Netzwerksicherheit

Seminararbeit im Proseminar IT-Sicherheit am Fachgebiet Programmiermethodik, Technische Universität Darmstadt

Daniel Szallies
Carsten Längsfeld

Juli 2001

Zusammenfassung

In dieser Arbeit werden Teilaspekte der Netzwerksicherheit betrachtet. Es werden Angriffsmethoden und Schutzmaßnahmen exemplarisch anhand der Themen Viren & Würmer, bzw. Firewalls diskutiert. Dabei wird schädigende Software nach ihrer Methodik klassifiziert, ihre möglichen Auswirkungen untersucht und Ansätze zur Abwehr aufgezeigt. Außerdem werden die grundsätzlichen Funktionen sowie die wichtigsten Elemente von Firewalls erläutert und anhand einiger Beispiele mögliche Firewall-Architekturen dargestellt und bewertet.

Schlagwörter: Netzwerksicherheit, Viren, Würmer, Trojanische Pferde, Firewalls, Router, Proxies

1. Einleitung

Im Zuge der Globalisierung und der zunehmenden Vernetzung von Konzernen, Organisationen und Behörden gewinnt das Thema Netzwerksicherheit immer mehr an Bedeutung. Die elektronischen Infrastrukturen und Ressourcen werden dabei durch Angriffe über Netzwerke, speziell das Internet, in immer stärkerem Maße gefährdet. Daher werden in dieser Arbeit Angriffsmaßnahmen auf Rechnerverbände am Beispiel von Viren und Würmern, sowie Abwehrmaßnahmen anhand von Firewalls dargestellt.

Diese Arbeit ist folgendermaßen gegliedert: Das zweite Kapitel behandelt die automatische Implantierung durch Viren, Würmer und Trojanische Pferde. Danach werden Nutzlasten und ihre möglichen Auswirkungen diskutiert. Im darauffolgenden Kapitel werden Ansätze zur Sicherung gefährdeter Systeme erörtert. Im fünften Kapitel werden zunächst die Grundidee und die Ziele von Firewalls, sowie die wichtigsten Firewall-Elemente vorgestellt. Im Anschluß werden verschiedene Firewall-Architekturen und ihre Vor- und Nachteile diskutiert Abschließend erfolgt der Versuch einer Bewertung des Status quo, sowie eine Prognose zukünftiger Angriffstrategien.

2. Prinzip der automatischen Infiltration

Angriffe auf Rechnersysteme haben nach gängiger Erfahrung das Ziel, Daten oder Abläufe des Systems im Sinne des Angreifers zu beeinflussen. Die häufigsten Motivationen sind fehlgeleiteter Spieltrieb oder, vor allem innerhalb von Firmen und Behörden, Rachsucht gegen eine vermeintliche oder tatsächliche ungerechte Behandlung eines Mitarbeiters durch Vorgesetzte. Seltener, aber wegen ihrer meist größeren „Zielstrebigkeit“ gefährlicher sind Versuche, die Kontrolle über ein Rechnersystem zu übernehmen; solche Fälle sind vor allem innerhalb der Wirtschaftskriminalität (z.B. Spionage) oder als „elektronische Kriegführung“ („Cyberwar“) von Bedeutung. Fremde Rechnersysteme sind in der Praxis nicht allgemein zugänglich, so dass sie nicht durch direkte Einwirkung manipuliert werden können (Zugriffe über Datennetze werden gesondert behandelt). Oft besteht jedoch für Außenstehende die Möglichkeit, das System zu infiltrieren, d.h. zusätzliche Software einzuschleusen, welche die gewünschten Manipulationen durchführt. Gerade das Einschleusen ist jedoch nicht trivial, da der direkte Zugriff (und damit das Einspielen von Programmen z.B. über Wechseldatenträger) eben nicht möglich ist; entsprechende Software muss sich daher selbst einschleusen. Da die Fähigkeit einer Software, sich selbst in ein System zu implantieren, völlig unabhängig ist von eventuell zusätzlichen (ggf. destruktiven) Manipulationen, trennt man diese sinnvollerweise auf in ein Implantierungsmodul und eine oder mehrere Nutzlasten. Wir wollen dabei im folgenden – speziell bei Betrachtung der Nutzlasten - voraussetzen, dass entsprechende Programmfunktionen mit Absicht implementiert wurden, da auch „normale“ Programme entsprechende und ggf. schädliche Nebenwirkungen entfalten können, wenn sie falsch ausprogrammiert oder benutzt werden.

2.1. Automatische Implantierung

Entsprechend dem obengesagten muss ein Infiltrationsprogramm selbständig in ein System eintreten und ablaufen können. In der Praxis können wir hier eine ausgezeichnete Gruppe von Programmen unterscheiden, welche die in [2] ausführlich diskutierten Techniken der *selbstreplizierenden (oder regenerativen) Codes* benutzt. Dabei handelt es sich im Kern um einen Programmierstil, bei welchem ein laufendes Programm Kopien seiner selbst herstellt und in geeigneter Weise verbreitet. Bis jetzt treten diese Techniken in der Praxis in zwei Varianten auf: *Computer-Viren* und *Würmer*. Unter den Infiltrationsprogrammen, die keine selbstreplizierende Codes verwenden, dominieren bis heute die *Trojanischen Pferde.* Wir entwickeln im Folgenden eine Klassifikation der einzelnen Typen entsprechend Fig. 1:

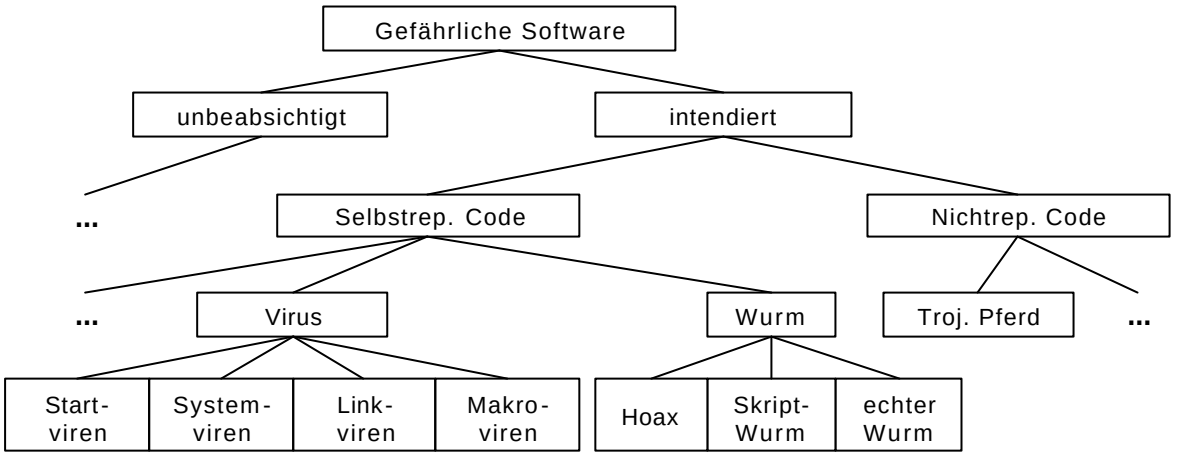

Fig. 1: Klassifikation potentieller Schadensträger

2.1.1. Computer-Viren

Der Begriff des Computer-Virus wurde zuerst von Fred Cohen in seiner Arbeit [3] definiert. Ein „Virusprogramm“ hat danach die Eigenschaft, bei seiner Ausführung Kopien seiner selbst herzustellen und diese in andere auf Rechnersystemen befindliche Programme einzubetten, sie zu „infizieren“, wobei diese Programme noch lauffähig bleiben und selbst wieder als Viren agieren. Die Möglichkeit eines Computervirus folgt offensichtlich aus der von-Neumann-Architektur; da Programme wie Daten erstellt und gespeichert werden, können sich offenbar Programme selbst kopieren. Viren lassen sich daher auf allen Abstraktionsebenen eines Rechners realisieren, auf denen das Prinzip der Egalität von Programmcode und Daten gegeben ist; unter den heute üblichen Rechnerarchitekturen für normale Anwendungen sind dies:

- Die **Hardwareebene,** auf welcher die sog. Startsektorviren den Programmcode des „Bootstrap“ auf Disketten oder Festplatten ersetzen, durch den das jeweilige Betriebssystem vom BIOS gestartet wird.

- Die **Systemebene** mit Manipulation des Betriebssystems (bis jetzt nur der unter MS-DOS® 5.0 lauffähige Verzeichnisinfektor „DIR-II“, dessen Funktion in [5] erläutert wird),

- Die **Anwendungsebene** mit Manipulation von ausführbaren Programmdateien (hier die große Menge der im Umlauf befindlichen „Linkviren“, welche den Programmablauf so modifizieren, dass zuerst der Virenkern abgearbeitet wird, der dann das ursprüngliche Programm nachlädt; ein bekannter Vertreter ist „Vienna“, vgl. [2].), sowie seit einiger Zeit

- Die **Dokumentenebene**, bei denen mit Büroanwendungen wie „Microsoft® Office“ oder „Starwriter“ erstellte Dokumente sogenannte Makros enthalten, die von der Anwendung interpretiert werden und sich selbst in andere Dokumente kopieren

Die genannten Beispiele gelten für IBM®-artige Systeme mit Microsoft®-kompatiblen Betriebssystemen; auf anderen Rechnern/Architekturen (z.B. mit Betriebssystem in ROM-Speicher) entfallen eventuell einige Ebenen oder es existieren weitere. Ein „Virus“ ist allerdings stets ein (allein meist nicht ausführbares) „Stück Software“, welches sich in der oben beschriebenen Weise vervielfältigt. Einen anderen Ansatz zur Verbreitung realisieren die Würmer:

2.1.2. Würmer

Als Würmer werden generell eigenständige Programme definiert, welche Kopien ihrer selbst anfertigen und diese – meist über ein Netzwerk – verbreiten. Entsprechend ihrem unterschiedlichen Maß an Eigenständigkeit lassen diese sich in drei Gruppen unterteilen:

2.1.2.1. **Hoaxes.** Die einfachste Version eines Wurmes entsprechend dem herkömmlichen Kettenbrief. Im Normalfall eine E-Mail mit der rhetorisch verkleideten Bitte, gerade diese E-Mail an alle Bekannte zu verschicken, wo sich dasselbe Spiel wiederholt.

2.1.2.2. **Skriptwürmer.** Eine weitere Spielart der E-Mail-Würmer, welche aus einer normalen Mail mit einem Dateianhang besteht, in der sich ein Skript befindet, durch welches ein Mailclient ferngesteuert und veranlasst wird, die Mail mitsamt Attachement an alle im Adressbuch eingetragenen Personen zu verschicken. Die bekanntesten Beispiele sind „Melissa“ und „I love You“; allerdings existiert das Prinzip schon seit 1986 mit dem „Tannenbaum-Wurm“ unter VM/CMS (siehe [2]).

2.1.2.3. **Echte Würmer.** Im Gegensatz zu den obigen, textbasierten Systemen sind echte Würmer Maschinenprogramme, welche die Netzwerkdienste von Systemen eigenständig nutzen und sich in Netzwerken frei bewegen. Das mittlerweile klassische Beispiel ist der in [14] besprochene Wurm „Happy99“ alias „Win32/Ska“.

2.1.3. Trojanische Pferde

Im Gegensatz zu den obigen Techniken ist das Prinzip des Trojanischen Pferdes seit der Antike bekannt und unverändert geblieben: Es handelt sich dabei um ein vorgeblich nützliches „Ding“ (bei einem Rechnersystem ein Spiel, Testprogramm etc.), welches die Einwohner von Troja/den Benutzer verleiten soll, die darin versteckten „Krieger“ (die Sabotagefunktion) freiwillig in die Festung/das System zu holen. Im Gegensatz zu den oben diskutierten Implantationsmethoden hat ein Trojanisches Pferd **grundsätzlich** schädigende Wirkung, abhängig von der eingestellten Nutzlast (siehe Punkt 3, „Nutzlasten“). „Trojaner“ mit datenzerstörender Wirkung sind meist nur kurze Zeit im Umlauf, da bei Bekanntwerden der destruktiven Wirkung das Programm natürlich gelöscht bzw. nicht weiterverbreitet wird (ein Trojanisches Pferd kann sich i.a. nicht selbst kopieren, obwohl Mischformen mit den oben diskutierten selbstreplizierenden Formen möglich sind). Wegen der großen Entdeckungsgefahr aktiviert sich die eingebaute Zusatzfunktion meist sofort und kann trotz der kurzen Umlaufzeit immensen Schaden anrichten (vgl. die Auswirkungen des „*PC-Cyborg Aids-Info-Disk*“-Trojaners nach [5]). Programme zur Datenausspähung oder Fernsteuerung eines Rechners dagegen können relativ lange in einem System verbleiben, da sie – im Gegensatz zu Viren und Würmern – keine verräterischen Zusatzaktivitäten im Zuge der „Vermehrung“ erzeugen. Da es sich bei ihnen überdies um „erwünschte“ Programme handelt, fällt ihre Anwesenheit natürlich nicht auf, bis die versteckten Nutzlasten entdeckt werden.

2.2. Erweiterte automatische Implantierung

Die oben beschriebenen Implantationsverfahren lassen sich noch ausbauen; in der Praxis findet man oft Zusatzfunktionen, welche das Entdecken oder Entfernen eingeschleuster Software verhindern sollen; bei Computer-Viren z.B. die sog. „Stealth-Fähigkeiten“, welche bei jedem Systemzugriff normale Verhältnisse vorspiegeln, oder noch gefährlichere Funktionen zur Verschlüsselung von Datenträgern, die ohne Virus/Wurm nicht mehr auslesbar sind (z.B. der MS-

DOS-Virus „One-Half/Freelove“), sowie Zerstörungsfunktionen, die beim Start eines Virensuchers ausgelöst werden; solche Mechanismen entsprechen dem Zusatzzünder einer Landmine, welcher die Explosion auslöst, sobald die Mine geräumt wird. Bei Trojanischen Pferden sind Zusatzfunktionen dagegen selten zu finden, zum ersten, da sie ihre Anwesenheit nicht verbergen müssen (lediglich die Nutzlasten sind geheim; um sie zu finden, müsste die fragliche Software zurückentwickelt werden, was nicht nur mit extremen Aufwand verbunden ist, sondern meist noch durch Lizenzverträge explizit untersagt wird). Überdies können sich Trojanische Pferde i.a. nicht selbst verbreiten und daher auch kaum auf Gegenmaßnahmen reagieren (allerdings enthalten Trojaner gelegentlich einen „Installer“, welcher die betreffenden Programme dauerhaft im System einpflanzt und automatisch starten lässt).

3. Nutzlast

Als Nutzlast gilt eine Programmfunktion, welche nicht zur oben angeführten Implantierung beiträgt, sondern jede andere Aufgabe ausführen kann; i.a. führt sie die eigentlich vom Programmierer/Angreifer gewünschten Manipulationen auf dem Zielsystem durch; das Implantierungsmodul (z.B. ein Wurm) dient dabei nur als Container, um die Funktion in den Rechner einzuschleusen. In der Praxis kommen meist Nutzlasten schädigenden Charakters vor, die sich in drei Gruppen einteilen lassen:

3.1. **Sabotagefunktionen.** Diese haben die Aufgabe, Datenverarbeitungsprozesse in den Zielsystemen zu stören oder unmöglich zu machen oder Daten zu löschen; die häufigsten Ausprägungen sind Löschung von Datenträgern, Vortäuschen von Systemfehlern und Verringerung der Systemperformanz.

3.2. **Systemfernsteuerung („Backdoor“).** Eine „Hintertür“ soll dem Angreifer widerrechtlich Zugriff auf das Zielsystem verschaffen; häufig handelt es sich um ein kleines Serverprogramm, welches eine Netzwerkverbindung (meist in das Internet) aufbaut, Befehle entgegennimmt und auf dem Zielsystem ausführt. Wird die Verbindung vom Zielsystem aus aufgebaut, so können Sicherheitssysteme (Firewalls) unterlaufen werden, da diese primär gegen unzulässige Zugriffe aus dem Netzwerk heraus gerichtet sind. Entsprechendes gilt für Fernsteuerung über ungesicherte Modem-Verbindungen, siehe 6).

3.3. **Spionagefunktionen.** Ein Spionagemodul versucht, nichtöffentliche Daten über das Zielsystem bzw. dessen Benutzer zu sammeln und über ein Netzwerk oder eine sonstige Verbindung aus dem Zielsystem zu übertragen; oft sind die Spionagefunktionen mit einem Fernsteuerungsmodul gekoppelt bzw. das Fernsteuerungsmodul kann auch geschützte Daten ausfindig machen. Ein Sonderfall sind die „Rückmeldungssysteme“ in werbefinanzierter Software (vergl. [1]); es handelt sich dabei um Module, welche neben dem sporadischen oder permanenten Einblenden von Werbung Informationen über den Benutzer zu gewinnen suchen, um die Werbeschaltungen entsprechend anzupassen (je nach System über Dialoge, durch Auswerten der Verlaufs-Einträge des Internet-Browsers etc.). Diese Daten werden entweder lokal gespeichert oder über das Internet an die Werbungsserver übertragen. Dabei besteht die Gefahr eines „gläsernen Surfers“, speziell wenn das Rückmeldesystem jedem Benutzer eine eindeutige Kennung zuweist; dann kann zugleich dessen Anonymität untergraben werden. In der Privacy-Erklärung von z.B. Cydoor Technologies [4] heißt es dazu sinngemäß: „Cydoor no longer assigns a unique user id“. Das ähnlich arbeitende System von Radiate (früher Aureate Media) [11] dagegen „sets a numerical identifier on your computer.“ Es sollen hier keineswegs den genannten Firmen (bei denen es sich nur um eine willkürliche Auswahl handelt) bösartige Absichten unterstellt werden, aber mit der Verbreitung persönlicher Daten steigt zwangsläufig die Gefahr des Missbrauchs (schon wegen der Möglichkeit, dass der Datenverkehr zum Werbeserver abgehört wird). Ein anderer Fall ist das Abspeichern von Informationen, die einen Rechner eindeutig identifizierbar machen, wie es z.B. angeblich seit neuestem von Blizzard praktiziert wird [7], vornehmlich um Störenfriede im Online-Spielnetzwerk „Battle.net“ aufzuspüren. Nach dem in [7] abgedruckten Teil des Lizenzvertrages „it is the policy of Blizzard to co-operate fully in criminal investigations, including but not limited to sharing all user information on file with the proper authorities on request“. Dies eröffnet u.a. die – durchaus akzeptable – Möglichkeit, illegale

Kopien des betreffenden Computerspieles aufzuspüren, aber auch die Gefahr, dass das System zur Überwachung von Nutzern durch staatliche Stellen ausgenutzt wird.

4. Lösungsansätze

Speziell um die Auswirkungen von Sabotagefunktionen zu mindern, sollten von allen wichtigen Datenbeständen Sicherheitskopien angefertigt werden; diese – seit Jahren erhobene – Forderung wird allerdings in der Praxis oft nicht umgesetzt. Neben der Möglichkeit, mit Suchprogrammen nach bereits bekannten Schadensträgern zu suchen (die bekannten „Virenscanner"), bildet sich seit einiger Zeit wieder der klassische Ansatz heraus, der in [13] mit den Worten charakterisiert wird: „Im Normalbetrieb arbeitet der Anwender mit eingeschränkten Rechten. Die ermöglichen ihm zwar seine normalen Tätigkeiten, gestatten aber keine Schreibzugriffe auf Systemdateien oder zentrale Programme wie das installierte Office-Paket". Im Endeffekt läuft dies auf eine Abkehr von der von-Neumann-Architektur hinaus; eine Möglichkeit, ein „sicheres" System zu erhalten, nennt [2] denn auch das Speichern der Programme auf nicht-schreibfähigen Medien (z.B. Chipkarten). Da eine solche Architektur den Aufwand für die Systempflege stark erhöht, muß im Einzelfall zwischen Sicherheit und Komfort abgewogen werden (für einen hauptsächlich zum Spielen benutzten Rechner lohnt sich der Mehraufwand kaum, bei einem Waffenkontrollsystem muss dagegen eine Fehlfunktion um jeden Preis verhindert werden). In der Praxis wird man wohl meist mehr oder weniger detaillierte Zugriffsbeschränkungen auf gefährdeten Systemen einsetzen, welche zwar die von-Neumann-Architektur belassen, aber schädigende Software daran hindern sollen, überhaupt in das System zu gelangen.

5. Firewalls

5.1. Der Begriff der Firewall

Ein Firewall-System besteht aus einer oder mehreren Komponenten, die den Zugriff zwischen einem geschützten Netzwerk und dem Internet oder zwischen mehreren Netzwerken beschränken. (Es ist durchaus möglich, dass Teilbereiche eines größeren Firmennetzes vor anderen durch eine Firewall geschützt werden, z.B. das Netzwerk der Forschungs- und Entwicklungsabteilung einer Firma vor dem restlichen, „unsicheren" Firmennetz). Eine Firewall ist gleichzeitig sowohl **„elektronischer Pförtner"**, als auch **„elektronische Brandschutzmauer"**. Die Aufgabe der „Brandschutzmauer" besteht darin, dass zu sichernde Netzwerk abzuschotten. Schäden, die im unsicheren Netz auftreten, also an der „Außenseite der Mauer", dürfen nicht in den inneren Bereich (das zu schützende Netzwerk) übergreifen. Die Aufgabe des „Pförtners" ist es, den Zugang zu dem gesicherten Netz und seinen Teilbereichen zu kontrollieren. Benutzer, die auf das zu schützende Netzwerk zugreifen wollen, müssen sich identifizieren und authentisieren. Der Pförtner muss prüfen welche Daten in das Netzwerk und welche aus dem Netzwerk gelangen dürfen. Zudem muss er alle Ereignisse protokollieren, um einen erfolgten Angriff besser nachvollziehen zu können und ggf. Rückschlüsse auf den/die Täter zu gewinnen. Die Firewall stellt den **Common-Point-Of-Trust** dar, d.h. es gibt nur einen Übergang zwischen dem zu schützenden Netz und dem unsicheren Netz, der über die Firewall läuft. Die Vorteile diese Konzepts liegen in

- **niedrigen Kosten**, da eine zentrale Sicherheitslösung günstiger ist als eine dezentrale Lösung auf jedem Rechner

- **Sicherheit**, da ein Firewall-System durch eine reduzierte Funktionalität möglichst wenig Angriffspunkte bieten sollte

- **Überprüfbarkeit** Durch den klaren Übergang zwischen den Netzen lässt sich die Protokollierung des Datenverkehrs einfach durchführen.

Prinzipiell soll eine Firewall die folgenden drei Dinge vor Angriffen aus dem Internet schützen:

- **Daten** (die Informationen auf den Computern)
- **Ressourcen** (die Computer selbst)
- **Eigene Identität**

5.1.1. **Daten.** Die **Vertraulichkeit**, **Integrität** und **Verfügbarkeit** der Daten muß gewährleistet sein. Vertraulichkeit meint, dass die Daten vor Unbekannten geheimgehalten werden. Es soll also verhindert werden, dass ein Angreifer aus dem Internet beispielsweise wichtige, firmeninterne Informationen lesen kann. Die Integrität soll sicherstellen, dass die Daten nicht von einem Angreifer manipuliert werden können. Als Verfügbarkeit wird die garantierte Nutzbarkeit der Daten bezeichnet, d.h. wenn die Daten benötigt werden, muss man auch darauf zugreifen können.

5.1.2. **Ressourcen** Es soll verhindert werden, dass Unbefugte sich unerlaubt Zugriff auf die Rechner z.B. eines Firmennetzwerks beschaffen können, um Computer-Ressourcen für eigene Zwecke zu verwenden.

5.1.3. **Eigene Identität.** Ein Angreifer der ein Firmennetzwerk kompromittiert hat, kann im Internet mit der Identität der Firma auftreten. In der Konsequenz heißt das, dass der Angreifer z.B. E-Mails im Namen der Firma verschicken, die Website des Unternehmens verändern oder die Identität für kriminelle Aktivitäten missbrauchen kann, was letztendlich vor allem dem Ruf der Firma schadet (wobei die entstehenden Kosten nicht vergessen werden dürfen).

5.2. Grundlagen der Netzwerktechnologie

Um die Funktionsweise von Firewalls verstehen zu können, ist ein grundlegendes Wissen über Datenpakete und Protokolle nötig. In diesem Abschnitt stellen wir daher in einer kurzen Einführung das ISO/OSI-Referenzmodell vor.

5.2.1. ISO/OSI-Referenzmodell. Um die verschiedenen Protokolle und Techniken der Vernetzung von Computer-Systemen zu vereinheitlichen wurde 1983 von der **ISO** (International Standardization Organization) das sog. **OSI-Referenzmodell** (Open Systems Interconnection) geschaffen. Das Ziel des Modells war es, durch die Verwendung von einheitlichen Schnittstellen unterschiedliche Netzkomponenten von verschiedenen Herstellern zusammen einsetzbar zu machen. Die Kommunikation wird bei diesem Modell in 7 Schichten (s. Fig.2) abgewickelt, die in beiden Rechnern implementiert sein müssen. Die Schichten

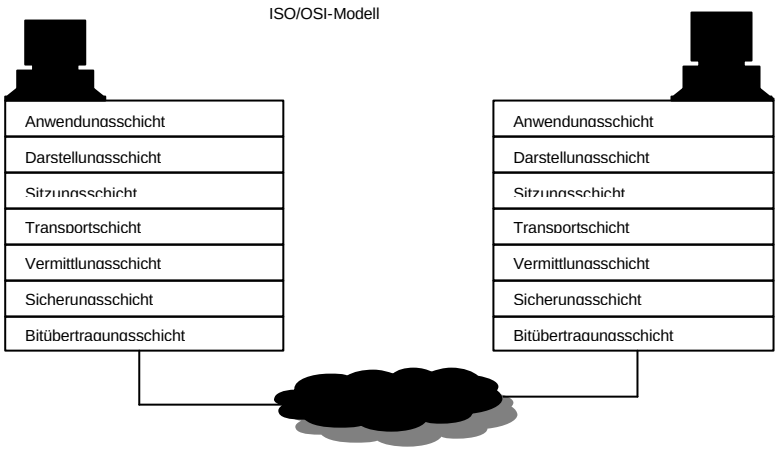

Fig. 2: OSI-Referenz-Modell

bauen dabei auf der Funktionalität der unteren Schichten auf, d.h. jede Schicht leistet für die jeweils übergeordnete Schicht spezielle Dienste. Die Kommunikation läuft immer auf gleicher Schichtebene ab, d.h. die Schicht *n* des einen Rechners kommuniziert immer mit der Schicht *n* des anderen Rechners, wobei ein spezifisches Protokoll verwendet wird. Dabei werden die Daten an die nächst tiefere Schicht weitergegeben, bis die unterste Schicht, das physikalische Übertragungsmedium erreicht ist. Dem eigentlichen Datenpaket werden von jeder Schicht Steuerinformationen hinzugefügt, der sog. **Header**. Man spricht bei diesem Vorgang von **Kapselung** (engl. encapsulation). Die Regeln mit deren Hilfe die Kommunikation gesteuert wird, werden **Protokoll** genannt. Zu den Hauptaufgaben der Protokolle gehören unter anderem die Adressierung des Quell- und Zielsystems, die Sicherstellung, dass die Daten unverfälscht, vollständig und in der richtigen Reihenfolge beim Empfänger ankommen, sowie die Festlegung der Übertragungsroute. Protokolle unterschiedlicher Schichten, die funktional oder historisch zusammengehören, bezeichnet man als **Protokoll-Stack**. Man unterscheidet zwischen **verbindungslosen** und **verbindungsorientierten Protokollen**. Bei verbindungslosen Protokollen werden die einzelnen Datenpakete unabhängig voneinander versandt, man spricht dann von **Datagrammen**. Bei dieser Art der Datenübertragung kann es passieren, dass ein später abgesendetes Paket vor einem früher gesendeten Paket beim Empfänger eintrifft. Im Unterschied dazu wird bei einem verbindungsorientierten Protokoll zunächst eine Verbindung zwischen den Kommunikationspartnern aufgebaut. Die Daten werden gesendet und nach der Übertragung wird die Verbindung wieder abgebaut. Auf die Funktion der einzelnen Schichten soll hier nicht näher eingegangen werden.

5.3. Firewall-Elemente

5.3.1. **Sicherheitspolitik.** Die Sicherheitspolitik definiert die Richtlinien für den Betrieb einer Firewall. Sie legt nach [6] unter anderem fest:

- **wie hoch der Schutzbedarf ist**, also welche Sicherheitsvorkehrungen tatsächlich getroffen werden müssen (reicht ein simples Firewallsystem oder lohnt sich die Anschaffung einer komplexen Variante, braucht man überhaupt eine Firewall?)
- **welche Systeme geschützt werden müssen**
- **welche Informationen verdeckt werden sollen**, z.B. der Aufbau der internen Netzstruktur oder Benutzernamen
- **wie das Kommunikationsprofil aussieht,** welche Benutzer gibt es?, wann benutzt ein bestimmter Benutzer normalerweise einen bestimmten Dienst? etc.
- **wie in Ausnahmesituationen verfahren wird**,
- **wer für Adminstration und Konfiguration der Firewall verantwortlich ist.**

5.3.2. **Security Management.** Ein weiteres wichtiges Element von Firewalls ist die Konfiguration und Admistration des Firewall-Systems. Die dafür eingesetzten Rechner und Programme werden als **Sicherheitsmanagement-Komponenten** bezeichnet. Diese sollten nach [6] unter anderem folgende Aufgaben erfüllen:

- Zugangskontrolle zur Konfigurationssoftware und den zugehörigen Dateien
- Eingabe und Kontrolle der Filterregeln zur Umsetzung der Sicherheitspolitik
- Eingabe und Kontrolle von Daten, die für den Betrieb der Firewall nötig sind (z.B. DNS-Informationen, Alias-Namen etc.)
- Protokolldatenmanagement (s. Intrusion Detection Systeme)
- Backupmanagement

5.3.3. **Paketfilterung.** Wie im letzten Abschnitt erläutert, wird der Datenverkehr in Rechnernetzen über Netzwerkprotokolle abgewickelt. Dabei werden Datenpakete vom Quell-Rechner zum Ziel-Rechner geschickt. Die Informationen über das jeweilige Paket finden sich in seinen Headern. Ein Paketfilter kann nun anhand dieser Headerinformationen ankommende bzw. abgehende Pakete kontrollieren. Über einen definierten Regelsatz (**Set of Rules**), der die Umsetzung der Sicherheitsrichtlinien darstellt, kann er dann entscheiden, ob ein Paket passieren darf oder nicht. Paketfilter können sowohl als Hardwarelösung (z.B. in Routern), als auch als Softwarelösung realisiert werden.. Die verwendeten Regeln können als **Erlaubnisregel** (alles ist verboten, bis auf die Dinge, die explizit erlaubt sind) oder als **Verbotsregel** (alles ist erlaubt, bis auf die Dinge, die explizit verboten sind) definiert werden. Die Anwendung von Verbotsregeln birgt allerdings die Gefahr, dass Regeln vergessen werden und somit Sicherheitslücken entstehen. Da Protokolle im allgemeinen bidirektional arbeiten, d.h. die eine Seite stellt eine Anfrage und die andere antwortet darauf, werden im Normalfall für jeden erlaubten Dienst zwei Regeln aufgestellt (eine für kommende und eine für gehende Pakete). Zur Analyse der Pakete nutzt der Paketfilter die Schichten 3 und 4 des ISO/OSI-Modells. Als Attribute verwendet er unter anderem die **Quell-/ und Zieladresse** des Pakets und die N**ummern der Sitzungs- und Anwendungsports**. Beispielsweise könnte man so auf einfache Weise alle Pakete blockieren, die versuchen von außen eine Verbindung über Telnet (Portnummer 23) aufzubauen oder die von einer bestimmten IP-Adresse gesendet wurden. Allerdings ist ein einfacher Paketfilter nicht in der Lage benutzerabhängige, oder

protokollspezifische Entscheidungen zu treffen. Sicherheitsbeschränkungen auf bestimmte Benutzer, Dateitypen oder Protokollbefehle wie put oder get bei FTP sind nicht möglich., da der Paketfilter auf den, ihm zugänglichen Protokollschichten, keinen Zugang zu diesen Informationen hat. Ein leistungsfähigeres System stellen zustandsgesteuerte oder dynamische Paketfilter dar. Diese erlauben eine Zustandsüberwachung und/oder Überprüfung der verwendeten Kommunikationsprotokolle.

5.3.4. **Application-Gateways.** Application-Gateways analysieren die Daten der Anwendungsschicht (Schicht 7 im ISO/OSI-Modell). Dazu gehören Authentisierungsinformationen oder spezifische Befehle der Protokolle der Anwendungsschicht. Sie verwenden dabei in der Regel sog. **Proxy-Prozesse**. Dabei handelt es sich um Programme, die als „Stellvertreter“ für einen internen Client mit einem externen, im unsicheren Netz liegenden, Server kommunizieren. Die Kommunikation mit dem Proxy läuft weitestgehend im Hintergrund und ist für den Client im Normalfall transparent. Da auf einem Rechner in der Regel mehrere Proxy-Prozesse installiert sind, spricht man auch manchmal von einem **Proxy-Server**. Allgemein unterscheidet man zwischen **Application-Level-Proxies** und **Circuit-Level-Proxies**.

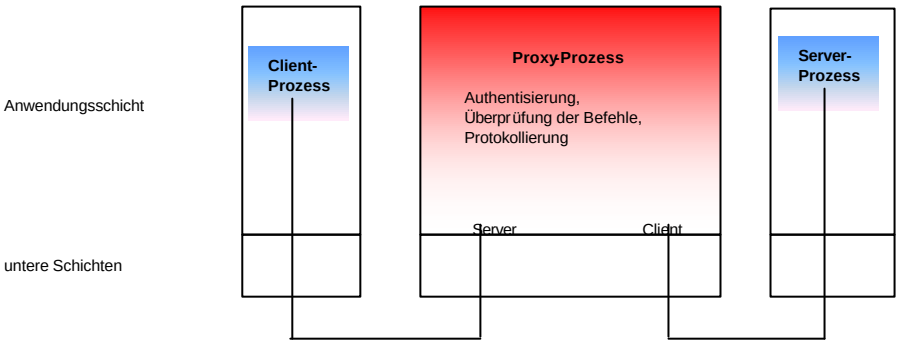

Fig. 3: Funktionsweise eines Proxies

- **Application-Level-Proxies.** Diese Proxies werden speziell für ein zu kontrollierendes Protokoll wie HTTP, FTP oder Telnet verwendet und verstehen die zum Protokoll gehörenden Befehle. Ein HTTP-Proxy kann zur Filterung zum Beispiel die URL oder den Inhalt eine HTML-Seite verwenden, ein FTP-Proxy könnte den Befehl put unterbinden.

- **Circuit-Level-Proxies.** Circuit-Level-Proxies sind generische Proxies, die für nahezu alle Protokolle eingesetzt werden können. Allerdings können sie keine protokollspezifischen Informationen zur Filterung und zur Protokollierung verwenden. Circuit-Level-Proxies benötigen daher modifizierte Clientprogramme, die sie mit zusätzlichen Informationen versorgen. Im Allgemeinen haben sie jedoch die gleiche Funktionalität wie Paketfilter, d.h. sie steuern den Datenverkehr anhand der Quell- und Zieladresse der Pakete. Sie werden meist eingesetzt, wenn für ein bestimmtes Protokoll kein Application-Level-Proxy existiert, das entsprechende Protokoll aber trotzdem durch die Firewall gelassen werden soll.

5.4. Firewall-Architekturen

Es gibt sehr viele verschiedene Möglichkeiten ein Firewall-System aufzubauen. Wir wollen im folgenden drei Beispiele vorstellen.

5.4.1. **Dual-Homed-Host.** Ein Dual-Homed-Host ist ein Computer mit mindestens zwei Netzwerkschnittstellen, der zwischen dem internen und dem externen Netzwerk platziert wird (s. Fig. 4). Das bedeutet er kann IP-Pakete vom einen in das andere Netzwerk weiterleiten. Diese Funktion wird aber abgeschaltet, so dass Datenpakete nicht direkt vom einen zum anderen Netzwerk geschickt werden. Die Kommunikation zwischen den Netzwerken läuft komplett über den Dual-Homed-Host, der **direkte Datenverkehr wird vollständig unterbunden**.
Dienste können demnach nur genutzt werden, wenn Proxies verwendet werden oder sich der Benutzer direkt am Host-Rechner einloggt. Die letzte Variante birgt nach [15] allerdings Risiken, da das Einrichten von User-Accounts auf dem Host zu Sicherheitsmängeln führen kann. Diese Architektur stellt eine sehr einfache Lösung dar (sog. **Single-Box-Architektur**, da die Firewall aus nur einem Gerät besteht), die dennoch einen hohen Kontrollgrad ermöglicht, aber auch viele Nachteile hat. Zum einen stellt der Dual-Homed-Host die einzige Schutzvorrichtung vor Angreifern aus dem unsicheren Netz dar (man spricht von einem **Bastion-Host**). Deshalb muss dafür gesorgt werden, dass der **Host an sich** sehr sicher eingerichtet ist, denn gelingt es einem Angreifer den Rechner zu kompromittieren, hat er Zugriff auf das gesamte innere Netzwerk. Außerdem sind diese Systeme nicht sehr leistungsfähig. Nach [12] muss ein Dual-Homed-Host bei jeder Verbindung mehr Arbeit erledigen und benötigt daher mehr Ressourcen als beispielsweise ein Paketfilter. Äquivalente Paketfiltersysteme erlauben zusätzlich auch einen höheren Datenverkehr.

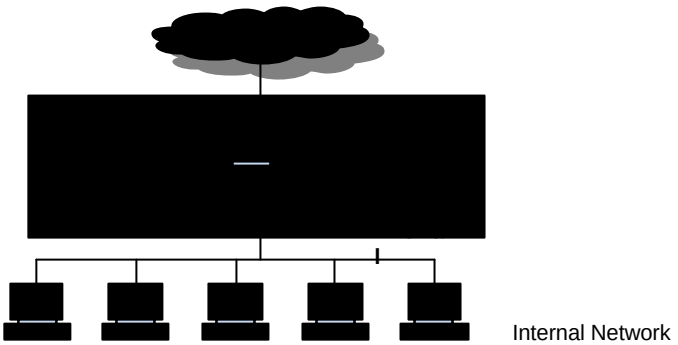

Fig. 4: Dual-Homed Host Architektur

5.4.2. **Screended Host.** Die Screened-Host-Architektur kombiniert einen Paketfilter mit einem Rechner, dem Bastion Host, der nur an das innere Netzwerk angeschlossen ist (s. Fig. 5). Der Paketfilter wird so konfiguriert, dass der Bastion Host das einzige System im inneren Netz ist, das aus dem Internet erreichbar ist, wobei auch hier nicht alle Arten von Verbindungen erlaubt sind. Gegebenfalls kann der Host auch „zulässige“ Verbindungen ins Internet aufbauen. (welche das sind, hängt allerdings davon ab wie die Sicherheitspolitik definiert wurde). Wegen der Position des Bastion Host im inneren Netzwerk gilt auch in diesem Fall, dass der Rechner möglichst sicher aufgesetzt werden muss. Der Paketfilter kann so konfiguriert werden, dass er entweder allen internen Rechnern die direkte Kommunikation mit dem Internet verbietet (d.h. es müssten die Proxies auf dem Bastion-Host verwendet werden) oder bestimmte Dienste zu externen Rechnern erlaubt (auch das hängt von der Sicherheitspolitik ab). Ein gravierender Nachteil dieser Architektur ist die Position des Bastion Host im internen Netzwerk, da es keine Schutzmöglichkeit mehr zwischen dem Bastion-Host und den restlichen Hosts im inneren Netzwerk gibt. Gelingt es

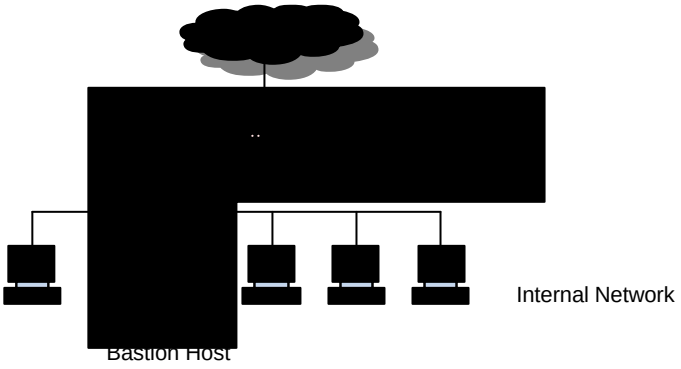

Fig. 5: Screened Host Architektur

außerdem einem Angreifer den Paketfilter zu überwinden kann er auf das gesamte Netzwerk zugreifen. Trotzdem bietet diese Lösung im Vergleich zum Dual-Homed-Host eine höheren Schutz und eine bessere Nutzbarkeit, da nach [15] „die Dual-Homed-Host-Architektur in der Praxis dazu neigt fehlerhaft zu arbeiten und Pakete aus dem externen Netzwerk in ein internes Netzwerk durchlassen“.

5.4.3. **Screened Subnet.** Bei dieser Architektur wird zwischen innerem und äußerem Netzwerk ein **Grenznetzwerk** eingefügt, das sogenannte **Perimeternetzwerk** (s. Fig. 6). Dieser Bereich wird manchmal auch als DMZ bezeichnet für De-Militarized Zone, nach der Zone zwischen Nord –und Südkorea. Bei der einfachsten Variante dieser Architektur gibt es einen **inneren Router**, der das Grenznetz mit dem inneren Netz verbindet und eine **äußeren Router**, der das Grenznetz mit dem externen Netz verbindet. Der Bastion Host wird dabei ebenfalls im Grenznetz platziert (s. Abb.). Der große Vorteil dieser Lösung ist nun, dass ein Angreifer, der in das innere Netzwerk einbrechen will, an beiden Routern vorbei muss. Selbst wenn es ihm gelingt Zugriff auf den Bastion Host zu erlangen, so hat er, im Unterschied zu den anderen beiden vorgestellten Lösungen keinen vollen Zugriff auf das innere Netzwerk, da er noch den inneren Paketfilter überwinden müsste. Es gibt also keinen „single point of

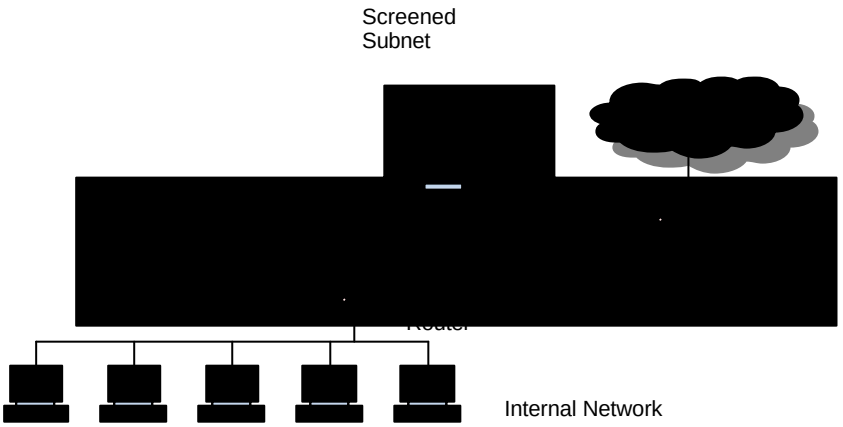

Fig. 6: Screened Subnet Architektur (mit zwei Routern)

failure“ mehr der das innere Netzwerk verwundbar macht., daher eignet sich diese Lösung nach [12] für fast alle Anwendungen.

5.5. Protokollierung, Analyse und Reaktion (Intrusion Detection Systeme)

Ein sehr wichtiger Aspekt beim Betrieb einer Firewall ist die Protokollierung und Analyse des Datenverkehrs, da anhand dieser Informationen mögliche Konfigurations- und Programmierfehler oder Angriffsversuche entdeckt werden können. Daher gilt, dass die einzelnen Firewall-Komponenten so viele Protokolldaten wie möglich erzeugen sollten. Da es sehr umständlich sein kann, die wirklich relevanten Informationen von den unwichtigen zu trennen, gibt es Programme, die diese Arbeit automatisieren (bsp. *swatch* und *logsurfer*). Die extrahierten Meldungen können dann dazu verwendet werden einen Angriffsversuch zu erkennen oder einen bereits erfolgten Angriff besser nachvollziehen zu können. Programme, die automatisch erkennen, ob ein Angriff auf das Netzwerk erfolgt ist bezeichnet man als Intrusion Detection Systeme (IDS). Intrusion Response-System (IR) ergreifen hingegen automatische Gegenmaßnahmen im Falle eines Angriffs. Auf letztere wird hier jedoch nicht näher eingegangen. Es gibt zwei prinzipielle Ansätze nach denen IDS funktionieren. Zum einen gibt es sog. **Anomalie-Erkennungsprogramme**, die die anfallenden Daten auf ungewöhnliche, vom „normalen Benutzerverhalten“ abweichende Aktionen überprüfen. Allerdings ist es sehr schwer das „normale Benutzerverhalten“ zu definieren und die Grenze zu finden wann es sich um einen Angriff handelt und wann nicht. Im Allgemeinen werden dazu über einen längeren Zeitraum Daten erfasst und anhand von bestimmten Merkmalen gespeichert (z.B. die „normale Arbeitszeit“ eines Benutzers). Erkennt das IDS eine Abweichung

von diesen gespeicherten Merkmalen, dann kann es eine Warnmeldung an den Administrator verschicken. Der andere Ansatz geht von der sog. Signaturanalyse aus, die ähnlich funktioniert wie ein Virenscanner. Dabei vergleicht das IDS die Protokolldaten mit einer Datenbank in der alle bis dato bekannten Angriffssignaturen gespeichert sind. Erkennt das IDS die Merkmale eines bestimmten Angriffs wird eine Warnmeldung an den Administrator geschickt. Nachteil dieser Methode ist, daß niemals alle Angriffssignaturen bekannt sind und daher neuartige Angriffe, für die sich keine Merkmale in der Datenbank befinden, von solchen Systemen nicht abgefangen werden können.

6. Ausblick

Betrachtet man die Schadensstatistiken über die Auslöser von Datenverlusten in den letzten Jahren, so stellt man fest, daß ca. 16% aller Datenverluste auf Infiltration mit schädigenden Programmen zurückzuführen sind; dagegen stellen die Datenverluste durch Hardwareschäden oder Benutzerfehler mit ca. 70% den größten Betrag dar. Insofern kann man die Gefährdung eines Rechnersystems durch automatische Infiltrationsverfahren als „eine Gefahr unter vielen“ ansehen, die jedoch mit Blick auf die bestehende Rechnerarchitektur betrachtet werden sollte. Ein völlig sicherer Schutz dürfte mit vertretbarem Aufwand nicht erreichbar sein; er entspräche einer Architektur ähnlich den heutigen Klein-Spielekonsolen, auf welche Programme nur von per se schreibgeschützten (und oft auch nicht auslesbaren) Kassetten eingespielt werden; können sonst keine Datenträger verwendet werden, ist das System völlig sicher, aber auch – für ernsthafte Anwendungen – bis zur Unbrauchbarkeit unflexibel.

Als Prognose für die nächste Entwicklung dürfte sich der Wunsch der Systemnutzer nach „sicheren“ Systemen wohl noch erhöhen; Grund ist, daß die meisten der heutigen Nutzer nicht mehr derart „intime“ Kenntnisse über den Aufbau ihrer Systeme besitzen wie in den ersten Jahren nach der Einführung des Personalcomputers – und auch nicht besitzen wollen: Nach unseren Erfahrungen will ein immer größerer Teil der Systemnutzer seine Aufgaben am Rechner erledigen können, ohne sich um die Interna kümmern zu müssen; entsprechend wächst der Anspruch, Rechnersysteme sollten „auf sich selbst aufpassen“. Die früher exzessiv verbreiteten Computerviren dürften in absehbarer Zeit ihre Bedeutung verlieren, da ihre Verbreitung durch Austausch „verseuchter“ Programme oder Datenträger geschieht; heutige Programmpakete sind derart umfangreich, daß ein umfangreicher Tauschhandel unattraktiv wird; die weitgehende Ablösung der Diskette durch nicht ohne weiteres startfähige CD-ROM oder CD-RW wird die klassischen Startsektorviren aussterben lassen. Der Trend geht eindeutig zu netzbasierter Schädigungssoftware, die in Hochsprachen oder den Skriptsprachen von Anwendungen geschrieben sind; hier dürften vor allem E-Mail-Würmer und der verteilte Denial-of-Service-Angriff eine Renaissance erleben. Schließlich wird sich die Person des Angreifers verändern: Während früher wirklich gefährliche, ausgeklügelte Assembler-Programme in wochenlanger Kleinarbeit „zusammengebastelt“ wurden, gestatten es seit dem Aufkommen der graphischen Benutzeroberflächen sogenannte „Virii-Creators“ (vgl. [12]) jedem Benutzer, ein Schadensprogramm nach seinen Wünschen zu erstellen. Da diese Programme meist E-Mail-Würmer erstellen, liegt die Hauptaufgabe des schadenswilligen Benutzers nun darin, eine attraktive „Verpackung“ zu gestalten. Die in [2] und [8] aufgezeigten „sinnvollen“ Verwendungen für automatische Infiltrationsprogramme (z.B. der „Kompressionsvirus“, automatischer Datenabgleich in verteilten Systemen, evolutionäre Programmentwicklung) werden sich in absehbarer Zeit nicht durchsetzen, erstens wegen des negativen Images der schädlichen Varianten, zweitens wegen der schwierigen Rechtslage (speziell des Urheberrechts) und schließlich wegen der Forderung der Endnutzer, sich um ihre Rechner nicht kümmern zu müssen. Auch Firewalls bieten keine absolute Sicherheit vor Angriffen auf das Netzwerk. Zum einen geht das Firewall-Konzept grundsätzlich davon aus, dass der Angreifer sich außerhalb des Netzwerks befindet. Dementsprechend können Attacken von „Insidern“ nicht verhindert werden. Dabei muss die betreffende Person nicht einmal absichtlich handeln. Die Problematik des „Social Engineering“ zeigt, dass ein Angreifer mitunter auf verblüffend einfache Weise z.B. an Benutzerpasswörter gelangen und sich damit das Ausspionieren auf technischem Wege ersparen kann. Diesen potentiellen Gefahren lassen sich nur durch interne Sicherheitsvorkehrungen, wie der Einführung von Benutzerrechten auf den Rechnern oder Mitarbeiterschulungen vermeiden. Aber selbst dann wäre keine absolute Sicherheit garantiert. In manchen Firmen finden sich auch „Hintertüren“ (sog. Backdoors), wie z.B. eine Modemverbindung ins Internet, die an der Firewall „vorbeiführen“. In diesem Fall kann selbst das teuerste Firewall-System nicht helfen, da die über diese

Leitung gehenden Daten von der Firewall natürlich nicht kontrolliert werden können. Firewalls bieten zudem nur eingeschränkten bis gar keinen Schutz vor Sabotageprogrammen, da es sehr schwierig ist in einem beliebigen kontrollierten Datenpaket beispielsweise die Funktion einen Virus zu entdecken. Außerdem besteht natürlich auch hier die Gefahr, dass ein interner Benutzer Viren in das zu schützende Netzwerk einschleust. Allerdings wird deutlich, dass es sich dabei nicht um grundlegende Nachteile der Firewall selbst handelt, sie ist nur ein Teil einer umfassenden Sicherheitspolitik. Erst das Zusammenspiel all dieser Komponenten kann zu einem relativ, wenn auch nicht absolut sicherem Netzwerk führen.

Literatur

[1] Jürgen Borngießer. *Der Wolf im Software-Pelz.* Internet-Magazin 6/2001, WEKA Verlag, München 2001

[2] Ralph Burger. *Computer viruses – a high-tech disease.* Abacus-Verlag, 1987

[3] Fred Cohen. *Computerviruses, Theory and Experiments.* University of Southern California, 1984.

[4] Cydoor Technologies Homepage, www.cydoor.com

[5] Thomas Dehn. *Virenschutz.* Deutscher Taschenbuch Verlag 1993

[6] Kai Fuhrberg: *Internetsicherheit. Browser, Firewalls und Verschlüsselung.* 2. Auflage, Hanser 2000

[7] Daniel Ch. Kreiss. *Neugierig wie der Teufel.* PC Games 7/2001 S.20-21, Computec Media, Nürnberg 2001

[8] Steven Levy: *Künstliches Leben aus dem Computer.* Knaur 1996

[9] Michael Matzer. *Sicherheitsrisiko Internet.* dtv 1999

[10] Norbert Pohlmann. *Firewall-Systeme*

[11] Radiate Homepage, www.radiate.com

[12] Jürgen Schmidt. *Virenbasteln für Dummies. Wachsende Gefahr durch Schädlinge aus dem Baukasten.* c't 13/2001 S. 98-101 Heise Verlag, Hannover 2001

[13] Jürgen Schmidt. *Schädlingsbekämpfung. Konzepte gegen Viren und andere Schädlinge.* c't 2/2001, S. 98-101, Heise Verlag, Hannover 2001

[14] Frank Ziemann. *Bei Anruf Update. Techniken moderner Malware.* c't 2/2001 S. 116-119. Heise Verlag, Hannover 2001

[15] Elisabeth D. Zwick, Simon Cooper, D. Brent Chapman. *Einrichten von Internet Firewalls.* o'Reilly 2001