6 Standarddienste in TCP/IP Netzwerken

In diesem Themenblock werden wir uns mit Diensten beschäftigen, die auf TCP/IP aufbauen. Dabei geht es in erster Linie darum, die angebotenen Dienste zu sichern, d.h. ein Angreifer soll es möglichst schwer haben, unerlaubt auf den Rechner zu kommen, der die Dienste anbietet. Da man nicht nur mittels Telnet auf einen Unix Rechner gelangen kann, sollte man auch die anderen Dienste begutachten.
Informationen zu den hier vorgestellten Diensten und noch vieles mehr finden Sie unter [Plat 02].

6.1 Domain Name System

In Kapitel 1 haben wir uns mit den Grundlagen von TCP/IP auseinandergesetzt. Dabei wurde deutlich, daß eine Grundlage der Kommunikation zwischen Rechnern die IP-Adresse darstellt. Menschliche Benutzer haben oft Schwierigkeiten, die numerischen IP-Adressen zu benutzen, und bevorzugen aussagekräftige und vor allem merkbare Namen. Außerdem ist es ein großer Nachteil der IP-Adressen, daß aus ihnen keinerlei geographische Information zu entnehmen ist. Man sieht einer Zieladresse nicht an, ob sie in Australien oder im Nebenzimmer lokalisiert ist.
Daher wurde das **Domain Name System** entwickelt, das den Aufbau von Rechnernamen regelt. Es ordnet jedem (weltweit eindeutigen) Namen eine IP-Adresse zu. Dabei gibt es einige Varianten. Eine Maschine mit einer IP-Adresse kann mehrere Funktionen haben und daher auch mehrere Namen, die auf diese Funktionen hinweisen. Genauso kann eine Maschine (z. B. ein Router) viele IP-Adressen haben aber nur einen Namen.

6.1.1 Namensraum und Adreßauflösung

Namen im DNS sind hierarchisch aufgebaut. Das gesamte Internet ist in Domains aufgeteilt, welche wiederum durch Subdomains strukturiert werden. In den Subdomains setzt sich die Strukturierung fort. Diese Hierarchie spiegelt sich im Namen wieder. Die entsprechenden Domains werden durch Punkte getrennt.
Beispiele:

```
www.uni-muenchen.de (141.84.120.25)
ftp.microsoft.com (207.46.133.140)
```

Es gibt verschiedene Möglichkeiten, die Namensauflösung zu testen. Eine ist der Befehl `nslookup`. Dies würde im einfachsten Fall folgendermaßen aussehen:

```
nslookup www.sun.de 53.122.34.10
```

```
Server:  nsi-str.daimlerchrysler.com
Address:  53.122.34.10

Non-authoritative answer:
Name:    www.sun.de
Address:  212.125.100.80
```

Die **Top Level Domain** (**TLD**) (im Beispiel: `de`) steht ganz rechts und wird durch den Country-Code abgekürzt (weitere Beispiele: `at` für Österreich, `au` für Australien, `fr` für Frankreich, `uk` für Großbritannien, ...). In Tabelle 9 sind einige, nicht länderbezogene Domains aufgelistet, die aber großteils in den USA verwendet werden.

TLD	**hauptsächliche Verwendung**
`com`	Kommerzielle Organisationen
`edu`	(education) Schulen und Hochschulen
`gov`	(government) Regierungsinstitutionen
`mil`	militärische Einrichtungen
`net`	Netzwerk betreffende Organisationen
`org`	Nichtkommerzielle Organisationen
`int`	Internationale Organisationen
`arpa`	ARPA-Net bzw. Rückwärtsauflösung von Adressen

Tabelle 9: Einige Top Level Domains

Ende 2000 sind neue TLDs von der **ICANN** (**Internet Corporation for Assigned names and Numbers** [ica 02]) genehmigt worden, die in Tabelle 10 aufgeführt sind.

TLD	**hauptsächliche Verwendung**
`aero`	Luftfahrtindustrie
`coop`	Firmen-Kooperationen
`museum`	Museen
`pro`	Ärzte, Rechtsanwälte und andere Freiberufler
`biz`	Business (frei für alle)
`info`	Informationsanbieter (frei für alle)
`name`	Private Homepages (frei für alle, nur dreistufige Domains Vorname.Name.name)

Tabelle 10: Neu genehmigte Top Level Domains

Unterhalb der Top Level Domain treten dann Domains wie `uni-muenchen` auf, die sich im Rahmen ihrer Organisationen auf diesen Namen, wie auch über die weitere Strukturierung

des Namensraumes, etwa daß Fachbereiche eine Subdomain bilden (z. B. `informatik`), geeinigt haben. Diese werden wieder strukturiert durch die Namen der einzelnen Lehrstühle und Institute. Als letztes Glied wird der einzelne Rechner mit seinem Hostnamen spezifiziert, z.B. `www.nm.informatik.uni-muenchen.de` ist so zu verstehen:
Rechnername.Lehrstuhl.Fachbereich.Organisation.TopLevelDomain
Dabei kann auch manchmal der Rechnername weggelassen werden. Die IP-Adressen, die z.B. bei `sun.de` aufgelöst werden, sind als Default-Rechner für diese Domain zu betrachten. Für die Aufnahme einer Verbindung zwischen zwei Rechnern muß in jedem Fall der Rechnername in eine zugehörige IP-Adresse umgewandelt werden.
Dabei ist `nslookup` der auszuführende Befehl, `www.sun.de` der aufzulösende Name und `53.122.34.10` die IP-Adresse des Nameservers, der abgefragt werden soll. Alle möglichen Optionen zu `nslookup` finden Sie auf Unix Rechnern mit `man nslookup`.
Eine weiter Möglichkeit ist der Befehl `host` (siehe auch `man host`). Eine Abfrage sieht hier so aus:

```
host www.sun.de 53.122.34.10
www.sun.de              A       212.125.100.80

host -t any sun.de 53.122.34.10
sun.de                  SOA     dns.sun.de hostmaster.germany.sun.com (
                        2002040955      ;serial (version)
                        28800   ;refresh period (8 hours)
                        14400   ;retry interval (4 hours)
                        604800  ;expire time (1 week)
                        86400   ;default ttl (1 day)
                        )
sun.de                  NS      dns1.cyberways.net
sun.de                  NS      dns2.cyberways.net
sun.de                  NS      dns.sun.de
sun.de                  A       212.125.100.80
```

Die Option `-t any` besagt, daß von der Domain `sun.de` alle verfügbaren Informationen angezeigt werden sollen. Auch der Befehl `dig` sei hier erwähnt (`man dig`).
Aus Sicherheitsaspekten ist es wünschenswert, auch den umgekehrten Weg zu gehen, nämlich zu einer sich meldenden Adresse den Namen und damit die organisatorische Zugehörigkeit offenzulegen.

```
host 53.122.207.1
Name: www.lufthansa.com
Address: 53.122.207.1
```

Es muß aber darauf hingewiesen werden, daß ein Rechner, der sich in der `.de` Domain befindet, nicht automatisch auch in Deutschland stehen muß. Es bedeutet nur, daß die

Organisation, die den Rechner in ihrer Domain beheimatet, die Top Level Domain in Deutschland beim DeNIC [den 02] beantragt hat.
Ebenso ist zu erwähnen, daß die verwendete Baumstruktur bei den DNS Namen nicht impliziert, daß auch die IP Struktur tief sein muß.

6.1.2 Nameserver und Resolver

Damit das DNS funktioniert, muß es Instanzen geben, die Namen in IP-Adressen und IP-Adressen in Namen umwandeln ('auflösen') können. Diese Instanzen sind durch Programme realisiert, die an größeren Maschinen ständig (meist im Hintergrund) im Betrieb sind und **Nameserver** heißen. Jeder Rechner, der an das Internet angeschlossen wird, muß die Adresse eines oder mehrerer Nameserver wissen, damit die Anwendungen auf diesem Rechner mit Namen anstatt IP-Adressen benutzt werden können. Die Nameserver sind für bestimmte Bereiche, sogenannte **domains** oder **Zonen**, zuständig (Institute, Organisationen, Regionen) und haben Kontakt zu anderen Nameservern, so daß jeder Name aufgelöst werden kann.
Insgesamt gibt es drei Hauptkomponenten, aus denen sich das DNS zusammensetzt:

- Der **Domain Name Space** ist ein baumartig, hierarchisch strukturierter Namensraum. **Resource Records**, auch **Zonenfiles** genannt, sind Datensätze, die den Knoten in diesem Baum zugeordnet sind.

- **Nameserver** sind Programme bzw. Rechner, die die Informationen über die Struktur des Domain Name Space verwalten und aktualisieren. Ein Nameserver hat normalerweise nur eine Teilsicht des Domain Name Space zu verwalten. Oft wird auch der Rechner, auf dem das Nameserverprogramm läuft, als **Nameserver** oder **DNS-Server** bezeichnet.

- **Resolver** sind die Programme, die für den Client Anfragen an den Nameserver stellen. Resolver sind einem Nameserver zugeordnet; bei Anfragen, die dieser zugeordnete Nameserver nicht beantworten kann (z.B. weil der Name in einem anderer Teilbereich des Domain Name Space liegt), kann er aufgrund von Referenzen andere Nameserver kontaktieren, um die Information zu erhalten.

Die Baumstruktur des DNS soll nun im Weiteren untersucht werden. Ausgehend von der **Wurzel** (**Root**) folgen die Top Level Domains. Diese Top Level Domains spalten sich in weitere Unterdomains auf.
Die Funktionsweise ist in Bild 57 dargestellt.

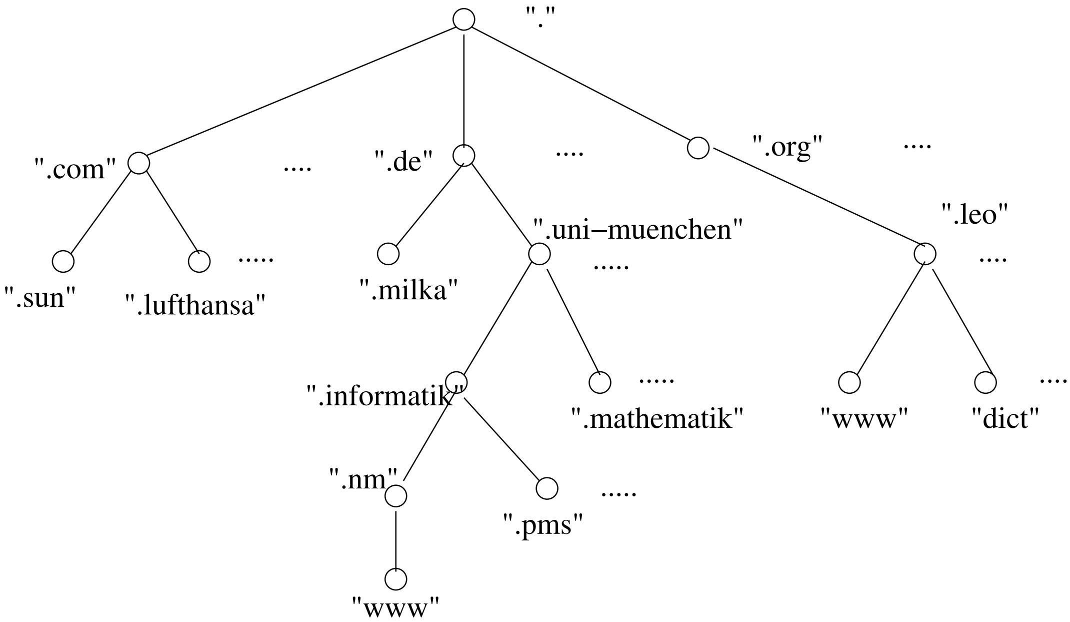

Abbildung 57: Prinzipielle Funktionsweise einer DNS Abfrage

Ein Nameserver des DNS verwaltet also einzelne Zonen, die einen Knoten im DNS-Baum und alle darunterliegenden Zweige beinhalten. Auf jeder Ebene des DNS-Baumes kann es Namesever geben, wobei jeder Nameserver seinen nächsthöheren und nächstniedrigeren Nachbarn kennt. Aus Sicherheitsgründen gibt es für jede Zone in der Regel mindestens zwei Nameserver (**primary**, auch **Master** genannt, und **secondary**, auch **Slave** genannt), wobei beide die gleiche Information halten. Nameservereinträge können nicht nur die Zuordnung Rechnername - IP-Adresse enthalten, sondern (neben anderem) auch weitere Nameneinträge für einen einzigen Rechner und Angaben für Mailverwaltungsrechner einer Domain (**MX**, **mail exchange**).
Basis des Nameservice bilden die Root Nameserver, die für die Top Level Domains zuständig sind. Die Mehrheit dieser Server ist in den USA beheimatet (siehe Abbildung 58 und Tabelle 11).

Wenn wir uns nun vorstellen, wir wollen von unserem Rechner aus den Server `ftp.leo.org` anpingen, so muß zuerst der Name `ftp.leo.org` in eine IP-Adresse aufgelöst werden. Dazu fragt unser Nameserverclient (Resolver) bei dem Nameserver nach, der in der Datei `/etc/resolv.conf` als Nameserver aufgeführt ist. Dieser ist aber nur verantwortlich für Namen in der Domain `secp.nm.informatik.uni-muenchen.de`. Er hat aber die Information, daß er für '.', also alles, wofür er nicht verantwortlich ist, die Root Nameserver befragen soll. Diese wissen dann, welcher Nameserver für `.org` zu befragen ist. Dieser kennt dann den Nameserver, der `leo.org` verwaltet und der kann Auskunft über `ftp.leo.org` geben. Die Abbildung 59 zeigt die prinzipielle Funktionsweise einer DNS Abfrage.

Abbildung 58: Location der Rootnameserver

Diese Informationen werden dann auf dem ersten Nameserver in der Kette zwischengespeichert, um eine weitere Anfrage zu `ftp.leo.org` selbstständig beantworten zu können. Das nennt man **Caching**. Es wäre aber fatal, wenn die Informationen für immer gespeichert bleiben würden. Änderungen würden dann nie publik werden. Deshalb gibt es die sogenannte **Time to Live** (**TTL**), vgl. Bsp. Seite 147. Diese TTL gibt an, wie lange ein Nameserver die Information behalten darf, bevor er bei dem für die Zone verantwortlichen Nameserver die aktuellen Infos erneut holen soll. Eine lange TTL bedeutet, daß die Daten lange gespeichert werden und damit die Anzahl der Anfragen geringer ist. Daraus folgt aber auch, daß Änderungen in einer Zone sich nur langsam verbreiten. Bei einer kurzen TTL werden die im Netz bekannten Daten schneller aktualisiert, die Performance sinkt aber durch die recht hohe Anzahl von Anfragen. Standardwert für die TTL bei `.de` Subdomains liegt bei einer Woche. Stehen Änderungen an, wird die TTL schrittweise vom Administrator für die Zone verringert.

6.1.3 Die Software „BIND"

Der Nameservice wird unter Unix beispielsweise von einem Programm namens **named** bewerkstelligt [AlLi 98]. Dieses Programm ist Teil des bind-Paketes, das von Paul Vixie für das Internet Software Consortium verwaltet wird. named ist in den meisten Linux-Distributionen enthalten und wird üblicherweise als `/usr/sbin/named` installiert. Falls kein Paket zur Verfügung steht, kann man ein Binärpaket von einem Linux-FTP-Server bekommen oder man holt sich den aktuellsten Source-Code von [bin 02]. In der `/etc/nsswitch.conf` steht in der Zeile `hosts`, in welcher Reihenfolge die `/etc/hosts` und

Name	**Typ**	**Betreiber**	**URL**
a	com	InterNic	`http://www.internic.org/`
b	edu	ISI	`http://www.isi.edu/`
c	com	PSINet	`http://www.psi.net/`
d	edu	UMD	`http://www.umd.edu/`
e	usg	NASA	`http://www.nasa.gov/`
f	com	ISC	`http://www.isc.org/`
g	usg	DISA	`http://nic.mil/`
h	usg	ARL	`http://www.arl.mil/`
i	int	NordUnet	`http://www.nordu.net/`
j	()	(TBD)	`http://www.iana.org/`
k	int	RIPE	`http://www.ripe.net/`
l	()	(TBD)	`http://www.iana.org/`
m	int	WIDE	`http://www.wide.ad.jp/`

Tabelle 11: Betreiber der Root DNS Server

der DNS verwendet werden. Default ist, daß zuerst die Informationen aus der `/etc/hosts` verwendet werden und erst, wenn die gewünschte Information dort nicht vorhanden ist, der in der `/etc/resolv.conf` eingetragene Nameserver verwendet wird. In die Datei `/etc/hosts` können lokale Zuordnungen von IP-Adresse zu Rechnernamen eingetragen werden.

Die `/etc/resolv.conf`:

```
nameserver 10.50.208.1
search muc.meinedomain.de
```

Wobei die Einträge folgendermaßen zu verstehen sind:

- `nameserver` gibt an, welcher Nameserver gefragt werden soll.
- `domain` legt den lokalen Domainnamen fest (kann durch `search` ersetzt werden).
- `search` listet die Domains auf, die durchsucht werden sollen, wenn keine Domain bei einem Rechnernamen angegeben ist.

Das Startscript des Nameservers befindet sich unter `/etc/init.d/named`. Um den Nameserver bereits beim Hochfahren des Rechners zu aktivieren, muß bis SuSE 7.3 die `/etc/rc.config` den Eintrag `START_NAMED=yes` enthalten. Ab SuSE 8.0 werden durch das Tool `YaST` Softlinks von den Runlevelverzeichnissen auf das Startscript `/etc/init.d/named` gelegt (dies ist auch von Hand möglich, Informationen zu den Runlevels finden Sie im Startscript).

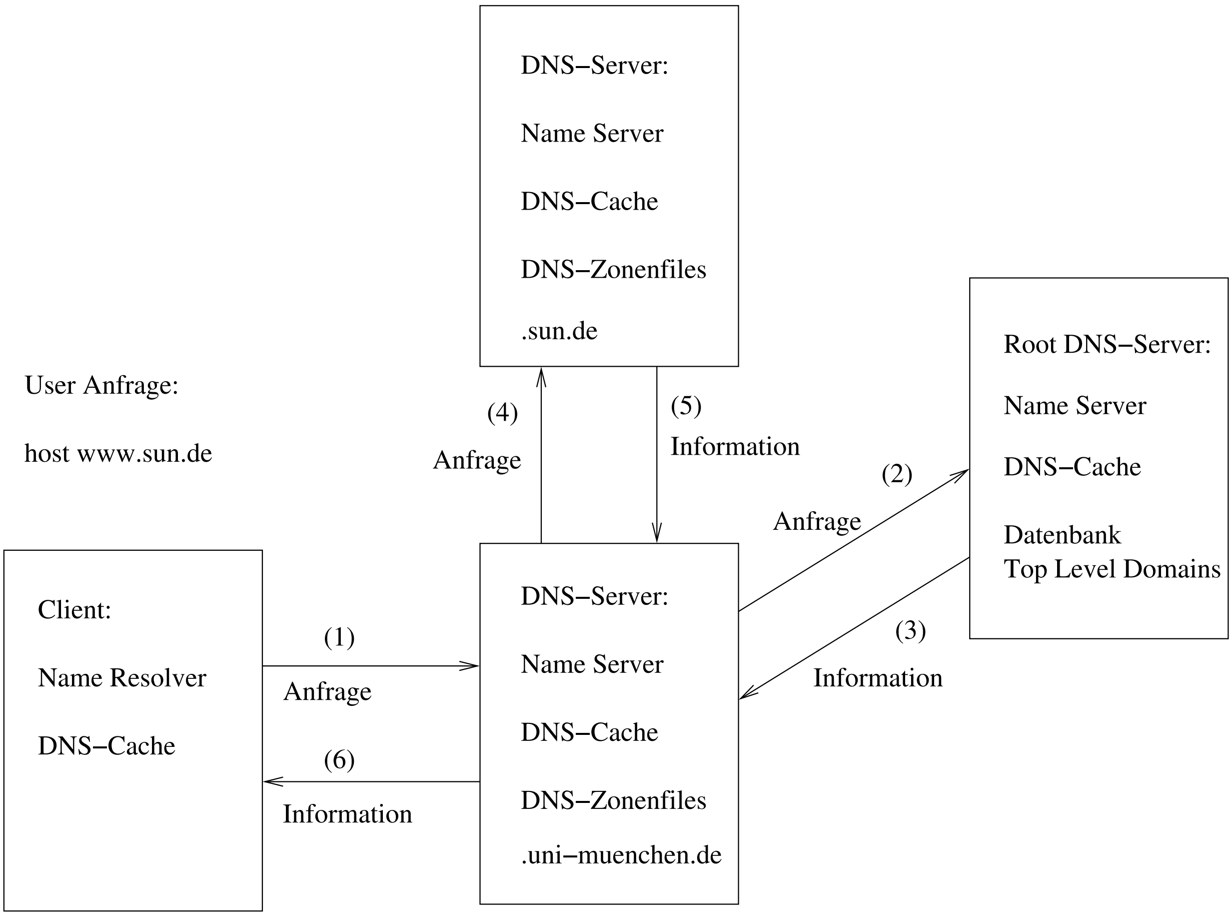

Abbildung 59: Prinzipielle Funktionsweise einer DNS Abfrage

Beim Starten von `named` werden Einträge in die `/var/log/messages` (kann bei anderen Systemen auch die `/var/log/syslog` oder die `/var/adm/messages` sein) geschrieben.
Überprüfung der Funktionstüchtigkeit des Nameservers kann mit Hilfe von `nslookup`, `host` oder `dig` erfolgen. In Kapitel 6.1.1 stehen einige Beispiele hierfür.

Caching-DNS

Ein **caching-only Nameserver** wird auf Namensanfragen antworten und sich bei der nächsten Anfrage an die alte Antwort erinnern. Dieses Vorgehen verkürzt vor allem die Wartezeit bei jeder weiteren Anfrage.
Die Konfigurationsdatei für den Bind heißt `/etc/named.conf`. Sie wird bei jedem Start vom `named` eingelesen. Im Folgenden ist der Standardrahmen der Datei `named.conf`, der von SuSE mitgeliefert wird, angegeben. Dabei ist '`#`' das Kommentarzeichen. Vorerst sollte sie nur das Folgende enthalten:

```
# Copyright (c) 2001 SuSE GmbH Nuernberg, Germany
#
# Author: Frank Bodammer <feedback@suse.de>
#
# /etc/named.conf
#
# This is a sample configuration file for the name server BIND8.
# It works as a caching only name server without modification.
#
# A sample configuration for setting up your own domain can be
# found in /usr/share/doc/packages/bind8/sample-config.
#
# A description of all available options can be found in
# /usr/share/doc/packages/bind8/html/options.html

options {

        # The directory statement defines the name server's
        # working directory

        directory "/var/named";

        # The forwarders record contains a list of servers to
        # which queries should be forwarded. Enable this line and
        # modify the IP-address to your provider's name server.
        # Up to three servers may be listed.

        #forwarders { 10.11.12.13; 10.11.12.14; };

        # Enable the next entry to prefer usage of the name
        # server declared in the forwarders section.

        #forward first;

        # The listen-on record contains a list of local network
        # interfaces to listen on. Optionally the port can be
        # specified. Default is to listen on all interfaces found
        # on your system. The default port is 53.

        #listen-on port 53 { 127.0.0.1; };

        # The next statement may be needed if a firewall stands
        # between the local server and the internet.
```

```
        #query-source address * port 53;

        # The allow-query record contains a list of networks or
        # IP-addresses to accept and deny queries from. The
        # default is to allow queries from all hosts.

        #allow-query { 127.0.0.1; };

        # The cleaning-interval statement defines the time interval
        # in minutes for periodic cleaning. Default is 60 minutes.
        # By default, all actions are logged to /var/log/messages.

        cleaning-interval 120;

        # Name server statistics will be logged to /var/log/messages
        # every <statistics-interval> minutes. Default is 60 minutes.
        # A value of 0 disables this feature.

        statistics-interval 0;

        # If notify is set to yes (default), notify messages are
        # sent to other name servers when the the zone data is
        # changed. Instead of setting a global 'notify' statement
        # in the 'options' section, a separate 'notify' can be
        # added to each zone definition.

        notify no;
};

# The following three zone definitions don't need any modification.
# The first one defines localhost while the second defines the
# reverse lookup for localhost. The last zone "." is the
# definition of the root name servers.

zone "localhost" in {
        type master;
        file "localhost.zone";
};

zone "0.0.127.in-addr.arpa" in {
        type master;
        file "127.0.0.zone";
```

```
};

zone "." in {
        type hint;
        file "root.hint";
};

# You can insert further zone records for your own domains below.
```

Im Bereich `options` können verschiedene Optionen für den named-Daemon gesetzt werden. Diese werden später noch im Einzelnen erklärt. Die `directory`-Zeile sagt dem `named`, wo er nach Dateien suchen soll. Alle noch folgenden Dateien gehören in dieses Verzeichnis (oder in ein Unterverzeichnis relativ hierzu). `/var/named` ist nach dem **Linux File System Standard** [lin 01] das für den Nameserver vorgesehene Verzeichnis für die Informationsdateien eines Nameservers.
Die Datei `/var/named/root.hint` beinhaltet die Root Nameserver:

```
.                         3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.       3600000      A     198.41.0.4
;
; formerly NS1.ISI.EDU
;
.                         3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.       3600000      A     128.9.0.107
;
; formerly C.PSI.NET
;
.                         3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.       3600000      A     192.33.4.12
;
; formerly TERP.UMD.EDU
;
.                         3600000      NS    D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.       3600000      A     128.8.10.90
;
; formerly NS.NASA.GOV
;
.                         3600000      NS    E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.       3600000      A     192.203.230.10
;
; formerly NS.ISC.ORG
;
.                         3600000      NS    F.ROOT-SERVERS.NET.
```

```
F.ROOT-SERVERS.NET.      3600000      A     192.5.5.241
;
; formerly NS.NIC.DDN.MIL
;
.                        3600000      NS    G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.      3600000      A     192.112.36.4
;
; formerly AOS.ARL.ARMY.MIL
;
.                        3600000      NS    H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.      3600000      A     128.63.2.53
;
; formerly NIC.NORDU.NET
;
.                        3600000      NS    I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.      3600000      A     192.36.148.17
;
; temporarily housed at NSI (InterNIC)
;
.                        3600000      NS    J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.      3600000      A     198.41.0.10
;
; housed in LINX, operated by RIPE NCC
;
.                        3600000      NS    K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.      3600000      A     193.0.14.129
;
; temporarily housed at ISI (IANA)
;
.                        3600000      NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.      3600000      A     198.32.64.12
;
; housed in Japan, operated by WIDE
;
.                        3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.      3600000      A     202.12.27.33
; End of File
```

Hier ist die 3600000 die TTL für die Root Nameserver-Informationen. Dieser Eintrag steht nur noch aus historischen Gründen in der `root.hint` Datei und ist eigentlich überflüssig. Die Datei `/var/named/127.0.0.zone` wird bereits standardmäßig bei der Installation angelegt und sieht so aus:

```
$TTL 2D
@               IN SOA          localhost.   root.localhost. (
                                42              ; serial
                                1D              ; refresh
                                2H              ; retry
                                1W              ; expiry
                                2D )            ; minimum

                IN NS           localhost.
1               IN PTR          localhost.
```

`/var/named/localhost.zone` sieht standartmäßig so aus:

```
$TTL 2D
@               IN SOA  @   root (
                                42              ; serial
                                1D              ; refresh
                                2H              ; retry
                                1W              ; expiry
                                2D )            ; minimum

                IN NS           @
                IN A            127.0.0.1
```

- **TTL** ist die sogenannte **time to live**, die angibt, wie lange die in dem Record enthaltenen Informationen gültig sind und gespeichert werden dürfen (2D = 2Tage).
- **@** übernimmt die Informationen aus der Zuordung von Zone zu Dateiname in der `named.conf`.
- **IN** steht für **Internet**.
- **SOA** steht für **start of authority**. Der SOA gibt an, daß dieser Nameserver autoritativ (verantwortlich) für diese Zone ist.
- **localhost.** oder **@** gibt den Namen des Primary Nameservers an.
- **root.localhost.** oder **root** ist die Mailadresse der Person, die für die Einträge zuständig ist, wobei der erste '.' für das '@'-Zeichen in einer Mailadresse steht.
- **serial** ist die **Seriennummer**, die bei jeder Änderung in diesem Zonenfile vergrößert werden muß, damit der Nameserver erkennt, daß sich in diesem File Änderungen ergeben haben.
- **refresh** gibt dem Slave Server an, wie oft er überprüfen soll, ob seine Daten aktuell sind (1D = 1 Tag).

- **retry** gibt an, in welchem Intervall ein Slave versuchen wird, seinen Master zu erreichen, falls der vorherige Versuch dazu fehlgeschlagen ist (2H = 2 Stunden).
- **expiry** ist das Zeitintervall, nachdem die Informationen auf dem Slave abgelaufen sind, wenn er für dieses Intervall nicht in der Lage war, seinen Master zu kontaktieren (1W = 1 Woche).
- **NS** listet alle Nameserver auf, die für diese Zone zuständig sind.
- **A** ist das **Name-to-Address** Mapping.
- **PTR** ist das **Address-to-Name** Mapping.

Es kann manchmal Sinn machen, die Anfragen nicht zu einem der Root Nameserver zu schicken, sondern sich einen sogenannten **Forwarder** einzutragen. Wenn sich der Forwarder an der Schnittstelle des lokalen Netzwerks zum Rest der Welt befindet, kann dadurch die Last außerhalb des lokalen Netzwerks zum Provider hin verkleiner werden, wenn mehrere interne Caching Nameserver diesen zentralen Forwarder abfragen. Die Konfiguration hierfür erfolgt in der `/etc/named.conf` unter den `options`:

```
forward first;
forwarders {
     10.0.0.1;
     10.1.0.1;
 };
```

Wenn ein interner DNS Server vorhanden ist, der nicht über das Internet propagiert wird, kann es vorkommen, daß interne Domains von verschiedenen internen Nameservern behandelt werden, die aber nichts voneinander wissen. Eine Möglichkeit, die Infos für den Cachingserver zugänglich zu machen, ist das **Forwarding** auf die unterschiedlichen Namserver:

```
zone "subdomain1.domain.de" {
        type forward;
        forward only;
        forwarders { 10.10.10.10; 10.11.12.13; };
};

zone "subdomain2.domain.de" {
        type forward;
        forward only;
        forwarders { 192.168.1.1; 172.28.8.1; };
};
```

Master Nameserver

Wenn ein Nameserver Zoneninformationen nicht nur vorhält, sondern die ursprüngliche Quelle der Informationen ist, so wir er als **Master** bezeichnet. Ein Master besitzt das **Masterzonenfile** (**Resource Record**) für eine Zone, das auch nur dort geändert wird. Es gibt pro Zone nur einen Master. Im Konfigurationsfile `/etc/named.conf` stehen nun nach den Defaulteinträgen noch weitere Einträge für die zusätzlichen Zonenfiles, die vom Master verwaltet werden:

```
zone "test.de" {
        type master;
        file "db.test.de";
};
```

Wobei die Namensgebung dem Administrator überlassen ist. Es sollte auch die **Reverse Zonen** für die verwalteten Namen geben. D.h. wenn der IP-Adressraum nicht in der eigenen Verwaltung liegt, muß der Provider darüber informiert werden, was er in seine Reverse Zonen einzutragen hat, ansonsten sehen im eigenen DNS die Einträge im Konfigurationsfile so aus:

```
zone "1.168.192.in-addr.arpa" {
        type master;
        file "db.1.168.192.rev";
};
```

Der Inhalt der Zonenfiles ist wieder analog zu oben:
`db.test.de`:

```
$TTL 2D
@               IN SOA  nameserver-master.provider.de   hostmaster.provider.de (
                                2002041600      ; serial
                                1D              ; refresh
                                2H              ; retry
                                1W              ; expiry
                                2D )            ; minimum
; Angabe der Nameserver
                IN NS           nameserver-master.provider.de.
                IN NS           nameserver-slave1.provider.de.
                IN NS           nameserver-slave2.provider.de.

; Mail Exchange
test.de                  IN MX 20       mailserver.provider.de.
test.de                  IN MX 10       mailserver.test.de.
```

```
; Delegierung einer Subdomain
namserver.test.test.de.  IN A           192.169.5.5
test.test.de.            IN NS          namserver.test.test.de.

; Rechnerinformationen
rechner1                 IN A           192.168.1.1
rechner2                 IN A           192.168.1.2
mailserver.test.de.      IN A           192.168.2.2
```

Es gibt zwei Möglichkeiten, die Namen der Rechner in das Zonenfile einzupflegen:

- nur der Hostname ohne Domain, da diese durch die Zuordnung Domain zu Zonenfile im Konfigurationsfile klar ist
- Hostname.Domain, wobei diese Darstellung durch einen Punkt abgeschlossen sein muß, da sonst der Domainname ein weiteres Mal angehängt wird

Mit **MX** gibt man die Mailzustellungsserver an. D.h. alle Mails, die an `@test.de` gehen, sollen an `mailserver.test.de` zugestellt werden. Wenn dieser Server nicht erreichbar ist, werden die Mails an `mailserver.provider.de` zugestellt. Die Zahlen geben die **Prioritäten** der Mailserver an, der Server mit der niedrigsten Zahl hat höchste Priorität.
`db.1.168.192.rev`:

```
$TTL 2D
@               IN SOA  nameserver-master.provider.de   hostmaster.provider.de (
                                2002041600      ; serial
                                1D              ; refresh
                                2H              ; retry
                                1W              ; expiry
                                2D )            ; minimum

                IN NS           nameserver-master.provider.de.
                IN NS           nameserver-slave1.provider.de.
                IN NS           nameserver-slave2.provider.de.

1.1.168.192.in-addr.arpa.       IN      PTR     rechner1.test.de.
2.1.168.192.in-addr.arpa.       IN      PTR     rechner2.test.de.
```

Slave Nameserver

Da es nur einen Master geben sollte (Datenkonsistenz), es aber wegen der Erreichbarkeit einer Domain ratsam ist, mehr als einen Nameserver mit den Informationen zu haben, werden weitere Nameserver als sogenannte **Secondary**- oder **Slave-Nameserver** konfiguriert. Im Konfigurationsfile `/etc/named.conf` sieht das folgendermaßen aus:

```
zone "1.168.192.in-addr.arpa" {
        type slave;
        file "sec-db.1.168.192.rev";
        masters { 10.10.10.10; };
};
zone "test.de" {
        type slave;
        file "sec-db.test.de";
        masters { 10.10.10.10; };
};
```

Wobei unter `masters` der oder die Nameserver angegeben werden, von denen die Zoneninformationen gezogen werden.

Die meisten Nameserver sind eine Mischung aus Caching-, Master-, Slave und Forwarding-Nameserver.

Sicherheitsaspekte

In der Vergangenheit gab es eine Reihe von Angriffen, die Außenstehenden Rootrechte auf dem Nameserver verschafft haben. Hier eine Liste von Angriffsmustern:

- **Denial of Service Attacken** (**DoS**): Dadurch wird zwar kein Rootzugriff möglich, der Dienst ist aber auch für andere nicht mehr verfügbar. Ziel dieser Attacke ist es, die Nameserver einer Firma lahmzulegen, damit keine Rechner dieser Firma im Internet mehr mit Namen bekannt sind. Dadurch kann hoher finanzieller Schaden entstehen, der Prestigeverlust ist aber meist weit größer.
- **Buffer Overflow / Remote Exploit**: Ein Angreifer überschreibt den Execution Stack des Prozesses, um dann einen von ihm übertragenen Code mit den Rechten des Prozesses auszuführen. Dies wird meist dazu benutzt, Rootrechte auf der Maschine zu erlangen. Dadurch hat der Angreifer alle Administratorprivilegien auf dem System und kann sich weitere Hintertüren und Spionagemittel einbauen. Meist werden von so kompromitierten Rechnern aus weitere Angriffe gefahren.
- **Cache Poison**: Falsche DNS Daten werden in den Cache des Nameservers eingeschleußt. Damit werden Anfragen über andere Maschinen geschickt, die dann falsche Antworten liefern können. So kann erreicht werden, daß alle Mails einer Firma, die an eine Partnerfima geschickt werden sollen, zuerst über eine andere Maschine geleitet, dort gelesen und evtl. modifiziert und dann erst an den eigentlichen Empfänger zugestellt werden. Das kann auch als **DNS Spoofing** bezeichnet werden.
- **Domain Hijack**: Übernahme einer kompletten Domain. Hier schaltet sich ein kompromitierter Nameserver in die Abfragekette ein und gibt direkt die falschen Antworten zurück. Die Auswirkungen sind analog zum Cache Poisoning.

Weiter Informationen zum Ausnutzen von Schwachstellen und den Auswirkungen wurden bereits in Kapitel 2.3 besprochen.
Möglichkeiten, um gegen diese Art der Attacken besser gerüstet zu sein:

- Prozeß nicht als User `root` laufen lassen (Prozeß mit -u ¡User-ID¿ -g ¡Group-ID¿ starten), das ist bei SuSE Default (sowohl die User-ID wie auch die Group-ID ist `named`).
- Bei Masterservern sollte der DNS-User keine Files und Directories schreiben können (Verzeichnis- und Dateirechte), auf dem Slave ist das anders zu betrachten, da ja bei den Zonentransferes die Daten in die sec-db-Files geschrieben werden müssen.
- Es sollte immer der neueste Patchlevel laufen.

Da durch DNS Abfragen Informationen über Netz- und Organisationsstrukturen gewonnen werden können, sollte das verhindert werden. Ein wichtiger Aspekt hierfür ist, daß stets interner und externer DNS getrennt auf verschiedenen Maschinen gepflegt werden sollten. Außerdem sollte es nur für die gewünschten Maschinen möglich sein, Informationen von den Nameservern zu bekommen. Z.B. muß ein Zonentransfer nur für die Slaveserver erlaubt sein, nicht aber von allen IP-Adressen aus. Diese Verhaltensweisen können noch genauer konfiguriert werden.
Im Bereich `options` können noch verschiedene Optionen angeben werden, wie sich der Bind verhalten soll:

- Daemon soll nur auf einem Interface Verbindungen annehmen: `listen-on { 192.168.1.1; };`
- Der Zonentransfer soll nur von bestimmten IP-Adressen aus erlaubt sein: `allow-transfer { 192.168.1.2 ; 10.10.10.10 ; };`
- DNS Queries sollen nur von erlaubten Adressen aus funktionieren: `allow-query { 192.168.1.0/24; 192.168.5.0/25; };` . Hierbei ist aber zu beachten, daß die Zone, für die der Server verantwortlich ist, abgefragt werden kann. Deshalb muß dann im Bereich, in dem die Zonenbehandlung der Master- und Slavezonen erfolgt, `allow-query { any; };` eingetragen werden.
- Von bestimmten Netzen soll keine Aktion erlaubt sein: `blackhole { 172.16.0.0/12; };`
- Um nicht jedem Abfrager sofort die Versionnummer und somit die möglichen Schwachstellen zu verraten, kann mit `version ‘‘Beliebiger Text’’` eine Fehlinformation ausgegeben werden.

Ein gutes DNS-HowTo finden Sie unter [uTW 00].
Die hier aufgezeigten Informationen zur Sicherheit von Bind stammen hauptsächlich aus [Thom 02], [Mart 01] und [Squi 01].

6.2 Telnet

Ein weiterer Netzwerkdienst ist Telnet. Zweck des Telnet-Programms ist es, von einem beliebigen TCP/IP-fähigen Terminal einen interaktiven Zugang zu anderen Computern zu schaffen. Diese Programme gehen im einfachsten Fall von einem Text-Terminal aus und verwenden oft die VT100/VT200-Emulation.

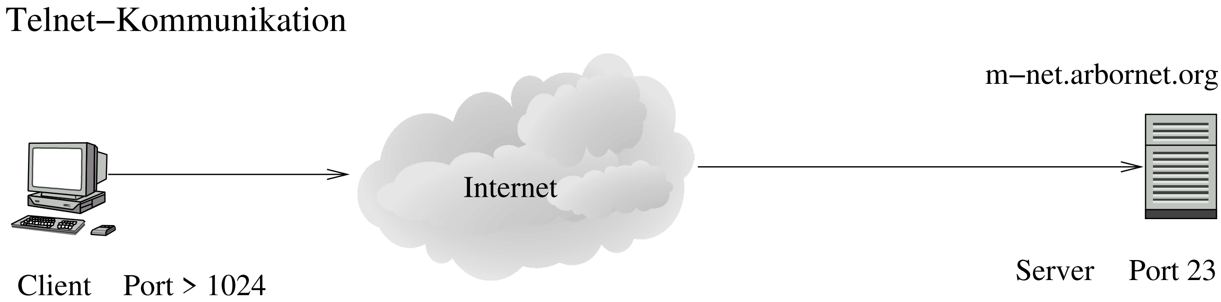

Abbildung 60: Prinzipielle Funktionsweise von Telnet

Für die Steuerung der Ausgabe, beispielsweise die Positionierung der Schreibmarke oder die Einstellung der Bildschirmfarbe, werden Steuerzeichen gesendet, welche die Terminal-Emulation 'verstehen' muß. Man hat also den gleichen Funktionsumfang wie ein lokal an dem jeweiligen Rechner angeschlossenes Terminal - egal wie weit der Rechner entfernt ist. Ist der Verbindungsaufbau erfolgreich, erscheint der Login-Prompt des fernen Rechners. Man kann sich prinzipiell an jedem Rechner im Internet einloggen - vorausgesetzt, man besitzt dort eine Zugangsberechtigung und der Telnetdienst wird angeboten.
Primär dient Telnet aber dem Shell-Zugang auf einem fernen Rechner. Sobald Sie die Verbindung zum fernen Rechner aufgebaut haben, können Sie all das auf dem Rechner machen, was man auch lokal dort an der Konsole machen kann. Für den 'Normalnutzer' spielt Telnet eine untergeordnete Rolle, dagegen kann ein Administrator alle ihm unterstehenden Computer bedienen, ohne seinen Arbeitsplatz verlassen zu müssen.

6.2.1 Sicherheitsrisiken und Schutzmechanismen

Da die Telnetverbindung unverschlüsselt über das Netz geht, kann jeder, der sich an den geeigneten Schnittstellen im Netz befindet, nicht nur den Datenverkehr, sondern auch User-ID und Passwort mitlesen.

Somit sollte, falls das organisatorisch möglich ist, der Zugriff auf Telnet nicht von überallher, sondern nur von bestimmten Bereichen aus erlaubt sein. Dazu ist bei SuSE und vielen andern Linuxdistributionen der `tcpwrapper` bereits installiert. Bei Solaris und einigen anderen Unixderivaten kann es sein, daß der Administrator das noch per Hand machen muß. In `/etc/inetd.conf` wird als Serverpfad der Pfad des `tcpwrapper` (`/usr/sbin/tcpd`) eingetragen und als Argument der Pfad des Telnetbinaries (`/usr/sbin/in.telnetd`). Der `tcpwrapper` sorgt dafür, daß nur die IP-Adressen, Netze, Domains, etc. auf den Dienst zugreifen dürfen, die in der `/etc/hosts.allow` aufgeführt sind. Die Bezeichnung des Dienstes in der `/etc/hosts.allow` muß mit der Bezeichnung in der Spalte für `<args>` in der `/etc/inetd.conf` übereinstimmen.
Hier ein Beispielauszug aus einer `/etc/inetd.conf`:

```
#<service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
telnet stream tcp nowait telnetd.telnetd /usr/sbin/tcpd /usr/sbin/in.telnetd
```

Beim Verbindungsaufbau wird am Zielrechner zuerst der Superserver `inetd` aufgerufen. Dieser prüft, ob für den verlangten Port ein Dienst angeboten wird. Ist das der Fall, so übergibt der `inetd` an den `tcpwrapper`. Dieser überprüft, ob die Quelladresse erlaubt ist, diesen Dienst anzusprechen. Ist das der Fall, so übergibt der `tcpwrapper` an den Telnetdaemon. Trifft das nicht zu, so wird die Verbindung unterbrochen und ein Eintrag in die Logdatei geschrieben. Das Gegenstück zu `/etc/hosts.allow` ist `/etc/hosts.deny`. In dieser Datei werden alle Rechner, Subnetze, etc. eingetragen, die keine Zugriffserlaubnis erhalten sollen.
Informationen zum `tcpwrapper` und `inetd` finden Sie vor allem in den Manpages (`man tcpd` und `man inetd`).
Die sicherste Lösung ist es, auf alle Dienste, die Passwörter im Klartext übertragen, ganz zu verzichten und diese in der `inetd.conf` abzuschalten. Ein Ersatz für Telnet ist beispielsweise die in Kapitel 6.4 besprochene Secure Shell.

6.3 File Transfer Protocol

6.3.1 Informationen zu FTP

Im Unterschied zu einer Telnet-Verbindung, die textorientiert ist, können mittels FTP beliebige Daten ausgetauscht werden (Bilder, Programme, Sounds, usw.). FTP dient primär zum Übertragen von Dateien (**file transfer**).

Der Verbindungsaufbau erfolgt wie bei Telnet, indem man beim FTP-Programm den gewünschten Zielrechner angibt. Bei erfolgreicher Verbindung kommt vom fernen Rechner ein Login-Prompt. FTP funktioniert aber auch, wenn man auf dem fernen Rechner keine Benutzerberechtigung hat, denn viele Rechner bieten große Dateibereiche über sogenannten **anonymous FTP** an. Man gibt in diesem Fall als Benutzernamen `ftp` (manchmal auch `anonymous`) ein und als Passwort die eigene Mailadresse. Danach kann man sich im

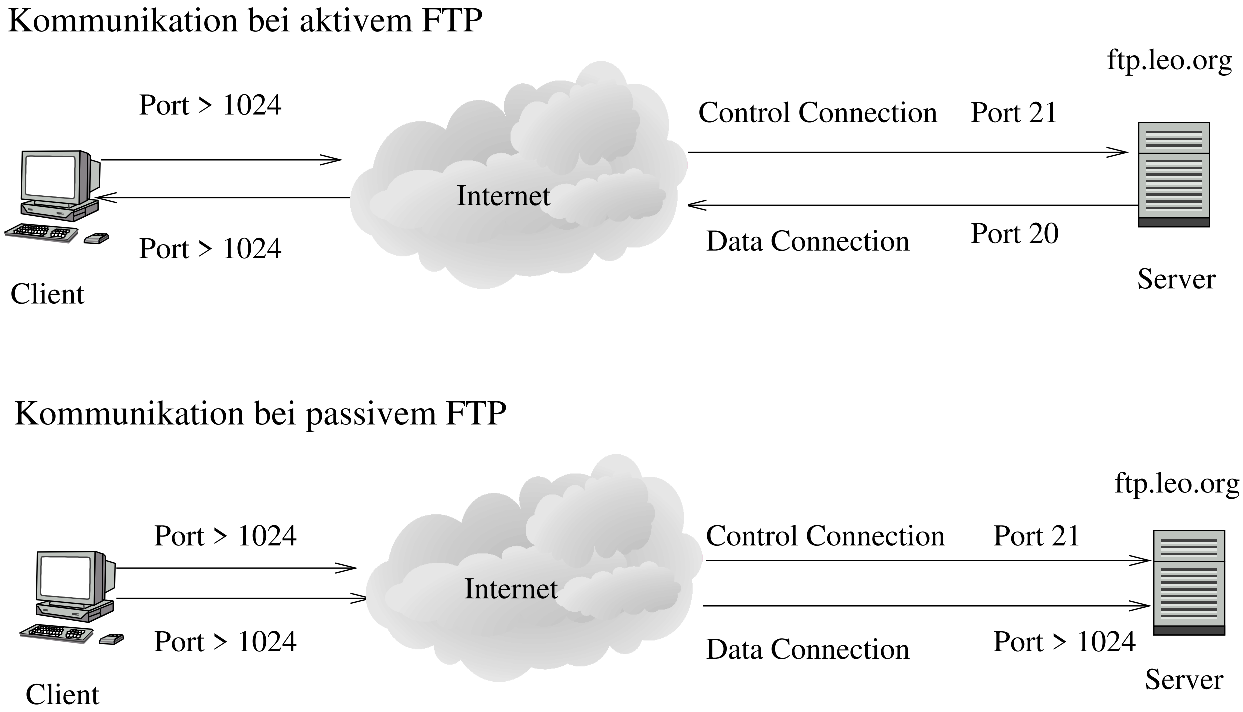

Abbildung 61: Prinzipielle Funktionsweise FTP

öffentlichen Dateibereich tummeln.
Die Besonderheit des Protokolls liegt in den getrennten Kanälen für die Daten und die Steuerung. Im RFC [PoRe 85] ist für FTP TCP-Port 21 als Steuerungskanal und TCP-Port 20 als Datenkanal festgelegt. FTP verwendet als Transportprotokoll immer TCP, da dieses bereits einen sicheren Datentransfer garantiert und die FTP-Software sich nicht darum zu kümmern braucht. Standardmäßig wird **activ FTP** verwendet. D.h. der Client baut von einem Port größer 1024 die Steuerverbindung auf Port 21 des Servers auf. Über diesen Kanal wird nun ein Port ausgehandelt, auf den der Server sich von Sourceport 20 aus auf den Client verbindet, um Daten zu transferieren. Somit wird eine zweite Verbindung vom Server zum Client aufgebaut. Beim **passiv FTP** erfolgt der Aufbau der Datenverbindung vom Client aus. Vom Server aus wird in diesem Fall keine Rückverbindung zum Client aufgebaut. FTP besitzt eine eigene Kommandooberfläche, die interaktiv bedient wird. Der Aufruf dieses Filetransferprogrammes erfolgt durch das Kommando `ftp`.
Die Vorteile von FTP liegen in den effizienten Verfahren zur Übertragung von Dateien beliebigen Formats und der Tatsache, daß der Zugriff seitens beliebiger Internet-Teilnehmer möglich ist.
Auch hier gilt, wie schon bei Telnet, daß die Verbindung unverschlüsselt über das Netz geht.

6.3.2 Die Software „ProFTPD“

ProFTPD [pro 03] ist aus dem Wunsch heraus entstanden, einen sicheren und einfach konfigurierbaren FTP Server zu haben. Als Vorbild für die Konfiguration wurde der Apache Webserver (wird in Abschnitt 7.2.4 behandelt) verwendet. Es gibt bereits eine Menge FTP Server für Unix Maschinen, einer der bekanntesten ist wahrscheinlich der WuFTPD [wuf 02], der sehr performant ist. Leider hat er aber eine umfangreiche, nicht immer glanzvolle Sicherheitshistory und läßt auch einige Konfigurationsmöglichkeiten vermissen, die zur Zeit für Windows FTP Server zur Verfügung gestellt werden.
ProFTPD hat ein Konfigurationsfile `/etc/proftpd.conf`, das alle Einstellungsmöglichkeiten beinhaltet. Es können verschiedene Berechtigungen für unterschiedliche Verzeichnisse mit je einem eigenen Authentisierungsfile definiert werden. Die FTP User müssen nicht in der `/etc/passwd` einen Systemaccount besitzen. Somit kann die Administration der Anwender auch von einem anderen Benutzer als `root` aufgeführt werden. Wird der ProFTPD als eigener Prozeß gestartet und nicht über die `/etc/inetd.conf`, so kann er unter einem User laufen, der nicht alle Berechtigungen von `root` braucht. Dadurch kann das Erlangen von `root`-Rechten durch Angriffe durch Exploits erschwert werden, da nach einem erfolgten Bufferoverflow der Angreifer nur die Rechte des nicht privilegierten Users bekommt, unter dem der FTP Server gelaufen ist. Als besonderes Sicherheitshighlight ist zu betrachten, daß über den ProFTPD keine Programme gestartet und ausgeführt werden können. Ebenso ist das Konzept versteckter Verzeichnisse und Files dahingehend umgesetzt, dass jeder Anwender darin beschränkt werden kann, wieviel er vom Verzeichnisbaum des Systems zu Gesicht bekommt. Das kann dann sehr interessant sein, wenn eine Firma mit mehreren anderen Firmen, die vielleicht Konkurrenten sind, Files über einen FTP Server austauschen möchten, so muß sichergestellt sein, daß jede Firma nur die für sie bestimmten Daten sehen, kopieren, verändern oder herunterladen kann.
In dem nun zu betrachtenden Beispiel gibt es ein Authentisierungsfile `/etc/proftpd/ftpd.passwd`, das folgende Einträge beinhaltet:

```
user1:Jib.R9cuz5.Vs:1006:1006:user1:/home/ftp/user1:/bin/false
user2:IhoWaBgFl9eVo:1006:1006:user2:/home/ftp/user2:/bin/false
user3:/Bi7nM9daRa8g:1006:1006:user3:/home/ftp/user3:/bin/false
```

Die User-ID 1006 ist auch in der `/etc/passwd` definiert. Mit diesem User kann der FTP Server gestartet werden.
Die User können mit dem Script `ftpasswd` administriert werden, das beim ProFTPD mitgeliefert wird.

```
ftpasswd --passwd --file=/etc/proftpd/ftpd.passwd --gid=1006 --uid=1006
          --home=/home/ftp/testuser --name=testuser --shell=/bin/false
```

Dieser Aufruf erzeugt in der `/etc/proftpd/ftpd.passwd` folgenden Eintrag und legt auch gleich ein Passwort fest:

```
testuser:KQsrKhQVuUz8I:1006:1006::/home/ftp/test:/bin/false
```

Mit dem Aufruf `ftpasswd --help` wird angezeigt, welche Optionen und Parameter zur Verfügung stehen.
In dem dazugehörigen Groupfile `/etc/proftpd/ftpd.group` ist die GroupID 1006 definiert, die man auch in der `/etc/group` wiederfindet.

```
ftp:x:1006:
```

Im Konfigurationsfile `/etc/proftpd.conf` werden alle Einstellungen für den FTP Daemon festgelegt [pro 01]. Der Tag `ServerName` legt fest, mit welchem Namen sich der FTP Server nach der Anmeldung des Benutzers meldet. Der String, der nach erfolgtem TCP-Verbindungsaufbau angezeigt wird, wird mit `ServerIdent` festgelegt. `ServerAdmin` legt die Mailadresse des Serveradministrators fest. Mit `ServerType` wird der Modus festgelegt, in dem Der ProFTPD läuft. Im Modus `standalone` wird der Server durch ein eigenes Startscript gestartet, gibt man als Modus `inetd` an, so erwarter der Server, vom `inetd` gestartet zu werden. Möchte man die Willkommensmeldung solange unterdrücken, bis sich ein Anwender erfolgreich angemeldet hat, so sollte man `DeferWelcome` auf `on` setzen. Mit `DefaultRoot` wird festgelegt, welches Rootverzeichnis ein Anwender bekommen soll. Ist `~` angegeben, so wird das für den User in der `/etc/proftpd/passwd` definierte Verzeichnis als sein Rootverzeichnis definiert, über das er nicht hinauskommt. Mit `ShowSymlinks on` werden symbolische Links für den Anwedner sichtbar. Baut ein Client eine Verbindung zu einem ProFTPD Server auf, so versucht der Server über das `ident` Protokoll[37] den externen Benutzernamen herauszubekommen. Läuft dieser Versuch in einen Timeout, da evtl. der Zugriff durch einen Firewall gesperrt ist, verzögert sich der Anmeldungsprompt. Um dem entgegenzuwirken, können die Ident Lookups mit `IdentLookups off` deaktiviert werden.

Hier eine mögliche Konfiguration des ProFTPD in `/etc/proftpd.conf`:

```
ServerName                      "secp-ftp.my-domain.de"
ServerIdent                     on  "FTP Server for SECP ready."
ServerAdmin                     "admin@my-domain.de"
ServerType                      standalone
DeferWelcome                    off
TransferLog                     /var/log/xferlog

DefaultRoot                     ~
ShowSymlinks                    on
IdentLookups                    off
UseReverseDNS                   off
```

[37] RFC 1413 [John 93]

```
MultilineRFC2228                on
DefaultServer                   on
ShowSymlinks                    on
AllowOverwrite                  on

TimeoutNoTransfer               7200
TimeoutStalled                  600
TimeoutIdle                     7200

DisplayLogin                    welcome.msg
DisplayFirstChdir               .message
LsDefaultOptions                "-l"

DenyFilter                      \*.*/

# Authentifizierung unabhaengig vom System
AuthPAM                         off
AuthUserFile                    /etc/proftpd/ftpd.passwd
AuthGroupFile                   /etc/proftpd/ftpd.group

# make listed files appear to be owned by the logged-in user
DirFakeUser on ~
DirFakeGroup on ~
DirFakeMode 0600

# Uncomment this if you are using NIS or LDAP to retrieve passwords:
PersistentPasswd                off

# Port 21 is the standard FTP port.
Port                            21

# To prevent DoS attacks, set the maximum number of child processes
# to 30.  If you need to allow more than 30 concurrent connections
# at once, simply increase this value.  Note that this ONLY works
# in standalone mode, in inetd mode you should use an inetd server
# that allows you to limit maximum number of processes per service
# (such as xinetd)
MaxInstances                    30

# Set the user and group that the server normally runs at.
User                            nobody
Group                           nogroup
```

```
# Normally, we want files to be overwriteable.
<Directory /*>
# Umask 022 is a good standard umask to prevent new files and dirs
# (second parm) from being group and world writable.
Umask                         022  022

AllowOverwrite                on

</Directory>
```

6.4 Secure Shell

Secure Shell (`ssh`) und Secure Copy (`scp`) bieten die Möglichkeit, die Internetdienste `telnet`, `rexec`, `rlogin`, `rsh`, `rcp` und `ftp` zu ersetzen. Bei diesen Diensten werden Daten und insbesondere Passwörter unverschlüsselt über die Netze geschickt, während bei SSH die gesamte Kommunikation verschlüsselt erfolgt. Zusätzlich verschlüsselt SSH X11-Verbindungen, setzt dabei automatisch die Display-Variable und gestattet, beliebige TCP/IP-Verbindungen zu tunneln.
Eigenschaften:

- Gedacht als Ersatz für die Unix-R-Kommandos (`rlogin`, `rsh`, `rcp`)
- Verschlüsselung der Verbindung
- Auch einsetzbar zur Sicherung von X-Window, `telnet`, `ftp` und `POP` durch Umleitung der Verbindung über einen gesicherten Kanal
- Automatische Umsetzung der Display-Variablen unter X
- Starke Host-Authentifizierung (public-private Key Verfahren) (siehe Kapitel 5)
- Unterstützte Plattformen:
  - nahezu alle Unix-Systeme
  - Windows 3.x, 95/98, NT, 2000 und XP
  - Macintosh

Mit der SSH besteht Schutz vor:

- Abhören von Passwörtern und Terminalverbindungen durch Sniffer
- DNS-Spoofing (siehe Seite 161)
- IP-Spoofing (Vortäuschen einer anderen IP-Adresse, um z.B. über Sicherungssysteme hinweg Angriffe auf interne Server auszuführen, siehe Seite 89)

6.4.1 Funktionsweise von SSH

Es wird vorausgesetzt, daß auf dem Zielrechner der SSH-Server und auf der Client-Workstation (von der aus man eine Verbindung zum Zielrechner herstellen will) der SSH-Client installiert ist.
Funktionsweise:

- Client baut Verbindung zum Server auf.
- Infos über die verwendete Protokollversion werden ausgetauscht.
- Beide Verbindungspartner schalten auf eine per CRC gesicherte Binärverbindung um.
- Der Server sendet seine öffentlichen Host- und Server-Schlüssel. Der Hostschlüssel dienen der Identifikation des Rechners und ist 'langlebig', Serverschlüssel werden in regelmäßigen Abständen neu generiert (z.B. einmal pro Stunde).
- Der Client überprüft den Host-Schlüssel mit Hilfe einer lokalen Datenbank (sofern schon gespeichert).
- Der Client authentisiert sich mit seinem Host-Schlüssel beim Server.
- Der Client generiert einen zufälligen Sitzungsschlüssel und chiffriert ihn mit Hilfe der beiden Schlüssel, die er vom Server erhalten hat.
- Die daraus entstehende Nachricht wird zum Server zurück übertragen.
- Die nachfolgende Übertragung läuft komplett verschlüsselt ab.

An dieser Stelle haben sich die beteiligten Rechner gegenseitig authentisiert und einen verschlüsselten Kanal aufgebaut. Bisher wurden noch keine Benutzerdaten übertragen.
Bei der Validierung der Zugangsdaten von Benutzern gibt es bei SSH im Wesentlichen zwei Möglichkeiten:

- Zugang mit Passwort-Authentisierung.
- Zugang mit RSA-Authentisierung (siehe auch Kapitel 5): SSH bietet zusätzlich die sogenannte RSA-Authentisierung an. Diese Form der Validierung ist noch sicherer. Der Benutzer muß sich ein RSA-Schlüsselpaar generieren. Zur Authentisierung wird der Public-Key und die Signatur einer Nachricht mit dem Private-Key verwendet. Der private RSA-Schlüssel wird, mit einer Passphrase verschlüsselt, gespeichert, um ihn vor Mißbrauch zu schützen. D.h. selbst wenn der RSA-Schlüssel in unerlaubte Hände gerät, kann er nur dann genutzt werden, wenn auch die Passphrase bekannt ist. Wegen der erhöhten Sicherheit wird empfohlen, möglichst diese Art der Validierung zu nutzen.

6.4.2 SSH Tunnel

Es ist möglich, E-Mail, POP3, etc. über eine SSH-Verbindung zu tunneln. Dies ist schematisch in den Abbildungen 62 und 63 dargestellt.
Wir wollen :
Von Rechner `majestix` aus soll einen SSH Tunnel zum Rechner `fw-admin` aufgebaut werden, über den SMTP getunnelt werden soll. Zuerst wird auf dem `majestix`

```
ssh fw-admin -L 50000:fw-admin:25
```

ausgeführt. Dazu muß man sich am `fw-admin` authentisieren. Solange diese SSH Verbindung offen ist, sieht man in der Prozeßliste auf dem `majestix`:

```
ps ax | grep ssh
16852 pts/102  S      0:00 ssh fw-admin -L 50000:fw-admin:25
netstat -an | grep 50000
tcp        0      0 127.0.0.1:50000         0.0.0.0:*               LISTEN
```

Damit hat man auf der lokalen Maschine den Port 50000 geöffnet. Wird dieser Port angesprochen, so erfolgt eine Weiterleitung und Umsetzung auf die Verbindung `fw-admin` Port 25. Auf `localhost:50000` läuft jetzt der SMTP-Dienst von `fw-admin:25` und kann z.B. mit `telnet localhost 50000` angesprochen werden.

Abbildung 62: Unverschlüsselte Verbindung

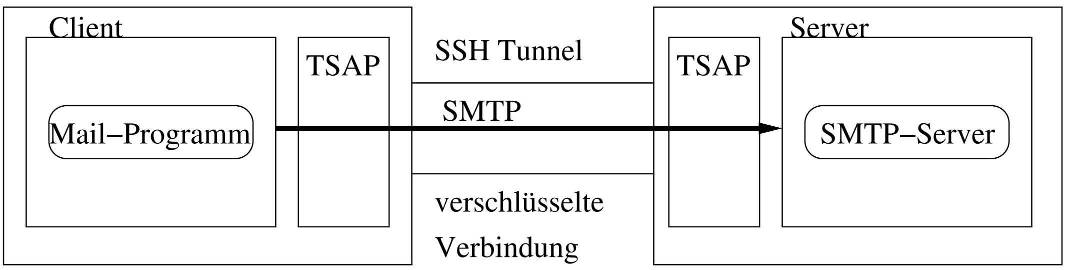

Abbildung 63: Verschlüsselte Verbindung

6.4.3 Kryptographische Verfahren

Bei SSH kommen kryptographische Verfahren an mehreren Stellen zum Einsatz. Ein **Asymmetrisches Verfahren** wird für die Authentifizierung der Kommunikationspartner und für die sichere Übertragung eines zufällig erzeugten und nur für kurze Zeit (in der Regel eine Stunde oder eine Sitzung) verwendeten Sitzungs-Key eingesetzt:

- **RSA** (Akronym der Erfinder Rivest, Shamir und Adleman). Dabei werden Key-Längen zwischen 768 und 1024 Bit empfohlen (siehe Kapitel 5)

Zur Verschlüsselung der Daten während der Verbindung wird wegen des höheren Durchsatzes ein **Symmetrisches Verfahren** verwendet. Je nach Installation der Software hat der Benutzer im allgemeinen die Auswahl zwischen verschiedenen Algorithmen, die Durchsatz-Angaben in folgender Liste beziehen sich auf einen Pentium-Prozessor (diese Angaben stammen von der Webseite [boe 98], auf der Sie auch noch weitere Informationen zur SSH finden können):

- None: 100% Durchsatz. Diese Variante ist nur für Debug- und Vergleichszwecke geeignet, da keine Verschlüsselung durchgeführt wird.
- DES (Data Encryption Standard): 56 Bit Key-Länge; 71% Durchsatz. Für sehr hohe Sicherheits-Anforderungen ist dieses Verfahren wegen seiner geringen Key-Länge inzwischen zu schwach.
- 3DES (Tripple DES): 112 Bit Key-Länge; 45% Durchsatz. Dieser Algorithmus kann empfohlen werden und ist als größter gemeinsamer Nenner bei allen SSH-Installationen immer vorhanden. Außerdem wird 3DES als Default verwendet und optionale, individuelle RSA-Keys sind durch 3DES geschützt.
- IDEA (International Data Encryption Algorithm): 128 Bit Key-Länge; 64% Durchsatz. Auch dieses Verfahren kann empfohlen werden.
- Blowfisch: 32 - 448 Bit Key-Länge möglich, SSH verwendet 128 Bit; 88% Durchsatz. Dem Autor des Dokuments sind keine Aussagen über die Güte dieses Algorithmus bekannt.
- Arcfour: Beliebige Key-Länge möglich, SSH verwendet 128 Bit; 91% Durchsatz. Dem Autor des Dokuments sind keine Aussagen über die Güte dieses Algorithmus bekannt.

Links zu Kryptographie und anderer Themen finden Sie in der Linksammlung unter [boe 02].

6.4.4 Konfiguration von SSH Server und SSH Client

Alle SSH Konfigurationsfiles sind unter `/etc/ssh/` zu finden. Darin befinden sich die Hostkeys (`ssh_host*key.pub` der Public-Key und `ssh_host*key`[38] der jeweilige Private-Key) der angegebenen Verschlüsselungsmethode ist. Die Konfigurationsdatei des SSH Clients ist `ssh_config`. In dieser Datei werden systemweite Einstellungen für den SSH Client getroffen, die aber durch Kommandozeile oder durch die persönlichen SSH Client Einstellungen des Anwenders unter `$HOME/.ssh/` unwirksam werden.

```
# This is ssh client systemwide configuration file.  See ssh(1) for more
# information.  This file provides defaults for users, and the values can
# be changed in per-user configuration files or on the command line.

# Configuration data is parsed as follows:
#  1. command line options
#  2. user-specific file
#  3. system-wide file
# Any configuration value is only changed the first time it is set.
# Thus, host-specific definitions should be at the beginning of the
# configuration file, and defaults at the end.

# Site-wide defaults for various options

# Host *
#   ForwardAgent no
#   ForwardX11 no
#   RhostsAuthentication no
#   RhostsRSAAuthentication yes
#   RSAAuthentication yes
#   PasswordAuthentication yes
#   FallBackToRsh no
#   UseRsh no
#   BatchMode no
#   CheckHostIP yes
#   StrictHostKeyChecking yes
#   IdentityFile ~/.ssh/identity
#   IdentityFile ~/.ssh/id_dsa
#   IdentityFile ~/.ssh/id_rsa
#   Port 22
#   Protocol 2,1
#   Cipher blowfish
```

[38] * steht für _, _rsa_, und _dsa_

```
#   EscapeChar ~
```

- `Protocol 2,1` bedeutet, daß der Client zuerst versucht, SSH Protokollversion 2 mit dem Server zu sprechen, wenn das nicht möglich ist, wird auf Version 1 zurückgegangen.
- `Port 22` gibt an, daß ohne weiter Portangaben der Standardport 22 für SSH vorausgesetzt wird.
- `Cipher blowfish` sagt, daß als Verschlüsselung `blowfish` verwendet wird.
- `ForwardAgent no`: Es erfolgt keine Weiterleitung zum Authenticationagent der entfernten Maschine.
- `ForwardX11 no`: Standardmäßig wird die DISPLAY Variable nicht gesetzt. D.h. es können keine X11 Weiterleitungen initiiert werden.
- `RhostsAuthentication no` sorgt dafür, daß die auf rhosts basierede Authentisierung nicht verwendet wird. Diese Einstellung ist nur auf Clientseite relevant.
- `RhostsRSAAuthentication yes`: Hier wird festgelegt, ob eine rhosts basierende Authentisierung mit RSA erlaubt ist.
- `RSAAuthentication yes` legt fest, daß eine Authentisierung mit RSA möglich ist. Diese Einstellung greift nur dann, wenn auf dem Client das `identity` File existiert.
- `PasswordAuthentication yes` erlaubt Passwortauthentisierung.
- `FallBackToRsh no` regelt, daß auch bei einem fehlgeschlagen Verbindungsaufbau vom Client aus nicht versucht wird, mit `rsh` zu kommunizieren.
- `UseRsh no` verbietet das Verwenden von `rsh`.
- `BatchMode no` legt fest, ob die SSH Kommunikation nicht als Batch läuft.
- `CheckHostIP yes` überprüft, ob der Eintrag in der `known_hosts` für diese IP-Adresse mit dem übermittelten Hostschlüssel übereinstimmt.
- `StrictHostKeyChecking yes` sorgt dafür, daß ein übermittelter Hostschlüssel nur nach Nachfrage beim Anwender in die `known_hosts` eingetragen wird. Stimmt der übertragene Hostschlüssel nicht mit dem Eintrag in der `known_hosts` überein, so wird die Verbindung nicht aufgebaut und der Anwender über den Grund informiert.
- `IdentityFile` legt fest, in welchem Verzeichnis und in welchen Files der private Schlüssel des Anwenders zu finden ist.
- `EscapeChar ~` legt das Escapezeichen fest.

Der SSH Server wird über `sshd_config` konfiguriert. Dort kann die Einschränkung auf Interfaces erfolgen, die Festlegung der Pfade für die Host-Keys erfolgt hier. `X11 Forwarding` kann aktiviert oder deaktiviert werden, der Loglevel wird festgelegt und einiges mehr.

```
Port 22
Protocol 2,1
#ListenAddress 0.0.0.0
#ListenAddress ::
HostKey /etc/ssh/ssh_host_key
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
ServerKeyBits 768
LoginGraceTime 600
KeyRegenerationInterval 3600
PermitRootLogin no
#
# Don't read ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
# Uncomment if you don't trust ~/.ssh/known_hosts for RhostsRSAAuthentication
#IgnoreUserKnownHosts yes
StrictModes yes
X11Forwarding yes
X11DisplayOffset 10
PrintMotd no
PrintLastLog no
KeepAlive yes

# Logging
SyslogFacility DAEMON
LogLevel DEBUG
#obsoletes QuietMode and FascistLogging

RhostsAuthentication no
#
# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
RhostsRSAAuthentication no
# similar for protocol version 2
HostbasedAuthentication no
#
RSAAuthentication yes
```

```
# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
PermitEmptyPasswords no

# Uncomment to disable s/key passwords
ChallengeResponseAuthentication no

# Uncomment to enable PAM keyboard-interactive authentication
# Warning: enabling this may bypass the setting of 'PasswordAuthentication'
#PAMAuthenticationViaKbdInt yes

# To change Kerberos options
# NB: Debian's ssh ships without Kerberos Support
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#AFSTokenPassing no
#KerberosTicketCleanup no

# Kerberos TGT Passing does only work with the AFS kaserver
#KerberosTgtPassing yes

#CheckMail yes
UseLogin no

#MaxStartups 10:30:60
#Banner /etc/issue.net
#ReverseMappingCheck yes

Subsystem       sftp    /usr/lib/sftp-server
```

- `ListenAddress` legt fest, auf welche IP-Adresse und somit auf welches Interface der Dienst gebunden wird.
- `Port 22` legt den Port fest, hinter dem der Dienst liegen soll. Das kann auch beliebig verändert werden.
- `Protcol 2,1` legt fest, daß zuerst versucht wird, mit dem Client SSH Protokollversion 2 zu sprechen. Versteht der Client das nicht, so wird auf Version 1 zurückgegangen.
- `UseLogin no` sorgt dafür, daß `login` beim Anmelden nicht verwendet wird, da in diesem Binary ein Buffer Overflow gefunden wurde.
- `PermitEmptyPasswords no` soll die User dazu erziehen, keine leeren Passwörter zu verwenden.

- `PermitRootLogin no` läßt nicht zu, daß sich Administratoren übers Netz direkt als `root` anmelden.
- `IgnoreRhosts yes` ist dafür verantwortlich, daß von Usern angelegte `.rhosts` und `.shosts` nicht ausgewertet werden.
- `HostKey` legt die Verzeichnisse und Files fest, in denen der Hostschlüssel zu finden ist.
- `ServerKeyBits 768` gibt die Länge des Serverschlüssels an.
- `LoginGraceTime 600`: nach dieser Zeitspanne beendet der Server die Verbindung, wenn sich der Anwender nicht erfolgreich anmelden konnte.
- `KeyRegenerationInterval 3600` gibt die Gültigkeitdauer des Serverschlüssels an.
- `StrictModes yes` überprüft die Rechte und Zugehörigkeiten im Userverzeichnis, bevor ein Login akzeptiert wird.
- `X11Forwarding yes` erlaubt das Weiterleiten von X11 Verbindungen.
- `X11DisplayOffset 10` gibt die erste verfügbare Displaynummer für X11 Forwarding an.
- `PrintMotd no` unterbindet die Ausgaben des Begrüssungstextes aus `/etc/motd`.
- `PrintLastLog no` unterbindet die Ausgabe der Information, wann sich dieser User das letzte Mal erfolgreich angemeldet hat und von welchem Rechner er gekommen ist.
- `KeepAlive yes` sorgt dafür, daß das System Keep Alive Meldungen rausschickt.
- `RhostsAuthentication no`: siehe Clientkonfiguration.
- `RhostsRSAAuthentication no`: siehe Clientkonfiguration.
- `HostbasedAuthentication no` ist analog zu `RhostsRSAAuthentication`.
- `RSAAuthentication yes`: siehe Clientkonfiguration.
- `PasswordAuthentication yes`: siehe Clientkonfiguration.
- `ChallengeResponseAuthentication no` verbietet die Verwendung von Einmalpasswortverfahren. Bei einigen SSH Serverversionen gibt es ein Problem mit dem S/KEY Verfahren. somit sollte zur Zeit die Unterstützung von Einmalpasswortverfahren deaktiviert sein.

6.5 Praktische Aufgaben

6.5.1 Topologie

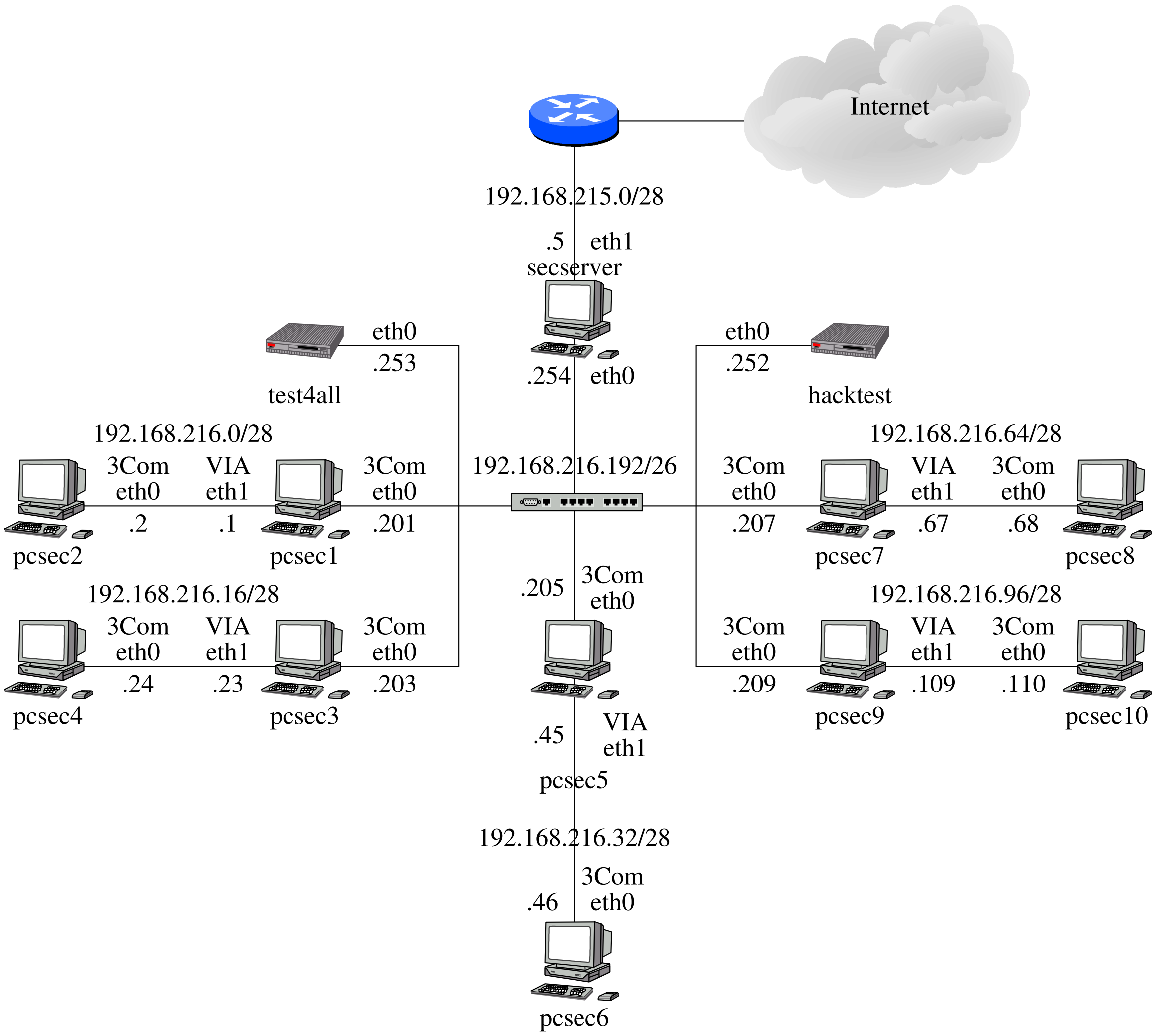

Abbildung 64: Der Versuchsaufbau für die weiteren Versuche des Praktikums

6.5.2 Konfiguration Bind

1. Installieren Sie über `YaST2` das Softwarepacket BIND 8 vom Installserver. Nach der Installation finden Sie das Nameserverbinary unter `/usr/sbin/named`, die Konfigurationsfiles unter `/var/named/`. Die Dienste werden bei Suse über den `YaST2` konfiguriert, um beim Booten gestartet zu werden. Dabei werden Links von den Runlevelverzeichnissen

- `/etc/init.d/rc0.d`
- `/etc/init.d/rc1.d`
- `/etc/init.d/rc2.d`
- `/etc/init.d/rc3.d`
- `/etc/init.d/rc4.d`
- `/etc/init.d/rc5.d`
- `/etc/init.d/rc6.d`
- `/etc/init.d/rcS.d`

die im Startscript `/etc/init.d/named` angegeben sind, auf das zu startende Startfile gelegt.

2. Was sehen Sie mit `netstat -an` und wie ist das zu interpretieren (Informationen über Manpage und / oder Internet)?

3. Ihr Nameserver soll folgende Eigenschaften erfüllen:

   (a) Der Daemon soll auf das Produktivinterface `eth0` hören. Nach einem Neustart des Dienstes durch das Startscript überprüfen Sie mit `netstat -an` und `telnet IP-Adresse 53` und anhand des Logfiles, ob Ihre Konfiguration erfolgreich war.

   (b) Für die Domain `secp.nm.informatik.uni-muenchen.de` ist der `pcsec4` als Master zuständig, für die Reverse-Zonen unseres Netzbereichs `192.168.216.0/24` ist der `pcsec10` als Master zu konfigurieren. Alle anderen sind Slave für diese Zonen. Überrüfen Sie Ihre Konfiguration anhand der Logfiles und mittels Zonentransfers: `host -l -a secp.nm.informatik.uni-muenchen.de ip-adresse-nameserver`. Namensgebung: z.B. `pcsec1` (`192.168.216.201`), `pcsec1-eth1` (`192.168.216.1`) und so weiter.

   (c) Für `uni-muenchen.de` verweisen Sie auf `test4all`. Somit ist der `test4all` für diese Domain als Forwarder einzutragen. Überprüfen Sie Ihre Konfiguration wieder anhand er Logfiles und mit den bereits bekannten Tools `host`, `dig` und / oder `nslookup`.

   (d) Zonentransfer soll nur von der IP-Adresse des `secservers` (`192.168.216.254`) und Ihres Partnerrechner erlaubt sein. Was hat das für Konsequenzen bzgl. der Masterzonen? Was ist hier zu beachten (siehe Seite 162)?

   (e) DNS Queries sollen nur von der eigenen Maschine und dem `secserver` erlaubt sein. Überprüfung mittels `nslookup` Abfragen. Welche Konsequenzen hat das für die Zonen, für die man Master oder Slave ist? Was ist hier zu beachten und an der Konfiguration zu ändern? (Siehe auch Seite 162 und in den im Literaturverzeichnis angegebenen Quellen).

   (f) Für alle anderen Anfragen ist der `secserver` als Forwarder einzutragen. D.h. es ist so zu verfahren, wie auf Seite 158 beschrieben ist.

4. Tragen Sie Ihren Nameserver in die `/etc/resolv.conf` ein. Welche Auswirkungen hat das?

5. Überprüfen Sie im Logfile, ob Ihr Nameserver korrekt starten konnte und dokumentieren Sie das in Ihrer Ausarbeitung.

6. Überprüfen Sie die Funktionalität Ihres Nameservers mit `host` und `nslookup` und dokumentieren Sie das in Ihrer Ausarbeitung.

Weitere Informationen zu den Befehlen `host`, `dig` und `nslookup` finden Sie in den Manpages.

6.5.3 Übungen zu Telnet und SSH

1. Stoppen Sie alle verfügbaren Dienste und testen Sie das Ergebnis mit `nmap`. Was sehen Sie? Aktivieren Sie zur Zeit nur Ihren Nameserver. Was sehen Sie mit `nmap`?

2. Starten Sie Telnet so, daß er vom `tcpwrapper` und `inetd` kontrolliert wird und sich User nur von der IP-Adresse Ihres Partnerrechners und vom `secserver` einloggen können.

3. Installieren Sie den SSH Daemon und konfigurieren Sie ihn so, daß er nur auf Ihrem Interface `eth0` hört und nur von Ihrem Partnerrechner und dem `secserver` aus angesprochen werden kann. Wie können Sie die Richtigkeit Ihrer Konfiguration überprüfen? Was sehen Sie, wenn Sie mit `namp` alle Ihre Interface anscannen? Was sehen Sie mit `netstat -an` und wie ist das zu interpretieren?

4. Verfolgen Sie mittels Programmen zum Netzsniffern sowohl eine Telnet, wie auch eine SSH Verbindung. Wenn Sie eine SSH Sitzung mittels `tcpdump` mitsniffern, was ist der Unterschied zu Telnet?

Informationen zu `nmap`, `tcpd`, `inetd` und so weiter finden Sie in den Manpages.

7 Weitere Dienste in TCP/IP Netzwerken

7.1 Electronic Mail

Das Verschicken von E-Mails ist einer der ältesten Dienste im Internet. Den ersten Teil der E-Mail nennt man **Header** (**Kopfzeilen**), den Inhalt und die Anlagen **Body Parts**. Der Umschlag heißt auf englisch **Envelope** (dieser hat hauptsächlich etwas mit dem Übertragungsprotokoll zu tun, der Normalanwender bekommt ihn nicht zu Gesicht). D. h. eine Mail, die an das Mail-System übergeben wird, besteht aus dem Envelope, dem Header und einem oder mehreren Body Parts. Die Postboten und Postämter nennt man **Message Transfer Agents** (**MTA**), die zusammen das **Message Transfer System** (**MTS**) bilden. Dieses MTS sorgt dafür, daß eine Mail vom Quellrechner zum Zeilrechner gelangt. Beim Ziel-MTA angelangt, wird die Mail in die Eingangs-Mailbox des Empfängers gelegt. Der Empfänger kann sich dann mit Hilfe eines Mail-Programms seine Mail aus dem Postfach in seine Eingangs-Mailbox holen und lesen. Wenn er sie danach nicht löscht, wird er sie in eine Ablage kopieren, die man **Folder** nennt. Die Benutzeroberfäche zum Erstellen einer Mail, die Eingangs- und Ausgangs-Mail-Boxen, die Folder und eine eindeutige Mailadresse zusammen nennt man den **Mail User Agent** (**MUA**) oder **Mailer**.
Je nach verwendetem Mail-System gibt es meist noch weitere Funktionen, z.B. das Weiterleiten von Nachrichten (ggf. mit Kommentar), Versenden von Nachrichten an mehrere Empfänger, Benachrichtigung des Versenders einer Nachricht, daß die Mail beim Empfänger angekommen ist.
Der **postmaster** ist die Mail-Adresse, bei der alle Fehlermeldungen, aber auch Anfragen von außen, anlaufen. Dahinter versteckt sich normalerweise der Systemverwalter.
Rechner mit direkter TCP/IP-Verbindung tauschen ihre E-Mails direkt aus. Das Protokoll heißt **SMTP** (**Simple Mail Transfer Protocol**). Hier wird die E-Mail dem Zielrechner direkt zugestellt.
Der erste Mail-RFC 822[39] [Croc 82] legt in erster Linie den Standard für Kopfzeilen (Header) in der elektronischen Post fest. Dort wurde unterstellt, beim Inhalt des Briefes handele es sich um reinen ASCII-Text. Wer Dateien versenden wollte, die Zeichen enthielten, welche nicht unter den 128 Zeichen des ASCII-Alphabets vorkamen, mußte die Datei so codieren, daß sie nur noch aus ASCII-Zeichen bestand.
MIME (**Multipurpose Internet Mail Extensions**) fügt diesem Standard vier weitere Felder hinzu, die genauer den Inhalt des Briefes spezifizieren. Aus diesen Feldern kann das Mailprogramm, so es diese berücksichtigt, entnehmen, welche anderen Programme aufzurufen sind, um z.B. ein Bild darzustellen. Das heißt nicht, daß die Daten im Brief nicht codiert würden, aber ein MIME-konformes Mailprogramm bietet die Möglichkeit, alle Codierungsvorgänge zu automatisieren.
Das erste Feld, welches der MIME-Standard definiert, heißt **MIME-Version**. Bislang gibt es nur die Version 1.0, so daß der Eintrag 1.0 dem Standard genügt. Mit der Verwendung

[39] geht in neuerer RFC 2822 [ReEd 01] auf