Sicherheit im Internet: Stand der Technik

Harald Weidner
weidner@ifi.unizh.ch

Universität Zürich
Institut für Informatik
März 1997

Inhalt

INHALT 2

EINLEITUNG 3

Das SINUS-Projekt 3

Ziel dieses Berichtes 4

Aufbau 4

1 KLASSIFIKATION VON SICHERHEITSPROBLEMEN 6

2 HILFSMITTEL AUS DER KRYPTOGRAPHIE 8

2.1 Klassische Verschlüsselungssysteme 8

2.2 Die Caesarchiffre 9

2.3 Klassifizierung von Chiffren 10

2.4 Der Data Encryption Standard 11

2.5 Der International Data Encryption Algorithmus 12

2.6 Public Key Kryptographie 13

2.7 RSA 16

2.8 Das El-Gamal Signaturverfahren **17**

2.9 Kryptographische Hashfunktionen **18**

2.10 Der Message Digest 5 **18**

2.11 PGP **19**

2.12 Kryptographische Protokolle **20**

2.12 Aktuelle Forschungsthemen **21**

3. FUNKTIONSWEISE UND SICHERHEITSEIGENSCHAFTEN VON TCP/IP-NETZEN **23**

3.1 Uebersicht **23**

3.2 Aufgabe und Arbeitsweise der Protokollebenen **25**

3.3 Die Netzzugangsschicht **27**

3.4 Das Internet Protocol (IP) **27**

3.5 Das Internet Control Message Protocol (ICMP) **28**

3.6 Das Transmission Control Protocol (TCP) **29**

3.7 Das User Datagram Protocol (UDP) **32**

3.8 Adressierung in TCP/IP-Netzen **32**

3.9 Routing in TCP/IP-Netzen **35**

3.10 Subnetze und Subnetz-Masken **37**

3.11 Zukünftige Entwicklungen **38**

3.12 Sicherheitsaspekte der Schichten I bis III **39**

Vertraulichkeit der transportierten Daten 39

Integrität der Daten 40

Authentizität 41

Verlust der Verfügbarkeit von Rechnern 43

4 WICHTIGE PROTOKOLLE DER ANWENDUNGSEBENE UND IHRE SICHERHEITSEINGENSCHAFTEN 45

4.1 Telnet 45

4.2 FTP 47

4.3 Electronic Mail 47

4.4 Das World Wide Web 50

4.5 Die Berkeley r-Tools 53

4.6 DNS 54

4.7 X11 56

4.8 SSH 58

4.9 Zukünftige Forschungsarbeiten 59

5 SICHERHEITSPROBLEME IN MEHRBENUTZERSYSTEMEN 61

5.1 Login und Passwörter 62

5.2 Dateien und Zugriffsrechte 63

5.3 Prozesse 64

5.4 Gemeinsame Dateisysteme im Netzwerk 66

5.5 Gemeinsame Nutzung von Druckern 67

6 FIREWALLS 68

6.1 Einleitung 68

6.2 Paketfilter 70

6.3 Application Gateways 75

6.4 Konzeption von Firewall-Lösungen 78

7 ZUSAMMENFASSUNG UND AUSBLICK 81

LITERATUR 83

Einleitung

Das SINUS-Projekt

Die Nutzung von Online-Diensten des Internet wird von immer mehr Unternehmen als ein wichtiges zukünftiges Werbe- und Geschäftsinstrument angesehen. Sowohl die Präsenz in einem schnell wachsenden Markt als auch die Möglichkeit, von den Informationen und Dienstleistungen anderer Anbieter zu profitieren, versprechen ein wichtiges Standbein des Unternehmenserfolges zu werden.

Mit der zukünftigen Kommerzialisierung des Internet, das ursprünglich als Kommunikationsinstrument zu akademischen Zwecken entwickelt wurde, spielen dabei Sicherheitsaspekte eine immer grössere Rolle. Obwohl die Notwendigkeit von Sicherheitsmassnahmen unumstritten ist und viele technische Lösungen zu Sicherheitsproblemen existieren, werden diese in der Realität kaum umgesetzt. Dies hängt nicht nur mit mangelndem Gefahrenbewusstsein und der Tatsache zusammen, dass Sicherheit Geld kostet und keinen direkten finanziellen Gewinn erbringt. Auch die Schwierigkeiten beim Erkennen und Bekämpfen von Sicherheitslücken, die auf mangelnde Kenntnis der Materie sowie Angst vor Inkompatibilität zu bestehenden Lösungen und Verlust an Funktionalität und Bequemlichkeit zurückzuführen sind, tragen dazu bei.

Das Projekt SINUS (Sichere Nutzung von Online-Diensten) wird vom Institut für Informatik der Universität Zürich in Zusammenarbeit mit SWITCH (Swiss Academic and Research Network) und der Telekurs Logistik AG Zürich durchgeführt [Teufel 95]. Es beschäftigt sich mit den Sicherheitsaspekten beim Anschluss eines Unternehmens an das Internet auf organisatorischer, technischer und personeller Ebene. Im Einzelnen wird

untersucht, inwieweit Nutzung von Online-Diensten, Angebot von Informationen und Durchführung von Geschäftsbeziehungen über das Internet in das Sicherheitskonzept und die informationstechnische Infrastruktur eines Unternehmens eingebettet werden kann. Ferner soll eine Aussage darüber gewonnen werden, ob frei verfügbare Sicherheitsoftware mit offener Spezifikation dem Anspruch gerecht wird, zur Abwehr von Angriffen aus dem Internet geeignet zu sein.

Ziel dieses Berichtes

Dieses Dokument behandelt Sicherheitsaspekte, die beim Anschluss an das Internet und der Nutzung von Online-Diensten relevant sind. Es beschreibt bekannte Sicherheitsprobleme, stellt Lösungen vor, die mit derzeit bekannten Techniken realisierbar sind, und vergleicht das Machbare mit dem tatsächlich Eingesetzten. Ziel ist es, eine Einführung in die Sicherheitsproblematik beim Anschluss an das Internet zu geben und das Bewusstsein für Sicherheits- und Datenschutzaspekte zu schärfen.

Aufbau

Der Text ist wie folgt aufgebaut: Im ersten Kapitel werden Sicherheitsanforderungen und -probleme klassifiziert und grundlegende Begriffsdefinitionen angegeben. Das zweite Kapitel stellt grundlegende Methoden aus der Kryptographie vor, die zur Umsetzung von Sicherheitsanforderungen benutzt werden können. Das dritte Kapitel beschäftigt sich mit Funktionsweise und Sicherheitseigenschaften der TCP/IP-Protokolle, auf denen das Internet und seine Dienste basieren. Das vierte Kapitel beschreibt gängige Netzwerkdienste, die auf TCP/IP basieren, geht auf ihre Sicherheitseigenschaften ein und zeigt Lösungen auf. Im fünften Kapitel wird auf Datenschutz- und Sicherheitsprobleme in Mehrbenutzersystemen eingegangen. Das sechste Kapitel stellt ein Werkzeug zur Kopplung von Netzen unterschiedlicher Vertrauenswürdigkeit vor, den Firewall. Im siebten Kapitel wird schliesslich das Wesentliche noch einmal zusammengefasst.

Einige Details in den Kapiteln 3 bis 6 hängen stark vom verwendeten Betriebssystem ab. Dieser Bericht konzentriert sich aus mehreren Gründen im Wesentlichen auf **Unix**.

- TCP/IP ist zwar ein systemunabhängiges Protokoll, besitzt jedoch aus historischen Gründen eine enge Verwandtschaft mit Unix. So war **BSD**, der Unix-Dialekt der

Universität von Berkeley, Kalifornien, die erste Plattform, auf der TCP/IP implementiert und eingesetzt wurde. Viele andere Implementierungen sind stark an BSD angelehnt.

- Unix kommt aus dem akademischen Bereich. Von den meisten Unix-Derivaten ist der Quellcode erhältlich, von manchen sogar kostenlos (Linux und einige BSD-Derivate). Dies macht das System durchschaubar und für das Studium neuer Konzepte, Techniken und Protokollimplementierungen geeignet.

- Viele der gängigen Betriebssysteme sind mangels Funktionalität nicht oder nur eingeschränkt für den Netzwerkbetrieb geeignet. MS DOS, Windows, OS/2 oder MacOS sind aufgrund ihres spartanischen Prozess- und Rechtekonzeptes zwar für Client-, aber kaum für Serveraufgaben zu gebrauchen. Andere Systeme sind veraltet (MVS), haben aufgrund ihrer Bindung an spezielle Hardware geringe Marktbedeutung (VMS) oder sind proprietäre Systeme, bei denen Implementierungsdetails geheimgehalten werden und allenfalls der Hersteller selbst eine Aussage über die Eignung in sicherheiskritischen Bereichen machen kann (Windows NT, AS 400).

- Die Marktbedeutung von Unix übertrifft die anderer Systeme bei weitem, und daran wird sich auf absehbare Zeit nichts ändern [Lubich 96].

1 Klassifikation von Sicherheitsproblemen

Um vorhandene Kommunikations- oder Datenverarbeitungssysteme auf ihre Sicherheitseigenschaften untersuchen zu können, muss zunächst definiert werden, was unter Sicherheit zu verstehen ist. Die Literatur orientiert sich dabei zumeist an Grundbedrohungen, vor denen Daten geschützt werden müssen. Die wichtigsten Schutzbedürfnisse sind [z.B. Stallings 95]:

- **Vertraulichkeit** bedeutet, dass Daten nur von befugten Personen gelesen werden dürfen.
- **Integrität** heisst, dass Daten nicht von unbefugten Personen, technischen Einrichtungen oder sonstigen Instanzen verändert werden dürfen.
- **Verfügbarkeit** besagt, dass Daten, Geräte oder andere Arbeitsmittel ohne Unterbrechung genutzt werden können.

Zur Beurteilung von Massnahmen zur Sicherstellung dieser Anforderungen muss das Klassifikationsschema verfeinert werden. Beispielsweise erfordert der Schutz von absichtlichen Verletzungen der Datenintegrität wesentlich andere Massnahmen als der Schutz vor unabsichtlichen Integritätsverletzungen. Ausserdem ist die Abgrenzung zwischen den oben genannten Begriffen fliessend. So kann ein Sabotageakt, der zur Löschung einzelner Dateien auf einem Datenträger führt, als Verletzung der Verfügbarkeit (dieser einzelnen Dateien) oder der Integrität (des gesamten Datenbestandes) angesehen werden.

Manche Autoren nennen weitere Anforderungen:

- **Authentifikation** [Bauknecht 95] bedeutet Echtheit des Autors eines Datensatzes oder einer Nachricht. Der Authentifikationsprozess beschäftigt sich mit der Identifizierung einer Instanz, z.B. durch Abbilden elektronischer Merkmale (Absenderadresse, Signatur) auf nichtelektronische Merkmale (Personalausweis, Bankkonto, Unterschrift).

  In manchen Büchern wird zwischen Authentifikation und Authentisierung unterschieden. Im ersten Fall identifiziert sich eine Instanz gegenüber einer anderen, entfernten Instanz; im zweiten Fall überprüft die Bezugsinstanz die Identität der anderen.

- **Verbindlichkeit** [Kersten 91] heisst, dass die Beteiligten einer Kommunikationsbeziehung sich auf die dabei getroffenen Vereinbarung berufen können.

- **Anerkennung** [Bauknecht 95] bedeutet, dass weder Sender noch Empfänger seine Teilnahme an der Kommunikationsbeziehung leugnen kann.

  In [Kersten 91] werden die beiden letzten Begriffe synonym verwendet. Andere Autoren beziehen die Verbindlichkeit auf die Inhalte der Kommunikation, Anerkennung auf die Kommunikation selbst.

- **Zweckbindung** [Holbein 96] besagt, dass ein Betriebsmittel oder eine Information nur zu einem bestimmten Zweck verwendet werden darf. Im Gegensatz zur Vertraulichkeit ist also nicht die Person/Instanz entscheidend, sondern der Verwendungszweck.

Andere Autoren ordnen die Authentifikation unter Vertraulichkeit oder Integrität und die Anerkennung unter die Integrität ein. Auch hier sind die Grenzen fliessend.

2 Hilfsmittel aus der Kryptographie

Um eine Zugriffskontrolle für Rechner, Dienste oder Daten zu realisieren, die bestimmten Benutzern mehr Privilegien als anderen einräumen, sind kryptographische Methoden unerlässlich. Das folgende Kapitel soll einen Ueberblick über gängige Techniken der Kryptographie geben, die Grenzen aufzeigen und den aktuellen Stand der Forschung in diesem Gebiet umreissen.

2.1 Klassische Verschlüsselungssysteme

Eine (klassische) Verschlüsselungsfunktion oder Chiffre ist abstrakt gesprochen eine Funktion $f_k: M \rightarrow C$. Dabei ist M die Menge aller Klartexte, C die Menge aller Chiffrate, und k ein Element aus K, der Menge aller Schlüssel. Die Entschlüsselung wird mit einer Funktion $f_k^{-1}: C \rightarrow M$ durchgeführt. An eine Verschlüsselungsfunktion werden folgende Anforderungen gestellt:

- Bei Kenntnis des Schlüssels k soll das Chiffrat $c = f_k(m)$ eines Klartextes m effizient berechenbar sein.
- Bei Kenntnis des Schlüssels k soll der Klartext $m = f_k^{-1}(c)$ aus einem Chiffrat c zurückgewonnen werden können.

Der Absender und der Empfänger einer Nachricht müssen über einen gemeinsamen Schlüssel k verfügen, den sie vorher über einen sicheren Kanal (am besten persönlich) ausgetauscht haben.

Ein Angreifer, der den Schlüssel k nicht kennt, soll keine Möglichkeit haben, aus der Kenntnis des Chiffrates c auf den Klartext m oder den Schlüssel k zu schliessen. Genauer:

- **Known Cipertext Attack.** Bei Kenntnis des Chiffrates c soll es keinen Weg geben, den Klartext m zu berechnen ohne den Schlüssel k zu kennen, d. h. die Funktion $m = f_k^{-1}(c)$ soll bei unbekanntem k nicht berechenbar sein.

- **Known Plaintext Attack.** Bei Kenntnis des Chiffrates c und des Klartextes m soll es nicht möglich sein, den Schlüssel k zu berechnen, d.h. es soll keine berechenbare Funktion $k = g(m, c)$ geben, so dass $c = f_{g(m,c)}(m)$ gilt.

- **Chosen Plaintext Attack.** Bei freier Wahl eines Klartextes m und Kenntnis des dazugehörigen Chiffrates $c = f_k(m)$ soll es nicht möglich sein, daraus den Schlüssel k zu berechnen, d.h. es soll keine berechenbare Funktion $k = g(m, c)$ geben, so dass $c = f_{g(m,c)}(m)$ gilt.

Die einzige bekannte Chiffre, die beweisbar sicher ist, ist der **One-Time-Pad**: Dabei ergibt sich das Chiffrat aus der bitweisen XOR-Verknüpfung von Klartext und Schlüssel. Der Schlüssel besitzt also die gleiche Länge wie das Chiffrat und wird nur einmal verwendet. Nimmt man ideale stochastische Eigenschaften der Schlüssel-Bitfolge an, so ist zu einem Chiffrat jede beliebige Bitfolge als dazugehöriger Klartext gleich wahrscheinlich. Somit ist die erste Forderung erfüllt. Die zweite und dritte Forderung sind, so wie sie oben formuliert sind, zwar nicht erfüllt, denn sind Klartext und Chiffrat bekannt, kann man den Schlüssel sofort errechnen. Da beim One-Time-Pad aber jeder Schlüssel nur einmal verwendet wird, sind Known Plaintext Attack und Chosen Plaintext Attack wirkungslos, denn sie liefern keine Infomationen für das Entschlüsseln zukünftiger Chiffrate.

2.2 Die Caesarchiffre

Die Caesarchiffre ist ein Beispiel für eine sehr einfache Chiffre, anhand der man sich obige Forderungen klarmachen kann. Sie geht auf Julius Caesar zurück, der damit Lageberichte in Schlachten verschlüsselt haben soll [Bauer 94]. Das Klartext- und Chiffretext-Alphabet besteht jeweils aus den 26 Buchstaben A,B,C,...,Z, denen die Zahlenwerte 0,1,2,...,25 zugeordnet werden. Der Schlüssel k ist eine Zahl zwischen 1 und 25. Beim Verschlüsseln werden die Buchstaben einfach um k Stellen im Alphabet

verschoben. Ist beispielsweise $k=3$, so wird der Text ARBEITSTAGUNG zu DUEHLWVWDJXQJ chiffriert. Mathematisch ausgedrückt ist

$$c = f_k(m) = (m + k) \bmod 26$$

die Verschlüsselungsfunktion der Caesarchiffre.

Die drei oben genannten Kriterien werden von der Caesarchiffre alle nicht erfüllt.

- Ist ein Chiffrat bekannt, so benötigt man lediglich 25 Versuche, um den Klartext herauszubekommen. Dies kann nicht als schwierig bezeichnet werden. Wenn man weiss, dass der Klartext in deutsche/englischer/... Sprache abgefasst ist, und kennt man Häufigkeitstabellen der Buchstaben in den jeweiligen Sprachen, dann genügen sogar erheblich weniger Versuche.
- Sind Klartext und Chiffrat bekannt, erhält man sofort den Schlüssel, indem man einfach die Zahlenwerte der Buchstaben voneinander subtrahiert.
- Darf der Klartext frei gewählt werden, ergibt die Eingabe eines A (Zahlenwert 0) sofort den Schlüssel.

Varianten der Caesarchiffre sind die **Viginerè-Chiffre**, bei der verschiedene Schlüsselwerte zyklisch zum Einsatz kommen, und die **Permutationschiffre**, bei der die Klartextbuchstaben nicht durch eine Verschiebung im Alphabet, sondern durch eine Permutation auf die Buchstaben des Chiffrates abgebildet werden. Auch diese Varianten halten zumindest keiner chosen-plaintext-Attacke stand: Die Eingabe von AAAAA... bei der Viginerè-Chiffre bzw. ABCD...Z bei der Permutationschiffre liefert sofort den Schlüssel.

2.3 Klassifizierung von Chiffren

Klassische Chiffren werden in der Literatur in Strom- und Blockchiffren unterteilt. Bei Stromchiffren ergibt sich ein Zeichen des Chiffrates aus jeweils einem Zeichen des Klartextes und des Schlüssels. Bei Blockchiffren erfolgt die Chiffrierung in Blöcken von fest vorgegebener Länge. Innerhalb eines Chiffratblockes hängt jedes Zeichen von allen Zeichen des dazugehörigen Klartextblockes ab. Strenggenommen gibt es keine Unterscheidung zwischen Strom- und Blockchiffren. Denn durch geeignete Definition des Begriffes “Zeichen” lässt sich jede Blockchiffre auch als Stromchiffre darstellen.

Beispielsweise ist die oben beschriebene Caesarchiffre eine Stromchiffre, denn jeder Buchstabe des Chiffrates ergibt sich aus genau einem Buchstaben des Klartextes. Definiert man als Zeichen jedoch nicht einen Buchstaben, sondern ein Bit der (ASCII-) Repräsentation des Buchstabens, so handelt es sich um eine Blockchiffre mit Blockgrösse 8 Bit. In der Literatur hat sich die Unterscheidung jedoch durchgesetzt, und sowohl die Caesar-Chiffre als auch der One-Time-Pad werden zu den Stromchiffren gezählt.

Abgesehen vom One-Time-Pad, dass u.a. zur Verschlüsselung des roten Telefons zwischen Moskau und Washington verwendet wird, ist heute keine ernsthafte Verwendung von Stromchiffren mehr öffentlich bekannt. Statt dessen kommen Blockchiffren zum Einsatz, wenn klassische Kryptographie benötigt wird.

2.4 Der Data Encryption Standard

Die wohl bekannteste Blockchiffre ist der Data Encryption Standard (DES), der 1977 von der NSA entworfen wurde und bis heute im Einsatz ist [ANSI 81]. Beim DES werden Klartextblöcke zu je 64 Bit auf 64 Bit lange Chiffretextblöcke abgebildet. Die Schlüssellänge beträgt 56 Bit. Die Sicherheit des DES ist umstritten. Sowohl die Geheimniskrämerei der NSA bezüglich der Entwurfskriterien für den DES als auch mehrere veröffentlichte Angriffsversuche geben Anlass zu der Vermutung, dass der DES knackbar ist. Es ist jedoch bis heute kein effizienterer Angriff als das vollständige Absuchen aller Schlüssel bekannt.

Der Hauptkritikpunkt am DES ist jedoch die geringe Schlüssellänge von 56 Bit. In [Diffie 77] wird beschrieben, wie mit Investitionskosten von einer Million US-Dollar eine Maschine gebaut werden kann, die eine DES-Verschlüsselung in einem Tag knackt. Berücksichtigt man den schnellen technischen Fortschritt, so kann man davon ausgehen, dass in wenigen Jahren eine solche Maschine erheblich billiger und schneller sein wird und somit eine Schlüssellänge von 56 Bit keinen ernsthaften Schutz mehr bieten kann. Ein Vorschlag zur Lösung dieses Problems ist der Triple-DES (3DES), bei dem drei DES-Verschlüsselungen hintereinander ausgeführt werden. Aus einem Klartext m wird beim 3DES das Chiffrat c mithilfe zweier 56-Bit-Schlüssel k und l berechnet:

$$c = DES_k(DES_l^{-1}(DES_k(m)))$$

Der Vorteil des 3DES besteht darin, dass zur Verschlüsselung weiterhin die alten DES-Hardwareimplementierungen verwendet werden können, die schnell arbeiten und billig

erhältlich sind. Die Vermutung, dass die Menge der DES-Verschlüsselungen bezüglich Komposition abgeschlossen ist und somit der 3DES die gleiche kryptographische Komplexität wie ein einfacher DES besitzt, konnte erst 1992 widerlegt werden [Campbell 92].

2.5 Der International Data Encryption Algorithmus

Eine neuere Blockchiffre, bei der die Entwurfskriterien öffentlich bekannt sind, ist der International Data Encryption Algorithm (**IDEA**) [Lai 92]. Die Blockgrösse von Klartext und Chiffrat beträgt wie beim DES 64 Bit, die Schlüssellänge jedoch 128 Bit. Eine IDEA-Verschlüsselung besteht aus neun Runden. In jeder Runde werden die Daten in 4 Wörter zu je 16 Bit aufgeteilt. Diese Wörter werden mittels dreier Verknüpfungsoperationen miteinander und mit 16 Bit langen Teilschlüsseln verknüpft. Zu den wesentlichen Stärken des IDEA gehört, dass beim Entwurf auf die Beweisbarkeit seiner kryptographischen Eigenschaften Wert gelegt wurde. Das Chiffrat soll in einer möglichst schwer vorhersagbaren Weise vom Klartext abhängen (**Konfusion**), und jedes Bit der Ausgabe soll von jedem Bit der Eingabe abhängen (**Diffusion**):

- Der Zusammenhang zwischen Chiffrat und Klartext ist sehr komplex. Dazu wurden die drei Verknüpfungen so gewählt, dass sie möglichst wenig mathematische Struktur haben. Zum Beispiel gelten für zwei verschiedene Verknüpfungen keine Assoziativ- und Distributivgesetze, es sind also für 16-Bit-Wörter a, b und c zwei verschiedene Verknüpfungen + und *

  $(a+b)*c = a+(b*c)$ oder $(a+b)*c = (a*c)+(b*c)$

  keine allgemeingültigen Formeln.

- Es gelten keine Isomorphiebeziehungen zwischen den Verknüpfungen, es gibt also beweisbar keine Funktion f, die

  $f(a+b) = f(a) * f(b)$

  für zwei unterschiedliche Verknüpfungen + und * erfüllt.

- Die Anordnung dieser Operationen sind so gewählt, dass die daraus entstehenden Ausdrücke nicht vereinfachbar sind. Insbesondere gilt für die vier Verknüpfungen im

mittleren Block jeder Runde, dass es beweisbar keine Verknüpfungen geben kann, die das Ergebnis mit nur drei Rechenschritten errechnen können.

- In jeder Runde hängen alle 64 Ausgabebits von allen 64 Eingabebits der Rundenfunktion ab. Kleine Aenderungen bei der Eingabe wirken sich also stark in der Ausgabe aus.

Aufgrund seiner übersichtlichen Struktur ist der IDEA sehr einfach in Software zu implementieren. Die Ausführungsgeschwindigkeit ist jedoch geringer als bei den anderen gängigen Blockchiffren.

Der wesentliche Nachteil des IDEA ist die Tatsache, dass der Algorithmus patentiert ist. Das Patent wird von der Schweizer Firma Ascom gehalten, die IDEA in Zusammenarbeit mit der ETH Zürich entworfen hat. Die Lizenzbedingungen erlauben die unentgeltliche private Nutzung des Algorithmus; bei gewerblicher Nutzung werden jedoch hohe Lizenzgebühren fällig [Schneier 96, S. 325].

2.6 Public Key Kryptographie

Public Key Kryptosysteme (PKK), auch asymmetrische Kryptosysteme genannt, stellen eine jüngere Entwicklung der Kryptographie dar. Unter diesem Begriff werden mehrere Grundfunktionen zusammengefasst:

- **Verschlüsselung.** Im Unterschied zu den klassischen (symmetrischen) Systemen werden Ver- und Entschlüsselung mit unterschiedlichen Schlüsseln durchgeführt. Jeder Teilnehmer in einem Public-Key-Verschlüsselungssystem besitzt zwei Schlüssel, einen geheimen und einen öffentlichen. Letzterer wird allgemein bekannt gemacht; er ermöglicht es, Dokumente zu verschlüsseln, die ausschliesslich mit dem privaten Schlüssel wieder entschlüsselt werden können.

  Anschaulich kann man sich ein Public-Key-Verschlüsselungssystem wie einen privaten Briefkasten vorstellen. Jeder kann Dokumente in einen Briefkasten einwerfen. Nur der Hausbewohner besitzt jedoch den Schlüssel, kann den Briefkasten öffnen und die Dokumente lesen.

- **Signaturen.** Der Begriff der digitalen Signatur ist erst mit der Erfindung der Public Key Kryptographie aufgekommen, obwohl sich Signaturschemata auch mit symmetrischer Kryptographie realisieren lassen. Bei einem Public-Key-

Signatursystem wird die Signatur zu einem Dokument mit dem geheimen Schlüssel eines Teilnehmers erstellt und ist mit dem öffentlichen Schlüssel von allen Teilnehmern überprüfbar. Sie stellt einen Beweis dafür dar, dass das Dokument mit dem zugehörigen privaten Schlüssel signiert und seitdem nicht mehr verändert wurde. Die digitale Signatur ist das Analogon zur herkömmlichen Unterschrift unter ein Dokument.

- **Blinde Signaturen.** Eine blinde Signatur ist ein Verfahren zur Signierung eines Dokumentes mit einem privaten Schlüssel, ohne dass der Signierende Einblick in das Dokument nehmen kann. Dennoch können andere, denen das Dokument offen vorliegt, die Signatur überprüfen.

  Den Einsatz eines blinden Signaturschemas kann man anschaulich mit einer Wahl erklären. Der Wähler füllt seinen Stimmzettel im Geheimen aus und steckt ihn in den Wahlumschlag. Diesen wirft er unter den Augen der Aufsichtspersonals in die Wahlurne. Das Personal bestätigt die Stimmabgabe in Form eines Eintrages in die Wahlunterlagen, ohne den Inhalt des Stimmzettels zu kennen.

In einem Public-Key-Kryptosystem besitzt jeder Teilnehmer zwei Schlüssel, einen privaten und einen öffentlichen. In der Literatur wird der öffentliche Schlüssel meist mit *e* (für encryption), der private mit *d* (für decryption) bezeichnet. Ein Public Key Kryptosystem besteht also aus zwei Funktionen f_e und g_d mit den Eigenschaften

$$c = f_e(m) \qquad \text{und} \qquad m = g_d(c)$$

sowie einer Schlüsselerzeugungsfunktion

$$(d, e) = \text{kgen}(random)$$

Die Arbeitsweise der Funktionen *f* und *g* ist allen Teilnehmern bekannt. Zwischen den beiden Schlüsseln besteht ein mathematischer Zusammenhang.

Jeder Teilnehmer in einem Public-Key-Verschlüsselungssystem besitzt also ein Schlüsselpaar (*e*,*d*). *e* wird als öffentlicher Schlüssel bezeichnet; er ist jedem anderen Teilnehmer des Systems bekannt. *d* ist dagegen der private Schlüssel und muss geheim gehalten werden. Der Vorteil eines Public-Key-Kryptosystems besteht darin, dass jeder Teilnehmer eine verschlüsselte Nachricht generieren kann, die nur von einem bestimmten anderen Teilnehmer entschlüsselt werden kann.

Für die Funktionen *f* und *g* muss also gelten

(1) Für jeden Klartext m muss $g_d(f_e(m)) = m$ sein, d.h. g_d ist die Umkehrfunktion von f_e.

(2) Ein Angreifer, der eine verschlüsselte Nachricht in die Hände bekommt, darf trotz Kenntnis von c und e in der Beziehung $c = f_e(m)$ nicht den Klartext m zurückgewinnen können.

(3) Ein Angreifer, der eine Nachricht mit dem öffentlichen Schlüssel eines anderen Teilnehmers verschlüsselt, darf trotz Kenntnis von m und c in der *Beziehung* $m = g_d(c)$ nicht den privaten Schlüssel d errechnen können.

Aus (1) folgt, dass zwischen den Schlüsseln e und d mathematisch gesehen eine Beziehung bestehen muss. (2) besagt, dass es nicht möglich sein darf, die Funktion f_e umzukehren, obwohl eine Umkehrfunktion (nämlich g_d) existiert. (3) heisst mathematisch ausgerückt, dass die Gleichung $m = g_d(c)$ nicht nach d auflösbar sein darf.

Es ist leicht einzusehen, dass zumindest die Forderung (2) nicht erfüllbar ist. Denn wenn eine Umkehrfunktion zu f_e existiert, kann man auch einen Algorithmus angeben, der sie berechnet: Nämlich das Durchprobieren aller Klartexte m, solange bis ein m mit $c = f_e(m)$ gefunden ist. In den real existierenden Public-Key-Kryptosystemen ist daher in (2) (und ebenso in (3)) die Formulierung dahingehend abzuschwächen, dass es keine Möglichkeit geben darf, innerhalb vernünftiger Zeit die gesuchte Grösse zu ermitteln. Auch diese Forderung ist (bisher) noch nicht mit konkreten Funktionen erfüllbar. Denn bisher ist noch kein schwieriges mathematisches Problem bekannt, das beweisbar eine Lösung hat, es aber beweisbar keinen effizienten Algorithmus zur Lösungsfindung gibt. Um Public Key Kryptosysteme realisieren zu können, bedient man sich daher mathematischer Funktionen, für die trotz intensiver Forschungsarbeit noch kein effizienter Algorithmus zur Berechnung der Umkehrfunktion gefunden wurde.

Ohne auf die Details konkreter Realisierungen von Public Key Kryptosystemen einzugehen, seien hier kurz einige Beispiele für Funktionen genannt, mit denen die genannten Bedingungen zu erfüllen versucht werden:

- **Das Faktorisierungsproblem.** Seien p und q zwei grosse Primzahlen (mit über 100 Dezimalstellen). Dann ist es leicht, die Funktion $n = h(p,q) := p*q$ zu berechnen, d.h. die beiden Primzahlen miteinander zu multiplizieren. Umgekehrt ist es jedoch sehr schwer, ein bekanntes Produkt zweier grosser Primzahlen zu faktorisieren, d.h.

p und q zu berechnen. Ein guter Ueberblick über das Faktorisierungsproblem findet sich in [Bressoud 89].

- **Der diskrete Logarithmus.** Sei p eine grosse Primzahl und (K[p], +, *) der endliche Körper mit p Elementen, dargestellt durch die natürlichen Zahlen {0, 1, ..., p-1} und die Verknüpfungsoperationen modulo p. Dann ist es leicht, zu festem $a>1$ und gegebenem x ein y mit $y=a^x \bmod p$ zu berechnen. Umgekehrt ist es dagegen schwer, aus einem bekannten y das x zurückzugewinnen, d.h. den Logarithmus in einem Primkörper zur Basis a zu berechnen. Ein Ueberblick über die Stand der Forschung zum diskreten Logarithmus-Problem findet man in [McCurley 90].

- **Das diskrete Wurzel-Problem.** Sei n eine grosse zusammengesetzte Zahl. Dann ist es leicht, zu einem festen $a>1$ und gegebenem x ein y mit $y=x^a \bmod$ n zu berechnen. Ist dagegen y bekannt, so ist es schwer, x zu berechnen. Falls n ein Produkt zweier Primzahlen ist, dann ist die Berechnung der diskreten Wurzel modulo n vom Aufwand her äquivalent zur Faktorisierung von n [Rabin 79].

“leicht” bedeutet dabei jeweils, dass es einen effizienten Algorithmus für die Berechnung gibt; “schwer” dagegen heisst, dass bisher kein Algorithmus gefunden wurde, der das Problem in vernünftiger Zeit löst. Sollte sich dies einmal ändern, sind die Public-Key-Verfahren, die auf diesen Problemen beruhen, gebrochen.

2.7 RSA

Das wohl bekannteste Public Key Kryptosystem ist das nach seinen Erfindern Rivest, Shamir und Adleman benannte RSA [Rivest 78]. Es basiert auf allen drei der oben genannten Einwegfunktionen; wird auch nur eines der beschriebenen Probleme gelöst, so ist RSA gebrochen. Dies ist jedoch in den letzten 21 Jahren nicht geschehen (oder zumindest ist nichts davon öffentlich bekannt).

Mit RSA können zwei der oben genannten Verfahren durchgeführt werden, nämlich Public-Key-Verschlüsselung und digitale Signaturen.

Die Schlüsselpaare können von den Benutzern selbst erzeugt werden, es ist keine Schlüsselvergabeinstanz notwendig. Der öffentliche Schlüssel besteht bei RSA aus zwei Zahlen, eine in der Grössenordnung von 400 bis 800 Dezimalstellen, die andere im

allgemeinen 1-2 Stellen. Der private Schlüssel enthält eine Zahl von etwa gleicher Länge wie die grosse Zahl des öffentlichen Schlüssels.

RSA-Berechnungen bestehen aus Langzahlarithmetik mit Zahlen in der Grössenordnung der Schlüssellänge. Daher sind sie erheblich langsamer als Chiffrier- und Dechiffrieroperationen bei Blockchiffren. Eine SPARC II braucht bei einem 1024-Bit-Schlüssel etwa eine Sekunde für die geheimen Berechnungen (Dechiffrierung und Signaturerstellung), die öffentlichen Operationen (Chiffrierung und Signaturprüfung) sind etwa um Faktor 10 schneller; dies ist eine Besonderheit bei RSA und hängt damit zusammen, dass die Zahlenwerte des öffentlichen Schlüssels speziell auf Geschwindigkeit optimiert werden können. Hardwareimplementierungen erreichen etwa die 5- bis 10-fache Geschwindigkeit [Schneier 96, S. 469].

Das RSA-Verfahren ist in den USA patentiert, in anderen Ländern ist dieses Patent jedoch ungültig. Das USA-Patent läuft im September 2000 aus [Schneier 96, S. 474].

2.8 Das El-Gamal Signaturverfahren

Ein Verfahren, mit dem auf Basis des diskreten-Logarithmus-Problems digitale Signaturen erstellt werden können, ist das El-Gamal-Verfahren [ElGamal 84, ElGamal 85]. Varianten des ElGamal-Verfahrens erlauben Identitätsbeweise [Beth 88], Passwort-Authentifikation [Chang 91] oder Schlüsselaustausch [Jaburek 90].

Die Komplexität der Berechnungen liegt bei gleicher Schlüssellänge etwas höher als bei RSA. Das El-Gamal-Verfahren basiert jedoch ausschliesslich auf dem diskreten-Logarithmus-Problem. Für dieses sind weniger effiziente Algorithmen bekannt als für das Faktorisierungsproblem, so dass ein El-Gamal-Kryptosystem bei gleicher Sicherheit mit kleinerer Schlüssellänge auskommt. Typische Schlüssellängen sind 60-200 Dezimalstellen lang. Ein öffentlicher Schlüssel besteht aus zwei Zahlen dieser Grössenordnung.

Das El-Gamal-Verfahren ist nicht patentiert. Jedoch behauptet die Firma Public Key Partners (PKP) in den USA, El-Gamal falle unter das Patent des **Diffie-Hellmann-Kryptosystems** [Hellmann 80]. Da jedoch auch dieses Patent 1997 ausläuft, wird die Streitfrage wohl nie geklärt werden [Schneier 96, S. 479].

2.9 Kryptographische Hashfunktionen

Ein wesentliches Hilfsmittel zur Konzeption kryptographischer Systeme stellen kryptographische Hashfunktionen (auch **Einwegfunktionen** genannt) dar.

Eine herkömmliche Hashfunktion h bildet eine Menge von Texten T auf eine Menge von Hashsummen S ab, wobei S wesentlich kleiner als T ist. Es gibt Kollisionen, also unterschiedliche Texte t_1, t_2 aus T mit $h(t_1) = h(t_2)$. Die Funktion h ist so gewählt, dass alle Hashwerte möglichst gleich häufig auftreten sollen. An eine **kryptographische Hashfunktion** werden zusätzlich noch folgende Forderungen gestellt:

(1) Es soll unmöglich sein, zu einer gegebenen Hashsumme s einen Text t zu finden, so dass $s = h(t)$ gilt.

(2) Es soll unmöglich sein, zwei unterschiedliche Texte t_1 und t_2 zu finden, so dass $h(t_1) = h(t_2)$ gilt.

Es ist leicht einzusehen, dass Forderung (2) nicht erfüllbar ist. Denn da Kollisionen existieren, lassen sie sich auch durch systematisches Probieren aller Kombinationen finden. Eine ähnliche Ueberlegung gilt für (1). Für praktische Realisierungen von kryptographischen Hashfunktionen ist daher das “unmöglich” durch “praktisch unmöglich” zu ersetzen: Die bekannten Realisierungen haben eine Hashsummenlänge von 128 Bit oder mehr, und erfordern daher das Durchprobieren von etwa 2^{128} Texten zum Verletzen der ersten, sowie etwa 2^{64} Texten zum Verletzen der zweiten Forderung. Natürlich ist nicht garantiert, dass zu einer konkreten Einwegfunktion einmal ein besserer Algorithmus gefunden wird, der die Umkehrfunktion effizienter berechnet; somit muss es in (1) und (2) strenggenommen sogar “mit den bekannten Algorithmen nicht durchführbar” heissen.

2.10 Der Message Digest 5

Der Message Digest 5 (**MD5**) von Rivest [Rivest 92] ist eine Realisierung einer kryptographischen Hashfunktion. Seine Eingabe kann eine Bitfolge beliebiger Länge sein, die Ausgabe besteht aus 128 Bit. Beim MD5 werden jeweils 128 Bit der Eingabe genommen und von einer Funktion mit Gedächtnis weiterverarbeitet. Das Ergebnis dieser Verarbeitung dient als Gedächtnis für die nächsten 128 Bit der Eingabe. Auf diese Weise

hängt die Ausgabe gleichermassen von der gesamten Eingabe ab. Am Anfang ist das Gedächtnis mit einem fest gewählten Initialisierungswert gefüllt.

MD5 ist die verbesserte Version von MD4, bei dem einige Regelmässigkeiten in der Berechnung der Hashsumme aufgedeckt wurden, die für die Kryptoanalyse möglicherweise relevant sein können. Der MD5 galt lange Zeit als sehr sicher. Erst 1996 wurde eine Schwachstelle entdeckt, durch die bei einem veränderten Initialisierungswert den Aufwand zum Auffinden einer Kollision stark verkleinert werden kann [Dobbertin 96]. Ob dieses Ergebnisse Auswirkungen auf die Sicherheit von MD5 hat, ist noch unbekannt.

2.11 PGP

Pretty Good Privacy (PGP) ist eine Implementierung eines Public-Key-Kryptosystems. Sie erlaubt das Verschlüsseln und/oder Signieren von Dateien und bietet eine komfortable Schlüsselverwaltung.

Intern basiert PGP auf RSA, IDEA und MD5. Jeder Teilnehmer kann sich ein Schlüsselpaar selbst erzeugen und seinen öffentlichen Schlüssel verbreiten; eine Vertrauensinstanz ist nicht notwendig.

Bei einer Verschlüsselung mit PGP wird zunächst ein zufälliger Session-Key generiert und die Datei mit diesem Key IDEA-verschlüsselt. Anschliessend wird der Session-Key mit dem öffentlichen RSA-Schlüssel des Empfängers IDEA-verschlüsselt und beide Chiffrate zusammen in eine Datei gepackt. Diese Datei kann man über ein unsicheres Netz an den Empfänger schicken, beispielsweise per E-Mail. Nur der richtige Empfänger kann mit seinem privaten RSA-Schlüssel die RSA-Chiffrierung rückgängig machen und den Session-Key wiedergewinnen. Damit kann schliesslich der Text dechiffriert werden.

Zur Berechnung einer Signatur wird zunächst die MD5-Hashsumme des zu signierenden Textes berechnet und mit dem privaten RSA-Schlüssel des Erstellers signiert. Klartext und Signatur werden zusammen veröffentlicht. Ein Leser, der den öffentlichen Schlüssel des Signierenden besitzt, kann die Signaturoperation rückgängig machen und erhält somit die MD5-Hashsumme zurück. Wenn sie mit einer selbstberechneten Hashsumme über den empfangenen Text übereinstimmt, war die Signatur korrekt; ansonsten ist entweder Text/Signatur unterwegs verfälscht worden, oder der Signaturschlüssel war nicht das Gegenstück zu dem bekannten öffentlichen Schlüssel.

Der Schwachpunkt bei dieser Art von Verschlüsselung ist allerdings der Austausch des öffentlichen Schlüssels. Wer eine Nachricht mit einem öffentlichen Schlüssel chiffriert, muss man sicherstellen, dass dies wirklich der Schlüssel des gewünschten Empfängers ist. Ansonsten könnte ein Angreifer einen selbstgenerierten Schlüssel unter dem Namen eines anderen Netzteilnehmers veröffentlichen. Er besässe dann den zugehörigen privaten Schlüssel und könnte Chiffrate an den öffentlichen Schlüssel dechiffrieren.

Zur Ueberprüfung der Authentizität eines Schlüssels bietet PGP einen Mechanismus an: Der **Fingerprint** ist eine kryptographische Hashsumme über den öffentlichen Schlüssel. Jeder Teilnehmer kann ihn selbst berechnen und auf einem sicheren Kanal (am besten persönlich; für normale Sicherheitsanforderungen genügt wohl auch ein Telefonat) mit dem Besitzer vergleichen.

Schlüssel, bei denen man sicher ist, dass sie dem vorgeblichen Besitzer gehören, kann man mit dem eigenen privaten Schlüssel signieren und signiert weitergeben. Auf diese Weise entsteht das **Web of Trust**: Wenn ein Teilnehmer *A* den öffentlichen Schlüssel von *B* signiert hat (also sicher ist, dass der zugehörige private Schlüssel dem Teilnehmer *B* gehört) und der Schlüssel von *C* eine Signatur von *B* trägt (angefertigt mit dem privaten Schlüssel von *B*), dann hat *A* damit einen Beweis der Authentizität des Schlüssels von *C*. Voraussetzung ist natürlich, dass *B* den Schlüssel wirklich vor dem Signieren überprüft hat. PGP findet solche Vertrauensketten automatisch bzw. gibt eine Warnung aus, wenn ein Schlüssel benutzt wird, zu dem keine Vertrauenskette besteht.

2.12 Kryptographische Protokolle

Basierend auf den Grundtechniken der klassischen Kryptograhie, der Public-Key-Kryptographie und der Einwegfunktionen lassen sich Protokolle generieren, mit denen sehr komplexe Anforderungen realisieren werden können. Beispiele dafür sind:

- **Coin Flipping** [z.B. Blum 82]: Von zwei Personen, die nur über Telekommunikationsleitungen miteinander kommunizieren können, wirft einer eine Münze (d.h. trifft eine Zufallsentscheidung) und beweist dem anderen das Ergebnis.
- **Key Escrow** [z.B. Simmons 92]: Ein Geheimnis, z.B. ein Schlüssel, wird auf mehrere Personen/Instanzen verteilt. Um es zu rekonstruieren, ist die Zusammenarbeit aller oder eines bestimmten Teils der Geheimnisträger nötig.

- **Vertragsabschluss** [z.B. Pfleeger 89]: Mehrere Parteien einigen sich auf ein Dokument/Vertrag/Zahlenwert. Die Einigung tritt gleichzeitig bei allen Teilnehmern in Kraft, und zwar genau dann, wenn alle Parteien den Vertrag unterschrieben haben.

- **Auktionen** [z.B. Reiter 95]: Mehrere Personen geben verdeckt Angebote ab. Am Ende wissen alle, welcher Teilnehmer das höchste Gebot abgegeben hat; die übrigen Gebote sind jedoch keinem anderen Teilnehmer bekannt. Die aktuellen Protokolle setzen jedoch voraus, dass sich mehr als zwei Drittel der Teilnehmer eines Auktionssystems protokollkonform verhalten.

- **Digital Cash** [z.B. Chaum 90]: Die Teilnehmer eines digitalen Zahlungssystems besitzen Münzen verschiedener Werte, die von einer Zentralstelle (der elektronischen Bank) ausgegeben wurden. Die Teilnehmer können mit diesen Münzen handeln, die Echtheit prüfen und sie bei der Bank einzahlen. Es ist nicht möglich, Münzen zu duplizieren. Die Bank kann den Weg der einzelnen Münzen nicht zurückverfolgen, d. h. der Handel mit diesen Münzen hinterlässt keine Datenspur.

Diese Techniken stecken jedoch grösstenteils noch in den Kinderschuhen. Zum einen erfordern sie teilweise einen sehr hohen Rechen- und Kommunikationsaufwand und sind deshalb nur in kleinen Teilnehmerkreisen durchführbar. Zum anderen befinden sich Entwurf und Kryptoanalyse komplexerer Protokolle in einen ständigen Wettlauf, und es werden noch häufig Fehler und Sicherheitslücken in bekannten Protokollen gefunden.

2.12 Aktuelle Forschungsthemen

Es gibt zahlreiche aktuelle Forschungsaktivitäten in der Kryptologie. Sie lassen sich im wesentlichen einteilen in

- Untersuchung der Sicherheit bekannter Algorithmen und Protokolle und Auffinden von Schwachstellen in bestehenden Systemen;

- Entwicklung neuer kryptographischer Verfahren, die trotz der sich ständig wachsenden Geschwindigkeit von Rechnern und Prozessoren auf absehbare Zeit nicht gebrochen werden können;

- Weiterentwicklung kryptographischer Protokolle für komplexe Anwendungen wie digitale Zahlungsmittel, Vertragsabschluss oder elektronische Wahlen.

3. Funktionsweise und Sicherheitseigenschaften von TCP/IP-Netzen

3.1 Uebersicht

Das Protokoll, das von allen Rechnern des Internet genutzt wird, ist TCP/IP. Dieses Protokoll zu beherrschen, ist also die Minimalvoraussetzung, um einen Rechner an das Internet anzuschliessen.

Prinzipiell ist TCP/IP für Vernetzungen von beliebig vielen Rechnern und beliebig grossen/kleinen Entfernungen geeignet. Dasselbe gilt für die Netzwerkhardware: TCP/IP funktioniert mit allen gängigen Netzwerktopologien (z.B. Ethernet, Token-Ring) und Kombinationen daraus. TCP/IP wird von allen gängigen Betriebssystemen unterstützt.

Das TCP/IP-Schichtenmodell besteht aus vier Schichten (Abbildung 3.1).

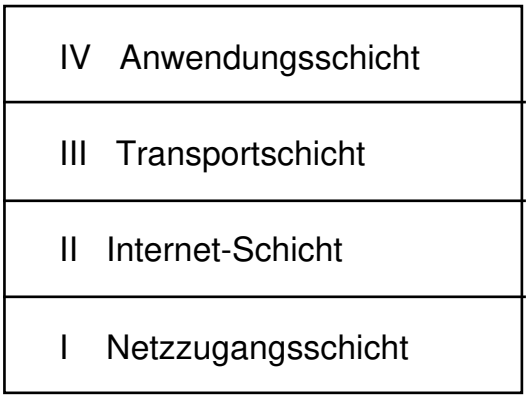

Abb. 3.1: Das TCP/IP-Schichtenmodell

Im Gegensatz dazu sieht das ISO/OSI Referenzmodell sieben Schichten vor. Abbildung 3.2 zeigt eine mögliche Einordnung von TCP/IP in das OSI-Modell [Hunt 92].

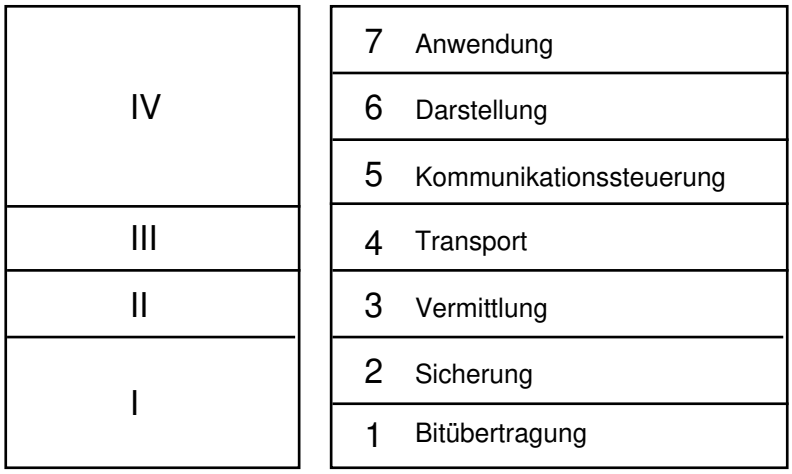

Abb. 3.2: Einordnung von TCP/IP in das OSI-Schichtenmodell

Die Zuordnung ist nicht eindeutig; insbesondere wird in einigen Büchern auch die Internet-Schicht (II) in die OSI-Schicht 4 eingeordnet.

3.2 Aufgabe und Arbeitsweise der Protokollebenen

Die vier Schichten des TCP/IP-Protokollstacks bilden eine Hierarchie im Bezug auf die Abstraktionsebene und auf den Datenfluss. Jede Schicht kann "direkt" mit der entsprechenden Schicht auf der Gegenseite kommunizieren, indem sie die Funktionen der darunterliegenden Schicht benutzt (siehe Abb. 3.3).

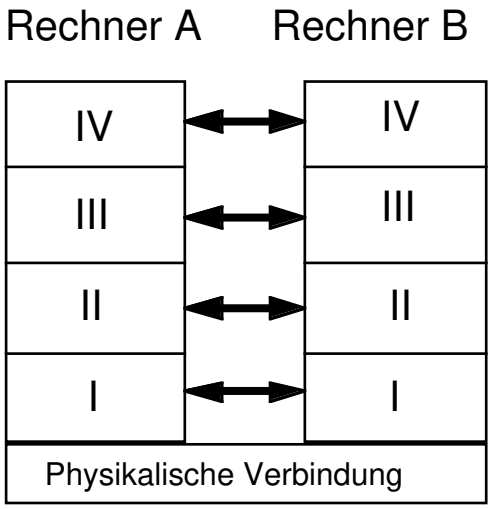

Abb. 3.3: Abstraktionsebenen des TCP/IP-Protokolls

Eine physikalische Verbindung zu einem anderen Rechner besteht nur unterhalb der Netzzugangsschicht. Ein Protokoll auf Schicht I kann mit dem entsprechenden Protokoll auf der anderen Seite Daten austauschen, indem es die physikalische Verbindung nutzt. Ein Protokoll auf der Schicht II kann dies tun, indem es die Transportfähigkeiten der Schicht I benutzt. Dasselbe gilt sinngemäss für die darüberliegenden Schichten.

Natürlich leiten die einzelnen Schichten nicht nur Daten weiter. Jede Schicht erfüllt eine spezifische Funktion. Die Aufgaben der einzelnen Schichten werden noch genauer erläutert.

Zur Erfüllung ihrer Aufgabe muss eine Schicht im Allgemeinen den Daten zusätzliche Steuerinformationen hinzufügen. Dies können Informationen über das Datenformat, die Beschaffenheit und Qualität der Verbindung oder eine Prüfsumme, die die Integrität der

Daten sichern soll, sein. Die Datenmenge wird also um so grösser, je weiter die Daten im Schichtenmodell nach unten wandern. Abbildung 3.4 verdeutlicht dies.

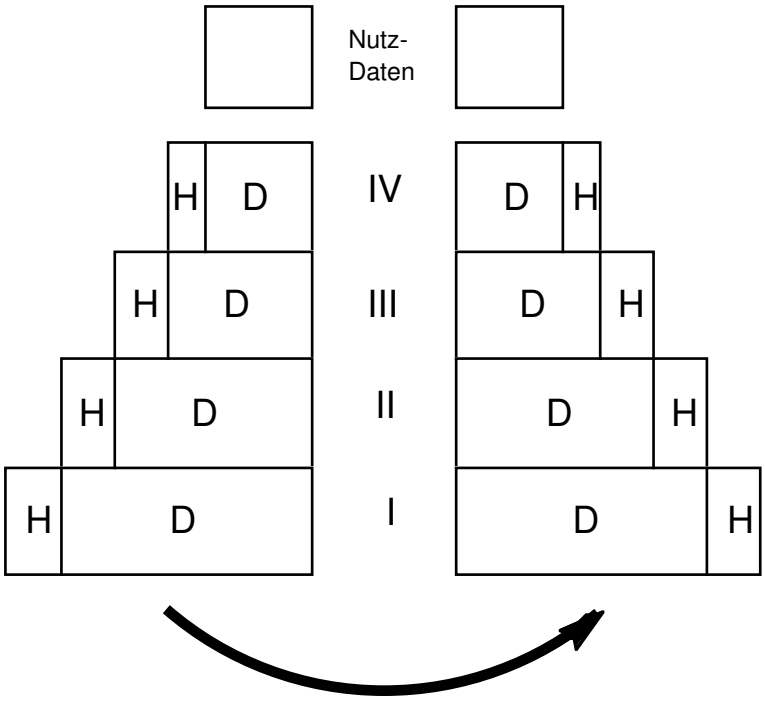

Abb. 3.4: Daten und Kontrollinformationen (D: Daten, H: Header der einzelnen Schichten)

Aus der Abbildung wird mehreres deutlich:

- Die Nutzdaten der Anwendung bilden die Daten (Data, D) der Anwendungsschicht IV. Die Schicht fügt ihre eigenen Kontrollinformationen (Header, H) hinzu und leitet das ganze Paket an die Schicht III weiter. Die übrigen Schichten verhalten sich entsprechend.
- Eine Schicht interessiert sich nicht für die Unterteilung in Daten und Header der darüberliegenden Schicht, sondern betrachtet das, was sie übergeben bekommt, komplett als zu transportierende Daten.

- Die Datenmenge, die tatsächlich über die Leitung geht, ist häufig erheblich grösser als die Nutzdaten auf der Anwendungsebene (auch wenn dies in der Abbildung übertrieben dargestellt ist).

Die Header werden meistens, aber nicht immer, an den Anfang der Nutzdaten gehängt. Eine Ausnahme bildet z.B. das Ethernet-Protokoll; dort gibt es Steuerinformationen sowohl vor als auch hinter den Nutzdaten.

In der Literatur zu TCP/IP [z.B. Hunt 92] werden die zu transportierenden Daten der Schicht I (Header und Nutzdaten zusammen) **Frames**, die der Schicht II **Datagramme** und die der Schicht III je nach Protokoll **Pakete** oder **Datenströme** genannt. Auf der Schicht IV gibt es keine einheitlichen Bezeichnungen.

3.3 Die Netzzugangsschicht

Aufgabe der Protokolle der Netzzugangsschicht ist es, sich um den Datentransport auf einem physikalischen Netzwerk zu kümmern. Mit den unterschiedlichen physikalischen Medien und Uebertragungsverfahren unterscheiden sich auch die Protokolle dieser Schicht; dennoch gibt es Aufgaben, die alle Protokolle der Schicht I gemeinsam haben, beispielsweise Kontrolle des Datenflusses auf der Leitung und Absicherung der Daten mittels Prüfsummenverfahren vor unbeabsichtigten Verfälschungen.

Gängige Betriebssysteme bieten Schicht-I-Protokolle mindestens für folgende Uebertragungsmedien: V.24/RS232, Centronics, ISDN, Ethernet, Token Ring.

3.4 Das Internet Protocol (IP)

IP ist das Schicht-II-Protokoll des TCP/IP-Protokollstacks. Es ermöglicht den Datentransport auch zwischen Rechnern, die nicht unmittelbar mit einer physikalischen Leitung verbunden sind. Das Finden eines Weges von einem Quell- zu einem Zielrechner in einem Netz wird als Routing bezeichnet. Eine weitere Aufgabe der Schicht II ist die einheitliche Adressierung aller Rechner eines Netzwerkes, unabhängig von der verwendeten Netzwerktechnologie.

Jeder Rechner in einem TCP/IP-Netzwerk besitzt (mindestens) eine eindeutige Kennung, genannt IP-Nummer. Die IP-Schicht jedes Rechners besitzt Informationen darüber, an

welche (direkt verbundenen) Rechner Datagramme für bestimmte Ziel-IP-Nummern weiterzuleiten sind, damit sie irgendwann ihr Ziel erreichen. Aufbau der IP-Nummern und Ablauf des Routings werden unten beschrieben.

Die Funktionalität von IP besteht also darin, Datagramme an bestimmte Rechner im Netz weiterzuleiten. Es findet keine Bestätigung über das Ankommen der Daten statt, die Daten werden nicht vor Verfälschung geschützt und einzelne Datagramme können sich unterwegs überholen. Alle drei Eigenschaften sind häufig unerwünscht; es ist Aufgabe der darüberliegenden Schichten, sich darum zu kümmern.

Der IP-Header kann 20 oder 24 Bytes lang sein; ersteres ist der Normalfall. Er enthält Steuerinformationen, die für das Weiterleiten der Datagramme und die Verarbeitung auf dem Zielrechner wichtig sind. Unter anderem sind dies:

- die IP-Nummer des absenden Rechners
- die IP-Nummer des Zielrechners
- die Protokollnummer des Schicht-III-Protokolles, von dem die Daten kommen (und an das sie weitergereicht werden sollen, wenn sie auf dem Zielrechner angekommen sind)
- die Länge des Datagrammes
- eine Prüfsumme über den Header, um Verfälschungen erkennen zu können
- Flags zum Aktivieren spezieller Optionen; auf eine davon wird später noch näher eingegangen.

3.5 Das Internet Control Message Protocol (ICMP)

ICMP ist ein Protokoll der Schicht III. Das bedeutet, dass ICMP-Datenpakete über IP transportiert werden. Die Aufgabe von ICMP besteht im Wesentlichen darin, Fehler- und andere Statusmeldungen zu versenden, die den IP-Datagrammtransport betreffen.

ICMP-Fehlermeldungen werden zum Beispiel generiert, wenn ein Datagramm empfangen wurde, für dessen Zieladresse keine Routinginformationen bekannt sind, oder wenn der IP-Header fehlerhaft ausgefüllt ist (die Prüfsumme aber korrekt ist). ICMP-Fehler werden nicht generiert, wenn die Prüfsumme im Header fehlerhaft ist.

Die Maxime lautet, dass Fehlermeldungen nur dann verschickt werden, wenn der absendende Rechner schuld an dem Fehler ist. Fehlerhafte Prüfsummen dagegen liegen meistens an Uebertragungsfehlern auf dem physikalischen Medium. Der Absender kann dafür nichts, und hat deshalb auch nichts von einer solchen Meldung. Ausserdem steht der Absender nicht zweifelsfrei fest, denn wenn die Header-Prüfsumme fehlerhaft ist, könnte ja auch die Absender-IP-Nummer verfälscht worden sein.

Des weiteren werden niemals Fehlermeldungen über fehlerhafte ICMP-Pakete verschickt (weil dies sonst zu einer endlosen Kette von Fehlermeldungen führen könnte), oder wenn ein fehlerhaftes Datagramm an mehrere Empfänger verschickt wurde (denn sonst würde die Netzlast explodieren).

Neben Fehlermeldungen gibt es auch noch andere Informationen, die über ICMP verschickt werden können. Die wichtigsten davon sind:

- ***echo request*** und ***echo reply*** ermöglichen einen Mechanismus zum Testen, ob ein Rechner im Netzwerk erreichbar ist. Wenn die TCP/IP-Implementierung ein *echo request* empfängt, schickt sie ein echo reply an den Absender. Dieser Mechanismus wird von dem Kommando *ping* verwendet.
- Ein ***redirect*** teilt der IP-Schicht mit, dass ein Rechner oder ein Netzwerk über eine neue Route erreichbar ist, ändert also die Routing-Tabelle der IP-Schicht.

3.6 Das Transmission Control Protocol (TCP)

TCP ist das zentrale Protokoll auf Schicht III des TCP/IP Protokollstacks. Es ermöglicht einen verbindungsorientierten, reihenfolgetreuen, bidirektionalen und gesicherten Datenstrom zwischen zwei Programmen auf verschiedenen Rechnern (oder auch dem gleichen Rechner). Die Aufgaben von TCP sind also

- Aufteilen eines beliebig langen Datenstromes in einzelne Segmente;
- Verschicken dieser Segmente über IP;
- Wiederholtes Verschicken von Segmenten, die defekt oder gar nicht beim Empfänger angekommen sind;
- Eliminieren duplizierter Segmente;

- Reihenfolgetreues Zusammensetzen des Datenstromes;
- Weitergabe des Datenstromes an das dazugehörige Programm;
- Management von Verbindungen zwischen Programmen.

Der Ablauf einer TCP-Verbindung sieht, stark vereinfacht, etwa so aus wie in Abbildung 3.5 dargestellt.

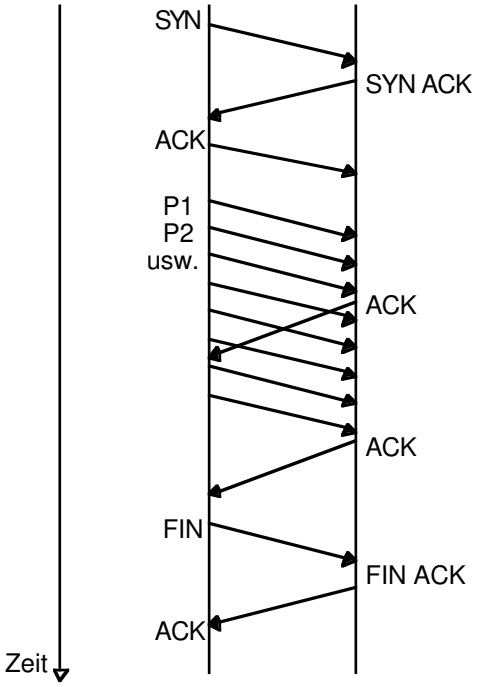

Abb. 3.5: TCP-Datentransport

Der Verbindungsaufbau besteht aus drei TCP-Steuerpaketen (die überhaupt keine Nutzdaten enthalten): SYN (Synchronise), SYN ACK (SYN Acknowledge) und ACK. Darauf folgt die Uebertragung von Datenpaketen (in der Abbildung zur Vereinfachung nur in eine Richtung dargestellt) und Bestätigungen korrekt empfangener Daten. Der Verbindungsabbau wird mit FIN eingeleitet; die Verbindung ist beendet, wenn die Gegenseite das FIN bestätigt hat.

Korrekt empfangene Datenpakete werden bestätigt. Eine Bestätigungsnachricht (ACK) kann mehrere Datenpakete bestätigen. Dadurch wird zum einen die Netzlast gesenkt, zum

anderen kann die sendende Seite neue Daten sofort nachschieben, auch wenn die letzten Pakete noch nicht bestätigt wurden. Pakete, die längere Zeit nicht bestätigt wurden, werden neu verschickt. Auf diese Weise werden Paketverluste und Verfälschungen korrigiert.

Die Reihenfolgetreue wird dadurch erreicht, dass die einzelnen Pakete mit **Sequenznummern** versehen sind. Aufeinanderfolgende Pakete in eine Richtung haben fortlaufende Sequenznummern; für beide Uebertragungsrichtungen gibt es getrennte Numerierungen. Dadurch kann der Datenstrom in seiner ursprünglichen Reihenfolge zusammengesetzt werden, auch wenn sich einzelne Pakete unterwegs überholen.

Um Datenverfälschung zu vermeiden, wird eine Prüfsumme über Header und Nutzdaten berechnet und beim Empfänger überprüft. Die Prüfsumme ist 16 Bit lang, so dass Uebertragungsfehler mit hoher Wahrscheinlichkeit erkannt werden.

Um die Zuordnung der einzelnen Datenpakete zu den Datenströmen der Programme zu ermöglichen, zwischen denen die Verbindung besteht, enthält der TCP-Header zwei Zahlen, die eine eindeutige Zuordnung der Pakete zu den Datenströmen ermöglicht: Absender- und Empfänger-Portnummer. Genauer: Eine TCP-Verbindung zwischen zwei Programmen auf unterschiedlichen Rechnern ist eindeutig gekennzeichnet durch die Angaben

- Absender-IP-Nummer
- Absender-Portnummer
- Empfänger-IP-Nummer
- Empfänger-Portnummer

Der Rechner, der eine TCP-Verbindung öffnet, wählt als Absender-Portnummer immer eine Zahl, der noch keine ausgehende TCP-Verbindung zugeordnet ist. Auf der Gegenseite können mehrere Verbindungen auf die gleiche Portnummer geöffnet werden, ohne dass die Eindeutigkeit verletzt wird. Portnummern liegen zwischen 0 und 65536.

Der Header eines TCP-Paketes enthält also unter anderem

- Absender-Portnummer
- Empfänger-Portnummer

- Sequenznummern (getrennt für beide Uebertragungsrichtungen)
- Prüfsumme
- Kennung, um was für eine Art von Paket es sich handelt (Daten, SYN, ACK, ...)

Weiterhin enthält er noch eine Längenangabe und Flags für Optionen, auf die teilweise im Kapitel über Paketfilter noch eingegangen wird.

3.7 Das User Datagram Protocol (UDP)

UDP ist ein weiteres Transportprotokoll auf Schicht III. Es bietet eine einfache Alternative zu TCP. UDP ermöglicht einen ungesicherten, verbindungslosen Transport zwischen zwei Programmen und garantiert auch keine Reihenfolgetreue.

Der Unterschied zum direkten Datentransport über IP besteht also lediglich darin, dass die Daten nicht zwischen zwei Rechnern, sondern zwischen Programmen transportiert werden. Um die Zuordnung zu den Programmen zu gewährleisten, wird das Portnummernkonzept von TCP verwendet. UDP überprüft die Integrität der Datenpakete mittels einer Prüfsumme. Einen Schutz gegen Paketverlust, Ueberholung oder Duplizierung gibt es dagegen nicht; es ist Aufgabe der Anwendungen, sich darum zu kümmern.

Der Vorteil von UDP gegenüber TCP besteht in der kleineren Datenmenge. Der UDP-Header ist lediglich 8 Bytes lang, der TCP-Header dagegen mindestens 20 Bytes. Der TCP-Verbindungsauf- und -abbau erfordert zudem, dass mindestens 5 IP-Datagramme über die Leitung gehen. UDP wird daher häufig für Dienste eingesetzt, bei denen kleine Datenpakete in unregelmässigen Abständen verschickt werden müssen.

3.8 Adressierung in TCP/IP-Netzen

Jeder Rechner in einem TCP/IP-Netz besitzt (mindestens) eine IP-Nummer; diese Nummer muss netzweit eindeutig sein. Am besten sollte diese Nummer weltweit eindeutig gewählt werden, damit es keinen Umstellungsaufwand beim Anschluss an das weltweite Internet gibt.

IP-Nummern sind immer von der Form *a.b.c.d*, wobei $0 \leq a,b,c,d \leq 255$ sind. Genauer gesagt ist eine IP-Nummer eine 32-Bit-Zahl, die üblicherweise durch vier 8-Bit-Zahlen dargestellt wird. Häufig ist es auch nützlich, die 8-Bit-Zahlen als Binärzahl darzustellen, also durch 8 Ziffern, die nur "0" oder "1" sein können. Zum Beispiel ist 11000110_2 = 128+64+4+2 = 198.

Eine IP-Nummer besteht aus zwei Teilen, dem Netzwerkteil und dem Rechnerteil. Sinn dieser Unterteilung ist, dass mehrere Rechner, die an dem selben physikalischen Netzwerkstrang hängen, IP-Nummern mit gleichem Netzwerk- und unterschiedlichem Rechnerteil bekommen sollen. Diese Zuordnung ist nicht bindend, sollte jedoch aus Uebersichtlichkeits- und Effizienzgründen eingehalten werden.

Die möglichen IP-Nummernbereiche sind in verschiedene Klassen unterteilt.

- **Class-A-Netze**: Die IP-Nummer ist von der Form *a.x.y.z*; dabei ist $1 \leq a \leq 126$. Das bedeutet, dass die Binärdarstellung von *a* mit einer 0 beginnt. In einem Class-A-Netz geben die ersten 8 Bits der IP-Nummer (also die Zahl *a*) die Netzwerknummer an; die restlichen 24 Bit enthalten die Rechnernummer. Es gibt also nur 126 verschiedene Class-A-Netze. Jedes dieser Netze kann aber über 16 Millionen Rechner enthalten.
- **Class-B-Netze**: *a.b.x.y* mit $128 \leq a \leq 191$ Die Binärdarstellung von *a* beginnt also mit 10. Bei einem Class-B-Netz besteht der Netzwerkteil der IP-Nummer aus den vorderen 16 Bit, der Rechnerteil aus den hinteren 16 Bit. Es gibt über 16000 verschiedene Class-B-Netze. Jedes davon kann über 65000 Rechner enthalten.
- **Class-C-Netze**: *a.b.c.x* mit $192 \leq a \leq 223$ Die Binärdarstellung von *a* beginnt mit 110. Hier ist der Netzwerkteil 24 Bit, der Rechnerteil 8 Bit lang. Somit gibt es über 2 Millionen Class-C-Netze. Jedes davon kann aber nur 254 Rechner enthalten.

Alle IP-Nummern, die nicht in eine dieser drei Kategorien passen, sind reserviert und dürfen nicht verwendet werden. Insbesondere sind reserviert:

- 0.*x.y.z*: Das Netz mit der Nummer Null am Anfang wird meist als Synonym für das Default-Netz genommen, also ein Netz, dass nicht in eine andere Beschreibung passt.
- 127.*x.y.z*: Dieses Netz ist das sog. loopback-Netz. Jeder Rechner in einem TCP/IP-Netz kann sich selbst (neben seiner regulären IP-Nummer) auch über die loopback-Nummer 127.0.0.1 ansprechen. Dies gilt selbst dann, wenn der Rechner gar keine Netzwerkkarte eingebaut hat.

- *a.b.c.d* mit *a*≥224: Diese Netze sind für spätere Erweiterungen reserviert.

Innerhalb eines Netzes dürfen die folgenden IP-Nummern nicht für Rechner vergeben werden, da sie eine besondere Bedeutung haben:

- Die IP-Nummer, deren Rechnerteil nur aus (binären) Nullen besteht: Sie gibt die Netzwerkadresse an.
- Die IP-Nummer, deren Rechnerteil nur aus (binären) Einsen besteht: Sie gibt die sog. Broadcast-Adresse an; dies ist eine IP-Nummer, unter der alle Rechner innerhalb eines Netzwerkes zu erreichen sind. Voraussetzung dafür ist aber, dass die verwendete Netzwerktechnologie dies unterstützt. Dies ist beispielsweise bei Ethernet der Fall.

Beispiele für IP-Nummern sind

- **132.17.1.95**: Es handelt sich um eine Class-B-Adresse, also um den Rechner 1.95 im Netz 132.17 (häufig als 132.17.0.0 geschrieben).
- **194.101.76.12**: Eine Class-C-Adresse: Rechner 12 im Netz 194.101.76 (bzw. 194.101.76.0)
- **10.255.240.65**: Eine Class-A-Adresse: Rechner 255.240.65 im Netz 10 (bzw. 10.0.0.0)
- **192.13.7.255:** Dies ist keine Rechnernummer, sondern die Broadcast-Adresse im Class-C-Netz 192.13.7
- **202.12.95.0**: Dies ist ebenfalls keine Rechnernummer, sondern die Netzwerkadresse des Class-C-Netzes 202.12.95
- **176.12.95.0**: Dies ist dagegen wieder eine Rechnernummer, nämlich die des Rechners 95.0 im Class-B-Netz 176.12

Abschliessend noch eine Bemerkung zur Terminologie. In der Literatur werden unterschiedliche, teils auch widersprüchliche Bezeichnungen für IP-Nummern und verwandte Konzepte verwendet. Daher sollen die Begriffe hier noch einmal abgegrenzt werden: IP-Nummern sind 32-Bit-Zahlen, die von IP zur Adressierung verwendet werden. Eine IP-Nummer kann eine Rechnernummer, eine Netzwerknummer oder eine Broadcast-Adresse sein. Jede IP-Nummer besteht aus einem Netzwerkteil und einem

Rechnerteil. Eine Rechnernummer ist also nicht dasselbe wie der Rechnerteil einer IP-Nummer!

3.9 Routing in TCP/IP-Netzen

Routing bedeutet das Zustellen eines IP-Datagrammes an den Rechner, dessen IP-Nummer im Datagrammheader als Ziel angegeben ist. Jeder Rechner im Netzwerk muss Informationen darüber besitzen, auf welche Weise er Datagramme zu allen anderen gültigen Rechnernummern weiterleiten muss, damit sie ihr Ziel erreichen.

Im Allgemeinen sollen, wie bereits erwähnt, Rechner innerhalb eines physikalischen Netzwerkes Rechnernummern mit dem selben Netzwerkteil bekommen. Das bedeutet, dass alle IP-Datagramme direkt an den Empfänger zugestellt werden können, wenn die Ziel-IP-Nummer den selben Netzwerkteil hat wie der eigene Rechner. Unterscheidet sich der Netzwerkteil dagegen, kann das Datagramm nicht direkt zugestellt werden, sondern wird an einen Router weitergegeben, der dem Ziel ein Stückchen näher ist.

Diese Unterscheidung ist nicht unbedingt bindend; es kann auch Ausnahmen geben. Diese sind jedoch betriebssystemabhängig und machen die Vernetzung kompliziert und unübersichtlich. Bei der Planung einer Rechnervernetzung sollte deshalb immer nach obigem Grundsatz vorgegangen werden.

Rechner, die als Router fungieren, müssen daher an mehrere Netze angeschlossen sein und in jedem Netz eine eigene IP-Nummer besitzen. Es hat also nicht jeder Rechner, sondern jeder Netzwerkanschluss in einem Rechner seine eigene IP-Nummer.

Informationen darüber, wie Ziel-IP-Nummern zu erreichen sind, ist in der Routing-Tabelle eines Rechners abgelegt. Abbildung 3.6 zeigt ein Beispiel für eine Netzanordnung, Abbildung 3.7 die dazugehörige Routingtabelle des schraffierten Rechners.

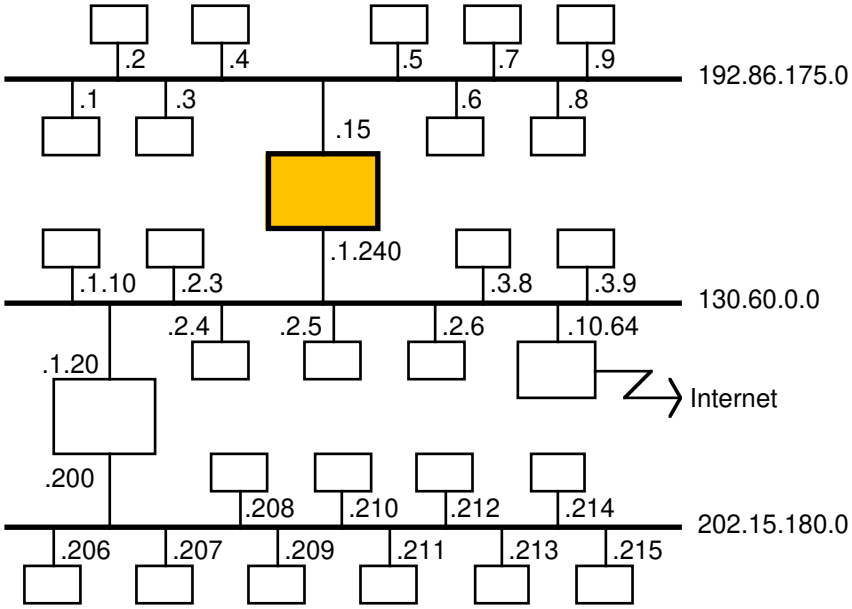

Abb. 3.6: Beispiel für eine TCP/IP-Vernetzung

Ziel	über	Interface
`127.0.0.0`	`direkt`	`lo0`
`192.86.175.0`	`direkt`	`eth0`
`130.60.0.0`	`direkt`	`eth1`
`202.15.180.0`	`130.60.1.20`	`eth1`
`0.0.0.0`	`130.60.10.64`	`eth1`

Abb. 3.7: Routingtabelle eines Routers

Jeder Rechner besitzt zunächst einmal eine Route zu sich selber über das Loopback-Device `lo0`. Der schraffierte Rechner hat zwei Ethernet-Anschlüsse (`eth0` und `eth1`), die mit den Netzen 192.86.175.0 und 130.60.0.0 verbunden sind, und somit auch zwei IP-Nummern. Er kann also alle Rechner in diesen Netzen direkt erreichen. Das Netz 202.15.180.0 ist über den Router 130.60.1.20 zu erreichen, der wiederum lokal erreichbar ist. Alle anderen Netze im Internet sind über den Router 130.60.10.64 zu erreichen.

3.10 Subnetze und Subnetz-Masken

Wie bereits erwähnt, passen in ein Class-A-Netz 16 Millionen Rechner, in ein Class-B-Netz immer noch 65000 Rechner. Es ist im allgemeinen nicht möglich bzw. sinnvoll, so viele Rechner in einem physikalischen Netz unterzubringen. Umgekehrt wäre es allerdings Verschwendung, wenn man z.B. ein Class-B-Netz aus nur 10 Rechnern aufbauen würde.

Die Lösung für dieses Problem besteht in einer weiteren Unterteilung einer gegebenen Netzadresse. Mit anderen Worten, es sollen mehr Bits der IP-Nummer für die Netzadresse und weniger für die Rechnernummer verwendet werden.

Zur Angabe, welche Bits zur Netz- und welche zur Rechneradresse gehören soll, dient die Subnetz-Maske. Sie ist ebenfalls eine 32-Bit-Zahl und wird meist in der Form *a.b.c.d* angegeben, wobei *a,b,c* und *d* 8-Bit-Zahlen sind.

In Binärdarstellung geschrieben, besteht eine Subnetz-Maske immer aus einer Folge von Einsen, gefolgt von einer Folge von Nullen. Ist ein Bit auf eins gesetzt, dann gehört das entsprechende Bit der IP-Nummer zur Netzadresse; ansonsten gehört es zu Rechnernummer.

Beispiele:

- Die IP-Nummer 128.66.12.1 gehört normalerweise zu einem Class-B-Netz; es handelt sich also um Rechner 12.1 im Netz 128.66.0.0.

  Wenn nun die Subnetz-Maske 255.255.255.0 lautet (das ist binär $11111111.11111111.11111111.00000000_2$), dann bedeutet das, dass jetzt die ersten 24 Bit der IP-Nummer die Netzadresse und die letzen 8 Bit die Rechnernummer sind. Also steht obige IP-Nummer für den Rechner 1 im Subnetz 128.66.12.0.

- Schwieriger wird es, wenn der Übergang zwischen Netz- und Rechnernummer nicht mehr an einer Bytegrenze stattfindet:

  Gegeben sei die IP-Nummer 130.97.16.132 mit der Subnetz-Maske 255.255.255.192. Hier empfiehlt es sich, alle beteiligten Nummern in Binärdarstellung aufzuschreiben:

  SN-Maske: 255.255.255.192 $11111111.11111111.11111111.11000000_2$

IP-Nr.: 130.97.16.132 $10000010.01100001.00010000.10000100_2$

Die Netzadresse berechnet sich, indem alle Bits der Rechnernummer auf Null gesetzt werden: $10000010.01100001.00010000.10000000_2$

also 130.97.16.128

Die Rechnernummer kann man ablesen, wenn man alle Bits der Netzwerknummer auf Null setzt: $00000000.00000000.00000000.00000100_2$

also handelt es sich um Rechner 4 im Netz 130.97.16.128.

Auf ähnliche Weise kann man noch die Broadcast-Adresse bestimmen: Man setzt dazu alle Bits der Rechneradresse auf Eins:

$10000010.01100001.00010000.10111111_2$

Also lautet die Broadcast-Adresse 130.97.16.191.

- Noch ein anderes Beispiel: IP-Nr. 132.90.132.5, SN-Maske: 255.255.240.0

  SN-Maske: 255.255.240.0 $11111111.11111111.11110000.00000000_2$

  (also 20 Bits Netznummer, 12 Bits Rechnernummer)

  IP-Nr.: 132.90.132.5 $10000100.01011010.10000100.00000101_2$

  Netzadresse: $10000100.01011010.10000000.00000000_2$

  also 132.90.128.0;

  Rechnernummer: $00000000.00000000.00000100.00000101_2$

  also 4.5.

3.11 Zukünftige Entwicklungen

Weltweit sind derzeit fast alle Class-A-Netze, ca. 50 Prozent aller Class-B-Netze und 5 Prozent aller Class-C-Netze vergeben. Wenn das Internet weiterhin so stark anwächst wie zur Zeit, dann wird es keine zwei Jahre mehr dauern, bis alle Netznummern vergeben sind. Schon heute werden einzelnen Organisationen mehrere Class-C-Netze zugewiesen, da ein Class-B-Netz zu gross wäre. Jedes Class-C-Netz benötigt einen Eintrag in den

Routingtabellen der Router weltweit. Zentrale Router haben derzeit etwa 30.000 Routen gespeichert.

Eine Erweiterung des Adressraumes ist also dringend nötig und zur Zeit auch in Arbeit: In **IP Next Generation (IPng),** der Version 6 des Internet-Protokolles [Deering 96], werden IP-Nummern 128 Bit lang sein. Das bedeutet, dass man jedem Quadratmeter der Erde über 10^{23} IP-Nummern zuweisen könnte. Es ist sehr wahrscheinlich, dass diese Länge auf absehbare Zeit ausreichend sein wird [Tanenbaum 96]. Mit dem grossen Adressraum wird es möglich, Netze streng hierarchisch nach organisatorischen und geographischen Gesichtspunkten zu vergeben. Beispielsweise werden alle IP-Nummern in den USA in den höchstwertigsten 8 Bit übereinstimmen. Ein Router in Europa erkennt also eine solche Adresse sofort und braucht für den Transport über den Atlantik nur noch eine oder wenige Routen zu kennen.

Neben der Erweiterung des Adressraumes bietet IPng noch andere Neuerungen, beispielsweise kryptographische Methoden zur Gewährleistung von Authentizität und Vertraulichkeit, Massnahmen für schnelleres Routing und eine verbesserte Behandlung überlanger IP-Datagramme.

3.12 Sicherheitsaspekte der Schichten I bis III

Vertraulichkeit der transportierten Daten

Ueblicherweise werden die Daten auf den Schichten I-III unverschlüsselt übertragen. Das bedeutet, dass jeder, der Zugriff auf das physikalische Uebertragungsmedium oder einen der beteiligten Router hat, die Daten abhören kann.

Je nach Uebertragungsmedium ist dies jedoch unterschiedlich schwierig. Eine Modemstrecke über eine Telefonleitung abzuhören, gilt als sehr schwer, insbesondere dann, wenn der Angreifer die Datenübertragung selbst nicht beeinflussen möchte oder kann. Die Kommunikation über Analogmodems erfolgt verbindungsorientiert; zu Beginn einer Verbindung werden mehrere Verbindungsparameter ausgehandelt, die sich jedoch im Laufe der Uebertragung ändern können. Wenn sich der Angreifer erst später zuschaltet, hat er es sehr schwer, diese Verbindungsdaten zu rekonstruieren. Bekommt ein empfangendes Modem einzelne Teile der Daten nicht mit (z.B. wegen einer kurzen Leitungsstörung), fordert es diese Daten neu an. Der Angreifer kann dagegen nicht so

einfach diese Daten neu anfordern. Ein Angriff, der auf blosem Mithören beruht, wird **passiver Angriff** genannt.

Wenn der Angreifer die Leitung nicht nur abhören, sondern auch beeinflussen oder unterbrechen kann, hat er es einfacher. Er kann dann gezielt Störungen auf die Leitung geben, so dass die Modems gezwungen werden, die Verbindungsdaten neu auszuhandeln. Auf diese Weise bekommt er die fehlenden Informationen nachgeliefert. Noch einfacher geht es, wenn er die Verbindung ganz unterbricht und sich dazwischenschaltet. Er gibt sich dann beiden Teilnehmern gegenüber als der jeweils andere aus und liest die Daten mit, während er sie weiterleitet. Ein Angriff, der auf einen Eingriff in die Datenübertragung oder den Protokollablauf beruht, nennt man **aktiven Angriff**. Er ist im Allgemeinen effektiver als ein passiver Angriff, kann aber auch leichter bemerkt werden.

Im Gegensatz zu einer Modemverbindung ist es sehr einfach, einen Ethernet-Strang oder ein Token-Ring-Netzwerk abzuhören. Frames werden auf diesen Medien in kleinen Blöcken übertragen; Blockanfang und -Ende sind klar definiert, und auszuhandelnde Verbindungsparameter gibt es nicht. Zum reihenfolgetreuen Zusammensetzen der einzelnen Blöcke werden natürlich Kenntnisse der Protokolle auf den höheren Schichten benötigt.

Integrität der Daten

Auf den unteren drei Schichten der TCP/IP Protokollhierarchie werden Daten gegen Uebertragungsfehler geschützt, indem jeweils Prüfsummen mitübertragen werden. TCP und UDP sichern Daten und Schicht-III-Header, IP den Schicht-II-Header; je nach Protokoll und Uebertragungstechnik werden auf Schicht I ganze Frames, nur Header oder auch überhaupt nichts gesichert.

Es ist möglich, aber unwahrscheinlich, dass Daten durch zufällige Fehler verfälscht wurden, die Prüfsumme aber dennoch korrekt ist. Die Algorithmen sind so gewählt, dass typische Fehler sich möglichst nicht gegenseitig aufheben. Gegen absichtliche Verfälschungen besteht jedoch kein Schutz; die Prüfsummenverfahren sind öffentlich bekannt, so dass ein Angreifer, der Zugriff auf die physikalische Verbindung oder einen der Router hat, die Daten verändern und die Prüfsumme neu berechnen kann.

Authentizität

Auf IP-Ebene ist es prinzipiell kein Problem, eine falsche Absender-IP-Nummer in die Datagramme einzutragen (**IP Spoofing**). Der Weitertransport erfolgt bei den gängigen Implementierungen ausschliesslich anhand der Ziel-IP-Nummer. Transport anhand Absender- und Zieladresse (und eventuell noch Protokollkennung, Portnummern und sonstiger Optionen) gibt es zwar für manche Betriebssysteme auch (sog. **Policy-Based Routing**), das ist aber aufwendig zu konfigurieren und kostet CPU-Zeit. Daher wird es selten eingesetzt.

Das blosse Fälschen der Absender-IP-Nummer bringt jedoch keinen direkten Gewinn, abgesehen davon, dass Angriffe schwerer rückverfolgbar sind. Voraussetzung dafür, dass man auf diese Weise wirklich an mehr Privilegien kommen kann, sind Dienste auf höherer Ebene, die bestimmten IP-Nummern mehr Privilegien einräumen als anderen, oder anders ausgedrückt, die eine Authentifizierung nach IP-Nummern vornehmen. Leider gibt es solche Dienste tatsächlich: Die Berkeley r-Tools (siehe Abschnitt 4.5) und der Line Printer Daemon (Abschnitt 5.5) sind bekannte Beispiele.

Für den Angreifer taucht jedoch hier noch ein zweites Problem auf. Die oben genannten Dienste arbeiten mit TCP-Verbindungen, benötigen also alleine zum Aufbau einer Verbindung bereits Pakettransport in beide Richtungen. Ist aber die Absenderadresse des ersten Paketes gefälscht, so wird das Antwortpaket an die vermeintliche IP-Nummer des Kommunikationspartners geschickt. Für den Angreifer ist es jedoch essentiell wichtig, das Antwortpaket zu bekommen. Denn es enthält die initiale Sequenznummer der Rückrichtung, ohne deren Bestätigung keine Verbindung zustande kommen kann. Er kann das Paket nur abhören, wenn er entweder Zugriff auf einen Router hat, über den diese Antworten transportiert werden (also sich hinreichend nahe an dem anzugreifenden Rechner befindet und somit leicht zu entdecken ist), oder indem er sich eines Tricks bedient: Dem **Source-Routing.** Dies ist eine spezielle IP-Option, mit der der Absender eines Datagrammes explizit die IP-Nummern von Routern angeben kann, über die das Datagramm laufen soll. Die Antwort wird ebenfalls über diese Knoten geroutet und somit dem Angreifer zugestellt.

Die Source-Routing-Option dient der Fehlersuche in IP-Netzen und sollte eigentlich nicht für TCP-Datentransport benutzbar ein. Bei den meisten kommerziellen Routern kann sie abgeschaltet werden, so dass TCP-Pakete mit eingeschaltetem Source-Routing einfach

verworfen werden. Leider machen noch nicht alle Organisationen an ihren Eingangsroutern von dieser Möglichkeit Gebrauch.

Falls Source Routing nicht möglich ist, z.B. weil es von einem Router nicht unterstützt wird oder im TCP/IP-Code des anzugreifenden Rechners nicht implementiert ist, kann durch ICMP-*redirect*-Meldungen dafür gesorgt werden, dass Antwortpakete an gefälschte IP-Nummern zum Absender zurückgelangen. Eine redirect-Meldung veranlasst einen Rechner oder Router dazu, eine Route in seiner Routingtabelle zu ändern. Daher sollten solche Meldungen nur von vertrauenswürdigen Routern akzeptiert werden, z.B. den Routern des Providers, was aber wegen der Fälschbarkeit von Absender-IP-Nummern schwierig zu realisieren ist.

Für den Fall, dass der Angreifer keine Möglichkeit hat, die Antworten zu erhalten, gibt es noch einen weiteren Trick, die richtige Sequenznummer zu bekommen: **Sequence Number Guessing** [Bellovin 89], Erraten der Sequenznummern. Manche Betriebssysteme vergeben ihre initialen TCP-Sequenznummern nach einem leicht nachvollziehbaren Schema, zum Beispiel in sequentieller Reihenfolge. In diesem Fall braucht er also lediglich vorher eine "harmlose" TCP-Verbindung an den anzugreifenden Rechner zu eröffnen, zum Beispiel ein *finger*, und sich die verwendeten Sequenznummern zu merken. Wenn dieser Rechner zwischenzeitlich keine andere Verbindungsanforderung bedient hat, dann ist die initiale Sequenznummer der Angriffsverbindung bekannt, auch wenn das Antwortpaket nicht empfangen werden kann.

Bei Protokollen, die UDP verwenden, existiert für den Angreifer das Problem mit den Sequenznummern nicht (ausser, wenn eine höhere Schicht einen ähnlichen Mechanismus benutzt). Protokolle, bei denen ein einzelnes UDP-Paket von einer "vertrauenswürdigen" IP-Nummer eine sicherheitsrelevante Aktion auslösen kann, sind somit prinzipiell zum Scheitern verurteilt.

Die Fälschbarkeit von IP-Nummern sollte dazu führen, dass IP-Nummern niemals zu Authentisierungszwecken benutzt werden sollen. Dienste, die bestimmten Benutzern oder Rechnern mehr Privilegien als anderen gewähren, sollten statt dessen auf einer höheren Ebene kryptographische Authentifizierungsverfahren benutzen. Zukünftige Erweiterungen der TCP/IP-Protokollfamilie sehen kryptographische Verfahren auch auf Schicht II und III vor (siehe Kap. 3.11). Standardisierung und Implementierung dieser Verfahren stehen jedoch in absehbarer Zeit noch nicht zur Verfügung.

Verlust der Verfügbarkeit von Rechnern

Die gängigen Implementierungen von TCP/IP bieten einige Ansatzpunkte für Angreifer, Rechner, Netze oder Router im Betrieb zu stören. Die meisten dieser Angriffe basieren auf der Ausnutzung von Fehlern in der Implementierung des TCP/IP-Systems, und es hängt vom System ab, ob eine bestimmte Sicherheitslücke vorhanden ist. Häufig zeigt sich jedoch, dass verschiedene Entwickler die gleichen Fehler machen, obwohl sie unabhängig voneinander arbeiten oder zumindest vorgeben, dies zu tun. Die nachfolgende Auflistung erhebt bei weitem keinen Anspruch auf Vollständigkeit, sondern liefert einige Beispiele für diese Form von Attacken.

Ein altbekannter Angriff sind sogenannte **SYN-Floods**. Dabei werden einem einzelnen Rechner innerhalb kurzer Zeit viele TCP-Pakete mit gesetzter SYN-Kennung geschickt; das sind Pakete, mit denen das Oeffnen einer neuen TCP-Verbindung eingeleitet wird. Der Rechner beantwortet diese Pakete alle mit einem SYN ACK und merkt sich den Vorfall, damit später das erwartete ACK-Paket richtig zugeordnet werden kann (siehe Abschnitt 3.6). Erst wenn diese nach Ablauf einer festgelegten Zeitspanne immer noch nicht eingetroffen sind, werden die gespeicherten Informationen wieder gelöscht. Viele TCP/IP-Implementierungen haben jedoch Schwierigkeiten, wenn mehrere tausend Verbindungsanforderungen gespeichert werden müssen. Es können dann keine sinnvollen Verbindungen mehr aufgebaut werden, weil Puffer überlaufen, oder der Ueberlauf kann sogar zum Absturz führen, weil dieser Fall bei der Implementierung nicht bedacht oder getestet wurde. Tragen diese Pakete gefälschte Absender-IP-Nummern, so sind sie auch sehr schwer von "ernsthaften" Verbindungsanforderungen zu unterscheiden. SYN-Floods sind in der letzten Zeit wieder in die öffentliche Diskussion geraten [CERT 96-21], weil im Usenet ein Programm veröffentlicht wurde, das diesen Angriff tatsächlich ausführt.

Eine andere Angriffsvariante ist das **Verschicken überlanger ping-Datenpakete** [CERT 96-26]. Sie basiert auf einem Fehler in der Implementierung der IP-Schicht vieler Betriebssysteme. Bekommt ein Rechner ein Datagramm mit einer Länge von über 64 Kilobytes, dann läuft ein interner Puffer über; es können wichtige Variablen des Betriebssystems überschrieben und das System zum Absturz gebracht werden. Obwohl dies ein reiner Implementierungsfehler ist, sind viele verschiedene Systeme davon betroffen, nicht nur Rechnerbetriebssysteme, sondern auch Router, Netzwerkdrucker und ähnliches. Diese Lücke existiert schon seit Jahrzehnten. Da IP-Datagramme über 64 Kilobytes verboten sind und von den meisten Betriebssystemen gar nicht verschickt

werden können, ist sie aber lange Zeit nicht aufgefallen. Erst die Implementierungen des *ping*-Programmes unter Windows 95 und Windows NT ermöglichten ein Verschicken solch langer Pakete.

Das Verschicken von **irreführenden Fehlermeldungen** [Bellovin 89] bezüglich der Erreichbarkeit von Rechnern ist eine weitere Angriffsvariante auf die Verfügbarkeit von Datenverarbeitungseinrichtungen. Mit einer speziellen ICMP-Meldung, *host unreachable*, zeigt ein IP-Router an, dass er für IP-Datagramme an eine bestimmte IP-Nummer keine Route kennt. Daraufhin löscht der absendende Rechner die entsprechende Route ebenfalls aus seiner Routingtabelle. Durch gezieltes Verschicken solcher Meldungen können fehlerhafte Informationen in Routingtabellen erzeugt werden, die zur Unerreichbarkeit anderer Rechner oder Netze führen können.

4 Wichtige Protokolle der Anwendungsebene und ihre Sicherheitseingenschaften

4.1 Telnet

Telnet (Teletype Network) dient zum Einloggen auf einem Rechner A von einem anderen Rechner B aus. Telnet ist, wie die meisten der hier besprochenen Dienste, eine **Client-Server-Applikation**: Auf A muss ein Telnet-Server (*telnet daemon*, *telnetd*) laufen, der einkommende Verbindungen entgegennimmt, auf B muss dem Benutzer ein Telnet-Client zur Verfügung stehen. Ein telnetd ist nur auf Systemen, die ein Benutzer- und Loginkonzept haben, sinnvoll, zum Beispiel Unix, VMS oder MVS. Clients gibt es für jedes gängige Betriebssystem.

Telnet arbeitet über TCP. Der Telnet-Server wird auf dem Rechner A über den Port 23 angesprochen. Das bedeutet, wenn auf A eine TCP-Verbindungsanforderung mit dem Zielport 23 ankommt, wird die TCP-Implementierung die einkommenden Daten an den telnetd weiterleiten. In TCP/IP-Terminologie heisst es, der telnetd **lauscht** auf Port 23. Die Portnummer auf der Absenderseite ist beliebig; der Absender muss nur sicherstellen, dass nicht mehrere Verbindungen mit gleichen Verbindungsparametern gleichzeitig benutzt werden.

Sobald die Verbindung aufgebaut ist, bietet der Server die übliche Loginprozedur über die Telnet-Verbindung an. Sie besteht im Allgemeinen aus einer Eingabeaufforderung für Benutzernamen und Passwort; stimmt beides mit den dem Server bekannten Informationen überein, ist der Benutzer eingeloggt.

Drei Dinge sind dabei sicherheitsrelevant:

- Der Telnet-Server auf Rechner A läuft mit relativ hohen Privilegien. Dies ist nötig, da er nach erfolgreicher Authentifizierung auf die Rechte umschalten kann, die der einloggende Benutzer besitzt. Der telnetd muss also mindestens die Vereinigung der Rechte besitzen, die alle von aussen erreichbaren Benutzeraccounts besitzen.

  Unter Unix läuft der telnetd unter der Benutzerkennung des Superusers, `root`. Nur dieser Benutzer hat die Möglichkeit, auf jede andere Benutzerkennung umzuschalten. Probleme ergeben sich, wenn der telnetd Programmierfehler hat. Denn nichts hindert ihn daran, beliebige Teile des Dateisystems oder des Speichers auszulesen oder zu verändern oder andere Programme auszuführen, die dann ebenfalls mit sehr vielen Rechten ablaufen.

- Auch ohne ein Passwort abgefangen zu haben, kann man mit telnet unter Umständen unerlaubten Zugriff auf fremde Rechner bekommen: durch das Raten von Passwörtern. Die Benutzernamen sind in der Regel öffentlich bekannt; sie kommen in Mailadressen vor, werden in die News gepostet oder auf WWW-Seiten veröffentlicht. Mit etwas Phantasie kann man Passwörter häufig erraten. Viele Benutzer verwenden Namen von Freunden, geographische Bezeichnungen oder sonstige Begriffe, die sie sich leicht merken können. Wörterbücher mit gängigen Passwörtern sind frei verfügbar, Programme zum Durchtesten über eine Telnet-Verbindung ebenfalls.

  Abhilfe können die Benutzer selber schaffen, indem sie Passwörter verwenden, die nicht so leicht zu erraten sind. Das Bewusstsein dafür ist allerdings kaum vorhanden. Besser ist es, wenn der Systemadministrator Software installiert, die darauf achtet, dass die Benutzer nur Passwörter wählen können, die hinreichend "kryptisch" sind, zum Beispiel weil sie Ziffern, gemischt Gross-/Kleinschreibung oder Sonderzeichen enthalten. Ausserdem sollte der telnetd die Eigenschaft haben, bei Fehleingaben von Benutzername oder Passwort etwas Zeit verstreichen zu lassen, bevor ein neuer Loginversuch gemacht werden kann. Zusätzlich sollte die maximale Zahl der gleichzeitig einkommenden Telnet-Verbindungen limitiert werden. Software, die dies leistet, ist für die meisten Unix-Varianten verfügbar, für andere Systeme allerdings weniger.

- Das grösste Sicherheitsproblem bei Telnet besteht allerdings darin, dass die Daten, die bei einer Telnet-Sitzung eingetippt werden oder auf dem Bildschirm erscheinen, unverschlüsselt über die TCP-Verbindung gehen. Insbesondere wird auch das

Passwort beim Einloggen unverschlüsselt übertragen. Jeder, der Zugriff auf die physikalische Leitung oder einen der Router hat, kann es mit wenig Mühe auslesen und dann selber damit einloggen. Aus diesem Grund sollte Telnet, wenn überhaupt, nur innerhalb eines gesicherten Bereiches, in dem sich Benutzer und Administratoren gegenseitig vertrauen, benutzt werden. Telnet von aussen auf die Rechner der eigenen Organisation sollte nicht zugelassen werden. Leider ist die Realität davon weit entfernt. Das liegt auch daran, dass die Alternativen teilweise umständlich zu bedienen und nicht für alle Plattformen verfügbar sind.

4.2 FTP

Das File Transfer Protocol (FTP) dient zum Uebertragen ganzer Dateien über das Netz. Das Grundprinzip ist ähnlich wie bei Telnet: Der Benutzer braucht einen FTP-Client, auf der Gegenseite läuft ein FTP-Server (auf Port 23). Der Client öffnet eine TCP-Verbindung, der Server führt eine Authentifizierung durch und erlaubt dann das Ablegen oder Herunterladen von Dateien.

Eine Besonderheit besteht darin, dass der Datei-Transport meist über getrennte TCP-Verbindungen abgewickelt wird; die ursprüngliche Verbindung wird nur zum Transport von Kommandos benutzt. Das ermöglicht einige zusätzliche Angriffsvarianten, die aber auch eine Abhörmöglichkeit der Leitung vorraussetzen und somit auch einfacher zu realisieren sind. Die Sicherheitsprobleme sind prinzipiell die gleichen wie bei Telnet. Erschwerend kommt aber hinzu, dass in den gängigen FTP-Server-Programmen relativ häufig sicherheitsrelevante Programmierfehler gefunden werden. Das liegt daran, dass FTP-Server viele Funktionen enthalten und damit komplexe und umfangreiche Programme sind.

4.3 Electronic Mail

Das Simple Mail Transfer Protocol (SMTP) ist das Protokoll, mit dem E-Mail im Internet transportiert wird. Es basiert auf TCP und liesse sich im Prinzip ebenfalls nach dem Client-Server-Prinzip betreiben, dass also der Absender eine Mail mit seinem Client direkt eine Verbindung zu einem SMTP-Server auf dem Rechner des Mailempfängers eröffnet. Dies ist jedoch aus verschiedenen Gründen nicht üblich: Erstens soll E-Mail häufig auf Rechnern empfangen werden, die an Arbeitsplätzen stehen und nicht ständig eingeschaltet

sind oder keine permanente Verbindung zum Internet haben, zweitens kennt der Absender einer Mail im Allgemeinen auch nicht die IP-Nummer oder den Rechnernamen der Maschine, auf die eine Mail ausgeliefert werden soll.

Die Architektur eines E-Mail-Systems besteht daher meistens aus einer Unterteilung in Mail User Agents (MUA) und Mail Transport Agents (MTA). Abbildung 4.1 zeigt eine mögliche Anordnung.

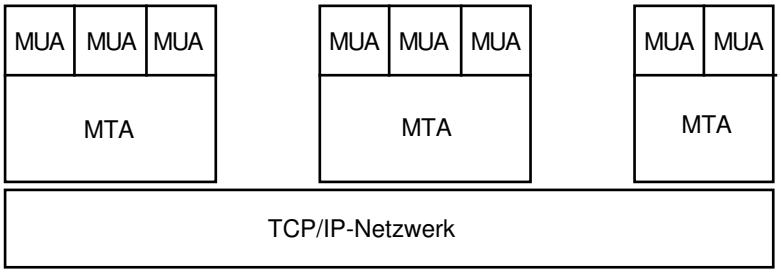

Abb. 4.1: Aufbau von E-Mail-Systemen im Internet

Die einzelnen MTA's kommunizieren über das SMTP-Protokoll über TCP miteinander. Dabei baut jeweils derjenige MTA die Verbindung auf, der eine Mail versenden möchte.

Der MTA und die MUA's können auf dem selben oder auf verschiedenen Rechnern laufen; die Kommunikation zwischen ihnen erfolgt entweder über das lokale Dateisystem eines Rechners oder Rechnerverbundes, oder nach dem Client-Server-Prinzip über eine TCP-Verbindung. Der MUA ist dabei der Client, er baut die Verbindungen auf und holt Mail ab bzw. übergibt Mail an den MTA. Gängige Protokolle zum Abholen von Mail sind das **Post Office Protocol** (POP) oder das **Internet Mail Agent Protocol** (IMAP). Zum Versenden von Mail wird meist SMTP benutzt.

Die Zuordnung zwischen MTA und MUA's kann technisch auf vielfältige Weise gemacht werden. Da die Kommunikation über TCP erfolgen kann, können MUA und MTA beliebig weit im Netz entfernt sein. Die Planung erfolgt meist nach organisatorischen oder Netzlast-Gesichtspunkten. Kleinere Organisationen können möglicherweise nur einen MTA verwenden, der für die gesamte E-Mail zuständig ist; in grösseren Unternehmen kann ein MTA pro Netzstrang, Gebäude, Stockwerk oder Abteilung Verwendung finden.

Der Einsatz von E-Mail ist nicht zwingend mit einer Internet-Anbindung verbunden. E-Mail ist sogar wesentlich älter als das Internet und wird sowohl in anderen Online-Transportsystemen (Bitnet, IPX) als auch in Offline-Systemen (UUCP, FTN) verwendet. Dank spezieller Gateways ist es möglich, dass Benutzer verschiedener Netze per E-Mail miteinander kommunizieren. Ein solches Gateway wird durch einen MTA realisiert, der das Versenden von Mail über (mindestens) zwei verschiedene Transportsysteme beherrscht.

E-Mail ist mit einer Reihe von Sicherheitsproblemen verbunden.

- Die Protokolle SMTP, POP und IMAP sehen keine Verschlüsselung vor. Damit sind Verletzungen der Vertraulichkeit und der Integrität einer Mail durch jeden möglich, der Zugriff auf das physikalische Uebertragungsmedium oder einen Netzknoten besitzt. Dieses Problem lässt sich durch den Einsatz kryptographischer Verfahren auf höherer Ebene lösen. Implementierungen dazu gibt es (siehe Kap. 4.9), jedoch sind diese teilweise komplex in der Handhabung, und es gibt ungeklärte Rechtsfragen.

- Die Protokolle POP und IMAP ermöglichen das Abholen bzw. Manipulieren der Mailbox eines Benutzers. Dazu authentisiert sich der Benutzer mit einem Passwort. Dieses Passwort wird, wie bei Telnet oder FTP, unverschlüsselt übertragen.

- Ein grosses Problem ist die Fälschbarkeit der Absenderadresse. Die Mailadresse des Absenders lässt sich, analog zu Absender-IP-Nummern, beliebig fälschen; sie hat keine Funktion für den Mailtransport. Manche MTA's protokollieren die IP-Nummer des Rechners, von dem eine Mail eingeliefert wurde, die meisten tun dies jedoch nicht. Damit ist Mail nicht nur ohne kryptographische Absicherung für verbindliche Absprachen oder geschäftliche Transaktionen ungeeignet, sondern birgt auch die Gefahr, dass die beiden nachfolgend beschriebenen Angriffsvarianten völlig anonym durchgeführt werden können.

- Unter einer **Mailbombe** versteht man eine sehr grosse Mail, deren Zweck es ist, die Leitungen auf der Empfängerseite zu blockieren oder die Mailplatte des Empfängers zum Ueberlaufen zu bringen und somit die Verfügbarkeit des E-Mail-Dienstes oder eines ganzen Rechners auf der Empfängerseite zu beeinträchtigen. Manche MTA's bieten einen Schutz vor solchen Mails, indem sie Mails oberhalb einer bestimmten Maximalgrösse nicht annehmen. Jedoch ist das Ziel einer Mailbombe auch mit vielen kleinen Mails zu erreichen.

- Die gängigen Implementierungen der Mail Transport Agents sind sehr komplizierte und umfangreiche Programme. Daher werden relativ häufig Programmierfehler gefunden, die sicherheitskritische Folgen haben. Beispielsweise ist es mit älteren Versionen von *Sendmail* unter Unix möglich, per Mail ferngesteuert Programme auszuführen. Man musste einfach die Empfängeradresse mit einem "|" beginnen lassen; der nachfolgende Text wurde dann als Unix-Kommando interpretiert und ausgeführt. Speziell bei *Sendmail* stehen meist sofort nach Bekanntwerden eines Fehlers neue Versionen zur Verfügung, bei denen das Problem beseitigt ist. Allerdings installieren nicht alle Administratoren die neuen Programmversionen sofort, so dass man noch lange Zeit verwundbare Mailserver finden kann.

4.4 Das World Wide Web

Das World Wide Web (WWW) ist ein System zum einfachen Abrufen und Darstellen von Dokumenten und Dateien in TCP/IP-Netzen. Im Prinzip ist es nichts weiter als die Möglichkeit, Dateien in mehreren standardisierten Formaten (Textdokumente, Bilder, Audio, Video) über verschiedene, teils altbekannte Protokolle der TCP/IP-Protokollfamilie zu übertragen und mit einem einzigen Programm, dem Web-Client oder WWW-Browser genannt, darstellen zu können.

Das zentrale Dateiformat ist die Hypertext Markup Language (HTML). HTML-Dokumente sind Textdateien mit Steueranweisungen (**Tags**). Der Text wird vom Client als Fliesstext betrachtet und entsprechend den Eigenschaften des Ausgabemediums (Bildschirmgrösse, Druckformat, Farbanzahl) und den Wünschen des Benutzers (Zeichensatz, Fenstergrösse) entsprechend dargestellt. Die Tags dienen dazu, Strukturelemente wie Ueberschriften, Listen oder Trennlinien in den Text einzubringen, Grafiken nachzuladen und einzelne Seitenelemente als Verweise (**Links**) auf andere Dokumente zu kennzeichnen. Daneben gibt es noch Tags, mit denen sich Eingabemasken realisieren lassen.

Ein im Web verfügbares Dokument wird über eine **Universal Resource Locater** (**URL**) angesprochen. Eine URL besteht aus einer Angabe zum Uebertragungsprotokoll, der Quelle, über die das Dokument zu beziehen ist, sowie optional einer Reihe von Parametern. Die Uebergabe von Parametern ist nur bei bestimmten Protokollen möglich und ermöglicht eine Interaktion des Benutzers mit dem Server.

HTML-Dokumente werden meist im Hypertext Transmission Protocol (HTTP), dem zentralen Protokoll des World Wide Web, übertragen. Die URL eines über HTTP verfügbaren Dokumentes kann zum Beispiel lauten:

```
http://www.ifi.unizh.ch/events/dexa.html
```

Dabei bezeichnet `http` den Protokolltyp und `www.ifi.unizh.ch` den DNS-Namen des Rechners, der das Dokument anbietet. `dexa.html` ist der Name des Dokumentes, `events/` das Verzeichnis, in dem das Dokument auf dem Server liegt. Auf dem genannten Rechner muss ein HTTP-Server (httpd) laufen. Die Portnummer für HTTP ist üblicherweise 80. Wegen der zentralen Bedeutung von HTTP für das WWW wird ein httpd häufig auch als WWW-Server bezeichnet.

Andere Uebertragungsprotokolle, die über einen WWW-Browser genutzt werden können, sind das oben beschriebene FTP, *gohper* (das Protokoll eines älteren, WWW-ähnlichen Informationssystems) oder **S-HTTP** (secure HTTP, ein HTTP-ähnliches Uebertragungsprotokoll mit verbesserten Sicherheitseigenschaften).

Durch das reine Herunterladen von öffentlich verfügbaren Dokumenten kann für den Betreiber des WWW-Servers kaum ein Schaden entstehen. Daher bietet das WWW vergleichsweise wenig Sicherheitsprobleme. Ganz ungefährlich ist es dennoch nicht:

- Durch die Uebergabe von Parametern beim Abruf eines Dokumentes hat der Benutzer die Möglichkeit, mit dem Server zu interagieren. Dies geschieht in den meisten Fällen durch Ausfüllen und Absenden eines HTML-Formulares. Die Information wird dabei unverschlüsselt übertragen; manche WWW-Browser ermahnen den Benutzer daher vor dem Durchführen einer solchen Transaktion, keine sensitiven Daten wie Passwörter oder Kreditkartennummern zu übertragen. Erweiterungen von HTTP (z.B. S-HTTP) oder für das WWW geeignete Verschlüsselungsprotokolle auf niedriger Netzwerkebene (z.B**. Secure Socket Layer, SSL**) beheben dieses Problem teilweise.

- Viele HTTP-Server erlauben die Angabe von URL's, die nicht direkt ein Dokument zurückliefern, sondern ein Programm auf dem Server ausführen. Zum Beispiel sind die oben genannten URL's mit Parametern meist von dieser Art. Die Kommunikation zwischen dem Server und einem solchen Programm erfolgt nach dem **Common Gateway Interface** (**CGI**) Standard, weshalb die Programme CGI-Programme oder CGI-Skripten genannt werden. Ein CGI-Programm dient dazu, die Parameter

auszuwerten und in Abhängigkeit ein neues Dokument zu erzeugen, das an den Client übertragen wird. Es kann jedoch auch Nebeneffekte haben, beispielsweise die Ablage der Parameter in einer Datenbank oder das Absenden einer Mail an den Serververwalter. Die Sicherheitsproblematik besteht hier bei der Programmierung der CGI-Programme. Da der Benutzer durch die Wahl der Parameter Einfluss auf die Programmausführung hat, muss das Programm alle möglichen Fälle berücksichtigen, um sich auch bei unsinnigen oder fehlerhaften Parametern noch definiert zu verhalten. Dies wird häufig bei der CGI-Programmierung vernachlässigt. Einfache CGI-Programme lassen sich mit geeigneten Skriptsprachen schnell und kurz erstellen, so dass Sicherheitsaspekte dabei leicht zu kurz kommen. Manche HTTP-Server wurden sogar zeitweise mit CGI-Programmen ausgeliefert, die Sicherheitslücken enthielten.

- Eine jüngere Entwicklung im WWW-Bereich ist ein Dokumentenformat, das nicht Text oder Grafik enthält, sondern Programmcode, der vom Browser ausgeführt wird. Die bekannteste Programmiersprache dieser Gattung ist **Java** von Sun Microsystems. Die Entwickler von Java waren sich durchaus dessen bewusst, dass es eine grosse Sicherheitslücke sein kann, Programme automatisch auszuführen, die über unsichere Quellen aus dem Netz bezogen wird. Daher ist Java so konzipiert, dass solche Programme in einer abgeschotteten Umgebung ablaufen und keinen Schaden am System anrichten können. Zum einen ist dies jedoch nur teilweise gelungen, zum anderen sind auch bei der Implementierung der gängigen Java-Interpreter Fehler gemacht worden, so dass Java derzeit alles andere als sicher ist. Die Ausführung von Java-Netz-Programmen, sog. Applets, kann bei den gängigen Web-Clients unterbunden werden. Da jedoch immer mehr Dienstangebote im WWW einen Java-Interpreter voraussetzen, ist dies keine zufriedenstellende Lösung.

- Eine (nicht standardisierte) Eigenschaft zweier kommerzieller Webbrowser, dem Netscape Navigator und dem Explorer von Microsoft, ist die Möglichkeit, Programmcode ausführen zu können, der direkt im HTML-Dokument steht. Die Programmiersprache wird **Javascript** genannt, was eine irreführende Bezeichnung ist, denn abgesehen von der Idee des mobilen Codes hat diese Sprache nichts mit Java zu tun. Javascript beinhaltet nicht nur viele Sicherheitslücken (u.A. die Möglichkeit, unbemerkt beliebige Dateien des lokalen Rechners per Mail an eine angegebene Adresse zu schicken), sondern auch datenschutzrechtlich umstrittene Funktionen wie das Aufzeichnen der Mausbewegungen eines Benutzers. Bei neueren Versionen der genannten Clients lässt sich diese Funktion abschalten.

4.5 Die Berkeley r-Tools

Die Kommandos **rlogin** (remote login), **rsh** (remote shell) und **rcp** (remote copy) stammen von der Universität Berkeley und waren ursprünglich Bestandteil von BSD.

rlogin ermöglicht, ähnlich wie Telnet, das Einloggen auf einem anderen Rechner im Netzwerk. Im Unterschied zu Telnet kann dieses Einloggen jedoch auch ohne Passwortschutz erfolgen, wenn der Zielrechner dem Ursprungsrechner vertraut. Die "Authentifizierung" des Rechners erfolgt dabei über IP-Nummern. Wie in Abschnitt 2.11 erwähnt wurde, lässt sich dieser Schutz mit Hilfe von IP Spoofing und Sequence Number Guessing oder Source Routing umgehen. Diese Form des Vertrauens lässt sich zwar global für das System abschalten. Jedoch können einzelne Benutzer ihre Accounts für bestimmte Benutzerkennungen auf anderen Rechnern öffnen. Dies zu unterbinden, ist nicht ohne Manipulation des *rlogin*-Daemons möglich.

rsh ermöglicht das Auführen einzelner Kommandos auf einem anderen Rechner, *rcp* das Kopieren von Dateien über Rechnergrenzen hinweg. Beide Kommandos funktionieren überhaupt nur dann, wenn der Zielrechner dem Ursprungsrechner vertraut; ist dies nicht der Fall, dann wird auch keine Passwort-Authentifizierung vorgenommen, sondern die Programme versagen ihren Dienst.

Die r-Tools werden mit gutem Recht als “broken as designed” bezeichnet und sollten am besten überhaupt nicht mehr verwendet werden, zumal mit der SSH (Secure Shell) eine wesentlich sicherere Alternative zur Verfügung steht (siehe Kapitel 6).

4.6 DNS

Der **Domain Name Service** (DNS) ist der Dienst, der im Internet die Zuordnung zwischen **Rechnernamen** und IP-Nummern durchführt. Sinn der Rechnernamen ist, dass sich IP-Nummern schwer merken lassen und die Zuordnung zwischen IP-Netznummern zu organisatorischen und geographischen Strukturen nicht offensichtlich ist.

Ein Beispiel: Der IP-Nummer `130.60.48.8` ist der Rechnername `frederic.ifi.unizh.ch` zugeordnet. Der Rechnername setzt sich dabei aus dem **Hostnamen** `frederic` und dem **Domainnamen** `ifi.unizh.ch` zusammen. Rechnernamen werden auch als **Fully Quantified Domain Name** (**FQDN**)

bezeichnet. Das Ermitteln der IP-Nummer zu einem bekannten Rechnernamen nennt man **Lookup** (Auflösen, Resolving), den umgekehrten Weg **Reverse Lookup**.

Im Internet gibt es keine zentrale Datenbank, in der alle Rechnernamen mit der dazugehörigen IP-Nummer gespeichert sind. Statt dessen gibt es viele einzelne Rechner, die einen Teil des Datenbestandes kennen, genannt **Nameserver**. Die Anordnung der Nameserver folgt dabei dem Aufbau der Domainnamen (Abbildung 4.2).

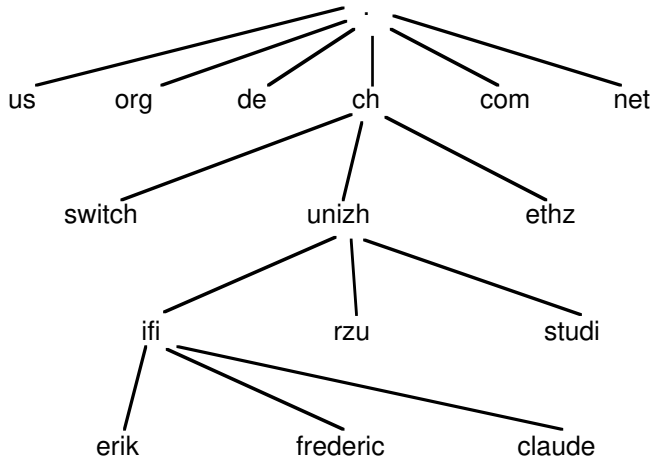

Abb. 4.2: Serverhierarchie beim Domain Name Service

Angenommen, ein Rechner irgendwo im Internet möchte die IP-Nummer des Rechners `frederic.ifi.unizh.ch` ermitteln. Dazu muss er einen beliebigen Nameserver kennen; üblicherweise betreibt jede Organisation mindestens einen davon. Der Nameserver wird feststellen, dass er den gesuchten Namen nicht kennt (ausser es handelt sich um den Nameserver des Institutes für Informatik der Uni Zürich, was hier aber ausgeschlossen werden soll). Er weiss aber, welches die Root-Nameserver sind, die alle Länderkürzel kennen (`us`, `org`, `ch`, `de`, `com`, `net` usw.). Root-Nameserver gibt es weltweit neun Stück, und alle Nameserver im Internet müssen diese kennen.

Der Nameserver befragt im ersten Schritt die Root-Nameserver nach dem gesuchten Namen `frederic.ifi.unizh.ch`. Aus Gründen der Fehlertoleranz werden alle neun auf einmal befragt; die erste Antwort wird verwendet. Die Antwort lautet, dass der Name selbst nicht bekannt ist, wohl aber die IP-Nummern der Nameserver für die Zone

ch. Im zweiten Schritt werden diese nach dem gesuchten Namen befragt. Wieder lautet die Antwort, dass der Name selbst nicht bekannt ist, aber die IP-Nummern der Nameserver von unizh.ch. Der dritte Schritt liefert die Nameserver von ifi.unizh.ch, der vierte schliesslich die gesuchte IP-Nummer zu frederic.ifi.unizh.ch.

Für die umgekehrte Richtung existiert ein ähnlicher Mechanismus. Eine Anfrage nach dem Rechnernamen, der zu der IP-Nummer 130.60.48.8 an den Reverse-Lookup-Root-Nameserver führt zu den IP-Nummern der Server für 130.x.x.x. Im zweiten Schritt werden Nameserver für den Bereich 130.60.x.x ermittelt, im dritten für 130.60.48.x. Die vierte Anfrage liefert schliesslich den gesuchten Rechnernamen.

Neben dem oben beschriebenen "normalen" Ablauf einer DNS-Anfrage kann es zahlreiche Besonderheiten geben. Beispielsweise könnte der Nameserver für 130.x.x.x auch direkt den für 130.60.48.x kennen und somit die Anfrage um einen Schritt verkürzen. Es ist möglich, einem Namen mehrere IP-Nummern zu geben und umgekehrt, und es gibt auch Aliasnamen, denen keine IP-Nummer, sondern ein anderer Name zugeordnet ist; beispielsweise ist www.ifi.unizh.ch ein Alias auf claude.ifi.unizh.ch.

Der Domain Name Service spielt aus der Sicht der Netzwerksicherheit eine Rolle, wenn die damit ermittelten Informationen zu Authentifizierungs- oder Auditing-Zwecken verwendet wird. Zum Beispiel liegt vielen Unix-Versionen ein sogenannter **TCP-Wrapper** bei, mit dessen Hilfe TCP-Verbindungen abgelehnt oder angenommen werden können. Entscheidungskriterium hierbei kann der Rechnername sein, von dem die Verbindung kam. Angenommen, ein Rechner erlaubt einkommende TCP-Verbindungen nur, wenn die Rechnernamen der Gegenseite auf *.unizh.ch (also auf alle Rechnernamen mit der Endung .unizh.ch) passen. Von der einkommenden Verbindung ist zunächst nur die IP-Nummer bekannt; über den Domain Name Service wird der zuständige Nameserver ermittelt und nach dem Rechnernamen gefragt. Ein Angreifer ausserhalb der Universität Zürich, der selbst den Nameserver für seine IP-Nummern betreibt, kann ihn so manipulieren, dass er tatsächlich einen Namen unterhalb von .unizh.ch zurückliefert, wenn er weiss, dass dies das Kriterium für den Zugang ist. Diese Technik wird **DNS-Spoofing** genannt.

Abhilfe gegen DNS-Spoofing ist relativ einfach: Sobald der Rechnername ermittelt ist, wird das Resolving in umgekehrter Reihenfolge durchgeführt, also die zum ermittelten

Rechnernamen gehörende IP-Nummer ermittelt. Diese Anfrage landet, sofern nicht gerade die Root-Nameserver oder die Server für `ch` korrupt sind, bei den Nameservern von `unizh.ch` und wird mit Sicherheit nicht mehr die ursprüngliche IP-Nummer zurückliefern (da sich diese ja nicht auf dem Campus befindet). In diesem Fall wird die Verbindung nicht erlaubt oder ein Alarm ausgelöst.

Neuere TCP-Wrapper besitzen die Fähigkeit, die ermittelten Rechnernamen zu überprüfen. Jedoch muss diese Eigenschaft extra aktiviert werden; die Begründung vieler Hersteller ist, dass sie zusätzliche Netzlast erzeugt.

4.7 X11

Das **X Window System 11** (kurz **X11**, noch kürzer **X**) ist ein verteiltes Graphiksystem, das als Unterbau für graphische Benutzeroberflächen gedacht ist. Es wurde ursprünglich am MIT entwickelt und wird überwiegend im Unix-Bereich eingesetzt, obwohl es von Betriebssystem, Hardwarearchitektur und sogar Netzwerkprotokoll völlig unabhängig ist. X11 besitzt eine Client-Server-Architektur.

Aufgabe des Servers ist es, Text und Grafik auf einem Bildschirm darzustellen, die ihm der Client über das Netzwerk liefert, und Tastatur- und Mauseingaben in einem hardwareunabhängigen Format an den Client weiterzuleiten. Der Client ist ein Programm, das auf dem selben oder einem anderen Rechner läuft und über die graphische Ausgabe mit dem Benutzer interagiert. X-Server gibt es für viele grafikfähigen Computersysteme wie MS Windows, Apple Macintosh, Atari ST und die meisten Unix-Derivate. X-Clients laufen prinzipiell auf allen netzwerkfähigen Systemen. Sinnvoll sind sie jedoch nur auf Multitaskingsystemen wie Unix, VMS oder MVS. Die Terminologie führt häufig zu Verwechslungen, denn im Gegensatz zu anderen Client-Server-Anwendungen läuft hier der Server auf dem Rechner, vor dem der Benutzer sitzt, während der Client auf irgendeinem Rechner im Netz gestartet werden kann.

Die Clients eröffnen zur Kommunikation mit dem X-Server eine TCP-Verbindung. Ueber diese Verbindung kann er beliebige Funktionen des Servers wie das Zeichnen von Bildschirmausgaben oder das Auslesen der Tastatur veranlassen. Damit nicht jeder beliebige Rechner im Internet eine TCP-Verbindung auf den Server eröffnen kann, gibt es zwei Arten von Zugriffskontrollmechanismen:

- **Zugriffskontrolle nach Rechnern**: Der Benutzer kann seinen X-Server anweisen, TCP-Verbindungen von bestimmten Rechnern anzunehmen, von allen anderen aber abzuweisen. Wie aus Abschnitt 3.12 bekannt, ist dies jedoch kein wirksamer Authentifizierungsmechanismus, denn unter Verwendung von IP-Spoofing in Kombination mit Source Routing oder Sequence Number Guessing kann dieser Schutz umgangen werden. Zudem ermöglicht er allen Benutzern der erlaubten Rechner Zugriff auf den eigenen X-Server und damit auf seinen Bildschirm und seine Tastatur. Der Benutzer kann auch bestimmten Rechnernamen den Zugriff gewähren; in diesem Fall kann der Schutz durch DNS-Spoofing umgangen werden. Schliesslich halten die Autoren mancher Bücher über X11 es gar nicht für nötig, auf Sicherheitsaspekte einzugehen. Sie nennen den Befehl, mit dem allen Rechnern im Internet der Zugriff erlaubt wird, und kümmern sich im Folgenden nicht mehr weiter darum.

- **Zugriffskontrolle nach Benutzern**: Beim Verbindungsaufbau übermittelt der Client eine Zeichenkette, die in einer Datei im Homeverzeichnis des Benutzers abgelegt ist, an den Server. Findet der Server diesen String ebenfalls im Homeverzeichnis, dann erhält der Client Zugriff. Diese Form der Authentifizierung ist nur in Mehrbenutzersystemen sinnvoll. Problematisch ist hier, dass die Authentifizierungsstrings unverschlüsselt über das Netzwerk transportiert werden, also abgehört werden können. Ausserdem ist die Aufbewahrung im Homeverzeichnis des Benutzers keine sichere Lösung; sie gewährt dem Administrator des Client-Rechners, der ja diese Datei lesen kann, Rechte über den X-Server.

4.8 SSH

Die **Secure Shell** (SSH) ist als ein sicherer Ersatz für Telnet, rlogin, rsh und rcp gedacht. Sie wurde 1995 von Tatu Ylonen aus Finnland entwickelt und wird frei im Internet vertrieben. Mittels kryptographischer Techniken ermöglicht sie eine weitgehende Resistenz gegen die bisher vorgestellten Angriffstechniken.

- **Verschlüsselung**. Alle beteiligten Rechner besitzen RSA-Schlüsselpaare. Am Anfang der Verbindung wird ein Sitzungsschlüssel generiert und mit RSA verschlüsselt ausgetauscht. Die gesamte Kommunikation geht mit diesem Sitzungsschlüssel chiffriert über die Leitung, wobei IDEA, DES oder andere symmetrische Algorithmen verwendet werden. Ein Angreifer kann durch reines

Abhören keine Informationen gewinnen, die ihn zum Einloggen in das System befähigen. Der RSA-Schlüssel, mit dem die symmetrischen Schlüssel ausgetauscht werden, wird jede Stunde neu erzeugt und niemals in einer Datei abgespeichert. Das verhindert, dass aufgezeichnete verschlüsselte Datenströme dechiffriert werden können, wenn ein Angreifer zu einem späteren Zeitpunkt Root-Rechte auf dem Serverrechner bekommt und die geheimen Schlüssel auslesen kann.

- **Authentifizierung**. Alle beteiligten Rechner besitzen RSA-Schlüssel und authentifizieren sich gegenseitig beim Verbindungsaufbau. Zunächst werden die öffentlichen Schlüssel ausgetauscht und mit den bereits bekannten verglichen. Wenn die korrekten öffentlichen Schlüssel bereits vorher bekannt sind, können auf diese Weise aktive Angriffe, bei der ein Angreifer den Schlüssel gegen seinen eigenen austauscht, erkannt werden. Lediglich beim allerersten Verbindungsaufbau besteht die Möglichkeit eines aktiven Angriffes, wenn die öffentlichen Schlüssel noch nicht vorher ausgetauscht wurden. Anschliessend findet eine Authentifizierung über RSA statt.

- **X11-Verbindungen**. Bei einem Login mit SSH auf einem entfernten Rechner können X-Clients gestartet werden, ohne dass diese eine TCP-Verbindung auf den X-Server öffnen müssen. Dadurch entfallen die in Abschnitt 4.7 geschilderten Sicherheitsprobleme.

- **Einloggen ohne Passwort** (wie bei den r-Tools) ist ebenfalls möglich. Allerdings verlässt sich SSH nicht auf IP-Nummern, Rechnernamen und Benutzernamen, sondern nur auf RSA-Authentifizierung.

- **TCP-Forwarding**. Beliebige TCP-Verbindungen können über eine SSH-Verbindung übertragen werden (Tunneling) und werden somit automatisch gegen Abhören und Spoofingattacken geschützt. Die Bedienung ist allerdings umständlich und nicht für jeden Dienst geeignet.

SSH-Server gibt es nur für Unix, frei erhältliche Clients für Unix und OS/2, kommerzielle Clients für MS Windows.

4.9 Zukünftige Forschungsarbeiten

Der Bereich der Netzwerksicherheit bietet noch ein weites Betätigungsfeld. Die Forschungsaktivitäten konzentrieren sich im wesentlichen auf den Einsatz kryptographischer Techniken, um bekannten Diensten zusätzliche Sicherheitseigenschaften zu verleihen.

Zur Wahrung der Vertraulichkeit von E-Mail kann Verschlüsselung, zur Wahrung der Integrität und der Authentizität Signaturen eingesetzt werden. Beides ist mit PGP (siehe Abschnitt 2.10) machbar. Jedoch obliegt es den Benutzern, Schlüssel zu erstellen, auszutauschen und die PGP-Berechnungen durchzuführen. Allerdings ist das mühsam, da Public-Key-Berechnungen lange dauern und PGP von vielen MUA's gar nicht oder nur unzureichend unterstützt wird. Daher werden kryptographische Techniken bei E-Mail kaum verwendet. Aktuelle Forschungsarbeiten beschäftigen sich mit Systemen, bei denen E-Mail automatisch mit rechnereigenen Schlüsseln ver- und auf der Empfängerseite entschlüsselt wird, ohne dass der Benutzer (im Normalfall) etwas davon mitbekommt.

Protokolle, die auf TCP beruhen, lassen sich über eine SSH-Verbindung tunneln (siehe Kap. 4.8). Derzeit entstehen einige Anwendungen, die davon Gebrauch machen, vor allem für das World Wide Web.

Zur Wahrung der Verfügbarkeit gibt es Arbeiten auf den Gebiet der Fehlertoleranz in Netzwerken. Das Grundprinzip der meisten Protokolle ist, Daten über mehrere unterschiedliche Pfade zum Ziel zu leiten, so dass fehlerhafte Knoten und Wege erkannt werden können [Lamport 82]. Problematisch an diesen Protokollen ist jedoch die hohe Netzlast und die Tatsache, dass nur eine begrenzte Anzahl von Knoten fehlerhaft sein darf, damit die Protokolle tatsächlich funktionieren.

5 Sicherheitsprobleme in Mehrbenutzersystemen

Zusätzlich zu den Sicherheitsrisiken, die der Anschluss eines Rechners an ein globales Netzwerk birgt, kommt bei Mehrbenutzersystemen ein weiteres Risiko hinzu: die Interaktionen zwischen den Benutzern.

Ein Mehrbenutzersystem ist ein Computersystem, auf dem mehrere Benutzer gleichzeitig (oder auch nacheinander) arbeiten können. Dabei kann es sich um einen einzigen Computer handeln, oder um mehrere, die über ein Netzwerk miteinander verbunden sind.

Prinzipiell ist jeder PC oder PC-Verbund, auf den mehrere Benutzer Zugriff haben, ein Mehrbenutzersystem. Die gängigen PC-Betriebssysteme MS-DOS, MS-Windows, OS/2 und MacOS bieten jedoch wenig Unterstützung für den Betrieb mit mehreren Benutzern. Mehrbenutzerbetrieb ist mit diesen Systemen nur dann sinnvoll, wenn jeder Benutzer seinen eigenen Rechner hat. Datenaustausch ist nur auf umständliche Weise möglich, z.B. über spezielle Rechner, die als Fileserver arbeiten.

Moderne Betriebssysteme sind von vorne herein für den Betrieb mit mehreren Benutzern konzipiert. Das bedeutet, dass mehrere Benutzer gleichzeitig mit dem Rechner arbeiten können und sich dabei gegenseitig nicht in die Quere kommen. Möglich wird dies durch mehrere Voraussetzungen:

- **Multitasking**. Das Betriebssystem muss die Fähigkeit besitzen, mehrere Prozesse (quasi-)parallel ablaufen zu lassen. Natürlich laufen die Prozesse nicht wirklich parallel; der Prozessor schaltet aber so schnell zwischen ihnen hin und her, dass es für den Benutzer so aussieht, als ob die Prozesse gleichzeitig ablaufen würden.

- **Unabhängige Konsolen**. An den Rechner sind mehrere Terminals (Bildschirm und Tastatur) angeschlossen. Die Programme sind an die Konsole gebunden, von der aus sie gestartet werden. Sie beeinflussen nicht Ein- und Ausgabe an den anderen Konsolen.

- **Benutzereigene Dateien**. Jeder Benutzer besitzt einen eigenen Bereich im Dateisystem, in dem er eigene Dateien anlegen kann. Für diese Dateien kann er selbst bestimmen, welche Benutzer sie lesen, verändern oder löschen dürfen.

- **Verteilte Dateisysteme**. Besteht das System aus mehreren Rechnern, so soll sich ein Benutzer auf jedem Rechner gleichermassen einloggen können. Systemumgebung und eigenen Dateien sollen überall erreichbar sein, egal wo sie tatsächlich physikalisch liegen.

Die Sicherheitsanforderungen an Mehrbenutzersysteme sind höher als die an Einbenutzersysteme. Neben der Einbruchsmöglichkeit von aussen ist es hier auch nötig, die einzelnen Benutzeraccounts, -daten und -prozesse voreinander zu schützen.

5.1 Login und Passwörter

Bevor ein Benutzer auf dem System arbeiten kann, muss er sich authentifizieren. Dies geschieht durch Eingabe seines Benutzernamens und des Passwortes. Es ist also nötig, dass das System das Passwort schon vorher kennt. Die Passwörter der Benutzer im Klartext abzuspeichern, wäre jedoch eine eklatante Sicherheitslücke. Denn erfahrungsgemäss verwenden viele Benutzer das selbe Passwort auf verschiedenen Mehrbenutzersystemen oder wählen sie zumindest ähnlich. Ein Hacker, dem es gelänge, die Passwörter auf einem System zu stehlen, hätte wohl erheblich mehr gewonnen als nur den Zugang zu diesen Accounts.

Unter Unix werden nicht die Passwörter selbst, sondern kryptographische Hashsummen der Passwörter gespeichert. Gibt der Benutzer beim Login sein Passwort ein, wird darüber ebenfalls die Hashumme berechnet und mit der gespeicherten verglichen. Nur im Falle einer Uebereinstimmung erhält der Benutzer Zutritt.

Der Hashsummenalgorithmus ist so gewählt, dass er nicht rückwärts berechnet werden kann; zumindest ist ausser dem Durchprobieren aller möglicher Passwörter kein Verfahren bekannt, mit dem aus der Hashsumme das Passwort zurückgewonnen werden

kann. Viele Unix-Versionen wiegen sich in Sicherheit und erlauben jedem Benutzer den Zugriff auf die Hashsummen der Passwörter: die Datei `/etc/passwd` ist für alle lesbar.

Ein Schwachpunkt ist jedoch die geringe Länge von nur 8 signifikanten Zeichen. Da ein Zeichen in 8 Bit abgespeichert wird, ist der Schlüsselraum theoretisch 64 Bit gross; praktisch werden aber meist nur Buchstaben und Ziffern für die Passwörter verwendet, wodurch die Zahl der Möglichkeiten stark reduziert wird. Noch einfacher sind Passwörter zu knacken, wenn sie aus sinnvollen Wörtern bestehen. Es gibt Programme, die automatisch ganze Wörterbücher von sinnvollen Wörtern und Variationen davon durch die Hashfunktion schleusen und mit den gespeicherten Hashwerten zu vergleichen. Auf neueren Unix-Systemen ist daher die Passwortdatei nicht mehr global lesbar, sondern die Programme, die die Authentifikation durchführen, haben spezielle Rechte, um sie zu lesen.

5.2 Dateien und Zugriffsrechte

Auf einem Mehrbenutzersystem sollen einzelne Benutzer bestimmen können, von wem und in welcher Weise auf ihre Dateien zugegriffen werden kann. Die gängigen Betriebssysteme bieten diese Funktionalität, wenn auch nicht in sehr flexibler Weise.

Unter Unix hat jede Datei und jedes Verzeichnis einen Besitzer und gehört zu einer Gruppe. Benutzer gehören ebenfalls zu (mindestens) einer Gruppe. Beim Einrichten des Benutzeraccounts bekommt jeder Benutzer ein Verzeichnis das ihm gehört (genannt Homeverzeichnis). Darin kann er Dateien und weitere Verzeichnisse anlegen, bearbeiten, löschen und die Zugriffsrechte einstellen.

Für Dateien gibt es drei Arten von Rechten: Das **Leserecht** erlaubt demjenigen, der es hat, eine Datei einzusehen oder zu kopieren. **Schreibrecht** bedeutet, dass der Dateiinhalt verändert werden darf. **Ausführungsrecht** ist nur bei Dateien sinnvoll, in denen Programme abgelegt sind; es bewirkt, dass ein Benutzer dieses Programm ausführen darf.

Für Verzeichnisse gibt es ebenfalls diese drei Rechtetypen, aber sie haben eine etwas andere Semantik: Das Ausführungsrecht entscheidet darüber, ob ein Benutzer überhaupt etwas in diesem Verzeichnis machen darf; besitzt er es nicht, sind die übrigen Rechte wirkungslos. Das Leserecht ermöglicht es, ein Inhaltsverzeichnis aller Dateinamen anzuzeigen. Ohne dieses Recht können zwar Dateioperationen durchgeführt werden, aber

nur auf Dateien, deren Namen dem Benutzer bekannt sind. Mit Schreibrecht auf ein Verzeichnis kann man neue Dateien anlegen, bestehende Dateien umbenennen oder löschen. Bei den meisten Unix-Versionen gibt es darüber hinaus noch das Sticky-Recht. Es schränkt das Schreibrecht dahingehend ein, dass nur noch Dateien angelegt werden können; Löschen oder Umbenennen ist jedoch nur dem Besitzer einer Datei möglich.

Die oben aufgezählten Rechte können bei jeder Datei oder jedem Verzeichnis getrennt für den Besitzer, die Gruppe sowie alle anderen Benutzer, die nicht der Gruppe angehören, eingestellt werden. Die Vergabe von individuellen Rechten an einer Datei für jeden einzelnen Benutzer ist nicht möglich. Benutzer, die häufig die selben Rechte an bestimmten Dateien brauchen, können aber vom Systemadministrator zu einer Benutzergruppe zusammengefasst werden. Die Gruppeneinteilung spiegelt daher häufig organisatorische Strukturen wieder, beispielsweise Arbeitsprojekte, gemeinsame Geschäftsprozesse oder Beschäftigungsstatus.

Manche Unix-Versionen erlauben über das oben beschriebene Rechte-Konzept hinaus, bis zu einer bestimmten Anzahl von Benutzern individuelle Rechte an einer Datei zu vergeben. Die Datenstruktur, in der diese Rechte gespeichert werden, wird **Access Control List** (**ACL**) genannt. Aus Effizienzgründen ist die Anzahl der Einträge begrenzt, beim Hewlett-Packard-Unix (HPUX) beispielsweise auf 10.

Der Zugriff auf Hardwarekomponenten wird unter Unix ebenfalls über das Dateirechtekonzept geregelt. Die Komponenten werden durch spezielle Dateien, sog. **Devices**, angesprochen. Die Devices wie andere Dateien im Dateisystem und besitzen den normalen Rechtemechanismus. Ein Beispiel: Sollen nur bestimmte Benutzer das Recht besitzen, das Modem benützen zu dürfen, dann kann dafür eine Gruppe `modem` eingerichtet werden; das Modem-Device wird dieser Gruppe zugeordnet und bekommt Lese- und Schreibrechte für die Gruppe.

5.3 Prozesse

Auf einem Mehrbenutzersystem kann jeder Benutzer Programme ausführen. Auf Multitasking-Betriebssystemen können diese Programme gleichzeitig laufen und werden Prozesse genannt. Auch hier gibt es abgestufte Rechte: Jeder Prozess unter Unix läuft unter einer Benutzer- und einer Gruppenkennung und hat genau auf diejenigen Dateien Zugriff, auf die auch der entsprechende Benutzer bzw. die Gruppe zugreifen darf. Die

Kennungen entsprechen dabei denen des Benutzers, der die Prozesse gestartet hat. Durch dieses Rechtekonzept wird das System vor den Benutzern, aber auch die Benutzer gegenseitig beschützt. Denn jedes von einem Benutzer aufgerufenen Programm kann maximal seine eigenen sowie öffentlich schreibbare Dateien zerstören, jedoch (korrekte Implementierung des Betriebssystems vorausgesetzt) keine anderen.

Manchmal ist es allerdings nötig, dass ein Prozess mehr Rechte benötigt, als ein Benutzer hat. Zum Beispiel gibt es Programme, die bestimmte Informationen in einer globalen Datei mitprotokollieren. Es wäre ungeschickt, wenn alle Benutzer das Schreibrecht für diese Datei hätten, denn dann könnten sie auch Eintragungen löschen oder unsinnige Angaben eintragen. Statt dessen bietet Unix die Möglichkeit, Programme nicht mit den Rechten des Aufrufers, sondern mit denen den Besitzers ablaufen zu lassen; analoges gilt für die Gruppe. Diese solche Programme werden **setuid** (set user id) bzw. **setgid** (set group id) genannt. Oben genanntes Problem kann zum Beispiel dadurch gelöst werden, dass eine neue Gruppe `log` angelegt wird, die Protokolldatei Schreibrechte für diese Gruppe erhält und das Programm beim Aufruf die Gruppenkennung `log` annimmt.

Programme dieser Art können wesentliche Sicherheitslücken darstellen, wenn die fehlerhaft programmiert sind: Wenn der Benutzer durch Parameter oder interaktive Eingaben das Programm dazu bringen kann, neben seinem vorgesehenen Zweck noch andere Funktionen auszuführen, dann geschieht dies mit den Rechten des Programmes. Im schlimmsten Fall, wenn das Programm unter der Benutzerkennung `root` läuft, sind dies sogar die Rechte an allen Dateien und Ressourcen des Rechners. Programmierfehler dieser Art sind nicht einfach zu finden: Sie brauchen nicht in dem Programm selber zu stecken, sondern zum Beispiel in anderen Programmen, die von dem setuid-Programm aufgerufen werden, oder den Bibliotheken.

Ein aktuelles Beispiel steht in [CERT 96-18]: Das Programm `rlogin` unter Unix ist setuid-`root`. Dies ist nötig, da es eine TCP-Verbindung öffnet und dabei einen Ursprungsport kleiner als 1024 verwendet, was nur von `root` möglich ist. `rlogin` ist der Client für den in Abschnitt 4.5 vorgestellten rlogin-Dienst; `rlogin` *<Rechnername>* ermöglicht das Einloggen auf dem angegebenen Rechner. Zur Umwandlung des Namens in die IP-Nummer benutzt `rlogin` eine Funktion aus der Systembibliothek. Diese Funktion ist bei einigen Versionen des Solaris-Betriebssystems von Sun fehlerhaft: Wird ein unsinniger Rechnername einer bestimmten Bauart angegeben, dann ist es möglich, ein beliebiges Programm zu starten. Das Programm läuft unter der Kennung seines Aufrufers - und der ist in diesem Fall `root`.

5.4 Gemeinsame Dateisysteme im Netzwerk

In Mehrbenutzersystemen sollen Benutzer Zugriff auf ihre Daten haben, unabhängig davon, an welchem Rechner sie gerade arbeiten. Ausserdem sollen Programme, die der Administrator einmal installiert hat, an allen Rechnern zur Verfügung stehen.

Das **Network File System** (**NFS**) hat die Aufgabe, Dateisysteme auf der Platte eines Rechners anderen Rechnern zur Verfügung zu stellen. Anwendungsprogramme merken dabei, abgesehen von der Geschwindigkeit, keinen Unterschied zwischen lokalen und importierten Verzeichnissen; der Zugriff erfolgt **transparent**. Insbesondere gelten die bekannten Unix-Zugriffsrechte auch auf Dateisystemen, die per NFS zugänglich gemacht werden.

Der Rechner, der ein Dateisystem exportiert, kann anderen Rechnern unterschiedliche Zugriffsbefugnisse zuweisen. Die Authentifizierung erfolgt dabei allerdings anhand von IP-Nummer oder Rechnername, und ist daher den bereits bekannten Sicherheitsproblemen im Zusammenhang mit IP- und DNS-Spoofing ausgesetzt (siehe Abschnitte 3.12 und 4.6).

In NFS-importierten Dateisystemen gelten die bekannten Unix-Zugriffsrechte. Der Schwachpunkt dabei ist allerdings, dass die Benutzeridentifizierung auf dem Client stattfindet. Ein Benutzer loggt sich an einem Rechner im Cluster ein, authentisiert sich mit seinem Passwort und ist dem Rechner fortan bei sämtlichen Datei- und Programmzugriffen als ein spezieller Benutzer bekannt. Gelingt es einem Angreifer, einen einzelnen Rechner auszutricksen und eine andere Benutzerkennung anzunehmen, dann hat er ab sofort an allen Dateien, die ihm lokal oder per NFS von anderen Rechnern zugänglich sind, die Rechte der falschen Benutzerkennung.

Von Sun existiert mit **Secure NFS** eine Lösung dieses Problems. Bei Secure NFS können Benutzer nach dem Einloggen zwar auf ihre Dateien von NFS-importierten Dateisystemen zugreifen, jedoch nur mit minimalen Rechten. Dazu existiert ein Benutzer `nobody`, der nur sehr eingeschränkte Rechte besitzt. Erst nachdem der Benutzer sich beim NFS-Server authentisiert hat, kann er mit seiner eigenen Benutzerkennung auf Dateien zugreifen. Die Authentifizierung geschieht dabei mit dem in Abschnitt 2.7 beschriebenen ElGamal-Signaturverfahren; öffentliche und private Schlüssel aller Benutzers sind von allen Rechnern lesbar; die privaten sind jedoch mit einem Passwort verschlüsselt, das der Benutzer bei der Authentisierung eingeben muss.

5.5 Gemeinsame Nutzung von Druckern

In einem Firmennetz mit vielen Rechnern und Benutzern ist es wünschenswert, dass im Netz vorhandene Drucker gemeinsam genutzt werden können. Viele Systeme bieten die Möglichkeit, einen an den lokalen Rechner angeschlossenen Drucker über das Netz mitbenutzen zu können. Unter Unix bietet der **Line Printer Daemon** (*lpd*) die Möglichkeit, über eine TCP-Verbindung zu druckende Daten einliefern zu können. Daneben kümmert er sich auch noch um andere Dinge wie das Konvertieren der Daten in ein für den Drucker geeignetes Format.

Dank seiner TCP-Schnittstelle ist der *lpd* eines Rechners prinzipiell vom gesamten Netz aus erreichbar, im Falle einer Internet-Anbindung also weltweit. Da es wohl nicht erwünscht ist, dass die ganze Welt den Drucker mitbenutzen kann, lassen sich Zugriffsbeschränkungen vergeben. Die Authentisierung erfolgt dabei nach IP-Nummern oder Rechnernamen, so dass sie mit den bekannten Spoofing-Techniken umgangen werden kann.

Nebenbei kümmert sich der *lpd* auch noch um andere Dinge wie das Konvertieren der Daten in ein für den Drucker geeignetes Format. Da er auf manchen Systemen unter der Benutzerkennung root läuft, ist das nicht ungefährlich: die Filtermechanismen werden mit `root`-Privilegien aufgerufen, Programmierfehler können grossen Schaden am System anrichten.

Beispielsweise müssen Postscript-Dateien gerastert werden, wenn sie auf nicht-Postscript-fähigen Druckern ausgegeben werden sollen. Dazu kann das frei erhältliche Programm *ghostscript* verwendet werden. Die Sprache Postscript ist weit mehr als eine Sprache zur Seitenbeschreibung; sie kann auch gewöhnliche Unix-Befehle enthalten, und *ghostscript* führt diese Befehle tatsächlich aus, wenn es nicht mit einer speziellen Sicherheitsoption aufgerufen wird. Ein Angreifer kann also im schlimmsten Fall Befehle unter root-Kennung auf einem Unix-System ausführen, indem er eine entsprechende Postscript-Datei an den *lpd* zum Drucken einliefert.

6 Firewalls

6.1 Einleitung

Rechneranlagen, wie sie heute von Firmen zur Datenverarbeitung eingesetzt werden, besitzen ein enormes Ausmass an Komplexität. Viele der in Kapitel 2 beschriebenen Sicherheitslücken und Angriffsvarianten basieren auf Fehlern in Betriebssystem oder der Netzwerksoftware. Wegen der grossen Komplexität dieser Softwareprodukte ist es nicht möglich, diese vollständig fehlerfrei zu halten. Bei Betriebssystemsoftware sind Programmierfehler häufig besonders sicherheitskritisch, da sie meist sehr hohe Privilegien hat, also auf viele Daten zugreifen kann.

Abhilfe gehen diese Angriffsvarianten ist im Allgemeinen nur möglich, indem das fehlerhafte Programm oder das Betriebssystem korrigiert wird. Bei kommerziellen Betriebssystemen ist man dabei meist auf den Hersteller angewiesen, der möglichst schnell einen Patch liefern sollte. Bei einem System, dessen Sourcecode frei verfügbar ist, wie Linux oder einigen BSD-Derivaten, ist dagegen häufig der Patch bereits verfügbar, wenn eine Sicherheitslücke bekannt wird. Bei komplexeren Problemen ist es zumindest möglich, die entsprechende Funktion solange ausser Kraft zu setzen, bis eine verbesserte Systemversion erhältlich ist. Dies setzt natürlich gewisse Kenntnisse von Programmierung und Betriebssystemarchitektur voraus.

Sicherheitslücken dieser Art werden in Form von Sicherheitsbulletins veröffentlicht, die von einer zentralen Stelle, dem **Computer Emergency Response Team** (**CERT**), verbreitet werden. Wenn eine solche Lücke entdeckt und an das CERT gemeldet wird, versucht dieses zunächst, die dadurch ermöglichten Angriffe nachzuvollziehen, die betroffenen Systeme zusammenzustellen und die Hersteller zu konsultieren. Daher

vergeht häufig einige Zeit, bis eine Lücke grossflächig bekannt wird und Massnahmen zur Beseitigung getroffen werden können. Bis der Systemverantwortliche diese erhalten, verstanden und umgesetzt hat, können andere die Lücken ausnutzen und in Rechnersysteme einbrechen oder Anlagen lahmlegen. Auch ist nicht garantiert, dass eine Lücke überhaupt an das CERT oder eine andere Organisation gemeldet wird; dies hängt stark von der Motivation des Entdeckers ab.

Aus dem Gesagten folgt, dass es letzten Endes nicht möglich ist, Angriffe auszuschliessen, die auf Fehlern in der Systemsoftware vernetzter Rechner basieren. Geht man jedoch von der Annahme aus, dass die Rechner im lokalen Netzwerk im Wesentlichen von vertrauenswürdigen Personen genutzt werden und die eigentliche Gefahr nur durch Netzwerkzugang von aussen droht, dann ergibt sich ein anderer Ansatzpunkt für die Vermeidung der in diesem Dokument beschriebenen Schwachstellen: Die Sicherung des Uebergangspunktes zwischem dem lokalen Netz und dem globalen Internet, oder allgemeiner, den Punkt, an dem das eigene, vertrauenswürdige Netz an ein anderes, weniger vertrauenswürdiges Netz angeschlossen ist.

Hinter dem Begriff **Firewall** verbirgt sich ein Sammelsurium verschiedener Konzepte und Techniken. Gemeinsam ist allen, dass sie versuchen, an der Schnittstelle (mindestens) zweier, unterschiedlich vertrauenswürdiger Netze einen kontrollieren Uebergang zu ermöglichen. Der zentrale Gedanke dabei ist, gerade diejenigen Daten passieren zu lassen, die für den ordnungsgemässen Betrieb der benötigten Dienste erforderlich sind. Nicht benötigte Dienste und Protokolle werden gesperrt. Denn je weniger Protokolle an dem Uebergangspunkt zugelassen werden, desto weniger können Sicherheitsmängel der Dienstimplementierungen zu Angriffen auf das interne Netz genutzt werden. Insbesondere sollen diejenigen Daten gesperrt werden, die dazu geeignet sind, bekannte Sicherheitslücken in der Systemsoftware auszunutzen.

Allen Firewall-Systemen gemeinsam ist also das Filtern der Daten anhand definierter Merkmale. Die Regeln für die Filterung werden in Form eines regelbasierten Systems formuliert.

Die verschiedenen Konzepte unterscheiden sich u.A.

- in der Ebene oder den Ebenen des zugrundeliegenden Netzwerkprotokolles, auf der die Daten untersucht werden,
- in der "Intelligenz" des regelverarbeitenden Systems,

- in den Kriterien, die zur Entscheidungsfindung herangezogen werden,
- in der Transparenz, also der Sicht, in der der Benutzer bzw. die Applikation die Firewall sieht,
- im Grad der Fehlertoleranz und der Ausfallsicherheit und
- im Angebot an zusätzlichen Funktionen, z.B. Protokollierung des Netzwerkverkehrs oder Alarmierung.

Im folgenden sind einige gängigen Konzepte anhand von Beispielen aufgeführt. Dabei wird nur noch auf das für uns wichtige TCP/IP Protokoll eingegangen; die Konzepte sind auch auf andere Protokolle wie das Bitnet oder Novell IPX übertragbar, aber die Details bei der Realisierung unterscheiden sich natürlich.

6.2 Paketfilter

Paketfilter verbinden zwei Netzwerke und filtern Daten auf der Vermittlungs- und Verbindungsschicht, also den Schichten 3 und 4 der ISO/OSI Schichtenhierarchie bzw. den Schichten II und III des TCP/IP-Protokolls. Ein Paketfilter ist üblicherweise ein Rechner, der mit mindestens zwei Netzwerkkarten ausgestattet und an verschiedene physikalische Netze angeschlossen ist. Er erbringt eine Routingfunktionalität zwischen dem inneren und dem äusseren Netz. Allerdings werden nicht alle, sondern nur bestimmte IP-Datagramme weitergeleitet. Daher werden Paketfilter häufig auch **Screening Router** genannt.

Einfache Paketfilter benötigen für die Filterung nur zwei Angaben, nämlich Ursprungs- und Ziel-IP-Nummer der einkommenden IP-Pakete. Anhand einer Tabelle entscheiden sie, welche Kombinationen durchgelassen und welche ausgefiltert werden. Diese beiden Informationen sind im Header jedes einzelnen IP-Paketes enthalten.

Beispiel. Eine Firma benutze das Class-C-Netz 193.75.101.0 und bietet ihren Mitarbeitern Zugriff auf das World Wide Web sowie Erreichbarkeit per E-Mail. Der Anschluss des internen Netzes zur Aussenwelt sei durch einen Paketfilter realisiert, der im inneren Netz die IP-Nummer 193.75.101.1 trägt. Die Rechner .2, .3 und .4 seien DNS-Server, Mailserver und WWW-Proxy, laufen unter UNIX, und werden von den Systemadministratoren verwaltet und auf Sicherheitsaspekte überprüft. Die übrigen IP-Nummern werden von Mac's und PC's benutzt, die in den Dienstzimmern der Mitarbeiter

stehen. Je nach Experimentierfreudigkeit der Benutzer haben die Administratoren kaum die Möglichkeit, auf die dort installierte Software und damit auf die Netzwerksicherheit dieser Rechner Einfluss zu nehmen. Das Setup ist in Abbildung 5.1 dargestellt.

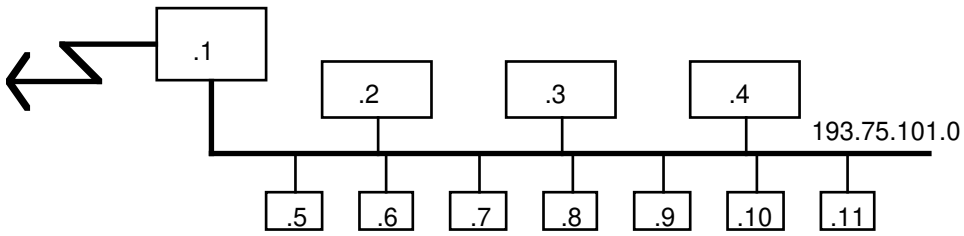

Abb. 5.1: Einsatz einesPaketfilters

Eine mögliche Zuordnung von Ursprungs- und Ziel-IP-Nummern zu Erlaubnis- oder Verbotsentscheidungen zeigt Abbildung 5.2.

Ursprung	**Ziel**	**Erlaubnis**
`193.75.101.1-4`	`*`	`+`
`*`	`193.75.101.1-4`	`+`
`193.75.101.5-255`	`*`	`-`
`*`	`193.75.101.5-255`	`-`

Abb. 5.2: Konfiguration eines einfachen Paketfilters

Die Konfiguration erlaubt also dem Firewall-Rechner und den Servern uneingeschränkten Zugriff auf alle Rechner im Internet. Die übrigen Rechner sind dagegen abgeschottet. Ihre Kommunikation mit der Aussenwelt kann nur mit den Servern erfolgen: Ausgehende E-Mail wird an die Server geschickt, die sie wiederum weiterverteilen; einkommende Mail wird vom Mailserver entgegengenommen und kann von den Benutzermaschinen abgeholt werden. WWW-Verbindungen sind nur über den Proxyserver möglich. Der Server nimmt die Anfragen der Clients entgegen, setzt sie um in Anfragen an die Server in der weiten

Welt, nimmt das Ergebnis (meist eine HTML-Seite) entgegen und liefert diese an den Client.

Der Sicherheitsgewinn besteht im Wesentlichen darin, dass keine IP-Pakete direkt von aussen an die Benutzermaschinen gelangen können. Hat zum Beispiel ein Benutzer versehentlich einen FTP-Server auf seinem Mac gestartet und kein Passwort gesetzt, könnten andernfalls sämtliche Dokumente auf der lokalen Platte von aussen abgerufen werden. Dies wird durch den Firewall verhindert.

Diese einfache Form der Paketfilterung hat jedoch erhebliche Nachteile: Die Entscheidung für Erlaubnis oder Verbot einer Kommunikation zwischen zwei Rechnern hängt ausschliesslich von den IP-Nummern der Rechner ab. Es ist nicht möglich, nur bestimmte Protokolle der Transportschicht (wie z.B. TCP, UDP), oder gar bestimmte Dienste, die auf der Transportschicht aufbauen, zu erlauben.

Beispielsweise gibt es in obigem Szenario Schwierigkeiten, wenn die Firma ihren Mitarbeitern den Zugriff auf einen Newsserver ausserhalb des lokalen Netzes ermöglichen möchte. Regeln, die eine TCP-Verbindung der Clients mit dem NNTP-Port eines Rechners ausserhalb des Firmennetzes erlauben, sonst aber alle anderen Formen der Kommunikation zwischen diesen Rechnern verbieten, sind mit obigem Firewall nicht zu machen. Es bleibt die Möglichkeit, entweder einen eigenen Newsserver auf einem der (als sicher angenommen) Unix-Rechner zu betreiben, oder ein News-Gateway analog zum WWW-Proxy zu betreiben. Beides sind ansonsten überflüssige Massnahmen, solange das Datenaufkommen gering ist.

Wegen dieser Einschränkungen sind Paketfilter, die nur nach IP-Nummern filtern, nicht mehr sehr verbreitet. Gängige Produkte ziehen mindestens noch die Protokollnummern aus dem IP-Header sowie die Portnummern aus dem TCP/UDP Header hinzu. In Anlehnung an die Konfigurationssprache der **sf Firewall Software** [Muchsel 96], die im SINUS-Projekt eingesetzt wird, können die Anforderungen aus obigem Beispiel folgendermassen gelöst werden:

```
    accept all from 193.75.101.2...193.75.101.4;
    accept all to 193.75.101.2...193.75.101.4;
    accept tcp from 193.75.101.0 mask 255.255.255.0 to port
119;
       accept tcp from port 119 to 93.75.101.0 mask
255.255.255.0;
```

```
block all;
```

Es werden also alle Pakete der beiden Server in beide Richtungen durchgelassen. Für die anderen Maschinen werden nur Pakete akzeptiert, deren Inhalt ein TCP-Paket ist und die an den NNTP-Port einer Maschine ausserhalb des lokalen Netzes gerichtet sind oder von dort kommen. Die fünfte Zeile regelt, was mit allen anderen Paketen gemacht werden soll, für die keine der Regeln greift. Es wird also vorausgesetzt, dass die erste passende Regel angewandt wird.

Filterung nach IP-Nummern, Protokollnummer und Portnummern ist die minimale Funktionalität, die man von einem Paketfilter erwarten kann. Im folgenden sind noch einige Verbesserungen beschrieben, die eine Software zusätzlich bieten kann:

Filterung nach TCP-Verbindungsaufbaurichtung. In obigem Regelsatz gibt es je eine Regel für beide Richtungen, in denen Pakete fliessen, wenn ein Benutzer eines der Arbeitsplatzrechner auf einen externen Newsserver zugreift. Dabei ist der Port des Servers spezifiziert, nicht jedoch der des eigenen Rechners, der ja von Verbindung zu Verbindung unterschiedlich ist. Obige Regeln haben nun den Nachteil, dass von jedem Rechner ausserhalb des lokalen Netzes eine Verbindung von Port 119 zu jedem beliebigen Port eines Rechners im inneren Netz aufgemacht werden kann. Sinnvoller ist es, einen TCP-Verbindungsaufbau nur in einer definierten Richtung zu erlauben, innerhalb der Verbindung dann aber beide Richtungen zuzulassen. Diese Vorgehensweise setzt eine Speicherung bestehender TCP-Verbindungen voraus und wird von einigen Paketfiltern angeboten. Auch die sf Firewall Software, die im SINUS-Projekt verwendet wird, besitzt diese Fähigkeit.

Filterung nach Benutzern. Bei einem solchen Paketfilter können im Bedingungsteil einer Regel neben den TCP/IP Verbindungsdaten auch Angaben über die Benutzerkennungen gemacht werden, für die der Ausführungsteil der Regel gelten soll. Beispielsweise kann eine Regel, die den Benutzern `hinz` und `kunz` den Zugriff auf das World Wide Web (genauer: den HTTP-Port) erlauben soll, lauten:

```
accept tcp from 193.75.101.0 users hinz, kunz to port 80;
```

Informationen über die Benutzernamen ist weder in den Headern der Schichten II und III, noch in den Nutzdaten der HTTP-Pakete enthalten. Die Informationen über Benutzer können über das *ident*-Protokoll gewonnen werden [St. Johns 93, Weidner 97]. Dazu

muss auf dem Rechner, auf dem der HTTP-Client läuft, ein *ident*-Server (*identd*) installiert sein, der auf Anfrage den Loginnamen des Benutzers, der eine TCP-Verbindung geöffnet hat, mitteilt. Nach aussen (d.h. an den WWW-Server) sollen diese Informationen natürlich nicht weitergegeben werden.

Filterung nach Datenmenge. Eine verbreitete Form eines Netzwerkangriffes ist der denial-of-service Angriff, siehe Abschnitt 3.12. Bei ihm werden Verbindungswege genutzt, die normalerweise erlaubt und harmlos sind, jedoch mit so grossen Datenpaketen, dass der Server, der Firewall oder die Netzanbindung der Organisation dadurch lahmgelegt wird. Ein bekanntes Beispiel ist die Mailbombe, bei der einem Empfänger sehr grosse E-Mails geschickt werden, so dass die Mailplatte überläuft und keine weiteren (sinnvollen) E-Mails mehr angenommen werden können. Das Problem kann nicht gelöst, wohl aber vermindert werden, wenn der Paketfilter die Angabe einer maximalen Datenmenge pro TCP-Verbindung oder pro IP-/Portnummer in einer bestimmten Zeiteinheit (z.B. einer Stunde) des absendenden Rechners erlaubt.

Filterung der TCP-/UDP-Nutzdaten spezieller Dienste. Bei manchen Diensten reicht eine Filterung nach IP- und Portnummern nicht aus, um einen sinnvollen Betrieb zu ermöglichen. Ein bekanntes Beispiel ist FTP (siehe Abschnitt 4.2). Das Problem wird gelöst, indem der Paketfilter die Daten, die auf dem FTP-Kommandokanal übertragen werden, mitliest. Taucht ein PORT-Kommando auf, interpretiert er die übertragene Portnummer und erlaubt es dem Server, einmalig die angegebene FTP-Datenverbindung zu öffnen. Ein andere Anwendung der Nutzdatenfilterung ist es, bestimmte Inhalte zu sperren, etwa Java-Applets oder den Zugriff auf CGI-Skripten. Hier kann man beliebig komplizierte Heuristiken anwenden, um unerwünschte Nutzdaten innerhalb der Pakete auszusortieren. Jedoch sind durch den Aufwand an CPU-Zeit Grenzen gesetzt.

Dynamische Regeln. Wie bei der Filterung nach Datenmenge erwähnt, kann es sinnvoll sein, bestimmten Rechnern ihre Verbindungsrechte zeitweilig zu entziehen, wenn bestimmte zusätzliche Bedingungen auftreten. Beispielsweise ist es sinnvoll, Rechnern, die zu viele Daten übertragen oder zu viele Verbindungen innerhalb einer kurzen Zeitspanne aufbauen, für eine Weile auszusperren. Dazu dienen dynamische Regeln, die zu den bisherigen (statischen) hinzugefügt werden und nach einiger Zeit wieder gelöscht werden können, beispielsweise bei Eintreten anderer Bedingungen, durch Eingriff des Systemadministrators oder automatisch nach Ablauf einer festgesetzten Zeitspanne.

Ersetzung von IP-Nummern. Die meisten Paketfilter akzeptieren oder blockieren IP-Pakete, verändern sie aber nicht. Daher trägt jedes nach aussen weitergeleitete Datagramm die IP-Nummer des absendenden Rechners im inneren Netz. Dies wird häufig als Sicherheitsrisiko angesehen; es ermöglicht Rückschlüsse über die interne Struktur des Firmennetzes und erlaubt eine Verfolgung der Aktivitäten einzelner Rechner. Um dies zu vermeiden, bieten einige Produkte die Möglichkeit, IP-Nummern des inneren Netzes durch eine feste IP-Nummer (meist die des Paketfilters oder des Routers) zu ersetzen. Dabei muss im allgemeinen auch die dazugehörige Portnummer ersetzt werden, um doppelte Sockets zu vermeiden. Bei einkommenden Pakete an diese Sockets werden IP- und Portnummer wieder zurückgesetzt, bevor sie an den Rechner weitergeleitet werden, für den sie bestimmt sind. Somit ist nur eine IP-Nummer des gesamten Unternehmensnetzes im Internet bekannt.

Audit/Alarmierung. Treten bestimmte, unerwünschte Dinge ein, so ist es sinnvoll, dies dem Systemadministrator mitzuteilen. Dies sollte je nach Schwere des Vorfalles mit unterschiedlichen Massnahmen geschehen, z.B. durch einen Eintrag im Logfile, eine E-Mail an den Administrator oder sogar dem Auslösen akustischer oder optischer Signale.

Protokollierung. Es sollte die Möglichkeit bestehen, Informationen über angenommene und abgewiesene Verbindungsanforderungen sonstige Betriebsdaten der Firewall dauerhaft zu speichern. Dabei stellt sich die Frage nach der Fälschbarkeit, wenn diese Daten als Beweismittel herhalten sollen. Jüngere Forschungsergebnisse der Kryptographie bieten die Erstellung eines ewigen Logfiles, bei dem in regelmässigen Zeitabständen Hashsummen über das File bei einer Vetrauensinstanz (z.B. einem Notar) hinterlegt werden [Anderson 96], ein anerkannter juristischer Beweis ist dies jedoch nicht. Eine andere, teurere Möglichkeit ist, die Logfiles auf einer einmal beschreibbaren CD zu speichern und diese zu hinterlegen.

6.3 Application Gateways

Die meisten kommerziellen Firewall-Produkte fallen in die Klasse der Application Gateways. Das Grundprinzip besteht darin, dass zwischen dem äusseren und dem inneren Netzwerk keine IP-Verbindung besteht. Statt dessen existieren Rechner, die mit beiden Netzen verbunden sind und für die gewünschten Dienste Gateways bereitstellen.

Ein Zugriff von innen auf einen äusseren Rechner kann nicht direkt gemacht werden. Statt dessen wird eine Anfrage an das Gateway gemacht, dass seinerseits den Zugriff auf den äusseren Rechner durchführt und das Ergebnis an den inneren weiterleitet. Das Prinzip ist ähnlich wie beim IP-Routing über einen Paketfilter. Der Unterschied besteht aber in der Ebene, auf der die Weiterleitung passieren soll, und der dadurch verfügbaren Informationen für die Filterregeln.

Der grosse Vorteil von Application Gateways ist die Sicht auf die Daten, die sie haben. Im Gegensatz zu Paketfiltern sehen Application Gateways den kontinuierlichen Datenstrom einer TCP-Verbindung. Gegenüber den beiden miteinander kommunizierenden Anwendungen treten sie jeweils als das andere Ende auf. Dadurch sehen sie nicht nur Verbindungs- und Kommunikationsparameter, sondern auch den Inhalt der übertragenen Daten.

Nachteilig ist dagegen, dass es für jeden Dienst einen eigenen Proxy geben muss. Beispielsweise muss ein Proxy für einkommende E-Mail sich nach aussen hin so verhalten wie der MTA der Organisation. Er muss also insbesondere alle Optionen des SMTP-Protokolles [Postel 82] kennen. Dennoch soll der Proxy nicht zu komplex werden; die üblichen Mailrouting- und -verarbeitungsfunktionen braucht er nicht zu besitzen.

Dabei tritt ein Problem auf: Die Client-Software muss wissen, dass sie keine direkte Verbindung zu einem gewünschten Server öffnen kann, sondern das Gateway verwenden muss. Bei manchen Diensten ist dies zumindest mit neuerer Software unproblematisch, da die Verwendung von Gateways (die auch als Proxy bezeichnet werden) in der Dienstspezifikation bereits vorgesehen ist.

Ein klassisches Beispiel ist E-Mail. Mail wird im Internet meist über das Simple Mail Transfer Protocol (SMTP) transportiert. Innerhalb der Organisation nehmen Mail Transport Agents (MTA) die Mails entgegen und stellen sie den Mail User Agents (MUA) zum Abholen zur Verfügung (siehe Abschnitt 4.3). Anstelle des eigentlichen MTA kann ein Applikation Gateway für E-Mail nach aussen bekannt gemacht werden. Es nimmt einkommende Mail entgegen und leitet sie nach einigen Ueberprüfungen an den MTA weiter. Somit kommt der MTA nie direkt mit anderen Hosts im Internet in Berührung. Für ausgehende E-Mail kann auf ähnliche Weise ein Zwischenschritt eingefügt werden.

HTTP besitzt von Hause aus die Möglichkeit, Proxy-Server zu verwenden. Gedacht war das ursprünglich dazu, dass der Proxy auch als Zwischenspeicher (Cache) für häufig gelesene Dokumente fungiert, diese also nicht jedesmal über das Netz geholt werden

müssen. FTP, **gopher** und **WAIS** (zwei ältere Protokolle für verteilte Informationssysteme) sind im Umweg über eine HTTP-Verbindung ebenfalls Proxy-tauglich.

Telnet (und auch die r-Tools) sind nicht unmittelbar Proxy-tauglich. Um einem Benutzer des internen Netzwerkes die Möglichkeit zu geben, sich auf einem äusseren Rechner einzuloggen, gibt es zwei Wege:

Auf dem Firewall-Rechner läuft ein spezieller *telnetd*, der Proxy-Funktionen übernimmt. Der Benutzer muss sich zunächst auf diesem Rechner einloggen; von dort kann er ein Login auf einen äusseren Rechner starten. Der Rechner, über den die Verbindung gehen muss, wird auch **Bastion Host** genannt. Diese Vorgehensweise hat den Nachteil, dass sich aus Benutzersicht die Bedienung ändert: Verbindungen über Bastion Hosts sind **nicht transparent**. Für einen menschlichen Benutzer ist das nicht weiter schlimm; ein automatischer Loginvorgang aus einem anderen Programm heraus kann dagegen möglicherweise nicht mehr funktionieren. Zum Beispiel kann eine WWW-Seite eine Telnet-URL enthalten; wenn der Benutzer sie anwählt, wird er jedoch keine Verbindung bekommen, da direkte Telnet-Verbindungen über den Bastion Host nicht mehr funktionieren.

Um die Bedienung zu vereinfachen, kann die Client-Software im inneren Netz so verändert werden, dass sie Zugriffe an einen äusseren Rechner automatisch an den Bastion Host umleitet. Meist geschieht das durch Veränderung bzw. Einfügung der Programmbibliotheken, über die ein Anwendungsprogramm (in diesem Fall der Telnet-Client) mit der TCP/IP-Implementierung des Betriebssystems kommuniziert. Der Vorteil dieser Methode ist die Transparenz aus Sicht der Anwendung bzw. des Benutzers. Nachteilig ist aber, dass an allen Programmen, Bibliotheken und/oder Betriebssystemen der Rechner im inneren Netz Veränderungen vorgenommen werden müssen. Dies ist zeitaufwendig und eine zusätzliche Fehlerquelle. Bei Systemen, deren Quelltext nicht öffentlich verfügbar ist, ist es manchmal sogar überhaupt nicht möglich.

6.4 Konzeption von Firewall-Lösungen

Wie bereits in der Einleitung erwähnt, werden Firewalls eingesetzt, um einen gesicherten Uebergang zwischen zwei Netzen unterschiedlicher Vertrauenswürdigkeit miteinander zu koppeln. Bei der Planung des Firewalleinsatzes sollte daher grundsätzlich die Frage im

Vordergrund stehen, welche schützenswerten Güter in einem Netz zur Verfügung stehen sollen, die gegen Uebergriffe aus einem anderen Netz gesichert werden müssen.

Zunächst einmal ist das gesamte interne Netz gegenüber dem restlichen Internet schützenswert. Wie in Abschnitt 3.12 beschrieben wurde, gibt es Möglichkeiten, Rechner oder Netzwerkkomponenten unter Ausnutzung von Protokollschwächen oder Implementierungsfehlern lahmzulegen. Darüber hinaus können Konfigurations- oder Bedienungsfehler weitere Lücken öffnen. Angriffe dieser Art können nicht pauschal verhindert werden. Das Risiko kann dadurch minimiert werden, dass am zentralen Anschlusspunkt des internen Netzes an das Internet nur diejenigen Dienste und Protokolle zugelassen werden, die für den geplanten Einsatzzweck benötigt werden.

Von den beiden Strategien

- “Erlaube alle Verbindungen ausser denen, die explizit verboten sind”
- “Verbiete alle Verbindungen ausser den explizit erlaubten”

ist also im Normalfall die erste vorzuziehen.

Die beiden oben vorgestellten Firewall-Typen, Paketfilter und Application Gateway, haben unterschiedliche Vor- und Nachteile. Daher ist es oft sinnvoll, am Verbindungspunkt zweier Netze beide Typen einzusetzen.

Application Gateways sind für Dienste vorzuziehen, die

- für den Betrieb mit Proxies geeignet sind (E-Mail, WWW, FTP, DNS)
- wegen Sicherheitsschwächen in den Applikationen eine Filterung der Nutzdaten erforderlich machen (E-Mail, WWW)
- von verschiedenen Benutzern im inneren Netz mit unterschiedlichen Privilegien genutzt werden können sollen und daher eine zusätzliche Authentisierungsstufe benötigen (Telnet, FTP)
- nur einen Teil ihrer üblichen Funktionalität nach aussen erlauben dürfen (z.B. nur Lese- und keine Schreibzugriffe auf NNTP-Server)
- neben Sicherheitsaspekten auch andere Vorteile (z.B. Lastverminderung oder Accounting) durch den Einsatz von Proxies erfahren (WWW, NNTP)

- Die Architektur des inneren Netzes nach aussen nicht sichtbar sein soll (z.B. durch Verwendung eigener, im Internet unbekannter IP-Nummern)

Paketfilter sind von Vorteil, wenn

- der Zugriff auf bestimmte Dienste über den Anschlusspunkt hinweg gar nicht erlaubt werden soll, weil diese Dienste unnötig sind oder schwere Sicherheitslücken enthalten (z.B. X11, NTP, TFTP, *bootp*, *lpd*)
- einzelne Organisationen auf einfache Weise von der Nutzung eines Dienstes ausgeschlossen werden sollen (z.B. Massenmail-Versender oder WWW-Suchmaschinen)
- Zugriffe von aussen nur auf bestimmte Rechner im inneren Netz erfolgen dürfen (SMTP-Server, WWW-Server, DNS-Server)
- ein Verbindungsversuch an einen bestimmten Dienst eine Warnung oder einen Alarm auslösen sollen (NTP, TFTP)
- Tools, mit denen Netzwerke nach Schwachstellen durchsucht werden können, erkannt werden sollen
- spezielle Optionen der Internet- und Transportschicht abgelehnt werden sollen (z.B. Source-Routing)

Beispielsweise kann der Zugriff auf den Mailserver der Organisation so gestaltet werden, dass zunächst ein Paketfilter SMTP-Verbindungen auf alle anderen Rechner ausser den SMTP-Proxy abweist und der Proxy die Mails entgegennimmt, prüft und an den eigentlichen Mailserver weiterleitet.

Analog dazu müssen ähnliche Ueberlegungen für die Gestaltung der Netzverbindungen im inneren Netz getroffen werden. Der Einsatz eines Firewalls ist immer dort notwendig, wo Netze unterschiedlicher Sicherheitsanforderungen innerhalb des Firmennetzes gekoppelt werden sollen, beispielsweise zwischen Netzen mit getrennter Administrationsverantwortung, Rechnerclustern, die sich gegenseitig über ihre IP-Nummern authentifizieren (müssen), konkurrierenden Abteilungen innerhalb der Organisation, oder auch zwischen einem Netz mit hohen Zuverlässigkeitsanforderungen und einem mit experimenteller Netzwerksoftware.

Weitere Hinweise zu Konzeption und Einsatz von Firewalls finden sich in [Cheswick 94, Chapman 95].

7 Zusammenfassung und Ausblick

In dieser Arbeit wurden Sicherheitsaspekte behandelt, die beim Anschluss eines organisationsinternen Netzes an das Internet zu berücksichtigen sind.

Probleme ergeben sich dadurch, dass die im Internet verwendete TCP/IP-Protokollfamilie ursprünglich als ungesichertes Netzwerkprotokoll zu Forschungszwecken entwickelt wurde. Auf Netzwerk- und Transportebene sind keine Mechanismen vorgesehen, die die Daten vor unbefugtem Einsehen oder vor absichtlicher Veränderung schützen. Absenderkennungen werden nicht überprüft und sind daher fälschbar. Dennoch verlassen sich viele Anwendungen auf vermeintliche Sicherheitseigenschaften und transportieren sicherheitsrelevante Daten wie Passwörter direkt über das Netz oder gewähren anderen Rechnern Privilegien alleine aufgrund der Absenderkennung.

Ein weiteres Problemfeld sind Programmierfehler in System- oder Netzwerksoftware. Unter Ausnutzung solcher Fehler kann ein Angreifer häufig Zugang zu sensitiven Daten bekommen, Kommandos auf fremden Rechnern ausführen oder Systemsoftware zum Absturz bringen.

Ein Hilfsmittel zur Absicherung eines organisationsinternen Netzwerkes gegenüber dem Internet ist der Firewall. Er ermöglicht die kontrollierte Verbindung zweier unterschiedlich vertrauenswürdiger Netze. Es gibt zwei Grundtypen von Firewalls, Paketfilter und Application Gateways. Sie arbeiten auf verschiedenen Netzwerkebenen und haben eine unterschiedliche Sicht auf die zu filternden Daten. Daraus ergeben sich Vor- und Nachteile für bestimmte Einsatzgebiete; es liegt nahe, beide Techniken zu koppeln. Das Aufsetzen einer Firewall-Anbindung erfordert bei den gängigen Produkten Expertenwissen. Ein falsch konfigurierter Firewall kann die Funktionsfähigkeit der Netzanbindung beeinträchtigen oder zusätzliche Sicherheitslücken öffnen.

Firewalls können das innere Netz vor dem äusseren beschützen oder umgekehrt, jedoch nicht einzelne Rechner oder Benutzer auf der gleichen Seite des Netzes voreinander. Der Einsatz eines Firewalls darf daher nicht dazu führen, die Sicherheit des internen Netzes zu vernachlässigen. Zur Verwaltung von Rechnerclustern mit vielen Benutzern gibt es komfortable Hilfsmittel, die jedoch Sicherheitsdefizite aufweisen und daher nicht in jeder Umgebung zu verwenden sind.

Sichere Kommunikation über ein unsicheres Netzwerk kann mit kryptographischen Techniken und Protokollen erreicht werden. Verschlüsselung bewahrt die Vertraulichkeit, Signierung die Integrität und Authentizität der Daten bei der Uebertragung. Die Schwierigkeit, vorher einen Schlüssel vereinbaren zu müssen, kann durch Public-Key-Kryptosysteme gemildert, jedoch nicht ganz ausgeräumt werden. Obwohl gute kryptographische Verfahren zur Verfügung stehen, werden sie von den gängigen Betriebssystemen und Anwendungsprogrammen kaum genutzt. Neben fehlendem Sicherheitsbewusstsein bei den Anwendern ist dies vor allem auf mässige Verfügbarkeit, Schwierigkeiten bei der Bedienung und rechtliche Unsicherheiten zurückzuführen.

Sicherheit im Internet wird in Zukunft einen hohen Stellenwert bekommen. Immer mehr Unternehmen schliessen sich an das Netz an, bieten Informationsangebote, kommunizieren mit Kunden und Vertragspartnern und treffen Unternehmensentscheidungen anhand von Daten, die aus dem Netz gewonnen wurden. Die Aufgaben des Sicherheitsmanagementes, Schaffung von Sicherheitsbewusstsein und technischem Know-How bei Angestellten und Führungskräften, wird sich daher in Zukunft zu einem wichtigen Bestandteil der Unternehmenskultur entwickeln.

Literatur

[Anderson 96] R. Anderson: The Eternity Service. Cambridge University Computer Laboratory, 1996

[ANSI 81] ANSI X3.92: American National Standard for Information Systems - Data Link Encryption. American National Standards Institute, 1981

[Bauer 94] F. L. Bauer: Kryptologie: Methoden und Maximen. Springer, 2. Auflage, 1994

[Bauknecht 95] K.Bauknecht, R. Hauser, R. Holbein, D. von Stockar: Sicherheit in der Informationstechnik. Universität Zürich, Institut für Informatik, Vorlesungsunterlagen SS 1995

[Bellovin 89] S. Bellovin: Security Problems in the TCP/IP Protocol Suite. Computer Communications Review, v. 19, n. 2, S. 32-48, 1989

[Beth 88] Th. Beth: Efficient Zero-Knowledge Identification Scheme for Smart Cards. EUROCRYPT'88 Proceedings, Springer 1988, S. 77-84

[Blum 82] M. Blum: Coin Flipping by Telephone. A Protocol for Solving Impossible Problems. Proceedings of the 24th IEEE Computer Conference, 1982, S. 133-137

[Bressoud 89] D. Bressoud: Factorization and Primality Testing. Springer, 1989

[Campbell 92] K. Campbell, M. Wiener: DES is not a group. CRYPTO ‘92 Proceedings, 1992, S. 512-520

[CERT 96-18] CERT Vendor-Initiated Bulletin VB-96.18 - Sun Microsystems, Inc.

[CERT 96-21] CERT Advisory CA-96.21: TCP SYN Flooding and IP Spoofing Attacks.

[CERT 96-26] CERT Advisory CA-96.26: Denial-of-Service Attack via ping.

[Chang 91] C. Chang, S. Hwang: Cryptographic Authentication of Passwords. Proceedings of the 25th Annual 1991 IEEE International Carnahan Conference on Security Technology, Taipei, Taiwan, 1991, S. 126-130

[Chapman 95] D. Chapman, E. Zwicky: Building Internet Firewalls. O'Reilly & Associated, 1995

[Chaum 90] D. Chaum, A. Fiat, M. Naor: Untraceable Electronic Cash. CRYPTO'88 Proceedings, Springer, 1990, S. 319-327

[Cheswick 94] W. Cheswick, S. Bellovin: Firewalls and Internet Security - Repelling the Wily Hacker. Addison-Wesley, 1994

[Deering 96] S. Deering, R. Hinden: Internet Protocol, Version 6 (IPv6) Specification. RFC 1883, 1996, `ftp.switch.ch:/pub/docs/rfc/`

[Diffie 77] W. Diffie, M. Hellmann: Exhaustive Cryptoanalysis of the NBS Data Encryption Standard. IEEE Computer, v. 10, n. 6, 1977, S. 74-78

[Dobbertin 96] H. Dobbertin: The Status of MD5 After a Recent Attack. Crypto Bytes, technical Newsletter of RSA Laboratories, Vol. 2, Nr. 2, 1996, S. 1-6

[ElGamal 84] T. ElGamal: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. CRYPTO'84 Proceedings, Springer, 1985, S. 10-18

[ElGamal 85] T. ElGamal: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Transactions on Information Theory, v. IT-31, n. 4, 1985, S. 469-472

[Hellmann 80] M. Hellmann. W. Diffie, R. Merkle: Public Key Cryptographic Apparatus and Method. U.S. Patent #4,200,770, 1980

[Holbein 96] R. Holbein: Secure Information Exchange in Organisations; An approach for solving the information misuse problem. Dissertation, Universität Zürich, Institut für Informatik, Shaker Verlag, 1996

[Hunt 92] C. Hunt: TCP Network Administration. O'Reilly Int. Thomson Verlag, 1992

[Jaburek 90] W. Jaburek: A Generalisation of ElGamal's Kryptosystem. EUROCRYPT '98 Proceedings, 1990, Springer Verlag, S. 23-28[McCurley 90] K. McCurley: The discrete Logarithm Problem. Proceedings of the Symposium on Applied Mathmatics, American Mathematics Society, 1990, S. 49-74

[Kersten 91] H. Kersten: Einführung in die Computersicherheit. Reihe IT-Sicherheit, Oldenbourg Verlag 1991

[Lamport 82] L. Lamport, R. Shostak, M. Pease: The Byzantine General Problem. ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, 1982, S. 382-401

[Lubich 96] H. Lubich: Von Mäusen und Elefanten - Versuch einer Beurteilung heutiger Betriebssysteme. Informatik/Informatique 5/1996, S. 6-9

[McCurley 90] K. McCurley. The Discrete Logarithm Problem. Cryptography and Computational Number Theory, Proceedings of the Symposium on Applied Mathematics, American Mathematics Society, 1990, S. 49-74

[Muchsel 96] R. Muchsel, R. Schmid: Der 80:20 Firewall - Konzeption und Entwicklung eines “Packet Filters”. In: Proceedings der SIS’96, 1996, S. 327-336

[Pfleger 89] C. Pfleeger: Security in Computing. Englewood Cliffs, New York, Prentice Hall, 1989

[Postel 82] J. Postel: Simple Mail Transfer Protocol. RFC 821, 1982

[Rabin 79] M. Rabin: Digital Signatures and Public-Key-Functions as Intractable as Factorization. MIT Lab. for Computer Science, Technical Report, Jan 1979

[Reiter 95] M. Reiter, M. Franklin: The Design and Implementation of a Secure Auction Service. Proceedings of the 1995 IEEE Symposium on Security and Privacy, 1995

[Rivest 78] R. Rivest, A. Shamir, L. Adleman: A Method for Obtaining Digital Signatures and Public Key Cryptosystems. Communications of the ACM, v. 21, n. 2, 1978, S. 120-126

[Rivest 92] R. Rivest: The MD5 Message Digest Algorithm. RFC 1321, 1992

[Schneier 96] B. Schneier: Applied Cryptography. John Wiley & Sons, New York, 1996

[Simmons 92] G. Simmons: An Introduction to Shared Secret and/or Shared Control Schemes and Their Application. Contemporary Cryptology: The Science of Information Integrity, IEEE Press, 1992, S. 441-497

[Stallings 95] W. Stallings: Network and Internetwork Security: Principles and Practice. Englewood Cliffs, 1995

[St. Johns 93] M. St. Johns: Identification Protocol. RFC 1413, 1993

[Tanenbaum 96] A. Tanenbaum: Computer Networks. 3. Auflage, Prentice Hall, New Jersey, 1996

[Teufel 95] S. Teufel, U. Zurfluh, H. Lubich: Sichere Nutzung von Online-Diensten (SINUS). Bewilligtes Forschungsgesuch des Schwerpunktprogrammes Informatikforschung SPP-IuK, Modul: Information und Kommunikation, Nr. 5003-45309, 1995

[Weidner 97] H. Weidner: Benutzeridentifizierung in Mehrbenutzersystemen. APS (Arbeitsplatz-Rechensysteme) ‘97, Koblenz, 1997.