäft. Mit etwas Glück gibt es in Ihrem Netz sogar einen DHCP-Server, so dass Sie auf Ihrem Rechner gar nichts einstellen müssen. Sie haben es gut.

Anders ist das möglicherweise, wenn Sie einen Linux-Rechner in ein lokales Funknetzwerk (neudeutsch »WLAN«) einbinden möchten. Bevor Sie an den Punkt kommen, wo Ihr bisheriges Wissen über IP-Adressen, Netzmasken oder DHCP greift, müssen Sie erst mal dafür sorgen, dass Ihr Rechner mit dem WLAN redet (oder das WLAN mit ihm).

Schicht 2

Das WLAN als solches ist eine Alternative zum kabelgebundenen Ethernet und somit in der Schicht 2 im ISO/OSI-Referenzmodell angesiedelt. Sie müssen also sozusagen »das Kabel in die Buchse stecken«, und da fangen in der Regel die Probleme an. Sie brauchen nämlich nicht nur ein paar wichtige Parameter wie den Namen des Netzwerks, mit dem Sie sich verbinden möchten (es könnte im Umkreis mehrere WLANs geben) und möglicherweise den Funkkanal, sondern auch Informationen über eine etwaige Verschlüsselung des WLAN, ohne die Sie keine Verbindung aufbauen können.

WLANs sind heute in aller Regel verschlüsselt, da man als Betreiber sonst Schwierigkeiten bekommen kann, wenn ein Client im WLAN zum Beispiel Raubkopien von Musik oder Filmen anbietet – was leicht passieren kann, wenn er sich an einem Peer-to-Peer-Filesharing beteiligt und eigentlich nur Sachen herunterladen möchte[1].

Verschlüsselungsverfahren

Für WLANs sind verschiedene Verschlüsselungsverfahren möglich. Das veraltete WEP sollten Sie nicht mehr verwenden, da es für einen Cracker mit der geeigneten Software in kürzester Zeit zu kompromittieren ist. WPA oder (besser) WPA2 sind derzeit noch ohne großes Magengrimmen zu verwenden; es gibt Angriffe, die aber noch sehr viel Aufwand erfordern, und es ist sowieso keine Alternative in Sicht. Wenn Sie sichergehen wollen, dann verschlüsseln Sie vertrauliche Informationen wie E-Mail, Kennwörter oder wichtige Web-Inhalte nicht nur auf der WLAN-Ebene, sondern auch im TCP, also auf ISO/OSI-Schicht 4 – zum Beispiel mit TLS –, oder verwenden Sie etwas wie OpenVPN.

4.2 WLAN-Grundkonfiguration

Treiber Bevor Sie sich an die eigentliche Konfiguration des WLAN-Zugangs machen können, sollten Sie sich vergewissern, dass das System Ihre WLAN-Karte (oder das in Ihren Rechner eingebaute WLAN-Interface) überhaupt erkennt. Manche WLAN-Hardware wird von Linux nämlich etwas stiefmütterlich unterstützt, weil die Hersteller keine Programmierinformationen liefern. Eine einfache Methode, um herauszufinden, ob das System Ihre WLAN-Hardware »kennt«, besteht darin, das Kommando »`ifconfig -a`« auszuführen:

```
# ifconfig -a
<<<<<
wlan0    Link encap: Ethernet  Hardware Adresse 00:24:d7:01:d1:a4
<<<<<
```

[1] Das Herunterladen ist nicht so eindeutig verboten wie das Anbieten, aber für die meisten Peer-to-Peer-Programme gilt, dass man beim Herunterladen auch anbietet, wenn man das nicht explizit ausschaltet (was viele Leute vergessen). Die Rechteinhaber verfolgen Sie, wenn, dann für das Anbieten – auch weil da die behaupteten Schadenssummen viel höher ausfallen.

»wlan0« ist ein typischer Name für ein WLAN-Interface, aber andere sind möglich.

Sollte nichts von einem WLAN-Interface zu sehen sein, dann kann das entweder daran liegen, dass kein passender Treiber zur Verfügung steht, oder daran, dass Sie zwar einen Treiber haben, aber dieser als erstes ein kleines Betriebssystem auf die WLAN-Karte laden muss, eine »Firmware«. Das Problem an dieser Stelle ist, dass die Lizenzlage dieser Firmwaredateien oft unklar ist, so dass die Linux-Distributoren davor zurückschrecken, sie mit dem Kernel zu verteilen. In diesem Fall müssen Sie sich die passende Firmware besorgen und dem Kernel zur Verfügung stellen. Prüfen Sie die Bootmeldungen des Kernels, um herauszufinden, ob Ihr Kernel nach einer bestimmten Firmware verlangt: Firmware

```
# dmesg
<KKKK
[   50.066719] iwlagn 0000:44:00.0: firmware: requesting ▷
 ◁ iwlwifi-6000-4.ucode
[   50.114207] iwlagn 0000:44:00.0: loaded firmware version 9.193.4.1
<KKKK
```

(Hier ist alles in Ordnung, ansonsten könnte auch eine Fehlermeldung erscheinen.) Firmware-Dateien werden normalerweise in /lib/firmware hinterlegt. /lib/firmware

Firmware finden Sie entweder in Ihrer Distribution – wenn der Hersteller nichts gegen eine Weiterverbreitung hat – oder mit etwas Glück per Google o. ä. im Internet. Intel zum Beispiel stellt Firmware für Intel-WLAN-Karten zur Verfügung, aber nicht als »freie Software« im Quellcode, sondern als binäre »Blobs«, die zwar kopiert werden dürfen, aber nicht verändert werden sollen/können. Diese Firmware ist also in Linux-Distributionen zu finden, wenn auch nicht notwendigerweise als Bestandteil der Standardinstallation; Debian GNU/Linux zum Beispiel tut sie in ein Paket namens firmware-iwlwifi im non-free-Teil der Distribution.

Für Freie-Software-Hardliner gibt es Varianten des Linux-Kernels, die keine Firmware enthalten, für die es keinen frei verfügbaren Quellcode gibt, und außerdem versuchen, die Namen der vom Kernel angeforderten externen Firmware-Dateien unlesbar zu machen (damit Benutzer nicht auf dumme Ideen kommen wie im Internet nach den unfreien Dateien zu suchen). Man könnte das als Übereifer auslegen.

iwconfig Sobald Sie sicher sind, dass Treiber und Firmware zur Verfügung stehen, und Sie auch den Gerätenamen Ihres WLAN-Interface wissen (in unserem Beispiel wlan0), können Sie versuchen, Ihren Rechner ins WLAN zu integrieren. Im einfachsten Fall dient dazu ein Programm namens iwconfig, das in Analogie zu ifconfig das Setzen diverser Parameter erlaubt. Zum Beispiel:

```
# iwconfig wlan0 essid "PinguNet"          WLAN-Name
# iwconfig wlan0 channel 1                 Funkkanal
# iwconfig wlan0 mode Managed              … als Client
# iwconfig wlan0 ap any                    beliebiger Access Point
```

Zumindest in einem unverschlüsselten Netz sollte Sie das so weit bringen, dass Sie anschließend entweder manuell mit ifconfig (Merke: »f«) die IP-Adresse usw. setzen oder einen DHCP-Client starten können.

Zu Testzwecken können Sie notfalls mal *ganz kurz* die Verschlüsselung Ihres WLAN ausschalten. Vergessen Sie aber nicht, sie später wieder einzuschalten, oder wundern Sie sich nicht über die vielen Leute mit den Laptops in Ihrem Vorgarten und die bösen Briefe, die Sie von den Anwälten der Musikindustrie geschickt bekommen.

`iwconfig` unterstützt viele fremdartige und wundervolle Einstellungsmöglichkeiten, die Sie im wirklichen Leben hoffentlich nie brauchen werden. Lesen Sie die Handbuchseite unter `iwconfig`(8).

4.3 WLAN-Verschlüsselung

Im Normalfall werden Sie es nicht mit unverschlüsselten WLANs zu tun haben, sondern mit solchen, die irgendein Verschlüsselungsverfahren einsetzen. Die einWEP fache WEP-Verschlüsselung – die Sie im wirklichen Leben bitte *nicht* verwenden wollen – können Sie noch mit `iwconfig` konfigurieren, indem Sie den Schlüssel wie folgt angeben:

```
# iwconfig wlan0 key s:SecretKey123                    oder was auch immer
```

(Den Schlüssel verrät Ihnen entweder Ihr freundlicher Systemadministrator, oder Sie schauen nach, was auf dem Etikett unten auf Ihrem WLAN-Router steht.)

WPA und WPA2 Um vernünftige Verschlüsselung, also WPA oder (besser) WPA2, zu verwen`wpa_supplicant` den, brauchen Sie ein weiteres Programm, nämlich `wpa_supplicant`. Dieses Programm kümmert sich um die Aushandelung der Verschlüsselung und tritt an die Stelle von `iwconfig`. Im einfachsten Fall können Sie eine Konfigurationsdatei `/etc/wpa_supplicant.conf` `/etc/wpa_supplicant.conf` anlegen, die Ihr WLAN beschreibt. Dabei hilft das Programm `wpa_passphrase`:

```
# wpa_passphrase PinguNet "SecretKey123" >/etc/wpa_supplicant.conf
# chmod 600 /etc/wpa_supplicant.conf
# cat /etc/wpa_supplicant.conf
network={
  ssid="PinguNet"
  #psk="SecretKey123"
  psk=b7256f1c680a0ce397f97fca8114d2231d2f1a2832e1c58aa821709c24fcf3f5
}
```

Je nachdem, welches Verschlüsselungsverfahren Sie verwenden, müssen Sie möglicherweise noch Informationen hinzufügen. Für »WPA2 Personal« zum Beispiel

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=wheel
network={
  ssid="PinguNet"
  key_mgmt=WPA-PSK
  psk=b7256f1c680a0ce397f97fca8114d2231d2f1a2832e1c58aa821709c24fcf3f5
}
```

Anschließend können Sie versuchen, sich mit dem WLAN zu verbinden:

```
# ifconfig wlan0 up
# wpa_supplicant -B -Dwext -i wlan0 -c /etc/wpa_supplicant.conf
```

Nach einer gewissen Wartezeit sollte die Ausgabe »`Associated:`« erscheinen, gefolgt von einer MAC-Adresse. In diesem Moment können Sie dann einen DHCP-Client starten oder die Adresse usw. manuell konfigurieren:

```
# dhclient wlan0
```

`wpa_supplicant` unterstützt eine sehr breite Auswahl von Verschlüsselungsverfahren und Konfigurationseinstellungen. Lesen Sie die Handbuchseiten `wpa_supplicant`(8) und `wpa_supplicant.conf`(5).

NetworkManager Im wirklichen Leben verwenden Sie wahrscheinlich eher eines der be`wicd` quemen Programme NetworkManager oder `wicd`, die die Verwaltung von WLAN-Zugängen wesentlich vereinfachen.

Übungen

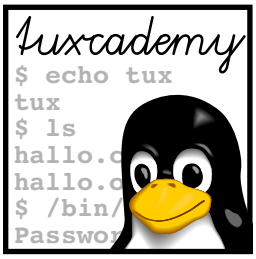
4.1 [2] Versuchen Sie, mit Hilfe des `wpa_supplicant` Verbindung zu einem geeigneten WLAN aufzunehmen.

Kommandos in diesem Kapitel

`iwconfig` Programm zur Konfiguration von WLAN-Geräten `iwconfig`(8) 57
`wpa_supplicant` Kümmert sich um WLAN-Zugang mit Verschlüsselung `wpa_supplicant`(8) 58

Zusammenfassung

- Die WLAN-Konfiguration ist nötig, um eine ISO/OSI-Schicht-2-Verbindung zum WLAN herstellen zu können. Die TCP/IP-Konfiguration der Schnittstelle erfolgt anschließend.
- Manche WLAN-Adapter benötigen proprietäre Firmware, damit sie funktionieren.
- Mit `iwconfig` können Sie einen einfachen WLAN-Zugang konfigurieren.
- Der Zugriff auf mit WPA oder WPA2 verschlüsselte WLANs erfordert den `wpa_supplicant`.
- Programme wie NetworkManager und `wicd` vereinfachen die WLAN-Konfiguration ungemein.



5

DNS: Grundlagen

Inhalt

5.1 Einführung in die Namensauflösung 62
5.2 Das Domain Name System 63
5.2.1 Eine kurze Geschichte 63
5.2.2 Aufbau eines DNS-Namens 64
5.2.3 Namensverwaltung 67
5.2.4 Domains und Zonen 68
5.2.5 Das Protokoll 69
5.3 Linux als DNS-Client 70
5.3.1 Die C-Laufzeitbibliothek 70
5.3.2 Der Resolver 71
5.3.3 Verhalten bei Zeitüberschreitungen 72
5.4 DNS testen 73
5.4.1 Einleitung 73
5.4.2 Das Kommando `dig` 73
5.4.3 Das Kommando `nslookup` 75
5.4.4 Das Kommando `host`. 77

Lernziele

- Namensauflösung unter Linux kennen lernen
- Das DNS-Protokoll grundsätzlich verstehen
- Linux als DNS-Client einrichten können
- Eine DNS-Installation auf korrekte Funktion prüfen können

Vorkenntnisse

- Kenntnisse über Linux-Systemkonfiguration
- TCP/IP-Kenntnisse

```
#
# hosts         This file describes a number of hostname-to-address
#               mappings for the TCP/IP subsystem.  It is mostly
#               used at boot time, when no name servers are running.
#               On small systems, this file can be used instead of a
#               "named" name server.
# Syntax:
#
# IP-Address  Full-Qualified-Hostname  Short-Hostname
#
# special IPv6 addresses

127.0.0.1       localhost
::1             localhost ipv6-localhost ipv6-loopback
fe00::0         ipv6-localnet
ff00::0         ipv6-mcastprefix
ff02::1         ipv6-allnodes

192.168.0.99    linux.example.com  linux
```

Bild 5.1: Die Datei `/etc/hosts` (SUSE)

5.1 Einführung in die Namensauflösung

Bereits in kleinen und mittelgroßen Netzwerken ist es schwer, die Übersicht über die IP-Adressen zu behalten. Hingegen fällt es menschlichen Benutzern meist sehr viel leichter, sich an die Rechnernamen zu erinnern. Ferner haben Namen den Vorteil, dass sie »sprechend« sind. Während wohl kaum jemand weiß, welcher Rechner sich hinter der IP-Adresse `204.152.189.116` verbirgt, lässt sich das anhand eines Namens wie `ftp.kernel.org` schon eher ahnen.

Rechnernamen sind leichter zu merken

verschiedene Dienste

In einer TCP/IP-Umgebung stehen verschiedene Dienste zur Verfügung, die Rechnernamen auf IP-Adressen abbilden.

Die einfachste Variante ist es, auf jedem Rechner eine Liste aller Namen in der Datei `/etc/hosts` abzulegen, die dann von den Applikationen nach dem gewünschten Rechnernamen durchsucht wird.

Lokale Datentabellen sind aber nur für sehr kleine LANs sinnvoll, die keinerlei Kontakt zur Außenwelt haben. Bereits bei größeren Netzwerken ist selbst ohne Internetverbindung die Größe der `hosts`-Datei und der damit verbundene Pflegeaufwand zu hoch. Für ein Netzwerk aus neun Rechnern, dem ein zehnter hinzugefügt wird, muss nicht nur auf dem neuen Rechner eine `hosts`-Tabelle mit zehn Einträgen erstellt werden. Ferner müssen auch die Datensätze auf den anderen neun Rechnern jeweils um einen neuen Eintrag ergänzt werden.

Offensichtlich ist für diesen Fall eine zentrale Verwaltung der gewünschten Informationen von Vorteil. Zur Implementierung einer solchen zentralen Datenbank exisitieren im Fall von Rechnernamen und IP-Adressen gleich zwei Verfahren, nämlich das **Network Information System (NIS)** und das **Domain Name System (DNS)**. Diese Dienste sind zwar schwieriger in der Grundkonfiguration, aber spätere Änderungen in der Netztopologie werden damit fast problemlos ermöglicht.

NIS

DNS

Lokale Namensauflösung mit `/etc/hosts` Die lokale Umsetzung von Rechnernamen in IP-Adressen verläuft in Linux-Systemen mittels der Datei `/etc/hosts`. Dabei handelt es sich um gewöhnlichen ASCII-Text, in dem neben Kommentarzeilen, die mit »#« eingeleitet werden, zeilenweise Einträge vorgenommen werden können. Diese enthalten spaltenweise mindestens die IP-Adresse und den vollständigen Namen (FQDN) des Rechners. Daneben ist es erlaubt, noch Kurznamen

für den Rechner anzugeben. Als Trennzeichen der einzelnen Spalten dienen Leer- und/oder Tabulatorzeichen, also »Freiplatz« (*white space*). Bild 5.1 zeigt den Inhalt einer exemplarischen `/etc/hosts`-Datei.

Diese Datei wird von jedem Client lokal gehalten und muss demzufolge auch auf jedem Client gepflegt werden. Sobald also ein neuer Rechner ins Netzwerk kommt, muss die Datei `/etc/hosts` auf jedem Rechner geändert werden. In größeren Netzwerken wird das sehr schnell unübersichtlich und umständlich zu konfigurieren, so dass man lieber auf zentrale Dienste ausweicht.

Zentrale Namensauflösung per NIS In lokalen Netzwerken ist es üblicherweise ein Ziel, den Benutzern eine Umgebung zur Verfügung zu stellen, die das Netzwerk transparent erscheinen lässt. Ein wichtiger Schritt in diese Richtung ist der Abgleich aller wichtigen Daten wie Benutzerinformationen zwischen allen Rechnern. Für die Auflösung von Rechnernamen existiert der *Domain Name Service*, DNS. Für alle anderen Aufgaben gibt es keinen so spezialisierten Dienst.

Aus diesem Grund entwickelte *Sun Microsystems* das *Network Information System*, kurz NIS. Diese Funktion dient der netzwerkweiten, auch subnetzübergreifenden zentralen Speicherung der üblicherweise in lokalen Dateien abgelegten Informationen (Benutzernamen, Kennwörter, Rechnernamen …) auf einem NIS-Server. NIS

NIS bietet die Möglichkeit, eine `/etc/hosts`-Datei auf einem zentralen Server zu verwalten und von den anderen Rechnern im Netz aus auf diese zentrale Kopie zuzugreifen. NIS wird in dieser Schulungsunterlage allerdings nicht weiter besprochen.

Verteilte Namensdatenbank: DNS In großen Systemen und für die Teilnahme am Internet führt an der Nutzung des *Domain Name System* (DNS) kaum ein Weg vorbei. Hierbei werden die Namen in einer weltweit verteilten Datenbank gehalten, auf die Sie über das Internet zugreifen können. Das DNS ist Thema des Rests dieser Schulungsunterlage.

Übungen

5.1 [2] Diskutieren Sie die Vor- und Nachteile einer statischen Tabelle wie `/etc/hosts` zur Namensauflösung.

5.2 [2] Unter welchen Umständen kann es sinnvoll sein, Namen in `/etc/hosts` einzutragen, auch wenn man sonst DNS benutzt?

5.2 Das Domain Name System

5.2.1 Eine kurze Geschichte

Im Rahmen des ursprünglich noch recht überschaubaren ARPAnet wurden die IP-Adressen aller Stationen in einer zentralen Datei `HOSTS.TXT` verwaltet und diese per FTP verteilt. Für die Pflege dieser Datei war das *Network Information Center* (NIC) des *Stanford Research Institute* (SRI) zuständig. Mit dem Wachstum des ARPAnet zeigten sich aber rasch die Grenzen dieser Methode. Nicht nur die Datei selbst wuchs immer weiter, auch die Last auf dem zentralen Server zur Aktualisierung der Informationen nahm stetig zu. Immerhin war bei einer zu seltenen Aktualisierung der Daten deren Konsistenz nicht mehr gegeben. ARPAnet

Zur Lösung dieses Problems erfand Paul Mockapetris ein neues Verfahren, das auf folgende Grundideen zurückgeht:

- Die Daten für kleine Teilbereiche des Gesamtnetzes werden lokal verwaltet.
- Diese lokal verwalteten Teildaten sind allgemein zugänglich.

- Die Benennung der Stationen erfolgt innerhalb eines hierarchischen Namensraumes.

Verteilte Datenbank

Aus diesen Rahmenbedingungen ergibt sich das Konzept einer riesigen verteilten Datenbank, die auch eine Lastverteilung auf verschiedene Maschinen ermöglicht.

Das Konzept wurde ursprünglich 1984 in [RFC0882, RFC0883] festgeschrieben. Die aktuellen Spezifikationen sind in [RFC1034, RFC1035] zu finden.

Das *Domain Name System* ordnet Rechnern neben IP-Adressen auch Namen zu.

Domain-Modell

Die Namen folgen dabei einer hierarchischen Struktur, dem sogenannten Domain-Modell. Der Dienst wird von speziellen Rechnern, den DNS-Servern, zur Verfügung gestellt, und die Adresse eines DNS-Servers (oder mehrerer) ist ein wichtiger Bestandteil der Netzkonfiguration eines Rechners.

Im Internet ist DNS zwingend erforderlich. Auch in isolierten LANs ist es manchmal sehr nützlich, bringt aber hier keine nennenswerten Vorteile gegenüber NIS, wenn dieses sowieso benutzt wird. Für ein kleines Ethernet mit bis zu vielleicht einem Dutzend Maschinen können Sie auf beides verzichten und sich wie beschrieben mit einer einfachen `/etc/hosts`-Datei begnügen.

Die heute vielleicht wichtigste Anwendung des DNS ist der Zugriff auf das World-Wide Web – ein Browser muss den Rechnernamen in einem URL wie `http://www.example.com/index.html` in eine IP-Adresse auflösen können, damit er weiss, wohin er seine HTTP-Verbindung aufbauen muss. Nur wegen des WWW ist es aber nicht nötig, alle Rechner im lokalen Netz für DNS zu konfigurieren, wenn Sie das nicht sowieso vorhaben – ein Web-Proxy mit DNS-Zugang genügt völlig.

Damit Sie ein Gefühl für den Umfang der DNS-Datenbank bekommen: Die monatliche Internet-Domain-Erhebung des ISC ergab im Januar 2011 eine Gesamtzahl von 818.374.269 Namen im DNS. Die Domains `net` und `com` gemeinsam machen gut die Hälfte davon aus; die größten länderspezifischen Domains sind `jp` (Japan), `it` (Italien) und `br` (Brasilien). `de` (Deutschland) steht mit knapp 21 Millionen Namen an sechster Stelle. Die Tendenz ist stark steigend; seit dem Juli 2006 hat die Größe des DNS sich fast verdoppelt. Die komplette Erhebung können Sie aktuell unter `https://www.isc.org/solutions/survey` finden.

5.2.2 Aufbau eines DNS-Namens

Damit die Namensauflösung eines Rechnernamens im Internet bzw. in großen Firmennetzen reibungslos funktioniert, sind die Namen der Rechner strukturiert aufgebaut. Diese Strukturierung funktioniert ähnlich wie die Struktur von Dateinamen in einem Unix-Dateisystem und wird als **Domain-Modell** bezeichnet. Die Idee ist, dass ein Namensteil nur unter den »Kindern« des unmittelbar darübergelegenen Namensteils eindeutig sein muss – innerhalb von `de` darf es also den Namen `linupfront.de` nur einmal geben, ein unabhängiges `linupfront.com` ist dagegen absolut erlaubt. Entsprechend sind auch `www.example.com` und `www.linupfront.com` gleichzeitig zulässig, da der Name `www` jeweils in verschiedenen »Teilhierarchien« des DNS auftaucht.

Domain-Modell

Namensstruktur

In DNS-Namen spielt der Punkt die Rolle des Schrägstrichs bei Unix-Dateinamen, und genau wie ein einzelner Schrägstrich die »Wurzel« des Unix-Dateibaums angibt, steht ein einzelner Punkt für die »Wurzel« des DNS-Namensbaums (hierzu gleich mehr). Der einzige andere Unterschied ist, dass wir Unix-Dateinamen von links nach rechts und DNS-Namen von rechts nach links betrachten.

Dass das so ist, hat historische Gründe. Schon bevor das Domainkonzept eingeführt wurde, waren in manchen Netzen E-Mail-Adressen der Form `HUGO@SONSTWO` populär (unsere amerikanischen Freunde lasen das als »Hugo at Sonstwo«; wenn es in Sonstwo nur einen Hugo gibt, dann ist das eine eindeutige Beschreibung). Eine Erweiterung dieser Idee nach rechts ist da ziemlich naheliegend.

Bevor das Internet in den 1990er Jahren in Großbritannien Fuß fassen konnte, gab es dort im akademischen Bereich einen unabhängigen Vorläufer namens JANET (das *Joint Academic Network*). Eine der Eigenarten von JANET war, dass Rechnernamen von links nach rechts geschrieben wurden. Die – hypothetische – Internet-Adresse `bill@cs.ed.ac.uk` (Benutzer `bill` im Department of Computer Science der University of Edinburgh, akademische Adresse, Großbritannien) wäre zu JANET-Zeiten also `bill@uk.ac.ed.cs` gewesen. Sie können sich vorstellen, dass es seinerzeit eine größere Herausforderung bedeutete, aus dem Internet Mail an JANET-Benutzer zu schicken – spezielle E-Mail-Gateways mussten die Adressen umdrehen, und so manche Nachricht an britische Informatiker landete statt dort in der Tschechoslowakei (`cs`). Übrigens ein Grund dafür, dass viele Informatik-Fachbereiche in Großbritannien heute noch `dcs` statt `cs` im Domainnamen führen!

mögliche Zeichen für DNS-Namen

Als mögliche Zeichen für DNS-Namen kommen Buchstaben, Ziffern, Unterstreichungen und der Bindestrich in Frage, letzterer aber nicht am Anfang eines Namensteils (engl. *label*). Zwischen Groß- und Kleinschreibung wird nicht unterschieden. Die maximale Länge eines Namensteils beträgt 63 Zeichen.

Länderspezifische Sonderzeichen (in Deutschland zum Beispiel Umlaute und »ß«) sind inzwischen im DNS prinzipiell erlaubt, auch wenn es an der tatsächlichen Umsetzung mitunter noch hapert. Welche Sonderzeichen wirklich akzeptiert werden, hängt von der Politik des jeweiligen Registrars (Abschnitt 5.2.3) ab.

Beginnend von rechts kann man verschiedene Bestandteile eines DNS-Namens identifizieren:

Die Root-Level-Domain Ganz rechts befindet sich die "Root-Level"-Domain namens ».«. Als Benutzer müssen Sie diesen abschließenden Punkt zum Glück so gut wie nie angeben; nur wenn Sie einen DNS-Server administrieren, ist das nicht so, denn der Punkt kennzeichnet unzweifelhaft das Ende des DNS-Namens. An Namen ohne Punkt in einer DNS-Server-Konfiguration hängt der Server gerne ungefragt eine »Standarddomain« an, und das bringt alles durcheinander. Seien Sie also vorsichtig!

Wie wir später sehen werden, können Sie auf einem DNS-*Client* eine oder mehrere Standard-Domains angeben, die an Namen ohne Punkt angehängt werden, bevor sie im DNS gesucht werden. Dies stellt oft eine Schreibvereinfachung dar.

Die Top-Level-Domain Die sieben ursprünglichen »generischen« Top-Level-Domains (»gTLDs«) dienten zur Unterteilung des Namensraums in organisatorische Einheiten. Seit 2000 wurden einige neue generische Top-Level-Domains ausgewählt, um den Namensraum besser zu strukturieren. (Es ist nicht ganz einsehbar, warum aus Dutzenden mehr oder weniger vernünftiger Vorschläge gerade diese ausgesucht wurden, aber gewöhnliche Menschen müssen das wohl auch nicht verstehen können). Eine Übersicht liefert Tabelle 5.1. Über einige davon, allen voran `xxx` und `mobi` gibt es auch noch massig Streit.

Unterteilung des Namensraums

Neben den generischen Top-Level-Domains existieren eine Anzahl zweibuchstabiger Länderkennungen oder »ccTLDs« (für *country-code TLDs*), etwa »`de`« für Deutschland, »`at`« für Österreich oder »`ch`« für die Schweiz. Diese Länderkennungen folgen dem ISO-Standard ISO-3166 (ISO – International Organization for Standardization), mit der Ausnahme, dass Großbritannien statt der ISO-Länderkennung »`gb`« aus historischen Gründen »`uk`« verwendet.

Länderkennungen

Zuletzt zu erwähnen wäre die »Infrastruktur-TLD«, `arpa`, die für interne administrative Zwecke des DNS verwendet wird – die Details sehen wir später.

Der Domainname Die nächste Ebene ist der Domainname, der innerhalb einer Top-Level-Domain nur ein einziges Mal benutzt werden kann. Damit diese Einmaligkeit gewährleistet ist, müssen die Domainnamen bei einem Registrar, oft ei-

Registrar

Tabelle 5.1: Die generischen Top-Level-Domains

gTLD	Typ	Bedeutung	Betreiber	Beispiel
arpa	I	IANA		
com	G	kommerzielle Organisationen	VeriSign*	ibm.com
edu	S	(US-)Bildungseinrichtungen	EDUCAUSE	berkeley.edu
gov	S	US-Regierungseinrichtungen	US GSA	nasa.gov
int	S	internationale Organisationen	IANA	nato.int
mil	S	US-Militäreinrichtungen	US DoD NIC	army.mil
net	G	Netzwerkinfrastruktur	VeriSign*	uu.net
org	G	andere Organisationen	Public Interest Registry*	linux.org
aero	S	Organisationen der Luftfahrtindustrie	SITA*	
asia	S	Asien und Pazifik	DotAsia Org. Ltd.	
biz	GR	kommerzielle Unternehmen	NeuLevel*	
cat	S	Katalanische Seiten	Fundacio puntCAT*	
coop	S	kooperative Organisationen	DotCooperation LLC*	
info	G	für beliebigen Gebrauch	Afilias Ltd*	
jobs	S	Arbeitsvermittler u. ä.	Employ Media LLC*	
mobi	S	Mobile Produkte/Dienste	mTLD Top Level Domain Ltd*	
museum	S	Museen	Museum Domain Mgmt Assoc*	
name	GR	Einzelpersonen	VeriSign*	
pro	GR	Freiberufler	RegistryPro*	
tel	S	Kontaktdatenbank	Telnic Ltd	
travel	S	Reise-Industrie	Tralliance LLC*	
xxx	S	»Erwachsenen-Unterhaltung«	ICM Registry LLC*	

Typ: G = generisch, GR = generisch mit Restriktionen, I = Infrastruktur, S = gesponsort
= Registrierungen über mehrere konkurrierende Registrars möglich

ner nationalen Institution, angemeldet und genehmigt werden.

DENIC In Deutschland kümmert sich darum das **DENIC (Deutsches Network Information Center)** mit Sitz in Frankfurt am Main. Das DENIC verwaltet eine zentrale Datenbank für die Top-Level-Domain `de`, mit deren Hilfe per `whois` abgefragt werden kann, ob ein Domainname bereits vergeben wurde und wenn ja, wem er gehört (`http://www.denic.de/`).

Subdomain und Rechnername Subdomains sind ein optionaler Bestandteil eines DNS-Namens, sie können vorhanden sein, müssen aber nicht. Es ist zulässig, mehr als eine Subdomain im Namen zu verwenden, aber in der Praxis werden Sie nicht mehr als zwei oder drei Subdomains finden, da der Name sonst zu komplex und unbequem wird. Die maximale Verschachtelungstiefe eines DNS-Namens ist auf 127 Ebenen begrenzt; insgesamt dürfen DNS-Namen bis zu 255 Zeichen lang sein. Je nach der Art des DNS-Namens kann der linkeste Namensteil einen Rechnernamen darstellen.

Eine wesentliche Eigenschaft von Subdomains und Rechnernamen ist, dass diese nicht mit dem DENIC bzw. dem äquivalenten lokalen Vertreter abgestimmt oder gar kostenpflichtig registriert werden müssen. Der Administrator eines Netzes kann hier jeden gültigen Namen vergeben – solange er innerhalb der entsprechenden Subhierarchie auf dieser Ebene eindeutig ist.

Ob Sie als DNS-Kunde eines Providers in der Position sind, Subdomains einrichten zu können, hängt vor allem davon ab, ob Ihr Provider Ihnen das erlaubt. Die großen Webspace-Provider zum Beispiel sind da möglicherweise sehr restriktiv, einerseits um der Vereinfachung willen und andererseits um umfassendere DNS-Dienste als Bestandteil »höherwertiger« Tarife verscherbeln zu können. Wenn Sie Ihren eigenen DNS-Server betreiben, dürfen Sie sich natürlich beliebig austoben.

5.2.3 Namensverwaltung

Die Namenshierarchie erfordert fast zwangsläufig eine Reihe von Institutionen und Organisationen, die darauf achten, dass die entsprechenden Konventionen eingehalten werden. Die wichtigste Organisation in diesem Zusammenhang ist ICANN (die *Internet Corporation for Assigned Numbers and Names*), ein Unternehmen, das als internationale nicht profitorientierte *private-public partnership* die Verantwortung für die Vergabe von Dingen wie IP-Adressräumen, Protokollnummern und eben auch die Verwaltung von Top-Level-Domains und die Administration der Root-Level-DNS-Server hat.

ICANN

Root-Level-DNS-Server

Die Root-Level-DNS-Server, weltweit 13 an der Zahl, stellen die Adressen von DNS-Servern für die verschiedenen TLDs zur Verfügung. Wer die Root-Level-DNS-Server kontrolliert, bestimmt im wesentlichen, welche TLDs es gibt. Pikanterweise ist das letzten Endes nicht ICANN – ICANN kümmert sich nur darum, dass die Server laufen –, sondern das US-Handelsministerium, ein Umstand, der vielen im Internet ein Dorn im Auge ist. Bisher beabsichtigen die USA trotz sanftem Druck vom Rest der Welt nicht, hieran irgendetwas zu ändern.

Kontrolle über die Root-Level-Server

Eigentlich gibt es keinen zwingenden Grund dafür, dass das ICANN die Oberhoheit über die TLDs haben muss – prinzipiell kann jeder, der will, einen Root-Level-DNS-Server betreiben. Das Problem ist neben der technischen Herausforderung nur, dass man einen interessanten Anteil der Internet-Benutzer dazu bringen muss, diesen Root-Level-DNS-Server tatsächlich zur Namensauflösung zu benutzen, sonst hat es natürlich keinen großen Sinn, Namen in den solchen »alternativen« TLDs zu registrieren. Hier liegt also ein klassisches »Henne-Ei-Problem« vor. Diverse Firmen und Organisationen, etwa OpenNIC oder `new.net`, versuchen das ohne bisher besonders durchschlagenden Erfolg; siehe zum Beispiel `http://en.wikipedia.org/wiki/Alternative_DNS_root` für einen Überblick.

Wer darf einen Root-Level-Server betreiben?

Früher gab es eine Organisation namens IANA (*Internet Assigned Numbers Authority*), die der ICANN übergeordnet war. Die IANA ist jetzt ein Teilbereich der ICANN und befasst sich mit denjenigen Sachen, die ICANN macht, die nichts mit dem DNS zu tun haben.

IANA

Die Registrierung und Vergabe von Namen innerhalb von ccTLDs (`de`, `fr`, `es` usw.) wird meistens von in der Regel staatlich bestallten Organisationen in den betreffenden Staaten übernommen (in Deutschland das DENIC). Diese Organisationen machen mitunter weitere Annahmen über die in »ihrer« ccTLD akzeptablen Namen. In Deutschland zum Beispiel sind sogar »einstellige« Domainnamen möglich (nach [RFC1035] ist die minimale Größe zwei Buchstaben), genau wie Domains, die nur aus Ziffern bestehen (!). Domains, die gegen gewisse strafrechtliche oder sittliche Normen verstoßen, etwa weil sie Begriffe der Naziherrschaft verherrlichen, sind nicht explizit in der DENIC-Satzung verboten, werden aber trotzdem nicht registriert.

Registrierung und Vergabe von Namen

Bis zum 23. Oktober 2009 waren in der ccTLD `de` nur Domainnamen von minimal drei Zeichen Länge erlaubt. Außerdem waren »Kraftfahrzeugkennzeichen-Domains« wie `mtk.de` oder `hro.de` nicht möglich, genausowenig wie Domains, die so heißen wie andere TLDs (`edu.de` oder `mil.de`). Hintergrund der Liberalisierung war ein Urteil des Bundesgerichtshofs, das der Volkswagen AG das Recht zusprach, die Domain `vw.de` benutzen zu dürfen.

Nicht erlaubt in `de`-Domains sind Bindestriche am Anfang oder am Ende des Namens oder – um Probleme mit Sonderzeichen zu vermeiden (fragen Sie nicht …) an dritter oder vierter Stelle des Namens.

Organisationen wie das DENIC verkehren in der Regel nicht direkt mit »Endkunden« (Leuten, die Second-Level-Domains registrieren wollen), sondern delegieren diesen Teil des Geschäfts an Subunternehmer, die die Domains praktisch

»weiterverkaufen«. (Das DENIC verkauft auch Domains an Endkunden, aber der Abschreckung halber zu völlig überzogenen Mondpreisen.) Bei einigen TLDs, etwa `com` und `net`, betreibt die in Tabelle 5.1 genannte Firma nur die Datenbank, während die Details der Domainregistrierung von sogenannten **Registrars** übernommen werden, die um die Domain-Anmelder, die *registrants*, konkurrieren.

Registrars

Oft gibt es Streit darum, wem ein spezieller DNS-Name gehören sollte. *Domain squatters* (»Domainbesetzer«) zum Beispiel registrieren DNS-Namen, von denen sie glauben, dass sie für Firmen interessant sein könnten (etwa weil die Firmen genauso heißen), in der Hoffnung, mit deren späterer Freigabe Geld verdienen zu können. In den meisten TLDs wird zur Beilegung solcher Probleme die *Uniform Domain Name Dispute-Resolution Policy* (UDRP) verwendet; in Deutschland verläßt das DENIC sich auf die Zivilgerichtsbarkeit.

Das Ganze kann beliebig kompliziert werden: Muss der (hypothetische) Privatmann Hugo Schulz seine (hypothetische) Domain `schulz.de` an den (hypothetischen) Konzern Schulz AG aus dem DAX abtreten und wenn ja, zu welchen Bedingungen? Was ist, wenn es nur die (hypothetische) Schulz GbR ist? In einer vielbeachteten Gerichtsentscheidung wurde immerhin das Primat der Stadt Heidelberg gegenüber einem privaten Internet-Provider zementiert, der unter der Domain `heidelberg.de` eine Regional-Datenbank anbot, und das dürfte im Analogieschluss auch für andere Kommunen richtungsweisend sein.

Wirklich haarig wird es da, wo Einzelunternehmen sich qua Domainname eine ganze Branche »sichern« wollen; notorisch geworden ist hier der Fall der `mitwohnzentrale.de`. Ein Verein von etwa 40 »Mitwohnzentralen« klagte gegen einen anderen Verein von etwa 25 anderen solchen Organisationen, der die Domain `mitwohnzentrale.de` führte, auf Unterlassung. Das OLG Hamburg gab dem Kläger Recht und stellte fest, dass eine solche Domain nur mit weiteren unterscheidenden Merkmalen benutzt werden darf. Allerdings wurde das Urteil später vom Bundesgerichtshof aufgehoben [Din01]. Laut BGH gilt das Prinzip »Wer zuerst kommt, mahlt zuerst«, und auch »Gattungsbegriffe« wie `mitwohnzentrale.de` sind davon nicht ausgenommen. Allerdings sei es nicht zulässig, sich alle möglichen Schreibweisen eines Begriffs zu sichern und damit Konkurrenten daran zu hindern, den Gattungsbegriff ebenfalls zu benutzen. Demzufolge wäre es nicht erlaubt, etwa gleichzeitig `www.autovermietung.de` und `www.auto-vermietung.de` zu benutzen. Das Ganze ist nach wie vor ein Minenfeld.

5.2.4 Domains und Zonen

Eine Domain ist ein »Teilbaum« der DNS-Namenshierarchie, etwa `example.com`. Eine **Zone** dagegen ist eine Menge von Namen, die vom selben einzelnen DNS-Server verwaltet werden. Dabei kann es sich um eine komplette Domain mit allen ihren untergeordneten Namen (inklusive eventuellen Subdomains und den Namen darin) handeln, oder ein DNS-Server kann die Verantwortung für eine oder mehre Subdomains an andere DNS-Server »delegieren«, die diese Subdomains dann in eigenen Zonen führen.

Zone

Der Zusammenhang zwischen Domains und Zonen ist analog zum Zusammenhang zwischen Verzeichnissen und Dateisystemen unter Unix. Genau wie die Verteilung von Verzeichnissen auf Dateisysteme anhand eines Unix-Dateinamens nicht zu erkennen ist, ist die Verteilung von Domains auf Zonen anhand eines DNS-Namens nicht zu sehen.

Ein DNS-Server kann auch für mehrere getrennte Zonen zuständig sein.

5.2.5 Das Protokoll

Wie funktioniert DNS denn nun, wenn zu einem Namen (etwa `www.linupfront.de`) die IP-Adresse gefunden werden soll? Ein Programm, zum Beispiel ein Web-Browser, wendet sich dafür über einen sogenannten DNS-Resolver (meist Bestandteil der Laufzeitbibliothek) an einen **rekursiven DNS-Server**. Die Aufgabe eines rekursiven DNS-Servers ist es, auf Anfragen von Resolvern hin das DNS zu durchsuchen und Antworten an die Resolver zurückzugeben. Dabei befragt er einen oder mehrere **autoritative DNS-Server**, deren Aufgabe es ist, direkte Anfragen zu beantworten.

rekursiver DNS-Server

autoritativer DNS-Server

Die wichtigste Feststellung ist, dass es sich beim DNS um eine verteilte Datenbank handelt. Kein einzelner DNS-Server auf der Welt hat also den kompletten Überblick über alle Namen. Statt dessen tastet der rekursive DNS-Server sich langsam an die richtige Antwort heran – bei der Suche nach `www.linupfront.de` zum Beispiel zunächst dadurch, dass er einen Root-Level-DNS-Server nach den Adressen von DNS-Servern fragt, die Informationen über die Top-Level-Domain `de` enthalten. Danach fragt er einen dieser DNS-Server nach den Adressen von DNS-Servern, die Informationen über `linupfront.de` enthalten, und schließlich diese Server nach `www.linupfront.de`. Die dabei resultierende Adresse ist das Ergebnis, das er an den Resolver zurückgibt.

verteilte Datenbank

Woher weiß der rekursive DNS-Server, wo die Root-Level-DNS-Server zu finden sind? Die Antwort darauf lautet, dass er deren Adressen im wesentlichen hartkodiert enthält. Im Kapitel 6 ist genauer erklärt, wie das gemacht wird. Erfreulicherweise ändern sich die Adressen der Root-Level-DNS-Server so gut wie nie.

In der Praxis wird allerdings nicht wirklich so vorgegangen – die vielen Anfragen würden das Internet über Gebühr belasten. Eine naheliegende Maßnahme ist, Resolver und rekursive DNS-Server einmal erhaltene Antworten eine Weile lang »cachen«, also zwischenspeichern zu lassen. Auf diese Weise können zusätzliche Suchvorgänge vermieden werden, wenn derselbe Name mehrmals hintereinander aufgelöst werden muss (etwa weil er in diversen Verweisen auf Bilder in einer HTML-Seite vorkommt). Jede DNS-Antwort wird vom autoritativen Server mit einer Haltezeit (engl. *time to live*, TTL) versehen, die angibt, wie lange die Antwort maximal in einem Cache-Speicher verbleiben soll.

Cache

Aus diesem Grund fragt der rekursive Server in Wirklichkeit nicht gezielt nach Teilen des gesuchten Namens wie `de` oder `linupfront.de`, sondern immer (also auch zum Beispiel den Root-Level-DNS-Server) nach dem kompletten Namen. Es könnte ja sein, dass einer der befragten Server die richtige Antwort schon im Cache hat, ohne dass er wirklich der letztendlich zuständige Server ist.

Eine Konsequenz des Cachens besteht darin, dass Änderungen an den Daten sich möglicherweise nur schleppend herum sprechen. Angenommen, die Adresse für `www.linupfront.de` ändert sich. Wenn Sie um 9 Uhr morgens `www.linupfront.de` aufgelöst und eine Antwort mit der alten Adresse und einer Haltezeit von 12 Stunden bekommen haben, dann bleibt diese bis um 21 Uhr in Ihrem Cache liegen, und Suchanfragen nach `www.linupfront.de` werden mit der gespeicherten alten Adresse beantwortet. Wenn nun um 10 Uhr die IP-Adresse geändert wird, auf die `www.linupfront.de` verweist, dann bekommen Sie das erst ab 21 Uhr mit, weil Ihr (rekursiver und cachender) DNS-Server vorher nicht auf die Idee kommt, den autoritativen Server noch einmal zu fragen. – Aus diesem Grund sollten Sie bei Änderungen an Ihren DNS-Daten im Hinterkopf behalten, dass diese nicht sofort für die ganze Welt sichtbar werden.

DNS verwendet die unterliegenden Protokolle TCP und UDP; für beide ist ihm der Port 53 zugeordnet. Einfache DNS-Anfragen werden aus Effizienzgründen normalerweise über UDP abgewickelt. TCP kommt zum Einsatz, wenn große Datenmengen transportiert werden müssen.

Transportprotokolle

Übungen

5.3 [1] Wofür steht die TLD `ag`? Um was für eine Art von TLD (generisch, gesponsort, Ländercode, …) handelt es sich?

5.4 [2] Was halten Sie von der gängigen Praxis, denselben Domainnamen in allen möglichen TLDs (`de`, `com`, `info`, …) zu registrieren? Diskutieren Sie die Vor- und Nachteile.

5.5 [3] Verwenden Sie `whois`, um Informationen über einige Ihnen bekannte Domains abzurufen (zum Beispiel `linupfront.de`). Probieren Sie nicht nur `de`-Domains, sondern auch zum Beispiel welche in den TLDs `com` oder `org`. Über welche Organisationen sind diese Namen registriert?

5.6 [2] (Recherchefrage.) Was ist der Unterschied zwischen dem Registrant, dem »Admin-C« und dem »Tech-C« einer Domain?

5.7 [3] (Recherchefrage.) Was ist das Problem mit der sTLD `mobi`? Schlagen Sie zum Beispiel auf `http://en.wikipedia.org` unter `.mobi` nach.

5.3 Linux als DNS-Client

5.3.1 Die C-Laufzeitbibliothek

Es gibt verschiedene Gründe, einen Linux-Rechner als DNS-Client zu konfigurieren: Etwa weil in dem lokalen Netz, wo der Rechner steht, DNS zur Namensauflösung verwendet wird oder weil der Rechner direkt (über einen Provider) ins Internet eingebunden ist und Sie im Web surfen wollen.

Diverse Anwendungsprogramme, zum Beispiel Web-Browser, die Secure Shell oder Diagnosewerkzeuge wie `ping`, akzeptieren Rechnernamen und versuchen diese in IP-Adressen umzuwandeln. In den allermeisten Fällen heißt das, dass die C-Laufzeitbibliothek (`libc`) dazu herangezogen wird – sie enthält zum Beispiel eine Funktion, die zu einem gegebenen Rechnernamen »irgendwie« nach einer IP-Adresse sucht, etwa indem sie in `/etc/hosts` nachschaut. Wenn auf Ihrem Rechner also DNS zur Namensauflösung verwendet werden soll, müssen Sie dafür sorgen, dass die C-Bibliothek das weiß.

Der Vorteil davon, die Namensauflösung komplett in die C-Bibliothek zu stecken, besteht darin, dass die betreffende Funktionalität nur einmal implementiert werden muss und nicht in jedem Programm, das Namensauflösung braucht. Außerdem ist es leicht, zusätzliche oder alternative Namensauflösungsmethoden (etwa NIS) mitzuverwenden.

Dienste zur Namensauflösung

Die aktuelle Linux-C-Bibliothek (`libc6`) verwendet zur Konfiguration der Namensauflösung die Datei `/etc/nsswitch.conf`. Darin ist beispielsweise festgelegt, welche Dienste zur Namensauflösung in welcher Reihenfolge verwendet werden. Daneben finden sich Einträge zur Auflösung von Benutzernamen, Gruppen usw., die uns an dieser Stelle nicht weiter interessieren. Eine genaue Beschreibung von Syntax und Funktionsweise kann in `nsswitch.conf`(5) eingesehen werden.

Der für die Auflösung von Rechnernamen interessante Teil von `/etc/nsswitch.conf` kann etwa so aussehen:

```
hosts: files dns
```

Hier wird also zunächst versucht, Rechnernamen mit Hilfe der lokalen Dateien (namentlich `/etc/hosts`) aufzulösen. Erst wenn dies scheitert, kommt das DNS zum Zug.

Erlaubt wäre in diesem Falle auch der Eintrag `nis`, um die Daten von einem zentralen NIS-Server zu beziehen, oder einer der anderen am Anfang von `/etc/nsswitch.conf` genannten Werte.

Tabelle 5.2: Optionen innerhalb `/etc/resolv.conf`

Option	Wirkung
`debug`	Die regulären Betriebsmeldungen werden auf `stdout` ausgegeben (meist nicht unterstützt).
`ndots` ⟨*n*⟩	Die minimale Anzahl von Punkten im Namen, bei welcher der Resolver direkt nachsieht, ohne auf die Suchliste zuzugreifen.
`attempts` ⟨*n*⟩	Die Anzahl der Anfragen an einen Server, bis der Resolver aufgibt. Maximalwert ist 5.
`timeout` ⟨*n*⟩	Der Anfangs-Timeout für Abfrageversuche in Sekunden. Maximalwert ist 30.
`rotate`	Nicht nur der erste, sondern alle Server werden abwechselnd befragt.
`no-check-names`	Deaktiviert die standardmäßige Überprüfung, ob zurückgelieferte Hostnamen nur gültige Zeichen enthalten.

5.3.2 Der Resolver

Resolver

Die eigentliche Arbeit der Namensauflösung übernimmt der sogenannte **Resolver** – er wird von der C-Bibliothek aufgerufen, formuliert eine DNS-Anfrage, schickt diese an den konfigurierten DNS-Server (oder die konfigurierten DNS-Server), wartet auf eine Antwort und reicht diese an die C-Bibliothek zurück, die sie wiederum an das aufrufende Programm weiterleitet.

Traditionell ist der Resolver Bestandteil eines DNS-Softwarepakets, das auch einen DNS-Server enthält – klassischerweise BIND (siehe Kapitel 6). Natürlich müssen Sie keinen DNS-Server betreiben, um einen Resolver benutzen zu wollen, und aus diesem Grund gehört bei den meisten heutigen Linux-Distributionen der Resolver »pakettechnisch« zur C-Bibliothek. Trotzdem hängen die Fähigkeiten des Standard-Linux-Resolvers bis zu einem gewissen Grad davon ab, welcher BIND-Version er ursprünglich entstammt.

Die zentrale Konfigurationsdatei für den Resolver heißt `/etc/resolv.conf`. Hier werden zum Beispiel die DNS-Server konfiguriert, die der Resolver konsultieren soll. Dazu existieren fünf Hauptdirektiven:

`domain` ⟨*Name*⟩ (lokale Domain) Anhand dieses Eintrags versucht der Resolver, unvollständige Rechnernamen (typisch solche, die keinen Punkt enthalten) um einen Domainanteil zu ergänzen.

Was ein unvollständiger Name ist, wird von der Option `ndots` (siehe Tabelle 5.2) bestimmt.

`search` ⟨$Domain_1$⟩ ⟨$Domain_2$⟩ **…** (Suchliste) Alternativ zu einem einzigen Eintrag mittels `domain` kann mit `search` auch eine Liste mit mehreren Ergänzungen für unvollständige Rechnernamen angegeben werden. Die Einträge in der Liste werden durch Leerzeichen getrennt. Zunächst wird versucht, den unveränderten Rechnernamen aufzulösen. Wenn dies scheitert, werden die Listeneinträge der Reihe nach angehängt und diese Namen ausprobiert. `domain` und `search` schließen sich gegenseitig aus; tauchen beide in der Konfiguration auf, gilt der textuell letzte Eintrag in der Datei.

Wie viele Einträge in der Liste erlaubt sind, hängt von der Version des Resolvers ab. Der von BIND 8 läßt maximal 6, der von BIND 9 sogar 8 Einträge zu.

```
nameserver 192.168.10.1
nameserver 192.168.0.99
search     foo.example.com bar.example.com example.com
```

Bild 5.2: Beispiel für `/etc/resolv.conf`

`nameserver` ⟨*IP-Adresse*⟩ (Lokaler DNS-Server) Der lokale Resolver wird den hier eingetragenen DNS-Server befragen. Es sind bis zu drei `nameserver`-Direktiven erlaubt, die bei Bedarf nacheinander abgefragt werden.

`sortlist` ⟨*IP-Adresse*⟩[/⟨*Netzmaske*⟩] (Sortierung) Falls zu einem Rechnernamen mehrere Adressen zurückgeliefert werden, so wird die hier eingetragene bevorzugt. Bis zu 10 Einträge sind in der Sortierliste möglich.

`options` ⟨*Option*⟩ (Optionen) Hiermit lassen sich besondere Resolver-Einstellungen vornehmen, die in der Tabelle 5.2 mit den Vorgabewerten aufgelistet sind. In der Praxis werden diese selten bis nie verändert.

Eine typische `/etc/resolv.conf`-Datei sehen Sie in Bild 5.2.

5.3.3 Verhalten bei Zeitüberschreitungen

Der Resolver befragt den ersten eingetragenen DNS-Server mit einem Timeout-Wert von fünf Sekunden oder dem optional in `/etc/resolv.conf` eingetragenen Wert. Das bedeutet, der Resolver wartet normalerweise 5 Sekunden, ob er eine Antwort erhält. Bekommt er keine bzw. stattdessen eine ICMP-Fehlermeldung (*network/host/port unreachable*), wird die Befragung auf unterschiedliche Arten weitergeführt.

Ist nur ein einziger Server bekannt, wird nach einer fehlgeschlagenen Anfrage der Timeout-Wert verdoppelt und die Anfrage an den gleichen Server wiederholt.

Sind hingegen mehrere Server in `/etc/resolv.conf` eingetragen, werden diese nacheinander in der angegebenen Reihenfolge befragt. Erst wenn von keinem eine Antwort erhalten wurde, wird auch hier der Timeout-Wert verdoppelt, dann aber durch die Anzahl der Server geteilt und die Runde erneut gestartet.

Schlägt auch dies fehl, wird wiederum der Timeout verdoppelt und die Anfrage wiederholt, bis die maximale Anzahl an Versuchen erreicht ist. Bis BIND 8.2 waren das insgesamt 4, ab BIND 8.2.1 nur noch zwei Versuche, was die Auflösung deutlich beschleunigt.

Übungen

5.8 [1] Nach welchen Namen fragt der Resolver die DNS-Server bei der folgenden Konfiguration in `/etc/resolv.conf`, wenn er eine Anfrage nach (a) `www` (b) `www.example` (c) `www.example.com` bekommt?

```
nameserver 10.11.12.13
nameserver 10.22.33.44
domain sub1.example.com
search sub1.example.com sub2.example.com example.com
```

5.9 [2] Betrachten Sie die `/etc/resolv.conf`-Datei auf Ihrem Rechner und erklären Sie die einzelnen Zeilen. Fällt Ihnen etwas Ungewöhnliches auf? Was würden Sie anders oder besser machen?

5.10 [2] Was ist der Unterschied zwischen den beiden Kommandos `domainname` und `dnsdomainname`? Können Sie mit `dnsdomainname` den DNS-Domainnamen eines Rechners einstellen?

5.4 DNS testen

5.4.1 Einleitung

Es ist unwahrscheinlich, dass ein komplexes und wichtiges Softwaresystem wie DNS immer auf Anhieb funktioniert. Als Administrator müssen Sie also damit rechnen, eine existierende DNS-Konfiguration testen zu müssen – entweder Ihre eigene oder auch nur eine Client-Konfiguration Ihrer Rechner. Hierzu stehen verschiedene Werkzeuge zur Verfügung, allen voran das moderne `dig`, aber auch traditionellere Programme wie `nslookup` oder `host`.

5.4.2 Das Kommando `dig`

`dig`, der *Domain Information Groper*, ist das wohl flexibelste und mächtigste Werkzeug für DNS-Tests. Im einfachsten Fall können Sie damit die IP-Adresse zu einem Namen herausfinden:

```
$ dig www.linupfront.de
```

`dig` enthält im wesentlichen seinen eigenen (instrumentierten) Resolver und kann nahezu beliebige DNS-Anfragen stellen und die dazugehörigen DNS-Antworten dekodieren. Es unterstützt eine Vielzahl von Optionen, mit denen Sie den Umfang und die Art der Ausgabe beeinflussen können.

Eine typische Anfrage mit `dig` liefert ein Ergebnis, das Sie mit Informationen nahezu erschlägt:

```
$ dig www.linux.org

; <<>> DiG 9.2.2 <<>> www.linux.org
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6775
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 0

;; QUESTION SECTION:
;www.linux.org.                 IN      A

;; ANSWER SECTION:
www.linux.org.          41436   IN      A       198.182.196.56

;; AUTHORITY SECTION:
linux.org.              41436   IN      NS      ns.invlogic.com.
linux.org.              41436   IN      NS      ns0.aitcom.net.

;; Query time: 75 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Fri Aug 22 08:35:10 2003
;; MSG SIZE  rcvd: 104
```

Die erste Zeile wiederholt die Versionsnummer von `dig` und unsere Kommandozeile. Der folgende »HEADER« gibt einen Überblick über die hier gezeigte Antwort: Sie hatte die Folgenummer 6775 und repräsentiert einen korrekt abgeschlossenen Vorgang (`NOERROR`). Die »`flags`« in der letzten Kopfzeile geben einige weitere Informationen: `qr` besagt, dass es sich bei dem gezeigten DNS-Paket tatschlich um eine Antwort handelt (da `dig` nur Antworten dekodiert und keine Anfragen, kommt das nicht als Überraschung). `rd` (oder *recursion desired*) heißt, dass der anfragende Resolver, in unserem Falle `dig`, »rekursive« Bearbeitung der Anfrage wünschte (der angefragte Server soll die komplette Arbeit machen und erst mit

der endgültigen Antwort wieder kommen) und `ra` (*recursion available*), dass der angefragte Server das tatsächlich zu machen bereit war. Ein `aa` bei den Flags besagt, dass die hier gezeigte Antwort autoritativ ist, also von einem »echten« DNS-Server für die Domain kommt und nicht aus irgendeinem Cache entlang des Wegs. Der Rest der Zeile gibt an, dass `dig` nach einem Namen gefragt und eine Adresse als direkte Antwort erhalten hat; die »AUTHORITY SECTION« enthält zwei Antworten und eine mögliche »ADDITIONAL SECTION« ist leer.

Die »QUESTION SECTION« gibt noch einmal die Anfrage wieder, die hier beantwortet wird, während die »ANSWER SECTION« die eigentliche Antwort liefert. Die »AUTHORITY SECTION« beschreibt die Herkunft der Antwort. Gegebenenfalls kommt auch noch eine »ADDITIONAL SECTION« dazu, die zusätzliche Informationen enthält; wenn Sie zum Beispiel eine Anfrage nach einem `MX`-Datensatz stellen, haben Sie gute Chancen, auch gleich den `A`-Datensatz für den Mailserver zu erhalten, für den Sie sonst eine gesonderte DNS-Anfrage stellen müßten.

`dig` befragt zwar standardmäßig die in `/etc/resolv.conf` genannten DNS-Server, aber ignoriert den Rest der Datei. Sie müssen es also mit kompletten Rechnernamen füttern, da zum Beispiel die `search`- und `domain`-Direktiven nicht beachtet werden.

Ganz am Schluss finden Sie noch Informationen darüber, welcher DNS-Server wann befragt wurde und wie lange die Bearbeitung der Anfrage gedauert hat.

Die Syntax von `dig` lautet allgemein

```
dig [@⟨Server⟩] ⟨Name⟩ [⟨Typ⟩] [⟨Klasse⟩]
```

Dabei bedeuten:

⟨*Server*⟩ Der abzufragende DNS-Server kann über seine IP-Adresse festgelegt werden. Ohne diese Angabe wird auf den oder die in `/etc/resolv.conf` vorgegebenen Server zugegriffen.

⟨*Name*⟩ Informationen über diesen DNS-Namen (Rechner- oder Domainnamen) werden gesucht.

⟨*Typ*⟩ Der Typ der gewünschten Antwort. Voreinstellung ist der Typ `a`, es wird also versucht, zu einem Rechnernamen eine oder mehrere Netzwerkadressen zu ermitteln.

Auch andere Typangaben wie `mx`, `ns` oder `soa` sind erlaubt. `axfr` versucht, einen Zonentransfer auszulösen.

⟨*Klasse*⟩ Voreinstellung ist `in` für »Internet«, daneben ist auch die Angabe `any` möglich. Andere Klassen sind nicht von praktischer Relevanz.

Wenn Sie sich schon etwas mit DNS auskennen (etwa weil Sie diese Unterlage zum zweiten Mal lesen), dann wird Ihnen auffallen, dass die Ausgabe von `dig` einer Zonendatei verdächtig ähnlich sieht. Tatsächlich können Sie über den Anfragetyp `axfr` etwas erhalten, was Sie mit etwas Editieren zu einer gültigen Zonendatei machen können. `dig` hilft Ihnen also zum Beispiel bei der Migration von einem DNS-Server, der seine Zonendaten in einem proprietären Format speichert, nach BIND.

`dig` erlaubt die Rückwärtsauflösung von IP-Adressen zu Rechnernamen auf die naheliegende Weise:

```
$ dig 136.52.162.212.in-addr.arpa ptr
```

liefert das gewünschte Ergebnis. Dasselbe geht aber auch bequemer, nämlich mit

```
$ dig -x 212.162.52.136
```

`dig` unterstützt eine ganze Menge von Kommandooptionen, die wir hier nicht alle ausführlich besprechen können (lesen Sie `dig`(1)). Hier nur ein paar Beispiele für die wichtigsten und interessantesten:

Kompakte Ausgabe Mit der Option `+short` beschränkt `dig` sich bei der Ausgabe auf das Wesentliche:

```
$ dig www.linupfront.de +short
212.162.52.136
```

Gekürzte Ausgabe Mit den Optionen `+noquestion`, `+noanswer`, `+noauthority`, `+noadditional` und `+nostats` können Sie selektiv die QUESTION SECTION, ANSWER SECTION, AUTHORITY SECTION, ADDITIONAL SECTION sowie die abschließende Statistik ausblenden. `+nocomments` unterdrückt fast sämtliche DNS-Kommentare (die Zeilen mit »;« am Anfang) und Leerzeilen in der Ausgabe.

TCP Mit der Option `+tcp` stellt `dig` seine Anfragen nicht wie üblich über UDP, sondern über TCP. Dies ist sinnvoll, wenn Sie zum Beispiel einen Paketfilter testen wollen, der DNS-Daten filtern oder durchlassen soll. `+notcp` verbietet die Verwendung von TCP, wobei bestimmte Arten von DNS-Anfragen dann nicht mehr ausgeführt werden können.

Suchliste Mit `+search` hält `dig` sich an die Suchliste, die in `/etc/resolv.conf` mit `domain` oder `search` definiert ist.

Rekursive und iterative Anfragen Mit `+norecurse` stellt `dig` keine rekursive, sondern eine iterative Anfrage (der angefragte DNS-Server soll den Namen also nicht komplett auflösen). Die Voreinstellung ist `+recurse`. `+trace` bringt `dig` dazu, alle Stationen einer Anfrage (beginnend bei den Root-Level-Servern) auszugeben. Auch in diesem Fall arbeitet `dig` iterativ.

Mehr über Optionen Zu jeder Option gibt es eine Option, die das Gegenteil bewirkt – wenn `+recurse` rekursive Namensauflösung einschaltet, dann verbietet `+norecurse` sie. Da rekursive Namensauflösung die Voreinstellung ist, scheint `+recurse` auf den ersten Blick nicht dringend nötig zu sein. Allerdings können Sie in der Datei `~/.digrc` Ihre Lieblingsoptionen ablegen; wenn Sie zum Beispiel keinen gesteigerten Wert auf die Abfragestatistik legen und normalerweise die Suchliste verwenden wollen, könnten Sie zum Beispiel die Zeilen

```
+search
+nostats
```

in `~/.digrc` aufnehmen. Ein `dig`-Aufruf wie

```
$ dig www.linupfront.de +nosearch
```

würde dann trotzdem die Suchliste ignorieren.

5.4.3 Das Kommando `nslookup`

Ein anderes Programm, mit dem Sie Daten von DNS-Servern abfragen können, ist `nslookup`. `nslookup` gibt es schon länger als `dig`; es ist nicht ganz so flexibel und

wird etwas anders bedient. Insbesondere können Sie es nicht nur »nichtinteraktiv« direkt von der Kommandozeile aus benutzen, sondern es unterstützt auch einen »interaktiven Modus«.

Ähnlich wie bei `dig` führt `nslookup` im nichtinteraktiven Modus einmalige Anfragen an einen DNS-Server aus, etwa so:

```
$ nslookup www.linupfront.de
Server:         127.0.0.1
Address:        127.0.0.1#53

Name:   www.linupfront.de
Address: 212.162.52.136
```

Das Programm `nslookup` befragt standardmäßig den oder die in der Datei `/etc/resolv.conf` angegebenen DNS-Server nach dem Namen.

Dabei ist die Strategie von `nslookup` anders als die des Resolvers; während der Resolver die DNS-Server der Reihe nach durchprobiert und am Schluss wieder mit dem ersten anfängt, wiederholt `nslookup` seine Anfragen an den ersten Server so lange, bis es aufgibt, und macht erst dann mit dem zweiten weiter. Der Sinn dahinter ist, dass die Fehlersuche im DNS vereinfacht wird, wenn `nslookup` immer nur mit einem Server redet statt mit mehreren gleichzeitig.

Wenn Sie einen anderen DNS-Server ansprechen wollen, könnnen Sie dessen Namen oder dessen Adresse als zweites Argument auf der Kommandozeile übergeben werden. Zum Beispiel:

```
$ nslookup www.heise.de ns1.example.com
```

Interaktiver Modus

Sie können `nslookup` im interaktiven Modus starten, indem Sie das Programm ohne Argumente aufrufen. Es meldet sich dann mit der Eingabeaufforderung »>«, wo Sie beliebige Namen angeben können. Per Voreinstellung fragt `nslookup` nach Einträgen, die die zum Namen gehörige IP-Adresse enthalten. Um eine Übersicht über alle Befehle zu bekommen, die `nslookup` bietet, verwenden Sie den Befehl `help` oder ein `?`.

Der interaktive Modus ermöglicht die Ausgabe weiterer Informationen. So können in diesem Modus nicht nur einzelne Namen aufgelöst werden, sondern auch nach beliebigen DNS-Einträgen gesucht oder die gesamte Zoneninformation einer Domäne aufgelistet werden.

```
$ nslookup
Default Name Server: dns.example.com
Address:  192.168.0.99
> foo.example.com
Name Server:  server.training
Address:  192.168.0.99
Non-authoritative answer:
Name:     foo.example.com
Address:  192.168.0.1
> exit
```

`nslookup` ist bekannt und verbreitet, gilt aber inzwischen als verpönt (*deprecated*) und wird irgendwann verschwinden. Der designierte Ersatz ist `dig` (Abschnitt 5.4.2). Neuere Versionen von `nslookup` machen Sie auf diesen Umstand mit einer fetten Warnmeldung aufmerksam, die Sie aber mit der Option `-silent` unterdrücken können.

5.4.4 Das Kommando `host`

`host` erlaubt ebenfalls die Abfrage von DNS-Informationen. Die Syntax lautet dabei:

`host` [⟨*Option*⟩ …] ⟨*Rechner*⟩ [⟨*nameserver*⟩]

Mit der Eingabe »`host rechner02`« würde also der voreingestellte Server nach der IP-Adresse des Rechners `rechner02` befragt werden. Das Ergebnis sähe dann so aus:

```
$ host rechner02
rechner02.example.com has address 192.168.0.2
```

Natürlich ist auch die Auflösung von Namen, die zu einer bekannten Adresse gehören, möglich:

```
$ host 192.168.0.250
250.0.168.192.in-addr.arpa domain name pointer quux.example.com
```

Mit der Option `-t` lassen sich verschiedene Informationen über eine Domain abfragen, etwa:

```
$ host -t ns example.com.
example.com. name server dns.example.com.
```

Kommandos in diesem Kapitel

dig	Sucht Informationen im DNS (sehr komfortabel)	dig(1)	73
host	Sucht Informationen im DNS	host(1)	76
nslookup	Sucht Informationen im DNS	nslookup(1)	75
whois	Gibt Informationen über DNS-Domains aus	whois(1)	65

Zusammenfassung

- Die Auflösung von Rechnernamen zu IP-Adressen ist eine zentrale Dienstleistung der TCP/IP-Netzinfrastruktur. Zur Auflösung können statische Tabellen, NIS oder das *Domain Name System* herangezogen werden.
- Domainnamen werden von rechts nach links gelesen, Namensteile (*labels*) werden durch Punkte getrennt und dürfen jeweils maximal 63 Zeichen lang sein. Namensteile können bis zu 127 Ebenen tief verschachtelt werden.
- An der Wurzel der Hierarchie steht die *root-level domain*; es gibt generische, gesponsorte, länderspezifische und Infrastruktur-Top-Level-Domains, deren Betreiber sich um die Registrierung der unmittelbar darunterliegenden Domains kümmern. Alles weitere ist Benutzersache.
- Die Verwaltung der TLD-Liste und der Root-Level-DNS-Server ist Aufgabe der ICANN, die dazu vom US-Handelsministerium beauftragt wurde. Für die einzelnen TLDs gibt es jeweils Organisationen, die die Datenbank verwalten und Domaineinträge direkt, über Subunternehmer oder über Registrars akzeptieren.
- Eine Zone ist eine Menge von Namen, die vom selben einzelnen DNS-Server verwaltet wird. Domains verhalten sich zu Zonen wie Unix-Dateinamen zu Unix-Dateisystemen.
- Der »Resolver« bildet die Client-Seite von DNS. Er ist Teil der C-Bibliothek und wird in der Datei `/etc/resolv.conf` konfiguriert. Die Datei `/etc/nsswitch.conf` enthält allgemeine Konfigurationseinstellungen für die Namensauflösung durch die C-Bibliothek – nicht nur Rechnernamen, sondern auch Benutzernamen, Gruppennamen und so weiter.
- `nslookup`, `host` und `dig` sind nützliche Programme zur Fehlersuche in einem DNS-System.

Literaturverzeichnis

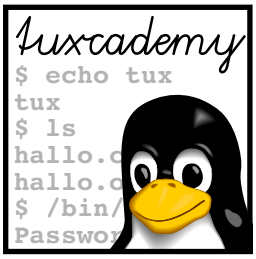
Din01 Daniel Dingeldey. »`Mitwohnzentrale.de` – Die Entscheidungsgründe des BGH«, Oktober 2001. Abgerufen am 19.4.2011. `http://www.domain-recht.de/magazin/domain-news-2001/mitwohnzentrale-de-die-entscheidungsgruende-des-bgh-id48.html`

RFC0882 P. Mockapetris. »Domain Names – Concepts and Facilities«, November 1983. `http://www.ietf.org/rfc/rfc0882.txt`

RFC0883 P. Mockapetris. »Domain Names – Implementation and Specification«, November 1983. `http://www.ietf.org/rfc/rfc0883.txt`

RFC1034 P. Mockapetris. »Domain Names – Concepts and Facilities«, November 1987. `http://www.ietf.org/rfc/rfc1034.txt`

RFC1035 P. Mockapetris. »Domain Names – Implementation and Specification«, November 1987. `http://www.ietf.org/rfc/rfc1035.txt`



6

Der DNS-Server BIND

Inhalt

6.1 Überblick 80
6.2 BIND-Grundlagen 81
6.2.1 Serverprogramm und Dateien 81
6.2.2 Die Konfigurationsdatei `named.conf` 82
6.3 Ein *caching-only* DNS-Server 84
6.4 BIND steuern mit `ndc` und `rndc` 85

Lernziele

- Den DNS-Server BIND kennenlernen
- Konfigurations- und einfache Zonendateien verstehen
- BIND als *caching-only* DNS-Server einrichten können
- Die Kommandos `ndc` und `rndc` einsetzen können

Vorkenntnisse

- Kenntnisse über Linux-Systemkonfiguration
- TCP/IP-Kenntnisse
- DNS-Grundkenntnisse (siehe Kapitel 5)

6.1 Überblick

BIND Die verbreitetste DNS-Serversoftware für Linux/Unix-Systeme ist BIND (kurz für *Berkeley Internet Name Domain*). BIND wurde ursprünglich – bis zur Version 4.8.3 – als Projekt der *Computer Systems Research Group* (CSRG) der Universität von Kalifornien in Berkeley entwickelt und von der *US Defense Advanced Research Projects Administration* (DARPA) finanziert. Seit Version 4.9 (ungefähr 1988) ist Paul Vixie, damals noch Angestellter der Digital Equipment Corporation (heute Bestandteil von Hewlett-Packard), Oberherr über BIND.

ISC BIND wird heute vom *Internet Systems Consortium* (ISC) gewartet. ISC ist eine nicht profitorientierte Organisation, die sich aus Spenden aus der Industrie und von anderen Stiftungen (inklusive der *National Science Foundation*, dem US-amerikanischen Pendant zur DFG) sowie durch Support- und Entwicklungsverträge und Spenden finanziert. Paul Vixie ist daran maßgeblich beteiligt und kümmert sich nach wie vor um BIND.

Zu den anderen Projekte des ISC gehört auch der ISC-DHCP-Server. Es dient auch als Heimat diverser anderer wichtiger Projekte (etwa INN und Lynx), stellt Infrastruktur für andere Open-Source-Projekte wie NetBSD oder `kernel.org` zur Verfügung und betreibt sekundäre DNS-Server für über fünfzig Top-Level-Domains sowie das *DNS Operations, Analysis and Research Center* (OARC).

Das ISC wurde 1994 als *Internet Software Consortium* gegründet, um sich um die Weiterentwicklung von BIND zu kümmern. 2004 erfolgte die Umbenennung in *Internet Systems Consortium*.

Von BIND sind verschiedene Versionen im Umlauf: Aktuell ist BIND 9, eine komplette Neuimplementierung unter Gesichtspunkten der Sicherheit und mit fortgeschrittenen Eigenschaften wie IPv6-Unterstützung, Views oder DNSSEC (*DNS Security Extensions*). Oft im Einsatz finden Sie aber noch die Vorläuferversion BIND 8; inzwischen im Linux-Umfeld glücklicherweise nahezu ausgestorben, aber in proprietären Unix-Versionen mitunter noch anzutreffen ist BIND 4. BIND-Versionen vor 9 sind notorisch für Sicherheitsprobleme – wenn Sie irgend können, sollten Sie also BIND 9 verwenden.

BIND 5 bis BIND 7 gab es nicht; die Versionsnummern aller wesentlichen Berkeley-Programme wurden mal in einem großen Rundumschlag willkürlich auf »8« gesetzt.

BIND 8 und BIND 9 werden sehr ähnlich konfiguriert, während BIND 4 eine völlig andere Syntax für seine Konfigurationsdatei verwendet (was die Datendateien angeht, sind die Unterschiede nicht so groß). In dieser Schulungsunterlage konzentrieren wir uns auf BIND 9, die empfehlenswerte Version.

Bis vor nicht allzulanger Zeit kam in den LPIC-Prüfungszielen sogar noch BIND 4 vor. Das ist glücklicherweise nicht mehr so. Wo nötig, weisen wir auf Unterschiede zwischen BIND 8 und BIND 9 hin.

andere Implementierungen Es gibt diverse andere Implementierungen von DNS-Servern, die aber zumeist nicht den vollen vorgeschriebenen Funktionsumfang unterstützen. Manche funktionieren zum Beispiel nur als autoritative und andere nur als rekursive DNS-Server. In der Praxis stellt das nicht notwendigerweise ein Problem dar, so dass Sie deren Vorzüge, etwa leichtere Konfigurierbarkeit, leistungsfähigere Datenhaltung oder höhere Sicherheit, in Anspruch nehmen können, solange Sie keine Funktionalität benötigen, die nicht angeboten wird. Mit etwas Umsicht ist es auch ohne weiteres möglich, etwa BIND als rekursiven und einen anderen DNS-Server als autoritativen Server einzusetzen.

DJBDNS Eine wegen ihrer Sicherheit (im Vergleich zu BIND) relativ populäre, aber in mancher Hinsicht unkonventionelle DNS-Implementierung ist *DJBDNS* von Dan

J. Bernstein, dem Autor des Mailservers *qmail*. Für kleine Systeme interessant ist möglicherweise `dnsmasq` von Simon Kelley, das einen sehr simplen DNS- und DHCP-Server implementiert, der den Inhalt von `/etc/hosts` auf dem betreffenden System per DNS zur Verfügung stellt und für andere DNS-Namen als cachender Server dient. `dnsmasq` braucht einen Bruchteil der Ressourcen von BIND und ist sehr einfach zu administrieren. dnsmasq

Übungen

6.1 [2] Welche DNS-Server stehen für Ihre Linux-Distribution zur Verfügung? Welche Einschränkungen haben die verschiedenen Programme – können sie etwa nur als autoritativer oder nur als rekursiver Server dienen? Welche anderen Stärken und Schwächen haben sie?

6.2 BIND-Grundlagen

6.2.1 Serverprogramm und Dateien

Der eigentliche DNS-Server im BIND-Paket heißt `named`. `named` ist ein typisches »freistehendes« Unix-Daemonprogramm und wird für gewöhnlich beim Hochfahren des Systems durch ein passendes Init-Skript automatisch gestartet. Alternativ können Sie Kommandos wie named

```
# /etc/init.d/bind9 start                    Debian, Ubuntu
# /etc/init.d/named start                    SUSE-Distributionen
# rcnamed stop                               SUSE-Distributionen
```

verwenden.

Damit BIND *wirklich* automatisch beim Systemstart mit gestartet wird, müssen Sie möglicherweise über die Paketinstallation hinaus noch weitere Schritte durchführen. Bei den Novell/SUSE-Distributionen ist zum Beispiel noch ein

```
# insserv named
```

nötig, damit BIND in die Runlevel-Struktur integriert wird. (Oder Sie verwenden den YaST-Runlevel-Editor.) BIND gehört zu den »Netzwerk-Serverdiensten« und sollte nach LSB daher in den Runlevels 3 und 5 laufen.

Gegebenenfalls können Sie das Programm auch »manuell« über seinen Namen aufrufen:

```
# /usr/sbin/named
```

Dies ist für gewisse Tests nützlich.

Beim Start liest `named` zunächst seine Konfigurationsdatei, `named.conf`. In Abhängigkeit von den darin enthaltenen Einstellungen greift er dann möglicherweise auf weitere Dateien zu, zum Beispiel Zonendateien (engl. *zone files*), die DNS-Daten enthalten, die der Server anbieten soll. named.conf Zonendateien

Die Distributionen sind sich uneins, wo die BIND-Konfiguration und insbesondere die Zonendateien abgelegt werden sollen. Bei den Red-Hat-Distributionen finden Sie die Konfigurationsdatei in `/etc/named.conf` und die Zonendateien in `/var/named`.

Die SUSE-Distributionen verhalten sich wie die von Red Hat. Allerdings stehen die Zonendateien seit SUSE 9.0 in `/var/lib/named`.

```
// BIND-Konfigurationsdatei

options {
        directory "/var/cache/bind";
        listen-on { 192.168.0.254; 127.0.0.1; };
        statistics-file "/var/log/bind/named.stats";
};

zone "." {
        type hint;
        file "/etc/bind/db.root";
};

zone "0.0.127.in-addr.arpa" {
        type master;
        file "/etc/bind/db.127";
};

zone "localhost" {
        type master;
        file "/etc/bind/db.local";
};

zone "example.com" {
        type master;
        file "/etc/bind/db/example.com";
};

zone "33.22.11.in-addr.arpa" {
        type master;
        file "/etc/bind/db/11.22.33";
};
```

Bild 6.1: Die Datei `named.conf` (Beispiel)

Debian GNU/Linux stellt die komplette Konfiguration (`named.conf` und Zonendateien) nach `/etc/bind`. Dasselbe gilt auch für Ubuntu.

Die meisten Distributionen haben Konventionen dafür, wie Sie Ihre eigenen Erweiterungen und Zonendateien in die BIND-Konfiguration integrieren können. Sie tun gut daran, die Beispieldateien zu studieren, die Ihre Distribution mitliefert, und zu verstehen, was dort wie gemacht wird – selbst wenn Sie sich letzten Endes entscheiden, die Systemvorgaben zu ignorieren und Ihre eigene Konfiguration umzusetzen.

Mit der Option `-c` können Sie beim Start von `named` eine alternative Konfigurationsdatei vorgeben. Dies ist nützlich zu Testzwecken oder wenn Sie, etwa aus Sicherheitsgründen, mehrere unabhängige BIND-Instanzen auf demselben Rechner laufen lassen wollen. In diesem Fall müssen Sie normalerweise auch das Init-Skript anpassen.

6.2.2 Die Konfigurationsdatei `named.conf`

Die Datei `named.conf` enthält allgemeine Konfigurationseinstellungen für den DNS-Server – neben Konfigurationsoptionen im engeren Sinne, die bestimmte Eigenschaften ein- oder ausschalten oder Vorgaben machen, auch kryptografische Schlüssel oder eine Liste der Root-Level-DNS-Server – sowie gegebenenfalls Verweise auf Zonendateien für Zonen, für die der Server autoritativ ist.

Syntax

Die Syntax der Konfigurationsdatei ist vage an die Programmiersprache C angelehnt. Kommentare in der Konfigurationsdatei können auf drei Arten kenntlich gemacht werden – sie beginnen mit # und reichen bis zum Zeilenende (wie bei der Shell), beginnen mit // und reichen bis zum Zeilenende (wie in C++) oder beginnen mit /* und reichen bis zu einem */, auch über Zeilenenden hinweg (wie in C).

Kommentare

Bild 6.1 zeigt ein kleines Beispiel für eine einfache named.conf-Datei.

Generelle Eigenschaften

Zuerst erfolgt im Abschnitt options die Festlegung einiger genereller Eigenschaften des DNS-Servers. Die Einträge der Beispieldatei in Bild 6.1 haben dabei folgende Bedeutung:

directory /etc/bind In diesem Verzeichnis befinden sich die Zonendateien. Das Verzeichnis /etc/bind können Sie problemlos durch einen anderen Pfad ersetzen; der Standardwert ist distributionsabhängig.

listen-on { 192.168.0.254; 127.0.0.1; } named wartet in der Regel an allen vorhandenen Netz-Schnittstellen auf Port 53 auf Anfragen. Falls ein DNS-Server mehrere IP-Adressen besitzt und nicht auf allen antworten soll, können Sie dies mit der Angabe listen-on einschränken. Im Beispiel beantwortet der DNS-Server also nur Anfragen auf den IP-Adressen 192.168.0.254 und 127.0.0.1. Das ist sinnvoll, wenn etwa auf einem Internet-Zugangsrechner ein DNS-Server läuft und Informationen über das innere Netz nicht von außen abgefragt werden können sollen. Mit Hilfe der listen-on-Anweisung können Sie dem named außerdem einen anderen Port als 53 zuzuweisen, etwa Port 1234 für das Netz 192.168.1.0 mittels »listen-on port 1234 { 192.168.1/24 }«. Viel bringen tut das allerdings nicht, da Resolver immer den Port 53 benutzen.

statistics-file "/var/log/bind" An die angegebene Datei werden Meldungen angehängt, wenn BIND ein SIGILL-Signal erhält.

Zoneninformationen

Nach den globalen Eigenschaften folgen verschiedene Abschnitte, jeweils mit zone eingeleitet. Jeder dieser Abschnitte beschreibt, für welchen Bereich von Namen (welche »Zone«) der Server in welcher Datei nachsehen soll, um die Auflösung von Namen zu IP-Adressen durchführen zu können.

Adressen der Root-Level-DNS-Server

Der zone-"."-Abschnitt dient dazu, dem Server die Adressen der Root-Level-DNS-Server mitzuteilen. Da eine rekursive Namensauflösung immer mit einer Anfrage an die Root-Level-DNS-Server beginnt, ist diese Konfiguration für einen rekursiven DNS-Server extrem wichtig; sie dient dazu, dass der lokale DNS-Server sich quasi an seinen eigenen Haaren aus dem Sumpf ziehen kann.

Die in der Konfiguration erwähnte Datei db.root wird in aller Regel zusammen mit BIND installiert. Die Daten der Root-Level-DNS-Server ändern sich extrem selten; sollten Sie jemals eine frische Version benötigen, finden Sie sie etwa unter ftp://ftp.ripe.net/tools/dns/named.root.

Es ist nicht notwendigerweise schlimm, wenn die Datei db.root nicht zu 100% aktuell ist, da sich normalerweise nicht alle Adressen darin gleichzeitig ändern; Anfragen an nicht (mehr) existente Adressen verlangsamen natürlich die Namensauflösung.

Bei älteren BIND-Versionen war es zwingend nötig, eine db.root-Datei in der Konfiguration zu haben, selbst wenn der Rechner gar nicht ans Internet angebunden war. Seit BIND 9 können Sie auf die Datei allerdings verzichten, da BIND die Liste der Root-Level-DNS-Server auch im Programm hart codiert enthält. Trotzdem schadet es nichts, sie hier explizit einzubinden.

Die nächsten beiden zone-Einträge ermöglichen es dem DNS-Server, dem Namen localhost die korrekte IP-Adresse 127.0.0.1 zuzuordnen und umgekehrt. Auch diese Einträge sollten auf jedem DNS-Server vorhanden sein.

Sie können eigene Zonendateien aufstellen und in named.conf in analoger Weise erwähnen, etwa wie in den letzten beiden zone-Einträgen in Bild 6.1 zu sehen. Damit kann Ihr DNS-Server »autoritative« Daten über Ihre Systeme zur Verfügung stellen – im lokalen Netz oder auch dem Internet.

```
options {
  directory "/var/cache/bind";
};

zone "." {
  type hint;
  file "/etc/bind/db.root";
};

zone "0.0.127.in-addr.arpa" in {
  type master;
  file "/etc/bind/db.127";
};
```

Bild 6.2: Konfiguration für einen *caching-only* DNS-Server

Wie die selbstdefinierten Zonendateien heißen und wo sie im Dateisystem liegen sollen, ist zumindest debattierbar. Wir folgen in dieser Unterlage locker der Konvention von Debian GNU/Linux, Zonendateien in `/etc/bind` abzulegen – es handelt sich dabei schließlich im weitesten Sinne um Konfigurationsdateien, die nichts in `/var/lib/bind` (oder so ähnlich) verloren haben. (Trotzdem ist das BIND-Arbeitsverzeichnis unterhalb von `/var` angesiedelt, weil dort transiente Daten abgelegt werden wie etwa Sicherheitskopien von Zonendateien, für die der Server sekundär ist.) Wir verwenden dafür ein Unterverzeichnis `/etc/bind/db` und nennen die Zonendateien so wie die Zonen selbst, weil das später den Umgang mit DNSSEC vereinfacht. Grundsätzlich steht es Ihnen aber frei, den Konventionen Ihrer Distribution zu folgen oder sich selbst eine eigene zu überlegen.

Übungen

6.2 [3] Installieren Sie das BIND-Paket Ihrer Distribution. Wo steht das ausführbare Programm, wo `named.conf`, wo die mitgelieferten Zonendateien und Root-Hints? Wird BIND nach der Installation automatisch beim Booten gestartet, oder müssen Sie noch weitere Schritte ausführen? Wenn ja, welche?

6.3 [2] Vergleichen Sie die `named.conf`-Datei auf Ihrem System mit der in Bild 6.1 gezeigten. Was ist gleich? Was ist anders?

6.3 Ein *caching-only* DNS-Server

Autoritative DNS-Server sind ärgerlich zu betreiben

In vielen Fällen werden Sie nicht selbst einen autoritativen DNS-Server betreiben wollen oder müssen – der administrative Aufwand dafür ist lästig, zumal die meisten Registrars darauf bestehen, dass autoritative Daten für das Internet aus Redundanzgründen nicht nur von einem einzigen, sondern von mindestens zwei autoritativen Servern vorgehalten werden. Es ist also in vielen Fällen eine gute Idee, den autoritativen DNS-Server beispielsweise dem eigenen Provider zu überlassen; die meisten Provider haben entsprechende Angebote und verfügen über die nötige Hardware und das erforderliche geschulte Personal. Oft haben Sie auch über eine geeignete Konfigurationsoberfläche direkten Zugriff auf die autoritativen Daten Ihrer Domain und genießen so beide Vorteile – totale Kontrolle über die Domaindaten und ungestörten Schlummer nachts ohne Angst vor DNS-Ausfällen.

Viel eher naheliegend ist der Betrieb eines ausschließlich *rekursiven* DNS-Servers, der Anfragen von Resolvern aus Ihrem Netz auflöst, ohne selbst Daten im

Internet zur Verfügung zu stellen. Durch seinen Cache hilft ein solcher Server Datenverkehr auf Ihrer Außenanbindung sparen und beschleunigt DNS-Recherchen ungemein, insbesondere wenn Sie vor allem auf ein relativ enges Repertoire von anderen DNS-Namen zugreifen. Der Konfigurationsaufwand für einen solchen *caching-only* Server ist sehr gering, wie Bild 6.2 zeigt – mehr ist (außer den Dateien `root.hint` und `127.0.0.zone`, auf die diese Konfiguration Bezug nimmt) nicht nötig!

Übungen

6.4 [3] Konfigurieren Sie BIND 8 oder 9 auf Ihrem Rechner als *caching-only* DNS-Server. Verwenden Sie `tcpdump` oder `wireshark`, um zu beobachten, was passiert, wenn Sie den Server z. B. mit `dig` dazu bringen, Namen aufzulösen. Was passiert, wenn Sie denselben Namen mehrmals anfragen?

6.4 BIND steuern mit `ndc` und `rndc`

Zum Starten und Stoppen von BIND liefern die meisten Linux-Distributionen passende Init-Skripte mit (Abschnitt 6.2.1). Je nach Distribution können Sie mit dem Init-Skript BIND auch dazu bringen, seine Konfiguration neu einzulesen (mit etwas wie »`/etc/init.d/named reload`«) oder abfragen, ob er läuft (»`/etc/init.d/named status`«). Sie können BIND auch mit Signalen wie `SIGHUP` zu Aktionen veranlassen (etwa seine Konfigurationsdatei neu einzulesen). Als BIND-Administrator werden Sie es aber begrüßen, dass BIND Ihnen weitaus mehr Möglichkeiten zur »Fernsteuerung« des Programms einräumt. Hierzu dienen die Programme `ndc` (für BIND 8 ab 8.2) und `rndc` (für BIND 9). Da wir Sie mit sanfter Gewalt dazu nötigen wollen, BIND 9 zu benutzen, erklären wir zuerst `rndc`. Am Schluss dieses Abschnitts kommen wir dann noch einmal kurz auf `ndc` zurück.

Kontrollkanal

`ndc` und `rndc` reden mit BIND über einen »Kontrollkanal«, entweder ein TCP- oder ein Unix-Domain-Socket. Dies erlaubt gegenüber Signalen wie `SIGHUP` natürlich ungeahnte Flexibilität, weil Sie nicht an eine begrenzte Nummer von Signalen gebunden sind und außerdem Zusatzdaten übertragen können (was Signale Ihnen nicht erlauben).

Die traditionelle Methode der Steuerung von BIND über Signale ist vom ISC aus inzwischen verpönt. Dort heißt es, der Kontrollkanal sei »der Weg der Zukunft«. Gewöhnen Sie sich also schon jetzt an die Kommandos.

Vorbereitungen Glücklicherweise hört BIND 9 nicht auf jeden hergelaufenen `rndc`-Prozess. Damit Ihr Server Sie zur Kenntnis nimmt, sind ein paar Vorbereitungen nötig. Zunächst müssen Sie in der Konfigurationsdatei `named.conf` erklären, dass Sie `rndc` zu benutzen gedenken:

```
controls {
  inet * allow { any; } keys { "rndc-key"; };
};
```

Dies bringt BIND zum Beispiel dazu, auf allen verfügbaren Schnittstellen auf Verbindungen von `rndc` zu lauschen.

`rndc` und BIND unterhalten sich, wenn Sie nicht in `controls` mit etwas wie

```
inet * port 8888 allow ◁◁◁◁◁
```

etwas Anderes verfügen, über den TCP-Port 953.

`rndc` muss sich im Beispiel durch den Schlüssel `rndc-key` identifizieren, den Sie wie folgt definieren können:

```
key "rndc-key" {
  algorithm hmac-md5;
  secret "LcZ+GcGU+VX4jAyCVtXnIA==";
};
```

Die magische Zeichenkette hinter `secret` ist dabei ein Base-64-codiertes Kennwort, das sowohl BIND als auch `rndc` zugänglich sein muss.

Wenn Sie keine besseren Ideen haben, können Sie das Programm `dnssec-keygen` verwenden, um einen passenden Schlüssel zu erzeugen. Versuchen Sie etwas wie

```
$ cd /tmp
$ dnssec-keygen -a HMAC-MD5 -b 128 -n HOST foo
Kfoo.+157+02283                                   Datei(en) mit dem Schlüssel
$ cut -d' ' -f7 <Kfoo.+157+02283.key
LcZ+GcGU+VX4jAyCVtXnIA==                                          Tadaa!!
```

Sie können sich auch ein nettes Kennwort ausdenken und es mit einem Programm wie `mimencode` codieren:

```
$ echo Hundekuchen | mimencode
SHVuZGVrdWNoZW4K
```

Diese Sorte Kennwort ist natürlich viel weniger zufällig (und damit viel weniger sicher) als das, was `dnssec-keygen` so produziert. Als Kompromiss könnten Sie sich Zufallsbytes aus der üblichen Quelle besorgen:

```
$ dd if=/dev/urandom bs=16 count=1 2>/dev/null | mimencode
rabUDzoYA5L3haoKKdKupw==
```

Oder Sie verwenden `rndc-confgen` (wie unten erwähnt).

Wenn Sie die `key`-Definition in der Datei `named.conf` stehen haben, sollten Sie dafür sorgen, dass diese Datei nicht für alle Benutzer lesbar ist (sonst kann jeder Benutzer an Ihrem BIND herumbasteln). Alternativ können Sie die `key`-Definition in eine eigene Datei – etwa `named-rndc.conf` – auslagern und diese dann mit

```
include "/etc/bind/named-rndc.conf";                  Verzeichnis evtl. anders
```

in die eigentliche `named.conf` einbinden.

Für `rndc` brauchen Sie eine Datei namens `rndc.conf` im selben Verzeichnis, wo auch die `named.conf`-Datei steht. Dort sollten Sie auf den Server und den gewünschten Schlüssel verweisen:

```
# rndc.conf
options {
  default-server localhost;
  default-key "rndc-key";
};
key "rndc-key" {
  algorithm hmac-md5;
  secret "LcZ+GcGU+VX4jAyCVtXnIA==";
};
```

(den `key`-Block können Sie aus der `named.conf`-Datei übernehmen).

Das Kommando rndc-confgen schreibt eine Datei auf seine Standardausgabe, die dem gezeigten Beispiel sehr ähnelt (der Schlüssel ist natürlich anders). Als Bonus liefert es auch noch eine für die named.conf-Datei geeignete controls-Klausel, die Sie nur kopieren und einsetzen müssen.

Sie können die Verbindung zum Beispiel mit dem Kommando »rndc status« testen:

```
# rndc status
version: 9.5.1-P3
number of zones: 14
◁◁◁◁◁
server is up and running
```

Seit BIND 9.2 geht es sogar noch einfacher: Wenn Ihre named.conf-Datei *keine* controls-Klausel enthält, lauscht BIND auf der Adresse 127.0.0.1 auf Verbindungen. Den Schlüssel für diese Verbindungen entnimmt er der Datei rndc.key im selben Verzeichnis, wo auch named.conf steht. Wenn es keine rndc.conf-Datei gibt, sucht rndc ebenfalls nach rndc.key.

Damit ist der Luxus natürlich praktisch komplett: »rndc-confgen -a« schreibt die Datei rndc.key, und Sie müssen sich um nichts mehr kümmern.

Fairerweise sollten wir anmerken, dass der »automatische« Mechanismus mit rndc.key zwar den gängigsten Anwendungsfall abdeckt, aber Sie bei irgendwelchen Sonderwünschen wieder auf die herkömmliche Methode zurückgreifen müssen. Sie haben zum Beispiel keine genaue Kontrolle über Parameter wie den Namen und die Länge des Schlüssels und können auch nur mit dem BIND auf demselben Rechner reden. Außerdem werden die Rechte so gesetzt, dass nur root und der Benutzer, mit dessen Identität BIND läuft, auf die Datei zugreifen dürfen. Wenn Sie also als normaler Benutzer rndc aufrufen können wollen, müssen Sie rndc.conf mit (typischerweise) einer eigenen Gruppe und Gruppenausführungsrecht verwenden.

Kommandos Nachdem Sie dafür gesorgt haben, dass Sie mit rndc auf Ihren BIND zugreifen können, sind hier die wichtigsten Kommandos aufgelistet. Sie übergeben sie einfach als Parameter auf der Shell-Kommandozeile:

```
$ rndc reload
```

Manche Kommandos haben Argumente.

reload Liest die Konfigurationsdatei und die Zonendateien neu ein.

reload ⟨*Zonenname*⟩ Liest nur die Zonendatei für ⟨*Zonenname*⟩ neu ein.

reconfig Liest die Konfigurationsdatei neu ein und lädt neue Zonen, aber liest existierende Zonendateien nicht neu ein (selbst falls sie sich geändert haben). Wenn Sie viele Zonen verwalten, ist das schneller, weil keine Zeitstempel auf den Zonendateien geprüft werden müssen.

stats Schreibt Statistik in die Statistikdatei. Typischerweise ist das named.stats im aktuellen Verzeichnis des BIND-Prozesses – versuchen Sie etwas wie

```
# ls -l /proc/`pidof named`/cwd
lrwxrwxrwx 1 bind bind 0 Nov 17 16:05 /proc/2753/cwd▷
 ◁ -> /var/cache/bind
```

oder setzen Sie einen anderen Namen (der Parameter in named.conf heißt statistics-file).

stop Hält den Server an (aber räumt vorher ordentlich auf).

trace Erhöht die »Debuggingstufe« von BIND um 1. Wenn die Debuggingstufe ungleich 0 ist, schreibt BIND interessante Informationen in die Datei `named.run` in seinem aktuellen Verzeichnis (siehe voriger Punkt).

Was BIND genau schreibt, hängt von der Debuggingstufe ab. Es ist auch nicht so, dass Stufe n alles enthält, was Stufe $n - 1$ geschrieben hätte, und noch einiges dazu – die Stufen haben eigentlich nichts miteinander zu tun. (Allerdings führen größere Werte tendenziell zu voluminöserer Ausgabe.) Details über die Stufen bleibt die offizielle Dokumentation uns leider schuldig; in [LA06, Kapitel 13] steht eine Liste.

trace ⟨*Zahl*⟩ Setzt die Debuggingstufe direkt auf ⟨*Zahl*⟩.

notrace Setzt die Debuggingstufe auf Null.

status Gibt Statusinformationen aus.

Sie können auch mehrere Server mit `rndc` steuern, indem Sie den Servernamen mit der `-s`-Option angeben:

```
# rndc -s ns1.example.com reload
```

Damit das funktioniert, müssen Sie in der Datei `rndc.conf` den passenden Schlüssel für den Server definiert haben. Dazu verwenden Sie einen `server`-Eintrag wie

```
server ns1.example.com {
  key "ns1-key";
};
```

mit einem entsprechenden `key`-Eintrag (hier für `ns1-key`).

ndc Das Programm `ndc` ist das Äquivalent zu `rndc` für BIND 8 (ab Version 8.2). Die wesentlichen Unterschiede zwischen `ndc` und `rndc` sind:

- `ndc` unterstützt im Gegensatz zu `rndc` keine Authentisierung.
- `ndc` und BIND kommunizieren standardmäßig nicht über TCP, sondern über ein Unix-Domain-Socket (`/var/run/ndc`). Da `ndc` keine Authentisierung macht, dienen die Zugriffsrechte auf dieses Socket dazu, Fremdlinge vom Herumbasteln am lokalen BIND abzuhalten.

  BIND 9 unterstützt keine Kommunikation über Unix-Domain-Sockets, und laut dem ISC ist das auch nichts, was er noch lernen wird.

  `ndc` und BIND 8 können auch über TCP miteinander reden, aber in Ermangelung eines Authentisierungsmechanismus ist das keine besonders tolle Idee.

- `ndc` unterstützt einen »interaktiven Modus«, in dem Sie nach einer Folge von Kommandos eingeben können. Dazu müssen Sie `ndc` ohne ein »direktes« Kommando aufrufen:

  ```
  # ndc
  Type   help  -or-   /h   if you need help
  ndc> _
  ```

- `ndc` hat ein paar Kommandos, die `rndc` (noch) nicht hat.

In Anbetracht der Tatsache, dass Sie lieber BIND 9 statt BIND 8 benutzen sollten und `ndc` für BIND 9 durch `rndc` ersetzt wurde, ist dieser Abschnitt allerdings von eher historischem Interesse.

Tabelle 6.1: BIND und Signale

Signal	BIND 8	BIND 9	ndc/rndc
HUP	Neu laden	Neu laden	reload
INT	Datenbank-Dump	Serverhalt	dumpdb
ILL	Statistik-Ausgabe	—	stats
USR1	Debugstufe erhöhen	—	trace
USR2	Debugging ausschalten	—	notrace
WINCH	Anfrageprotoll an/aus	—	querylog
TERM	Serverhalt	Serverhalt	stop

Signale Bevor ndc und rndc auf der Bildfläche erschienen, waren Signale die einzige Möglichkeit, mit einem laufenden BIND zu interagieren. Heutzutage ist das nicht mehr *à la mode*, und das, was Sie mit Signalen machen können, wird auch von BIND-Version zu BIND-Version weniger.

Tabelle 6.1 zeigt die möglichen Signale, ihre Wirkung auf BIND 8 und BIND 9 und wie Sie den jeweiligen Effekt auch mit ndc oder rndc erreichen können (heute die empfohlene Methode).

Wenn Sie Ihrem BIND ein Signal schicken wollen, brauchen Sie noch dessen Prozess-ID. Diese speichert er freundlicherweise in einer Datei ab, typischerweise /var/run/named.pid (oder so ähnlich – bei Debian GNU/Linux etwa /var/run/bind/run/named.pid). Damit können Sie dann etwas sagen wie

```
# kill -HUP `cat /var/run/named.pid`
```

oder (für Bash-Anwender)

```
# kill -HUP $(</var/run/named.pid)
```

(Aber das wussten Sie bestimmt schon.)

Kommandos in diesem Kapitel

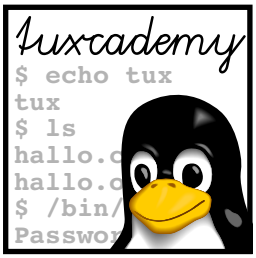
dnssec-keygen Erzeugt Schlüssel für DNSSEC dnssec-keygen(8) 86
named Der Berkeley-Internet-Name-Daemon (BIND) named(8) 81
ndc Steuerprogramm für BIND ndc(8) 85, 88
rndc Programm zur „Fernsteuerung“ von BIND (moderne Version) rndc(8) 85
rndc-confgen Legt Konfigurationsdateien für rndc (und BIND) an rndc-confgen(8) 86

Zusammenfassung

- DNS wurde Mitte der 1980er Jahre standardisiert. Die kanonische Implementierung ist BIND (aber es gibt andere).
- BIND hat seine Konfiguration in /etc/named.conf; die Datei enthält globale Einstellungen und verweist auf »Zonendateien«.
- Die Programme rndc (für BIND 9) und ndc (für BIND 8) dienen zur »Fernsteuerung« von BIND.

Literaturverzeichnis

LA06 Cricket Liu, Paul Albitz. *DNS and BIND*. Sebastopol, CA: O'Reilly Media, Inc., 2006, 5. Auflage. ISBN 978-0-596-10057-5.
http://www.oreilly.com/catalog/dns5/



7

Zonendateien

Inhalt

7.1 Zonendateien und Ressourcendatensätze 92
7.2 SOA-Records 93
7.3 NS-Records 95
7.4 A-Records 95
7.5 CNAME-Records 96
7.6 Rückwärts-Auflösung von Namen und PTR-Records 97
7.7 MX-Records 98
7.8 SRV-Records 99
7.9 Andere Typen von Ressourcendatensätzen 100

Lernziele

- Zonendateien für BIND aufstellen können
- Die wichtigsten Typen von Ressourcendatensätzen verstehen und einsetzen können

Vorkenntnisse

- Kenntnisse über Linux-Systemkonfiguration
- TCP/IP-Kenntnisse
- DNS-Grundkenntnisse (siehe Kapitel 5)
- BIND-Grundkenntnisse (siehe Kapitel 6)

7.1 Zonendateien und Ressourcendatensätze

Während die Konfigurationsdatei von BIND angibt, wie der Server seine Funktion erfüllt, enthalten Zonendateien die tatsächlichen Daten, die der Server zur Verfügung stellt. Ihr Format entstammt den RFCs für DNS ([RFC1034, Abschnitt 3.6.1], [RFC1035, Abschnitt 5]) und wird trotz einiger extrem nervender Eigenschaften auch von vielen anderen DNS-Servern verwendet.

Ressourcendatensätze
RR-Typ

Eine Zonendatei (Bild 7.1) ist eine Folge von Ressourcendatensätzen (engl. *resource records* oder kurz »RRs«). Ressourcendatensätze haben einen Typ, der angibt, welche Art von Daten sie darstellen, und ein davon abhängiges Format (davon später). Das grundlegende Format eines Ressourcendatensatzes sieht etwa so aus:

⟨*Name*⟩ [⟨*TTL*⟩] [⟨*Klasse*⟩] ⟨*Typ*⟩ ⟨*weitere Daten*⟩

Dabei ist der ⟨*Name*⟩ salopp gesagt etwas, wonach Sie den DNS-Server fragen können. DNS-Anfragen beziehen sich immer auf einen Namen und einen bestimmten Typ und liefern (im Erfolgsfall) die ⟨*weiteren Daten*⟩ aus dem betreffenden RR. Für Namen in DNS-Zonendateien gelten einige wichtige Spielregeln:

- Namen *mit* einem Punkt am Ende sind »absolut« und werden ausgehend von der Root-Level-Domain interpretiert. An Namen *ohne* einen Punkt am Ende wird der Name der Zone angehängt, so wie er entweder bei der Einbindung der Zonendatei in die BIND-Konfiguration angegeben wurde oder (mit Vorrang) in einer `$ORIGIN`-Direktive in der Zonendatei selbst steht.

  
  Sie sollten dringend darauf achten, in Zonendateien an den richtigen Stellen einen Punkt oder keinen Punkt anzugeben. Ein Name wie `example.com` in einer Zonendatei für `example.com` wird als `example.com.example.com` angesehen, was selten stimmt.

- Das Zeichen »`@`« ist eine Abkürzung für den Namen der Zone (aus der BIND-Konfigurationsdatei oder `$ORIGIN`).

- Das Namensfeld darf auch ganz frei bleiben, in welchem Fall der Name aus dem vorigen RR (gegebenenfalls auch `@`) weiter benutzt wird.

Die ⟨*TTL*⟩ (kurz für *time to live*) gibt an, wie lange der betreffende RR von Caches zwischengespeichert wird, ohne den ursprünglichen Server zu behelligen. Sie können die TTL bei den einzelnen RRs oder (bequemer) global am Anfang der Zonendatei angeben – dazu dient dann die Direktive `$TTL`.

```
$ORIGIN example.com.          ; Daten über example.com
$TTL 3h                       ; Standard-TTL 3 Stunden
example.com.  IN SOA  ns.example.com. hostmaster.example.com. (
              2009102201      ; Seriennummer
              1d              ; Slave-Refresh (1 Tag)
              6h              ; Slave-Retry (6 Stunden)
              2w              ; Slave-Expiry (2 Wochen)
              1h              ; TTL für nicht vorhandene Namen
              )
@             IN NS    ns.example.com.      ; ein DNS-Server
              IN NS    ns.anderswo.de.      ; noch ein DNS-Server
              IN MX    10 mail.example.com. ; Mailserver
ns       1d   IN A     10.0.0.1
mail          IN A     10.0.0.2
www           IN CNAME mail
```

Bild 7.1: Beispiel für eine Zonendatei


Was ist eine günstige TTL? Grundsätzlich gilt, dass eine lange TTL zwar die Last auf Ihren DNS-Servern verringert, aber auch dazu führt, dass Änderungen länger brauchen, bis sie sich herumsprechen. Bei einer TTL von einem Tag werden Clients nach einer Veränderung der Zonendatei maximal einen Tag lang durch veraltete RRs in die Irre geschickt, die noch auf Caches herumhängen. Umgekehrt bedeutet eine kurze TTL natürlich eine schnellere Reaktion des »Internet« auf Veränderungen Ihrer Zonen, während Ihre DNS-Server mehr zu tun bekommen, weil sie öfter nach aktuellen Daten gefragt werden.


Für A-Records ist eine TTL von einem Tag durchaus vernünftig. Ein Extremfall in Richtung extrem kurzer TTLs sind Dienste wie dyndns.org, die DNS-Namen für Rechner zur Verfügung stellen, deren IP-Adresse oft wechselt (etwa weil sie über einen Wählzugang ins Internet gehen). A-Records von dyndns.org haben standardmäßig eine TTL von einer Minute.


TTLs werden traditionell in Sekunden angegeben. Aktuelle BIND-Versionen erlauben aber eine bequemere Schreibweise mit Einheiten – 5m steht zum Beispiel für »5 Minuten«, 12h für »12 Stunden«, 2d für »2 Tage« und 1w für »1 Woche«. (Für Monate, Jahre, und Jahrhunderte gibt es leider keine Abkürzungen.)


Mit etwas wie

```
$TTL 3h
```

können Sie eine Standard-TTL für alle Records vorgeben, die der $TTL-Zeile folgen. Die Records müssen dann keine explizite TTL mehr enthalten – aber wenn Sie eine angeben, dann gilt diese und nicht der Standardwert.

Die ⟨*Klasse*⟩ gibt an, für welche Sorte Netzwerkprotokoll der RR gilt. Heute ist das in der ganz überwiegenden Mehrheit der Fälle IN (kurz für »Internet«); es gibt ein paar andere Möglichkeiten, die aber kaum praktische Bedeutung haben. IN ist deshalb der Standardfall und kann auch weggelassen werden.

Wie im Beispiel zu sehen ist, können Sie Kommentare in die Zonendatei schreiben, wenn Sie einen Semikolon davorsetzen. Wenn ein RR sehr lang ist (normalerweise bei SOA-Records der Fall), dann können Sie sie mit Hilfe von runden Klammern über mehrere Zeilen in der Datei verteilen.

Übungen


7.1 [!1] Geben Sie eine Zeile an, mit der Sie in einer Zonendatei eine Standard-TTL von einem Tag, drei Stunden und 27 Minuten einstellen können. Wie müßten Sie dies bei einer älteren BIND-Version angeben, die nur TTLs in Sekunden unterstützt

7.2 SOA-Records

Jede Zone im DNS braucht ein SOA-Record (kurz für *start of authority*), das den Haupt-DNS-Server, eine E-Mail-Adresse für Fragen oder Beschwerden und einige Parameter für die Steuerung sekundärer DNS-Server angibt. Im Beispiel

```
example.com.  IN SOA  ns.example.com. hostmaster.example.com. (
              2009102201      ; Seriennummer
              1d              ; Slave-Refresh (1 Tag)
              6h              ; Slave-Retry (6 Stunden)
              2w              ; Slave-Expiry (2 Wochen)
              1h              ; TTL für nicht vorhandene Namen
              )
```

Haupt-DNS-Server

benennt `ns.example.com` den Haupt-DNS-Server für die Zone `example.com`.

Adresse eines Verantwortlichen

`hostmaster.example.com` repräsentiert die Adresse eines Verantwortlichen für die Zone. Möglicherweise sind Sie befremdet, weil diese Zeichenkette gar nicht aussieht wie eine E-Mail-Adresse – aber erinnern Sie sich daran, dass das »@« als Abkürzung für den Zonennamen fungiert. Eine Adresse wie

```
hostmaster@example.com.
```

würde also zu

```
hostmasterexample.comexample.com.
```

und das möchte niemand haben. Statt dessen hat man die haarsträubende Konvention eingeführt, dass der erste Punkt in der Adresse als »@« interpretiert wird.

»Was ist mit Adressen wie `hugo.schulz@example.com`?« werden Sie sich fragen. Hier müssen Sie den ersten Punkt natürlich hinter einem Rückwärts-Schrägstrich verstecken:

```
hugo\.schulz.example.com
```

Wobei Adressen wie `hugo.schulz.example.com` in einer DNS-Konfiguration sowieso nichts zu suchen haben. Schließlich wollen Sie nicht jedesmal, wenn Herr Schulz in Urlaub geht, Ihre Zonen auf die Adresse von seinem Vertreter umbiegen. Sie tun viel besser daran, eine neutrale Adresse wie `hostmaster.schulz.example.com` zu verwenden und eine Umleitung auf der Ebene Ihres Mailservers zu veranlassen – das ist viel einfacher zu warten.

Die Parameter in Klammern beziehen sich auf den Betrieb von sekundären DNS-Servern und werden in Kapitel 8 genauer erklärt. Wichtig ist allenfalls die »Seriennummer«; Sie sollten sich schon jetzt angewöhnen, bei jeder Änderung einer Zonendatei die Seriennummer zu erhöhen.

An der Seriennummer kann ein sekundärer DNS-Server sehen, ob er seine Informationen über die Zone aktualisieren muss. Wenn Sie also die Zonendatei ändern, *ohne* die Seriennummer zu erhöhen, dann bekommen die sekundären DNS-Server Ihre Änderungen nicht mit, und das führt zu Chaos.

Eigentlich ist die Seriennummer nur eine ganze Zahl, und es spricht fundamental nichts dagegen, bei 1 anzufangen und bei Bedarf immer 1 hinzuzuaddieren. Eine gängige Konvention besteht aber darin, die Seriennummer so zu strukturieren, dass die ersten acht Stellen Jahr, Monat und Tag wiedergeben; in Anbetracht der Tatsache, dass die Seriennummer eine 32-Bit-Zahl ist, bedeutet das, dass zumindest für die nächsten 2200 Jahre noch 2 Stellen übrig sind, in denen Sie die Modifikationen eines Tages fortlaufend zählen können. Solange Sie die Zonendatei also nicht öfter als hundertmal am Tag ändern, sind Sie auf der sicheren Seite.

Die Klammern dienen wirklich nur dazu, den RR auf mehrere Zeilen in der Datei verteilen zu können. Sie könnten genausogut etwas schreiben wie

```
example.com. IN SOA ns.example.com. ◁◁◁◁◁ 2w 1h
```

(auf einer Zeile, ohne Klammern).

Übungen

7.2 [!1] Verwenden Sie `dig`, um die `SOA`-Records einiger Ihnen bekannter Domains abzurufen. Welche Konventionen (wenn überhaupt) verwenden diese für die Namen von Haupt-DNS-Servern und die E-Mail-Adressen der Domainverantwortlichen?

7.3 [3] Angenommen, Sie haben die Seriennummer in einer Ihrer Zonendateien versehentlich *zu sehr* erhöht. Was können Sie tun, um sie wieder auf einen vernünftigen Wert zu bringen?

7.3 NS-Records

NS-Records geben an, welche DNS-Server für die betreffende Zone »autoritativ« sind, also direkte (und nicht aus früheren Anfragen zwischengespeicherte) Antworten für Namen in der Zone zu bieten haben. Ob sie diese direkten Antworten aus der offiziellen Zonendatei lesen (was der primäre DNS-Server tut) oder per Zonentransfer holen (was allfällige sekundäre DNS-Server tun), ist irrelevant.

In unserem Beispiel benennen die NS-Records die DNS-Server für die Zone selbst. NS-Records spielen aber auch eine wichtige Rolle sozusagen als »Kleber«, der das DNS zusammenhält, da sie aus in der Hierarchie übergeordneten Zonen auf die DNS-Server untergeordneter Zonen verweisen. Dieser Vorgang heißt Delegation und wird in einem eigenen Kapitel (Kapitel 9) beschrieben.

Delegation

Der im SOA-Record benannte DNS-Server kann, muss aber nicht, in einem NS-Record für die Zone auftauchen. Nur die DNS-Server in den NS-Records werden allerdings mit Anfragen aus dem Internet behelligt. Eine gängige Konfiguration ist die eines »versteckten primären DNS-Servers« (engl. *hidden primary*), bei der Sie Ihre Zonendatei auf einem privaten Server haben, der sie den DNS-Servern Ihres Providers zugänglich macht. Diese beantworten dann Anfragen aus dem Internet, während Ihr Server überhaupt nicht aus dem Internet zugänglich sein muss (jedenfalls nicht für DNS). Aus Sicherheits-, Effizienz- und Wartbarkeitsgründen ist das ein guter Kompromiss, falls Ihr Provider sich darauf einläßt.

7.4 A-Records

Die nominelle Hauptaufgabe des DNS – die Abbildung von Rechnernamen auf IP-Adressen – wird von den A-Records erbracht. Ein A-Record wie

```
server.example.com. IN A 11.12.13.14
```

ordnet dem Rechner mit dem Namen `server.example.com` die Adresse `11.12.13.14` zu.

Ob der betreffende Rechner sich unter der betreffenden Adresse tatsächlich angesprochen fühlt, ist eine völlig andere Frage, nämlich eine der Konfiguration dieses Rechners und nicht eine des DNS. (Auch wenn Sachen wie DHCP mit dynamischer DNS-Aktualisierung da gewisse Querverbindungen hinter den Kulissen herstellen.)

Es ist durchaus erlaubt und auch gängig, demselben Namen mehrere A-Records mit verschiedenen Adressen zuzuordnen. Auf eine Anfrage nach dem betreffenden Namen werden dann alle A-Records zurückgegeben, allerdings (bei BIND, wenn nicht anders konfiguriert) in nach einem Ringelreihen-Verfahren (engl. *round robin*) variierender Reihenfolge. Bei etwas wie

Mehrere A-Records

```
www.example.com. IN A 11.12.13.1
                 IN A 11.12.13.2
                 IN A 11.12.13.3
```

bekommt die erste Anfrage die Antwort

```
11.12.13.1, 11.12.13.2, 11.12.13.3
```

Die zweite dann zum Beispiel

```
11.12.13.2, 11.12.13.3, 11.12.13.1
```

und die dritte

```
11.12.13.3, 11.12.13.1, 11.12.13.2
```

Danach geht es wieder von vorne los. – Da DNS-Clients in der Regel nur die erste Adresse aus der Antwort anschauen, bekommt jeder der drei Web-Server (wenn wir mal davon ausgehen, dass jede der Adressen auf einen anderen Rechner verweist) ein Drittel der Anfragen zugeschanzt. Dies ist natürlich eine extrem krude Form der Lastverteilung, da sie keine Rücksicht auf die Verfügbarkeit und Auslastung der betreffenden Server sowie die Komplexität der Anfragen nimmt, aber manchmal bringen einen auch die einfachen Ansätze weiter.

Übungen

7.4 [2] Unter welchen Bedingungen ist es sinnvoll, eine DNS-basierte »Lastverteilung« mit mehreren `A`-Records für denselben Namen zu verwenden?

7.5 `CNAME`-Records

Mit `CNAME`-Records (kurz für *canonical name*) können Sie einen Namen auf einen anderen abbilden. Im Beispiel

```
mail   IN A     11.12.13.14
www    IN CNAME mail
```

wird festgelegt, dass `www.example.com` ein anderer Name für den Rechner `mail.example.com` ist. – Bei `CNAME`-Records wird der Client im Prinzip dazu veranlasst, eine neue Anfrage nach dem Namen auf der rechten Seite des `CNAME`-Records zu stellen. Freundliche DNS-Server wie BIND liefern das Ergebnis auf diese Anfrage allerdings gleich mit der ursprünglichen Anfrage mit, um dem Client einen unnützen Rundweg über das Internet zu ersparen.

Die Namen, die Sie mit `CNAME`-Records definieren, sind nicht überall erlaubt – insbesondere nicht auf der rechten Seite von anderen RRs. Verkneifen Sie sich also Konstruktionen wie

```
example.com.     IN NS    ns.example.com.
ns.example.com.  IN CNAME www.example.com.
www.example.com. IN A     11.12.13.14
```

Andere Programme – etwa Mailserver – mögen so etwas gar nicht.

`CNAME`-Records, die auf andere `CNAME`-Records verweisen – so wie hier:

```
foo.example.com.  IN CNAME  bar.example.com.
bar.example.com.  IN CNAME  baz.example.com.
```

– sind nicht ausdrücklich verboten und werden auch von BIND unterstützt. Allerdings sind sie auch nicht ausdrücklich erlaubt, so dass andere DNS-Implementierungen sie möglicherweise nicht anbieten. Machen Sie im Interesse Ihres vollen Haupthaars und Ihrer Reputation auf dem Internet einen Bogen um solche Sperenzchen.

```
$ORIGIN 0.0.10.in-addr.arpa.  ; Rückwärtszone für 10.0.0.0/8
$TTL 3h                       ; Standard-TTL 3 Stunden
example.com.  IN SOA  ns.example.com. hostmaster.example.com. (
              2009102201      ; Seriennummer
              1d              ; Slave-Refresh (1 Tag)
              6h              ; Slave-Retry (6 Stunden)
              2w              ; Slave-Expiry (2 Wochen)
              1h              ; TTL für nicht vorhandene Namen
              )
@             IN NS    ns.example.com.      ; ein DNS-Server
              IN NS    ns.anderswo.de.      ; noch ein DNS-Server
1             IN PTR   ns.example.com.
2             IN PTR   mail.example.com.
```

Bild 7.2: Beispiel für eine Rückwärts-Zonendatei

Übungen

7.5 [!2] Welche Vorteile und Nachteile haben CNAME-Records, die auf A-Records verweisen, gegenüber der Idee, mehrere A-Records für dieselbe IP-Adresse einzuführen?

7.6 [2] Angenommen, Ihre komplette *Domain* ändert ihren Namen – etwa von example.com nach beispiel.de. Können Sie das im DNS über ein

```
example.com. IN CNAME beispiel.de.
```

erledigen?

7.6 Rückwärts-Auflösung von Namen und PTR-Records

Das DNS soll nicht nur IP-Adressen zu gegebenen Namen liefern können, sondern auch Namen zu gegebenen IP-Adressen. Dies allein auf der Basis von A-Records zu lösen ist praktisch unmöglich (Sie müßten im Wesentlichen das komplette DNS aufzählen), so dass man sich eines genialen, aber etwas verqueren Tricks bedient, um die DNS-Infrastruktur aus Zonen, Zonendateien, Caching und so weiter auch für diese Anwendung nutzbar zu machen.

Der Trick besteht darin, eine DNS-Anfrage nach einer IP-Adresse wie 11.12.13.14 zu behandeln wie eine DNS-Anfrage nach einem *Namen* wie 14.13.12.11.in-addr.arpa. (die Domain in-addr.arpa. fällt hier vom Himmel). Beachten Sie, dass die Oktette der IP-Adresse hier in umgekehrter Reihenfolge auftauchen! Ähnlich wie bei den »normalen« Namen kann ein DNS-Resolver hier ausgehend von der Root-Level-Domain einen DNS-Server finden, der für eine passende Zone – hier zum Beispiel 13.12.11.in-addr.arpa. – autoritativ ist und eine Antwort auf die Frage nach dem Namen 14.13.12.11.in-addr.arpa. liefern kann.

Für diese »Rückwärts-Auflösung« werden PTR-Records verwendet, auf deren rechter Seite keine IP-Adresse (wie bei A-Records), sondern ein anderer DNS-Name steht. Diese werden parallel zu den A-Records in eigenen Zonendateien gesammelt (da es sich ja um völlig separate Zonen handelt). Diese Zonendateien haben entsprechende SOA- und NS-Records, enthalten aber statt den A- oder CNAME-Records ansonsten nur PTR-Records (Bild 7.2).

Es ist möglich, für dieselbe IP-Adresse mehrere PTR-Records zu haben. Allerdings interessieren sich die wenigsten Programme dafür.

Achten Sie beim Erstellen der Rückwärtszonen besonders pingelig darauf, dass Sie die Namen auf der rechten Seite Ihrer PTR-Records mit einem Punkt abschließen. Die implizite Erweiterung endpunktloser Namen durch den Namen der Zone führt sonst zu Entgleisungen wie

```
mail.example.com.0.0.10.in-addr.arpa.
```

Ein gängiger Kritikpunkt am System der DNS-Zonendateien ist die Notwendigkeit, getrennte Vorwärts- und Rückwärtszonen anlegen und warten zu müssen. Wir geben zu, dass das nicht wirklich genial ist, geben aber zu bedenken, dass dieser Ansatz der einzige ist, der wirklich maximale Flexibilität ermöglicht. Betrachten Sie das Erlernen dieser arkanen Spielregeln als Initiationsritus in den (kleinen) Kreis der Leute, die DNS wirklich verstehen.

Übungen

7.7 [2] Unter welchen Bedingungen könnte es sinnvoll sein, in Rückwärtszonen CNAME-Records zu verwenden? Geht das überhaupt?

7.7 MX-Records

MX-Records dienen dazu, Mail an Adressen wie hugo@example.com zuzustellen. Wenn ein Mailserver die Adresse hugo@example.com verarbeitet, fragt er zunächst nach MX-Records für die Domain example.com. Die Zone könnte zum Beispiel Folgendes definieren:

```
example.com.  IN  MX  10 mail1.example.com.
              IN  MX  10 mail2.example.com.
              IN  MX  20 mail-backup.example.net.
```

MX-Records bestehen aus einem Rechnernamen und einer Priorität. Kleinere Zahlen entsprechen höheren Prioritäten, wobei die tatsächlichen Werte irrelevant sind – nur die Einsortierung zählt. Im Beispiel hier würde der Mailserver zunächst versuchen, die Nachricht an hugo@example.com an einen der beiden Rechner mail1.example.com oder mail2.example.com auszuliefern. Nur wenn *beide* nicht erreichbar sind, würde er es bei mail-backup.example.net versuchen.

MX-Records *müssen* auf Namen zeigen, die A-Records haben – CNAME-Records sind ausgeschlossen. Das heißt, etwas wie

```
example.com.              IN  MX     10 mail.example.com.
mail.example.com.         IN  CNAME  postman-pat.example.com.
postman-pat.example.com.  IN  A      11.12.13.14
```

ist nicht erlaubt.

Wenn Mailserver für eine Mail-Domain kein MX-Record finden, benutzen sie als Notnagel auch gerne A-Records. Allerdings sollten Sie sich darauf nicht verlassen und lieber ein MX-Record installieren.

Mehr über Mail und DNS erfahren Sie zum Beispiel aus der Linup-Front-Schulungsunterlage *Linux als Mailserver*.

Übungen

7.8 [1] Welchen Zweck hat es, mehrere MX-Records mit verschiedenen Prioritäten anzugeben?

7.9 [1] Welchen Zweck hat es, mehrere MX-Records mit derselben Priorität anzugeben?

7.8 SRV-Records

Ein allgemeineres Problem als das, das MX-Records lösen, ist das der Suche nach einem Server für irgendein gegebenes Protokoll innerhalb einer Zone. Der gängige Ansatz dafür, etwa einen WWW- oder FTP-Server für eine bestimmte Firma – nennen wir sie mal example.com – zu finden, verwendet Konventionen für die tatsächlichen DNS-Namen: Den WWW-Server vermuten wir unter www.example.com, den FTP-Server unter ftp.example.com.

CNAME-Records sorgen dafür, dass der Rechner, auf dem der betreffende Dienst läuft, nicht tatsächlich so heißen muss. Wir können immer eine Definition wie

```
www.example.com.  IN  CNAME  server.example.com.
```

verwenden.

Diese Methode funktioniert in der Praxis recht gut, aber auf nette Eigenschaften wie die Prioritätensteuerung der MX-Records müssen wir hier verzichten.

Eine umfassende – wenn auch nicht weit verbreitete – Lösung des Problems bilden SRV-Records gemäß [RFC2782]. Wie MX- und CNAME-Records verweisen SRV-Records auf andere DNS-Namen, für die es dann A-Records geben muss. Der Unterschied besteht einerseits in einer besonderen Struktur für die Namen der SRV-Records und andererseits in einer ausgefeilten Prioritätensteuerung, die noch über die von MX-Records hinausgeht.

Sie könnten zum Beispiel das SRV-Record

```
_ldap._tcp.example.com.  IN  SRV  0 0 389 ldap.example.com.
```

veröffentlichen, um LDAP-Anfragen für die Domain example.com an den Rechner ldap.example.com zu verweisen.

Wenn Sie sich ein bisschen mit LDAP auskennen, dann wissen Sie, dass ein DN-Suffix wie dc=example,dc=com für die Suche nach genau so einem SRV-Record benutzt werden kann.

Die DNS-Namen von SRV-Records folgen der allgemeinen Form

⟨*Dienstname*⟩.⟨*Protokollname*⟩.⟨*Domain*⟩

Die Unterstreichungen dienen dazu, den Dienstnamen und den Protokollnamen von möglicherweise tatsächlich existierenden anderen Namen abzusetzen (nur für den Fall, dass Sie tatsächlich eine Subdomain tcp haben sollten). Dienstnamen kommen aus /etc/services (und damit aus der offiziellen Liste der IANA) oder sind lokal eindeutig festgelegt. Protokollnamen kommen aus /etc/protocols, wobei Sie im wirklichen Leben wahrscheinlich nichts außer _tcp und _udp antreffen werden.

Die vier Parameter rechts vom SRV haben die folgende Bedeutung:

Priorität Die Priorität (eine Zahl von 0 bis 65535) ist zu interpretieren wie die Priorität eines MX-Records: Bei der Suche nach einem Dienst ist mit den SRV-Records anzufangen, deren Priorität die kleinste Zahl ist. Erst wenn alle von diesen SRV-Records benannten Server erfolglos kontaktiert wurden, geht es mit den SRV-Records mit der nächsthöheren Zahl weiter.

Gewicht Innerhalb einer Gruppe von SRV-Records mit derselben Priorität können Sie hiermit steuern, welcher Anteil der Anfragelast auf welchen Server entfällt, nämlich so viel, wie das Gewicht des betreffenden SRV-Records in Relation zur Summe aller Gewichte von SRV-Records derselben Priorität ausmacht. Ein Beispiel: Mit einem Satz von SRV-Records wie

```
_ldap._tcp.example.com.  IN  SRV  0 1 389 ldap1.example.com.
                         IN  SRV  0 2 389 ldap2.example.com.
                         IN  SRV  0 3 389 ldap3.example.com.
```

entfällt auf ldap3.example.com die Hälfte der Anfragen, auf ldap2.example.com ein Drittel und auf ldap1.example.com ein Sechstel.

Verantwortlich dafür, dass das eingehalten wird, ist der Client, der typischerweise eine Zufallszahl erzeugt und schaut, welcher Server sich daraus ergibt. In unserem Beispiel könnte er zum Beispiel eine uniform verteilte Zufallszahl $r \in [0, 1)$ bestimmen und ldap1 ansprechen, wenn $r < 1/6$, ldap2, wenn $1/6 \leqslant r < 1/2$, und ldap3 sonst.

Laut [RFC2782] sollen Sie das Gewicht auf 0 setzen, wenn keine Serverauswahl auf Gewichtsbasis stattfinden soll. Wenn es sowohl SRV-Records mit Gewicht 0 als auch solche mit anderen Gewichten gibt, sollen die mit Gewicht 0 höchstens eine sehr geringe Chance haben, ausgewählt zu werden.

Portnummer Mit der Portnummer können Sie als Systemadministrator Server auf ungewöhnlichen Ports laufen lassen. Für LDAP ist der TCP-Port 389 normal, aber wenn Ihre Clients Ihren LDAP-Server über das SRV-Record finden, spräche nichts dagegen, ihn auf einem anderen anderweitig unbenutzten Port laufen zu lassen (vielleicht 38900).

Servername Der Name des Rechners, der den Dienst tatsächlich anbieten. Wie bei MX-Records muss unter diesem Name (mindestens) ein A-Record zu finden sein; CNAME-Records sind nicht erlaubt.

Um zu dokumentieren, dass ein bestimmter Dienst gar nicht zur Verfügung steht, können Sie einen Punkt (».«) als Servernamen angeben:

```
_gopher._tcp.example.com.  IN  SRV  0 0 0 .
```

SRV-Records setzen voraus, dass die Clients, die auf die betreffenden Dienste zugreifen wollen, mit ihnen umgehen können. Bei den Autoren von Web-Browsern stößt das Konzept bisher auf dröhnendes Desinteresse[1], während es zum Beispiel im Jabber/XMPP-, LDAP-, Kerberos- oder SIP-Umfeld eher populär ist.

7.9 Andere Typen von Ressourcendatensätzen

Es gibt noch etliche andere Typen von RRs, auf die wir hier aus Platz- und Zeitgründen nicht im Detail eingehen können. Ziehen Sie die RFCs oder einschlägige Bücher zu Rate.

[1] Ein entsprechender Bug-Report – mit Patch – ist für Firefox zum Beispiel seit dem 20. September 1999 anhängig; siehe https://bugzilla.mozilla.org/show_bug.cgi?id=14328.

Zusammenfassung

- Informationen über die tatsächlichen DNS-Daten stehen in Zonendateien.
- Zonendateien enthalten Ressourcendatensätze (*resource records*. Diese haben einen Typ, eine Klasse, eine Haltezeit (TTL) und vom Typ abhängige weitere Daten.
- `SOA`-Records enthalten Verwaltungsinformationen für eine Zone.
- `NS`-Records geben autoritative DNS-Server für eine Zone an.
- `A`-Records bilden Namen auf IP-Adressen ab.
- Mit `CNAME`-Records können Sie auf andere Namen verweisen (mit Einschränkungen).
- Die »Rückwärtsauflösung« von IP-Adressen in Namen funktioniert gemäß denselben Prinzipien und mit denselben Mechanismen wie die Vorwärtsauflösung, wobei die Zonennamen aus den IP-Adressen konstruiert werden.
- `PTR`-Records bilden IP-Adressen auf Namen ab.
- `MX`-Records bezeichnen den oder die Mailserver für eine Domain.
- Mit `SRV`-Records können Sie Server benennen, die bestimmte Dienste in einer Domain anbieten. Die Unterstützung dafür ist allerdings noch lückenhaft.
- Es gibt zahlreiche weitere Arten von Ressourcendatensätzen.

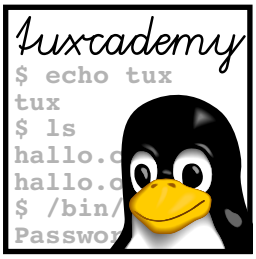
Literaturverzeichnis

RFC1034 P. Mockapetris. »Domain Names – Concepts and Facilities«, November 1987. `http://www.ietf.org/rfc/rfc1034.txt`

RFC1035 P. Mockapetris. »Domain Names – Implementation and Specification«, November 1987. `http://www.ietf.org/rfc/rfc1035.txt`

RFC2672 M. Crawford. »Non-Terminal DNS Name Redirection«, August 1999. `http://www.ietf.org/rfc/rfc2672.txt`

RFC2782 A. Gulbrandsen, P. Vixie, L. Esibov. »A DNS RR for specifying the location of services (DNS SRV)«, Februar 2000. `http://www.ietf.org/rfc/rfc2782.txt`



8

Primäre und sekundäre DNS-Server

Inhalt

8.1 Redundanz im DNS 104
8.2 Konfiguration 104
8.3 Zonentransfer: Grundlagen 108
8.4 Automatische Benachrichtigung bei Änderungen 109
8.5 Zonentransfer absichern mit TSIG 110

Lernziele

- Das Redundanzkonzept von DNS verstehen
- Primäre und sekundäre Server für Zonen definieren können
- Zonentransfers verstehen
- Automatische Benachrichtigung bei Änderungen konfigurieren können
- Zonentransfers über TSIG absichern können

Vorkenntnisse

- Kenntnisse über Linux-Systemkonfiguration
- TCP/IP-Kenntnisse
- DNS-Grundkenntnisse (siehe Kapitel 5)
- BIND-Grundkenntnisse (siehe Kapitel 6)

8.1 Redundanz im DNS

Das DNS ist eine weltweit verteilte Datenbank und bildet gewissermaßen das »Rückgrat« des Internet. Bei einem Ausfall strategisch wichtiger DNS-Server könnte es sein, dass große Teile des Internet nicht mehr zu erreichen wären – oder nur noch über die numerischen IP-Adressen der Stationen, was natürlich auf dasselbe herausläuft (oder wissen Sie spontan die IP-Adresse von `www.google.com`?). Aus diesem Grund war es den Entwicklern von DNS von Anfang an wichtig, ein Redundanzkonzept vorzusehen, damit solche gravierenden Fehlersituationen gar nicht erst auftreten können.

primärer Server

»Autoritative« DNS-Server sind solche, die über offizielle Zonendaten verfügen. Der »primäre« primäre Server für eine Zone ist derjenige, wo die Zonendaten verwaltet werden. Eine Zone hat in der Regel aber einen oder mehrere »sekundäre« Server, die die Zonendaten vom primären Server beziehen und dann ebenfalls als autoritative DNS-Server für die Zone auftreten können. Im Idealfall sind die sekundären Server für eine Zone netztopologisch gesehen »anderswo« untergebracht, so dass bei einer Katastrophe – Meteorit fällt aufs Haus, Bagger trennt das Internet-Kabel durch, … – mindestens einer der autoritativen Server verfügbar bleibt.

sekundäre Server

Früher sprach man statt von »primären« und »sekundären« Servern von »Master-« und »Slave-Servern«. Heute ist das anscheinend nicht mehr politisch korrekt, auch wenn diese Begriffe immer noch, wie wir gleich sehen werden, in der Konfigurationsdatei auftauchen.

obere Grenze

Grundsätzlich gibt es keine obere Grenze für die Anzahl der sekundären DNS-Server, die eine Zone haben kann. Allerdings ist irgendwann ein Punkt erreicht, wo sich ein weiterer Server nicht mehr wirklich lohnt. Selbst Domains mit hohem Anfrageaufkommen, etwa `google.com`, beschränken sich darum auf ein halbes Dutzend DNS-Server (oder so).

Außerdem möchte man, dass DNS-Antworten in 512 Bytes passen (damit sie noch mit UDP übertragen werden können). Das setzt eine Grenze bei 10 oder 11 `NS`-Records.

Mindestanforderungen

Je nach der Top-Level-Domain, in der Sie Ihren Namen registriert haben, kann es aber Mindestanforderungen geben. Das DENIC zum Beispiel besteht darauf, dass es für `de`-Domains mindestens zwei autoritative Server geben muss, die außerdem »komplett unterschiedliche« IPv4-Adressen haben müssen [DEN10]. Gemeinerweise wird das sogar überprüft, bevor Ihr Name offiziell eingetragen wird.

Übungen

8.1 [!2] (Wenn Sie eine Domain haben:) Wie viele DNS-Server gibt es für Ihre Domain? Welcher DNS-Server ist primär? Welche sekundär? Können Sie Aussagen darüber machen, wo diese Server stehen? (Betrachten Sie ersatzweise eine bekannte Domain, zum Beispiel `heise.de` oder `linupfront.de`.)

8.2 [1] Wissen Sie eine Zone mit mehr als 10 sekundären DNS-Servern? Wenn ja, welche?

8.2 Konfiguration

Wie Sie einen primären DNS-Server für eine Zone definieren können, haben Sie bereits in Kapitel 6 gesehen: Eine Deklaration der Form

```
zone "example.com" {
    type master;                                    Primärer Server
```

```
    file "/etc/bind/db/example.com";
};
```

in der BIND-Konfiguration tut alles Nötige. Für einen sekundären Server sieht die Deklaration gar nicht viel anders aus, nämlich zum Beispiel so:

```
zone "example.com" {
    type slave;                                   Sekundärer Server
    file "bak/example.com";                       Sicherheitskopie
    masters { 11.22.33.44; };                     Adresse des primären Servers
};
```

Statt »master« müssen Sie »slave« einsetzen. Außerdem müssen Sie in masters die IP-Adresse des primären Servers angeben. Während der primäre Server die Zonendaten aus der mit file angegebenen Datei liest, verwendet der sekundäre Server diese Datei dazu, eine Sicherheitskopie der vom primären Server erhaltenen Daten anzulegen.

Nötig wäre das nicht, aber es ist nützlich, falls der sekundäre DNS-Server startet und den primären aus irgendwelchen Gründen nicht erreichen kann. In diesem Fall kann er nämlich die Sicherheitskopie einlesen und Daten verteilen (auch wenn diese möglicherweise ein bisschen veraltet sind). Eine eventuell nötige Aktualisierung holt der sekundäre Server dann automatisch nach, sobald der primäre Server wieder zur Verfügung steht.

Grundsätzlich könnten Sie in der Konfiguration des sekundären Servers auf die file-Klausel verzichten – er würde auch ohne funktionieren –, aber wir raten Ihnen sehr dringend davon ab.

Wie Sie die Datei *nennen*, ist einzig und allein Ihnen überlassen. Wir benutzen in dieser Unterlage Namen der Form bak/* für Sicherheitskopien von Zonendaten auf sekundären Servern, wobei bak ein Unterverzeichnis des Arbeitsverzeichnisses von BIND – typischerweise /var/cache/bind – ist. Sie können das natürlich anders machen. In Ihrem eigenen Interesse sollten Sie sich aber ein Schema überlegen und dann so weit wie möglich dabei bleiben.

Außerdem müssen Sie noch dafür sorgen, dass die Zone selbst die richtigen Informationen enthält. Zunächst sollten Sie im SOA-Record geeignete Einstellungen vornehmen:

SOA-Record

```
example.com.  IN SOA  ns.example.com. hostmaster.example.com. (
              2009102201      ; Seriennummer
              1d              ; Slave-Refresh (1 Tag)
              6h              ; Slave-Retry (6 Stunden)
              2w              ; Slave-Expiry (2 Wochen)
              1h              ; TTL für nicht vorhandene Namen
              )
```

Wichtig ist hier vor allem der Teil zwischen den Klammern (der Rest wurde ja bereits besprochen), namentlich die Steuerparameter für sekundäre DNS-Server. Der Reihe nach sind das:

Seriennummer Die Seriennummer muss jedes Mal erhöht werden, wenn sich an den Zoneninformationen etwas ändert. Die sekundären DNS-Server rufen in periodischen Abständen das SOA-Record ab und prüfen, ob die Seriennummer höher ist als die letzte, die sie gesehen haben; falls ja, wird ein Zonentransfer ausgelöst. Siehe hierzu auch Abschnitt 7.2.

Refresh-Zeit Gibt den Abstand an, in dem der sekundäre DNS-Server prüft, ob seine Daten noch aktuell sind. Die hier angegebene Zeit von 1 Tag ist relativ hoch; eine sinnvolle Untergrenze wäre zum Beispiel 3 Stunden, und Werte zwischen 6 und 12 Stunden sind realistisch.

Parameter	Minimum	Maximum
Refresh-Zeit	1h	24h
Retry-Zeit[a]	15m	6h
Verfallszeit	1w	1000h
negative TTL	3m	1d

Tabelle 8.1: SOA-Parameter gemäß DENIC (Quelle: [DEN10])

Seit der Version 8 erlaubt BIND die komfortable Angabe dieser und der anderen Zeitangaben mit Einheiten; sollten Sie es noch mit steinalten Versionen von BIND zu tun haben, müssen Sie die Haltezeiten in Sekunden (ohne Einheit) angeben (ein Tag hat 86400 Sekunden, eine Woche 604800). Wenn Sie sadomasochistisch veranlagt sind, können Sie das natürlich auch bei BIND 9 noch tun …

Heutzutage ist es nicht mehr nötig, die Refresh-Zeit abzuwarten, da der primäre Server die sekundären Server proaktiv benachrichtigen kann, wenn es Änderungen gibt. Siehe hierzu Abschnitt 8.4.

Retry-Zeit Wenn der sekundäre DNS-Server den primären DNS-Server bei einer Routineprüfung nicht erreichen kann, dann versucht er es in dem Zeitabstand wieder, den dieser Parameter angibt. Üblicherweise ist dieser Zeitabstand kürzer als die Refresh-Zeit, aber das muss nicht unbedingt so sein. Auch hier ist »6 Stunden« ein eher hoch angesetzter Wert.

Verfallszeit (engl. *expiry time*) Wenn der sekundäre DNS-Server so lange keinen Kontakt zum primären Server hatte, wie dieser Zeitraum angibt, dann hört er auf, Anfragen für die Zone zu beantworten. Eine Woche ist hier ein vernünftiger Wert; längere Zeiträume sind möglich, wenn die Zone sich nicht oft ändert. In jedem Fall sollte die Verfallszeit nicht kürzer sein als die Refresh-Zeit.

TTL für nicht vorhandene Namen Wie alle RRs haben auch Antworten der Form »Diesen Namen gibt es gar nicht« eine Haltezeit, die angibt, wie lange der Empfänger sie in seinem Cache liegen lassen soll. Diese Haltezeit wird hier angegeben.

Früher – bis BIND 8.2 – galt dieser Parameter nicht nur für die negative Haltezeit, sondern auch für die Standard-Haltezeit für RRs ohne genaue Angabe. Inzwischen gibt es dafür die $TTL-Direktive.

Das Maximum für die negative Haltezeit beträgt 3 Stunden.

DENIC

Sollten Sie es mit dem DENIC zu tun bekommen, werden Sie herausfinden, dass die Herrschaften dort relativ konkrete Vorstellungen darüber haben, welche Einstellungen für diese Parameter akzeptabel sind (Tabelle 8.1).

NS-Records

Schließlich müssen Sie sicherstellen, dass für alle autoritativen DNS-Server NS-Records in der Zone vorhanden sind. Die NS-Records dienen dazu, dass anfragende Server wissen, wo sie Informationen über die Zone herbekommen können.

hidden primary

Es spricht nichts dagegen, kein NS-Record für den *primären* DNS-Server zu haben, jedenfalls solange zwei unabhängige sekundäre DNS-Server für die Zone vorhanden und in NS-Records erwähnt sind. In diesem Fall muss der primäre DNS-Server nicht aus dem Internet zugänglich sein, was grundsätzlich einen Sicherheitsgewinn bedeutet. Man spricht von einer *hidden-primary*-Konfiguration.

Eng verwandt mit der *hidden-primary*-Konfiguration ist die Idee eines *stealth slave server*. Dies ist ein sekundärer Server, für den es kein NS-Record gibt. Auf den ersten Blick klingt das ziemlich nutzlos – aber es kann Situationen geben, in denen Sie zum Beispiel in einem bestimmten Subnetz aus Effizienzgründen einen autoritativen DNS-Server haben wollen, ohne dass dieser Antworten aus dem Internet beantwortet (oder beantworten kann). Wenn Sie für diesen Server kein NS-Record installieren, dann wird er von entfernten DNS-Servern nicht angesprochen, aber Sie können die clientseitigen Resolver so einstellen, dass sie direkt auf ihn verweisen.

stealth slave server

Zum Schluss sollten wir erwähnen, dass die Begriffe »primär« und »sekundär« streng genommen nicht für Server gelten, sondern für *Zonen*. Derselbe DNS-Server kann gleichzeitig primärer Server für Zone *X* und sekundärer Server für Zone *Y* sein. Entscheidend ist die Typangabe in der Definition der jeweiligen Zone.

Übungen

8.3 [2] Konfigurieren Sie einen sekundären DNS-Server für eine Zone auf Ihrem primären DNS-Server und die dazugehörige Rückwärts-Zone. (Verwenden Sie dazu einen zweiten Rechner, eine (zweite) virtuelle Maschine oder – etwas mühseliger – einen zweiten BIND auf Ihrem Rechner. In einem Präsenztraining: Verabreden Sie sich mit Ihrem Nachbarn und installieren Sie die sekundären Server »über Kreuz«.) Stellen Sie sicher, dass Ihr primärer und Ihr sekundärer Server dieselben Antworten geben, indem Sie mit `dig` oder `host` gezielt die betreffenden Server abfragen.

8.4 [2] Überzeugen Sie sich, dass Aktualisierungen korrekt an den sekundären Server übertragen werden. Ändern Sie dazu Ihre Zone auf dem primären Server (etwa indem Sie ein weiteres A-Record hinzufügen) und laden Sie (nur!) dessen Konfiguration neu. Prüfen Sie mit `dig` oder `host`, dass der primäre Server die neuen Daten liefert und der sekundäre Server die alten (dort sollte der betreffende Name nicht definiert sein). Starten Sie dann den sekundären DNS-Server neu (oder lösen Sie mit `rndc` einen expliziten Zonentransfer aus) und prüfen Sie, ob die neuen Daten danach auch auf dem sekundären Server zur Verfügung stehen. (*Wichtig:* Diese und die folgende Aufgabe funktionieren nur dann wirklich, wenn Sie in die Konfiguration des primären Servers ein

```
options { notify no; };
```

aufnehmen. Die Details dazu stehen in Abschnitt 8.4.)

8.5 [2] Stellen Sie die Parameter im SOA-Record Ihrer Zone so ein, dass der sekundäre DNS-Server in kurzen Abständen (etwa einmal pro Minute) eine Aktualitätsprüfung vornimmt. Ändern Sie die Zone auf dem primären Server und laden Sie diesen neu. Überzeugen Sie sich, dass der sekundäre Server eine Weile später über die neuen Daten verfügt.

8.6 [3] Stellen Sie die Refresh-, Retry- und Verfallszeiten im SOA-Record Ihrer Zone respektive auf die Werte »60 Sekunden«, »10 Sekunden« und »120 Sekunden«. Laden Sie die Zone auf dem primären Server neu und lösen Sie einen Zonentransfer aus. Halten Sie anschließend den primären DNS-Server an und beobachten Sie mit einem Programm wie `tcpdump` oder `wireshark`, ob und ggf. wie der sekundäre DNS-Server versucht, mit dem primären Server Kontakt aufzunehmen. Überzeugen Sie sich, dass der sekundäre Server nach Ablauf der Verfallszeit keine Anfragen nach Daten in der Zone mehr beantwortet.

8.3 Zonentransfer: Grundlagen

Wenn ein sekundärer DNS-Server feststellt, dass sein Datenbestand veraltet ist (Stichwort: Seriennummern-Vergleich in den SOA-Records), dann versucht er, einen »Zonentransfer« auszulösen und sich auf diesem Weg aktuelle Daten zu besorgen. Je nachdem, wie umfangreich die Zonendaten sind, kann das eine Weile dauern und den primären Server merklich belasten.

Sicherheitskritisch sind Zonentransfers sowieso; die Namensstruktur einer Installation läßt oft Schlüsse auf ihre Netztopologie und möglicherweise sogar die verwendeten Betriebssysteme oder angebotenen Serverdienste zu.

Wenn Sie wissen möchten, welche Daten so alles übertragen werden, können Sie einen Zonentransfer testhalber zum Beispiel mit `dig` auslösen:

```
$ dig @ns.example.com example.com. axfr
```

Zonentransfer auslösen

Um einen sekundären DNS-Server dazu zu bringen, einen Zonentransfer durchzuführen, der dann tatsächlich bei Bedarf die betreffende Zone aktualisiert, können Sie `rndc` (oder `ndc`) verwenden:

```
# rndc refresh example.com                                    BIND 9
# ndc reload example.com                                      BIND 8
```

Zonentransfer erzwingen

Die Betonung liegt dabei auf »bei Bedarf« – die Seriennummer der Zone auf dem primären Server muss größer sein als die auf dem sekundären. Um einen Zonentransfer gewaltsam zu erzwingen, müssen Sie auf dem sekundären Server die Sicherheitskopie der Zonendatei löschen und den BIND dort neu starten (neu laden reicht nicht aus).

Auf einem Server, der für viele Zonen autoritativ ist, kann ein »`rndc refresh`« ziemlich lange dauern, da alle Zonendateien geprüft werden.

Zonentransfers werden übrigens über TCP abgewickelt – die zu übertragenden Datenmengen sind in der Regel zu groß für UDP.

Einschränkung

Weil Sie nicht jeder beliebigen Station auf dem Internet erlauben wollen, Ihre Zonendaten *en gros* abzurufen, sollten Sie Zonentransfers zumindest auf diejenigen Rechner beschränken, die als sekundäre DNS-Server für Ihre Zonen fungieren. Dazu können Sie in der Definition der Zone auf dem primären Server eine `allow-transfer`-Klausel verwenden:

```
zone "example.com" {
    type master;                                          Primärer Server
    file "/etc/bind/db/example.com";
    allow-transfer { 11.22.33.55; };                      Sekundäre(r) Server
};
```

Der Parameter von `allow-transfer` ist eine Adressensuchliste. Adressensuchlisten besprechen wir im Detail in Abschnitt 10.1.

Übungen

8.7 [!1] Verwenden Sie `dig`, um an Ihrem primären DNS-Server einen Zonentransfer auszulösen. Können Sie auch `host` für einen Zonentransfer verwenden?

8.8 [2] Benutzen Sie `allow-transfer`, um Ihrem sekundären Server Zonentransfers zu verbieten. Überzeugen Sie sich (mit `dig`, `host` oder auch BIND selbst), dass sie tatsächlich nicht mehr akzeptiert werden.

8.4 Automatische Benachrichtigung bei Änderungen

Der primäre Server für eine Zone kann die sekundären Server bei Änderungen auch direkt benachrichtigen, so dass Sie nicht die Refresh-Zeit abwarten müssen. Der entsprechende Mechanismus heißt DNS NOTIFY und ist in [RFC1996] standardisiert; BIND 8 und BIND 9 unterstützen ihn. DNS NOTIFY

Mit DNS NOTIFY verschickt der primäre Server, wenn die Seriennummer einer Zone sich geändert hat, eine Benachrichtigung an alle in `NS`-Records für die Zone benannten sekundären Server. Diese Benachrichtigung enthält das `SOA`-Record und damit die aktuelle Seriennummer der Zone. Ein sekundärer Server bestätigt diese Benachrichtigung (damit der primäre Server ihm keine weiteren schicken muss) und tut anschließend so, als ob die Refresh-Zeit abgelaufen wäre, d. h., er holt das `SOA`-Record der Zone vom primären Server, vergleicht die Seriennummern und löst, falls nötig, einen Zonentransfer aus.

Nach einem erfolgreichen Zonentransfer verschickt der sekundäre Server selbst auch nochmal eine Runde NOTIFYs an die anderen autoritativen DNS-Server der Zone. Der Sinn dahinter ist, dass der primäre Server möglicherweise nicht alle sekundären Server direkt erreichen kann (eventuell verwenden sie einen anderen sekundären Server als primären Server). Das ist allerdings erst ab BIND 8.2.3 implementiert.

DNS NOTIFY ist seit BIND 8 standardmäßig eingeschaltet. Sollten Sie keine Benachrichtigungen wünschen, können Sie global die Option »`notify no`« setzen: keine Benachrichtigungen

```
options {
    notify no;
};
```

Außerdem können Sie Benachrichtigungen auch für einzelne Zonen abschalten, indem Sie die `notify`-Option in der Zonendefinition unterbringen:

```
zone "example.com" {
    type master;
    file "/etc/bind/db/example.com";
    notify no;
};
```

Wenn Sie für eine Zone einen *stealth slave server* definiert haben (siehe Abschnitt 8.2), für den es kein `NS`-Record gibt, dann bekommt dieser auch keine NOTIFY-Nachrichten. Um ihn trotzdem zu versorgen, können Sie in der Zonendefinition (oder auch in einer globalen `options`-Deklaration) eine `also-notify`-Klausel unterbringen: `also-notify`

```
also-notify { 11.22.33.44; };
```

Der (oder die) benannte(n) Server bekommen dann ebenfalls NOTIFY-Nachrichten geschickt.

Wenn Sie zum `also-notify` noch ein »`notify explicit`« angeben, werden *nur* die im `also-notify` genannten Server benachrichtigt. `notify explicit`

Übungen

8.9 [!2] Wenn die NOTIFY-Nachricht des primären Servers schon das neue `SOA`-Record enthält, warum holt der sekundäre Server es dann noch einmal und macht eine weitere Prüfung, bevor er tatsächlich einen Zonentransfer auslöst?

8.10 [!2] Bearbeiten Sie noch einmal Übung 8.4, aber entfernen Sie vorher das in der Aufgabenstellung erwähnte »`notify no`«. Überzeugen Sie sich, dass der primäre Server die NOTIFY-Benachrichtigung verschickt (`wireshark`!), der sekundäre Server sie bestätigt und dass ein Zonentransfer stattfindet, nach dem der sekundäre Server dann über die aktuellen Daten verfügt.

8.5 Zonentransfer absichern mit TSIG

Mit `allow-transfer` können Sie zwar IP-Adressen von Rechnern benennen, die Zonentransfers durchführen dürfen, aber eine Authentisierung über IP-Adressen ist bekanntlich nicht das Gelbe vom Ei: Sie sind viel zu leicht zu fälschen. Seit Version 8.2 unterstützt BIND TSIG (kurz für *transaction signatures*), einen Mechanismus, bei dem entfernte DNS-Server, die einen Zonentransfer durchführen wollen, sich über einen kryptografischen Schlüssel identifizieren müssen. Nur solche Server, die den korrekten Schlüssel vorweisen können, bekommen die Zonendaten ausgehändigt.

TSIG [RFC2845] ist nicht nur für die Absicherung von Zonentransfers zu gebrauchen, sondern auch für die von dynamischen Aktualisierungen. Dies wird hier allerdings nicht weiter ausgeführt.

TSIG ist nicht zu verwechseln mit dem wesentlich komplizierteren DNSSEC, das nicht nur Transaktionen zwischen Servern absichert, die sich »kennen«, sondern es erlaubt, die Authentizität von beliebigen DNS-Daten sicherzustellen. DNSSEC wird hier nicht weiter besprochen.

Bei TSIG fügt der Sender einer DNS-Anfrage (oder -Antwort) der Nachricht ein `TSIG`-Record hinzu, das eine »Signatur« der Nachricht enthält. (Genaugenommen wird die MD5-Prüfsumme über den Rest der Nachricht und den Schlüssel bestimmt.) Dadurch kann der Empfänger nachvollziehen, dass der Absender den richtigen Schlüssel verwendet hat und dass die Nachricht nicht unterwegs verfälscht wurde.

Schlüssel erzeugen Damit TSIG zur Absicherung eines Zonentransfers funktioniert, müssen der primäre und der sekundäre DNS-Server über denselben Schlüssel verfügen. Einen solchen Schlüssel erzeugen Sie am bequemsten mit dem Kommando `dnssec-keygen`:

```
# dnssec-keygen -a HMAC-MD5 -b 128 -n HOST ns0-ns1.example.com.
Kns0-ns1.example.com.+157+21300
```

Die Optionen `-a` und `-b` dienen hierbei zur Auswahl des Algorithmus, für den der Schlüssel gedacht ist, und der Schlüssellänge. Beide Werte liegen für TSIG in der Praxis fest; Sie müssen sie trotzdem angeben, da `dnssec-keygen` (wie der Name andeutet) auch Schlüssel für andere Zwecke aus dem Dunstkreis von DNSSEC erzeugen kann. Die `-n`-Option schließlich gibt an, dass wir einen Schlüssel für TSIG erzeugen möchten.

Der letzte Parameter ist der Name des Schlüssels. Er wird für die Schlüsseldefinition in der BIND-Konfiguration gebraucht und muss ebenfalls beim primären und beim sekundären Server übereinstimmen.

[RFC2845] empfiehlt, für jede Kombination aus primärem und sekundärem Server einen eigenen Schlüssel zu definieren und die Namen der betreffenden Rechner in den Schlüsselnamen zu integrieren. Deshalb `ns0-ns1.example.com` – die beteiligten Rechner in unserem Beispiel heißen `ns0` und `ns1` und es geht um die Domain `example.com`.

`dnssec-keygen` hat zwei Dateien angelegt, deren Namen sich von der Zeile ableiten, die das Programm ausgegeben hat:

```
# cat Kns0-ns1.example.com.+157+21300.key
ns0-ns1.example.com. IN KEY 512 3 157 QB/Dhg2V+Iqc+poGbi+eoA==
# cat Kns0-ns1.example.com.+157+21300.private
Private-key-format: v1.3
Algorithm: 157 (HMAC_MD5)
Key: QB/Dhg2V+Iqc+poGbi+eoA==
Bits: AAA=
Created: 20110429142655
Publish: 20110429142655
Activate: 20110429142655
```

Der eigentliche Schlüssel ist dabei »QB/Dhg2V+Iqc+poGbi+eoA==«.

Es hält Sie niemand davon ab, Ihren eigenen Schlüssel festzulegen und zum Beispiel mit `mmencode` zu base64-codieren. Auf der anderen Seite geht `dnssec-keygen` sorgfältig mit der Kryptografie um und es wäre strategisch unklug, sich das nicht zunutze zu machen.

Den Schlüssel können Sie jetzt in Ihre `named.conf`-Datei(en) einbauen, etwa so:

```
key ns0-ns1.example.com. {
    algorithm hmac-md5;
    secret "QB/Dhg2V+Iqc+poGbi+eoA==";                    Siehe oben
};
```

(Achten Sie darauf, dass Sie den Schlüssel gegebenenfalls so übertragen, dass niemand mithören kann – etwa über `ssh` oder `scp`.)

Eine gängige Fehlerquelle bei TSIG ist, dass die Uhren der beteiligten Rechner zu sehr voneinander abweichen. Ein Unterschied von 5 Minuten ist noch tolerabel, mehr führt zu Problemen. Verwenden Sie sinnvollerweise NTP, um auf allen Rechnern eine gemeinsame Zeit zu etablieren.

Konfiguration auf dem sekundären Server Damit der sekundäre Server den Schlüssel benutzt, müssen Sie ihn in einer `masters`-Klausel in der Definition der betreffenden Zone angeben. Etwas wie

`masters`

```
zone "example.com" {
    type slave;
    masters { 10.0.0.1 key ns0-ns1.example.com.; };
    file "bak.example.com";
};
```

sorgt dafür, dass der sekundäre BIND alle Anfragen an den primären Server für die Zone mit dem benannten Schlüssel signiert.

Sie könnten auch eine `server`-Klausel angeben und damit bewirken, dass *alle* Anfragen an den primären Server signiert werden. Das erlaubt Ihnen aber nicht, für verschiedene Zonen verschiedene Schlüssel zu verwenden:

`server`

```
server 10.0.0.1 {
    keys { ns0-ns1.example.com.; };
};
```

Konfiguration auf dem primären Server Auf dem primären Server können Sie
allow-transfer in der allow-transfer-Klausel in der Zonendefinition angeben, dass nur Anfragen bearbeitet werden, die mit dem richtigen Schlüssel signiert sind:

```
zone "example.com" {
    type master;
    file "/etc/bind/db/example.com";
    allow-transfer { key ns0-ns1.example.com.; };
};
```

Sie können auch noch trickreicher werden, etwa wenn Sie dafür sorgen wollen, dass die Zonentransfer-Anfragen von einem Rechner mit einer bestimmten IP-Adresse *und* dem korrekten Schlüssel kommen sollen. Allerdings geht das von hinten durch die Brust ins Auge:

```
acl not-ns1.example.com { !10.0.0.2; any; };

zone "example.com" {
    type master;
    file "/etc/bind/db/example.com";
    allow-transfer {
        !not-ns1.example.com;          Anfragen von falscher Adresse abweisen
        key ns0-ns1.example.com.;      Richtigen Schlüssel verlangen
    };
};
```

Die ACL-Definition passt auf alle Rechner, die *nicht* die IP-Adresse von ns1.example.com haben (Schauen Sie in Kapitel 10 nach, wie ACLs funktionieren). Wenn die allow-transfer-Klausel ausgewertet wird, wird zunächst geprüft, ob die Absenderadresse der Anfrage auf die ACL not-ns1.example.com passt. Ist das der Fall, wird die Anfrage abgelehnt (dafür steht das »!«). Kommt die Anfrage dagegen vom Rechner mit der Adresse 10.0.0.2, wird der Schlüssel geprüft.

Die allow-transfer-Klausel kann auch in der options-Deklaration stehen statt in einer Zonendefinition und gilt dann für alle Zonen. Allerdings können Sie dann keinen zonenspezifischen Schlüssel mehr verwenden.

Testen Sicherlich möchten Sie sich überzeugen, dass diese Konfiguration auch das tut, was sie soll. Versuchen Sie zuerst einen Zonentransfer ohne TSIG mit dig:

```
$ dig @10.0.0.1 example.com. axfr

; <<>> DiG 9.7.3 <<>> @10.0.0.1 example.com. axfr
; (1 server found)
;; global options: +cmd
; Transfer failed.
```

dig und TSIG Um mit dig einen per TSIG authentisierten Zonentransfer auszulösen, können Sie den Schlüssel mit der Option -y direkt auf der Kommandozeile angeben:

```
$ dig -y ns0-ns1.example.com.:QB/Dhg2V+Iqc+poGbi+eoA==▷
 ◁ @10.0.0.1 example.com. axfr
```

Besser ist es allerdings, wenn der Schlüssel in einer Datei steht. Dafür können Sie die Ausgabe von dnssec-keygen verwenden:

```
$ dig -k Kns0-ns1.example.com.+157+21300.key▷
 ◁ @10.0.0.1 example.com. axfr
```


Wenn der Schlüssel auf der Kommandozeile übergeben wird, könnte es sein, dass er in der Ausgabe von `ps` oder in `/proc` zu sehen ist. Das ist eine mögliche Sicherheitslücke.

Übungen


8.11 [!2] Erzeugen Sie wie oben beschrieben einen TSIG-Schlüssel und installieren Sie ihn auf Ihrem primären und sekundären DNS-Server. Vergewissern Sie sich, dass Zonentransfers mit dem Schlüssel funktionieren und ohne nicht.

Kommandos in diesem Kapitel

`dnssec-keygen` Erzeugt Schlüssel für DNSSEC `dnssec-keygen`(8) 110

Zusammenfassung

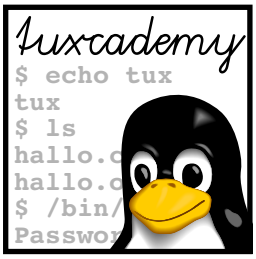
- DNS erlaubt Redundanz durch primäre und sekundäre Server für Zonen.
- Auf einem primären Server werden Zonendaten gewartet. Sekundäre Server holen sich die Zonendaten von einem primären Server.
- Sekundäre Server sind einfach zu konfigurieren.
- Der Parameter `allow-transfer` erlaubt die Beschränkung von Zonentransfers auf Rechner mit bestimmten IP-Adressen.
- DNS NOTIFY erlaubt bei Änderungen an Zonendaten die automatische Benachrichtigung der sekundären Server durch den primären Server.
- Mit TSIG können Sie Zonentransfers über Kryptografie authentisieren.

Literaturverzeichnis

DEN10 DENIC e. G. »Dokumentation Nameserver Predelegation Check«. Abrufbar über *Weiterführende Links und Dokumente* auf `http://www.denic.de/hintergrund/nast.html`; das DENIC macht eine clevere Authentisierung (!?), die kein direktes Link zuläßt, September 2010. Version 0.18.

RFC1996 P. Vixie. »A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)«, August 1996. `http://www.ietf.org/rfc/rfc1996.txt`

RFC2845 P. Vixie, O. Gudmundsson, D. Eastlake 3rd, et al. »Secret Key Transaction Authentication for DNS (TSIG)«, Mai 2000. `http://www.ietf.org/rfc/rfc2845.txt`



9

Subdomains und Delegation

Inhalt

9.1 Einfache Subdomains 116
9.2 Delegation an andere DNS-Server 117
9.3 Delegation für Rückwärts-Zonen 119
9.4 Stub-Zonen 121

Lernziele

- Subdomains innerhalb einer Zone definieren können
- Subdomains an andere Server delegieren können
- Delegation für Rückwärtsauflösung verstehen
- Stub-Zonen einsetzen können

Vorkenntnisse

- Kenntnisse über Linux-Systemkonfiguration
- TCP/IP-Kenntnisse
- DNS-Grundkenntnisse (siehe Kapitel 5)
- BIND-Grundkenntnisse (siehe Kapitel 6)

9.1 Einfache Subdomains

Wenn Ihr Netz eine gewisse Größe erreicht, kann es sich anbieten, den Namensraum zu untergliedern – einerseits um Namenskollisionen zu vermeiden (wenn es mehrere Web-Server gibt, die alle `www` heißen wollen) und andererseits, weil es oft sinnvoll ist, die Namensstruktur an die Organisationsstruktur anzupassen. Wenn in Ihrem Intranet zum Beispiel die Entwicklungsabteilung, die Marketingabteilung und die Personalabteilung eigene Webseiten unterhalten, würde es sich anbieten, die Namen

```
www.entwicklung.example.com
www.marketing.example.com
www.personal.example.com
```

zu verwenden – sogar, wenn diese Webseiten in Wirklichkeit alle auf demselben Web-Server liegen. Dieses Arrangement macht es nämlich einfach, später eine oder mehrere dieser Webpräsenzen auf eigene Server zu verlagern, was mit Adressen der Form

```
http://www.example.com/entwicklung/
```

umständlicher wäre.

Auf der DNS-Seite ist dies sehr einfach zu erreichen: Die Einträge

```
; Zonendatei für example.com
<<<<<
www.entwicklung  IN A  192.168.1.1
www.marketing    IN A  192.168.1.1
www.personal     IN A  192.168.1.1
<<<<<
```

realisieren das Gewünschte.

Ein leistungsfähiger Web-Server wie Apache hat kein Problem damit, die drei unabhängigen Web-Präsenzen anzubieten; wie das im Detail funktioniert, können Sie zum Beispiel der Linup-Front-Unterlage *Der Web-Server Apache* entnehmen.

Subdomain in derselben Zone

Solange eine Subdomain nicht zu viele Namen enthält und Sie als Betreiber der übergeordneten Domain auch für die Verwaltung der Subdomain zuständig sind, ist es am einfachsten, die betreffenden Namen einfach komplett in der Zonendatei der übergeordneten Domain unterzubringen:

```
; Zonendatei für example.com
<<<<<
sky.blue      IN A     192.168.2.1
navy.blue     IN A     192.168.2.2
steel.blue    IN A     192.168.2.3
cadet.blue    IN A     192.168.2.4
www.blue      IN CNAME steel.blue
<<<<<
```

Da diese Namen alle nicht mit einem Punkt aufhören, wird implizit der Name der Zone angehängt. Wir reden also in Wirklichkeit über `sky.blue.example.com` & Co.

`$ORIGIN`

Mit der `$ORIGIN`-Direktive können Sie sich etwas Tipparbeit ersparen. Das Folgende entspricht dem vorigen Beispiel:

```
; Zonendatei für example.com
<<<<<
```

```
$ORIGIN blue.example.com.
sky           IN A     192.168.2.1
navy          IN A     192.168.2.2
steel         IN A     192.168.2.3
cadet         IN A     192.168.2.4
www           IN CNAME steel
◁◁◁◁◁
```

(Beachten Sie den Punkt am Ende des Arguments von `$ORIGIN`.)

`$ORIGIN` legt fest, womit Namen erweitert werden, die nicht mit einem Punkt aufhören, aber ändert nicht die Bedeutung von `@`.

Wenn die Subdomains größer werden, räumlich anderswo liegen oder gar jemand anders die Verantwortung für ihre Wartung und ihren Betrieb hat, dann ist dieser einfache Ansatz nicht mehr tragfähig genug. Im nächsten Abschnitt lernen Sie, wie Sie eine Subdomain komplett an einen anderen DNS-Server *delegieren*.

Übungen

9.1 [!1] Wir haben die Parallele zwischen DNS-Namen und Unix-Dateinamen hervorgehoben. Was ist in der Welt der Unix-Dateinamen das Äquivalent zu `$ORIGIN`?

9.2 Delegation an andere DNS-Server

Wenn eine Subdomain an einen anderen Server *delegiert* wird, bedeutet das, dass sie in eine eigene Zone ausgegliedert wird. Während es im einfachen Beispiel aus dem vorigen Abschnitt nur das `SOA`-Record für die Zone `example.com` gab und die Subdomain `blue` ebenfalls Teil dieser Zone war, besteht der erste Schritt bei der Delegation darin, für die Subdomain eine eigene Zonendatei mit einem eigenen `SOA`-Record anzulegen:

```
; db/blue.example.com
$TTL 3h
@       IN SOA   sky.blue.example.com. hostmaster.example.com (
                 2008101501 3h 1h 1w 1h )

        IN NS    sky
        IN NS    navy

sky     IN A     192.168.2.1
navy    IN A     192.168.2.2
steel   IN A     192.168.2.3
cadet   IN A     192.168.2.4
www     IN CNAME steel
```

Sie können diese Subdomain auf einem DNS-Server ablegen (mit einem geeigneten Eintrag in der Datei `named.conf`, selbstverständlich) und sie dort mit `dig` ausprobieren. (Über die Rückwärtsabbildung von IP-Adressen auf Namen reden wir später.)

Verkneifen Sie sich die naheliegende Idee, zum Testen die Subdomain im selben BIND unterzubringen wie die übergeordnete Domain. Das funktioniert nicht. Sie brauchen auf jeden Fall zwei verschiedene BINDs; ob das zwei Rechner, zwei virtuelle Maschinen oder zwei unabhängige BIND-Prozesse auf demselben Rechner sind, ist Ihnen überlassen.

Verbindung herstellen

Wenn die Subdomain auf ihrem eigenen DNS-Server gut funktioniert, ist es an der Zeit, die Verbindung zwischen der übergeordneten Domain und der Subdomain herzustellen – sie auf dem DNS-Server für die übergeordnete Domain zu *delegieren*. Solange das nicht passiert, existiert die Subdomain für das »wirkliche« DNS nämlich nicht. Wir müssen erreichen, dass ein rekursiver DNS-Server, der beim Auflösen eines Namens wie `www.blue.example.com` beim DNS-Server für `example.com` ankommt, von diesem an einen DNS-Server für `blue.example.com` weitergereicht wird. Dazu brauchen wir in der Zonendatei für `example.com` entsprechende `NS`-Records:

```
; db/example.com
<<<<<
blue      IN NS  sky.blue.example.com.
          IN NS  navy.blue.example.com.
<<<<<
```

Wenn Sie gut aufgepasst haben, sollte Ihnen etwas sehr Unangenehmes aufgefallen sein: Wir verweisen den anfragenden DNS-Server für Namen in der Subdomain `blue.example.com` an den zuständigen DNS-Server `sky.blue.example.com`. Dieser DNS-Server ist aber selbst in der fraglichen Zone, so dass seine erste Aufgabe darin bestehen müsste, uns seine IP-Adresse mitzuteilen, die wir brauchen, um ihn zu kontaktieren, damit er uns seine IP-Adresse mitteilt … die Katze beißt sich in den Schwanz.

glue records

Um dieses Problem zu lösen, verwendet man sogenannte *glue records*. Das heißt, außer den `NS`-Records für die DNS-Server der Subdomain fügen wir in die übergeordnete Zone auch noch `A`-Records für diese DNS-Server ein – eben die *glue records*, die die beiden Domains quasi »zusammenkleben«. Das sieht dann so aus:

```
; db/example.com
<<<<<
blue      IN NS sky.blue.example.com.
          IN NS navy.blue.example.com.
sky.blue  IN A  192.168.2.1
navy.blue IN A  192.168.2.2
<<<<<
```

Glue records werden nur gebraucht, wenn die DNS-Server für die Zone tatsächlich in der Zone selbst liegen. Sind die DNS-Server woanders, dann genügt das »normale« DNS, um die Namen aufzulösen: Eine Konfiguration wie

```
; db/example.com
<<<<<
blue      IN NS ns1.provider.net.
          IN NS ns2.provider.net.
<<<<<
```

würde ohne weitere Anstrengungen und »Klebstoff« funktionieren.

Es versteht sich von selbst, dass die Verantwortlichen für die Domains `example.com` und `blue.example.com` hier eng zusammenarbeiten müssen, damit die `NS`- und `A`-Records, die die Verbindung zwischen den beiden Domains herstellen, aktuell bleiben. In Abschnitt 9.4 lernen Sie eine Methode kennen, die diese Kooperation erleichtert.

Delegation ist eines der Fundamente, auf denen das DNS errichtet ist. Was in unserem Beispiel zwischen `example.com` und `blue.example.com` passiert ist, ist genau dasselbe, was auch auf einer etwas globaleren Ebene zwischen `com`

und example.com passieren müßte: Jemand muss in der com-Zone NS-Records und *glue records* für die DNS-Server anlegen, die für example.com autoritativ sind. Dito natürlich auch für . und com – aber das ist endgültig nicht mehr unsere Baustelle …

Übungen

9.2 [!2] Definieren Sie auf einem anderen DNS-Server (weiterer Rechner, andere virtuelle Maschine, notfalls andere BIND-Instanz auf Ihrem Rechner, in Präsenztrainings der Rechner Ihres Platznachbarn, …) eine Subdomain Ihrer Domain und sorgen Sie dafür, dass die Delegation funktioniert, indem Sie alle nötigen NS- und Glue-Records definieren.

9.3 Delegation für Rückwärts-Zonen

Was wir im vorigen Abschnitt für die Vorwärts-Abbildung von Namen auf Adressen kennengelernt haben, wird natürlich auch für die Rückwärts-Abbildung von Adressen auf Namen gebraucht, wenn die Verantwortung für die Vergabe von IP-Adressen von einer Person (oder Organisation) zu einer anderen wechselt. Leider sind die Dinge hier ein gutes Stück fremdartiger und komplizierter.

Delegation an Oktett-Grenzen Am einfachsten ist es, wenn der Verantwortungsübergang an einer Oktett-Grenze stattfindet – anschaulicher gesagt: Ihr Provider hat Ihnen einen Adressbereich zugeteilt, der genau 256 oder 65536 Adressen umfasst, zum Beispiel 11.12.13.0/24 oder 22.33.0.0/16. Dies läßt sich nämlich sehr einfach auf die Rückwärts-Zonen abbilden.

Delegation an Grenzen anderswo Hier wird es unappetitlich: Entweder haben Sie in derselben Rückwärts-Zone mehrere Subnetze (was zu administrativen Schwierigkeiten führen kann) oder Sie haben pro Subnetz mehrere Rückwärts-Zonen (was auch nicht viel besser ist).

Nehmen wir mal an, Sie wollen das Netz 10.0.0.0/8 (also die Adressen der Form $10.x.y.z$ in 16.384 Subnetze mit jeweils maximal 1.024 Adressen unterteilen. Die Netzmaske für jedes dieser Netze ist 255.255.252.0. Ein mögliches Subnetz ist zum Beispiel 10.1.128.0/22, und es enthält die Adressen 10.1.128.0 bis 10.1.131.255. Um die entsprechende Rückwärts-Auflösung in der Zone 10.in-addr.arpa zu konfigurieren, müssten Sie also ein Bündel NS-RRs der Form

```
128.0.10.in-addr.arpa.  IN NS  ns1.example.com.
128.0.10.in-addr.arpa.  IN NS  ns2.example.com.
129.0.10.in-addr.arpa.  IN NS  ns1.example.com.
129.0.10.in-addr.arpa.  IN NS  ns2.example.com.
130.0.10.in-addr.arpa.  IN NS  ns1.example.com.
130.0.10.in-addr.arpa.  IN NS  ns2.example.com.
131.0.10.in-addr.arpa.  IN NS  ns1.example.com.
131.0.10.in-addr.arpa.  IN NS  ns2.example.com.
```

anlegen (wir brauchen ja mindestens zwei DNS-Server pro Zone). Das ist ärgerliche Tipperei und nicht schön.

Aktuelle BIND-Versionen (BIND 9.1.0 und später) erlauben eine Direktive namens $GENERATE, mit der Sie die Erstellung dieser RRs etwas vereinfachen können: $GENERATE

```
$GENERATE 128-131 $.0.10.in-addr.arpa. IN NS ns1.example.com.
$GENERATE 128-131 $.0.10.in-addr.arpa. IN NS ns2.example.com.
```

erzeugt eine Reihe von RRs, bei denen das `$` sukzessive durch die Werte zwischen 128 und 131 ersetzt werden.

Angenommen, Ihre Firma verfügt über ein »Class-C«-Netzwerk[1], also einen Satz von 256 IP-Adressen – etwa `11.12.13.0/24` –, und Sie möchten dieses Netz weiter unterteilen, zum Beispiel in vier gleich große Subnetze mit den Adressen `11.12.13.0/26`, `11.12.13.64/26`, `11.12.13.128/26` und `11.12.13.192/26`. Um die Rückwärts-Auflösung für diese Subnetze administrativ zu trennen, haben Sie die Wahl zwischen drei relativ unappetitlichen Möglichkeiten:

- Sie können dasselbe Verfahren einsetzen wie eben beschrieben. Das heißt, Sie spucken in die Hände und stellen eine lange Liste von `PTR`-RRs der Form

```
1.13.12.11.in-addr.arpa.   IN NS ns1.example.com.
1.13.12.11.in-addr.arpa.   IN NS ns2.example.com.
2.13.12.11.in-addr.arpa.   IN NS ns1.example.com.
2.13.12.11.in-addr.arpa.   IN NS ns2.example.com.
◁◁◁◁◁
63.13.12.11.in-addr.arpa.  IN NS ns1.example.com.
63.13.12.11.in-addr.arpa.  IN NS ns2.example.com.
;
65.13.12.11.in-addr.arpa.  IN NS ns1.example.com.
65.13.12.11.in-addr.arpa.  IN NS ns2.example.com.
◁◁◁◁◁
127.13.12.11.in-addr.arpa. IN NS ns1.example.com.
127.13.12.11.in-addr.arpa. IN NS ns2.example.com.
◁◁◁◁◁
```

  auf (Brr.) (Mit `$GENERATE` können Sie sich eine gewisse Palliativbehandlung angedeihen lassen.)

- Eine etwas weniger schmerzhafte Methode geht zurück auf Glen Herrmansfeldt und wurde seitdem in [RFC2317] offiziell gemacht. Sie besteht darin, die IP-Adressen über `CNAME`-Records auf neue Subdomain-Namen abzubilden, die dann über separate Zonendateien delegiert werden können. Beispielsweise würde sich etwas anbieten wie

```
0-63.13.12.11.in-addr.arpa.   IN NS    ns1.example.com.
                              IN NS    ns2.example.com.
64-127.13.12.11.in-addr.arpa. IN NS    ns1.example.com.
                              IN NS    ns2.example.com.
◁◁◁◁◁
1.13.12.11.in-addr.arpa.      IN CNAME 1.0-63.13.12.11.in-addr.arpa.
2.13.12.11.in-addr.arpa.      IN CNAME 2.0-63.13.12.11.in-addr.arpa.
◁◁◁◁◁
65.13.12.11.in-addr.arpa.     IN CNAME 65.64-127.13.12.11.in-addr.arpa.
66.13.12.11.in-addr.arpa.     IN CNAME 66.64-127.13.12.11.in-addr.arpa.
◁◁◁◁◁
```

  Jede von den neuen Subdomains enthält dann nur noch die `PTR`-Records für die jeweiligen abgebildeten Namen, die Zonendatei für `11.12.13.128/26` zum Beispiel etwas wie

```
$ORIGIN 128-191.13.12.11.in-addr.arpa.
◁◁◁◁◁
129    IN PTR foo.marketing.example.com.
130    IN PTR bar.marketing.example.com.
◁◁◁◁◁
```

[1] Sowas gibt es offiziell ja nicht mehr, aber wir tun mal so …

und kann auf einem separaten Server administriert werden.

- Zu guter Letzt können Sie das ganze Problem hinter sich lassen und die Rückwärtszone einfach komplett selber verwalten. Wir würden Ihnen das nicht wirklich übelnehmen.

Wenn Sie Provider sind (oder einen im Fernsehen spielen), gibt es auch noch den perfiden Trick, einen Namen aus der Rückwärts-Zone per `CNAME`-Record in die Vorwärts-Zone zu verlegen, so dass der Kunde (der die Vorwärts-Zone direkt warten kann) sich dann einen Namen für die Rückwärtsabbildung aussuchen kann. In der Rückwärts-Zone steht dann etwas wie

```
14.13.12.11.in-addr.arpa.  IN CNAME 14.rev.example.com.
```

und in der Vorwärts-Zone etwas wie

```
14.rev.example.com. IN PTR  server.example.com.
```

(Die üblichen Schreibvereinfachungen kommen in Frage.)

Übungen

9.3 [!2] Verwenden Sie `$GENERATE`, um `A`-Records für die Rechner `pc1.example.com`, `pc2.example.com`, ..., `pc100.example.com` anzulegen. Diese sollen die IP-Adressen `10.0.1.1` bis `10.0.1.100` zugeordnet bekommen.

9.4 [2] Geben Sie die `NS`- und `PTR`-Records an, die nötig sind, um die Rückwärts-Zone für ein Netz mit der Adresse `192.168.1.0/24` zu definieren, das in zwei Subnetze mit je 128 Adressen unterteilt ist.

9.4 Stub-Zonen

In Abschnitt 9.2 haben Sie gesehen, wie Sie einen Teil einer Zone auf einen anderen DNS-Server auslagern – delegieren – können. Dabei war es unbequemerweise nötig, die »Verbindung« zwischen Ihrem DNS-Server und dem entfernten Server manuell über `NS`-Records und *glue records* herzustellen. BIND unterstützt allerdings auch eine (experimentelle) Methode, dies zu vereinfachen, die sogenannten Stub-Zonen.

Wenn Sie in einem DNS-Server eine Stub-Zone für eine Delegation konfigurieren, dann schickt der Server in periodischen Abständen Anfragen nach den `SOA`- und `NS`-Records an den DNS-Server der Stub-Zone. Auch nach *glue records* wird gegebenenfalls gefragt. Die Delegation richtet sich dann nach diesen Datensätzen. Ändern die Administratoren der delegierten Zone etwas an ihrer Konfiguration (etwa indem die `NS`-Records modifiziert werden), bemerkt Ihr Server das anhand der neuen Seriennummer im `SOA`-Record und kann sich die neuen Daten für die Delegation holen.

Eine Stub-Zone könnte ungefähr so aussehen:

```
zone "blue.example.com" {
    type stub;
    masters { 192.168.2.1; };
    file "stub.blue.example.com";
};
```

Die `masters`-Direktive verweist dabei auf den primären DNS-Server der delegierten Zone.


Bei BIND 9 ist es notwendig, die Stub-Zone für `blue.example.com` in allen (auch den sekundären) DNS-Servern für `example.com` zu definieren. Das liegt daran, dass BIND 9 die Informationen über die Delegation von `blue.example.com` nicht in die Zonendaten für `example.com` aufnimmt, so dass die sekundären Server für `example.com` sie nicht über einen Zonentransfer bekommen können. Die separate Definition der Stub-Zone auf jedem Server sorgt dafür, dass die sekundären Server aktuell bleiben.


Stub-Zonen funktionieren natürlich entsprechend auch für Rückwärts-Zonen.

Übungen


9.5 [2] Überzeugen Sie sich, dass Stub-Zonen funktionieren: Definieren Sie in der Konfiguration Ihres DNS-Servers, der eine Subdomain an einen anderen DNS-Server delegiert, eine Stub-Zone für diese Subdomain und entfernen Sie die manuell eingebauten `NS`- und Glue-Records für die Delegation. Testen Sie die Namensauflösung für die Zone mit `dig` oder `host`.

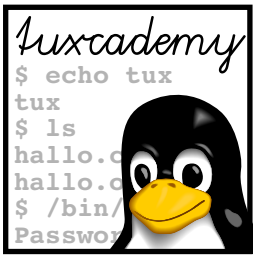

9.6 [2] Machen Sie dasselbe auch mit der korrespondierenden Rückwärts-Zone und prüfen Sie die Funktionalität.

Zusammenfassung

- Mit `$ORIGIN` können Sie bequem »Subdomains« in einer Zone definieren.
- Sie können Subdomains an andere DNS-Server delegieren, indem Sie entsprechende `NS`-Records setzen.
- Mit *glue records* können Sie das Problem lösen, dass der DNS-Server für eine delegierte Zone in der delegierten Zone selbst angesiedelt ist.
- Rückwärts-Zonen können (bzw. müssen) genauso delegiert werden wie Vorwärts-Zonen. Wenn der Verantwortungsübergang an Oktett-Grenzen in der Adresse stattfindet, ist das einfach, ansonsten umständlich bis ekelhaft.
- Stub-Zonen dienen zur automatischen Verwaltung von Delegationsinformationen.

Literaturverzeichnis

RFC2317 H. Eidnes, G. de Groot, P. Vixie. »Classless IN-ADDR.ARPA delegation«, Mârz 1998. `http://www.ietf.org/rfc/rfc2317.txt`



10

DNS-Tipps und -Tricks

Inhalt

10.1 Adressensuchlisten 124
10.2 Weiterleitung an andere Server 125
10.3 Weiterleitungs-Zonen 127

Lernziele

- Adressensuchlisten verstehen und definieren können
- Weiterleitung für DNS-Server und Zonen definieren können

Vorkenntnisse

- Kenntnisse über Linux-Systemkonfiguration
- TCP/IP-Kenntnisse
- DNS-Grundkenntnisse (siehe Kapitel 5)
- BIND-Grundkenntnisse (siehe Kapitel 6)

10.1 Adressensuchlisten

Adressensuchlisten

Bei BIND ist es möglich, Zugriff auf diverse Eigenschaften des Servers (etwa Zonentransfers oder die Daten in bestimmten Zonen) auf Anfragen zu beschränken, deren Absender-IP-Adressen gewissen Regeln genügen. Diese Regeln werden in **Adressensuchlisten** (engl. *address match lists*) beschrieben.

Adressensuchlisten bestehen aus einer Folge von Einträgen, die jeweils mit einem Semikolon abgeschlossen werden müssen. Als Einträge kommen in Frage:

IP-Adressen Diese werden wie üblich als *dotted quads* notiert.

IP-Präfixe Ein IP-Präfix besteht aus der Adresse eines IP-Netzes (als *dotted quad* oder Anfangsstück eines solchen), gefolgt von einem Schrägstrich, gefolgt von der Anzahl der 1-Bits in der dazugehörigen Netzmaske.

Benannte Adressensuchregeln Eine Adressensuchliste kann einen Namen haben (siehe unten), und mit diesem Namen können Sie in anderen Adressensuchlisten auf die dazugehörige Liste verweisen.

Die Adressensuchliste

```
127.0.0.1; 11.22/16; "secondaries";
```

zum Beispiel verweist auf den einzelnen Rechner mit der IP-Adresse `127.0.0.1`, das Netz `11.22.0.0/16` und die in der Adressensuchliste `secondaries` beschriebenen Rechner oder Netze.

Außerdem dürfen Sie auch Namen von TSIG-Schlüsseln (Abschnitt 10.1) in der Form

```
key dhcp-server.example.com.
```

angeben. Dies ist zum Beispiel in `allow-transfer` nützlich.

vordefinierte Adressensuchlisten

Es gibt vier vordefinierte Adressensuchlisten:

`none` Die leere Adressensuchliste

`any` Alle IP-Adressen

`localhost` Alle IP-Adressen des Rechners, auf dem der DNS-Server läuft

`localnets` Alle IP-Adressen in Netzen, die direkt an irgendeine Netzwerkschnittstelle des Rechners angeschlossen sind, auf dem der DNS-Server läuft

Adressensuchlisten dürfen an diversen Stellen in der BIND-Konfiguration auftauchen; einige davon haben Sie schon kennengelernt.

Einzelne Einträge in Adressensuchlisten können auch »negiert« werden, indem Sie ein Anführungszeichen davor schreiben. Etwas wie

```
!11.22/16;
```

läßt eine Operation für Rechner im beschriebenen Netz *nicht* zu.

Adressensuchlisten werden »von links« abgearbeitet, bis ein passender Eintrag gefunden wird. Bei etwas wie

```
allow-transfer { !11.22.33.44; 11.22/16; };
```

gilt zum Beispiel Folgendes:

- Eine Anfrage vom Rechner `11.22.33.55` passt nicht auf den ersten Eintrag (`!11.22.33.44`), aber auf den zweiten. Dieser Rechner dürfte also einen Zonentransfer vornehmen.
- Eine Anfrage vom Rechner `11.22.33.44` passt schon auf den ersten Eintrag. Da der Eintrag durch ein vorangestelltes »!« negiert ist, wird der Zonentransfer abgelehnt.

Wenn überhaupt kein passender Eintrag in der Liste steht, wird die gewünschte Operation (hier der Zonentransfer) ebenfalls abgelehnt.

Verfallen Sie nicht dem Irrglauben, bei etwas wie

```
allow-transfer { !11.22/16; };
```

würde die Operation für alle Rechner, die *nicht* im beschriebenen Netz sind, *zugelassen*. Wenn Sie das wollen, müssen Sie

```
!11.22/16; any;
```

schreiben.

Name vergeben

Sie selbst können einer Adressensuchliste einen Name vergebenNamen geben, indem Sie einen `acl`-Eintrag in der Datei `/etc/named.conf` machen, etwa wie folgt:

```
acl "meine-liste" { 127.0.0.1; 11.22/16; };
```

Anschließend können Sie sich anderswo in der Konfiguration auf diese Liste beziehen:

```
acl "weitere-liste" { "meine-liste"; };
```

Die Abkürzung `acl` kennen Sie vielleicht als *access control list*. Lassen Sie sich davon nicht verwirren: BIND verwendet Adressensuchlisten, die Sie mit `acl` definieren können, an diversen Stellen und nicht nur zur Zugriffskontrolle.

Übungen

10.1 [!1] Geben Sie eine Adressensuchliste an, die (etwa mit `allow-transfer`) Zugriffe von Stationen aus dem Netz `10.11.12.0/24` erlaubt, von anderen Stationen aus dem Netz `10.11.0.0/16` verbietet und von allen übrigen Stationen erlaubt.

10.2 Weiterleitung an andere Server

Mitunter ist es sinnvoll, dass ein DNS-Server eine rekursive Anfrage nicht selber zu beantworten versucht, sondern sie an einen anderen Server zur Beantwortung weiterleitet. Hierfür sprechen mehrere mögliche Gründe:

- Der andere Server hat einen riesengroßen Cache und kann viele Anfragen direkt aus seinem Cache beantworten, was Zeit spart. Beispielsweise könnten in einer großen Firma oder Hochschule die DNS-Server in den einzelnen Abteilungen oder Fachbereichen so konfiguriert sein, dass sie alle Anfragen an einen oder mehrere zentrale DNS-Server weiterleiten. Diese kümmern sich dann um die tatsächliche Beantwortung und speichern das Ergebnis zwischen, so dass eine identische Anfrage aus einer anderen Abteilung unmittelbar aus dem Cache beantwortet werden kann.

- In einer Firewall-Infrastruktur haben die DNS-Server im internen, vertrauenswürdigen Netz nicht die Möglichkeit, DNS-Server im Internet anzusprechen und von diesen Antworten geschickt zu bekommen, sondern alle externen DNS-Anfragen sollen über einen DNS-Server in der DMZ geleitet werden.
- Eine Installation ist nur über eine sehr langsame Verbindung an die Außenwelt angeschlossen, und überflüssiger Datenverkehr soll nach Möglichkeit vermieden werden.

Sie können BIND dazu bringen, alle Anfragen an einen bestimmten anderen DNS-Server (oder mehrere) zu schicken, indem Sie in der Konfigurationsdatei eine `forwarders`-Option verwenden:

```
options {
    forwarders { 11.12.13.14; 11.12.13.15; };
};
```

(Die DNS-Server, die das *Ziel* der Weiterleitung darstellen, müssen nicht besonders konfiguriert werden.)


Der Parameter von `forwarders` ist natürlich eine Adressensuchliste im Sinne von Abschnitt 10.1.

Wenn in einem DNS-Server Weiterleitung konfiguriert wurde, versucht dieser wie bisher zunächst, eine Anfrage aus dem lokalen Cache zu befriedigen. Stehen die gewünschten Informationen nicht im Cache, schickt der DNS-Server die Anfrage (rekursiv) an einen der Forwarder und wartet einen kurzen Moment. Kommt keine Antwort, versucht er – wie sonst auch – die Anfrage selbst (iterativ) zu beantworten. Diese Betriebsart bezeichnet die BIND-Konfiguration auch als »`forward first`«, eben weil zunächst weitergeleitet und dann gegebenenfalls iterativ gesucht wird.

Mit der Option »`forward only`« können Sie dafür sorgen, dass BIND *nur* die Weiterleitungsziele befragt und die Anfrage nicht selber iterativ aufzulösen versucht:

```
options {
    forwarders { 11.12.13.14; 11.12.13.15; };
    forward only;
};
```

In diesem Fall müssen auch tatsächlich `forwarders` konfiguriert sein.


Im wirklichen Leben ist es wahrscheinlich günstiger, »`forward only`« zu verwenden als »`forward first`«. Das liegt daran, dass die Wartezeit auf einen als Weiterleitungsziel definierten DNS-Server so lang sein kann, dass der Resolver, der die ursprüngliche DNS-Anfrage gestellt hat, schon Misserfolg an die Anwendung signalisiert (oder kurz davor ist). Damit hängt das Resultat – aufgelöster Name oder Fehlermeldung – davon ab, wie schnell die iterative Auflösung funktioniert, und das führt letzten Endes zu verwirrenden Ergebnissen für den Client.


BIND – jedenfalls in halbwegs neuen Versionen – konsultiert die Server in der `forwarders`-Liste nicht in der angegebenen Reihenfolge, sondern wählt den ersten anhand der Antwortzeiten auf frühere Anfragen. Dies ist vorteilhaft, wenn der erste DNS-Server in der Liste nicht erreichbar ist, da dieser dann nicht zwecklos angesprochen wird, bevor BIND sich an einen anderen (funktionierenden) DNS-Server wendet.

Übungen

10.2 [1] Einen DNS-Server ohne eigene Zonendaten und mit einer globalen Weiterleitung bezeichnet man auch als *forward-only*-Server. Wo könnte man so einen DNS-Server sinnvoll einsetzen?

10.3 [!2] Überzeugen Sie sich, dass die globale Weiterleitung von Anfragen an einen anderen DNS-Server funktioniert, etwa indem Sie BIND so konfigurieren, dass er alle Anfragen an den DNS-Server Ihres Schulungszentrums oder Ihres DSL-Routers weiterleitet.

10.3 Weiterleitungs-Zonen

Statt *alle* Anfragen an Ihren DNS-Server an andere Server weiterzuleiten, können Sie auch dafür sorgen, dass das nur für bestimmte Anfragen passiert. Zum Beispiel können Sie alle Anfragen nach Namen in einer bestimmten Zone an einen festen Server weiterleiten (oder mehrere), während alle anderen Anfragen iterativ aufgelöst werden. Dazu gibt es »Weiterleitungs-Zonen«, die Sie wie folgt definieren können:

```
zone "example.net" {
    type forward;                                    Weiterleitungs-Zone
    forwarders { 12.13.14.15; 12.13.14.16; };        Nur für diese Zone
};
```

Die Weiterleitung im Beispiel gilt für alle Namen, die mit `example.net` aufhören, auch wenn sie in »Unterzonen« liegen, die wiederum auf anderen DNS-Servern zu finden sind.

Umgekehrt können Sie Weiterleitungs-Zonen auch verwenden, um einen DNS-Server, der eine globale Weiterleitung nutzt, bestimmte Zonen *nicht* weiterzuleiten, sondern auf dem üblichen Weg aufzulösen. Diese Art von Weiterleitungs-Zone wird nicht als »`type forward`« definiert, sondern wie eine gewöhnliche Zone, allerdings mit einer `forwarders`-Direktive:

```
options {
    forwarders { 10.0.0.253; 10.0.0.254 };
};
◁◁◁◁◁
zone "example.com" {
    type slave;
    masters { 10.0.0.1; };
    file { "bak.example.com" };
    forwarders {};
};
```

Sie brauchen diese Definition allerdings nur, wenn die Zone nicht komplett auf dem betreffenden Server liegt, sondern Teile davon auf andere Server ausgelagert sind. Namen, für die Ihr Server autoritativ ist – hier zum Beispiel `foo.example.com` –, beantwortet er natürlich aus seinem eigenen Datenbestand, aber Namen in einer delegierten Subdomain – vielleicht `sky.blue.example.com` – würden an die `forwarders` weitergeleitet. Die leere `forwarders`-Direktive dagegen sorgt dafür, dass Ihr Server diese Namen auf dem normalen Weg auflöst, also indem er sich die `NS`-Records für die Subdomain anschaut und die betreffenden DNS-Server direkt fragt.

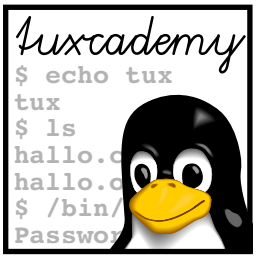
Übungen

10.4 [2] Vergewissern Sie sich, dass Weiterleitungs-Zonen funktionieren, indem Sie auf Ihrem Rechner eine Weiterleitungs-Zone für eine Zone installieren, für die der DNS-Server auf einem anderen Rechner autoritativ ist. Testen Sie die Weiterleitung, indem Sie mit `dig` oder `host` gezielt Ihren Rechner nach DNS-Ressourcen fragen.

10.5 [2] Testen Sie die Nicht-Weiterleitung von Anfragen über Zonen, für die Ihr DNS-Server autoritativ ist, per »`forwarders {}`«. Delegieren Sie dazu eine Subdomain auf einen anderen Rechner (denken Sie an Übung 9.2) und konfigurieren Sie die *übergeordnete Domain* dann auf Ihrem Rechner als Weiterleitungs-Zone mit leerem `forwarders`. Das funktioniert natürlich am besten, wenn Ihr Rechner ansonsten eine globale Weiterleitung verwendet, aber das haben Sie ja gerade in Übung 10.3 gemacht.

Zusammenfassung

- Adressensuchlisten (*address match lists*) erlauben das Einschränken diverser Leistungen von BIND auf Anfragesteller, deren IP-Adressen bestimmten Anforderungen genügen.
- Sie können BIND dazu bringen, alle Anfragen oder Anfragen an Daten in bestimmten Zonen gezielt an andere DNS-Server weiterzuleiten.



11

DNS-Sicherheit: Grundlagen

Inhalt

11.1 DNS und Sicherheit 130
11.2 BIND ohne `root` 132
11.3 BIND und `chroot` 133
11.4 Getrennte rekursive und autoritative Server 135
11.4.1 Warum trennen? 135
11.4.2 Umsetzung 136

Lernziele

- Wichtige Sicherheitsprobleme mit DNS kennen und einschätzen können
- BIND für den Einsatz in einer `chroot`-Umgebung konfigurieren können
- Getrennte rekursive und autoritative DNS-Server parallel betreiben können

Vorkenntnisse

- Kenntnisse über Linux-Systemkonfiguration
- TCP/IP-Kenntnisse
- DNS-Grundkenntnisse (siehe Kapitel 5)
- BIND-Grundkenntnisse (siehe Kapitel 6)

11.1 DNS und Sicherheit

DNS gehört zum »Urgestein« des Internet und darum ist es vieleicht nicht überraschend, dass Sicherheit beim Entwurf des Protokolls keine bedeutende Rolle gespielt hat. Beispielsweise werden DNS-Anfragen und -Antworten nicht authentisiert, so dass ein DNS-Client streng genommen nie sicher sein kann, ob die Antwort, die er auf eine Anfrage bekommt, (a) von einem Server kommt, der »befugt« ist, überhaupt Antworten auf die Anfrage zu geben (Caching, obwohl aus anderen Gründen wichtig und nützlich, macht das natürlich nicht einfacher), und (b) nicht von irgendeinem bösen Buben (oder Mädchen) auf dem Weg vom Server verfälscht wurde (»DNS-Spoofing«).

Die DNS-Gemeinde hat das letzte Jahrzehnt (oder so) damit verbracht, auf dieses Problem mit dem großen Hammer der Kryptografie einzuprügeln. Die Resultate – unter dem Namen »DNSSEC« sind inzwischen halbwegs vorzeigbar.

DNSSEC

Neben diesem fundamentalen (und zum Glück anscheinend halbwegs erledigten) Problem gibt es noch einige andere Ärgernisse, die einen DNS-Administrator plagen können:

Cache-Vergiftung (engl. *cache poisoning*) Hierbei bringt ein Angreifer gefälschte Daten in den Cache eines DNS-Servers ein, die dieser dann an andere DNS-Server (und damit letzten Endes an Clients wie Web-Browser) weitergibt. DNS-Cache-Vergiftung ist ein wichtiger Bestandteil von »Phishing«- und »Pharming«-Angriffen, deren Ziel es ist, zum Beispiel Kreditkartennummern oder PIN/TAN-Paare von Online-Banking-Kunden zu erbeuten.

Cache-Vergiftung kann grundsätzlich auf zwei Arten passieren:

- Der Angreifer kann versuchen, den DNS-Server dazu zu bringen, eine rekursive Anfrage nach dem zu verfälschenden Namen zu stellen. Dann versucht der Angreifer, ihm eine Antwort zu schicken, die vor der des legitimen entfernten DNS-Servers eintrifft. Gelingt das, legt der angegriffene DNS-Server die gefälschte Antwort in seinem Cache ab und verwirft die später eintreffende richtige Antwort.

  Um dieses Problem zu entschärfen, sieht das DNS-Protokoll vor, dass eine Anfrage mit einer 16 Bit breiten »Transaktionsnummer« versehen wird, die in der Antwort wiederholt werden muss. Moderne DNS-Server wie BIND wählen diese Nummer zufällig, so dass ein Angreifer – wenn er nicht in einer Position ist, die Anfrage an den autoritativen Server für den gesuchten Namen abzufangen – den richtigen Wert raten muss. Damit ist diese Sorte Angriff sehr aufwendig und auch auffällig.

  Früher pflegten DNS-Server die Transaktionsnummer fortlaufend zu vergeben, so dass ein Angreifer, der einmal die aktuelle Nummer gesehen hatte – etwa indem er den anzugreifenden Server als Erstes eine rekursive Anfrage an einen vom Angreifer kontrollierten DNS-Server stellen ließ –, mit weitaus größeren Erfolgschancen die Nummer der späteren »Vergiftungs«-Anfrage raten konnte.

- Der Angreifer kann versuchen, zusätzliche RRs an eine legitime Antwort anzuhängen, die der angegriffene DNS-Server dann naiverweise in seinen Cache aufnimmt (denken Sie an die ADDITIONAL SECTION in Abschnitt 5.4.2). Um einem Opfer zum Beispiel ein `A`-Record unterzuschieben, das `www.example.com` auf die (gefälschte) Adresse `11.22.33.44` abbildet, muss der Angreifer also nur einen DNS-Server unter seine Kontrolle bringen, der dann an seine legitimen Antworten dieses RR anhängt. Leichtgläubige andere DNS-Server legen es dann ungeprüft in ihrem Cache ab, und deren Clients bekommen die falsche Adresse

serviert, so dass zum Beispiel Web-Benutzer unbemerkt auf den Server `11.22.33.44` umgeleitet werden – wo der Angreifer natürlich eine Webseite abgelegt hat, die der legitimen Seite `www.example.com` täuschend ähnlich sieht.

Grundsätzlich sollte HTTPS dazu beitragen, dass solche Machenschaften zumindest im Web nicht funktionieren – der Angreifer müßte seinen gefälschten Web-Server auch dazu bringen, ein beglaubigtes Zertifikat für `www.example.com` auszuliefern. Erfahrungen der jüngeren Zeit haben jedoch gezeigt, dass es anscheinend grundsätzlich nicht schwer ist, sich beliebige beglaubigte Zertifikate zu besorgen, und die Details von HTTPS sind der breiten Masse von Anwendern auch so wenig bekannt, dass Anomalien selten den Argwohn erwecken, den sie eigentlich erwecken sollten.

Aktuelle DNS-Server ignorieren zusätzliche RRs, die sich nicht auf die Domain beziehen, von der in der Anfrage die Rede war. Damit ist ein plumper Angriff nicht mehr möglich.

Mitte 2008 kam eine trickreiche Variation der Cache-Vergiftung ans Licht, bei der der Angreifer die Domain-Einschränkung umgeht, indem er den angegriffenen DNS-Server dazu bringt, sehr viele Anfragen nach Namen der Form `aaaaaa.example.com`, `aaaaab.example.com` usw. zu stellen. (Das ist zum Beispiel über eine Webseite möglich, die etliche tausend Verweise auf unsichtbare Bilder enthält.) Wenn viele Anfragen ausstehen, ist die Wahrscheinlichkeit höher, dass eine vom Angreifer gefälschte Antwort mit einer geratenen Transaktionsnummer zu irgendeiner ausstehenden Anfrage passt[1]. Der Angreifer sorgt aber dafür, dass jede einzelne gefälschte Antwort als zusätzliches RR die Abbildung von `www.example.com` auf `11.22.33.44` enthält, die der angegriffene DNS-Server auch ohne weiteres akzeptiert, da sie ja dieselbe Domain betrifft wie die aktuelle Anfrage. Dadurch ist die Wahrscheinlichkeit, dass die Cache-Vergiftung gelingt, deutlich höher als beim Raten von Transaktionsnummern einzelner Anfragen zwecks direkter Cache-Vergiftung.

Eine Milderung (keine Verhinderung) dieses Angriffs wird dadurch erzielt, dass der DNS-Server den UDP-Quellport für seine DNS-Anfragen zufällig wählt, statt etwa konstant den Port 53 zu verwenden. Das macht Zufallstreffer, bei denen Transaktionsnummer *und* Quellport übereinstimmen, deutlich unwahrscheinlicher. BIND 9 macht das seitdem so.

Auch Cache-Vergiftung ist nicht mehr möglich, wenn die Authentizität von RRs über DNSSEC gesichert ist.

Distributed Denial of Service (DDoS) Diese Angriffsform zielt darauf ab, DNS-Server mit Anfragen zu überlasten, so dass andere (legitime) Anfragen nicht mehr beantwortet werden können. Auf diese Weise kann zum Beispiel ein Web-Server praktisch unerreichbar gemacht werden, wenn Anfragen nach seiner IP-Adresse untergehen. DDoS funktioniert am besten, wenn dem Angreifer ein Botnet, also eine große Menge von (typischerweise) Windows-PCs mit »trojanischer« Software, zur Verfügung steht.

Gegen DDoS-Angriffe kann man als Opfer wenig bis nichts unternehmen. Selbst wenn man auf die Anfragen nicht zu reagieren versucht, verstopfen

[1] So etwas nennt man »Geburtstagsangriff« (engl. *birthday attack*), vor dem folgenden Hintergrund: Eine Gruppe von Personen muss außer Ihnen mindestens 253 andere Personen enthalten, damit die Wahrscheinlichkeit, dass mindestens eine davon am selben Tag Geburtstag hat wie *Sie*, größer ist als 50%. Damit die Wahrscheinlichkeit, dass mindestens *irgend zwei* Personen in der Gruppe am selben Tag Geburtstag haben, größer ist als 50%, muss die Gruppe aber nur aus 23 Personen bestehen.

sie doch die eigene Netzanbindung. Im Falle von DNS hilft natürlich eine möglichst weit verstreute Auswahl von sekundären Servern, die alle Ziel des Angriffs sein müssen, um einen Web-Server unerreichbar zu machen.

DNS Amplification Diese Angriffsform richtet sich nicht in erster Linie gegen DNS-Server, sondern instrumentalisiert diese im Rahmen von DDoS-Angriffen auf andere Rechner. Sie nutzt den Umstand aus, dass ein Angreifer durch relativ kleine Anfragen an einen DNS-Server ziemlich große Antworten provozieren kann. Wenn der Angreifer Anfragen an zahlreiche DNS-Server schickt und bei den entsprechenden UDP-Datagrammen die IP-Adresse des anzugreifenden Rechners als Absenderadresse einträgt, bekommt dieser jede Menge Antworten geschickt, was seine Netzanbindung überlasten kann.

Neben diesen Angriffen auf die fehlerfreie Funktion des DNS (oder anderer Netzwerkdienste) an sich gibt es natürlich auch immer die Möglichkeit, dass Angreifer versuchen, Sicherheitslücken in der DNS-*Implementierung* auszunutzen, um sich Zugriff auf den Rechner zu verschaffen, auf dem der DNS-Server läuft. Sie können die Latte für diese Sorte Angriff wesentlich höher legen, indem Sie BIND nicht mit `root`-Rechten laufen lassen und außerdem in eine `chroot`-Umgebung einsperren. Wie das geht, erklären wir in den nächsten Abschnitten.

Eine der wichtigsten Sicherheitsvorkehrungen, die Sie als Administrator eines DNS-Servers treffen können, besteht darin, immer die aktuellste Version von BIND zu verwenden. Die meisten Linux-Distributionen sind gut darin, BIND aktuell zu halten. Unterrichten Sie sich aus den Sicherheits-Mailinglisten Ihrer Distribution und werden Sie gegebenenfalls zeitnah aktiv.

Sie können herausfinden, welche Version von BIND bei Ihnen läuft, indem Sie nach `TXT`-Records für den Namen `version.bind` in der `CHAOSNET`-Klasse fragen:

```
$ host -t txt -c chaos version.bind 127.0.0.1
version.bind descriptive text "9.7.3"
```

Übungen

11.1 [2] (Recherche-Aufgabe:) Angenommen, Sie wollen der Welt nicht verraten, welche Version von BIND Sie einsetzen. Was können Sie unternehmen? Ist das eine gute Idee?

11.2 BIND ohne `root`

Grundsätzlich sollten Netzwerkdienste immer mit den geringstmöglichen Privilegien laufen. Eigentlich braucht BIND für nichts `root`-Rechte außer dafür, beim Start den Port 53 zum Lauschen zu öffnen; danach kann er auch die Rechte eines gewöhnlichen Benutzers verwenden, solange sichergestellt ist, dass er auf alle benötigten Dateien zugreifen darf.

Am besten legen Sie einen neuen Benutzer für BIND an (mit einer korrespondierenden Gruppe, die dieser Benutzer als primäre Gruppe zugeordnet bekommt) und sorgen dafür, dass BIND mit der Option »`-u` ⟨*Benutzername*⟩« aufgerufen wird. ⟨*Benutzername*⟩ ist dabei der Name des betreffenden Benutzers. BIND startet dann als `root`, nimmt die erforderlichen Initialisierungen vor und geht dann zum angegebenen Benutzernamen über.

BIND 9 verwendet immer die primäre Gruppe des betreffenden Benutzers als Gruppe. BIND 8 tut das nur, wenn Sie nicht auch die Option »`-g` ⟨*Gruppenname*⟩« angegeben haben, um eine bestimmte Gruppe auszuwählen. BIND 9 unterstützt die Option `-g` nicht mehr.


Wenn BIND nicht als root läuft, müssen Sie möglicherweise Einschränkungen in Kauf nehmen. Zum Beispiel ist es dann nicht möglich, dass BIND dynamisch hinzukommende Netzwerkschnittstellen erkennt. Auch mit gewissen Spielarten von IPsec oder IP-über-IP-Tunnelung kann es Probleme geben. Wenn Sie sowas benutzen: Wir haben Sie gewarnt.

Es ist gut möglich, dass Ihre Distribution schon für Sie organisiert hat, dass BIND nicht als root läuft.


Bei Debian GNU/Linux und den aktuellen Ubuntu-Distributionen verwendet BIND standardmäßig den Benutzer bind (und die Gruppe bind). Wenn Sie das nicht so haben wollen, können Sie die Definition von OPTIONS in der Datei /etc/default/bind9 (oder den konkreten Aufruf von BIND im Init-Skript /etc/init.d/bind9) ändern.

Übungen


11.2 [!1] Wie geht Ihre Distribution mit dem Thema um? Wo wird der Benutzer für BIND festgelegt und wie könnten Sie das ändern?


11.3 [2] Falls Ihre Distribution das nicht sowieso schon so macht: Sorgen Sie dafür, dass BIND einen Benutzer außer root verwendet.

11.3 BIND und chroot

Wenn BIND nicht als root, sondern als normaler Benutzer läuft, dann ist schon mal ausgeschlossen, dass ein Angreifer Zugriff auf alle Dateien des Computers bekommt, auf dem BIND läuft. Sie können das Gefahrenpotential noch weiter vermindern, indem Sie BIND in einer chroot-Umgebung laufen lassen.


Mit chroot können Sie dafür sorgen, dass ein Prozess (und seine Kinder) nur einen Teilbaum des Dateisystems wahrnehmen. Konzeptuell wird ein beliebiges Verzeichnis im Dateibaum das Wurzelverzeichnis des Dateibaums, den dieser Prozess sieht (deswegen chroot, kurz für *change root*). Es gibt ein Shellkommando namens chroot und einen gleichnamigen Systemaufruf.

Wie Sie BIND (9) dafür fit machen, können Sie aus dem »Chroot-BIND HOWTO« [Chroot-BIND HOWTO] lernen. Hier ist eine konzentrierte und angepasste Zusammenfassung der notwendigen Schritte.

1. Als erstes sollten Sie dafür sorgen, dass BIND als normaler Benutzer läuft (Abschnitt 11.2). Das ist dringend nötig, da ein Prozess, der als root läuft, aus einer chroot-Umgebung ausbrechen kann.

2. Dann ist es Zeit, die chroot-Umgebung zu präparieren. Nehmen wir mal an, die Umgebung soll in /chroot/bind untergebracht werden. Dann müssen Sie die folgenden Kommandos ausführen:

```
# mkdir -p /chroot/bind
# chmod 700 /chroot
# chown bind:bind /chroot/bind
# chmod 700 /chroot/bind
# cd /chroot/bind
# mkdir -p dev etc/bind var/run
# chown bind:bind var/run
```

3. In der chroot-Umgebung müssen einige Dateien vorhanden sein, die BIND benötigt. Führen Sie die folgenden Kommandos aus:

```
# mknod dev/null c 1 3                    Null-Device
# mknod dev/zero c 1 5                    Lieferant für Nullbytes
# mknod dev/random c 1 8                  Zufallsgenerator
# chmod 666 dev/*
# cp /etc/localtime etc                   Zeitzone des Servers
```

4. Kopieren Sie die BIND-Konfiguration aus dem `/etc`-Verzeichnis Ihres Servers in die `chroot`-Umgebung. Wie das genau geht, hängt von Ihrer Linux-Distribution ab; für Debian GNU/Linux funktioniert zum Beispiel

```
# cp /etc/bind/* etc/bind
```

Wenn Ihr BIND sekundärer DNS-Server für bestimmte Zonen ist, braucht er ein Verzeichnis in der `chroot`-Umgebung, wo er die Sicherheitskopien der Zonendaten ablegen kann. Dieses Verzeichnis muss für den BIND-Benutzer schreibbar sein.

5. Wenn BIND Protokollnachrichten über den Syslog-Dienst verschicken soll, braucht er Zugriff auf `/dev/log`. Da diese Datei außerhalb der `chroot`-Umgebung liegt, ist sie nicht mehr zugänglich. Je nachdem, wie Ihre Protokollierungs-Infrastruktur aussieht, müssen Sie sich also etwas Anderes überlegen. Zum Beispiel könnten Sie das `/dev/log` des Systems über einen Bind-Mount in `/chroot/bind/dev` aufnehmen oder BIND sein Protokoll einfach in Dateien schreiben lassen (in welchem Fall Sie in der `chroot`-Umgebung ein geeignetes für den BIND-Benutzer schreibbares Verzeichnis vorsehen müssen).

Neue Versionen von `syslogd` unterstützen eine Option »`-a`«, mit der Sie ein alternatives Unix-Domain-Socket für die Protokollierung angeben können. Wenn Sie `syslogd` mit der Option »`-a /chroot/bind/dev/log`« aufrufen, sollte das schon reichen. Nach einem Neustart legt `syslogd` das Socket an.

6. Nach dem Abschluss dieser Vorarbeiten können Sie die Schrauben noch etwas fester anziehen, etwa indem Sie bestimmte Dateien und Verzeichnisse unveränderbar machen:

```
# chattr +i etc etc/localtime var
```

7. Jetzt sind Sie schon fast fertig. Sie müssen nur dafür sorgen, dass BIND mit den folgenden Optionen aufgerufen wird:

```
-u bind                                   Normaler Benutzer
-t /chroot/bind                           chroot in die Umgebung
-c /chroot/bind/etc/bind/named.conf       Konfigurationsdatei
```

(Der Pfad der Konfigurationsdatei ist abhängig davon, wie Sie die `chroot`-Umgebung ausgestaltet haben.) Sie können das normalerweise erreichen, indem Sie entweder das Init-Skript für BIND oder die Datei mit den Voreinstellungen (etwa `/etc/default/bind9` oder `/etc/sysconfig/named`) anpassen.

Prüfen Sie unbedingt das Protokoll, das BIND beim Start schreibt, und überzeugen Sie sich, dass keine Dateien vermisst werden.

Sie können auch BIND 8 in einer `chroot`-Umgebung laufen lassen. Die Installation ist aber viel komplizierter, und da Sie ja sowieso BIND 9 benutzen sollen, sparen wir uns hier die Details.

Übungen

11.4 [2] Falls Ihre Distribution das nicht sowieso schon so macht: Sorgen Sie dafür, dass BIND in einer `chroot`-Umgebung ausgeführt wird.

11.4 Getrennte rekursive und autoritative Server

11.4.1 Warum trennen?

Die meisten Angriffe auf DNS beruhen darauf, dass der angegriffene DNS-Server über eine rekursive Anfrage dazu gebracht wird, auf einen DNS-Server zuzugreifen, der unter der Kontrolle des Angreifers steht – und der dann zum Beispiel »vergiftete« Daten liefern kann, die der angegriffene Server in seinem Cache ablegt und an Clients weitergibt. (Siehe hierzu die Ausführungen über Cache-Vergiftung in Abschnitt 11.1.)

Im Grunde muss ein DNS-Server zwei Arten von Anfragen beantworten können:

- Anfragen von (den Resolvern auf) lokalen Clients, die irgendwelche beliebigen Namen aufgelöst haben möchten.
- Anfragen von entfernten DNS-Servern, die Namen in einer Zone auf dem Server aufgelöst haben möchten.

Mit anderen Worten: Anfragen, die nicht aus Ihrem lokalen Netz kommen, sollten sich nur auf DNS-Daten beziehen, für die Ihr Server *autoritativ* ist. Umgekehrt sollten Anfragen, die Ihren Server für *rekursive* Auflösung in Anspruch nehmen, nur aus Ihrem lokalen Netz kommen. Problematisch wird es da, wo eine Anfrage von irgendwo aus dem Internet kommt und Ihren Server als *rekursiven* Server in Anspruch nehmen will.

Sie können dieses Einfallstor schließen, indem Sie Ihren DNS-Server in zwei Komponenten trennen: Die eine kümmert sich ausschließlich um rekursive Anfragen aus dem lokalen Netz, die andere ausschließlich um Anfragen beliebiger Clients nach autoritativen Daten. Dies macht es Angreifern aus dem Internet unmöglich, den Cache Ihres DNS-Servers zu »vergiften«.

Nur autoritativer DNS-Server Im einfachsten Fall können Sie BIND daran hindern, rekursive Anfragen zu bearbeiten, indem Sie die Option `recursion` in der Konfiguration entsprechend setzen:

```
options {
    recursion no;
};
```

Außerdem sollten Sie, wie in Abschnitt 8.3 beschrieben, dafür sorgen, dass nur autorisierte Rechner Zonentransfers durchführen dürfen.

Bei BIND 8 ist es ferner sinnvoll, den DNS-Server davon abzubringen, dass er automatisch versucht, IP-Adressen für die Namen auf der rechten Seite von `NS`-Records herauszufinden:

```
options {
    fetch-glue no;
};
```

BIND 9 macht das von Haus aus nicht.

Nur rekursiver DNS-Server Sie können BIND sagen, dass er nur von bestimmten Rechnern oder Netzen Anfragen annehmen soll:

```
acl "internal" { 127.0.0.1; 11.12.13/24; }
options {
    allow-query { "internal"; };
};
```

läßt nur Anfragen zu, die vom Rechner selbst (`127.0.0.1`) oder von anderen Stationen im Netz `11.12.13.0/24` ausgehen.

In neueren Versionen von BIND 8 sowie in BIND 9 gibt es außerdem die Option `allow-recursion`, die lediglich rekursive Anfragen ausschließt. Dies ist `allow-query` vorzuziehen.

Der Grund dafür ist, dass ein DNS-Server möglicherweise Anfragen von entfernten Servern bekommt, die versuchen, Namen in einer delegierten Subdomain aufzulösen. Solche relativ unverfänglichen Anfragen werden unter `allow-query` abgewiesen, aber unter `allow-recursion` bearbeitet. In jedem Fall kommt es aber nicht dazu, dass der DNS-Server zur Beantwortung dieser Anfragen selbst DNS-Anfragen stellt (was die Wurzel allen Übels ist).

Für BIND 8 ist außerdem noch die Option `use-id-pool` interessant. Diese wurde in BIND 8.2 eingeführt und sorgt dafür, dass die Transaktionsnummern, die der Server vergibt, zufällig bestimmt werden. Dies erschwert Angriffe mit »roher Gewalt«, die bei älteren DNS-Servern möglich waren, weil die Anfragenummern in Folge vergeben wurden und Angreifer raten konnten, welche Nummern ausstehende Anfragen trugen:

```
options {
    use-id-pool yes;                                    BIND 8 ab Version 8.2
};
```

BIND 9 verwendet von selbst zufällige Anfragenummern, so dass Sie nichts Besonderes unternehmen müssen.

11.4.2 Umsetzung

Im einfachsten (und aufwendigsten) Fall installieren Sie einfach zwei verschiedene Computer, auf denen jeweils eine BIND-Instanz läuft, und konfigurieren den einen als autoritativen und den anderen als rekursiven DNS-Server. Im Zeitalter der Virtualisierung klingt das nicht mehr ganz so extravagant wie früher, aber trotzdem ist diese Lösung mit dem größten Konfigurations- und Wartungsaufwand verbunden. In diesem Abschnitt stellen wir Ihnen einige weniger aufwendige Ansätze vor.

Ein BIND für alles In Bild 11.1 sehen Sie eine einfache komplette Konfiguration, die die Hinweise aus dem vorigen Abschnitt aufgreift. Wir beschränken rekursive Anfragen auf die Rechner im internen Netz, so dass externe DNS-Server nur noch autoritative Antworten bekommen können. Außerdem dürfen nur die offiziellen sekundären DNS-Server Zonentransfers durchführen.

Zwei separate Prozesse Wenn Sie sichergehen wollen, dass Ihr autoritativer und Ihr rekursiver DNS-Server nichts miteinander zu tun haben, dann können Sie auch zwei BIND-Instanzen auf demselben Rechner laufen lassen. Dabei sind nur ein paar Kleinigkeiten zu beachten:

- Die beiden BINDs brauchen separate Konfigurationsdateien und sinnvollerweise auch separate Verzeichnisse für ihre Konfiguration.

```
acl "internal" {
    127.0.0.1; 10.0.0/24;                          Rechner in unserem LAN
};

acl "secondary" {
    33.44.55.66; 99.100.101.102;                   Unsere sekundären Server
};

options {
    directory "/var/cache/bind";
    allow-recursion { "internal"; };
    use-id-pool yes;                                          Für BIND 8
};

zone "example.com" {
    type master;
    file "/etc/bind/db/example.com";
    allow-transfer { "secondary"; };
};

zone "0.0.10.in-addr.arpa" {
    type master;
    file "/etc/bind/db/10";
    allow-transfer { "secondary"; };
};

zone "." {
    type hint;
    file "/etc/bind/db.root";
};
```

Bild 11.1: Abgesicherte BIND-Konfiguration

```
acl "secondary" {
    33.44.55.66; 99.100.101.102;                    Unsere sekundären Server
};

options {
    directory "/var/cache/bind-auth";
    pid-file "/var/run/bind-auth.pid";
    listen-on { 10.0.0.1; };
    recursion no;
    fetch-glue no;                                  Für BIND 8
};

zone "example.com" {
    type master;
    file "/etc/bind/db/example.com";
    allow-transfer { "secondary"; };
};

zone "0.0.10.in-addr.arpa" {
    type master;
    file "/etc/bind/db/10";
    allow-transfer { "secondary"; };
};
```

Bild 11.2: Getrennte DNS-Server: Der autoritative Server

```
acl "internal" {
    127.0.0.1; 10.0.0/24;                           Rechner in unserem LAN
};

options {
    directory "/var/cache/bind-rec";
    pid-file "/var/run/bind-rec.pid";
    listen-on { 127.0.0.1; 10.0.0.101; };           IP-Alias?
    allow-query { "internal"; };
    use-id-pool yes;                                Für BIND 8
};

zone "." {
    type hint;
    file "/etc/bind/db.root";
};
```

Bild 11.3: Getrennte DNS-Server: Der rekursive Server

- Die beiden BINDs müssen auf separaten IP-Adressen lauschen (da immer nur ein Prozess den Port 53 zum Lauschen geöffnet haben kann). Wenn der betreffende Rechner nicht sowieso schon verschiedene physikalische »LAN-« und »Internet« Schnittstellen hat, dann können Sie zum Beispiel über IP-Aliasing eine weitere Adresse zur Verfügung stellen.

  
  Im Extremfall kann der autoritative BIND auf der »echten« IP-Adresse des Rechners lauschen und der rekursive auf der Loopback-Adresse `127.0.0.1`. Dann steht er nur dem lokalen Resolver zur Verfügung, aber möglicherweise reicht das ja.

- Wahrscheinlich brauchen Sie für die beiden BINDs auch separate Init-Skripte. Am besten kopieren Sie das BIND-Init-Skript Ihrer Distribution und passen allfällige Pfadnamen an.

Bild 11.2 zeigt die Konfigurationsdatei für den autoritativen und Bild 11.3 die für den rekursiven BIND. Dabei nehmen wir an, dass der Rechner über eine physikalische Netzwerkschnittstelle verfügt, die im `10.0.0.0/24`-Netz beheimatet ist. Für den rekursiven Server ist ein IP-Alias vergeben worden. Sie müssen auf den Clients nur darauf achten, dass in der Datei `/etc/resolv.conf` die erste `nameserver`-Zeile so aussieht:

```
nameserver 10.0.0.101                                    Der rekursive DNS-Server
```


In den `NS`-Records für die Domain steht natürlich nur der Name des *autoritativen* Servers (mit einem `A`-Record für die Adresse `10.0.0.1`).

Ein BIND (9) mit Views »Views« sind eine neue Eigenschaft von BIND 9, mit der es möglich ist, in einer einzigen BIND-Instanz einen Effekt zu erzeugen, der der aufgeteilten Konfiguration aus dem vorigen Beispiel ähnelt. Die entsprechende BIND-Konfiguration ist in Bild 11.4 gezeigt.

Hierbei sind die folgenden Punkte zu beachten:

- In einer `view`-Deklaration entscheidet eine `match-clients`-Klausel darüber, für welche Anfragen die View gilt. Dabei werden die verschiedenen Views in der Reihenfolge angeschaut, wie sie in der Konfiguration stehen; die erste passende Deklaration gewinnt.

  
  Wenn in Bild 11.4 die `authoritative`-View vor der `recursive`-View stünde, würde die `recursive`-View völlig ignoriert, da die `authoritative`-View durch ihr

  ```
  match-clients { any; };
  ```

  auf jede Anfrage passt.

- Views müssen in der Konfiguration nach (aber nicht unmittelbar nach) einer `options`-Klausel stehen.

- Views dürfen fast alles enthalten, was auch sonst in der Konfiguration stehen darf. Eine notable Ausnahme sind `acl`-Deklarationen. Optionen, die sonst in `options` stehen würden, werden in Views direkt hingeschrieben (siehe etwa die `recursion`-Option in Bild 11.4).

Die beiden Views in unserem Beispiel unterscheiden sich nicht nennenswert. In der einen ist `recursion` eingeschaltet und in der anderen nicht. Allerdings wäre es ohne weiteres möglich, zum Beispiel unterschiedliche Zonendefinitionen zu verwenden, damit in der rekursiven View alle und in der autoritativen nur die für »externe« Anfragesteller interessanten RRs (`SOA`, `NS`, `MX` und `A`-Records für Namen wie `www.example.com`, `mail.example.com` und Ähnliches zu sehen sind.

```
acl "internal" {
    127.0.0.1; 10.0.0/24;                          Rechner in unserem LAN
};

acl "secondary" {
    33.44.55.66; 99.100.101.102;                   Unsere sekundären Server
};

options {
    directory "/var/cache/bind";
};

view "recursive" {
    match-clients { "internal"; };
    recursion yes;

    zone "example.com" {
        type master;
        file "/etc/bind/db/example.com";
    };
    zone "0.0.10.in-addr.arpa" {
        type master;
        file "/etc/bind/db/10";
    };
    zone "." {
        type hint;
        file "/etc/bind/db.root";
    };
};

view "authoritative" {
    match-clients { any; };
    recursion no;

    zone "example.com" {
        type master;
        file "/etc/bind/db/example.com";
        allow-transfer { "secondary"; };
    };
    zone "0.0.10.in-addr.arpa" {
        type master;
        file "/etc/bind/db/10";
        allow-transfer { "secondary"; };
    };
    zone "." {
        type hint;
        file "/etc/bind/db.root";
    };
};
```

Bild 11.4: Getrennte DNS-Server: Konfiguration mit Views

Übungen

11.5 [!2] Probieren Sie die einfache Konfiguration aus »Ein BIND für alles« aus. (Für diese Aufgabe lohnt sich Virtualisierung – konfigurieren Sie Ihren DNS-Server mit zwei Netzwerkkarten, eine für das LAN und eine für das »Internet«, und überzeugen Sie sich, dass aus den jeweiligen Richtungen nur die erwünschten Arten von Abfragen möglich sind. Natürlich können Sie das Ganze auch mit physikalischen Rechnern aufbauen – dann sollten es aber mindestens zwei sein, oder Sie müssen `dig` mit der Option »`-b`« verwenden.)

11.6 [3] Versuchen Sie sich an der in »Zwei separate Prozesse« beschriebenen Konfiguration. Zum Experimentieren reicht es aus, wenn Sie den zweiten BIND-Prozess manuell – und nicht über ein angepasstes Init-Skript – starten. (Für das Init-Skript gibt es Sonderpunkte.)

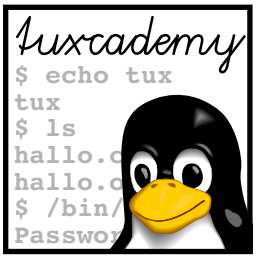
11.7 [3] Setzen Sie die in »Ein BIND (9) mit Views« erklärte Konfiguration um. Sorgen Sie dabei auch dafür, dass für die Zugriffe von außen nur die dafür wichtigen RRs zu sehen sind.

Zusammenfassung

- DNS in seiner ursprünglichen Form ist empfindlich gegenüber verschiedenen Angriffen wie DNS Spoofing oder Cache-Vergiftung.
- Die meisten dieser Probleme werden mit DNSSEC gelöst.
- BIND kann nach der Initialisierung mit den Rechten eines gewöhnlichen Benutzers laufen.
- Es ist möglich, BIND in einer `chroot`-Umgebung auszuführen. Bei BIND 9 ist das relativ einfach, bei BIND 8 deutlich aufwendiger.
- In Abwesenheit von DNSSEC bedeutet es einen großen Sicherheitsgewinn, die Funktion eines autoritativen von der eines rekursiven DNS-Servers zu trennen. Mit BIND ist das auf verschiedene Weisen möglich.

Literaturverzeichnis

Chroot-BIND HOWTO Scott Wunsch. »Chroot-BIND HOWTO«, Dezember 2001.
`http://tldp.org/HOWTO/Chroot-BIND-HOWTO.html`



A

Musterlösungen

Dieser Anhang enthält Musterlösungen für ausgewählte Aufgaben.

2.6 Die Terminal-Gerätedateien gehören normalerweise dem daran arbeitenden Benutzer und der Gruppe `tty`. Das Kommando `mesg` tut nichts anderes als das Gruppen-Schreibrecht auf der Gerätedatei des betreffenden Terminals zu setzen (»`mesg y`«) oder zu löschen (»`mesg n`«). Das Programm `wall` ist Set-GID `tty` und kann darum im Namen beliebiger Benutzer auf alle Terminal-Gerätedateien schreiben, sofern Gruppenschreibrecht existiert; `root` ist nicht an die Schreibrechte gebunden und darf natürlich immer schreiben.

2.7 `shutdown` entfernt `/etc/nologin` unmittelbar vor dem Wechsel des Runlevels oder wenn es beendet wurde, ohne tatsächlich das System herunterzufahren. Zur Sicherheit sorgen die gängigen Distributionen auch beim Neustart dafür, dass eine etwa noch herumliegende `/etc/nologin`-Datei gelöscht wird.

3.1 Wenn Sie 20 Oktette für einen TCP-Kopf und 20 Oktette für einen IP-Kopf addieren, kommen Sie gerade auf 1500 Oktette – die maximale Größe für die »Nutzlast« eines Ethernet-Frames und damit MTU (engl. *maximum transmission unit*) in einem Ethernet-LAN.

Im Normalfall strebt man an, dass die maximale Fenstergröße ein ganzzahliges Vielfaches der MSS ist. Wenn das nicht so ist, dann kommt es bei einem fast vollen Fenster zum Versand von fragmentierten Daten und somit zu Ineffizienzen.

3.2 Etwas wie

```
# tcpdump -w log.pcap not tcp port ssh
```

ignoriert auch andere möglicherweise gerade aktive SSH-Verbindungen. Wenn Sie sichergehen wollen, dass nur Ihre ausgeschlossen wird, dann müssen Sie etwas wie

```
# tcpdump -w log.pcap not \( tcp port ssh and host red and www \)
```

angeben (vorausgesetzt, dass die Auflösung für `red` und `www` klappt).

3.3 Das erste Paket einer TCP-Verbindung zeichnet sich dadurch aus, dass das `SYN`-Flag gesetzt ist, aber nicht das `ACK`-Flag. Kombinieren Sie das mit einem Ausdruck für die Absenderadresse, der auf Ihren Rechner passt:

```
# tcpdump -w tcp.pcap src host red and tcp[tcpflags] = tcp-syn
```

3.5 Versuchen Sie es mal mit

```
not (ip.addr == 192.168.61.254)
```

Dies funktioniert, weil die Bedingung

```
ip.addr == 192.168.61.254
```

umgangssprachlich ausgedrückt bedeutet: »Entweder die Absender- oder die Empfänger-Adresse des Pakets oder beide sind `192.168.61.254`«. Das logische Gegenteil davon ist »Weder die Absender- noch die Empfänger-Adresse des Pakets ist `192.168.61.254`«.

5.1 In kleinen Netzen ist eine statische Tabelle leicht aufgestellt und mit vertretbarem Aufwand zu warten, während ein DNS-Server nur hierfür meist überflüssig ist. In der statischen Tabelle werden Daten schnell gefunden. Auf der anderen Seite erlaubt ein DNS-Server die Verwaltung verschiedener Daten, für die keine statische Tabelle vorgesehen ist. Ferner kann ein lokaler DNS-Server im Gegensatz zu einer statischen Tabelle auch Zugriffe auf das weltweite DNS per Caching beschleunigen.

5.2 Namen in `/etc/hosts` stehen auch dann zur Verfügung, wenn das DNS aus irgendwelchen Gründen nicht zu erreichen ist. Es ist also sinnvoll, zumindest den Namen des lokalen Rechners und der wichtigsten Server im selben LAN in die Datei `/etc/hosts` einzutragen, um gegen allfällige DNS-Ausfälle gefeit zu sein.

5.3 `ag` ist die ccTLD für Antigua und Barbuda, einen Inselstaat in der östlichen Karibik.

Es zeigt sich, dass die Betreiber der `ag`-TLD nicht darauf bestehen, Domains darin nur an Personen oder Organisationen zu vergeben, die auch dort beheimatet sind; diverse deutsche Aktiengesellschaften unterhalten zum Beispiel `ag`-Domains. Der 5. Zivilsenat des OLG Hamburg entschied im Juli 2004 in zweiter Instanz, dass nur deutsche Aktiengesellschaften `ag`-Domains registrieren dürfen, und zwar nur Domains, die so heißen wie die Aktiengesellschaft selbst. (Der Rest der Welt – außerhalb der Reichweite des langen Arms der deutschen Jurisprudenz – darf vermutlich machen, was er will.)

5.4 Auf der einen Seite ist es natürlich verlockend für die (hypothetische) Schulz AG, sich die (hypothetischen) Namen `schulz.de`, `schulz.com`, `schulz.eu`, `schulz.ag` usw. zu sichern (getreu dem Pokémon-Motto »Schnapp sie Dir alle!«). Das schützt das Unternehmen immerhin vor Ärgernissen wie dass die konkurrierende (hypothetische) Huber GmbH unter dem Namen `schulz.com` eine Webseite etabliert, auf der *ihre* Produkte oder Leistungen beworben werden, und der Schulz AG so Kunden abspenstig macht – statt einen langwierigen und teuren Prozess um Markenverletzungen oder unlauteren Wettbewerb zu führen, tut man sich leichter, präventiv alle in Frage kommenden Domains selbst zu registrieren.

Auf der anderen Seite untergräbt das natürlich die Idee der Top-Level-Domains. Wenn jeder sowieso seinen Namen in allen Top-Level-Domains registriert, dann bräuchte man auch gar keine Top-Level-Domains mehr. Ferner würde man aus dem Bauch heraus zum Beispiel die Top-Level-Domain `eu` mit Organisationen auf der »europäischen Ebene« in Verbindung bringen und nicht (so wie es aktuell gehandhabt wird) mit irgendwelchen Leuten, die Lust auf eine Adresse mit `eu` am Ende hatten. Hier konkurriert das abstrakte Interesse an einem »sauberen« DNS mit dem konkreten Interesse der Registrars und Provider, möglichst viele `eu`-Domains unter die Leute zu bringen, indem man seinen Kunden suggeriert, sie bräuchten zu ihrer `de`-, `at`- oder `fr`-Adresse unbedingt auch noch die gleichnamige `eu`-Domain.

5.6 Der Registrant ist im wesentlichen die Person, die sich den Namen der Domain ausgesucht hat. Der Admin-C (*administrative contact*) ist die Person, die bestimmen darf, was mit der Domain passiert (etwa ob sie zu einem anderen Registrar umgezogen wird) und die Domaindaten ändern darf. Der Tech-C (*technical contact*) ist die Person, die die DNS-Server für die Domain verwaltet. Wenn Sie eine »Billigdomain« über einen großen Webspace-Provider registrieren, dann sind Sie in der Regel Registrant und Admin-C, während jemand beim Provider als Tech-C fungiert. – Sie sollten bei einer Domainregistrierung *unbedingt* darauf achten, als Admin-C eingetragen werden, da Sie sich sonst vom andauernden guten Willen Ihres Providers abhängig machen. Zum Glück ist das heutzutage selten ein Problem; früher konnte es durchaus sein, dass der Provider sich selbst als Admin-C eintragen ließ, so dass Sie zum Beispiel nicht ohne weiteres mitsamt Ihrer Domain zu einem anderen Provider umziehen konnten, wenn Sie Ärger mit dem bisherigen hatten.

5.7 Die Idee hinter der sTLD `mobi` ist, dass Anbieter zum Beispiel auf Servern mit `.mobi`-URL Seiten ablegen können, die darauf optimiert sind, mit mobilen Geräten angeschaut zu werden. Mobile Geräte (jedenfalls, als `mobi` neu war) hatten kleine Bildschirme und schwachbrüstige Browser und in der Regel auch sehr langsame Netzanbindungen. Heute – im Zeitalter von UMTS – ist Netzbandbreite nicht mehr unbedingt ein großes Problem, und die Bildschirmauflösungen gängiger mobiler Geräte, etwa Smartphones oder Netbooks, unterscheiden sich nicht nennenswert zumindest von dem, was Anfang der 2000er Jahre auf Desktop-PCs völlig normal war. Auch die Browser sind heutzutage ihren »stationären« Cousins sehr ähnlich geworden. Es gibt also keinen zwingenden Grund, warum man spezielle Webserver mit »mobilen« Inhalten anbieten sollte, wenn es ohne großen Aufwand möglich ist, die »normalen« Inhalte auch auf mobilen Geräten nutzbar zu machen. Dieselben Ziele, die `mobi` über eine parallele Serverlandschaft erreicht, lassen sich zum Beispiel auch durch zusätzliche Rechnernamen (etwa `m.example.com`), durch HTTP-Inhaltsaushandlung, geschickte Style-Sheets oder andere Anpassungsmethoden erreichen. (Im Extremfall wird die sTLD `mobi` vor allem zu einem Vehikel für Domain-Anbieter, um mehr Namen verkaufen zu können.)

Tatsächlich kritisieren Leute wie Tim Berners-Lee, der Erfinder des World-Wide Web, dass etwas wie `mobi` die Geräteunabhängigkeit des WWW untergräbt. Zum Beispiel könnte man eine Reiseplanung mit derselben Berechtigung auf dem PC im Büro anschauen wollen wie auf dem Smartphone am Flughafen, und es wäre unsinnig, dafür zwei verschiedene URLs verwenden zu müssen.

5.8 Im Fall (a): `www.sub1.example.com`, dann gegebenenfalls `www.sub2.example.com` und `www.example.com`. In den Fällen (b) und (c) wird jeweils nur der angegebene Name gesucht.

5.10 `domainname` dient zum Anzeigen oder Einstellen des NIS-Domainnamens eines Rechners, `dnsdomainname` zum Abfragen des DNS-Domainnamens. Die beiden Domainnamen haben technisch absolut nichts miteinander zu tun. Mit `dnsdomainname` können Sie den DNS-Domainnamen eines Rechners nicht einstellen, er ergibt sich aus dem Namen, den das DNS für den Rechner liefert, indem Sie die linkeste Komponente weglassen.

6.4 Der *caching-only* Server sollte nur bei der ersten Anfrage nach einem Namen tatsächlich andere Server befragen. Bei der zweiten Anfrage sollte die Antwort (nicht autoritativ) aus dem Cache kommen, so dass in der Ausgabe des Paketsniffers gar keine Pakete erscheinen.

7.1 Richtig wäre zum Beispiel

```
$TTL 1d3h27m                                    aktueller BIND
$TTL 97227                                      älterer BIND
```

7.3 Eine Möglichkeit besteht darin, alle sekundären DNS-Server anzuhalten, ihre Informationen über die falsche Seriennummer zu extirpieren und sie die Zonendateien neu laden zu lassen. Das ist nicht wirklich schön und bedarf auch einigen Aufwands, falls Sie nicht alle Ihre sekundären DNS-Server selber kontrollieren.

Eine elegantere Methode nutzt etwas Mathematik aus. DNS-Seriennummern haben Werte zwischen 0 und 4.294.967.295, und die Methode, mit der sie verglichen werden, führt dazu, dass – egal, welchen Wert eine Seriennummer hat – knapp die Hälfte der verfügbaren Seriennummern (2.147.483.647 oder $2^{31} - 1$ Stück) »kleiner« sind als eine gegebene Seriennummer und knapp die andere Hälfte »größer«. Angenommen, die Seriennummer ist 17. In diesem Fall gelten die Seriennummern 18 bis 2.147.483.664 (oder $17+2^{31}-1$) als größer und 2.147.483.666 bis 16 als kleiner. (Die Seriennummer 2.147.483.665 – oder $17 + 2^{31}$ – sitzt genau auf dem Zaun; ob sie »kleiner« oder »größer« ist, ist implementierungsabhängig. Vermeiden Sie sie.) Beachten Sie auch, dass die 16 sich daraus ergibt, dass die Seriennummern sich bei 2^{32} »herumwickeln«.

Nehmen wir jetzt an, die Seriennummer 17 ist Ihnen zu hoch und Sie hatten eigentlich 2 gemeint. Der Trick besteht darin, das »Herumwickeln« auszunutzen, und funktioniert so: Addieren Sie $2^{31} - 1$ zu Ihrer Seriennummer. (Wenn das Ergebnis größer als $2^{32} - 1$, also 4.294.967.295, ist, müsen Sie es »herumwickeln«, indem Sie 2^{32} subtrahieren.) Tragen Sie diese Seriennummer – in unserem Beispiel $17+2^{31}-1 = 2.147.483.664$ – in Ihre Zonendatei ein und warten Sie darauf, dass alle sekundären DNS-Server sich aktualisieren. Setzen Sie dann die Seriennummer, die Sie *eigentlich* haben wollten, in die Zonendatei ein. Wie oben angedeutet gelten aus der Sicht der »künstlichen« neuen Seriennummer $17 + 2^{31} - 1$ alle Seriennummern von $17 + 2^{31}$ bis $17 + 2^{31} - 1 + (2^{31} - 1)$, genauer gesagt (mit »Herumwickeln«) 15, als »größer«, so dass die sekundären DNS-Server sich beim nächsten Aktualisieren die Zonendatei mit der Seriennummer 2 holen werden – denn 2 ist als DNS-Seriennummer betrachtet »größer« als 2.147.483.664.

7.4 Diese Strategie lohnt sich am meisten, wenn der fragliche Dienst von mehreren unabhängigen Servern in praktisch identischer Form erbracht werden kann. Das gilt für viele Arten von Web-Servern, aber zum Beispiel auch replizierte LDAP-Server. Schwierig wird es da, wo es nötig wird, dass immer derselbe von mehreren Servern eine Anfrage beantwortet, etwa wenn auf dem Server Zustand gehalten wird.

7.5 Wenn sich die Adresse ändert, unter der der Dienst erbracht wird, müssen Sie nur ein `A`-Record ändern und alle Namen, die über `CNAME`-Records auf dieses `A`-Record verweisen, liefern automatisch die neue Adresse. Ansonsten müßten Sie alle betroffenen `A`-Records finden und anpassen. Der Nachteil ist, dass theoretisch für die Auflösung des `CNAME`-Namens eine weitere Anfrage nach dessen `A`-Record nötig wäre – ein cleverer DNS-Server wie BIND ahnt das jedoch voraus und liefert das betreffende `A`-Record gleich mit.

7.6 Nein – `CNAME`-Records beziehen sich immer nur auf »Blätter« des DNS-Baums. Namen, denen andere RRs zugeordnet sind (hier vermutlich ein `SOA`- und ein paar `NS`-Records), können nicht außerdem `CNAME`-Records haben. (Lesen Sie RFC 2672 über `DNAME`-Records, die BIND 9 implementiert.)

7.7 Das ist absolut erlaubt und wird für gewisse Anwendungsfälle auch dringend gebraucht, etwa wenn es um Delegation geht (betrachten Sie zum Beispiel Abschnitt 9.3).

7.8 Es ist durchaus gängig, für eine Domain einen Haupt- und einen oder mehrere Ersatz-Mailserver zu konfigurieren. Wenn der Haupt-Mailserver aus irgendwelchen Gründen nicht erreichbar ist, dann können die Ersatz-Mailserver eingehende Mail annehmen und bei Gelegenheit an den Hauptserver weiterreichen. Die MX-Records dafür könnten ungefähr so aussehen:

Hauptserver

```
example.com.  IN  MX  10 mail.example.com.
              IN  MX  20 mail-backup.example.com.
```

In der Praxis ist das oft mehr Mühe, als es wert ist. Wenn Ihr Mailserver nicht erreichbar ist, dann versuchen standardkonforme entfernte Mailserver bis zu fünf Tage lang, ausstehende Nachrichten zuzustellen, und das sollte Ihnen genug Zeit geben, Ihren Mailserver wieder flott zu machen. Normalerweise liegen ja die Benutzerpostfächer auf dem Mailserver, so dass, selbst wenn ein Ersatz-Mailserver konfiguriert ist, Ihre Benutzer nicht an ihre Mail kommen würden, weil sie ihre Postfächer nicht erreichen können. Außerdem ist es wichtig, dass ein Ersatz-Mailserver über dieselbe Konfiguration wie der Haupt-Mailserver verfügt, was Spamfilterung angeht, da Spammer ihre unerwünschten Nachrichten gerne absichtlich an den Ersatz-Mailserver schicken, selbst wenn der Haupt-Server zu erreichen ist – in der Annahme, dass jener bei der Prüfung weniger genau hinschaut und dieser die Nachrichten vom Ersatz-Mailserver ohnehin kritiklos annimmt. Je nachdem, wo Ihr Ersatz-Mailserver steht und wer ihn wie administriert, sind aber wichtige Informationen wie etwa Ihre Benutzerdatenbank dort überhaupt nicht zugänglich.

7.9 Das ist interessant, wenn Sie eine große Firma oder Universität oder ein Internet-Provider sind und riesige Mengen von Mail umsetzen. Da Clients sich mehr oder weniger zufällig eines der MX-Records aussuchen, können Sie so die eingehende Mail auf mehrere Server verteilen. Es ist dann auch nicht schlimm, wenn mal einer Ihrer Mailserver ausfällt, weil dann die anderen Mailserver die Last mit übernehmen. Sie müssen allerdings sicherstellen, dass alle Ihre Mailserver Zugriff auf die Benutzerpostfächer haben – etwa indem Sie einen anderen Rechner (oder mehrere) als IMAP- und/oder POP3-Server konfigurieren und die Mail zum Beispiel mit LMTP vom Mailserver zum IMAP-Server schicken, der die Benutzerpostfächer vorhält.

8.1 Informationen über die DNS-Server bekommen Sie zum Beispiel mit `dig` oder `host`, indem Sie sich die SOA- und NS-Records anschauen:

Primärer Server

```
$ host -t soa heise.de
heise.de has SOA record ns.heise.de.
postmaster.ns.heise.de. 2011042901 10800 3600 604800 3600
$ host -t ns heise.de
heise.de name server ns2.pop-hannover.net.
heise.de name server ns.pop-hannover.de.
heise.de name server ns.heise.de.
heise.de name server ns.plusline.de.
heise.de name server ns.s.plusline.de.
```

Ortsangaben können Sie zum Beispiel über `http://www.geobytes.com/IpLocator.htm` herausfinden. Zahlreiche andere interessante Informationen bietet WHOIS auf der Basis der IP-Adresse:

```
$ whois $(host -t a ns.heise.de | cut -d' ' -f4)
```

8.2 Die Root-Zone (».«) hat zwölf, genau wie `com.` und `net.`. (`org.` gibt sich mit fünf zufrieden und auch `de.` kommt anscheinend mit vieren aus.)

8.7 `host` führt mit der Option `-l` einen Zonentransfer durch und listet anschließend die `NS`-, `PTR`- und `A`-Records in der Zone auf. Zusammen mit der Option `-a` werden alle RRs aufgelistet. Außerdem können Sie natürlich

```
$ host -t AXFR example.com. 10.0.0.1
```

sagen, um ähnlich wie bei `dig` explizit einen Zonentransfer auszulösen (die Ausgabe entspricht dann der von »`host -la`«).

8.9 Ein Angreifer könnte NOTIFY-Nachrichten fälschen und damit einen Denial-of-Service-Angriff auslösen, bei dem die sekundären Server massenhaft Zonentransfers anfordern und damit den primären Server und/oder die Verbindung zwischen den beiden überlasten.

9.1 `$ORIGIN` spielt bei DNS-Zonendateien die Rolle, die das Kommando `cd` (oder, für Programmierer, der Systemaufruf `chdir`(2)) in Unix bzw. Linux ausfüllt. (Es kann sein, dass das die Analogie etwas sehr strapaziert, aber vielleicht hilft es ja trotzdem irgendwem.)

9.3 Versuchen Sie etwas wie

```
$GENERATE 1-100 pc$.example.com A 10.0.1.$
```

`$GENERATE` funktioniert nur für eine begrenzte Auswahl von RR-Typen, namentlich `A`, `AAAA`, `CNAME`, `DNAME`, `NS` und `PTR`. Außerdem ist es BIND-spezifisch, das heißt, andere DNS-Server verstehen es nicht unbedingt, selbst wenn sie ansonsten RFC-konforme Zonendateien verarbeiten können.

9.4 Dies ist eine etwas vereinfachte Form des gezeigten RFC-2317-Beispiels. Sie können etwas verwenden wie

```
0-127.1.168.192.in-addr.arpa.    IN NS ns1.example.com.
                                 IN NS ns2.example.com.
128-255.1.168.192.in-addr.arpa.  IN NS ns1.example.com.
                                 IN NS ns2.example.com.
$ORIGIN 1.168.192.in-addr.arpa.
$GENERATE 1-126   $ CNAME $.0-127.1.168.192.in-addr.arpa.
$GENERATE 129-254 $ CNAME $.128-255.1.168.192.in-addr.arpa.
$ORIGIN 0-127.1.168.192.in-addr.arpa.
1   IN PTR foo.marketing.example.com.
◁◁◁◁◁
$ORIGIN 128-255.1.168.192.in-addr.arpa.
129 IN PTR bla.entwicklung.example.com.
◁◁◁◁◁
```

Die beiden Zonen `0-127.1.168.192.in-addr.arpa.` und `128-255.1.168.192.in-addr.arpa.` können Sie natürlich an entfernte DNS-Server delegieren.

10.1 Etwas wie

```
10.11.12/24; !10.11/16; any;
```

sollte das Gewünschte erledigen.

10.2 Sie könnten einen solchen Server beispielsweise in einer Firewall-Infrastruktur als rekursiven DNS-Server in der DMZ verwenden und damit Anfragen an die DNS-Server Ihres Providers weiterleiten. Damit ist es möglich, den ausgehenden Paketfilter so zu konfigurieren, dass DNS-Daten nur zwischen Ihrem DNS-Server und den DNS-Servern Ihres Providers ausgetauscht werden; andere DNS-Anfragen von außen können Sie abweisen. (Das setzt natürlich voraus, dass Sie in Ihrer DMZ keinen autoritativen DNS-Server für das Internet anbieten müssen, etwa weil Sie einen *hidden primary* installiert haben, der die DNS-Server Ihres Providers als offizielle DNS-Server für Ihre Domains bekannt gibt.

11.1 Im einfachsten Fall können Sie die tatsächliche Versionsnummer unterdrücken (oder ein bisschen flunkern), indem Sie die `version`-Option verwenden:

```
options {
    ◁◁◁◁◁
    version "11.3.2";                          Zurück in die Zukunft …
    ◁◁◁◁◁
};
```

Wenn Sie überhaupt keine Antwort liefern oder diese Information nur bestimmten Fragestellern zugänglich machen wollen, müssen Sie mehr arbeiten. Definieren Sie zuerst eine CHAOSNET-Zone mit der gewünschten Information:

```
$TTL 1d
@ CH SOA ns.example.com. hostmaster.example.com. (
         2011050201 86400 3600 604800 3600 )
  CH NS  ns.example.com.

version.bind. CH TXT "BIND 11.3.2"
```

Diese Zone können Sie bei BIND 8 wie üblich einbinden:

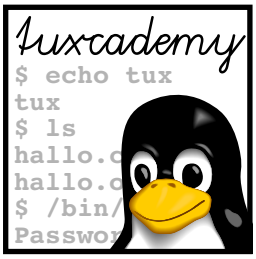
```
zone "bind" chaos {
    type master;
    file "/etc/bind/db.bind";                  Die Datei von eben
    allow-query { localnets; };                Lokale Rechner
};
```

Bei BIND 9 ist das leider nicht ganz so einfach, da Zonen standardmäßig in einer View der Klasse »Internet« angelegt werden und die Zonen auch zu dieser Klasse gehören müssen. Sie müssen also eine CHAOSNET-View anlegen und die Versions-Zone darin platzieren. Daraus folgt unangenehmerweise aber, dass Sie außerdem eine Internet-View anlegen müssen, damit Sie die anderen Zonendefinitionen dort unterbringen können. Also etwas wie

```
view internet in {
    zone "example.com" {
        ◁◁◁◁◁
    };
};
view chaosnet chaos {
    zone "bind" {
        ◁◁◁◁◁                                  Inhalt siehe oben
    };
};
```

Ob das Ganze letzten Endes etwas bringt, ist eine andere Frage. Natürlich sollten Sie immer den aktuellsten verfügbaren BIND verwenden und möglichst sicher

konfigurieren. Angreifer werden sich sicherlich nicht davon abhalten lassen, Ihrem BIND auf den Zahn zu fühlen, nur weil er keine Versionsnummer veröffentlicht.



B

X.509-Crashkurs

Dieser Anhang gibt einen schnellen Überblick über X.509 und die Generierung von Zertifikaten. Wenn Sie genau wissen wollen, wie das alles funktioniert, lesen Sie die Linup-Front-Schulungsunterlagen *Apache und SSL* (APW2) oder *Linux als Web- und FTP-Server* (WEBF).

B.1 Einleitung: Kryptografie, Zertifikate und X.509

symmetrische Kryptografie

Die verschlüsselte Übertragung von Daten ist grundsätzlich ein gelöstes Problem: Symmetrische Kryptoverfahren wie AES machen es möglich, große Datenmengen effizient und für alle praktischen Zwecke unknackbar zu übertragen. Das Problem bei diesen Verfahren ist lediglich der Schlüsselaustausch: Da beide Kommunikationspartner über denselben geheimen Schlüssel verfügen müssen, müssen sie sich irgendwie vertraulich auf einen Schlüssel einigen können.

asymmetrische Kryptografie

Dieses Dilemma löst man über asymmetrische Kryptografie. Bei asymmetrischen Kryptoverfahren wie RSA hat jeder Teilnehmer zwei Schlüssel, einen privaten (geheimen) und einen öffentlichen. Der öffentliche Schlüssel kann allgemein bekannt gemacht werden, solange der private Schlüssel vertraulich bleibt. Dadurch wird es möglich, zwei verschiedene Probleme zu adressieren:

Verschlüsselung Alice[1] möchte eine vertrauliche Nachricht an Bob schicken. Sie verschafft sich Bobs öffentlichen Schlüssel und verschlüsselt damit die Nachricht. Bob verwendet seinen privaten Schlüssel, um die Nachricht wieder lesbar zu machen.

Digitale Signatur Alice möchte kundtun, dass sie ein bestimmtes Dokument verfasst hat. Sie »signiert« (verschlüsselt) das Dokument mit ihrem privaten Schlüssel. Bob (oder wer auch immer sonst über Alices öffentlichen Schlüssel verfügt) kann mit Alices öffentlichem Schlüssel verifizieren, dass Alice (und nicht sonst jemand) das Dokument signiert hat und dass der Inhalt authentisch ist, also dem entspricht, was Alice geschrieben hat.

Probleme mit asymmetrischer Kryptografie

Ein praktisches Problem ist, dass man mit asymmetrischen Kryptoverfahren nur relativ kleine Datenmengen verschlüsseln kann und dass sie relativ ineffizient sind – symmetrische Kryptoverfahren beruhen im Wesentlichen auf Bitoperationen, die sich sogar in Hardware realisieren lassen, während asymmetrischen Kryptoverfahren aufwendige mathematische Operationen zugrunde liegen. (RSA zum Beispiel rechnet mit sehr langen Zahlen herum.) In der Praxis verwendet man

[1] In der kryptografischen Literatur sind die beiden Kommunikationspartner traditionell immer »Alice« und »Bob«.

asymmetrische Kryptografie deshalb zusammen mit anderen kryptografischen Verfahren:

Sitzungsschlüssel **Verschlüsselung** Alice wählt einen zufälligen Schlüssel (den »Sitzungsschlüssel« oder *session key*) für ein geeignetes symmetrisches Kryptoverfahren (etwa AES). Sie verschlüsselt diesen Schlüssel mit Bobs öffentlichem Schlüssel und schickt das Resultat an Bob. Bob entschlüsselt den AES-Schlüssel mit seinem privaten Schlüssel. Anschließend können Alice und Bob vertraulich kommunizieren, indem sie ihre Nachrichten mit AES verschlüsseln.

Prüfsumme **Digitale Signatur** Alice bildet eine »kryptografische Prüfsumme« über das Dokument, also eine unumkehrbare Funktion, die von jedem Bit der Eingabe abhängt (Stichwort: MD5, SHA-1 und Ähnliches). Anschließend signiert sie diese Prüfsumme mit ihrem privaten Schlüssel. Zur Prüfung der Signatur bildet Bob die kryptografische Prüfsumme über das Dokument. Wenn die Entschlüsselung der Signatur mit Alices öffentlichem Schlüssel (den Bob ja hat) dasselbe Ergebnis liefert, ist die Signatur gültig, und das Dokument ist authentisch.

Auf diese Weise wird das Problem der Schlüsselverteilung gelöst, aber an seine Stelle tritt ein neues Problem: Angenommen, Bob findet irgendwo im Netz Alices öffentlichen Schlüssel (oder einen öffentlichen Schlüssel, der behauptet, zu Alice zu gehören). Wie kann Bob sicher sein, dass dieser Schlüssel *wirklich* Alices Schlüssel ist?

Eine mögliche Lösung für dieses Problem besteht darin, den Zusammenhang zwischen Alice und Alices Schlüssel von einer vertrauenswürdigen[2] Instanz bestätigen zu lassen. Diese Instanz heißt »Zertifizierungsstelle« (*certificate authority* oder kurz »CA«), und die Bestätigung des Zusammenhangs nennt man »Zertifikat«. X.509 ist ein Standard dafür, wie solche Zertifikate aussehen. Man spricht in diesen Zusammenhang auch von einer *public-key infrastructure* oder PKI.

Zertifizierungsstelle
Zertifikat
PKI

Ein Zertifikat besteht im Wesentlichen aus vier Komponenten:

- Einem Namen für die Person (oder den Server), von dem die Rede ist. X.509 verwendet dafür sogenannte *distinguished names* der Form

  ```
  cn=Hugo Schulz,o=Beispiel GmbH,l=Musterdorf,c=DE          Person
  cn=www.example.com,o=Beispiel GmbH,l=Musterdorf,c=DE      Server
  ```

- Einem öffentlichen Schlüssel, der zu der benannten Person (oder dem benannten Server) gehört. Genau diesen Zusammenhang soll das Zertifikat bestätigen.
- Eine digitale Signatur für das Zertifikat. Diese sichert dessen Authentizität.
- Einen Namen (DN) für die Zertifizierungsstelle.

Sie können die Authentizität eines solchen Zertifikats prüfen, indem Sie sich den öffentlichen Schlüssel der Zertifizierungsstelle besorgen (der dazugehörige Name steht ja im Zertifikat) und damit die Signatur nachrechnen. Diesen öffentlichen Schlüssel bekommen Sie natürlich auch in Form eines X.509-Zertifikats.

Wenn Sie clever sind, dann fragen Sie sich an dieser Stelle, wer wohl das Zertifikat der Zertifizierungsstelle signiert haben mag, um zu dokumentieren, dass *dieses* echt ist. Die Antwort darauf lautet: Das macht die Zertifizierungsstelle selber. Bevor Ihnen das komisch vorkommt (was im ersten Moment absolut entschuldbar wäre), sollten Sie sich überlegen, dass Sie in diesem Geschäft *irgendwem* vertrauen müssen. Da Sie der Zertfizierungsstelle glauben, dass sie zum Beispiel Alices Zertifikat zuverlässig signieren

[2] »Vertrauenswürdig« heißt in einem Kontext wie diesem gemäß Peter Gutmann dasselbe wie »man verläßt sich drauf, weil man ohnehin keine andere Wahl hat.«

kann, können Sie ihr auch glauben, dass sie ihr eigenes Zertifikat zuverlässig signieren kann – das ist kein großer Schritt. Man spricht bei einem solchen von der Zertifizierungsstelle selbst signierten Zertifikat auch von einem »Wurzelzertifikat« (*root certificate*). Wurzelzertifikat

Wobei natürlich die Frage bleibt, woher Sie wissen, dass das Wurzelzertifikat *wirklich* echt ist und Ihnen nicht – komplett mit gültiger Signatur – von einem geschickten Angreifer untergeschoben wurde. Sie können dieser Sache natürlich nachgehen, indem Sie sich bei der Zertifizierungsstelle vergewissern – oder (was wahrscheinlicher ist) Sie vertrauen blind dem Hersteller Ihres Browsers, der Ihnen hilfreicherweise (?) ein paar Dutzend Wurzelzertifikate von verschiedenen Zertifizierungsstellen fest eingebaut und (hoffentlich …) vorher seine Hausaufgaben ordentlich gemacht hat.

Grundsätzlich gibt es auch die Möglichkeit, eine Hierarchie von Zertifizierungsstellen zu bilden. Das heißt, ein Zertifikat für eine Person oder einen Server kann mit einem Zertifikat signiert sein, das nicht direkt das Wurzelzertifikat einer Zertifizierungsstelle ist, sondern wiederum mit einem anderen Zertifikat signiert wurde. Diese »Kette« von Zertifikaten läßt sich weiterverfolgen, bis irgendwann ein selbstsigniertes Zertifikat erreicht ist. Hierarchie

Um einen Server über X.509 authentisieren zu können, brauchen Sie zumindest ein Zertifikat für diesen Server. Wenn es sich dabei um einen Web-Server handelt, werden Sie kaum anders können, als sich dieses Zertifikat von einer kommerziellen Zertifizierungsstelle wie VeriSign ausstellen zu lassen, damit die gängigen Browser es ohne weitere Mühe verifizieren können. Server-Zertifikat

CAcert (`http://www.cacert.org/`) ist eine Organisation, die auf nichtkommerzieller Basis (unter anderem) Zertifikate für Server ausstellt. CAcert arbeitet im Moment an einer Akkreditierung durch die wesentlichen Browser-Hersteller; dieser Prozess ist aktuell (Mai 2011) aber noch nicht abgeschlossen. CAcert

Wenn Sie Ihre Zertifikate nur für »interne« Zwecke brauchen – etwa für eine VPN-Infrastruktur oder einen Web-Server im Intranet –, gibt es keinen vernünftigen Grund, VeriSign und Konsorten Geld in den Rachen zu werfen. Sie können ohne weiteres selbst als Zertifizierungsstelle für Ihre eigenen Zertifikate agieren und so nicht nur jede Menge Kosten sparen, sondern auch eine wesentlich sicherere Infrastruktur aufbauen (da Sie sich selbst deutlich mehr vertrauen dürften als den Windhunden vom kommerziellen Zertifizierungsmarkt). Eigene Zertifizierungsstelle

Im Rest dieses Kapitels erklären wir, wie Sie X.509-Zertifikate auf der Basis von OpenSSL (`http://www.openssl.org/`) verwalten können. Es gibt alternative frei verfügbare Implementierungen von X.509 sowie bequemere Oberflächen zur Verwaltung von Zertifizierungsstellen, aber es handelt sich auch nicht um Hexenwerk.

B.2 Eine Zertifizierungsstelle generieren

Um zur Zertifizierungsstelle zu werden, müssen Sie ein Schlüsselpaar (privater und öffentlicher Schlüssel) für die Zertifizierungsstelle erzeugen und damit ein selbstsigniertes Wurzelzertifikat generieren. Im Idealfall tun Sie das aus Sicherheitsgründen auf einem Rechner, den Sie für nichts Anderes brauchen – ein Laptop-Computer, den Sie sicher wegschließen, wenn Sie nicht gerade ein Zertifikat ausstellen müssen, ist am besten. Schlüsselpaar Sicherheit

Da alles Nötige auf der Kommandozeile stattfinden kann, reicht als Rechner für die Zertifizierungsstelle irgendeine alte Möhre, die sonst keiner mehr benutzen möchte, dicke aus. Machen Sie für den Fall des Falles Sicherheitskopien, die Sie mindestens genauso sicher aufbewahren wie den Rechner.

Manche Linux-Distributionen – etwa der »SUSE Linux Enterprise Server« von Novell – bieten hilfreicherweise an, im Rahmen der Installation Ihres Web-Servers, LDAP-Servers, … auch gleich eine Zertifizierungsstelle mitzugenerieren. Sowas lehnen Sie natürlich dankend ab.

Verwaltungsinformationen

Außerdem müssen Sie als Zertifizierungsstelle einige Verwaltungsinformationen speichern, etwa eine Liste der Zertifikate, die Sie ausgestellt haben. Im einfachsten Fall legen Sie für die Zertifizierungsstelle einen eigenen Benutzer an, in dessen Heimatverzeichnis Sie die benötigte Dateistruktur etablieren:

```
# useradd -m ca
# /bin/su - ca
$ mkdir private certs
$ chmod 700 private certs
$ echo 01 >serial
$ touch index.txt
```

Die Datei `serial` enthält die »Seriennummer« des nächsten zu erzeugenden Zertifikats und in `index.txt` steht die Zertifikatsliste. Im Verzeichnis `private` legen wir den privaten Schlüssel der Zertifizierungsstelle ab, und in `certs` landen die von der Zertifizierungsstelle ausgestellten Zertifikate.

Für die spätere Arbeit ist es am günstigsten, eine Konfigurationsdatei anzulegen, in der die wichtigsten Parameter für die verschiedenen OpenSSL-Werkzeuge stehen. Dies vermeidet Verwirrung und Fehler durch Vergesslichkeit und macht die Vorgänge nachvollziehbar. Eine mögliche Konfigurationsdatei sehen Sie in Bild B.1.

Wurzelzertifikat erzeugen

Wenn Sie die Konfigurationsdatei aus Bild B.1 in `/home/ca/openssl-ca.cnf` abgelegt haben, können Sie das Wurzelzertifikat für die Zertifizierungsstelle wie folgt erzeugen:

```
$ export OPENSSL_CONF=~/openssl-ca.cnf
$ openssl req -x509 -days 1825 -newkey rsa:2048 -out ca-cert.pem
Generating a 2048 bit RSA private key
.............................................+++
.....................+++
writing new private key to '/home/ca/private/ca-key.pem'
Enter PEM pass phrase:geheim
Verifying - Enter PEM pass phrase:geheim
-----
$ _
```

Die Passphrase, die Sie hier festlegen, müssen Sie jedesmal wieder eingeben, wenn Sie mit dieser Zertifizierungsstelle ein neues Zertifikat für einen Benutzer oder Server ausstellen wollen.

Es versteht sich von selbst, dass Sie im wirklichen Leben eine deutlich komplexere Passphrase verwenden wollen als in unserem Beispiel. Von der Sicherheit des privaten Schlüssels Ihres Wurzelzertifikats hängt die Sicherheit Ihrer kompletten Zertifikatsinfrastruktur ab!

Mit der `-days`-Option geben Sie an, wie lang das Wurzelzertifikat gilt. Da mit dem Ablauf dieses Zertifiakts die von Ihrer Zertifizierungsstelle ausgestellten Zertifikate nicht mehr verifiziert werden können, sollten Sie diese Dauer mit Bedacht wählen. Eine zu kurze Gültigkeitsdauer zwingt Sie dazu, schon bald alle ausgestellten Zertifikate zu erneuern, während eine zu lange Gültigkeitsdauer im Extremfall bedeuten kann, dass Fortschritte in der Computertechnik es möglich machen, den öffentlichen Schlüssel der Zertifizierungsstelle zu brechen, und die Zertifizierungsstelle dadurch kompromittiert wird. Mit einem 2048-Bit-Schlüssel sollten Sie allerdings für die vorhersehbare Zukunft Ruhe haben.

```
[ca]
default_ca = CA

[CA]
dir              = /home/ca
certificate      = $dir/ca-cert.pem                    CA-Zertifikat
private_key      = $dir/private/ca-key.pem             Privater Schlüssel
database         = $dir/index.txt                      Zertifikatsliste
new_certs_dir    = $dir/certs                          Neue Zertifikate
serial           = $dir/serial                         Seriennummer

default_crl_days = 7                                   Rhythmus für CRL-Veröffentlichung
default_days     = 730                                 Gültigkeitsdauer für Zertifikate
default_md       = sha1                                Prüfsumme für Zertifikate

policy           = CA_policy
x509_extensions  = certificate_extensions

[CA_policy]                                            Namensregeln für die Zertifikate
commonName             = supplied                      Muss angegeben sein
emailAddress           = supplied
organizationalUnitName = optional                      Darf fehlen
organizationName       = match                         Muss mit CA-DN übereinstimmen
localityName           = match
countryName            = match

[certificate_extensions]
basicConstraints = CA:false

[req]                                                  Regeln für CA-Zertifikat
default_bits       = 2048
default_keyfile    = /home/ca/private/ca-key.pem
default_md         = sha1
prompt             = no
distinguished_name = CA_dn
x509_extensions    = CA_extensions

[CA_dn]                                                DN für die Zertifizierungsstelle
commonName       = CA
emailAddress     = ca@example.com
organizationName = Beispiel GmbH
localityName     = Musterhausen
countryName      = DE

[CA_extensions]
basicConstraints = CA:true
```

Bild B.1: Konfigurationsdatei für eine OpenSSL-basierte CA

Sie können das gerade erstellte Zertifikat mit dem Kommando

```
$ openssl x509 -in ca-cert.pem -text -noout
```

anschauen.

B.3 Server-Zertifikate generieren

Der Prozess für die Erzeugung von Server-Zertifikaten ist derselbe, egal ob Sie ein Zertifikat von einer kommerziellen Zertifizierungsstelle wünschen oder ob Sie das Zertifikat selbst ausstellen:

CSR

1. Sie (als Inhaber des Servers) erzeugen ein Schlüsselpaar und auf dieser Basis einen *Certificate Signing Request* (CSR) – eine Datei, die Ihren öffentlichen Schlüssel und einen DN für den Server enthält. (Den privaten Schlüssel aus dem Schlüsselpaar behalten Sie natürlich für sich.)
2. Die Zertifizierungsstelle überzeugt sich davon, dass Ihr CSR vernünftig aussieht, stellt das dazugehörige (signierte) Zertifikat aus und schickt es Ihnen zurück.
3. Sie installieren das Zertifikat auf Ihrem Server.

Wenn Sie Ihre eigene Zertifizierungsstelle sind, dann übernehmen Sie den zweiten Schritt natürlich selbst.

Schlüsselpaar und CSR erzeugen Wir müssen Ihnen also als erstes erklären, wie Sie ein Schlüsselpaar und einen CSR generieren. Das funktioniert wieder mit OpenSSL:

```
$ unset OPENSSL_CONF
$ cd /tmp
$ openssl req -newkey rsa:2048 -keyout server-key.pem -out server-csr.pem
Generating a 2048 bit RSA private key
..........................+++
........+++
writing new private key to 'server-key.pem'
Enter PEM pass phrase:abc123
Verifying - Enter PEM pass phrase:abc123
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:DE
State or Province Name (full name) [Some-State]:.
Locality Name (eg, city) []:Musterhausen
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Beispiel GmbH
Organizational Unit Name (eg, section) []:↩
Common Name (eg, YOUR name) []:www.example.com
Email Address []:webmaster@example.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:↩
```

```
An optional company name []:↩
$ _
```

Was Sie als Bestandteile des DN eingeben, ist wahlfrei, solange es zur »Policy« der Zertifizierungsstelle passt (siehe Bild B.1).

In unserem Beispiel kann der »state or province name« leer bleiben, weil die Policy darüber keine Aussagen macht. Ebenso darf der »organizational unit name« leer bleiben (oder einen beliebigen Wert haben). Der »organization name« muss mit dem im Wurzelzertifikat der Zertifizierungsstelle übereinstimmen. Der »common name« ist grundsätzlich beliebig, aber *muss* für ein Server-Zertifikat dem FQDN des betreffenden Servers entsprechen.

Beim Erzeugen des Schlüsselpaars besteht OpenSSL darauf, dass Sie eine Passphrase zur Verschlüsselung des privaten Schlüssels eingeben. Grundsätzlich ist das lobenswert, aber es gibt Situationen – etwa wenn der Schlüssel für einen Server gebraucht wird, der auch starten soll, ohne dass jemand an der Konsole steht und die Passphrase eingibt –, wo Sie vielleicht lieber einen privaten Schlüssel *ohne* Passphrase hätten. Für diesen Fall können Sie wie folgt eine unverschlüsselte Kopie Ihres privaten Schlüssels erzeugen:

```
$ openssl rsa -in server-key.pem -out server-key-np.pem
Enter pass phrase for server-key.pem:abc123
writing RSA key
```

Anschließend steht der unverschlüsselte private Schlüssel in `server-key-np.pem`. Behandeln Sie ihn mit Sorgfalt.

Wenn Sie direkt einen nicht verschlüsselten privaten Schlüssel erzeugen wollen, können Sie »`openssl req`« mit der Option `-nodes` aufrufen.

Zertifikat besorgen Den CSR schicken Sie entweder an eine Zertifizierungsstelle Ihres Vertrauens oder Sie stellen sich selbst ein Zertifikat aus. Für Letzteres müssen Sie im Heimatverzeichnis des Benutzers `ca` stehen, und die Umgebungsvariable `OPENSSL_CONF` muss auf die Konfigurationsdatei für die Zertifizierungsstelle zeigen:

```
$ cd                                                    Zurück ins Heimatverzeichnis
$ export OPENSSL_CONF=$HOME/openssl-ca.cnf
$ openssl ca -in /tmp/server-csr.pem
Using configuration from /home/ca/openssl-ca.cnf
Enter pass phrase for /home/ca/private/ca-key.pem:geheim
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'DE'
localityName          :PRINTABLE:'Musterhausen'
organizationName      :PRINTABLE:'Beispiel GmbH'
commonName            :PRINTABLE:'www.example.com'
emailAddress          :IA5STRING:'webmaster@example.com'
Certificate is to be certified until May  3 10:00:56 2013 GMT (730 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Certificate:
  ◁◁◁◁◁
Data Base Updated
$ _
```

Das neue Zertifikat landet dann im Verzeichnis `certs`. Die Zertifikate dort sind nach ihrer Seriennummer benannt (das gerade angelegte Zertifikat heißt `certs/01.pem`), so dass Sie gegebenenfalls in der Datei `serial` nachschauen müssen, was die letzte vergebene Nummer war. (Denken Sie daran, dass `serial` immer die Nummer des *nächsten* Zertifikats enthält. Wenn Sie schlau sind, schauen Sie statt dessen in die Datei `serial.old`.) (Denken Sie auch daran, dass die Seriennummern hexadezimal sind.)

Das neue Zertifikat (und möglicherweise den dazugehörigen privaten Schlüssel) sollten Sie anschließend auf den gewünschten Rechner kopieren. Wenn Sie – wie empfohlen – einen dedizierten Rechner für die Zertifizierungsstelle verwenden, dann involviert dieser Prozess sinnvollerweise einen USB-Stick, denn es ist besser, wenn der Rechner mit der Zertifizierungsstelle nicht ans Netz angeschlossen ist.

Client-Zertifikate

Client-Zertifikate Zertifikate für Benutzer – sogenannte Client-Zertifikate – können Sie übrigens ganz analog erstellen. Dafür verwenden Sie einfach einen CSR, bei dem das »common name«-Feld keinen FQDN enthält, sondern den Namen der betreffenden Person.

Im Idealfall erzeugen nicht Sie diesen CSR (und das dazugehörige Schlüsselpaar), sondern der spätere Inhaber des Zertifikats. Nur auf diese Weise kann sicher gestellt werden, dass der private Schlüssel des Benutzers wirklich privat bleibt.

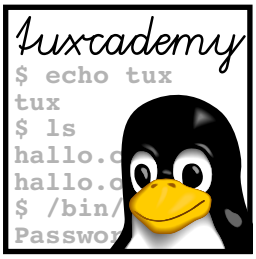
PKCS#12

Web-Browser erwarten Client-Zertifikate gerne im »PKCS#12«-Format, das das Zertifikat und den dazugehörigen privaten Schlüssel zusammenfasst. Wenn Sie sowohl das Zertifikat als auch den privaten Schlüssel haben, können Sie ein PKCS#12-Zertifikat mit dem Kommando

```
$ openssl pkcs12 -export -in hugo-cert.pem -inkey hugo-key.pem -out hugo.p12
Enter pass phrase for hugo-key.pem:x8o,AqS!k
Enter Export Password:foo-bar
Verifying - Enter Export Password:foo-bar
```

erzeugen.

Das »Export Password« müssen Sie wieder angeben, wenn Sie das PKCS#12-Zertifikat mit dem Browser einlesen. Es muss (bzw. sollte) nichts mit dem Kennwort für den privaten Schlüssel zu tun haben.



C

LPIC-2-Zertifizierung

C.1 Überblick

Das *Linux Professional Institute* (LPI) ist eine herstellerunabhängige, nicht profitorientierte Organisation, die sich der Förderung des professionellen Einsatzes von Linux widmet. Ein Aspekt der Arbeit des LPI ist die Erstellung und Durchführung weltweit anerkannter, distributionsunabhängiger Zertifizierungsprüfungen beispielsweise für Linux-Systemadministratoren.

Mit der „LPIC-2"-Zertifizierung des LPI können Sie nachweisen, dass Sie über fortgeschrittene Kenntnisse der Administration von Linux verfügen. Die Zertifizierung richtet sich an Kandidaten mit einigen Jahren Linux-Erfahrung und prüft das Wissen, das Sie benötigen, um ein kleines bis mittleres Netzwerk aus Linux- und Windows-Rechnern zu planen, einzuführen, warten und konsistent und sicher zu erhalten. (Zu Ihrer Beruhigung: Es geht hier nur um die Linux-Aspekte.) Das schließt einen Samba-Server, Internetdienste wie einen Firewall, Proxy-Server, Mail-Server, Web-Server und FTP-Server ein.

Die Zertifizierung besteht aus zwei Prüfungen, LPI-201 und LPI-202. Diese werden in Form von computerorientierten Multiple-Choice- und Kurzantworttests über die Prüfungszentren von Pearson VUE und Thomson Prometric angeboten oder können auf Veranstaltungen wie dem LinuxTag oder der CeBIT zu vergünstigten Preisen auf Papier abgelegt werden. Das LPI veröffentlicht auf seinen Web-Seiten unter `http://www.lpi.org/` die **Prüfungsziele**, die den Inhalt der Prüfungen umreißen.

Prüfungsziele

Die vorliegende Unterlage ist Teil eines Kurskonzepts der Linup Front GmbH zur Vorbereitung auf die Prüfung LPI-201 und deckt damit einen Teil der offiziellen Prüfungsziele ab. Details können Sie den folgenden Tabellen entnehmen. Eine wichtige Beobachtung in diesem Zusammenhang ist, dass die LPIC-2-Prüfungsziele nicht dazu geeignet oder vorgesehen sind, einen Einführungskurs in Linux didaktisch zu strukturieren. Aus diesem Grund verfolgt unser Kurskonzept keine strikte Ausrichtung auf die Prüfungen oder Prüfungsziele in der Form „Belegen Sie Kurs x und y, machen Sie Prüfung p, dann belegen Sie Kurs a und b und machen Sie Prüfung q". Ein solcher Ansatz verleitet viele Kurs-Interessenten zu der Annahme, sie könnten als absolute Linux-Einsteiger n Kurstage absolvieren (mit möglichst minimalem n) und wären anschließend fit für die LPIC-2-Prüfungen. Die Erfahrung lehrt, dass das in der Praxis nicht funktioniert, da die LPI-Prüfungen geschickt so angelegt sind, dass Intensivkurse und prüfungsorientiertes „Büffeln" nicht wirklich helfen.

Entsprechend ist unser Kurskonzept darauf ausgerichtet, Ihnen in didaktisch sinnvoller Form ein solides Linux-Basiswissen zu vermitteln und Sie als Teilnehmer in die Lage zu versetzen, selbständig mit dem System zu arbeiten. Die LPIC-2-

Zertifizierung ist nicht primäres Ziel oder Selbstzweck, sondern natürliche Folge aus Ihren neuerworbenen Kenntnissen und Ihrer Erfahrung.

C.2 Prüfung LPI-201

Die folgende Tabelle zeigt die Prüfungsziele der Prüfung LPI-201 und die Unterlagen, die diese Prüfungsziele abdecken. Die Zahlen in den Spalten für die einzelnen Unterlagen verweisen auf die Kapitel, die das entsprechende Material enthalten.

Nr	Gew	Titel	SYAP	NADM
201.1	2	Kernel-Komponenten	1–2	–
201.2	2	Einen Kernel übersetzen	2	–
201.3	1	Einen Kernel patchen	2	–
201.4	2	Einen angepassten Kernel und Kernelmodule übersetzen und installieren	2	–
201.5	3	Kernel und Kernelmodule zur Laufzeit verwalten und abfragen	1	–
202.1	4	System-Startprozesse anpassen	3	–
202.2	4	Systemreparatur	3	–
203.1	4	Das Linux-Dateisystem betreiben	–	–
203.2	3	Linux-Dateisysteme warten	–	–
203.3	2	Dateisystem-Optionen anlegen und konfigurieren	–	–
203.4	1	udev-Geräteverwaltung	8	–
204.1	2	RAID konfigurieren	–	–
204.2	1	Zugriff auf Speichergeräte einstellen	–	–
204.3	3	Logical Volume Manager	–	–
205.1	3	Grundlegende Netz-Konfiguration	–	1, 3
205.2	4	Fortgeschrittene Netz-Konfiguration und Problembehebung	–	1–2, 4
205.3	5	Problembehebung im Netz	–	1
205.4	1	Benutzer über Systemangelegenheiten benachrichtigen	–	–
206.1	4	Programme vom Quellcode aus übersetzen und installieren	4	–
206.2	3	Sicherungskopien anlegen und restaurieren	–	–
207.1	2	Grundlegende DNS-Server-Konfiguration	–	5–6, 8
207.2	2	DNS-Zonen erstellen und warten	–	5–10
207.3	2	Einen DNS-Server absichern	–	8, 10–11

C.3 LPI-Prüfungsziele in dieser Schulungsunterlage

205.1 Grundlegende Netz-Konfiguration

Gewicht 3

Beschreibung Kandidaten sollten in der Lage sein, Netz-Hardware so zu konfigurieren, dass sie Verbindung mit einem lokalen kabelgebundenen oder drahtlosen Netz oder einem Weitverkehrsnetz aufnehmen kann. Dieses Prüfungsziel umfasst auch die Kommunikation zwischen verschiedenen Subnetzen in einem einzigen Netz.

Wichtigste Wissensgebiete

- Werkzeuge zur Konfiguration und Manipulation von Ethernet-Schnittstellen
- Konfiguration drahtloser Netze

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- /sbin/route
- /sbin/ifconfig
- /sbin/ip
- /usr/sbin/arp
- /sbin/iwconfig

205.2 Fortgeschrittene Netz-Konfiguration und Problembehebung

Gewicht 4

Beschreibung Kandidaten sollten in der Lage sein, Netz-Hardware so zu konfigurieren, dass sie verschiedene Netz-Authentisierungsverfahren umsetzt. Dieses Prüfungsziel umfasst auch die Konfiguration eines Rechners mit mehreren Netz-Schnittstellen und eines VPN-Clients sowie die Behebung von Kommunikationsproblemen.

Wichtigste Wissensgebiete

- Werkzeuge zur Manipulation von Routing-Tabellen
- Werkzeuge zur Konfiguration und Manipulation von Ethernet-Schnittstellen
- Werkzeuge zur Analyse des Status von Netz-Hardware
- Werkzeuge zur Überwachung und Analyse des TCP/IP-Verkehrs
- OpenVPN

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- /sbin/route
- /sbin/ifconfig
- /bin/netstat
- /bin/ping
- /usr/sbin/arp
- /usr/sbin/tcpdump
- /usr/sbin/lsof
- /usr/bin/nc
- /sbin/ip
- /etc/openvpn/*
- openvpn
- nmap
- wireshark

205.3 Problembehebung im Netz

Gewicht 5

Beschreibung Kandidaten sollten in der Lage sein, gängige Netz-Konfigurationsprobleme zu erkennen und zu beheben. Dies umfasst auch Wissen über die Orte üblicher Konfigurationsdateien und Kommandos im Dateisystem.

Wichtigste Wissensgebiete

- Ort und Inhalt von Zugangsbeschränkungsdateien
- Werkzeuge zur Konfiguration und Manipulation von Ethernet-Schnittstellen
- Werkzeuge zur Verwaltung von Routing-Tabellen
- Werkzeuge zum Auflisten von Netz-Zuständen
- Werkzeuge zur Bestimmung der Netz-Konfiguration
- Methoden zur Information über die erkannte und verwendete Netz-Hardware
- System-Initialisierungsdateien und ihr Inhalt (System-V-Init-Prozess)

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- /sbin/ifconfig
- /sbin/route
- /bin/netstat
- /etc/network oder /etc/sysconfig/network-scripts/
- System-Protokolldateien wie /var/log/syslog und /var/log/messages
- /bin/ping
- /etc/resolv.conf
- /etc/hosts
- /etc/hosts.allow und /etc/hosts.deny
- /etc/hostname oder /etc/HOSTNAME
- /bin/hostname
- /usr/sbin/traceroute
- /usr/bin/dig
- /bin/dmesg
- /usr/bin/host

207.1 Grundlegende DNS-Server-Konfiguration

Gewicht 2

Beschreibung Kandidaten sollten in der Lage sein, BIND als nur cachenden DNS-Server zu konfigurieren. Dieses Prüfungsziel umfasst auch die Fähigkeit, ältere BIND-Konfigurationsdateien in neuere Formate umzuwandeln, die Verwaltung eines laufenden Servers und die Konfiguration des Protokolls.

Wichtigste Wissensgebiete

- Konfigurationsdateien, Begriffe und Hilfsprogramme für BIND 9.x
- Definition des Orts der BIND-Zonendateien in BIND-Konfigurationsdateien
- Neuladen von modifizierten Konfigurations- und Zonendateien

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `/etc/named.conf`
- `/var/named/*`
- `/usr/sbin/rndc`
- `kill`

207.2 DNS-Zonen erstellen und warten

Gewicht 2

Beschreibung Kandidaten sollten in der Lage sein, eine Zonendatei für eine Vorwärts- oder Rückwärts-Zone oder einen Root-Level-Server anzulegen. Dieses Prüfungsziel umfasst das Setzen korrekter Werte für Records, das Hinzufügen von Rechnern zu Zonen und das Hinzufügen von Zonen zum DNS. Ein Kandidat sollte ferner in der Lage sein, Zonen an einen anderen DNS-Server zu delegieren.

Wichtigste Wissensgebiete

- Konfigurationsdateien, Begriffe und Hilfsprogramme für BIND 9
- Hilfsprogramme, die Daten von einem DNS-Server abrufen
- Aussehen, Inhalt und Ort im Dateisystem für BIND-Zonendateien
- Verschiedene Methoden, um einen neuen Rechner in die Zonendateien einzufügen, einschließlich Rückwärtszonen

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `/var/named/*`
- Zonendatei-Syntax
- Formate von Resource-Records
- `dig`
- `nslookup`
- `host`

207.3 Einen DNS-Server absichern

Gewicht 2

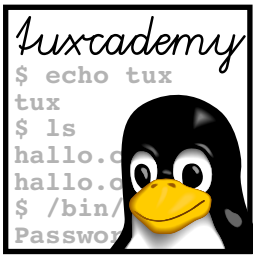
Beschreibung Kandidaten sollten in der Lage sein, einen DNS-Server so zu konfigurieren, dass er als Benutzer außer root und in einer chroot-Umgebung läuft. Dieses Prüfungsziel umfasst auch den sicheren Datenaustausch zwischen DNS-Servern.

Wichtigste Wissensgebiete

- BIND-9-Konfigurationsdateien
- Konfiguration von BIND 9 für eine chroot-Umgebung
- Geteilte Konfiguration von BIND mit der forwarders-Direktive

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `/etc/named.conf`
- `/etc/passwd`
- DNSSEC
- `dnssec-keygen`

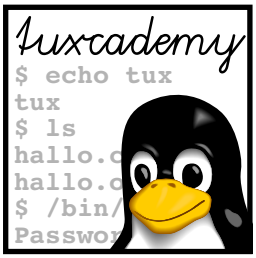


D

Kommando-Index

Dieser Anhang fasst alle im Text erklärten Kommandos zusammen und verweist auf deren Dokumentation sowie die Stellen im Text, wo die Kommandos eingeführt werden.

dig Sucht Informationen im DNS (sehr komfortabel) dig(1) 73
dnssec-keygen Erzeugt Schlüssel für DNSSEC dnssec-keygen(8) 110, 86
host Sucht Informationen im DNS host(1) 76
ifconfig Konfiguriert Netzwerk-Schnittstellen ifconfig(8) 19
ifdown Schaltet eine Netzwerk-Schnittstelle aus (Debian) ifdown(8) 23
ifup Schaltet eine Netzwerk-Schnittstelle ein (Debian) ifup(8) 23
inetd Internet-Superserver, überwacht Ports und startet ggf. Dienste inetd(8) 17
ip Verwaltet Netzwerkschnittstellen und Routing ip(8) 22
iwconfig Programm zur Konfiguration von WLAN-Geräten iwconfig(8) 57
mesg Schaltet wall-Nachrichten für ein Terminal ein oder aus mesg(1) 29
named Der Berkeley-Internet-Name-Daemon (BIND) named(8) 81
ndc Steuerprogramm für BIND ndc(8) 85, 88
nslookup Sucht Informationen im DNS nslookup(1) 75
rndc Programm zur „Fernsteuerung“ von BIND (moderne Version) rndc(8) 85
rndc-confgen Legt Konfigurationsdateien für rndc (und BIND) an rndc-confgen(8) 86
route Verwaltet die statische Routing-Tabelle im Linux-Kern route(8) 20
tcpdump Netzwerk-Sniffer, protokolliert und analysiert Netzwerkverkehr tcpdump(1) 32
wall Schreibt eine Nachricht auf die Terminals aller angemeldeten Benutzer wall(1) 29
whois Gibt Informationen über DNS-Domains aus whois(1) 65
wireshark Paket-Sniffer, liest und analysiert Netzwerkverkehr (Ex-ethereal) wireshark(1) 41
wpa_supplicant Kümmert sich um WLAN-Zugang mit Verschlüsselung wpa_supplicant(8) 58
xinetd Verbesserter Internet-Superserver, überwacht Ports und startet ggf. Dienste xinetd(8) 17



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.

`127.0.0.zone`, 84

`acl` (`/etc/named.conf`), 125, 139
Adressensuchlisten, **124**
`allow-query` (`/etc/named.conf`), 136
`allow-recursion` (`/etc/named.conf`), 136
`allow-transfer` (`/etc/named.conf`), 108, 110–112, 124
`also-notify` (`/etc/named.conf`), 109
autoritativer DNS-Server, **68**

Bernstein, Dan J., 80
BIND, 80
`bind`
    `-g` (Option), 132
    `-u` (Option), 132
Broadcast-Adresse, **17**

`cd`, 148
`chroot`, 133

Datagramme, **16**
`db.root`, 83
Definitionen, **12**
DENIC, **65**, 104
`/dev`, 19
`/dev/log`, 134
`dig`, 72–76, 85, 94, 107–108, 112, 117, 122, 139, 147–148
    `-b` (Option), 139
    `+noadditional` (Option), 75
    `+noanswer` (Option), 75
    `+noauthority` (Option), 75
    `+nocomments` (Option), 75
    `+noquestion` (Option), 75
    `+norecurse` (Option), 75
    `+nostats` (Option), 75
    `+notcp` (Option), 75
    `+recurse` (Option), 75
    `+search` (Option), 75
    `+short` (Option), 75
    `+tcp` (Option), 75
    `+trace` (Option), 75
    `-y` (Option), 112
`~/.digrc`, 75
`directory` (`/etc/named.conf`), 83
DJBDNS, 80
DNS, **62!–63**
`dnsdomainname`, 72, 145
`dnsmasq`, 80
`dnssec-keygen`, 86, 110–112
    `-a` (Option), 110
    `-b` (Option), 110
    `-n` (Option), 110
`domain` (`/etc/resolv.conf`), 71
Domain-Modell, **64**
`domainname`, 72, 145
`dumpcap`, 41

`echo`, 21
`emacs`, 29
`/etc/bind`, 83
`/etc/bind/db`, 83
`/etc/default/bind9`, 133–134
`/etc/ethers`, 34
`/etc/hosts`, 62–63, 70, 80, 144
`/etc/init.d/bind9`, 133
`/etc/init.d/networking`, 23
`/etc/issue`, 28–30
`/etc/issue.net`, 28
`/etc/motd`, 29
`/etc/named.conf`, 81, 125
    `acl`, 125, 139
    `allow-query`, 136
    `allow-recursion`, 136
    `allow-transfer`, 108, 110–112, 124
    `also-notify`, 109
    `directory`, 83
    `file`, 105
    `forward`, 126
    `forwarders`, 126–127
    `listen-on`, 83
    `masters`, 105, 111, 121

match-clients, 139
notify, 109
options, 109, 112, 139
recursion, 135
server, 111
statistics-file, 83
use-id-pool, 136
version, 149
view, 139
/etc/network/interfaces, 23, 25
/etc/networks, 34
/etc/nologin, 30, 143
/etc/nsswitch.conf, 70
/etc/protocols, 99
/etc/resolv.conf, 71–72, 74–76, 139
domain, 71
nameserver, 71, 139
options, 72
search, 71
sortlist, 72
/etc/services, 17, 34–35, 99
/etc/sysconfig/named, 134
/etc/sysconfig/network, 23
/etc/sysconfig/network-scripts, 24
/etc/sysconfig/network-scripts/ifcfg-eth0, 24
/etc/sysconfig/network/config, 23
/etc/sysconfig/network/routes, 23
/etc/sysconfig/static-routes, 24
/etc/wpa_supplicant.conf, 58
ethereal, 41, 52, 165

file (/etc/named.conf), 105
forward (/etc/named.conf), 126
forwarders (/etc/named.conf), 126–127
Fragmentierung, **16**

gated, 21
getty, 28–29

Herrmansfeldt, Glen, 120
host, 72, 76, 107–108, 122, 147–148
-a (Option), 147
-l (Option), 147
-t (Option), 77

ifconfig, 19–21, 56–57
-a (Option), 56
ifdown, 23–24
-a (Option), 23
ifup, 23–24
-a (Option), 23
inetd, 17
Internet Systems Consortium, 80
ip, 22
addr (Option), 22
help (Option), 22
link (Option), 22
route (Option), 22

ISC, 64, 80
iwconfig, 57–58

Kelley, Simon, 80

/lib/firmware, 57
listen-on (/etc/named.conf), 83
login, 29
ls, 30

masters (/etc/named.conf), 105, 111, 121
match-clients (/etc/named.conf), 139
mesg, 29–30, 143
mimencode, 86
mmencode, 111
Mockapetris, Paul, 63

named, 81–83
-c (Option), 82
named.conf, 81–84, 86, 117
named.run, 87
named.stats, 87
nameserver (/etc/resolv.conf), 71, 139
ndc, 79, 85, 88–89, 108
Netzmaske, **17**
Netzwerkadresse, **17**
NIS, **62**
notify (/etc/named.conf), 109
nslookup, 72, 75–76
-silent (Option), 76

openssl, 157
-days (Option), 154
openssl req
-nodes (Option), 157
OPENSSL_CONF (Umgebungsvariable), 157
options (/etc/named.conf), 109, 112, 139
options (/etc/resolv.conf), 72

ping, 32, 35, 70
Portnummern, **16**
Ports, 17
/proc, 112
Prüfungsziele, **159**
ps, 112

rcnetwork, 23
recursion (/etc/named.conf), 135
Registrars, **67**
rekursiver DNS-Server, **68**
Resolver, **70**
rndc, 79, 85–89, 107–108, 165
-s (Option), 88
rndc-confgen, 86–87
-a (Option), 87
rndc.conf, 86–88
rndc.key, 87
root.hint, 84
route, 20–23
Runlevel, 81

scp, 111

search (/etc/resolv.conf), 71
server (/etc/named.conf), 111
shutdown, 30, 143
sortlist (/etc/resolv.conf), 72
ssh, 29, 111
sshd, 29
statistics-file (/etc/named.conf), 83
syslogd, 134
    -a (Option), 134

tcpdump, 11, 31–34, 36–41, 43–44, 47, 52, 85, 107
    -e (Option), 38
    -i (Option), 33
    -N (Option), 39
    -n (Option), 33, 39
    -q (Option), 37
    -t (Option), 40
    -tt (Option), 40
    -ttt (Option), 40
    -tttt (Option), 40
    -ttttt (Option), 40
    -v (Option), 39
    -vv (Option), 39
    -vvv (Option), 39
    -w (Option), 33
    -X (Option), 38
    -x (Option), 38
traceroute, 32
tshark, 41

Umgebungsvariable
    OPENSSL_CONF, 157
use-id-pool (/etc/named.conf), 136

/var/cache/bind, 105
/var/lib/named, 81
/var/run/named.pid, 89
/var/run/ndc, 88
verbindungsloses Protokoll, **16**
version (/etc/named.conf), 149
vi, 29
view (/etc/named.conf), 139
Vixie, Paul, 80

wall, 29–30, 143, 165
whois, 65, 70
wicd, 58
wireshark, 11, 31–32, 40–41, 43–44, 47–48, 50–52, 85, 107, 109
wpa_passphrase, 58
wpa_supplicant, 58–59
write, 30

xinetd, 17

Zone, **68**