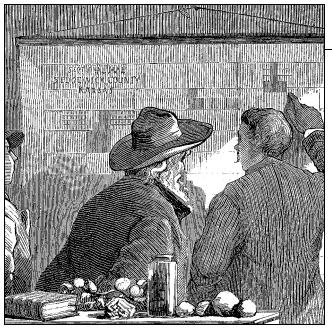


KAPITEL 6

Domain Name Services (DNS) absichern

Der Domain Name Service (DNS) ist einer der grundlegendsten und notwendigsten Internet-Dienste. Ohne DNS müssten sowohl Benutzer als auch Anwendungen sämtliche Rechner im Internet über ihre Internet Protocol-(IP-)Adresse ansprechen und könnten nicht einfach einen Namen angeben, den man sich viel leichter merken kann. Man kann darüber streiten, ob das Internet ohne DNS eine Randerscheinung im Bereich der Universitäten und des Militärs geblieben wäre, anstatt sich zu einem integralen Bestandteil von Gesellschaft und Kultur zu entwickeln. (Wer außer einem Computernarren würde denn schließlich dieses Buch lieber unter 62.206.71.154 und nicht unter *www.oreilly.de* suchen wollen?)

Dennoch werden in der aktuellsten Version des Konsensdokuments des SANS-Instituts, »The Twenty Most Critical Internet Security Vulnerabilities« (Die zwanzig wichtigsten Sicherheitslücken im Internet; Version 4.0 vom 8. Oktober 2003, zu finden unter *http://www.sans.org/top20.htm*), auf Grundlage der Berichte von Teilnehmern Schwachstellen von BIND als häufigste Unix-Sicherheitslücke aufgeführt. Die Berkeley Internet Name Domain (BIND) ist das OpenSource-Paket, das auf der Mehrheit der Internet-DNS-Server eingesetzt wird. Nach anderen Aussagen von SANS ist »eine ungewöhnlich große Zahl« von BIND-Installationen anfällig für die Ausnutzung von wohlbekannten (in vielen Fällen alten) Sicherheitslücken.

Dass es so viele Rechner mit Sicherheitslücken in einem essenziellen Dienst gibt, ist in der Tat eine schlechte Nachricht. Die gute Nachricht für Sie besteht darin, dass Sie mithilfe von einigen einfachen Konzepten und Technologien die Sicherheit von BIND auf Ihrem Linux- (oder einem anderen Unix-)DNS-Server erheblich verbessern können. Dieses Kapitel beginnt deshalb zwar mit einigen Hintergrundinformationen zu BIND, im Wesentlichen geht es im Folgenden aber um Sicherheit. Wenn Sie also ein absoluter DNS-Neueinsteiger sind, sollten Sie vielleicht zusätzlich kurz einen Blick in die ersten ein oder zwei Kapitel des Buchs *DNS and BIND* (dt. Titel *DNS und BIND*) von Albitz und Liu (erschienen bei O'Reilly) werfen.

Sollten Sie auch nach diesen Erläuterungen BIND noch misstrauisch gegenüberstehen oder es einfach nicht mögen und etwas anderes ausprobieren wollen, erhalten Sie im

Anschluss an die BIND-Erklärungen Informationen zu djbdns, einer hoch angesehenen Alternative zu BIND. Wir werden uns nicht nur mit einigen Argumenten für und wider djbdns auseinander setzen, sondern auch kurze Hinweise zur Installation und Absicherung von djbdns geben.

DNS-Grundlagen

Auch wenn ich gerade gesagt habe, dass in diesem Kapitel Kenntnisse über DNS vorausgesetzt werden, lassen Sie uns zunächst einige wichtige Begriffe und Konzepte klären, die im Zusammenhang mit DNS stehen.

Nehmen wir an, jemand (*meinrechner.irgendeinisp.com* in Abbildung 6-1) surft im Web und möchte sich die Site *http://www.dogpeople.org* anschauen. Nehmen wir weiter an, dass die Maschine, die dieser Jemand benutzt, so konfiguriert ist, dass sie auf den Nameserver *ns.irgendeinisp.com* zugreift, um DNS-Lookups durchzuführen. Da der Name »www.dogpeople.org« für die Router, die die Webanfrage und die Antworten darauf passieren müssen, keinerlei Bedeutung hat, muss der Webbrowser des Benutzers erst die Internet Protocol-(IP-)Adresse in Erfahrung bringen, mit der der Name *http://www.dogpeople.org* assoziiert wird. Erst dann kann er versuchen, die Anfrage durchzuführen.

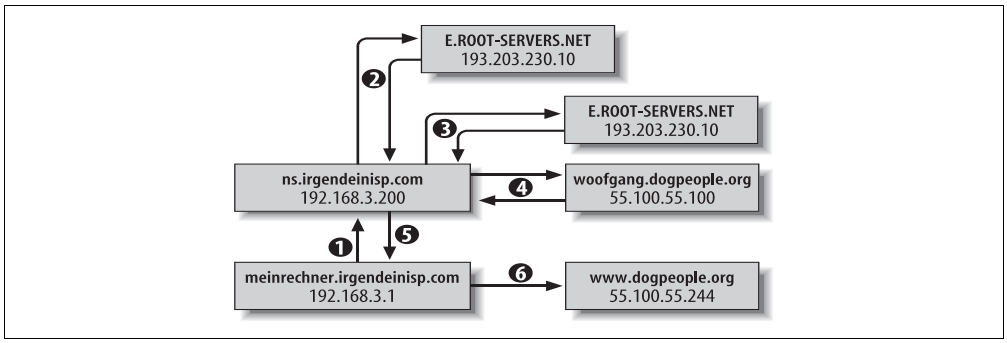


Abbildung 6-1: Eine rekursive DNS-Anfrage

meinrechner fragt also zuerst den Server *ns*, ob er die IP-Adresse kennt. Da *dogpeople.org* nicht zum Kontrollbereich von *ns.irgendeinisp.com* gehört und auch nicht vor kurzem mit einem anderen Rechner kommuniziert hat, der *dogpeople.org* kontrolliert, leitet er eine Abfrage für den Benutzer ein. Das Initiieren von einer oder mehreren Abfragen zur Beantwortung einer vorhergehenden Abfrage bezeichnet man als *Rekursion*.

ns.irgendeinisp.com beginnt seine rekursive Anfrage, indem er zunächst einen *Root-Nameserver* nach der IP-Adresse des Rechners fragt, der die Kontrolle über die generische Top Level Domain *.org* hat. (Alle Internet-DNS-Server benutzen eine statische »hints«-(Hinweis-) Datei, um die ungefähr 13 offiziellen Root-Nameserver zu identifizieren. Diese Liste wird unter *ftp://ftp.rs.internic.net/domain* unter dem Namen *named.root* geführt.) In unserem Beispiel fragt *ns E.ROOT-SERVERS.NET* (einen echten Root-Ser-

ver, dessen IP-Adresse derzeit 193.203.230.10 ist), und dieser antwortet, dass der DNS für *.org* von *TLD1.ULTRADNS.NET*, dessen IP-Adresse 204.74.112.1 ist, verwaltet wird.

Nun fragt *ns* den Rechner *TLD1.ULTRADNS.NET* nach dem Namen und der IP-Adresse des DNS-Servers, der für die Zone *dogpeople.org* zuständig ist. *TLD1.ULTRADNS.NET* gibt als Antwort zurück, dass der DNS-Server von *dogpeople.org* unter dem Namen *woofgang.dogpeople.org* und der IP-Adresse 55.100.55.100 zu erreichen ist.

Anschließend fragt *ns woofgang* (unter Verwendung von *dessen* IP-Adresse 55.100.55.100) nach der IP von *www.dogpeople.org*. *woofgang* beantwortet die Frage (55.100.55.244), und diese Antwort wird von *ns* wiederum an *meinrechner.irgendeinisp.com* weitergeleitet. Schließlich ruft *meinrechner* direkt die IP-Adresse 55.100.55.244 über HTTP auf und führt die Webabfrage durch.

Bei dieser Vorgehensweise handelt es sich um den gängigsten Weg zur Ermittlung eines Namens bzw. einer IP-Adresse. Diese und andere Arten von Nachfragen zu einzelnen Rechnern bezeichnet man auch einfach als *Abfragen*. DNS-Abfragen werden über den UDP-Port 53 abgewickelt.

Allerdings geht es nicht bei allen DNS-Transaktionen darum, einzelne Rechner zu ermitteln. Manchmal ist es notwendig, ganze Domain-(Zonen-)Datenbanken zu übertragen: Man bezeichnet das auch als *Zonen-Transfer*. Eine derartige Übertragung wird durchgeführt, wenn Sie den Endbenutzerbefehl *host* mit dem *-l*-Flag und den Befehl *dig* unter Angabe der Abfrageart *axfr* eingeben. Das Resultat einer solchen Abfrage ist eine vollständige Liste aller DNS-Einträge, die sich auf die angefragte Zone beziehen.

host und *dig* werden normalerweise eingesetzt, um die Ursachen von Fehlern festzustellen. Zonen-Transfers dienen Nameservern mit Kontrollfunktion für dieselbe Domain dazu, sich miteinander zu synchronisieren (z.B. für Updates »vom Master zum Slave«). Tatsächlich ist es sogar so, dass ein Master-Server Anfragen nach Zonen-Transfers zurückweisen sollte, wenn diese nicht von einem Rechner stammen, der ein bekannter und zulässiger Slave-Server ist. Zonen-Transfers werden über den TCP-Port 53 abgewickelt.

Das letzte allgemeine DNS-Konzept, das wir hier kurz ansprechen möchten, ist das *Cachen*. Nameserver cachen alle lokalen Zonen-Dateien (d.h. ihre *hints*-Dateien sowie alle Informationen über die Zone, für die sie zuständig sind) und darüber hinaus die Ergebnisse aller rekursiven Abfragen, die sie, seitdem sie zuletzt gestartet wurden, durchgeführt haben – also fast alle. Jeder *Resource-Record* (RR, dt. Quelleintrag) verfügt über eine eigene Time-to-live-Einstellung (TTL, dt. Lebensdauer-Einstellung) oder über einen Standardeintrag, den er von seiner Zonen-Datei geerbt hat. Dieser Wert gibt an, wie lange jeder RR gecacht werden darf, bevor er aktualisiert werden muss.

Diese Hinweise sind natürlich nicht annähernd ausreichend, um BIND vollständig zu verstehen und zu benutzen. Sie reichen aber dazu aus, sich mit der Sicherheit von BIND auseinander zu setzen.

DNS-Sicherheitsrichtlinien

Die DNS-Sicherheitsgrundsätze kann man in zwei Maximen zusammenfassen: Benutzen Sie immer die aktuellste Version der von Ihnen gewählten DNS-Software, und bieten Sie Fremden nicht unnötigerweise Informationen oder Dienste an. Anders ausgedrückt: Bleiben Sie aktuell, und seien Sie geizig mit Informationen!

Die Anwendung dieser Maximen führt uns zu einer Reihe von speziellen Techniken. Bei der ersten geht es darum, die Rekursion zu begrenzen oder sogar völlig zu deaktivieren, weil Rekursion leicht bei DNS-Attacken, wie zum Beispiel dem Cache-Poisoning, missbraucht werden kann. Die Rekursion kann man leicht über Parameter in der Konfigurationsdatei begrenzen. Ob die Rekursion auch komplett deaktiviert werden kann, hängt von der Rolle des Nameservers ab.

Wenn es sich bei dem Server zum Beispiel um einen *externen* DNS-Server handelt, dessen einzige Aufgabe darin besteht, Abfragen hinsichtlich der öffentlich zugänglichen Server der Organisation zu beantworten, zu der er gehört, gibt es keinen Grund, warum dieser Server Nachforschungen über nicht-lokale Host-Namen durchführen können sollte (was wiederum genau die Definition von Rekursion ist). Bietet ein Server aber andererseits DNS-Auflösungsdienste für Endbenutzer in einem Local Area Network (LAN) an, muss er definitiv rekursive Abfragen bei lokalen Rechnern durchführen können. Möglicherweise kann er aber so eingerichtet werden, dass er rekursive Abfragen, wenn nicht sogar alle Abfragen, von nicht-lokalen Adressen nicht zulässt.

Eine andere Möglichkeit zur Begrenzung der DNS-Aktivität besteht darin, *DNS-Dienste aufzuspalten* (vgl. Abbildung 6-2). Bei aufgespaltenem DNS handelt es sich um ein Beispiel für das »Diensteaufteilungskonzept«, das ich in Kapitel 2 im Abschnitt »Entscheiden, was in die DMZ gehört« vorgestellt habe. In diesem Fall bezieht es sich auf die Trennung zwischen *öffentlichen* und *privaten* Datenbanken für jede lokale Domain (Zone). Die öffentliche Zonen-Datenbank enthält so wenige Informationen wie möglich: Sie sollte NS-Einträge für die öffentlich zugänglichen Nameserver, MX-Einträge für SMTP-(E-Mail-)Gateways und A-Einträge (Adressen) für öffentliche Webserver beinhalten. Hinzu kommen Einträge, die sich auf beliebige andere Rechner beziehen, über die die Welt etwas erfahren soll.

Bei der privaten Zonen-Datenbank kann es sich entweder um eine Obermenge der öffentlichen handeln, oder sie kann völlig andere Einträge zu bestimmten Kategorien oder Rechnern enthalten.

Ein weiterer Aspekt des DNS-»Geizes« bezieht sich auf den Inhalt der Zonen-Dateien. Sogar öffentliche Zonen-Datenbanken enthalten oftmals mehr Informationen, als wirklich notwendig ist. Die Rechner könnten unnötigerweise sprechende Namen haben (z.B. könnten Sie dadurch den falschen Leuten verraten, wofür welcher Server zuständig ist), oder es könnten zu ausführliche Kontaktinformationen angegeben sein. Einige Organisationen führen sogar die Namen und Versionsnummern der Hard- und Software auf, die auf den einzelnen Systemen zum Einsatz kommen! Derartige Informationen sind fast

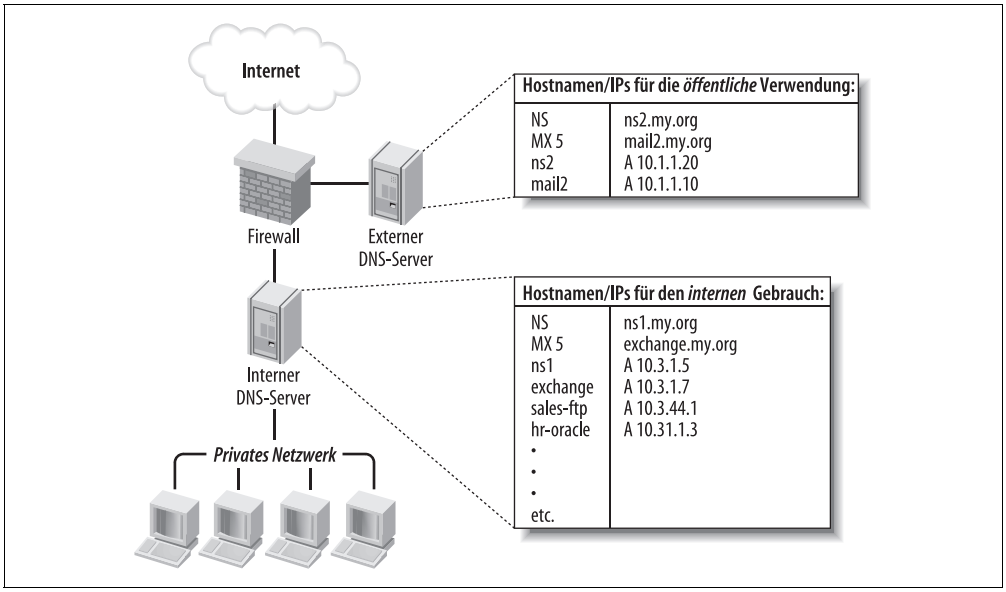


Abbildung 6-2: Aufgespaltenes DNS

immer für potenzielle Angreifer von größerem Nutzen als für diejenigen, für die die Einträge eigentlich bestimmt waren.

Auf dem neuesten Software-Stand zu bleiben und sich über bekannte DNS-Schwachstellen auf dem Laufenden zu halten ist mindestens genauso wichtig wie der Schutz der eigentlichen DNS-Daten. Außerdem ist es einfacher: Die neueste Version von BIND kann man jederzeit unter *ftp://ftp.isc.org* kostenlos herunterladen. Die neueste Version von djb-dns steht unter *http://cr.yp.to* zur Verfügung. Informationen über allgemeine DNS-Sicherheitsfragen und spezielle BIND- und djbdns-Schwachstellen werden über einige Mailinglisten und Newsgroups verbreitet (ein paar davon werden am Ende dieses Kapitels aufgeführt).

Eigentlich gibt es darüber hinaus auch noch eine dritte und vierte Maxime für die DNS-Sicherheit, die allerdings nicht nur auf DNS zutreffen: Nehmen Sie sich die Zeit, die Sicherheitsfunktionalität Ihrer Software zu verstehen, und verwenden Sie die Sicherheitsdienste, die von dem Provider angeboten werden, bei dem Sie Ihre DNS-Anmeldung vorgenommen haben. Network Solutions und andere Top-Level-Domain-Verwalter bieten verschiedene Sicherheitsoptionen für die Durchführung von Änderungsanfragen, darunter PGP. Stellen Sie sicher, dass Ihr Provider mindestens die E-Mail-Adresse, die in Änderungsanfragen für Ihre Domains angegeben ist, überprüft!

Auswahl einer DNS-Software

Die beliebteste und bewährteste DNS-Software ist BIND. BIND war ursprünglich ein Abschlussprojekt von Studenten an der University of California in Berkeley, auf das sich

mittlerweile mehr als 1.000 Sites weltweit verlassen. Die neuste Version von BIND, Version 9, wurde von der Nominum Corporation im Auftrag des Internet Software Consortium (ISC) entwickelt. Die ISC ist die Organisation, die BIND heute offiziell betreut.

BIND war und ist die Referenzimplementierung für die DNS-Standards der Internet Engineering Task Force (IETF). Zum Beispiel enthält die BIND-Version 9 die bislang vollständigste Implementierung des neuen DNSSEC-Standards für die DNS-Sicherheit von der IETF. Aufgrund der Bedeutung und Verbreitung von BIND wird es im größten Teil dieses Kapitels darum gehen, wie man BIND sicherer machen kann.

BIND hat aber auch seine Kritiker. Ähnlich wie bei Sendmail sind auch bei BIND im Laufe der letzten Jahren verschiedene Sicherheitslücken aufgetreten, von denen einige zu schwerwiegenden Konsequenzen geführt haben. BIND ist darüber hinaus, ebenfalls wie Sendmail, kontinuierlich gewachsen und immer komplexer geworden: Es ist nicht mehr so schlank und schnell, wie es einmal war, und auch nicht mehr so stabil. Daher gibt es Menschen, die annehmen, dass BIND unsicher und bei entsprechender Last nicht mehr verlässlich sei.

Daniel J. Bernstein ist so ein BIND-Kritiker, allerdings einer, der tatsächlich etwas dagegen unternommen hat: Er ist der Entwickler von djbdns, einem (abhängig von den Voraussetzungen, die Sie daran stellen) kompletten DNS-Paket. djbdns verfügt über einige wichtige Funktionen:

Modularität
: Anders als BIND, bei dem ein einziger monolithischer Daemon, nämlich *named*, für alles eingesetzt wird, verwendet djbdns verschiedene Prozesse zur Erfüllung unterschiedlicher Aufgaben. Zum Beispiel benutzt djbdns nicht nur unterschiedliche Prozesse zum Auflösen von Namen und zur Beantwortung der Anfragen von anderen Resolvern (Namensauflösern). Es geht vielmehr sogar so weit, dass diese Prozesse auf verschiedenen IP-Adressen lauschen müssen. Diese Modularität resultiert sowohl in besserer Performance als auch in besserer Sicherheit.

Einfachheit
: Die Anhänger von djbdns behaupten, dass es einfacher zu konfigurieren sei als BIND, auch wenn das natürlich subjektiv ist. Allerdings hat djbdns, zumindest vom Programmierstandpunkt aus betrachtet, als Folge des weitaus geringeren Code-Umfangs ein einfacheres Design.

Sicherheit
: Das primäre Ziel bei der Entwicklung von djbdns war Sicherheit. Außerdem führen die weniger umfangreiche Code-Basis und die Einfachheit seiner Architektur dazu, dass djbdns besser überprüfbar ist als BIND: Je weniger Code durchgesehen werden muss, desto weniger Bugs werden übersehen. Bis heute sind keine Sicherheitslücken in irgendeiner Produktionsversion von djbdns bekannt geworden.

Performance
: D. J. Bernstein behauptet, dass djbdns viel schneller und zuverlässiger ist als BIND und dass es viel weniger RAM-Kapazität in Anspruch nimmt. Einige meiner Bekannten, die DNS-Server mit extrem vielen Anfragen warten, setzen daher djbdns ein.

Daraus folgt also, dass djbdns BIND in jeder Hinsicht überlegen ist und dass die große Mehrheit der DNS-Administratoren, die BIND einsetzen, die Dummen sind, oder nicht? Vielleicht ja, aber ich bezweifle das. djbdns wartet mit einigen überzeugenden Vorteilen auf, insbesondere mit seiner Performance. Falls Sie einen Nur-Cache-Nameserver und keine echte DNS-Verwaltung für Ihre Domain benötigen, ist djbdns eindeutig eine elegantere Lösung als BIND. Allerdings bewegt die IETF DNS in zwei Hauptrichtungen, die Mr. Bernstein offenbar für Fehlentwicklungen hält, weswegen er sich weigert, sie in djbdns zu unterstützen.

Bei der ersten Entwicklung handelt es sich um DNSSEC. Um sichere Zonen-Transfers zu gewährleisten, muss djbdns mit rsync oder OpenSSH eingesetzt werden, weil djbdns weder TSIGs noch irgendeinen anderen DNSSEC-Mechanismus unterstützt. Die zweite Entwicklung betrifft IPv6, die djbdns nicht in der vom IETF empfohlenen Weise unterstützt (was nicht heißen soll, dass Mr. Bernstein IPv6 komplett ablehnt; er widerspricht nur der Art, wie es laut IETF-Empfehlung von DNS verwendet werden soll).

Für welche Lösung sollen Sie sich also entscheiden? Wenn Performance Ihr Hauptanliegen ist und wenn Sie glauben, dass djbdns von Grund auf sicherer ist als BIND (und sogar als BIND, das so konfiguriert ist, wie ich es im Folgenden beschreiben werde) oder wenn Sie ein weniger umfangreiches und modulareres Paket als BIND haben möchten, halte ich djbdns für eine gute Wahl.

Sollten Sie aber andererseits DNSSEC verwenden wollen, bereits mit der Administration von BIND vertraut sein oder mit anderen DNS-Servern kooperieren müssen, auf denen BIND läuft (und sich im Stande fühlen, die bereits bekannten und vielleicht in der Zukunft noch auftretenden Sicherheitslücken dadurch zu kompensieren, dass Sie es sorgfältig konfigurieren und ständig bezüglich Sicherheitshinweisen und Updates auf dem Laufenden halten), bin ich nicht der Meinung, dass BIND eine schlechte Wahl für Sie ist.

Ich bin also der Ansicht, dass beide Pakete jeweils über ihre eigenen Vorzüge verfügen: Sie müssen selbst entscheiden, welches Paket eher mit Ihren Ansprüchen übereinstimmt. BIND ist mit Abstand die verbreitetste DNS-Software im Internet, und die meisten meiner Erfahrungen in der Absicherung von DNS-Servern habe ich mit BIND gemacht. Aus diesem Grund konzentriert sich ein Großteil dieses Kapitels auf die DNS-Sicherheit, soweit sie mit den BIND-Versionen 8 und 9 im Zusammenhang steht. Die zweite Hälfte dieses Kapitels behandelt die Grundlagen von djbdns.

Wenn Sie sich weder von BIND noch von djbdns angesprochen fühlen und sich für ein ganz anderes Produkt entscheiden, können Sie, wenn Sie möchten, direkt zum Abschnitt »Sicherheit von Zonen-Dateien« übergehen. Dieser Abschnitt bezieht sich auf alle DNS-Server, unabhängig davon, welche Software darauf zum Einsatz kommt.

Absicherung von BIND

Eine BIND-Installation, mit der Sie sich sicher fühlen können, verlangt eine ganze Menge Arbeit. Das bezieht sich sowohl darauf, wie der Daemon ausgeführt wird, als auch darauf, wie seine Konfigurationsdateien mit Kommunikation umgehen.

Die Versionierung von BIND verstehen

Derzeit sind, trotz der größten Bemühungen des ISC, eine der Versionen zurückzuziehen, drei Hauptversionen von BIND im Einsatz. BIND v9 ist die neueste Version davon, und die aktuellste Update-Version ist zum Zeitpunkt der Erstellung dieses Buchs 9.2.3.

Bedingt durch ein Reihe von praktischen und historischen Gründen haben die Benutzergemeinde von BIND und die meisten UNIX-Händler bzw. -Distributoren BIND v9 nur langsam angenommen. Daher ist BIND v8 immer noch weit verbreitet in Gebrauch. In dieser Version von BIND, BIND v8, sind einige unangenehme Buffer-Overflow-Sicherheitslücken aufgetreten, die zu einer Kompromittierung von root führen können. Es ist daher von grundlegender Bedeutung, dass jeder, der BIND v8 benutzt, auf jeden Fall die neuste Version, also derzeit 8.4.4, einsetzt oder, noch besser, direkt ein Upgrade auf BIND v9 durchführt. BIND v9 enthält keinerlei Code aus BIND v8 oder früher.

Wo wir gerade von älteren Versionen sprechen, muss erwähnt werden, dass einige Benutzer immer noch BIND v4 einsetzen, obwohl BIND v.8.1 im Mai 1997 veröffentlicht wurde. Tatsächlich gibt es sogar ein paar Unix-Händler und -Distributoren (z.B. OpenBSD), die immer noch BIND v4 mit ihrem Betriebssystem zusammen ausliefern. Das ist größtenteils auf Stabilitätsprobleme und Sicherheitslücken in BIND v8 und Misstrauen gegenüber BIND v9 zurückzuführen. Als Konsequenz davon bietet das Internet Software Consortium, wenn auch nur sehr ungern, weiterhin von Zeit zu Zeit Patches für die Version 4 an, obwohl die sonstige Entwicklung dieses Codes schon vor mehreren Jahren eingestellt wurde.

Für welche Version sollten Sie sich jetzt also entscheiden? Meiner Meinung nach ist Version 9 bei weitem die stabilste und sicherste Version von BIND. Außerdem hat sich seit ihrer ursprünglichen Veröffentlichung gezeigt, dass sie von den meisten in BIND 4 und 8 entdeckten Sicherheitslücken nicht betroffen ist. (Diese Tatsache steht im Gegensatz zu den Andeutungen mancher Kritiker, wonach BIND 9 noch immer Code aus den Versionen 4 und 8 enthalten würde.) Bis heute wurden in BIND 9 nur zwei Sicherheitsprobleme entdeckt, wobei es sich bei beiden um Möglichkeiten für Denial-of-Service-Angriffe handelt (beide wurden schnell durch Patches behoben). In BIND 9 wurden bisher keine Schwachstellen entdeckt, die es ermöglichen würden, von einem entfernten Rechner aus einzubrechen und root-Rechte zu erlangen.

Falls Sie sich aus irgendeinem Grund zwischen BIND 4 und BIND 8 entscheiden müssen, sollten Sie die neueste Version von BIND 8 wählen (aufgrund seiner Vorgeschichte in puncto Sicherheit kann ich BIND 8 ansonsten aber nicht empfehlen). Die Unterstützung, die BIND 8 für signierte Transaktionen bietet, die Möglichkeit, es mit chroot abzusichern, und seine Einstellungsmöglichkeiten, um es mit den Rechten nicht privilegierter Benutzer und Gruppen auszuführen (was wir alles demnächst behandeln werden), überwiegen bei weitem die bessere Stabilität, die BIND 4 zu bieten scheint. Da BIND 8 noch immer weit verbreitet ist, werde ich in diesem Kapitel Beispiele für beide Programmversionen, BIND 8 und BIND 9, behandeln. Trotzdem kann ich es nur noch einmal wiederholen: Falls Sie die Wahl haben, sollten Sie unbedingt BIND 9 verwenden!

BIND beschaffen und installieren

Sollten Sie ein vorkompiliertes Binary (z.B. RPM, tgz usw.) verwenden oder BIND selbst aus dem Source-Code kompilieren? Für die meisten Anwender sollte es völlig akzeptabel sein, ein Binary zu verwenden, solange es aus einer vertrauenswürdigen Quelle stammt. Praktisch alle Unix-Varianten beinhalten BIND als Teil ihrer »Standard«-Installationen. Stellen Sie nur sicher, dass Sie wirklich die neueste Version installiert haben.

Falls Sie mit der »Updates«-Webseite Ihres Linux-Distributors noch keine Bekanntschaft gemacht haben sollten, ist es jetzt an der Zeit, sie kennen zu lernen. BIND ist eines der fundamentalen Pakete, von denen die meisten Distributoren die aktuellen Versionen jederzeit bereithalten (d.h. ohne eine neue Update-Version der gesamten Distribution abzuwarten, um neue Pakete bereitzustellen).

Mit folgendem Befehl Ihres Red Hat-Paketmanagers können Sie die Versionsnummer des bei Ihnen installierten BIND-Pakets überprüfen:

```
rpm -q -v paket-name
```

wenn das Paket bereits installiert wurde, oder:

```
rpm -q -v -p /pfad/zu/paket.rpm
```

wenn Sie ein Paket haben, das noch nicht installiert wurde. Der rpm-Paketname von BIND lautet gewöhnlich *bind9* oder *bind*.

Wenn Sie diese Abfrage ausführen und erkennen, dass Sie eine alte (vor 9.2.3 liegende) Version installiert haben, unterstützen die meisten Paketformate eine Upgrade-Funktion. Laden Sie einfach ein aktuelleres Paket von der Website Ihrer Linux-Distribution herunter, und führen Sie das Upgrade mithilfe des Paketmanagers aus. Im Fall von *rpm* können Sie dazu folgende Befehlssyntax verwenden (vorausgesetzt, Sie benötigen keine speziellen Installationsoptionen):

```
rpm -U /pfad/zu/paket.rpm
```

Sollte die vorige Syntax nicht funktionieren, probieren Sie diese aus:

```
rpm -U --force /pfad/zu/paket.rpm
```

Wenn Sie keine passende Binary-Distribution finden, kompilieren Sie eine aus dem Source-Code. Achten Sie aber darauf, dass Sie vorher *gcc* und die übliche Zusammenstellung von Bibliotheken installiert haben.

Die zum Kompilieren von BIND 9 notwendigen Anweisungen sind in der README-Datei dokumentiert. Im Normalfall kann BIND 9 mit folgenden Befehlen kompiliert werden:

```
./configure
make
make install
```

Falls Sie für BIND 9 ein eigenes Installationsverzeichnis angeben möchten, können Sie dazu *configure*s Option `--prefix` verwenden. Beispiel:

```
./configure --prefix=/pfad/zu/installations_root
```

(Der Pfad */pfad/zu/installations_root* ist dabei der absolute Pfad des Verzeichnisses, in dem Sie BIND 9 installieren möchten.)

Wenn Sie sich dazu entschließen, BIND in einem anderen Verzeichnisbaum als dem Standardverzeichnis zu installieren, empfehle ich Ihnen nicht, dafür denselben Baum zu verwenden, den Sie als chroot-Gefängnis einsetzen möchten. (Sollten Sie nicht wissen, was das ist, sollten Sie schon jetzt die ersten paar Absätze des nächsten Abschnitts lesen.) Meiner Meinung nach resultiert eine der Befürchtungen, die zum Einsatz eines chroot-Gefängnisses führen, aus der Vermutung, dass ein Angreifer BIND in seine Gewalt bringen könnte. Falls das so ist, möchten Sie bestimmt nicht, dass der Eindringling BIND-Bibliotheken oder -Binaries ändert oder ersetzt. Kurzum, Sie sollten nicht alle Ihre BIND-Eier zusammen in einem Korb (bzw. in diesem Fall in einem Verzeichnisbaum) aufbewahren.

Falls Sie Transaktionssignaturen oder DNSSEC (beide Features werden in diesem Kapitel noch behandelt) einsetzen möchten, müssen Sie *configure* zusätzlich noch die Option `--with-openssl=yes` mitgeben.

Wenn das *configure*-Skript durchgelaufen ist, geben Sie `make` ein. Wurde das erfolgreich ausgeführt, tippen Sie `make install`. Alle BIND-Binaries und die dazugehörigen Dateien werden nun an den Ort installiert, der von Ihnen angegeben wurde.

Vorbereitungen zum Ausführen von BIND (oder: die Zelle ausstatten)

BIND selbst ist jetzt zwar installiert, wir sind aber noch nicht in der Lage, *named* zu starten. Ich habe auf die befleckte Vergangenheit von BIND, was Sicherheit angeht, hingewiesen, und unser gesunder Menschenverstand sagt uns dazu, dass ein Programm mit einer Vergangenheit voller Sicherheitsprobleme leicht Gefahr läuft, wieder angegriffen zu werden. Es ist daher eine gute Idee, BIND vom Rest des Systems, auf dem es läuft, zu isolieren. Eine Möglichkeit, diese Isolierung durchzuführen, die auch explizit von den BIND-Versionen 8 und 9 unterstützt wird, besteht darin, das Wurzelverzeichnis von *named* zu ändern.

Wenn BIND der Meinung ist, dass ein anderes Verzeichnis als / das Wurzelverzeichnis ist, sitzt der potenzielle Cracker zum Beispiel dann in der Falle, wenn er versucht, eine obskure Buffer-Overflow-Schwachstelle auszunutzen, die es ihm erlaubt, *named* zu werden. Wird *named* mit dem geänderten Wurzelverzeichnis */var/named* ausgeführt, befindet sich eine Datei, die aus der Sicht von *named* scheinbar in */etc* liegt, in Wirklichkeit in */var/named/etc*. Einer, der *named* in seine Gewalt bringt, sieht nicht alle Konfigurationsdateien des Systems, sondern nur die Dateien, die Sie in das Verzeichnis */var/named/etc* gelegt haben (d.h. Dateien, die nur von *named* verwendet werden).

Das Hilfsmittel, das wir normalerweise verwenden, um einen Prozess in einem veränderten Wurzelverzeichnis auszuführen, ist *chroot*. Auch wenn diese Funktionalität integraler Bestandteil von BIND ist (d.h., der *chroot*-Befehl selbst wird nicht benötigt), bezeichnet man das veränderte/gefälschte Wurzelverzeichnis, das wir für *named* ausgesucht haben, dennoch als *chroot-Gefängnis*.

Beachten Sie aber, dass wir *named* zusätzlich als unprivilegierten Benutzer oder unprivilegierte Gruppe und nicht standardmäßig als root ausführen sollten, um die Möglichkeiten des Crackers, das chroot-Gefängnis zu verlassen, zu reduzieren. Diese Funktionalität ist ebenfalls in die BIND-Versionen 8 und 9 integriert.

Wir möchten *named* ohne Zugriff auf das komplette Dateisystem ausführen. Daher müssen wir unsere ausgepolsterte Zelle mit Kopien von allem versorgen, was *named* zur Erledigung seines Jobs braucht. Diese Versorgung reduziert sich auf das Folgende:

1. Wir richten eine geschrumpfte Replik unseres »echten« Wurzeldateisystems ein (z.B. */etc*, */bin*, */sbin*, */var* usw.).
2. Wir kopieren einige Dateien, die BIND in diesem Dateisystem erwartet und benutzt.
3. Wir versehen die Benutzer- und Zugriffsrechte auf diese Dateien und Verzeichnisse mit angemessen paranoiden Einstellungen.

Einrichten eines chroot-Gefängnisses für BIND v8

Die einfachste Möglichkeit, die notwendigen Schritte zum Einrichten eines chroot-Gefängnisses aufzuzählen, besteht darin, einfach das ausgiebig dokumentierte Skript aufzuführen, das ich benutze, um meine BIND-v8-chroot-Gefängnisse einzurichten:

Beispiel 6-1: Einrichten des chroot-Gefängnisses in BIND v8

```
#! /bin/bash
# (Aendern Sie den oben angegebenen Pfad entsprechend, wenn sich Ihr bash-
# Binary an einer anderen Stelle befindet.)
# Befehle zur Einrichtung eines BIND-v8-chroot-Gefaengnisses, mit Aenderungen
# uebernommen aus einem Skript von Kyle Amon
#  (http://www.gnutec.com/~amonk)
# SIE MUESSEN ROOT SEIN, UM DIESES SKRIPT AUSZUFUEHREN!

# Zuerst sollten Sie einige Pfade definieren. BINDJAIL ist das
# Wurzelverzeichnis von BINDs chroot-Gefaengnis.

BINDJAIL = /var/named

# BINDBIN ist das Verzeichnis, in dem sich named, rndc und andere BIND-
# Executables befinden.

BINDBIN = /usr/sbin

# Als Zweites legen Sie das chroot-Gefaengnis mit allen Unterverzeichnissen an:
```

Beispiel 6-1: Einrichten des chroot-Gefängnisses in BIND v8 (Fortsetzung)

```
mkdir -m 2750 -p $BINDJAIL/dev $BINDJAIL/etc
mkdir -m 2750 -p $BINDJAIL/usr/local/libexec
mkdir -m 2770 -p $BINDJAIL/var/run
mkdir -m 2770 $BINDJAIL/var/log $BINDJAIL/var/tmp
mkdir -m 2750 $BINDJAIL/master
mkdir -m 2770 $BINDJAIL/slave $BINDJAIL/stubs

# Als Drittes richten Sie unprivilegierte Benutzer und Gruppen fuer named ein.
# (Diese koennte es schon geben, wenn Sie SUSE oder Mandrake verwenden.
# Achten Sie aber darauf, dass der passwd-Eintrag auf
# /bin/false und nicht auf eine echte Shell zeigt.)

echo "named:x:256: " >> /etc/group
echo "named:x:256:256:BIND:$BINDJAIL:/bin/false" \
>>  /etc/passwd

# Aendern Sie als Viertes einige Benutzer- und Zugriffsrechte.


chown -R root:named $BINDJAIL

# Kopieren Sie fuenftens einige wichtige Dateien in das chroot-Gefaengnis.

# Die naechste Zeile kann in den meisten Faellen ausgelassen werden.
cp $BINDBIN/named $BINDJAIL

# Die uebrigen Zeilen sind aber gewoehnlich notwendig -
# es handelt sich um Dateien, die BIND im chroot-Gefaengnis benoetigt, um
# richtig zu funktionieren.
cp $BINDBIN/named-xfer $BINDJAIL/usr/local/libexec
cp $BINDBIN/ndc $BINDJAIL/ndc
cp /etc/localtime $BINDJAIL/etc
mknod $BINDJAIL/dev/null c 1 3
chmod 666 $BINDJAIL/dev/null
mknod $BINDJAIL/dev/random c 1 8
chmod 666 $BINDJAIL/dev/random
```

Beachten Sie, dass Sie */var/named* durch den vollständigen Pfad zu dem Verzeichnis ersetzen sollten, das Sie als Wurzelverzeichnis von root verwenden möchten (viele benutzen dazu */var/named*). Auf vergleichbare Weise müssen Sie in der Zeile *chown -R named* durch den Namen der Gruppe ersetzen, die */named/root* besitzen soll (ich empfehle *named* oder eine andere Gruppe speziell für BIND – d.h. eine Gruppe, die keine echten Benutzer oder andere Anwendungskonten als Mitglieder hat). Darüber hinaus sollten Sie sich vergewissern, dass die Variable `$BINDBIN` den wirklichen Pfad für die Binaries *named* und *ndc* enthält (beide werden normalerweise entweder unter */usr/local/sbin* oder unter */usr/sbin* installiert).

ndc, das *Name Daemon Control*-Interface aus BIND v8, und sein Nachfolger *rndc* in BIND v9 (das *Remote Name Daemon Control*-Interface) können zur Kontrolle von *named* eingesetzt werden: Beide sind Bestandteil des jeweiligen BIND-Source-Codes und der

Binary-Distributionen. Beide Befehle dienen dazu, die Zonen-Dateien neu zu laden. Ich persönlich halte es aber für ebenso einfach, BIND-Startup-Skripten dazu zu verwenden, z.B. `/etc/init.d/named reload`.

Im Folgenden erhalten Sie weitere Hinweise dazu, wie man *ndc* und *rndc* in einem chroot-Gefängnis ausführen kann. Lesen Sie aber auch, wenn Sie weitere Informationen über ihre allgemeinen Einsatzmöglichkeiten benötigen, die jeweilige *ndc(8)*- bzw. *rndc(8)*-Manpage.

Beispiel 6-1 kann mit minimalen Anpassungen als Skript verwendet werden. Achten Sie nur darauf, dass Sie die Werte von `BINDJAIL` und `BINDBIN` bei Bedarf ändern.

Es gibt noch einen weiteren notwendigen Schritt, der allerdings zu distributionsspezifisch ist, als dass wir ihn in Beispiel 6-1 hätten aufnehmen können: Sagen Sie *syslogd*, dass er Log-Daten von *named* von einem Socket im chroot-Gefängnis akzeptieren soll. Natürlich könnten Sie auch *named* stattdessen so konfigurieren, dass er die Logs direkt in Dateien im chroot-Gefängnis schreibt. Die meisten Benutzer finden es jedoch viel bequemer, einige oder alle ihrer named-Ereignisse in das syslog zu schreiben, indem Sie ein *-a*-Flag in ihrem syslog-Startup-Skript ergänzen.

Auf meinem Red Hat-Linux-System wird *syslogd* zum Beispiel durch das Skript */etc/rc.d/init.d/syslog* gestartet. Um *syslogd* mitzuteilen, dass er auf diesem System Log-Daten von einem *named*-Prozess akzeptieren soll, der mit einem anderen Wurzelverzeichnis unter */var/named* läuft, habe ich die folgende Zeile geändert:

```
daemon syslogd -m 0
```

und zwar folgendermaßen:

```
daemon syslogd -m 0 -a /var/named/dev/log
```

Beachten Sie, dass Sie *ndc*, um damit einen *named*-Prozess in einem chroot-Gefängnis zu kontrollieren, zunächst als statisches Binary neu kompilieren müssen. Dabei muss der chroot-Pfad in der Datei *src/bin/ndc/pathnames.h* hinterlegt sein. Führen Sie folgende Schritte aus, um *ndc* auf diese Weise neu zu kompilieren:

1. Wechseln Sie mit *cd* in das Wurzelverzeichnis Ihres BIND-v8-Source-Codes.
2. Editieren Sie die *.settings*, um die Zeile mit den gcc-Optionen zu ändern (die z.B. den String `-CDEBUG=...` enthält) und das Flag `-static` darin zu ergänzen.
3. Editieren Sie die Datei *bin/ndc/pathnames.h*, um die Angabe des Pfads von `/var/run/ndc` in */pfad/zu/chroot_gefaengnis*`/ndc` zu ändern.
4. Kompilieren Sie *ndc* neu, und kopieren Sie das neue Binary in das Wurzelverzeichnis Ihres chroot-Gefängnisses.

Von jetzt an müssen Sie den *chroot*-Befehl benutzen, um *ndc* z.B. wie folgt aufzurufen:

```
chroot /pfad/zu/chroot_kaefig ./ndc [ndc befehl]
```

Einrichten eines chroot-Gefängnisses für BIND v9

Die Vorgehensweise in BIND v9 ist mit der in BIND v8 vergleichbar, wie in Beispiel 6-2 gezeigt wird.

Beispiel 6-2: Einrichten des chroot-Gefängnisses in BIND v9

```
#!/bin/bash
# (Aendern Sie den oben angegebenen Pfad entsprechend, wenn sich Ihr bash-
# Binary an einer anderen Stelle befindet.)
#
# Befehle zur Einrichtung eines BIND-v9-chroot-Gefaengnisses, mit Aenderungen
# uebernommen aus einem Skript von Kyle Amon
# (http://www.gnutec.com/~amonk)
# und aus dem Chroot-BIND-HOWTO (http://www.linuxdoc.org).
# SIE MUESSEN ROOT SEIN, UM DIESES SKRIPT AUSZUFUEHREN!

# Zuerst sollten Sie einige Pfade definieren. BINDJAIL ist das
# Wurzelverzeichnis von BINDs chroot-Gefaengnis.

BINDJAIL = /var/named

# BINDBIN ist das Verzeichnis, in dem sich named, rndc und andere BIND-
# Executables befinden.

BINDBIN = /usr/sbin

# Als Zweites legen Sie das chroot-Gefaengnis mit allen Unterverzeichnissen an:
# Beachten Sie, dass meine Zugriffsrechte restriktiver sind als die im CHROOT-BIND HOWTO --
# es gibt keinen Grund, warum named seine eigenen Dateien ändern sollte

mkdir -m 2750 -p $BINDJAIL/dev $BINDJAIL/etc
mkdir -m 2770 -p $BINDJAIL/var/run
mkdir -m 2770 $BINDJAIL/var/log $BINDJAIL/var/tmp
mkdir -m 2750 $BINDJAIL/master
mkdir -m 2770 $BINDJAIL/slave $BINDJAIL/stubs

# Die folgende Zeile ist unter Debian 3.0 und vielleicht auch bei anderen Distributionen
# notwendig (diese Zeile schadet aber auch nicht, falls sie nicht benötigt wird)
mkdir -m 2770 -p $BINDJAIL/var/cache/bind

# Als Drittes richten Sie unprivilegierte Benutzer und Gruppen fuer named ein.
# (Diese koennte es schon geben, wenn Sie SUSE oder Mandrake verwenden.
# Achten Sie aber darauf, dass der passwd-Eintrag auf
# /bin/false und nicht auf eine echte Shell zeigt.)

echo "named:x:256:" >> /etc/group
echo "named:x:256:256:BIND:$BINDJAIL:/bin/false" \
>>  /etc/passwd

# Geben Sie als Viertes named einige Kontrolle ueber seine eigenen
# Dateien, die Aenderungen unterworfen sind.
chown -R root:named $BINDJAIL
```

Beispiel 6-2: Einrichten des chroot-Gefängnisses in BIND v9 (Fortsetzung)

```
# Kopieren Sie fuenftens einige wichtige Dateien in das chroot-Gefaengnis.

# Die naechste Zeile kann in den meisten Faellen ausgelassen werden.
cp $BINDBIN/named $BINDJAIL

# Die uebrigen Zeilen sind aber gewoehnlich notwendig -
# es handelt sich um Dateien, die BIND im chroot-Gefaengnis benoetigt, um
# richtig zu funktionieren.
cp /etc/localtime $BINDJAIL/etc
mknod $BINDJAIL/dev/null c 1 3
chmod 666 $BINDJAIL/dev/null
mknod $BINDJAIL/dev/random c 1 8
chmod 666 $BINDJAIL/dev/random
```

chroot-Gefängnisse für BIND mit SUSE und Fedora

Fedora und SUSE nehmen Ihnen die ganze Arbeit zum Einrichten eines chroot-Gefängnisses für BIND 9 bereits ab. Fedora verfügt dazu über ein separates RPM namens *bind-chroot*: Es richtet das chroot-Gefängnis ein, setzt alle notwendigen Zugriffsrechte usw. Die Installation von *bind-chroot* erfordert, dass das normale *bind*-Paket bereits installiert ist.

Bei SUSE ist das Ganze noch einfacher: Das Standardpaket für *bind9* verfügt bereits über ein chroot-Gefängnis, was heißt, dass *named* standardmäßig in einem chroot-Gefängnis läuft. Diese überaus vernünftige Default-Installation von BIND kann dem Security Team von SUSE nur positiv angerechnet werden.

named aufrufen

Da wir bislang noch keine Konfigurations- oder Zone-Dateien abgesichert haben, ist es zu früh, *named* zu starten, um Namen aufzulösen. Wo wir aber gerade bei dem Thema sind, *named* in einem chroot-Gefängnis laufen zu lassen, lassen Sie uns darüber sprechen, wie man *named* aufrufen kann, damit er nicht nur in dem Käfig gestartet wird, sondern auch da bleibt. Wir erreichen das durch die folgenden Kommandozeilen-Flags:

- *-u username*
- *-g group name* (nur BIND v8)
- *-t verzeichnis_das_als_Wurzelverzeichnis_dienen_soll*
- *-c /pfad/von/named.conf*

Das erste Flag *-u* bewirkt, dass *named* unter einem bestimmten Benutzernamen (und nicht als root) ausgeführt wird. Wie ich zuvor erwähnt habe, ist es besser, wenn derjenige, der es schafft, den Prozess in seine Gewalt zu bringen, und damit zu *named* wird, ein unprivilegierter Benutzer und nicht root wird. Wird *named* in einem chroot-Gefängnis ausgeführt, fällt es dem Angreifer viel schwerer – falls es nicht sogar unmöglich ist – aus dem chroot-Gefängnis »auszubrechen«, wenn *named* nicht unter root läuft.

BIND v9 unterstützt das Flag `-u` nur auf Linux-Systemen mit einem Kernel ab Version 2.3.99-pre3 (in der Praxis also ab Version 2.4). Falls Sie aus irgendeinem Grund noch immer einen 2.2er Kernel einsetzen, bedeutet das für Sie, dass Sie BIND v9 mit keinem anderen Benutzer als *root* ausführen können.

Allerdings gibt es keinen Grund, warum Sie noch immer Linux 2.2 einsetzen müssten. Zum Zeitpunkt, da diese Zeilen geschrieben wurden (Oktober 2004), ist Linux 2.4 bereits vier Jahre lang optimiert und verbessert worden, so dass es sich hinsichtlich seiner Stabilität und Sicherheit bereits mehr als bewiesen hat. Deshalb sollten Sie auf Ihren Linux-Bastion-Servern unbedingt einen 2.4er Kernel einsetzen.

Die *-g*-Option von BIND v8 bewirkt, dass *named* unter dem angegebenen Gruppennamen läuft. Diese Option wurde in BIND v9 fallen gelassen, weil es ungewöhnlich wäre, *named*, der die Rechte eines bestimmten Benutzers hat, mit anderen Gruppenrechten auszuführen als mit denen dieses bestimmten Benutzers. Mit anderen Worten wird *named* in BIND v9 später unter der ID der Gruppe ausgeführt, die Sie ausgewählt haben, als Sie das unprivilegierte Benutzerkonto für *named* angelegt haben.

Die *-t*-Option sorgt dafür, dass alle Pfade, die von *named* referenziert werden, an das geänderte Wurzelverzeichnis (chroot) angepasst werden. Beachten Sie bei der Verlegung von *named* in ein chroot-Gefängnis, dass dieses neue Wurzelverzeichnis sogar schon angewendet wird, bevor die Datei *named.conf* eingelesen wird. Aus diesem Grund müssen Sie zusätzlich die Option `-c` verwenden, um den Pfad der *named.conf* explizit anzugeben.

Anders ausgedrückt, sucht *named* (v8), wenn Sie es mit dem Befehl

```
named -u named -g wheel -t /var/named -c /etc/named.conf
```

aufrufen, nach */var/named/etc/named.conf* und nicht nach */etc/named.conf*.

Seltsamerweise muss man das *-c*-Flag nicht angeben, wenn *named* nicht in einer root-Umgebung ausgeführt wird (und *named.conf* in */etc* bleibt); man *muss -c* angeben, wenn *named* in einer chroot-Umgebung laufen soll (und zwar unabhängig davon, wo Sie *named.conf* unterbringen). Eigentlich sollte man erwarten, dass *named* automatisch unter */chroot/pfad/etc* nach *named.conf* sucht, aber aus irgendwelchen Gründen muss man ihm explizit mitteilen, dass er in */etc* nachschauen muss, wenn das Wurzelverzeichnis nicht wirklich */* ist.

Beim *named9*-Paket von Debian 3.0 liegt die Konfigurationsdatei unter */etc/bind/named.conf*. Wenn Sie nun aber die Konfigurationsdateien Ihres unter Debian eingerichteten chroot-Gefängnisses in *$BINDJAIL/etc* statt in *$BINDJAIL/etc/bind* ablegen, müssen Sie den Parameter `-c` immer noch mit `-c /etc/named.conf` übergeben.

Diese Flags haben gemeinsam (sofern sie richtig verwendet werden) die Auswirkung, dass die Zugriffsrechte, die Umgebung und sogar das Dateisystem von *named* stark eingegrenzt sind. Wenn ein unberechtigter Benutzer *named* irgendwie in seine Gewalt brin-

gen sollte, würde er die Berechtigungen eines unprivilegierten Benutzerkontos erhalten. Außerdem würde er sogar noch weniger vom Dateisystem des Servers sehen als ein normaler Benutzer: Aus der Sicht von *named* würden Verzeichnisse, die sich auf einer höheren Ebene im Verzeichnisbaum befinden als sein chroot-Verzeichnis, nicht einmal existieren.

Absichern von named.conf

Das Ausführen von *named* in einer Gummizelle ist an sich schon paranoid und bewundernswert genug. Aber das ist nur der Anfang! Die Konfigurationsdatei von BIND, *named.conf*, beinhaltet eine Vielzahl von Parametern, die es Ihnen erlauben, *named* bis ins kleinste Detail zu kontrollieren.

Betrachten Sie zum Beispiel die Datei *named.conf*, die in Beispiel 6-3 aufgeführt wird:

Beispiel 6-3: Eine exemplarische named.conf-Datei für einen externen DNS-Server

```
# Uebrigens koennen Kommentare in der Datei named.conf so aussehen ...
// oder so ...
/* oder so. */
acl trustedslaves { 192.168.20.202; 192.168.10.30};
acl kerle { 10.10.1.17; 10.10.2.0/24; };
acl keine_kerle { localhost; !kerle; };

options {
    directory "/";
    listen-on { 192.168.100.254; };
    recursion no; fetch-glue no;
    allow-transfer { trustedslaves; };
};

logging {
    channel seclog {
        file "var/log/sec.log" versions 5 size 1m;
        print-time yes; print-category yes;
    };
    category xfer-out { seclog; };
    category panic { seclog; };
    category security { seclog; };
    category insist { seclog; };
    category response-checks { seclog; };
};

zone "coolfroods.ORG" {
    type master;
    file "master/coolfroods.hosts";
};

zone "0.0.127.in-addr.arpa" {
    type master;
    file "master/0.0.27.rev";
};
```

Beispiel 6-3: Eine exemplarische named.conf-Datei für einen externen DNS-Server (Fortsetzung)

```
zone "100.168.192.in-addr.arpa" {
    type master;
    file "master/100.168.192.rev";
};
```

Bei dem hypothetischen Server, dessen Konfigurationsdatei hier dargestellt ist, handelt es sich um einen externen DNS-Server. Da seine Aufgabe darin besteht, der externen Welt Informationen über die öffentlich zugänglichen Server von *coolfroods.org* zu geben, wurde er so konfiguriert, dass Rekursion nicht zulässig ist. Tatsächlich enthält die Datei keinen ».«-Zonen-Eintrag (d.h. keinen Verweis auf eine *hints*-Datei). Er kann also nichts über Rechner erfahren, die nicht in seinen eigenen lokalen Zonen-Dateien beschrieben sind. Übertragungen seiner lokalen Zonen-Datenbanken werden durch die Festlegung der entsprechenden IP-Adressen auf eine Gruppe von vertrauenswürdigen Slave-Servern beschränkt. Der Logging-Dienst wurde für verschiedene Ereignistypen aktiviert.

Wie machen wir also diese und noch hübschere Sachen mit *named.conf*?

Grundsätzlich ist *named.conf* aus BIND v9 rückwärtskompatibel. Somit beziehen sich die folgenden Hinweise auf beide Versionen, es sei denn, es ist explizit anders angegeben.

acl{}-Abschnitte

Zugriffskontrolllisten (Access Control Lists, ACLs) sind, auch wenn sie optional sind, eine bequeme Möglichkeit, Gruppen von IP-Adressen und Netzwerken zu kennzeichnen. Da wir vorsichtig sind, möchten wir definitiv bestimmte Aktionen und Daten auf ausgewählte IP-Adressen beschränken.

Eine ACL kann an einer beliebigen Stelle innerhalb von *named.conf* deklariert werden. Da die Datei aber von oben nach unten geparst wird, muss jede ACL vor ihrer ersten Instanziierung in einem Parameter deklariert werden. Somit ist es sinnvoll, die ACL-Definitionen an oberster Stelle in der *named.conf* aufzuführen.

Das Format eines *acl{}*-Eintrags wird in Beispiel 6-4 vorgestellt.

Beispiel 6-4: Format von Zugriffskontrolllisten

```
acl acl_name { IPadresse; Netzwerkadresse; acl_name; usw. };
```

Die Elementliste in den geschweiften Klammern kann eine beliebige Kombination folgender Einträge enthalten:

IP-Host-Adressen
: Im Format *x.x.x.x* (z.B. 192.168.3.1)

IP-Netzwerkadressen (in der BIND-Dokumentation als *IP Prefixes* bezeichnet)
: Im CIDR-Format *x.x.x.x/y* (z.B. 172.33.0.0/16)

Namen von ACLs
: Namen von ACLs, die in anderen *acl{}*-Abschnitten definiert wurden, inklusive der vordefinierten ACLs »any«, »none«, »localhost« und »localnets«.

Key-Namen
: Namen von Schlüsseln, die zuvor in der Datei *named.conf* in *key{}*-Abschnitten eingeführt worden sind.

Jedes dieser Elemente kann durch ein vorangestelltes »!« negiert werden, d.h., »!192.168.3.1« bedeutet »nicht 192.168.3.1«. Achten Sie nur darauf, dass Sie speziellere Elemente vor umfassenderen Elementen verwenden. Die ACL-Elementliste wird nämlich von links nach rechts geparst. Ein Element, das zum Beispiel »alle Adressen im Netzwerk 10.0.0.0/8 außer 10.1.2.3« angibt, könnte folgendermaßen aussehen:

```
{!10.1.2.3; 10.0.0.0/8; }
```

aber *nicht* so:

```
{ 10.0.0.0/8; !10.1.2.3; }
```

Jedes Element, das in den geschweiften Klammern aufgeführt wird, muss mit einem Semikolon enden. Das gilt sogar dann, wenn nur ein einziges Element in den Klammern steht.

Dieser Auszug aus Beispiel 6-3 zeigt ACLs mit verschiedenartigen Elementen:

```
acl kerle { 10.10.1.17; 10.10.2.0/24; };
acl keine_kerle { localhost; !kerle; };
```

Jedes Mal, wenn die Datei *named.conf* in diesem Beispiel gelesen wird, werden die Wörter *kerle* und *keine_kerle* durch den Inhalt der jeweiligen ACL-Elementliste ersetzt.

Globale Optionen: Der options{}-Abschnitt

In der Konfigurationsdatei folgt als Nächstes eine Liste von globalen Optionen. Einige der Optionen, die in diesem Abschnitt zulässig sind, können auch in Zone-Abschnitten verwendet werden. Berücksichtigen Sie aber, dass der Parametereintrag im Zone-Abschnitt höher gewertet wird als der im *options{}*-Abschnitt, falls ein bestimmter Eintrag in beiden Abschnitten erscheint. Anders ausgedrückt: Die Werte werden von den Parametern in den Zone-Abschnitten als Ausnahmen von den globalen Werten verstanden.

Im Folgenden finden Sie einige nützliche Parameter, die in *options{}*-Abschnitten verwendet werden können:

`listen-on` *`[port#]`* `{` *`Liste von IPs mit lokalem Interface`* `; };`
: Legen Sie das oder die Interface(s) fest, auf dem bzw. denen auf DNS-Abfragen und Zonen-Transfer-Anfragen gelauscht werden soll. Diese und alle anderen Adresslisten, die in {}-Klammern stehen, müssen durch Semikola getrennt werden. Die Portnummer ist optional (der Standard ist 53).

`listen-on-v6 [port#] { any | none ; };`
: Geben Sie an, ob auf allen Interfaces mit einer IPv6-Adresse gelauscht werden soll (nur BIND v9).

`allow-recursion { Liste von IP-Adressen/-Netzwerken; };`
: Führen Sie eine rekursive Abfrage für eine angegebene IP-Liste durch, die einfach nur aus dem Wort `none`; bestehen kann.

`allow-transfer { Liste von IP-Adressen/-Netzwerken oder none ; };`
: Geben Sie an, welche Adressen und/oder Netzwerke Zonen-Transfers bekommen dürfen, wenn sie welche anfordern.

`allow-query { IP/acl-list ; };`
: Einfache DNS-Abfragen von diesen IP-Adressen/ACLs/Netzwerken (oder `none`) sind erlaubt.

`version "[nachricht]";`
: Geben Sie Ihre Versionsnummer an. Es gibt keinen legitimen Grund, warum irgendjemand außer Ihrem Netzwerkadministrator Ihre BIND-Versionsnummer kennen sollte. Manche verwenden diesen Parameter, um auf Versionsanfragen mit irreführenden oder witzigen Informationen zu reagieren.

`recursion [yes | no];`
: Schalten Sie die Rekursion global an oder aus. Ist keine Rekursion erlaubt, sollten Sie auch fetch-glue auf `no` setzen (vgl. den nächsten Eintrag in dieser Liste).

`fetch-glue [yes | no];`
: In BIND v9 zwar erlaubt, aber überflüssig. Wenn Sie diesen Parameter auf `no` setzen, wird verhindert, dass Ihr Nameserver die IPs von anderen Nameservern, mit denen er in Berührung kommt, auflöst und zwischenspeichert. Auch wenn das Glue-Fetching für besser lesbare Log-Einträge sorgt, ist es in den vergangenen Jahren häufiger geschickten Cache-Poisoning-Angriffen ausgesetzt gewesen. In BIND v8 werden derartige Einträge im Verlauf von normalen Abfragen angelegt, es sei denn, Sie deaktivieren das hier. In BIND v9 werden unabhängig davon, was Sie hier angeben, niemals solche Einträge erzeugt.

Logging

Zusätzlich zu den globalen Optionen sollten Sie vielleicht einige Logging-Regeln einrichten. Standardmäßig loggt *named* nicht mehr als ein paar wenige Startup-Nachrichten (wie zum Beispiel Fehlermeldungen und geladene Zonen), die an den syslog-Daemon geschickt werden (der sie wiederum in */var/log/messages* oder in irgendeine andere Datei schreibt). Um sicherheitsrelevante Vorkommnisse, Zonen-Transfers usw. zu erfassen, müssen Sie einen *logging{}*-Abschnitt in der Konfigurationsdatei *named.conf* ergänzen.

Der *logging{}*-Abschnitt besteht aus zwei Teilen: einer oder mehreren *channel{}*-Definitionen, die den oder die Orte angeben, an die Log-Nachrichten geschickt werden sollen, gefolgt von einem oder mehreren *category{}*-Abschnitten, die jeden Ereignistyp, den Sie loggen möchten, einem Channel zuordnen. Channels verweisen entweder auf Dateien

oder auf den lokalen syslog-Daemon. Kategorien müssen aus einer Menge von vordefinierten Ereignistypen ausgewählt werden.

Das Format von Channel-Definitionen ist in Beispiel 6-5 dargestellt:

Beispiel 6-5: Syntax von Log-Channeln

```
channel channel-name {
          dateiname [ datei-optionen-liste ] | syslog syslog-einrichtung | null ;
          [ print-time yes|no; ]
          [ print-category yes|no; ]
          [ print-severity yes|no; ]
          [ severity schwere-grad; ]
};
```

Die über *dateiname* referenzierte Datei wird standardmäßig im Arbeitsverzeichnis von *named* abgelegt. Es kann aber stattdessen auch ein vollständiger Pfad angegeben werden. (Falls *named* in einer chroot-Umgebung läuft, wird der Pfad als relativ zum chroot-Wurzelverzeichnis angesehen.) Mithilfe der Optionen *size* und *versions* können Sie festlegen, wie groß die Datei höchstens werden darf bzw. wie viele alte Kopien jeweils parallel aufbewahrt werden sollen.

Beachten Sie aber, dass das Datei-Rotationsverfahren nicht annähernd so elegant ist wie bei *syslogd*. Sobald eine Datei die angegebene Größe erreicht hat, hört *named* einfach auf, in diese Datei zu schreiben (anstatt sie unter einem anderen Namen abzuspeichern und eine neue Datei zu erzeugen, wie wir das von *syslogd* kennen). Die Datei wird so lange nicht »aussortiert«, bis *named* das nächste Mal gestartet wird. Dann geschieht nämlich, was die *versions*-Option eigentlich spezifiziert: Sie gibt an, wie viele Kopien einer Datei ausgehend von der Zahl der Aufrufe von *named* aufbewahrt werden sollen. Das Aussortieren hängt nicht mit der Dateigröße zusammen. In Kapitel 12 finden Sie Informationen über bessere Methoden zum Rotieren von Log-Dateien.

Wenn Sie anstelle von *dateiname* `syslog` angeben und einen *`syslog-type`* festlegen, schickt der Channel Nachrichten an den lokalen *syslogd*-Prozess (oder an *syslog-ng*, sofern vorhanden). Dazu verwendet er die Einrichtung, die als *`syslog-einrichtung`* definiert wurde. (Eine Liste dieser Einrichtungen mit Beschreibungen finden Sie in Kapitel 12.) Standardmäßig benutzt *named* die *daemon*-Einrichtung für die meisten Meldungen nach dem Startup.

Die Optionen *print-time*, *print-category* und *print-severity* geben an, ob zusätzlich pro Ereignis auch Uhrzeit und Datum sowie die Kategorie bzw. der Schweregrad vor jedem Log-Eintrag festgehalten werden sollen. Die Reihenfolge dieser Optionen spielt dabei keine Rolle. Sie werden unabhängig davon folgendermaßen hintereinander aufgeführt: *uhrzeit/datum*, *kategorie*, *schweregrad*. Es lohnt sich in *syslog*-Channels nicht, die Zeit, zu der ein Eintrag generiert wurde, explizit festzuhalten, weil *syslogd* automatisch alle Einträge um die Angabe der Uhrzeit erweitert.

Schließlich erlaubt Ihnen die *severity*-Option, den Schweregrad zu bestimmen, ab dem *named*-Meldungen an den Channel geschickt werden müssen. Bei *`schwere-grad`* kann es

sich um eine der syslog-»Prioritäten« (die ebenfalls in Kapitel 12 beschrieben werden) mit Ausnahme von `debug` handeln. Die Priorität `debug` kann nur gefolgt von einem numerischen Argument zwischen `1` und `10` angegeben werden, das den Debug-Level bestimmt. Als *`schwere-grad`* wird standardmäßig der Wert `info` verwendet.

Im Folgenden sehen Sie einen weiteren Auszug aus Beispiel 6-3 vom Anfang des Kapitels:

```
logging {
    channel seclog {
        file "var/log/sec.log" versions 3 size 1m;
        print-time yes; print-category yes;
    };
```

Aufgrund dieser *logging{}*-Anweisung werden Ereignistypen, die an den *seclog*-Channel weitergeleitet werden, in eine Log-Datei namens */var/log/sec.log* geschrieben (der führende / zu Beginn des Pfads wird implizit vorausgesetzt, weil das Verzeichnis / weiter oben in diesem Beispiel als das Arbeitsverzeichnis von *named* definiert wurde). Wenn diese Datei eine Größe von mehr als 1 MByte erreicht, hört *named* auf, Log-Daten an diesen Channel und somit an diese Datei zu schicken. Jedes Mal, wenn *named* gestartet wird, wird die aktuelle Version dieser Datei umbenannt – z.B. *sec.log.1* in *sec.log.2*, *sec.log.0* in *sec.log.1* und *sec.log* in *sec.log.0*. Die Log-Einträge, die in diese Datei geschrieben werden, beginnen mit der Angabe von Datum und Kategorie, der Schweregrad wird aber ausgelassen.

Die Festlegung von Kategorien ist viel einfacher (vgl. Beispiel 6-6):

Beispiel 6-6: Syntax von Log-Kategorien

```
category kategorie-name { channel-liste ; };
```

Wie die ACL-Elementlisten ist auch die *channel-liste* eine durch Semikola unterteilte Liste und muss einen oder mehrere Channels enthalten, die vorher in einer *channel{}*-Anweisung definiert worden sein müssen. (Wenn Sie möchten, können Sie die den verschiedenen Kategorien zugeordneten Meldungen je nach ihrer Kategorie mehreren Channels zuordnen.) Tabelle 6-1 beinhaltet eine Liste von Kategorien, die, unter Sicherheitsaspekten betrachtet, besonders wichtig sind. Eine vollständige Beschreibung aller unterstützten Kategorien finden Sie im *BIND v8 Operator's Guide* (BOG) oder im *BIND 9 Administrator Reference Manual* (ARM).

Tabelle 6-1: Log-Kategorien, die in Zusammenhang mit Sicherheit stehen

Bezeichnung der Kategorie	Unterstützung in BIND v8	Unterstützung in BIND v9	Inhalt der Meldungen
`default`	✓	✓	Nachrichten aus allen Kategorien, die keinem Channel zugeordnet sind; wenn für `default` keine Channels festgelegt werden, werden die `default`-Nachrichten an die integrierten Channels `default_syslog` und `default_debug` geschickt.
`config`	✓	✓	Ergebnisse des Parsens und Ausführens von `named.conf`
`security`	✓	✓	Fehlgeschlagene und erfolgreiche Transaktionen

Tabelle 6-1: Log-Kategorien, die in Zusammenhang mit Sicherheit stehen (Fortsetzung)

Bezeichnung der Kategorie	Unterstützung in BIND v8	Unterstützung in BIND v9	Inhalt der Meldungen
`xfer-in`	✓	✓	Eingehende Zonen-Transfers (d.h. solche, die von lokalen Zonen-Anfragen hervorgerufen werden)
`xfer-out`	✓	✓	Ausgehende Zonen-Transfers (d.h. solche, die von externen Zonen-Anfragen ausgelöst werden)
`load`	✓		Ladevorgang von Zonen-Dateien
`os`	✓		Probleme mit dem Betriebssystem
`insist`	✓		Fehlschlagen von internen Konsistenzprüfungen
`panic`	✓		Unerwartetes Herunterfahren (Crashs)
`maintenance`	✓		Routinemäßige Aktivitäten zur eigenen Wartung
`general`		✓	Keiner Kategorie zugeordnete Nachrichten
`client`		✓	Client-Anfragen

Die *named.conf*-Optionen, mit denen wir uns bisher beschäftigt haben, beziehen sich auf alle Nameserver inklusive derer, die nur cachen und keine Kontrolle über irgendwelche Zonen haben (d.h. die weder Master noch Slave oder nicht einmal Stub für irgendetwas sind) und daher naturgemäß einfacher und leichter abgesichert werden können als andere Arten von DNS-Servern. Einige der verbleibenden Konfigurationsmöglichkeiten in der Datei *named.conf* beziehen sich auf Server, die nur cachen.

Die Hauptsicherheitslücke von Servern, die nur cachen, besteht im Cache-Poisoning. Die beste Verteidigung dagegen ist der rigorose Einsatz der globalen Optionen *allow-recursion{}*, *allow-query{}*, *fetch-glue* und *recursion* (in Ergänzung zur Verwendung der neusten Version Ihrer DNS-Software). Auf einem Server, der nur cacht, muss *recursion* auf *yes* gesetzt werden, weil seine Hauptaufgabe in der Rekursion besteht. Schränken Sie also mithilfe der *allow-recursion{}*-Anweisung die Zahl der Rechner ein, die rekursive Abfragen auslösen dürfen.

zone{}-Abschnitte

Der letzte mögliche *named.conf*-Abschnitt, mit dem wir uns hier beschäftigen, ist der *zone{}*-Abschnitt. Wie bei *options{}* gibt es viel mehr Parameter als die, die hier beschrieben werden. Weitere Informationen über diese Parameter finden Sie in den Handbüchern BOG und ARM.

Die folgenden drei Parameter sind besonders nützlich, um die Sicherheit des zonenübergreifenden Datenaustauschs zu verbessern:

`allow-update { `*`element-liste`*` ; };`

Erlaube dynamische DNS-Aktualisierungen von den Rechnern/Netzwerken, die in der Elementliste angegeben sind. Die Elementliste kann eine beliebige Kombination aus IP-Adressen, IP-Netzwerken oder ACL-Namen enthalten. (Alle referenzierten ACLs müssen an anderer Stelle in der Datei *named.conf* definiert sein.)

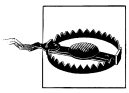
`allow-query { element-liste ; };`
: Erlaube DNS-Abfragen von diesen Entities.

`allow-transfer { element-liste ; };`
: Antworte auf Anfragen nach Zonen-Transfers von diesen Entities.

Alle drei Parameter können in *options{}*-Abschnitten und *zone{}*-Abschnitten angegeben werden. Sollten sie in beiden Abschnitten aufgeführt sein, überschreiben die zonenspezifischen Einstellungen die globalen Einstellungen.

Split-DNS und BIND v9

Zu Beginn dieses Kapitels habe ich die verbesserte Unterstützung von Split-DNS durch BIND v9 erwähnt. Die verbesserte Unterstützung wird durch die neue *view{}*-Anweisung hervorgerufen. Sie kann in der Datei *named.conf* dazu verwendet werden, jedem Zonen-Namen mehrere Zonen-Dateien zuzuordnen. Auf diese Weise können verschiedene Clients unterschiedlich behandelt werden. Zum Beispiel erhalten externe Benutzer andere Antworten zu einem bestimmten Domain-Namen als interne Benutzer.



Wenn Sie die *view{}*-Funktion in Bezug auf eine Zone einsetzen, müssen Sie sie auf alle Zonen anwenden. Es müssen also *alle zone{}*-Anweisungen in einer *view{}*-Anweisung eingebettet sein. Allein stehende (nicht eingebettete) *zone{}*-Anweisungen dürfen nur benutzt werden, wenn überhaupt keine *view{}*-Anweisung vorhanden ist.

Die Syntax von *view{}*-Anweisungen wird in Beispiel 6-7 dargestellt.

Beispiel 6-7: Syntax von Zonen-Ansichten (zone-views)

```
view "view-name" {
    match-clients { uebereinstimmungsliste; };
    recursion yes|no;
    zone "domain.name" {
           // hier stehen die ueblichen BIND 8/9 zone{}-Inhalte
    };
    // Es koennen auch zusaetzliche Zonen fuer diese Ansicht (view) definiert werden.
};
```

Die unter *match-clients* aufgeführte Liste, die auf passende Einträge geprüft wird, kann die gleichen Formate und vordefinierten Kennzeichen enthalten wie die Elementliste, die ich in diesem Kapitel bereits im Absatz »acl{}-Abschnitte« beschrieben habe. Eingebettete *zone{}*-Anweisungen unterscheiden sich nicht von normalen, allein stehenden *zone{}*-Angaben.

Beispiel 6-8 stellt zwei Ansichten dar, die für eine Split-DNS-Situation definiert wurden. Danach werden Anfragen von internen Benutzern mit vollständigen Zonen-Informationen beantwortet. Hingegen bekommen externe Benutzer Informationen aus einer Zonen-Datei, die nur eine Teilmenge der Informationen enthält. Interne Benutzer kön-

nen darüber hinaus Informationen über eine interne Zone, *intranet.unsereorg.org*, erfragen, über die der DNS-Server *keine* Informationen an externe Anfragen erteilt.

Beispiel 6-8: Einige Beispiel-Ansichten

```
view "intern" {
    // Bei unseren internen Rechnern handelt es sich um:
    match-clients { 192.168.100.0/24; };
    // ... fuer die werden auch rekursive Abfragen ausgefuehrt ...
    recursion yes;
    // Die folgenden Zonen halten wir fuer die internen Benutzer bereit:
    zone "unsereorg.ORG" {
           type master;
           file "master/unsereorg_int.hosts";
    };
    // Folgende Unter-Domain kann von externen Benutzern nicht durchsucht werden:
    zone "intranet.unsereorg.ORG" {
           type master;
           file "master/intranet.unsereorg.hosts";
    };
};

view "extern" {
    //Client-Ansicht von "nichts von dem, was oben aufgefuehrt ist".
    match-clients { any; };
    // Wir fuehren keine rekursiven Abfragen fuer die breite Oeffentlichkeit durch:
    recursion no;
    // Externe Anfragen werden mit den Informationen aus einer abgespeckten
    // Zonen-Datei beantwortet:
    zone "unsereorg.ORG" {
           type master;
           file "master/unsereorg_ext.hosts";
    };
};
```

Die Kommentare im Beispiel 6-8 weisen bereits darauf hin, dass die *view{}*-Definition von oben nach unten geparst wird: Wenn die IP-Adresse eines Benutzers mit dem View verglichen wird, wird die Liste so lange weiter nach unten durchsucht, bis ein übereinstimmender Eintrag gefunden wird.

Sicherheit von Zonen-Dateien

Unserer sicherer DNS-Dienst ist in seiner Gummizelle eingesperrt und achtet sehr genau darauf, wem er welche Informationen gibt. Anders ausgedrückt, er nimmt allmählich die gewünschte Form an. Was ist aber eigentlich mit den Zonen-Datenbanken?

Die gute Nachricht zu diesem Thema ist, dass wir hier wesentlich weniger Einstellungsmöglichkeiten haben als in der Konfigurationsdatei *named.conf*. Also ist auch weniger zu tun. Die schlechte Nachricht ist, dass es mindestens eine Art von Quelleintrag gibt, die sowohl überflüssig als auch gefährlich ist. Sicherheitsbewusste Administratoren sollten diesen Eintrag daher vermeiden.

Beispiel 6-9 zeigt eine beispielhafte Zonen-Datei für die hypothetische Domain *holzkoepfe.com*.

Beispiel 6-9: Beispielhafte Zonen-Datei

```
$TTL 86400
// Anmerkung: Eine globale/standardmaessige TTL muss oben angegeben werden.
// BIND v8 hat das nicht ueberprueft.
// Das ist aber bei BIND v9 der Fall.
@ IN SOA  laus.holzkoepfe.com. hostmaster.holzkoepfe.com.  (
            2000060215      ; serial
            10800           ; refresh (3H)
            1800            ; retry (30m)
            120960          ; expiry (2w)
            43200 )         ; RR TTL (12H)
         IN    NS      ns.anderedomain.com.
         IN    NS      laus.holzkoepfe.com.
         IN    MX 5    laus.holzkoepfe.com.
blorp    IN    A       10.13.13.4
laus     IN    A       10.13.13.252
laus     IN    HINFO   MS Windows NT 3.51, SP1
@        IN    RP      john.smith.holzkoepfe.com. dumm.holzkoepfe.com.
dumm     IN    TXT     "John Smith, 612/231-0000"
```

Der erste relevante Eintrag in diesem Beispiel ist der Start of Authority-(SOA-)Eintrag (Start des Einflussgebiets). In Beispiel 6-9 wird die Seriennummer (fortlaufende Nummer) im Format *jjjjmmtt##* angegeben. Das ist sowohl bequem als auch gut für die Sicherheit, weil dadurch die Wahrscheinlichkeit verringert wird, dass irrtümlich eine alte (nicht mehr gültige) Zonen-Datei geladen wird. Die laufende Nummer (2000060215 in Beispiel 6-9) dient sowohl als Index als auch als Zeitstempel.

Das Aktualisierungsintervall (refresh) ist mit 10.800 Sekunden (drei Stunden) angegeben. Andere gebräuchliche Werte für diese Einstellung sind zum Beispiel 3.600 Sekunden (eine Stunde) und 86.400 Sekunden (ein Tag). Je kürzer das Aktualisierungsintervall gewählt wird, desto kürzer ist die Zeit, in der Änderungen an den Zonen-Einstellungen weiterverbreitet werden. Allerdings nehmen der Netzwerk-Traffic und die Systemaktivität auch entsprechend zu.

Die Ablaufdauer (expiry) ist mit zwei Wochen angegeben. Es handelt sich dabei um die Dauer, für die die Zonen-Datei noch als gültig angesehen wird, falls der Zonen-Master aufhören sollte, auf Aktualisierungsanfragen zu antworten. Einerseits gewährleistet die Angabe einer langen Ablaufdauer, dass die Slaves mit zwischengespeicherten Zonen-Informationen weiterarbeiten und die Domain erreichbar bleibt, falls der Master-Server über einen längeren Zeitraum von Denial-of-Service-Angriffen bombardiert wird. (Vermutlich wird der Haupt-DNS-Server dann jedoch nicht weiterarbeiten.) Andererseits könnten sich sogar in so einem Fall die Zonen-Informationen ändern. Außerdem verursachen veraltete Daten in der Regel mehr Unheil als gar keine Daten.

Wie das Aktualisierungsintervall sollte auch das Time to live-Interval (TTL; Lebenszeit) kurz genug sein, um die Bekanntmachung von aktualisierten Einträgen innerhalb eines

vernünftigen Zeitrahmens zu ermöglichen. Auf der anderen Seite sollte es lang genug sein, um das Netzwerk nicht unnötig zu belasten. Die TTL legt fest, wie lange die Quelleinträge zu einzelnen Zonen im Cache von anderen Nameservern, die sie abgefragt haben, bleiben dürfen.

Unsere andere Sorge in Bezug auf diese Zonen-Datei gilt der Frage, wie man die unnötige Bekanntgabe von Informationen reduzieren kann. Zunächst möchten wir die Zahl der Adresseinträge (`A`-Einträge) und Aliase (`CNAME`-Einträge) generell verringern, damit nur die Rechner darin erscheinen, die vorhanden sein müssen.

Wenn überhaupt, sollten wir die Einträge zum Verantwortlichen (Responsible Person; `RP`) und die `TXT`-Einträge nur nach gründlicher Abwägung der Risiken benutzen. Wir sollten aber niemals irgendwelche sinnvollen Informationen in einem `HINFO`-Eintrag angeben. `HINFO` ist ein Relikt aus alten Zeiten: `HINFO`-Einträge werden benutzt, um Angaben zum Betriebssystem, zur Version des Systems und sogar zur Hardware-Konfiguration des Rechners zu machen, auf den sie sich beziehen.

Zu der Zeit, als der größte Teil von Internetknoten im universitären Bereich und in anderen offenen Umgebungen angesiedelt waren (und als Computer noch exotisch und neu waren), schien es vernünftig zu sein, seinen Benutzern diese Informationen zu geben. Heutzutage gibt es für `HINFO` aber keine vernünftigen Einsatzmöglichkeiten auf öffentlichen Servern mehr, Sie können damit höchstens Verwirrung stiften, z.B. potenzielle Angreifer bewusst mit falschen Informationen versorgen). Um es kurz zu machen: Benutzen Sie keine `HINFO`-Einträge!

`RP` wird dazu verwendet, die E-Mail-Adresse von demjenigen anzugeben, der die Domain administriert. Das Beste ist es, hier eine Adresse anzugeben, die so uninteressant wie nur möglich ist, z.B. *information@wuzza.com* oder *hostmaster@wuzza.com*. Auf vergleichbare Weise dienten `TXT`-Einträge früher dazu, weitere Kontaktmöglichkeiten anzugeben (Telefonnummern usw.). Heute sollten diese Einträge allerdings auf die absolut notwendigen Informationen begrenzt oder, noch besser, komplett ausgelassen werden.

Wenn wir uns jetzt Beispiel 6-5 noch einmal anschauen, sehen wir, dass die letzten paar Einträge im besten Fall überflüssig und im schlechtesten Fall eine Goldmine für einen Cracker sind. Ich wiederhole: Wenn Sie meinen, dass Sie `RP` und `TXT` angeben müssen, sollten Sie die damit verbundenen Risiken sorgfältig abwägen. `HINFO` sollten Sie unter gar keinen Umständen verwenden.

Weitergehende BIND-Sicherheit: TSIGS und DNSSEC

Die meisten der Sicherheitskontrollen, mit denen wir uns bislang in diesem Kapitel beschäftigt haben, standen im Zusammenhang mit der Begrenzung der Daten, die der DNS-Server bereitstellt, und damit, wann er sie bereitstellt. Wie sieht es aber mit der Authentifizierung aus? Was hindert zum Beispiel einen Angreifer daran, seinen Rechner als einen vertrauenswürdigen Master-Server in Ihrer Domain auszugeben und falsche Zonen-Dateien auf Ihre Slaves hochzuladen? Und was hindert ihn daran, dazu gefälschte

(gespoofte) Pakete (d.h. Pakete mit falschen IP-Quelladressen) einzusetzen, um die ACLs zu umgehen? Und wie sieht es mit der Datenintegrität aus: Was sollte einen solchen Angreifer davon abhalten, »Man in the middle«-Angriffe zu benutzen, um die Inhalte von Ihren echten DNS-Abfragen und -Antworten zu ändern?

Glücklicherweise können Transaktionssignaturen (TSIGs) Authentifizierung und ein gewisses Maß an Schutz für die Datenintegrität während der Transaktionen zwischen DNS-Servern gewährleisten. Unglücklicherweise garantieren TSIGs aber nicht, dass die DNS-Informationen vor der Übertragung nicht kompromittiert worden sind. Wenn es einem Angreifer gelingt, »root« auf einem DNS-Server zu werden oder sich eine Kopie von der TSIG des Servers zu beschaffen, kann er falsche DNS-Informationen signieren.

Allerdings beschäftigt sich die IETF nun schon seit einigen Jahren mit DNS-Sicherheitserweiterungen (DNS Security Extensions). (Dabei handelt es sich um die DNSSEC, die in RFC 2535 und anderen Dokumenten von der dnsext-Arbeitsgruppe der IETF beschrieben sind.) Durch diese Erweiterungen zu DNS (die hauptsächlich aus neuen Quelleinträgen für Schlüssel und Signaturen bestehen) besteht jetzt die Möglichkeit, DNS-Einträge kryptographisch zu signieren und die Einträge selbst zu verifizieren. Zusammen verwendet sollten die TSIG- und die DNSSEC-Funktionalität DNS im Internet viel vertrauenswürdiger machen.

Die DNSSEC-Spezifikation ist aber noch nicht endgültig fertig gestellt. Obwohl DNSSEC in Bind v9 weitgehend implementiert ist, vermittelt es den Eindruck, ein bisschen kompliziert und unhandlich zu sein. Da die TSIG-Funktionalität von BIND viel ausgereifter und einfacher zu verwenden ist und sowohl von BIND v8.2 und höher als auch von BIND v9 unterstützt wird, beenden wir unsere Ausführungen zu BIND mit einigen Hinweisen zum Einsatz von TSIGs.

Wenn Sie sich für die höchste Ausformung von DNS-Sicherheit mit DNSSEC interessieren (hoffentlich tun das viele, um damit zu seiner weiteren Entwicklung und weit verbreiteten Anwendung beizutragen), kann ich Ihnen Kapitel 11 aus dem Buch *DNS and BIND* (dt. Titel *DNS und BIND*, O'Reilly) von Albitz und Liu wärmstens empfehlen. Jeder, der sich für DNS-Sicherheit interessiert, sollte die neuste Ausgabe dieses Buchs besitzen.

Transaktionssignaturen (TSIGs)

Wenn Sie TSIGs dazu einsetzen möchten, alle Zonen-Transfers zwischen dem Master und dem Slave einer Zone zu signieren, müssen Sie nur das Folgende tun:

1. Erzeugen Sie einen Schlüssel für die Zone.
2. Erzeugen Sie auf jedem Server in der Datei *named.conf* einen *key{}*-Eintrag, der den Schlüssel enthält.
3. Erzeugen Sie auf jedem Server in der Datei *named.conf* einen *server{}*-Eintrag für den Remote-Server, der den in Schritt 2 deklarierten Schlüssel referenziert.

Schritt 1 können Sie am einfachsten mit dem *dnskeygen*-Befehl von BIND durchführen. Um einen 512-Bit-Signaturschlüssel zu generieren, der sowohl vom Master als auch vom Slave verwendet werden kann, geben Sie das Folgende ein:

```
dnskeygen -H 512 -h -n schluesselname
```

Die Resultate werden in zwei Dateien gespeichert, die so ähnlich heißen wie *Kschluesselname.+157+00000.key* und *Kschluesselname.+157+00000.private*. In diesem Fall sollte der Schlüssel-String in beiden Dateien identisch sein. Er sieht etwa folgendermaßen aus:

```
ff2342AGFASsdfsa55BSopiue/ u2342LKJDJlkjVVVvfjweovzp2OIPOTXUEdss2jsdfAAlskj==
```

Die Schritte 2 und 3 bestehen aus der Erstellung von Einträgen in der Datei *named.conf*, die so aussehen wie die, die in Beispiel 6-10 dargestellt sind. Diese Schritte müssen auf jedem Server wiederholt werden. Dabei müssen Sie *schluesselname* durch einen beliebigen Namen für den Schlüssel ersetzen. Dieser String muss allerdings auf beiden Servern identisch sein.

Beispiel 6-10: key{}- und server{}-Syntax

```
key schluesselname {
    algorithm hmac-md5;
    secret "Geben Sie den Schluessel aus irgendeiner Schluesseldatei hier ein.";
}
server IP-Adresse des Remote-Servers {
    transfer-format many-answers;       # (Verschicke Antworten stapelweise
                                        # und nicht einzeln.)
    keys { schluesselname; };
};
```

Sogar ohne eine entsprechende *server{}*-Anweisung teilt eine *key{}*-Angabe einem DNS-Server mit, dass er alle Antworten auf Anfragen signieren soll, die mit dem angegebenen Schlüssel signiert waren. Durch eine *server{}*-Anweisung weiß *named*, dass er alle Anfragen und Aktualisierungen, die er an den Server schickt, mit dem angegebenen Schlüssel signieren soll. Beachten Sie, dass die *key{}*-Anweisungen immer vor allen anderen Anweisungen stehen müssen, die sich darauf beziehen. Dazu gehören zum Beispiel *server{}*-Anweisungen. Ich empfehle Ihnen daher, die *key{}*-Anweisungen zusammen mit den ACL-Definitionen oben in Ihrer *named.conf*-Konfigurationsdatei aufzuführen.

Sobald Sie den Schlüssel erzeugt und die entsprechenden *key{}*- und *server{}*-Einträge in den *named.conf*-Dateien auf beiden Rechnern vorgenommen haben, müssen Sie nur noch *named* auf beiden Rechnern neu starten. Dazu können Sie auf beiden Servern einen der folgenden Befehle verwenden: `kill -HUP`, `ndc restart` (bei BIND v8) oder `rndc restart` (bei BIND v9).

Alle Zonen-Daten, die im Anschluss daran zwischen diesen beiden Servern ausgetauscht werden, werden mit dem geteilten TSIG-Schlüssel kryptographisch signiert. Unsignierte oder nicht richtig signierte Zonen-Daten werden zurückgewiesen.

Zusätzliche Einsatzmöglichkeiten von TSIGs

Ein Schlüssel, der in einem *key{}*-Eintrag in der Datei *named.conf* aufgeführt ist, kann auch in den *acl{}*-, *allow-transfer{}*-, *allow-query{}*- und *allow-update{}*-Einträgen in der Elementliste jeder Anweisung verwendet werden. Dadurch gewinnen Sie viel mehr Flexibilität beim Erstellen der Elementlisten und bei den Anweisungen, die sie benutzen. Auf diese Weise können Sie auch das Verhalten von *named* besser steuern. Außerdem steht Ihnen ein weiteres Kriterium neben der Quell-IP-Adresse zur Authentifizierung von Client-Requests zur Verfügung. Damit ist BIND weniger anfällig für IP-Spoofing.

Beispiel 6-11 zeigt eine *key{}*-Definition, gefolgt von einer Zugriffskontrollliste.

Beispiel 6-11: Ein TSIG-Schlüssel in einer Zugriffskontrollliste

```
key af_fe {
    algorithm hmac-md5;
    secret "ff2342AGFASsdfsa55BSopiue/u2342LKJDJlkjVVVvfjweovzp2OIPOTXUEdss2jsdfAAlskj==";
}
acl guteaffen { 10.10.100.13; key af_fe ; };
```

Eine Übersetzung für diese ACL wäre etwa: »Das Etikett *guteaffen* bezieht sich auf den Rechner mit der IP-Adresse 10.10.100.13, dessen Daten mit dem Schlüssel *af_fe* signiert werden.« Die `key` *`schluesselname`* `;`-Syntax, die in der Elementliste der ACL angewendet wird, wird unverändert auch in *allow-transfer*|*query*|*update{}*-Anweisungen verwendet.

Nehmen wir einmal an, dass wir in unserer fiktiven *named.conf*-Datei, aus der wir den Auszug in Beispiel 6-11 kennen, folgenden Eintrag sehen:

```
allow-transfer { guteaffen; };
```

Dieser Eintrag, der entweder in eine *options{}*- oder in eine *zone{}*-Anweisung eingebettet sein könnte (abhängig davon, ob er global oder nur zonenspezifisch gültig ist), besagt, dass Zonen-Transfers nur berücksichtigt werden, wenn sie mit der ACL *guteaffen* übereinstimmen, d.h., nur wenn die Anfrage von der IP-Adresse 10.10.100.13 stammt *und* mit dem Schlüssel *af_fe* signiert ist.

Informationsmöglichkeiten zu BIND (und IS-Sicherheit)

Die Richtlinien und Techniken, mit denen wir uns beschäftigt haben, sollten Ihnen einen guten ersten Einblick in das Absichern Ihrer BIND-Server vermittelt haben. Um sich ein tieferes Verständnis dieser Techniken anzueignen, empfehle ich Ihnen den *BIND v8 Operators' Guide* und das *BIND v9 Administrators' Reference Manual*. Zumindest für mich gehören die beiden zu den nützlichsten Dokumenten, die jedem OSS-Paket beiliegen. Eine weitere ausgezeichnete Informationsquelle zur BIND-Sicherheit ist die Präsentation »DNS Security« von Liu. Das Quellenverzeichnis am Ende dieses Kapitels führt weitere Informationen über diese und andere BIND-Quellen auf.

Ebenso wichtig ist es für jeden BIND-Benutzer, dass er mindestens eine der E-Mail-Listen mit Sicherheitshinweisen abonniert. Ich persönlich bevorzuge BUGTRAQ, weil die

Informationen darin sowohl rechtzeitig kommen als auch umfassend sind. (Allerdings ist die Liste auch stark frequentiert, und ich empfehle Ihnen daher die Digest-Version.) Unter *http://www.securityfocus.com/cgi-bin/subscribe.pl* finden Sie eine Möglichkeit, sich online für diese Liste anzumelden. Eine andere ausgezeichnete Liste ist VulnWatch. Für diese Liste gibt es zwar keinen Digest, allerdings ist sie auch nicht so stark frequentiert wie BUGTRAQ. Weitere Informationen über diese Liste finden Sie unter *http://www.vulnwatch.org/subscribe.html*.

Darüber hinaus empfehle ich Ihnen, die CERT-Hinweise, die im Quellenverzeichnis am Ende dieses Kapitels aufgeführt sind, nachzuschlagen und durchzulesen. Sie müssen die früheren Sicherheitslücken von BIND kennen, um die BIND-Sicherheit zu verstehen.

djbdns

Wenn Sie, nachdem Sie meine Hinweise zu BIND gelesen oder überflogen haben, immer noch misstrauisch gegenüber der Größe, Komplexität und der Geschichte von BIND sind, sollten Sie vielleicht *djbdns* ausprobieren. Dabei handelt es sich um die weniger komplexe, aber robuste Alternative von Daniel J. Bernstein.

Auch wenn es in diesem Abschnitt insbesondere um die Sicherheits-Features von *djbdns* geht, haben wir auch die Absicht, eine allgemeine Einführung zu *djbdns* anzubieten. Dies ist (hoffentlich) aus zwei Gründen gerechtfertigt. Zum einen hat die bloße Auswahl von *djbdns* anstelle von BIND positive Auswirkungen auf die Sicherheit, zumindest weil dadurch »der DNS-Genpool variiert«. Zum anderen ist *djbdns*, obwohl es häufig eingesetzt wird, bisher nicht ausreichend in den Printmedien behandelt worden, so dass diese Einführung eine der ersten ihrer Art ist (wenn nicht sogar *die* erste).

Sollte keine dieser Aussagen Sie überzeugen, brauchen Sie sich allerdings auch nicht schuldig zu fühlen, wenn Sie bei BIND bleiben (vorausgesetzt, Sie verwenden Version 9 und nehmen sich die Zeit, es sorgfältig zu konfigurieren, abzusichern und zu warten). Nicht zuletzt bin ich selbst ja schließlich auch ein BIND v9-Benutzer.

Was ist djbdns?

BIND kann man sich als das atomar angetriebene Spülbecken, den Mixer und Bodenwischer der DNS-Software in einem vorstellen. Er plätschert in einer Ecke vor sich hin, und ab und zu kommt es zu einem Leck oder zu einer Explosion. Trotz seines Marktanteils handelt es sich um eine alte Maschine mit makelhaften Wartungsberichten.

djbdns ist dagegen eine Sammlung von Geräten, die Sie in einem DNS-Spezialitätengeschäft finden können: einfach, abgesichert, schnell und sicher, wenn sie nach Vorschrift eingesetzt wird. Fast unbemerkt werden Millionen von Domain-Namen bei großen Anbietern von Internet-Domains und andere hochfrequentierte Seiten wie z.B. DirectNIC, NameZero, Interland und TicketMaster jeden Tag von diesem Paket bedient. Es wird Sie vielleicht überraschen zu erfahren, dass *tinydns* (die zum Betrieb öffentlicher

Nameserver vorgesehene Komponente von djbdns) der im Internet am zweithäufigsten eingesetzte Nameserver ist. Eine im Jahr 2002 durchgeführte Studie über 22 Millionen *.com*-Domains (*http://cr.yp.to/surveys/dns1.html*) ergab, dass 70% dieser Domains von BIND und 8% von tinydns verwaltet werden. Eine 2004 durchgeführte Studie über beinahe 38 Millionen Domains (*http://mydbs.bboy.net/survey/*), die die Top Level Domains *.com*, *.net*, *.org*, *.info* und *.biz untersuchte*, kam zu dem Schluss, dass tinydns einen Marktanteil von 15.5% besitzt. Durchschnittlich verwaltete *tinydns* pro Server mehr Domains (446) als BIND (72) oder der Microsoft DNS Server (21). Das Programm ist überaus zuverlässig. Die Software läuft einfach, ohne dass dazu ein weiteres menschliches Eingreifen als die Änderung von Domain-Daten erforderlich ist. Die Speicherbelegung ist sehr begrenzt, die Prozesse werden überwacht und bei Bedarf neu gestartet, und die Log-Dateien werden automatisch rotiert, um eine zu hohe Belegung von Plattenplatz zu verhindern. Man muss sich um eine Installation von tinydns also kaum kümmern, was bereits viel über diese Software aussagt.

Wie bei BIND handelt es sich auch bei *djbdns* um freie Software für Unix und Unix-ähnliche Systeme. *djbdns* kann BIND ersetzen oder parallel als primärer oder sekundärer Nameserver eingesetzt werden.

djbdns besteht aus Servern, Clients, Bibliotheken und Hilfsdiensten (vgl. Tabelle 6-2).

Tabelle 6-2: Die Komponenten von djbdns und andere dazugehörige Pakete

djbdns-Paket	Beschreibung
dnscache	Nameserver mit Cache-Funktion
tinydns	Autoritativer Nameserver
axfrdns	Zonen-Transfer-Server
axfr-get	Zonen-Transfer-Client
walldns	Eine umgekehrte DNS-Mauer; sie bietet umgekehrte Auflösungen, ohne etwas über das interne Netzwerkdesign zu verraten
rbldns	IP-Adresslisten-Server, geeignet für unsichtbare Listen
dnsip, dnsname, dnsmx, dnsipq, dnsfilter	DNS-Hilfs-Clients
dnsq, dnsqr, dnstrace	DNS-Debugging-Clients
dns	Eine C-Bibliothek für DNS
Dazugehörige Pakete	**Beschreibung**
daemontools	Hilfsmittel für das Service-Management, die vom *dnscache* und von *tinydns* verwendet werden
ucspi-tcp	TCP-Client-Server-Interface, das von *axfrdns* und *axfr-get* benutzt wird

Wir beschäftigen uns in Kürze damit, wie man die Hauptkomponenten installiert und konfiguriert. Zunächst aber sollten Sie erfahren, warum *djbdns* entwickelt wurde und welche Probleme dadurch gelöst werden.

Warum nicht BIND?

Zusammenfassend kann man sagen, dass *djbdns* als Antwort auf Probleme von BIND im Bereich der Sicherheit, der Komplexität und der Performance entwickelt wurde. Es ist daher sinnvoll, *djbdns* über seine Unterscheidungsmerkmale zu BIND zu charakterisieren. Tabelle 6-3 enthält einen solchen Vergleich zwischen *djbdns* und BIND.

Tabelle 6-3: BIND und djbdns im Vergleich

Charakteristische Eigenschaft	BIND	djbdns
Sicherheit	BIND hatte schon viele Sicherheitsprobleme. Da es normalerweise mit root-Privilegien ausgeführt wird, kann jede Ausnutzung (von Buffer-Overflows oder etwas anderem) den Server kompromittieren. Es erfordert zusätzlichen Aufwand, wenn man es als normaler Benutzer oder in einer chroot-Umgebung ausführen möchte. Es gibt keine Sicherheitsgarantien.	Jedes *djbdns*-Programm wird unter einem speziellen Nicht-root-Benutzer in einem chroot-Gefängnis ausgeführt. Sogar wenn es gehackt werden sollte, kann es nicht dazu benutzt werden, die Kontrolle über den Server oder etwas anderes zu gewinnen. Der Entwickler hat eine Prämie von $ 500 ausgesetzt für »denjenigen, der als Erster eine nachweisbare Sicherheitslücke in der neusten Version von *djbdns* entdeckt«.
Bedienbarkeit	BIND ist dafür berüchtigt, schwer zu erlernen, einzusetzen und zu verwalten zu sein. Das Dateiformat ist kryptisch, schwer zu lesen und verzeiht keine Fehler (auch wenn BIND 9 besser ist). Es gibt keine automatische Fehlerkontrolle. Die Integrität des Systems ist daher von dem Wissen und der Disziplin der Administratoren abhängig.	Das *djbdns*-Zonen-Dateiformat (*tinydns-data*) ist einfach aufgebaut. Eingabefehler werden automatisch überprüft. Der Nameserver wird daher nur mit korrekten Daten aktualisiert. Es gibt intelligente Standardwerte z.B. für TTL und Zeitstempel. Sie brauchen daher nichts anzugeben. PTR-Berichte werden automatisch erzeugt. Die Realisierung von DNS mit geteilten Ansichten (Split-DNS) ist einfach.
Marktanteile	Die Nummer Eins.	Die Nummer Zwei.
Änderungen	Häufige Veröffentlichung von Updates und Patches für BIND 8, weniger häufige Updates für BIND 9.	Seit der ersten Ausgabe dieses Buches (2002) wurde keine neue Version mehr veröffentlicht.
Effizienz	BIND ist ein Ressourcenfresser. Es frisst Arbeitsspeicher wie ein hungriger Raubritter. Manchmal kippt es als Folge seiner Fressorgie einfach um – und zieht das Tischtuch mit nach unten.	Die Standardgröße des *dnscache*-Arbeitsspeichers beträgt 1 MByte. Diese Größe kann aber im laufenden Betrieb geändert werden. Verringert sich der freie Cache, werden automatisch die ältesten Einträge im Cache gelöscht.
Klarheit	Wie Orson Welles ist BIND umfangreich, komplex und schwer zu verdauen. Seine eingebaute Logik ist zum Teil verschlungen und funktioniert nicht wie gewünscht. Unerwartete Code-Interaktionen zwischen dem Cache- und dem autoritativen Server tragen dazu bei, dass BIND anfällig für Angriffe wie Cache Poisoning ist.	*djbdns* ist einfach. Jedes seiner Programme tut weniger und verfügt über weniger Code. Es gibt also weniger Angriffsfläche für Probleme. *dnscache* beginnt mit dem Root-Server, um den wirklich zuständigen (autoritativen) Server für Domains zu finden. Man kann es nicht hereinlegen und dazu bewegen, gekidnappten Nameservern zu vertrauen.

Tabelle 6-3: BIND und djbdns im Vergleich (Fortsetzung)

Charakteristische Eigenschaft	BIND	djbdns
Modularität	BIND ist ein Caching-Server, ein für die Zone maßgeblicher Server (»Authoritative Server«[a]) und ein Zonen-Transfer-Server und -Client. Wenn Sie nur eine Funktion benötigen, müssen Sie die anderen deaktivieren und sicherstellen, dass Ihre Firewall keine Zugriffe auf deren Ports zulässt. Die Komplexität des Codes hat zu vielen Bugs und Sicherheitsproblemen geführt.	Verschiedene Funktionen werden von unterschiedlichen Servern wahrgenommen. Jeder Server ist klein, einfacher zu erlernen, zu verstehen und im Alltag zu gebrauchen. Sie installieren nur, was Sie wirklich brauchen: *dnscache* für das Cachen, *tinydns* für den (autoritativen) Server, *axfrdns* und/oder *axfr-get* für Zonen-Transfers.
Verfügbarkeit der Daten	Bei Zonen-Transfers fällt BIND in Trance und kommuniziert mit niemand anderem.	*tinydns* bietet jederzeit Zugriff auf eine konsistente, autoritative Datenbank. Dadurch stehen Name-Services auch zur Verfügung, wenn die Datenbank aktualisiert wird und Zonen-Transfers durchgeführt werden.
Datenintegrität	Standardmäßig werden Zonen-Daten im Klartext und ohne eventuell vorhandene Kommentare übertragen. DNSSEC wurde zur Verschlüsselung des Datenstroms vorgeschlagen, funktioniert aber bis jetzt noch nicht richtig.	Mit den Standard-Tools *rsync* und *ssh* können sichere inkrementelle Transfers von Zonendaten zwischen verschiedenen tinydns-Servern durch Kopieren der Datendateien vorgenommen werden. Es sind keine speziellen Protokolle oder Tools erforderlich. Die ursprünglichen Kommentare und die Formatierung der Dateien werden beibehalten. AXFR-Zonen-Transfers an und von BIND werden ebenfalls unterstützt.
Verfügbarkeit	BIND ist fester Bestandteil jeder Unix-Version. Ablageorte für Dateien und verfügbare Versionen und Patches können sich zwischen den einzelnen Systemen stark unterscheiden.	*djbdns* ist keine Standardkomponente einer Linux- oder BSD-Installation. Das erklärt auch, warum die meisten Leute noch nie davon gehört haben. Die Lizenz von *djbdns* schreibt vor, dass jede weiter vertriebene Version sich auf allen Plattformen gleich verhalten muss und dass die Dateinamen und deren Pfade nicht geändert werden dürfen. Das steht im Gegensatz zur gängigen Praxis von Paketmanagern (BSD-Ports, Red Hat RPM usw.), die das Paket normalerweise so formen, dass es in die Distribution passt. Mit den Worten des Autors *(http://cr.yp.to/compatibility.html)*: »Es ist eine schreckliche Idee, die plattformübergreifende Kompatibilität zugunsten einer paketübergreifenden Ähnlichkeit aufzugeben.« Es ist erlaubt, Patches und den Programmquellcode in Umlauf zu bringen.
RFC-Erfüllung	BIND unterstützt fast alles, was mit DNS zusammenhängt. BIND 9.1.1 beinhaltet mehr als 60 DNS-bezogene RFCs und mehr als 50 Internet-Entwürfe (Internet Drafts)!	*djbdns* unterstützt einige RFCs nicht: IXFR (RFC 1995), DNSSEC (RFC 2535, 2931, 3008), TSIG (RFC 2845), Dynamic DNS (RFC 2136), A6 (RFC 2874) und DNAME (RFC 2672). In jedem Fall argumentiert Bernstein damit, dass diese Standards entweder nicht funktionieren oder dass es eine bessere Lösung gibt.

a Ein »Authoritative Server« besitzt verbindliche Informationen über die abgefragte Zone, im Gegensatz zu einem Server, der diese Informationen nur cacht und mit den gecachten Informationen antwortet.

Auswahl von djbdns-Diensten

djbdns ist modular aufgebaut: Sie entscheiden sich nur für die Teile und führen auch nur die Teile aus, die Sie auf einem bestimmten System benötigen. Zu *djbdns* gehören im Wesentlichen drei Server und ein Client, die die Hauptfunktionen abbilden.

dnscache
: Bei *dnscache* handelt es sich um einen *Caching-Nameserver*. Er verfügt nicht selbst über irgendwelche Daten, sondern verwaltet lediglich einen *lokalen DNS-Cache* für lokale Clients wie zum Beispiel Webbrowser. Die DNS-Anfragen von den Clients werden an *dnscache* weitergegeben. *dnscache* fragt wiederum den öffentlichen Root-Nameserver, folgt der Spur zu autorisierten (autoritativen) Nameservern, erhält die Ergebnisse und hält diese lokal im Speicher, um spätere Anfragen zu beschleunigen. Er kann sowohl eine einzelne Maschine als auch eine Gruppe bedienen. Er hat niemals Kontrollfunktion über eine Domain. *dnscache* nimmt nur rekursive Abfragen entgegen.

tinydns
: *tinydns* ist ein *autoritativer (verbindlicher) Nameserver*. Er gibt Informationen über Ihre Domains an das öffentliche Internet weiter. Er hält nichts im Speicher und beantwortet keine Fragen zu Domains, für die er nicht zuständig ist. *tinydns* gibt Anworten auf wiederholte Anfragen.

axfrdns
: Überträgt Zonen-Daten von einem primären *tinydns*-Nameserver an einen sekundären Nameserver wie BIND.

axfr-get
: Erfragt Übertragungen von Zonen-Daten von einem primären Nameserver wie BIND an einen sekundären *tinydns*-Nameserver.

Dadurch dass es eine Unterscheidung zwischen diesen Funktionen bei *djbdns* gibt, ist es erforderlich, dass Sie sich entscheiden, welche Name-Services Sie überhaupt anbieten möchten und, wenn ja, wo. Im Folgenden finden Sie einige Hinweise zu den gängigsten Situationen:

- Wenn Sie nur eine Unix-Maschine haben und nur Caching-Name-Services für lokale Client-Programme anbieten möchten, installieren Sie einen *internen DNS-Cache* mit *dnscache*.
- Wenn Sie mehrere Maschinen haben, können Sie einen *internen DNS-Cache* mit *dnscache* auf jeder Maschine oder einen *externen DNS-Cache* auf einer Maschine (*dnscachex*) installieren, um Anfragen Ihrer Nachbarrechner zu beantworten.
- Wenn Sie einige Domains verwalten und Nachschlagedienste für diese Domains über das Internet anbieten möchten, installieren Sie den *autoritativen DNS-Server*, *tinydns*.
- Wenn Sie einige Domains verwalten und Redundanz erreichen wollen, installieren Sie *tinydns* auf mehr als einem Server und übertragen die Daten zwischen den Servern mithilfe von *rsync* und *ssh*.

- Wenn Sie *tinydns* installieren, aber auch Zonen-Daten an BIND übertragen müssen (mit *tinydns* als *primärem* oder *Master*-Server), installieren Sie *axfrdns*.
- Wenn Sie *tinydns* installieren, aber auch Zonen-Daten *von* BIND übertragen bekommen müssen (mit *tinydns* als *sekundärem* oder *Slave*-Server), installieren Sie *axfr-get*.

Wie djbdns funktioniert

Abbildung 6-3 zeigt die Komponenten und den Datenfluss von einem *dnscache*. Dieser Server verwendet lediglich einen Arbeitsspeicher-Cache. Ist der gesuchte Eintrag im Cache enthalten und noch nicht abgelaufen, wird er direkt zurückübermittelt. Andernfalls sucht *dnscache* danach. Handelt es sich um eine neue Domain, beginnt er dabei mit den autoritativsten Servern und folgt der Delegationskette weiter nach unten. Dadurch wird *Cache-Poisoning* (manipulierte Daten in einem DNS-Cache) durch das Verfolgen eines gefälschten *Glue-Eintrags* verhindert (Auflösung eines Server-Namens auf dem kürzesten Weg).

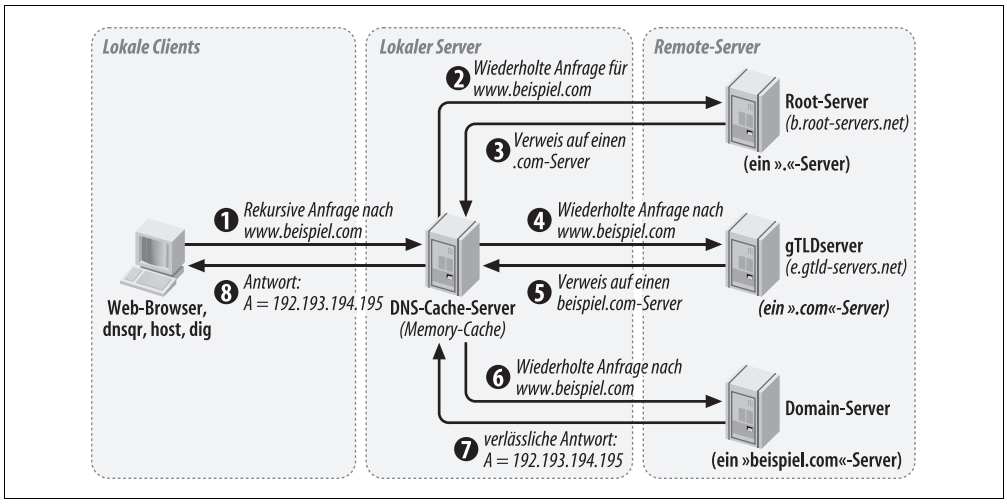


Abbildung 6-3: dnscache-Architektur und -Datenfluss

Abbildung 6-4 demonstriert, wie *tinydns*, *axfrdns* und *axfr-get* jeder unterschiedliche Funktionen wahrnehmen:

A Fügt einen Nameserver-Eintrag für einen Rechner wie zum Beispiel *www.beispiel.com* hinzu oder ändert ihn. Wenn Sie im Internet verlässliche Host-Daten über *beispiel.com* bereitstellen möchten, ist das die Stelle, an der Sie aktiv werden müssten.

B Fragt einen autoritativen *tinydns*-Nameserver nach einem *www.beispiel.com*-Eintrag ab. Externe Clients und Server, die nach dem Rechner *beispiel.com* suchen, würden diesen Weg einschlagen.

C Überträgt die Zonen-Daten für *www.beispiel2.com* auf einen sekundären Nameserver wie BIND. *axfrdns* könnte eine *notify*-Anfrage an den sekundären Server

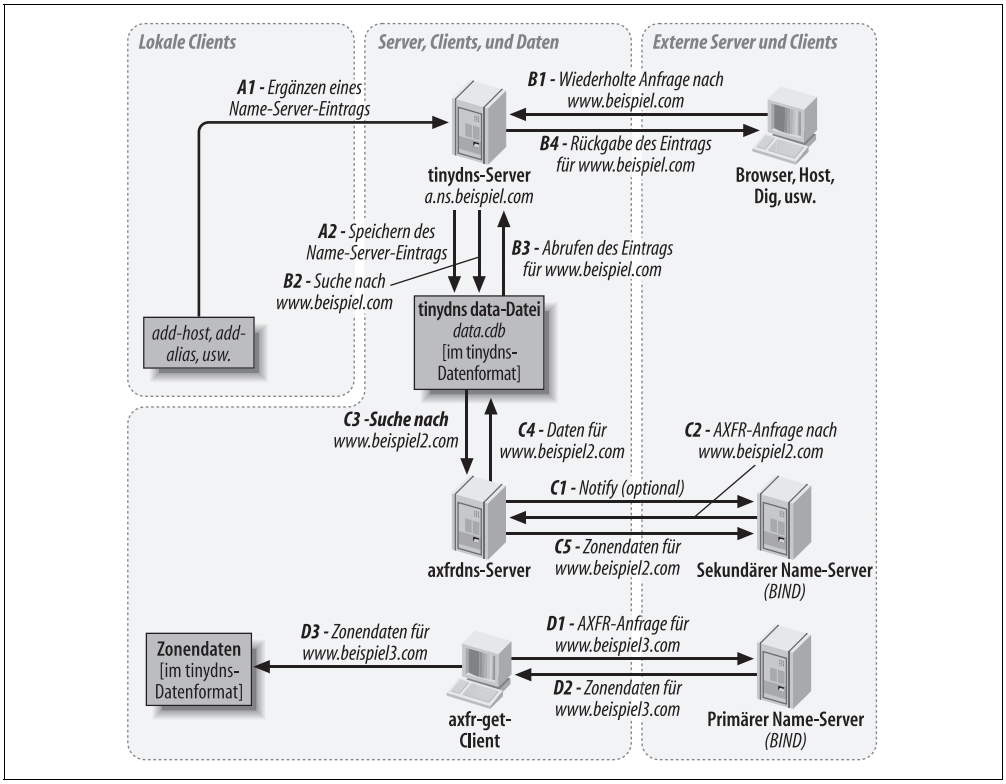


Abbildung 6-4: Architektur der tinydns-Familie und der Datenfluss dadurch

geschickt haben, um diesen zu bewegen, die Daten jetzt zu erfragen und nicht erst, wenn das Ablaufdatum des Eintrags erreicht ist.

D Überträgt Zonen-Daten von *www.beispiel3.com* auf einen primären Nameserver wie BIND. Die Daten werden lokal in einer Datei im *tinydns-data*-Format abgelegt. Die Daten werden normalerweise nicht automatisch mit den Daten aus den Hauptdateien abgeglichen, die von A oder B verwendet werden.

Beachten Sie, dass es keinen Zusammenhang zwischen *dnscache* und einer dieser Funktionen gibt.

Installation von djbdns

Sobald Sie festgelegt haben, welche Rolle oder Rollen Ihr *djbdns*-Nameserver übernehmen soll, können Sie die entsprechenden Pakete installieren. Es gibt einige Pakete, die bei jeder *djbdns*-Installation erforderlich sind.

Installation des Service-Managers: daemontools

Bei der Standardinstallation von *djbdns* ist es erforderlich, zunächst die *daemontools* zu installieren. Diese Hilfswerkzeuge starten die *djbdns*-Server und halten sie am Laufen. Wieso ist das jetzt wieder eine andere Gruppe von Werkzeugen? Auch diese Programme wurden wiederum als Antworten auf Bugs und Unstimmigkeiten in beliebten Unix-Diensten wie *syslogd* und *inetd* entwickelt. Die *daemontools* sind sehr zuverlässig und einfach zu installieren. Probieren Sie sie also aus, und beurteilen Sie selbst, ob Sie sie mögen. Auch wenn RPMs von verschiedenen Stellen angeboten werden, empfehle ich Ihnen doch die Installation aus dem Source-Code. Dieser Installationsweg ist gut dokumentiert. Und zwar erfolgt die Installation folgendermaßen:

1. Laden Sie den *daemontools*-Tarball mit *wget* (oder Ihrem favorisierten HTTP-Client) herunter (die neuste Version finden Sie unter *http://cr.yp.to/daemontools/install.html*):

   ```
   $ wget http://cr.yp.to/daemontools/daemontools-0.76.tar.gz
   ```

2. Entpacken Sie die Distribution:

   ```
   $ tar xvzf daemontools-0.76.tar.gz
   $ rm daemontools-0.76.tar.gz
   $ cd admin/daemontools-0.76
   ```

3. Werden Sie root, und kompilieren und konfigurieren Sie das Paket:

   ```
   # ./package/install
   ```

Dieses Installationsskript macht Folgendes:

- Es kompiliert das Programm.
- Es erstellt das Verzeichnis */command* und füllt es mit einigen Programmen.
- Es erzeugt symbolische Links aus */usr/local/bin* auf die Programme unter */command*.
- Es erzeugt das Verzeichnis */service*.
- Es hängt folgende Zeile an die Datei */command/svscanboot* an:

  ```
  SV:123456:respawn:/command/svscanboot
  ```

  Dadurch wird ein */command/svscan* gestartet, der das */service*-Verzeichnis überwacht. Wir werden ihm in Kürze etwas zu tun geben.

Der Installationsprozess erstellt einige Verzeichnisse, die an manchen Stellen vielleicht nicht erlaubt sind. Wenn Sie keine symbolischen Links als Workaround benutzen können, müssen Sie vielleicht den Source-Code ändern. Diese rigide Installationspraxis garantiert, dass bei jeder Installation von *djbdns* alles an derselben Stelle abgelegt wird, verhindert aber unter Umständen, dass *djbdns* mehr Verbreitung findet.

Installation von djbdns selbst

Sind die *daemontools* kompiliert und am vorgesehenen Ort abgelegt, ist es an der Zeit, den eigentlichen *djbdns* zu installieren:

1. Laden Sie den neuesten Tarball herunter (die neuesten Versionsinformationen finden Sie unter *http://cr.yp.to/djbdns/install.html*):

```
$ wget http://cr.yp.to/djbdns/djbdns-1.05.tar.gz
```

2. Entpacken Sie die Distribution.

```
$ tar xvzf djbdns-1.05.tar.gz
$ rm djbdns-1.05.tar.gz
$ cd djbdns-1.05
```

3. Falls Ihr System glibc 2.3.x oder höher verwendet (z.B. Red Hat 9, Fedora), müssen Sie die Deklaration von `errno` anpassen, da es sich dabei nicht mehr länger um einen ganz gewöhnlichen globalen Integer handelt. Ziemlich am Anfang der Datei *error.h* müssen Sie

```
extern int errno;
```

   in

```
#include <errno.h>
```

   abändern.

4. Kompilieren Sie *djbdns*:

```
$ make
```

5. Werden Sie root, und installieren Sie die Programme im Verzeichnis */usr/local/bin*:

```
# make setup check
```

Installation eines internen Cache: dnscache

Wenn Sie einer oder mehreren lokalen Maschinen DNS-Caching anbieten möchten, müssen Sie *dnscache* installieren.

1. Richten Sie einen Benutzer für *dnscache* und einen anderen für das Logging ein:

```
# adduser -s /bin/false dnscache
# adduser -s /bin/false dnslog
```

2. Legen Sie fest, welche IP-Adresse für *dnscache* verwendet werden soll. Wenn sich der DNS-Cache nur auf Ihre lokale Maschine bezieht, ist Ihre *localhost*-Adresse, 127.0.0.1, eine gute Wahl. (Es ist auch gleichzeitig der Standardeintrag, wenn Sie keine Adresse angeben.) Wenn Sie einen DNS-Cache für mehrere Maschinen einrichten möchten, finden Sie weitere Informationen im Abschnitt *dnscachex*.
3. Wählen Sie ein Verzeichnis für den Server und die dazugehörigen Dateien aus. Üblicherweise wird dazu */etc/dnscache* verwendet.
4. Legen Sie das Verzeichnis *dir* für den *dnscache*-Service an, und stellen Sie dann eine Verbindung zwischen dem Server und dem *dnscache*-Konto *acct*, dem Log-Account *logacct* und dem Port 53 (UDP und TCP) auf der Adresse *ip* her. Der folgende Befehl erfüllt alle diese Aufgaben (abgesehen vom Erstellen des Service-Verzeichnisses, das Sie manuell anlegen müssen):

```
dnscache-conf acct logacct dir ip
```

Wenn wir unsere Beispielauswahl zugrunde legen, erhalten wir Folgendes:

```
# /usr/local/bin/dnscache-conf dnscache dnslog /etc/dnscache 127.0.0.1
```

5. Die Adressen von mehreren ICANN-Rootservern (**.root-servers.net*) haben sich seit dem Release von djbdns 1.0.5 geändert. Die Rootserver-Datei von djbdns (*/etc/dnscache/root/servers/@*) muss deshalb angepasst werden. Dort ist pro Zeile jeweils eine IP-Adresse aufgeführt.

   Sie können diese Datei direkt abändern und dazu folgende Adressen verwenden, die zumindest Anfang 2004 aktuell waren:

   ```
   198.41.0.4
   192.228.79.201
   192.33.4.12
   128.8.10.90
   192.203.230.10
   192.5.5.241
   192.112.36.4
   128.63.2.53
   192.36.148.17
   192.58.128.30
   193.0.14.129
   198.32.64.12
   202.12.27.33
   ```

   Oder Sie verwenden die von djbdns bereitgestellten Tools, um die aktuellen Adressen aus dem Internet zu beziehen:

   ```
   dnsip
    `dnsqr ns . | awk '/answer:/ { print $5 ; }' | sort` \
      > /etc/dnscache/root/servers/@
   ```

   Ein weiterer Weg besteht darin, die Datei *ftp://ftp.internet.net/domain/named.root* herunterzuladen. Anschließend müssen Sie nur noch die Server-Adressen aus den A-Records herauskopieren und sie in der Datei */etc/dnscache/root/servers/@* speichern.

6. Teilen Sie den *daemontools* mit, dass sie die neuen Services verwalten sollen:

   ```
   # ln -s /etc/dnscache /service
   ```

7. Achten Sie darauf, dass Ihr lokaler Resolver den neuen Server benutzt. Editieren Sie die Datei */etc/resolv.conf*, um die Tatsache anzugeben, dass Sie jetzt *dnscache* einsetzen:

   ```
   nameserver 127.0.0.1
   ```

8. Das war's! Sie sind jetzt der stolze Besitzer eines Caching-Nameservers. Führen Sie einige Anwendungen aus, die Ihre Resolver-Bibliotheken aufrufen. *djbdns* enthält die Hilfsmittel *dnsqr*, *dnsip* und *dnsname* (die alle weiter hinten in diesem Kapitel beschrieben werden). Sie können aber auch *ping* oder *host* benutzen. Vermeiden Sie aber *nslookup*, das in diesem Zusammenhang unvorhersehbar ist (vgl. *http://cr.yp.to/djbdns/faq/tinydns.html#nslookup*).

Damit wir beurteilen können, was unter der Oberfläche passiert, lassen Sie uns einmal anschauen, welche Einträge wir in den *dnscache*-Logs finden, nachdem wir die Adresse *www.slashdot.org* gesucht haben:

```
$ tail /service/dnscache/log/main/current
@400000003bd238e539184794 rr 401c4337 86400 ns slashdot.org. ns1.andover.net.
@400000003bd238e539185f04 rr 401c4337 86400 ns slashdot.org. ns2.andover.net.
@400000003bd238e53918728c rr 401c4337 86400 ns slashdot.org. ns3.andover.net.
@400000003bd238e539188614 rr 401c4337 86400 cname www.slashdot.org. slashdot.org.
@400000003bd238e539189d84 cached 1 slashdot.org.
@400000003bd238e53918a93c sent 627215 64
@400000003bd238f62b686b4c query 627216 7f000001:1214:a938 12 20.113.25.24.in-addr.arpa.
@400000003bd238f62b689644 cached 12 20.113.25.24.in-addr.arpa.
@400000003bd238f62b68a9cc sent 627216 88
```

Das Log ist ASCII, aber nicht wirklich für Menschen lesbar. Bei dem ersten Feld handelt es sich um einen TAI64-Zeitstempel, der eine sekundenscharfe Auflösung und eine überaus beeindruckende Reichweite von Billionen von Jahren hat. (Die Unix-Zeit wird im Jahr 2038 einen 32-Bit-Signed-Integer überschreiten.) Die anderen Felder verschlüsseln verschiedene Aspekte der DNS-Nachricht. Lassen Sie die Logs durch einen Filter wie zum Bespiel *tinydns-log.pl* (verfügbar unter *http://tinydns.org/tinydns-log.pl.txt*) laufen, um sie in einem zweckmäßigeren Format betrachten zu können:

```
10-20 21:54:19 rr 64.28.67.55 086400 a slashdot.org. 64.28.67.150
10-20 21:54:19 rr 64.28.67.55 086400 ns  slashdot.org. ns1.andover.net.
10-20 21:54:19 rr 64.28.67.55 086400 ns  slashdot.org. ns2.andover.net.
10-20 21:54:19 rr 64.28.67.55 086400 ns  slashdot.org. ns3.andover.net.
10-20 21:54:19 rr 64.28.67.55 086400 cname  www.slashdot.org. slashdot.org.
10-20 21:54:19 cached a slashdot.org.
10-20 21:54:19 sent 627215
10-20 21:54:36 query 627216 127.0.0.1:4628:43320 ptr 20.113.25.24.in-addr.arpa.
10-20 21:54:36 cached ptr 20.113.25.24.in-addr.arpa.
10-20 21:54:36 sent 627216
```

Installation eines »externen« Cache: dnscachex

Wenn Sie einen DNS-Cache für mehr als eine Maschine in einem lokalen Netzwerk zur Verfügung stellen möchten, müssen Sie eine Adresse wählen, auf die alle diese Maschinen zugreifen können. Diese Adresse ist aus Sicht der Clients »extern«, befindet sich aber innerhalb Ihrer Firewall. Falls Sie sich in einem geschützten Netzwerk befinden, können Sie die Adresse dieser Maschine verwenden. Für *dnscache* und *tinydns* kann man nicht dieselbe Adresse verwenden, weil beide den UDP-Port 53 benutzen.

Für gewöhnlich bezeichnet man den Dienst als *dnscachex*, wenn mehrere Clients bedient werden und als *dnscache,* wenn es nur einen Client gibt. Lassen Sie uns beispielsweise annehmen, dass die Adresse des Diensts *192.168.100.9* lautet und das lokale Netzwerk Adressen verwendet, die alle mit 192.168.100 beginnen :

1. Richten Sie die Benutzer *dnscache* und *dnslog* ein, wie zuvor für den *dnscache* beschrieben:

   ```
   # adduser -s /bin/false dnscache
   # adduser -s /bin/false dnslog
   ```

2. Legen Sie ein Verzeichnis für den Dienst *dnscachex* fest:

   ```
   # /usr/local/bin/dnscache-conf dnscache dnslog /etc/dnscachex 192.168.100.9
   ```

3. Starten Sie *dnscachex*, indem Sie den Dienst mit den *daemontools* verbinden:

   ```
   # ln -s /etc/dnscachex /service
   ```

   Erlauben Sie anderen Maschinen im lokalen Netzwerk den Zugriff auf diesen externen Cache:

   ```
   # touch /etc/dnscachex/root/ip/192.168.100
   ```

   Dazu müssen Sie den Server nicht neu starten.
4. Ändern Sie die */etc/resolv.conf-Datei* auf jeder Maschine ab, die den *dnscachex*-Server verwenden soll:

   ```
   nameserver 192.168.100.9
   ```
5. Testen Sie die Client-Maschinen mithilfe von *ping* oder einer anderen Anwendung, wie zuvor für den *dnscache* beschrieben.

Einen »externen« Forwarding-Cache einrichten

Für jede Maschine, auf der Sie *dnscache* einrichten, müssen Sie für UDP-Port 53 ein Loch in Ihre Firewall reißen. Wenn Sie einen einzigen externen Ccache (*dnscachex*) verwenden, können Sie die dadurch entstehende Gefährdung jeoch auf eine einzige Maschine reduzieren. Sie können auch mehrere Caches hintereinander schalten, so dass ein *dnscache*, der sich innerhalb der Firewall befindet, nur mit einem anderen *dnscache* kommunizieren kann, der sich seinerseits jenseits der Firewall oder in der DMZ befindet. Falls Sie innerhalb Ihrer Firewall einen *dnscachex*-Server aufgesetzt haben, sollten Sie auf allen Client-Rechnern folgendes Kommando ausführen:

```
echo 1 > /service/dnscache/env/FORWARDONLY
```

Führen Sie diesen Befehl auf keinen Fall auf dem *dnscachex*-Server aus. Ändern Sie auf dieser Maschine lediglich die Nameserver-Adresse in */etc/named.conf* in die Adresse des *dnscache*-Servers, der sich auf der anderen Seite der Firewall befindet.

Split Horizon

Sie möchten vielleicht einen so genannten *Split Horizon*-DNS-Dienst bereitstellen, wobei jeder Client innerhalb Ihres Netzwerks Zugriff auf interne wie externe Nameserver bekommt. Um es mit einer beliebten Redewendung der Perl Community zu sagen, hier führen mehrere Wege ans Ziel:

- Verwenden Sie einen Forwarding-Cache. Für jede interne Domain, die Sie gesondert behandeln möchten, erstellen Sie unter */service/dnscache/root/servers* eine Datei, die als Dateinamen den Namen der Domain bekommt. Als Inhalt speichern Sie in dieser Datei lediglich die IP-Adresse des zuständigen Nameservers. Wenn Sie beispielsweise einen internen Nameserver mit der Adresse 192.168.1.23 haben, der für das riesige interne Netzwerk von *hackenbush.com* zuständig ist, führen Sie dieses Kommando aus:

  ```
  echo 192.168.1.23 > /service/dnscache/root/servers/hackenbush.com
  ```

- Verwenden Sie für Ihre internen tinydns-Nameserver so genannte *Tagged Records*. Diese verhalten sich ähnlich wie die von BIND verwendeten *Views* und werden später im Abschnitt *tinydns* noch genauer behandelt.

Installation eines DNS-Servers: tinydns

Wenn Sie einen autoritativen Nameserver für Ihre Domains einrichten möchten, installieren Sie *tinydns*:

1. Richten Sie einen Benutzer für *tinydns* und einen anderen für sein Logging ein (wenn Sie *dnscache* installiert haben, ist der zweite Benutzer bereits vorhanden):

   ```
   # adduser -s /bin/false tinydns
   # adduser -s /bin/false dnslog
   ```

2. Legen Sie eine IP-Adresse für *tinydns* fest. Die Dienste *dnscache* und *tinydns* müssen auf unterschiedlichen IP-Adressen laufen, weil sie beide den UDP-Port 53 benutzen. Wenn Sie beide auf einer Maschine ausführen, verwenden Sie die Loopback-Adresse (127.0.0.1) für *dnscache* und die öffentliche Adresse für *tinydns*. Für den Fall, dass Sie *dnscachex* auf der öffentlichen Adresse der Maschine laufen lassen, weisen Sie der Maschine mit *ifconfig* eine andere IP-Adresse zu und benutzen diese für *tinydns*. Die Syntax von *tinydns-conf* ist mit der von *dnscache-conf* vergleichbar:

   ```
   tinydns-conf acct logacct dir ip
   ```

   Angenommen, Sie haben beschlossen, die öffentliche Adresse 208.209.210.211 zu verwenden, dann richten Sie den Dienst folgendermaßen ein:

   ```
   # /usr/local/bin/tinydns-conf tinydns dnslog /etc/tinydns 208.209.210.211
   ```

3. Aktivieren Sie den Dienst, indem Sie für *svscan* einen Link einrichten, mit dem er arbeiten kann:

   ```
   # ln -s /etc/tinydns /service
   ```

4. *tinydns* läuft jetzt, allerdings ohne irgendwelche Daten, mit denen er arbeiten kann. Lassen Sie uns etwas dagegen unternehmen.

tinydns ausführen

Es ist nun an der Zeit, Ihrem Nameserver einige Daten hinzuzufügen. Dies kann auf zweierlei Weise geschehen:

- Benutzen Sie die *tinydns-Hilfsanwendungen*. Dabei handelt es sich um Shell-Skripten, die *tinydns-edit* mit Standardwerten aufrufen und die Datenbank auf Konsistenz überprüfen, wenn Sie Änderungen vornehmen.
- Editieren Sie direkt die *tinydns*-Daten. Dadurch gewinnen Sie mehr Möglichkeiten, haben aber auch weniger automatische Kontrollmöglichkeiten.

Hilfsanwendungen

Lassen Sie uns zuerst die Helferlein benutzen. Diese modifizieren die Textdatei *data* und führen eine Überprüfung mit der autoritativen Datenbank-Datei, *data.cdb*, durch:

1. Werden Sie *root*.
2. Wechseln Sie in das *tinydns*-Datenverzeichnis:

   ```
   # cd /service/tinydns/root
   ```
3. Ergänzen Sie einen Eintrag, der einen primären Nameserver für Ihre Domain angibt:

   ```
   # ./add-ns hackenbush.com 192.193.194.195
   ```
4. Ergänzen Sie einen Eintrag, der einen sekundären Nameserver für Ihre Domain angibt:

   ```
   # ./add-childns hackenbush.com 200.201.202.203
   ```
5. Ergänzen Sie einen Host-Eintrag:

   ```
   # ./add-host hugo.hackenbush.com 192.193.194.200
   ```
6. Ergänzen Sie einen Alias-Namen für dieselbe Adresse:

   ```
   # ./add-alias another.hackenbush.com 192.193.194.200
   ```
7. Ergänzen Sie einen Mail-Server-Eintrag:

   ```
   # ./add-mx mail.hackenbush.com 192.193.194.201
   ```
8. Veröffentlichen Sie diese Änderungen (konvertieren Sie die Datei *data* in *data.cdb*):

   ```
   # make
   ```

tinydns wird dann sofort mit den geänderten Einträgen arbeiten. Lassen Sie uns nun näher betrachten, was diese Hilfsanwendungen eigentlich gemacht haben, damit wir uns im Anschluss ansehen können, wie man diese Änderungen manuell durchführen kann.

Das tinydns-Datenformat

Die Hilfsanwendungen ändern die *data*-Datei. Bei dieser Datei handelt es sich um eine Textdatei im *tinydns-data*-Format. Dieses Format ist einfach, kompakt und leicht zu modifizieren. Die folgenden Zeilen zeigen die Einträge, die die Hilfsanwendung im Verlauf unseres Beispiels im letzten Abschnitt erzeugt hat:

```
.hackenbush.com:192.193.194.195:a:259200
&hackenbush.com:200.201.202.203:a:259200
=hugo.hackenbush.com:192.193.194.200:86400
+another.hackenbush.com:192.193.194.200:86400
@mail.hackenbush.com:192.193.194.201:a::86400
```

Anstatt die Hilfsanwendung zu benutzen, hätten wir diese Einträge auch mit einem Texteditor erstellen und mit den Standard-*ttl*-Werten füllen können:

```
.hackenbush.com:192.193.194.195:a
&hackenbush.com:200.201.202.203:a
=hugo.hackenbush.com:192.193.194.200
+another.hackenbush.com:192.193.194.200
@mail.hackenbush.com:192.193.194.201:a
```

Wenn sich der primäre Nameserver in unserer Domain befindet (unter *a.ns.hackenbush.com)*, der sekundäre Nameserver aber unter *ns.schwungrad.com*, müsste man das folgendermaßen spezifizieren:

```
.hackenbush.com:192.193.194.195:a
&hackenbush.com::ns.schwungrad.com
```

Würde sich der primäre Nameserver unter *ns.schwungrad.com* befinden, könnte man das folgendermaßen angeben:

```
.hackenbush.com::ns.schwungrad.com
```

Ein paar Zeichen nehmen Ihnen viel Arbeit ab und tragen zur Vermeidung von häufigen Fehlern in BIND-Zonen-Dateien bei:

- Einträge, die mit einem Punkt (.) beginnen, erzeugen einen SOA-Eintrag, einen NS-Eintrag und einen A-Eintrag, wenn eine IP-Adresse angegeben ist.
- Einträge, die mit einem Gleichheitszeichen (=) beginnen, erzeugen A- und PTR-Einträge.

tinydns-data-Referenz

Jeder Eintrag (jede Zeile) in einer *tinydns-data*-Datei beginnt mit einem charakteristischen Zeichen. Die einzelnen Felder werden durch Doppelpunkte voneinander abgetrennt. Nachfolgende Felder und ihre Doppelpunkte können ausgelassen werden. In diesem Fall werden die Standardwerte verwendet. Tabelle 6-4 beschreibt einige Felder, die in vielen Arten von *tinydns-data*-Einträgen vorkommen:

Tabelle 6-4: Häufig verwendete tinydns-data-Felder

Feld	Beschreibung	Standardwert
dom	Der Name einer Domain wie *hackenbush.com*	Keiner
fqdn	Ein vollständig qualifizierter Domain-Name wie *hugo.hackenbush.com*. Es kann auch eine Wildcard verwendet werden: **.fqdn* steht zum Beispiel für alle Namen, die auf *.fqdn* enden, es sei denn, es gibt einen zutreffenderen Eintrag für einen Namen.	Keiner
ip	Eine IP-Adresse wie zum Beispiel *192.193.194.195*	Keiner
ttl	Lebensdauer (die Zeitdauer in Sekunden, für die die in diesem Eintrag enthaltenen Daten gecacht werden dürfen).	SOA: 2560 (42.6 Minuten); NS: 259200 (3 Tage); MX, A, andere: 86400 (1 Tag)
ts	Wenn *ttl* fehlt oder ungleich null ist, gibt dieses Feld die Startzeit für die Gültigkeit der Informationen in dieser Zeile an. Ist der Wert von *ttl* null, wird die Endzeit festgelegt. *ts* wird durch einen externen TAI64-Zeitstempel angegeben. Dabei handelt es sich um einen 16 Zeichen langen Hexadezimal-String in Kleinschreibweise mit einer Auflösung von einer Sekunde. Der Hexadezimalwert `4000000000000000` entspricht der ISO-Zeit `1970-01-01 00:00:00`, der Referenz-Startzeit für Unix-Systeme.	Leer. Dieser Wert bedeutet, dass der Eintrag aktiv ist.
loc	Ein aus einem oder zwei Zeichen bestehender String, der eine Ortsangabe enthält und dazu benutzt wird, verschiedene Informationen in Abhängigkeit von der Position des Clients weiterzugeben. Weitere Informationen darüber finden Sie in der djbdns-Dokumentation.	Keiner

Die nächste Tabelle, Tabelle 6-5, enthält eine Übersicht über den Zusammenhang zwischen den *tinydns*-Hilfsanwendungen und den entsprechenden Einträgen in der *data*-Datei. Sie können Ihre Daten auf die von Ihnen gewünschte Weise angeben. Beachten Sie dabei, dass bei den Hilfsanwendungen die Angabe von IP-Adressen und nicht von Namen erforderlich ist. Falls Sie stattdessen einen Namen oder die *ttl*-, *ts*- oder *loc*-Felder angeben möchten, müssen Sie die *data*-Datei editieren.

Tabelle 6-5: Die Syntax der Hilfsanwendungen und das tinydns-data-Format im Vergleich

Syntax der Hilfsanwendung	data-Format	Beschreibung
`add-ns` *dom ip*	.*dom*:*ip*:*x*:*ttl*:*ts*:*loc*	Legt einen *primären Nameserver* für die Domain *dom* fest. Erzeugt einen SOA-Eintrag für die Domain und einen NS-Eintrag für den Nameserver, der mit *x* und/oder *ip* angegeben ist. Enthält *x* irgendwelche Punkte, wird es als wörtlicher Host-Name interpretiert, andernfalls als *x*`.ns.`*dom*. Wenn eine *ip* vorhanden ist, wird ein A-Eintrag angelegt.
		Mithilfe von *add-ns* werden die aufeinander folgenden Werte `a`, `b` usw. für *x* angelegt. Diese entsprechen `a.ns.`*dom*, `b.ns.`*dom* usw. Dieses Standardverhalten erzeugt *inneramtliche* (intradomain) Namen für die Nameserver. Die Festlegung eines Nameservers für die Domain innerhalb der Domain selbst trägt dazu bei, dass nicht extra der Root-Nameserver zur Namensauflösung herangezogen werden muss.
`add-childns` *dom ip*	&*dom*:*ip*:*x*:*ttl*:*ts*:*loc*	Legt den *sekundären Nameserver* für eine Domain fest. Erzeugt nur einen NS-Eintrag für den Nameserver, der mit *x* und/oder *ip* angegeben ist. Enthält *x* irgendwelche Punkte, wird es als wörtlicher Host-Name interpretiert, andernfalls als *x*`.ns.`*dom*. Wenn eine *ip* vorhanden ist, wird ein A-Eintrag angelegt.
		Add-childns erzeugt ebenfalls `a`, `b` usw. für *x*.
`add-host` *fqdn ip*	=*fqdn*:*ip*:*ttl*:*ts*	Ergänzt einen Rechner: Erzeugt einen A-Eintrag (*fqdn* nach *ip*) und einen PTR-Eintrag (*umgekehrte-ip*`.in-addr.arpa` nach *fqdn*).
`add-alias` *fqdn ip*	+*fqdn*:*ip*:*ttl*:*ts*	Legt einen Alias-Namen fest: Erzeugt einen A-Eintrag (*fqdn* nach *ip*).
`add-mx` *fqdn ip*	@*dom*:*ip*:*x*:*dist*:*ttl*:*ts*	Legt einen Mail-Server fest: Erzeugt einen MX-Eintrag. Enthält *x* irgendwelche Punkte, wird es als wörtlicher Host-Name interpretiert, andernfalls als *x*`.ns.`*dom*`.`*dist*, und hat einen Standardwert von 0.
		Add-mx erzeugt auch die aufeinander folgenden Host-Namen `a`, `b` usw. für *x*.

Für die weniger üblichen Eintragsarten, die in Tabelle 6-6 aufgeführt werden, gibt es keine Hilfsanwendungen.

Tabelle 6-6: Weniger gebräuchliche Eintragsarten

Syntax der Hilfsanwendung	data-Format	Beschreibung
(Kein Helferlein)	`Zdom:fqdn:con:ser:ref:ret:exp:min:ttl:ts:loc`	Erzeugt nur einen SOA-Eintrag mit dem Kontakt `con`, der laufenden Nummer `ser`, der Refresh-Zeit `ref`, der Retry-Zeit `ret`, der Expire-Zeit `exp` und der Mindestzeit `min` für `dom`.
(Kein Helferlein)	`Chost2:fqdn:ttl:ts:loc`	Richtet einen CNAME-Eintrag für `host2` ein, um auf `host` zu verweisen.
(Kein Helferlein)	`'fqdn:text:ttl:ts:loc`	Erzeugt einen TXT-Eintrag für `fqdn`. Dabei kann `text` nur Oktalwerte als Escape-Code enthalten (z.B. `\272`), um Nicht-ASCII-Werte einzurichten.
(Kein Helferlein)	`^fqdn:ip:ttl:ts:loc`	Erzeugt einen PTR-Eintrag für `fqdn` nach `ip`.
(Kein Helferlein)	`:fqdn:type:data:ttl:ts:loc`	Erzeugt einen Eintrag vom Typ `type` (einem Integer zwischen 1 und 65.535). `data` kann oktalen Escape-Code enthalten.

Nachdem Sie eine Datendatei geändert haben, geben Sie `make` ein. Dadurch wird das *tinydns-data*-Programm, das *data* in *data.cdb* konvertiert, ausgeführt. Durch die Konvertierung wird die vorhandene Datenbasis nur überschrieben, wenn die Ausgangsdaten konsistent sind. Anschließend arbeitet *tinydns* sofort mit den neuen Daten.

In manchen mit *tinydns* verwalteten Netzwerken werden die Zonendaten aufgrund der leichteren Änderbarkeit in Datenbanken (SQL oder LDAP) oder in separaten Dateien gespeichert. Die eigentliche tinydns-data-Datei wird in diesen Installationen bei Bedarf automatisch generiert.

Ausführen von djbdns-Client-Programmen

Zusätzlich zu seinem Server-Daemon und den unterstützenden Prozessen enthält *djbdns* verschiedene Client-Programme (siehe Tabelle 6-7). Diese dienen zur Ausführung der gleichen Funktionen wie die alten BIND-Tools *nslookup* und *dig* und sind im Bereich des Troubleshootings und zum Testen Ihrer DNS-Infrastruktur nützlich. Diese Programme können aber nicht nur mit *tinydns*, sondern auch mit jedem anderen DNS-Server verwendet werden.

Tabelle 6-7: In djbdns enthaltene Client-Programme

Programm	Syntax	Beschreibung
dnsip	`dnsip fqdn1 [fqdn2...]`	Gibt die IP-Adresse von einem oder mehreren vollständig qualifizierten Domain-Namen aus.
dnsname	`dnsname ip1 [ip2...]`	Gibt den ersten Domain-Namen von einer oder mehreren IP-Adressen aus.
dnsmx	`dnsmx fqdn`	Gibt den MX-Eintrag von `fqdn` aus.
dnstxt	`dnstxt fqdn`	Gibt den TXT-Eintrag von `fqdn` aus.
dnsq	`dnsq type fqdn server`	Schickt eine nicht rekursive Anfrage an den `server`, um die Einträge vom Typ `type` für `fqdn` zu bekommen.

Tabelle 6-7: In djbdns enthaltene Client-Programme (Fortsetzung)

Programm	Syntax	Beschreibung
dnsqr	`dnsqr` *`type fqdn`*	Holt die Einträge vom Typ *`type`* für *`fqdn`*. Dadurch wird eine rekursive Abfrage beim Nameserver ausgelöst, der unter */etc/resolv.conf.dnsqr* angegeben ist. Dieses Programm ist vergleichbar mit den Programmen `dig`, `host` und `nslookup`.
dnstrace	`dnstrace` *`type fqdn server1 [server2…]`*	Ermittelt alle DNS-Server, die Einfluss auf die Auflösung der Einträge vom Typ *`type`* für *`fqdn`* haben können, angefangen von einem oder mehreren *root*-Nameservern *`server1`*`, ... serverx`.
dnsfilter	`dnsfilter` *`[-c queries] [-n lines]`*	Ersetzt den Host-Namen am Anfang von Textzeilen durch IP-Adressen. Liest von der Standardeingabe und schreibt in die Standardausgabe. *`queries`* gibt die Höchstzahl von parallel zulässigen DNS-Abfragen an (der Standardwert ist 10). *`lines`* steht für die Anzahl der Zeilen, die weitergelesen werden sollen (Standard ist 1.000).

Gemeinsamer Einsatz mit BIND

Sie könnten sich dafür entscheiden, einige Komponenten von *djbdns* auf Ihren Servern zur Abwicklung von Name-Service-Aufgaben zu installieren. Sie könnten wahlweise oder zwangsweise in die Situation geraten, dass diese Aufgaben teilweise auch über eine vorhandene BIND-Installation übertragen werden müssen. Dieser Abschnitt enthält Informationen darüber, wie Zonen-Daten zwischen Nameservern ausgetauscht werden können, auf denen zum Teil *djbdns* und zum Teil BIND läuft.

Installation von ucspi-tcp

Als Erstes müssen Sie ein kleines externes Toolkit namens *ucspi-tcp* installieren, das ebenfalls von Bernstein entwickelt wurde. Es enthält die Programme *tcpserver* und *tcpclient*. Ähnlich wie *inetd* verwalten diese Programme den externen Zugriff auf TCP-basierte Clients und Server. Allerdings sind sie dabei zuverlässiger als *inetd*, weil sie bessere Kontrollmöglichkeiten für Last und Ressourcenbelegung bieten. Um *ucspi-tcp* zu installieren, führen Sie folgende Schritte aus:

1. Laden Sie mithilfe von *wget* (oder einem anderen HTTP-Tool Ihrer Wahl) den neusten Tarball von *http://cr.yp.to/ucspi-tcp/install.html* herunter:

   ```
   $ wget http://cr.yp.to/ucspi-tcp/ucspi-tcp-0.88.tar.gz
   ```

2. Entpacken Sie ihn:

   ```
   $ tar xvzf ucspi-tcp-0.88.tar.gz
   ```

3. Passen Sie bei Bedarf die Datei *errno.h* an:

   ```
   $ cd ucspi-tcp.0.88
   $ vi error.h
   ```

   Ändern Sie

   ```
   extern int errno;
   ```

   in:

   ```
   #include <errno.h>
   ```

4. Starten Sie den Kompilierungsprozess:

```
$ make
```

5. Werden Sie *root* und installieren Sie das Toolkit im Verzeichnis `/usr/local/bin`:

```
$ make setup check
```

Ausführen von axfr-get

Der *axfr-get*-Client stößt einen Zonen-Transfer von einem Nameserver über AXFR an. Dabei wird folgende Syntax verwendet:

```
axfr-get dom datei tmpdatei
```

In diesem Beispiel wird ein Zonen-Transfer für die Domain *dom* angestoßen. Die Daten werden zuerst im *tinydns-data*-Format in die Datei *tmpdatei* geschrieben. Die erste Zeile, die in die Datei *tmpdatei* geschrieben wird, enthält einen Kommentar mit der Seriennummer der Zone. Nach erfolgreicher Übertragung wird die Datei *datei.tmp* in *datei* umbenannt.

Achten Sie darauf, dass Sie nur Daten aus Zonen erfragen, in denen Ihr *tinydns*-Server ein sekundärer Server ist. Kombinieren Sie diese Daten dann in der *tinydns*-Datendatei */service/tinydns/root/data* mit den Daten der Zonen, für die Ihr *tinydns*-Server ein primärer Server ist.

Eine einfache Lösung ist die Ergänzung von folgenden Einträgen im */service/tinydns/root/Makefile*. Unser Beispiel-*tinydns*-Server ist *a.ns.hackenbush.com*, und wir bieten sekundäre Name-Services für die Domain *schwungrad.com* an, deren Nameserver *ns.schwungrad.com* ist:

```
all: data.cdb
schwungrad.data:
    /usr/local/bin/tcpclient -i \
    a.ns.hackenbush.com \
    53 \
    /usr/local/bin/axfr-get \
    schwungrad.com \
    schwungrad.data \
    schwungrad.tmp
data: hackenbush.data schwungrad.data
    cat *.data > data
data.cdb: data
    usr/local/bin/tinydns-data
```

Führen Sie `make` so lange aus wie notwendig, um die Daten von *schwungrad* zu bekommen.

Axfr-get unterstützt weder `NOTIFY` (RFC 1996) noch `IXFR` (RFC 1995). Er schickt keine automatischen `AXFR`-Anfragen an den primären externen Nameserver, wenn der SOA-Refresh-Timeout ausläuft. Sie müssen deswegen sicherstellen, dass *axfr-get* oft genug aufgerufen wird (zum Beispiel durch einen stündlichen Cron-Job). Er holt dann zunächst den SOA und überprüft seine Seriennummer. Wenn diese Nummer höher ist als die lokal angegebene, stößt er eine Abfrage nach Zonen-Daten über AXFR an.

Es wäre schön, eine Server-Version von *axfr-get* zu haben, die primäre BINDs genauso behandelt wie sekundäre BINDs. Wir hätten dann eine komplette Ersatzlösung für einen sekundären BIND (es sei denn, Sie verwenden DNSSEC oder ein experimentelles Protokoll).

Installation von axfrdns

axfrdns benutzt den TCP-Port 53 und kann sich daher eine IP mit *tinydns* teilen, der UDP-Port 53 benutzt. Wenn Sie die IP *192.193.194.195* verwenden, führen Sie zur Installation die folgenden Schritte aus:

1. Legen Sie ein Verzeichnis für den Dienst an:

```
# axfrdns-conf axfrdns dnslog /etc/axfrdns /etc/tinydns 192.193.194.195
# cd /etc/axfrdns
```

2. Editieren Sie die `tcp`-Datei, um Zonen-Transfers von *200.201.202.203* auf *hackenbush.com* zuzulassen und umgekehrt:

```
200.201.202.203:allow,AXFR="hackenbush.com,194.193.192.in-addr.arpa"
```

3. Wandeln Sie `tcp` in ein Binary-Format um:

```
# make
```

4. Informieren Sie *daemontools* über den neuen Dienst:

```
# ln -s /etc/axfrdns /service
```

Ausführen von axfrdns

Der sekundäre Server stößt einen Zonen-Transfer von *axfrdns* an, wenn die TTL des SOA-Eintrags der Zone abgelaufen ist. *axfrdns* bedient die Zone aus derselben autoritativen Datenbasis, die von *tinydns* verwendet wird, nämlich *data.cdb*. Sie können auch veranlassen, dass der sekundäre Server direkt einen Zonen-Transfer anstößt, indem Sie ihm eine *notify*-Nachricht schicken. Auch wenn es kein Standard-Bestandteil von *djbdns* ist, kann das Perl-Skript *tinydns-notify* (das unter *http://www.sericyb.com.au/tinydns-notify* verfügbar ist) dazu verwendet werden.

axfrdns antwortet ausschließlich auf AXFR-Anfragen, und es überträgt ganze Zonen. Wenn ein externer Nameserver wie BIND eine IXFR-Anfrage an *axfrdns* richtet, schlägt diese fehl. RFC 1995 besagt, dass der Anfragende dann AXFR ausprobieren soll (RFC 1995). Das wird allerdings durch einen Bug verhindert, der in manchen Versionen von BIND vorhanden ist. Das Problem kann auf eine der folgenden Weisen behoben werden:

- Patchen Sie *axfrdns* so, dass es IXFR akzeptiert. Den notwendigen Patch bekommen Sie unter der URL *http://www.fefe.de/dns/djbdns-1.05-ixfr.diff.gz.*.
- Wechseln Sie von Ihrer BIND-Version zur BIND-Version 9.2 oder höher.
- Konfigurieren Sie BIND mit `request-ixfr no;`.

Bernstein empfiehlt für sich ergänzende und sichere Transfers die Verwendung von *rsync* und *ssh* anstelle von AXFR und IXFR.

Verschlüsselung von Zonen-Transfers mit rsync und ssh

Wenn Sie *djbdns* auf allen Ihren Servern einsetzen, brauchen Sie keine Domain-Daten mit AXFR zu übertragen. Stattdessen können Sie *rsync* und *ssh* für sich ergänzende, sichere Transfers verwenden:

1. Installieren Sie die *rsync*- und *ssh*-Server und -Clients, falls Sie das bisher noch nicht getan haben.
2. Starten Sie die *rsync*- und *sshd*-Daemons auf dem sekundären Server.
3. Erteilen Sie dem primären Server die Erlaubnis, Daten auf den sekundären Server über *ssh* zu schreiben.
4. Editieren Sie Ihr */service/tinydns/root/Makefile*. Wenn die Server-Adresse Ihres sekundären Servers *192.193.194.195* ist, sollte Ihr *Makefile* folgendermaßen aussehen:

```
remote: data.cdb
    rsync -az -e ssh data.cdb 192.193.194.195:/service/tinydns/root/data.cdb
data.cdb: data
    /usr/local/bin/tinydns-data
```

Normalerweise werden Sie von *ssh* zur Angabe eines Passworts aufgefordert. Um das zu vermeiden, erzeugen Sie ein Schlüsselpaar und kopieren den öffentlichen Schlüssel in das *user*-Verzeichnis auf dem sekundären Server. Detaillierte Informationen darüber finden Sie in den entsprechenden Abschnitten über SSH in Kapitel 4.

Das war's! Jedes Mal, wenn Sie jetzt Änderungen an *tinydns* vornehmen, sei es über die Hilfsanwendungen oder indem Sie direkt die Zonen-Dateien editieren und anschließend `make` eingeben, um sie zu veröffentlichen, wird die Datenbank *data.cdb* auf den sekundären Server kopiert. Durch die Verwendung von *rsync* ist gewährleistet, dass nur die geänderten Teile übernommen werden. Durch *ssh* ist sichergestellt, dass die Daten während der Übertragung verschlüsselt und gegen Diebstahl oder Veränderungen geschützt sind.

Alternativ können Sie auch *rsync* auf die Datendatei und nicht auf die Datenbank *data.cdb* anwenden und dann *make* auf dem sekundären Server ausführen, um die Datenbank zu erstellen.

Daten-Migration aus BIND

Wenn Sie BIND nur als Caching-Server verwenden, wird die Installation von *dnscache* BIND vollständig ersetzen. Vergessen Sie nicht, den *named*-Prozess auszuschalten.

Wenn BIND Ihre Domain mit Daten bedient und auf die übliche Weise konfiguriert ist, kann es durch *tinydns* ersetzt werden. Einige neuere Funktionen wie DNSSEC und IXFR werden nicht unterstützt, aber *ssh* und *rsync* bieten eine einfachere und bessere Funktionalität.

Bernstein hat ausführlich beschrieben, wie man eine Site von BIND auf *tinydns* migriert. Diese Beschreibung finden Sie unter *http://cr.yp.to/djbdns/frombind.html*. Sie enthält folgende Informationen:

- Die Verwendung von *axfr-get*, um Zonen-Daten von einem BIND-Server zu holen und in das *tinydns-data*-Format zu konvertieren
- Das Ersetzen von Seriennummern und TTLs durch automatische Werte
- Das Zusammenführen von verschiedenen Eintragstypen
- Das Testen Ihrer Einrichtung, während BIND läuft, und dessen würdige Ersetzung

Ressourcen

Hoffentlich habe ich Ihnen eine gute Einführung in die Absicherung Ihrer BIND- oder *djbdns*-basierten DNS-Server gegeben. In den folgenden Quellen finden Sie bei Bedarf weitergehende Informationen.

Allgemeine Ressourcen für die DNS-Sicherheit

comp.protocols.tcp-ip.domains
: USENET-Gruppe

http://www.intac.com/~cdp/cptd-faq/
: *comp.protocols.tcp-ip.domains*'s Häufig gestellte Fragen zu DNS.

comp.protocols.tcp-ip.domains USENET-Gruppe: »FAQ« Website: *http://www.intac.com/~cdp/cptd-faq/*.

Rowland, Craig. »Securing BIND« (*http://www.guides.sk/psionic/dns/*) Hinweise zur Absicherung von BIND auf OpenBSD und Red Hat Linux.

Einige RFCs, die mit DNS in Zusammenhang stehen (verfügbar unter http://www.rfc-editor.org)

- 1035 (allgemeine DNS-Spezifikation)
- 1183 (zusätzliche Resource-Record-Spezifikation)
- 2308 (Negatives Caching)
- 2136 (Dynamische Updates)
- 1996 (DNS-Notify)
- 2535 (DNS-Security-Extensions)

Einige DNS/BIND-Sicherheitshinweise (verfügbar unter http://www.cert.org)

CA-2002-31
: »Multiple Vulnerabilities in BIND« (Mehrere Schwachstellen in BIND; betrifft die Versionen 4 und 8)

CA-2002-15
: »Denial-of-Service Vulnerability in ISC BIND 9« (Denial of Service-Sicherheitslücke im ISC BIND 9)

CA-2000-03
: »Continuing Compromises of DNS Servers« (Anhaltende Kompromittierung von DNS-Servern)

CA-99-14
: »Multiple Vulnerabilities in BIND« (Verschiedene Sicherheitslücken in BIND)

CA-98.05
: »Multiple Vulnerabilities in BIND« (Verschiedene Sicherheitslücken in BIND)

CA-97.22
: »BIND« (Cache-Poisoning)

Ressourcen zu BIND

Internet Software Consortium, »BIND Operator's Guide« (»BOG«).
: Wird separat vom BIND 8-Source-Code zum Herunterladen angeboten. Die aktuelle Version befindet sich unter *ftp://ftp.isc.org/isc/bind/src/8.3.3/bind-doc.tar.gz*. Der BOG ist der wichtigste und nützlichste Teil der offiziellen BIND 8-Dokumentation.

Internet Software Consortium, »BIND 9 Administrator Reference Manual«.
: Dieses Handbuch ist fester Bestandteil der BIND 9-Source-Code-Distributionen. Es befindet sich im Verzeichnis *doc/arm*, der Dateiname ist *Bv9ARM.html*. Es steht auch im PDF-Format unter *http://www.nominum.com/resources/documentation/Bv9ARM.pdf* zur Verfügung. Das ARM ist der wichtigste und nützlichste Teil der offiziellen BIND 9-Dokumentation.

Internet Software Consortium, »Internet Software Consortium: BIND«
: (*http://www.isc.org/products/BIND/)*. Die ultimative Quelle für sämtliche BIND-Software und -Dokumentation.

Liu, Cricket, »Securing an Internet Name Server«.
: Eine Präsentation, die unter *http:// www.acmebw.com/papers/securing.pdf* zur Verfügung steht. Sie stammt von Cricket Liu, einem der Autoren des Buchs *DNS und BIND* (O'Reilly) (alias »The Grasshopper Book«, Das Grashüpfer-Buch).

Ressourcen zu djbdns

Bernstein, D. J., »djbdns: Domain Name System Tools« (*http://cr.yp.to/djbdns.html*).
: Die ultimative Quelle für *djbdns*-Software und -Dokumentation.

Brauer, Henning, »Life with djbdns« (*http://lifewithdjbdns.org*).
: Ein umfassendes Handbuch zur Verwendung von *djbdns*, das Beispiel-Konfigurationen und Links auf andere Websites beinhaltet.

Nelson, Russell, »djbdns Home Page« (*http://www.tinydns.org*).
Von anderen Leuten beigesteuerter Code und Bezugsquellen für Support.

Luterman, Greg, »Grumpy Badger's Introduction to djbdns« (http://djbdns.wolf-home.com/).
Eine behutsame Einführung.

»FAQTS – Knowledge Base... djbdns« (*http://djbdns.faqts.com/)*.
Brian Coogans gesammelte Informationen zu djbdns.

»Linux notebook/djbdns« (*http://binarios.com/lnb/djbdns.html)*.
Nützliche Tabellen, Skripten und Hinweise für und zu djbdns.