DNS & BIND

Markus Dahms

23. November 2002

1 Sinn & Unsinn von DNS

DNS, das *„Domain Name System“*, macht aus Rechnernamen IP-Adressen, also Nummern. Prinzipiell funktioniert das Internet ganz gut ohne benannte Rechner, doch man hat sich ziemlich daran gewöhnt.
In den 70er Jahren, damals, das Internet hieß noch `ARPANET` und wahr noch überschaubar, war es üblich, alle bekannten Rechnernamen mit den dazugehörigen IPs in der Datei `/etc/hosts` zu hinterlegen. Diese wurde zur damaligen Zeit aus der Datei `HOSTS.TXT` generiert, welche per FTP vom *Network Information Center* des *Stanford Research Institute* (`SRI-NIC`) bezogen werden konnte.
Wie man sich vorstellen kann, wurde diese Möglichkeit der Namensauflösung immer unhandlicher, da die Zahl der am Internet hängenden Rechner in den folgenden Jahren explodierte.
Nebenbei hatte niemand wirklich die Kontrolle darüber, wer wie Namen im Internet vergibt, sodass Inkonsistenzen wie doppelt vergebene Namen ein großes Problem waren.
1984 wurde dann mit den RFCs 882 & 883 der Grundstein für das DNS gelegt, mittlerweile wurden diese durch die RFCs 1034 & 1035 ersetzt.

2 Aufbau & Struktur

Grundstruktur des *Domain Name System* ist ein Baum, ein hierarchisch aufgebautes Gebilde. So kann man ein Hauptmerkmal realisieren, das DNS ist eine große, *verteilte* Datenbank.
Ein Rechnername besteht nun aus der Aneinanderreihung der Namen der Knoten dieses Baums (jeweils durch einen Punkt getrennt), die man durchlaufen muss, um von einem Blatt zur Wurzel zu gelangen.

Jetzt gibt es Rechner, üblicherweise *Name Server* bzw. eingedeutscht Nameserver genannt, die über einen Teil dieses Baums Bescheid wissen. Möchte man eine fremde Adresse herausbekommen, so fragt man am Besten den Nameserver im lokalen Netz. Kennt dieser die Adresse nicht, konsultiert er einen *Root Server*. Dies ist ein Nameserver, der sich in der obersten Hierarchieebene des DNS-Baums auskennt, den sogenannten *Top Level Domains* (`TLDs`). Diese befinden sich also in einer validen Rechneradresse, welche aus dem Hostnamen des Rechners sowie den Domainnamen besteht, hinter dem letzten Punkt.
Jetzt werden solange die für die ensprechenden Äste im DNS-Baum (*Domains* bzw. *Subdomains*) zuständigen Nameserver gefragt, bis einer die vollständige IP-Nummer des Computers weiß.
Im Sinne des Internet sind die Nameserver meistens redundant, jede Domain sollte mindestens zwei Nameserver besitzen, Root Server gibt es sogar 13 an der Zahl.
Dabei unterscheidet man zwischen primären und sekundären Nameservern. Erstere haben einen eigenen, konfigurierten Datenbestand, letztere syncronisieren sich mit den primären Servern oder dienen nur als Cache.

3 BIND - eine kleine Einführung

Der bekannteste unter den Nameservern ist BIND, ausgeschrieben *Berkeley Internet Name Domain*.
Er ist als Open Source vom *Internet Software Consortium*[1] verfügbar. Neben Linux lässt sich BIND auch auf anderen unixoiden und nichtunixoiden Systemen einsetzen.

3.1 Installation

Der vermutlich einfachste Weg der Installation ist das Benutzen des Pakets, welches jede große Linux-Distribution mitliefert.
Alternativ dazu kann man natürlich auch die Quellen vom ISC[2] herunterladen und selbst kompilieren. Welchen Weg man auch immer wählt, nachher hat man einen Nameserver (zum Beispiel `/usr/sbin/named`), ein Programm, das DNS-Anfragen stellen kann (meist `/usr/bin/nslookup` und / oder `/usr/bin/host`), sowie einige Bibliotheken, die von eigenen Programmen genutzt werden können.

[1] ISC, http://www.isc.org
[2] http://www.isc.org/products/BIND/

3.2 Konfiguration

Wie unter UNIX üblich, konfiguriert man den BIND durch eine Reihe von Textdateien, welche zu editieren unerlässlich ist.
Die Position dieser Dateien ist von Distribution zu Distribution zum Teil verschieden, weshalb ich hier die Konfiguration von *Debian GNU/Linux* zugrunde lege. In den Beispielen werden im übrigen Konfigurationdateien von BIND Version 9 verwendet.

3.2.1 /etc/bind/named.conf

Zuerst ein paar generelle Optionen, bei dem doppelten Slash („`//`“) und der Raute („`#`“) handelt es sich übrigens um Kommentare.

```
// Anfang "/etc/bind/named.conf"

options {
        directory "/var/cache/bind";
        // query-source address * port 53;
};
```

Zuerst wird das Zone- und Cache-Verzeichnis festgelegt. Die zweite Option bestimmt, auf welchen Adressen und Ports der Server läuft, da mir der Standard genehm ist, kann diese Option auskommentiert bleiben.

```
key "key" {
        algorithm       hmac-md5;
        secret "...";
};
```

Legt einen Schlüssel namens „key“ zur Zugriffskontrolle fest.

```
controls {
        inet 127.0.0.1 allow { 127.0.0.1; } keys { "key"; };

};
```

Erlaubt von `localhost` Zugriffe, die sich mit dem Schlüssel „key“ authentifizieren können.

Als nächstes werden die sogenannten Zonen, bei denen es sich bei genauerer Betrachtung um Domains im Sinne des DNS handelt, konfiguriert.

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

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

Die *Root Zone*. In `db.root` sind alle Root Server angegeben. Für mein vom Internet abgeschottetes Beispielnetz habe ich diese durch eine quasi leere Datei (`db.noroot`) ersetzt, sodass bei Anfragen auf Adressen außerhalb dieses Netzes schnell ein Fehler zurückgegeben wird. Anderenfalls würde der Nameserver ziemlich lange auf ein Timeout warten, da er die Root Server nicht kontaktieren kann.

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

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

zone "0.in-addr.arpa" {
        type master;
        file "/etc/bind/db.0";
};

zone "255.in-addr.arpa" {
        type master;
        file "/etc/bind/db.255";
};
```

Dies sind einige Standardadressen, die jeder Nameserver auflösen können sollte, insbesondere `localhost` und Broadcast-Adressen.

```
zone "madsworld.lan" {
        type master;
        file "/etc/bind/db.madsworld.lan";
};

zone "1.168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/db.192.168.1";
};

// Ende "/etc/bind/named.conf"
```

Mein kleines Beispielnetz. Die TLD „.lan“ ist bis jetzt noch nicht vergeben und wird einfach von mir mißbraucht.
Man beachte die zwei `zone`-Sektionen: erstere ist für die Auflösung von Rechnernamen zuständig, die zweite legt Informationen für die Rückwärtsauflösung fest.

Der folgende Abschnitt soll den Aufbau und die Funktionsweise der Zonendateien klären.

3.2.2 /etc/bind/db.*

Als Beispiel müssen wieder die Zone-Konfigurationen von „madsworld.lan“ herhalten, zuerst `/etc/bind/db.madsworld.lan`.

```
$TTL    43200
```

Time To Live - die Zeit, nach welcher ein beliebiger Nameserver spätestens die Informationen über diese Domain überprüfen sollte.

```
@  IN  SOA  scotty.madsworld.lan.  hostmaster.madsworld.lan. (
               1          ; Serial  (increment after change)
               3600       ; Refresh (in seconds) - 1h
               900        ; Retry   (in seconds) - 15m
               1209600    ; Expire  (in seconds) - 14d
               43200      ; Default TTL (in seconds) - 12h
               )
```

Der *Start Of Authority*-Record.
Das At-Zeichen (`@`) dient als Platzhalter für die aktuelle Zone (in diesem Fall wäre das `madsworld.lan`).
In diesem Abschnitt werden Timeouts und andere Zeitparameter für die Zone festgelegt.

```
   IN  MX   5  bigtin.madsworld.lan
```

Der *Mail Exchanger*-Record.
Hier wird festgelegt, wer die Mails für die Zone annimmt und weiterleitet.

```
   IN  NS   scotty.madsworld.lan
```

Der *Name Server*-Record.
Dies bestimmt den zuständigen Nameserver.

```
   IN  A    192.168.1.100
```

Der *Address*-Record.
Welche IP-Adresse ist der Zone zugeordnet.

```
bigtin       IN  A     192.168.1.100
             IN  A     192.168.1.101
             IN  WKS   192.168.1.100 TCP ssh http https pop-3
             IN  HINFO "2 x Pentium III 1000" "Debian GNU/Linux"

www          IN  CNAME bigtin.madsworld.lan.
mail         IN  CNAME bigtin.madsworld.lan.

scotty       IN  A     192.168.1.4
             IN  MX    scotty.madsworld.lan.
             IN  HINFO "Pentium 75" "Debian GNU/Linux"

slippy       IN  A     192.168.1.2
             IN  HINFO "Compaq Aero Contura 4/25" "DOS / XLinux"
             IN  TXT   "it's a nice old notebook"
```

Für jeden Rechnernamen folgen nun weitere Einträge, wobei ein *Address*-Record (`A`) pro Name natürlich das Wichtigste ist. Zusätzlich kann man noch Spitznamen (`CNAME`), Systeminformationen (`HINFO`) oder weitere Infos (`TXT`) unterbringen.
Und jetzt das Ganze in die andere Richtung, aus einer IP-Adresse werde ein Name: `/etc/bind/db.192.168.1`

```
$TTL    43200

@   IN  SOA  scotty.madsworld.lan.  hostmaster.madsworld.lan. (
               1          ; Serial  (increment after change)
               3600       ; Refresh (in seconds) - 1h
               900        ; Retry   (in seconds) - 15m
               1209600    ; Expire  (in seconds) - 14d
               43200      ; Default TTL (in seconds) - 12h
               )
    IN  NS   scotty.madsworld.lan
```

Dieser Abschnitt ist bereits aus der vorherigen Datei bekannt.

```
2   IN  PTR  slippy.madsworld.lan.
4   IN  PTR  scotty.madsworld.lan.
100 IN  PTR  bigtin.madsworld.lan.
101 IN  PTR  bigtin.madsworld.lan.
```

Ein *Pointer*-Record (`PTR`) ist nun dafür zuständig, einer IP-Adresse wieder einen Namen zuzuordnen.

Literatur

[1] Paul Albitz & Cricket Liu: *DNS and BIND in a Nutshell*, O'Reilly & Associates, Inc. (1995)