Fachbereich Informatik der Universität Hamburg

Vogt-Kölln-Straße 30 ♢ D-22527 Hamburg / Germany

University of Hamburg – Computer Science Department

Bericht Nr. 224/00 • Report No 224/00

Sicherheit in vernetzten Systemen

Seminar Sommersemester 1999

Dr. Hans-Joachim Mück, Carsten Benecke, Stefan Kelm (Hrsg.)

FBI-HH-224/00

Februar 2000

In die Reihe der Berichte des Fachbereichs Informatik aufgenommen durch

Accepted for Publication in the Report Series of the Department of Computer Science by

Prof. Dr. Klaus Brunnstein und Prof. Dr. Karl Kaiser

Zusammenfassung

Der vorliegende Band enthält die schriftlichen Ausarbeitungen des Seminars „Sicherheit in vernetzten Systemen“, welches im Sommersemester 1999 am Fachbereich Informatik der Universität Hamburg durchgeführt wurde. Im Rahmen dieses Seminars wurden verschiedene Sicherheitsrisiken analysiert sowie bestimmte Lösungen vorgestellt und aus informationstechnischer Sicht bewertet. Die Teilnehmer erhielten einen Einblick in aktuelle Aspekte der Rechner- und Netzwerksicherheit; die erarbeiteten Lösungsansätze wurden kritisch diskutiert, um Möglichkeiten und Grenzen im Bereich der IT-Sicherheit zu vermitteln. Die einzelnen Thematiken bildeten einen roten Faden und bauten somit aufeinander auf: von allgemeinen Problemen der Netzwerksicherheit über den Einsatz von Firewalls und kryptographischen Verfahren bis hin zu speziellen Fragestellungen wie Schlüsselmanagement und mobilem Code wurden essentielle Inhalte bearbeitet.

Abstract

This publication contains the written contributions of the students who participated in the seminar “Security in Networked Systems,” which took place during the Summer semester 1999 at the Computer Science Dept. of the University of Hamburg. The students held lectures covering a number of important security related subjects; after the problems were described and analyzed, solutions were introduced which were subsequently evaluated in a discussion session at the end of each lecture. Not only the possibilities, but also the limitations of IT security were thus illuminated. The topics discussed covered general security problems, traditional subjects such as firewalls and cryptographic techniques, as well as current issues of particular interest including key management and mobile code. This report therefore offers an overview of many important aspects in the field of IT security.

Vorwort

Die Motivation zur Durchführung dieses Seminars entstand aus der langjährigen Projektarbeit der Drittmittelprojekte DFN-CERT, DFN-PCA und DFN-FWL (`http://www.cert.dfn.de/`), welche am FB Informatik angesiedelt sind. Das Ziel des Seminars war die Vermittlung hoch aktueller Fragestellungen aus den Bereichen der Rechner- und Netzwerksicherheit.

Rechnernetze sind unsicher. Angefangen bei „einfachen“ Host-Rechnern innerhalb eines LANs über VPNs (Virtual Private Networks) bis zu weltweit verteilten Firmennetzen gibt es unterschiedlichste Sicherheitsrisiken und Angriffsmöglichkeiten. Dies ist insbesondere aus der Sicht einer zunehmenden Kommerzialisierung (Stichwort: Electronic Commerce) und den ständig steigenden Anschlußzahlen im Internet problematisch. In dem Seminar „Sicherheit in vernetzten Systemen“, das im Sommersemester 1999 am Fachbereich Informatik der Universität Hamburg abgehalten wurde, wurden Sicherheitsrisiken analysiert, bestimmte Lösungen vorgestellt und aus informationstechnischer Sicht kritisch bewertet. Die Teilnehmer erhielten somit einen Einblick in aktuelle Aspekte der Rechner- und Netzwerksicherheit.

Die Themenwahl war von den Veranstaltern vorgegeben. Die einzelnen Beiträge bildeten dabei einen roten Faden, beginnend von allgemeinen Problemen bis hin zu ausgewählten, spezielleren Fragestellungen. Von den Teilnehmern wurde eine überdurchschnittliche Leistung erwartet, die deutlich über den üblichen Anforderungen an ein Hochschulseminar lag. Das Ausbildungsziel dieses Seminars bestand darin, sich in ein ausgewähltes Gebiet der Netzwerksicherheit einzuarbeiten, den Umgang mit wissenschaftlicher Literatur zu vertiefen, die Ergebnisse anschließend in ansprechender Form vorzutragen und schriftlich zu dokumentieren. Die Ursachen für die unterschiedliche Qualität der einzelnen Beiträge liegt zum einen in dem unterschiedlichen Vorwissen der Teilnehmer an der Veranstaltung, zum anderen daran, daß die Teilnehmer an die Ausarbeitung und die von den Veranstaltern gemachten Betreuungsangebote äußerst unterschiedlich herangegangen sind.

Die intensive Betreuung der Veranstalter bestand in ausführlichen Vorgesprächen mit den Vortragenden, um diese mit den ausgewählten Themen und den Lernzielen vertraut zu machen, in zahlreichen Diskussionen während der einzelnen Vorträge, in Nachbesprechungen mit den Vortragenden sowie bei der Erstellung der hier zusammengefaßten Beiträgen. Dieser sehr hohe Betreuungsaufwand erlaubt leider keine regelmäßige Durchführung des Seminars; um so wichtiger erscheint es uns daher, die erarbeiteten Ergebnisse in dieser Form schriftlich festzuhalten.

Als Veranstalter möchten wir uns bei allen Teilnehmern an diesem Seminar herzlichst bedanken. Ihre aktive Teilnahme am Seminar hat zu dessen hoher Qualität beigetragen!

Dr. Hans-Joachim Mück (DFN-CERT)
Carsten Benecke (DFN-FWL)
Stefan Kelm (DFN-PCA)

Termine / Themen

Seminar 18.416: Sicherheit in vernetzten Systemen, SS 1999

Inhaltsverzeichnis

Kapitel 1

Grundlagen: Internet-Protokolle

Klaus Möller
DFN-CERT GmbH

Zusammenfassung

Der folgende Artikel gibt eine kurze Einführung in die Protokolle IP, TCP und UDP, welche die Grundlage der TCP/IP Protokoll-Suite bilden. Es werden die Datenformate und die Bedeutung der einzelnen Felder vorgestellt. Begleitend dazu werden die dazugehörenden Programmierschnittstellen unter UNIX dargestellt. Den Abschluß bildet eine kurze Einführung in die Verwendung von symbolischen Namen, speziell DNS bei TCP/IP.

1.1 Einleitung

In diesem Artikel soll eine Einführung in die TCP/IP Protokoll-Suite gegeben werden, wobei besonders auf die Programmierschnittstelle (API) der einzelnen Protokolle unter UNIX eingegangen wird.

Zunächst wird im Abschnitt 1.2 der Protokollbegriff erläutert und das OSI-Schichtenmodell vorgestellt. Nach einer kurzen Gegenüberstellung des IP-Protokollstapels und des OSI-Modells sollen die beiden wichtigsten Schichten der TCP/IP-Suite näher vorgestellt werden.

Dabei wird „von unten“ kommend zunächst im Abschnitt 1.3 die Netzwerkschicht mit dem Protokoll IP vorgestellt. Neben dem Paketformat wird das Adressierungskonzept (Abschnitt 1.3.2), die Wegewahl (Abschnitt 1.3.3) und die Fragmentierung von IP-Paketen (Abschnitt 1.3.4) erläutert. In Abschnitt 1.3.5 folgt eine kurze Einführung in das Protokoll ICMP, welches Steuerungsaufgaben im IP-Stack übernimmt.

Es folgen die Protokolle der Transportschicht, zunächst TCP als verbindungsorientiertes Protokoll in Abschnitt 1.4 mit Erläuterung von Verbindungsaufbau und -abbau sowie dem Datenaustausch. Das verbindungslose Protokoll UDP wird in Anschnitt 1.5 eingeführt.

Den Abschluß bildet ein Abriß über die Verwendung symbolischer Namen bei TCP/IP in Abschnitt 1.6 mit dem „Domain Name System“ (DNS) in Abschnitt 1.6.1 als Beispiel.

1.2 Protokollstapel und Schichten

Ein Protokoll ist eine Sammlung von Regeln zur Datenübertragung, die das Verhalten zweier oder mehrerer Kommunikationspartner beschreiben und festlegen. Da verschiedene Anwendungen auch unterschiedliche Anforderungen an die Datenübertragung haben, ergeben sich hierfür sehr komplexe Regeln. Es ist daher nicht sinnvoll, alle Aufgaben in nur einem Protokoll abzuwickeln. Deshalb werden in der Regel mehrere Protokolle gleichzeitig eingesetzt, die in mehreren Schichten angeordnet sind, wobei jede Schicht bestimmte Teilaufgaben erbringt. Jede Schicht nimmt dabei nur die Dienste der unmittelbar darunterliegenden Schicht in Anspruch und bietet ihrerseits nur „höherwertige“ Dienste für die darüber liegende Schicht an.

Um eine einheitliche Betrachtungsweise zu ermöglichen, hat die International Standard Organisation (ISO) ein Referenzmodell entworfen: das sog. „Open System Interconnection“ (OSI) 7-Schichten-Modell. Dieses in Abbildung 1.1 dargestellte Modell dient als Rahmen zur Beschreibung der Protokollcharakteristika und Dienste. Die einzelnen Schichten werden folgendermaßen unterteilt.

Bitübertragungsschicht: Regelt den Austausch einzelner Bits über ein Übertragungsmedium. Hier werden Kodierungsverfahren, physikalische Anschlüsse, usw. normiert.

Sicherungsschicht: Aufgabe dieser Schicht ist die Erkennung von Übertragungsfehlern zwischen direkt angeschlossenen Partnerinstanzen der Schicht 2 und ggfs. die Korrektur dieser Fehler. In dieser Schicht wird auch eine Adressierung der Endgeräte spezifiziert, die jedoch nicht für alle verwendeten Schicht-2 Technologien einheitlich ist und auch nicht mit der Adressierung auf der Vermittlungsschicht identisch sein muß. Bitübertragungs- und Sicherungsschicht sind meist so eng miteinander verknüpft, daß sie zusammen spezifiziert werden, z.B. in Protokollen wie Ethernet oder Token-Ring.

Vermittlungsschicht: Aufgabe dieser Schicht ist die Vermittlung von Dateneinheiten zwischen Endgeräten auf der Grundlage eindeutiger Adressen. Dazu gehört insbesondere die Weiterleitung von Datenpaketen über Vermittlungsknoten im Netz, die die einzelnen Teilnetzwerke auf Schicht 2 miteinander verbinden.

Transportschicht: Übernimmt den Transport von Nachrichten zwischen den eigentlichen Endgeräten. Dazu gehört die Steuerung des Datenflusses, die Sicherung gegen Datenverlust sowie das Aushandeln von Dienstqualitätsparametern (Quality of Service, QoS).

Kommunikationssteuerungsschicht: Aufgabe dieser Schicht ist die Steuerung der Kommunikation zwischen den Kommunikationspartnern. Außerdem können Synchronisationspunkte gesetzt werden, so daß bei einem Fehler des Netzes die Kommunikation beim letzten Synchronisationspunkt wieder aufgesetzt werden kann.

Darstellungsschicht: Aufgabe dieser Schicht ist die Konvertierung zwischen den unterschiedlichen Datenformaten der beteiligten Rechner, z.B. die Konvertierung zwischen verschiedenen Zeichensätzen wie ASCII oder EBCDIC. Dazu wird i.d.R. eine gemeinsame Syntax zur Datenrepräsentation ausgewählt (Transfersyntax) und die Daten werden vor bzw. nach der Übertragung in bzw. aus diese(r) Syntax transformiert.

Anwendungsschicht: Auf dieser Schicht sind die Funktionen realisiert, die die Anwendungen zur Erbringung ihrer Dienste definiert haben, z.B. E-mail, File-Transfer oder Remote-Login.

Im Gegensatz zum OSI-Modell verwendet TCP/IP nicht sieben, sondern nur vier Protokollschichten. Dabei entsprechen die OSI-Schichten 5–7 grob der TCP/IP Schicht 4 (Anwendung). Die Vermittlungs- und Transportschicht (3 und 4) des OSI-Modells haben ihre Entsprechung in der Internet- und Transportschicht (2 und 3) von TCP/IP. Die OSI-Schichten 1 und 2 werden aus Sicht von TCP/IP zusam-

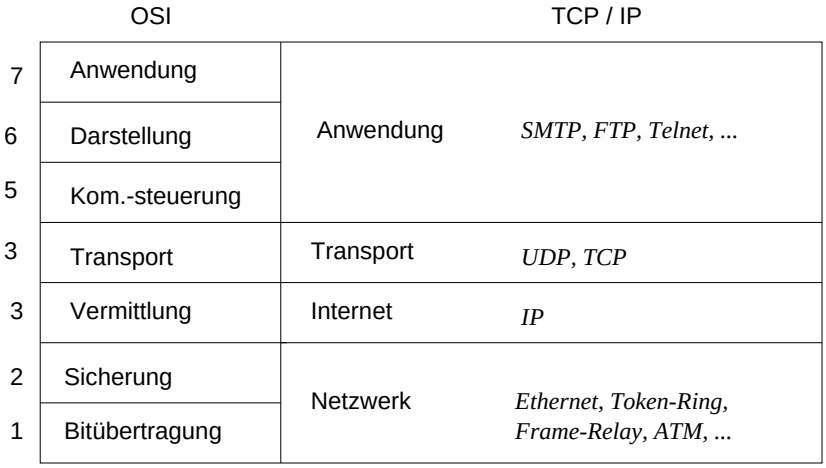


Abbildung 1.1: IP und OSI Schichtenmodell

mengefaßt und als Netzwerkschicht betrachtet. Im weiteren Verlauf dieses Artikels wird die OSI-Terminologie verwendet werden, weil sie allgemein als das Referenzmodell zur Beschreibung von Protokollsuiten angesehen wird.

Obwohl dieses Modell auf den ersten Blick einfacher erscheint, bringt es auch Nachteile mit sich. Viele Anwendungen benötigen gemeinsame Funktionalitäten wie Checkpointing innerhalb einer Sitzung (OSI-Schicht 5), Umwandlung der Datenformate zwischen verschiedenen Recherarchitekturen (OSI-Schicht 6), Remote-Procedure-Call Konventionen (ROSE, OSI-Schicht 7), usw. Im OSI-Modell werden diese Funktionen von den Schichten 5–7 bereitgestellt und können als Teil des Protokollstapels von allen Anwendungen genutzt werden. TCP/IP Anwendungen müssen diese Funktionalität selbst implementieren, was den Entwicklungsaufwand und den Speicherverbrauch erhöht und evtl. Inkompatibilitäten zwischen Anwendungen erzeugen kann. Allerdings werden einige der benötigten Funktionalitäten mittlerweile durch eigene Standards bereitgestellt, speziell im Bereich „Remote Procedure Call“ mit ONC RPC (siehe [RFC 1831, RFC 1832]) oder CORBA (siehe [CORBA]).

Andererseits ist auch Standardisierung nicht nur von Vorteil. Anwendungen, die Funktionalitäten speziell der OSI-Schichten 5 und 6 nicht benötigen, müssen sie trotzdem verwenden, da im OSI-Modell kein direkter Zugriff über mehr als eine Protokollschicht hinweg vorgesehen ist. Ein einheitlicher Satz von Funktionen kann in diesem Fall auch die Benutzung alternativer Ansätze (auf Schicht 5 und 6) verhindern, weil die notwendige Funktionalität nicht bereitgestellt wird. Wird dagegen versucht, möglichst viel Funktionalität im „Standard“ bereitzustellen, kann dies die Implementierung erschweren, evtl. zu hohe Anforderungen an Betriebssystem oder Hardware stellen und zu schlechtem Laufzeitverhalten führen.

1.3 IP

Als Protokoll der Vermittlungsschicht dient IP dazu, von den Besonderheiten der jeweiligen Schicht-2-Protokolle wie Ethernet, Token Ring oder ATM zu abstrahieren. Dazu stellt es eine Ende-zu-Ende Adressierung bereit, d.h. eine einheitliche Adressierung für alle in einem IP-Netz erreichbaren Geräte, unabhängig davon, mit welchem Schicht-2-Protokoll das Gerät tatsächlich angeschlossen ist.

Zwischen den einzelnen Schicht-2-Netzwerken leiten Vermittlungsknoten, sog. Router, die Datenpakete weiter (siehe Abschnitt 1.3.3). Ein wesentlicher Teil der IP-Spezifikation beschreibt die Art der Weiterleitung der Datenpakete vom Absender zum Empfänger.

IP ist ein paketorientiertes und verbindungsloses Protokoll, d.h. ein Rechner kann ohne vorherigen Verbindungsaufbau Daten versenden. Dabei werden diese Daten in Paketen und nicht als Zeichenstrom versendet. Das genaue Gegenteil (verbindungsbasiert und datenstromorientiert) ist z.B. die Übertragung von Telefongesprächen. Den Aufbau eines IP-Paketes erläutert der folgende Abschnitt. Die vollständige Spezifikation befindet sich in [RFC 791].

1.3.1 IP-Paketformat

Ein IP-Paket besteht aus einem Kopf und einem Datenanteil (siehe Abbildung 1.2). Der Kopf enthält alle Informationen zur Verarbeitung des Paketes auf IP-Ebene. Der Datenanteil wird auf IP-Ebene nicht ausgewertet sondern nur an die höheren Protokollschichten weitergereicht.

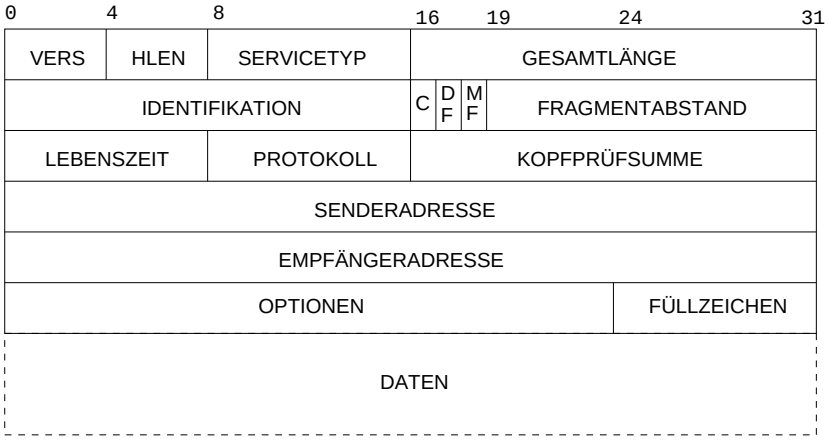


Abbildung 1.2: IP Protokollkopf (aus [Santi 1998, S. 38])

Es gibt mehrere Versionen des IP-Protokolls. Version 4 ist die derzeit aktuelle; die Versionen 1–3 werden praktisch nicht mehr verwendet. Das Versionsfeld dient einem Host dazu, ein Datenpaket einer Protokollversion zuordnen zu können, da die Paketformate der einzelnen Versionen unterschiedlich sind. Damit eine IP-Implementierung erkennen kann, ob es sich um ein Paket einer neueren Version handelt, befindet sich das Versionsfeld in allen IP-Versionen an derselben Stelle, d.h. in den ersten vier Bits des Protokollkopfs.

Die Gesamtlänge eines IP-Paketes (manchmal auch IP-Datagramm genannt) wird durch das gleichnamige Feld angegeben. Sie beinhaltet auch die Länge des IP-Kopfs. Die Länge des Protokollkopfes wird im eigenen Feld Kopflänge (HLEN) angegeben. Mit der Kopfprüfsumme können Übertragungsfehler im IP-Kopf erkannt werden.

Das Lebenszeit-Feld dient dazu, IP-Pakete, die in einer Schleife weitergeleitet werden, nach Ablauf einer endlichen Zeitspanne aus dem Netz zu entfernen. Jeder Router, der ein IP-Paket weiterleitet, ist verpflichtet, diesen Zähler mindestens um eins zu verringern, oder um die Anzahl Sekunden, die das Paket auf dem betreffenden Router verweilt hat. Erreicht der Zähler den Wert Null, wird das Paket nicht mehr weitergeleitet und aus dem Speicher gelöscht. Der Absender wird darüber mit einer

ICMP-Meldung informiert (siehe 1.3.5). De-facto wird aus Implementierungsgründen bzw. aufgrund der heutigen Bearbeitungsgeschwindigkeit der Wert stets um eins verringert, so daß die Bezeichnung dieses Feldes als „Hop-Count“ seine Berechtigung hat, da dieser Wert letztlich bestimmt, über wieviele Router ein IP-Paket weitergeleitet werden kann.

Der Kopf eines IP-Paketes enthält eine rudimentäre Unterstützung für Dienstqualitäten. Die angeforderte Qualität wird im Feld „Servicetyp“ angegeben. Allerdings ist die Unterstützung innerhalb des IP-Kopfes für heutige Bedarfe nicht ausreichend, da sich die Dienstqualität nicht hinreichend genau spezifizieren läßt und von Hosts und Routern im Netz keine Garantie für eine Dienstqualität gegeben werden kann. Hinzu kommt, daß viele IP-Implementierungen dieses Feld nicht auswerten. Somit gibt es de-facto in IP keine allgemeine Unterstützung von Dienstqualitäten.

Im Protokollfeld wird angegeben, welchem Protokoll der Transportschicht der Inhalt des IP-Paketes zur weiteren Bearbeitung übergeben werden muß. So hat beispielsweise TCP die Nummer 17 und UDP die Nummer 6.

Die Felder Identifikation, DF, MF und Fragmentabstand werden im Abschnitt 1.3.4 näher erläutert.

Für besondere Zwecke können im IP-Kopf Optionen angegeben werden. So kann z.B. der Weg eines IP-Paketes durch das Netzwerk mittels sog. „Source Routing Optionen“ genau festgelegt werden. Dazu wird dem IP-Paket im Optionsbereich eine Liste von IP-Adressen mitgegeben, die vom Paket wie bei einem Laufzettel der Reihe nach zu besuchen sind.

Da die Länge des Kopfes aus Implementierungsgründen immer ein Vielfaches von vier Bytes sein muß, wird der Kopf bei der Verwendung von Optionen ggfs. mit Nullbytes auf ein Vielfaches von vier aufgefüllt.

1.3.2 IP-Adressen

Die Felder Sender- und Empfängeradresse (vgl. Abb 1.2) im Protokollkopf dienen zur Identifizierung des Absenders und Empfängers eines IP-Paketes. Eine IP-Adresse stellt eine global eindeutige Identifizierung eines Rechners dar. Sie wird als eine 32-Bit Ganzzahl dargestellt. Die Notation erfolgt überlicherweise als sog. „Dotted-Quad“, d.h. jedes der vier Bytes wird durch Punkte getrennt als Dezimalzahl aufgeschrieben. So schreibt sich z.B. $86640E0F_{hex}$ als 134.100.14.15.

Jedem Betreiber eines Netzwerkes wird von einer übergeordneten Stelle (im deutschen Teil des Internet ist dieses das DE-NIC) ein Bereich von IP-Adressen zugewiesen. Innerhalb dieses Adreßbereichs kann der Betreiber des (Teil-)Netzes die Adressen frei seinen Rechnern zuordnen. Der vom Betreiber frei zu vergebende Bereich wird als Hostanteil, der von der übergeordneten Instanz zugewiesene Anteil als Netzanteil bezeichnet.

Die Funktion des Netzanteils ist analog zu der einer Vorwahl im klassischen Telefonnetz, als das hiermit eine größere Anzahl von Hosts (Telefonen) unter einem gemeinsamen Prefix zusammengefaßt werden kann. Bei IP-Adressen ist der Netzanteil ein Präfix der Host-Adresse, was bedeutet, daß die ersten n Bits der IP-Adresse (von der übergeordneten Instanz) festgelegt sind.

In der Vergangenheit gab es für den Netzanteil nur drei feste Größen, 8, 16 und 24 Bit, auch als Class A, B und C Netzwerke bezeichnet. Die Unterscheidung danach, wie groß der Netzanteil einer gegebenen IP-Adresse war, wurde anhand der ersten zwei bzw. drei Bits der Adresse getroffen.

Seit Einführung von „Classless Inter Domain Routing“ (CIDR) wird der Netzanteil von den übergeordneten Instanzen auch in anderen Größen vergeben. Die Angabe der Größe des Netzanteils (d.h. wieviele Bits einer Adresse sind Netz bzw. Hostanteil) wird nun außerhalb des eigentlichen IP-Paketes

bereitgestellt, beispielsweise durch Angabe der Netzmaske durch den Administrator eines Hosts. Eine Netzmaske ist ein Feld von 32 Bit, das einer IP-Adresse zugeordnet wird. Die Bits, die in der Netzmaske auf 1 gesetzt sind, geben an, daß die Bits an der gleichen Stelle in der IP-Adresse zum Netzanteil gehören. Die auf 0 gesetzten Bits geben an, daß diese Bits der IP-Adresse zum Hostanteil gehören. Der Bereich der auf 1 gesetzten Bits muß zusammenhängend sein. Notwendig wurde diese Vorgehensweise durch die Verknappung des insgesamt zur Verfügung stehenden Adreßraums, speziell bei Adressen mit kurzem (8 oder 16 Bit) Netzanteil.

Um auch innerhalb eines zugewiesenen Adreßbereichs die Administration des Gesamtnetzes in einzelne Teilnetze untergliedern zu können, beispielsweise innerhalb eines größeren Unternehmens, kann der Netzanteil einer IP-Adresse verlängert werden, um so innerhalb eines zugewiesenen Bereichs Subnetze bilden zu können. Diese Subnetze werden i.d.R. 1 : 1 auf Schicht-2 Netzwerke abgebildet. Dies führt in der Praxis dazu, daß häufig Schicht-2 Netze mit IP-Subnetzen identifiziert werden, obwohl es sich um verschiedene Konzepte handelt.

1.3.3 Wegewahl

Liegt der Empfänger innerhalb des gleichen Subnetzes wie der Absender, kann das IP-Paket direkt über das entsprechende Protokoll der Sicherungsschicht an den Empfänger ausgeliefert werden. Anderenfalls muß das IP-Paket über Zwischenstationen an den Empfänger weitergeleitet werden.

Die Auswahl der geeigneten nächsten Zwischenstation („Next-Hop“) wird als Wegewahl bezeichnet. Dazu hat jeder Host eine sog. Wegewahl (Routing)-Tabelle, aus der zu entnehmen ist, ob die betreffende IP-Adresse direkt erreichbar ist, oder an welche Zwischenstation das IP-Paket zu senden ist oder ob die IP-Adresse nicht erreichbar ist. Alle Next-Hops müssen selbst direkt über angeschlossene Schicht-2 Netze erreichbar sein.

Um die Größe der Routingtabelle zu beschränken, wird nicht jede IP-Adresse einzeln eingetragen, sondern nur Teilnetze mit ihrem Präfix. Zusätzlich kann eine sog. „Default Route“ eingetragen werden, die für alle nicht explizit eingetragenen Teilnetze und Adressen gilt. Eine einzelne IP-Adresse kann hierbei als ein Sonderfall eines Teilnetzes mit einem 32-Bit langen Präfix (eine sog. Host-Route) betrachtet werden. Treffen auf eine Empfängeradresse mehrere Präfixe zu, so wird der längste übereinstimmende Präfix genommen, da man davon ausgeht, daß dieser den genauesten Weg zum Ziel beschreibt. Trifft keine Route zu, wird das Paket verworfen und der Absender benachrichtigt (vgl. 1.3.5).

Die Routing-Tabelle kann entweder statisch durch den Administrator des Hosts eingetragen werden oder auch automatisch über sog. Routing-Protokolle gebildet werden. Eine Betrachtung der einzelnen Routing-Protokolle würde den Rahmen dieses Artikels sprengen, hier sei auf [Comer 1995] verwiesen. Gebräuchliche Routing-Protokolle sind beispielsweise *Routing Information Protocol* (RIP), *Open Shortest Path First* (OSPF) oder *Border Gateway Protocol* (BGP). Zum Transport ihrer Informationen können Routing-Protokolle sowohl auf Transportprotokolle (z.B. RIP: UDP, BGP: TCP), als auch direkt auf IP aufsetzen (z.B. OSPF). Die Routinginformation wird aus Sicht von IP dann wie ein weiteres Schicht-4-Protokoll (mit entsprechendem Eintrag im Protokollfeld, z.B. für OSPF die Nr. 89) transportiert.

Die Entscheidung, an welche Zwischenstation ein Datenpaket weitergeleitet wird, erfolgt ausschließlich lokal im Rechner oder Router, der das IP-Paket gerade bearbeitet. Weder Absender noch Empfänger haben einen Einfluß auf den Weg des Datenpaketes (eine Ausnahme ist die Source-Routing Option, siehe Abschnitt 1.3.1). Die Entscheidung wird für jedes eintreffende Paket neu getroffen,

so daß die einzelnen Pakete eines Datenstroms auf unterschiedlichen Wegen durch das Netz geleitet werden können; dies gilt ebenso für den Weg, den eine Antwort durch das Netz zurück nimmt.

Die zugrundeliegende Philosophie bei der Weiterleitung von IP-Paketen ist „Best-Effort“, d.h. jeder Host bemüht sich so gut er kann, die Pakete auszuliefern; es werden aber keinerlei Garantien gegeben, auch nicht, daß ein Paket überhaupt sein Ziel erreicht. So dürfen Hosts in Überlastsituationen IP-Pakete verwerfen, ohne andere Hosts darüber zu informieren.

1.3.4 Fragmentierung

Zum Transport über das Netzwerk werden IP-Pakete in die Pakete der darunterliegenden Schicht-2 Protokolle eingepackt. Es kann vorkommen, daß ein IP-Paket größer ist als die maximale Größe eines Schicht-2-Paketes. So definiert z.B. Ethernet eine maximale Nutzlastgröße von 1500 Bytes (d.h. soviel Platz steht zum Transport von Paketen der Vermittlungsschicht bereit), während IP-Pakete bis zu 65535 Bytes groß sein können. In diesem Fall gibt es zwei Möglichkeiten:

1. Da das Paket, so wie es ist, nicht weitergeleitet werden kann, wird das Paket verworfen und eine Fehlermeldung an den Absender geschickt (siehe Abschnitt 1.3.5). Dies geschieht, wenn im IP-Kopf das „Don’t Fragment“ Bit (DF) gesetzt ist.

2. Das IP-Paket wird in mehrere kleinere IP-Pakete aufgebrochen, die jeweils kleiner als die erlaubte Nutzlastgröße des Schicht-2 Paketes sind. Dieser Vorgang wird als Fragmentierung bezeichnet.

Fragmentierung findet auf der IP-Schicht statt. Jeder Host, den ein IP-Paket passiert, kann fragmentieren, ohne daß ein anderer Host darüber benachrichtigt werden muß. Auch die anderen Protokollschichten werden nicht benachrichtigt.

Ein einmal fragmentiertes IP-Paket wird erst wieder beim Empfänger zusammengesetzt (defragmentiert), auch wenn auf dem Weg zum Ziel Teilstrecken existieren, die den Transport des IP-Paketes als Ganzes erlauben würden. Allerdings kann ein Teilpaket (Fragment) durchaus von anderen Hosts nochmals fragmentiert werden, falls dies notwendig (d.h. das Fragment zu groß) sein sollte.

Bei der Fragmentierung wird letztlich nur der Datenanteil des IP-Paketes wirklich unterteilt, jedes Fragment erhält denselben IP-Kopf wie das ursprüngliche Paket, inklusive Optionen, mit folgenden Änderungen: Abbildung 1.3 zeigt ein Beispiel eines Schicht-2 Netzes mit einer maximalen Paketgröße von 620 Bytes und einem IP-Paket von 1420 Bytes Länge.

- Das Längenfeld wird auf die Größe des Fragments gesetzt.

- Das Offset-Feld bestimmt die Position des Fragments innerhalb des ursprünglichen Paketes. Ein Fragment enthält neben dem Kopf die Bytes von $Offset \times 8$ bis $Offset \times 8 + Länge - Kopflänge - 1$ (mit $Kopflänge = HLEN \times 4$).

- In allen Fragmenten wird das “More Fragments“ Bit gesetzt, mit Ausnahme des letzten Fragments. Das letzte Fragment ist das, welches die Bytes bis zum Ende des ursprünglichen Paketes enthält.

Bei der Defragmentierung des Paketes werden alle Fragmente mit dem gleichen Identifikationsfeld (vgl. 1.3.1) als demselben ursprünglichen Paket zugehörig erkannt. Über das Fragmentabstandfeld

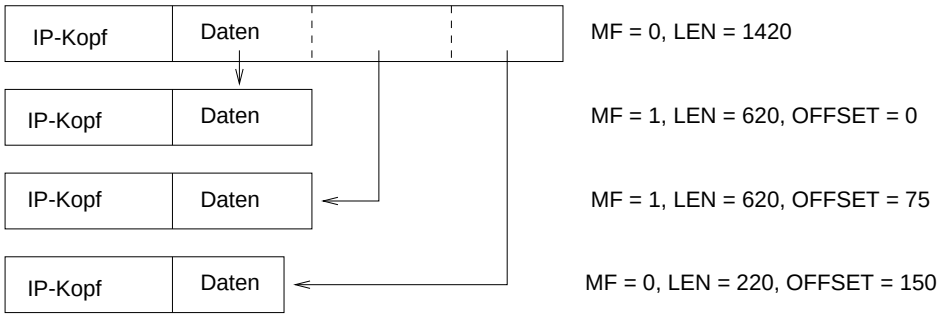


Abbildung 1.3: IP-Fragmentierung

wird die Position der Fragmente innerhalb des ursprünglichen Paketes bestimmt. Erst nach Empfang des „letzten“ Paketes (das ohne More-Fragment Bit) kann der Empfänger bestimmen, ob er alle Fragmente erhalten hat, da nur aus diesem Paket die Größe des ursprünglichen Paketes (Fragmentabstand $\times$ *8 + Länge - Kopflänge*) rekonstruierbar ist. Anhand der Fragmentabstand- und Längenfelder der anderen Fragmente kann der Empfänger bestimmen, ob noch Fragmente fehlen. Werden innerhalb eines implementierungsabhängigen Zeitfensters nicht alle Fragmente empfangen, werden alle Fragmente des IP-Paketes gelöscht und das IP-Paket ist damit verloren. Über diesen Vorgang wird keine Fehlermeldung an den Absender geschickt (vgl. „Don’t Fragment“-Fall weiter oben).

1.3.5 ICMP

Innerhalb von IP gibt es keinen Mechanismus zur Meldung von Fehlern an andere Hosts im Netzwerk. Dieser Mechanismus ist in das „Internet Control Message Protocol“ (ICMP) ausgelagert. ICMP dient als Steuer- und Fehlerbenachrichtigungsprotokoll und wird nicht nur von IP, sondern auch von den Protokollschichten oberhalb von IP, z.B. TCP und UDP benutzt. Dazu wird bei einigen ICMP-Meldungen der IP-Kopf und evtl. weitere Teile des Datenpaketes (z.B. der UDP oder TCP Protokollkopf) der ICMP-Meldung hinzugefügt. Ein wichtiges Merkmal von ICMP ist, daß Fehler bei der Bearbeitung von ICMP-Paketen niemals neue ICMP-Fehlermeldungen hervorrufen; es werden dadurch mögliche Endlosschleifen vermieden, die dazu führen könnten, das Netz mit ICMP-Fehlermeldungen zu fluten.

ICMP setzt als eine Art Pseudoprotokoll auf IP auf. Zwar wird ICMP zur Übertragung wie andere (Transport)Protokolle in IP eingepackt, die Bearbeitung des ICMP Paketinhalts erfolgt jedoch auf Schicht 3 und nicht in der Transportschicht. Dort hätte die Bearbeitung keinen Einfluß auf das Verhalten der Netzwerkschicht (IP).

Die beiden wesentlichen Felder des in Abbildung 1.4 gezeigten ICMP-Paketes sind das Typfeld und das Kodefeld. Das Typfeld legt grob die Klasse des Fehlers bzw. der Steuerungsaufgabe fest, das Kodefeld bestimmt die Details des Typs. Die Prüfsumme stellt sicher, daß der Kopf des ICMP-Paketes korrekt übertragen wurde, der weitere Inhalt eines ICMP-Paketes ist abhängig vom Typenfeld und wird aus Platzgründen hier nicht weiter behandelt.

Tabelle 1.1 beschreibt einige der vielen Typen von ICMP-Paketen und einige der darin auftretenden Kodes, für weitere Details sei auf [RFC 792] verwiesen.

Der Vorteil der Auslagerung der Fehler- und Steuermeldungen an ein eigenes Protokoll (IP-Protokollnummer 1) liegt in der größeren Flexibilität: Meldungstypen können hinzugefügt oder geändert

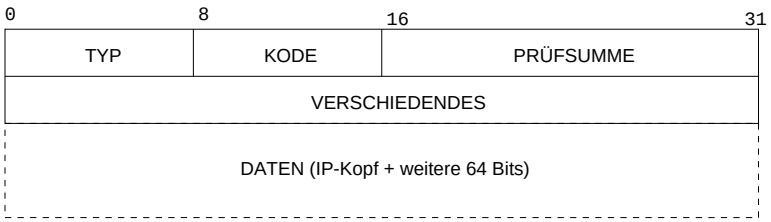


Abbildung 1.4: ICMP-Paketformat

Typ		Kode	Erläuterung
0	Echo Reply	0	Antwort auf Echo Request Paket
3	Destination Unreachable	1	Ziel nicht erreichbar, Host nicht erreichbar
		7	Ziel nicht erreichbar, Empfänger unbekannt
		8	Ziel nicht erreichbar, Absender isoliert
5	Redirect	1	Anweisung, einen anderen Router zu benutzen
8	Echo Request	0	Prüfen, ob Empfänger erreichbar
11	Time-Exceeded	0	Lebenszeit-Feld abgelaufen
12	Parameter Problem	0	Fehler, der durch andere Typen nicht abgedeckt ist

Tabelle 1.1: ICMP Typen und Kodes

werden, ohne daß an den übrigen Protokollen etwas geändert werden muß. So kann etwa ein neues Protokoll der Transportschicht eine neue Klasse von ICMP-Meldungen erfordern, ohne daß hierfür das IP-Protokoll erweitert werden muß. Sollten sich durch das neue Protokoll Probleme auf der Vermittlungsschicht ergeben, so können neue ICMP-Typen hinzugefügt werden, ohne daß am Transportprotokoll Änderungen notwendig werden.

1.4 TCP

TCP ist ein Protokoll der Transportschicht, mit dem eine gesicherte, bidirektionale, Ende-zu-Ende Verbindung bereitgestellt wird. (Daher wird TCP auch als *verbindungsorientiertes* Protokoll bezeichnet.) Gesichert bedeutet in diesem Zusammenhang nur eine Absicherung gegen Übertragungsfehler der darunterliegenden Protokollschichten, primär gegen:

- **Auslieferung der Pakete beim Empfänger in anderer Reihenfolge als beim Absender abgeschickt:** Dies kann beispielsweise passieren, wenn die Datenpakete unterschiedliche Wege durch das Netzwerk nehmen und dadurch in anderer Reihenfolge ihr Ziel erreichen als sie abgesendet wurden.
- **Verlust von Datenpaketen:** Wie in Abschnitt 1.3.3 gezeigt, ist es für Hosts in einem IP-Netzwerk legitim, unter bestimmten Umständen IP-Pakete zu verwerfen. Die darüberliegenden Schichten oder andere Hosts werden darüber u.U. nicht benachrichtigt.
- **Mehrfache Auslieferung desselben Datenpaketes beim Empfänger:** Auch dieser Fall kann sich aus einem Zusammenwirken mehrerer Faktoren in einem IP-Netz ergeben, beispielsweise bei ungeschickt gewählten Timeout-Werten in einem Netzwerk unter Überlast.

1.4.1 TCP-Paketformat

Abbildung 1.5 zeigt den Aufbau eines TCP-Datenpaketes. Wie bei IP besteht das Paket aus einem Kopf und einem Datenteil. Der Datenteil wird in der Transportschicht nicht ausgewertet, sondern unverändert an die höheren Schichten weitergegeben. Die Länge des Datenteils ergibt sich aus der Gesamtlänge des IP-Paketes, in dem das TCP-Paket transportiert wurde, minus der Längen der IP- und TCP-Protokollköpfe, die in den jeweiligen HLEN-Feldern angegeben sind (siehe auch 1.2). Die Prüfsumme dient der Integritätsprüfung des TCP-Kopfs. Zur Erklärung der im folgenden nicht aufgeführten Felder und Flags sei auf [RFC 793] verwiesen.

Die Felder Sequenznummer, Bestätigungsnummer und Fenstergröße dienen zur Steuerung des Datenflusses zwischen Kommunikationspartnern. Da sie sich am besten an einem konkreten Beispiel erklären lassen, sei auf Abschnitt 1.4.4 verwiesen.

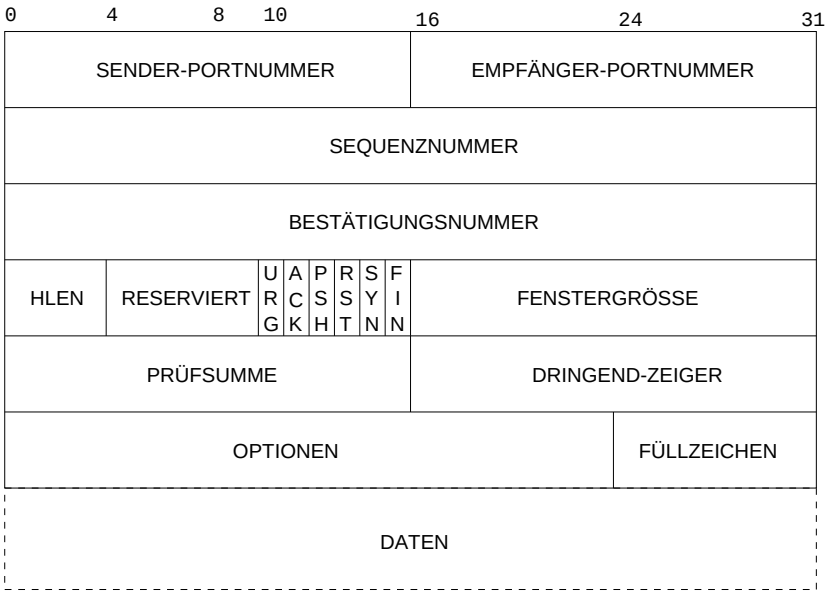


Abbildung 1.5: TCP Paketformat (aus [Santi 1998, S. 55])

Zwischen zwei Rechnern können gleichzeitig mehrere TCP-Verbindungen bestehen. Um die einzelnen Verbindungen auf TCP-Ebene auseinanderhalten zu können, werden Portnummern verwendet. Der Initiator einer Verbindung benutzt die Empfängerportnummer zur Auswahl des Dienstes beim Empfänger. Die Portnummer des Absenders dient dazu, eventuelle Mehrfachverbindungen vom Absender zum gleichen Port beim Empfänger auseinanderzuhalten. Jede TCP-Verbindung wird so durch ein 4-Tupel, bestehend aus Absender-IP-Adresse, Absender-TCP-Portnummer, Empfänger-IP-Adresse und Empfänger-TCP-Portnummer eindeutig bestimmt. Die Portnummern werden mit jedem TCP-Paket in den Feldern Empfängerportnummer und Absenderportnummer verschickt.

Da ein Dienst über die Portnummer ausgewählt wird, muß der Absender wissen, welcher Dienst sich hinter welcher Portnummer verbirgt. Dazu gibt es das Konzept der „Well Known Services“. Für eine Reihe von Standard-Diensten werden (im Rahmen von TCP/IP) allgemeinverbindlich Portnummern festgelegt. So befindet sich hinter Portnummer 25 beispielsweise der Email-Dienst SMTP, hinter Portnummer 20 und 21 der Filetransfer-Dienst FTP, usw. Für Dienste, die nicht als „Well Known Service“ definiert wurden, gibt es zwei Möglichkeiten. Die Programmiererin kann die Portnummern fest in die

Client- und Serverprogramme des Dienstes kodieren, oder das Serverprogramm des Dienstes läßt sich bei einem Verzeichnisdienst registrieren, der seinerseits ein Well Known Service ist. Mehr dazu in Abschnitt 1.6.

1.4.2 TCP-Programmierschnittstelle und Zustandsautomat

Die Arbeitsweise eines komplexen Protokolls läßt sich gut mittels eines endlichen Automaten veranschaulichen. Abbildung 1.7 zeigt einen solchen Automaten für TCP.[1]

Der Ausgangszustand ist `CLOSED`, der Zustand, in dem die eigentliche Datenkommunikation stattfindet, ist `ESTABLISHED`. Auf die einzelnen Übergänge und Aktionen wird in den folgenden Abschnitten genauer eingegangen werden.

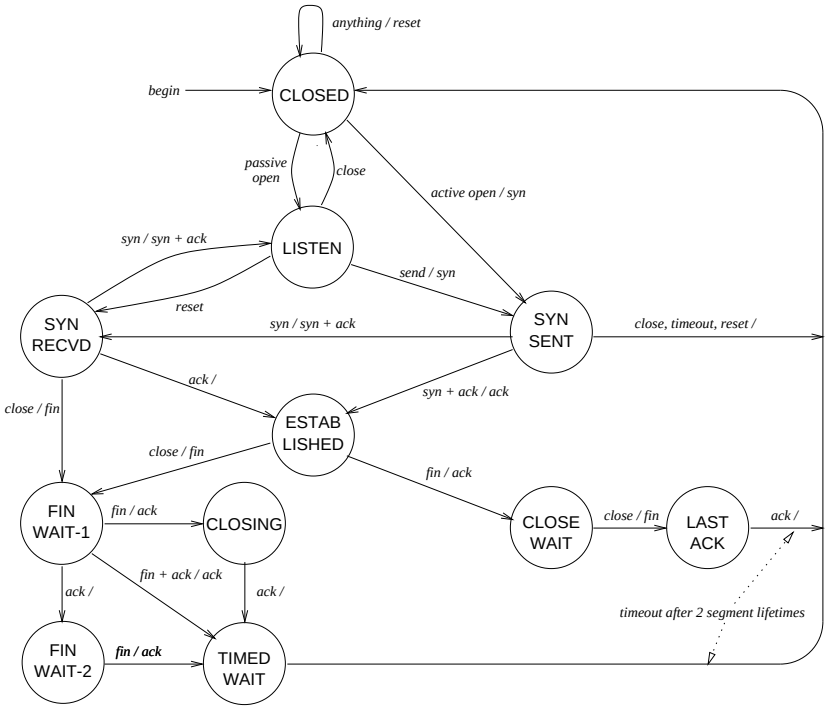


Abbildung 1.6: TCP als endlicher Automat (aus [Comer 1995, S. 220])

UNIX TCP-Programmierschnittstelle

In UNIX[2] werden die Endpunkte von Datenströmen durch Dateideskriptoren repräsentiert. Auf einem Dateideskriptor sind grundsätzlich die Operationen Öffnen (Anlegen), Lesen, Schreiben und Schlie-

[1] Die Kreise entsprechen den Zuständen und die Pfeile stellen die möglichen Übergänge zwischen den Zuständen dar. Die Beschriftung der Pfeile gibt an, welche Aktion den Übergang veranlaßt hat, bzw. welche Ausgabe (d.h. welche Pakete mit welchen Flags) die TCP-Implementierung beim Zustandsübergang aussendet. Die Eingabe kann dabei sowohl der Empfang eines Datenpaketes sein, als auch eine Operation des Anwendungsprogramms, die mit dem TCP-Automaten über das Betriebssystem kommuniziert. Die auslösende Aktion (die Eingabe) steht vor, die Ausgabe hinter dem Schrägstrich.

[2] Die Schnittstelle unter Windows, WINSOCK, funktioniert ähnlich und viele andere Systeme bauen auf den Konzepten der UNIX-Schnittstelle auf, so daß diese Schnittstelle sich gut zur Einführung eignet.

ßen (Abbauen) definiert. Der Endpunkt eines Netzwerk-Datenstroms wird in der OSI-Terminologie als „Service Access Point“ (SAP) bezeichnet. In UNIX heißt die Implementierung eines SAPs *Socket*. Ein (TCP)Socket verhält sich aus Sicht der Programmiererin ähnlich wie ein Dateideskriptor (d.h. die Operationen Öffnen, Lesen, Schreiben und Schliessen für Dateideskriptoren können auch auf Sockets angewendet werden), wobei das Lesen von Daten einer Empfangsoperation entspricht und das Schreiben von Daten einer Sendeoperation entspricht.

Netzwerk-Dateideskriptoren (Sockets) werden mit dem Aufruf `socket()` erzeugt. Danach gibt es zwei Wege. Soll der Socket zur Annahme von Verbindungen dienen, (z.B. für Server) oder soll lediglich eine Verbindung zu einem anderen Rechner aufgebaut werden (typischerweise bei Clients). Die Art des (TCP)Sockets wird nur durch die (für die Programmiererin nicht direkt sichtbaren) Zustände des TCP-Automaten bestimmt. Welchen Zustand der Automat einnimmt, bestimmt die Programmiererin durch die weiteren Systemaufrufe.

Zunächst kann dem Socket eine IP-Adresse und eine Portnummer mittels `bind()` zugeordnet werden. Für Server-Sockets ist dieser Schritt zwingend, bei Client Sockets wird dies notfalls vom Betriebssystem während des `connect()` Aufrufs nachgeholt (das Betriebssystem ordnet dann eine nach betriebssysteminternen Regeln gewählte Portnummer zu).

Für Server-Sockets wird der Socket mit `listen()` in den gleichnamigen Zustand versetzt (in Abbildung 1.7 der Übergang mit *passive open* nach LISTEN). In diesem, auch als *halboffen* bezeichnetem Zustand, wartet der Socket auf Verbindungsaufbauwünsche. Der Aufruf `accept()` nimmt einen solchen Wunsch entgegen und bewirkt zweierlei: Vom Socket wird eine Kopie erzeugt und der Programmiererin zurückgegeben, während der alte Socket im Zustand LISTEN bleibt, damit weitere Verbindungen angenommen werden können. Der alte Socket repräsentiert weiter den Endpunkt einer halboffenen Verbindung (nur an Empfänger-IP-Adresse und Empfänger-TCP-Portnummer gebunden), während der neue, von `accept()` zurückgelieferte Socket den Endpunkt einer vollständigen Verbindung repräsentiert und zur weiteren Kommunikation mit dem fremden Host genutzt wird.

Clients nehmen mittels `connect()` Verbindung zum Server auf (was dem Übergang *active open* von `CLOSED` nach `SYN-SEND` in Abbildung 1.7 entspricht) und können unmittelbar danach mit der Kommunikation beginnen.

Die weiteren Aufrufe und ihr Zusammenhang mit den Zustandsübergängen des TCP-Automaten und den einzelnen Feldern im TCP-Kopf wird nun in den folgenden Abschnitten 1.4.3, 1.4.4 und 1.4.5 genauer behandelt.

1.4.3 Verbindungsaufbau

Der Verbindungsaufbau bei TCP wird als *Three-Way-Handshake* bezeichnet, weil dabei drei Datenpakete zwischen den beteiligten Rechnern ausgetauscht werden, wie Abbildung 1.7 zeigt.

Der Initiator der Verbindung beginnt, in dem er ein Paket mit dem SYN-Flag zum Kommunikationspartner sendet. Die Sequenznummer wird dabei vom Betriebssystem zufällig gewählt. Der Partner antwortet mit einem Datenpaket, das zwei Aufgaben hat. Erstens wird der Empfang des ersten Paketes bestätigt. Dazu wird im Antwortpaket das ACK-Flag gesetzt und eine Bestätigungsnummer gewählt, die der Sequenznummer des ersten Paketes entspricht. Zweitens wird durch eine eigene Sequenznummer und das SYN-Flag die Annahme der Verbindung dem Initiator gegenüber bestätigt. Der Initiator bestätigt den Empfang des zweiten Paketes durch Senden eines Antwortpaketes, in dem das ACK-Flag gesetzt ist und die Bestätigungsnummer der Sequenznummer $+1$ des Partners entspricht. Obwohl das

dritte Paket nicht zwingend notwendig wäre, um eine Verbindung aufzubauen, dient es der Sicherung gegen den Verlust des zweiten Paketes und zur Bestätigung der initialen Sequenznummer des Empfängers. Beide Seiten können so sicher sein, daß ihr Verbindungsaufbau erfolgreich war.

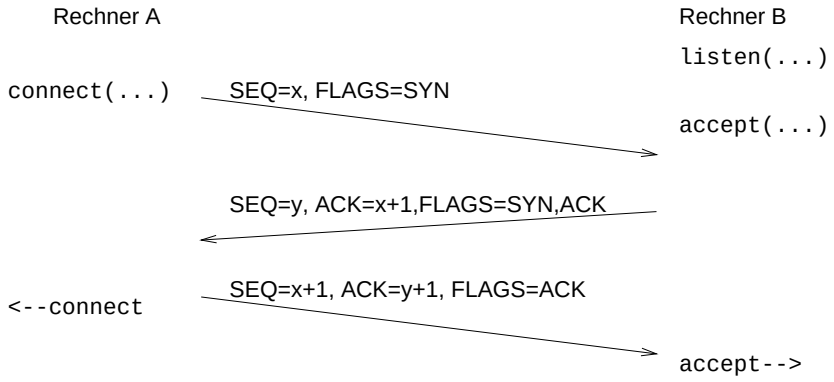


Abbildung 1.7: TCP-Verbindungsaufbau

Aus Sicht einer Programmiererin wird beim Initiator zunächst ein Socket mit Hilfe des `socket()` Aufrufs angelegt und danach `connect()` aufgerufen, um die Verbindung aufzubauen. Dies ist der Moment, zu dem das erste Paket gesendet wird. Der endliche Automat von Rechner A tritt in den Zustand `SYN-SEND` ein (siehe Abbildung 1.7). Nach Empfang des ersten Paketes tritt der endliche Automat von Rechner B vom Zustand `LISTEN` in den Zustand SYN-RECEIVED. Wird die Verbindung mit `accept()` auf Serverseite angenommen, tritt der Zustandsautomat von Rechner B in den Zustand `SYN-SEND` und das zweite Pakete wird gesendet. Nach Empfang dieses Paketes tritt der Zustandsautomat von Rechner A in den Zustand `ESTABLISHED`, der `connect()` Aufruf wird beendet und das dritte Paket wird gesendet. Nach Empfang dieses Paketes erreicht auch der endliche Automat von Rechner B den Zustand `ESTABLISHED` und der `accept()` Aufruf gibt die Kontrolle an das aufrufende Programm zurück. Die TCP-Verbindung ist nun etabliert.

1.4.4 Datenaustausch

Der Datenaustausch unter TCP ist bidirektional, d.h. beide Kommunikationspartner können gleichzeitig senden und empfangen. Der Austausch ist dabei nicht an bestimmte Blockgrößen gebunden, sondern jede Seite kann in einer Sendeoperation beliebig viele Zeichen versenden; die TCP/IP Implementierung bestimmt selbst, in wievielen TCP bzw. IP Datenpaketen die Daten versendet werden.

Die Felder Sequenznummer, Bestätigungsnummer und Fenstergröße dienen dabei zur Steuerung des Datenflusses zwischen den Kommunikationspartnern. Beide Partner erzeugen beim Verbindungsaufbau je eine zufällig gewählte Sequenznummer. Diese Nummer wird sich im Verlauf der Verbindung ändern, und zwar wird sie jeweils um die Anzahl der empfangenen Bytes erhöht. Der Empfänger bestätigt den Empfang der Daten dadurch, daß er ein Antwortpaket sendet, in dem 1. das ACK-Flag gesetzt ist, und 2. die Sequenznummer des Absenders, erhöht um die Anzahl der erfolgreich empfangenen Bytes, als Bestätigungsnummer eingetragen ist.

In Abbildung 1.8 ist ein einfacher TCP-Datenaustausch zu sehen. A sendet fünf Bytes an B, dieser antwortet, indem er den Empfang der fünf Bytes bestätigt (ACK-Flag gesetzt und Bestätigungsnummer

$ACK = x+5$) und im selben Paket 10 Bytes an A schickt. Dieser anwortet mit seiner neuen Sequenznummer, die um fünf erhöht ist, und bestätigt (ACK-Flag und Bestätigungsnummer $ACK = y+10$) den Empfang der Daten von B.

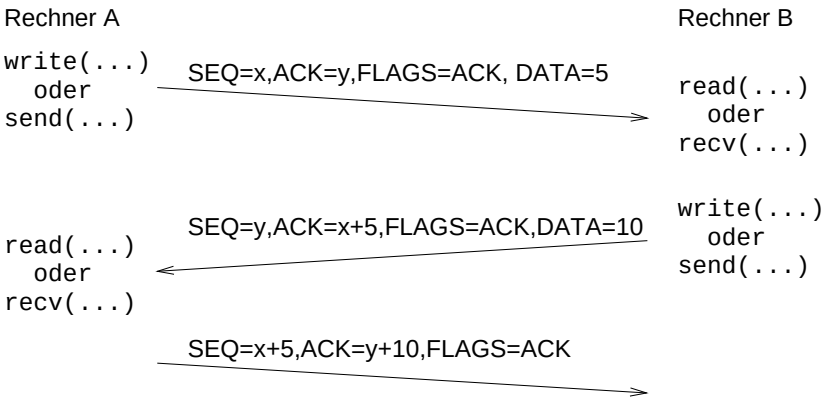


Abbildung 1.8: Einfacher TCP-Datenaustausch

Da TCP einen kontinuierlichen Datenstrom zur Verfügung stellt und ein Socket unter UNIX einen Dateideskriptor darstellt, können zum Senden und Empfangen von Daten die UNIX-Systemaufrufe zum Schreiben und Lesen auf Dateien, `write()` und `read()` verwendet werden. Anstelle eines gewöhnlichen Dateideskriptors wird der Socket-Deskriptor benutzt. Stehen an einem Socket weniger Daten zur Verfügung, als der `read()` Aufruf lesen will, so wird dieser Aufruf im Normalfall (sog. „synchronous IO“) solange blockiert, bis genügend Daten eingetroffen sind. Umgekehrt ist es dem Betriebssystem freigestellt (und damit implementierungsabhängig), in wievielen Paketen die Daten eines `write()` Aufrufs versendet werden. Randbedingungen sind hierbei nur die maximale Paketgröße des Schicht-2 Netzwerks, an den der Absender angeschlossen ist, und das momentan offene TCP-Fenster. Anstelle der Aufrufe `read()` und `write()` können auch die Aufrufe `recv()` und `send()` verwendet werden. Letztere erlauben es, einige zusätzliche Eigenschaften von TCP auszunutzen, die in diesem Artikel nicht erläutert werden.

Während eines Datenaustausches ändert sich am Zustand der endlichen Automaten nichts. Beide Seiten verbleiben im Zustand `ESTABLISHED`, bis eine der Seiten die Verbindung abbaut (siehe Abschnitt 1.4.5) oder ein schwerer Fehler auftritt. Dann sendet eine der Seiten ein Paket, in dem das RST-Flag (RST: Reset) gesetzt ist, und beide Seiten gehen in den Ausgangszustand `CLOSED`.

Sliding Window und Window Advertisement

Das TCP-Protokoll arbeitet nach dem Prinzip des *Sliding Window*: Jeder Kommunikationspartner unterhält ein „Fenster“, dessen „Rahmen“ aus zwei Zahlen besteht: Aus der letzten empfangenen Bestätigungsnummer und aus der Summe der Bestätigungsnummer und der Fenstergröße. Im „Inneren“ des Fensters befindet sich die Sequenznummer des Kommunikationspartners. Mathematisch bedeutet dies, daß

$$\textit{Bestätigungsnummer} \leq \textit{Sequenznummer} \leq \textit{Bestätigungsnummer} + \textit{Fenstergröße}.$$

Solange die Sequenznummer kleiner der Bestätigungsnummer plus der Fenstergröße ist, (die Sequenznummer sich also noch „innerhalb des Fensters“ befindet) kann der Kommunikationspartner

Daten senden, ohne auf eine Bestätigung des Kommunikationspartners warten zu müssen. Die Datenmenge (in Bytes) ergibt sich aus der Differenz von Bestätigungsnummer plus Fenstergröße minus der Sequenznummer.

Die eintreffenden Bestätigungen „verschieben (slide)“ das Fenster vorwärts (in Richtung höherer Sequenznummern), da der „Rahmen“ des Fensters abhängig von der empfangenen Bestätigungsnummer ist.

Ein Beispiel für einen solchen Datentransfer gibt Abbildung 1.9, bei der gleichzeitig auch der Fall eines nicht ausgelieferten Paketes dargestellt ist. Zunächst sendet A zwei Datenpakete (drei und fünf Bytes), die von B in einem Paket bestätigt werden (Bestätigungsnummer $ACK = x + 8$).

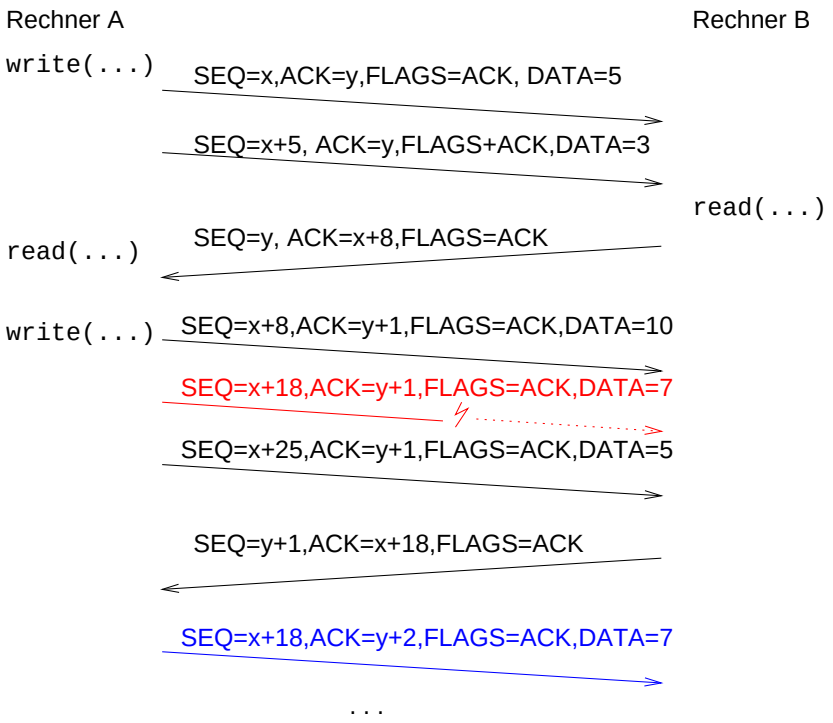


Abbildung 1.9: TCP-Datentransfer mit Sliding Window

Die Größe des Fensters wird im Feld „Fenstergröße“ des TCP-Protokollkopfes (siehe 1.5) dem Kommunikationspartner mitgeteilt. Das TCP-Protokoll erlaubt es den Kommunikationspartnern, diese Größe dynamisch den jeweiligen Gegebenheiten anzupassen, auch während einer laufenden Kommunikation. Dieses Verfahren wird als *Window Advertising* bezeichnet.

In Abbildung 1.9 wird im unteren Teil gezeigt, was geschieht, wenn ein Paket verlorengeht. A sendet drei Datenpakete mit 10, 7 und 5 Bytes, wobei das Paket mit 7 Bytes verlorengeht. B bestätigt deshalb nur den Empfang des ersten Paketes (Bestätigungsnummer $ACK = x + 18$). A weiß daraus (da er selbst noch Kopien aller gesendeten Pakete behält, bis die Bestätigung eingetroffen ist) welches Paket nochmals gesendet werden muß. Nach dem Empfang dieses Paketes könnte B den Empfang der restlichen 12 Bytes bestätigen, da das dritte der ursprünglichen Pakete erfolgreich empfangen wurde und nur der Empfang noch nicht bestätigt wurde (In der Abbildung ist dieses Paket nicht gezeigt.).

1.4.5 TCP-Verbindungsabbau

Soll die Verbindung zwischen zwei Rechnern wieder abgebaut werden, so sendet der Rechner, der die Verbindung abbauen will, ein Datenpaket, in dem das FIN-Flag (FIN: Finish) gesetzt ist, wie in Abbildung 1.10 gezeigt. Der TCP-Zustandsautomat dieses Rechners befindet sich dann um Zustand FIN-WAIT-1 (siehe 1.6) Derandere Rechner nimmt dieses FIN-Flag zur Kenntnis (sein Zustandsautomat geht in den Zustand CLOSE-WAIT, vgl. 1.6) und sendet ein Datenpaket mit dem ACK-Flag zum Zeichen, daß das Paket mit dem FIN-Flag angekommen ist.

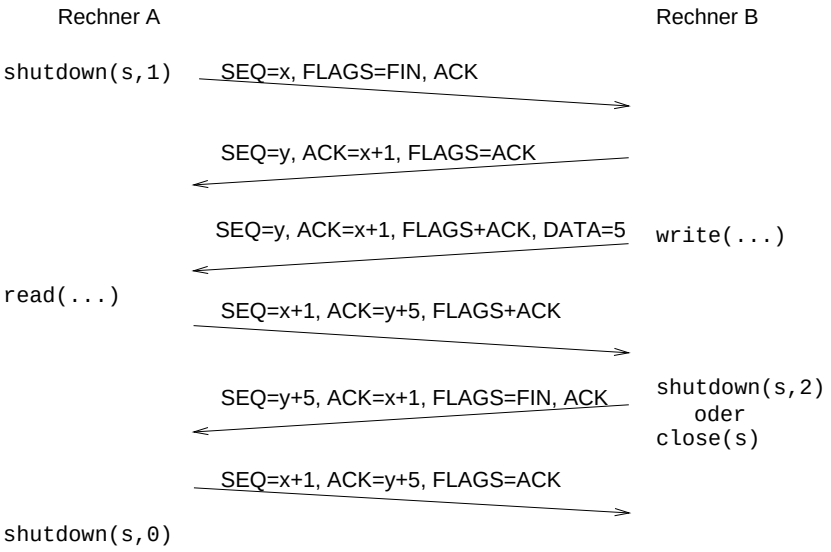


Abbildung 1.10: Abbau einer TCP-Verbindung

Das FIN-Flag signalisiert lediglich, daß eine Seite das Senden von Daten beendet hat. Die andere Seite kann weiterhin Daten senden, bis auch sie mit dem FIN-Flag das Ende ihres Datenstroms signalisiert. Erst nachdem beide Seiten jeweils ein Datenpaket mit FIN-Flag gesendet und den Empfang des FIN-Flags der Gegenseite bestätigt haben, ist die Verbindung vollständig abgebaut. Solange befindet sich der TCP-Automat in einem der Zustände `FIN-WAIT-1, FIN-WAIT-2, CLOSING oder TIMED-WAIT` (siehe Abbildung 1.6).

Zum Abbau der Verbindung stehen der Programmiererin zwei Systemaufrufe zur Verfügung. Entweder kann die Verbindung mittels `shutdown()` einseitig geschlossen werden oder die Verbindung kann mit `close()` vollständig abgebaut werden. Im letzteren Fall nimmt das aufrufende Programm keine Daten mehr an, ihr Empfang wird jedoch vom Betriebssystem quittiert und die Daten werden verworfen, bis die Gegenseite ihrerseits die Verbindung abbaut. Dem Programm wird beim Aufruf signalisiert, ob sich noch ungelesene Daten in den Puffern des Betriebssystems befinden. Der Aufruf wird in diesem Falle ohne weitere Aktionen abgebrochen, es sei denn, die Programmiererin signalisiert explizit, daß die Daten verworfen werden sollen.

1.5 UDP

Im Gegensatz zu TCP ist UDP ein sehr einfach aufgebautes Protokoll. UDP weist daher auch beträchtlich weniger Funktionalität auf als TCP. Der wichtigste Unterschied zu TCP ist, daß UDP ein

verbindungsloses Protokoll ist. Somit bietet UDP keinerlei Schutz gegen den Verlust von Datenpaketen, gegen Paketduplikate oder die Auslieferung von Paketen in einer anderen Reihenfolge.

Der Vorteil liegt in der wesentlich einfacheren Implementierung von UDP, was besonders Systemen mit knappen Ressourcen zugute kommt. Der zweite Grund für die Existenz von UDP liegt darin, daß für Anwendungen, die die volle Funktionalität von TCP nicht benötigen, ein einfaches Protokoll bereitgestellt wird. So besteht die Möglichkeit, die notwendige Funktionalität nur im für die Anwendung benötigten Umfang anzufordern. Die folgende Abbildung 1.11 zeigt schematisch den Aufbau eines UDP-Paketes.

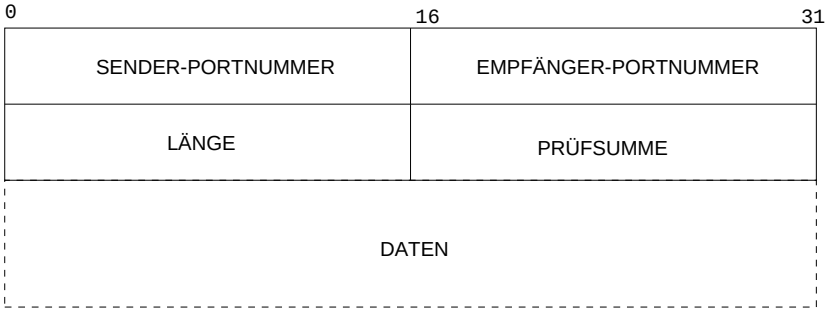


Abbildung 1.11: UDP Paketformat (aus [Santi 1998, S. 68])

Das Längenfeld gibt die Länge des UDP-Paketes in Bytes an und das Prüfsummenfeld wird wie bei IP und TCP nur dazu benutzt, die Integrität der Datenfelder im Protokollkopf sicherzustellen, nicht jedoch die der Nutzdaten. Im Gegensatz zu TCP bezieht UDP auch Felder des IP-Kopfes in die Prüfsummenberechnung mit ein, so daß hier von einer „Pseudoprüfsumme“ gesprochen wird. Hier zeigt sich ein weiterer Vorteil von UDP gegenüber TCP, nämlich der wesentlich geringere Overhead, ein UDP-Kopf ist nur acht Byte lang, gegenüber 20 (oder mehr bei Verwendung von Optionen) Byte bei TCP.

Analog zu TCP benutzt UDP das Konzept der Portnummern, um die verschiedenen Anwendungen unter einer IP-Adresse anzusprechen. Wie unter TCP werden 16 Bit große Zahlen für die Portnummern des Absenders und Empfängers verwendet. TCP und UDP Portnummern sind nicht identisch, eine unter TCP bereits vergebene Portnummer kann unter UDP nochmals verwendet werden und u.U. eine völlig andere Anwendung identifizieren.

1.5.1 UDP-API

Das 4-Tupel (SenderIP, SenderPort, EmpfängerIP, EmpfängerPort) abstrahiert unter UDP nicht eine Verbindung wie unter TCP, sondern eine Kommunikationsbeziehung. Der Unterschied wird aus Sicht der Programmiererin deutlicher, wenn das API der UDP-Sockets betrachtet wird.

UDP-Sockets werden wie TCP-Sockets mit dem Aufruf `socket()` vom Betriebssystem bereitgestellt, lediglich die Parameter sind anders. Der `connect()` Aufruf kann entfallen, da UDP ein verbindungsloses Protokoll ist. Ebenso kann der `accept()` Aufruf für Sockets entfallen, die von Servern benutzt werden. Zwingend notwendig bleibt für Server jedoch der Aufruf von `bind()`, um dem Betriebssystem mitzuteilen, unter welcher IP-Adresse und Portnummer das Programm ansprechbar ist.

Die Funktion `sendto()` kann dazu benutzt werden, auch an verschiedene Empfänger Daten zu versenden, allerdings müssen die Empfängeradressen beim Aufruf übergeben werden. Mit TCP-Sockets ist

das nicht möglich, da der Aufruf von `connect()` dort den Empfänger festlegt. Analog ist die Entgegennahme von Daten ohne den Aufruf von `accept()`, direkt mit `recv()` oder `recvfrom()` möglich. Der erste Aufruf empfängt Daten von beliebigen Absendern, der zweite Aufruf ermöglicht die Auswahl eines Absenders durch Übergabe der Absenderadresse beim Aufruf. Beide Aufrufe blockieren, wenn keine Daten vorhanden sind. Ein UDP-Socket abstrahiert somit nur das eine Ende (IP-Adresse, Portnummer) einer Kommunikationsbeziehung und nicht den Endpunkt einer bidirektionalen Verbindung, die es unter UDP nicht gibt.

1.6 Symbolische Namen

Im Verlauf dieses Vortrags wurden oft Portnummern und IP-Adressen erwähnt. Leider sind numerische Werte für menschliche Anwender oft schwer zu merken und bringen auch nicht den notwendigen Abstraktionsgrad mit sich, den Menschen erwarten. Anstelle von „Anwendung hinter TCP-Port 80, auf Adresse 123.234.132.231“ wird eher etwas wie „WWW-Server auf Rechner www.cert.dfn.de“ verwendet. Die Verwendung solcher symbolischer Namen erfordert eine Umsetzung zwischen IP-Adressen und Dienstkennungen.

Dies geschieht durch Bereitstellung von Bibliotheksfunktionen, die von der Anwendung benutzt werden, um die symbolischen Namen zu übersetzen. In UNIX sind das Funktionen wie `gethostbyname()` zur Umsetzung von Namen in IP-Adressen, oder `getservbyname()` zur Bestimmung der zu einem Dienst gehörenden Portnummern. Die Routinen selbst sind teilweise nur ein Einstiegspunkt in zum Teil aufwendige Such- und Abfrageschemata, deren Details an dieser Stelle nicht erläutert werden können.

Die eigentliche Abbildungsinformation muß vom Administrator des Systems entweder lokal oder auf einem Server im Netzwerk bereitgestellt werden. Im letzteren Fall müssen Anfragen an diesen Server weitergeleitet werden.

Zur lokalen Bereitstellung werden typischerweise Dateien an standardisierter Stelle im System verwendet. Unter UNIX sind dies z.B. die Dateien `/etc/hosts` und `/etc/services`.

Zur Bereitstellung der Informationen im Netzwerk stehen standardisierte Informationsdienste wie DNS [RFC 1034, RFC 1035] oder LDAP (X.500) zur Verfügung. Welcher Dienst benutzt wird, ist von der jeweiligen Umgebung abhängig. In UNIX-Umgebungen wird häufig DNS verwendet, andere Umgebungen benutzen evtl. herstellereigene Informationsdienste, deren Spezifikation jedoch nicht immer offengelegt ist, so daß sie hier nicht weiter betrachtet werden.

Stehen mehrere Informationsdienste zur Verfügung, so muß eine Auswahl getroffen werden, welche Dienste benutzt werden sollen. Außerdem muß für den Fall, daß die gleiche Information (nicht notwendigerweise der gleiche Inhalt) von mehreren Diensten bereitgestellt wird, entschieden werden, welchem Dienst die höhere Glaubwürdigkeit eingeräumt wird.

In UNIX wird dies durch die Datei `/etc/nsswitch.conf` geregelt: Für jede Art von Information wird in der Datei eine priorisierte Liste von Diensten spezifiziert. Die Dienste werden sequentiell abgefragt, bis einer der Dienste ein positives Ergebnis liefert oder das Ende der Liste erreicht ist. Damit bestimmt die Position eines Dienstes in der Liste seine Wichtigkeit bzw. Glaubwürdigkeit.

1.6.1 DNS

Der am weitesten verbreitete Namensdienst im Internet ist das „Domain Name System“, abgekürzt DNS. Er wurde 1987 entwickelt, um der wachsenden Anzahl von Hosts im Internet Rechnung zu tragen, da Abgleich und Pflege der zentralen Hosts-Datei nicht mehr möglich waren. Die Spezifikation wurde in [RFC 1034] und [RFC 1035] veröffentlicht.

Um zukünftig mit der Anzahl der Hosts skalieren zu können, wurde der DNS so konzipiert, daß Administration und Speicherung der Informationen verteilt sind. Jede Örtlichkeit kann die sie betreffenden Informationen lokal verwalten.

Um Namenskonflikte aufgrund der dezentralen Administration zu vermeiden (z.B. nennen viele Verwalter ihren Server schlicht „server“), ist der Namensraum im DNS hierarchisch unterteilt. Entlang der Hierarchiestufen können Abschnitte des Namensraums an andere Stellen vergeben werden. Der übergeordnete Teil des Namens sorgt dafür, daß Namen global eindeutig sind. So bildet die Hierachie des DNS eine Baumstruktur, deren Wurzel (root) unbenannt ist. Abbildung 1.12 zeigt einen Ausschnitt der Hierarchie.

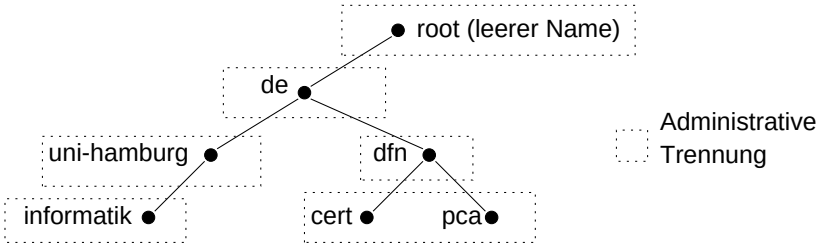


Abbildung 1.12: Hierarchischer Aufbau des DNS

Bei der Notation der Namen im DNS wird zwischen dem lokalen oder relativen Namen und dem absoluten Namen unterschieden. Der absolute Name wird durch Konkatenation der Namensbestandteile aller Hierarchiestufen gebildet, wobei Punkte als Trennzeichen eingefügt werden. In Abb. 1.12 ist das bespielsweise *informatik.uni-hamburg.de.* für den linken Unterbaum. Da der Name der Wurzel das leere Wort ist, endet die Notation eines vollständigen Namens auf einen Punkt. Beim relativen Namen wird nur der Namensbestandteil relativ zu einem bestimmten Punkt in der Baumstruktur angegeben.

Das DNS-System hat eine Client-Server Architektur. Der Client (Resolver) stellt seine Anfragen zunächst an einen lokalen Server (Nameserver). Dieser kann entweder die Anfrage selbst beantworten und selbständig weitere Nameserver abfragen (sog. *recursive Query*), oder dem Resolver einen Verweis an andere Nameserver zurückliefern. Diese kann der Resolver dann direkt kontaktieren (*non-recursive Query*).

Als Transportprotokoll wird normalerweise UDP benutzt, Abfragen über TCP sind jedoch möglich. Für beide Protokolle ist der Port 53 als sog. „Well Known Service“ reserviert.

1.7 Zusammenfassung

Im Verlaufe dieses Artikels wurden die wichtigsten Protokolle der TCP/IP Suite dargestellt und ein Einblick in ihren inneren Aufbau und die UNIX/Socket Programmierschnittstelle gegeben. Die in Abschnitt 1.3 vorgestellten Protokolle IP und ICMP bilden die Grundlage für die in Abschnitt 1.4 und

1.5 vorgestellten Protokolle TCP und UDP der Transportschicht, welche von den Benutzern direkt über ihre Programmierschnittstelle verwendet werden können. Im letzten Abschnitt wurde kurz die Verwendung symbolischer Namen und ihres APIs eingeführt.

TCP/IP ist seit einigen Jahren die am häufigsten verwendete Protokoll-Suite. Obwohl andere Protokoll-Suiten existieren, z.B. OSI und SNA, die teilweise bessere Lösungen (z.B. im Bereich Dienstqualitäten) bereitstellen, haben sich diese nicht im selben Maße durchsetzen können. TCP/IP hat einige Schwächen, vor allem im Bereich Sicherheit und bei der Unterstützung von Dienstqualitäten. Mit ersterem werden sich weitere Artikel näher beschäftigen.

Literaturverzeichnis

[Comer 1995] Comer, D.: „Internetworking with TCP/IP, Volume 1: Principles, Protocols and Architecture“, 3. Auflage, Prentice-Hall International 1995

[CORBA] Object Managment Group: „CORBA 2.2 Specification“, 2. 3. 1998, Online unter `http://www.omg.org/corba/corbaiiop.htm`

[Santi 1998] Santifaller, M.: „TCP/IP und ONC/NFS, Internetworking mit UNIX“, 4. Auflage, Addison-Wesley

[RFC 791] Postel, J.: „Internet Protocol (IP)“, 1981, Online unter `ftp://ftp.ietf.org/rfc/rfc0791.txt`

[RFC 792] Postel, J.: „Internet Control Message Protocol (ICMP)“, 1981, Online unter `ftp://ftp.ietf.org/rfc/rfc792.txt`

[RFC 793] Postel, J.: „Transmission Control Protocol“, 1981, Online unter `ftp://ftp.ietf.org/rfc/rfc0793.txt`

[RFC 768] Postel, J.: „User Datagram Protocol (UDP)“, 1980, Online unter `ftp://ftp.ietf.org/rfc/rfc0768.txt`

[RFC 1034] Mockapetris, P.: „Domain names - concepts and facilities“, 1987, Online unter `ftp://ftp.ietf.org/rfc/rfc1034.txt`

[RFC 1035] Mockapetris, P.: „Domain names - implementation and specification“, 1987, Online unter `ftp://ftp.ietf.org/rfc/rfc1035.txt`

[RFC 1831] Srinivasan, R.: „RPC: Remote Procedure Call Protocol Specification Version 2“, August 1995, Online unter `ftp://ftp.ietf.org/rfc/rfc1831.txt`

[RFC 1832] Srinivasan, R.: „XDR: External Data Representation Standard“, August 1995, Online unter `ftp://ftp.ietf.org/rfc/rfc1832.txt`

Kapitel 2

Überblick Sicherheitsprobleme

Olaf Gellert
Universität Hamburg
Fachbereich Informatik

Zusammenfassung

Dieser Artikel gibt einen Überblick über die wesentlichen Sicherheitsprobleme in vernetzten Systemen. Zunächst werden die Eigenschaften sicherer Systeme vorgestellt. Es folgt eine Klassifizierung von verschiedenen Arten von Sicherheit. Danach werden jeweils zur hostbasierten Sicherheit und zur Netzwerksicherheit Schwächen der Systeme aufgezeigt und Angriffe vorgestellt, die diese Schwächen ausnutzen.

2.1 Einleitung

Mit der wachsenden Vernetzung von Computern und dem vermehrten Bereitstellen von kommerziellen Diensten über Netzwerke steigt auch die Anzahl der Angriffe auf die Computersysteme. Lag die Zahl der vom Computer Emergency Response Team gemeldeten Sicherheitsvorfälle im Jahre 1989 nur bei 132, so waren es im Jahre 1996 bereits 2573, wobei die Zahl der betroffenen Netzwerke bei mehr als 40.000 lag [Kyas 1998, S. 23]. Durch die zunehmende Verarbeitung sensitiver Daten auf Computern sind die Folgekosten von Angriffen deutlich gestiegen [Kyas 1998, S. 55].

Die meisten dieser Angriffe nutzen Schwächen der Betriebssysteme, Netzwerkprotokolle und Netzwerkhardware aus. Insbesondere im Bereich der Netzwerkprotokolle und ihrer Implementationen sind seit Mitte der achtziger Jahre viele Schwächen aufgezeigt geworden [Morris 1985, Bellovin 1989, Joncheray 1995].

Um das Risiko eines Angriffs einschätzen zu können, sind detaillierte Kenntnisse der Systemschwächen nötig. Dieser Artikel soll daher einen Überblick über die Sicherheitsprobleme von vernetzten Systemen geben. Dazu wird zunächst auf den Begriff der Sicherheit eingegangen und eine Klassifikation von Sicherheit gegeben. Darauf folgt eine Beschreibung der wichtigsten Schwachpunkte und Probleme anhand von Beispielen.

2.2 Sicherheit

Beim Einsatz von Computern gibt es i.d.R. zwei Zielsetzungen, die den Begriff der Sicherheit definieren:

- Verhinderung unbefugter Aktivitäten an, mit oder durch das eigene System.
- Gewährleisten gewollter Aktivitäten an, mit oder durch das System.

Zunächst muß also definiert werden, welche Aktivitäten im Umgang mit den Systemen gewollt und nötig sind und welche Handlungen diesen Zielen entgegenstehen. Dies geschieht ganz allgemein in einer Policy und detaillierter in Standards und Handbüchern [Großklaus 1999]. Gibt eine Policy die Grundregeln für die Systemnutzung vor, so bleibt das Problem bestehen, daß sich nicht unbedingt jeder Anwender an diese Regeln hält. Dies gilt insbesondere bei Systemen, die nicht nur einer beschränkten Gruppe von Mitarbeitern zugänglich sind, sondern über ein Netzwerk auch Zugriff von außen ermöglichen. Der Zugriff auf die Ressourcen des Systems wie z.B. Daten, Netzwerk, Computer oder Drucker muß also durch das System selbst so eingeschränkt werden, daß zumindest grobe Verstöße gegen die Policy unmöglich sind.

Folgende Eigenschaften sollen bei der Umsetzung eines Sicherheitskonzeptes durchgesetzt werden [Stallings 1995]:

Verfügbarkeit: Nutzungsmöglichkeit der Betriebsmittel für gewollte Aktivitäten

Vertraulichkeit: Einschränkung des Zugriffs auf Daten

Integrität: Unverändertheit von Daten

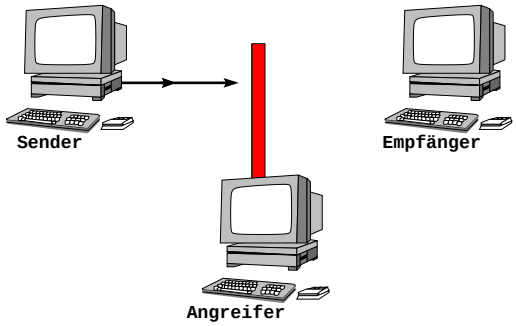
Authentizität: Identität des Verfassers

Der Aspekt der Korrektheit von Systemen soll hier nicht betrachtet werden. Abbildung 2.1 zeigt exemplarisch die Angriffsmöglichkeiten auf, die diese Eigenschaften verletzen.

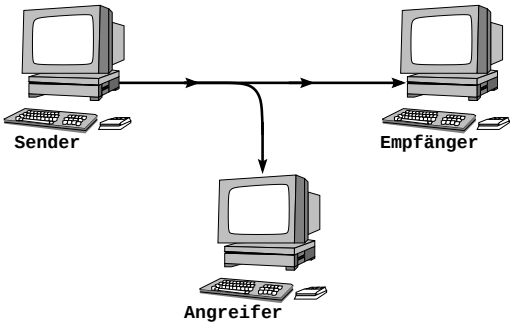
2.2.1 Historische Entwicklung von Sicherheitsproblemen

Die Durchsetzung von Sicherheit erweist sich in heutigen Systemen als schwierig. Dies liegt daran, daß beim Entwurf dieser Systeme die Funktionalität das hauptsächliche Kriterium war und auch noch ist. Der Entwurf soll also zunächst nur sicherstellen, daß das System seine Funktion erfüllt. Die typischen Umgebungen der achtziger Jahre bestehen zum Großteil aus isolierten Einzelplatzrechnern, die jeweils einem bestimmten Benutzer als Arbeitsplatz dienen. Dies erfordert nur eine begrenzte Verwaltung der Betriebsmittel, da alle Ressourcen des Rechners dem einzigen Benutzer uneingeschränkt zur Verfügung stehen. In der Regel gibt es auf einem solchen System keine Benutzerverwaltung und keine Trennung zwischen den Dateien und Prozessen des Benutzers und des Betriebssystems. Die Grundvoraussetzung für einen geregelten Betrieb eines solchen Systems ist also die Kooperation des Anwenders, der z.B. keine Systemdateien löschen darf.

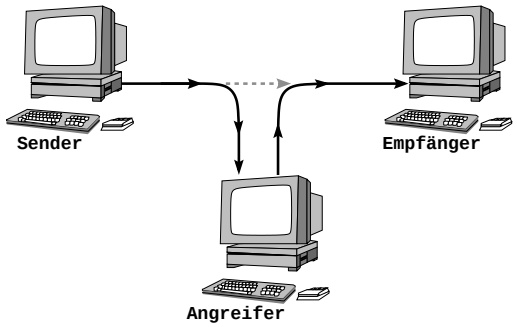
Die Einführung der verteilten Betriebsmittelnutzung (also gemeinsames Nutzen von Druckern, Dateien etc.) erforderten einen Neuentwurf der Systeme. Auch bei diesem Entwurf stand einzig die Funktionalität der Systeme im Vordergrund. Somit wurden Protokolle und Software nur soweit umgestaltet,



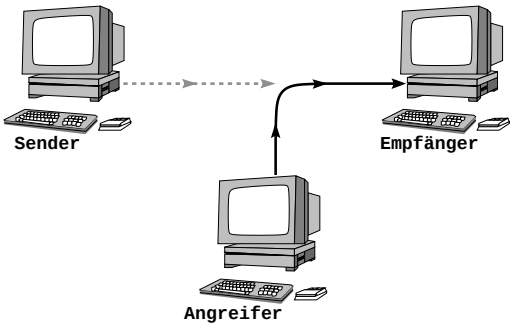
Denial of Service: Angriff auf die Verfügbarkeit



Sniffing: Angriff auf die Vertraulichkeit



Man in the Middle: Angriff auf die Integrität



Spoofing: Angriff auf die Authentizität

Abbildung 2.1: Verletzungen gewünschter Systemeigenschaften

daß Unfälle durch gleichzeitigen Zugriff auf Betriebsmittel verhindert wurden. Beispielsweise wurden Zugriffe auf Drucker durch Warteschlangen sequentialisiert und Dateizugriffe durch exklusiven Zugriff beschränkt. Auch hierbei ist die Kooperation der Anwender und Anwendungen notwendige Voraussetzung für einen ordnungsgemäßen Betrieb.

Auf Grund dieser historischen Entwicklung verfügen heutige Systeme i.d.R. nicht über entsprechende Mechanismen, um Sicherheit durchzusetzen. Der Entwurf eines sicheren Systems muß unkooperatives Verhalten von Benutzern als Tatsache akzeptieren und dieses verhindern können.

Als Beispiel für die Verletzlichkeit heutiger Systeme sei ein simpler Angriff auf die Verfügbarkeit eines PostScript-Druckers genannt. Der in Abbildung 2.2 gezeigte Druckjob versetzt einen PostScript-Drucker in eine Endlosschleife, so daß keine weiteren Druckjobs mehr bearbeitet werden können, bis der Druckjob am Drucker selbst manuell abgebrochen wird.

```
%!PS
{} loop showpage
```

Abbildung 2.2: Beispiel: Endlosschleife eines PostScript-Druckers

Ein weiteres einfaches Beispiel ist das Sperren von Dateien, ohne diese wieder freizugeben.

2.3 Klassifikation von Sicherheit

In den folgenden zwei Abschnitten erfolgt eine Klassifikation von Angriffsarten und von Arten der Sicherheit.

2.3.1 Angriffsarten

Die in Abbildung 2.1 gezeigten Angriffe lassen sich auf Grund ihrer Eigenschaften in folgende zwei Klassen einteilen:

Passive Angriffe: Abhören

Aktive Angriffe: Unterbrechung, Modifikation, Fälschung

Die passiven Angriffe sind i.d.R. nur sehr schwer zu entdecken, da sie keine sichtbare Veränderung des Systems mit sich bringen. Die einzigen Entdeckungsmöglichkeiten sind, den Täter auf frischer Tat zu ertappen (z.B. einen heimlich ans Netz angeschlossenen Rechner) oder spürbare Folgen des Abhörens (jemand kennt offensichtlich vertrauliche Daten des Unternehmens). Passiven Angriffen kann man jedoch vorbeugen, indem sensitive Daten nur verschlüsselt gespeichert und übertragen werden.

Die aktiven Angriffe weisen gegenteilige Eigenschaften auf. Sie sind kaum zu verhindern, da dies die Kenntnis aller Angriffsmöglichkeiten voraussetzt. Weiterhin müßte jedes übertragene Paket inhaltlich analysiert werden. Hier hilft nur, diese Angriffe zu entdecken und entsprechende Maßnahmen gegen den Täter einzuleiten (z.B. Strafverfolgung). Solche Maßnahmen sollten langfristig auch abschreckende Wirkung haben. Im Falle eines Angriffs aus einem externen Netz ist es jedoch oftmals unmöglich, den Täter ausfindig zu machen; Firewalls sollten zur Erhöhung der Sicherheit des internen Netzes eingesetzt werden. Im Gegensatz zu den passiven Angriffen können aktive Angriffe jedoch erkannt

werden. Voraussetzung dafür sind Maßnahmen, die ungewöhnliche Vorfälle auf den Systemen protokollieren. Dies geschieht in Form eines ausführlichen *Auditing*, das sowohl ein Mitprotokollieren des Netzwerkverkehrs (z.B. Meldungen über ungewöhnlich viele ICMP-Pakete) als auch das Erkennen von Veränderungen von wichtigen Systemdateien (mittels Prüfsummen, die aus den Originaldateien berechnet werden, wie es z.B. das Werkzeug `tripwire` anbietet) beinhalten sollte.

2.3.2 Arten der Sicherheit

Die Möglichkeiten, Sicherheit in einer Umgebung vernetzter Rechner zu gewährleisten, lassen sich einteilen nach den Schwachstellen, die für Angriffe ausgenutzt werden können. Angriffspunkte sind die einzelnen logischen Ebenen, aus denen das System besteht. Auf jeder dieser Ebenen muß das System abgesichert werden.

Physikalische Sicherheit

Angriffe können jederzeit erfolgen, wenn der Angreifer physikalischen Zugang zum System hat. Schwachstellen sind hierbei öffentlich zugängliche Systemkomponenten, also Rechner, Router, Switches, Verteilerschränke und Netzkabel. Beispiele für Angriffe auf die Verfügbarkeit, die Vertraulichkeit, die Authentizität oder die Integrität sind z.B. die folgenden:

- Ausbau von Festplatten
- Neustart eines Systems von einer Diskette, um unkontrollierten Zugriff auf die Festplatten zu erlangen
- Anschluß eines Rechners an das Netz, um den Netzwerkverkehr abzuhören
- Umkonfigurieren von Routern und Switches
- Versenden von Emails vom Account eines Benutzers, der sich nicht „ausgeloggt“ hat

Nicht zuletzt sind öffentlich zugängliche Büros eine Möglichkeit, nach Paßwörtern von Mitarbeitern (z.B. auf Zetteln in der Umgebung des Rechners) oder vertraulichen Daten in gedruckter Form zu suchen. Eine Beschränkung des physikalischen Zugangs zum System ist also Grundvoraussetzung für einen gesicherten Betrieb.

Hostsicherheit

Jeder einzelne Rechner eines Systems bietet Ansatzpunkte für einen Angriff. Ursache dafür sind Schwächen in Entwurf und Implementierung der Algorithmen. Wesentliche Schwachstellen sind hierbei der Paßwort-Mechanismus, Systemaufrufe, SetUID-Programme und unsichere Anwendungen und Dienste. Angriffe erfolgen i.d.R. über das Netzwerk, so daß insbesondere die Dienste, die Netzwerkzugriffe ermöglichen, abgesichert werden müssen. Die Sicherheitsmaßnahmen, die die Sicherheit eines einzelnen Rechners gewährleisten sollen, werden unter dem Begriff der *Hostsicherheit* zusammengefaßt.

Netzwerksicherheit

Auch das Netzwerk selbst ist eine Schwachstelle, die das System angreifbar macht. Dies betrifft zum einen die Protokolle, die das Versenden und Empfangen von Daten ermöglichen, zum anderen die verwendete Technologie und Topologie eines Netzwerkes. Zu den entsprechenden Angriffsmöglichkeiten zählen hier z.B. das Fälschen von Absenderadressen, das Verändern der dynamischen Routingtabellen eines Rechners oder Routers, um den Datenstrom umzuleiten, oder die Unterbrechung des Netzwerkzugangs. Maßnahmen zur Beseitigung dieser Angriffsmöglichkeiten dienen der *Netzwerksicherheit*. Die exakte Trennung zwischen Hostsicherheit und Netzwerksicherheit ist hierbei genau so schwierig wie die Trennung zwischen Netzwerk und Rechner selbst, der Übergang ist fließend.

Weitere Einflüsse auf die Sicherheit

Es gibt weitere Faktoren, die die Sicherheit eines Systems beeinflussen. Insbesondere menschliche Schwächen lassen sich ausnutzen, um an sicherheitsrelevante Informationen zu gelangen. Diese Art von Angriffen wird häufig als *Social Hacking* oder *Social Engineering* bezeichnet. Hierzu gehört das Auskundschaften von Paßwörtern im Gespräch, oft per Telefon. Oftmals geben sich Angreifer als Systemadministratoren aus und erklären, sie bräuchten das Paßwort des Angerufenen für dringliche administrative Aufgaben. Insbesondere in großen Firmen kennen die Mitarbeiter die Systembetreuer i.d.R. nicht und geben ihr Paßwort preis. Weiterhin sind vielen Mitarbeitern die Gefahren und Sicherheitsrisiken des Systems nicht bewußt. Auch Systemadministratoren einzelner Abteilungen können unabsichtlich durch leichtfertige Konfiguration einzelner Rechner die Sicherheit des Gesamtsystems gefährden. Eine gute Schulung der Mitarbeiter ist deshalb wichtig, um das Sicherheitsbewußtsein der Anwender zu schärfen. Auch eine ständige Weiterbildung der Systemverwalter trägt dazu bei, daß Sicherheitslücken frühzeitig erkannt und geschlossen werden oder gar nicht erst entstehen.

2.4 Angriffe auf die Hostsicherheit

In diesem Kapitel werden Beispiele für die Möglichkeiten von Angriffen gegeben. Zunächst werden Angriffe auf die Hostsicherheit (d.h. auf Paßwort-Mechanismus, Systemaufrufe, Dienste und Anwendungen) aufgeführt, im nächsten Kapitel folgt die Beschreibung von Angriffen auf die Netzwerksicherheit, die i.d.R. die Schwächen von Netzwerkprotokollen und -diensten ausnutzen.

2.4.1 Paßwort-Mechanismus

Die Identifikation und Autorisierung von Benutzern erfolgt heutzutage weitestgehend über einen Paßwort-Mechanismus. Dies erfordert, daß dem System eine Liste zur Verfügung stehen muß, die den Benutzernamen die zugehörigen Paßwörter zuordnet. Unter Unix-Systemen ist dies die Datei `/etc/passwd`. Ein Eintrag in diese Datei sieht wie folgt aus:

```
gellert:1/S1ocEzgyZA2:10000:10000:Olaf Gellert:/home/olaf:/bin/bash
```

Der Eintrag besteht aus dem Benutzernamen, dem verschlüsselten Paßwort, einer Benutzer- und einer Gruppenidentifikationsnummer, einem beschreibenden Feld (in dem i.d.R. der volle Name des Benutzer steht), der Angabe des Heimatverzeichnisses des Benutzers und seiner Login-Shell. Diese Datei ist für jeden Benutzer lesbar, damit z.B. die Angabe über das Heimatverzeichnis auch Anwendungen zur Verfügung steht. Dies birgt jedoch auch die Gefahr, daß versucht werden kann, die Paßwörter anderer

Benutzer zu erraten. Dazu wählt man ein mögliches Paßwort aus, verschlüsselt es mit dem Systemcall `crypt` und vergleicht das Ergebnis mit den Paßwortfeldern der `/etc/passwd`. Kommt die gesuchte Verschlüsselung vor, so hat man das richtige Paßwort geraten. Durch das Verschlüsseln ganzer Lexika lassen sich effizient häufig benutzte Paßwörter, wie z.B. Vornamen oder Nachnamen, herausfinden. Solche Angriffe lassen sich erschweren, indem das Paßwort gemeinsam mit einem Zeitstempel verschlüsselt wird, der ebenfalls in der Paßwortdatei abgelegt wird (sog. Versalzen, engl. *Salt*). So werden identische Paßwörter, die zu unterschiedlichen Zeiten angelegt werden, unterschiedlich verschlüsselt. Verhindern lassen sich Angriffe auf den Paßwort-Mechanismus durch den Einsatz von sog. *Shadow Passwords*, wobei die Paßworteinträge in `/etc/passwd` durch ein 'x' ersetzt und die Paßwörter in der Datei `/etc/shadow` abgelegt werden. Diese Datei ist nur für das System und den Systemadministrator lesbar.

Weiterhin kann versucht werden, über ein *Trojanisches Pferd*[1] an Paßwörter zu gelangen. Angeführt sei hier nur ein Programm, das ebenso wie das Login-Kommando einen Benutzernamen und ein Paßwort einliest, diese jedoch nicht nur an das System übermittelt, sondern zusätzlich im Klartext in eine Datei einträgt oder per Mail verschickt.

Paßwörter können oftmals auch leicht durch das Abhören des Netzwerkverkehrs erhalten werden, da Protokolle wie z.B. `ftp` oder `telnet` die Paßwörter unverschlüsselt an den Zielrechner übertragen.

2.4.2 Systemaufrufe

Jedes Betriebssystem bietet den Anwendungen eine Bibliothek von Funktionen an, die den Zugriff auf die Hardware ermöglichen (z.B. `open`, `read`, `write` und `close` zum Zugriff auf Dateien). Diese Funktionen überprüfen i.d.R. nicht die Korrektheit von übergebenen Parametern, was bei unerwarteten Parametern zum Systemabsturz führen kann. Zudem weisen die Systemfunktionen häufig konzeptionelle Schwächen in Bezug auf die Sicherheit auf. Folgende Beispiele mögen dies illustrieren:

gets: Liest eine Zeichenkette von der Tastatur in einen angegebenen Pufferspeicher ein. Dabei wird die Größe des Pufferspeichers nicht als Parameter übergeben. Somit kann der Speicherbereich hinter dem Puffer verändert werden, indem eine Zeichenkette eingegeben wird, die größer als der Pufferspeicher ist (sog. *buffer overruns*).

malloc: Fordert eine angegebene Menge Speicher vom System an. Der reservierte Speicher wird jedoch nicht vorher vom Betriebssystem gelöscht, so daß hier oftmals Informationen anderer Prozesse zu finden sind, die diesen Speicher vorher benutzt haben.

2.4.3 SetUID-Programme

Eine weitere Gefährdung des Systems stellen Programme dar, die mittels SetUID mit `root`-Rechten versehen sind. Dies dient dazu, Programme von Anwendern ausführen zu lassen, die für ihren Ablauf jedoch `root`-Rechte benötigen, da sie privilegierte Systemfunktionen aufrufen. Ein Beispiel ist das Programm `lpc`, mit dem Drucker angehalten oder gestartet werden können. Die SetUID-Programme erhalten beim Aufruf die Rechte ihres Eigners, i.d.R. ist dies der Systemverwalter `root`.

[1] Ein Trojanisches Pferd ist ein Programm, das nicht nur die Aufgaben erledigt, für die es vorgeblich geschaffen wurde, sondern heimlich weitere Tätigkeiten ausübt, z.B. Sammeln von Paßwörtern und anderen vertraulichen Informationen.

Eine Gefährdung bedeuten diese Programme, wenn es möglich ist, aus ihnen heraus unkontrolliert andere Prozesse zu starten, die dann ebenfalls `root`-Rechte erhalten. Bei der Ausführung von Shell-Scripten mit SetUID `root` kann es möglich sein, die Ausführung des Scriptes zu beenden, ohne daß die Shell selbst beendet wird (sog. *Shell Escapes*). Damit steht einem Anwender eine privilegierte Shell zur Verfügung. Oftmals verändern SetUID-Programme auch Dateien, deren Name nicht immer statisch im Programmtext abgelegt ist, so daß evtl. die `/etc/passwd` oder andere wichtige Systemdateien überschrieben werden können.

Werden solche SetUID-Programme wirklich benötigt, so sollten sie in einer *Change Root* Umgebung ablaufen, d.h. sie erhalten keinen Zugriff auf das reale Wurzelverzeichnis des Dateisystems, sondern erhalten ein entsprechend vorbereitetes Unterverzeichnis als Wurzel. Somit können sie nicht mehr auf die Systemdateien in höheren Verzeichnissen (z.B. `/etc`) zugreifen.

2.4.4 Anwendungen

Eine weitere Gefahr stellen Anwendungen dar, die eine Kommandosprache besitzen. Dies sind z.B. alle gängigen Office-Pakete, die über eigene Makrosprachen (z.B. Visualbasic) verfügen. Beim Laden eines Dokumentes kommen Befehle zur Ausführung, die im Dokument selbst stehen. Bei Dokumenten von fremden Personen gibt es keine Gewißheit, daß keine schädlichen Befehle wie z.B. das Löschen von Dateien im Dokument versteckt sind. Es gibt sogar Viren, die in solchen Kommandosprachen verfaßt wurden (Word Makro Viren).

Dasselbe gilt auch für PostScript Interpreter wie z.B. GhostScript. Auch in PostScript gibt es Befehle, die den Zugriff auf Dateien und das Starten von Prozessen erlauben (`deletefile`, `renamefile`, `file`, `%pipe`). Ebenso kann das Ausführen von unbekanntem Code (also Java, JavaScript, ActiveX) durch einen WWW-Client gefährlich sein.

Weiterhin gibt es keine Sicherheit bei der Ausführung von Anwendungen, die als ausführbare Dateien aus dem Internet importiert wurden. Einige Trojanische Pferde wurden auf diese Weise bereits verbreitet.

2.4.5 Dienste

Bei der Installation eines Betriebssystems sind viele Dienste bereits vorkonfiguriert, um dem Benutzer ein eigenes Einrichten dieser Dienste abzunehmen. Dies bedeutet jedoch, daß viele Dienste bei der Systeminitialisierung gestartet werden, die evtl. gar nicht benötigt werden und die von Angreifern benutzt werden können. Beispiele sind `ftpd` und `httpd`; die wenigsten Rechner sollen `ftp`-, `http`- oder Druck-Serverdienste anbieten. Auch wenn manche Dienste benötigt werden, so sind diese selten sicher konfiguriert. Ein Nacharbeiten an den Einstellungen sollte gewährleisten, daß diese Dienste nur beschränkt von anderen Rechnern benutzbar sind. Wichtig ist z.B. die Einschränkung von Lesezugriffen und insbesondere Schreibzugriffen von außen (für `ftp` und `http`).

Auch das lokale Ausführen von Programmen auf externe Veranlassung (CGI-Scripte von Webservern, Remote-Logins) sollte soweit wie möglich unterbleiben. Weiterhin bedarf es der Beschränkung der über den Internetdaemonen (`inetd`) angebotenen Dienste, da dieser bei Anfragen von außen die entsprechenden Dienste automatisch aktiviert (also z.B. bei ftp-Anfragen einen ftp-Daemonen startet).

Besondere Schwachstelle sind die sog. „`r`-Dienste“ `rsh`, `rcp` und `rlogin`. Sie ermöglichen ein Login bzw. die Kommandoausführung von fremden Rechnern aus ohne Überprüfung eines Paßworts. Dies geschieht über die Konfiguration von sog. *Trusted Hosts* in den Dateien `/etc/hosts.equiv` oder

`~/.rhosts`, die eine Liste von vertrauenswürdigen Systemen beinhalten. Durch `~/.rhosts` kann jeder Anwender einstellen, von welchen Systemen aus ein Login ohne Paßwortabfrage möglich sein soll. Der Zugriff auf ein System sollte jedoch nur durch den Systemadministrator konfigurierbar sein, die „r-Dienste“ ermöglichen die (ungewollte) Delegation von Rechten an die Benutzer des Systems.

`rsh` und `rlogin` erlauben eine Ausführung von Befehlen, wenn die Anfrage folgende Kriterien erfüllt [Cheswick & Bellovin 1996]:

- Die Verbindung stammt von einem privilegierten Port
- Name und IP-Adresse des Quellsystems korrespondieren:
  `gethostbyname(gethostbyaddr(Absenderadresse)) == Absenderadresse`
- Quellsystem steht in `/etc/hosts.equiv` oder in `~/.rhosts`

Da sich Absenderadressen einfach fälschen lassen und jedes Unix-System über Standard-Accounts verfügt, ist diese Authentisierung unzureichend, die Dienste `rsh` und `rlogin` sollten durch ihre sichereren Äquivalente `ssh` und `slogin` ersetzt werden oder ganz abgeschaltet werden, falls sie nicht benötigt werden.

2.5 Angriffe auf die Netzwerksicherheit

In diesem Kapitel werden Angriffe auf die Netzwerksicherheit vorgestellt. Diese nutzen i.d.R. Schwächen in den Protokollen selbst oder in ihren Implementierungen aus. Alle heutzutage verwendeten Protokolle bieten Angriffsmöglichkeiten. Aus diesem Grunde werden die Angriffe nach Protokollebenen aufgeführt, d.h. von den unteren, hardware-nahen Ebenen angefangen geht die Aufzählung bis in die Anwendungsebene. Abbildung 2.3 zeigt die logische Anordnung der Internet-Protokolle. Es gibt weitere Schwachstellen auf der Ebene der Netzwerkhardware (z.B. ist Ethernet wenig geeignet, Vertraulichkeit zu gewährleisten, da es sich um ein Broadcast-Medium handelt), auf die hier jedoch nicht näher eingegangen wird.

2.5.1 ARP

Das *Address Resolution Protocol* (ARP) ermöglicht die Zuordnung von physikalischen Adressen zu IP-Adressen. Dies ist notwendig, da das verwendete Netzwerkmedium nicht mit IP-Adressen arbeitet, sondern in der Regel eigene Adreßformate benutzt.

Die Umsetzung geschieht wie folgt: Ein Rechner, der die Ethernet-Adresse zu einer gegebenen IP-Adresse benötigt, sendet einen Ethernet Broadcast, der die gefragte IP-Adresse und die Ethernet-Adresse des Absenders enthält, an alle angeschlossenen Rechner (sog. *Arp Request*). Die anderen Rechner empfangen dieses Paket und der Rechner, zu dem die angefragte IP-Adresse gehört, antwortet, indem er seine eigene Ethernet-Adresse an den anfragenden Rechner sendet (*Arp Reply*). Damit nicht für jedes zu übertragende Paket ein *Arp Request* benötigt wird, werden die bisherigen Antworten im sog. *Arp Cache* für eine Weile zwischengespeichert [Plummer 1982].

Dieses Protokoll enthält zwei Schwachstellen, die Ansatzpunkte für Angriffe sein können. Zum einen ist es möglich, beliebig viele *Arp Requests* zu versenden. Diese beschäftigen alle an das Segment angeschlossenen Rechner. Dieser Denial-of-Service-Angriff wird als *Arp-Sturm* bezeichnet. Zum anderen gibt es keine Möglichkeit, die Korrektheit der Antworten sicherzustellen. Ein Angreifer könnte

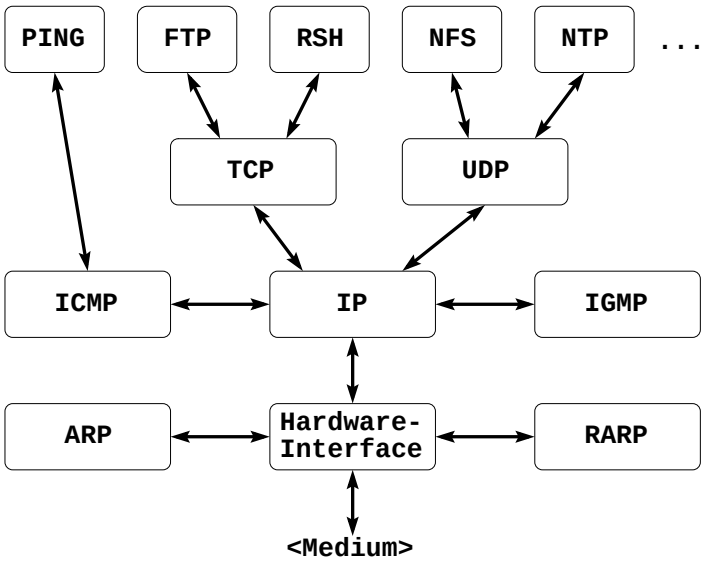


Abbildung 2.3: Protokollstapel der Internet-Protokolle

auf einen beliebigen *Arp Request* mit seiner eigenen Ethernet-Adresse antworten und so die folgenden Datagramme zu sich umleiten. Diese Umleitung bleibt im Cache noch eine Weile erhalten, so daß mit geringem Aufwand die Kommunikation umgeleitet werden kann, um *Man in the Middle*-Angriffe vorzubereiten (vgl. Abbildung 2.1).

2.5.2 IP

Auch das Internetprotokoll, das für das Routing von Paketen zuständig ist, ist angreifbar. Einen Ansatzpunkt für einen Angriff bietet der Fragmentierungsmechanismus des Internetprotokolls. Muß ein Paket fragmentiert werden, so wird es erst beim Empfänger wieder zusammengesetzt. Die Fragmente behalten die ursprüngliche Identifikationsnummer des fragmentierten Paketes bei, es wird ein Fragmentbit in den Flags gesetzt und der Fragmentoffset, der den Byteabstand zum ersten Byte des ursprünglichen Pakets angibt, wird eingetragen. Das Längenfeld wird auf die Länge des einzelnen Fragmentes gesetzt. (Abbildung 2.4 zeigt das Format eines IP-Paketes; eine ausführliche Beschreibung der Fragmentierung gibt [Stevens 1994].) Somit ist es möglich, mehrere Fragmente zu generieren, die zusammengesetzt größer sind als 64 KByte, also die erlaubte Größe von IP-Paketen übersteigen [Möller 1999]. Viele ältere IP-Implementationen verursachen Pufferüberläufe beim Zusammensetzen solcher Fragmente, der Empfang eines solchen Paketes führt dann zu einem Systemabsturz. Dieser Denial-of-Service-Angriff wird auch als *Ping of Death* bezeichnet.

Zum anderen kann eine falsche Absenderadresse in ein IP-Paket eingetragen werden (sog. *IP-Spoofing*). Ein Angreifer ist somit in der Lage, Pakete zu versenden, die aus einem bestimmten Netzwerk oder von einem bestimmten Rechner zu kommen scheinen. Eine Autorisierung auf der Basis von IP-Nummern, wie sie durch die `r`-Dienste vorgenommen wird, ist somit unsicher. Häufig lassen sich durch Wahl einer Absenderadresse aus einem internen Netz auch Paketfilter täuschen, die das Paket empfangen und erst beim Absenden prüfen, ob es weitergeleitet werden darf. Das Paket schaut dann auf Grund der gefälschten Absenderadresse so aus, als ob es aus dem internen Netz stammt, und wird weitergeleitet, obwohl es von einem externen Angreifer gesendet wurde.

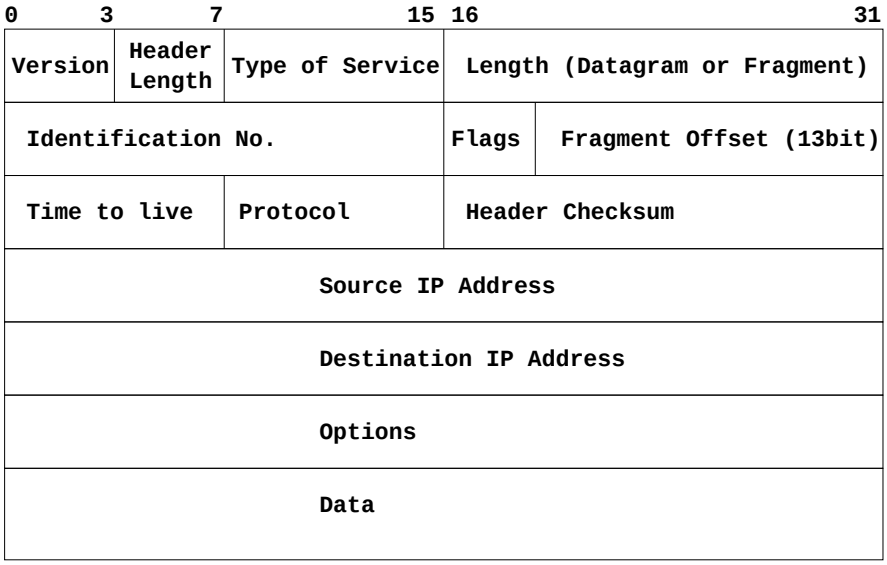
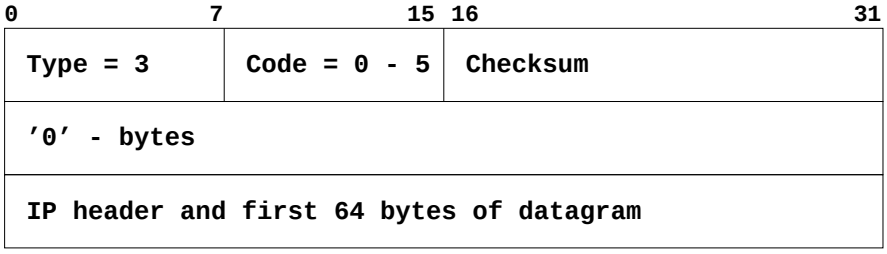


Abbildung 2.4: Format eines IP-Datagramms (aus [Stevens 1994])

2.5.3 ICMP

Das *Internet Control Message Protocol* dient dazu, Kontrollnachrichten auf der IP-Ebene zu versenden. Es ist integraler Bestandteil von IP und informiert den Absender von IP-Datagrammen über aufgetretene Fehler und gibt Routing-Informationen weiter.

Für Angriffe nutzbar ist zum einen die ICMP-Nachricht „*unreachable*“. Sie gibt an, daß ein Subnetz, ein Rechner oder ein Dienst nicht erreichbar ist. Durch einfaches Senden einer „ICMP unreachable“ Nachricht mit einem gefälschten Absender kann ein Ausfall des Empfängers oder eines Teilnetzes vorgetäuscht werden und der Sender nimmt an, der Empfänger sei nicht erreichbar. Einige ältere ICMP-Implementierungen werten auch die Port-Informationen, die im Rumpf des Paketes stehen, nicht aus; sie trennen bei Empfang einer *unreachable* Nachricht alle TCP- und UDP- Verbindungen zu diesem Rechner.

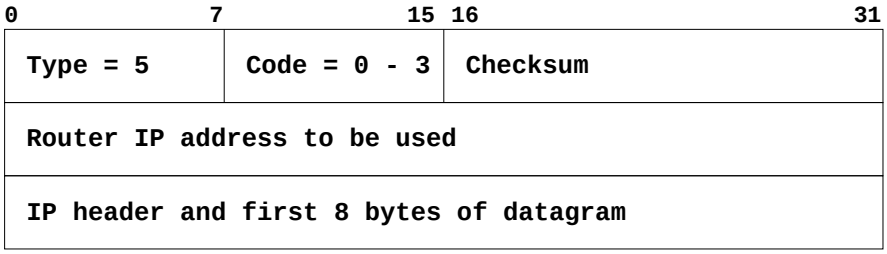


Code:
0 Network unreachable
1 Host unreachable
2 Protocol unreachable
3 Port unreachable
4 Fragmentation impossible
5 Source route failed

Abbildung 2.5: „ICMP unreachable“ Datagramm

ICMP stellt auch Mechanismen bereit, um Routinginformationen weiterzuleiten. Ein Router R1, der

ein Paket vom Rechner C über seine Schnittstelle S1 erhält und feststellt, daß er das Paket über S1 an den Router R2 weiterleiten muß, kann C eine *ICMP redirect* (Abbildung 2.5) Nachricht senden, um ihm mitzuteilen, daß weitere Pakete mit diesem Ziel direkt an den Router R2 geschickt werden können. Ein Angreifer kann ein *ICMP redirect* mit gefälschter Absenderadresse an einen Rechner senden, so daß dieser seine Pakete nicht an einen Router, sondern an den Rechner des Angreifers absendet [Bellovin 1989]. Auch reagieren manche Router auf *ICMP redirect* Nachrichten mit einer Änderung ihrer Routing-Tabellen. Dies sollten sie i.d.R. nicht tun, da die Netztopologie in der näheren Umgebung des Routers bekannt sein dürfte und Informationen über Topologieveränderungen nicht von einem beliebigen, entfernten Rechner stammen können [Cheswick & Bellovin 1996].



Code: **0** for Network **1** for Host
2 for Service **3** for Service & Host

Abbildung 2.6: „ICMP redirect“ Datagramm

2.5.4 TCP

TCP (*Transmission Control Protocol*) bietet die verbindungsorientierte, korrekte Übertragung von Anwendungsdaten an. Zu diesem Zweck setzt TCP bestimmte Mechanismen ein, die mögliche Reihenfolgevertauschungen von Datagrammen auflösen und Übertragungsfehler korrigieren. Hierfür werden die Pakete jeweils mit einer Sequenznummer versehen und empfangene Bytes durch sog. *Acknowledgements* bestätigt. Auch Verbindungen können von fremden Rechnern unterbrochen, gefälscht und sogar übernommen werden. Wesentlicher Mechanismus ist dabei das Fälschen der IP-Absenderadresse. Das Problem des Angreifers besteht darin, daß die Antworten des Servers[2] an den im Paket angegebenen Absender zurückgeschickt werden; sie sind für den Angreifer also i.A. nicht sichtbar. Dies bedeutet, daß die entsprechenden Sequenznummern dieser Antwortsegmente geraten werden müssen, um den Empfang zu bestätigen. Zudem wird ein Angreifer verhindern müssen, daß vom realen Client keine Reset-Pakete an den Server gesendet werden, da ein unvorhergesehenes Paket die Verbindung beenden kann. I.d.R. wird als Absender die IP-Adresse eines Rechners angegeben, der entweder nicht in Betrieb ist oder aber durch einen gezielten Denial-of-Service-Angriff lahmgelegt wurde (z.B. *Syn Flooding*, Abschnitt 2.5.4).

Denial-of-Service-Angriffe über TCP

Mittels TCP läßt sich ein Rechner daran hindern, weitere Verbindungen entgegenzunehmen. Dies geschieht durch das Senden von Verbindungsanfragen (gesetztes SYN Bit), wobei das zurückkommende

[2]Im folgenden wird der Rechner, der einen Verbindungsaufbau initiiert (Systemaufruf `connect`), als Client bezeichnet, und der Rechner, der diese Verbindung entgegennimmt (Systemaufrufe `listen`, `accept`), wird Server genannt.

Bestätigungspaket nicht mehr bestätigt wird (also ein unvollständiger Dreiwege-Handshake). Es entsteht somit eine sog. halboffene Verbindung. TCP Implementationen beinhalten einen sog. *Backlog* Wert, der angibt, wie viele gleichzeitige, halboffene Verbindungen zu einem Socket bestehen können. Dies verhindert, daß für zu viele halboffene Verbindungen entsprechender Speicher alloziert werden muß. Der *Backlog* Wert liegt bei vielen Implementationen unterhalb von zehn. Werden also genügend halboffene Verbindungen initiiert, so ist der angesprochene Dienst bis zum Timeout der Verbindungen blockiert; eingehende Verbindungen werden nicht mehr angenommen.

TCP-Sequenznummern-Angriff

Um eine unerlaubte Verbindung aufzubauen kann als Absenderadresse die eines vertrauenswürdigen Rechners angegeben werden. Da beim Verbindungsaufbau die Bestätigung (SYN und ACK Bit gesetzt) des Empfängers wiederum bestätigt werden muß, muß die Sequenznummer dieses Paketes geraten werden. Bereits 1985 zeigte Morris, daß dies bei vielen TCP-Implementierungen möglich ist, da die initiale Sequenznummer für neue Verbindungen nur langsam und deterministisch verändert wird [Morris 1985].

Ein Angriff geschieht dann in folgenden Schritten:

1. Aufbau von harmlosen, erlaubten Testverbindungen (finger, mail) zum Empfänger, um die vergebenen Sequenznummern und deren Veränderung zu bestimmen.
2. Anforderung einer gefährlichen Verbindung (rlogin) mit gefälschter Absenderadresse durch Paket (`syn i`)
3. Antwortpaket (`syn j, ack i+1`) ist unsichtbar
4. Verbindungsaufbau durch Paket (`ack geschätztes j+1`)

Nun kann weiterhin gesendet werden unter fortlaufendem Erhöhen der Sequenznummer. Eine Verbindung wurde von einem entfernten Rechner unter Angabe eines falschen Absenders aufgebaut.

TCP-Hijacking

Es ist auch möglich, bestehende TCP-Verbindungen von einem unbeteiligten Rechner aus zu übernehmen. Dies wird als Hijacking bezeichnet (engl. *to hijack* = ein Flugzeug entführen). Insbesondere bei Systemen, bei denen beim Verbindungsaufbau die Berechtigung überprüft wird (z.B. durch Einmalpaßwörter oder durch Rückfrage durch einer Firewall), bietet dieser Angriff die Möglichkeit, eine bereits gewährte Verbindung zu übernehmen. Realisiert wird dies durch das Senden eines *Reset*-Paketes mit gefälschter Absenderadresse an den Client. Dieser nimmt an, der Server habe die Verbindung abgebrochen, und beendet seinerseits die Verbindung. Diese Art des Verbindungsabbaus heißt *abortive release*, stellt also einen Abbruch und keinen geregelten, beidseitigen Verbindungsabbau dar (siehe [Stevens 1994]). Der Client kennt also keine bestehende Verbindung mehr, der Server hat vom Verbindungsabbruch jedoch nichts bemerkt. Somit kann nun wie beim Sequenznummernangriff fortgefahren werden, der Angreifer sendet nun an den Server. Abbildung 2.7 zeigt den Ablauf eines solchen Hijacking-Angriffs direkt beim Verbindungsaufbau.

Es besteht auch die Möglichkeit, Verbindungen beidseitig zu übernehmen, so daß sowohl der Client als auch der Server weiterhin glauben, sie würden direkt miteinander kommunizieren, obwohl ein

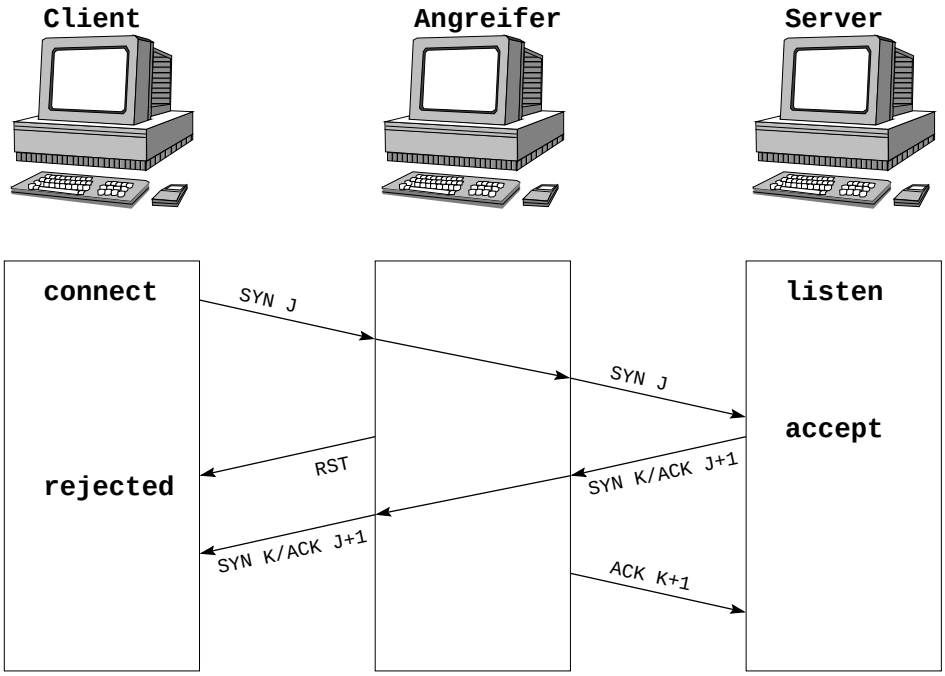


Abbildung 2.7: TCP Hijacking Angriff

Angreifer für beide die Verbindung vortäuscht [Joncheray 1995]. Dies geschieht, indem ein Angreifer durch einen *Reset* an den Server und einen direkt folgenden Neuaufbau der Verbindung dafür sorgt, daß die Sequenznummern von Client und Server nicht mehr synchron sind. Die Bestätigungen des einen werden somit vom anderen verworfen, da sie eine unerwartete Sequenznummer haben. Der Angreifer kann somit für den Client und den Server weiterhin eine Verbindung vorspiegeln, in dem er selbst die Datagramme mit den erwarteten Sequenznummern generiert. Abbildung 2.8 zeigt den Ablauf des Angriffs.

Voraussetzung für diesen Angriff ist i.d.R. die Möglichkeit, die Verbindung abzuhören, um zu wissen, wieviel der Client sendet, sonst kann die Sequenznummer der Bestätigung nicht generiert werden. Innerhalb eines Ethernetsegments oder durch Umlenken von Paketströmen über Routerkonfigurationen ist dies jedoch durchaus möglich. Die bei diesem Angriff erzeugte Menge an Bestätigungen, die jeweils verworfen werden und zu einem erneuten Übermitteln der Sequenznummer führen, stellen eine hohe Netzlast dar, behindern diesen Angriff jedoch nicht, wie folgende Überlegung zeigt:

- Jedes Paket, das auf Grund seiner falschen Sequenznummer verworfen wird, erzeugt eine Bestätigung
- Diese enthält wiederum eine unerwartete Sequenznummer
- Nur ein Paketverlust beendet die Endlosschleife
- Die Pakete enthalten keine Daten, bei Verlust werden sie nicht neu erzeugt
- Das zum Pakettransport verwendete IP ist nicht verlustfrei, insbesondere bei hoher Netzauslastung gehen Pakete verloren.

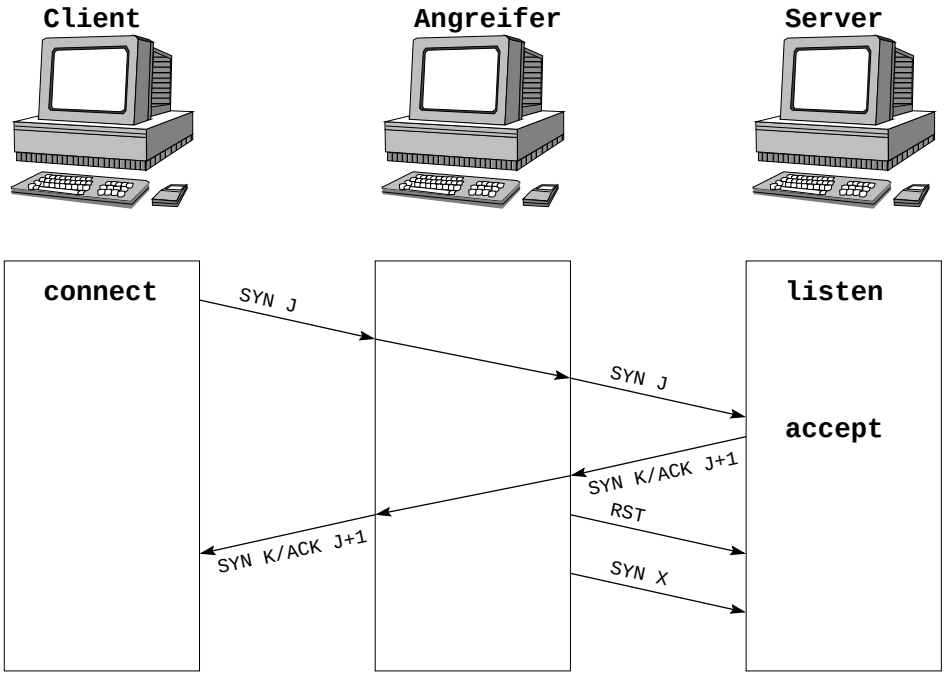


Abbildung 2.8: TCP Hijacking Angriff

- Je mehr solcher Endlosschleifen entstehen, desto mehr Pakete gehen verloren

2.5.5 Network Time Protocol (NTP)

Das *Network Time Protocol* dient zur präzisen Synchronisation von Uhren über das Netzwerk, wobei Rechner mit sehr genauen Uhren (z.B. DCF 77 Funkuhren) als Timeserver auftreten und andere Rechner mit der genauen Uhrzeit versorgen. Durch IP-Spoofing kann auf Rechnern, die die Uhrzeit mittels NTP erhalten, die Systemuhr auf eine falsche Uhrzeit gestellt werden. Dies ermöglicht u.a. sog. *Replay*-Angriffe auf Rechner, die eine zeitabhängige Authentisierung durchführen. Ein durch Abhören erhaltenes Paßwort kann somit ein zweites Mal verwendet werden, indem die Systemuhr des Rechners auf die zum Paßwort gehörende Uhrzeit zurückgesetzt wird [Kyas 1998].

2.5.6 Network File System (NFS)

Das *Network File System* ermöglicht den transparenten Zugriff auf entfernte Dateien und Verzeichnisse. Die älteren Implementierungen setzen auf UDP auf, die Authentisierung beruht auf IP-Adressen (vgl. „r-Dienste“, Abschnitt 2.4.5). Somit ist der Authentisierungsmechanismus anfällig für IP-Spoofing. Beim Öffnen einer Datei über NFS wird vom Server ein Datei-Deskriptor vergeben, die für die weiteren Zugriffe benutzt wird. Diese Datei-Deskriptoren ändern sich nie, da der NFS-Server zustandslos ist. Dies bedeutet, daß nach einem Ausfall und Reboot des NFS-Servers die Clients weiterhin zugreifen können, da die Datei-Deskriptoren auch nach dem Reboot aktuell sind. Dieses Vorgehen mindert zwar die Folgen eines Server-Absturzes, bedeutet jedoch auch, daß ein einmal erhaltener Datei-Deskriptor im Prinzip ewig gültig ist. Eine nachträgliche Veränderung der Zugriffsrechte einer Datei ändert nichts, wenn ein Client den entsprechenden Datei-Deskriptor bereits besitzt. Da diese

Datei-Deskriptoren beim ersten Zugriff auf die Datei an den Client übertragen werden, können sie von einem Angreifer abgehört und verwendet werden, so daß er den Authentisierungsmechanismus von NFS umgehen kann.

2.5.7 Network Information Service (NIS)

Der *Network Information Service* dient zur Verteilung wichtiger Dateien von einem zentralen Server zu den Clients. Dies wird zum Beispiel genutzt, um die Paßwortdatei oder Hostadreßtabellen zentral zu verwalten. Da die Übertragung dieser Dateien im Klartext geschieht, können Angreifer diese Informationen durch Abhören erhalten. Gravierender ist jedoch ein NIS-Server-Spoofing Angriff, wobei der Angreifer durch IP-Spoofing vorgibt, der NIS-Server des Clients zu sein und eine eigene Paßwortdatei überträgt. Somit kann durch einen Angriff sofort jeglicher Zugang zum NIS-Client erlangt werden, da der Angreifer dem Client eine bekannte Paßwortdatei „unterschiebt“.

2.5.8 Domain Name Service (DNS)

Der *Domain Name Service* führt die Zuordnung von IP-Namen zu IP-Adressen durch, da zur Zustellung eines Datagramms immer die IP-Adresse nötig ist. Innerhalb eines lokalen Netzwerkes verfügen die Rechner i.d.R. über eine eigene Zuordnungstabelle (unter Unix `/etc/hosts`). Ist ein Empfänger nicht in dieser Tabelle enthalten, so wird ein festgelegter Rechner (Nameserver) über DNS kontaktiert. Kennt dieser den gefragten Namen, so sendet er die entsprechende IP-Adresse zurück. Ist der Rechnername jedoch unbekannt, so sendet der DNS-Server entweder die Adresse des nächst höheren DNS-Servers der Namenshierarchie an den Client, damit dieser eine erneute Anfrage stellen kann (sog. Iteratives Verfahren), oder aber er befragt selbst den nächsten DNS-Server, um dem Client die Antwort zuzusenden (Rekursives Verfahren). Das bedeutet, daß bei Anfragen nach weit entfernten Rechnernamen die Antwort i.A. von einem externen Rechner stammt. Solche Antworten sind nicht vertrauenswürdig, ein Angreifer könnte solche Anfragen abfangen und seine eigene IP-Nummer zurücksenden. Damit hat er die nachfolgenden Kommunikation zu seinem Rechner umgelenkt [Kyas 1998].

2.6 Zusammenfassung

Wie anhand vieler Beispiele deutlich geworden ist, ist das Funktionieren von Rechnern und Netzwerken nach wie vor von der Kooperation der Anwender abhängig. Es gibt zwar Bestrebungen, bestehende Protokolle sicherer zu machen (z.B. existiert in der Version 6 des Internetprotokolls ein sichererer Authentisierungsmechanismus und die Möglichkeit der verschlüsselten Datenübertragung [Huitema 1996]), jedoch wird es noch eine Weile dauern, bis sich solche neuen Protokolle durchgesetzt haben. Zudem werden für viele Anwendungszwecke neue Protokolle geschaffen, bei deren Entwurf die Sicherheit nach wie vor kaum beachtet wird. Insbesondere die wachsende Komplexität der Systeme macht das Erkennen von Schwachstellen nahezu unmöglich. Zudem gibt es noch immer zahlreiche fehlerhafte Implementationen der verschiedenen Protokolle. Und mit großer Wahrscheinlichkeit werden in vielen Implementationen noch Fehler entdeckt werden.

Hinzu kommen die Schwierigkeiten bei der sicheren Konfiguration der einzelnen Rechner. In Anbetracht der steigenden Anzahl von Anwendungen und Diensten werden diese Konfigurationsprobleme weiterhin zunehmen.

Allerdings gibt es durchaus Möglichkeiten, die Sicherheit zu erhöhen. Der Einsatz von Firewalls, Verschlüsselungs- und Authentisierungsmechanismen trägt dazu bei, viele Sicherheitslücken zu schließen. Jedoch ist eine ständige Schulung von Administratoren nötig, um in puncto Sicherheit auf dem neuesten Stand zu bleiben.

Literaturverzeichnis

[Bellovin 1989] Steven M. Bellovin. *Security Problems in the TCP/IP Protocol Suite*. Computer Communication Review, Vol. 19, Nr. 2, 1989.

[Cheswick & Bellovin 1996] William R. Cheswick, Steven M. Bellovin. *Firewalls und Sicherheit im Internet*. Addison Wesley, Bonn, 1996.

[Großklaus 1999] Axel Großklaus. *Policy, Vorfallsbearbeitung, Schwachstellenanalyse*. Vortrag im Seminar „Sicherheit in vernetzten Systemen“, FB Informatik, Universität Hamburg, Mai 1999.

[Huitema 1996] Christian Huitema. *IPv6: The new Internet Protocol*. Prentice Hall, 1996.

[Joncheray 1995] Laurent Joncheray. *A Simple Active Attack Against TCP*. Proceedings of the Fifth USENIX UNIX Security Symposium, USENIX Assoc, 1995.

[Kyas 1998] Othmar Kyas. *Sicherheit im Internet*. International Thomson Publishing, Bonn, 1998.

[Möller 1999] Klaus Möller. *Grundlagen: Internet Protokolle*. Vortrag im Seminar „Sicherheit in vernetzten Systemen“, FB Informatik, Universität Hamburg, April 1999.

[Morris 1985] R. Morris *A Weakness in the 4.2BSD UNIX TCP/IP Software*. Computing Science Technical Report No 117, AT&T Bell Laboratories, 1985.

[Plummer 1982] David C. Plummer. *An Ethernet Address Resolution Protocol*. RFC 826.

[Stallings 1995] William Stallings. *Sicherheit im Datennetz*. Prentice Hall, München, 1995.

[Stevens 1994] W. Richard Stevens. *TCP/IP Illustrated, Volume 1: The Protocols*. Addison Wesley, Reading, Massachusetts, 1994.

Kapitel 3

Übersicht über Lösungsmöglichkeiten für die Sicherheitsproblematik

Peter Janitz
DFN-CERT GmbH

Zusammenfassung

In diesem Artikel wird eine Übersicht über verschiedene Möglichkeiten gegeben, die Sicherheitsprobleme in vernetzten Unix-Systemen zu bewältigen. Dabei wird die Sicherheitsproblematik anhand der formalen Aspekte von Sicherheit aufgeschlüsselt und es werden separate Lösungsansätze vorgestellt. Zunächst wird die Absicherung eines einzelnen Rechners beschrieben und anschließend auf die des gesamten Netzes eingegangen. Es wird sich zeigen, daß zur Erzielung einer möglichst hohen Sicherheit sowohl die Host- als auch die Netzwerksicherheit im Zusammenspiel mit einer Sicherheits-Policy zu beachten sind.

3.1 Einleitung

Mit fortschreitender Vernetzung von Computersystemen und der damit wachsenden Abhängigkeit von der ordentlichen Funktion dieser Systeme gewinnt die Frage nach der Sicherheit solcher Netze immer stärkere Bedeutung.

Die in diesem Artikel beschriebenen Beispiele beziehen sich größtenteils auf UNIX-Systeme, da die Vielzahl der Multiuser-Maschinen unter einem UNIX Betriebssystem arbeitet. Die allgemeineren, nicht auf UNIX bezogenen Beispiele und Hinweise, z.B. über die Auswahl „richtiger“ Passwörter, lassen sich auch auf andere Betriebssysteme anwenden, die über ähnliche Mechanismen verfügen.

Eine pragmatische, nicht unbedingt formale Definition von Rechnersicherheit lautet:

> Ein Rechner ist dann sicher, wenn er sich so verhält, wie es von einem autorisierten Benutzer erwartet wird.

Durch diesen Satz soll verdeutlicht werden, daß nicht nur Angriffe und mißbräuchliche Nutzung sicherheitsrelevant sind, sondern daß auch mangelhafte Soft- und Hardware Sicherheitsprobleme auslösen können.

Etwas formaler läßt sich Sicherheit in die folgenden vier Bereiche einteilen, auf die in den nachfolgenden Abschnitten genauer eingegangen wird:

- Physikalische Sicherheit
- Personelle Sicherheit
- Hostsicherheit
- Netzwerksicherheit

Da die physikalische und die personelle Sicherheit nicht das eigentliche Thema dieses Artikels sind, werden diese beiden Bereiche nur kurz angerissen. Das Kapitel Hostsicherheit soll zeigen, wie Benutzer oder Administratoren mögliche Schwachstellen eines Rechners beseitigen können und so bereits einen stärkeren Schutz vor mißbräuchlicher Nutzung gewinnen. Das nachfolgende Kapitel über Netzwerksicherheit soll die wichtigsten Maßnahmen zur Sicherung eines Netzwerkes aufzeigen. Damit all diese Schutzmaßnahmen koordiniert durchgeführt werden können, werden im letzten Kapitel Policies erklärt.

Eigenschaften von Sicherheit

Bevor versucht wird, die einzelnen Bereiche der Sicherheit zu analysieren und zu verbessern, sollte klargestellt werden, was Sicherheit bedeutet. Die wichtigsten Eigenschaften von Sicherheit sind:

- Verfügbarkeit
- Vertraulichkeit
- Integrität
- Authentizität

Hinzu kommt die Verläßlichkeit von Systemen und die gerade im Hinblick auf den elektronischen Handel wichtiger werdende Nichtabstreitbarkeit (non repudiation) von Handlungen.

3.2 Physikalische Sicherheit

Dieser Aspekt der Sicherheit, der fälschlicherweise oftmals vernachlässigt wird, umfaßt alle Maßnahmen, die dem physikalischen Schutz der Hard- und Software vor äußeren Einflüssen dienen.

Da es sich bei Computern und deren Bauteilen um sehr empfindliche Geräte handelt, ist dafür Sorge zu tragen, daß sie keinen schädlichen Umgebungseinflüssen ausgesetzt sind. Zu diesen schädlichen Einflüssen zählen Feuer, Rauch, Feuchtigkeit, Staub, Erschütterungen, usw. Durch spezielle Rechnerräume sollten die meisten dieser Faktoren ausgeschaltet werden können.

Da auch Spannungsschwankungen und -ausfälle Schäden nach sich ziehen können, sollten wichtige Systeme zum einen elektrisch getrennt abgesichert und eventuell mit einer eigenen unterbrechungsfreien Stromversorgung (USV) ausgerüstet werden.

Eine der wichtigsten Maßnahmen ist jedoch der physische Zugangsschutz. Hierbei gilt es nicht nur für einen abgeschlossenen Raum zu sorgen, sondern es sollten auch weitere Faktoren wie abgehängte Decken, große Klimaanlagen und Fensterfronten mit in die Überlegungen einbezogen werden. Durch diese Maßnahmen gelingt es, manuelle Sabotagen zu verhindern oder zumindest stark zu erschweren.

3.3 Personelle Sicherheit

Bereits bei der Einstellung neuer Arbeitnehmer sollte besondere Sorgfalt auf deren Auswahl gelegt werden. Dies beinhaltet neben persönlichen Gesprächen auch die Nutzung aller verfügbaren Informationen (Zeugnisse, Tests, ...). Weiterhin sollten gegebenenfalls Sondervereinbarungen mit den Angestellten betreffs Verschwiegenheit und Treue getroffen werden.

Ein sehr wichtiger Punkt besteht in der regelmäßigen Weiterbildung der Angestellten, da gerade im IT-Bereich die Entwicklung sehr schnell voranschreitet. Neben der Belehrung über die Pflichten sollte die Information über die Rechte der Mitarbeiter ebenfalls nicht vernachlässigt werden, damit es nicht zu der bekannten Situation kommt, in der ein Angestellter telefonisch dem vermeintlichen Administrator sein Passwort mitteilt. Diese Belehrungen haben das Ziel, ein Sicherheitsbewußtsein beim Benutzer selbst zu erreichen. Der Benutzer sollte verstehen, weshalb er manche Sachen machen darf und andere wiederum nicht. Ferner sollte er darüber informiert werden, welcher Personenkreis innerhalb der Firma für bestimmte Rechnerprobleme ansprechbar (und befugt) ist.

3.4 Hostsicherheit

Unter Hostsicherheit wird die Sicherheit des einzelnen Rechners verstanden, unabhängig davon, ob es sich um einen „stand-alone“ oder einen vernetzten Rechner handelt.

3.4.1 Benutzernamen und die Passwort-Datei

Unter Unix sollte jeder Benutzer einen Benutzernamen und ein Passwort besitzen. Dabei beginnt eine wirkungsvolle Hostsicherheit bereits mit der Verwendung «guter» Passwörter. Unter «guten» Passwörtern sind solche zu verstehen, die nicht allzu einfach erraten werden können. Sie sollten

- aus Groß- und Kleinbuchstaben bestehen,
- Zahlen und Sonderzeichen enthalten,
- so leicht zu merken sein, daß sie nicht notiert werden müssen und
- möglichst lang sein.

Als Beispiel sind hier «`Dwe-g-Pw`» (`Dies war ein gutes Passwort`) oder «`hmU8UsIn`» (`heute morgen Um 8 Uhr schlafe Ich noch`). Diese Regeln lassen sich erzwingen, indem der Systemadministrator

z.B. die „cracklib“-Routine [Crack,1996] in den Passwortmechanismus einbindet, wodurch nur noch Passwörter akzeptiert werden, die bestimmten Kriterien genügen.

Ein weiterer Schutz besteht darin, die Passwörter mit einem Verfallsdatum auszustatten und eine History über bereits verwendete zu führen, so daß die Benutzer in regelmäßigen Abständen zur Änderung aufgefordert werden und ihr Passwort auch tatsächlich ändern müssen. Da die Passwörter in dieser History vielleicht auf anderen Systemen noch benutzt werden und sich aus der History eventuell das neue Passwort eines Benutzers ableiten läßt, ist sicherzustellen, daß sie nicht mißbräuchlich genutzt werden kann.

Da auf die Passwortdatei von vielen Programmen zugegriffen wird, ist sie standardmäßig für jeden Benutzer lesbar. Da dies einen Brute-Force-Angriff, das (gezielte) Ausprobieren der verschiedenen Passworte, stark vereinfacht, ist es angeraten, die eigentlichen Passwörter nur mit Root-Rechten lesen zu können (shadow) und lediglich die sonstigen Benutzerinformationen allgemein lesbar zu belassen. Eine Veränderung der crypt()-Routine kann ebenfalls die Sicherheit der „Accounts“ erhöhen, da ein Brute-Force-Angriff meist mit Standardtools erfolgt. Hierzu zählt beispielsweise eine Erhöhung der Verschlüsselungsiterationen. Durch den Einbau eines Zählers, welcher Fehlversuche mitprotokolliert und Benutzerkonten gegebenenfalls temporär sperrt wird das mehrfache Einloggen zum Ausprobieren eine Passwortes stark erschwert. Der Einsatz eines verbesserten Zufallszahlengenerators, der die standardmäßig in den Betriebssystemen enthaltenen Generatoren ersetzt und keine erratbaren Ergebnisse liefert, trägt ebenfalls zu einer Erhöhung der Sicherheit bei.

Natürlich nutzen die sichersten Passwörter nicht viel, wenn der Loginvorgang beispielsweise durch einen Sniffer mitprotokolliert werden kann. Deshalb ist es angeraten, von außerhalb nur sichere (=verschlüsselte) Verbindungen zuzulassen, d.h. z.B. POP3 und FTP-Zugänge, bei denen Benutzername und Passwort unverschlüsselt übertragen werden, zu sperren sowie den telnet-Zugang nicht mehr zuzulassen und durch SSH [SSH, 1999] zu ersetzen.

Nach dem Motto „Vertrauen ist gut, Kontrolle ist besser“ sollte der Administrator regelmäßig versuchen, unsichere oder nicht mehr benutzte „Accounts“ selbst aufzufinden. Dies kann durch Passwort-Cracker und andere Hilfsmittel erfolgen. Es sollte allerdings darauf geachtet werden, daß etwaige Ausgaben und Logfiles nicht auf dem geprüften Rechner verbleiben, da sie einem potentiellen Angreifer ebenfalls sehr nützlich sein können.

Der sicherste – und auch komplizierteste – Mechanismus zum Schutz der „Accounts“ besteht aus der Verwendung von Einmalpasswörtern (z.B. [S/Key, 1993]). Diese haben den Vorteil, daß sie jeweils nur einmal gelten und somit Passwörter, die durch „Sniffer“ entdeckt werden, nutzlos sind.

3.4.2 Das Dateisystem

In die Überlegungen zur Hostsicherheit sollte auch das Dateisystem mit seinen bereits vorhandenen Schutzmechanismen einbezogen werden.

Bei den gängigen Unix-Dateisystemen wird zwischen dem Besitzer (`user`), der Gruppe (`group`) und anderen Benutzern (`other`) unterschieden. Für jede Kategorie lassen sich eigene Zugriffsrechte zum Lesen (`read`), Schreiben (`write`) und Ausführen (`execute`) von Dateien und Verzeichnisse setzen. So können vertrauliche Dateien oder Programme bereits durch `-rwx------` vor unbefugtem Zugriff anderer Nutzer geschützt werden. In Arbeitsgruppen können Dateien mit `-rwxrwx---` ausgestattet werden, wodurch gewährleistet ist, daß zusätzlich Benutzer der jeweiligen Gruppe auf die Dateien zugreifen können. Dies setzt jedoch eine sinnvolle Verwaltung der verschiedenen Gruppen voraus.

Diese Schutzmaßnahmen des Dateisystems gelten jedoch nicht für den Superuser. Wenn Daten auch vor dem Administrator oder einem Benutzer mit Root-Rechten geschützt werden sollen, ist der Einsatz von Verschlüsselungssoftware erforderlich, z.B. CFS (Cryptographic File System) (siehe [CFS, 1997] und Abschnitt 3.4.7).

3.4.3 Dienste

Da die meisten Angriffe gegen Rechner über Schwachstellen in den laufenden Netzwerk-Diensten (fingerd, telnetd, ftpd, httpd, ...) geführt werden, sind grundsätzlich auf jedem Rechner nur solche Dienste zu starten, die unbedingt zum ordnungsgemäßen Betrieb benötigt werden.

Ist es nicht möglich, auf einen bestimmten, bekanntermaßen unsicheren Dienst zu verzichten, so kann dieser eventuell durch einen sichereren ersetzt werden, der über die gleiche Funktionalität verfügt, aber zusätzliche Sicherheitsfunktionen beeinhaltet. Beispielsweise ist es sehr ratsam, die r-Dienste (rlogind, rshd, ...) durch das SSH-Paket zu ersetzen. Als weiteres Beispiel dient der „finger"-Daemon, welcher in seiner Standardkonfiguration sehr viele Informationen über die Benutzer eines Rechners liefert (Loginname, Zeit, Homeverzeichnis, ...), die einem Angreifer hilfreich sein können. Abhilfe kann hier z.B. der „restricted finger"-Daemon liefern, bei dem die Ausgaben frei konfigurierbar sind.

Darüberhinaus sollte der „TCP-Wrapper" ([TCP-Wrapper, 1999]) eingesetzt werden, der den Zugriff auf die Netzwerk-Dienste nur von bestimmten Rechnern aus gestattet.

3.4.4 SetUID- / SetGID-Root-Programme

Dateien, bei denen das SetUID- bzw. das SetGID-Bit gesetzt ist, werden mit den Rechten des Eigentümers bzw. der Gruppe ausgeführt. Dies gestaltet sich besonders problematisch, wenn bei diesen Programmen Root als Eigentümer ausgewiesen ist. Denn in diesem Fall kann der Ausführende, beispielsweise duch einen „Buffer Overflow" oder andere Fehler innerhalb des Programms, ebenfalls Root-Rechte erlangen. Ein Beispiel für solch ein Programm ist `passwd`:

```
-r-sr-xr-x   3 root     sys        12345 Apr 27  1995 /bin/passwd
```

Programme wie `passwd` oder `su` benötigen dieses Bit. Meist gibt es allerdings in einer Standardinstallation etliche SetUID-Programme, die entweder nicht benötigt werden oder die unnötigerweise mit diesen Rechten ausgestattet worden sind. Daher sollten nach der Installation von neuer Software oder Patches regelmäßig nach diesem Bit gesucht werden. Der folgende Aufruf erledigt dies, wobei es in Abhängigkeit vom jeweiligen Betriebssystem wiederum zu Unterschieden kommen dürfte:

```
find / \( -perm -004000 -o -perm -002000 \) -type f -print
```

Die SetUID / SetGID-Bits der gefundenen Programme können anschließend mit dem `chmod`-Befehl zurückgesetzt werden. Als Schutzmaßnahme vor unbekannten Dateisystemen können diese mit der Option „nosuid" in das Dateisystem einhängt werden, wodurch eine generelle Ignorierung der SetUID / SetGID-Bits erreicht wird. Sollte auf bekanntermaßen unsichere SetUID-Root-Programme nicht verzichtet werden können, so sind diese in einer „Chroot"-Umgebung auszuführen. Eine Chroot-Umgebung ist ein Verzeichnisbaum, der alle zur Ausführung benötigten Dateien enthält und von den dort laufenden Programmen nicht ohne größeren Aufwand verlassen werden kann. Auf diese Art und Weise erhält ein Angreifer nach einem „Buffer-Overflow" lediglich Zugriff auf eben diesen einzelnen Verzeichnisbaum – und nicht auf das komplette Dateisystem.

3.4.5 Logging / Auditing

Um zu gewährleisten, daß ein Angriff bzw. eine mißbräuchliche Nutzung eines Rechners bemerkt wird, ist es erforderlich, alle relevanten Daten und Vorgänge zu protokollieren – und auch auszuwerten. Dabei muß allerdings beachtet werden, daß Logfiles auf einem kompromittierten Rechner üblicherweise verändert oder gelöscht werden. Eine Änderung dieser Situation könnte sich z.B. mit Einführung eines neuen Syslog-Daemon ergeben, der in der Lage ist, Logfiles zu signieren (Bestrebungen dazu s. [Syslog-ng, 1999]).

Daher bietet es sich an, einen besonders gesicherten Loghost zu betreiben, der zentral alle Logmeldungen entgegen nimmt. Der immer noch bestehenden Gefahr des „Logfloodings“, dem drohenden Überlauf des Logmediums mit dem damit verbundenen Verlust an wichtigen Logmeldungen, kann z.B. durch entsprechende Filterung der Meldungen oder einem ausreichend dimensionierten Speicherplatz begegnet werden. Bei einer Filterung ist darauf zu achten, daß nicht gerade diejenige Information, die ein Angreifer in den vielen Logmeldungen verstecken will, verloren geht. Bei allen Logvorgängen darf nicht außer acht gelassen werden, daß teilweise gesetzlich festgelegt ist, welche Daten gespeichert werden dürfen, wie lange diese maximal aufzubewahren sind, usw. [BDSG, 1997].

Weiterhin empfiehlt sich der Einsatz von Prüfsoftware, die Prüfsummen der zu schützenden Dateien oder Verzeichnisse erstellt und somit eine Veränderung dieser Dateien erkennen läßt. Wichtig ist bei der Nutzung solcher Software vor allem, daß sie auf ein „sauberes“ System aufsetzt, von dem bekannt ist, daß noch keine Daten verändert wurden. Sobald Patches oder neue Softwarekomponenten eingespielt werden, ist ein Update der Prüfsummen erforderlich, um diese „legalen“ Veränderungen zu autorisieren. Weiterhin müssen die Prüfsummen auf einem Medium abgelegt sein, daß auch für Root nur lesbar ist, beispielsweise CD-Roms oder Festplatten mit Hardware-Schutz, da im Falle eines Einbruchs davon auszugehen ist, daß der Angreifer Root-Rechte erlangen kann und somit in der Lage ist, selbst neue Prüfsummen zu erstellen.

3.4.6 Der Superuser

Da der Superuser (Root) auf einem Unix-System unbegrenzte Rechte besitzt, gilt es, ihn besonders zu schützen. Neben den bereits beschriebenen Aspekten zur Passwortvergabe sind noch ein paar weitere Punkte zu beachten. So darf der Pfad zur Ausführung von Programmen (`$PATH`) des Superusers nicht das aktuelle Verzeichnis (`./`) enthalten, da sonst unter Umständen nicht der gewünschte Befehl, sondern ein von einem Angreifer installiertes, gleichnamiges Programm aufgerufen wird, welches sich im aktuellen Verzeichnis befindet. Kann trotzdem nicht auf das aktuelle Verzeichnis im Pfad verzichtet werden, dann sollte es nur am Ende des Suchpfades aufnommen werden, um zu gewährleisten, daß erst die Systemverzeichnisse durchsucht werden.

Um illegale Root-Logins bzw. deren Versuche rechtzeitig zu erkennen, sollte regelmäßig nach „bad logins“ in den Dateien `/var/log/messages/` und/oder `/var/log/sulog` gesucht werden. In Abhängigkeit vom Betriebssystem können sich diese Dateien auch unter `/var/adm/` oder `/usr/adm/` befinden. Um zu verhindern, daß bei der Kompromittierung eines Root-Passwortes auch alle anderen bekannt werden, ist es ratsam, jedem Rechner ein anderes Root-Passwort zu geben. Eine Maßnahme, die sich bei mehreren zu verwaltenden Rechnern allerdings schwierig und umständlich gestalten dürfte.

3.4.7 Verschlüsselung und digitale Signaturen

Die Zugriffskontrollen des Dateisystems sind aufgehoben, sobald ein Benutzer Administratorrechte (Root-Rechte) besitzt. Deshalb sollte nicht nur auf diesen eingebauten Dateischutz zur Wahrung der Vertraulichkeit und Integrität vertraut werden, sondern die sensiblen Daten sollten zusätzlich verschlüsselt werden.

Möglichkeiten

Die Verschlüsselung erfüllt verschiedene Eigenschaften von Sicherheit. Zunächst wird durch Verschlüsselung sowohl die Vertraulichkeit einzelner Daten auf einem Rechner als auch die Vertraulichkeit des Datentransports erreicht. Ferner können durch digitale Signaturen die Integrität und die Authentizität von Daten sichergestellt werden.

Mit dem Programm PGP (Pretty Good Privacy [PGPi, 1999]) erhält jeder Benutzer die Möglichkeit, seine Daten wirkungsvoll zu verschlüsseln. Da mit PGP ebenfalls Daten signiert und somit etwaige Veränderungen erkannt werden können, stellt PGP ein gutes Werkzeug dar, um Vertraulichkeit und Integrität von einzelnen Dateien sicherstellen zu können.

Durch CFS (Cryptographic File System) werden komplette Verzeichnisse verschlüsselt gespeichert, wodurch die Vertraulichkeit der Daten selbst bei einem Hardwarediebstahl gewahrt bleibt. Wird ein NFS-Server betrieben, so ist es möglich, verschlüsselte Dateisysteme zu exportieren und erst auf dem Klienten wieder zu entschlüsseln, um eine vertrauliche Übertragung zu gewährleisten.

Zu diesem Zweck und zur sicheren Authentisierung des Kommunkationspartners wird SSH [SSH, 1999] eingesetzt. SSH eignet sich besonders als Ersatz für telnet und die sog. r-Dienste, da einerseits keine Klartext-Passwörter mehr verschickt werden müssen und andererseits sowohl auf Server- als auch auf Klientenseite eine starke Authentisierung stattfindet. Dies stellt einen wirkungsvollen Mechanismus zur Verhinderung von Passwortsniffing, IP-Spoofing „„Man-in-the-middle“-Angriffen und TCP-Hijacking dar.

Grenzen

Verschlüsselung besitzt prinzipbedingte Grenzen. Eine verschlüsselte Datei ist zwar nur noch für den Schlüsselinhaber lesbar, kann aber, die entsprechenden Zugriffsrechte vorausgesetzt, unwiderruflich gelöscht oder modifiziert werden, so daß die Vertraulichkeit zwar gewahrt bleibt, die Verfügbarkeit jedoch nicht.

Eine weitere Gefahr besteht darin, daß eine wichtige Datei bereits vor der Verschlüsselung angegriffen werden kann.

Auch die Verschlüsselungssoftware selbst ist ein mögliches Angriffsziel. Ein solcher Angriff kann zur Folge haben, daß beispielsweise der Verschlüsselungsalgorithmus falsch funktioniert oder vielleicht die Passphrase, das Passwort oder der Klartext aufgezeichnet und weiterversandt wird.

Falls ein Angreifer einen bis dato unbekannten Weg der Entschlüsselung findet, ist davon auszugehen, daß die Vertraulichkeit und gegebenenfalls die Authentizität nicht mehr länger gewahrt werden kann.

Da bei verschlüsselten Emails zwar der eigentliche Inhalt, die Angaben im Header über Adressat und Absender jedoch frei lesbar sind, besteht die Möglichkeit von Verkehrsanalysen bzw. der Erstellung von Kommunikationsprofilen. Die Folgen können von der Zusendung unerwünschter Email bis hin zur Entlassung eines Mitarbeiters reichen, wenn dieser mit einer „feindlichen“ Firma Daten austauscht.

Heutzutage kommt es bei vielen Nachrichten auf die ordentliche und exakte Zustellung an. Gerade im Hinblick auf den elektronischen Handel können Replay oder Delay-Angriffe, d.h. das wiederholte oder das verzögerte Senden einer Nachricht, große Schäden anrichten. Diese Gefahren werden leider nicht durch bloße Verschlüsselung behoben, da auch eine verschlüsselte und signierte Nachricht aufgehalten oder mehrfach versendet werden kann. Um auch hier die Sicherheit wieder herzustellen, sind umfangreiche organisatorische und administrative Maßnahmen erforderlich, vor allem der Einsatz von Zeitstempeln. Diese Maßnahmen sind jedoch nicht Gegenstand dieses Beitrags.

3.5 Netzwerksicherheit

Es muß Benutzern und Administratoren verdeutlicht werden, daß der Anschluß eines Rechners an ein Netz grundsätzlich Gefahren in sich birgt. Dies gilt nicht nur für den Anschluß eines lokalen Netzes, sondern ebenfalls für die temporäre Verbindung eines einzelnen Rechners an ein Netz.

3.5.1 Grundsätzliche Überlegungen

Es sollten nur diejenigen Rechner vernetzt werden, die diese Anbindung unbedingt benötigen. Oftmals gibt es Alternativen zu einem Netzanschluß.

Die einfachste Methode ist der Datentranport mittels Wechseldatenträgern, eine Praxis, die in Hochsicherheitsumgebungen täglich angewandt wird. So betreibt die DFN-PCA den Datenaustausch mit dem „stand-alone“ Zertifizierungsrechner nur über Disketten, wie es in der Policy gefordert wird.

3.5.2 Grundregeln

Wenn ein Rechner vernetzt werden soll, haben bezüglich der laufenden Dienste die gleichen Punkte Beachtung zu finden, die bereits in Abschnitt 3.4.3 zu den Diensten hinsichtlich der Hostsicherheit genannt wurden. Das heißt, daß alle nicht benötigten Dienste abgeschaltet und alle unbedingt benötigten unsicheren Dienste durch sicherere Programme ersetzt werden müssen. Dies gilt im besonderen Maße für das Ersetzen der unsicheren r-Dienste durch die entsprechenden SSH-Versionen. Vor allem ist hervorzuheben, daß diese Sicherungsmaßnahmen *vor* dem Anschluß des Rechners an ein Netz zu erfolgen haben, da ein potentieller Angreifer unter Zuhilfenahme von Scripten in wenigen Sekunden einen Rechner automatisch auf Schwachstellen untersuchen und die gefundenen Lücken sofort ausnutzen kann, um den Rechner zu kompromittieren.

Jeder einzelne Rechner eines Netzwerkes läßt sich absichern. Der damit verbundene Sicherungs- und Administrationsaufwand dürfte bei kleinen Netzen noch praktikabel sein. Doch schon bei größeren Netzen mit mehreren unterschiedlichen Rechnertypen und Betriebssystemen wird dieser Aufwand nicht mehr zu bewältigen sein.

Daher bietet es sich an, an zentraler Stelle einen abgesicherten Rechner bzw. Rechner-Router-Kombination aufzustellen, welche die Sicherheit der nachgeschalteten internen Rechner gegenüber externen Angriffen gewährleistet.

3.5.3 Firewalls

Definition einer Firewall nach Ellermann [Ellermann, 1994]:

> Eine Firewall ist eine Schwelle zwischen Netzen mit verschiedenen Sicherheitsniveaus. Durch technische und administrative Maßnahmen wird dafür gesorgt, daß jede Kommunikation zwischen den Netzen über die Firewall geführt werden muß. Auf der Firewall sorgen Zugriffskontrolle und Audit dafür, daß nur berechtigte Verbindungen zwischen den Netzen aufgebaut werden und potentielle Angriffe schnellstmöglich erkannt werden.

Einer der wichtigsten Punkte dieser Definition ist, daß es nur genau eine Verbindung zwischen den Netzen gibt. Sollte es Abweichungen von dieser Regel geben, hier sind vor allem Modemverbindungen zu nennen, muß dies explizit in einer Policy (siehe Abschnitt 3.6) festgehalten werden. Als Ausnahme kann beispielsweise eine Modemverbindung zu einem Mailserver zugelassen werden, damit Mitarbeiter auch von auswärts Zugriff auf ihre Daten haben.

Meist wird davon ausgegangen, daß eine Firewall ein lokales Netz (LAN) vom Internet abtrennt. Aber gerade bei größeren Firmen wird das interne Netz oftmals weiter unterteilt und durch „interne“ Firewalls geschützt. Beispielsweise kann durch eine „interne“ Firewall das Rechnernetz der Personalabteilung vom restlichen Firmennetz geschützt werden, damit nicht jeder Angestellte Zugang zu vertraulichen Daten aus dem Personalwesen erlangen kann.

Es gibt verschiedene Möglichkeiten, eine Firewall zu realisieren. Sie reichen von ganz einfachen Konfigurationen bis hin zu komplizierten, sehr aufwendigen Implementationen.

Eine grobe Unterteilung kann in vier Klassen erfolgen:

- Packet Screens,
- Gateways,
- Kombinationen von Packet Screen und Bastion sowie
- Mischtechniken.

In den folgenden zwei Abschnitten werden nur die beiden ersten Konzepte genauer dargestellt. Aus [Kohlrausch, 1999] und [Ellermann, 1994] können weitergehende Informationen zu den verschiedenen Firewallkonzepten entnommen werden.

Beispiel: Packet Screen

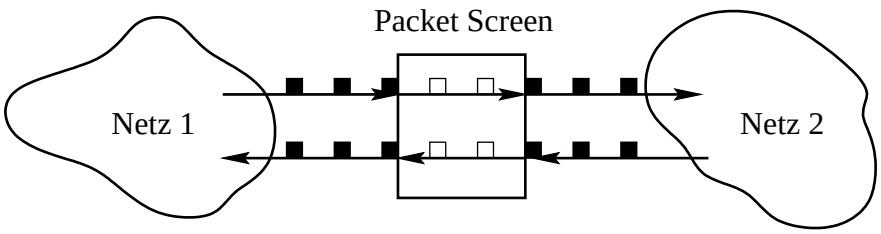


Abbildung 3.1: Zwei Netze durch eine Packet Screen verbunden

Packet Screens bilden die einfachste Möglichkeit, eine Firewall aufzubauen. Dafür kann ein bereits vorhandener Router so konfiguriert werden, daß er nur bestimmte Pakete durchläßt und andere hingegen abblockt (vgl. [Carl-Mitchell et al., 1992]). Daher werden Packet Screens auch oftmals als „Screening Router“ bezeichnet.

Mit einer Packet Screen ist es möglich, eine Zugriffssteuerung anhand der Quell- und Zieladressen sowie der Quell- und Zielports zu realisieren. Durch Filterregeln für Quell- und Zieladressen läßt sich die Erreichbarkeit einzelner Rechner oder Subnetze festlegen. Die Einschränkung der erreichbaren Dienste (http, ftp, telnet, . . .) erfolgt durch die Filterung der Quell- und Zielports. Bei den Filterregeln wird zwischen „Deny Filtern“ und „Pass Filtern“ unterschieden [Ranum, 1992].

Ein Vorteil von Packet Screens sind ihre geringen Kosten, da die notwendige Hardware, in Form eines Routers, meist bereits vorhanden ist und der Router lediglich umkonfiguriert werden muß.

Ein weiterer Vorteil ist die Transparenz einer Packet Screen. Solange keine unerlaubten Dinge versucht werden, wird ein Benutzer die Packet Screen nicht bemerken.

Eine Packet Screen besitzt allerdings auch einige Nachteile, von denen im folgenden einige genannt werden.

Die nur indirekte Kontrolle der übertragenen Protokolle ist ein solcher Nachteil. Da nicht nach Protokollen, sondern nach Portnummern gefiltert wird, ist es nicht gesichert, daß auf den verschiedenen Ports auch nur die gewünschten Protokolle übertragen werden. Durch „Tunneling“-Techniken können so auch Ports und Rechner angesprochen werden, die nicht erreichbar sein sollten [Bellovin 1989].

Desweiteren besteht die Gefahr, daß Rechnern mit einer gefälschten IP-Adresse ebenfalls Zugang gewährt wird.

Durch die Prüfung aller einzelnen Pakete kann es in Abhängigkeit der Vielzahl der Filterregeln zu Performanzeinbußen kommen.

Da ein Router keine Auditdaten liefern kann, müssen im Falle eines erfolgreichen Einbruchs die Auditdaten der einzelnen internen Hosts gesammelt und verglichen werden. Aufgrund dieses fehlenden Audits kann eine Packet Screen nicht als „richtige“ Firewall bezeichnet werden (vgl. Definition in 3.5.3).

Beispiel: Gateway mit Proxy

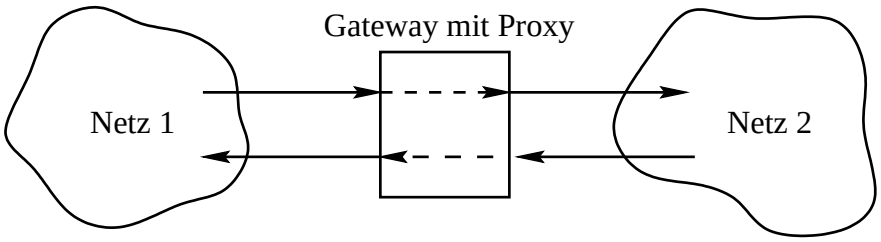


Abbildung 3.2: Zwei Netze durch ein Gateway mit Proxy-Server verbunden

Ein Rechner, der Verbindung zu zwei Netzen hat, wird als Gateway bezeichnet, wenn Verbindungen, die über diesen Rechner laufen, auf Applikationsebene realisiert werden. Bieten diese Applikationen weiterhin die Möglichkeit einer Zugriffskontrolle und eines Audits, so kann der Rechner als Gateway-Firewall genutzt werden.

Als einen Proxy bzw. Proxy-Server wird ein Dämon bezeichnet, der auf einem vorher definierten Port des Gateways auf eine Verbindung eines Klienten wartet. Der Proxy-Server prüft, ob der Verbindungswunsch des Klienten erlaubt ist. Ist dies der Fall, stellt der Proxy die Verbindung zum Zielrechner her.

Proxy-Server können für beliebige Protokolle eingesetzt werden. Am gebräuchlichsten sind Proxies für Telnet, FTP und HTTP.

Ein wichtiger Vorteil liegt in der großen Auswahl an Kriterien für die Zugangskontrolle. Neben den Quell- und Zieladressen sowie den Quell- und Zielports können z.B. auch Benutzername und Passwort oder Zeitangaben zusätzlich zur Überprüfung herangezogen werden.

Weiterhin ist es möglich, innerhalb eines Protokolls einzelne Befehle herauszufiltern, um beispielsweise zu verhindern, daß Daten über FTP exportiert werden. Ein anderes Beispiel wäre ein HTTP-Proxy, der nur bekannten, ungefährlichen Code zurückliefert und alles andere blockiert bzw. für eine spätere Auswertung speichert.

Ein Nachteil einer Gateway-Firewall sind wiederum die zu erwartenden Einbußen in der Performanz, da die einzelnen Pakete nicht nur geroutet, sondern bis zur Applikationsebene „hochgereicht“ werden müssen.

Grenzen

Durch eine Firewall erfolgt lediglich eine Zugriffskontrolle und ein entsprechendes Audit. Sie bietet jedoch keinen Schutz einzelner Verbindungen, z.B. vor Abhören. Deshalb ist es für einen äußeren Angreifer beispielsweise möglich, eine von der Firewall autorisierte telnet-Verbindung „zu übernehmen“ (hijacking) und somit in das interne Netz einzudringen.

Trotz einer Firewall kann ein potentieller Angreifer einem „geschützten“ Netz großen Schaden zufügen oder dieses ausspionieren, indem er einen autorisierten, internen Benutzer dazu veranlaßt, ein Trojanisches Pferd oder einen simplen Virus im internen Netz zu starten. Als Trojanische Pferde werden Programme bezeichnet, die neben ihrer eigentlichen noch eine weitere, versteckte Aufgabe erledigen. Dabei kann es sich um die Zerstörung von Daten oder das portionierte Herausschleusen von Informationen per Email handeln.

Ein ebenfalls nicht zu unterschätzendes Problem stellen interne Angriffe dar, d.h. von einem internen Rechner auf einen anderen im gleichen Netz. Da eine Firewall nur Verbindungen zwischen Netzen kontrolliert und keine Verbindungen interner Rechner untereinander, können Angriffe eines internen Rechners auf einen anderen im gleichen Netz nicht durch diese Firewall verhindert oder protokolliert werden.

3.6 Policy

Bisher wurden Techniken und Maßnahmen zur Erhöhung der Sicherheit genannt. Um diese wirkungsvoll einsetzen zu können, muß aber erst festgestellt werden, was schützenswertes Gut ist und wie die möglichen Bedrohungen aussehen.

3.6.1 Risikoanalyse

Vor der übereilten Durchführung von Sicherungsmaßnahmen, die unnötig hohe Kosten verursachen können und eventuell keinen Gewinn an Sicherheit liefern, steht eine genaue Kosten-Nutzen-Analyse (Risk Management). Dabei sind die folgenden Punkte zu beachten:

- Erfassung des zu schützenden Gutes

- Definition der benötigten Dienste, Netzverbindungen, Ressourcen, ...
- Erfassung aller möglichen Bedrohungen
- Aufstellung eines Schutzplans gegen die wahrscheinlichen Bedrohungen
- ...

Das Ziel dieser Analyse ist die Aufstellung einer geeigneten Sicherheits-Policy (weitere Informationen in [Großklaus, 1999]).

3.6.2 Grundregeln einer Policy

Eine Sicherheitspolicy ist für alle Mitarbeiter ohne Ausnahme unbedingt verbindlich. Wenn beispielsweise durch die Sicherheitspolicy nur Netzzugänge durch eine Firmenfirewall gestattet sind, so dürfen auch keine leitenden Angestellten ein privates Modem im Büro betreiben, um einen Netzzugang an der Firewall vorbei zu betreiben, da auf diese Weise gegebenenfalls unkontrollierte Hintertüren entstehen.

Desweiteren sollte eine Policy immer allgemein, verständlich und knapp formuliert sein, damit sie jeder Mitarbeiter im Gedächtnis halten kann, ohne erst langwierig nachschlagen zu müssen.

Im Gegensatz zur teilweise herrschenden Meinung ist eine Sicherheitspolicy öffentliche, allgemein zugängliche Information, die nicht geheimgehalten, sondern publik gemacht werden sollte.

Durch eine gut formulierte Policy läßt sich sogar zusätzliches Vertrauen schaffen.

In einer Sicherheitspolicy ist festgeschrieben, was zu schützen ist und was zum Schutz unternommen wird. Desweiteren wird eine Person benannt, die für die Umsetzung und Durchsetzung der Policy verantwortlich ist.

Da die beste Policy nur etwas nützt, wenn sie durchgesetzt und eingehalten wird, stellt die Überprüfbarkeit eine wichtige Anforderung dar. Dazu ist es erforderlich, entsprechende Kriterien festzulegen, mit deren Hilfe die Einhaltung der Policy geprüft werden kann.

Beispiele verschiedener Policies

Bisher war meist die Rede von einer Sicherheitspolicy. Es kann jedoch ein weitaus größerer Bereich an Richtlinien durch Policies abgedeckt werden. Beispielsweise gibt es in vielen Firmen, die in sicherheitsrelevanten Bereichen tätig sind, spezielle Verhaltensrichtlinien für die Mitarbeiter, die in einer Policy zusammengefaßt sind.

Ein weiteres Beispiel ist eine Backup-Policy. In ihr wird festgelegt, wann ein Backup zu erfolgen hat, was gesichert wird und wer hierfür zuständig ist.

Die DFN-PCA hat ihre Zertifizierungsrichtlinien in verschiedenen Policies zusammengefaßt. Es gibt die *World Wide Web Policy* (für SSL, S/MIME, ...), die *Medium-Level Policy* (für PEM) und die *Low-Level Policy* (für PEM und PGP). Unter [DFN-PCA Policies, 1999] sind diese Policies veröffentlicht.

3.7 Zusammenfassung

Trotz aller Bestrebungen zur Erhöhung der Netzwerksicherheit darf die Sicherheit der einzelnen Rechner (Hostsicherheit) weiterhin nicht vernachlässigt werden, damit ein zusätzlicher Schutzmechanismus existiert. Dieser kann notwendig werden, wenn z.B. die Firewall wegen eines Implementations- bzw. Konfigurationsfehlers oder im Zuge eines Angriffs ausfällt und dadurch Netzwerkverkehr ungehindert durchgelassen wird.

Sowohl kryptographische Verfahren als auch Firewalls besitzen ihre jeweiligen Vor- und Nachteile, die in den Abschnitten 3.4.7 und 3.5.3 beschrieben wurden. Daher bietet es sich an, eine Kombination beider Verfahren anzuwenden. Eine Firewall, die nur verschlüsselte Verbindungen erlaubt, ist eine solche Kombination, die die Vorteile beider Verfahren ausnutzt. Einerseits werden durch die Verschlüsselung die Vertraulichkeit, Integrität und Authentizität der Verbindungen sichergestellt und andererseits sorgt die Firewall für den Schutz aller nachfolgenden Rechner.

Eine vernünftige Sicherheitspolicy bildet die Grundlage für jedes Sicherheitskonzept. Nur so kann geregelt werden, welche Maßnahmen genau zu ergreifen sind. Das Fehlen einer Policy kann zu unkoordinierten Maßnahmen führen, die keine Verbesserung der Sicherheit mit sich bringen und unnötige Kosten verursachen.

Literaturverzeichnis

[BDSG, 1997] `http://www.datenschutz-berlin.de/recht/de/bdsg/bdsg1.htm`

[Bellovin, 1990] Bellovin S.: „Pseudo-Network Drivers and Virtual Networks“, Murray Hill, NJ, 1990

[Carl-Mitchell et al., 1992] Carl-Mitchell S., Quartenman J.,: „Building Internet Firewalls“, UNIX World, Feb. 1992

[CFS, 1997] `ftp://ftp.replay.com/replay/mirror/ftp.cryptography.org/disk/`

[Cheswick, 1994] Cheswick W., Bellovin S.: „Firewalls and Internet Security“, Addison-Wesley Publishing Company 1994

[Crack,1996] `ftp://ftp.cert.dfn.de/pub/tools/password/Crack/`

[DFN-PCA Policies, 1999] `http://www.pca.dfn.de/dfnpca/policy/`

[Ellermann, 1994] Ellermann U.: „Firewalls: Isolations- und Audittechniken zum Schutz von lokalen Computer-Netzen“, DFN-Bericht Nr. 76, DFN-Verein 1994

[Garfinkel, Spafford, 1996] Garfinkel S., Spafford G.: „Practical Unix & Internet Security“, 2nd Edition, O'Reilly & Associates, Inc. 1996

[Großklaus, 1999] Großklaus, A.: „Policy, Vorfallsbearbeitung, Schwachstellenanalyse“,Seminar 18.416, Sicherheit in vernetzten Systemen SS99, 1999

[Kohlrausch, 1999] Kohlrausch: „Klassische Firewalls in IP-Netzen“, Seminar 18.416, Sicherheit in vernetzten Systemen SS99, 1999

[PGPi, 1999] http://www.pgpi.com/

[Ranum, 1992] Ranum M.: „Thinking about Firewalls“, Greenbelt, MD, 1992

[S/Key, 1993] ftp://ftp.cert.dfn.de/pub/tools/password/SKey/

[SSH, 1999] http://www.ssh.fi/

[Syslog-ng, 1999] http://www.balabit.hu/products/syslog-ng/

[TCP-Wrapper, 1999] ftp://ftp.cert.dfn.de/pub/tools/net/TCP-Wrapper/

Kapitel 4

Policy, Vorfallsbearbeitung, Schwachstellenanalyse

Axel Großklaus
Universität Hamburg
Fachbereich Informatik

Zusammenfassung

Der folgende Artikel stellt die Bedeutung einer Policy im Rahmen eines Konzeptes zur Rechner- und Netzwerksicherheit dar und beschreibt Aufbau und Erstellung einer solchen. Nach der Definition des Begriffes werden Voraussetzungen der Erstellung und mögliche Ziele vorgestellt, sowie die Erstellung selbst Schritt für Schritt besprochen. Abschließend werden die Umsetzung einer Policy und mögliche Probleme dabei gezeigt und einzelne, ausgewählte Aspekte genauer besprochen.

4.1 Einleitung

Die Absicherung eines vernetzten Systems stellt, heute mehr denn je, eine komplexe Aufgabe dar. Nicht nur in Bereichen mit von Natur aus hohem Sicherheitsbedürfnis, wie zum Beispiel bei Banken oder dem Militär, sondern im alltäglichen Einsatz von vernetzten Systemen sind die Verfügbarkeit und Verlässlichkeit, und damit die Sicherheit dieser Systeme, ein wichtiger Faktor.

Vernetzte Systeme werden immer mehr in unternehmenskritischen Bereichen eingesetzt, deren Sicherheit unbedingt gewährleistet werden muß. Auch die zunehmende Einführung von „eCommerce“ und Online-Banking, also der Abwicklung von Geschäften mittels Computern über das Internet, stellt einen Bereich mit hohem Schutzbedürfnis dar. Hier werden durch einen Angriff auf die Sicherheit eines Systems nicht allein Daten des Betreibers gefährdet, sondern auch solche von Kunden und Geschäftspartnern (persönliche Daten, Kreditkartennummern usw.).

Mit den wachsenden Aufgaben ist die Komplexität der eingesetzten Systeme stark gestiegen, was ihre Absicherung weiter erschwert. Wie in [Gellert 99] gezeigt, sind die Angriffsmöglichkeiten auf ein System vielfältig, und die Anzahl dieser Angriffe hat, besonders mit der Vernetzung durch das

Internet, stark zugenommen. Neben der größeren Anzahl an Servern, die über das Internet erreichbar sind, ist ein Grund hierfür besonders die Verfügbarkeit von „Standard Software“ (sog. Cracker-Tools), um Angriffe auf entfernte Systeme automatisch und ohne großes technisches Wissen durchzuführen (siehe [Delgado Friedrichs 99]).

„You can have easy, cheap or secure. Pick two!“
[eine „Internet-Weisheit“]

Ebenso vielfältig wie die Zahl der möglichen Angriffe auf ein vernetztes System ist die Zahl der vorgeschlagenen Lösungsmöglichkeiten [Janitz 99]. Einen guten Überblick geben hier [Fraser et al. 97] und [Garfinkel & Spafford 1996].

Je nach Prioritäten und gewünschtem Grad von Sicherheit gibt es ganz unterschiedliche Ansätze und Vorgehensweisen, die sich oft ergänzen, manchmal aber auch behindern oder gegenseitig ausschließen.

Sicherheitsmaßnahmen verhindern leider nicht nur die unrechtmäßige Nutzung eines Systems, sondern sie behindern meistens auch die Arbeit der rechtmäßigen Benutzer. Auch die Kosten für die Absicherung eines Systems sind oft beträchtlich. Somit stellen Sicherheitsmaßnahmen immer auch einen Kompromiß von Sicherheit gegenüber Benutzbarkeit und Wirtschaftlichkeit dar.

Aus all diesen Gründen ist eine Policy, also ein Gesamtkonzept, das Ziele, Mittel und Verantwortliche der Bemühungen um Sicherheit definiert und gliedert, hier zwingend erforderlich. Nur wenn klar definiert ist, *was* von *wem wie* geschützt werden soll, kann die Fülle der möglichen Sicherheitsmaßnahmen effektiv angewendet und so der Aufgabe der Absicherung effektiv und effizient begegnet werden.

4.2 Definition und Aufbau einer Policy

4.2.1 Sicherheit im Sinne einer Policy

„A system is secure if it behaves the way you expect it will.“
[Garfinkel & Spafford 1996]

Die Sicherheit eines Systems ist *immer* ein relativer Begriff. Dies gilt für jeden einzelnen der Teilbereiche, die den Begriff der System-Sicherheit ausmachen (nach [Garfinkel & Spafford 1996]):

- Verfügbarkeit
- Vertraulichkeit
- Integrität
- Konsistenz
- Kontrolle
- Audit

Um nun die Sicherheit eines Systems zu bewerten, wird ein Ist-Zustand mit einem Soll-Zustand verglichen. Ein System ist dann sicher, wenn diese beiden Zustände übereinstimmen. Während sich der Ist-Zustand meistens unmittelbar aus dem betrachteten System ergibt, muß der Soll-Zustand vorab definiert und niedergeschrieben werden. Dabei wird jeder einzelne dieser Teilbereiche bewertet und festgelegt, wieweit und durch welche Maßnahmen die Sicherheit hier gewährleistet werden soll. Ausschlaggebend ist hier der Wert der zu schützenden Systeme und Daten (materiell wie immateriell) bzw. der Schaden, der bei einem Bruch der Sicherheit auftritt, sowie außerdem die technischen Möglichkeiten der Absicherung.

Jedes System und jede Gruppe von Nutzern des Systems hat individuell unterschiedliche Bedürfnisse, auf die die Maßnahmen jeweils zugeschnitten werden müssen.

4.2.2 Inhalt und Struktur einer Policy

„Policy, n.; pl. Policies
[...] 3. The method by which any institution is administered;
system of management; course.“
[Webster 1913]

Eine Policy ist eine Niederlegung von Zielen, Konzepten und Methoden in einer allgemeinen und nicht-technischen Weise. Sie ist das Gesamtkonzept für einen genau definierten Gültigkeitsbereich, zum Beispiel eine Firma. Obwohl die obige Definition aus einer Zeit vor Computern oder Netzwerken stammt, hat sich diese bis heute praktisch nicht geändert, und sie deckt sich weitgehend mit dem Begriff der Policy, wie er heute bei der Absicherung von vernetzten Systemen gebraucht wird.

Die Regelungen, die in einer Policy getroffen werden, stellen immer Kompromisse oder Abwägungen dar. Neben technischen Aspekten spielen hier besonders betriebswirtschaftliche und (firmen-)politische Überlegungen eine Rolle. Da absolute Sicherheit nie zu erreichen ist, muß für jeden Bereich entschieden werden, wie weit Bemühungen um Sicherheit gehen dürfen, damit sich deren Anwendung und Umsetzung noch lohnt und diese nicht zu einschränkend sind.

Beispielsweise könnte eine Policy zwischen einem filternden HTTP-Proxy auf der eine Seite und einem freien Zugang zum WWW auf der anderen Seite abwägen. Ist zum Beispiel die Bedrohung durch aktive Inhalte wie JavaScript so groß, daß die Benutzer keinen freien WWW-Zugang erhalten sollten?

Ein anderer Kompromiß könnte bei der Absicherung des Netzwerkverkehrs notwendig werden. Sollen Daten schon auf der Schicht 3 (z.B. IP) verschlüsselt werden, was den Nutzen von Packet-Screens erheblich einschränkt oder lieber allein auf eine Firwall vertraut werden? Hier behindern sich also zwei Sicherheitsmaßnahmen gegenseitig, und eine Entscheidung muß getroffen werden.

Dabei müssen die Regelungen jedoch immer umsetzbar bleiben. Es ist nicht sinnvoll, ein sehr hohes Sicherheitsmaß vorzuschreiben, das theoretisch zwar alles abdeckt, aber in der Praxis nicht umsetzbar ist.

Es muß auch festgeschrieben sein, wer dann für die Umsetzung der einzelnen Teilbereiche der Policy verantwortlich ist. Eine Vorschrift, *daß* etwas zu sein hat, ist wenig effektiv, wenn nicht gleichzeitig festgelegt ist, *wer* dafür Sorge trägt, daß dies auch passiert.

Eine Policy klärt also:

Was? (Objekte)	Auf welche Dinge wird Bezug genommen? Welche Ziele sollen erreicht werden?
Wer? (Subjekte)	Welche Mitarbeiter sind dafür verantwortlich?
Wie? (Vorgehen)	Wie soll dabei vorgegangen werden?

Die Regelungen, die dabei von der Policy getroffen werden, sind absolut. Sie dürfen von niemandem umgangen werden und es gibt keine Ausnahmen. Es können jedoch innerhalb der Policy selbst Ausnahmen von Regelungen festgelegt werden. Dies sind dann aber keine Ausnahmen *von* der Policy, sondern Ausnahmen *im Sinne* der Policy. Wenn eine Policy als zu einschränkend empfunden wird, kann sie geändert werden, dies sollte jedoch nur sehr selten vorkommen.

Insgesamt kann man bei den Formulierungen einer Policy zwei Grundrichtungen wählen:

Positiv: „Es ist alles erlaubt, was nicht explizit verboten ist.“

oder

Negativ: „Es ist alles verboten, was nicht explizit erlaubt ist.“

Beide Formen haben unterschiedliche Anwendungsgebiete. Die positive Form ist für eine Umgebung mit niedrigem bis mittlerem Sicherheitsniveau besser geeignet, da die Anzahl der notwendigen Regelungen kleiner gehalten wird, als bei einem hohen Niveau, bei dem viele Dinge durch die Policy ausgeschlossen werden sollen, und so wesentlich mehr Regelungen notwendig würden. Die positive Form ist jedoch möglicherweise zu liberal, da ein Fehler in der Policy wie das Vergessen einer Regelung zur Folge hätte, daß etwas erlaubt ist, was der eigentlichen Intention der Policy widerspricht. Ein Anwendungsgebiet für eine Policy mit positiven Formulierungen sind firmenweite, allgemeine Richtlinien.

Die negative Form der Policy hingegen findet ihre Anwendung eher in Bereichen mit restriktiven Regelungen und höherem Sicherheitsbedürfnis. Diese Art ist jedoch möglicherweise zu restriktiv. Ein Fehler in Form einer fehlenden oder falsch formulierten Regulierung führt hier dazu, daß etwas verboten ist, was eigentlich erlaubt sein sollte. Die negative Grundrichtung bietet sich besonders für restriktive Spezial-Policies wie zum Beispiel Sicherheits- oder Datenschutz Policies an. Zu verschiedenen Arten von Policies, siehe auch 4.2.3.

Diese beiden Arten stellen natürlich nur Grundformen dar, die in ihrer reinen Form in der Praxis wenig zu gebrauchen sind. Hier wird man meistens eine Mischform wählen, die jedoch einen klaren Schwerpunkt auf einer der beiden Arten hat.

Für eine effektive Umsetzung ist es unbedingt notwendig, daß das ganze Dokument öffentlich ist. Idealerweise zugänglich für wirklich jeden, mindestens aber öffentlich für alle, die von ihr in irgendeiner Weise betroffen sind, also alle Mitarbeiter, aber auch Kunden oder Lieferanten. Firmengeheimnisse oder Dinge, die die Sicherheit gefährden könnten, wenn sie bekannt würden, gehören nicht in eine Policy. Manche Firmen sind inzwischen dazu übergegangen, ihre Policies, sofern sie für Kunden von Interesse sein könnten, zum Beispiel im WWW zu veröffentlichen und mit ihnen als eine Art „Qualitätssiegel“ zu werben. Policies, die für eine solche Veröffentlichung in Frage kommen, sind zum Beispiel solche über firmeninternen Datenschutz.

Auf spezifische Systeme oder Mitarbeiter wird dabei kein Bezug genommen, sondern nur auf Arten von Systemen sowie Positionen und Funktionen von Mitarbeitern und Benutzern.

Eine mögliche Formulierung lautet daher *nicht*

„Herr Schulz ist für die S/Key-Paßwörter auf dem Server `foobar.fnord.com` *zuständig.“*,

sondern besser

„Für die Zugangskontrolle eines Servers ist der jeweilige Administrator verantwortlich.“

So ist sichergestellt, daß auch bei Änderungen von Systemen oder in der Personalstruktur keine Änderungen der Policy notwendig werden. Zusätzlich ist es im allgemeinen besser, Gebote anstatt von Verboten erlassen, soweit diese den gleichen Sinngehalt haben. Eine Policy, die aus einer langen Aufzählung von Dingen besteht, die ein Mitarbeiter alle nicht darf, wird sicher nicht so gut aufgenommen wie eine Policy mit eher positiven Formulierungen.

„Jeder Mitarbeiter hat für die Geheimhaltung seines Paß wortes Sorge zu tragen.“

ist besser als

„Mitarbeiter dürfen auf keinen Fall anderen Personen ihr Paßwort zugänglich machen“

Durch die allgemeinen Formulierungen in der Policy bleiben aber auch auf jeden Fall Lücken und Unklarheiten in den Anweisungen, die sie trifft. Daher gibt es zu jeder Policy weiterführende Dokumente wie zum Beispiel Richtlinien, die die Policy „mit Leben füllen“, also klar definieren, wie die Anweisungen der Policy umzusetzen sind. Auf diese Dokumente wird im Abschnitt 4.3 gesondert eingegangen.

Zusätzlich zu den Vorschriften, die mit einer Policy erlassen werden, sollten auch Sanktionen erarbeitet werden, die bei einem Verstoß gegen die Policy verhängt werden, sowie Vorgehensweisen, wie und wann dies passiert.

Um zu gewährleisten, daß auch immer die gültige Version der Policy verfügbar ist und Anwendung findet, muß das Dokument zusätzlich noch Informationen über einen Gültigkeitsbereich und -zeitraum, eine Version und eine eindeutige Dokumenten-Identifikation tragen, sowie einen Hinweis, wo die jeweils aktuelle Version der Policy erhältlich ist. Gedruckte Versionen sollten eine Unterschrift der Person tragen, die sie in Kraft gesetzt hat (Abteilungsleiter, Vorstandsmitglied), digitale Versionen eine kryptographische Signatur. Wenn die Unterschrift auf der Papierform nur gedruckt ist, sollte vermerkt sein, wo eine Version mit einer Originalunterschrift einsehbar ist.

4.2.3 Verschiedene Arten von Policies

Neben Unterschieden in Formulierung und Art der Regelungen, gibt es auch in Bezug auf den Gültigkeitsbereich verschiedene Klassen, in die sich Policies einteilen lassen.

Es gibt *allgemeine Policies*, von einer solchen wurde im vorigen Abschnitt ausgegangen, und *Spezial-Policies* für einen bestimmten Bereich. Welche man wählt, hängt hauptsächlich davon ab, ob man ein festes Gerüst für *alle* Vorgänge erstellen, oder nur einen einzelnen Bereich, wie zum Beispiel die Sicherheit von EDV-Anlagen, genauer regeln möchte.

Eine allgemeine Policy regelt *alle* Vorgänge innerhalb ihres Gültigkeitsbereiches. Ein solches Dokument kann jedoch, besonders bei großen Firmen, schnell unübersichtlich und unhandlich werden, sowohl für den Leser als auch für diejenigen, die die Policy betreuen und pflegen.

Daher kann es sinnvoll sein, eine Policy in verschiedene einzelne Dokumente aufzuspalten.

Beispiele für Policies und mögliche Regelungen:

Benutzer-Policy:	Welche Arten von Accounts gibt es? Welche Rechte und Pflichten haben diese jeweils?

Netzwerk-Policy:	Was ist „normaler“ Netzwerk-Verkehr? Wer darf welche Ressourcen nutzen? Was ist bei Ausfällen zu tun?
Backup-Policy:	Welche Daten werden wann gesichert? Wer sichert und verwaltet die Daten?
Audit-Policy:	Welche Audit-Daten werden gesammelt? Wie werden diese aufbereitet und gespeichert? Wer hat Zugriff auf diese Daten?

Bei dieser Aufteilung wird nun jeder Teilbereich durch eine eigene Policy geregelt, wobei alle diese Teil-Policies zusammen wieder ein Gesamtkonzept ergeben. Durch die kleineren Einheiten bleiben diese gut verständlich und sind einfacher zu handhaben und zu pflegen.

Die Erstellung einer Gesamtpolicy ist ein aufwendiger und komplexer Vorgang, der viel Zeit benötigt und damit Kosten erzeugt. Besonders kleinere Firmen werden hiervor zurückscheuen und meistens eine schlankere Lösung bevorzugen, die nur einzelne Bereiche abdeckt, die von besonderem Interesse sind. Dies kann duch eine Sicherheits-Policy geschehen. Dies ist eine Policy, die nur Punkte der Sicherheit der Daten und Dienste betrachtet, Dinge von nur allgemeiner Bedeutung jedoch außen vor läßt.

Als dritte Form von Policies gibt es Spezial-Policies, die nur einen einzelnen Bereich abdecken, der besondere Beachtung benötigt; zum Beispiel die Zertifizierungs-Policy einer Policy Certification Authority (PCA). Hier hat die Policy neben der alleinigen Regelung auch die Funktion, Dienste und Vorgänge nach außen zu dokumentieren.

4.2.4 Ziele einer Policy

Durch eine gemeinsame Policy wird innerhalb ihres Geltungbereiches eine gemeinsame Arbeits-Grundlage über Ziele, Vorgehensweisen und Begrifflichkeiten gelegt.

Erst durch eine Policy ist eine effektive Umsetzung von Sicherheit, sowie die Kontrolle deren Umsetzung möglich. Die Policy regelt und legt Ziele fest, gibt Anleitung, wie diese umzusetzen sind, und bietet einen Maßstab für die Qualität der Umsetzung. Damit kann auch eine Dokumentation von Leistungen und deren Qualität nach außen, also an Kunden und Vertragspartner, stattfinden.

4.3 Zusätzliche Dokumente: Standards und Richtlinien

Die Regelungen und Formulierungen innerhalb einer Policy sind zwar bindend für den jeweiligen Bereich, aber sie sind auch im Normalfall nur allgemein formuliert und legen lediglich ein Gerüst fest, nach dem vorgegangen werden soll. Damit diese Regelungen wirklich effizient angewendet werden können, müssen sie noch „mit Leben gefüllt“ werden, das heißt, es muß festgelegt werden, wie diese Regelungen zu verstehen sind und wie sie in der Praxis angewendet und durchgeführt werden sollen. Zu diesem Zweck werden der Policy weitere Dokumente beigestellt. Das Verhältnis ist hier ähnlich wie bei einem Gesetz und dessen Durchführungsverordnungen und Kommentartexten. Besonders bei ausführlichen Policies gibt es hier mehrere Ebenen an Dokumenten, die unterhalb der Policy angeordnet werden. Die Ausführlichkeit hängt jedoch auch hier wieder stark von der jeweiligen Situation ab.

Eine mögliche Hierarchie ist:

Policy: Die Policy ist absolut bindend, jedoch nur sehr kurz und allgemein gefaßt. Sie sollte jedem Mitarbeiter ausgehändigt werden und bekannt sein.

Standard: Ein Standard gestattet *seltene* Ausnahmefälle, ist aber ausführlicher und detaillierter. Er richtet sich an eine kleinere Personengruppe, wie zum Beispiel eine Abteilung oder eine bestimmte Art von Mitarbeitern.

Richtlinie bzw. Handbuch: Eine Richtlinie ist nur schwach bindend, aber sehr ausführlich und spezifisch. Sie richtet sich nur an die Person, die die beschriebene Aufgabe in der Praxis durchführt. Richtlinien für einzelne Bereiche können zu Handbüchern zusammengefaßt werden, zum Beispiel die Richtlinien zu Datensicherung und Speicherung von Daten zu einem Handbuch „Datenhaltung“.

Je weiter oben ein Dokument in dieser Hierarchie steht, desto bindender ist es und um so weniger Ausnahmen läßt es zu. Je weiter unten ein Dokument steht, desto ausführlicher und spezifischer werden die Anweisungen und Erklärungen und desto kleiner wird der Personenkreis, für den das Dokument relevant ist.

Da sich alle Dokumente nur auf Funktionen von Mitarbeitern und Einrichtungen beziehen, muß zu jeder Zeit klar ersichtlich sein, welche Mitarbeiter jeweils aktuell diese Funktionen ausüben. Legt z.B. ein Standard eine Aufgabe in den Bereich eines Operators, dann muß klar ersichtlich sein, welche Mitarbeiter als Operatoren tätig sind und wann diese Dienst haben. Auch diese Aufstellungen sollten jedem Mitabeiter zugänglich sein, für den ein jeweiliges Dokument gilt.

Eine Regelung zur Datensicherung könnte im obigen Modell folgendermaßen formuliert werden:

Policy: „Alle kritischen Daten müssen regelmäßig gesichert werden.“

Standard: „Backups sollen wöchentlich durch den Operator auf ein geeignetes Medium vorgenommen werden.“

Richtlinie: “Backups sollten an jedem Freitagabend mit dem Programm `tar` auf DAT-Band gespeichert und mindestens ein Jahr gelagert werden. Nach dem Schreiben sollte das Band durch Wiedereinlesen auf Korrektheit geprüft werden.“

Bei diesen Beispielen stellen die Formulierungen für Richtlinie und Standard jeweils natürlich nur einen kleine Auszug des Textes zur Umsetzung des jeweils darüberliegenden Dokumentes dar.

Während die Aufgabe einer Policy weitgehend darin besteht, Ge- und Verbote auszusprechen, haben der Standard und besonders die Richtlinien zusätzlich auch helfende und erklärende Aspekte. Sie legen nicht nur fest, welche Dinge getan werden sollen, sondern geben Hilfen wie dies am sinnvollsten und effizientesten erreicht werden kann und idealerweise auch, in begrenztem Rahmen, *warum* eine Regelung so gewählt wurde. Die Dokumente, die einer Policy beigefügt sind, sollen den Benutzern in erster Linie helfen, die Policy korrekt und ohne große Behinderungen umzusetzen bzw. anzuwenden.

4.4 Erstellung und Umsetzung einer Policy

4.4.1 Voraussetzungen und Vorbereitungen

Die Erarbeitung einer Policy ist ein komplexes und aufwendiges Projekt, das sorgfältig geplant und vorbereitet werden muß, um zu gelingen. Dies liegt zum einen daran, daß keine zwei Policies gleich sind und für jede Firma individuell ein Konzept erstellt werden muß, zum anderen daran, daß Nachbesserungen, aufgrund der Natur dieses Dokumentes, besonders ärgerlich sind und in der Umsetzung abermals große Kosten erzeugen können, wenn ein ganzer Betrieb auf die geänderte Policy umgestellt wird.

Die Erstellung einer Policy für eine Firma kann durch eigene Mitarbeiter geschehen oder als Auftrag an eine externe Beraterfirma gegeben werden. Eine Erstellung durch eigene Mitarbeiter hat den Vorteil, daß diese die Betriebsabläufe sehr wahrscheinlich besser kennen und einschätzen können. Eine Policy zur Sicherheit der EDV-Systeme und Netzwerke kann zum Beispiel durch Mitglieder der EDV-Abteilung erstellt werden. Es muß jedoch darauf geachtet werden, daß diese Mitarbeiter die notwendige Fachkenntnis bereits mitbringen, oder genügend Zeit haben, sich sorgfältig darin einzuarbeiten, da eine fehlerhafte Policy sehr wahrscheinlich schlimmere Folgen hat als gar keine Policy. Gegebenenfalls sollte dann zusätzlich eine Beraterfirma hinzugezogen werden. Diese Lösung hat u.a. den Vorteil, daß diese Berater (hoffentlich) sowohl die notwendige Fachkenntnis als auch Routine in der Erstellung mitbringen. Es enstehen jedoch hierdurch auch zusätzliche Kosten durch Honorare.

Ein guter Kompromiß ist es, eine Gruppe aus eigenen Mitarbeitern und externen Beratern für diese Aufgabe zusammenzustellen.

Die Erstellung der Policy läuft dann in mehreren Schritten ab:

- Autorisierung der Erstellung
- Informationsbeschaffung und Koordination
- Risiko- und Schwachstellenanalyse („Risk Analysis“)
- Erstellung der Policy selbst
- Erstellung von Plänen zur Vorfallsbearbeitung („Incident Handling“)
- Inkrafttreten der Policy und Umsetzung
- Kontrolle der Umsetzung und regelmäßige Prüfung der Regelungen

4.4.2 Erstellung: Informationsbeschaffung und Koordination

Bevor mit der Erstellung der Policy begonnen werden kann, muß, falls dies nicht schon geschehen ist, eine Autorisierung durch den höchsten Entscheidungsträger erfolgen, also zum Beispiel durch die Gesellschafter oder den Vorstand. Dies ist notwendig, damit die Regelungen, die die Policy trifft, auch rechtlich untermauert sind und umgesetzt werden. Eine solche Autorisation muß vor Beginn der Arbeiten erfolgen, um den Verfassern Einblick in alle nötigen Bereiche der Firma zu geben, und auch nach Fertigstellung, wenn die erarbeitete Policy in Kraft treten soll.

Der erste Punkt der Erstellung sollte eine Beratung mit allen Abteilungen sein, für die diese Policy gelten soll. Hier sind Informationen über den normalen Geschäftsbetrieb wichtig, genauso wie bestehende Probleme und Sicherheitslücken. Ein Bild des Ist-Zustandes der Firma muß erstellt werden.

Besonders wichtig sind hier auch bereits bestehende Dokumente zu (Sicherheits-)Konzepten und Arbeitsvorgängen. Diese Dokumente sollten möglichst in die Policy und ihre zugehörigen Dokumente eingearbeitet werden und danach nicht mehr als eigenständige Dokumente Anwendung finden. Wenn diese weiterbestehen sollen, muß auf jeden Fall darauf geachtet werden, daß diese Policy-konform sind und bleiben.

Personengruppen, die an der Erstellung der Policy beteiligt werden sollten, sind:

Mitarbeiter: Die einzelnen Mitarbeiter und Abteilungen können detailliert Auskunft geben über bestehende Arbeitsabläufe, Vorgehensweisen und gegebenenfalls auch über bestehende Probleme.

Management: Autorisierung und Umsetzung von Policy und Sanktionen (s.o.)

Betriebsrat & -datenschützer: Wahrung der (Persönlichkeits-)Rechte der Mitarbeiter und Information über neue Rechte und Pflichten sowie Koordination mit bestehenden Regelungen.

Sicherheitsdienst: Vereinheitlichung der Policy mit bestehenden Sicherheitskonzepten, Hinweise auf bestehende Sicherheitsprobleme sowie Klärung von Art und Machbarkeit der Umsetzung von neuen Regelungen.

Weitere Informationen sollten extern eingeholt werden:

Jurist: Es ist wichtig bzw. unumgänglich, daß die Policy mit dem jeweils geltenden Recht verträglich ist. Er kann Hilfen bei der Festsetzung von Sanktionen und deren Formulierungen geben sowie Ratschläge zu Vorgehensweisen bei Sicherheitsbrüchen (Beweissicherung usw.).

Datenschützer: Methoden zur Absicherung eines Systems (besonders im Bereich Audit) können schnell in Konflikt mit geltenden Datenschutz-Gesetzen führen. Wenn Daten über Aktivitäten von Benutzern erhoben und verarbeitet werden sollen, muß vorab geklärt werden, ob dies datenschutzrechtlich legitim ist. Der Datenschutzbeauftragte des jeweiligen Bundeslandes kann hier Auskunft
geben.

Versicherungen: Eine Beratung durch Versicherungen kann bei der Erstellung der Risiko-Analyse (siehe 4.4.3) helfen, da die Risiko-Analyse für eine Policy sehr ähnlich der einer Versicherung ist. Außerdem kann eine Versicherung bei der Schätzung bestehender Werte helfen und Auskunft geben über Wahrscheinlichkeiten von bestimmten Vorfällen (Stromausfall, Einbruch etc.).

Die Informationen aus den jeweiligen Quellen müssen zusammengetragen, strukturiert und aufeinander abgestimmt werden. Es ist hier besonders wichtig, bestehende Widersprüchlichkeiten aufzudecken und zu verhindern, daß durch die Policy neue Unverträglichkeiten dazukommen.

Wenn dies alles geschehen ist, kann mit der Schwachstellen- und Risiko-Analyse begonnen werden.

4.4.3 Risiko- und Schwachstellenanalyse

Bevor eine Policy (Schutz-)Maßnahmen definieren kann, muß klar sein, *was* geschützt werden soll, welche Gefahren bestehen, bzw. welche Ereignisse eintreten können und wie groß die Wahrscheinlichkeit des Eintretens und der entstehende Schaden jeweils sind.

Um diese Daten zu sammeln, ist eine Risiko- und Schwachstellen-Analyse notwendig, in der diese Fragen Schritt für Schritt geklärt werden.

Was soll geschützt werden?
Wie groß ist das Schutzbedürfnis im einzelnen?

Am Anfang steht eine Bestandsaufnahme an bestehenden Systemen, Daten und sonstigen schützenswerten Gütern, soweit dies nicht schon beim anfänglichen Zusammentragen von Informationen geschehen ist. Dabei sollten die Systeme und ihre Daten in Gruppen mit jeweils gleicher Sensitivität eingeteilt werden. Es werden „Sicherheits-Zonen“ erstellt. Abbildung 4.1 zeigt eine Einteilung eines Netzwerkes in ein internes Netz, ein externes Netz, das Internet sowie die verbindenden Komponenten. Für jede dieser einzelnen Zonen wird nun festgestellt, wie kritisch deren Sicherheit für den Gesamtbetrieb jeweils ist. Dabei ist auch von Bedeutung, wie sich ein Sicherheits-Bruch in einer Zone gegebenenfalls auf andere Zonen auswirkt.

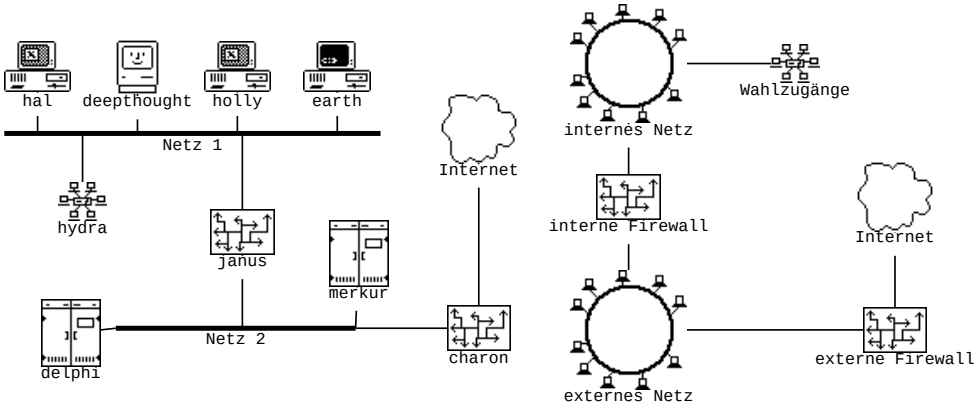


Abbildung 4.1: Einteilung der Systeme (links) in Zonen (rechts).

Welche Risiken bestehen jeweils?

Als nächster Schritt muß für jedes der Systeme bzw. der Zonen ermittelt werden, welche möglichen Brüche von Sicherheit bzw. sonstigen Risiken für deren Betrieb bestehen. Schäden durch böswillige Einwirkung (Sabotage, Spionage) sind hier genauso von Interesse wie solche durch Unfälle, Fehler in der Benutzung oder „höhere Gewalt“ (Stromausfall, Feuerschaden). Neben materiellen Schäden spielen auch immaterielle Schäden wie zum Beispiel Image- oder Vertrauensverlust eine Rolle.

Zu jedem der betrachteten Schäden sollte möglichst genau die Wahrscheinlichkeit des Auftretens in einem bestimmten Zeitraum und der tatsächliche Schaden, beziehungsweise die Kosten für eine Wiederherstellung des Ursprungszustandes ermittelt oder geschätzt werden.

Die Multiplikation von Wahrscheinlichkeit (zum Beispiel für ein Jahr) und Schaden ergibt einen Richtwert für die Kosten, die durch die jeweilige Bedrohung laufend entstehen.

Wenn zum Beispiel die Reparatur-Kosten bei einem Hardware-Ausfall 10.000 DM betragen und die Wahrscheinlichkeit für das Auftreten innerhalb eines Jahres bei 5% liegt, dann entstehen durch dieses Risiko im Jahr 10.000 DM $*$ 5% = 500 DM an Kosten.

Gegen welche von diesen Risiken können und sollen Maßnahmen getroffen werden?

Nachdem nur klar ist, welche Risiken jeweils im einzelnen zu beachten sind, wird geklärt, gegen welche von diesen Risiken eine Absicherung möglich ist, und aus diesen dann ausgewählt, gegen welche Risiken Absicherung getroffen werden soll.

Dabei wird man im Normalfall eine betriebswirtschaftliche Kosten-Nutzen Rechnung für jeden der einzelnen Punkte anstellen, Kosten für Schutzmaßnahmen gegen Kosten für Reparatur und Wiederherstellung gegeneinander abwiegen und sich für die günstigere Alternative entscheiden. Bei seltenen Vorfälle, gegen die eine Absicherung sehr teuer würde, wird man sich dafür entscheiden, lieber im Ernstfall die Kosten für eine Wiederherstellung zu bezahlen. Bei häufigeren Vorfällen, bei denen auch eine Absicherung nicht zu aufwendig ist, wird man sich für Vorsorgemaßnahmen entscheiden.

Hier jeweils ein Beispiel für die beiden Fälle (aus [Garfinkel & Spafford 1996, S.32]):

1. Beispiel: *Soll zur Benutzer-Authentisierung das herkömmliche Passwort-System durch ein besseres System mit Chip-Karten ersetzt werden?*

   Das zu sichernde System enthält Forschungs-Ergebnisse im Wert von ca. 1.000.000 DM. Zu diesem System haben 50 Benutzer Zugang. Bei jedem der Benutzer wird die Wahrscheinlichkeit, daß dieser sein Paßwort unabsichtlich preisgibt oder ausspioniert wird, auf 2% pro Jahr geschätzt. Die Wahrscheinlichkeit, daß das Paßwort eines Benutzers bekannt wird ist also $1-(1,0-0,02)^{50} \approx 63,6\%$, und somit beläuft sich der jährliche Schaden auf 1.000.000 DM $*$ 63,6% pro Jahr $\approx$ 636.000 DM/Jahr.

   Die Kosten für ein Chipkarten-System, das 5 Jahre eingesetzt werden kann, betragen 20.000 DM für das System, sowie 50 $*$ 75 DM für die Chipkarten, zusammen also 4750 DM/Jahr.

   Somit stehen 636.000 DM/Jahr Kosten durch den potentiellen Schaden nur 4750 DM/Jahr für die Absicherung entgegen und es wird klar, daß die Anschaffung des Chipkarten-Systems auf jeden Fall ratsam ist. Man entscheidet sich als *für* die Absicherung und *gegen* das Risiko.

2. Beispiel: *Soll für einen bestimmten Server eine unterbrechungsfreie Stromversorung angeschafft werden um die Verfügbarkeit zu sichern?*

   Der Schaden bei einem Stromausfall wird bei diesem System auf ca. 35.000 DM geschätzt. Darin sind Kosten für eine eventuelle Reparatur sowie die Kosten für den Arbeitsausfall der Mitarbeiter enthalten, die mit diesem System arbeiten. Nach Anfrage beim zuständigen Elektrizitäts-Versorger und einer Versicherung wird eine Ausfall-Wahrscheinlichkeit mit 0.5%/Jahr angesetzt. Der Schaden pro Jahr beträgt in etwa also 175 DM.

   Die Kosten für eine USV, die 10 Jahre betrieben werden kann, betragen rund 150.000 DM, also 15.000 DM/Jahr.

   In diesem Fall stehen 15.000 DM/Jahr für die Absicherung einem potentiellen Schaden von 175 DM/Jahr für das Risiko entgegen. Die Entscheidung wird also *gegen* die Absicherung und *„für“* das Risiko getroffen werden.

Natürlich sind die obigen Beispiele grob vereinfacht dargestellt und die Unterschiede in den Beträgen besonders deutlich gewählt. In der Praxis wird sich die Abwägung deutlich schwieriger gestalten. Das liegt zum einen daran, daß wesentlich mehr Faktoren in die Berechnung mit einfließen müssen (z.B. Lager- und Wartungskosten, immaterielle Schäden, Restrisiken), zum anderen ist das Zahlenmaterial, besonders die Wahrscheinlichkeiten, meistens wesentlich ungenauer und unsicherer. Das Prinzip, nach dem diese Abwägungen getroffen werden, ist jedoch das gleiche.

Welche Risiken werden eingegangen?
Welche Restrisiken bleiben?

Wenn die Entscheidungen für oder gegen einzelne mögliche Sicherheitsmaßnahmen getroffen sind, bleibt eine Anzahl an Risiken über, gegen die keine Absicherung getroffen wird. Entweder aus dem Grund, daß eine Lösung unpraktikabel oder teuer war, oder daß eine Absicherung überhaupt nicht möglich war, wie zum Beispiel bei schweren Naturkatastrophen.

Für (möglichst) alle diese Risiken sollten später Pläne erstellt werden, wie bei Eintreten des jeweiligen Vorfalls vorgegangen werden soll (siehe Abschnitt 4.4.4).

Wenn alle Risiken und möglichen Sicherheitsmaßnahmen eingeschätzt und entsprechend die Entscheidungen für oder gegen diese getroffen wurden, kann aus diesen Informationen die Policy erstellt werden, in der die getroffenen Entscheidungen dann zu festen Regeln werden und entsprechend ausgearbeitet sind.

4.4.4 Vorfallsbearbeitung

In Abschnitt 4.4.3 wurden Risiken gegen Sicherheitsmaßnahmen gegeneinander abgewägt und Entscheidungen darüber getroffen, ob eine Absicherung stattfinden soll. Dabei blieb eine Anzahl an Risiken, die in Kauf genommen werden, weil eine Absicherung zu teuer oder nicht praktikabel ist. Für jedes dieser Risiken muß nun ein Plan zur Vorfallsbearbeitung erarbeitet werden, damit man dem Vorfall trotz fehlender Absicherung nicht unvorbereitet gegenüber steht.

Auch hier muß wieder geklärt werden:

Was?	Was soll unternommen werden, um die Auswirkungen zu begrenzen und den Schaden zu beheben? Wie soll dabei vorgegangen werden und wie weit soll die Schadensbehebung gehen?
Wer?	Welche Mitarbeiter sind für die einzelnen Teile der Vorfallsbearbeitung zuständig? Welche externen Stellen sollen kontaktiert werden?

Durch die Erarbeitung von Plänen zur Vorfallsbearbeitung kann die Schadensbegrenzung effizienter durchgeführt und der Schaden somit reduziert werden. Auch werden Fehler vermieden, die durch unkoordinierte Bemühungen einzelner Personen entstehen können („operative Hektik“). Die Reaktion auf einen Vorfall ist im allgemeinen für alle Beteiligten ein Sonderfall, für den keine Routine und wenig bis gar kein Training besteht.

Wenn also zum Beispiel ein Cracker ein System kompromittiert hat, sollte klar sein, welche Schritte einzuleiten sind, um die Integrität des Systems wiederherzustellen, in welchem Maße Spurensicherung stattfinden soll und ob Anzeige bei der Polizei erstattet wird.

Besonders bei schweren Vorfällen wie Bränden, Personenschäden oder Ausfällen von weiten Teilen der Systeme z.B. durch Gewitterschaden stellt für alle Beteiligten eine starke psychische Belastung dar, unter der es schwer ist, aus dem Stand sinvoll und effizient zu handeln. In manchen Fällen kann es sogar unmöglich sein, auf einen Vorfall sinnvoll zu reagieren, wenn nicht vorher schon Vorbereitungen getroffen wurden. So sollte zum Beispiel eine Kopie der Pläne zur Vorfallsbearbeitung an mehreren Stellen gelagert werden, damit diese bei einem Brand auch zur Verfügung stehen und nicht verbrennen.

Idealerweise bieten die Pläne zur Vorfallsbearbeitung die „Vorbereitung auf das Unvorhersehbare“ und zu jedem möglichen Vorfall gibt es Handlungsanweisungen, wie vorzugehen ist. Solch ein „Zauberbuch“ wird es jedoch in der Praxis natürlich nicht geben. Trotzdem ist es sinnvoll, möglichst viele Möglichkeiten bei der Erstellung dieser Pläne in Betracht zu ziehen, denn jede Stunde die für die Erstellung eines solchen Dokumentes verwendet wird, zahlt sich im Ernstfall um ein Vielfaches aus.

4.4.5 Zusammenspiel

In den vorigen Abschnitten wurden drei Teile vorgestellt, aus denen ein vollständiges Sicherheitskonzept bestehen sollte, die Risikoanalyse, die Policy selbst und die Vorfallsbearbeitung.

Am Anfang steht eine *Risikoanalyse*, die Risiken und Schwachstellen aufdeckt und Gegenmaßnahmen abwägt.

Diese Menge von Risiken wird dadurch in zwei Teile aufgespalten. Der größte Teil sollte in der *Policy* durch entsprechende Maß nahmen verhindert oder verringert werden. Die restlichen Risiken sollten dann in den Plänen zur Vorfallsbearbeitung Beachtung finden.

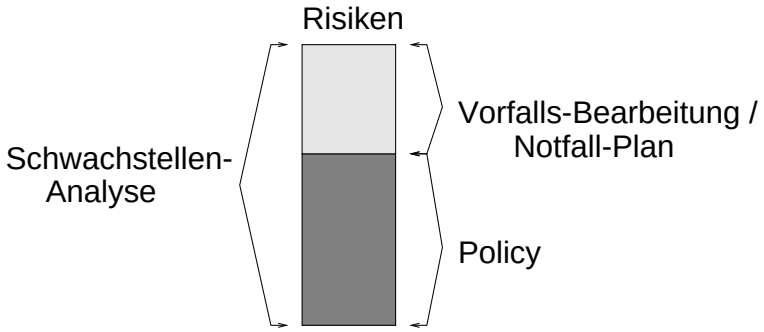


Abbildung 4.2: Das Zusammenspiel der drei Teile

Idealerweise gibt es also kein Risiko, das weder in der Policy noch bei der Vorfallsbearbeitung behandelt wird (siehe Abbildung 4.2). Natürlich ist auch dies wieder ein Idealbild, das in der Praxis kaum zu erreichen ist. Es sollte aber auf jeden Fall das angestrebte Ziel sein, dieses Bild so weit wie möglich zu verwirklichen.

4.4.6 Umsetzung der Policy und Kontrolle

Nach der erfolgreichen Erstellung der Policy und ihrer Zusätze (Handbücher, Standards usw.) kann diese „in Betrieb genommen“ werden. Sie wird also nochmals durch den zuständigen Entscheidungsträger abgesegnet und ist ab sofort gültig und bindend. Jedoch ist die Arbeit an der Policy nicht abge-

schlossen. Die Umsetzung einer Policy und die Kontrolle dieser Umsetzung sind ein ebenso wichtiger Teil wie die eigentliche Erstellung.

Am Anfang der Umsetzung steht die Unterrichtung aller Mitarbeiter. Die Policy und ihre einzelnen Regeln sollten nicht einfach nur eingesetzt werden, sondern alle „Betroffenen“ müssen ausführlich darüber informiert werden. Dabei ist es wichtig, daß die einzelnen Regelungen erklärt werden und auch die Gründe genannt werden, *warum* die jeweiligen Regelungen so getroffen wurden. Zusätzlich sind Informationen sinnvoll, wie die Regelungen am besten umzusetzen sind.

Bei der ersten Einsetzung der Policy sollte eine Schulung für alle Mitarbeiter durchgeführt werden, in der die Policy und das zugrundeliegende Sicherheitskonzept vermittelt werden. Diese Schulung muß, um effektiv zu sein, in regelmäßigen Abständen wiederholt werden. Auch sollten neue Mitarbeiter zu Beginn ihrer Tätigkeit über die Policy informiert werden. Zusätzlich zu den Schulungen sollte auch regelmäßig Feedback von den Mitarbeitern gesammelt werden, wie die Umsetzung in der Praxis funktioniert. So können Fehler und Probleme mit der Policy frühzeitig erkannt und behoben werden.

Es gibt eine ganze Reihe von Dingen, die die Umsetzung einer Policy behindern können oder diese ineffektiv machen:

- *Unverständnis bei Mitarbeitern oder Managern:*
  Besonders in den Anfängen ist es wahrscheinlich, daß Mitarbeiter oder Manager nicht mit den Regelungen einverstanden sind; besonders in Bereichen, in denen Sicherheitsmaßnahmen ihre Möglichkeiten oder Kompetenzen beschränken. Und allein der Widerstand, von einer Vorgehensweise in eine andere zu wechseln, kann beachtlich sein. Eingefahrene Strukturen in Arbeitsabläufen halten sich oft hartnäckig („das haben wir schon immer so gemacht“).

- *Mangel an Erfahrung bei der Umsetzung:*
  Für viele Mitarbeiter wird es neu sein, nach einer Policy zu arbeiten. Erfahrungen, wie Regelungen am besten umzusetzen sind, oder gar Routine fehlen oft oder sind nur unzureichend. Durch das Fehlen von Routine wird die Umsetzung der Policy für die einzelne Mitarbeiter schwieriger und aufwendiger. Es ist wichtig, daß sowohl die weiterführenden Dokumente wie Richtlinien ausführlich und verständlich sind, als auch daß Schulung ausreichend durchgeführt wird. Nur wenn Mitarbeiter ein Sicherheits*bewußtsein* haben, nach dem sie *intuitiv* handeln, kann die Policy effizient umgesetzt werden. Ein solches Bewußtsein hilft auch bei Bereichen, die nicht explizit durch die Policy geregelt sind, im Sinne der Policy zu handeln.

- *Mangelnde Durchsetzung von Sanktionen:*
  Ein weiterer wichtiger Punkt ist die tatsächliche Durchsetzung von Sanktionen, die in der Policy bei Verstößen definiert werden. Wenn dies nicht passiert, ist die Policy ein „Tiger ohne Zähne“. Fälle, in denen ein Mitarbeiter verschont wird, weil er in einem anderen Projekt gebraucht wird, untergraben die Autorität der Policy unmittelbar.

  In [Garfinkel & Spafford 1996, S.39] wird ein Fall beschrieben, in dem ein Programmierer versucht, in das System der Personalbuchhaltung einzudringen. Auf Drängen des Abteilungsleiters werden keine Sanktionen verhängt, da der Programmierer in einem Projekt benötigt wird. Drei Monate später gibt es einen Einbruch in das Buchhaltungssystem und der Systemverwalter, der mit dem System betraut war, wird entlassen. Obwohl es sehr wahrscheinlich ist, daß der Einbrecher derselbe Programmierer ist, bleibt dieser unbehelligt.

- *Mangelnde Methoden zur Kontrolle der Umsetzung und Datenschutz:*
  Nicht zuletzt ist ein großes Problem bei der Kontrolle der Umsetzung einer Policy, daß es wenig bewährte Methoden oder Maßstäbe gibt, an denen die Güte der Umsetzung wirklich festgemacht werden kann. Wenn man über eine „gut/mittel/schlecht“-Skala hinaus die Entwicklung der Umsetzung beobachten und bewerten will, tritt schnell das Problem auf, daß wenig konkrete Daten erhoben werden können. Zusätzlich ist diese Datenerhebung an den Punkten, wo sie technisch möglich ist, zum Beispiel bei Log-Dateien von Rechnern problematisch, da sie schnell mit dem Datenschutzrecht in Konflikt kommt.

Wie zu sehen ist, kann sich die Umsetzung einer Policy deutlich schwieriger gestalten als deren Erstellung. Auf jeden Fall ist die Erstellung kein einmaliger, statischer Vorgang, sondern sollte ein Kreislauf sein, der immer wieder durchlaufen wird.

Eine Rückkopplung von Informationen, die bei der Umsetzung gewonnen werden, und eine ständige Kontrolle der Regelungen ist notwendig, damit die Policy auch effektiv *bleibt*.

Hin und wieder kann es notwendig werden, Regelungen anzupassen oder neue Regelungen einzuführen. Mögliche Gründe, die neben Fehlern in der Policy zu Änderungen führen können, sind Änderungen in der Risiko-Struktur, zum Beispiel durch neue Tätigkeitsfelder der Firma, Änderungen in der Struktur der Firma, zum Beispiel durch neue Installationen oder Mitarbeiter, und nicht zuletzt Änderungen in der Gesetzeslage, die in einigen seltenen Fällen auftreten können. Manche Änderungen können durch einen sorgfältigen Entwurf der Policy von vornherein abgefangen werden, so daß evtl. nur Änderungen in Richtlinien und Standards erforderlich sind.

Im allgemeinen sollte man eine Policy so selten wie möglich, aber so oft wie nötig ändern und anpassen.

4.5 Geschichte und Entwicklung der Policy

Policies zur Sicherheit von EDV-Anlagen gibt es seit Beginn des Einsatzes von Computern. Dabei fanden diese zu Beginn hauptsächlich bei militärischem Einsatz von Computern Anwendung, da das Militär von Natur aus ein Interesse an einem hohen Sicherheitsniveau der eingesetzten Systeme hat. Viel Arbeit im Bereich von Forschung und Entwicklung wurde vom Militär geleistet. Besonders das „Department of Defense“ und das „National Computer Security Center“ der USA haben auf diesem Gebiet viel Arbeit investiert und viele Konzepte und Paradigmen im Bereich der Computer-Sicherheit entwickelt. Diese Konzepte wurden schnell in Regelungen gefaßt, die für alle amerikanischen Behörden und anderen staatlichen Einrichtungen gelten. Viele davon sind in den „Federal Information Processing Standards“ (FIPS) niedergelegt. Bekannt ist hier zum Beispiel die sogenannte „Rainbow Series“, deren bekanntester Vertreter das Orange Book ist.

Heute ist Computer-Sicherheit längst nicht mehr allein Sache des Militärs und einiger ausgewählter Firmen, sondern jede Firma, die EDV ernsthaft einsetzen will, muß sich Gedanken um deren Sicherheit und damit auch um eine Policy machen.

4.6 Zusammenfassung

Mit dem immer stärkeren Einsatz von Computern in nahezu allen Bereichen des Lebens und der zunehmenden Vernetzung dieser Systeme wird auch die Sicherheit und Verläßlichkeit dieser Systeme

immer wichtiger. Im Zuge dieser Entwicklung wird auch die Notwendigkeit einer Policy für viele Firmen immer dringender, da die EDV eine kritische Komponente geworden ist, deren Verfügbarkeit und Sicherheit unerläßlich ist. Eine Policy ist nicht eine isolierte Sicherheitsmaßnahme für sich, sondern *notwendige Grundlage*, um Sicherheit für die eingesetzten Systeme effektiv und gezielt umsetzen zu können. Ohne Policy werden Sicherheitsmaßnahmen immer eine unzusammenhängende Ansammlung von Einzelbemühungen bleiben. Eine Policy verbindet die Theorie der Sicherheitskonzepte mit der Praxis der wirklichen Maßnahmen.

Jedoch ist die Erstellung ein kompliziertes und aufwendiges Projekt, das nicht unterschätzt werden darf. Es sollte mit größter Sorgfalt geplant und durchgeführt werden. Eine schlechte oder fehlerhafte Policy ist unter Umständen ein größeres Risiko, als gar keine Policy zu haben.

Die Umsetzung einer Policy ist genauso wichtig wie die Erstellung. Nur in Verbindung mit Information und Schulung kann eine Policy in der Praxis auch wirklich durchgesetzt werden. Dabei sollte die Policy immer wieder überprüft und, falls notwendig, angepaßt werden.

Wenn all diese Punkte berücksichtigt werden, bietet eine Policy ein festes Grundgerüst, auf dem alle anderen Bemühungen um Sicherheit festen Halt finden und wirklich erreichen können, wofür sie gedacht waren.

Literaturverzeichnis

[Delgado Friedrichs 99] Delgado Friedrichs, F.: „Cracker-Tools am Beispiel“. Seminar 18.416: „Sicherheit in vernetzten Systemen“, Universität Hamburg, FB Informatik, SS 99

[Fraser et al. 97] Fraser, B. (Hrsg.) et al.: “RFC 2196: Site Security Handbook“, IETF, September 1997, `ftp://ftp.ietf.org/rfc/rfc2196.txt`

[Garfinkel & Spafford 1996] Garfinkel, S. & Spafford, G.: „Practical Unix & Internet Security“, 2nd Edition, O’Reilly & Associates 1996, 1991

[Gellert 99] Gellert, O.: „Überblick Sicherheitsprobleme“. Seminar 18.416: „Sicherheit in vernetzten Systemen“, Universität Hamburg, FB Informatik, SS 99

[Janitz 99] Janitz, P.: „Überblick über Lösungsmöglichkeiten für die Sicherheitsproblematik“. Seminar 18.416: „Sicherheit in vernetzten Systemen“, Universität Hamburg, FB Informatik, SS 99

[Webster 1913] Webster’s Revised Unabridged Dictionary, G & C. Merriam Co., 1913

Kapitel 5

Kryptographische Verfahren

Martin Johns
Universität Hamburg
Fachbereich Informatik

Zusammenfassung

Im folgendem Beitrag werden die wichtigsten kryptographischen Begriffe und Verfahren erläutert. Dabei wird der Fokus auf Verfahren liegen, die Kommunikationsbeziehungen zwischen zwei Kommunikationspartnern absichern.

5.1 Übersicht

In den vorhergehenden Beiträgen konnte man sehen, daß, in ihrem Rohzustand, vernetzte Systeme alles andere als sicher sind. Dies ist natürlich ein Zustand, der in Zeiten der globalen Vernetzung und zunehmenden Bedeutung des Internet im täglichen Leben nicht akzeptabel ist. Die Kryptographie bietet mächtige und vielseitige Hilfsmittel, die es ermöglichen können, die Sicherheit in vernetzten Systemen zu erhöhen. Im folgenden werden die wichtigsten kryptographischen Prinzipien und Algorithmen vorgestellt. Dabei liegt der Schwerpunkt auf den Chiffren (sowohl den symmetrischen wie auch den asymmetrischen), aber auch andere notwendige Aspekte (wie Einweg-Hashfunktionen oder digitale Signaturen) kommen zur Sprache. Leider kann hier nicht alles so ausführlich dargestellt werden, wie es eventuell nötig wäre. Wer mehr über die hier vorgestellten Dinge erfahren möchte, sei auf das Buch von Bruce Schneier [Schnei97] verwiesen, das auch als Quelle für diese Ausarbeitung diente.

5.2 Grundlegende Begriffe

5.2.1 Begriffsdefinitionen

Zum besseren Verständnis hier zuerst einige Definitionen und Abkürzungen von Begriffen, die im folgenden häufiger Verwendung finden werden:

Kryptologie: Wissenschaft der Kryptographie und Kryptoanalysis.

Kryptographie: Lehre vom geheimen Schreiben.

Kryptoanalysis: Lehre vom "Brechen" von Chiffren.

Chiffre (cipher) : Methode des Verschlüsselns.

Klartext (plaintext): unverschlüsselter Text (M für "Message", K für "Klartext" oder P für "Plaintext").

Chiffretext (ciphertext): verschlüsselter Text (C für "Chiffretext").

Chiffrieren (encrypt): verschlüsseln ($E_K(M) = C$).

Dechiffrieren (decrypt): entschlüsseln ($D_K(C) = M$).

Weiterhin wird später ab und zu von sogenannten unsicheren Kanälen zu lesen sein. Darunter sind Datenwege zu verstehen, bei denen nicht ausgeschlossen werden kann, daß dritte Personen Zugriff auf die auf diesen Datenwegen übermittelten Daten erhalten. Zur Erläuterung von einigen Vorgehensweisen wird in diesem Text außerdem mit den Namen Alice und Bob gearbeitet. Dabei handelt es sich natürlich nicht um real existierende Personen, sondern vielmehr um Statthalter für jede Art von handelnden Akteuren der digitalen Welt (also sowohl Menschen wie auch Maschinen).

5.2.2 Ziele

Der Einsatz von kryptographischen Methoden in der Praxis hat i.a. die Durchsetzung einer oder mehrerer Anforderungen zum Zuel:

Vertraulichkeit, Geheimhaltung: Hierunter ist zu verstehen, daß die betreffenden Dokumente bzw. die betreffende Kommunikation vor den neugierigen Blicken nicht autorisierter Personen zu schützen sind. Dieses wird normalerweise mittels Verschlüsselung der Dokumente erreicht.

Authentifizierung: Besonders auf elektronischen Kommunikationswegen ist es sehr einfach, sich falsche Identitäten zuzulegen. Durch das Fehlen von persönlichem Kontakt ist man darauf angewiesen, den digitalen Informationen zu vertrauen. Ein vergleichbares Problem ist auch das Feststellen der Identitäten von Kommunikationspartnern. Was benötigt wird, sind Methoden zur zweifelsfreien Identifikation von Akteuren in der digitalen Welt (wobei hier Akteure selbstverständlich nicht nur Menschen, sondern z.B. auch Rechner oder Prozesse sein können).

Integrität: Es muß möglich sein festzustellen, ob Daten von dritter Seite aus manipuliert wurden. Dieses bezieht sich sowohl auf Manipulation, die auf unsicheren Kommunikationswegen erfolgt sein mag, wie auch auf Datenmanipulation, die ein Eindringling eventuell im System durchgeführt hat (um z.B. Sytemsoftware durch eigene Programme zu ersetzen, die Hintertüren enthalten). In diesem Kontext ist unter Datenmanipulation Modifikation, Vernichtung und Ersetzung von Daten zu verstehen.

Unwiderrufbarkeit: Es sollte Methoden geben, die verhindern, daß ein Teilnehmer an einem Kommunikationsaustausch dessen Stattfinden im Nachhinein leugnen kann.

5.2.3 Der Begriff der Chiffre

Verschlüsselung von Daten erfolgt mittels eines Algorithmus, der Chiffre genannt wird. Dieser Algorithmus teilt sich auf in zwei Transformationen, eine zum Verschlüsseln der Daten (to encrypt, in Zeichen *E*) und einer zum Entschlüsseln (to decrypt, in Zeichen *D*). Sowohl die Ver- wie auch die Entschlüsselung wird unter Anwendung von Schlüsseln durchgeführt. Dabei können die Schlüssel zum Ver- und Entschlüsseln gleich (symmetrische Chiffren) oder unterschiedlich (asymmetrische Chiffren) sein. Um praktikabel zu sein, gibt es noch weitere Anforderungen an Chiffrieralgorithmen. Sie sollten einfach zu benutzen sein, d.h. es sollte kein Problem sein, passende Schlüssel zu generieren und anhand der Kenntnis des Schlüssels sollte es einfach sein, aus der Chiffriertransformation die Dechiffriertransformation zu berechnen.

Weiterhin sollte die Sicherheit des Algorithmus ausschließlich von der Geheimhaltung des Schlüssels abhängen und nicht auf der Geheimhaltung des Algorithmus beruhen (keine “security through obscurity”). Denn man kann davon ausgehen, daß jeder noch so gut geschützte Algorithmus eines Tages aufgedeckt wird. Und schließlich sollten die Chiffrier- und Dechiffrier- Transformationen für alle Schlüssel effizient berechnet werden können, denn niemandem ist mit einem Algorithmus geholfen, der zwar nahezu unbrechbar ist, aber unverhältnismäßig aufwendig in Zeit und Systemressourcen ist (man kann sich Szenarien vorstellen, wo solch ein Algorithmus akzeptabel wäre, aber im Mittelpunkt dieses Aufsatzes stehen kryptographische Methoden, die in erster Linie Anwendung in der Kommunikation in Datennetzen finden).

5.2.4 Die perfekte Geheimhaltung

Das Ziel einer jeden Chiffre ist eine möglichst gute Geheimhaltung. Deswegen lohnt es sich, als Referenzmarke den Begriff der *perfekten Geheimhaltung* einzuführen. Eine Chiffre unterliegt perfekter Geheimhaltung, wenn ungeachtet der Menge an vorliegendem Chiffretext aus dem Chiffretext nicht genügend Informationen abgeleitet werden können, um den Klartext zu bestimmen. Das bedeutet, daß die Kenntnis des Chiffretextes **keine** zusätzlichen Informationen über den Klartext gewährt und daß jeder mögliche Klartext, als Urbild des Chiffretextes, gleich wahrscheinlich ist. Die Voraussetzung für eine derartige Chiffre ist, daß die Anzahl der möglichen Schlüssel mindestens so groß sein muß wie die Anzahl der möglichen Klartexte. Nur so kann gewährleistet werden, daß für einen Chiffretext alle Klartexte gleich wahrscheinlich sind, denn es können höchstens so viele Klartexte Urbild eines Chiffretextes sein, wie es Schlüssel gibt. Also könnte eine Kryptanalytikerin bei einer Chiffre, deren Schlüsselraum kleiner ist als der Klartextraum, bei einem bekannten Chiffretext eventuell bestimmte Klartexte von vornherein ausschließen.

Es gibt tatsächlich einen Algorithmus, der perfekte Geheimhaltung bietet - das sogenannte *One-Time-Pad*. Beim One-Time-Pad wird der Klartext mit einem echt zufälligen Schlüssel der selben Größe exklusiv “verodert” (XOR). Der so entstandene Chiffretext ist ohne Kenntnis des Schlüssels absolut sicher, denn für jeden möglichen (sinnvollen oder auch sinnlosen) Klartext derselben Größe existiert ein Schlüssel, der den einen Text in den anderen überführt. Doch One-Time-Pads sind nicht praktikabel: Jeder Schlüssel darf nur ein einziges Mal verwendet werden. Sobald man einen One-Time-Pad-Schlüssel zweimal benutzt, gibt es sehr effektive Methoden zur Rückgewinnung des Klartextes aus dem Chiffretext. Außerdem sind die Schlüssel sehr groß (ebenso groß wie der zu verschlüsselnde Text) und müssen für einen sicheren Nachrichtenaustausch zuvor sicher an den Empfänger übermittelt werden. Und nicht zuletzt hängt die Sicherheit auch von der Güte des Schlüssels ab. Denn perfekte

Sicherheit kann es, auch mit diesem Verfahren, nur geben, wenn man über echte Zufallszahlen (oder einen hinreichend guten Zufallszahlengenerator) verfügt.

5.2.5 Klassische Chiffren

Chiffren sind schon seit langer Zeit in Gebrauch. Schon alte Feldherren wie Julius Cäsar haben erkannt, daß es sicherer ist, die Marschbefehle ihrer Truppen verschlüsselt zu übermitteln, für den Fall, daß der Nachrichtenbote in die Hände des Feindes geraten sollte. Klassische Chiffren kann man in zwei Kategorien aufteilen:

Transpositions-Chiffren: Eine Transpositions-Chiffre verändert die Reihenfolge der Zeichen des Klartextes. So wird z.B. aus BLUTWURST nach der Verschlüsselung TURBLUWTS. Die Art, wie die Zeichen ihre Position verändern, wird durch den Algorithmus bestimmt. So können die Zeichen zeilenweise in eine Matrix geschrieben werden und dann spaltenweise wieder herausgelesen werden. Auch andere Transpositions- Chiffren sind denkbar, ihnen allen gemein ist jedoch, daß sie modernen kryptanalytischen Methoden nicht standhalten können.

Substitutions-Chiffren: Bei Substitutions-Chiffren werden die Zeichen des Klartextes zeichenweise durch andere Zeichen ersetzt. Die einfachste Chiffre dieser Art ist die sogenannte Cäsar-Chiffre. Bei ihr wird zu den einzelnen Ordinalia der Zeichen ein fester Betrag addiert. Das resultierende Chiffretext-Zeichen ist die Summe modulo 26 (z.B. $A+4=E$ bzw. $Y+4=C$). Es gibt andere Substitutions-Chiffren, z.B. *homophone Substitution-Chiffren*, die Zeichen des Alphabetes auf verschiedene Chiffretext-Zeichen abbilden (So kann A auf 3, 12, 88 oder 92 abgebildet werden) oder *polygraphische Substitutions-Chiffren*, die Blöcke von Klartext mit Blöcken von Chiffretext ersetzt. Wie schon bei den Transpositions-Chiffren gilt auch hier, daß sie modernen kryptanalytischen Verfahren nicht standhalten.

Moderne Chiffren sind i.A. *Produkt-Chiffren*, d.h. eine Komposition von mehreren Funktionen, die jeweils entweder Substitutions- oder Transpositions-Operationen durchführen. Sie arbeiten üblicherweise auch nicht mehr auf Zeichen-, sondern auf Bit-Ebene.

5.3 Symmetrische Chiffren

5.3.1 Definition

Wenn man sich noch nicht eingehender mit Verschlüsselungsalgorithmen beschäftigt hat, sind es wahrscheinlich symmetrische Chiffren, die man sich vorstellt, wenn man an Verschlüsselung denkt. Hier paßt auch die Analogie des Schlüssels am besten. Wie bei der Tür, die vom gleichen Schlüssel zu- und auch wieder aufgeschlossen werden kann, wird auch der Text bei symmetrischen Chiffren mit dem gleichen Schlüssel ver- und wieder entschlüsselt.

Bei einem sicheren Nachrichtenaustausch zwischen verschiedenen Parteien muß dieser Schlüssel allen Beteiligten bekannt sein. Und hier liegt auch das Hauptproblem, auf das man bei der Benutzung von symmetrischen Chiffren stößt: der sichere Schlüsselaustausch. Bevor man sicher vor ungewollten Lauschern mittels verschlüsselter Botschaften kommunizieren kann, muß man den Schlüssel an alle Gesprächspartner verteilen, falls er ihnen noch nicht bekannt ist. Doch dieses ist kein triviales Problem, da ja offensichtlich kein sicherer Kanal zu den Gesprächspartnern vorhanden ist (sonst müßte

man den Nachrichtenaustausch schließlich nicht verschlüsseln). Mehr zu dieser Problematik findet sich in dem Aufsatz über Keymanagement.

In der Familie der symmetrischen Chiffren gibt es zwei Kategorien: Die *Stromchiffren*, welche die Daten bitweise verschlüsseln, und die *Blockchiffren*, die den Text in Blöcken fester Größe verschlüsseln.

5.3.2 Stromchiffren

Prinzip der Stromchiffren

Eine Stromchiffre verschlüsselt den Klartext i.a. bitweise (manchmal aber auch in größeren Einheiten). Dieses geschieht in den meisten Fällen durch eine bitweise exklusive Veroderung (XOR) mit einem Schlüsselstrom ($C_i = K_i \oplus S_i$, wobei C_i, K_i, S_i jeweils das i-te Bit des Chiffretextes, Klartextes und Schlüsselstroms sind). Die Entschlüsselung findet ebenfalls durch bitweises XOR mit dem gleichen Schlüsselstrom statt. ($C_i \oplus S_i = K_i \oplus S_i \oplus S_i = K_i \oplus 0 = K_i$). Der Schlüsselstrom wird durch einen Algorithmus (dem sogenannten Schlüsselstromgenerator) erzeugt, der als Eingabe zur Initialisierung einen Wert (den eigentlichen Schlüssel) bekommt. Der gleiche Schlüssel erzeugt immer den gleichen Schlüsselstrom, so daß den einzelnen Kommunikationspartnern lediglich der Schlüssel bekannt sein muß, um den Strom an Chiffretext-Bits zu entschlüsseln. Die Sicherheit einer Stromchiffre beruht auf der Güte des Schlüsselstromgenerators. Der Schlüsselstromgenerator produziert eine Folge von Bits, deren Auftreten möglichst zufällig ist. Je besser der Generator den Zufall approximiert, desto sicherer wird die Chiffrierung. Bei wahrer Zufälligkeit (nicht Pseudo-Zufälligkeit) hätte man ein One-Time-Pad. Bei der Verwendung von Stromchiffren sollte man darauf achten, niemals den gleichen Schlüssel (und somit den gleichen Chiffrestrom) zweimal zu verwenden. Falls man dieses tut, könnte eine dritte Person, die die beiden Chiffretexte abgefangen hat, diese beiden miteinander exklusiv verodern. Als Ergebnis erhält sie eine exklusive Veroderung der beiden Klartexte (die Schlüsselstrom-Bits fallen bei diesem Vorgehen weg). Dieses ist leicht zu entschlüsseln, da das Vorkommen der einzelnen Zeichen in Sprache alles andere als zufällig ist. Durch die Kenntnis eines der Klartexte kann man nun den Schlüsselstrom berechnen (durch XOR des Klar- und des Chiffretextes) und hat dadurch die Möglichkeit, alle weiteren Nachrichten, die mit diesem Schlüssel chiffriert wurden, problemlos zu lesen.

Ebenfalls sollte man darauf achten, daß, wenn man einen Schlüsselstromgenerator mit einem periodischen Schlüsselstrom verwendet, die Größe der zu verschlüsselnden Daten kleiner ist als die Größe der Periode. Falls dieses nicht erfüllt ist, gibt man eventuellen Lauschern einen Ansatzpunkt zum erfolgreichen Entschlüsseln des Textes: Es besteht so die Möglichkeit, den Chiffretext in Blöcke der Größe der Periode aufzuteilen und diese, ähnlich zu dem oben erwähnten Angriff, mit einander exklusiv zu verodern. Da sich der Schlüssel, nach dem Durchlaufen der Periode wiederholt, fallen auch bei diesem Vorgehen die Schlüsselbits weg (s.o.).

Durch die oben beschriebenen Eigenschaften (Strom von Chiffretext-Bits durch bitweise Verschlüsselung, keine Notwendigkeit, spezielle Blockgrößen der zu verschlüsselnden Daten einzuhalten) eignen sich Stromchiffren besonders für den Einsatz an Orten, wo es auf die Verschlüsselung kontinuierlicher Datenströme ankommt (z.B. Standleitung zwischen zwei Computern).

Arten von Stromchiffren

Man kann zwischen zwei unterschiedlichen Arten von Stromchiffren unterscheiden: den *synchronen* und den *selbstsynchronisierenden* Stromchiffren (vgl. Abbildung 5.1).

Bei den synchronen Stromchiffren ist der Schlüsselstrom unabhängig vom Datenstrom, d.h. der generierte Strom von Schlüsselbits wird allein bestimmt durch den verwendeten Algorithmus und den Schlüssel. Der Vorteil einer solchen Verfahrensweise ist, daß die Berechnung des Schlüsselstroms im vorhinein erledigt werden kann und beim eigentlichen Nachrichtenaustausch nur noch die XOR-Verknüpfung mit den Daten stattfinden muß. Besonders bei rechenintensiven Algorithmen kann man so den Datendurchsatz der Kommunikation erheblich erhöhen. Der Nachteil von synchronen Stromchiffren ist, daß bei nur einem verlorengegangenen Bit der gesamte folgende Chiffretext unbrauchbar ist. Dieses ist der Fall, weil sowohl auf Sender- wie auch auf Empfängerseite die Schlüsselstromgeneratoren die jeweiligen Daten-Bits mit dem selben Schlüsselstrom-Bit (de)kodieren müssen, sie müssen also synchron vorgehen (daher auch der Name). Falls ein Bit verloren geht, geraten die beiden in Asynchronismus. Während der Sender schon bei Bit Nummer *i+1* ist, ist der Empfänger noch bei Bit Nummer *i* und versucht somit eine Dechiffrierung anhand eines falschen Schlüsselstrom-Bits. Dieses setzt sich auf den Rest des Chiffretextes fort. Fehlerhaft übertragene Bits wiederum machen keine größeren Probleme. Die fehlerhaften Bits werden zwar falsch dechiffriert, aber der Rest der Nachricht bleibt entschlüsselbar.

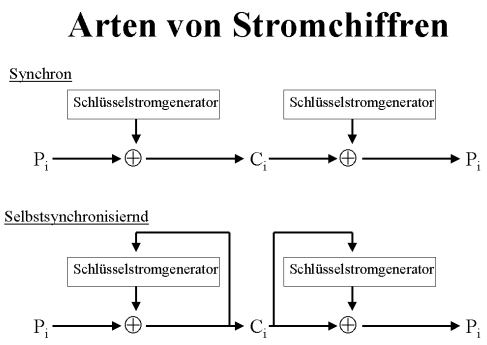


Abbildung 5.1: Synchrone und selbstsynchronisierende Stromchiffren

Bei selbstsynchronisierenden Stromchiffren ist jedes Schlüsselstrom-Bit eine Funktion einer festen Anzahl (*n*) vorhergehender Chiffretext-Bits. Dadurch bekommt die Chiffre die Eigenschaft, daß auch unter Verwendung des gleichen Schlüssels unterschiedliche Daten mit unterschiedlichen Schlüsselströmen kodiert werden. Somit fällt der Angriffsansatz aus 5.3.2 weg. Fehlerhaft übertragene oder fehlende Bits führen dazu, daß die nächsten *n* Bits falsch dechiffriert werden, da die Funktion zur Berechnung des Schlüsselstroms auf Sender- und Empfängerseite mit unterschiedlichen Chiffretext-Bits arbeitet. Nach *n* Bits haben sich die beiden aber wieder synchronisiert (deswegen selbstsynchronisierend) und arbeiten wieder korrekt (die Kommunikationspartner haben allerdings keine Möglichkeit zu entscheiden, ab welcher Stelle wieder synchron gearbeitet wird – dieses kann bestenfalls aus den empfangenen Daten geschlossen werden). Dieses eigentlich sehr wünschenswerte Verhalten ermöglicht aber auch einen Angriff auf die Kommunikation durch Wiedereinspielung. Eine dritte Partei, die aus einem früher stattgefundenen Nachrichtenaustausch Chiffretextbits gespeichert hat, kann diese später in einen Nachrichtenaustausch wieder einschleusen. Auf Empfängerseite wird dann für *n* Bits unbrauchbarer Datenmüll entschlüsselt, danach hat sich der Schlüsselstromgenerator aber mit den alten Daten synchronisiert und entschlüsselt nun einwandfrei die eingeschleusten Daten. Dies

funktioniert natürlich nur, wenn der Schlüssel in der Zwischenzeit nicht geändert wurde.

Ein Beispiel für eine häufig verwendete Stromchiffre ist RC4 (von Ron Rivest, 1987). RC4 ist eine synchrone Stromchiffre mit einer variablen Schlüssellänge und findet unter anderem Einsatz in Lotus Notes und Oracle Secure SQL.

5.3.3 Blockchiffren

Blockchiffren sind wie Stromchiffren symmetrische Chiffrieralgorithmen. Im Unterschied zu den Stromchiffren findet die Verschlüsselung aber nicht bitweise statt, sondern in Blöcken fester Größe (üblich sind im allgemeinen 64 Bit). Bei einer Blocklänge von 64 Bit werden also immer jeweils 64 Bit Klartext in 64 Bit Chiffretext transformiert. Bei einem Klartext, dessen Größe ungleich einem Vielfachen der Blockgröße ist, muß der letzte Block mit Bits aufgefüllt werden ("padding"). Dieses kann durch einfaches Auffüllen mit Nullen oder Einsen stattfinden, man kann sich aber auch raffiniertere Methoden des Auffüllens vorstellen. Somit hat der Chiffretext nicht unbedingt dieselbe Größe wie der Klartext, was auch ein weiterer Unterschied zu den Stromchiffren ist, wo Klartext- und Chiffretext-Größe in jedem Fall gleich sind. Unterschiedliche Größe von Chiffre- und Klartext ist eine wünschenswerte Eigenschaft, da in manchen Fällen die alleinige Kenntnis über die genaue Größe einer Nachricht Hinweise auf ihren Inhalt liefern kann.

Der Data Encryption Standard (DES)

Die wohl am weitesten verbreitete Blockchiffre ist der Data Encryption Standard (DES), entwickelt 1975 von IBM als eine Fortentwicklung von LUCIFER. Der DES verschlüsselt 64 Bit-Blöcke mit einem 56 Bit großen Schlüssel. 1977 wurde der DES vom "National Institute of Standards and Technology" (NIST) als Verschlüsselungsstandard mit dem Ziel spezifiziert, daß der DES in US-amerikanischen Software als Standard-Verschlüsselungsalgorithmus verwendet werden sollte. Die Software-Branche begegnete dem DES mit Mißtrauen. Dieses Mißtrauen rührte u.a. daher, daß die National Security Agency (NSA) eine nicht geklärte Rolle bei der Entwicklung des Algorithmus gespielt hatte und sich somit beständig Gerüchte gehalten haben, daß der DES eine von der NSA eingebaute Hintertür enthielte (eine Hintertür bei einem Verschlüsselungsalgorithmus ist eine Möglichkeit, ohne Kenntnis des Schlüssels chiffrierte Daten zu dechiffrieren). Der DES ist leicht in Hardware zu realisieren und wird auch heutzutage noch vielseitig verwendet (u.a. in Unix zur Verschlüsselung der Paßwörter).

Funktionsweise von DES

Die Verschlüsselung beim DES findet in 16 Runden statt. In einer Runde finden immer dieselben Operationen statt: Der 64 Bit große Datenblock wird in eine linke und eine rechte Hälfte zu je 32 Bit geteilt. Aus dem 56 Bit großen Schlüssel wird mittels Bitauswahl ein für die jeweilige Runde spezifischer Rundenschlüssel von 48 Bits erzeugt. Mit der rechten Hälfte der Daten und dem Schlüssel wird eine Funktion f berechnet, deren Ergebnis wieder ein 32 Bit großer Datenblock ist. Dieser Datenblock wird mit der linken Hälfte der zu verschlüsselnden Daten exklusiv verodert. Das Ergebnis dieser Veroderung stellt die neue linke Datenhälfte dar. Abschließend werden die linke und die rechte Hälfte vertauscht. Nach der letzten Runde findet keine Vertauschung der Hälften statt.

Als Formeln:

$L_i = R_i$

$R_i = L_i \oplus K_i$

Vor der ersten und nach der letzen Runde finden zusätzlich jeweils eine Permutation der Datenbits statt. Die Abschluß-Permutation ist die inverse Permutation zu der Eingangs-Permutation. Die Permutationen tragen nicht zur Sicherheit des Algorithmus bei und werden bei manchen Implementationen auch weggelassen. Die Entschlüsselung von mit DES chiffrierten Daten findet wie die Verschlüsselung statt, nur daß die Rundenschlüssel in umgekehrter Reihenfolge verwendet werden. Man kann sich leicht überzeugen, daß das Ergebnis der Dechiffrierung tatsächlich der Klartext ist.

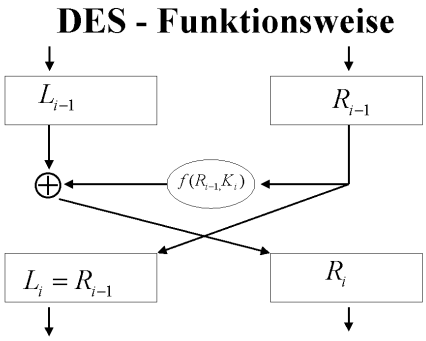


Abbildung 5.2: Funktionsweise des DES

Sicherheit von DES

Das Hauptproblem des DES ist die für heutige Rechenkapazitäten zu kleine Schlüssellänge. Der effektivste Angriff auf den DES ist ein sogenannter *Brute-Force-Angriff*, bei dem der gesamte Schlüsselraum durchsucht wird. Dies findet dann im schlimmsten Fall mit einem einzig für die Aufgabe des Findens von DES Schlüsseln gebauten, massiv parallelen Rechner statt, der nach und nach alle möglichen DES Schlüssel durchprobiert, bis er einen Klartext gefunden hat (an dieser Stelle sollte erwähnt werden, daß das Erkennen eines Klartextes kein triviales Problem darstellt). Bei einer Schlüsselgröße von 56 Bits gibt es 2^{56} mögliche Schlüssel für den DES, was heutzutage kein unüberwindbares Hindernis mehr darstellt, zumal statistisch nur 50% der Schlüssel probiert werden müssen. Eine Maschine zum Ermitteln eines DES-Schlüssels durch einen Brute-Force Angriff in durchschnittlich 3,5 Stunden hätte 1993 eine Million Dollar gekostet (z.B. der EFF-Cracker[1]).

Abhilfe gegen solche sehr speziellen Maschinen schafft das Benutzen von Varianten des DES. Diese Varianten (die z.B. mit einer leicht veränderten Funktion *f* arbeiten) sind zwar nicht unbedingt hundert-prozentig so sicher wie der DES, aber man kann davon ausgehen, daß potentielle Datendiebe nicht über Hardware-Implementationen dieser Varianten verfügen und somit auch über keinen Spezialrechner zur Schlüsselfindung.

Eine andere und bessere Möglichkeit, den DES sicherer zu machen, ist *Triple DES*. Bei Triple DES wird der Klartext insgesamt dreimal mit DES verschlüsselt. Dabei werden je nach Implementierung

[1] http://www.eff.org/pub/Privacy/Crypto_misc/DESCracker

entweder 2 oder 3 56-Bit DES Schlüssel verwendet. Genaugenommen wird der Text zuerst mit Schlüssel Nummer eins verschlüsselt, dann mit Schlüssel Nummer 2 entschlüsselt und abschließend entweder mit Schlüssel Nummer 3 oder nochmals Schlüssel Nummer 1 wieder verschlüsselt. Durch dieses Vorgehen verdoppelt bzw. verdreifacht man die Größe des Schlüsselraums. Man hat dann de facto einen 112 bzw. 168 Bit großen Schlüssel, also statt 2^{56} entweder 2^{112} oder 2^{168} mögliche Schlüssel, die es für einen Angreifer mit einem Brute Force Angriff auszuprobieren gilt.

Kurz erwähnt werden sollten hier auch die sogenannten “weak”- und “semi weak”-keys des DES. Dieses sind Schlüssel, die im Falle der “weak keys” den Text unverschlüsselt lassen und im Falle der “semi weak keys” zweifache Verschlüsselung mit dem selben Schlüssel wieder zum Klartext führt (was den Sicherheitsgewinn des Triple DES zunichte macht). Die Zahl dieser Schlüssel ist sehr gering und es wird davon ausgegangen, daß alle bekannt sind, so daß es für eine gewissenhafte Implementation des DES einfach sein sollte, diese Schlüssel zu vermeiden.

5.3.4 Betriebsmodi von Blockchiffren

Blockchiffren können in verschiedenen kryptographischen Modi betrieben werden. Die verschiedenen Modi ermöglichen eine der jeweiligen Situation angepaßte Nutzung der Chiffre, je nachdem ob es z.B. wichtiger ist, daß die Verschlüsselung schnell vonstatten geht oder ob der Gefahr des Entfernens oder Einschleusens von fremden Daten begegnet werden soll. Auch ist es möglich, einen Blockchiffre-Algorithmus so zu benutzen, daß er wie eine Stromchiffre arbeitet, z.B. in Fällen, in denen es wichtig ist, daß Klar- und Chiffretext dieselbe Größe haben. Kryptographische Modi fügen einer Chiffre keine weitere Sicherheit zu; die Sicherheit beruht weiterhin ausschließlich auf dem verwendeten Algorithmus. Im folgenden werden vier häufig verwendete Modi vorgestellt:

ECB (Electronic Code Book)

Der Electronic Code Book Modus (ECB) ist die wohl naheliegendste Art, eine Blockchiffre zu benutzen. Jeder Block des Klartextes wird separat in einen Block Chiffretext verschlüsselt. Gleiche Klartext-Blöcke liefern bei diesem Vorgehen gleiche Chiffretext-Blöcke. Hier liegt auch die Gefahr dieses Betriebsmodus’. Es ist möglich, unerkannt Chiffretext-Blöcke zu entfernen oder auszutauschen, bzw. weitere Blöcke einzufügen, weil die einzelnen Blöcke unabhängig voneinander sind. Um dieses zu tun, ist es nicht nötig, den Schlüssel zu kennen. Man fügt z.B. einfach zwischen zwei Chiffretext-Blöcken einen eventuell zuvor abgefangenen älteren Chiffretext- Block ein, dessen Inhalt man rekonstruieren konnte (indem man z.B. auf anderem Weg an den zugehörigen Klartext gekommen ist). Nach der Manipulation des Chiffretextes ist es nicht möglich diese festzustellen, da jeder Chiffretext-Block unabhängig von den anderen dechiffriert wird. Eine weitere Konsequenz dieses Modus’ ist es, daß Muster im Klartext nur schlecht verborgen werden. Ein nicht zu unterschätzender Vorteil des ECB ist es, daß die Chiffrierung massiv parallel durchgeführt werden kann. Besonders auf Medien mit hohen Übertragungsraten (z.B. ATM) kann man so die Übertragungskapazität des Mediums trotz Verschlüsselung der übertragenen Daten ansatzweise ausnutzen.

CBC (Chipher Block Chaining)

Der Cipher Block Chaining Modus (CBC, vgl. Abbildung 5.3) begegnet der Gefahr des ECBs, daß Chiffretext-Blöcke unerkannt ausgetauscht werden können. Beim CBC wird vor der Verschlüsselung eines Klartext-Blockes dieser mit dem vorhergehenden Chiffretext-Block exklusiv verodert. Bei

der Entschlüsselung wird der Chiffretext-Block zuerst dechiffriert und anschließend mit dem vorhergegangenen Chiffretext-Block exklusiv verodert. Durch dieses Vorgehen werden gleiche Klartext-Blöcke zu unterschiedlichen Chiffretext-Blöcken chiffriert, falls im Text zuvor mindestens ein Unterschied aufgetreten ist. Gleiche Nachrichten werden aber immer noch bis zum ersten Unterschied gleich verschlüsselt. Eine mögliche Abhilfe gegen dieses Verhalten ist es, als ersten Block einen zufälligen Initialblock zu verwenden. Beim CBC ist es nicht mehr möglich, unerkannt Chiffretext-Blöcke auszutauschen oder zu entfernen. Sobald eine Veränderung im Chiffretext stattfindet, wird der folgende Block fehlerhaft dechiffriert, weil zu einer einwandfreien Dechiffrierung der korrekte vorhergehende Chiffretext-Block nötig ist. Gleiches gilt für Entfernen oder Hinzufügen von Blöcken.

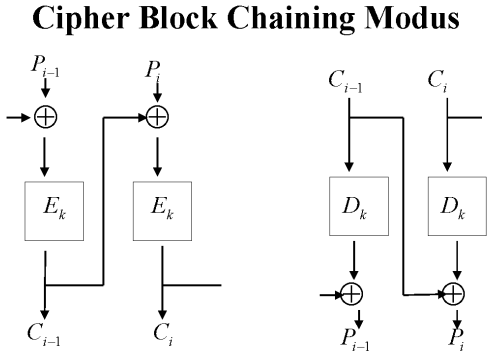


Abbildung 5.3: CBC

CFB (Cipher Feedback)

Beim Cipher Feedback Modus (CFB) wird die Blockchiffre als selbstsynchronisierende Stromchiffre verwandt. So ist es möglich, Daten in kleineren Einheiten als der Blockgröße zu chiffrieren. Die Größe dieser Einheiten ist variabel zwischen einem Bit und der normalen Größe eines Blockes. Die Verschlüsselung der Daten findet, falls die Chiffrierung bitweise vonstatten gehen soll, wie bei Stromchiffren üblich durch bitweises exklusives Oder (XOR) mit einem Strom von Schlüsselbits statt. Falls die Größe der Chiffierungseinheiten größer als ein Bit sein soll, wird in einem Schritt mehr als ein Schlüsselstrom-Bit generiert. Um den Schlüsselstrom zu generieren, wird der Chiffretext in ein Schieberegister zurückgeführt. Bei einer Blockgröße von *m* Bits werden die ersten *m* Bits des Schieberegisters mit dem zugrundeliegenden Blockchiffrierungs-Algorithmus verschlüsselt. Falls Dateneinheiten von *n* Bit chiffriert werden sollen, bilden nun die ersten *n* Bits des resultierenden Chiffretext-Blocks den Schlüsselstrom. Anschließend werden die ersten *n* Bits des Schieberegisters verworfen und die neu entstandenen *n* Bits Chiffretext nachgeschoben. Die Entschlüsselung funktioniert analog. Man kann sich vor Augen führen, daß so eine selbstsynchronisierende Stromchiffre, mit all ihren Vor- und Nachteilen (siehe 5.3.2), entsteht.

OFB (Output Feedback)

Es ist ebenso möglich, eine Blockchiffre als synchrone Stromchiffre einzusetzen. Dieses geschieht im Output Feedback Modus (OFB). Das Prinzip gleicht dem des CFB mit dem Unterschied, daß nicht der Chiffretext in das Schieberegister zurückgeführt wird, sondern der Schlüsselstrom. Dadurch wird der Schlüsselstrom unabhängig von den zu verschlüsselnden Daten. Dieses hat den Vorteil, daß der Schlüsselstrom schon vor der Kommunikation berechnet werden kann, so daß der eigentliche Nachrichtenaustausch sehr schnell vonstatten gehen kann, da der rechenaufwendige Teil der Chiffrierung schon erledigt ist und nur noch die Klartext-Bits mit dem Schlüsselstrom exklusiv verodert (XOR) werden müssen.

5.3.5 Weitere Blockchiffren

Neben DES existieren natürlich noch eine Menge weiterer hochentwickelter Blockchiffren. Zwei bekanntere sind:

IDEA (International Data Encryption Algorithm (1992)): Der IDEA-Algorithmus ist eine Weiterentwicklung des PES' (Proposed Encryption Standard). IDEA verschlüsselt den Klartext in Blöcken von 64 Bit und benutzt dabei einen 128 Bit großen Schlüssel. Die Verschlüsselung erfolgt in 8 Runden und ist insgesamt etwa doppelt so schnell wie der DES. IDEA findet unter anderem in PGP (Pretty Good Privacy) Verwendung.

Skipjack (1998): Der Skipjack-Algorithmus war der Vorschlag der NSA zur Nachfolge des DES. Einher mit der Verwendung des Algorithmus sollte ein Treuhändermechanismus gehen, der eine Kopie jedes benutzten Schlüssel bei der NSA hinterlegen sollte. Um dieses durchzusetzen, wurde die Funktionsweise des Algorithmus geheim gehalten und der Skipjack war nur als Clipper- oder Capstone-Chip in Hardware zu erhalten. Erst 1998 wurden seine Einzelheiten veröffentlicht und stellten sich bald darauf als nicht sicher heraus, weshalb er sich bislang nicht durchsetzen konnte. Beim Skipjack werden 80 Bit große Blöcke mit einem 80 Bit großem Schlüssel chiffriert.

5.4 Asymmetrische Chiffren

5.4.1 Definition

Asymmetrische Chiffren verfolgen einen anderen Ansatz als die oben erläuterten symmetrischen Chiffren. Anstatt daß der gleiche Schlüssel sowohl zur Ver- wie auch zur Entschlüsselung verwendet wird, gibt es bei asymmetrischen Chiffren zwei oder mehr verschiedene Schlüssel (in diesem Beitrag werde ich mich in meiner Beschreibung allerdings auf asymmetrische Chiffren mit zwei Schlüsseln beschränken). Im Fall von asymmetrischen Chiffren mit zwei Schlüsseln spricht man oft auch von *public key Algorithmen*, da üblicherweise ein Schlüssel veröffentlicht wird und ein Schlüssel geheim bleibt. Bei einer Kommunikation dient der öffentliche Schlüssel (*public key*) zum Chiffrieren der Daten und der geheime Schlüssel (*private key*) zum Dechiffrieren. So ist es möglich, daß jeder dem Besitzer des geheimen Schlüssels verschlüsselte Nachrichten zukommen lassen kann, ohne daß vorher ein gemeinsamer Schlüssel vereinbart werden mußte. Man verwendet einfach den öffentlichen Schlüssel der betreffenden Person, welcher auch auf unsicheren Kanälen übermittelt werden kann, und verschlüsselt

die zu sendenden Daten mit diesem. Der so entstandene Chiffretext kann einzig und allein mit dem geheimen Schlüssel dechiffriert werden, der nur dem Empfänger bekannt sein darf. Damit dies auch funktioniert, muß aber gewährleistet sein, daß durch Kenntnis des öffentlichen Schlüssels und des Algorithmus keine Möglichkeit entsteht, den geheimen Schlüssel zu ermitteln. Mit anderen Worten: Es muß unmöglich (bzw. unverhältnismäßig schwer) sein, unter Kenntnis des öffentlichen Schlüssels den geheimen Schlüssel zu berechnen.

Die Rollen (öffentlich/privat) der Schlüssel sind nicht bei allen Algorithmen von vornherein festgelegt, sondern häufig ist es egal, mit welchem Schlüssel ver- und mit welchem entschlüsselt wird.

5.4.2 RSA

Der wohl am weitesten verbreitetste public-key-Algorithmus ist RSA (benannt nach seinen Entwicklern Ron Rivest, Adi Shamir und Leonard Adleman (1977)). Anders als bei symmetrischen Chiffren wie dem DES, die auf geschickter Bit-Ersetzung und Permutation beruhen, ist das Funktionsprinzip des RSA ein rein zahlentheoretisches. Seine Schlüssellänge ist variabel, wobei Größen zwischen 512 und 1024 Bit üblich sind. Der RSA ermöglicht sowohl Verschlüsselung wie auch digitale Signatur (siehe 5.6.1) und hat sich weltweit zu einem Quasi-Standard für asymmetrische Verschlüsselung entwickelt.

Funktionsweise von RSA

Der Algorithmus des RSA ist verblüffend einfach:

Schlüsselgenerierung:

1. Wähle zwei etwa gleich große Primzahlen p,q.
2. Diese bestimmen den Modul $n = p * q$.
3. Wähle Zahl e mit $\text{ggt}(e, (p-1)(q-1)) = 1$.
4. Wähle Zahl d mit $e * d \bmod (p-1)(q-1) = 1$.
5. e und n werden öffentlich gemacht und bilden den *public key*.
6. d, p und q bleiben geheim (p und q werden üblicherweise aus Sicherheitsgründen nach der Schlüsselerzeugung gelöscht). d bildet den *private key*.

Ver- und Entschlüsselung funktionieren wie folgt:

Verschlüsselung: $C = M^e \bmod n$

Entschlüsselung: $M = C^d \bmod n$

Die Reihenfolge der Anwendung der Schlüssel ist beliebig, was aus der Kommutativität der Multiplikation folgt ($M = C^d = (M^e)^d = M^{ed} = M^{de} = (M^d)^e = C'^e = M$). Dies ist jedoch eine spezifische Eigenschaft des RSA.

Sicherheit von RSA

Die Sicherheit des RSA beruht darauf, daß das Problem der Faktorisierung von großen Zahlen ein hartes und somit sehr zeitaufwendiges ist. Ohne die Kenntnis der Primfaktoren p und q (und somit ohne Kenntnis von $(p-1)*(q-1)$) ist es nur sehr schwer möglich, aus dem öffentlichen Schlüssel *e* den geheimen Schlüssel *d* zu berechnen. Andere mögliche Arten die Chiffre anzugreifen, wie z.B. *d* mittels eines Brute Force Angriffs zu raten, sind noch aufwendiger (was bei einer Schlüsselgröße von über 512 Bit und einem zu durchsuchenden Raum von 2^{512} möglichen Schlüsseln keine Überraschung sein sollte). Deswegen ist die Gewinnung von *p* und *q* aus *n* die effektivste Methode, die RSA-Verschlüsselung zu brechen. Die Faktorisierung großer Zahlen kann durch eine mathematische Methode, genannt das Zahlenkörpersieb (*number field sieve, NFS*), beschleunigt werden und liegt dann irgendwo zwischen polynominellem und exponentiellem Aufwand. Es ist auch möglich, spezialisierte Hardware zum Faktorisieren großer Zahlen zu bauen: 1999 wurden die Kosten für einen Computer, der es ermöglichen soll, 512 Bit große Schlüssel zu finden, auf etwa 5000 US-Dollar geschätzt. Abhilfe für solche Bedrohungen ist es, größere Schlüssel zu verwenden. Denn mit Vergrößerung der Schlüssellänge (bzw. der Modul-Länge) wächst auch subexponentiell der Aufwand, den Modul zu faktorisieren. Heutzutage sollte man als User einen Schlüssel mit einer Länge von mindestens 1024 Bit und als Unternehmen von mindestens 1536 Bit benutzen.

Fazit: Die Sicherheit des RSA ist unbewiesen. Es ist bis heute lediglich keine Möglichkeit bekannt, eine RSA-Verschlüsselung effizient zu brechen. Ob eine derartige Möglichkeit existiert, ist unbekannt, aber nicht ausgeschlossen.

5.4.3 Andere Ansätze für asymmetrische Chiffren

Neben dem oben beschriebenen Ansatz für asymmetrische Chiffren, deren Sicherheit auf der Schwierigkeit des Faktorisierens großer Zahlen beruht, existieren weitere Ansätze für asymmetrische Chiffren, in denen andere, nur mit großem Aufwand zu lösende Probleme, als Grundlage benutzt wurden:

Berechnen des diskreten Logarithmus über endlichen Gruppen: Ähnlich wie das Problem des Faktorisierens ist das Berechnen des diskreten Logarithmus über endlichen Gruppen ein hartes Problem. Asymmetrische Chiffren, wie z.B. der ElGamal-Algorithmus, benutzen als Gruppe die ganzen Zahlen modulo einer Primzahl. Für diese Gruppe hat das Problem subexponentiellen Aufwand.

Lösen von NP-vollständigen Problemen: Es existieren mehrere Ansätze, NP-vollständige Probleme für asymmetrische Chiffren zu verwenden. Hierbei versucht man das Problem so zu modifizieren, daß es mit Hilfe einer zusätzlichen Information einfach zu lösen ist, ohne die Information der Aufwand aber weiterhin NP-Vollständig bleibt. Man spricht in solchen Fällen auch davon, daß man eine Hintertür in das Problem einbaut. Das modifizierte Problem bildet den öffentlichen Schlüssel, die Zusatzinformation den geheimen. Die zu verschlüsselnden Daten werden in eine Instanz (bzw. in eine Aufgabestellung) des NP-Vollständigen Problems transformiert. Um die Daten zurückzugewinnen, muß man das Problem lösen. Das Lösen des Problems ist ohne die Kenntnis der Zusatzinformation (also dem geheimen Schlüssel) ab einer gewissen Größe des Problems unmöglich, bei Kenntnis der Information aber im Idealfall mit linearem Zeitaufwand zu erledigen. Leider wurde bis jetzt noch kein überzeugender Algorithmus dieses Ansatzes gefunden.

Elliptische Kurven: Asymmetrische Chiffren, deren Sicherheit auf der Schwierigkeit des Lösens des diskreten Logarithmus über endlichen Gruppen beruht, benutzen als Gruppe i.a. die ganzen Zahlen modulo einer Primzahl. Diese Gruppe hat leider einige Besonderheiten, durch die das Berechnen des diskreten Logarithmus nur subexponentiellen Aufwand hat. Um ausreichende Sicherheit zu bieten, ist es nötig, relativ große Schlüssel zu verwenden. Dieses wiederum erschwert den Einsatz solcher Chiffren in Anwendungen, die nur über beschränkten Speicherplatz verfügen, wie es z.B. bei Chipkarten der Fall ist. Darüber hinaus wird ein Zusammenhang zwischen dem Problem der Faktorisierung und dem des diskreten Logarithmus vermutet. Falls sich dieses als wahr herausstellt und darüber hinaus ein Durchbruch im Lösen eines der beiden Probleme erzielt wird, werden mit einem Schlag fast alle verwendeten asymmetrischen Chiffren nutzlos.

Mit den elliptischen Kurven, deren Punkte eine "Abelsche Gruppe" bilden, glaubt man, einen potenten Ersatz für die Gruppe der ganzen Zahlen modulo einer Primzahl gefunden zu haben. Das so gewonnene Problem hat exponentiellen Aufwand und kommt somit mit kleineren Schlüsseln aus. Darüber hinaus ist das Problem des Lösens des Logarithmus über einer elliptischen Kurve unabhängig vom Problem des Faktorisierens großer Zahlen. Weiter kommt positiv hinzu, daß die bekannten und bereits als sicher bewerteten Algorithmen, die momentan noch die Gruppe der ganzen Zahlen modulo einer Primzahl verwenden, auch mit der Gruppe der Punkte einer elliptischen Kurve funktionieren. Der verbreiteten Benutzung dieses Ansatzes steht aber noch im Weg, daß die Schlüsselgenerierung für derartige Algorithmen ein noch nicht befriedigend gelöstes Problem ist.

Als Beispiel für einen Algorithmus, der erfolgreich mit elliptischen Kurven als Gruppe arbeitet, sei hier der *Elliptic Curve Digital Signature Algorithm (ECDSA)* (1992) genannt. Wie der Name schon vermuten läßt, handelt es sich hierbei um eine Variante des DSA (siehe 5.6.1). ECDSA wurde 1999 als ANSI-Standard (ANSI x9.62) akzeptiert.

5.5 Vergleich von symmetrischen und asymmetrischen Chiffren

In der Praxis stellt sich häufiger die Frage, ob für eine bestimmte Aufgabenstellung eher symmetrische oder asymmetrische Chiffren geeignet sind. Deswegen lohnt sich eine vergleichende Betrachtung der beiden Ansätze. Ein Vorteil von symmetrischen Chiffren ist die vergleichbar geringe Komplexität der Berechnung und die damit einhergehende höhere Geschwindigkeit. Hinzu kommt, daß die Schlüssellänge bei symmetrischen Chiffren weit unter der von asymmetrischen Chiffren liegt (56 Bit gegenüber mind. 512 Bit). Die hohe Komplexität von asymmetrischen Chiffren macht ihren Einsatz bei großen Datenmengen unpraktikabel.

Auf der anderen Seite liegen die Probleme bei symmetrischen Chiffren klar beim Schlüssel-austausch und der damit verbundenen Schlüsselverwaltung. Bei *n* Kommunikationspartnern müssen mind. *n-1* Schlüssel auf sicheren Kanälen ausgetauscht werden und diese im Anschluß sicher aufbewahrt werden; beides keine trivialen Probleme. Diese Probleme fallen bei der Verwendung von asymmetrischen Chiffren weg. Der einzige Schlüssel, der sicher aufbewahrt werden muß, ist der eigene geheime Schlüssel. Die anderen öffentlichen Schlüssel der Kommunikationspartner können an unsicheren Stellen gelagert werden, und wenn sie noch nicht vorhanden sind, einfach vor Beginn der sicheren Kommunikation auf ungeschützten Kanälen angefordert werden.

Aber genau hier zeigt sich eine nicht zu unterschätzende Gefahr, die bei der Benutzung von asymmetrischen Chiffren besteht: Die Authentizität der öffentlichen Schlüssel der Kommunikationspartner ist

nicht in jedem Fall gewährleistet. Man kann nicht ohne weiteres davon ausgehen, daß der Schlüssel, der als öffentlicher Schlüssel von Alice gesendet wurde, auch tatsächlich zu Alice gehört. Genausogut kann sich eine dritte Person als Alice ausgeben und ihren öffentlichen Schlüssel anstelle des Schlüssels von Alice verschicken und so die im guten Glauben für Alice chiffrierten Nachrichten lesen und diese anschließend mit dem echten öffentlichen Schlüssel von Alice erneut verschlüsseln und weitersenden. Es müssen in solchen Fällen also Mechanismen zum Einsatz kommen, welche die Authentizität der Schlüssel gewährleisten. Diese Verifikation der Identität eines Schlüsselinhabers ist nicht durch die asymmetrischen Algorithmen geregelt und muß unabhängig von diesen erfolgen.

Wie man sehen kann, sind die Nachteile der einen Chiffreart genau die Vorteile der anderen (und umgekehrt). Somit fällt es schwer, eine allgemeine Empfehlung für eine der beiden Arten zu geben. In den meisten Fällen wird es vielmehr zu einem kombinierten Einsatz der beiden Arten kommen, z.B. eine asymmetrische Chiffre zum anfänglichen Austausch von (Sitzungs–) Schlüsseln und eine symmetrische Chiffre für den eigentlichen Datenaustausch. Somit können die jeweiligen Nachteile vermieden und die Vorteile der Chiffrearten genutzt werden.

Der Vollständigkeit halber noch ein kurzer Blick auf die verschiedenen Arten, die beiden Chiffrearten anzugreifen. Symmetrische Chiffren wird man im allgemeinen durch einen Brute Force Angriff (also erschöpfendes Durchsuchen des Schlüsselraums) zu brechen versuchen. Falls man einen Algorithmus verwendet, der einen effektiveren Angriff als den Brute Force Angriff zuläßt, sollte man soweit möglich von diesem Algorithmus Abstand nehmen und einen sichereren verwenden. Asymmetrische Chiffren wird man i.a. angreifen, in dem man das zugrundeliegende Problem löst (z.B. beim RSA das Faktorisieren des Moduls n).

5.6 Weitere notwendige Algorithmen

5.6.1 Einweg-Hashfunktionen

Ein wichtiges Werkzeug, das in vielen kryptographischen Protokollen Verwendung findet, sind *Einweg-Hashfunktionen*. Eine Hashfunktion H erzeugt aus einem beliebig großen Datenblock D einen eindeutigen Hashwert $h = H(D)$ mit fester Länge m. Hashfunktionen finden auch in anderen Bereichen der Informatik Verwendung. Um für kryptographische Anwendungen zweckmäßig zu sein, muß aber noch mehr von der Funktion gefordert werden: Zu einem gegebenen Datenblock D muß es einfach sein, den Hashwert h zu berechnen, aber die Umkehrung, aus einem Hashwert h' einen Datenblock D' zu generieren mit $h' = H(D')$, sollte berechnungsmäßig unmöglich sein. Daher kommt auch die Bezeichnung “Einweg”-Hashfunktion. Ebenfalls berechnungsmäßig unmöglich sollte es sein, zwei Datenblöcke D und D' mit dem selben Hashwert $h = H(D) = H(D')$ zu finden. Daraus resultiert, daß auch kleinste Änderungen am Text zu unterschiedlichen Hashwerten führen müssen.

Authentifizierung ohne Verschlüsselung mittels Einweg-Hashfunktionen

Einweg-Hashfunktionen ermöglichen eine einfache Möglichkeit zur Authentifizierung von Kommunikationspartnern. Die Voraussetzung dafür ist, daß die an der Kommunikation beteiligten Parteien ein Geheimnis G teilen. Zur Erläuterung des Vorgehens betrachte man folgende Situation: Alice will Bob überzeugen, daß eine Nachricht M von ihr stammt. Um dieses zu tun, bildet Alice $h = H(M,G)$ also den Hashwert der Konkatenation der Nachricht und des Geheimnisses, wobei H eine Einweg-Hashfunktion ist, auf die sich Alice und Bob im Vorfeld geeinigt haben und G das gemeinsame Ge-

heimnis. Alice sendet h zusammen mit M an Bob. Bob, der G kennt, kann ebenfalls $H(M,G)$ bilden und das Ergebnis mit h vergleichen. Wenn die Werte übereinstimmen, weiß Bob, daß die Nachricht M wirklich von Alice geschickt wurde. Da nur er und Alice das Geheimnis G kennen, hätte niemand außer Alice den Hashwert h generieren können, und da H eine Einweg-Hashfunktion ist, ist es für dritte auch trotz der Kenntnis von h und M unmöglich G zu berechnen. Bob weiß außerdem, daß der Text M unverändert angekommen ist. Hätte eine dritte Partei die Nachricht M durch eine andere Nachricht M' ersetzt, wäre der resultierende Hashwert $h' = H(M',G)$ unterschiedlich zu dem mitgeschickten $h = H(M,G)$. Diese Art der Authentifizierung findet u.a. bei IPSec Verwendung.

Der Nachteil dieses Verfahrens ist es, daß es dritten gegenüber keine Authentifizierung bietet ("non-repudiation"). Alice kann sich zwar von der Identität Bobs überzeugen, hat aber keine Möglichkeit, einer dritten Person Clara zu beweisen, daß die Nachricht tatsächlich von Bob stammt, da Clara weder das Geheimnis G kennt, noch, bei Offenlegung des Geheimnisses, aus G auf Bob schließen kann.

Der Geburtstagsangriff

Der Geburtstagsangriff beruht auf dem sogenannten Geburtstagsparadoxon und verdeutlicht, warum man darauf achten sollte, keine Einweg-Hashfunktion mit zu kleinem Hashwert zu benutzen. Mit dem Geburtstagsparadoxon hat es folgende Bewandtnis: Die kleinste Anzahl an Personen in einem Raum, so daß die Wahrscheinlichkeit, daß eine dieser Personen den selben Geburtstag wie man selbst hat, größer als 50% ist, beträgt 253. Doch die kleinste Anzahl an Personen, die in einem Raum sein müssen, so daß die Wahrscheinlichkeit, daß zwei dieser Personen am selben Tag Geburtstag haben, größer als 50% ist, beträgt nur 23. Dieses erklärt sich durch die Tatsache, daß man mit 23 Personen 253 verschiedene Paare bilden kann. Diesen Umstand macht sich der Geburtstagsangriff zu nutze: Der gewöhnliche Brute Force Angriff auf einen Hashwert ist, daß man für einen Hashwert $H(M)$ einen Text M' sucht mit $H(M') = H(M)$. Bei einer Hashlänge von 64 Bit sind so bis zu 2^{64} Hashwerte zu berechnen. Bei einer Rechenleistung von 1.000.000 Berechnungen pro Sekunde wäre die Dauer eines solchen Angriffs 600.000 Jahre. Der Geburtstagsangriff verschiebt die Aufgabenstellung des Angriffs ein wenig. Anstatt daß zu einem existierenden $H(M)$ ein M' gefunden werden soll, werden hier zwei Dokumente M und M' gesucht, die den selben Hashwert ergeben. Dieses geschieht i.a. so, daß am Anfang zwei Texte existieren, die so modifiziert werden sollen, daß sie am Ende den gleichen Hashwert ergeben. Dieses geschieht, indem an beiden Dokumenten unauffällige, nur bei eingehender Untersuchung der Dokumente erkennbare Veränderungen vorgenommen werden (z.B. Einfügen von Leerzeichen oder ähnliches). Die Hashwerte für die verschiedenen Versionen der beiden Dokumente werden berechnet und zwischengespeichert. Nun haben wir die Situation, in der wir nur ein Paar mit gleichem Hashwert benötigen. Im allgemeinen sind bei einem Hashwert von 64 Bit nicht mehr als 2^{32} Hashwerte zu berechnen, bis zwei passende Versionen gefunden sind. Bei einer Rechenleistung von 1.000.000 Berechnungen pro Sekunde wäre die Dauer eines solchen Angriffs etwa eine Stunde. Also sollte man von Hashfunktionen mit einem Hashwert, der kleiner als 128 Bit ist, Abstand nehmen. Bedenkt man weiterhin, daß in vielen kryptographischen Protokollen, die zum Abschließen von Verträgen dienen, nicht der eigentliche Text des Vertrags signiert wird, sondern nur der Hashwert des Vertrags, ist es in solchen Fällen ratsam, bevor man den Vertrag unterzeichnet, eine kleine unmerkliche Veränderung an dem Dokument vorzunehmen und den Hashwert neu zu bilden. Wenn man nun diesen Hashwert unterzeichnet, kann man sicher sein, daß der Vertragspartner nicht noch einen weiteren, mittels des Geburtstagsangriffs generierten, Vertrag in der Hinterhand hat, der über denselben Hashwert verfügt, aber einen anderen Inhalt hat. Denn im nachhinein wäre dann nicht mehr festzustellen, welcher der beiden Verträge unterzeichnet wurde.

Beispiele für Einweghashfunktionen

Hier nun noch eine kurze Erwähnung von zwei verbreiteten Einweghashfunktionen, die als weitgehend sicher gelten:

Secure Hash Algorithm, SHA1 (NIST, 1994): SHA1 ist als Teil des DSA (siehe 5.6.1) von der NSA entwickelt worden und berechnet einen Hashwert von 160 Bit.

RIPE-MD 160, Europäische Union: RIPE-MD 160 ist eine sichere Weiterentwicklung von MD4 und berechnet 160 Bit große Hashwerte.

Digitale Signaturen

Für viele Kommunikationsprotokolle hätte man gerne ein elektronisches Äquivalent zur menschlichen Unterschrift, also eine Möglichkeit, digitale Dokumente zu unterzeichnen. Die exakten Anforderungen an eine sogenannte digitale Signatur lassen sich wie folgt auflisten:

Authentizität: Der Empfänger kann sich von der Identität des Unterzeichners überzeugen (Es muß für jede Person problemlos möglich sein festzustellen, von wem die digitale Signatur erstellt wurde. Die digitale Signatur identifiziert ihren Ersteller eindeutig).

Fälschungssicherheit: Nur dem Unterzeichner ist es möglich, die Signatur zu erzeugen.

Überprüfbarkeit: Eine dritte Partei kann jederzeit die Signatur verifizieren.

Keine Wiederverwendbarkeit: Die Signatur bezieht sich nur auf das unterzeichnete Dokument und kann keinesfalls für andere Dokumente verwendet werden.

Keine Veränderbarkeit, Integrität: Nachdem das Dokument unterzeichnet ist, kann es nicht mehr verändert werden.

Es ist ersichtlich, daß, wenn alle diese Forderungen erfüllt sind, Vorgänge wie digitales Unterzeichnen von Verträgen und ähnliches möglich sind. Digitale Signaturen kann man weiterhin zur Authentifizierung von Kommunikationspartnern oder Autoren digitaler Dokumente einsetzen.

Digitale Signaturen mittels asymmetrischer Chiffren

Vorausgesetzt die Reihenfolge der Anwendungen des geheimen und öffentlichen Schlüssels ist unerheblich, eignen sich asymmetrische Chiffren gut, um digitale Signaturen zu erstellen. Dabei wird wie folgt vorgegangen: Alice chiffriert das betreffende Dokument mit ihrem geheimen Schlüssel. Danach sendet sie das so unterzeichnete Dokument an Bob. Bob dechiffriert das Dokument mit Alices öffentlichem Schlüssel. Betrachten wir kurz, ob alle unsere Anforderungen an digitale Signaturen bei einem derartigen Vorgehen erfüllt werden:

Authentizität: Da Alices öffentlicher Schlüssel den Klartext ergibt, weiß Bob, daß das Dokument von Alice stammt.

Fälschungssicherheit: Nur Alice kennt ihren geheimen Schlüssel. Niemand anderes hätte die Signatur erstellen können.

Überprüfbarkeit: Jeder, der Alices öffentlichen Schlüssel kennt, kann die Signatur bestätigen. Alice kann sie nicht zurücknehmen.

Keine Wiederverwendbarkeit: Die Unterschrift bezieht sich nur auf dieses Dokument, andere Dokumente ergeben andere Signaturen.

Keine Veränderbarkeit, Integrität: Bei Veränderung des Dokuments ergibt die Dechiffrierung mit Alices öffentlichem Schlüssel kein sinnvolles Ergebnis.

In der Praxis wird oftmals nicht das gesamte Dokument unterzeichnet, (also chiffriert) sondern nur der Hashwert des Dokuments. Dieses geschieht vor allem aus Gründen der Effizienz (außerdem bleibt so das Dokument selbst, auch ohne Überprüfung der digitalen Signatur, lesbar). Man kann sich davon überzeugen, daß bei Benutzung einer Einweg-Hashfunktion mit einem ausreichend großem Hashwert sich die Sicherheit der digitalen Signatur dabei nicht ändert.

Digitale Signaturen mittels DSA

Bei digitalen Signaturen mit asymmetrischen Chiffren geht mit der Signierung eine Verschlüsselung des Dokuments einher (falls das ganze Dokument und nicht nur der Hashwert signiert wird). Dies ist nicht in allen Fällen wünschenswert. Weiterhin muß man beachten, daß in manchen Ländern Benutzung oder Export von Software, die Verschlüsselung ermöglicht, restringiert ist. Also gibt es einen Bedarf nach Algorithmen, die digitale Signaturen ohne einhergehende Verschlüsselung ermöglichen. Der Digital Signature Algorithm (DSA), der 1991 von der NSA entwickelt wurde und Teil des DSS (Digital Signature Standard) ist, verfolgt einen derartigen Ansatz. Bei korrekter Implementation eignet er sich nur zum Signieren, aber nicht zum Verschlüsseln. Auch beim DSA wird mit öffentlichen und geheimen Schlüsseln gearbeitet und die Signatur wird auch hier unter Verwendung des öffentlichen Schlüssels überprüft. Die Sicherheit des DSA beruht auf der Schwierigkeit, den diskreten Logarithmus über endlichen Körpern zu berechnen. Die Schlüssellänge ist variabel zwischen 512 und 1024 Bit. Wie auch schon beim DES wurde dem DSA bei seiner Einführung mit Mißtrauen begegnet, da an seiner Entwicklung die NSA beteiligt war. Tatsächlich besteht bei maliziöser Implementation des Algorithmus die Möglichkeit, einen verdeckten Kanal in den DSA einzubauen, der nach und nach den geheimen Schlüssel des Anwenders nach außen schleust. Da ein solcher Kanal unmöglich zu entdecken ist, sollte man nur vertrauenswürdige Implementationen des DSA verwenden.

5.6.2 Weiteres wichtiges Zubehör

Für viele kryptographische Anwendungen und Protokolle benötigt man neben den bereits vorgestellten Werkzeugen wie Chiffren und Einweg-Hashfunktionen noch weiteres Handwerkszeug.

Zufallszahlengeneratoren: Als erstes seien hier die kryptographisch starken Zufallszahlengeneratoren genannt. In vielen Anwendungen und Protokollen werden Schlüssel zur Datensicherung oder Kommunikation aus Zufallszahlen gewonnen. Für diese Fälle benötigt man einen Zufallszahlengenerator, dessen Ausgaben nicht vorhersehbar sind. Genauer gesagt muß es unmöglich sein, ohne Kenntnis des Initialwertes des Generators, ausschließlich aus Kenntnis der vorherigen Zufallszahlen und des Algorithmus, den nächsten Zufallswert vorherzusagen. Falls ein Algorithmus Verwendung findet, der dieses nicht leistet, erschließt sich, z.B. in einer Situation, in der die Zufallszahl zur Schlüsselgenerierung benutzt wird, ein möglicher Angriffspunkt auf

die Verschlüsselung. Anstatt die Chiffre oder den Schlüssel angreifen zu müssen, können potentielle Angreifer versuchen, den verwendeten Schlüssel zu "erraten". Falls sie in Besitz von vorher verwendeten Schlüssel-Informationen sind und der Zufallszahlengenerator noch nicht neu initialisiert wurde, ist ein derartiges Vorgehen äußerst vielversprechend.

Synchronisierte Uhren: Viele kryptographische Protokolle benutzen Zeitstempel, um Angriffe, die durch das Wiedereinspielen von abgefangenen Daten stattfinden, abzuwehren. Die einzelnen Nachrichten eines Protokolls werden mit einem Zeitstempel, der Informationen über Datum und Uhrzeit enthält, versehen. Wenn eine Nachricht eintrifft, deren Zeitstempel älter als eine festgelegte Schranke ist, wird diese Nachricht nicht mehr akzeptiert. Ein solches Verfahren ist nur möglich, wenn die Teilnehmer an dem Protokoll über synchronisierte Uhren verfügen. Doch dieses ist kein triviales Unterfangen, besonders bei Kommunikation, die über die Grenzen eines kleineren Netzwerkes hinausgeht.

Hochauflösende Zeitstempel: Die Notwendigkeit für hochauflösende Zeitstempel hängt eng mit dem Punkt der synchronisierten Uhren zusammen. Viele Betriebssysteme liefern nur sehr grob gemessene Zeitangaben. Für manche Protokolle ist es aber nötig, Unterschiede in Zeitstempeln, die in den Bereich von Zeiträumen unter einer Nanosekunde fallen, zu erkennen. Dafür reichen dann oft die vom Betriebssystem zur Verfügung gestellten Hilfsmittel nicht aus.

5.6.3 Kryptographische Protokolle

Ein kryptographisches Protokoll ist ein Kommunikationsprotokoll, das mindestens ein Sicherheitsziel wie beispielsweise Authenzität, Vertraulichkeit oder Integrität der übertragenen Daten gewährleisten kann. Es definiert eindeutig eine festgelegte Abfolge von Handlungen der Kommunikationspartner. Dabei wird, je nach Aufgabe des Protokolls, bestimmt, welches Format die verschickten Nachrichten haben, welche Algorithmen verwendet werden, wie die Schlüssel ausgetauscht werden und ähnliches. Dabei finden üblicherweise mehrere kryptographische Algorithmen Verwendung. Man findet kryptographische Protokolle in allen Schichten der TCP/IP-Familie. Nur um ein paar Namen zu erwähnen, seinen hier als Beispiele in der Transportschicht *TLS* und *SSL* und in der Anwendungsschicht *SSH*, *Kerberos* und *PGP* genannt.

5.7 Schlußbemerkung

Als abschließende Worte kann man sagen, daß Algorithmen zur hinreichend guten Absicherung von Daten vorhanden sind. Mit den hier vorgestellten Algorithmen können die eingangs erwähnten Ziele erreicht werden. Die Voraussetzung dafür ist aber, daß alle Komponenten des verwendeten kryptographischen Protokolls aufeinander abgestimmt sind. Es nutzt niemandem, wenn man den Schlüsselaustausch mittels 2048 Bit großer öffentlicher Schlüssel sichert, aber dann für die eigentlichen Daten nur einen 56 Bit großen DES Schlüssel verwendet. Ein Protokoll ist immer nur so stark wie seine schwächste Komponente. Auch die verwendeten Protokolle sollten auf ihre Sicherheit überprüft werden, denn auch bei perfekt sicherer Verschlüsselung kann ein unzureichendes Protokoll Sicherheitslücken eröffnen. Am wichtigsten ist eine konsequente und korrekte Benutzung der kryptographischen Algorithmen und Protokolle. Die größten Gefahren für die Datensicherheit sind noch immer ein nachlässiger Benutzer und schlechte Implementierungen.

Literaturverzeichnis

[Schnei97] Bruce Schneier: “Applied Cryptography – Protocols, Algorithms and Source Code in C”, 2. Auflage, John Wiley & Sons 1997

Kapitel 6

Keymanagement und Kommunikationsverbindungen

Marcel Kronberg
Universität Hamburg
Fachbereich Informatik

Zusammenfassung

Der Begriff Keymanagement bezeichnet die Verwaltung von Schlüsseln für kryptographische Algorithmen. Zu den Aufgaben zählen Schlüsselgenerierung, Schlüsselübertragung, Schlüsselverifizierung, Schlüsselspeicherung und Schlüsselzerstörung. Die Herausforderung im Keymanagement gründet sich auf der Tatsache, daß Verschlüsselungsalgorithmen meist öffentlich bekannt sind, um ihre Stärke und Sicherheit zu beweisen, die Schlüssel aber geheimgehalten werden müssen. Die Sicherheit eines kryptographischen Systems basiert somit hauptsächlich auf der Geheimhaltung von Schlüsseln. Dazu bedarf es komplexer Sicherheitskonzepte. In diesem Dokument wird auf die Schlüsselübertragung im Zusammenhang mit der Absicherung von Kommunikationsbeziehungen eingegangen. Dabei wird zuerst im Abschnitt 1 die Aufgabe des Keymanagements bei der Absicherung von Kommunikationsverbindungen definiert. Danach werden im Abschnitt 2 Grundlagenverfahren vorgestellt. Abschließend wird im Abschnitt 3 ein Überblick über die Funktionsweise von zwei Keymanagement-Protokollen gegeben.

6.1 Aufgabe des Keymanagements bei der Absicherung von Kommunikationsbeziehungen

Eine Kommunikationsbeziehung wird als sicher angesehen, wenn sie Eigenschaften wie Vertraulichkeit, Integrität, Authentizität und Unleugbarkeit besitzt. Der Einsatz von kryptographischen Verfahren verleiht Kommunikationsbeziehungen diese Eigenschaften. Einige kryptographische Verfahren wie beispielsweise Verschlüsselungsverfahren benötigen Schlüssel. Die Kommunikationspartner einer Kommunikationsbeziehung müssen in Besitz der Schlüssel sein, um miteinander einen sicheren Datenverkehr zu etablieren. Folglich muß ein Austausch der Schlüssel stattfinden. Die Aufgabe des

Keymanagements bei der Absicherung von Kommunikationsbeziehungen besteht in der sicheren Etablierung von kryptographischem Schlüsselmaterial, um den Datenverkehr abzusichern.

6.2 Verfahren zur Absicherung von Kommunikationsbeziehungen

6.2.1 Verschlüsselungsverfahren – Vertraulichkeit – Authentizität

Verschlüsselungsverfahren sind ein zentraler Bestandteil bei der Absicherung von Kommunikationsbeziehungen. Es werden sowohl symmetrische als auch asymmetrische Verschlüsselungsverfahren verwendet.

Symmetrische Verschlüsselungsverfahren

Symmetrische Verschlüsselungsverfahren werden für die Ver- und Entschlüsselung der Daten während der Kommunikation eingesetzt. Sie gewährleisten höhere Geschwindigkeiten für Ver- und Entschlüsselung sowie eine kürzere Schlüssellänge als asymmetrische Verschlüsselungsverfahren. Am Anfang der Kommunikationsbeziehung wird zwischen den Kommunikationspartnern Schlüsselmaterial ausgetauscht, aus dem bei beiden Kommunikationspartnern symmetrische Schlüssel errechnet werden. Für jede neue Kommunikationsbeziehung werden immer wieder neue symmetrische Schlüssel etabliert. Das hat den Vorteil, daß die Schlüssel nicht auf der Festplatte gespeichert werden müssen, von der sie unter Umständen leicht gestohlen werden können. Ferner schützt das Brechen eines Schlüssels die Nachrichten anderer Kommunikationsbeziehungen vor unbefugten Zugriff.

Asymmetrische Verschlüsselungsverfahren

Der Einsatz symmetrischer Verschlüsselungsverfahren erfordert einen vorherigen Austausch von geheimem Schlüsselmaterial. Dafür werden asymmetrische Verfahren eingesetzt. Diese sichern die geheime Übertragung und Authentisierung von symmetrischem Schlüsselmaterial. Es werden nun zwei asymmetrische Verfahren vorgestellt.

Austausch von Schlüsselmaterial nach Diffie Hellman

1976 wurde von Whitfield Diffie und Martin Hellman ein Algorithmus publiziert [Schnei97], der es beiden Kommunikationspartnern nach Austausch von zwei öffentlichen Nachrichten ermöglicht, einen gemeinsamen geheimen Schlüssel zu errechnen. Dazu berechnet Kommunikationspartner A einen Wert $X = g^x \bmod n$ und schickt diesen an Kommunikationspartner B. Dieser sendet einen Wert $Y = g^y \bmod n$ zurück zu Kommunikationspartner A. Nun berechnet Kommunikationspartner A den Schlüssel $k = Y^x \bmod n$ und Kommunikationspartner B den Schlüssel $k' = X^y \bmod n$. Die Schlüssel k und k' stimmen überein, denn es gilt: $k = k' = g^{xy} \bmod n$. Es wurde mit öffentlichem Schlüsselmaterial (X, Y, g, n) ein gemeinsamer geheimer Schlüssel $k = k' = g^{xy} \bmod n$ vereinbart. Die Sicherheit des Algorithmus beruht auf der Schwierigkeit, diskrete Logarithmen schnell und effizient zu berechnen. Für große Zahlen ist die Berechnung diskreter Logarithmen praktisch unmöglich. Die Werte x und y sind geheim und werden in keiner Nachricht an den jeweiligen Kommunikationspartner übermittelt. Das Verfahren beinhaltet nicht die Authentisierung des öffentlichen Schlüsselmaterials (X, Y, g, n).

Es ist möglich, Nachrichten abzufangen und manipulierte Nachrichten an Stelle der Ursprungsnachrichten weiterzuleiten (Man in the middle Angriff). Darum werden für die Authentisierung andere asymmetrische Verfahren wie Verschlüsselung mit öffentlichen Schlüsseln oder digitale Signierung angewendet.

Verschlüsselung mit öffentlichen Schlüsseln

Im Gegensatz zum Diffie Hellman Verfahren wird bei der Verschlüsselung mit öffentlichen Schlüsseln das Schlüsselmaterial nicht mit öffentlichen, sondern mit geheimen Nachrichten übertragen. Die Kommunikationspartner besitzen jeweils zwei Schlüssel, einen öffentlichen und einen privaten. Der öffentliche Schlüssel dient zur Verschlüsselung und ist öffentlich bekannt. Der private Schlüssel ist geheim und dient zur Entschlüsselung von Nachrichten, die mit dem öffentlichen Schlüssel verschlüsselt wurden. Bei der Schlüsselübertragung wird mit dem vorher ausgetauschten öffentlichen Schlüssel das Schlüsselmaterial für die Übertragung verschlüsselt und zum jeweiligen Kommunikationspartner übertragen. Nur der Inhaber des zum jeweiligen öffentlichen Schlüssel dazugehörigen privaten Schlüssel kann die gesendeten Nachrichten entschlüsseln und darauf reagieren. Damit ist es möglich, geheimes Schlüsselmaterial zu übertragen. Die Authentisierung durch öffentliche Schlüssel erfordert jedoch weitere zusätzliche Verfahren, da die Zusammengehörigkeit von einem öffentlichen Schlüssel zu einem Kommunikationspartner nicht unmittelbar feststellbar ist. Es ist möglich, daß Dritte ihren öffentlichen Schlüssel unter Vorgabe einer falschen Identität in Umlauf bringen und somit eine scheinbare Kommunikation mit dem erwünschten Kommunikationspartner vortäuschen.

Zertifikate

Um sicherzustellen, daß die Kommunnikationsbeziehung auch mit dem erwünschten Kommunikationspartner stattfindet, bedarf es zusätzlicher Instanzen, die die Zusammengehörigkeit von öffentlichem Schlüssel zu einem Kommunikationspartner beglaubigen. Diese Instanzen sind Zertifizierungsstellen in Zertifizierungsinfrastrukturen, die nach den Richtlinien ihrer Policy [DFN-PCA, 1999] arbeiten und Zertifikate ausstellen. Zertifikate beglaubigen die Zusammengehörigkeit eines öffentlichen Schlüssels zu einer Identität. Identitäten können Personen, Prozesse oder Rechner sein. Technisch betrachtet ist ein Zertifikat eine digitale Signatur über Identitätsinformationen mit dem privaten Schlüssel der Zertifizierungsstelle. Dadurch kann zwar zweifelsfrei nachgewiesen werden, welche Zertifizierungsstelle das Zertifikat ausgestellt hat, jedoch ist nicht ersichtlich, unter welchen Regeln die Beglaubigung erfolgte. Das ist vor allem bei unbekannten Zertifikaten problematisch. Die Prüfung des Zertifikates kann durch das Herunterladen des Zertifikates von der Zertifizierungsstelle oder einem Verzeichnisdienst erfolgen. Der Echtheitsnachweis muß nicht während der Kommunikationsbeziehung stattfinden. Weitere Informationen über Zertifizierungsinfrastrukturen und Zertifikate findet man im PKI-Beitrag in diesem Band (Kapitel 10).

6.2.2 Prüfsummen – Integrität

In den voherigen Abschnitten wurde dargestellt, mit welchen Verfahren Vertraulichkeit und Authentizität hergestellt werden kann. Die Eigenschaft der Datenintegrität bezeichnet die Fähigkeit zu erkennen, ob die Daten während der Übertragung verändert wurden. Dafür werden kryptographische Prüfsummenverfahren verwendet. Beispiele für kryptographische Prüfsummenverfahren sind MD5

[RFC 1321] oder SHA-1 [NIST, 1994]. Prüfsummenverfahren werden meistens über ganze Nachrichtenketten angewendet. Da beide Kommunikationspartner das Verfahren kennen, können sie selbst eine Prüfsumme über die empfangenen Nachrichten ermitteln und mit der erhaltenen Prüfsumme vergleichen. Bei Manipulation an Nachrichten ändern sich die Prüfsummen und stimmen nicht mehr mit den selbst errechneten überein. Somit sind Manipulationen an Nachrichten erkennbar.

6.2.3 Zusammenfassung

Bei der kryptographischen Absicherung von Kommunikationsbeziehungen werden symmetrische Verschlüsselungsverfahren zum Verschlüsseln (Vertraulichkeit) des Datenverkehrs benutzt. Die dafür notwendigen geheimen Schlüssel werden während des Verbindungsaufbaus errechnet und sind nur während der Kommunikationsverbindung gültig. Für jede Verbindung werden neue geheime Schlüssel errechnet. Das Schlüsselmaterial, das für die Ermittlung des geheimen Schlüssels notwendig ist, wird mit Hilfe von asymmetrischen Verfahren übermittelt. Hier können nun mehrere Verfahren gleichzeitig zum Einsatz kommen. Zum Schlüsseltransport eignet sich das Diffie Hellman Verfahren. Mit zwei öffentlichen Nachrichten wird ein gemeinsamer geheimer Schlüssel etabliert. Das Diffie Hellman Verfahren umfaßt nicht die Authentisierung der öffentlichen Nachrichten. Darum kommt zusätzlich die Authentisierung mit öffentlichen Schlüsseln in Verbindung mit Zertifikaten zum Einsatz. Zertifikate beglaubigen die Zugehörigkeit eines öffentlichen Schlüssels zu einer Identität durch die digitale Signatur der im Zertifikat eingetragenen Identitäts- und Schlüsselangaben. Zertifikate werden von einer Zertifizierungsstelle ausgestellt, die sich in einer Zertifizierungsinfrastruktur befindet und nach den Richtlinien ihrer Policy arbeitet. Zur Integritätssicherung werden kryptographische Prüfsummen eingesetzt.

6.3 Keymanagement-Protokolle

Kryptographische Protokolle stellen Dienste bereit, die Authentizität, Vertraulichkeit, Integrität und Unleugbarkeit von Kommunikationsbeziehungen anbieten. Voraussetzung für die Bereitstellung der Dienste kryptographischer Protokolle ist die Einigung über kryptographische Verfahren, die beide Kommunikationspartner unterstützen, sowie die sichere Übertragung der dazugehörigen Parameter. Hierzu werden sogenannte „Keymanagement-Protokolle“ eingesetzt. Ziel ist der Abschluß von Abkommen, nach denen die kryptographischen Protokolle ihre Dienste bereitstellen. Momentan werden zwei Ansätze von Keymanagement verfolgt. Entweder wird das Keymanagementprotokoll auf der gleichen Schicht wie das kryptographische Protokoll angesiedelt, oder es gibt ein universelles Keymanagementprotokoll, welches kryptographischen Protokollen auf anderen Ebenen den Dienst zum Verhandeln kryptographischer Verfahren und der dazugehörigen Parameter zur Verfügung stellt. Im folgenden wird das „Internet Key Exchange Protokoll (IKE)“ auf der Anwendungsschicht und das „Transport Layer Security Handshake Protokoll (TLS Handshake)“ als Keymanagementprotokoll für das „Transport Layer Security Protokoll (TLS)“ vorgestellt.

6.3.1 Internet Key Exchange Protokoll (IKE)

Das IKE Protokoll entstand aus der Abbildung des „Oakley Key Determination Protokoll (Oakley)“ [RFC 2412] auf das „Internet Security Association and Key Management Protocol (ISAKMP)“. Das ISAKMP Protokoll [RFC 2408] legt dabei allgemeine Formate und Prozeduren für die Erzeugung,

Modifikation und Vernichtung von „Security Associations (SAs)“ [RFC 2401] fest. Oakley ist eine konkrete Schlüsseltauschdefinition nach ISAKMP. Es regelt den Austausch und die Verifikation von Schlüsselmaterial. Die Standardisierung von ISAKMP / Oakley wurde IKE [RFC 2409] genannt. IKE ist ein Protokoll auf der Anwendungsebene. Es kann Protokollen auf anderen Schichten den Dienst zur Etablierung von kryptographischen Verfahren erbringen, falls diese das Konzept der „Security Associations (SAs)“ unterstützen. Dabei kann es sowohl für die Etablierung von „Authentication Header“ SAs [RFC 2402] wie auch für „Encapsulated Security Payload“ SAs [RFC 2406] der „IP Sicherheitserweiterungen (IPSec)“ genutzt werden. Dazu verwendet IKE das zweiphasige Verhandlungskonzept aus ISAKMP. In der ersten Phase wird ein sicherer Kommunikationskanal durch Etablierung einer ISAKMP SA aufgebaut, unter dessen Schutz in der zweiten Phase die Etablierung der SAs anderer Protokolle stattfindet. Der für die Etablierung von Abkommen notwendige Austausch von Schlüsselmaterial erfolgt in unterschiedlichen Betriebsarten nach dem Diffie Hellman Verfahren. Als Betriebsarten der ersten Phase werden Main und Aggressive Mode unterschieden. Die zweite Phase stellt den Quick Mode zur Etablierung von SAs anderer Protokolle bereit.

Authentisierung

Während der beiden Phasen werden „Anti Clogging Tokens“ [RFC 2422] zur schwachen Authentisierung benutzt. Diese dienen als Bezeichner der ISAKMP SA in Phase I. In Phase II werden zusätzlich Nachrichten-IDs zur Bezeichnung der SAs anderer Protokolle genutzt. Phase I stellt vier Authentisierungsverfahren zur Authentisierung des „Diffie Hellman“ Austausches bereit. Es kann zwischen Authentisierung mit digitalen Signaturen, Authentisierung mit vorher ausgetauschten öffentlichen Schlüsseln, einem erweiterten Modus der Authentisierung mit vorher ausgetauschten öffentlichen Schlüsseln und der Authentisierung mit privaten Schlüsseln gewählt werden. Der Ablauf des Austausches und die Generierung des Sitzungsschlüssels variiert nach der Authentisierungsmethode.

Protokollphasen: Phase I – Main und Agressive Mode

In Phase I können zwei Betriebsarten (Main Mode und Aggressive Mode) zur Etablierung einer ISAKMP SA benutzt werden. Im Main Mode werden dafür sechs Nachrichten benötigt. In den ersten beiden Nachrichten wird verhandelt, mit welchen kryptographischen Verfahren (SA) die Kommunikationsbeziehung abgesichert werden soll. Dazu gehören der Verschlüsselungsalgorithmus, Hashalgorithmus und die Authentisierungsmethode. In der dritten und vierten Nachricht erfolgt der Austausch von Schlüsselmaterial. Hierbei handelt es sich um DH Exponenten (KE) und Zufallszahlen (Ni und Nr). Jetzt wird bei beiden Kommunikationspartnern ein gemeinsamer Schlüssel (Sitzungsschlüssel) errechnet. Von nun an können nachfolgende Nachrichten verschlüsselt übertragen werden. Die fünfte und sechste Nachricht authentisieren den Austausch, indem Prüfsummen über Nachrichtenbestandteile der vorherigen Nachrichten und Identitätsinformationen (IDii und IDir) übertragen werden.

Der Aggressive Mode benötigt für die Etablierung der ISAKMP SA nur drei Nachrichten. In den ersten beiden Nachrichten erfolgt die Einigung, mit welchen kryptographischen Verfahren (SA) die Kommunikationsbeziehung abgesichert werden soll, die Übertragung von Schlüsselmaterial (KE, Ni, Nr) und Identitätsinformation (IDii, IDir). Darüber hinaus enthält die zweite Nachricht bereits eine Prüfsumme zur Authentisierung des Responder. Die dritte Nachricht authentisiert den Initiator.

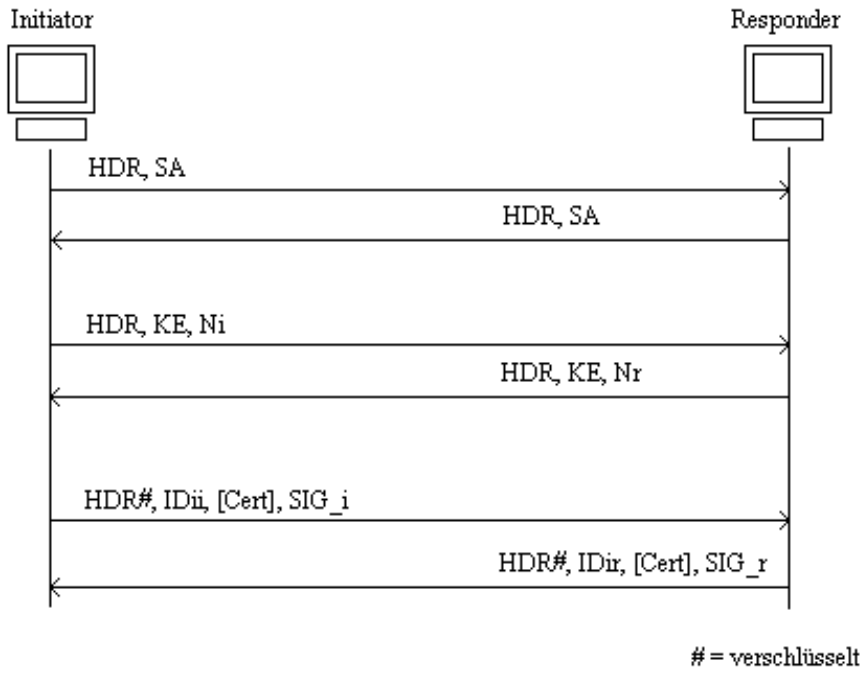


Abbildung 6.1: Main Mode - Authentisierung mit digitalen Signaturen

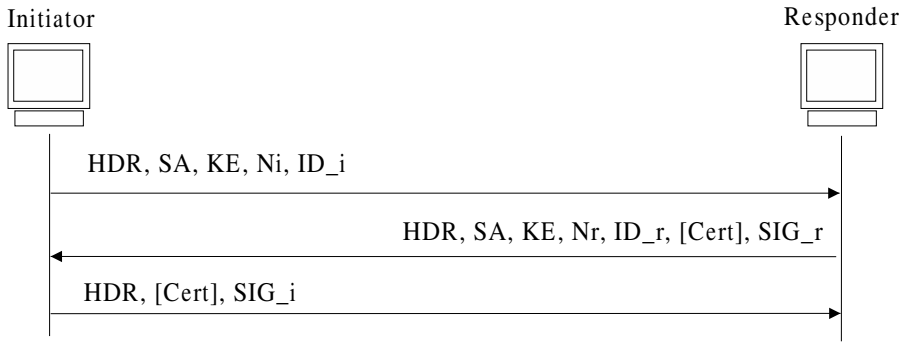


Abbildung 6.2: Aggressive Mode

Protokollphasen: Phase II – Quick Mode

In der zweiten Phase werden SAs für andere Protokolle (z.B. IPSec) etabliert. Die dazu verwendete Betriebsart heißt Quick Mode. Die Etablierung erfolgt bereits unter dem Schutz der in Phase eins vereinbarten ISAKMP SA. Der Quick Mode wird an den Nachrichten-IDs im ISAKMP Header erkannt. Im wesentlichen besteht der Quick Mode aus der Verhandlung einer SA für ein Protokoll, sowie den Austausch von Zufallszahlen, um neues Schlüsselmaterial bereitzustellen und Wiederholungsangriffe (replay attacks) zu erkennen. Die Etablierung der SA erfolgt in drei Nachrichten. In der ersten Nachricht übermittelt der Initiator einen Vorschlag über die zu verwendenen kryptographischen Verfahren (SA) und gegebenenfalls das Schlüsselmaterial zur Ableitung von Sitzungsschlüsseln. Die zweite Nachricht enthält die von dem Empfänger ausgewählten Verfahren und gegebenenfalls das Schlüsselmaterial zur Ableitung von Sitzungsschlüsseln. Die dritte Nachricht authentisiert die vorherigen Nachrichten.

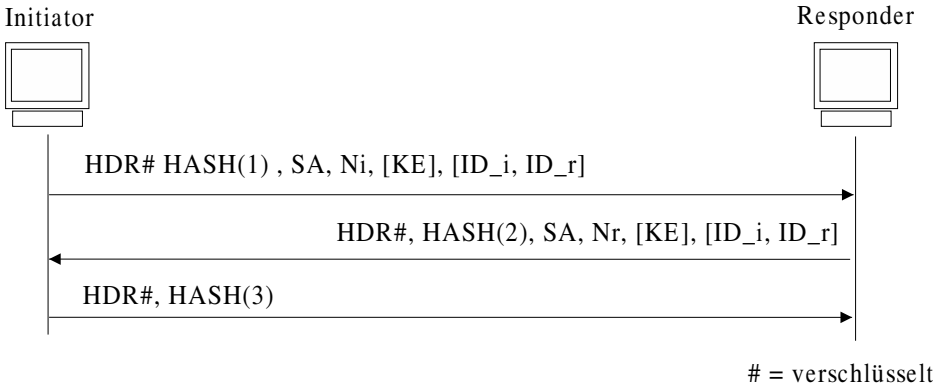


Abbildung 6.3: Quick Mode

Weiterhin gibt es in IKE noch Betriebsarten, die keiner Phase genau zuzuordnen sind. Diese werden hier jedoch nicht besprochen.

Zusammenfassung

IKE ist ein Keymanagementprotokoll auf der Anwendungsebene, welches Protokollen auf anderen Ebenen den Dienst zur Etablierung von SAs bereitstellt. Dafür werden zwei Verhandlungsphasen genutzt. In Phase I erfolgt die Etablierung einer ISAKMP SA, mit der die Etablierung der SAs anderer Protokolle in Phase II geschützt wird. Der Austausch von Schlüsselmaterial während der Phasen erfolgt in unterschiedlichen Betriebsarten mit dem Diffie Hellman Verfahren. In Phase I kann zwischen Main und Aggressive Mode gewählt werden. Die Authentisierung erfolgt mit digitalen Signaturen, vorher ausgetauschten öffentlichen Schlüsseln oder privaten Schlüsseln. Phase II stellt den Quick Mode zur Etablierung der SAs anderer Protokolle bereit. Die Authentisierung erfolgt durch die in Phase I etablierte ISAKMP SA. IKE bricht mit dem Entwurfsprinzip, nach dem jede Protokollschicht

den Dienst einer darunterliegenden Schicht annimmt und der nächst höheren Schicht einen Dienst anbietet.

6.3.2 Transport Layer Security Protokoll (TLS)

Das TLS Protokoll [RFC 2246] ist die standardisierte und leicht überarbeitete Version des von Netscape entwickelten SSL Protokolls [Frier, 1992]. Es ist ein zusätzliches Protokoll im Protokollstack, welches sich zwischen Transport und Anwendungsschicht befindet. Dadurch ist es möglich TLS auch mit anderen verbindungsorientierten Protokollen einzusetzen. Ziel von TLS ist es, die Geheimhaltung von Daten zwischen zwei Anwendungen sicherzustellen. Das TLS Protokoll ist aus zwei Schichten aufgebaut. In der oberen Schicht befinden sich Handshake Protokoll, Change Chipher Spec Protokoll, Alert Protokoll und Application Data Protokoll. In der unteren Schicht befindet sich das TLS Record Protokoll.

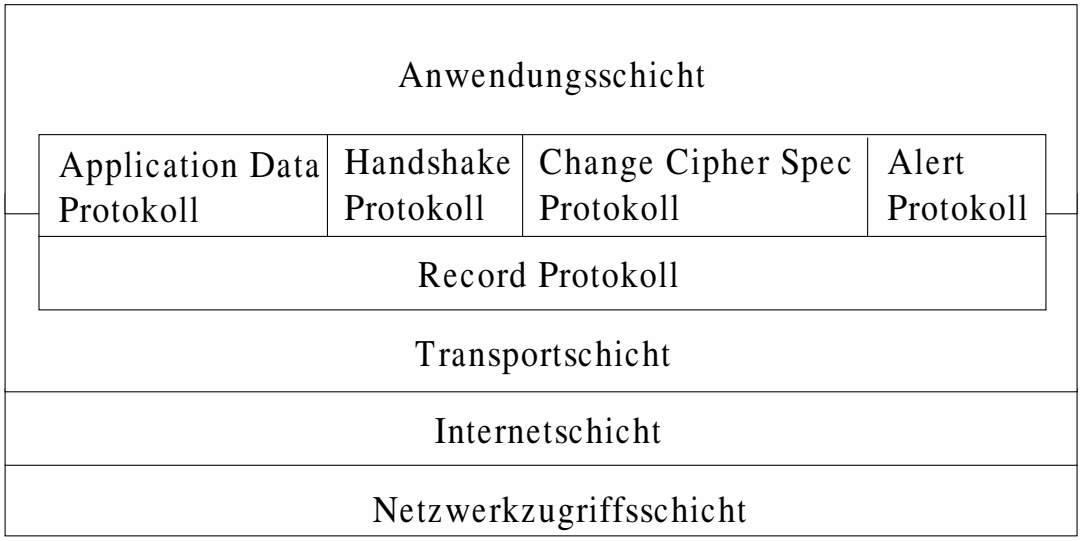


Abbildung 6.4: Lage von TLS im Protokollstack

Die Authentisierung sowie die Etablierung der symmetrischen Schlüssel für die Verschlüsselung regelt das TLS Handshake Protokoll. Im Change Chipher Spec Protokoll wird das Wechseln der Verschlüsselungsmethode während einer bestehenden Verbindung geregelt. TLS Handshake und Change Cipher Spec Protokoll bilden somit die Keymanagement Protokolle im TLS Protokoll. Fehlernachrichten der Kommunikationspartner meldet das TLS Alert Protokoll. Die Weiterleitung der Daten aus der Anwendungsschicht in die Record Schicht führt das Application Data Protokoll aus. Kompression, Verschlüsselung und Integritätsprüfung der Nachrichten der Protokolle der oberen Schicht regelt das TLS Record Protokoll.

Das TLS Handshake Protokoll – Authentisierung der Kommunikationspartner

Die Authentisierung in TLS findet zertifikatbasiert statt. Es werden die Authentisierungsmodi anonym, Server-Authentisierung sowie Client-und Server-Authentisierung unterschieden. Im anonymen Authentisierungsmodus findet keine Authentisierung der Partner statt. Bei der Server-Authentisierung authentisiert sich nur der Server gegenüber dem Client. Client und Serverauthentisierung erfordert einen gegenseitigen Zertifikatsaustausch und ermöglicht somit eine gegenseitige Authentisierung.

TLS Handshake – Schlüsseletablierung

Zuerst wird mit den „Hello“-Nachrichten verhandelt, wie die Verbindung aufgebaut wird. Hierbei handelt es sich um die Einigung über Verschlüsselungs- und Kompressionsverfahren. Danach erfolgt eine Authentisierung mit Zertifikaten. Der zur Übertragung des „Premaster Geheimnisses“ notwendige Austausch von Schlüsselmaterial kann entweder mit RSA [Rivest, 1978] oder Diffie Hellman erfolgen. Vom Client wird ein „Premaster Geheimnis“ gebildet und verschlüsselt zum Server übertragen. Der Server entschlüsselt das „Premaster Geheimnis“ mit seinem privaten Schlüssel. Client und Server errechnen mit dem „Premaster Geheimnis“ das „Master Geheimnis“. Das „Master Geheimnis“ dient der Berechnung eines Schlüsselblocks. Aus dem Schlüsselblock werden Sitzungsschlüssel entnommen, mit denen der Datenverkehr verschlüsselt wird. Dabei unterscheidet sich der Sitzungsschlüssel des Client von Sitzungsschlüssel des Servers. Der Datenstrom vom Client zum Server wird also mit einem anderen Schlüssel verschlüsselt als der Datenstrom vom Server zum Client.

Letztlich erfolgt die Überprüfung des Handshake. Beide Kommunikationspartner überprüfen in der letzten Nachricht des Handshake („Finished“-Nachricht) ihr „Master Geheimnis“, indem sie eine Prüfsumme über das „Master Geheimnis“ und alle vorher im Handshake getauschten Nachrichten übermitteln. Die Prüfsumme der „Server Finished“-Nachricht wird dabei auch über die „Client Finished“-Nachricht gebildet.

Nachrichtenabfolge

Der Client leitet den Verbindungsaufbau mit einem „Client Hello“ ein. Die „Client Hello“-Nachricht enthält einen Zeitstempel und eine Zufallszahl, die später für die Ableitung von Sitzungsschlüsseln benutzt wird. Bestand bereits eine Verbindung, so kann der Client die Sitzungs-ID der damaligen Sitzung angeben, um diese wieder aufzunehmen oder eine identische Sitzung zu eröffnen. Darüber hinaus enthält die „Client Hello“-Nachricht eine sortierte Liste mit den Verfahren, die die Verbindung schützen sollen.

Der Server antwortet mit einer „Server Hello“-Nachricht. Diese enthält eine Protokollversionsnummer, einen Zufallswert, die Sitzungs-ID sowie die gewählten kryptographischen Verfahren (Authentisierungsmethode, Verschlüsselungsalgorithmus) und die gewählte Komprimierungsmethode. Der Server bestimmt dabei über die Verfahren, die verwendet werden. Die Sitzungs-ID stimmt im Fall einer wieder aufgenommenen Sitzung mit der vom Client vorgeschlagenen überein. Andernfalls wird vom Server eine neue Sitzungs-ID vergeben.

Abhängig von der verwendeten Authentisierungsmethode wird nun eine „Certificate“-Nachricht vom Server zum Client übermittelt. Diese enthält eine Liste von Zertifikaten. Optional kann der Server mit einer „Certificate Request“-Nachricht ein Zertifikat vom Client anfordern, um eine Client-Authentisierung zu ermöglichen. Mit einer „Server Hello Done“-Nachricht zeigt der Server danach an, daß von ihm keine weiteren Nachrichten außer der Finished Nachricht übermittelt werden.

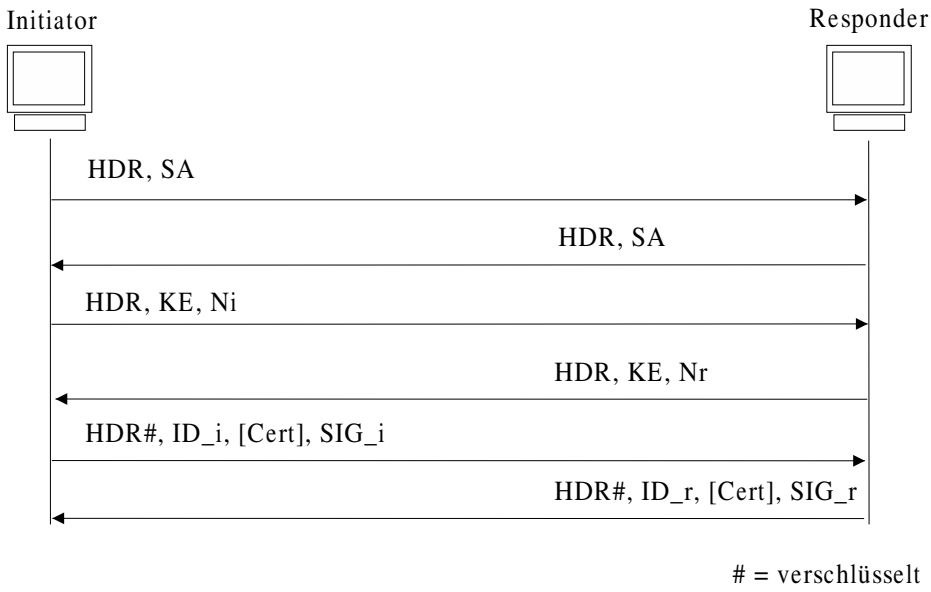


Abbildung 6.5: TLS Handshake

Falls der Server ein Client-Zertifikat angefordert hat, erfolgt nun die Übermittlung des Client-Zertifikates. Unabhängig von der „Client Certificate“-Nachricht folgt vom Client nach einer „Server Hello Done“-Nachricht immer eine „Client Key Exchange“-Nachricht. Diese enthält ein vom Client errechnetes „Premaster Geheimnis“. Das „Premaster Geheimnis“ ist verschlüsselt. Der dafür notwendige Schlüssel wurde je nach Vereinbarung und Authentisierungsmethode entweder aus dem Serverzertifikat entnommen, in einer „Server Key Exchange“-Nachricht übertragen oder mittels Diffie Hellman Austausch zwischen den Kommunikationspartnern ausgehandelt.

Wurde vorher eine „Client Certificate“-Nachricht mit einem Zertifikat übermittelt, mit dem signiert werden kann, so sendet der Client nun eine „Client Verify“-Nachricht, die eine kryptographische Prüfsumme über alle vorher im Handshake ausgetauschten Nachrichten enthält. Dadurch werden Manipulationen Dritter an den während des Handshakes ausgetauschten Nachrichten erkennbar.

Beide Kommunikationspartner errechnen nun aus dem „Premaster Geheimnis“ ein „Master Geheimnis“.

Der Client sendet eine vom „Change Cipher Spec“ Protokoll initiierte „Change Cipher Spec“-Nachricht und zeigt damit an, daß der nachfolgende Datenverkehr verschlüsselt übertragen wird.

Mit dem „Master Geheimnis“ wird nun in der Record Schicht beider Kommunikationspartner ein Schlüsselblock errechnet, aus dem ein Sitzungsschlüssel entnommen wird. Damit wird der Datenstrom vom Client zum Server durch die Recordschicht verschlüsselt.

Die erste mit dem Sitzungsschlüssel verschlüsselte Nachricht ist die „Finished“-Nachricht vom Client zum Server. Sie enthält eine Prüfsumme über das „Master Geheimnis“ und allen vorher gesendeten Handshake Nachrichten. Dabei werden nur Handshake-Protokollnachrichten berücksichtigt. „Finished“-Nachrichten enthalten keine „Alert“ und „Change Cipher Spec“-Nachrichten.

Nach erfolgreicher Prüfung der „Client Finished“-Nachricht sendet der Server ebenfalls eine vom „Change Cipher Spec Protokoll“ initiierte „Change Cipher Spec“-Nachricht. In der Recordschicht beider Kommunikationspartner wird ein weiterer Sitzungsschlüssel aus dem Schlüsselblock entnommen, mit dem der nachfolgende Datenverkehr vom Server zum Client verschlüsselt wird.

Danach sendet der Server eine „Finished“-Nachricht. Die Prüfsummen in der „Server Finished“-Nachricht beziehen jedoch zusätzlich noch die „Client Finished“-Nachricht mit ein.

Wurde die „Server Finished“-Nachricht vom Client erfolgreich überprüft, beginnt mit Hilfe des Application Data Protokolls der Austausch von verschlüsselten Anwendungsdaten.

Zusammenfassung

Das TLS Protokoll ist die standardisierte Version des SSL Protokolls der Firma Netscape Communication. Es befindet sich zwischen Anwendungs- und Transportschicht und besteht aus zwei Schichten. In der oberen Schicht befinden sich Application Data Protokoll, Handshake Protokoll, Change Cipher Spec Protokoll und Alert Protokoll. In der unteren Schicht befindet sich das Record Protokoll. Kompression, Verschlüsselung und Integritätsprüfung regelt das Record Protokoll. Das Schlüsselmanagement wird in TLS vom TLS Handshake Protokoll übernommen, welches den Dienst der Etablierung von kryptographischen Verfahren und deren Parametern für das TLS Record Protokoll erbringt. Die Authentisierung erfolgt Zertifikatsbasierend. Es wird zwischen anonymer Authentisierung, Serverauthentisierung und Client-Serverauthentisierung unterschieden. Anonyme Authentisierung kann nicht vor „man in the middle“ Angriffen schützen. Bei den anderen Authentisierungsverfahren erfolgt die Prüfung des Zertifikates vom Kommunikationspartner.

Die Übertragung des Schlüsselmaterials (Premaster Geheimnis) vom Client zum Server erfolgt verschlüsselt. Der dafür notwendige Schlüssel wird entweder mit Diffie Hellman ausgehandelt oder mit RSA vom Server zum Client übertragen. Aus dem „Premaster Geheimnis“ wird bei Client und Server ein „Master Geheimnis“ errechnet, aus dem ein Schlüsselblock abgeleitet wird. Client und Server entnehmen jeweils unterschiedliche Schlüssel aus dem Schlüsselblock. Der Datenstrom vom Client zum Server wird somit mit einem anderen Schlüssel ver-und entschlüsselt, als der Datenstrom vom Server zum Client. Das TLS Protokoll bricht nicht mit dem Entwurfsprinzip, daß jede Protokollschicht einen Dienst der darunterliegenden in Anspruch nimmt und der nächst höheren Protokollschicht einen erweiterten Dienst anbietet.

Literaturverzeichnis

[ANSI, 1983] ANSI X3.106: „American National Standard for Information Systems-Data Link Encryption“, American National Standards Institute, 1983

[Frier, 1992] Frier, A. et al: „The SSL 3.0 Protocol“, Netscape Communications Corp., Nov 18, 1996

[Lai, 1992] Lai, X.: „On the Design and Security of Block Ciphers“, ETH Series in Information Processing, v. 1, Konstanz: Hartung-Gorre Verlag, 1992

[DFN-PCA, 1999] DFN-PCA: „Die Policies der DFN-PCA“, Online unter `http://www.pca.dfn.de/dfnpca/policy/`

[NIST, 1994] NIST FIPS PUB 180-1: „Secure Hash Standard“, National Institute of Standards and Technology, U.S. Department of Commerce, Work in Progress, May 31, 1994

[RFC 1321] Rivest R.: „The MD5 Message Digest Algorithm“, April 1992, Online unter `ftp://ftp.ietf.org/rfc/rfc1321.txt`

[RFC 2401] Kent S., Atkinson R.: „Security Architecture for the Internet Protocol“, November 1998, Online unter `ftp://ftp.ietf.org/rfc/rfc2401.txt`

[RFC 2402] Kent S., Atkinson R.: „IP Authentication Header“, November 1998, Online unter `ftp://ftp.ietf.org/rfc/rfc2402.txt`

[RFC 2406] Kent S., Atkinson R.: „IP Encapsulating Security Payload (ESP)“, November 1998, Online unter `ftp://ftp.ietf.org/rfc/rfc2406.txt`

[RFC 2408] Maughan D., Schneider M., Turner J., Schertler M.: „Internet Security and Key Management Protocol (ISAKMP)“, November 1998, Online unter `ftp://ftp.ietf.org/rfc/rfc2408.txt`

[RFC 2409] Hawkins D., Carrel D.: „The Internet Key Exchange (IKE)“, November 1998, Online unter `ftp://ftp.ietf.org/rfc/rfc2409.txt`

[RFC 2412] Orman H.: „The Oakley Key Determination Protocol“, November 1998, Online unter `ftp://ftp.ietf.org/rfc/rfc2412.txt`

[RFC 2422] Karn P., Simpson B.: „Photuris: Session Key Management Protokoll“, März 1999, Online unter `ftp://ftp.ietf.org/rfc/rfc2422.txt`

[RFC 2246] Dierks T., Allen C.: „Transport Layer Security Protocol Version 1.0“, Januar 1999, Online unter `ftp://ftp.ietf.org/rfc/rfc2246.txt`

[RFC 2268] Rivest R., „A Description of the RC2(r) Encryption Algorithm“, Januar 1998, Online unter `ftp://ftp.ietf.org/rfc/rfc2268.txt`

[Rivest, 1978] Rivest, A. et al: “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”, Communications of the ACM, v. 21, n. 2, Feb 1978, pp. 120-126

[Schnei97] Schneier, B.: “Applied Cryptography – Protocols, Algorithms and Source Code in C”, 2. Auflage, John Wiley & Sons 1997

[Tuchman, 1979] Tuchman, W.: “Hellman Presents No Shortcut Solutions To DES”, IEEE Spectrum, v. 16, n. 7, July 1979, pp40-41.

Kapitel 7

„Klassische“ Firewalls in IP-Netzen

Jan Kohlrausch
Universität Hamburg
Fachbereich Informatik

Zusammenfassung

In dem vorliegenden Dokument wird eine Einführung in die grundlegenden Konzepte von Firewalls in IP-Netzen gegeben. Ziel ist es, Elemente und Architekturen von Firewalls zu beschreiben und einen Überblick über die Möglichkeiten und Grenzen von Firewalls zu geben. In dem Artikel werden zwei in der Praxis eingesetzte Firewall-Architekturen als Beispiele vorgestellt.

7.1 Einleitung

Es ist einsichtig, daß lokale Netzwerke, die an das Internet angeschlossen sind, gegen Angriffe aus dem Internet geschützt werden müssen, um die Vertraulichkeit, Integrität und Verfügbarkeit der Daten im lokalen Netzwerk zu gewährleisten. Große lokale Netzwerke können in der Praxis nicht durch Hostsicherheit alleine gesichert werden. Die Gründe dafür sind:

- Jeder Host muß eigenständig gesichert und überwacht werden.
- Es gibt i.a. eine große Anzahl privilegierter Benutzer und heterogene Hard- und Softwaresysteme. Dadurch entstehen diverse Schwachstellen durch fehlerhafte Administration und Softwarefehler.
- Es ist kein effizientes zentrales Auditing des Netzwerkes möglich.
- Es ist aufwendig, Einbrüche zu entdecken und nachzuvollziehen.
- Der administrative Aufwand, um das lokale Netzwerk zu sichern, ist extrem hoch.

Im Gegensatz zu dem Sicherheitskonzept der Hostsicherheit bieten Firewalls die Möglichkeit, ein lokales Netzwerk als Einheit gegen Angriffe von außen abzusichern. Ein Firewall ist ein System zwischen zwei Netzwerken, das

- die einzige Verbindung zwischen den beiden Netzwerken darstellt und
- Zugriffskontrolle und Auditing des Datentranfers zwischen beiden Netzwerken durchführt.

Der „klassische“ IP-Firewall wird zur Zugriffskontrolle zwischen dem Internet und einem lokalen Netzwerk eingesetzt. Firewalls können aber auch sicherheitsrelevantere interne Netzwerke (Kaskade von Firewalls) gegen Angriffe von außen schützen.

Dieser Beitrag gliedert sich in folgende Abschnitte: Im ersten Abschnitt werden die Elemente eines Firewalls (Packet-Screen, Proxy-Server und Bastion) beschrieben. Im zweiten Abschnitt wird gezeigt, wie sich die Elemente von Firewalls in die grundlegenden Architekturen eingliedern. In diesem Abschnitt wird auch die Sicherheit der einzelnen Firewall-Architekturen verglichen. Danach wird anhand von zwei Firewallarchitekturen beschrieben, wie die im letzten Abschnitt beschriebenen Konzepte in der Praxis eingesetzt werden können. Zum Abschluß werden die Grenzen von Firewalls aufgezeigt.

7.2 Elemente eines Firewalls

7.2.1 Packet-Screen

Die Zugriffskontrolle durch eine *Packet-Screen* erfolgt auf der Grundlage der Header-Informationen in den TCP/IP, UDP und ICMP Datagrammen (OSI Schichten 3 und 4, siehe auch [Möller 99]). Filterregeln, die vom Administrator vorgegeben werden, geben an, welche Datagramme die Packet-Screen passieren dürfen oder gesperrt werden. Es gibt zwei verschiedene Filterstrategien:

Gebotsregeln: Alles, was nicht explizit erlaubt ist, ist verboten.

Verbotsregeln: Alles, was nicht explizit verboten ist, ist erlaubt.

Eine größere Sicherheit gewährleistet die erste Filterstrategie. Nur Datagramme, deren Weiterleitung durch eine Gebotsregel explizit erlaubt worden ist, können die Packet-Screen passieren. Fehler bei der Festlegung der Regeln bewirken deshalb im allgemeinen, daß erlaubte Datagramme durch eine fehlerhafte Regel gesperrt werden. Bei der zweiten Filterstrategie bewirkt eine fehlerhafte Verbotsregel, daß Datagramme, die durch diese Regel gesperrt werden sollten, die Packet-Screen passieren können.

Es ist i.a. nicht möglich, alle Schwachstellen im internen Netzwerk zu schließen. Z.B. kann ein Host im internen Netzwerk einen Dienst anbieten, den ein Benutzer ohne Kenntnis des Administrators eingerichtet hat. Werden nur die in den Verbotsregeln angegebenen Dienste gesperrt, kann dieser Dienst aber immer noch von außen angegriffen werden. Da der Administrator nicht weiß, daß dieser Dienst eingerichtet worden ist, kann er diesen Dienst nicht explizit in den Regeln berücksichtigen. Werden allerdings nur die erlaubten Dienste in den Gebotsregeln angegeben, ist der Dienst von außen nicht angreifbar. Ohne Kenntnis des Systemadministrators kann in diesem Fall also kein eventuell angreifbarer Dienst eingerichtet werden, der dem Internet zur Verfügung steht.

Gefiltert wird nach :

1. IP-Quell- und Zieladresse
2. Datagrammtyp (TCP / UDP / ICMP / IP)

3. TCP / UDP-Quell- und Zielportnummer (Dienstkennung)
4. SYN-Flag / ACK-Flag (Regeln beim TCP Protokoll die Richtung des Verbindungsaufbaus)
5. ICMP-Nachrichtentyp
6. IP-Optionen (z.B. „source routing“)

Durch Angabe der IP-Quell- und Zieladresse in den Filterregeln werden die IP-Adressen von einzelnen Hosts oder Subnetzen festgelegt, zwischen denen Datagramme durch die Packet-Screen weitergeleitet werden. Die Integrität der IP-Adressen ist aber nicht hoch; d.h. sie können leicht geändert oder gefälscht werden (*IP Spoofing*). Durch Angabe der Datagrammtypen wird festgelegt, welche Protokolle (TCP, UDP und ICMP) die Packet-Screen passieren können. Beispielsweise verwendet das *Network-File-System (NFS)* in einigen Implementierungen das UDP-Protokoll. Wird das UDP-Protokoll von der Packet-Screen gesperrt, können keine NFS-Zugriffe aus dem externen Netzwerk auf das interne Netzwerk durchgeführt werden. I.a. sind für Serverprogramme von Internetdiensten bestimmte Portnummern zugeordnet worden. Durch diese Filteroption kann erreicht werden, daß nur ausgewählte Dienste (z.B. HTTP- oder SMTP-Server) des internen Netzwerkes von außen erreichbar sind.

Durch das SYN-Flag und das ACK-Flag wird die Richtung des TCP-Verbindungsaufbaus festgelegt. Die Bedeutung dieser TCP-Flags wird in dem weiter unten gezeigten Beispiel verdeutlicht. Es gibt mehrere ICMP-Nachrichten. ICMP-Nachrichten, deren Weiterleitung von der Packet-Screen sinnvoll ist, sind z.B. „Echo-Request“ und „Echo-Reply“ Datagramme. Alle anderen ICMP-Nachrichten sollten gesperrt werden. Durch die letzte Option wird verhindert, daß Datagramme, in denen die IP-Option *„source routing“* verwendet wird, die Packet-Screen passieren können (siehe auch Abschnitt 7.3).

Ein Beispiel für die Aufstellung von Filterregeln ist in Abb. 7.1 dargestellt. Ziel der Regeln ist es, eine Packet-Screen so zu konfigurieren, daß nur Datagramme, die für das SMTP-Protokoll (regelt u.a. das Versenden und Empfangen von E-Mails) notwendig sind, die Packet-Screen passieren können. Die Regeln ‘A’ und ‘B’ lassen eingehende SMTP-Datagramme passieren. Analog dazu bewirken die Regeln ‘C’ und ‘D’, daß ausgehende SMTP-Datagramme die Packet-Screen passieren können. Regel ‘E’ sperrt alle anderen Datagramme, die die Regeln ‘A’ bis ‘D’ nicht erfüllen.

Die ersten zwei Spalten von Abb. 7.1 enthalten die IP-Adressen des externen und inneren Netzwerkes. Durch sie werden die IP-Adressen von den Hosts oder Subnetzen festgelegt, zwischen denen eine TCP-Verbindung durch die Packet-Screen aufgebaut werden kann. Die folgenden beiden Spalten enthalten die Portnummern des SMTP-Servers (25) und des SMTP-Clients (> 1023). Die Portnummer des SMTP-Clients ist für Werte größer als 1023 beliebig wählbar. Durch die Werte in der fünften Spalte kann die Richtung des Aufbaus einer TCP-Verbindung kontrolliert werden. Ein „-“ Zeichen in dieser Spalte bedeutet, daß Datagramme, in denen das SYN-Flag gesetzt und das ACK-Flag nicht gesetzt sind (diese Datagramme kennzeichnen den Aufbau einer TCP-Verbindung), gesperrt werden. Die Werte in dieser Spalte bewirken, daß Serverprogramme mit Portnummern größer 1023 nicht von Clients mit der privilegierten Portnummer 25 angegriffen werden können. Da bei den „Windows“ Betriebssystemen von Microsoft nicht zwischen privilegierten und nicht privilegierten Portnummern unterschieden wird, ist es i.a. kein Problem, einen Client mit privilegierter Portnummer einzurichten. Die folgende Spalte besagt, daß die Regeln ausschließlich TCP-Verbindungen betreffen. In der letzten Spalte wird schließlich festgelegt, ob die Regel eine Verbots- oder Gebotsregel ist.

R	Quelladr.	Zieladr.	Quellp.	Zielp.	SYN	Prot.	Aktion
A	Extern	Intern	> 1023	25		TCP	Permit
B	Intern	Extern	25	> 1023	-	TCP	Permit
C	Intern	Extern	> 1023	25		TCP	Permit
D	Extern	Intern	25	> 1023	-	TCP	Permit
E	Beliebig	Beliebig	Alle	Alle		Alle	Deny

Abbildung 7.1: Beispiel für Filterregeln.

Vor- und Nachteile

Der Vorteil einer Packet-Screen ist die Transparenz. Werden keine verbotenen Datagramme versendet, bleibt die Funktion der Packet-Screen für die Benutzer vollständig verborgen. Da die Funktionalität einer Packet-Screen in vielen Screening-Routern bereits vorhanden ist, ist der Installationsaufwand i.a. gering.

Nachteil einer Packet-Screen ist, daß die Möglichkeiten der Zugriffskontrolle gering sind. Es kann keine vom Benutzer oder von der Verwendung des Dienstes (z.B. Inhalt einer URL) abhängige Zugriffskontrolle durchgeführt werden. Einige Protokolle sind für Packet-Screens ungeeignet (z.B. *Remote procedure calls (RPC)* wegen variabler Portnummern). Ein weiterer Nachteil ist die unzureichende Integrität der Portnummern und IP-Adressen. Sie können leicht gefälscht werden (*IP Spoofing*). In der Praxis ist es aufwendig, die Funktion der Packet-Screen für eine große Anzahl von Filterregeln zu überprüfen.

7.2.2 Proxy-Server

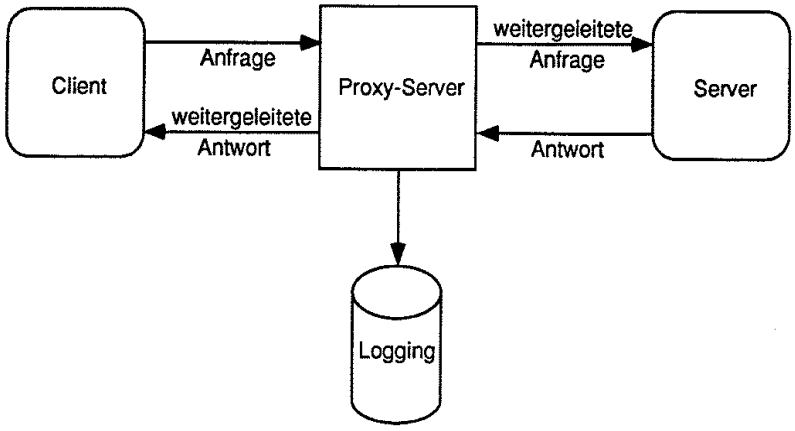


Abbildung 7.2: Funktionsweise eines Proxy-Servers (Aus [Ellermann 94]).

Ein *Proxy-Server* ist ein Prozeß auf Anwendungsebene, der eine fein granulierte Zugriffskontrolle und Auditing auf Anwendungsebene (OSI Schichten 5-7) durchführt. Er stellt Anfragen anstelle eines

Clients an einen Server und leitet die Antworten des Servers an den Client weiter. Die Funktionsweise eines Proxy-Servers ist in Abb. 7.2 dargestellt.

Vor- und Nachteile

Der Vorteil eines Proxy-Servers ist die detaillierte Zugriffskontrolle. Es können z.B. der Benutzername, der Dienst (z.B. FTP Befehle *put* und *get*), zeitliche Einschränkungen von Diensten und Benutzeraktivitäten und der Kontext des Zugriffes (z.B. FTP: Dateilänge) berücksichtigt werden. Durch die feine Granularität der Zugriffskontrolle ist ein aussagekräftiges Auditing möglich. Beim Verbindungsaufbau kann die Authentizität des Benutzers geprüft werden.

Nachteil eines Proxy-Servers ist die fehlende Transparenz. Alle Client-Programme müssen an den Proxy angepaßt werden, oder es wird eine Kooperation der Benutzer verlangt. Die Zugriffskontrolle auf Anwendungsebene erfordert eine höhere Rechenleistung als die Überprüfung der Protokoll-Header in einer Packet-Screen. I.a. ist der Installationsaufwand eines Proxy-Servers größer als bei einer Packet-Screen. Für einige wichtige Dienste (z.B. Telnet und FTP) sind aber Proxy-Server und Client-Programme bereits frei vorhanden (z.B. „SOCKS“ und „TIS Internet Firewall Toolkit for Proxying“, siehe [Chapman et al. 95]).

7.2.3 Bastion

Eine *Bastion* ist ein zum Firewall gehörender Host, der als einziger Computer aus dem (äußeren) Netzwerk erreichbar ist. Da die Bastion der einzige von außen angreifbare Rechner ist, ist höchste Hostsicherheit erforderlich. Die Softwareausstattung sollte deshalb so einfach wie möglich gehalten werden (*Least privilege*). Zusätzlich ist die Erstellung eines Notfallplans für die Bastion wichtig, um im Fall eines erfolgreichen Angriffs die Schäden so gering wie möglich zu halten.

Die Funktion einer Bastion ist das Erbringen von Internet Diensten (z.B. *anonymous FTP*) und das Weiterleiten von Internet Diensten. Benutzer-Accounts dürfen wegen der deutlichen Reduzierung der Hostsicherheit nicht eingerichtet werden. Eine höhere Sicherheit bietet die Installation von Proxy-Servern (z.B. Telnet und FTP) auf der Bastion bzw. die Weiterleitung von Protokollen (z.B. SMTP).

Die Bastion darf nicht die Funktionalität eines Routers erfüllen. Deshalb muß das *„IP-Forwarding“* abschaltet werden und die *„IP-Source routing“* Option muß unterbunden werden. Mit der *IP-Source routing* Option kann die Route eines IP-Datagramms unabhängig von Routingtabellen festgelegt werden. Dadurch kann die Weiterleitung von Datagrammen durch die Bastion erzwungen werden. Die Sicherheit des Netzes wird durch ein umfangreiches Auditing unterstützt. Deshalb sollte ein zentrales Auditing auf der Bastion (z.B. Überwachung der Proxy-Server) eingerichtet werden.

7.3 Firewall-Architekturen

In diesem Abschnitt werden Firewall-Architekturen vorgestellt und es werden die Vor- und Nachteile der Architekturen beschrieben.

7.3.1 Packet-Screen (Screening-Router)

Der einfachste Firewall ist durch eine Packet-Screen gegeben (Abb. 7.3), die zwischen dem Internet und dem lokalen Netzwerk eingesetzt wird. Die Zugriffskontrolle durch die Packet-Screen übernimmt

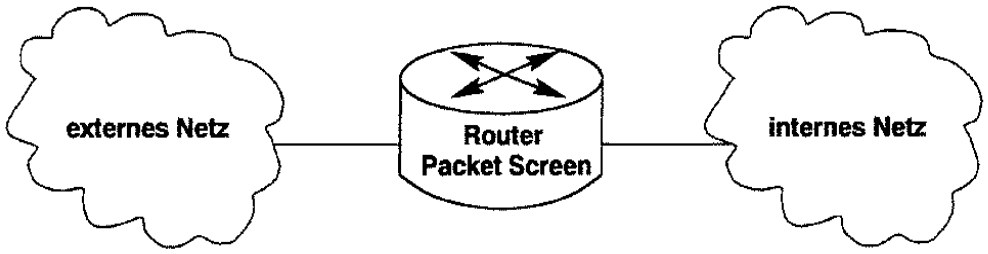


Abbildung 7.3: Packet-Screen Firewall: konzeptionelle Sicht (Aus [Benecke et al. 98]).

i.a. ein Screening-Router. Die Vor- und Nachteile dieser Architektur sind durch die Vor- und Nachteile der Packet-Screen gegeben.

Vor- und Nachteile

Der Vorteil dieser Architektur ist die einfache Installation. Es müssen i.a. keine weiteren Hard- und Softwarekomponenten eingesetzt werden.

Nachteil dieser Firewall-Architektur ist, daß die Sicherheit allein vom Router abhängt. Gelingt es dem Angreifer, die Screening Funktion des Routers abzuschalten, ist das interne Netz ungeschützt. Ein weiterer Nachteil dieser Architektur ist die grobe Zugriffskontrolle des Screening-Routers und die begrenzten Möglichkeiten des Auditings. Beschränkt sich die Zugriffskontrolle auf Internet Protokolle, gibt es eine hohe Anzahl von angreifbaren Hosts in dem internen Netzwerk. Die Sicherheit des internen Netzwerkes hängt dann wesentlich von der Sicherheit der internen Hosts ab.

Bewertung

Die Sicherheit beim ausschließlichen Einsatz einer Packet-Screen ist gering.

7.3.2 Gateway Firewall

Abbildung 7.4: Gateway Firewall: konzeptionelle Sicht (Aus [Benecke et al. 98]).

Eine weitere Firewall-Architektur ist der Gateway Firewall (Abb. 7.4). In dieser Architektur wird die Packet-Screen in der vorherigen Architektur durch eine Bastion ersetzt. Die Bastion stellt die einzige Verbindung zwischen den beiden Netzwerken dar und übernimmt die Funktion des Gateways zwischen den Netzwerken. Die Bastion wird technisch durch einen *dual homed* Host realisiert.

Da auf der Bastion nicht die Funktionen eines Routers vorhanden sind, gibt es keine direkte Verbindung zwischen den beiden Netzwerken. Die Zugriffskontrolle erfolgt i.a. über Proxy-Server (siehe Abschnitt 7.2.2), die auf der Bastion installiert sind. Es ist auch möglich, auf dem Gateway eine Packet-Screen zu installieren.

Vor- und Nachteile

Vorteil dieser Architektur ist die umfangreiche Zugriffskontrolle, die durch den Einsatz von Proxy-Servern gegeben ist. Da das Auditing auf dem Gateway sehr umfangreich gestaltet werden kann, können Angriffe auf das Gateway und das interne Netzwerk gut nachvollzogen werden. Für diese Firewall-Architektur sind viele kommerzielle Lösungen erhältlich, so daß diese Produkte in der Praxis eingesetzt werden können.

Ein Nachteil ist der hohe Installationsaufwand, den die Proxy-Server mit sich bringen, wenn für die gewünschte Anwendung kein Proxy vorhanden ist oder dessen Funktionalität nicht ausreicht. Die Firewall-Architektur bietet keine Erweiterungsmöglichkeiten. Reicht die Performanz des Gateways nicht mehr aus, muß das Gateway durch einen anderen Rechner ersetzt werden.

Bewertung

Die Sicherheit der Gateway Firewall-Architektur ist durch die umfangreichere Zugriffskontrolle und die besseren Möglichkeiten des Auditings höher, als bei dem alleinigen Einsatz einer Packet-Screen. Bestehende Schwachstellen des Gateway können durch das Auswerten der Informationen des Auditings erkannt und beseitigt werden.

7.3.3 Kombination von Packet-Screen und Bastion

Im folgenden werden die Firewall-Architekturen beschrieben, die sich aus der Kombination von Packet-Screens und Bastions-Rechnern ergeben (Abb. 7.5 bis 7.7).

Bastion innen

In Abschnitt 7.3.1 sind die Nachteile des alleinigen Betriebs einer Packet-Screen beschrieben worden. Diese Nachteile werden durch die Kombination einer Packet-Screen und einer Bastion vermieden. Die Packet-Screen wird so konfiguriert, daß die Bastion als einziger Host im internen Netzwerk von außen erreicht werden kann. Dadurch wird die Anzahl der angreifbaren Hosts auf die Bastion beschränkt. Die Zugriffskontrolle erfolgt i.a. durch die Packet-Screen und Proxy-Server, die auf der Bastion installiert sind. Um erfolgreiche Angriffe auf die Packet-Screen zu erkennen, durch die der Schutz des internen Netzwerkes aufgehoben wird, ist ein detailliertes Auditing auf der Bastion erforderlich.

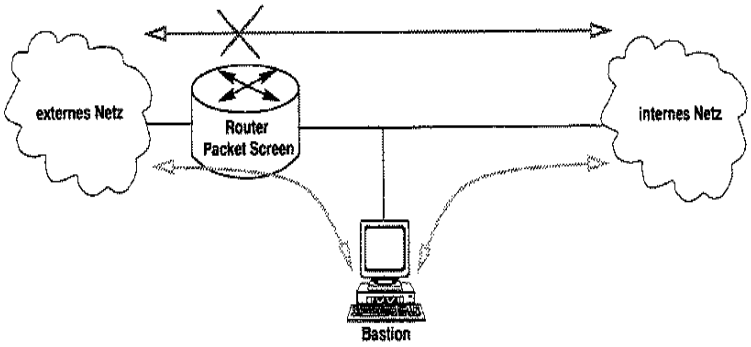


Abbildung 7.5: Bastion innen: konzeptionelle Sicht (Aus [Benecke et al. 98]).

Vor- und Nachteile

Die Bastion im inneren Netzwerk ist gegen Angriffe aus dem externen Netz durch einen Screening-Router geschützt. Im Gegensatz zu der Gateway-Architektur ist diese Architektur durch den Einsatz von mehreren Bastions-Rechner erweiterbar, auf denen dann Proxy-Server für verschiedene Dienste installiert werden können.

Nachteil dieser Architektur gegenüber dem Gateway sind zusätzliche Kosten durch den Screening Router. Der Netzwerkverkehr des inneren Netzes kann mitgehört werden (*sniffing*), falls es dem Angreifer gelingt, die Bastion zu kompromittieren. Dadurch ist die Sicherheit des inneren Netzes deutlich reduziert, weil z.B. unverschlüsselte Passworte mitgehört werden können.

Bastion außen

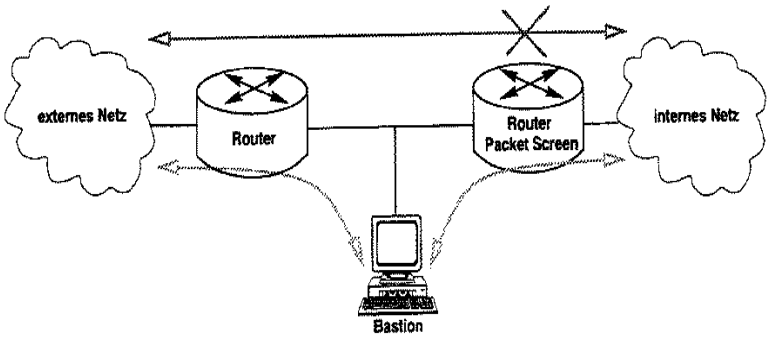


Abbildung 7.6: Bastion außen: konzeptionelle Sicht (Aus [Benecke et al. 98]).

Ist die Bastion im inneren Netzwerk, kann der Netzwerkverkehr des internen Netzes mitgehört werden (*sniffing*), falls es dem Angreifer gelingt, die Bastion zu kompromittieren. Dieser Nachteil wird vermieden, wenn die Bastion von dem inneren Netz durch einen Screening-Router abgeschirmt wird. Die Bastion ist dann in einem vom inneren Netzwerk unabhängigen Subnetz untergebracht. Die Funktion der Bastion ist analog zu der Funktion der Bastion, die im vorherigen Abschnitt beschrieben wurde.

Vor- und Nachteile

Vorteil ist, daß Angriffe auf die Bastion besser entdeckt werden können, weil der Schutz des äußeren Screening-Routers entfällt. Die Bastion ist von dem internen Netzwerk abgeschirmt. Deshalb kann der Netzwerkverkehr des internen Netzes nicht mitgehört werden (*sniffing*), falls es dem Angreifer gelingt, die Bastion zu kompromittieren.

Ein Nachteil ist der fehlende Schutz der Bastion durch die äußere Packet-Screen. Dieser Nachteil ist vorher auch als Vorteil aufgezählt worden. Man muß also im konkreten Fall entscheiden, ob der zusätzliche Schutz der Bastion oder das Erkennen von Angriffsversuchen auf die Bastion wichtiger ist.

Bastion mittig

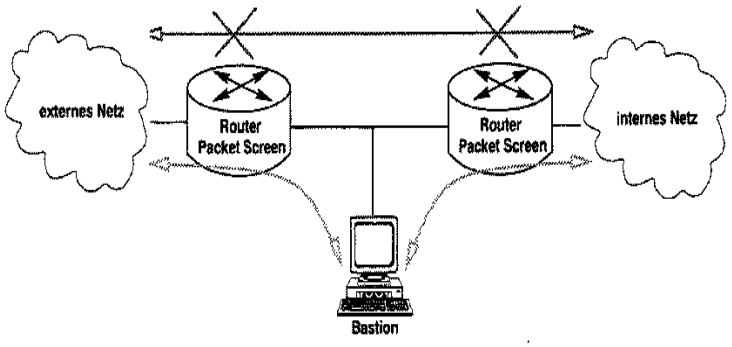


Abbildung 7.7: Bastion mittig: konzeptionelle Sicht (Aus [Benecke et al. 98]).

In dieser Architektur wird die Bastion vom inneren und äußeren Netzwerk durch Screening-Router abgetrennt.

Vor- und Nachteile

Die Vor- und Nachteile dieser Architektur sind im wesentlichen analog zu den Vor- und Nachteilen der Firewall-Architektur mit äußerer Bastion. Ein weiterer Vorteil ist allerdings der zusätzliche Schutz der Bastion durch die äußere Packet-Screen. Dadurch können aber Angriffsversuche auf die Bastion schlechter erkannt werden.

Bewertung

Die Sicherheit dieser Firewall-Architekturen ist durch die umfangreiche Zugriffskontrolle und die guten Möglichkeiten des Auditings hoch. Der innere Screening-Router erhöht die Sicherheit des Firewalls deutlich.

In [Chapman et al. 95, S. 66] wird behauptet, daß die Sicherheit der Firewall-Architektur mit innerem Screening-Router höher ist als die Sicherheit der Architektur eines Gateway-Firewalls.

7.4 Firewall-Architekturen in der Praxis

In diesem Abschnitt werden zwei Firewall-Architekturen beschrieben, die bei AT&T und DEC eingesetzt werden. Die Beispiele sind aus [Ellermann 94] entnommen. Anhand der beiden Beispiele soll gezeigt werden, wie die in den vorherigen Abschnitten vorgestellten Konzepte für Firewall-Architekturen in der Praxis kombiniert werden können. Die beiden Firewall-Architekturen wurden ausgewählt, weil beide Architekturen interessante Konzepte enthalten und auf unterschiedlichen Policies beruhen (Policies werden ausführlich in [Großklaus 99] behandelt).

7.4.1 AT&T Firewall

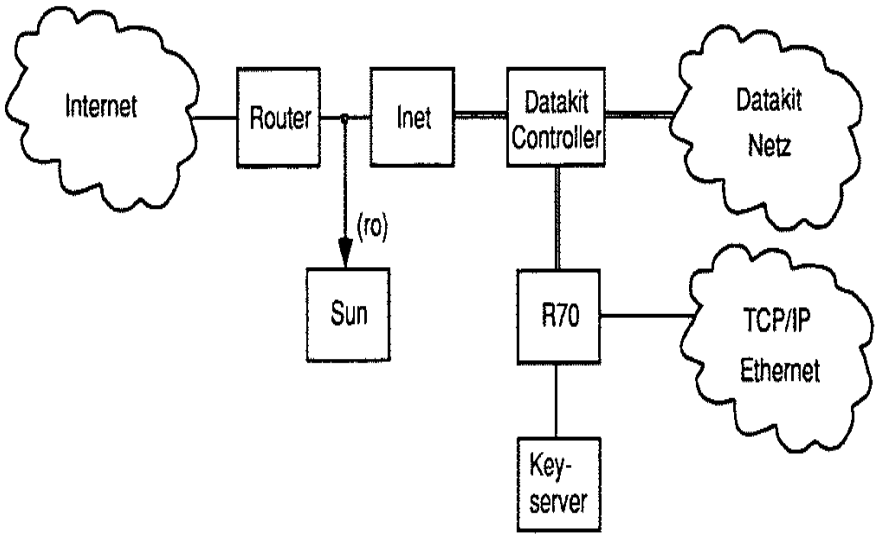


Abbildung 7.8: AT&T Firewall (Aus [Ellermann 94])

Der AT&T Firewall ist in Abb. 7.8 dargestellt. Die Zugriffskontrolle erfolgt durch die beiden Gateways *Inet* und *R70*. Der AT&T Firewall besitzt also eine Gateway-Architektur, wobei das Gateway aus Abb. 7.4 in zwei Komponenten aufgeteilt worden ist.

Der Host *Inet* ist ein nicht vertrauenswürdiges Gateway, dessen Aufgabe die Annahme von Verbindungsanfragen aus dem Internet und der Aufbau von Verbindungen aus dem inneren TCP/IP Netzwerk sind. Obwohl der Host vertrauensunwürdig ist, wird eine hohe Hostsicherheit gefordert. Die hohe Hostsicherheit wird erreicht, indem keine Benutzer-Accounts eingerichtet werden und das Programm *sendmail* wurde durch das sicherere *upas* ersetzt. Auf dem Host *Inet* sind Proxy-Server für FTP und Telnet und ein ‘anonymous FTP-Server’ eingerichtet worden.

Die zweite Komponente des Gateways stellt der als vertrauenswürdig eingestufte Host *R70* dar. Er ist der einzige Computer im internen Netzwerk, den *Inet* direkt erreichen kann. Die Aufgabe von *R70* ist die Authentisierung von Verbindungen, die aus dem externen Netzwerk aufgebaut werden. Ein Host aus dem externen Netzwerk, der eine Verbindung zu einem Computer aus dem internen Netzwerk aufbauen will, stellt zuerst eine Anfrage an das Gateway *Inet*. *Inet* leitet die Anfrage an das Gateway *R70* weiter. Das Gateway *R70* führt eine Benutzerauthentisierung mittels eines „Challenge/Response“ Verfahrens durch und leitet danach im Erfolgsfall die Verbindung an den internen Host weiter. Die

kryptographischen Schlüssel für die Benutzerauthentisierung sind auf dem Host *Keyserver* gespeichert.

Das Auditing erfolgt durch den Computer *Sun*, der mit dem Gateway über eine 'read-only' Leitung verbunden ist. Es ist also nicht möglich, die Daten des Auditings von außen zu verändern.

7.4.2 DEC Firewall

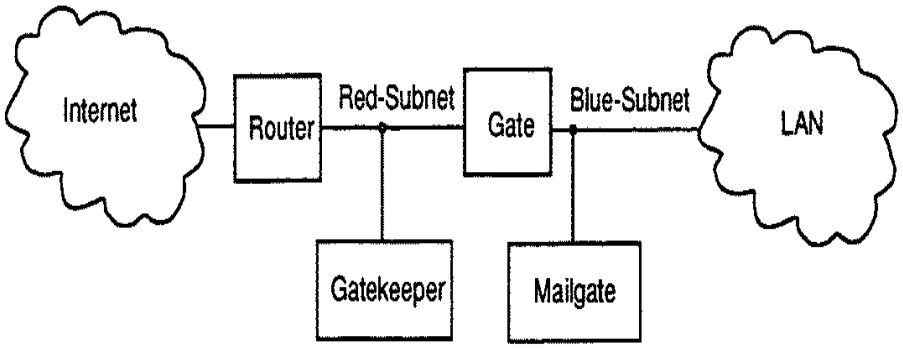


Abbildung 7.9: DEC Firewall (Aus [Ellermann 94])

Die Architektur des DEC Firewalls ist in Abb. 7.9 dargestellt. Die wichtigsten Komponenten sind die Packet-Screen *Gate* und die Bastion *Gatekeeper*. Der Router enthält keine Filterfunktionen. Die Architektur entspricht der Kombination von Packet-Screen und äußerer Bastion aus Abschnitt 7.4.3.

Gate ist ein Unix Host, auf dem *screend* läuft und der als Screenig-Router verwendet wird. Das Routing ist so konfiguriert, daß die gesamte Kommunikation zwischen Internet und internem Netzwerk über die Bastion *Gatekeeper* geführt wird. Es besteht keine Protokollbeschränkung. Routingprotokolle (z.B. *RIP*) werden in dem Host *Gate* nicht unterstützt. Dadurch sollen Angriffe, die die Routing Tabelle ändern, abgewehrt werden.

Die Bastion *Gatekeeper* wird durch die Packet Screen *Gate* abgeschirmt (Architektur: Bastion außen). Auf dem Host sind Proxy-Server („*Application Gateways*“) für Telnet und FTP installiert. Wie bei dem AT&T Firewall wird für die Benutzerauthentisierung ein „Challenge/Response“ Verfahren verwendet. Um das Ausschleusen geheimer Dokumente aus dem inneren Netzwerk nach außen zu verhindern, werden verschiedene Maßnahmen getroffen. Bei dem FTP-Protokoll ist nur Datenimport möglich. Für das Telnet-Protokoll ist eine Beschränkung des Datenflusses nach außen hin auf 1200 Baud eingeführt worden in der Hoffnung, daß das Versenden von geheimen Dokumenten rechtzeitig bemerkt wird. Für die Sicherung der Bastion sind verschiedene Abwehrmaßnahmen gegenüber Angriffen von außen eingerichtet worden. So werden z.B. falsche Paßwortdateien zurückgeliefert und es sind „*Sucker Traps*“ eingerichtet worden. „*Sucker Traps*“ werden anstelle von Internetdiensten installiert. Sie simulieren diesen Dienst und lassen den Angreifer in dem Glauben, daß er durch eine Schwachstelle des Dienstes in das System eingebrochen ist. Zugleich wird automatisch versucht, den potentiellen Angreifer zu identifizieren (z.B. durch eine *finger* Abfrage). Zusätzlich werden die Aktionen des Angreifers protokolliert.

7.4.3 Vergleich der Firewall-Architekturen

Während der AT&T Firewall auf der Gateway-Architektur (Abschnitt) basiert, besteht der DEC Firewall aus der Kombination einer Packet-Screen und einer Bastion. Bei beiden Firewalls wird eine Benutzerauthentisierung mittels eines Challenge/Response Verfahrens durchgeführt. Diese Tatsache unterstreicht die Bedeutung von kryptographischen Verfahren für den Betrieb von Firewalls.

Die Aufgaben der beiden Firewall-Architekturen unterscheiden sich. Während der AT&T Firewall in erster Linie das innere Netzwerk gegen Bedrohungen aus dem äußerern Netzwerk sichern soll, richteten sich die Sicherheitskonzepte des DEC Firewalls auch gegen Mißbrauch ausgehend von dem inneren Netzwerk. Durch Beschränkung des Datenexports soll erreicht werden, daß keine geheimen Dokumente aus dem inneren Netzwerk in das externe Netzwerk gelangen können.

7.5 Grenzen der Sicherheit durch Firewalls

In diesem Abschnitt werden die Sicherheitsgrenzen der Firewalls beschrieben. Da ein Firewall keine Zugriffskontrolle im inneren Netzwerk durchführt, besteht kein Schutz des inneren Netzwerkes gegenüber Angriffen aus dem inneren Netzwerk. Gerade durch diese Angriffe ist aber ein großes Risiko gegeben, weil der Angreifer aus dem inneren Netzwerk die Schwächen dieses Netzwerkes i.a. besser kennt als ein Angreifer von außen. Zudem weiß er, wo und in welcher Form die unternehmenskritischen oder wertvollen Daten gespeichert sind.

Ein Firewall bietet alleine keinen Schutz der einzelnen Kommunikationsbeziehung. Die Portnummern bzw. die IP-Adressen können leicht gefälscht werden. Ein Angreifer kann dadurch eine TCP-Verbindung übernehmen (*TCP-Hijacking*) oder Daten verändern, die während der Verbindung übertragen werden. Um diesen Mißbrauch zu verhindern, ist der Einsatz kryptographischer Protokolle notwendig.

Durch die größere Bandbreite in Hochgeschwindigkeitsnetzen oder durch aufwendige Zugriffskontrollen können Performanzprobleme auftreten. Dadurch können die Benutzer u.U. erzwingen, daß die Sicherheit des Netzwerkes reduziert wird, um den ordnungsgemäßen Betrieb des Netzwerkes aufrecht zu halten.

7.6 Zusammenfassung

- Ein Firewall ist ein System zwischen zwei Netzwerken, das die einzige Verbindung zwischen den Netzen darstellt und eine Zugriffskontrolle und Auditing durchführt.
- Ein Firewall bietet effektive Methoden, ein Netzwerk gegenüber Angriffen von außen zu schützen, ohne daß jeder interne Host aufwendig geschützt werden muß. Trotzdem müssen die Hostrechner (z.B. Bastion) des Firewalls aufwendig gesichert werden, weil sie nach wie vor angreifbar sind.
- Ein Firewall kann dazu beitragen, die Risiken des Datenmißbrauchs ausgehend von dem inneren Netzwerk abzuschwächen (z.B. durch Zugriffskontrolle des Datenexportes).
- Durch Einführung eines Firewalls ist ein sehr umfangreiches Auditing des Netzwerkverkehrs zwischen den beiden Netzen möglich.

- Firewalls bieten einen zusätzlichen Schutz gegen Fehler bei der Administration von internen Hosts und gegen Softwarefehler von internen Hosts. Die Schwachstellen des internen Netzwerkes können durch den Firewall z.T. verborgen werden.
- Firewalls und Kryptographie sind z.T. orthogonale Verfahren. Beide Verfahren ergänzen sich sehr gut und sollten parallel eingesetzt werden.

Literaturverzeichnis

[Benecke et al. 98] Benecke, Carsten & Ellermann,Uwe: „Nutzung von Kryptographie im Zusammenhang mit Firewalls“, Studie des DFN-FWL, DFN Bericht Nr. 86, Novenber 98

[Chapman et al. 95] Chapman, D. Brent & Zwicky, Elizabeth: „Building Internet Firewalls“, O’Reilly, 1995

[Ellermann 94] Ellermann,Uwe: „Firewalls, Isolations- und Audittechniken zum Schutz von lokalen Computer-Netzen“, DFN Bericht Nr. 76, September 94

[Großklaus 99] Großklaus, A.: „Policy, Vorfallsbearbeitung, Schwachstellenanalyse“. Seminar 18.416: „Sicherheit in vernetzten Systemen“, Universität Hamburg, FB Informatik, SS 99

[Möller 99] Möller, K.: „Grundlagen: Internet-Protokolle“. Seminar 18.416: „Sicherheit in vernetzten Systemen“, Universität Hamburg, FB Informatik, SS 99

Kapitel 8

Firewalls in Hochgeschwindigkeitsnetzen am Beispiel von ATM

Jens Nedon
ConSecur GmbH
Schulze-Delitzsch-Straße 2, 49716 Meppen
nedon@consecur.de

Zusammenfassung

Firewalls bieten die Möglichkeit, den Datenverkehr zwischen Netzen zu kontrollieren und zu blockieren. Die Performanz heutiger Firewalls stößt jedoch an Grenzen, wo ein Datendurchsatz jenseits des herkömmlichen Ethernet erzielt werden soll.

In diesem Artikel sollen diese Grenzen aufgezeigt und Firewallarchitekturen dargestellt werden, die in Hochgeschwindigkeitsnetzen zum Einsatz kommen können. Dies wird am Beispiel der ATM-Technologie erläutert.

8.1 Einleitung

Firewalls stellen eine Möglichkeit dar, an zentraler Stelle den Datenverkehr zwischen Netzwerken zu kontrollieren und ggf. zu blockieren und so die Sicherheitspolitik eines Unternehmens auch im Netzverkehr umzusetzen. Mit dem Trend zu immer schnelleren Netzen geht jedoch das Problem einher, daß heutige Firewalls einen Engpaß darstellen, der die Vorteile eines schnellen Netzes zunichte macht. Zwei Beispiele sollen das erläutern:

Während in Netzen heute gewöhnlicherweise Technologien zum Einsatz kommen, die einen Datendurchsatz von 10–100 Mbit/s (z.B. Ethernet) erlauben, wird in den nächsten 5 Jahren der Einsatz von Technologien erwartet, die einen Durchsatz im Bereich von 1 Gbit/s bis zu 100 Tbit/s erlauben. Die Technik heutiger Firewalls stellt jedoch bereits bei einem Durchsatz von 100 Mbit/s einen Engpaß dar.

Eine zusätzliche Grenze heutiger Firewalls ist die geringe Skalierbarkeit. Tritt ein Engpaß in der Verarbeitungskapazität des Firewalls auf, so kann versucht werden, den Engpaß durch Aufteilung der

Funktionalität auf mehrere Rechnerkomponenten zu verteilen. Derzeit ist das nur durch Aufteilung der Firewallbastionen möglich, der Paketfilter bleibt ein Engpaß.

Da künftig mit erweiterten Aufgaben an der Firewall zu rechnen ist, z.B. der kryptographischen Absicherung von Verbindungen oder auch der Kontrolle von Verbindungsinhalten (Virenfilter, URL-Filter, Video-/Audio-Filter), kann davon ausgegangen werden, daß die heutige Firewalltechnologie in Zukunft nicht mehr ausreichend dimensionierte Netzverbindungen zuläßt.

ATM (Asynchronous Transfer Mode) bietet hier einen Ansatz, neben einem vergrößerten Durchsatz auch eine bessere Skalierbarkeit der Firewalls zu ermöglichen.

In den folgenden Abschnitten soll ein solcher Ansatz am Beispiel des Hochgeschwindigkeitsnetzes ATM untersucht werden. Hierzu wird zunächst die ATM-Techologie kurz vorgestellt. Ein Mechanismus, der zur Kontrolle der Verbindungen in ATM geeignet ist, wird dann im Abschnitt 8.4 vorgestellt. Die Möglichkeiten für den Einsatz dieser Kontrollmechanismen werden hier nur kurz angesprochen und dann im zweiten Teil dieser Gemeinschaftsarbeit genauer untersucht.

8.2 Die ATM-Technologie

ATM (Asynchronous Transfer Mode) ist eine Technologie für Hochgeschwindigkeitsnetze. Mit bereits erreichbaren Geschwindigkeiten von 622 Mbit/s (bidirektional) wird es im Bereich von Backbones, Video- und Multimediaanwendungen eingesetzt.

ATM sieht die Definition von Dienstgüteparametern (Quality of Service) für Verbindungen vor. Damit können Anforderungen und Garantien z.B. zum Mindestdurchsatz umgesetzt werden. Im Gegensatz zu IP ist ATM ein verbindungsorientiertes Protokoll. Eine mögliche Kontrolle der Verbindung wird demzufolge nicht paketweise erfolgen, sondern zum Zeitpunkt des Verbindungsaufbaus.

Ein praktischer Vorteil von ATM ist die universelle Einsetzbarkeit und Skalierbarkeit, die den Einsatz sowohl im WAN- als auch im LAN-Bereich ermöglicht. Beim Übergang zwischen Netzwerken wäre dann kein Technologiewechsel mehr erforderlich.

8.2.1 ATM-Switches

Die Knoten in ATM-Netzwerken werden als Switches bezeichnet. Beim Verbindungsaufbau wird zwischen den Kommunikationspartnern eine virtuelle Verbindung aufgebaut, die von ATM-Switch zu ATM-Switch durchgeschaltet wird. Auf diese Weise entsteht ein Pfad zum Zielrechner, der dann für die Dauer der Kommunikation geschaltet ist.

Abb. 8.1 zeigt ein typisches Blockschaltbild eines ATM-Switches.

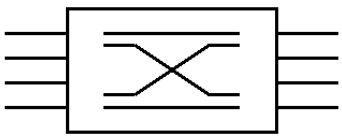


Abbildung 8.1: ATM-Switch, Blockschaltbild

8.2.2 ATM-Zellen

Über diese Verbindung werden Datenpakete ausgetauscht, die in ATM-Terminologie als Zellen bezeichnet werden. Diese Datenpakete haben eine feste Größe von 53 Byte (5 Byte Protokolldaten sowie 48 Byte Nutzdaten). Hieraus lassen sich einige wesentliche Vorteile gegenüber Technologien mit variabler Paketlänge ableiten:

- Durch die Festlegung der Zellgröße lassen sich genaue Vorhersagen über Last, Verzögerung und Übertragungsdauer treffen, was zu einer Lastverteilung genutzt werden kann. Das Aushandeln von Dienstgüteparametern kann daher bereits beim Verbindungsaufbau erfolgen.
- Durch die feste Zellgröße sind einfachere Algorithmen zum Switchen der Zellen in Hardware implementierbar. Hierdurch kann ein sehr hoher Durchsatz durch Switches erreicht werden, was die potentiell hohen Übertragungsgeschwindigkeiten über ATM-Verbindungen erst ermöglicht.

Der 5-Byte-große Header einer ATM-Zelle für die Datenübertragung zwischen Switch und Endgerät an der „User-Network“-Schnittstelle (UNI) setzt sich aus den folgenden Informationen zusammen (Abb. 8.2):

1. Die ersten 4 Bit Daten werden zur Flußkontrolle und zur Behebung von kurzzeitiger Überlast verwendet.
2. Die folgenden 8 Bit kennzeichnen einen sog. virtuellen Pfad (VPI, Virtual Path Identifier),
3. die nächsten 16 Bit kennzeichnen einen virtuellen Kanal (VCI, Virtual Channel Identifier); der Switch führt das Mapping von Eingangsschnittstelle zur Ausgangsschnittstelle aufgrund der VPI/VCI-Informationen für jede einzelne Zelle durch.
4. Die dann folgenden 12 Bit bestehen aus einer Kennzeichnung der Nutzdaten (PT, Payload Type) sowie Informationen zur Fehlererkennung und -kontrolle.

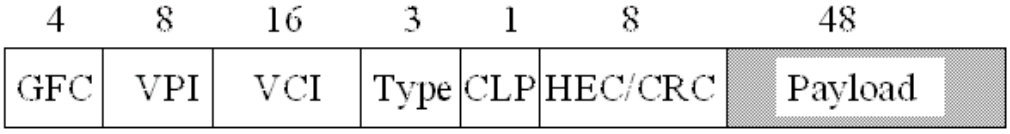


Abbildung 8.2: ATM-Zelle (UNI-Schnittstelle)

8.2.3 Virtuelle Kanäle und Pfade

Nachdem eine Verbindung aufgebaut wurde, also das Routing vollzogen und alle zwischen den Endpunkten liegenden ATM-Switches die Verbindung durchgeschaltet haben, werden ATM-Zellen über diese Verbindung transportiert. Die Zellen werden vom ATM-Switch anhand ihrer VPI/VCI-Kennung am Eingang des Switches identifiziert und auf dem beim Verbindungsaufbau durch das Routing ermittelten Ausgang des Switches weitergeleitet. Der Switch aktualisiert dabei jeweils die VPI/VCI-Informationen in jeder Zelle. VPI und VCI haben somit nur lokale Bedeutung zwischen zwei benachbarten Switches bzw. Endgeräten. Die Zelle wandert auf diese Weise von Switch zu Switch und

schließlich zum Empfänger, dessen ATM-Adresse nur einmal beim Verbindungsaufbau angegeben wurde.

Da die VCI-Kennung nur zwischen jeweils zwei Switches festgelegt ist, kann sich die VCI während des Transports ändern. Es können aber auch Virtuelle Kanäle zwischen zwei Switches in sog. virtuellen Pfaden zusammengefaßt werden. Alle Verbindungen zwischen zwei Switches haben dann denselben VPI. Dies ermöglicht den Einsatz von einfacheren „Cross Connect"-Switches, bei denen lediglich die VPIs umgesetzt werden.

Virtuelle Kanäle und Pfade sind die Basis für virtuelle Verbindungen. Da ATM ein verbindungsorientiertes Protokoll ist, wird das Routing und das Aushandeln von Dienstparametern vor dem Transport der ersten ATM-Datenzelle ausgeführt. Dazu gibt es auch in ATM verschiedene Protokollelemente, z.B. zum Verbindungsaufbau. In den Datenzellen selbst werden dann keine Informationen über die Zieladresse der Zelle mehr mitgeführt, sondern nur die eben dargestellten VPI/VCI-Informationen.

8.3 Die Einbindung von ATM in bestehende Netze

Da ATM eine neue Netzwerktechnologie ist, ist sie zunächst nicht mit der derzeitigen Technologie von Internet und Intranets – in der Regel Ethernet-Technologie – kompatibel; sie kann nur dann wirtschaftlich in diesem Kontext eingesetzt werden, wenn es einerseits gelingt, vorhandene Nicht-ATM-Subnetze in das ATM-Umfeld zu integrieren und wenn andererseits zwischen Kommunikationspartnern Internetverbindungen basierend auf ATM realisierbar sind.[1]

ATM bietet durch die Definition mehrerer ATM Adaptionsschichten (AAL, ATM Adaption Layer) die Möglichkeit, neben reinen ATM-Verbindungen (AAL 3/4) auch Verbindungen mit anderen Protokollen zu realisieren. Dafür kann die Adaptionsschicht AAL 5 genutzt. Dadurch können auch nicht-ATM-Netze in ATM integriert werden.

8.3.1 Punkt-zu-Punkt vs. Broadcast

An den Schnittstellen von ATM zu herkömmlichen LANs (etwa Token-Ring oder Ethernet) stoßen ein verbindungsorientiertes (ATM) und ein verbindungsloses Protokoll aufeinander. In ATM werden Ende-zu-Ende-Verbindungen zwischen jeweils genau zwei Kommunikationspartnern realisiert, während Token-Ring und Ethernet auf Broadcast-Mechanismen beruhen.

Das Protokoll IP basiert auf der Technologie verbindungsloser Netze. Um eine IP-Verbindung über ein ATM-Netz hinweg zu realisieren, würde es genügen, die IP-Protokollelemente einzukapseln und einen Tunnel durch das ATM-Netz hindurch aufzubauen. Wenn jedoch an der Kommunikation sowohl Stationen im herkömmlichen LAN als auch Stationen im ATM-Netz beteiligt sein sollen, müssen wesentliche Mechanismen der klassischen Netze in ATM realisiert oder simuliert werden. Das schwerwiegendste Problem dabei ist die in ATM realisierte Adreßauflösung, die im Gegensatz zu IP nicht durch Broadcast-Nachrichten realisiert werden kann, da es in ATM keine Broad- oder Multicast-Adressierung gibt.

Es gibt verschiedene Ansätze, das Internetprotokoll IP auf Basis von ATM zu realisieren, die kurz erwähnt werden sollen. Für eine detaillierte Beschreibung siehe [Scharf 96].

[1] Bei der derzeitigen Verbreitung von Endgeräten und IP-Kommunikationssoftware kann davon ausgegangen werden, daß ATM nur dann eingesetzt wird, wenn die bisherige Kommunikationssoftware einsetzbar bleibt.

8.3.2 LANE

Ein Ansatz ist LANE (LAN-Emulation), das vom ATM-Forum vorgeschlagen und standardisiert wurde.
Die Umsetzung der Adressen geschieht über einen dedizierten Server (LAN-Emulation-Server, LES), der über eine Tabelle die Zuordnung von ATM-Adressen zu virtuellen MAC-Adressen vornimmt. Eine Anfrage nach einer MAC-Adresse aus dem LAN heraus wird dann von diesem Server beantwortet.

Multicast und Broadcast geschieht ebenfalls über einen eigens dafür vorgesehenen Server (Broadcast and Unknown-Server, BUS), der die Anfrage entgegennimmt und dann seinerseits die Multicast-Verbindung über Punkt-zu-Mehrpunkt-Verbindungen zu den jeweiligen Zielstationen simuliert. Die Umsetzung der Multicast- oder Adreßanfragen in Verbindungen zum Multicast- oder Adreßserver wird durch spezielle LAN-Emulation-Clients vorgenommen, die in der LAN/ATM-Bridge und im ATM-Endgerät installiert sind.

Woher kennt ein ATM-Client nun die Adressen der beiden Server LES und BUS? Die Adressen könnten in jedem Client eingetragen werden. Dies wäre jedoch eine unflexible Lösung. Stattdessen gibt es einen weiteren Server (LAN Emulation Configuration Server, LECS), der die Verwaltung der Serveradressen übernimmt. Die LECS-Adresse ist somit die einzige auf den LECs manuell zu konfigurierende Adresse.

8.3.3 CLIP

CLIP (Classical IP and ARP over ATM) wird in [RFC2225] und [RFC1577] beschrieben und stellt einen Standardisierungsvorschlag der IETF dar. Die Vorgehensweise ähnelt der von LANE mit folgenden Unterschieden:

1. LANE betrachtet die Adressierung anhand von MAC-Adressen, während CLIP anhand von IP-Adressen adressiert. Dafür müssen Anfragen nach ARP-Adressen behandelt werden, was analog zu LANE mit einem dedizierten Server realisiert wird.

2. IP-Pakete werden bis zum Zielrechner in ATM-Zellen gekapselt.

8.3.4 MPOA

Um das Routing verschiedener Protokollemulationen über ATM zu ermöglichen, ohne mit jedem Protokoll eine eigene Routinginfrastruktur realisieren zu müssen, wurde vom ATM-Forum das Protokoll MPOA (Multiprotocol over ATM) entwickelt. Grundlage hierfür waren u.a die Standards der IETF zu CLIP [RFC2332] sowie das Protokoll LANE des ATM-Forum.

8.4 Firewalls in ATM-Netzen

Wenn Firewalls zur Kontrolle der Verbindungen in ATM-Netzen eingesetzt werden, so ergeben sich grundsätzliche Unterschiede zu derzeitigen Firewalls:

- Firewalls in ATM-Netzen sind nicht notwendigerweise IP-Firewalls. ATM ist ein Protokoll, mittels dessen auch IP-Dateneinheiten gekapselt und versendet werden können (beispielsweise mit CLIP, LANE, MPOA).

- Ein IP-Firewall in einem ATM-Netz muß zur Kontrolle der emulierten oder gekapselten IP-Pakete diese zunächst in IP-Pakete zurückverwandeln. Dies ist aufwendig und kann zu Performanz-Problemen führen.

Andererseits kann ein Firewall in einem ATM-Netz auch ATM-Spezifika nutzen, um Performanz-Verluste auszugleichen oder den Datendurchsatz zu erhöhen:

- Durch die Definition von Quality-of-Service-Parametern und die Möglichkeit, Bandbreite zu reservieren, kann bereits beim Verbindungsaufbau ein Mindestdurchsatz durch die Firewall garantiert werden oder falls Engpässe auftreten könnten, der Verbindungswunsch abgelehnt werden.

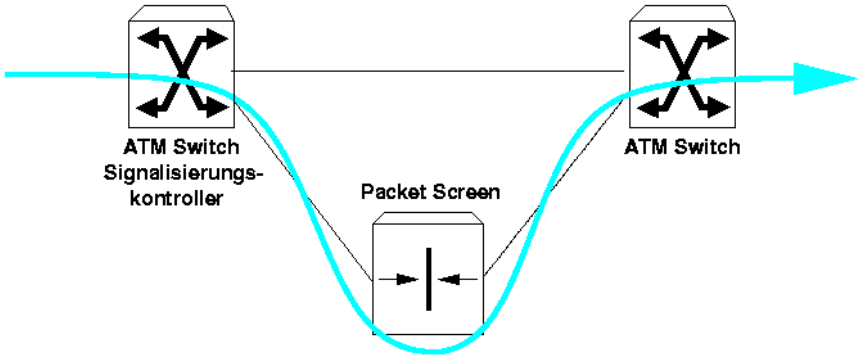


Abbildung 8.3: Einschleifen eines Proxy (aus [Ellermann 99])

- Da ATM ein verbindungsorientiertes Protokoll ist, kann beim Verbindungsaufbau durch einen ATM-Switch ein Virtueller Pfad realisiert werden, der durch einen dedizierten Firewallrechner führt. Dadurch kann transparent ein Proxy in die Verbindung eingefügt werden (vgl. Abb. 8.3). Bei Überlast auf diesem Proxy kann auch ein zweiter oder dritter Proxy anstelle des ersten bei weiteren Verbindungen gewählt werden.

- Firewallproxies müssen nicht notwendigerweise physikalisch in die Verbindung eingesetzt sein, es genügt, eine Virtuelle Verbindung durch den Proxy hindurch aufzubauen. Dadurch können auch zentrale Gruppen von Proxies vorgehalten werden.

- Eine weitere Möglichkeit ist die Bildung virtueller privater Teilnetze in Unternehmen, die durch logisch eingehängte Firewalls auch untereinander gesichert werden können. Es ist hier kein aufwendiges Tunneling wie in reinen IP-Netzen notwendig; die Kopplung von Endgeräten zu Teilnetzen geschieht rein durch Konfiguration der ATM-Switches (vgl. Abb. 8.4).

In allen den beschriebenen Szenarien kommt den ATM-Switches eine besondere Bedeutung zu. Die Kontrolle von ATM-Verbindungen, also der Aufbau oder die Ablehnung der Verbindung sowie das Einschleifen von Firewallproxies, geschieht beim Verbindungsaufbau durch einen ATM-Switch.

Im folgenden soll deshalb der Verbindungsaufbau in einem ATM-Switch (ATM-Signalisierung) hinsichtlich der Kontrollmöglichkeiten beim Verbindungsaufbau untersucht werden.

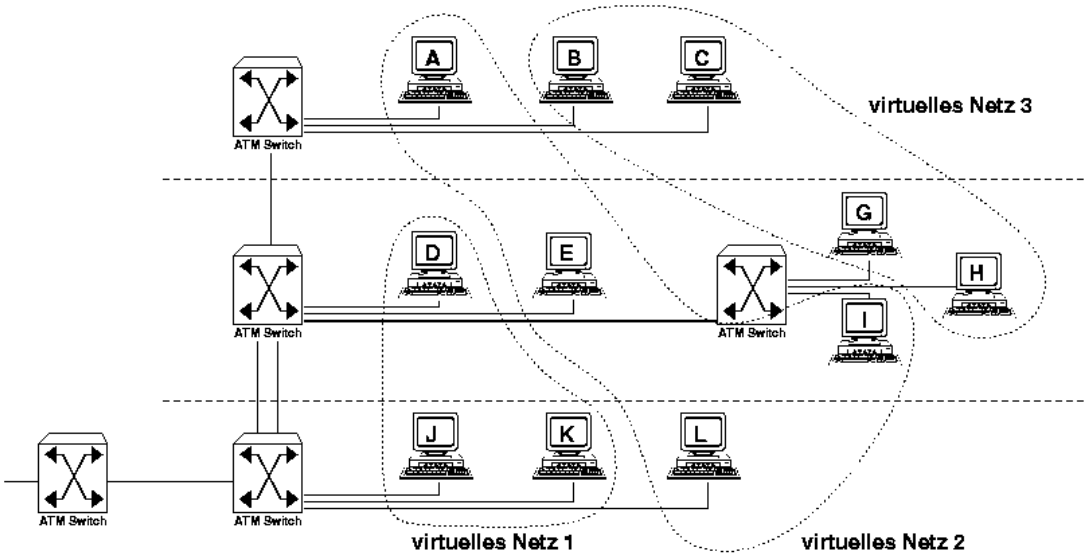


Abbildung 8.4: Aufbau virtueller LANs (aus [Ellermann 99])

8.5 ATM-Signalisierungsnachrichten

Der Aufbau einer ATM-Verbindung geschieht durch Senden standardisierter Nachrichten, die in der ATM-Terminologie als Signalisierungsnachrichten bezeichnet werden. Der Nachrichtenaustausch zum Zweck des Verbindungsauf/abbaus wird daher auch Signalisierung genannt.

Beispiele für Signalisierungsnachrichten sind:

SETUP, ADD_PARTY, DROP_PARTY, CONNECT, RELEASE, RESTART, u.a.

Am Beispiel des Verbindungsaufbaus ist der Austausch der erforderlichen Signalisierungsnachrichten in Abb. 8.5 dargestellt.

Signalisierungsnachrichten können mit zusätzlichen Parametern versehen sein, diese können zur Auswertung und Zugriffskontrolle auf ATM-Firewalls herangezogen werden. Beispielsweise enthält die SETUP-Nachricht die folgenden Parameter:

Notwendige Parameter sind:

Protocol discriminator
Call reference
Message Type = SETUP
Message Length
ATM traffic descriptor
Broadband bearer capability
Called party Number
QoS-Parameter
Endpoint reference (nur bei Punkt-zu-Mehrpunkt-Signalisierung)

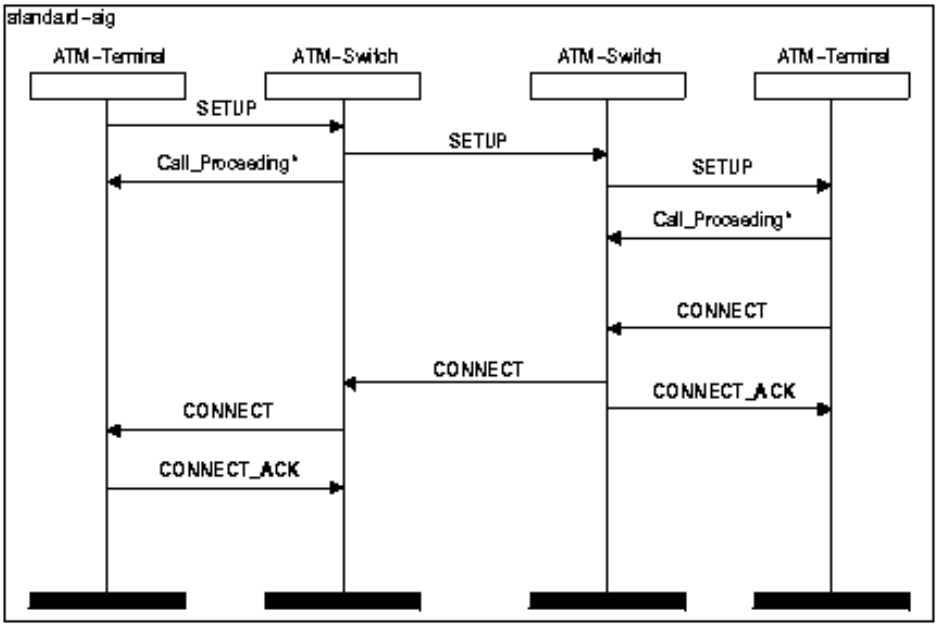


Abbildung 8.5: Verbindungsaufbau (aus [Ellermann 99])

Optionale Parameter sind:

- AAL-Parameter
- Broadband high layer information
- Broadband low layer information
- Calling party number
- Calling party subaddress
- Called party subaddress
- Transit network information
- Endpoint reference

Ergibt die Auswertung der SETUP-Nachricht, daß die Verbindung nicht zulässig ist, so kann der Verbindungsaufbau abgelehnt werden. Handelt es sich um eine zulässige Verbindung, so kann die Verbindung entweder direkt oder mit Umweg über einen Proxy weiter aufgebaut werden.

8.6 Zusammenfassung zum Teil 1

In diesem ersten Teil wurde die ATM-Technologie als ein Beispiel für Hochgeschwindigkeitsnetze beschrieben. Künftig werden Hochgeschwindigkeitsnetze eingesetzt werden, da heutige Netztechnologien für multimediale und datenintensive Anwendungen eine zu geringe Bandbreite aufweisen. Damit ergibt sich auch die Notwendigkeit der Kontrolle des Datenverkehrs zum Schutz vor Angriffen, was heute durch IP-Paketfilter und Applikations-Proxies erreicht wird.

In ATM kommt der Kontrolle des Verbindungsaufbaus eine entscheidende Bedeutung zu, da nur hier das Einschleifen von Applikations-Proxies erfolgen kann. Zusätzlich kann eine Lastverteilung vorgenommen werden, so daß auch Dienstgüteparameter garantiert werden können.

Durch die oben dargestellten Möglichkeiten zur Zugriffskontrolle auf Signalisierungsnachrichten können auch sehr effizient mehrere virtuelle Netze über einem physikalischen ATM-Netz betrieben werden, die untereinander und nach außen flexibel durch Firewalls abgesichert sind. Im Gegensatz zu paketorientierten Netzen, wie beispielsweise Fast Ethernet, erfolgt diese Zugriffskontrolle nur einmal beim Verbindungsaufbau und muß nicht für jedes Paket einzelnd erfolgen.

Konkrete Architekturen für Firewalls in ATM-Netzen werden im Teil 2 des Vortrages beschrieben.

Anmerkung zum Teil 2

Teil 2 der schriftlichen Ausarbeitung fehlt. Obwohl Herr Marienhagen mehrfach auf die Abgabe der schriftlichen Ausarbeitung angesprochen wurde, ist dieser Teil leider nicht fertiggestellt worden.

Literaturverzeichnis

[Ellermann 99] Ellermann, U. & Benecke, C.: „Einsatz von Firewalls in reinen ATM-Netzen, Studie des DFN-FWL“, Universität Hamburg, undatiert

[RFC1577] Laubach, M.: „Classical IP and ARP over ATM“, RFC 1577, Network Working Group, Januar 1994

[RFC2225] Laubach, M. & Halpern, J.: „Classical IP and ARP over ATM“, RFC 2225, Network Working Group, April 1998

[RFC2332] Luciani, J. et al.: „NBMA Next Hop Resolution Protocol (NHRP)“, RFC 2332, Network Working Group, April 1998

[Scharf 96] Scharf, A.: „ATM ohne Geheimnis“, Hannover : Verlag Heinz Heise 1996

Kapitel 9

„Active Content“ und mobiler Code

Andreas Lessig
Universität Hamburg
Fachbereich Informatik

Zusammenfassung

Das vorliegende Dokument betrachtet die mit „Active Content“ (Java, JavaScript, VBScript, Plugins, Active X) und mobilen Agenten einhergehenden Gefahren und Probleme und wirft einen kurzen Blick auf Lösungsmöglichkeiten.

9.1 Einleitung

Seit seiner Entstehung ca. 1992 hat sich das World Wide Web von einem textbasierten Informationssystem für Wissenschaftler zu einer bunten interaktiven Erlebniswelt für jedermann gewandelt. Im folgenden werden die Schattenseiten der Technologien beleuchtet, die dies ermöglicht haben, und die vielleicht in Zukunft die Möglichkeiten des Web noch erweitern werden.

Der erste Abschnitt beschäftigt sich mit aktiven Inhalten, die es erlauben, eine Webseite interaktiv auf die Benutzer reagieren zu lassen. Dazu werden die heute gängigen Technologien vorgestellt, ihr Potential für Angriffe beleuchtet und einen Blick auf Lösungsmöglichkeiten geworfen.

Im zweiten Teil geht es dann um eine Technologie, die heute erst eine Vision in den Köpfen einiger Forscher ist, die morgen aber völlig neue Möglichkeiten in der Benutzung des World Wide Web und im E-Commerce erschließen könnte. Die mobilen Agenten könnten es erlauben, eigene Hilfsprogramme zu entsenden, um für ihren Besitzer autonom im Netz zu agieren. Es soll kurz erklärt werden, wie man sich dies vorzustellen hat, welche Probleme heute schon abzusehen sind und welche Lösungsansätze in der Forschung z.Z. diskutiert werden.

9.2 Active Content

9.2.1 Begriffsbestimmung

Unter „Active Content“ oder aktiven Inhalten sollen all diejenigen Technologien verstanden werden, die beim Betrachten einer HTML-Seite auf dem Rechner des Betrachters wie ein Programm ausgeführt werden. Es ist dabei unerheblich, ob die HTML-Seite mittels eines Browsers von einem Webserver geladen oder z.B. per E-Mail empfangen und mit einem HTML-fähigen E-Mail-Klienten betrachtet wird.

Wir werden dabei die folgenden Technologien betrachten:

1. Java
2. JavaScript
3. VBScript
4. Active X
5. Netscape-Plug-Ins

Wie aus der Aufstellung zu erkennen ist, handelt es sich um ein weites Feld, das von interpretierten Sprachen mit eingebauten Sicherheitskontrollen bis hin zu Technologien reicht, bei denen neue Funktionsbibliotheken in das System gebracht werden, die ohne jegliche Überwachung ausgeführt werden. Aus diesem Grunde sollen die einzelnen Technologien im folgenden etwas genauer dargestellt werden.

9.2.2 Technologien

Java

Java entstand 1991 als Forschungsprojekt von Sun Microsystems und sollte ursprünglich als einfache und kompakte Sprache zur Programmierung von Elektrogeräten wie Fernsehern und Kühlschränken dienen. 1994 stellte Sun dann aber den Browser HotJava vor, der erstmals demonstrierte, daß sich die Sprache auch zur Programmierung komplexerer Anwendungen (Applications) eignet. Darüber hinaus wurde demonstriert, wie man kleine Codebibliotheken, sogenannte Applets, in Webseiten einbinden und so die bis dato eher statischen Webseiten dynamisch mit dem Benutzer interagieren lassen können.

Java ist eine objektorientierte Sprache, die keine globalen Variablen oder Funktionen kennt, sondern nur Objekte, die ihrerseits Methoden und Attribute besitzen und konkrete Ausprägungen einer Klasse sind. Fast jede Klasse liegt dabei als eigenständige Datei mit der Endung `.class` vor und kann bei Bedarf dynamisch zur Laufzeit des Programms geladen werden.

Aus der Sicht des Programmierers sind Applications dabei spezielle Objekte, die eine Methode `main()` implementieren, während es sich bei Applets um ein von dem Objekt `java.applet` abgeleitetes Kindobjekt handelt. Dabei können Applications direkt ausgeführt werden. Sie entsprechen einem normalen Programm. Applets sind dagegen darauf angewiesen, daß schon eine Anwendung existiert, die eine Laufzeitumgebung zur Verfügung stellt. Üblicherweise handelt es sich dabei um einen Browser.

Im Gegensatz zu Applications sind Applets keine vollständigen Programme. Sie dienen vielmehr dazu, Webseiten um interaktive Anteile zu bereichern. Sie werden in der Regel vom selben Server geladen, von dem auch die Webseite stammt, die sie referenziert und ermöglichen es, z.B. ein graphisches Benutzerinterface für ein Bestellsystem eines Online-Händlers zu realisieren.

Sie besitzen keine Methode, die sofort aufgerufen wird, sondern können erst aktiv werden, wenn bestimmte Ereignisse eintreten. Dabei kann es sich z.B. um das Laden des Applets in eine Webseite handeln, einen Mausklick oder das Entladen der Webseite. Es ist die Aufgabe der Laufzeitumgebung, die mit diesen Ereignissen verknüpften Methoden aufzurufen und dem Applet ein Fenster zur Verfügung zu stellen, in dem es sich darstellen kann.

Die Ausführung von Java-Code übernimmt dabei eine „Virtual Machine“, d.h. ein emulierter Prozessor, der eine spezielle Maschinensprache, den sogenannten „Bytecode“ ausführt, die mit der tatsächlich vom Prozessor des Rechners verstandenen Maschinensprache nichts gemein zu haben braucht. Diese „virtuelle Maschinensprache“ wurde von Sun genormt und ist auf allen Betriebssystemen gleich. Auf diese Weise kann Java-Code auf jeder Plattform ausgeführt werden, für die eine „Virtual Machine“ implementiert wurde.

Selbstverständlich ist es aber nicht erforderlich, tatsächlich in Bytecode zu programmieren. Vielmehr existiert ein Compiler, der die Hochsprache Java in Bytecode übersetzt, welcher dann wie ein normales Programm an die Anwender weitergegeben werden kann.

Bei der Entwicklung von Java wurden auch Sicherheitsüberlegungen angestellt. Als Folge davon hat die objektorientierte Sprache z.B. ein strenges Typenkonzept, verzichtet auf Pointer und hat eingebaute Sicherheitsfunktionen, die Aufrufe bestimmter Funktionen auf ihre Rechtmäßigkeit überprüfen sollen.

Dies geschieht generell in drei Mechanismen, dem Bytecode Verifier, dem Class Loader und dem Security Manager (vgl. [MGFe 99]). In Java 2 kommt zusätzlich noch der Access Manager als vierte Komponente hinzu.

Soll ein Objekt einer Javaklasse instanziiert werden, die noch nicht von der virtuellen Maschine in den Hauptspeicher geladen wurde, so wird dies nachgeholt. Dabei wird die zu ladende Klasse vom Bytecode-Verifier untersucht, der fest in die Virtual Machine eingebaut ist. Er führt diverse Überprüfungen durch, die sicherstellen sollen, daß die `.class-Datei` das korrekte Format hat, Stacks nicht überlaufen, die Argumente für Funktionen immer den richtigen Typ haben, keine illegalen Typumwandlungen geschehen, die Sichtbarkeitsregeln für Objekte eingehalten werden und alle Zugriffe auf Register der virtuellen Maschine gültig sind. Da einige Eigenschaften nicht zufriedenstellend durch eine statische Untersuchung sichergestellt werden können, existieren weitere Überprüfungen zur Laufzeit.

Um eine Javaklasse laden zu können, wird ein Classloader benötigt. Hierbei handelt es sich entweder um ein Objekt, das von `java.lang.ClassLoader` abgeleitet wurde, oder um den Urlader, der direkt in die Virtual Machine eingebaut ist.

Ein Class Loader hat dabei drei Aufgaben. Er muß die nötigen Verfahren implementieren, um Klassen entweder von der Festplatte oder mittels eines Netzwerkprotokolls in den Hauptspeicher zu laden, wenn sie nicht schon früher geladen wurden. Darüber hinaus ist es seine Aufgabe, sicherzustellen, daß importierte Klassen nicht ohne weiteres Systemklassen ersetzen. Schließlich ist er für die Verwaltung von Namensräumen zuständig. Nur Klassen im selben Namensraum können gegenseitig ihre Methoden aufrufen. Dies wird dazu benutzt, Applets von verschiedenen Servern gegeneinander abzuschirmen.

Vertrauenswürdige Anwendungen können eigene Class Loader definieren und so Einfluß darauf nehmen, wie von ihnen benötigte Klassen geladen werden.

Beim Security Manager handelt es sich schließlich um ein Objekt, das von allen potentiell gefährlichen Funktionen der Java API aufgerufen wird. Widerspricht der Aufruf der im Security Manager implementierten Politik, so wird eine Exception[1] ausgelöst, die an das aufrufende Programm zurückgegeben wird. Andernfalls kann der Aufruf stattfinden.

Pro virtueller Maschine kann nur ein Security Manager zur Zeit aktiv sein. Wurde ein Security Manager installiert, so kann er nicht ohne Neustart der Virtual Machine deinstalliert werden. Anwendungen können aber während ihrer Initialisierung einen eigenen Security Manager installieren, der spezielle Politiken umsetzt.

Die Politiken, die der Security Manager umsetzt, haben sich im Verlauf der Entwicklung von Java gewandelt. Bisher wurden dabei drei Stationen durchlaufen. Das JDK[2] 1.0 implementierte eine recht simple Sicherheitspolitik. Zugriffe von Applications wurden grundsätzlich gestattet, Applets liefen in einer „Sandbox“. Unter diesem Begriff wurde ein Satz von Einschränkungen verstanden, der verhindern sollte, daß Applets irgendwelche schädlichen Nebenwirkungen ausüben konnten.

Sie durften (Auszug):

- keine Zugriffe auf das lokale Dateisystem tätigen,
- keine Netzwerkverbindungen zu anderen Rechnern öffnen, als dem von dem sie geladen wurden,
- nicht auf Netzwerkverbindungen warten oder diese entgegennehmen,
- keine Angaben zu ihrem Benutzer abfragen,
- keine Systemeigenschaften umdefinieren,
- keine externen Programme ausführen oder dynamische Bibliotheken laden,
- nicht die Virtual Machine beenden und
- weder einen Class Loader noch einen Security Manager installieren.

Mit JDK 1.1 wurde die Möglichkeit geschaffen, Code digital zu signieren. Ein Applet, das auf diese Weise für „vertrauenswürdig“ erklärt wurde, wurde mit denselben Rechten wie eine Application ausgeführt.

War bis jetzt Java-Code entweder „vertrauenswürdig“ oder nicht, womit er entweder alle oder keine Rechte hatte, änderte sich dies mit Java 2 (JDK 1.2). Nun ist es möglich, gezielt Politiken aufzustellen, welche Rechte ein bestimmtes Applet oder eine Application erhalten soll. Um dies zu ermöglichen, wurde mit dem Access Controller eine neue Sicherheitskomponente geschaffen.

Soll der Security Manager einen Funktionsaufruf beurteilen, tut er dies nicht mehr selbst, sondern ruft den Access Controller auf, der die gewünschte Funktion mittels einer Liste mit Politiken überprüft, in denen der Benutzer oder Systemadministrator definiert hat, welche Rechte Code eines bestimmten Ursprungs hat. Dabei kann auch festgelegt werden, ob eine digitale Signatur einer bestimmten Instanz notwendig ist.

[1] Dies ist ein spezieller Mechanismus der Fehlerbehandlung. Ein gerade behandelter Codebereich wird beendet, und es wird zu einer Behandlungsroutine gesprungen.

[2] Die Abkürzung steht für Java Development Kit. Bis JDK 1.2 war es üblich, auf diese Weise die Javaversion anzugeben. Die in JDK 1.2 vertriebene Version wird von SUN dagegen als „Java 2“ bezeichnet.

JavaScript

Bei JavaScript handelt es sich um eine Skriptsprache, die ursprünglich von Netscape entwickelt wurde, inzwischen aber auch von vielen anderen Browsern unterstützt wird. Auch der Internet Explorer versteht z.B. einen Dialekt von JavaScript, der allerdings nicht immer voll kompatibel zur Netscape-Variante ist.

Trotz ihres Namens hat die Sprache recht wenig mit Java gemein. Weder hat sie das selbe Maß an eingebauten Sicherheitschecks noch ein strenges Typenkonzept. Routinen zur Manipulation von Dateien fehlen, allerdings finden sich immer wieder indirekte Wege, trotzdem auf lokale Dateien zuzugreifen.

Programme in JavaScript können direkt als Quelltext in einer HTML-Seite stehen, deren Inhalt sie auch dynamisch verändern können. Auch das Öffnen neuer Browserfenster sowie das Erscheinungsbild des Browsers können über JavaScript beeinflußt werden. Tatsächlich werden beim Netscape Navigator die Benutzereinstellungen als JavaScript-Datei abgelegt, die bei jedem Neustart ausgeführt wird. Dies geschieht, bevor die Sicherheitseinstellungen in Kraft treten, die es dem Browser z.B. verbieten, JavaScript auszuführen. Dies ist auch notwendig, da auch die Sicherheitseinstellungen als JavaScript-Befehle in besagter Konfigurationsdatei stehen.

VBScript

Auch Microsoft entwickelte eine Skriptsprache zum Einsatz auf Webseiten. Diese lehnte sich syntaktisch an „Visual Basic“ an und wurde daher „Visual Basic Scripting Edition“ (kurz: VBScript) genannt.

Außer der Tatsache, daß VBScript bislang ausschließlich vom Internet Explorer verstanden wird, liegen die Hauptunterschiede zu JavaScript darin, daß VBScript zum einen nur einen Datentyp kennt, während es zum anderen in das Microsoft eigene „Active X-Konzept“ eingebunden ist (s.u.). So kennt VBScript die Möglichkeit, im System registrierte Objekte zu instanziieren, wenn die Sicherheitseinstellungen des Browsers dies zulassen. Beispiele für solche Objekte sind das `FileSystemObject` bzw. das `WordDocumentObject`. Während ersteres das Fehlen von Funktionen für die Manipulation von Dateien ausgleicht, erlaubt letzteres das komfortable Lesen und Ändern von Dokumenten in MS-Office-Formaten inklusive des Einbringens neuer Makros[3].

Active X

Active X ist ein komplexes Konzept,[4] dessen vollständige Erörterung den Rahmen dieses Dokumentes sprengen würde. In diesem Zusammenhang sollen daher nur die „COM-Objekte“[5] vorgestellt werden,[6] die es erlauben, unter Windows Objekte („Controls“) zu registrieren, die auf eine definierte Weise aufgerufen werden können, um bestimmte Aufgaben zu übernehmen.

So existiert z.B. ein Objekt, das ein einfaches Editorfenster implementiert. Um aus diesem Control ein lauffähiges Programm zu machen, reicht es im Prinzip, ein Rahmenfenster und ein Menü zu erzeugen. Die Verarbeitung und Darstellung der vom Benutzer eingegebenen Zeichen erfolgt dabei

[3] Unter Makros werden in Dokumente eingebettete Befehle verstanden. Die Makrosprache der Microsoft-Office-Produkte ist umfassend genug, daß darin sogar Viren geschrieben werden können, die andere Dokumente infizieren.

[4] Laut [Box 98] dauert es ein halbes Jahr, es wirklich zu verstehen.

[5] Die Abkürzung steht für Component Object Model.

[6] Active X umfaßt sehr viel mehr, wenn aber von Active-X-Controls gesprochen wird, sind in der Regel COM-Objekte gemeint.

autonom durch das Control. Das Rahmenprogramm ist lediglich dafür verantwortlich, vom Benutzer angewählte Menüpunkte in Steuerbefehle für das Control umzusetzen, sowie den eingegebenen Text bei Bedarf auszulesen und in einer Datei zu speichern, bzw. bei Bedarf aus einer Datei neu einzulesen.

Technisch werden solche Objekte in der Regel als spezielle dynamische Bibliotheken (DLLs[7]) realisiert, die bei Bedarf über Systemfunktionen angesprochen werden können. Sie werden dabei über CLSIDs[8] genannte Bezeichner referenziert. Dabei handelt es sich um 128-Bit-Zahlen, die auf eine Weise generiert werden können, die sicherstellen soll, daß die generierte CLSID weltweit eindeutig ist.[9]

Controls können auch ähnlich wie Applets von Webseiten gestartet werden.[10] Diesen Vorgang bezeichnet Microsoft als „Scripting“. Dazu müssen nur die CLSID sowie u.U. eine URL angegeben werden, von der das Control heruntergeladen werden kann.

Letzteres ist nicht nötig, wenn es schon lokal installiert ist. Dies kann insbesondere dann der Fall sein, wenn es Teil eines Software-Paketes ist. Neben der Benutzung in Webseiten ist COM auch eine Technologie, die es erlaubt, Anwendungen in Komponenten aufzuspalten, die bei Bedarf beliebig neu kombiniert werden können. Als Folge davon sind in einem Windows-System in der Regel die unterschiedlichsten Controls vorhanden, die in ihrer Funktion von der Darstellung von Tabellen über das Betreiben eines Telnet-Servers bis zur Formatierung der lokalen Festplatte reichen.

Um dennoch eine Kontrolle darüber zu haben, welche lokal installierten Controls von Webseiten aufgerufen werden können, dürfen normalerweise nur solche Controls von Webseiten aufgerufen werden, die von ihrem Programmierer als „sicher für Scripting“ markiert wurden.[11]

Im Gegensatz zu Applets existiert kein Mechanismus, um die Ausführung eines Controls zu überwachen. Einmal aufgerufen verhält sich ein Control wie ein lokales Programm. Es ist daher möglich, alle Funktionalitäten als Control zu programmieren, die auch in einem normalen Programm realisierbar wären (Festplatte löschen, Tastatureingaben mitlesen, Daten ausspionieren, Netzwerkverbindungen öffnen, . . .).

Die einzige Kontrollmöglichkeit besteht in der Entscheidung, ob ein Control geladen und ausgeführt werden soll. Soll ein Control aus dem Internet mit dem Internet Explorer 5 mit Standardeinstellungen heruntergeladen und ausgeführt werden, so geschieht dies nur, wenn es signiert ist, der Benutzer zugestimmt hat und das Control von seinem Autor als „sicher für Scripting“ markiert wurde. Die Frage, wann eine Signatur gültig ist, soll dabei erst einmal zurückgestellt werden.

Netscape-Plug-Ins

Plug-Ins wurden ursprünglich für Bildbearbeitungsprogramme entwickelt. Hierbei handelt es sich um dynamisch nachladbare Komponenten (unter Windows meist DLLs), die es erlauben, die Fähigkeiten eines Programmes nachträglich zu erweitern. Auch der Netscape Navigator kennt Plug-Ins, die es ihm ermöglichen, neue Dateitypen darzustellen.

[7] Die Abkürzung steht für Dynamic Link Library.

[8] Die Abkürzung steht für CLasS IDentification. Tatsächlich stellt COM auch den Versuch dar, ein Programmiersprachen übergreifendes Binärformat für objektorientierte dynamische Bibliotheken zu schaffen.

[9] Dazu wird auf die MAC-Adresse einer eingebauten Ethernet-Karte zurückgegriffen. Existiert im Rechner keine Ethernet-Karte, so wird ein Pseudozufallszahlengenerator benutzt, dessen Ausgabe zumindest mit großer Warscheinlichkeit eine CLSID generiert, die bisher noch nicht benutzt wird.

[10] Tatsächlich bezeichnet der Internet Explorer in seinen Warnmeldungen jegliche aktiven Inhalte in Webseiten als ActiveX-Steuerelemente.

[11] Dies ist in den Einstellungen des Internet Explorers konfigurierbar.

Im Rahmen dieser Betrachtung ist der Unterschied zwischen Plug-Ins und Active X-Controls vernachlässigbar. Beide Verfahren kennen als einzige Schutzmaßnahme Zertifikate, anhand derer der Benutzer entscheiden kann, ob die jeweilige Datei heruntergeladen und ausgeführt werden soll. Ist dies erst einmal geschehen, so hat der Benutzer keinerlei Kontrolle über ihre Auswirkungen.

9.2.3 Probleme und Gefahren

Grundlegendes

Wenn man aktive Inhalte aus dem Internet herunterlädt und startet, dann muß man sich darüber im Klaren sein, daß man Programme ausführt, von denen man i.A. weder genau weiß woher sie stammen, noch was sie tun. Auch wenn Java und die hier vorgestellten Skriptsprachen im Gegensatz zu Plug-Ins und Active X-Controls in ihrer Funktionalität eingeschränkt sind, so heißt dies nicht, daß sie deshalb völlig ungefährlich sind. Zum einen werden mit steter Regelmäßigkeit neue Implementationsfehler gefunden, die Zugriffe erlauben, die in der Spezifikation ausdrücklich verboten wurden (vgl. [Guninski 98]), zum anderen erlauben es die einzelnen Technologien auch sich gegenseitig aufzurufen.

Auf diese Weise kann die Tatsache ausgenutzt werden, daß die vorgesehenen Einschränkungen der einzelnen Technologien nicht deckungsgleich sind. So könnte ein JavaScript-Programm Java-Routinen aufrufen, um auf Dateien zuzugreifen, während ein Java-Programm mittels JavaScript bestimmte Browsereigenschaften abfragen kann, die ihm sonst nicht zugänglich wären. Das bedeutet, daß auf diese Weise hybride Programme geschrieben werden können, deren Möglichkeiten über das hinausgehen, was jeweils nur in einer Sprache realisierbar wäre.

Im folgenden sollen einige ausgewählte Beispiele für Angriffe mittels Skriptsprachen oder Java betrachtet werden, die die erwähnten Probleme näher beleuchten. Dabei sollte beachtet werden, daß, obwohl die gefährlicheren Probleme auf Implementationsfehler der Browser herrühren, sie keineswegs leicht abgetan werden können. Die Aussage, ein Problem sei ja „nur ein Bug“, hat nur dann einen Wert, wenn dies bedeutet, daß das Problem nach der Behebung des Fehlers beseitigt wurde. Dem ist aber nicht immer so. Viele der im folgenden vorgestellten Probleme tauchen in jeder neuen Browserversion wieder in einem neuen Gewand auf. Es soll daher auch nicht auf die genaue Funktionsweise der jeweiligen Angriffe eingegangen werden. Vielmehr soll die Aufstellung verdeutlichen, welche Probleme aktive Inhalte dem Betrachter von Webseiten bereiten können.

Belästigung

Die Belästigung stellt die Klasse von Angriffen dar, die am einfachsten zu implementieren ist. Diese Angriffe zielen darauf ab, die Benutzung eines Rechners unangenehm zu machen oder zu verhindern. Grundsätzlich gilt aber, daß Angriffe dieser Art keine Auswirkungen haben, die nach einem Neustart des Rechners fortbestehen.

Die einfachsten Beispiele sind Programme, die übermäßig Rechenzeit verbrauchen und so den Rechner verlangsamen, die ungewöhnliche Geräusche erzeugen, die Tonausgabe blockieren oder eine große Anzahl von Fenstern öffnen. Auch sind unter Windows 95/98 Angriffe bekannt, die dazu führen, daß ein Browserfenster sich nicht einmal mehr durch den Taskmanager[12] beenden läßt. Im schlimm-

[12] Ein Systemprogramm, mit dem Prozesse beendet werden können.

sten Fall können diese Angriffe sogar dazu führen, daß der Browser abstürzt und u.U. sogar das Betriebssystem in einen Zustand versetzt, der einen Neustart erfordert.

Angriffe dieser Klasse sind in der Regel nicht schwer zu programmieren und können oft nicht wirksam verhindert werden, da alle ausgeführten Aktionen für sich genommen völlig legal sind. Wenn ein Applet z.B. Fibonacci-Zahlen berechnet, so mag das zwar bedeuten, daß der Rechner unerträglich langsam wird, der Browser kann aber kaum automatisch entscheiden, ob es einen guten Grund für besagte Berechnungen gibt.

Angriffe mit konkretem Schaden

Sind Belästigungs-Angriffe oft eher lästig als wirklich gefährlich, so sind andere Angriffe deutlich problematischer. Regelmäßig werden neue Methoden gefunden, die es einem Angreifer erlauben, wichtige Daten unbefugt aus dem Rechner auszulesen. Die Webseiten von [Guninski 98] enthalten Beispiele, wie es für diverse Versionen von Internet Explorer und Navigator möglich ist, lokale Dateien, „Bookmarks“ oder die Liste der zuletzt besuchten URLs auszulesen.

In [FeBaDeWa 97] wird ein Angriff beschrieben, der darauf basiert, einen Besucher auf eine gefälschte Webseite zu locken, die sich auf einem Rechner befindet, den der Angreifer kontrolliert. Diese Fälschung erweckt den Eindruck, es handele sich um eine Seite eines bekannten Servers. Sie entspricht dem Original, mit Ausnahme der in ihr enthaltenen Verknüpfungen. Diese wurden so manipuliert, daß sie nicht direkt auf ihre Ziele zeigen, sondern auf ein CGI-Skript auf dem Rechner des Angreifers, welches die angeforderten Seiten lädt, um dann alle in ihr enthaltenen Verweise wieder in der selben Art zu manipulieren. Als Folge werden alle Seiten, die der Betrachter für den Rest seiner Sitzung lädt, über den Rechner des Angreifers angefordert. Dieser kann sie dabei nicht nur betrachten und so u.U. vertrauliche Daten (Web-E-Mails, Paßwörter, Kreditkarteninformationen, ...) mitlesen, er kann sie auch manipulieren, um so gezielt Fehlinformationen zu verbreiten.

Eine extrem primitive Variante dieses Angriffs wurde dazu verwendet, einen Aktienkurs zu manipulieren. Glücklicherweise konnte der Angreifer aber gefaßt und vor Gericht gebracht werden. Immerhin war es ihm gelungen, eine großen Anzahl von Anlegern zu überzeugen, seine Firma würde in Kürze von einer größeren Gesellschaft übernommen. Dies erreichte er, indem er Anleger aus einem Diskussionsforum auf eine gefälschte Seite von Bloomberg TV lockte, deren Verknüpfungen wieder auf echte Seiten des Senders zeigten. Dies trieb den Aktienkurs derartig in die Höhe, daß er einen recht ansehnlichen Gewinn von immerhin 93.000 Dollar verbuchen konnte [Newsticker 99].

Schließlich sind auch noch Angriffe bekannt geworden, die es erlauben, lokale Dateien zu löschen oder mit Viren zu versehen.

Umgehung von Sicherheitsmaßnahmen

Bevor ein Angriff gelingen kann, ist es manchmal erforderlich, bestehende Sicherheitsmaßnahmen zu umgehen. Auch hierbei können HTML-Seiten mit aktiven Inhalten eine tragende Rolle spielen.

Das bekannteste Beipiel für einen derartigen Angriff wäre das Anzeigen einer Dialogbox mit einem Inhalt der Art „Es tut uns leid, aber sie müssen ihr Paßwort erneut eingeben“. Angriffe dieser Art sind weit verbreitet und einfach zu programmieren. Die Dialogboxen sind dabei oft nicht von tatsächlich existierenden Meldungen eines Betriebssystems oder Internet-Content-Providers zu unterscheiden.

Andere Angriffe zielen darauf ab, den Browser so zu manipulieren, daß er Inhalte ausführt, die er normalerweise auf Grund von Sicherheitseinstellungen ignorieren würde. Ein Ansatz ist z.B. seine Sicherheitseinstellungen zurückzusetzen. Dies war über einen Fehler in einem Netscape Navigator 4.07 möglich. Eine speziell geformte URL konnte dazu führen, daß das Programm beim nächsten Start seine Konfigurationsdatei für defekt erklärte und mit den (u.U. nicht so sicheren) Standardeinstellungen überschrieb. Eine ganze Reihe anderer Angriffe zielte darauf ab, den Internet Explorer ein Skript so ausführen zu lassen, als wäre es von einer „vertrauenswürdigen“ URL geladen worden.

Schließlich soll noch erwähnt werden, daß einige Benutzer anonymisierende Proxyserver benutzen, um ihre Identität beim Surfen im Web geheimzuhalten. Auch ist es gängige Praxis, die IP-Adressen von Rechnern hinter einer Firewall mit ähnlichen Mitteln geheim zu halten. Diese Maßnahmen können recht einfach unwirksam gemacht werden, wenn ein Browser wie z.B. eine 4er Version des Netscape Navigators eine veraltete Java Implementation benutzt. Diese erlaubte es nämlich, die IP-Adresse des lokalen Rechners abzufragen. Es ist dabei noch nicht einmal erforderlich, daß wirklich ein Applet nachgeladen wird, da auch JavaScript Java-Methoden aufrufen kann.

9.2.4 Schutzmaßnahmen

Schutzmechanismen der Browser

Generelles Verbot aktiver Inhalte

Aktive Inhalte werden vom Browser in das System eingebracht und durch ihn ausgeführt oder interpretiert. Er sollte daher auch einen gewissen Schutz vor unangenehmen Nebeneffekten dieser Inhalte bieten, wie sie in den vorherigen Abschnitten diskutiert wurden.

In den einzelnen Browsern werden dabei unterschiedliche Ansätze verfolgt, diesen Schutz zu realisieren. So erlaubt es der Netscape Navigator nur, generell zu entscheiden, ob Java oder JavaScript ausgeführt werden sollen oder nicht. Dieser Schutz ist recht effektiv und kann höchstens umgangen werden, indem ein Fehler des Browsers dazu benutzt wird, sämtliche Einstellungen zurückzusetzen. Allerdings wurde besagter Implementationsfehler inzwischen behoben und war auch alles andere als unauffällig. Es erschienen Fehlermeldungen und auch eventuell notwendige Einstellungen (Proxies, Mailserver, E-Mail-Identität, ...) wurden gelöscht.

Dem fast optimalen Schutz vor aktiven Inhalten steht allerdings die Tatsache gegenüber, daß es immer mehr Seiten im Internet gibt, die ohne aktiviertes JavaScript nicht darstellbar oder navigierbar sind. Für diese Seiten jeweils JavaScript anzuschalten, um dann sofort wieder zurückzustellen, oder den Quelltext im Kopf auszuwerten, ist der Mehrheit der „normalen“ Benutzer nicht zuzumuten.

Sicherheitszonen

Der Internet Explorer verfolgt seit Version 4.0 ein Konzept mit deutlich feineren Kontrollen. Statt „JavaScript Ja/Nein“ existieren Dutzende von Einstellungen mit so bezeichnenden Namen wie „ActiveX-Steuerelemente und Plugins ausführen, die für Scripting sicher sind“. Hierdurch können die eigenen Sicherheitsbedürfnisse tatsächlich genauer umgesetzt werden. Einem „normalen“ Benutzer ist dies allerdings nicht zuzumuten. Aus diesem Grund können stattdessen auch die generischen Sicherheitsstufen „Hoch“, „Mittel“, „Niedrig“ und „Sehr niedrig“ benutzt werden.[13]

[13] Die Darstellung bezieht sich auf den Internet Explorer 5.

Darüber hinaus kennt der Browser das Konzept unterschiedlicher Sicherheitszonen. Dabei erfolgt die Einteilung der Rechner anhand ihrer Adresse und des verwendeten Protokolls in:

Arbeitsplatz Nur der lokale Rechner ist Mitglied dieser Zone. Sie ist im Internet Explorer nicht konfigurierbar. Dazu sind spezielle Administrationsprogramme erforderlich, welche auch neue Zonen einrichten können.

lokale Intranetzone Diese Zone ist so voreingestellt, daß all diejenigen Rechner in ihr Mitglied sind,

- deren URL keine Punkte enthält (z.B. `http://rechner/`), und die keiner anderen Zone zugeordnet sind,
- die nicht über einen Proxyserver zugegriffen werden, oder
- auf die über das Common Internet Filesystem (CIFS, auch bekannt als NetBIOS) zugegriffen wird (z.B. `\\rechner\freigabe\datei.txt`).

Internetzone Diese Zone enthält alle Rechner, die keiner anderen Zone zugeordnet sind.

Zone für vertrauenswürdige Sites In diese Zone werden Rechner nicht automatisch eingetragen. Es können aber manuell Rechner eingetragen werden, für die weniger restriktive Einstellungen gelten sollen, als dies für Rechner der Internetzone der Fall wäre.

Zone für eingeschränkte Sites Diese Zone stellt das Gegenteil der „Zone für vertrauenswürdige Sites“ dar. Sie ist bestimmt für Rechner, deren Inhalte mit größeren Restriktionen ausgeführt werden müssen, als Inhalte von Rechnern der Internetzone. Auch in sie werden Rechner nicht automatisch eingetragen.

Obwohl die Einordnung prinzipiell automatisch abläuft, so kann der Benutzer auch einzelne Rechner gezielt in bestimmte Zonen einordnen. Dabei ist allerdings zu beachten, daß es prinzipiell möglich ist, Rechner sowohl unter ihrem logischen Namen als auch unter ihrer IP-Adresse einzutragen. Erfolgen diese Einträge in verschiedene Zonen, so wird der jeweilige Rechner unterschiedlich behandelt, je nachdem, ob er über seine IP-Adresse oder mit seinem logischen Namen angesprochen wird (vgl. [Microsoft 99b]).

Für jede Zone können nun die Sicherheitseinstellungen getrennt getroffen werden, womit es prinzipiell möglich ist, das System recht gut auf seine eigenen Bedürfnisse zu konfigurieren. So wäre es z.B. denkbar, den Download von Applets von dem Rechner zu erlauben, auf dem man Homebanking betreibt, von allen anderen aber den Download streng zu untersagen.

Allerdings gibt es doch einige Probleme, die diese Vorteile wieder in Frage stellen. So sind die Voreinstellungen des Browsers alles andere als sicher. Auch die Möglichkeit, das Schutzbedürfnis statt in vielen Einzeleinstellungen in „Hoch“, „Mittel“, „Niedrig“ und „Sehr niedrig“ zu spezifizieren, wird dadurch zur Falle für den arglosen Benutzer, daß die Ausführung von Applets und Skripten auch in der Einstellung „Hoch“ aktiviert ist (vgl. [Microsoft 99b]).

Der einzige Weg zu wirklich sicheren Einstellungen besteht somit darin, alle Einstellungen von Hand auf die gewünschten Werte zu setzen. Auch finden sich nicht alle relevanten Einstellungen unter „Ansicht→Internetoptionen→Sicherheit“, auch unter „Erweitert“ sind u.a. mit „Zählen der übertragenen Seiten aktivieren“, „Auf zurückgezogene Serverzertifikate prüfen“, „Auf zurückgezogene Zertifikate von Herausgebern prüfen“, „Bei ungültigen Site-Zertifikaten warnen“ Einstellungen zu finden, die ein sicherheitsbewußter Benutzer besser überprüfen sollte.

Allerdings sind damit immer noch nicht alle Probleme behoben. So existiert kein Menüpunkt, um die Zone „lokaler Rechner“ zu konfigurieren, womit aktive Inhalte von allen Seiten, die lokal auf dem Rechner gespeichert sind, oder sich auf einer im Rechner eingelegten CD-ROM befinden, immer ausgeführt werden können. Dies mag auf den ersten Blick harmlos erscheinen, da die Seiten ja erst einmal durch den Benutzer heruntergeladen werden müssen. Tatsächlich aber wurden in der Vergangenheit immer wieder Angriffe demonstriert, die es erlaubten, den Browser davon zu überzeugen, eine Seite würde lokal geladen, obwohl sie tatsächlich aus dem Internet stammte. Solange nicht sichergestellt ist, daß derartige Angriffe nicht mehr vorkommen können, führen sie das gesamte Zonenkonzept ad absurdum.

Zertifikate

Sowohl Navigator als auch Internet Explorer räumen aktiven Inhalten weitergehende Rechte ein, wenn sie von jemandem digital signiert wurden, der ein Zertifikat einer Instanz besitzt, deren Root-Zertifikat in einer Datenbank des jeweiligen Browsers eingetragen ist. Beispiele für solche Instanzen wären z.B. die Browserhersteller selber oder eine kommerzielle Instanz wie VeriSign. Da neben den Organisationen, die ein Abkommen mit dem Browserhersteller haben, noch eine Vielzahl weiterer Instanzen existiert, erlauben es die Browser dem Benutzer, bei Bedarf weitere Rootzertifikate hinzuzufügen.

Liegt eine gültige Signatur vor, so wird ein Active X-Control im Internet Explorer heruntergeladen und ausgeführt[14], während ein Java-Applet im Netscape Navigator das Recht erhält, Funktionen aufzurufen, um erweiterte Rechte zu erhalten, die ihm sonst durch die „Sandbox“ verwehrt würden (z.B. Dateizugriff). In beiden Fällen findet in den Standardeinstellungen eine Sicherheitsabfrage durch den Browser statt.

Die Gültigkeit einer Signatur wird dabei dann als gegeben angenommen, wenn

- der zur Signatur verwendete Schlüssel mit demjenigen identisch ist, der im Zertifikat des Signierenden eingetragen ist,
- das Zertifikat des Signierenden von einer CA ausgestellt wurde, deren Root-Zertifikat in der Datenbank des Browsers eingetragen ist, und
- das Zertifikat mit dem Schlüssel signiert wurde, der im Root-Zertifikat eingetragen ist.

Ob das jeweilige Zertifikat abgelaufen ist oder zurückgezogen wurde, wird dabei nicht zwangsläufig überprüft. Microsoft führte z.B. schon im Internet Explorer 3.0 mit Authenticode eine Technologie für Zertifikate ein. Erst seit Authenticode 2.0 (dem Internet Explorer 4.0) sind aber besagte Überprüfungen prinzipiell möglich (vgl. [Microsoft 97]). Auch im Internet Explorer 5 sind sie standardmäßig abgestellt (vgl. [Microsoft 99a]).

In diesem Zusammenhang stellt sich die Frage, welche Aussage eine gültige Signatur beinhaltet. Tatsächlich lautet diese nicht „Diese Datei wurde signiert und ist deshalb sicher“ sondern im besten Falle „Eine Person, die einer bestimmten CA bekannt ist, hat diese Datei signiert“. Schließlich ist noch nicht einmal sicher, daß der Signierende mit dem Programmautor identisch ist. Er könnte auch ein Applet/Control gefunden, für nützlich oder interessant erachtet und selber signiert haben, damit die Betrachter seiner Webseiten nicht jedesmal Fehlermeldungen wegen der fehlenden Signatur erhalten.

[14] Um in den Standardeinstellungen ausgeführt zu werden, muß es darüber hinaus als „Sicher für Scripting“ markiert sein. Da dies aber durch den Programmierer des Controls geschieht, kann es in diesem Zusammenhang vernachlässigt werden.

In so einem Falle kann er selber nicht genau wissen, was besagte Datei tut, da er höchstwahrscheinlich nicht den Quelltext kennt.

Nehmen wir aber einmal an, der Signierende sei mit dem Programmautor identisch. Nun bedeutet die Existenz eines Zertifikats, daß diese Person bei einer Zertifizierungsinstanz vorstellig geworden ist und die Ausstellung des Zertifikats beantragt hat. Die Zertifizierungsinstanz führt aber in der Regel weder Überprüfungen der Programmierkenntnisse des Autors durch, noch verlangt sie Leumundszeugnisse. Damit kann und wird sie nicht garantieren, daß besagte Person niemals Dateien signieren wird, die unerwünschte Nebeneffekte haben.[15] Die einzige Aussage, die sich dann im hier betrachteten Kontext aus einem Zertifikat ableiten läßt, ist die Identität des Zertifizierten.

Damit kann eine Signatur keine Aussage über die Gefährlichkeit des signierten Objektes machen. Sie erlaubt es allenfalls, im Nachhinein einen Verantwortlichen für ein maliziöses Objekt zu benennen. Dabei wird allerdings vorausgesetzt, die im Zertifikat enthaltenen Informationen können nach einem Vorfall rekonstruiert werden, und die Zertifizierungsinstanz hat die Identität des Zertifizierten sorgfältig überprüft.

Beide Voraussetzungen sind nicht immer erfüllt. Schließlich könnte ein maliziöses Skript die Festplatte löschen und damit alle Spuren beseitigen. Ein Applet könnte seine unerwünschte Funktion erst mit einer gewissen zeitlichen Verzögerung aktivieren, wodurch es schwierig wird, einen Bezug zwischen Aktion und Urheber herzustellen. Auch eine mangelhafte Überprüfung eines Antragstellers durch eine Zertifizierungsinstanz ist schon vorgekommen. So konnte sich Kevin McCurley von [Digicrime 98] durch VeriSign Zertifikate ausstellen lassen, die als E-Mail-Adresse „root@localhost“ oder „Administrator“ enthielten, wie Abb. 9.1 zeigt.

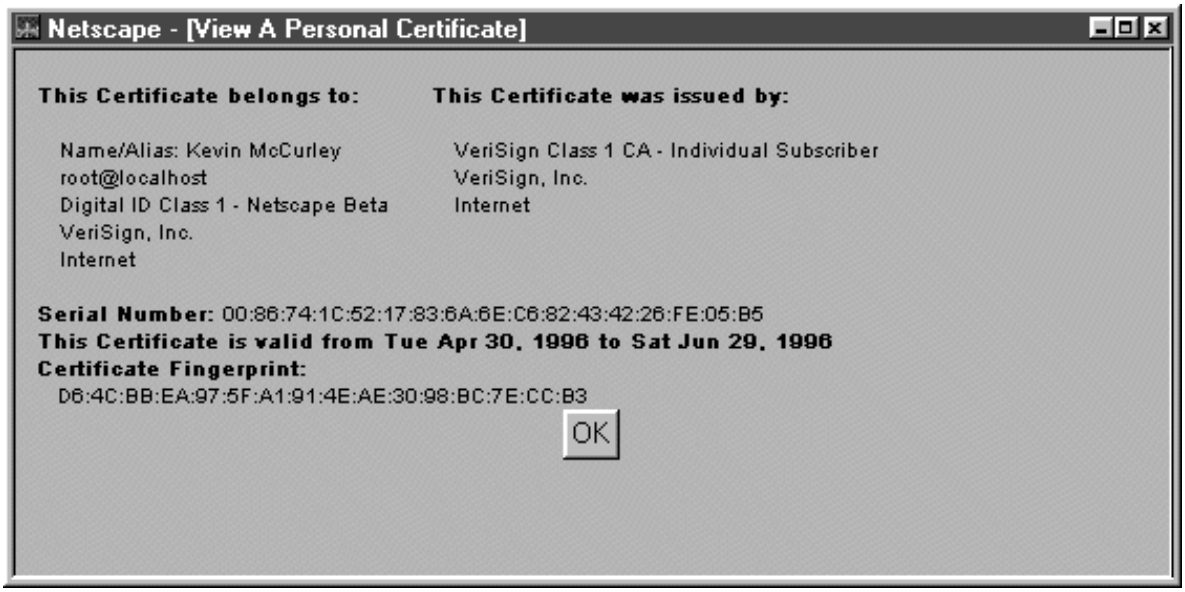


Abbildung 9.1: Zertifikat für root@localhost

[15] Es wäre prinzipiell denkbar, daß Zertifizierungsinstanzen tatsächlich derartige Garantien abgeben. Dies ist aber heute normalerweise nicht gegeben.

On-Access-Virenscanner

Seit einiger Zeit existieren sogar HTML-Seiten, die nicht mehr sofort eine Schadensfunktion ausführen, sondern ihren Programmcode in andere lokal vorhandene Dateien schreiben. Dabei infizieren diese „HTML-Viren“ nicht nur HTML-Dateien, sondern z.T. auch Office-Dokumente. Da nun virale aktive Inhalte existieren, liegt der Gedanke nahe, einem Teil der hier beschriebenen Probleme mit Virenscannern zu begegnen, die ständig im Hintergrund laufen und den Zugriff auf als maliziös erkannte Dateien verhindern (On-Access-Scanner).

Leider hat diese Idee einige konzeptionelle Schönheitsfehler. Zum ersten kann sie nur den Zugriff auf aktive Inhalte filtern, die als eigene Datei vorliegen (z.B. Java, Active X). HTML-Seiten, die eingebettete Skripte enthalten, werden vom Browser aber normalerweise direkt in den Hauptspeicher geladen, so daß kein Plattenzugriff erfolgt, den der Scanner blocken könnte. Zum zweiten ist es relativ einfach, neue Angriffe zu schreiben, die vom Scanner noch nicht erkannt werden. Und schließlich muß leider festgestellt werden, daß das Hauptaugenmerk der meisten Scannerhersteller immer noch auf Viren gerichtet ist. Für andere Gefahren fühlen sie sich schlicht nicht zuständig.

Das bedeutet aber nicht, daß der Einsatz eines Virenscanners auf einem Rechner überflüssig wäre, der zum Besuch des Internets genutzt wird. Schließlich können sich nur die wenigsten Benutzer davor zurückhalten, Programme herunterzuladen und auszuführen. In diesem Falle besteht immer die Gefahr der Verseuchung des Rechners mit Viren und Trojanern.

Schutzmechanismen des Betriebssystems

Da es sich gezeigt hat, daß nicht alle Angriffe auf der Anwendungsebene abgehalten werden können, stellt sich die Frage, ob nicht die Konfigurationsmöglichkeiten des Betriebssystems genutzt werden könnten, um Angriffe abzuwehren.

Im Gegensatz zu Windows 95/98 besitzen echte Mehrbenutzer-Betriebssysteme typischerweise eine Rechteverwaltung, die es erlaubt, Benutzer gegeneinander abzuschirmen. So kann ein spezieller Benutzer eingerichtet werden, dessen einziger Zweck der Besuch des Internets ist. Dieser darf als einziger den Browser aufrufen, dafür aber keine anderen Programme ausführen.[16] Er hat damit z.B. nicht das Recht, auf wichtige Dokumente zuzugreifen, die damit nicht durch eine bösartige Webseite ausspioniert oder manipuliert werden können.

Diese Maßnahme allein schützt allerdings nicht vor Angriffen, die darauf abzielen, bestimmte gemeinsam genutzte Ressourcen vollständig zu belegen. Diese „Denial-of-Service“-Angriffe schaffen es z.B., eine vollständige Belegung der Festplatte zu erreichen oder alle verfügbare Rechenzeit zu nutzen.

Dieser Problematik kann allerdings mit einer geeigneten Konfiguration des Systems entgegengewirkt werden. Durch eine geschickte Partitionierung der Platte und die Einführung von „quotas“, d.h. einer Begrenzung des Speichers, den ein bestimmter Benutzer auf der Platte belegen kann, und der Möglichkeit, den einzelnen Prozessen Prioritäten zuzuweisen, kann ein Benutzer sehr effektiv daran gehindert werden, seine Mitbenutzer übermäßig zu beeinträchtigen.

Darüber hinaus kennt man aus der UNIX-Welt das Konzept der „Root-Reserve“. Dies bedeutet, daß das Betriebssystem einen bestimmten Anteil der Systemressourcen normalen Benutzern vorenthält, der dazu dienen soll, im Ernstfall dem Systemadministrator (Root) den nötigen Spielraum zu verschaffen, um die aufgetretenen Probleme zu beheben. So besitzen die meisten Dateisysteme unter

[16] Das gleiche gilt natürlich für Prozesse, die mit seinen Rechten laufen (z.B. da sie von ihm gestartet wurden).

UNIX einen Anteil, der nur von Root beschrieben werden darf. Auch dürfen normale Benutzer die Priorität ihrer Prozesse nur verringern, während Root sie auch erhöhen und damit Prozesse schaffen kann, die so bevorrechtigt sind, daß sie von normalen Benutzerprozessen nicht behindert werden können.

Filternde Proxies

Auch einige Firewallhersteller haben die von aktiven Inhalten ausgehenden Gefahren erkannt. Darum haben sie HTTP-Proxies entwickelt, die die von ihnen weitergeleiteten Webseiten auf aktive Inhalte untersuchen und diese gegebenenfalls entfernen.

Hierbei ist prinzipiell eine relativ feine syntaktische Filterung bestimmter Technologien (Java, JavaScript,...) möglich, wie auch die Entscheidung, von welchen URLs aktive Inhalte erlaubt sind und von welchen nicht.

Allerdings entstehen dabei auch Probleme. So ist es einem Proxy z.B. prinzipiell nicht möglich, verschlüsselte Datenströme wie HTTP über SSL zu untersuchen. Auch unbekannte Archivformate[17] sind ein Problem für filternde Proxies.

Darüber hinaus erschweren die Browserhersteller dem Proxy die Filterung, indem sie immer neue Aufrufmethoden definieren. So benutzt z.B. der Navigator den ungenormten Tag „Embed“ anstelle von „Applet“ oder „Object“. Auch erlauben es die spezifischen Erweiterungen von Java und JavaScript in den einzelnen Browsern, z.B. aus einem Java-Applet heraus JavaScript-Code auszuführen bzw. in einem Skript Java-Methoden aufzurufen. Dies macht den Ansatz, als gefährlich erachtete Technologien zu blockieren, „harmlose“ aber durchzulassen, undurchführbar.

Auch die Idee, man könne gezielt JavaScript-Funktionen ausfiltern, die als schädlich erachtet werden, solche aber durchlassen, die „ungefährlich“ seien, wird dadurch vereitelt, daß Möglichkeiten existieren, selbstmodifizierenden Code zu schreiben. Dieser Programmcode, dessen einzelne Anweisungen als harmlos erscheinen, verändert sich bei seiner Ausführung in solcher Weise, daß in ihm völlig neue Befehle eingesetzt werden, die sonst gefiltert worden wären.

Selbst eine Politik, die darauf basiert, nur als „harmlos“ erkannte Befehle zu erlauben, alle anderen aber zu verbieten, stellt wahrscheinlich keinen Ausweg dar. Das Problem ist darin zu suchen, daß die meisten Befehle nicht von sich aus „harmlos“ oder „gefährlich“ sind. Erst ihre Kombination macht z.B. aus einer harmlosen Rechenaufgabe einen polymorphen Code, der Befehle ausführt, die eigentlich verboten wurden.

Eine „sichere“ Untermenge der JavaScript-Befehle wäre vermutlich zu eingeschränkt, um sinnvoll zu sein. Tatsächlich brauchen die meisten Skripte, die heute in Webseiten eingesetzt werden

- Stringvariablen,
- einfache Operatoren (Zuweisung, Konkatenation) und
- einen Befehl, um eine neue Seite zu laden.

Ohne diese Untermenge von Befehlen ist ein sinnvoller Einsatz daher nicht gegeben. Verbietet man nur einen dieser Befehle, so kann man JavaScript auch gleich generell verbieten.

[17] Die Packer Tar, Gzip, Zip und Arj sind allgemein bekannt. Es kann aber z.B. nicht zwangsläufig davon ausgegangen werden, daß auch Rar und Ace unterstützt werden.

Erlaubt man diese Befehle aber, so kann folgendes Programm ausgeführt werden (die Notation erfolgt in einer Pseudosprache):

```
a:="bo"
b:=ßcript:"
c:=öm"
d:="java"
e:="()"
f:=d+b+a+c+e              // --> f="javascript:boom()"
Seite_laden(f)
```

Dabei wird letztlich ein Befehl `boom()` ausgeführt, der eigentlich durch den Proxy ausgefiltert werden sollte.

Dieses Beispiel basiert auf der Tatsache, daß bestimmte Browser eine Protokollangabe „javascript“ kennen. URLs mit diesem Protokoll werden nicht über das Netz geladen, sondern ausgeführt. Das Programm läßt sich dabei beliebig komplizieren, um zu verhindern, daß eine Analyse des Proxies erkennt, welchen Wert `f` annehmen wird.

Fazit

Trotz der aufgezeigten Schwächen sollten die aufgezeigten Schutzmechanismen genutzt werden. Geeignet miteinander kombiniert können sie die mit aktiven Inhalten verbundenen Risiken deutlich vermindern.

Schon eine geeignete Wahl der Browsereinstellungen kann einen gewissen Schutz bieten. Dabei hängt der zu erreichende Schutzgrad allerdings stark von der Art der Nutzung des Internets ab. Wer dort in erster Linie nach Informationen sucht, kann in der Regel aktive Inhalte und Cookies abschalten und wird nur selten Seiten finden, die ihn zwingen, diese wieder einzuschalten.

Wer allerdings häufig Bestellungen und Bankgeschäfte über das Internet tätigt, der wird bald feststellen, daß er ohne Cookies und aktive Inhalte nicht sehr weit kommt. Um nun aber auf diese Nutzung nicht verzichten zu müssen, bieten sich mehrere Lösungsmöglichkeiten an. Der Internet Explorer könnte so konfiguriert werden, daß alle Zonen restriktive Sicherheitseinstellungen erhalten, wobei in der „Zone für vertrauenswürdige Sites“ die Einschränkungen soweit gelockert werden, daß die gewünschten Anwendungen möglich sind.

Obwohl so im Normalfall nur aktive Inhalte von „vertrauenswürdigen“ Rechnern ausgeführt werden, hat dieser Ansatz Schönheitsfehler. Zum einen sind Angriffe bekannt geworden, die das Zonenkonzept aushebeln, zum anderen stellen aktive Inhalte nicht die einzige Gefahr im Internet dar. Auch ein eventuell heruntergeladener Trojaner oder Virus könnte einen finanziellen Schaden beim Benutzer hervorrufen. Dies wäre z.B. der Fall, wenn er übermittelte Kreditkartendaten mitliest oder beim Online-Banking kurz vor dem Absenden eines Auftrages die Tastatur sperrt, die eingegebenen Werte manipuliert und dann den Auftrag selber abschickt. Aus diesem Grunde sollte erwogen werden, getrennte Systeme für verschiedene Aufgaben zu nutzen. Ein System könnte dabei für Büroarbeiten (Textverarbeitung, Steuererklärung, ...), eines zur Unterhaltung (Spiele, Browsen beliebiger Webseiten, ...) und eines für Online-Transaktionen dienen.

Unter einem „System“ ist dabei entweder ein eigener Rechner oder eine Festplatte in einem Wechselrahmen zu verstehen, die nur bei Bedarf eingeschoben wird. Mehrere Konfigurationen, die jeweils

von einem Bootmanager gestartet werden, scheiden aus, da eine Wechselwirkung zwischen ihnen nicht ausgeschlossen werden kann. Zum einen könnte eine Konfiguration mit einem Virus verseucht sein, der die gesammte Festplatte löscht, zum anderen existieren Programme, die den Zugriff auf die jeweils inaktiven Partitionen selbst dann ermöglichen, wenn sie von einem anderen Betriebssystem angelegt wurden.

Zertifikate können im Rahmen dieser Betrachtungen nur einen minimalen Beitrag leisten. Wie oben angeführt, sind sie in der gegenwärtigen Form kaum dazu zu gebrauchen, als Grundlage für automatische Sicherheitsentscheidungen zu dienen. Lediglich, wenn Zertifikataussteller und Zertifkatnehmer bekannt sind und diese Information beim Akzeptieren von aktiven Inhalten auch durch den Benutzer überprüft wird, kann ein Zertifikat als Grundlage für eine manuelle Entscheidung dienen. Dieses Szenario wird aber wohl nur in wenigen Fällen zutreffen.

On-Access-Virenscanner sollten unabhängig von der hier betrachteten Problematik auf jedem Rechner installiert sein, auf dem gelegentlich neue Software installiert, oder Dokumente in Formaten empfangen werden, die Makros enthalten könnten. Um vor den Gefahren aktiver Inhalte zu schützen, sind Virenscanner allerdings kaum geeignet.

Alternative Betriebssysteme könnten die Sicherheit eines Rechners erhöhen. Ist es aber z.B. nötig, eine Webseite aufzusuchen, die speziell für den Internet Explorer und Active X ausgelegt ist, so wird kein Weg an Windows 95/98 oder NT vorbeiführen.

Filternde Proxies können schließlich eine zusätzliche Verteidigungslinie bilden, die einen Teil der Angriffe verhindert, bevor aktive Inhalte überhaupt in den Browser gelangen. Sie sind daher eine sinnvolle Ergänzung zu anderen Schutzmaßnahmen, sollten aber nicht als alleiniges Mittel eingesetzt werden, da sie oft nicht jede Art aktiver Inhalte ausfiltern können.

Abschließend soll noch ausgeführt werden, daß technische Maßnahmen alleine nicht ausreichend sind. Nur wenn der Benutzer ihren Sinn auch versteht und die Wirkung der getroffenen Maßnahmen durch sein Verhalten unterstützt, kann ein ausreichender Grad an Schutz erreicht werden.

9.2.5 Zwischenstand

Wie die obige Diskussion zeigt, stellen aktive Inhalte eine reale Bedrohung dar. Ein Verzicht auf sie macht allerdings den Zugang auf bestimmte Webseiten schwer bis unmöglich. Obwohl viele Ansätze existieren, die Gefahren aktiver Inhalte zu reduzieren, stellt sich doch immer wieder heraus, daß Sicherheit mit Disziplin und Verzicht auf Features erkauft werden muß.

Es muß dabei im Einzelfall entschieden werden, wie die Abwägung zwischen Sicherheit und unbeschwertem Surfen getroffen wird. Es wäre allerdings schon viel gewonnen, wenn der Benutzer sich dabei wirklich bewußt wäre, welche Konsequenzen seine Entscheidung haben kann. Bedauerlicherweise hat sich in der Vergangenheit gezeigt, daß dieses Wissen nicht sehr weit verbreitet ist.

9.3 Mobiler Code (Agenten)

9.3.1 Die Idee

Agenten sind Programme, die selbständig für einen Benutzer handeln sollen. Ein Beispiel dafür ist ein Programm, das lernt, nach welchen Kriterien der Benutzer seine E-Mails liest oder löscht, und das

ihm nach einer gewissen Lernphase diese Arbeit abnimmt. Auf diese Weise bekommt der Anwender nur noch die Post zu sehen, die ihn wirklich interessiert.

Im Zeitalter des E-Commerce hat dieses Konzept eine neue Ausprägung bekommen. Hierbei handelt es sich um autonom handelnde Programme, die sich im Internet von Rechner zu Rechner bewegen und dort im Namen des Anwenders agieren. Da diese Agenten nicht mehr an einen Rechner gebunden sind, nennt man sie „mobil“.

Eine typische Anwendung könnte dabei die Buchung einer Reise im Internet sein. Hierzu wird ein Agent gestartet, dem die gewünschten Eigenschaften einer solche Reise eingegeben werden (z.B. Ziel, Preis, Datum, Unterkunft, Unterhaltungsprogramm, Gewichtung der einzelnen Faktoren . . .).

Der Agent stellt nun eine Verbindung zu einem Rechner eines Reisebüros im Internet her und erstellt dort eine identische Kopie. Anschließend beendet sich die Instanz auf dem Rechner des Benutzers. Dieser Vorgang wird „Migration“ genannt.

Auf dem Rechner des Reisebüros wird nun der migrierte Agent ausgeführt. Dieser fordert ein Angebot ab und migriert dann zum nächsten Rechner eines anderen Reisebüros, wo sich der Vorgang wiederholt.

Nachdem der Agent so eine gewisse Anzahl von Angeboten eingeholt hat, überprüft er anhand der ihm mitgegebenen Kriterien, welches Angebot den Anforderungen am besten entspricht. Dieses teilt er entweder dem Benutzer mit oder bucht es automatisch.

Für letztere Funktion könnte der Agent eine Form „elektronischen Geldes“ mit sich führen. Denkbar wären z.B. Kreditkarteninformationen, „elektronische Münzen“ wie bei DigiCash (vgl. [Chaum 92]) oder kryptographische Schlüssel sein, die es erlauben, eine Bestellung digital zu signieren.

9.3.2 Die Probleme

Bisher sind mobile Agenten nicht mehr als ein ambitioniertes Forschungsprojekt. Allerdings läßt sich wohl schon heute sagen, daß es Leute geben wird, die versuchen werden, diese Technik auf illegale Weise zu ihrer persönlichen Bereicherung zu nutzen. Neuartige Technik hat schon immer neuartige Verbrechen hervorgebracht.

Aus Sicht eines Händlers sind mobile Agenten auf ähnliche Weise problematisch, wie es aktive Inhalte für den normalen Internetbenutzer sind. Auch er muß sich Programme herunterladen, von denen er nicht im Vorwege wissen kann, was sie tun werden. Auch die vorgeschlagenen Basistechnologien sind ähnlich. So favorisiert ein Teil der Forscher Java, während andere Skriptsprachen wie z.B. Agent Tcl (vgl. [GrCyKoRu 97]) verwenden.

Eine andere Gefahr ist der Diebstahl oder die Manipulation eines Agenten auf dem Übertragungsweg. Diese ist insbesondere dann gegeben, falls er in der Lage ist, selbständig eine Buchung vorzunehmen. In diesem Fall enthält er zwangsläufig Daten, die es erlauben, im Internet zu bezahlen. Es wäre dann nur eine Frage der Zeit, bis kriminelle Elemente versuchen würden, in den Besitz dieser Daten zu gelangen.

Auch der Wirtsrechner kann eine mögliche Gefahrenquelle sein. Schließlich könnte auch der Händler ein Interesse haben, an das Geld des Agenten zu kommen, ohne dafür eine Ware zu liefern. Er könnte aber auch versuchen, einen Auftrag zu erhalten, ohne das Angebot abgegeben zu haben, das die Wünsche des Benutzers am besten trifft.[18] Hierbei könnte er sich zu Nutze machen, daß der Agent

[18] Sein Angebot könnte z.B. zu teuer sein.

die Angebote seiner Mitbewerber gespeichert haben muß, um am Ende seine Entscheidung treffen zu können. Wenn diese Daten nun gelöscht oder manipuliert würden, so würde das seiner Entscheidung eine völlig falsche Grundlage geben. Ergänzt werden könnte dieser Angriff noch um eine falsche Ausführung des Agenten, die direkt zu der Stelle im Code springt, an der die Suche beendet und ein Geschäft getätigt wird.

9.3.3 Die Lösungsansätze

Es werden Lösungsansätze diskutiert, um diese Gefahren einzudämmen. Das Problem „bösartiger Agenten“ wird dabei auf eine ähnliche Weise behandelt, wie das Problem maliziöser Webseiten oder das der Trojaner. Auch hier wird der Einsatz von „Sandboxen“ und eine restriktive Ressourcenzuteilung durch das Betriebssystem vorgeschlagen. Darüber hinaus wird angedacht, Programme zu verifizieren. [FeLe 97] beschreibt eine Technik, bei der Programme den Beweis dafür enthalten, bestimmte Eigenschaften zu erfüllen (Proof Carrying Code).

Obwohl es nicht möglich ist, maschinell zu beweisen, daß ein Programm eine bestimmte Eigenschaft erfüllt, kann dies für konkrete Programme doch oft manuell geschehen. Existiert so ein Beweis, so ist eine maschinelle Überprüfung durchaus realisierbar. Es wäre daher zwar etwas komplizierter, Agenten zu schreiben, dafür wären aber bestimmte vorher definierte Eigenschaften am Zielrechner überprüfbar.

Für den Schutz von Datenpaketen gibt es seit langem Lösungsansätze. Diese laufen in der Regel darauf hinaus, daß die Daten vor dem Absenden signiert und chiffriert werden. Auf diese Weise kann nur der Empfänger sie lesen und anhand der digitalen Unterschrift überprüfen, ob die Daten auch wirklich unverändert bei ihm angekommen sind.

Deutlich schwieriger zu lösen ist das Problem maliziöser Wirtsrechner. Die naheliegendste Lösung wäre warscheinlich, den Wirtsrechnern einfach zu vertrauen. Dies könnte erreicht werden, indem „vertrauenswürdige Einkaufszentren“ geschaffen würden [Nedon 98]. Hierbei würde es sich um Dienstleister handeln, die sowohl Händlern als auch Käufern eine vertrauenswürdige Infrastruktur[19] zur Verfügung stellen und auch dafür haften, daß Agenten unverfälscht ausgeführt werden. Eine regelmäßige Überprüfung durch eine unabhängige Instanz würde dabei sicherstellen, daß diese Dienstleister ihren Pflichten auch gewissenhaft nachkommen.

Einen anderen Ansatz verfolgen T. Sander und C.F. Tschudin (vgl. [Sander 97b, Sander 97c, Sander 98, Sander 97a]). Sie beschreiben verschlüsselte Funktionen, die ausgeführt werden, ohne daß der Wirtsrechner ihren Inhalt kennt. Auf diesem Wege könnten Funktionen vertrauenswürdig in einer nichtvertrauenswürdigen Umgebung ausgeführt werden.

Die Idee dabei ist folgende: Alice will von Bob eine Funktion $\mathtt{f}$ ausführen lassen, die als Eingabe einen von Bob bestimmten Wert x erhält. Bob soll aber nicht wissen, um welche Funktion es sich handelt, da er das Ergebnis der Berechnung sonst beeinflussen könnte.

Um dies zu erreichen, wählt Alice eine weitere Funktion $\mathtt{s}$ und sendet $\mathtt{s}\circ\mathtt{f}$. Bob berechnet nun $y = (\mathtt{s}\circ\mathtt{f})(x)$ und sendet das Ergebnis y zurück an Alice, die $z = \mathtt{s}^{-1}(y) = (\mathtt{s}^{-1}\circ\mathtt{s}\circ\mathtt{f})(x) = \mathtt{f}(x)$ ermittelt.

Dieses Verfahren funktioniert prinzipiell, wenn es sich bei $\mathtt{s}$ um eine lineare Abbildung (Homomorphismus) handelt, d.h. es existiert eine Abbildung $\oplus$ für die gilt: $\mathtt{s}(x+y) = \mathtt{s}(x)\oplus\mathtt{s}(y)$. Darüber hinaus muß $\mathtt{s}^{-1}$ für Alice mit geringem Aufwand berechenbar sein und aus der Berechnung von $\mathtt{s}(x)\oplus\mathtt{s}(y)$ dürfen x und y nicht ableitbar sein.

[19] „Vertrauenswürdig“ bedeutet hier: „Der Dienstleister tut alles in seiner Macht stehende, um ordnungsgemäße Abläufe zu gewährleisten und sich das Vertrauen seine Kunden zu erhalten.“

Die Autoren wenden das besprochene Verfahren auf polynomielle und rationale Funktionen an und erweitern es schrittweise, um einigen Angriffen entgegen zu wirken. In ihren Papieren drücken sie die Hoffnung aus, das Verfahren so verallgemeinern zu können, daß jedes beliebige Programm auf eine Funktion `f` abbildbar wäre. Mit einem praktischen Einsatz dieses Verfahrens in absehbarer Zukunft ist aber wohl nicht zu rechnen.

9.4 Fazit

Sicherheit bedeutet immer eine Einschränkung der Funktionalität.[20] Im Fall der aktiven Inhalte gingen die Entwickler in erster Linie daran, Funktionalität bereitzustellen, ohne dabei an die Sicherheit zu denken. Mit Ausnahme von Java sind alle hier besprochenen Technologien erst nachträglich um Sicherheitskonzepte ergänzt worden. Die Erfahrung zeigt, daß derartige Versuche in der Regel fehlschlagen, da sie zu tiefe Einschnitte in bereits genutzte Systeme erfordern würden.

Bei den mobilen Agenten läßt sich positiverweise feststellen, daß Sicherheitsüberlegungen bereits angestellt werden, bevor die Technik entwickelt und auf den Markt gebracht wurde. Es bleibt daher zu hoffen, daß hier nicht wieder die selben Fehler gemacht werden. Allerdings muß klar gesehen werden, daß Sicherheit ihren Preis hat. Auch die oben aufgeführten Lösungsansätze bedeuten bei ihrer Realisierung einen zusätzlichen Aufwand, der sich für den Implementierenden in Form von Kosten bemerkbar macht. Aus diesem Grunde sollte man immer vorsichtig sein, wenn eine neue Technologie als sicher angepriesen wird. In der Regel stellen sich die Probleme erst im laufenden Betrieb heraus.

Literaturverzeichnis

[Box 98] Don Box, „Essential COM“, Addison-Wesley, 1998

[Chaum 92] David Chaum, „Archieving Electronic Privacy“, Scientific American, August 1992,S.96-101,
Online unter `http://ganges.cs.tcd.ie/mepeirce/Project/Chaum/sciam.html`

[Digicrime 98] Webseiten von Digicrime
Online unter `http://www.digicrime.com`

[EnLe 99] Andreas Engel, Andreas G. Lessig,
„Internetgestützte Angriffe und ausgewählte Gegenmaßnahmen“, Diplomarbeit, 1999,
Online unter `http://agn-www.informatik.uni-hamburg.de/People/1lessig/dipl.ps.gz`

[FeBaDeWa 97] E. Felten, D. Balfanz, D. Dean, D. Wallach, „Web Spoofing: An Internet Con Game“,
Online unter `http://www.cs.princeton.edu/sip`

[FeLe 97] J. Feigenbaum, P. Lee, „Trust Management and Proof-Carrying Code in Secure Mobile-Code Applications“, Proceedings of the DARPA Workshop on foundations for secure mobile code, Monterey, USA, März 1997

[20] Ein System, in dem jeder Benutzer/Prozeß jede Aktion durchführen darf, ist per definitionem nicht sicher. Erst wenn eine Instanz existiert, die Zugriffe anhand von Regeln auf ihre Rechtmäßigkeit überprüft, kann ein gewisser Grad an Sicherheit erreicht werden.

[GrCyKoRu 97] R. Gray, G. Cybenko, D. Kotz, D. Rus, „Agent Tcl: A flexible and secure mobile-agent system“, in: W. Cockayne, M. Zypa (Hrsg.): Mobile agents, Explanations and Examples, Manning Publishing, 1997,
Online unter http://www.cs.dartmouth.edu/~agent/papers/chapter.ps.Z

[Guninski 98] Georgi Guninski, „Browser exploits“,
Online unter http://www.nat.bg/~joro/

[MGFe 99] Gary McGraw, Edward W. Felten, „Securing Java“, 2. Auflage, John Wiley & Sons, 1999

[Microsoft 97] „Microsoft Announces Innovative Security Zones“, Microsoft 3.Juni 1997,
Online unter http://www.microsoft.com/PressPass/press/1997/jun97/securipr.htm

[Microsoft 99a] „Chapter 6 - Digital Certificates“, Microsoft,
Online unter http://technet.microsoft.com/cdonline/content/complete/Internet/Client/IE/reskit/ie5/ch06digi.htm

[Microsoft 99b] „Chapter 7 - Security Zones and Permission-Based Security for MS Virtual Machine“, Microsoft,
Online unter http://technet.microsoft.com/cdonline/content/complete/Internet/Client/IE/reskit/ie5/ch07zone.htm

[Microsoft 99c] „Microsoft Security Management Architecture White Paper – 3. Establishing Trust on the Web“, Microsoft,
Online unter http://www.microsoft.com/technet/intranet/whpapers/secure/secure03.htm

[Nedon 98] Jens Nedon, „Sicherheitsmechanismen für Mobile Agenten in elektronischen Dienstemärkten“, Studienarbeit am Fachbereich Informatik der Universität Hamburg, 27. August 1998

[Newsticker 99] „Fünf Jahre Hausarrest für Falschmeldung im Web“, Heise Newsticker 31.8.1999,
Online unter http://www.heise.de/newsticker/data/cp-31.08.99-000/

[Sander 97a] T. Sander, C.F. Tschudin, „Protecting Mobile Agents against malicious Hosts“, in: G. Vigna (Hrsg.), „Mobile Agents and Security“, LNCS 1419, Springer-Verlag Berlin,
Online unter http://www.icsi.berkeley.edu/~tschudin/ps/ma-security.ps

[Sander 97b] T. Sander, C.F. Tschudin, „On the Cryptographic Protection of Mobile Code“, Folien zu: Workshop on Mobile Agents and Security, University of Maryland Baltimore County, 27./28. Oktober 1997
Online unter http://www.icsi.berkeley.edu/~tschudin/mobag-prot-slides.ps.gz

[Sander 97c] T. Sander, C.F. Tschudin, „Towards Mobile Cryptography“, November 1997,
Online unter http://www.icsi.berkeley.edu/~tschudin/ps/ieee-sp98.ps.gz

[Sander 98] T. Sander, C.F. Tschudin, „On Software Protection via Function Hiding“, Eingereicht in: Second International Workshop on Information Hiding, Dezember 1998,
Online unter http://www.icsi.berkeley.edu/~tschudin/ps/iwh98.ps.gz

Kapitel 10

Public-Key-Zertifizierung und Public-Key-Infrastrukturen

Britta Liedtke und Ingmar Camphausen
DFN-PCA
Universität Hamburg
Fachbereich Informatik

Zusammenfassung

Mittels Verfahren der Public-Key-Kryptographie kann man den Grundbedrohungen einer sicheren elektronischen Kommunikation begegnen. Voraussetzung dafür ist das Vorhandensein einer entsprechenden Zertifizierungsinfrastruktur, die öffentliche Schlüssel in nachprüfbar authentischer Form bereitstellt. Ein wesentlicher Bestandteil solcher Infrastrukturen sind Zertifizierungsinstanzen, die die Unverfälschtheit von Public Keys und deren jeweilige Bindung an einen bestimmten Namen durch digitale Zertifikate bestätigen. Derartige Instanzen oder Infrastrukturen gibt es in Deutschland bereits seit einigen Jahren, auch in Europa sind sie zur Zeit im Aufbau, und mittlerweile haben auch der deutsche Gesetzgeber und die EU einen rechtlichen Rahmen für die Tätigkeit von Zertifizierungsstellen geschaffen.

10.1 Einleitung

Die Bereitstellung und der Austausch elektronischer Informationen, die eine immer größere Bedeutung erlangen, finden häufig ungeschützt in unsicheren Netzen auf unsicheren Netzwerkprotokollen statt. Diese grundsätzlichen Schwachpunkte, die sich nicht so einfach beheben lassen, erfordern die Einführung effizienter Maßnahmen, um die grundlegenden Sicherheitsanforderungen wie Vertraulichkeit und Integrität der Informationen in der Praxis zu realisieren.

Nachdem im folgenden Kapitel 10.2 zunächst die grundlegenden Begriffe und Probleme beim Einsatz von Public-Key-Verfahren geklärt werden, sollen im Anschluß in Kapitel 10.3 Lösungsmöglichkeiten mittels Zertifizierung der Schlüssel aufgezeigt werden. Der entscheidenden Rolle, die Zertifizierungsinstanzen dabei zukommt, trägt Kapitel 10.4 Rechnung, der ihre grundlegenden Dienste, die Besonderheiten ihrer Arbeit und den Unterschied zu Trustcentern darlegt.

Kapitel 10.5 behandelt Public-Key-Infrastrukturen (PKI), jene in denen Zertifizierungsinstanzen eine wesentliche Rolle spielen. Dabei werden Aspekte der Skalierbarkeit ebenso erörtert wie Cross-Zertifizierungen und die Vorzüge des „Web of Trust“.

Das folgende Kapitel gibt einen Überblick über die in der Praxis der Public-Key-Anwendung relevanten Protokolle, Standards und Applikationen, sowie die bereits existierenden Infrastrukturen und deren rechtlichen Rahmen.

Der Beitrag schließt mit einem Ausblick auf die absehbaren weiteren Entwicklungen im Bereich der Public-Key-Standardisierung und -Anwendung.

10.2 Einführung in Public-Key-Verfahren

Einen hohen Stellenwert zum Schutz von Informationen hat seit jeher der Einsatz von *kryptographischen Verfahren*. Dazu gehören in erster Linie Verschlüsselungsverfahren, die die Vertraulichkeit der Informationen gewährleisten sollen und auf die in Abschnitt 10.2.1 kurz eingegangen wird. Dabei ergeben sich zum Teil nicht unerhebliche Probleme bei der Verteilung der kryptographischen Schlüssel, auf die zusammen mit entsprechenden Lösungsansätzen in Abschnitt 10.2.2 näher eingegangen wird. In diesem Zusammenhang erhalten aber auch Signatur-Verfahren (vgl. Abschnitt 10.2.3), die die Authentizität der Informationen gewährleisten sollen, eine zunehmende Bedeutung.

10.2.1 Verschlüsselungsverfahren

Nachdem M. Johns im Rahmen dieses Seminars (vgl. [Johns]) bereits die wichtigsten kryptographischen Begriffe und Verfahren erläutert hat, soll an dieser Stelle nur kurz die symmetrische und asymmetrische Verschlüsselung rekapituliert werden.

Charakteristisch für die **symmetrische Verschlüsselung** ist die Verwendung eines Schlüssels sowohl bei der Verschlüsselung als auch bei der Entschlüsselung: Eine Nachricht, die mit einem Schlüssel K verschlüsselt wurde, kann nur mit dem identischen Schlüssel K wieder in den Klartext umgewandelt werden. Dies bedeutet jedoch, daß pro Kommunikationspartner ein geheimer Schlüssel existieren muß, um vertrauliche Informationen austauschen zu können. Demzufolge haben Kommunikationspartner, die verschlüsselte Nachrichten austauschen wollen, das Problem, daß sie vor der eigentlichen Kommunikation Maßnahmen einleiten müssen, um den geheimen Schlüssel auf sicheren Wegen auszutauschen.

Bei der **asymmetrischen Verschlüsselung** werden hingegen zwei unterschiedliche – aber mathematisch voneinander abhängige – Schlüssel zum Ver- und Entschlüsseln benutzt, von denen ein Schlüssel veröffentlicht werden kann (*Public Key*), der andere jedoch geheim gehalten werden muß (*Private Key*). Asymmetrische Verfahren werden deshalb auch Public-Key-Systeme oder Public-Key-Verfahren genannt.

Üblicherweise wird der öffentliche Schlüssel (*Public Key*) eines Kommunikationspartners zum Verschlüsseln eines Klartextes benutzt; der Chiffretext kann dann nur mit dem korrespondierenden privaten Schlüssel (*Private Key*) entschlüsselt werden.

Ein Vorteil gegenüber der symmetrischen Verschlüsselung ist der erheblich geringere Bedarf an Schlüsseln. Jeder Benutzer benötigt genau einen privaten Schlüssel zum Entschlüsseln und einen öffentlichen Schlüssel pro Kommunikationspartner. Der entscheidende Vorteil gegenüber der symmetrischen Verschlüsselung ist jedoch, daß hierbei nur der öffentliche Schlüssel (und nicht mehr der geheime) an die

Kommunikationspartner verteilt werden muß. Aber auch die Verteilung des öffentlichen Schlüssels ist nicht ganz unproblematisch und soll im folgenden Abschnitt näher erläutert werden.

10.2.2 Schlüsselverteilung

Gerade bei Public-Key-Verfahren ist die Authentizität der öffentlichen Schlüssel ein wichtiges Kriterium, da sie die Voraussetzung für „Sicherheit“ ist. Allerdings können die öffentlichen Schlüssel bei der Schlüsselverteilung, die häufig auf unsicheren Wegen, z.B. per E-Mail, erfolgt, diversen Angriffen ausgesetzt sein. Man denke nur an das in Abbildung 10.1 dargestellte Szenario: Benutzer A bittet Benutzer B um Zusendung seines Public Keys. Diese Kommunikation wird von einem Angreifer X abgehört, der anschließend den von B abgeschickten Public Key abfängt und seinerseits einen entsprechend gefälschten Public Key an Benutzer A weiterleitet mit dem Vermerk „Hier kommt der Public Key von Benutzer B“. Die Folge dieses gefälschten untergeschobenen Schlüssels ist die Kompromittierung des Verfahrens. Daraus wiederum resultiert ein Verlust der Vertraulichkeit, da in diesem Beispiel Benutzer A nicht mehr vertraulich mit Benutzer B kommunizieren kann.

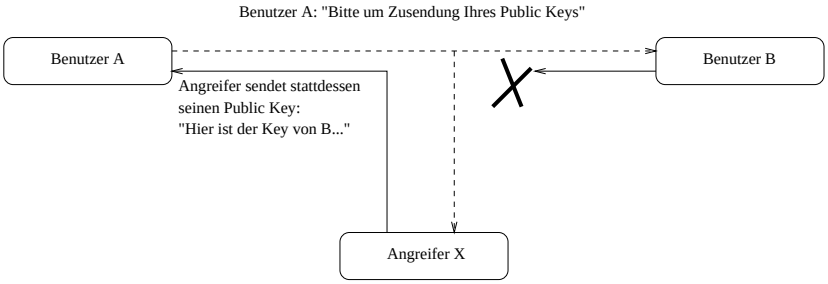


Abbildung 10.1: Ein mögliches Angriffsszenario

Bei der Verteilung des öffentlichen Schlüssels muß deshalb zum einen die Authentizität des Public Keys sichergestellt werden, zum anderen muß der Empfänger des Public Keys, diesen auch auf geeignete Art und Weise verifizieren können. Bei „PGP“ (*Pretty Good Privacy*, vgl. hierzu auch [PGP]) erfolgt die Verifikation des öffentlichen Schlüssels über den Abgleich des Fingerprints, der sich (z.B. auf Visitenkarten) gut verteilen läßt. Gerade im Zusammenhang mit PGP treten immer wieder Mißverständnisse bzgl. der Schlüsselverteilung und Authentizität dieser Schlüssel auf. In den letzten Jahren haben sich PGP-Key-Server zur Verteilung der öffentlichen Schlüssel durchgesetzt. Was jedoch häufig übersehen wird: Diese Key-Server dienen nur der Verteilung der öffentlichen Schlüssel und bieten keine Garantie für die Echtheit der dort vorgehaltenen Schlüssel. Die Überprüfung der Echtheit des Schlüssels bleibt dem jeweiligen Benutzer überlassen.

Der mit Abstand sicherste Weg die öffentlichen Schlüssel zu verteilen, ist diese bei einem persönlichen Kontakt zu übergeben. Innerhalb kleiner, lokal begrenzter Benutzergruppen läßt sich dies vielleicht noch realisieren, aber wie soll der öffentliche Schlüssel mit einem Kommunikationspartner in Australien ausgetauscht werden. Und wer will ständig herumreisen, um seinen öffentlichen Schlüssel an die Kommunikationspartner zu verteilen, wenn dafür auch wesentlich einfachere Methoden wie E-Mail zur Verfügung stehen. Demzufolge müssen Lösungen zur gesicherten Schlüsselverteilung gefunden werden. Ein Ansatz ist zunächst, einen sicheren Kanal zur Übermittlung der öffentlichen Schlüssel zu nutzen. Dies kann beispielsweise eine Standleitung sein. Aber bei vielen Kontakten, z.B. nach Australien, läßt sich ein sicherer Kanal nicht immer und nicht überallhin etablieren. In diesem Fall könnte

ein vertrauenswürdiger Dritter[1] eingeschaltet werden. Dies ist letztendlich nur eine Verlagerung des Problems auf eine sichere Übermittlung des öffentlichen Schlüssels an den vertrauenswürdigen Dritten.

Es sind also noch zusätzliche Informationen erforderlich, die die Verifikation vereinfachen und eine nachprüfbare Bestätigung der Authentizität des öffentlichen Schlüssels ermöglichen. Solch eine Bestätigung kann z.B. in Form eines digitalen Zertifikates (vgl. Kapitel 10.3) an den öffentlichen Schlüssel gebunden werden.

10.2.3 Signaturverfahren

Digitale Signaturen stellen ein elektronisches Pendant zu handschriftlichen Unterschriften dar und sollen die eindeutige Zuordnung eines elektronischen Dokumentes zu einem bestimmten Benutzer ermöglichen. Eine elektronisch erzeugte Unterschrift wird als *digitale Signatur* bezeichnet und soll wie eine handschriftliche Unterschrift die Echtheit des unterschriebenen Dokumentes garantieren.

Die Signaturverfahren basieren, wie auch die asymmetrischen Verschlüsselungsverfahren, auf dem Zwei-Schlüssel-Prinzip (Public-Key-Verfahren). Allerdings existieren Unterschiede in der Verwendung der beiden Schlüsselpaare: Der private Schlüssel (*Private Key*) wird zur Erzeugung der Signatur verwendet, die wiederum mit dem öffentlichen Schlüssel (*Public Key*) verifiziert werden kann.

Damit gewährleisten Digitale Signaturen die Authentizität des „unterschriebenen“ Dokuments, denn nur der Signierer kann das Dokument signiert haben und muß im Besitz des *Private Key*s sein. Da die Signatur nur mit dem geheimen privaten Schlüssel (*Private Key*) erzeugt werden kann, ist sie nicht fälschbar und kann auch nicht geleugnet werden, es sei denn, der private Schlüssel wurde kompromittiert. Ein weiteres Merkmal ist, daß ein signierter Text unveränderlich ist, zumindest dahingehend, daß nach erfolgten Änderungen die Signaturprüfung fehlschlägt. Damit gewährleisten Signaturverfahren zwei ganz entscheidende Sicherheitsfunktionen: Zum einen die Integrität (Unversehrtheit) des Dokumentes und zum anderen die Authentizität (Echtheit). Aus diesen Gründen wird häufig die Auffassung vertreten, daß Digitale Signaturen eine höhere Sicherheit bieten als handschriftliche Unterschriften.

10.3 Public-Key-Zertifizierung

Digitale Zertifikate stellen eine kryptographische Bindung zwischen einem Benutzer (einer Identität) und einem Schlüssel her. Dies wird technisch gesehen darüber realisiert, daß eine Digitale Signatur um einen Public Key erzeugt wird. Ein Zertifikat kann damit als elektronische Beglaubigung aufgefaßt werden, die bestätigt, daß der Schlüssel X dem Benutzer Y gehört. Allerdings enthält ein solches Zertifikat keine Aussage über die Vertrauenswürdigkeit des zertifizierten Benutzers/Schlüsselinhabers.

Wenn es darum geht, wer wen zertifiziert, gibt es derzeit zwei verschiedene Ansätze. Zum einen gibt es die Möglichkeit, daß sich die Benutzer gegenseitig zertifizieren, wie z.B. bei PGP („Pretty Good Privacy“). Diese Art der Zertifizierung wird im folgenden Abschnitt 10.3.1 näher erläutert. Zum anderen können lokale Zertifizierungsinstanzen eingerichtet werden, die die einzelnen Benutzer zertifizieren, worauf in Abschnitt 10.3.2 näher eingegangen wird.

[1] Dies kann ein Kurier als Überbringer oder auch ein sicherer Server zum Abrufen der öffentlichen Schlüssel sein.

10.3.1 Zertifikate von Benutzern

Die gegenseitige Zertifizierung durch andere Benutzer hat sich bei PGP durchgesetzt und erfolgt auf sogenannten „Key Signing Parties“. Man trifft sich dort, überprüft die Identität der anderen Benutzer und tauscht mit ihnen Informationen (z.B. den Fingerprint) der öffentlichen Schlüssel aus. Diese Vorgehensweise funktioniert recht gut innerhalb kleiner Benutzergruppen, läßt sich aber nicht mit weit verstreuten Benutzergruppen realisieren. Bei dieser Art von Zertifizierung entsteht jeweils (z.B. auf jeder „Key Signing Party“) ein relativ kleines „Web-of-trust“, das in 10.5.4 näher erläutert wird.

Damit ergibt sich das Problem, daß nicht notwendigerweise zu jedem Kommunikationspartner Vertrauenspfade existieren. Des weiteren ist die Qualität der Zertifikate unklar, da man nicht alle Benutzer kennt, und vor allem nicht weiß, nach welchen Kriterien diese andere Benutzer zertifizieren. Auch enthalten die PGP-Zertifikate (PGP 2.6*) keine Aussagen über die Gültigkeit der Zertifikate, und noch problematischer: Ein einmal zertifizierter öffentlicher Schlüssel bleibt solange zertifiziert, wie der Schlüssel nicht widerrufen wurde. Es gibt keinerlei Möglichkeit, ein Zertifikat zurückzuziehen. Auch gibt es keine festen Regeln für die Verteilung der Zertifikate und ein sehr wichtiger Aspekt ist inwieweit andere Kommunikationspartner auf diese Art der Zertifizierung vertrauen. Als Alternative bietet sich das Einrichten von Zertifizierungsinstanzen, sogenannten „Certification Authorities“ (CAs), an.

10.3.2 Zertifikate von Instanzen

Durch die Einrichtung eines vertrauenswürdigen Dritten (im Sinne einer übergeordneten Zertifizierungsstelle oder Instanz) können diese einen hohen Bekanntheitsgrad erlangen, und ihre Zertifikate werden möglicherweise von einem größeren Benutzerkreis anerkannt. Ein Vorteil ist, daß diese Instanzen meist ein allgemein anerkanntes Format für Zertifikate benutzen, z.B. X.509v3 (vgl. hierzu auch [X.509]). Derartige Zertifikate enthalten Versions- und Seriennummern, Gültigkeitsdauer sowie diverse andere Attribute.

Die Einrichtung von Zertifizierungsinstanzen bzw. Zertifizierungsstellen oder „Certification Authorities“ ermöglicht auf jeden Fall feste Vertrauenspfade und den Aufbau einer hierarchischen Struktur, auf die in Abschnitt 10.5.2 näher eingegangen wird.

Gerade bei der Einrichtung von Instanzen stellt sich immer wieder die Frage, wer letztendlich wem vertraut. Grundsätzlich muß der Zertifikatnehmer der CA, bei der er seinen Schlüssel zertifizieren läßt, nicht vertrauen. Er kann sich durchaus aus strategischen Gründen dort zertifizieren lassen, selbst wenn er persönlich ihr nicht vertraut. Vielleicht ist diese CA aber bei vielen *anderen* Nutzern – Kunden oder Kommunikationspartnern dieses Schlüsselinhabers – hoch angesehen, die ihr bzw. ihren Zertifikaten Glauben schenken, weswegen der Nutzer trotz seiner eigenen Vorbehalte die Dienste *dieser* CA in Anspruch nimmt. Der Nutzen, den er von ihrer Zertifizierung seines Schlüssels hat, hängt nicht davon ab, ob *er* dieser CA vertraut, sondern ob die Personen ihr vertrauen, die seinen öffentlichen Schlüssel verifizieren wollen und dafür das Zertifikat dieser CA für seinen Schlüssel benutzen (müssen).

Zertifikate von Instanzen ermöglichen feste Vertrauenspfade und tragen zur Glaubwürdigkeit von Zertifikaten bei. Bei der Etablierung realer CAs werden meist sogenannte „Policy Certification Authorities“ (PCAs) darübergestellt, die die einzelnen CAs zertifizieren, so daß eine hierarchische Struktur (vgl. 10.5.2) entsteht. Diese PCA legt die Zertifizierungsrichtlinien (Policy) fest, die Vorgaben dazu enthalten, wie bei der Überprüfung der Identität vorzugehen ist. Es werden aber auch Richtlinien zum Widerruf von Zertifikaten und Vorschriften zum Schutz der zu Zertifizierungszwecken benutzten Hard- und Software festgelegt.

Ein solches Regelwerk einer PCA (oder einer CA) wird auch Policy genannt und muß von der PCA veröffentlicht werden. Die untergeordneten CAs müssen sich an diese Policy halten und haben die Möglichkeit, die Policy zu übernehmen oder entsprechend ihrer lokalen Gegebenheiten eine eigene Policy zu veröffentlichen, die nicht im Widerspruch zu der der PCA stehen darf. Ein Beispiel für eine Policy ist die der DFN-PCA, die unter [DFN-PCA99] veröffentlicht wurde. Damit hat jeder Benutzer die Gelegenheit, diese Richtlinien genau zu überprüfen und danach zu entscheiden, ob diese CA ein vertrauenswürdiger Dritter sein kann oder auch nicht. Damit ist die Qualität der Policy überprüfbar, z.B. dahingehend, ob ein persönlicher Kontakt stattfinden muß oder ob die Zertifikate online ausgestellt werden. Als Konsequenz ist damit dann auch die Qualität der Zertifikate überprüfbar: Letztendlich hat jeder einzelne Benutzer die Gelegenheit, die CA durch gefälschte oder fehlerhafte Schlüssel bzw. Anträge auf den Prüfstand zu stellen.

10.3.3 Widerruf von Zertifikaten

Ein Grund für den Widerruf eines Zertifikats vor Ablauf der Gültigkeit könnte ein Mißbrauch des zertifizierten Schlüssels sein. Dies könnte z.B. mit dem Zertifizierungs-Schlüssel einer CA passieren, nachdem bekannt geworden ist, daß nicht immer eine korrekte Identitätsprüfung der Benutzer stattgefunden hat. Ein weiterer Grund wäre z.B. die Kompromittierung des privaten Schlüssels. In diesem Fall wäre die Authentizität der Signatur nicht mehr gewährleistet.

Ein wichtiger Punkt in diesem Zusammenhang ist, daß grundsätzlich nur der Zertifizierer (also die CA) ein Zertifikat widerrufen kann. Jeder einzelne Benutzer kann zwar bei seiner Zertifizierungsstelle einen Antrag auf Widerruf stellen (für den es ebenso wie vor der Zertifizierung festgelegte Regeln geben muß), den eigentlichen Widerruf muß jedoch die CA vornehmen.

Die Verteilung der widerrufenen Zertifikate erfolgt meist in CRLs (Certificate Revocation Lists). Dies sind zeitgestempelte, von der CA signierte Listen der widerrufenen Zertifikate (bzw. genauer meist deren Seriennummern), die regelmäßig veröffentlicht werden und frei zugänglich sein müssen, damit die Benutzer darauf zugreifen können. Normalerweise sollte es so aussehen, daß sich der Benutzer die CRL auf seinen lokalen Rechner herunterlädt und seine Anwendungen nicht nur die Signatur bzw. das Zertifikat sowie dessen Gültigkeit überprüfen, sondern auch die „lokale“ CRL.

Ein Vorteil dieses Verfahrens ist, daß die CRLs genau wie die Zertifikate über unsichere Kanäle (z.B. via E-Mail oder FTP) verteilt werden können, da sie von der herausgebenden CA digital signiert sind, so daß evtl. Manipulationen sofort auffallen würden. Eine Einschränkung ist jedoch, daß die Aktualität der CRL nicht unbedingt gewährleistet ist: Einige CAs geben täglich eine CRL heraus, andere wöchentlich bzw. monatlich. Damit ist allerdings nicht gewährleistet, daß nicht doch schon wieder weitere Zertifikate widerrufen wurden, die in keiner veröffentlichten CRL auftauchen, da sie kurz nach der Veröffentlichung widerrufen wurden und erst in der nächsten CRL mit veröffentlicht werden.

Als Alternative bietet sich hier die Einführung eines „Online Revocation Checking“[2] an, bei dem die Anwendung (z.B. ein Browser) vor dem Akzeptieren eines Zertifikats z.B. auf einen Server zugreift, auf dem die CA ihre widerrufenen Zertifikate bereitstellt. Allerdings erfordert dies dann neue Sicherheits-Mechanismen: Der Validierer muß Vertrauen in den Validierungsservice haben, nicht jedoch in den Server, auf dem die Widerrufe bereitgestellt werden.

[2] PKIX definiert ein Protokoll OCSP [OCSP], mit dem solch eine Überprüfung des Status’ des Zertifikats durchgeführt werden kann.

10.4 Zertifizierungsinstanzen

Wodurch zeichnet sich eine Zertifizierungs*instanz* (oft auch als Zertifizierungs*stelle* bezeichnet) gegenüber einer „normalen“ Zertifizierenden aus, wie sie beispielsweise an einer gegenseitigen Zertifizierung zwischen PGP-Nutzern beteiligt sein könnte?

- Eine CA operiert nach *festgelegten, veröffentlichten* Regeln, einer sog. *Policy*.
- Eine CA ist – moralisch, vertraglich oder sogar gesetzlich – zur Einhaltung ihrer Policy verpflichtet.
- Die Glaub- und Vertrauenswürdigkeit einer CA ist die Grundvoraussetzung für ihre Akzeptanz und daher für ihre Tätigkeit, insofern ist dies ihr wertvollstes und zugleich kritischstes Gut.
- Die CA übernimmt die Garantie für die Richtigkeit der Daten in den von ihr ausgestellten Zertifikaten und ist u.U. sogar dafür haftbar.
- Eine CA ist in den meisten Fällen eine juristische Person, da längerfristige Stabilität und Kontinuität angestrebt werden und sich eher in dieser Rechtsform gewährleisten lassen.[3]

10.4.1 Dienstangebot

Die Kern-Dienstleistungen, die jede Zertifizierungsstelle erbringen muß, sind die **Identitätsprüfung (Registrierung)** der Zertifikatnehmer (die zu diesem Zeitpunkt noch die Antragsteller sind, denn noch sind sie nicht zertifiziert), die **Zertifizierung**, d.h. das digitale Signieren der Angaben im Zertifikat, die **Bereitstellung** bzw. **Verteilung** der Zertifikate an den Inhaber und an andere Nutzer und das **Sperrmanagement** für die eigenen Zertifikate.

Die Veröffentlichung der Zertifikate kann dabei über einen Verzeichnisdienst oder über Subskriptionsdienste („Abonnement“, z.B. per E-Mail oder auf CD-ROM per Post) erfolgen, die Bekanntgabe der widerrufenen (gesperrten) Zertifikate ebenfalls mittels eines Verzeichnisdienstes, über den Sperrlisten verbreitet werden, oder mittels Online-Statusabfrage (OCSP) einzelner Zertifikate. Die Bereitstellung oder Verteilung der Zertifikate und der Sperr-Informationen kann allerdings auch durch einen Dritten im Auftrag der CA erfolgen, z.B. durch den Anbieter eines Verzeichnisdienstes. Die auf diese Weise verbreiteten authentisierten Informationen müssen aber nach wie vor von der Zertifizierungsstelle selbst stammen bzw. erzeugt werden.

Meist wird eine Zertifizierungsinstanz den Einsatz der von ihr herausgegebenen Zertifikate für den Nutzer bis zu einer Maximalhöhe für den Mißbrauchsfall absichern, d.h. sie übernimmt eine gewisse (Zahlungs-)Garantie oder Haftung im Schadensfalle. Hierzu wird sich eine CA üblicherweise versichern, wenn sie nicht ohnehin aufgrund gesetzlicher Vorschriften oder wegen Zulassungsbestimmungen zum Abschluß einer entsprechenden Police verpflichtet ist.

Darüber hinaus kann eine CA weitere Dienste anbieten: Sie kann die Möglichkeit einer *pseudonymen* Zertifizierung alternativ zu einer normalen Zertifizierung anbieten. Dabei wird durch das Zertifikat nicht der Klarname des Zertifikatnehmers, sondern ein von ihm gewähltes (und von der CA gebilligtes) Pseudonym an einen Schlüssel gebunden. Die CA wird sich dann verpflichten, den Klarnamen –

[3] Das bedeutet allerdings nicht, daß nicht auch eine einzelne natürliche Person eine CA sinnvoll betreiben könnte; diese CA steht und fällt dann aber mit dem Engagement, der Zuverlässigkeit und Erreichbarkeit dieser Person.

und mögliche weitere Adreß- oder Identifikationsdaten – zu einem Pseudonym nur unter genau festgelegten, besonderen Umständen offenzulegen (z.B. bei Vorliegen einer richterlichen Anordnung oder auf Anfragen von Ermittlungsbehörden).

Bei einem längerfristigen Betrieb einer CA wird diese auch die Möglichkeit einer Re-Zertifizierung anbieten, d.h. die Ausstellung eines neuen Zertifikates mit anderer (längerer) Gültigkeitsdauer für einen Schlüssel, den sie bereits zuvor zertifiziert hat. Beratungsdienste dürften schließlich auch fast zwangsläufig zum Angebot einer CA gehören, denn die komplexen Abläufe und Zusammenhänge rund um die Zertifizierung und Nutzung von öffentlichen Schlüsseln werden bei den Nutzern immer wieder Fragen aufwerfen.

10.4.2 Abgrenzung CA – Trustcenter

Neben den im vorigen Abschnitt erwähnten Basis- oder Zusatzdiensten, die eine Zertifizierungsinstanz typischerweise anbieten wird, sind weitere Dienstangebote möglich: So kann eine Zertifizierungsstelle zusätzlich die Erzeugung (und Qualitätsprüfung bzw. -sicherstellung) der Nutzerschlüssel übernehmen; sie kann eine Sicherungskopie des geheimen Schlüssels aufbewahren, auf die der Nutzer zurückgreifen kann, wenn sein „Original“ aus irgendeinem Grund nicht mehr verfügbar ist; die Personalisierung und Ausgabe von Chipkarten als Speichermedium für die Schlüssel kann von ihr übernommen werden, sie kann eine Archivierung abgelaufener Schlüssel anbieten usw. Eine CA, die solche Dienste anbietet, wird dann allerdings eher als *Trustcenter* oder als *Trusted Third Party* (TTP) bezeichnet.[4]

Das ‘Trust’ (engl. ‘Vertrauen’) im Namen deutet schon an, woran man die Unterscheidung zwischen einer Zertifizierungsinstanz und einem Trustcenter festmachen kann: am Vertrauen, das ihnen entgegengebracht wird bzw. vom Zertifikatnehmer entgegengebracht werden muß. Da ein Trustcenter oder eine TTP typischerweise die Schlüssel für den Anwender generiert, hat sie auch Zugriff auf dessen geheimen Schlüssel. Der Anwender muß also seinem Trustcenter dahingehend vertrauen, daß es sich die Kenntnis seines geheimen Schlüssels nicht zunutze macht und ihn mißbraucht oder ihn Dritten offenbart. Das unterscheidet ein Trustcenter von einer reinen Zertifizierungsinstanz, die nur Kenntnis vom *öffentlichen* Schlüssel ihrer Zertifikatnehmer hat.

10.4.3 Besonderheiten der CA-Arbeit

Die Aktivitäten einer Zertifizierungsinstanz finden unter besonderen Rahmenbedingungen statt. Dieser sollten sich die CA-Mitarbeiter ständig bewußt sein und ihr Handeln danach ausrichten:

Eine Zertifizierungsinstanz „lebt“ von dem Vertrauen, das die Nutzer ihrer Zertifikate ihr entgegenbringen, d.h. die Personen, die sich auf die von ihr ausgestellten Zertifikate verlassen, wenn es darum geht, einen *authentischen* öffentlichen Schlüssel einer anderen Person oder Instanz zu erhalten. Wird dieses Vertrauen durch irgendeinen Vorfall (oder auch nur ein Gerücht!) erschüttert, wird der Arbeit der CA im schlimmsten Falle die Grundlage entzogen.

Gleichzeitig steht die CA aber mit ihren Handlungen quasi jederzeit auf dem öffentlichen „Prüfstand“. Was sie tut, mündet typischerweise in einem öffentlich zugänglichen Dokument (einem Zertifikat),

[4] Nach der Erfahrung der Autoren wird vor allem in EU-Dokumenten diese Unterscheidung ziemlich konsequent vorgenommen. Ist dort von ‘TTP’ oder von einem ‘Trust Center’ statt von einer Zertifizierungsinstanz oder einem Zertifizierungsdienstleister oder -anbieter die Rede, dann ist fast immer nicht nur eine Zertifizierung von öffentlichen Schlüsseln, sondern meist auch die Erzeugung des geheimen Schlüssels durch das Trustcenter oder seine Hinterlegung dort vorgesehen (*key escrow*)!

dessen Urheberschaft und Unverfälschtheit jederzeit und von jedem, der über den öffentlichen Schlüssel dieser CA verfügt, nachvollzogen werden kann. Fehler oder Fehlhandlungen, Verstöße gegen die eigene Zertifizierungs-Policy, können so eindeutig nachgewiesen und dann von der CA aufgrund ihrer digitalen Signatur unter der betreffenden Information kaum noch abgestritten werden.

Ein weiterer, etwas weniger gravierender Punkt sollte hier auch nicht unerwähnt bleiben: Die Entscheidung eines Zertifikatnehmers für eine bestimmte Zertifizierungsstelle (und/oder ein bestimmtes kryptographisches Verfahren) wird gelegentlich fälschlicherweise als Parteinahme des Zertifikatnehmers (über)interpretiert – etwa in dem Sinne „Derjenige läßt sich von der XY-CA zertifizieren, also vertraut er ihr auch“ oder „YYY benutzt Programm A für seine gesicherte Kommunikation, also findet er wohl B unsicher/weniger sicher.“ Diese Auffassung kann zutreffen, muß aber nicht in jedem Fall richtig sein (siehe 10.4.2).

10.5 Public-Key-Infrastrukturen

10.5.1 Skalierbarkeit

Public-Key-Verfahren werden aufgrund ihrer breiten Anwendbarkeit zu einer Querschnittstechnik werden. Etliche andere Dienste werden sich der Krypto-Mechanismen bedienen, die solche Verfahren bieten. Das setzt aber voraus, daß auch eine entsprechende Infrastruktur vorhanden ist, die die Schlüssel bzw. Zertifikate bereitstellt und ihre Nutzung ermöglicht. Letztlich muß eine solche Zertifizierungsinfrastruktur dazu geeignet sein, die unter Umständen mehreren tausend lokalen Nutzer z.B. eines Universitätsfachbereiches, die mehreren hunderttausend landesweiten Nutzer z.B. eines Großunternehmens oder die weltweit mehreren Millionen Nutzer (Beispiel: Kreditkartenunternehmen) mit den benötigten öffentlichen Schlüsseln zu versorgen. Sie muß also „mitwachsen“ können bzw. so strukturiert sein, daß auch sehr große Nutzerzahlen noch effizient bedient werden können.

Dabei taucht das Problem auf, daß einerseits die Nähe einer Zertifizierungsstelle zu den Nutzern wichtig und sinnvoll ist, damit für den erforderlichen persönlichen Kontakt bei der Identitätsprüfung die Wege kurz gehalten werden, andererseits die baulichen, technischen und organisatorischen Sicherungsmaßnahmen für eine CA sehr aufwendig sind, was wiederum eine einzige *zentrale* Zertifizierungsstelle nahelegen würde. Eine erste Maßnahme, um dieses Dilemma zu umgehen, ist die Einrichtung sogenannter „Registrierungsstellen“ (*registration authority*, RA). Mittels solcher RAs wird eine räumlich-organisatorische Trennung vorgenommen zwischen dem Ort, an dem die Registrierung der späteren Zertifikatnehmer stattfindet, und der Stelle, an der die Zertifikate schließlich erzeugt werden (die eigentliche CA). Die RAs befinden sich dann typischerweise räumlich nahe bei den Nutzern der CA. Sie nehmen die Identitätsprüfung vor und leiten dann die Daten der Zertifizierungswilligen an die CA weiter. Diese stellt dann die Zertifikate aus, die – insbesondere bei hardware-basierten Schlüsseldatenträgern („persönliche Sicherheitsumgebung“, engl. *personal secure environment*, PSE) wie Chipkarten – dann häufig bei den Registrierungsstellen von den Zertifikatnehmern abgeholt werden müssen. Als RAs fungieren häufig Organisationseinheiten oder Einrichtungen, die auch sonst ähnliche Aufgaben wahrnehmen, wie beispielsweise eine Personalstelle oder ein Immatrikulationsbüro.

10.5.2 Zertifizierungshierarchien

Bei sehr großen Nutzerzahlen, z.B. in großen Organisationen oder bei einer offenen, zahlenmäßig nicht beschränkten Nutzergruppe, gelangt man selbst bei Einrichtung von Registrierungsstellen bald

an die Grenze dessen, was mit einer Zertifizierungsstelle noch handhabbar ist. Schließlich müssen in der Regel die Wege für die Nutzer der CA (Zertifikatnehmer) weiterhin ausreichend kurz gehalten werden, so daß es ihnen noch möglich ist, die Zertifizierungsstelle (oder eine der RAs) aufzusuchen, wenn sie sich ein Zertifikat ausstellen lassen möchten. Zudem ist oft die Nähe der CA zu denjenigen Mitarbeitern oder Abteilungen einer Organisation gewünscht, die am sinnvollsten bestimmte Entscheidungen im Rahmen der Zertifizierung treffen oder bestimmte Informationen bestätigen können, so z.B. die, ob jemand tatsächlich Angehöriger der betreffenden Einrichtung ist oder wirklich bestimmte Berechtigungen erhalten darf. Aus diesen Gründen versucht man häufig, bestehende Organisationsstrukturen durch die Zertifizierungsinfrastruktur abzubilden. So gelangt man relativ rasch zu Strukturen, bei denen mehrere Zertifizierungsinstanzen zusammenwirken.

Da Organisationen meist in mehrere Hierarchie-Ebenen gegliedert sind (Geschäftsleitung, Abteilungsleitungen, Projektleiter, Mitarbeiter, ...), wird ihre Struktur oft in einer mehrstufigen Zertifizierungshierarchie abgebildet. In ihr wirken mehrere CAs in koordinierter Weise zusammen. Dabei sind sie in einer Baumstruktur durch Zertifizierungen miteinander vernetzt. Die Knoten des Baumes bilden Zertifizierungsstellen, die Blätter entsprechen den Zertifikatnehmern, und die Zweige des Baumes entsprechen den Zertifizierungen. Ganz oben steht dabei die Wurzel-Instanz (Root CA), die – meist ausschließlich – untergeordnete CAs zertifiziert. Diese wiederum zertifizieren die Nutzer (oder/und bei Bedarf auch weitere nachgeordnete CAs). Aufbauend auf dieser Struktur erfolgt die Verifikation eines Zertifikates über Zwischenschritte mit Hilfe einer Kette von Zertifikaten (Vertrauenspfad, certificate chain/chain of trust), indem mittels eines Zertifikates der Signierschlüssel verifiziert wird, mit dem das jeweils nächste Zertifikat in einer solchen Zertifikatkette unterzeichnet wurde usw.

Eine solche streng hierarchische Struktur hat zwei wichtige Vorzüge aufzuweisen, wenn es darum geht, den Vertrauenspfad (also einen Pfad innerhalb des Zertifizierungsbaumes) zwischen zwei Knoten oder Blättern zu ermitteln: Die Baumstruktur stellt sicher, daß ein solcher Pfad in jedem Fall existiert, also von jedem Teilnehmer bzw. von der Wurzelinstanz zu jedem Teilnehmer der Zertifizierungshierarchie ein Vertrauenspfad ermittelt werden kann, und sie erleichtert die Konstruktion dieses Pfades, da es in einem ausreichend ist, von beiden betrachteten Knoten ausgehend im Baum in Richtung Wurzel aufzusteigen, bis man am ersten Knoten – der ersten CA – angelangt ist, der beiden anderen übergeordnet ist. Spätestens bei der Root-CA ist dieser Punkt in einer streng hierarchischen Zertifizierungsstruktur immer erreicht. Es sind daher keine aufwendigen Suchverfahren erforderlich, um diesen Pfad zu ermitteln.

Eine rein hierarchische Zertifizierungsinfrastruktur weist aber auch einen nicht unerheblichen Nachteil auf: Sie ist anfällig gegenüber Störungen, durch die eine der beteiligten CAs ausfällt. Muß aus dem Zertifizierungsbaum ein Knoten nebst zugehöriger Kanten gelöscht werden, weil die von ihm repräsentierte Zertifizierungsinstanz ausfällt (was durch technische Defekte, Lizenz-Entzug durch die übergeordnete CA, Konkurs u.a. bedingt sein kann), so zerfällt der vorher zusammenhängende Baum in isolierte, nicht mehr miteinander verbundene Teilbäume. Aus dem einen der Teilbäume existieren dann keinerlei Zertifizierungspfade mehr zu Nutzern oder CAs im jeweils anderen Teilbaum.

10.5.3 Cross-Zertifizierung

Zu den strukturell bedingten Problemen einer streng hierarchischen Zertifizierungsinfrastruktur zählt, wie sich erst in jüngerer Zeit herausgestellt hat, die Wurzel-Instanz. In der Theorie müßte es eine „Welt-Wurzelinstanz“ geben, die alle anderen Top-Level-CAs einzelner PKIs zertifiziert und so die Verbindung zwischen diesen PKIs herstellt. Es zeigt sich aber, daß eine solche Instanz aufgrund nichttechnischer Erwägungen und Probleme nicht ohne weiteres etabliert werden kann: Die Einigung auf

eine bestimmte Instanz als „Root“ ist schwierig, da die (geographische) Standortfrage eine (macht- bzw. wirtschafts-)politische Frage ist. Unterschiedliche Rechtssysteme in den verschiedenen Ländern und damit verbundene unterschiedliche Urkundsbegriffe und Haftungsregeln erschweren es daneben zusätzlich, eine solche Instanz finden zu können. Im Bereich der Wirtschaftsunternehmen gibt es zudem wenig Bereitschaft, sich ohne überzeugenden Grund einer anderen Partei – womöglich einem Konkurrenzunternehmen – unterzuordnen. Divergierende Policies einzelner existierender PKIs tun ein Übriges, die Suche nach einer für alle Beteiligten akzeptablen Wurzel-Instanz zu erschweren.

Das Fehlen einer verbindenden Wurzel-Instanz führt dazu, daß die Gesamtheit aller PKIs unzusammenhängend und damit in ihrem Nutzen beschränkt ist, dabei wäre eigentlich das Ganze (d.h. global durch Zertifizierungen verbundene PKIs) für alle Beteiligten mehr als die Summe seiner Teile, da bei einer durch Zertifizierungen verbundenen globalen Struktur alle Teilnehmer einer PKI auch mit denen einer beliebigen anderen PKI gesichert kommunizieren könnten.

Eine gewisse Erleichterung schafft in dieser mißlichen Situation die Möglichkeit der Cross-Zertifizierung. Hierdurch können sich zwei CAs jenseits aller Hierarchien gegenseitig und ohne daß damit eine Unterordnung eines der Beteiligten verbunden wäre („peer to peer“) zertifizieren. Auch eine partielle Cross-Zertifizierung, also die (Cross-)Zertifizierung nur in einer Richtung ist möglich; auch in diesem Fall drückt sie anders als bei der „normalen“ Zertifizierung *keine* Unterordnung des Zertifizierten gegenüber dem Zertifizierer aus. Derartige Cross-Zertifizierungen können zwei Funktionen erfüllen: Sie können ansonsten disjunkte Zertifizierungsinfrastrukturen miteinander verbinden und so den Nutzern beider PKIs die Zertifikate der jeweils anderen PKI sinnvoll zugänglich machen, und sie können „Abkürzungen“ bzw. alternative Verbindungen im Zertifizierungsbaum darstellen, die im Sinne eines Ausfallschutzes (Redundanz) oder einer einfacheren Verifikation (kürzere Vertrauenspfade) durchaus sinnvoll sein können. – Leider ist in gängigen Public-Key-nutzenden Programmen die Unterstützung und Handhabung von Cross-Zertifikaten bisher kaum oder nur rudimentär implementiert.

10.5.4 Web of Trust

Den Vorteilen einer hierarchischen Public-Key-Infrastruktur – garantierte Existenz eines Vertrauenspfades zwischen zwei beliebigen Knoten, einfache Ermittlung des Vertrauenspfades, klare, überschaubare Strukturen und Verifikationsregeln, Eignung auch für große Anwendergruppen – stehen bei der Zertifizierung und Nutzung der Zertifikate die Vorzüge eines (in der Regel) unstrukturierten sog. „Web of Trust“ (WoT) oder Vertrauensnetzes gegenüber, wie man es beispielsweise bei PGP antrifft, bei dem es keine per se herausgehobenen Zertifizierer gibt und jeder Teilnehmer Zertifikatnehmer und Zertifizierer zugleich sein kann und sich Teilnehmer wechselseitig ihre öffentlichen Schlüssel zertifizieren: Das WoT ist im allgemeinen robust gegenüber dem Ausfall einzelner Knoten; es ist flexibel insofern, als es Hierarchien als eine Ausprägung des WoT mit einschließt; es bildet etablierte Geschäftspraktiken (Verträge von gleich zu gleich zwischen zwei Partnern) besser ab als die (zwangsweise) Unterordnung in einer hierarchischen PKI und das WoT ist sehr viel einfacher zu realisieren, wenigstens bei kleinen, überschaubaren Gruppen von Teilnehmern. Das WoT kann hier ohne größere Vorarbeiten oder Infrastruktur-Investitionen relativ rasch und mit wenig Aufwand etabliert werden. Ein weiterer Vorteil des WoT liegt darin, daß es verschiedene disjunkte Vertrauenspfade zwischen zwei Teilnehmern geben kann. Die sich hierdurch eröffnenden, weitergehenden Möglichkeiten werden in 10.7.4 näher skizziert.

Aufgrund der genannten Faktoren trifft man daher in der Praxis in den meisten Fällen *Mischformen* zwischen einem völlig unstrukturierten WoT und einer streng hierarchischen PKI an.

10.5.5 Kennzeichen einer PKI

Das Ensemble aus Zertifizierungsstelle(n), ggf. Registrierungs- und Ausgabestellen (RAs) nebst den Abruf- oder Verteil-Möglichkeiten für Zertifizierungs- und Sperrinformationen faßt man unter dem Namen 'Public-Key-Infrastruktur' zusammen. Häufig ist im gleichen Sinne auch von einer 'Sicherungs-' oder 'Sicherheitsinfrastruktur', von einer 'Zertifizierungsinfrastruktur' oder auch einer 'Zertifizierungshierarchie' die Rede, wenn diese Gesamtheit aus CA(s), Registrierungsstellen und Verzeichnisdienst(en) bezeichnet werden soll.[5]

Egal unter welchem der Begriffe sie firmieren, zeichnen sich doch alle solchen Infrastrukturen typischerweise durch folgende Eigenschaften aus:

- Skalierbarkeit
- Anwendungsbezug (d.h. es werden Schlüssel und Zertifikate für eine bestimmte Anwendung oder Klasse von Anwendungen bereitgestellt)
- Einbettung in einen umfassenderen, nicht-technischen Rahmen, der u.a. die rechtliche Aussagekraft der Zertifikate oder die Haftung der Beteiligten, insbesondere der CA, z.B. bei Haftung oder Mißbrauch, festlegt (siehe dazu auch 10.6.3)
- Verknüpfung mit anderen Zertifizierungshierarchien/PKIs (z.B. durch vertragliche Anerkennung oder explizite Cross-Zertifizierung)
- weitergehende Dienstangebote: Eine PKI stellt meist nicht nur Zertifizierungs- und Sperrdienste zur Verfügung, sondern bietet auch Möglichkeiten zur Schlüsselerzeugung, -Verteilung, -Archivierung oder dem turnusmäßigen oder außerplanmäßigen Schlüsselwechsel

10.6 PKI-Praxis

Dieser Abschnitt gibt einen kurzen Überblick über gängige Anwendungen und Standards mit Bezug zu Public-Key-Verfahren und -Infrastrukturen sowie über bereits in Betrieb befindliche oder demnächst in Betrieb gehende PKIs bzw. entsprechende Pilot- oder Forschungsprojekte und den aktuellen Stand der relevanten Gesetzgebung in Deutschland und auf EU-Ebene.

10.6.1 Anwendungen, Protokolle, Standards

Bekannte und bereits etablierte Anwendungen und Standards, die sich Public-Key-Verfahren bedienen, existieren im Internet vor allem im Bereich der sicheren E-Mail-Kommunikation. Dort existieren mit PGP [PGP] und S/MIME[6] gleich zwei verbreitete Standards, die vor allem im Fall von PGP bereits seit etlichen Jahren erprobt und etabliert sind. Sie ermöglichen, ebenso wie der in Deutschland vom TeleTrusT e.V. entwickelte MailTrusT-Standard [MTT], einen authentischen, integritätsgeschützten und vertraulichen Mail-Austausch nebst den zugehörigen Schlüssel- bzw. Zertifikat-Übermittlungen. S/MIME und MailTrusT greifen dabei das Zertifikat-Format auf, das in den verschiedenen Versionen

[5] Siehe [Hammer] für eine ausführlichere Diskussion dieser Begrifflichkeiten

[6] `http://www.ietf.org/html.charters/smime-charter.html`

des ITU-Standards X.509 definiert ist. Dieses flexible Format hat sich in seiner Version 3 (X.509v3) von 1997 [X.509] als Standardformat bei den meisten Anwendungen etablieren können.

Daneben existiert mit SSL (Secure Socket Layer) bzw. dessen designiertem Nachfolger TLS (Transport Layer Security) [TLS] im Bereich des sicheren Zugriffs auf Web-Server und der Authentifizierung solcher Server bereits ein weitverbreitetes Public-Key-basiertes Protokoll, das sich grundsätzlich auch zur Absicherung anderer, von Hause aus unsicherer Protokolle wie z.B. SMTP oder FTP verwenden läßt. An einem umfassenderen, nicht anwendungsspezifischen Konzept und entsprechenden Protokollen für eine allgemein nutzbare X.509-basierte Public-Key-Infrastruktur arbeitet die PKIX-Arbeitsgruppe[7] der Internet Engineering Task Force (IETF). Sie hat dazu bereits einige Vorschläge in Form von *informational* bzw. *draft* RFCs vorgelegt. (Einen guten Überblick über die diversen Drafts gibt [PKIX Roadmap].)

10.6.2 Existierende Infrastruktur

Bereits vor den ersten Entwürfen zum deutschen Signaturgesetz (vgl. 10.6.3) gab es Zertifizierungsstellen, Trustcenter und ganze Public-Key-Infrastrukturen in der Bundesrepublik Deutschland. Als Beispiele für nichtkommerziell betriebene CAs bzw. PKIs sollen hier die DFN-PCA[8], die pgpCA der Computerzeitschrift *c't*[9], die „Trustfactory“ der GMD[10] und die Zertifizierungshierarchie des Individual Network[11] dienen. Sie stellen PGP- und/oder X.509-Zertifikate aus. Auch kommerziell tätige CAs/Trustcenter gab es bereits vor dem Entwurfsstadium des Signaturgesetzes. So sind z.B. TC TrustCenter for Security in Data Networks GmbH[12], Hamburg, die Competence Center Informatik (CCI) GmbH[13], Meppen, die IKS GmbH[14], Jena, und das Produktzentrum Telesec der Deutschen Telekom AG[15], Netphen, bereits seit einiger Zeit am Markt tätig.

Seit Inkrafttreten des Signaturgesetzes 1997 hat bislang erst ein Unternehmen die Genehmigung als gesetzeskonforme Zertifizierungsstelle erhalten: die Deutsche Telekom für ihre Telesec-Tochter (s.o.). In Kürze dürfte mit der PostCom-Tochter der Deutsche Post AG eine zweite SigG-CA hinzukommen. Weitere Bewerber bzw. Interessenten stehen bereits in den Startlöchern; zu ihnen zählen D-Trust (Tochter von debis und Bundesdruckerei), DE-CODA des DIHT, TC Trustcenter, DATEV, TÜV-IT u.a.

Neben den CAs bzw. Trustcentern gab und gibt es eine ganze Reihe von Forschungs- und Pilotprojekten, die sich mit verschiedenen Aspekten von Publik-Key-Zertifizierung und -Infrastrukturen befass(t)en. Zu nennen sind hier u.a. das Projekt DFN-PCA des DFN-Vereins, der Pilotversuch SPHINX, in dessen Rahmen sichere „Ende-zu-Ende-Kommunikation“ für die Bundesverwaltung etabliert werden soll (vergleichbare Projekte gibt es auch auf Landesebene in einzelnen Bundesländern), der Einsatz von Public-Key-Verfahren im Hochschulverwaltungsbereich an der FernUniversität Hagen oder die Kooperation der TC Trustcenter GmbH mit den Hamburger Meldestellen, die in diesem Rahmen neben herkömmlichen Ausweisen auch Public-Key-Zertifikate an interessierte Bürger ausgeben sollen.

[7] http://www.ietf.org/html.charters/pkix-charter.html
[8] http://www.pca.dfn.de/dfnpca/
[9] http://www.heise.de/ct/pgpCA/
[10] http://www.darmstadt.gmd.de/trustfactory/
[11] http://www.in-ca.individual.net
[12] http://www.trustcenter.de
[13] http://www.cci.de
[14] http://www.iks-jena.de
[15] http://www.telesec.de

Auch auf EU-Ebene wurde und wird in mehreren Projekten zu Public-Key-Verfahren, -Infrastrukturen und Trustcentern geforscht bzw. werden digitale Signaturen und Zertifikate bereits im Wirkbetrieb eingesetzt:

- Das Projekt **ICE-TEL** und sein Nachfolge-Projekt **ICE-CAR**[16] befassen sich mit dem Aufbau einer Zertifizierungsinfrastruktur für Europa und der Entwicklung und Förderung entsprechender Anwendungen in Wissenschaft, Wirtschaft und Verwaltung.

- Im **EUROTRUST**-Projekt[17] wird ein ähnliches Ziel wie in ICE-CAR verfolgt (Aufbau einer CA/TTP-Infrastruktur), wobei der Schwerpunkt stärker auf den kommerziellen Aspekten des CA-Betriebs sowie auf *key recovery* bzw. *key escrow* (vgl. 10.4.2) liegt als bei ICE-CAR.

- Alcatel und die Belgische Firma GlobalSign wurden Ende Februar 1999 von der EU-Kommission damit beauftragt, ein mit Public-Key-Verfahren abgesichertes Kommunikationsnetz zwischen Regierungen und Verwaltungen der EU-Mitgliedsstaaten aufzubauen.[18]

- Die Europäische Kommission will die Verwendung elektronischer Kommunikation in Verbindung mit ihrem Fünften Rahmenprogramm (FP5) fördern. Aus diesem Grund bietet sie die Möglichkeit, Projekt-Vorschläge komplett auf elektronischem Wege einreichen zu können. Mittels digitaler Zertifikate und kryptographischer Verfahren werden dabei Authentizität und Vertraulichkeit der elektronisch eingereichten Proposals sichergestellt. Der FP5-Zertifizierungsdienst bietet dazu kostenlos digitale Zertifikate für die elektronische Kommunikation mit der Europäischen Kommission an.[19]

10.6.3 Gesetzgebung

Für den Betrieb einer Zertifizierungsstelle sind viele gesetzliche Vorschriften relevant. Die wichtigsten, weil am konkretesten auf Zertifizierungsstellen abzielenden, sind das deutsche Signaturgesetz mit der zugehörigen Signaturverordnung und die EU-Richtlinie zur elektronischen Signatur. (Daneben sind natürlich auch die Datenschutzgesetze zu berücksichtigen.)

Signaturgesetz

Das Gesetz zur digitalen Signatur [SigG], auch Signaturgesetz oder kurz SigG, ist seit 1. August 1997 in Kraft. Es zielt darauf ab, Rahmenbedingungen zu schaffen, „unter denen digitale Signaturen als sicher gelten und Fälschungen digitaler Signaturen oder Verfälschungen von signierten Daten zuverlässig festgestellt werden können“ (§ 1 Abs. 1) – insofern ist es eher ein „Sicherungsinfrastruktur-Gesetz“ als ein „Signaturgesetz“. Die Verwendung anderer Verfahren stellt das SigG ausdrücklich frei.

Das Signaturgesetz sieht eine zweistufige Zertifizierungshierarchie vor: Auf oberster Ebene befindet sich die „zuständige Behörde“ genannte Wurzelzertifizierungsinstanz, die die eigentlichen, privatwirtschaftlich betriebenen Zertifizierungsstellen (zweite Ebene) zertifiziert. Diese wiederum stellen

[16] `http://ice-car.darmstadt.gmd.de/`
[17] `http://www.baltimore.ie/projects/eurotrust.html`
[18] `http://www.globalsign.net/company/press/alcatel.htm`
[19] `http://fp5-csp.org/news.html`

die Zertifikate für die Anwender aus. Der Anwendungsbereich des SigG umfaßt Zertifizierungsstellen, für deren Betrieb es einer Genehmigung der „zuständigen Behörde“ bedarf. Die Aufgabe dieser „Wurzelinstanz“ wird im SigG der Regulierungsbehörde für Telekommunikation und Post zugewiesen. Die Erteilung einer Lizenz zum Betrieb einer Zertifizierungsstelle ist nach dem SigG an drei Voraussetzungen geknüpft: Zuverlässigkeit des Antragstellers, Fachkunde des in der Stelle tätigen Personals und Erfüllung der technischen und organisatorischen Sicherheitsanforderungen des Gesetzes an die Zertifizierungsstelle.

Im SigG wird die Bundesregierung ermächtigt, durch Rechtsverordnung Details zur Digitalen Signatur und zu SigG-Zertifizierungsstellen zu regeln, die im Signaturgesetz selbst ausgespart wurden. Dies betrifft alle wesentlichen Einzelheiten der Zertifizierung, der Pflichten einer Zertifizierungsstelle und der Maßnahmen, mit denen ihre Einhaltung kontrolliert werden soll.

Die Bundesregierung hat in ihrer *Verordnung zur digitalen Signatur (Signaturverordnung – SigV)* die entsprechende Ausgestaltung der o.g. Bereiche geregelt. Auch die Signaturverordnung bleibt noch bewußt allgemein und unspezifisch, was geeignete technische Sicherungsverfahren und Schutzmaßnahmen angeht; derartige Festlegungen wurden aus der SigV in zwei Maßnahmenkataloge für technische Komponenten und für Zertifizierungsstellen ausgelagert, die von der Regulierungsbehörde geführt und im Bundesanzeiger veröffentlicht werden.

Im Rahmen der 1999 vorzunehmenden Überprüfung des Informations- und Kommunikationsdienstegesetzes (IuKDG), zu dem als ein Bestandteil auch das SigG gehört, wurde von der Bundesregierung kein wesentlicher Korrektur- oder Veränderungsbedarf am SigG festgestellt; im wesentlichen wurde im Evaluationsbericht [IuKDG-Eval] lediglich Anpassungsbedarf gesehen, der aus der mittlerweile verabschiedeten EU-Richtlinie zu elektronischen Unterschriften (siehe nächster Abschnitt) resultiert. Kritische Stimmen, die bereits bei der Verabschiedung etliche aus ihrer Sicht verbesserungswürdige Punkte am SigG moniert hatten, fanden in diesem Bericht kaum Niederschlag.[20]

EU-Richtlinie zu elektronischen Signaturen

Am 30. November 1999 wurde die EU-Richtlinie „über gemeinschaftliche Rahmenbedingungen für elektronische Signaturen“ verabschiedet.[21] Sie wird aller Voraussicht nach Anfang Januar 2000 in Kraft treten und soll „die Verwendung elektronischer Signaturen erleichtern und zu ihrer rechtlichen Anerkennung beitragen.“ Im Sinne der Interoperabilität und eines funktionierenden Binnenmarktes legt sie dazu entsprechende rechtliche Rahmenbedingungen – auch für bestimmte Zertifizierungsdienste – fest.

Die EU-Richtlinie unterscheidet „elektronische Signaturen“, „fortgeschrittene elektronische Signaturen“ und „fortgeschrittene elektronische Signaturen mit qualifiziertem Zertifikat“.

Die Mitgliedsstaaten der EU werden in dieser Richtlinie dazu aufgefordert, dafür Sorge zu tragen, daß sog. fortgeschrittene elektronische Signaturen, die auf einem qualifizierten Zertifikat beruhen, handschriftlichen Unterschriften unter Papierdokumenten gleichgestellt werden und vor Gericht als Beweismittel zugelassen werden. Doch auch den „einfachen“ elektronischen Signaturen darf zumindest die Zulässigkeit als Beweismittel vor Gericht nicht grundsätzlich abgesprochen werden.

[20] siehe dazu z.B. Andre Reisen: *Signaturgesetz und -verordnung: Die ersten Schritte*, 1997, online unter `http://www.bsi.bund.de/pbdigsig/download/siswv.pdf`

[21] Pressemitteilung dazu unter `http://europa.eu.int/rapid/start/cgi/guesten.ksh?p_action.gettxt=gt&doc=IP/99/915|0|RAPID&lg=EN`

Weiterhin sieht die EU-Richtlinie im Unterschied zum deutschen Signaturgesetz Haftungsregelungen für die Anbieter von Zertifizierungsdiensten vor. Sie legt auch fest, daß es kein obligatorisches Akkreditierungsverfahren für Zertifizierungsstellen geben darf. Außerdem gestattet die EU-Richtlinie ausdrücklich die Verwendung von Pseudonymen anstelle von Namen in Zertifikaten.

Geplante Gleichstellung von Schrift- und elektronischer Form

Nach der Einführung der digitalen Signatur durch den deutschen Gesetzgeber (siehe 10.6.3) steht einer breiten Anwendung in vielen Fällen noch das Schriftform-Erfordernis des Bürgerlichen Gesetzbuches (BGB) im Wege (§ 126 BGB). Dieses macht die Anerkennung oder Beweiserleichterung vom Vorliegen einer verkörperten Urkunde, abhängig. Um alternativ dazu in Zukunft auch digital signierte Dokumente in elektronischer Form mit der gleichen rechtlichen Bedeutung einsetzen zu können, sollen in 2000 die entsprechenden Vorschriften des BGB um eine „elektronische Form“ ergänzt werden, die grundsätzlich als Option zur Schriftform dient.[22] Die elektronische Form wird dabei Bezug auf die Anforderungen des Signaturgesetzes nehmen. Dadurch wird – bis auf wenige, explizit benannte Ausnahmen wie Eheschließungen oder Grundstückskäufe – in allen Fällen, in denen auf die Schriftform gemäß BGB rekurriert wird, als gleichgestellte Alternative zur Papierform auch eine digitale Signatur gemäß SigG zulässig sein.

Krypto-Regulierung

Innerhalb Deutschlands gibt es – anders als in einigen anderen Ländern wie z.B. Rußland oder Frankreich[23] – keinerlei gesetzliche Beschränkungen der Nutzung von Verschlüsselungsverfahren. Versuche, entsprechende Restriktionen politisch durchzusetzen (Stichwort: Krypto-Regulierung), waren bislang nicht mehrheitsfähig.

Exportkontrolle

Soweit es sich nicht um Krypto-Produkte handelt, die ausschließlich zur Erzeugung von digitalen Signaturen eingesetzt werden können, sondern um solche, die auch die Verschlüsselung von Nachrichten ermöglichen, müssen bei der Ausfuhr der Produkte oder entsprechender Entwicklungswerkzeuge die gesetzlichen Exportbestimmungen beachtet bzw. Ausfuhrgenehmigungen eingeholt werden. Den Rahmen hierfür gibt vor allem das Wassenaar-Übereinkommen[24] vor, eine multilaterale zwischenstaatliche Übereinkunft, die (auf politischer Ebene, rechtlich nicht verbindlich) regelt, welche Produktkategorien einer Exportkontrolle unterworfen werden sollen. Zu den kontrollierten Kategorien gehört dabei u.a. die Verschlüsselungstechnik.

10.7 Ausblick

Zum Abschluß sollen kurz einige der sich abzeichnenden zukünftigen Entwicklungen umrissen werden, die auf Public-Key-Verfahren oder -Infrastrukturen aufsetzen oder eng mit diesen verbunden sind.

[22] Entwurf online unter `ftp://ftp.pca.dfn.de/pub/pca/docs/SigG/BGBE5-99.doc`,
Begründung zum Entwurf online unter `http://www.dud.de/dud/files/bgbebegr.zip`

[23] siehe B.-J. Koops: *Crypto Law Survey*, `http://cwis.kub.nl/~frw/people/koops/lawsurvy.htm`

[24] `http://www.wassenaar.org`

10.7.1 Entwicklungsrichtung der Public-Key-Standards

Der Fortschritt in der Rechenleistung moderner Supercomputer, unterschiedlich leistungsfähige Hardware (Mobilkommunikation, Chipkarten), neue Analyseverfahren oder völlig neue Ansätze (Stichwort: Quanten- oder DNA-Computer) machen es erforderlich, daß immer wieder neue Kryptoalgorithmen entwickelt und eingesetzt werden. Die bereits existierenden Standards zu kryptographischen Verfahren werden – so sie es nicht heute schon vorsehen – entsprechend überarbeitet werden (müssen), so daß sie auch neue Verfahren integrieren und unterstützen können bzw. daß zumindest ein Austausch der Krypto-Algorithmen möglich ist. Wichtig wäre hier z.B. eine Rückfallposition bzw. ein Ersatzverfahren („fallback“) für den unwahrscheinlichen, aber letztlich doch nicht mit Sicherheit auszuschließenden Fall, daß ein Verfahren wie z.B. RSA durch Fortschritte der Mathematik oder der Kryptoanalyse unsicher wird.

Ein weiteres Feld, an dessen Etablierung und Standardisierung bereits gearbeitet wird, stellen die sogenannten Attribut-Zertifikate dar. Sie dienen, anders als die bisher gebräuchlichen Public-Key-Zertifikate, nicht mehr dazu, eine Identität nachzuweisen, sondern sie enthalten darüber hinaus Angaben zu Eigenschaften (Attribute) der betreffenden Person oder Instanz.

In je mehr Bereichen PKIs zu einer unverzichtbaren technischen Grundlage werden, desto wichtiger und zugleich komplexer wird die Aufgabe, die Kommunikation und die Abläufe zwischen den beteiligten Instanzen zu managen. Insofern werden bald (einheitliche) Standards und Schnittstellen zur Verwaltung von Zertifikaten oder ganzen PKIs erforderlich werden. Erste Schritte in dieser Richtung werden bereits unternommen (siehe 10.6.1).

10.7.2 Entstehende Public-Key-nutzende Standards

In den Arbeitsgruppen der Internet Engineering Task Force (IETF)[25] wird an etlichen Standards gearbeitet, die sich auf Public-Key-Verfahren und -Zertifizierung stützen. Dazu zählen die Digital Signatur-Label für Inhalts-Informationen[26] (PICS) ebenso wie ein „fälschungssicherer“ Namensdienst [DNSSEC] für das Internet.

Daneben gibt es etliche weitere im Entstehen befindliche Internet-Standards, die letztlich auf einer u.a. mit Public-Key-Verfahren gesicherten Verbindung aufbauen. Hierzu zählen z.B. der E-Mail-Austausch über eine SSL-gesicherte Verbindung, eine kryptographische anstelle der Passwort-basierten Authentifizierung in Protokollen wie Telnet oder FTP oder die Verteilung von Schlüsseln im OpenPGP-Format unter Verwendung von TLS.

10.7.3 Zukünftige Anwendungen

Einige Bereiche, mit denen die meisten Menschen in Berührung kommen, werden in näherer Zukunft Anwendungsgebiete von Kryptographie und Public-Key-Infrastrukturen sein. Dazu zählen u.a. die Übertragung von Abrechnungsdaten im Gesundheitswesen (z.B. zwischen behandelndem Arzt, Krankenkasse, Kassenärztlicher Vereinigung, Krankenhaus usw.), die Möglichkeit, seine Steuererklärung fast komplett auf elektronischem Weg an das Finanzamt zu übermitteln[27], und der Schriftverkehr zwischen Notaren, Anwälten, Rechtsverwaltung und Gerichten.

[25] `http://www.ietf.org`
[26] `http://www.w3.org/TR/WD-DSIG-label-970605.html`
[27] `http://www.elster.de`

10.7.4 Vertrauensbewertung (Trust Metrics)

Mit steigender Durchdringung des Alltages durch Public-Key-Verfahren oder PKIs und der wachsenden praktischen Erfahrung im Umgang damit wird deutlich werden, daß die heute üblichen, eher „schlichten“ Modelle zur Vertrauensbewertung oder -berechnung, wie sie in gängigen Anwendungen implementiert sind, für manche Anwendungsgebiete unzureichend sind. Weiterhin sind die bisher realisierten Ansätze wenig robust in Bezug auf den Ausfall einzelner Knoten des Zertifizierungsnetzes. Komplexe Verfahren zur Vertrauens- oder Glaubwürdigkeitsbewertung von Zertifikat-Ausstellern und zur Erstellung und Einbeziehung von verschiedenen Zertifikat-Ketten (Vertrauenspfaden) werden daher erprobt und realisiert werden (müssen). Erste Forschungsarbeiten dazu gibt es bereits, und im bekannten Verschlüsselungsprogramm PGP ist immerhin bereits ein etwas flexibleres Modell der Vertrauensbewertung und -berechnung realisiert.

Literaturverzeichnis

[DFN-PCA99] Kelm, S. & Liedtke, B.: “DFN-PCA: Die World Wide Web-Policy: Zertifizierungsrichtlinien für das PCA-Projekt.”, Version 1.0, 1. April 1999, online unter `http://www.pca.dfn.de/dfnpca/policy/wwwpolicy.html`

[DNSSEC] Eastlake, D.: “Domain Name System Security Extensions”, RFC 2535, März 1999

[Johns] Johns, M.: „Kryptographische Verfahren“, Seminar 18.416, Sicherheit in vernetzten Systemen SS99, 1999

[Hammer] Hammer, V.: „Wie nennen wir Infrastrukturen für die Schlüsselverwaltung?“, *Datenschutz und Datensicherheit* (DuD) 2/1998, Jahrgang 22, S. 91–92, online unter `http://www.provet.org/iukdg/sis-sis.htm`

[IuKDG-Eval] „Bericht der Bundesregierung über die Erfahrungen und Entwicklungen bei den neuen Informations- und Kommunikationsdiensten im Zusammenhang mit der Umsetzung des Informations- und Kommunikationsdienste-Gesetzes (IuKDG) gemäß Beschluß des Deutschen Bundestages vom 11. Juni 1997“, BT-Drucksache 13/7935, online unter `http://www.iid.de/iukdg/BERICHTiukdg-neu-2.html`

[MTT] Bauspieß, F.: „MailTrusT Spezifikation“, Version 1.1, TeleTrusT e.V., 10. Dezember 1996, online unter `http://www.darmstadt.gmd.de/mailtrust/MTTv1/mttspc11.pdf`

[OCSP] Myers, M., Ankney, R., Malpani, A., Galperin, S., und Adams, C.: “X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP”, RFC 2560, Juni 1999

[PGP] Garfinkel, S.: *PGP: Pretty Good Privacy*, O’Reilly & Associates, Inc., 1994

[PKIX Roadmap] Arsenault, A. & Turner, S.: “Internet X.509 Public Key Infrastructure PKIX Roadmap”, <draft-ietf-pkix-roadmap-04.txt>, Oktober 1999, online unter `ftp://ftp.nordu.net/internet-drafts/draft-ietf-pkix-roadmap-04.txt`

[SigG] Gesetz zur digitalen Signatur (Signaturgesetz – SigG), 22. Juli 1997, Artikel 3 des *Gesetz zur Regelung der Rahmenbedingungen für Informations- und Kommunikationsdienste (Informations- und Kommunikationsdienste-Gesetz – IuKDG)*, BGBl. I S. 1870, 1872, online unter `http://www.iid.de/rahmen/iukdgbt.html`

[TLS] Dierks, T. & Allen, C.: “The TLS Protocol – Version 1.0”, RFC 2246, Januar 1999

[X.509] ITU-T Recommendation X.509 (1997) = ISO/IEC 9594-8: *Information Technology – Open Systems Interconnection – The Directory: Authentication Framework*, Juni 1997 (E)

Kapitel 11

Cracker-Tools am Beispiel

Friedrich Delgado Friedrichs
Universität Hamburg
Fachbereich Informatik

Zusammenfassung

Im Seminarvortrag sollte, nachdem in den anderen Beiträgen vom Administrator ausgegangen worden war, der auf seinen Systeme Verläßlichkeit, Vertraulichkeit, Datenintegrität und Verfügbarkeit der Ressourcen für autorisierte Benutzer sicherstellen will, die „andere Seite“ gezeigt und vom Cracker ausgegangen werden, der, aus welchem Grund auch immer, diese Interessen unterminieren will.

Zunächst wird im Abschnitt 11.1 ein wenig über den Hintergrund des Begriffs, des Bildes, des Selbstverständnisses und der Ideologie des Hackers oder Crackers gesagt, wie es sich als Gesamteindruck darstellt, in aller Kürze, ohne Anspruch auf wissenschaftliche Korrektheit.

Dann werden in den Abschnitten 11.2 bis 11.2.5 sowohl die vorgeblichen, als auch die tatsächlichen Ziele von Crackern untersucht, letztere etwas ausführlicher, wie sie sich aus den Handlungen von Crackern ableiten lassen. Auch die momentane Rechtslage und die möglichen, durch Cracker angerichteten Schäden sollen Erwähnung finden.

Es muß auch noch einmal auf die Programmiertechnischen und Datenverkehrstechnischen Grundlagen des Crackings eingegangen werden. Dies wird im Abschnitt 11.3 getan. Insbesondere sind hier „Buffer-Overrun“ Techniken und die TCP-Protokollsuite von Bedeutung.

Anschließend wird so etwas wie eine Standardstrategie eines Crackerangriffs dargelegt (siehe Abschnitt 11.4), die in vielen Publikationen und auch an konkreten Fallbeispielen deutlich wurde.

Dann werden zunächst Crackertools kategorisiert und an kurzen Beispielen vorgestellt, und dann anhand eines bestimmten Tools (nmap) verschiedene „Scantechniken“ genauer erläutert (Abschnitte 11.5 bis 11.6).

Vorbemerkung

Ein Merkmal der Cracker-Kultur und der Sprache der Jugendlichen „Warez-d00dz“ und „Script-Kiddies“ ist der Gebrauch einer Art „alternativer Rechtschreibung“, in der (wie in einigen Beispielen in diesem Text ersichtlich) Buchstaben durch phonetisch oder optisch ähnliche ersetzt werden.

Da diese Ausdrücke teilweise in diesem Paper zitiert werden und auch für einige englische Begriffe aus der Crackerszene (wie z.B. den Begriff „Cracker“ selbst) keine deutschen Worte existieren, kann die Ästhetik und Lesbarkeit dieses Textes beträchtlich gelitten haben. Wenn möglich, wurde versucht, dies zu kompensieren.

Im übrigen wurden viele der hier verwendeten Informationen im Internet recherchiert, weswegen auch die Literatur im wesentlichen aus Dokumenten besteht, welche (oft ausschließlich) online verfügbar sind. Sofern vorhanden wurden im Literaturverzeichnis die URLs angegeben, unter denen die Autoren ihre Texte aktuell halten, in den meisten Fällen mit einer alternativen URL, falls einer der Server mal nicht verfügbar sein sollte.

11.1 Hintergrund

Ach, was muß man oft von bösen
Kindern hören oder lesen!
Wie zum Beispiel hier von diesen,
Welche Max und Moritz hießen,
Die, anstatt durch weise Lehren
Sich zum Guten zu bekehren,
Oftmals nur darüber lachten
Und sich heimlich lustig machten.
Ja, zur Übeltätigkeit,
Ja, dazu ist man bereit!
Menschen necken, Tiere quälen;
Äpfel, Birnen, Zwetschen stehlen -
Das ist freilich angenehmer
Und dazu auch viel bequemer,
Als in Kirche oder Schule
Festzusitzen auf dem Stuhle.
Aber wehe, wehe, wehe!
Wenn ich auf das Ende sehe!!
Ach, das war ein schlimmes Ding,
Wie es Max und Moritz ging.
Drum ist hier, was sie getrieben,
Abgemalt und aufgeschrieben. Wilhelm Busch, Max & Moritz

In vielen Publikationen, vor allem im Internet, ist vieles über den Hintergrund des Begriffs „Hacker“ geschrieben worden. Prominente Beispiele sind viele deutsche Wochen- und Tageszeitungen, aber deutlich umfangreicher sind hier Publikationen aus der Crackerszene selbst, sowie aus der Open-Source-Gemeinde, die versucht, den Begriff des „Hackers“ als einen positiven gegenüber dem des maliziösen „Crackers“ abzugrenzen.

Mein persönlicher Eindruck, der beim Lesen dieser Veröffentlichungen und bei privaten Unterhaltungen über diese Thematik entstanden ist, soll diesem Haufen von persönlichen Meinungen einfach hinzugefügt werden, um den Lesern gewissermaßen einen Aufguß dieser Thematik präsentieren zu können.

11.1.1 Cracker vs. Hacker

Das „Jargon File“ [Raymond 1996][1] definiert den Begriff „Hacker“ wie folgt:

> hacker /n./ [originally, someone who makes furniture with an axe]
>
> 1. A person who enjoys exploring the details of programmable systems and how to stretch their capabilities, as opposed to most users, who prefer to learn only the minimum necessary.
> 2. One who programs enthusiastically (even obsessively) or who enjoys programming rather than just theorizing about programming.
> 3. A person capable of appreciating hack value.
> 4. A person who is good at programming quickly.
> 5. An expert at a particular program, or one who frequently does work using it or on it; as in ‘a Unix hacker’. (Definitions 1 through 5 are correlated, and people who fit them congregate.)
> 6. An expert or enthusiast of any kind. One might be an astronomy hacker, for example.
> 7. One who enjoys the intellectual challenge of creatively overcoming or circumventing limitations.
> 8. [deprecated] A malicious meddler who tries to discover sensitive information by poking around. Hence ‘password hacker’, ‘network hacker’. The correct term for this sense is cracker. [...]
>
> The term ‘hacker’ also tends to connote membership in the global community defined by the net (see *network, the* and *Internet address*). It also implies that the person described is seen to subscribe to some version of the hacker ethic (see *hacker ethic*).
>
> It is better to be described as a hacker by others than to describe oneself that way. Hackers consider themselves something of an elite (a meritocracy based on ability), though one to which new members are gladly welcome. There is thus a certain ego satisfaction to be had in identifying yourself as a hacker (but if you claim to be one and are not, you’ll quickly be labeled *bogus*). See also *wannabee*.

Aha! Und ein Cracker ist dann bitteschön?

> cracker /n./ One who breaks security on a system. Coined ca. 1985 by hackers in defense against journalistic misuse of hacker (q.v., sense 8). An earlier attempt to establish ‘worm’ in this sense around 1981-82 on Usenet was largely a failure. [...]
>
> Use of both these neologisms reflects a strong revulsion against the theft and vandalism perpetrated by cracking rings. While it is expected that any real hacker will have done some playful cracking and knows many of the basic techniques, anyone past *larval stage* is expected to have outgrown the desire to do so except for immediate, benign, practical reasons (for example, if it’s necessary to get around some security in order to get some work done).
>
> Thus, there is far less overlap between hackerdom and crackerdom than the *mundane* reader misled by sensationalistic journalism might expect. Crackers tend to gather in small, tight-knit, very secretive groups that have little overlap with the huge, open poly-culture this lexicon describes; though crackers often like to describe *themselves* as hackers, most true hackers consider them a separate and lower form of life.

[1] In diesem Fall „http://www.tf.hut.fi/cgi-bin/jargon?search=hacker“, die generische URL scheint unter „http://www.tuxedo.org/ esr/jargon/“ auf den Webseiten von Eric S. Raymond zu liegen.

> Ethical considerations aside, hackers figure that anyone who can't imagine a more interesting way to play with their computers than breaking into someone else's has to be pretty *losing*. Some other reasons crackers are looked down on are discussed in the entries on *cracking* and *phreaking*. See also *samurai*, *dark-side hacker*, and *hacker ethic*. For a portrait of the typical teenage cracker, see *warez d00dz*.

Einige der im vorigen Zitat angegebenen Referenzen führen den Unterschied zwischen einem Hacker und einem Cracker noch weiter aus. Im vorliegenden Text wird dem alten Begriff des „Hackers“ in der obengenannten Bedeutung Ehre erwiesen und stattdessen der Ausdruck „Cracker“ für die Urheber der im Referat angesprochenen destruktiven und zum Teil kriminellen Aktivitäten verwendet.

Man kann hinzufügen, daß der Terminus „Cracker“ sehr oft auch für Personen benutzt wird, die den Kopierschutz von Softwareprodukten umgehen. Desweiteren hat sich der Begriff „Script Kiddie“ für eine (zumeist jugendliche, oder moralisch unreife) Person eingebürgert, die vornehmlich die Erkenntnisse anderer Personen bezüglich Sicherheitslücken ausnutzt und sich mittels fertiger Software Informationen über und Zugang zu fremden Systemen verschafft. „Script Kiddies“ sind die Ursache eines Großteils der bekanntgewordenen Sicherheitsvorfälle, wenn es um Computersicherheit geht.

„Cracker-Tools“ ist ein Sammelbegriff für die zu diesem Zweck verwendete Software, die oft von Personen geschrieben wurde, die sich von Script-Kiddies durch ihre Kenntnisse absetzen, wenn auch nicht unbedingt in ihren Zielsetzungen.

Man kann hinzufügen, daß die Sicherheitsvorfälle zugenommen haben, seit Crackertools im Netz verfügbar sind.

11.2 Rechtfertigungen der Cracker

Während über die Rechtfertigungen der Cracker eine ganze Menge gesagt werden kann, vor allem da in den Publikationen einzelner Cracker und „Cracker-Ringe“, deren vorgebliche Ethik in oft sehr ausführlicher Weise dargelegt wird, ist deren tatsächliche Motivation hier ein Gegenstand persönlicher Spekulation (die ich mir erlauben will), bzw. sofern es zu Straftaten kommt, die Angelegenheit von Strafverfolgungsbehörden. Einiges läßt sich aber auch deutlich am Verhalten der Cracker und Script-Kiddies erkennen, die man bisweilen im irc[2] antreffen kann, oder auch an ihren Äußerungen, die man auf ihren eigenen Webseiten bzw. auf Webseiten, die von ihnen „defaced“[3] wurden, finden kann.

11.2.1 Motivation des Crackers

Hier sind meine Vermutungen:

- Materieller Gewinn
- Langeweile
- Statusgewinn bei anderen Crackern

[2] Internet Relay Chat, eines der älteren Chatsysteme des Internet, das Protokoll wird in RFC 1459 dargelegt.

[3] defaced, im Cracker Jargon: Die Information auf einer Webseite wird nach einer Cracker-Attacke durch meist sinnlosen oder anstößigen Inhalt ersetzt.

- Machtgefühl
- Neugier

Die meisten Leser können vermutlich nachvollziehen, daß Neugier eine Motivation des Crackers sein kann.

Aus einigen Veröffentlichungen (die hier zum Teil zitiert werden) läßt sich ablesen, daß es eine Art „Crackordnung“[4] unter jugendlichen Crackern gibt, in der sich ein Statusvorteil daraus ergibt, den Rechner eines anderen zu übernehmen. In den Schriften wird sich teilweise über andere (Cracker oder Administratoren) lustig gemacht, deren Server übernommen oder „0wned“ wurden. Beispiele für derartige Äußerungen finden sich in [Fyodor 1998], die den Verdacht bestärken, daß Statusgewinn und subjektives Empfinden eines Machtgefühls und die damit verbundene Steigerung des Selbstwertgefühls Motive für Cracker sein können.

Ein besonders abschreckendes Beispiel für dieses Verständnis einer „Gemeinschaft“ findet sich allerdings im Verhalten von „Script-Kiddies“ und „Bot-Owners“[5] im irc.

11.2.2 Hackerethik, Hackerideologie

Ein weiteres Zitat aus dem Jargon-File [Raymond 1996]:

> hacker ethic /n./
>
> 1. The belief that information-sharing is a powerful positive good, and that it is an ethical duty of hackers to share their expertise by writing free software and facilitating access to information and to computing resources wherever possible.
> 2. The belief that system-cracking for fun and exploration is ethically OK as long as the cracker commits no theft, vandalism, or breach of confidentiality.
>
> Both of these normative ethical principles are widely, but by no means universally, accepted among hackers. Most hackers subscribe to the hacker ethic in sense 1, and many act on it by writing and giving away free software. A few go further and assert that *all* information should be free and *any* proprietary control of it is bad; this is the philosophy behind the *GNU* project.
>
> Sense 2 is more controversial: some people consider the act of cracking itself to be unethical, like breaking and entering. But the belief that 'ethical' cracking excludes destruction at least moderates the behavior of people who see themselves as 'benign' crackers (see also *samurai*). On this view, it may be one of the highest forms of hackerly courtesy to (a) break into a system, and then (b) explain to the sysop, preferably by email from a *superuser* account, exactly how it was done and how the hole can be plugged - acting as an unpaid (and unsolicited) *tiger team*.
>
> The most reliable manifestation of either version of the hacker ethic is that almost all hackers are actively willing to share technical tricks, software, and (where possible) computing resources with other hackers. Huge cooperative networks such as *Usenet*, *FidoNet* and Internet (see *Internet address*) can function without central control because of this trait; they both rely on and reinforce a sense of community that may be hackerdom's most valuable intangible asset.

[4] in Abwandlung des Begriffs „Hackordnung“

[5] Ein „Bot“ (Abk. für Roboter) im irc ist ein Programm, welches sich mit dem irc verbindet und mehr oder weniger die Anwesenheit eines normalen Benutzers vortäuscht.

In ein paar Schriften von selbsternannten Hackern oder Autoren von Crackertools, wie den bereits erwähnten [Fyodor 1998] und [Cornwall 1985], wird versichert, daß einer Art Ehrenkodex gefolgt und versucht werde, sich vom unreifen Verhalten der „Script-Kiddies“ abzusetzen. Insbesondere [Fyodor 1998] warnt aus diesem Grunde davor, sein Tool `nmap` dazu zu verwenden, eine große Anzahl von Sites zu scannen, um sich dann aus einer so erzeugten Datenbasis von Hosts einen herauszusuchen, der eine bestimmte Sicherheitsschwäche (Im Jargon: „vulnerability“ - Verwundbarkeit) hat. Allerdings ist genau dies der häufigste Verwendungszweck dieses Tools.

Das Verhalten bei Einbrüchen in Rechner sieht oft anders aus, als es oben unter Punkt 2 definiert und gefordert wird. In der Regel werden z.B. Log-Dateien ohne Rücksicht auf Verluste gelöscht, um der Entdeckung zu entgehen.

Der mögliche Schaden, den ein Cracker-Angriff anrichten kann, läßt sich folgendermaßen beschreiben:

Personalkosten, die dadurch entstehen, daß sich Systemadministratoren und Operatoren darum kümmern müssen, wieder einen normalen Betriebszustand herzustellen. Auch die damit verbundenen Verzögerungen und Ausfallzeiten sind ein Faktor, der auch materiell Kosten verursacht.

Datenverlust: Nach einem „Root-Compromise“[6] kann über den Zustand des Systems in der Regel keine Aussage mehr gemacht werden. Programme im System können Trojaner[7] enthalten, persönliche Daten von Benutzern (wie z.B. auch elektronische Schlüssel) können entwendet worden sein, so daß es nötig sein kann, das System neu zu installieren, oder zumindestens ein Backup eines Zustands einzuspielen, der begründet als nicht kompromittiert gelten kann. Außerdem werden durch das Vorgehen des Crackers unter Umständen wichtige Daten gelöscht. Das Entfernen eines Systemlogs kann z.B. für einen Internet-Service-Provider auch einen monetären Verlust bedeuten (Verlust von Accounting-Daten).

Instabilität: Das Auswechseln von Programmen oder gar Programmbibliotheken durch Cracker (mit dem Ziel, Trojaner zu installieren) kann (unvorhergesehene) Instabilitäten zur Folge haben.

Es hängt meist von der persönlichen Einschätzung der „Opfer“ ab, was als Schaden oder Verlust verstanden wird, allerdings gibt es auch unstrittige Beispiele von destruktivem Verhalten durch Cracker:

In „An evening with Berferd“ [Cheswick 1991] wird beschrieben, wie der Angreifer an einem Punkt (offenbar aus Frustration, da die von Bill Cheswick per Hand simulierte Umgebung ihn zu verwirren schien) den Unix-Befehl „`rm -rf / &`“ als root[8] auszuführen versucht. Das ist nun offensichtlich destruktives Verhalten. Allerdings gibt es auch Beispiele für unabsichtlichen Schaden, den Cracker verursacht haben. Der von Clifford Stoll [Stoll 1988] beschriebene Angreifer drang vor seiner Entdeckung in einen Rechner ein, der bei der Echtzeitüberwachung eines medizinischen Experiments benutzt wurde. Wenn das Eindringen nicht rechtzeitig entdeckt worden wäre, hätte das laut Stoll eine ernsthafte Gefährdung eines Patienten bedeuten können. Stoll beschreibt, daß derselbe Cracker, obwohl er offensichtlich versucht hat, keinen Schaden zu verursachen, unwillentlich die Daten eines physikalischen Experiments beschädigt hat.

Damit wurde wohl dargelegt, warum bei den folgenden „Motivationen“, wie sie unter anderem Hugo Cornwall (siehe [Cornwall 1985]) angibt, eher von Rechtfertigungen gesprochen werden sollte.

[6] root compromise: Ein Unbefugter erhält die Privilegien des Systemverwalters, oder des Benutzers „root“ unter Unix.

[7] Trojaner: Ein Programm, das, meistens als legitime Anwendung oder Systemdienst getarnt, Crackern Zugriff zu einem System erlaubt, oder andere Funktionen erfüllt, die für Cracker nützlich sind, wie z.B. verfälschen von Angaben über den Systemzustand. Prominente Trojaner sind unter Windows z.B. „Back-Orifice 2000“ und „Netbus“.

[8] Die Ausführung dieses Befehls als Benutzer root löscht alle auf dem System vorhandenen Daten unwiderruflich.

11.2.3 Cracker, Script-Kiddies und Vandalen

Cracker rechtfertigen ihre Tätigkeiten zum Teil als:

- „educational und recreational sport“ [Cornwall 1985]
- Aufzeigen von Sicherheitslücken (u.a. laut [Raymond 1996] wie bereits oben zitiert)
- Politische Aufklärung gegen blinden Technikglauben (wie es scheint der vorherrschende Ductus des Hamburger CCC, u.a. in [CCC 1985])
- „quest for knowledge“ (siehe [deicide 1993])

Insbesondere zu dem Vorschlag, Cracking als „Sport“ zu bezeichnen, läßt sich einwenden, daß bei einem Sport die „Mitspieler“ in der Regel informiert und einverstanden sind. Sofern Cracking in einem solchermaßen von allen Beteiligten definierten Umfeld praktiziert wird, ist sicher nichts dagegen einzuwenden. Dem Wissenserwerb scheint die Bemühung, laufende und sichere Server zu implementieren und zu warten, dienlicher zu sein. Dem CCC darf man vielleicht einräumen, daß seine Aktionen in den 80er Jahren einige öffentliche Aufmerksamkeit hinsichtlich der Datensicherheit von BTX und ähnlichen Diensten erregt haben. Ob die gewählte Form dafür legitim ist, kann man anzweifeln.

In keinem Falle kann man dem Verhalten von Script-Kiddies, die Server willkürlich angreifen, um auf einer „defaced site“ anzugeben, irgendeinen Nutzen einräumen. Das „Script-Kiddie“ ist so verstanden gewissermaßen der Prototyp des Crackers im Sinne dieses Papers. Wer sich als Hacker fühlen möchte, mag sich verbal davon abgrenzen wollen, letztlich sind Script-Kiddies immer an ihrem Verhalten leicht zu erkennen.

11.2.4 Rechtslage

Cracken ist strafbar u.a. nach[9]

- Bundesdatenschutzgesetz §43, (Widerrechtliches Speichern, Weitergeben oder Abrufen von personenbezogenen Daten), bis zu 1 Jahr Haft, nur auf Anzeige
- StGB §202.a (Datenausspähen) Ausspähen von Daten, bis zu 3 Jahren Haft
- StGB §263.a (Computerbetrug) (Verschaffen eines rechtswidrigen Vermögensvorteils), bis zu 5 Jahren Haft
- StGB 303.a (Datenveränderung)
- StGB 303.b (Computersabotage) (hier und beim vorhergehenden nur auf Antrag, es sei denn, es liegt öffentliches Interesse vor, bis zu 5 Jahren Haft)

Das Bundesdatenschutzgesetz betrifft vor allem Behörden und öffentliche Stellen und ist hier nur der Vollständigkeit halber erwähnt. In der Regel werden Straftaten im Sinne dieser Gesetze nur verfolgt, sofern Strafanzeige gestellt wurde, oder öffentliches Interesse vorliegt (siehe [Jaeger 98]).

[9] siehe [Jaeger 98] und [Dammann 1999]

11.2.5 Ziele von Crackern

Abgesehen von den in Abschnitten 11.2.1 und 11.2.3 besprochenen Motivationen lassen sich folgende praktische Ziele aus dem Vorgehen von Crackern erkennen:

- Spionage (Datendiebstahl)
- Datenverfälschung (auch deren Herkunft)
- Ressourcendiebstahl
- Sabotage

Im folgenden wird gezeigt, wie sich diese Ziele realisieren lassen.

11.3 Grundlagen (Wiederholung)

Einige Grundlagen sind zum Verständnis der Technik und des Vorgehens von Crackern nötig. Es wird zunächst kurz auf die Technik des „Buffer-Overflows“ eingegangen, wie er in [Aleph One 1997] genauer erläutert wurde. Ebenso kurz sollen die Probleme der TCP-Protokollsuite wiederholend erwähnt werden.

11.3.1 Buffer-Overflow Techniken

Unter Unix gehören zu einem Prozess drei Speicherbereiche. Zum einen der sogenannte Text, in dem das Programm selbst (also die Maschineninstruktionen) gespeichert sind, dann gibt es zum zweiten ein Datensegment und zum dritten den Stack. Im Datensegment sind statische Variablen abgelegt, während der Stack bei Funktionsaufrufen zum Ablegen der Parameter, zum Abspeichern des vorherigen Programmzustandes und für die lokalen Variablen dient.
Man betrachte das folgende fehlerhafte c-Programm: (nach [Aleph One 1997])

```
void function(char *str) {
   char buffer[16];

   strcpy(buffer,str);
}

void main() {
  char large_string[256];
  int i;

  for( i = 0; i < 255; i++)
    large_string[i] = 'A';

  function(large_string);
}
```

Hier wird der im Ausführungsstack liegende Speicherbereich `buffer`, der für den übergebenen Parameter `string` viel zu klein ist, ohne Bereichsprüfung überschrieben. Die Anordnung auf dem Stack

entspricht hierbei der in Abbildung 11.1. Neue Variablen werden auf dem Stack beginnend mit hohen Speicheradressen und zu den niedrigeren Speicheradressen fortschreitend angelegt. Man sagt, daß der Stack „von oben nach unten“ wächst[10] (auf Abbildung 11.1 in Pfeilrichtung), so daß in unserem Beispiel der Stackpointer und die Parameter gewissermaßen „über“ dem string `buffer` liegen, weil diese *vorher* dort abgelegt wurden.

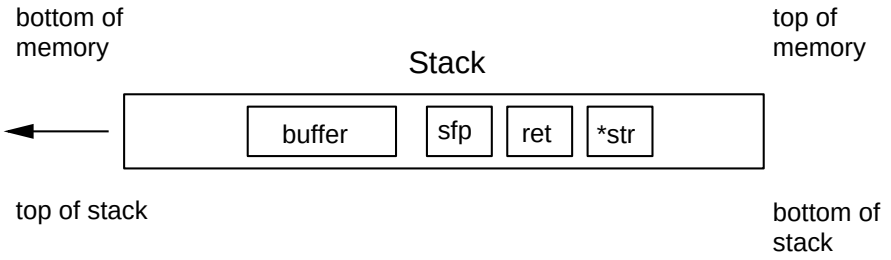


Abbildung 11.1: Der Ausführungsstack unter Unix (Beispiel) (nach [Aleph One 1997])

Bei dem in der Funktion enthaltenen `strcpy()` Befehl wird der Puffer `buffer` jedoch von unten nach oben überschrieben, so daß tatsächlich der vom Unterprogramm abgespeicherte Framepointer(`sfp`), die Rücksprungadresse(`ret`), und sogar die Variable `*str` *selbst* überschrieben wird.

In der Praxis ist die Folge meistens eine Schutzverletzung (segmentation fault), da bei Beenden des Unterprogramms versucht wird, die Rücksprungadresse aufzusuchen, die durch zufällig im String vorhandene Daten überschrieben wurde. Wenn allerdings (z.B. in einer Stringvariable) im Speicher ein Maschinenprogramm abgelegt wird, das etwa unter Unix eine shell aufruft, und der Puffer gezielt mit der Adresse dieses Maschinenprogramms überschrieben wird, ist der Effekt deutlich kontrollierter:

Da die Rücksprungadresse der Funktion mit der Adresse eines gültigen Maschinenprogramms überschrieben wurde, wird nach Beendigung der Funktion die Kontrolle an eben dieses Programm übergeben. So läßt sich an ein Programm, das diesen Implementierungsfehler[11] enthält, beliebiger Programmcode übergeben, der dann mit den Rechten des laufenden Prozesses ausgeführt wird. Da unter Unix einige Prozesse mit den Rechten des Benutzers `root` laufen, ist dies eine Möglichkeit, sich unautorisiert root-Rechte auf einem System zu verschaffen.

Es ist wichtig zu bemerken, daß diese Technik nur bei Programmen funktioniert, die auf diese spezielle Art falsch implementiert wurden. Allerdings bekommt man beim Lesen der Mailingliste `bugtraq` (die unter [securityfocus] archiviert wird), den Eindruck, daß dieses Problem auf einen Großteil der eingesetzten Programme zutrifft. Fast alle gängigen Anwendungen oder Serverprogramme enthalten oder enthielten zu irgendeiner Zeit ein derartiges Problem[12].

Wird ein „buffer overrun“ Problem bei einer Software bekannt, schreibt irgendjemand ein Programm, was diese Sicherheitslücke ausnutzt (im allgemeinen werden solche Programme als „exploits“ bezeichnet). In der Regel wird das Programm dann auf einer Mailingliste (wie z.B. bugtraq) veröffentlicht, um zu demonstrieren, daß die Sicherheitslücke ausgenutzt werden kann, aber auch um Administratoren die Möglichkeit zu geben, zu testen, ob ihr Server verwundbar ist. Die Etikette gebietet zwar, mit der Veröffentlichung eines „exploits“ zu warten, bis ein Patch für das Problem erhältlich ist, aber selbst wenn sich daran gehalten wird, bedeutet das noch lange nicht, daß alle betroffenen

[10] dieses Wachsen „nach unten“ ist laut [Aleph One 1997] auf vielen Prozessorarchitekturen die Regel.

[11] Die Möglichkeit, daß ein Puffer in nicht vom Programmierer beabsichtigter Weise überschrieben werden kann.

[12] So z.B. sendmail, syslog, XFree86, wu-ftp, pro-ftp, pine, CDE, die rpc-Suite und viele, viele mehr.

Systeme sofort „gepatcht“ werden können, so daß die Script Kiddies nun mit einem fertigen Exploit viele Server kompromittieren können[13].

11.3.2 Probleme bei TCP

Wenn man sich den Aufbau einer TCP-Verbindung durch den sogenannten „dreifachen Handschlag“ (threeway handshake) und ihren Abbau ins Gedächtnis ruft, wie es z.B. in „TCP/IP Illustrated“ [Stephens 1994] erläutert ist, so fallen folgende Fälle ins Auge:

1. Wenn ein Host lediglich ein `syn`[14] an einen Server schickt, so kommt bei einem „offenen Port“[15] ein `ack` zurück.

2. Wenn man auf einen offenen Port ein `fin` schickt, ohne daß eine Verbindung besteht, so kommt kein Paket zurück, wohingegen auf einem geschlossenen Port laut rfc 793 (TCP) gefordert wird, daß der Absender ein `rst` zurückerhält.

Beide genannten Verhaltensweisen des TCP-Protokolls bieten die Möglichkeit des sogenannten „Portscanning“, d.h. des systematischen Prüfens eines Hostrechners auf „offene“ Server-Ports. Allerdings halten sich z.B. Windows und OpenBSD nicht an die in dem RFC geforderte Verfahrensweise bei geschlossenen Ports[16]. Im ersten Fall würde der Server erwarten, daß ein Client eine Verbindung aufbauen will, aber ein Angreifer, der lediglich „portscannen“ möchte, muß nicht notwendigerweise tatsächlich eine Verbindung öffnen wollen. Früher war die Regel, daß solche unvollständigen Verbindungsversuche nicht in den Systemlogs auftauchten, weswegen ein Portscanner, der mit dieser Technik arbeitete, unentdeckt bleiben konnte.

Auf eine dem Scannen durch halbgeöffnete Verbindungen verwandte Technik, wird in Abschnitt 11.6.1 eingegangen.

Neben Portscanning bietet sich auch noch die Möglichkeit, das angegriffene System daran zu hindern, weitere TCP-Verbindungen anzunehmen, indem einfach bis zur maximalen Anzahl möglicher Verbindungen `syn` Pakete an den Server geschickt werden, so daß der TCP-Stack mit unvollständigen Verbindungen „vollgeladen“ ist (sogenanntes *Syn-Flooding*).

Weitere Probleme der TCP/IP-Protokollsuite sind:

Voraussagbarkeit von Sequenznummern: Eine Implementation des Protokollstacks muß für jedes TCP-Paket neue Sequenznummern generieren. Wenn der Algorithmus, nach dem diese erzeugt werden, so beschaffen ist, daß die jeweils nächste Sequenznummer leicht zu erraten ist, wird es für einen Angreifer möglich, eine bestehende TCP-Verbindung zu übernehmen (sog. „Hijacking“)[17].

[13] Es gibt viele Argumente, die man für Mailinglisten wie bugtraq anführen kann, in denen Sicherheitsprobleme *en detail* besprochen werden. Das wichtigste ist wohl, daß die Information jedem zur Verfügung steht, der seine Server sichern möchte, ohne daß die Notwendigkeit besteht, einen Service Vertrag oder ähnliches abzuschließen. Aber diese Praxis führt auch dazu, daß Script-Kiddies ohne jede Sachkenntnis an ihre „spl0itz“ kommen.

[14] Ein „`syn`“ soll hier kurz ein beliebiges TCP-Segment bezeichnen, in dessen Optionen das `syn` Flag gesetzt ist, desgleichen für alle anderen Flags, d.h. `fin, ack, rst` usw.

[15] d.h. wenn auf dem Host ein Serverprozeß läuft, der auf diesem Port Verbindungen annimmt.

[16] Die Nonkonformität könnte in diesem Falle bei den genannten Systemen unterschiedliche Gründe haben. Bei OpenBSD ist bekannt, daß diese Art der Implementation aus Sicherheitserwägungen gewählt wurde.

[17] Einzelheiten werden in [Bellovin 1989] erläutert.

Verhalten bei unspezifizierten Daten: Wenn bedeutungslose oder zufällige Pakete an ein System geschickt werden, oder solche, bei denen eine nicht protokollkonforme Kombination von Flags gesetzt sind, oder die in anderer Weise nicht den Spezifikationen entsprechen, sollte ein TCP-Stack im Idealfall diese Pakete verwerfen oder einen Fehler generieren. Es gibt aber TCP/IP Implementationen, bei denen solche Pakete zum Absturz des TCP/IP Stacks oder sogar des gesamten Systems führen.

Viele Systeme, die nicht gegen solche „denial of service attacks“ anfällig sind, schicken aber auf die ungültigen Pakete ganz bestimmte Antworten, die typisch für das Betriebssystem, oder sogar für eine ganz bestimmte Version des TCP/IP Protokollstapels sein können. So läßt sich das sogenannte „OS-Fingerprinting“ durchführen, mit dem sich ermitteln läßt, welches Betriebssystem auf einem entfernten Rechner läuft. Auf „OS-Fingerprinting“ wird noch einmal kurz in Abschnitt 11.6.2 am Beispiel des Tools „nmap“ eingegangen.

IP-Spoofing: „IP-Spoofing“ bezeichnet Pakete, in denen als Absender eine IP-Adresse angegeben ist, die nicht zu dem tatsächlichen Absender gehört, Pakete, die also quasi „im Namen eines anderen“ verschickt werden. Durch IP-Spoofing lassen sich unsichere Dienste, die Authentisierung durch IP-Adressen betreiben, in die Irre führen. Außerdem kann ein Angreifer dadurch versuchen, seine Identität zu verbergen, wie in Abschnitt 11.6.3 dargelegt wird.

11.4 Allgemeine Crackertaktik

Es ist bereits erwähnt worden, daß der Großteil der bekanntgewordenen Sicherheitsvorfälle von Angreifern kommt, die kein spezielles Interesse an einem bestimmten System haben. Wer ein Internet-Gateway betreibt, kann sich mittels geeigneter Software (z.B. argus, oder unter Linux: scanlogd) leicht davon überzeugen, daß aus aller Herren Länder IP-Pakete mit den merkwürdigsten Zielports eintreffen, wie es jemand auf der Bugtraq-Liste ausdrückte: „The background noise of Script-Kiddies probing our gateway[...]“. Script-Kiddies haben so etwas wie eine Taktik oder Strategie nicht, sondern suchen den schnellen „kill“. Jemand, der ein bestimmtes Ziel verfolgt, also nicht aus Gründen des Prestige in eine Site oder einen Server eindringen will, wird in der Regel sehr behutsam vorgehen und nach Möglichkeit versuchen, möglichst Präzise zu planen und Informationen zu sammeln, ohne aufzufallen.

Für die taktischen Betrachtungen hier ist es interessanter von einem (möglicherweise bezahlten) fiktiven Profi-Cracker auszugehen, der das Ziel hat, möglichst unbemerkt viel Informationen zu sammeln, oder das konträre Ziel, möglichst effektiv und plötzlich den größtmöglichen Schaden anzurichten.

Die folgenden Punkte bauen aufeinander auf, werden aber nicht unbedingt in strikter Reihenfolge, sondern eher iterativ angewandt:

Ziele auswählen: Das „Script-Kiddie“ macht sich die Zielauswahl leicht. Aus eigenen Portscans oder per „trading“ erworbene Listen von Hostrechnern mit installierten Softwareversionen erlauben es, sobald ein Exploit für eine bestimmte Sicherheitslücke veröffentlicht wird, diese Server zu „r00ten“[18].

Der fiktive Profi-Cracker, der beispielsweise von einer Regierungsbehörde beauftragt ist, Informationen eines bestimmten Typs zu beschaffen, oder bestimmte Systeme zu sabotieren, wird

[18] r00ten: root-Rechte auf einem fremden Server erlangen.

ein grobes Angriffsziel schon vorgegeben haben, sich dann aber aufgrund der Netzstruktur zum Beispiel für einen bestimmten Host entscheiden wollen.

Allgemeine Informationen sammeln: Interessant sind hier z.B. Fragen über die Netzwerktopologie des Systems. Wo sind Netzsegmente (physikalisch oder durch Switches), wo kann z.B. ein Netzwerksniffer sinnvoll eingesetzt werden? Wo in dieser Topologie befinden sich Router, Gateways und Firewalls?

Interessant kann aber auch sein, ob das Gebäude, in dem die Rechner stehen, öffentlich zugänglich sind, und ob es Bereiche gibt, in denen Publikumsverkehr herrscht. Aufschlußreich ist auch, wie die Systeme genutzt und administriert werden, um z.B. beurteilen zu können, ob ein Ausfall eines Systems eher zur Tagesordnung gehört oder ein ungewöhnlicher Vorfall ist.

Die Gewohnheiten von Usern können zu den allgemeinen Informationen gehören, die hier gesammelt werden.

Ob zwischen den Systemen eines Netzes bestimmte Grade von „Trust“ (Vertrauen) bestehen, ist sehr interessant, da z.B. manche Sites von jedem Rechner im eigenen Netz aus den Shell-Zugang per rlogin erlauben. Somit ließen sich ganze Sites übernehmen, zwischen deren Rechnern ein solches Vertrauensverhältnis zu erkennen ist.

Diese allgemeinen Informationen haben zum einen den Wert, mögliche Angriffspunkte auszumachen, aber auch Möglichkeiten zu finden, wie ein Angriff verborgen oder verdeckt werden kann.

Zu den möglichen Methoden gehört neben dem schon angesprochenen „Portscanning“ und dem Betrachten der Login-Meldungen verschiedener Dienste auch das sogenannte „Social-Engineering“. Hierbei werden durch sozialen Kontakt zu Mitarbeitern der betreffenden Behörde/Firma Informationen „ausgehorcht“ (z.B. per Email, oder im irc, denkbar ist aber auch etwa ein fingierter Anruf eines Support-Technikers[19] oder eines potentiellen Kunden).

Informationen über einen bestimmten Host sammeln: Hat man einen Hostrechner oder Router als Ziel ausgewählt, ist die Art der Hardware, das Betriebssystem, die User und vorhandene Ressourcen interessant. Auch will der Cracker herausfinden, welche Dienste laufen, welche Version einer bestimmten Serversoftware stellt den jeweiligen Dienst bereit und unter welchen Benutzerrechten diese läuft.

Über einen Portscanner lassen sich die Serverdienste leicht herausfinden (siehe Abschnitt 11.5.1).

Wenn an irgendeinem Punkt eine Sicherheitslücke offenbar wird, kann es opportun sein, sich Zugang zu dem System zu verschaffen, um sich erst einmal genauer umzusehen. Der Cracker kann aber auch entscheiden, noch eine Weile zu lauern, um mit genaueren Informationen leichter und unauffälliger eindringen zu können.

Shell-Zugang erlangen: Zugang zu einer Kommandoshell kann mit normalen Benutzerprivilegien, aber auch mit erweiterten Privilegien erfolgen. Wenn ein Dienst gefunden wird, der eine Sicherheitslücke aufweist (wie z.B. die Möglichkeit, einen „Buffer overrun“ auszuführen, wie in Abschnitt 11.3.1 beschrieben), kommt es darauf an, wie sich dieser „Bug“ ausnutzen läßt, und mit welchen Privilegien dieser Dienst läuft.

[19] wie [Fyodor 1998] vorschlägt.

Shell-Zugang läßt sich aber auch durch das Mithören des Verkehrs in einem Ethernet (sniffing), wobei telnet- und ftp-Passworte aufgezeichnet werden, oder durch unvorsichtige Benutzer erlangen, wenn diese leere oder triviale Passwörter verwenden.

Administrator-Privilegien erhalten: Gesetzt den Fall, es wurden im vorherigen Schritt nur ein oder mehrere Shellzugänge von Benutzern erlangt, so können nun die im System vorhandenen Programme z.B. auf Unix-Systemen auf das setuid-bit untersucht werden. Die Auswahl an lokalen Programmen mit root-Privilegien dürfte noch zusätzliche Möglichkeiten aufzeigen, Sicherheitslöcher auszunutzen, nachdem man bereits widerrechtlich in das System eingedrungen ist. Bei Sicherheitslücken in setuid-Programmen handelt es sich oft ebenfalls um „Buffer-overflows“, mit denen sich leicht eine „rootshell“ erzeugen läßt.

Spuren verwischen: Dies ist ein heikler Punkt. Auf einem System, auf dem intensiv protokolliert wird, ist es kaum möglich, alle Spuren zu verwischen.

Wenn z.B. die Logdateien auf einem entfernten Host gehalten werden, ist ein Fälschen der Logs entweder mit der Notwendigkeit eines Angriff auf den Loghost verbunden, was deutlich schwieriger sein kann, als der bereits gelungene Angriff, da ein „Loghost“ speziell gesichert sein sollte. Wenn die Logdateien z.B. ständig auf einem Zeilendrucker mit genügend Endlospapier ausgedruckt werden, besteht keine Chance diese zu fälschen.

Das Fälschen von Logdateien, die im Klartext vorliegen, ist an sich trivial. Binäre Logdateien (wie z.B. utmp) sind in der Regel sehr gut dokumentiert. Auch Prüfsummen des Systems, welche mit dem Tool „Tripwire“ erstellt wurden, erweisen sich als nutzlos, wenn die Prüfsummen oder das Tripwire Programm auf einem Medium liegen, auf das vom Cracker zugegriffen werden kann. Das Medium läßt sich schreibbar mounten und die Prüfsummen fälschen, oder das Programm kann durch einen Trojaner ausgetauscht werden, so daß es immer korrekte Prüfsummen liefert[20].

Mit einem so modifizierten „Tripwire“ Programm, können z.B. die im folgenden Absatz „Hintertüren und Verstecke“ erwähnten Trojaner verborgen werden.

Ein oft angewandtes Mittel, um die eigenen Spuren (von vorneherein) nicht einfach aufspüren zu lassen, besteht darin, sich auf einen bereits gestohlenen Benutzeraccount auf einem fremden Systems einzuloggen, von dem aus man sich wieder auf einen anderen Account begibt. Dies kann über mehrere Accounts hinweg geschehen, was zwar die Antwortzeiten für den Cracker deutlich vergrößert, aber die Spuren undurchsichtiger werden läßt, da sie schwerer zurückzuverfolgen sind.

Hintertüren und Verstecke: Die oben angesprochene Möglichkeit, z.B. das Programm Tripwire zu verändern, um die eigenen Aktivitäten zu verbergen, läßt sich auch auf viel grundlegendere Teile des Systems anwenden. Systemdienste lassen sich durch Versionen ersetzen, die Hintertüren enthalten, so daß z.B. das login-Programm jeden hineinläßt, der ein bestimmtes Paßwort eingibt. Programme, die Auskunft über den Zustand des Systems geben (wie z.B. top, ps oder who) lassen sich durch Versionen austauschen, die die Aktivitäten des Crackers verbergen.

Eine einfache Hintertür besteht in einem Eintrag in `/.rhosts` oder `/etc/hosts.equiv` auf Systemen, auf denen „rlogin“ Server verfügbar sind. Über einen einfachen Eintrag in diesen

[20] Darum wird auch empfohlen, sowohl die Daten als auch das Programm „tripwire“ auf einem separaten Medium, z.B. einer Diskette, einem Tape o.ä. zu halten.

Dateien kann dem Cracker jederzeit Zugriff gewährt werden. Allerdings ist so eine simple „Backdoor“ relativ leicht zu entdecken.

Wie man einen neuen Administrator bekommt, ohne Bewerbungsgespräche zu führen: Schlimmstenfalls ist nun das gesamte Netzwerk unter der Kontrolle des Crackers, ohne daß es jemand merkt. Letzteres wird aber (hoffentlich) kaum vorkommen, sofern sich überhaupt jemand um das System kümmert. Wenn nicht, ist der neue Administrator ja vielleicht auch besser als der alte...

Ob es in der Praxis vorkommt, daß eine ganze Site über längere Zeit von einem Cracker mißbraucht wird, ist mehr oder minder eine akademische Frage, da das Vorhandensein des Eindringlings möglicherweise sehr lange nicht entdeckt wird.

Was aber sicherlich vorkommt, ist, daß sich ein Cracker ein kompromittiertes System eine ganze zeitlang durch Einbauen von Hintertüren (s.o) „warmhält“, um es dazu zu verwenden, andere Systeme zu cracken, oder mittels Netzwerksniffer weitere Paßwörter interessanterer Hosts zu sammeln.

11.5 Crackertools

Crackertools lassen sich in verschiedene Kategorien einteilen:

Scanner: Es gibt hier relativ harmlose Varianten. So ist z.B. ein einfaches Shellscript, das per `telnet` oder `netcat` Verbindungen zu anderen Hosts aufbaut, auch ein Portscanner. Das Spektrum geht dann über Scanner, die noch weitere Tests machen, und Maßnahmen ergreifen, ihre Aktivität zu verheimlichen, bis hin zu Tools, die die entsprechenden Verwundbarkeiten auch aktiv ausprobieren[21], wie z.B. `Pandora` oder `nessus`.

Nicht nur „Portscanner“ fallen in die Kategorie der Scanner, auch die guten alten „Wardial-Programme“, die Bereiche von Telefonnummern nach Einwahlknoten absuchen, wie sie im Film „Wargames“ dargestellt werden. Diese Tools haben auch 20 Jahre nach ihrer Erfindung noch durchaus einen Nutzen für Cracker. Auch Anwendungen, die per snmp nach schreibbaren Attributen mit vordefinierten Paßwörtern suchen, fallen in diese Kategorie.

Netzwerksniffer: Um den Datenverkehr auf einem Kabel zu belauschen (sniffing), braucht man physikalischen Zugriff zu dem Kabel selbst. Allerdings ist das in einem Broadcast-Netz wie z.B. Ethernet leicht möglich, da alle Daten an alle Hosts und Router im LAN übertragen werden. Bei Verwendung von Switches werden allerdings zwei Hosts direkt miteinander verbunden, so daß hier ein Ethernet-Sniffing nicht mehr möglich ist. Mit „Switching-Hubs“ kann immerhin das Netz in mehrere Broadcast-Bereiche unterteilt werden, so daß ein Sniffer nur noch innerhalb eines (möglicherweise dynamisch zugeordneten) Bereichs erfolgreich ist.

Tools wie `snoop` oder `tcpdump` sind eher zur Diagnose von Netzwerkproblemen gedacht, während das Programm `sniffit` gezielt nach Passwörtern sucht und diese aufzeichnet. Das Crackertool „`hunt`“ läßt sich nicht nur als sniffer verwenden, sondern hat auch die Funktionalität, TCP-Verbindungen zu übernehmen, ARP-Antworten zu fälschen und ähnliche praktische Möglichkeiten.

[21] So wird das Tool „`Pandora`“ als „point, click and attack, security audit tool with full metal jacket ninja kungfu action“ beschrieben.

Auch hier zeigt sich, daß sowohl Programme, die für den Einsatz durch Netzwerkadministratoren konzipiert wurden, als auch von Crackern entwickelte Tools Einsatz finden können.

Allerdings kann dies auch umgekehrt gelten. Ein Administrator mag sich entscheiden, ein spezielles Crackertool zur Diagnose oder zum Testen der Systemsicherheit anzuwenden.

„Exploits“: Exploits beruhen in der Regel auf einer ganz bestimmten Sicherheitslücke eines speziellen Programms, wie bereits in Abschnitt 11.3.1 erläutert wurde.

Die Unterscheidung zwischen lokalen Exploits, also solchen, für die ein Shell-Zugang erforderlich ist, und remote-Exploits, die den Zugang über einen Netzwerkdienst erlauben, ist für die Technik ihrer Implementierung deutlich weniger wichtig als für ihre praktische Anwendung.

Passwordcracker: Hat ein Cracker eine Datei mit verschlüsselten Passwörtern erhalten, lassen sich diese durch einen Lexikonangriff berechnen. „John“[22] ist ein prominentes Beispiel eines Paßwort-Crackprogramms, das nicht nur Wörter aus einem Lexikon, sondern auch sehr ausgefeilte Permutationen und Kombinationen ausprobiert.

Logeditoren: Logeditoren können, wie auch Exploits, von erfahrenen Crackern und Hackern sehr leicht für ihre speziellen Bedürfnisse selbst geschrieben werden. Sie suchen einfach nach bestimmten Mustern in Klartext- oder binären Logs, die dann entfernt oder modifiziert werden.

Root-Kits: Unter einem „Root-Kit“ versteht man eine Sammlung von Trojanern, die mit einem Installationsscript versehen werden, das die Trojaner installiert, und gegebenenfalls Spuren verwischt.

In einem im Sommer 1999 am Fachbereich Informatik aufgetretenen Fall wurde ein Rootkit aufgespielt, das unter dem Dateinamen „neet.tar“ auf einem öffentlichen FTP-Bereich zu erhalten war. Obwohl die Installation darin erfolgreich war, einen laufenden Netzwerksniffer zu verbergen, fiel die Anwesenheit eines Endringlings dadurch auf, daß Logdateien durch das Installationsskript gelöscht wurden.

Ein Root-Kit wird wohl dem Cracker am meisten nützen, wenn es wirklich speziell auf das konkrete System maßgeschneidert wird, das angegriffen wurde, da vor der Implementierung des Installationsskriptes erforscht werden muß, wohin z.B. die Logs geschrieben werden und ob noch andere Sicherheitsmaßnahmen existieren, so daß das System sowohl mit einer effektiven Hintertür versehen, als auch die Aktivität des Angreifers möglichst verborgen wird.

11.5.1 Beispiele für Crackertools

Es folgt ein Vergleich veschiedener Crackertools.

Die IP-Adressen bzw. -Hostnamen wurden in allen Beispielen unkenntlich gemacht, um Hinweise auf die Testsysteme zu verbergen.

Saint

`Saint` ist eher als Auditing-Programm für Systemverwalter gedacht, die Sicherheitsprobleme in ihrer eigenen Site überprüfen wollen. Das Paper von Farmer und Venema [Farmer & Venema] beschreibt

[22] Es handelt sich bei „John the Ripper“ um einen Nachfolger des Programs „Jack the Ripper“.

die grundsätzlichen Sicherheitsprobleme und Überlegungen, die zu dem Saint-Vorgänger „Satan“ geführt haben.

Die Bestimmung Saints als Sicherheitstool wird dadurch unterstrichen, daß zum einen alle Scans durch volle TCP-Verbindungen durchgeführt werden, ohne irgendwelche Maßnahmen, die Aktivität zu verbergen. Zum anderen ist im Lieferumfang von Saint auch ausführliche Dokumentation zu den gefundenen Sicherheitslücken enthalten, inklusive Anleitungen, wie diese behoben werden können.

Abbildung 11.2 zeigt eine Beispielsitzung mit Saint, in der das Ergebnis eines Portscans angezeigt wird. Bei dem hier geprüften System wurden lediglich Probleme gefunden, die von Saint mit einem braunen Punkt bewertet wurden. Die Sicherheitsstufen gehen von grün (alles ok), über braun (möglicherweise problematisch) bis hin zu rot (Sicherheitslücke, möglichst sofort schließen).

Die Kurzbeschreibungen der gefundenen Probleme lassen sich anklicken und führen zur Dokumentation der speziellen Verwundbarkeit.

Abbildung 11.2: Ergebnisse eines „Saint“ Laufs

Man kann anhand dieser Angaben sehr gut entscheiden, welche Dienste auf dem betreffenden Server

abgesichert, deaktiviert, oder gegen andere ausgetauscht werden sollen.[23]

nessus

`Nessus` ist ein Portscanner mit einer eigenen grafische Benutzeroberfläche, der einige Sicherheitsprobleme aktiv austestet, indem Exploits ausprobiert werden. Man könnte sagen, daß nessus deutlich aggressiver testet, als saint. Auch ist es vorgesehen, daß „stealth-scan“ Techniken verwendet werden, die ein Systemadministrator nicht benötigt. `nessus` ist daher eindeutig als „Crackertool“ zu klassifizieren. Es gibt wenig Dokumentation und die Anzeige der „Vulnerabilities“ ist deutlich weniger aussagekräftig als bei Saint. Außerdem werden Verwundbarkeiten auch falsch identifiziert. Auf einem Server mit einer frisch installierten Version eines NFS-Servers, der kurz vorher auf eine damals aktuelle Verwundbarkeit getestet wurde, wurde diese Sicherheitslücke von nessus „gefunden“, hingegen gab es von Saint nach dem Upgrade grünes Licht.

sscan

`sscan` ist ein reines Kommandozeilenprogramm, das ähnlich aggressiv Scans ausführt wie `nessus`. Im folgenden sieht man eine kurze Beispielausgabe von `sscan`:

```
moppel:/usr/src/local/sscan # ./sscan  -v -o xxxxx
-------------------------<[ * report for host xxxxxxxxx.xxx.xxxxxxxxxxx.de *
<[ tcp port: 80 (www) ]>        <[ tcp port: 25 (smtp) ]>
<[ tcp port: 22 (ssh) ]>
--<[ *OS*: xxxxxxxxx.xxx.xxxxxxxxxxx.de: os detected: solaris 2.x
-------------------------<[ * scan of xxxxxxxxx.xxx.xxxxxxxxxxx.de completed *
```

Die Ausgabe dieses Programms kann deutlich länger ausfallen, da z.B. bei einem Fileserver noch geprüft wird, welche Verzeichnisse mit welchen Rechten freigegeben sind.

Übrigens handelt es sich bei dem Betriebssystem *nicht* um Solaris! Auch das wird also durch die hier verwendete Version von `sscan` teilweise unzuverlässig erkannt.

Im Systemlog des angegriffenen Rechners liest man:

```
Jun 24 12:44:51 xxxxxx scanlogd: From nnn.nnn.nn.nn to nnn.nnn.nn.nn ports 80,
23, 143, 110, 111, 6000, 79, 53, 31337, \dots, flags f?rp?u, TOS 00, TTL 64,
started at 12:44:51
```

Dies ist auch ein Beispiel, wie man Angriffe leicht erkennen kann. Das Tool `scanlogd` läuft im Hintergrund und hat in diesem Falle erkannt, daß von einem Rechner aus in kurzer Folge Verbindungsversuche auf verschiedene Ports stattgefunden haben, was als „Portscan“ identifiziert wurde. Scanlog kann auch auf Systemen, auf denen halbgeöffnete Verbindungen oder `fin`-Pakete nicht geloggt werden, „Portscans“ identifizieren. Allerdings hat der in diesem Beispiel verwendete Daemon „scanlogd“ ein kleines Problem, das noch im Abschnitt 11.6.3 diskutiert wird.

[23] Die hier verwendete, ältere Version von Saint arbeitete sehr zuverlässig, während bei der aktuellen Version eine Tendenz zur Positiven Falschmeldung zu bestehen scheint. Auf einem Linuxrechner wurde beim Schreiben dieser Zeilen ein verwundbarer Microsoft IIS (Internet Information Server) gefunden, der normalerweise unter Windows NT läuft. Es wäre nun zu prüfen, ob Bill Gates in das System eingedrungen ist. :-) Natürlich ist bei einem Tool für einen Administrator eine übervorsichtige Warnung besser als keine Meldung, während der Cracker möglichst nur Verwundbarkeiten angezeigt bekommen möchte, die auch funktionieren.

nmap

Im folgenden sieht man das Ergebnis eines Scans, bei dem ein einzelner Host geprüft wurde. Mit dem angegebenen Parameter „-I“ wurde zusätzlich auf jedem offenen Port eine Abfrage an den „identd“ des Zielhosts gestellt, so daß im Ergebnis erkennbar ist, unter welchen Rechten das Programm (vermutlich) läuft.[24]

```
> nmap -I xxxxxx
Starting nmap V. 2.12 by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
Interesting ports on xxxxxx.xxxxx.xxxxxxxxxxx.de (nnn.nnn.nn.nn):
Port    State       Protocol  Service         Owner
25      open        tcp        smtp            root
79      open        tcp        finger          root
80      open        tcp        http            root
111     open        tcp        sunrpc          bin
113     open        tcp        auth            nobody
139     open        tcp        netbios-ssn     root
515     open        tcp        printer         root
2049    open        tcp        nfs             root
6000    open        tcp        X11             root
Nmap run completed -- 1 IP address (1 host up) scanned in 3 seconds
```

In der Regel kann man an einer solchen Bildschirmausgabe schon sehen, welchen Dienst man als Cracker genauer betrachten sollte, um sich Zugang mit root-Privilegien zu verschaffen, und welche wahrscheinlich weniger lohnend sind.

11.6 Portscanner-Attacken am Beispiel nmap

Nmap bietet unter anderem folgende Funktionen:

„Stealth“ Portscanning: Es wird nicht gescannt, indem der volle „threeway handshake“ durchgeführt wird, sondern z.B. durch halbgeöffnete Verbindungen, oder mit ungewöhnlichen TCP-Flags. Dadurch kann der Scan auf vielen ungesicherten UNIX-Systemen verschleiert (stealth) werden, da ein Eintrag in die „Logs“ durch den „inetd“ oder durch den Dienst selbst erst *nach* einem ordentlichen Verbindungsaufbau geschieht.

Ident Lookups: Wie im vorherigen Beispiel gesehen, kann nmap auch Portscans mit ident Abfragen kombinieren.

„Sweep-Scans“: Man kann einen kompletten Netzbereich (nicht nur A-, B- oder Class C Netzwerke, sondern z.B. auch mehrere Class B Netze oder Subnetze) angeben. Obwohl sich der Autor Fyodor ausdrücklich von Script-Kiddie-Verhalten distanziert (siehe auch [Fyodor 1998]), finden die dieses nmap-Feature sicherlich praktisch. Insbesondere die Möglichkeit, sehr große Netzbereiche zu scannen, ist nur für Script-Kiddies sinnvoll, da diese möglichst schnell viele verwundbare Systeme finden möchten.

[24] Die Auskunft des ident Dienstes sind nicht verläßlich, da grundsätzlich beliebige Informationen über das auth-Protokoll zurückgesendet werden können. Typischerweise sind dies jedoch die Benutzerkennungen, unter denen die Dienste ausgeführt werden.

Decoys: Mittels IP-Spoofing werden Portscans von beliebigen IP-Adressen vorgetäuscht, um die Herkunft des eigentlichen „Portscans“ zu verschleiern. (Siehe Abschnitt 11.6.3)

UDP-Portscanning: Die Prüfung von offenen UDP-Ports ist vor allem zeitaufwendiger als die von TCP-Services, da ein offener Port kein Antwortpaket auf einen Verbindungsversuch zurückgibt, so daß über die ausbleibende Reaktion mittels „Timeouts“ entschieden werden muß, ob ein UDP-Dienst auf dem Port Verbindungen annimmt.

GUI-Frontend: `Nmap` besitzt auch auch ein grafisches Frontend. Script-Kiddies, die nicht gerne Kommandozeilen bedienen, haben damit auch eine Möglichkeit „Portscans“ durchzuführen.

Im Folgenden werden beispielhaft noch drei weitere Möglichkeiten des Tools nmap genauer vorgestellt.

11.6.1 Fin-Scans

Wie bereits in Abschnitt 11.3.2 angedeutet, verlangt die rfc 793 das in Abbildung 11.3 gezeigte Verhalten: Ein offener Port sendet (ähnlich wie bei einem Verbindungsversuch bei einem geöffneten UDP-Port) kein Feedback auf ein Fin-Paket zurück, sofern zu diesem Paket keine offene Verbindung existiert, während der geschlossene ein `rst` zurückliefert. Anhand eines Timeouts wird entschieden, ob ein Server auf dem entsprechenden TCP-Port lauscht.

Diese Art des Kontaktversuchs wird von vielen älteren Systemen nicht aufgezeichnet.

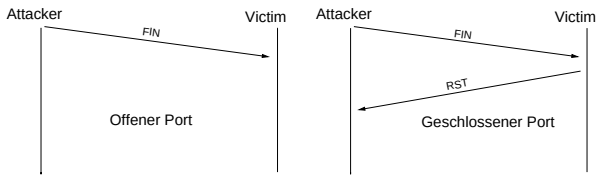


Abbildung 11.3: Zeitlicher Ablauf einer Probe mit gesetztem `fin`-Flag

11.6.2 Betriebssystemerkennung

Der TCP-Stack verschiedener Betriebssysteme liefert auf bestimmte IP-Pakete erkennbar unterschiedliche Ergebnisse, so daß Rückschlüsse auf das Betriebssystem durch folgende Tests gezogen werden können:

- „Probes“ mit verschiedenen Flags (auch ungültigen) und SYN-Floods,
- Beobachten des Verhaltens und der Vorhersagbarkeit von Sequenznummern und der TCP-Fenstergröße,
- Prüfen der im ACK Feld enthaltene Informationen bei Antworten auf „Scan-Probes“,
- Verhalten bezügl. verschiedener ICMP-Pakete,
- Reaktion des TCP-Stacks auf das TOS (Type of Service) Feld,

- Fragmentierung von Paketen und
- TCP-Optionen.

In einer eher scherzhaften Anmerkung befindet [Fyodor 1998], daß man verschiedene Versionen des Betriebssystems Windows bis auf Patchlevel an der Reaktion auf „Denial of Service“-Angriffe erkennen könne, die man in einer bestimmten Reihenfolge oder Chronologie anwenden müsse. Er fügt hinzu, es sei eine gewisse Versuchung, diese Funktionalität in nmap zu integrieren.

Nmap erkennt aber die meisten Betriebssysteme schon zuverlässig genug, um Angreifern die Auswahl der erforderlichen „Exploits“ zu ermöglichen.

11.6.3 Decoys

Abbildung 11.4 zeigt ein fiktives Netzwerk, in dem ein Rechner „victim“ von einem anderen „h4x0r“ gescannt wird, der mehrere andere Portscanning-Attacken mit falschen Absenderadressen losschickt („spooft“).

Für den angegriffenen Rechner sieht es so aus, als würde er von *allen* Rechnern gescannt, deren Adressen „h4x0r“ benutzt.

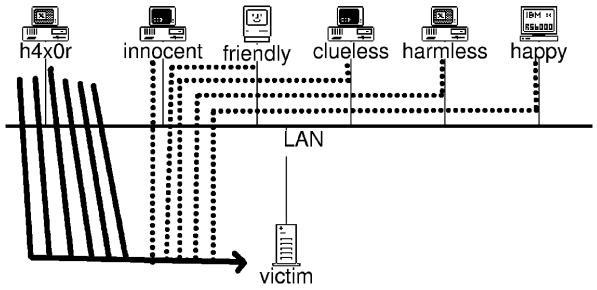


Abbildung 11.4: Nmap Attacke mit gefälschten IP-Adressen

Diese Methode hat auch für den Cracker den Vorteil, daß das Tool „scanlogd“ etwa ab dem fünften gefundenen Portscan nur noch lakonisch „more possible portscans follow“ in das Systemlog einträgt. Damit läßt sich die Herkunft des Angriffs (hier Rechner „h4x0R“) verbergen.

Sofern der Angriff aus dem LAN erfolgte, läßt sich allerdings dennoch der wahre Urheber feststellen, indem mit der Software „argus“ alle ungewöhnlichen IP-Pakete mitsamt den Hardware-Adressen aufgezeichnet werden. Alle Portscan-Pakete von „h4x0r“ tragen auch seine MAC-Adresse, durch die er zu identifizieren ist. Allerdings ändert sich die Lage, wenn der angegriffene Rechner außerhalb des LANs befindlich ist. Sobald die IP-Pakete über eine andere Netzwerkhardware, wie z.B. einen Vermittlungsrechner (Router), übertragen wurden, kann die ursprüngliche Hardwareadresse nicht mehr festgestellt werden[25].

Nmap bietet somit eine wirksame Methode, die Herkunft von Portscannings zu verschleiern.

[25] Die Möglichkeit, auch die MAC-Adressen zu fälschen, sofern die Netzwerkhardware diese Möglichkeit zur Verfügung stellt, bietet nmap nicht. Sie wäre wohl auch, wenn überhaupt, deutlich schwieriger zu implementieren.

Zusammenfassung

Nach dem allgemeinen Eindruck hat die Aufmerksamkeit für Sicherheitsprobleme bei Computersystemen in den letzten Jahren zugenommen, was aber auch mit der drastisch angestiegenen Zahl der Vorfälle zusammenhängt.

Nach einem Blick nicht nur auf die technischen Hintergründe von Cracker-Attacken wurde in diesem Paper eine allgemeine und funktionale Strategie eines Angriffs aufgestellt. Ein Blick auf einige Tools und deren spezielle Funktionalität ermöglicht es dem Leser, sich ein Bild von den Methoden und Werkzeugen der Cracker zu machen.

Was sich in diesem Artikel nicht findet, sind Anleitungen für den Netzwerkverwalter, wie das System zu sichern und im Falle eines Angriffs die Schäden zu beheben sind. In diesem Zusammenhang soll auf die technischen Tips des CERT im WWW verwiesen werden [Cert], aber auch empfohlen werden, einfach einmal im Sinne des Papers von Venema und Farmer [Farmer & Venema] im eigenen System ein Eindringen von außen zu versuchen. Dies sollte auch mit Hilfe der hier vorgestellten Tools wertvolle Erkenntnisse über das eigene System bringen, und vielleicht ein wenig der durch die Beschäftigung mit dem Thema entstandene kriminelle Energie, auf legale Weise sublimieren. In vielen der Texte und URLs in der Literaturliste befinden sich auch weitere Hinweise auf hier fehlende Details und auch Hinweise, wie ein System zu sichern ist.

Nach meiner Erfahrung ist es am wichtigsten und am sinnvollsten, sich immer auf dem neuesten Stand zu halten, was Sicherheitslücken angeht, und diese nach ihrem Auftauchen möglichst zügig zu schließen. Dies wird dadurch erleichtert, daß so gut wie alle Hersteller oder Distributoren von Betriebssystemen diese Probleme ebenfalls sehr ernst nehmen, und in der Regel Patches umgehend zur Verfügung stellen.

Literaturverzeichnis

[rootshell] `http://www.rootshell.org/beta/documentation.html` Nützliche Sammlung, von dort [Stoll 1988] und [Cheswick 1991]

[Stoll 1988] „Stalking the wily hacker“, Clifford Stoll, Communications of the ACM, May 1988 vol. 31 nr. 5 page 484(14) enthält die Beschreibung eines (rechtzeitig bemerkten) Hackerangriffs aus Sicht eines Administrators. Es mag interessant sein anzumerken, daß im Zusammenhang mit dem dort geschilderten Angriff auch eine Gruppe von drei deutschen Crackern aufgespürt wurde, die im Zusammenhang mit dem sogenannten „KBG-Hack“ standen, der in der Presse und neuerdings im Film „23“ hinreichend geschildert wurde. Als Buch: Clifford Stoll, „The Cuckoo’s Egg“, Mass Market Publishing, 1995

[Cheswick 1991] „An evening with berferd“, Bill Cheswick, 1991, ähnlich wie [Stoll 1988], dem Cracker wird ein speziell vorbereiteter Rechner als Angriffsziel und Gefängnis „untergejubelt“, in dem seine Aktivitäten studiert werden. `http://www.rootshell.org/docs/berferd_cheswick.ps.gz` auch `http://www.securityfocus.com/data/library/berferd.ps`

[Farmer & Venema] „Improving the Security of your Unix-System by breaking into it“, Farmer & Venema `http://www.rootshell.org/docs/improve_by_breakin.txt` generischere URL: `ftp://ftp.porcupine.org/pub/security/admin-guide-to-cracking.101.Z` (leicht aktualisiert) auf der Homepage von Wietse Venema

[Cert] http://www.cert.org/tech_tips/

[securityfocus] http://www.securityfocus.com/ u.a. das Bugtraq-Archiv, viel Lesestoff, „Vulnerabilities“ und „Exploits“.

[Raymond 1996] Das „Jargon-File“ http://www.tuxedo.org/~esr/jargon/ Interessant und witzig! In Buchform aber weniger aktuell: Eric S. Raymond (Hrsg.) „The New Hacker’s Dictionary“, 3. Auflage, MIT-Press 1996

[Fyodor 1998] http://www.insecure.org/nmap/nmap-fingerprinting-article.html Eine Erklärung der Funktionsweise der Betriebssystemermittlung mittels „OS-Fingerprinting“

[Cornwall 1985] Hacker’s Handbook, Hugo Cornwall, Century Communications Limited, 1985, als ascii: http://www.rootshell.org/docs/Hackers-Handbook auch unter http://www.dopeman.com/g-files/hack/hacker_hnd.txt

[CCC 1985] Die Hackerbibel, Chaos Computer Club, Hamburg, Grüner Zweig, 1985

[deicide 1993] The Neophyte’s Guide to Hacking, deicide, http://www.rootshell.org/docs/hacking_guide.txt auch http://newdata.box.sk/neworder/docs/hacking_guide.txt

[Jaeger 98] „Sind Hacker-Tools strafbar?“, Stefan Jaeger, c’t 5/98, S.18

[Dammann 1999] „Computerkriminalität aus Sicht von Ermittlungsbehörden“, Marc-Oliver Dammann, Tagungsunterlagen 6. Workshop „Sicherheit in vernetzten Systemen“, DFN-Cert und DFN-PCA, Hamburg 1999

[Aleph One 1997] „Smashing The Stack For Fun And Profit“, Aleph One, Phrack Vol. 7, Issue 49, http://www.securityfocus.com/data/library/P49-14.txt auch http://www.fc.net/phrack/files/p49/p49-14

[Stephens 1994] „TCP/IP Illustrated, Volume 1 : The Protocols“, W. Richard Stephens, Addison-Wesley, 1994

[Bellovin 1989] „Security Problems in the TCP/IP Protocol Suite“, Steven M. Bellovin, 1989, online unter http://www.deter.com/unix/papers/tcpip_problems_bellovin.ps.gz und http://rootshell.com/docs/tcpip_problems_bellovin.ps.gz