WIRTSCHAFTSUNIVERSITÄT WIEN

MAGISTERARBEIT

Titel der Magisterarbeit:

Firewalls unter dem Aspekt von Virtuellen Maschinen

Verfasserin/Verfasser: Florian Kogelbauer BSc (Wu)

Matrikel-Nr.: 0350819

Studienrichtung: WINF-M03

Beurteiler: Univ. Prof. Dipl.-Ing. Mag. Dr. Wolfgang Panny

Betreuender Assistent: Dipl.-Ing. Mag. Dr. Albert Weichselbraun

Ich versichere:

dass ich die Magisterarbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe.

dass ich dieses Magisterthema bisher weder im In- noch im Ausland (einer Beurteilerin/ einem Beurteiler zur Begutachtung) in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

dass diese Arbeit mit der vom Begutachter beurteilten Arbeit übereinstimmt.

Datum

Unterschrift

Firewalls unter dem Aspekt von Virtuellen Maschinen

Florian Kogelbauer

10. September 2009

Wirtschaftsuniversität Wien

Abstract

Die vorliegende Arbeit befasst sich im ersten theoretischen Teil mit Firewalls, als auch den gegenständlich verwendeten Protokollen der Netzwerke sowie deren Schwächen. Im zweiten Teil werden Virtuelle Maschinen von Grund auf, bis hin zu konkreten Implementierungen behandelt. Ebenso ist die sicherheitstechnische Betrachtung dieser Technologie in diesem Teil beschrieben. Die ersten beiden Kapitel dienen somit dem Leser als Einleitung sowie als Grundlage zum Verständnis der beiden Technologien. Im dritten Teil der Arbeit werden spezifische Gesichtspunkte der Technologien bei gemeinsamer Verwendung behandelt. Gleichfalls wird im dritten Teil auf eine praktische Realisierung sowie deren Erweiterung zur gekoppelten Verwendung eingegangen. Dies wird mittels eines selbst erstellten Tools zur Generierung von Firewall-Rules erreicht und soll dem Leser konzeptionell verdeutlicht werden.

Stichworte:

Firewalls, Firewallkonzepte, Paketfilter mit iptables, Firewall Builder, Protokollsicherheit, Virtuelle Maschinen, VM Produkte, Xen, KVM, VM Sicherheit, VM Cluster, Firewalls unter VMs

Kernpunkte:

Die vorliegende Arbeit soll verdeutlichen, dass eine kombinierte Verwendung der beiden Technologien zu einer Erhöhung der Sicherheit dienen kann. Die Adaption vorhandener Software soll dem Trend der fortwährenden Virtualisierung Rechnung tragen.

- Sinnvoller Einsatz sowie abgestimmte Konfiguration der Komponenten können die Sicherheit in virtualisierten Umgebungen erhöhen.
- Firewalls können in virtuellen Maschinen, welche zur Bereitstellung von Gast-OS geeignet sind, anstandslos eingesetzt werden, dies jedoch nur unter Berücksichtigung der dezidierten Gegebenheiten.
- Firewalls können die Sicherheit zwar maßgeblich erhöhen, jedoch ist der Sicherheitsgewinn gegenüber von Hypervisor-Lösungen aufgrund der Ausführung in der Applikationsebene geringer.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

Der nachfolgende Abschnitt soll grundlegende Eigenschaften der vorliegenden Arbeit beschreiben, sowie die damit verbundenen Rahmenbedingungen abklären. Eingangs werden die Zielsetzung der Arbeit, sowie die damit verbundene Forschungsfrage erläutert. Anschließend wird auf die Struktur der Arbeit eingegangen und die Erstellung dieser in einen zeitlichen Rahmen gefasst. Abschließend soll der Zusammenhang zu anderen, von meiner Person, publizierten Arbeiten abgegrenzt werden.

1.1. Zielsetzung

Zielsetzung der Arbeit ist die Darstellung der Besonderheiten, welche bei der Verwendung von Firewalls in Verbindung mit Virtuellen Maschinen auftreten. Darüber hinaus sollen grundlegende Konzepte der Technologien dargestellt werden, als auch die dadurch determinierten Einsatzgebiete. Weiters sollen konkrete Realisierungen, in den entsprechenden Teilbereichen dargestellt werden und Verbesserungen angesprochen werden.
Zusätzlich soll eine konkrete Softwarelösung – Firewall Builder – in Hinblick auf Virtuelle Maschinen vorgestellt werden. Hierbei soll auf die konkreten Anforderungen sowie die damit verbundenen Problembereiche eingegangen werden. Weiters soll eine experimentelle Erweiterung zur Unterstützung von VMs erstellt, sowie erläutert werden.
Die Dreiteilung der Arbeit soll ermöglichen, die einzelnen Themenbereiche gesondert, aber auch in Kombination betrachten zu können. Der Leser soll die Möglichkeit erhalten, sich umfassendes Wissen in den unterschiedlichen behandelten Gebieten anzueignen, wobei auf eine konzeptionelle Darstellung geachtet wird, um die Portierbarkeit des Wissens zu ermöglichen.

1.2. Forschungsfrage

Die grundlegende Forschungsfrage dieser Masterarbeit ist die Feststellung von Besonderheiten bei der Verwendung von Firewallsystemen in Verbindung mit Virtuellen Maschinen. Die Feststellung dieser Eigenheiten auf theoretischer Basis, als auch die konkrete Realisierung sollen Gegenstand dieser Arbeit sein. Weiters soll untersucht werden, in wie weit die Besonderheiten bei den konkreten Realisierungen berücksichtigt werden können. Die Ausarbeitung dieser Fragestellungen basiert vorwiegend auf folgenden Methoden:

- Literaturrecherche zu den Themengebieten
- Praktische Umsetzung ausgewählter Lösungen und Dokumentation der Ergebnisse

Weiters sollen etwaige Besonderheiten bei der Verbundbildung von Virtuellen Maschinen und dessen Auswirkungen auf Firewalls untersucht werden.

1.3. Thesenstruktur

Um die Zusammenhänge erklären zu können, werden zuvor die Themen Firewalls, als auch der Virtuellen Maschinen getrennt behandelt. Dies soll dem Leser die Möglichkeit geben, einen Überblick über die Leistungsfähigkeit sowie Zweckmäßigkeit der beiden Technologien zu bekommen. Darüber hinaus werden konkrete Realisierungen dargestellt, welche den Bezug zur Praxis herstellen sollen.
Bezüglich der Firewalls werden, nach allgemeiner Definition (Aufgaben, Abgrenzung, Einsatzgebiete), zuerst grundlegende Netzwerkkenntnisse vermittelt, welche den Leser für sicherheitstechnische Missstände sensibilsieren sollen. Darüber hinaus werden klassische Angriffe und Angriffsmethoden erläutert. Weiters soll der Einsatz von Firewalltechnologie zur Vermeidung beziehungsweise Unterbindung dieser Angriffe dargestellt werden. Danach wird auf die unterschiedlichen Konzepte der Firewalls eingegangen und eine konkrete Realisierung (Packetfilter mittels iptables) vorgestellt. Abschließend wird erläutert, welche Angriffe auf Firewallsysteme es gibt und wie diese erkannt werden können.
Bezüglich der Virtuellen Maschinen werden, nach allgemeiner Definition, die Aufgaben, sowie die geschichtliche Entwicklung dargestellt. Anschließend wird auf die unterschiedlichen Virtualisierungstechniken eingegangen. Weiters werden die Konzepte des Bridgings,

als auch der Virtuellen Netzwerkadapter erläutert, welcher im Abschnitt der gemeinsamen Nutzung besondere Bedeutung zukommen. Danach werden konkrete Virtualisierungslösungen vorgestellt und auf die sicherheitstechnischen Fragen eingegangen.
Im letzten Abschnitt der Arbeit wird der kombinierte Einsatz der zuvor vorgestellten Technologien dargestellt. Zu Beginn werden die speziellen Anforderungen, welche an Firewalls unter Virtuellen Maschinen gestellt werden, dargestellt. Anschließend wird der Einsatz von Firewalls im VM-Verbund und der damit verbundenen Besonderheiten betrachtet. Abschließend soll eine konkrete Lösung zur Erstellung von Firewall Policies vorgestellt werden. In diesem Zusammenhang soll untersucht werden, ob eine hinreichende Unterstützung für Virtuelle Maschinen und der damit verbundenen virtuellen Netzwerkadapter gegeben ist. Weiters werden konkrete Erweiterungen dargestellt und eine experimentelle Realisierung vorgestellt.

1.4. Abgrenzung zu anderen publizierten Arbeiten

Die Abgrenzung bezieht sich im Speziellen auf die bereits veröffentlichten Bakkalaureatsarbeiten des Autors, wovon eine in Zusammenarbeit mit einem, hier nicht explizit genannten Unternehmen erarbeitet wurde. Die zweite Arbeit stellt eine theoretische Ausarbeitung eines sicherheitstechnischen Themas dar [Kog07, Kog08].

Bezüglich der Bakkalaureatsarbeit [Kog07][1], welche in Zusammenarbeit mit einem Unternehmen eine praktische Realisierung eines Softwaremoduls darstellt, ist keinerlei Überschneidung festzustellen. Die darin behandelten Themen der Erstellung eines Softwaremoduls für die Dokumentenverteilung an die Kunden, als auch die Verzeichnisdienste (insb. OpenLDAP) werden daher ausgegrenzt.

Bei der Bakkalaureatsarbeit [Kog08][2] hingegen sind auf Grund der Themenverwandtschaft (Sicherheitstechnik und grundlegende Netzwerkstrukturen) Überschneidungen festzustellen. Im Speziellen betrifft dies die Grundlagen der Netzwerktechnik, als auch die darin definierten Referenzmodelle [Kog08, S. 6–21]. Basierend auf diesen Grundlagen wird eine Vertiefung bis hin zur Bit–(Sequenz)Ebene angestrebt, welche als Grundlage für das Filtern beziehungsweise den Einsatz von Firewalls verstanden werden kann.

Bei der Thematik der Virtuellen Maschinen ist keine Themenverwandtschaft zu den bisher publizierten Arbeiten festzustellen, weshalb eine klare Abgrenzung leicht möglich ist.

[1] [Kog07]
Kogelbauer, Florian: *Elektronische Dokumentenverwaltung und -bereitstellung auf einer Website durch Implementierung eines Zusatzmoduls DokuSERVE.* Wirtschaftsuniversität Wien, 2007.

[2] [Kog08]
Kogelbauer, Florian: *Virtual Private Networks - Grundlagen, Geschichte sowie aktuelle Entwicklungen.* Wirtschaftsuniversität Wien, 2008.

Teil I.

Firewalls

2. Firewall - Grundlagen

Das nachfolgende Kapitel soll dem Leser Auskunft über die Grundlagen sowie die Aufgaben von Firewalls und -systemen geben. Dieses Kapitel dient somit der Erläuterung der Grundideen, welche hinter der Firewall stehen und auf konzeptueller Basis betrachtet werden sollen. Weiters wird versucht eine passende Begriffsdefinition zu finden und den Leser für das Sicherheitskonzept zu sensibilisieren. Dies soll auf einer möglichst abstrakten und theoretischen Ebene erfolgen, um etwaige technische Verständnisprobleme zu minimieren oder gar auszuschließen. Darüber hinaus sollen die Aufgaben gegenüber anderen Sicherheitskonzepten abgegrenzt werden und ein Überblick über derzeitige (und sinnvolle) Einsatzbereiche gegeben werden.

2.1. Begriffsdefinition

Entsprechend der weitverbreiteten Verwendung von Firewalls und -systemen haben sich im Laufe der Zeit eine Vielzahl unterschiedlicher Definitionen gebildet. Diese unterscheiden sich zumeist sowohl im Detailierungsgrad als auch im definierten Funktionsumfang. Nachfolgend werden einige Beispiele (unterschiedlicher Komplexität) – ohne Anspruch auf Vollständigkeit – genannt und eine eigenständige Definition geliefert.
Die nachfolgenden Definitionen sollen einen Ausschnitt - ohne Anspruch auf Vollständigkeit - möglicher Ansichten geben:

> „Unter einer Firewall verstehen wir ... ein Produkt, das anhand definierter Regeln in der Lage ist, den Netzwerkverkehr zu filtern, so dass nur erwünschte Netzwerkpakete ihr Ziel erreichen“ [FG05].

> „Firewalls sollen als Zugangsschutzsysteme den Zugriff auf diverse Netze oder Rechner kontrollieren und teilweise auch den Traffic von innen nach außen beschränken“ [WP07].

> „. . . eine Firewall . . . kann man sich wie das Burgtor einer mittelalterlichen Stadt vorstellen. . . . Jeglicher Verkehr muss sie passieren und wird von ihr untersucht . . . “ [Les06].

Für diese Arbeit definieren wir die allgemeine Funktionalität einer Firewall wie folgt:

> Unabhängig vom Anwendungsgebiet besteht die Aufgabe der Firewall darin, den Netzwerkverkehr anhand von vordefinierten oder angelernten Regeln zu filtern und nur für gültig befundene Pakete passieren zu lassen.

Dies bedeutet, dass unabhängig von der Ausführung (Paketfilter, Proxy, Personal Firewall) eine Firewall den Netzwerkverkehr präventiv filtert. Die Filterung basiert dabei auf definierten Regeln, wobei diese vordefiniert sein können oder aber auch im Betrieb angelernt wurden. Es sei hervorgehoben, dass sonstige Funktionen (Logging, Alerting, Verschlüsselung, . . .) bei dieser allgemeinen Definition ausgeklammert wurden, da diese zumeist in Verbindung mit speziellen Softwareprodukten angeboten werden und bei diesen zumeist als Erweiterungen implementiert sind. Diese „sonstigen Aufgaben“ werden unter anderem im nächsten Unterpunkt behandelt, worauf an dieser Stelle verwiesen werden soll.

2.2. Aufgaben der Firewall

Um die Aufgaben, sowie den Nutzen von Firewalls zu veranschaulichen, wird die allgemein (in der Literatur) verbreitete Ansicht der Firewall als Stadttor verwendet [Les06]. Ähnlich zu einem Stadttor, welches den einzigen Eingang zur Stadt repräsentiert, fungiert die Firewall als Eingang zum Rechner beziehungsweise zur Netzwerkinfrastruktur.

Dieser „Knotenpunkt“ beziehungsweise „Schleusen“-Ansatz hat den enormen Vorteil, dass jede Person, beziehungsweise bei uns jedes (Netzwerk-)Paket, begutachtet beziehungsweise durchsucht werden kann. Dieses Durchsuchen findet, wie bereits in der Definition hervorgehoben, basierend auf gewissen Regeln beziehungsweise Regelsätzen statt, welche in detailierter Form in den entsprechenden Realisierungen behandelt werden. Regelkonforme Pakete beziehungsweise Anfragen werden sinngemäß weitergeleitet, wobei an dieser Stelle auch festgelegt werden kann, welche Dienste überhaupt (und wem) zur Verfügung stehen. Ein unsicherer Dienst, der keine zureichende Verschlüsselung des

Benutzernamen//Passwort-Paares gewährleistet, kann so zum Beispiel im Vorhinein ausgegrenzt werden.
Darüber hinaus können nicht nur Pakete sondern auch Inhalte von einer Firewall gefiltert werden. Dies kann zum Beispiel Vorteile bei der Verwendung von Webservices bieten. Weiters werden meist nur jene Pakete weitergeleitet, welche zuvor aus dem lokalen Netz angefordert wurden, wodurch eine systematische Suche von Schwächen im lokalen Netzwerk erschwert wird.

Ein weiterer Vorteil dieses Ansatzes des Stadttores beziehungsweise der Schleuse ist die naturgemäße Zentralisierung, welche auch als Bündelung der „Kraft“ angesehen werden kann. Dies hat den enormen Vorteil, dass nicht alle, der Firewall nachgelagerten Systeme lokal verwaltet, administriert und aktualisiert werden müssen. Dadurch werden sozusagen präventiv bereits gewisse Gefahrenquellen (zum Beispiel Out-of-Date Situationen, mutwillige Beschädigung, ...) ausgeschlossen und ein Umgehen erschwert. Unter der Bündelung der Kraft kann eine Konzentration auf die zentralisierte Stelle beschrieben werden. Dies ermöglicht eine Ausarbeitung und Implementierung eines hochsicheren Systems, welches aufgrund der Vielfältigkeit bei einer verteilten Realisierung kaum möglich beziehungsweise sinnvoll wäre. Diese Bündelung reduziert zudem die Angriffsfläche um ein Vielfaches.

Firewalls werden zudem verwendet um Netzwerkresourcen vor unbefugtem Zugriff zu schützen. Dieser Ansatz, bei dem ein Netzwerk von anderen durch eine oder mehrere Firewalls gesichert wird, bezeichnet man als die Bildung einer sogenannten „DMZ“ (Demilitarized Zone) was frei als demilitarisierte Zone übersetzt werden kann.

Die primären Aufgaben der Firewall sind:

- **Filterung des Netzwerkverkehrs sowie -zugriffs** und gegebenenfalls auch deren Inhalte in vielschichtiger Art und Weise zur Schaffung einer kontrollierbaren Netzwerkumgebung beziehungsweise -infrastruktur.

Neben dieser „Basisaufgabe“ gibt es noch eine Fülle von weiteren Möglichkeiten beziehungsweise Aufgaben die eine Firewall wahrnehmen kann [FG05]:

- **Logging** stellt als ereignisbasierte Aufzeichnung eine der wichtigsten Zusatzfunktionen dar. Die hohe Relevanz dieser Aufgabe kann zum einen durch die Hilfestellung bei Problemen, als auch durch die Analysefähigkeit von Angriffsversuchen erklärt werden. Ein sogenannter Log-Eintrag besteht zumindest aus einem Zeitstempel sowie der IP Adressen und Ports. Besonders interessant

sind hier die Einträge der abgewiesenen Pakete, wobei hier noch zusätzlich der Grund für das Abweisen aufgezeichnet wird. Diese Aufzeichnungen ermöglichen eine nachträgliche Analyse über etwaige Angriffsversuche sowie das Setzen von erforderlichen Maßnahmen um diese zukünftig zu verhindern.

- **Alerting** tritt im Wesentlichen in Kombination mit Logging auf und bezeichnet die Verständigung einer Person beziehungsweise eines Programms (Überwachungskonsole). Dieses Verständigen kann zum Beispiel mittels Versenden einer E-Mail an den Systemadministrator über ein geblocktes Paket realisiert werden.

- **Accounting** kann als Spezialfall des Loggings beschrieben werden. Hier werden bestimmte Anfragen (zum Beispiel HTTP-Requests und -Replies), sowie die damit verbundene Byte-Anzahl ausgewertet und anhand der IP-Adressen zugewiesen. So wird zum Beispiel eine Verrechnung zu Cost-Centers (innerbetrieblich) ermöglicht. Weiters können bei dieser Art des Loggings Sammelberichte über interessante Teilbereiche (zum Beispiel Verwendung von unsicheren Protokollen) erstellt werden.

- **Caching** bezeichnet das Halten von lokalen Duplikaten bestimmter Online-Inhalte um eine Reduktion des externen Datentransfers zu erreichen. Aufgrund des vielfältigen Angebots an spezialisierten (und eigenständigen) Caching-Lösungen soll hier nur erwähnt sein, dass dieses auch direkt als Firewall-Funktionalität implementiert werden könnte.

- **Authentifizierung** beschreibt den Umstand, dass ein Benutzer gegenüber eines Dienstes (der Firewall) seine Identität eindeutig bescheinigt. Damit geht einher, dass jenem Benutzer bestimmte Zugriffsrechte und auch -ebenen zugesprochen werden. Bei der Firewall Authentifizierung wirkt die Firewall als Client gegenüber dem Authentifizierungs-Server, welcher eine Rückmeldung an die Firewall gibt, ob ein Benutzer entsprechende Rechte hat oder nicht. Auf dieser Antwort basierend, werden dann die entsprechenden Dienste beziehungsweise Freigaben freigeschaltet. Dies ist besonders wichtig, falls extern auf unternehmensinterne Dienste (Mail-Server, Fileserver, ...) zugegriffen wird. In diesem Zusammenhang seien die schwache, sowie die starke Authentifizierung genannt [FG05]. Wobei erstere einem statischen Passwort entspricht und mittlerweile als obsolet, da unsicher, angesehen wird. Die starke Authentifizierung wird zumeist als Einmal-Passwort-System implementiert, wobei sogenannte Hardware-Tokens Verwendung finden. Ist zu

diesem Einmal-Passwort, welches seine Gültigkeit nach einmaliger Verwendung verliert, noch zusätzlich ein PIN erforderlich so spricht man von einer Zwei-Faktor-Authentifizierung. Wobei der Besitz von nur einer Ressource, dem Einmal-Passwort oder dem PIN, nicht berechtigt die freizugebenden Ressourcen zu nutzen.

- **Adressübersetzung oder NAT** (Network Address Translation) bezeichnet die Umwandlung von IP-Adressen und Ports (ebenso TCP und UDP) in eine andere IP-Adressen und Ports Kombination. Für die Anwendung dieser Übersetzungstätigkeit gibt es primär zwei relevante Gründe. Zum einen ist dies die Begrenzung der verfügbaren offiziellen IP-Adressen, weshalb netzwerkintern private Adressen (RFC 1918) eingesetzt werden. Zum anderen ist es nicht wünschenswert, dass Rechner direkt mit dem Internet verbunden sind, da auf ihnen möglicherweise bestimmte Ports ständig „offen“ sind.
  Bei der Adressübersetzung werden die internen, also nicht offiziellen, IP-Adressen in der Firewall auf eine öffentliche IP-Adresse des Unternehmens umgewandelt. Dies ist deshalb sinnvoll, da die interne Adresse aus dem Internet nicht erreichbar wäre. Weiters kann eine Unterscheidung der Destination anhand von Portnummern gewährleistet werden, welche dann zusätzlich einer Übersetzung unterzogen werden müsste. Grafisch kann das wie folgt dargestellt werden:

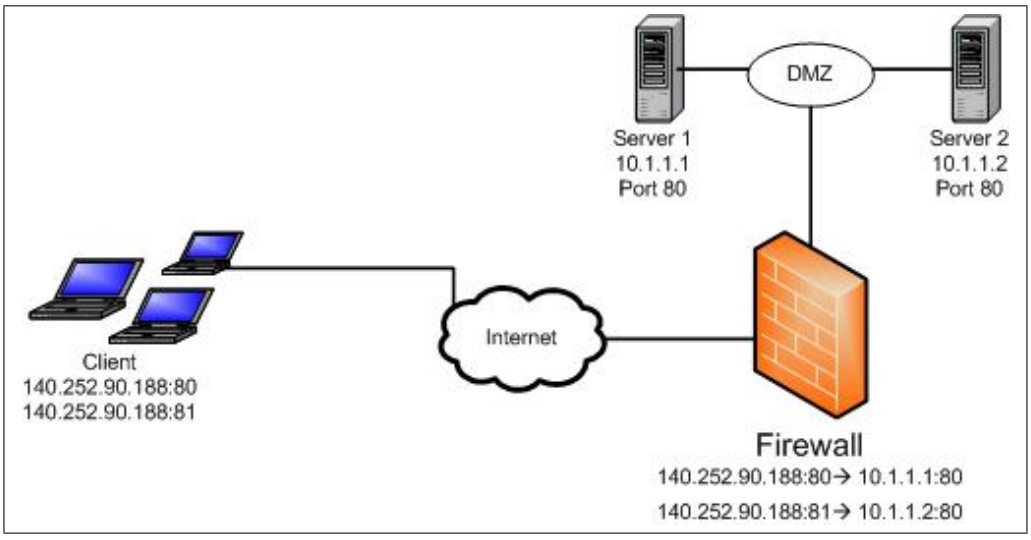

Abbildung 2.1.: NAT für IP-Adressen [vgl. FG05, S. 61]

Zu beachten gilt, dass nicht alle Anwendungen beziehungsweise Protokolle in Verbindung mit NAT eingesetzt werden können. Hier seien AH (Authentication Header), IKE (Internet Key Exchange) und IPSec ohne UDP Encapsulation genannt.

- **Verschlüsselung** ist hinsichtlich der sicheren Übertragung von Daten über ein unsicheres Netzwerk (Internet) ausschlaggebend für die verteilte Vernetzung von Standorten beziehungsweise Firmen. In diesem Zusammenhang können VPNs (Virtual Private Networks) zur Datenverschlüsselung eingesetzt werden. Firewalls mit ihrer zentralen Stellung eignen sich daher hervorragend als VPN-Endpunkte zwischen welchen diese „sicheren“ Tunnel durch das Internet aufgebaut werden. Selbst wenn die Firewall keine eigene VPN-Funktionalität bietet, deren Nutzung jedoch angestrebt wird, muss zumindest eine Weiterleitung dieser Anfragen ins interne Netz (zur VPN-Gegenstelle) ermöglicht werden. Der Vorteil von Firewalls, die VPN Funktionalität unterstützen ist jener, dass ungültige Anfragen sofort abgewiesen werden können und somit nicht einmal ins interne Netz vordringen können.

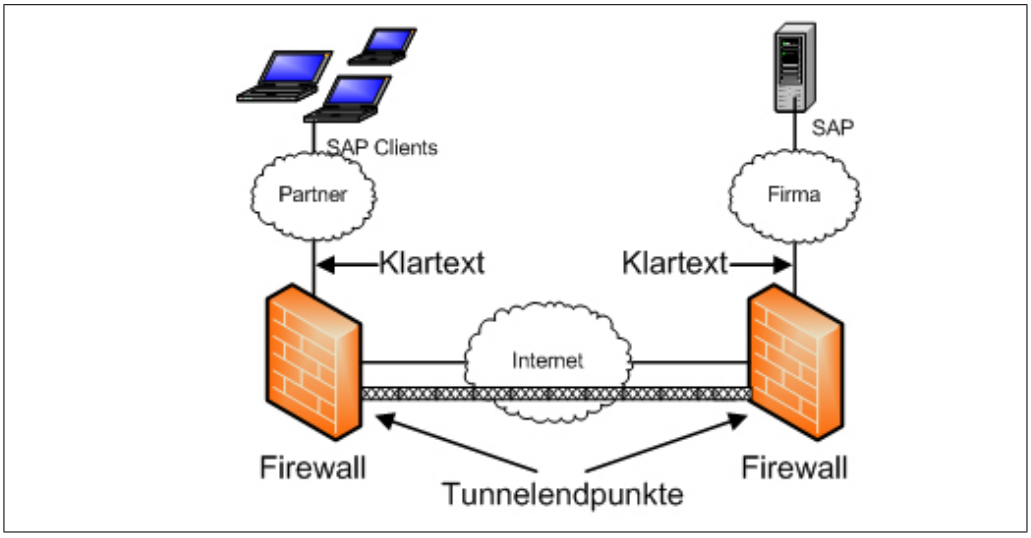

Abbildung 2.2.: VPN Tunnel [vgl. FG05, S. 63]

- **Content Security** (oder kurz CS) bezeichnet Maßnahmen um zu verhindern, dass schädlicher Content (daher Inhalt) in das interne Netzwerk gelangt und wird

vorwiegend für Internetanwendungen eingesetzt (zum Beispiel HTTP, FTP, SMTP). CS erfordert die Zerlegung der einzelnen Pakete in seine Protokollteile und die semantische Analyse von diesen. Weiters ist auch möglich, und bei manchen Produkten inkludiert, die Inhalte auf Viren zu durchsuchen und gegebenenfalls zu blockieren. CS-Lösungen bestehen technisch gesehen aus einem Proxy je zu kontrollierendem Protokoll. In Verbindung mit dem SMTP-Protokoll kann auch das Synonym „Mail-Firewall“ verwendet werden. Viele Firewall-Produkte kommen ohne CS-Funktionalität auf den Markt, die meisten jedoch bieten eine Schnittstelle zu Produkten von spezialisierten Anbietern.

- **Intrusion Prevention und Detection** sind neuere Ansätze zum Erkennen und Umgang mit Angriffen. Beide basieren grundlegend auf der Headerinformation der Datenpakete, können aber aufgrund des Verhaltens (beziehungsweise der Vorkommnisse) Angriffe erkennen. Um fortgeschrittene Angriffe zu erkennen, ist es keinesfalls ausreichend einzelne Pakete zu analysieren, es müssen hingegen die Fragmente eines Angriffes „virtuell“ reassembliert werden. Erst die Betrachtung mehrerer Fragmente (Paketteile) bilden die Unterscheidung zur herkömmlichen Firewall (zum Beispiel Paketfilter).
  Eine Weiterentwicklung stellt die sogenannte „Deep Inspection“ dar, welche zusätzlich zu den Header-Informationen auch die Protokollstruktur und somit -einhaltung überprüft. Es wird somit verhindert, dass protokollfremde Anwendungen eine Art „Einkapselung“ über diese Protokolle verwenden. In diesem Zusammenhang sind zum Beispiel Internet Messenger genannt, die über HTTP eine Verbindung zum Server aufbauen.
  Die Unterscheidung in IPS (Intrusion Prevention Systems) und IDS (Intrusion Detection Systems) liegt in der Behandlung der Angriffe. IPS erkennen und wehren aktiv Angriffe ab, während IDS diese nur wahrnehmen und aufzeichnen (loggen). Wieder ermöglicht die Zentralität des Firewall-Systems eine Erweiterung zum IPS beziehungsweise IDS.

Bei der Auswahl der Features, welche ein Firewall-System bieten soll, gilt zu unterscheiden, welche Aufgaben unbedingt notwendig sind und welche als selbstständige Lösung sinnvoller (da höhere Performance) zu realisieren sind [Spe06]. So ist zum Beispiel das Logging und Alerting aus keiner gängigen Lösung wegzudenken, wohingegen die Deep-Inspection oder CS-Funtkionalität ausgelagert werden könnten. Es gilt darüber hinaus abzuwägen, ob eine hinreichende Performance möglich ist und ob die zusätzlichen Features sicherheitstechnisch (bekannte Bugs und ähnliches) unbedenklich sind.

2.3. Abgrenzung der Aufgaben von Firewalls

Die Fülle an möglichen Aufgaben die eine Firewall wahrnehmen kann verleitet nur all zu oft zu dem Glauben, dass diese allmächtig sei. In der Tat ist sie es nicht, sondern kann als einzelnes Modul (von mehreren) zur Absicherung des Systems eingesetzt werden.

Der Einsatz einer Firewall ist generell gesprochen begrenzt, um ein Netzwerk nach außen hin abzusichern. Aktuellen Studien zu Folge werden jedoch ca. 80% aller Delikte von Insidern begangen [Les06]. In solch einer Situation ist die Firewall ungeeignet um diesen Problemen zu begegnen.

Weiters ist eine Firewall verständlicherweise nutzlos, wenn diese übergangen wird und sie somit ihre Schutzfunktion verliert. Dies ist zum Beispiel möglich, wenn ein Mitarbeiter sein eigenes Modem (zur Herstellung einer Internetverbindung) einschleust, oder einen verschlüsselten Tunnel zu einem unsicheren Netzwerkknoten aufbaut.

Ein weiterer Problempunkt besteht in der Verwendung von mobilen Endgeräten, wie zum Beispiel eines Laptops. Die Verwendung dieser Gerätschaften in einem unsicheren Umfeld (also außerhalb der von uns errichteten DMZ) kann zur Verseuchung (zum Beispiel durch Viren oder Trojaner) führen. Diese kann durch die Verwendung im eigenen Netzwerk auf das gesamte Netzwerk überschlagen und erhebliche Probleme mit sich bringen.

Bei Diensten, die öffentlich (über das Internet) zur Verfügung gestellt werden sollen, wie zum Beispiel der Mail-Server oder aber auch ein Web-Server, ist eine Abschottung durch die Firewall nicht zielführend, da diese ja im Generellen den Zugang zu unterbinden versucht. In Verbindung zu diesen öffentlich zugänglichen Services sei genannt, dass auch die programmiertechnische Sicherheit der jeweilig eingesetzten Komponenten nicht außer Acht gelassen werden kann. So kann zum Beispiel eine fehlerhafte Implementierung zu einen bedrohlichen Sicherheitsrisiko werden, wenn dem Angreifer die Möglichkeit erwächst, das System „unbemerkt“ zu kapern. Bei der Verwendung solcher öffentlichen Dienste wird in der Praxis eine eigene DMZ für diese Systeme eingerichtet, welche wiederum von dem eigentlichen internen Netzwerk (durch eine Firewall) abgeschirmt wird.

Weiters sind Firewalls kein adäquates Mittel um Angriffe, welche sich gegen die Kommunikationsfähigkeit richten, zu bekämpfen. Als Beispiel seien diverse DoS-Angriffe genannt, bei denen die rigorose Anzahl von sinnlosen Datenpakten eine Art Verstopfung

der Kommunikationsleitung verursachen sollen. Dies führt dann dazu, dass die eigenen Anfragen (an externe Dienste) einfach in der Menge untergehen und nicht mehr abgehandelt werden können. Die Firewall leistet in diesem Fall „nur“ den wesentlichen Beitrag, dass diese manipulierten Pakete nicht den Weg ins interne Netz finden.

Um die Thematik dieser Arbeit abzugrenzen sei an dieser Stelle erwähnt, dass diese Arbeit und die darin enthaltenen Erläuterungen zum Thema der Firewall grundsätzlich auf Fireall-Basisfunktionen abzielen. Zudem werden Zusatzfunktionen erläutert, welche durch die standardmäßige Implementierung in diversen Software-Lösungen einen „quasi“ Basisfunktionscharakter aufweisen.

2.4. Einsatzbereiche

Der Einsatz sowie die -gebiete von Firewall(-systemen) sind mannigfaltig und werden grob in drei Kategorien unterteilt. Als Unterscheidungskriterium dienen die Anzahl der zu schützenden Clients, als auch die Sicherheitsansprüche, welche berücksichtigt werden müssen. Darüber hinaus gibt es eine Vielzahl von Merkmalen beziehungsweise Eigenschaften, welche bei der Installation beziehungsweise beim Betrieb einer Firewall berücksichtigt und an die jeweiligen Bedingungen angepasst werden müssen. Hier seien auszugsweise die Anzahl der Benutzer, das Schutzbedürfnis, als auch die einzusetzende Hardware genannt.

Die drei (nicht bindenden) Kategorien, in steigender Komplexität sind folgende [Les06]:

- **Privathaushalt beziehungsweise Heim-PC** In dieser Umgebung gibt es eine geringe Anzahl von Nutzern und Endgeräten, die mit dem Internet verbunden sind.

  In dieser Kategorie finden sich vorwiegend kleinere Firewall-Produkte, wie Disketten-Linux basierte Firewalls, als auch Personal Firewalls. Die Vorteile der minimalen Linux Version stellen die einfache (wenngleich auch beschränkte) Konfiguration, als auch der ressourcenschonende Betrieb dar [Les06]. Darüber hinaus ist die Logik unabhängig von den einzelnen Netzwerk-Clients und kann zentral verwaltet werden. Ein weiterer Vorteil dieser Lösung ist die Schaffung der zuvor schon beschriebenen DMZ.

Ist es nicht notwendig eine DMZ zu schaffen, zum Beispiel weil nur ein Rechner Zugang zum Internet bietet, so drängt sich die Verwendung der Personal Firewall auf. Diese Art der Firewall-Systeme etablierte sich während der Zeit als vorwiegend Firewall-Systeme für Firmen den Markt beherrschten. Diese Produkte hatten neben dem erheblichen Kaufpreis zusätzlich noch den Nachteil der langwierigen Installation und Konfiguration. Durch die stetig steigende Anzahl der privaten Nutzer und deren Bedürfnis für Sicherheit entstanden die sogenannten Personal Firewalls, welche vorwiegend als Applikation im Hintergrund des Betriebssystems ihre Dienste verrichten [Jan07]. Zu dieser Gattung zählen die prominenten Vertreter wie ZoneAlarm, Norton Internet Security (sowie Norton 360) als auch McAfee Internet Security.

Festzuhalten gilt, dass diese Systeme nicht komplementär anzusehen sind, sonder auch in Kombination mit dem jeweils anderen auftreten beziehungsweise angewandt werden können. Vor allem die Einfachheit der Installation einer Personal Firewall spricht für die Anwendung dieser, um eine Erhöhung der Sicherheit zu erzielen.

- **(Studenten-)Wohnheim** In dieser Umgebung gibt es ein Vielzahl von Nutzern und Endgeräten, die mit dem Internet verbunden sind. Weiters beschreibend für diese Situation ist, die stark limitierte monetäre Lage und die daraus resultierende fehlende Investitionsfähigkeit.

  Die Vielzahl der Nutzer bedingt eine Vergabe von privaten Adressen, da bei Verwendung von öffentlichen Adressen erhebliche Kosten entstehen würden. Die Hauptaufgaben einer Firewall in dieser Umgebung stellen der Verbindungsaufbau nach außen, das Verbergen der Rechnerinfrastruktur (intern) sowie das Blocken von unberechtigten Zugriffen von außen dar.

  Die Finanzmittelknappheit verbietet den Einsatz von kommerziellen Lösungen, weshalb die FW-Funktionalität durch einen „ausreichend“ performanten Rechner auf Basis einer Standarddistribution (zum Beispiel SuSe Linux) bereitgestellt wird. Die Vielzahl der Benutzer schließt aufgrund der nicht realisierbaren Policies die Verwendung einer Disketten-Distribution aus.

  Bedingt durch die Vielzahl der Benutzer ist die Verwendung eines cachenden Proxys durchaus als sinnvoll zu erachten. Durch die Standardinstallation ist diese Funktionalität ohne größere Schwierigkeiten schnell nachzurüsten.

Einen bekannten Vertreter dieser Gattung stellt der PF (Packet Filter) unter OpenBSD, mit seiner NAT Funktionalität dar, welcher später in dieser Arbeit noch erläutert beziehungsweise beschrieben wird.

- **Firmennetzwerk** In dieser Umgebung gibt es oft eine definierbare Anzahl von Nutzern und Endgeräten, die mit dem Internet verbunden sind. Die Infrastruktur beherbergt neben den Benutzer-Clients auch eine Sever-Landschaft zur Bereitstellung von Datei- und Druckerdiensten. Auch soll zur Repräsentation der firmeneigenen Produkte ein Web-Server betrieben werden.

  Zusätzlich zu den bereits bei dem Wohnheim genannten Features (Zugang regeln, Cache, NAT) muss in dieser Situation das Zulassen von wohl definierten (bestimmte Ports) Anfragen auf den eigenen Web-Server berücksichtig werden. Da dies zu Problemen führen könnte, wenn sich dieser im selben Netz wie die Benutzer-Clients, Datei- und Druckserver befände (da der Zugriff von außen ungehindert und ungefiltert nach innen dringen würde), ist es wünschenswert eine zweite (von der internen abgesicherten) Zone einzurichten. Dies bewirkt ein Abschotten des internen Netzes vom Internet (DMZ 1) und ermöglicht gleichsam die Zugriffe auf den Web-Server von außen (DMZ 2).

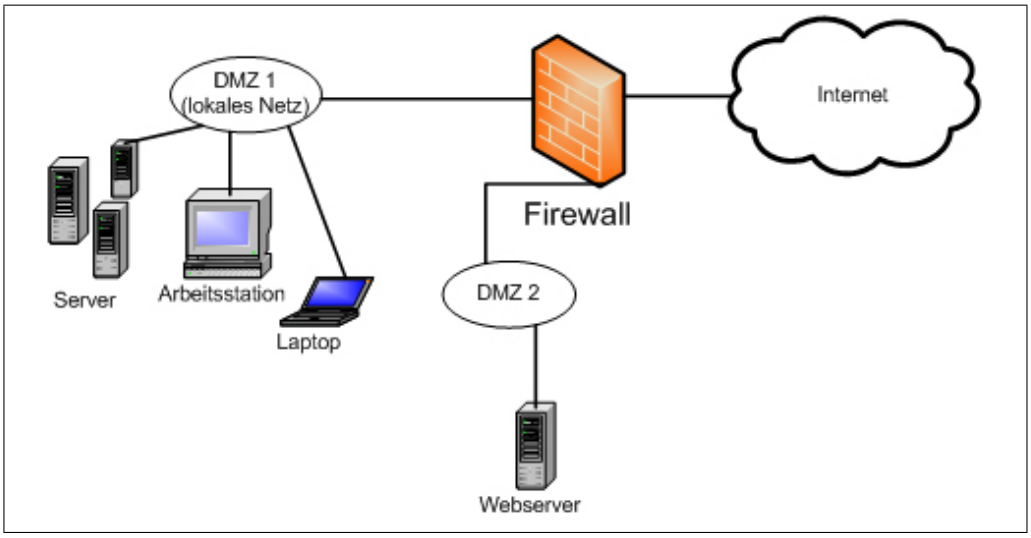

Abbildung 2.3.: Firewall im Firmennetzwerk [vgl. Les06, S. 13]

Da die finanzielle Situation es ermöglicht, kann moderne Hardware angeschafft werden, sowie kommerzielle Produkte zum Einsatz kommen.

3. Grundlagen der Netzwerktechnik

Das nachfolgende Kapitel soll dem Leser einen Überblick über die notwendigen Techniken beziehungsweise Technologien zur Datenvermittlung im Internet sowie der damit verbundenen Protokolle und Dienste geben. Dieses Kapitel dient somit der Erläuterung der Grundlagen, wie Firewalls Daten kontrollieren und gegebenenfalls verwerfen beziehungsweise abändern. Der Schwerpunkt wird dabei auf der Darstellung der jeweiligen Protokolle, sowie gegebenenfalls deren Schwächen liegen. Für eine detailierte (und technische) Erläuterung der einzelnen Protokolle und Dienste, sowie auf deren geschichtliche Entwicklung sei auf die einschlägige (Fach-)Literatur verwiesen[1].

3.1. TCP / IP Modell

Um den Datenvermittlungsvorgang im Internet erläutern zu können wird häufig auf die diversen Schichtenmodelle zurückgegriffen. Die zwei wohl bekanntesten Modelle sind das ISO/OSI (ISO = International Standards Organization / OSI = Open Systems Interconnection) sowie das tatsächlich implementierte TCP/IP (TCP = Transmission Control Protocol / IP= Internet Protocol) Modell. In diesem Zusammenhang muss erläutert werden, dass das ISO/OSI Modell aufgrund der relativ späten Entwicklung (mit seinen idealtypischen Eigenschaften) durch das bereits verfügbare und eingesetzte TCP/IP Protokoll (mit den wohlbekannten Problemen) obsolet beziehungsweise ersetzt wurde. Die Entstehung des TCP/IP Modells folgte somit, in Hinblick auf die Entwicklung der Funktionalität, den Praxisanforderungen sowie den verfügbaren Technologien. Zusammenfassend lässt sich somit sagen, dass es sich beim ISO/OSI Modell um ein theoretisches, in der Praxis nicht in größerem Umfeld implementiertes Modell handelt, welches uns als Referenzmodell dient. Um die dennoch vorhandenen Zusammenhänge dieser beiden Modelle darzustellen, soll die nachfolgende Grafik als beschreibendes Instrument dienen.

[1] zum Beispiel: [Tan02] – Tanenbaum, Andrew S.: *Computer Networks.* Prentice Hall International, 4. Auflage, August 2002.

Beide Modelle können wie folgt gegenübergestellt werden (Abbildung enthält einige Beispielprotokolle):

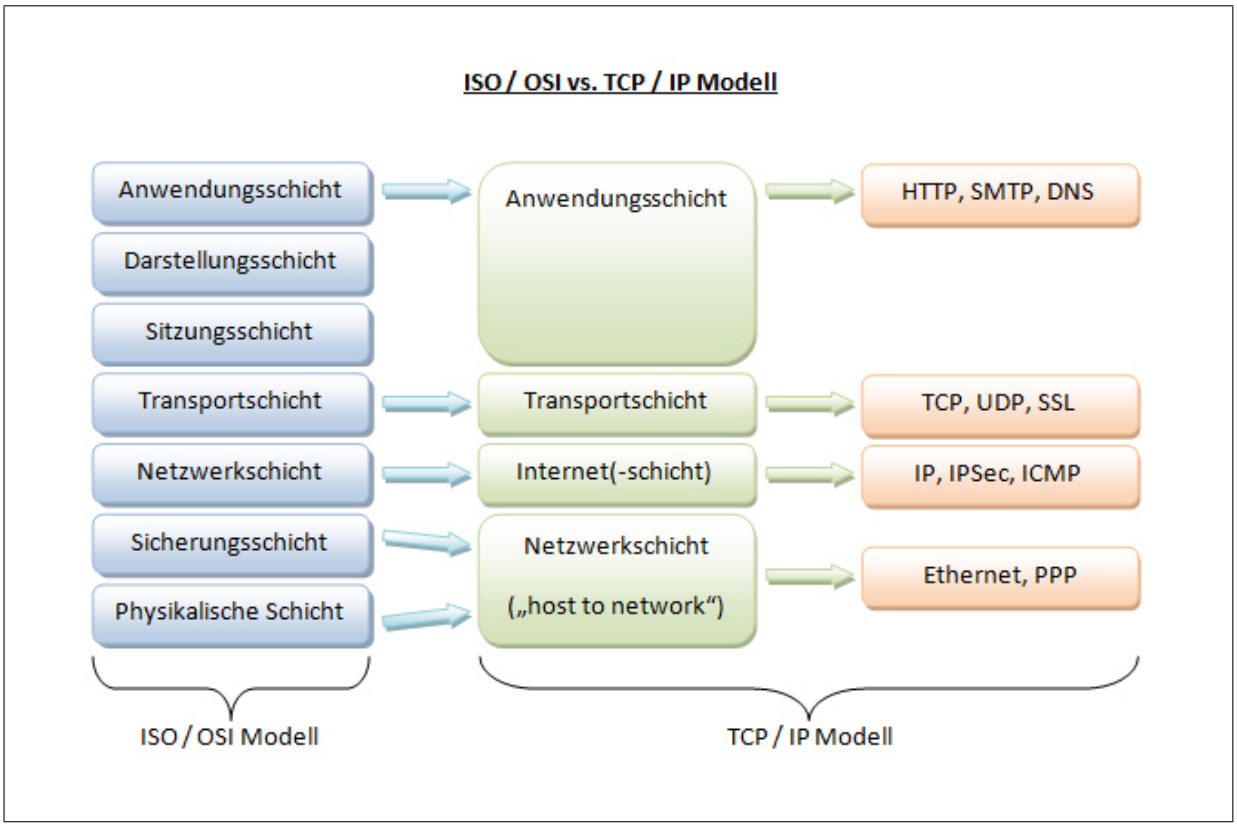

Abbildung 3.1.: ISO/OSI vs. TCP/IP Modell [vgl. WP07, S. 96]

Aufgrund der Verbreitung sowie der Bedeutung in der Praxis, als auch für diese Arbeit, wird nachfolgend das TCP/IP Modell erläutert, wobei das ISO/OSI Modell ausgeklammert wird. Die Beschreibung dieser wird auf einer abstrakten Ebene stattfinden, da die einzelnen Protokolle in den nachfolgenden Unterpunkten im Detail beleuchtet werden. Für interessierte Leser wird wieder auf die einschlägige Literatur verwiesen[2].

Die einzelnen Schichten sowie die damit verbundenen Aufgaben sind [Tan02]:

[2]zum Beispiel: [Tan02] – Tanenbaum, Andrew S.: *Computer Networks*. Prentice Hall International, 4. Auflage, August 2002.

- **Netzwerkschicht** Die Netzwerkschicht wird generell als freistehend und minimalistisch beschrieben betrachtet. Ihre Aufgabe liegt darin, es einem Endgerät beziehungsweise Nutzer Zugang zum Netzwerk (insb. Internet) zu ermöglichen, wobei die einzusetzenden Technologien nicht näher spezifiziert werden. Als einzige Restriktion ist lediglich die Kompatibilität zur Verwendung von IP - Paketen beziehungsweise deren Einkapselung (Encapsulation) vorgeschrieben. Die zugrunde liegende Technologie, sei es eine UMTS Verbindung oder ein Glasfaseranschluss, wird somit ausgeklammert, wobei diese die Hauptaufgabe der Verbindungsmöglichkeit inne hat.

  Als eine der wichtigsten Aufgaben der Netzwerkschicht kann die Übersetzung der aus der Internetschicht übergebenen IP - Adresse zu der physikalischen Adresse (Hardwareadresse des Netzwerkadapters) angesehen werden. Diese Übersetzung geschieht mittels ARP, welches teilweise auf hinterlegte Tabellen zugreift.

- **Internetschicht** Die Aufgabe der Internetschicht besteht darin, ein IP - Paket (beziehungsweise IP - Datagramm) eindeutig mit einer Quell- und Zieladresse (beziehungsweise Rechner) zu versehen und die Vermittlung (Routing) dieser zu steuern.
  Die Quell- sowie Zieladressen müssen zum erfolgreichen Routing weltweit eindeutig sein, womit auch das Problem von IPv4 definiert ist. Der Aufbau entspricht einer 32bit langen punktseparierten Zeichenkette, welche aus vier Teilen besteht (zum Beispiel 127.0.0.1).
  Bei der derzeitig vorherrschenden Version des Protokolls (Version 4) ist die Anzahl der Adressen allmählich erschöpft, weshalb eine Folgeversion (Version 6) entwickelt wurde. Hier wurde der Adressbereich auf eine 128bit lange Zeichenkette erweitert, welche aus Gründen der Leserlichkeit, in hexadezimaler Schreibweise einzutragen ist (zum Beispiel 1000:1cba:aaaa:1111:2449:8a1b:-1360:2345).
  Das IPv6 besitzt zusätzlich zu dem erweiterten Adressbereich noch einige Entwicklungen, welche der bemängelten Sicherheitslücken der Vorversion Folge tragen sollen. So sei zum Beispiel das IPSec - Protokoll genannt, das erweiterte Sicherheitseigenschaften bietet, welche jedoch auf Kosten der Verständlichkeit realisiert wurden. So wird derzeit noch angezweifelt, ob das IPSec - Protokoll in all seinen Facetten überhaupt von einer einzigen Person komplett verstanden werden kann. Dies führt gegebenenfalls auch zu schwachen Implementierungen des Protokolls, womit Sicherheitslücken entstehen können.

Die ausgefeilten Techniken der Adressübersetzung (und Subnetzbildung) der gegenwärtig verbreiteten Version, sowie die Rückwärtskompatibilität der wichtigsten implementierten Sicherheitsmechanismen, der zukünftigen Version, lassen Zweifel über die in näherer Zukunft nicht zu erwartende Umsetzung des neuen Protokolls erwachsen, da diese mit erheblichen Kosten bei den Telekommunikationsanbietern verbunden wäre. Die gegenständlich eingesetzte Version des Protokolls determiniert zudem keine Mechanismen zur Sicherstellung einer Mindestqualität der Übertragungsleistung, welche deshalb zumeist im Backbone-Netz der Kommunikationsdienstleister realisiert wird.

Die Internetschicht ist zudem für die Übermittlung etwaiger Statusmeldungen des Übertragungs- und Vermittlungsprozesses verantwortlich, welche durch das ICM - Protokoll realisiert ist. Als Beispiel sei hier die Rückmeldung „Destination unreachable“ genannt, welche signalisiert, dass ein Paket nicht ausgeliefert werden konnte.

- **Transportschicht** Die Transportschicht gewährleistet Verbindung sowie die Verwaltung von Kommunikationssitzungen als auch die Sicherstellung einer verlustfreien Übertragung. In der Transportschicht werden das verbindungsorientierte Transmission Control Protocol (kurz TCP), als auch das verbindungslose User Datagram Protocol (kurz UDP) definiert und spezifiziert.

  TCP ermöglicht eine Fragmentierung des zu übertragenden Bit - Stroms, wobei ein Fehlererkennung sowie -korrektur der Verbindung gewährleistet wird. Die TCP - Fragmente werden an die Internetschicht zur Einkapselung weitergereicht, wobei zudem ein flussgesteuerter Kontrollmechanismus das unabsichtliche „Flooding“ beziehungsweise Überlasten der Gegenstelle zu vermeiden sucht. Dies ist zum einen durch die nicht spezifizierte „Quality of Service“ Eigenschaften und zum anderen der unterschiedlichen Übertragungsleistungen der zugrunde liegenden Medien notwendig und sinnvoll.

  Das verbindungslose UDP steht für Anwendung zur Verfügung welche eine Fragmentierung beziehungsweise Flusskontrolle selbst realisieren beziehungsweise nicht benötigen. Die dadurch ermöglichten Performancegewinne eignen sich besonders für auf Geschwindigkeit optimierte Anwendungsfälle, bei denen eine Sicherheit der Übertragung weniger wichtig als die erzielte Übertragungsleistung ist. Zu den klassischen Anwendungsfällen zählen etwa die

Video- und Sprachübertragung, aber auch die Übertragung, sich nur langsam verändernder Messwerte, kann mittels UDP realisiert werden.

- **Anwendungsschicht** Die Anwendungsschicht beinhaltet ein wahres Sammelsurium an sogenannten "höheren" Protokollen, welche auf TCO und IP zur Übermittlung aufbauen. Hier sei etwa das HTTP (Hypertext Transfer Protocol) und SMTP (Simple Mail Transfer Protocol) genannt. Aber auch der in Verbindung mit dieser Arbeit besonders interessante Dienst der Namensauflösung (DNS – Domain Name Service) ist in dieser Schicht angesiedelt. Dieser Service ist für die Aufschlüsselung und Umwandlung von Ressourcennamen, wie zum Beispiel `http://www.example.org/`, in die entsprechenden IP - Adressen verantwortlich.

Aus den vorangehend beschriebenen Schichten sowie der damit verbundenen Aufgaben sollte ersichtlich sein, dass die Schichten in einer Art Interaktion zu einander stehen. Hierbei wird jeweils von einer höheren Schicht auf die jeweils nachfolgende, niedriger zugegriffen und eine Einkapselung der übergebenen Daten durchgeführt. Ein Verständnis dieser Datenkapselung ist unerlässlich, um den Prozess der Filterung beziehungsweise Inspektion von Datagrammen durch die Firewall prinzipiell verstehen zu können. Diese Kapselung soll durch nachfolgende Grafik (schemenhaft) erneut verständlich gemacht werden:

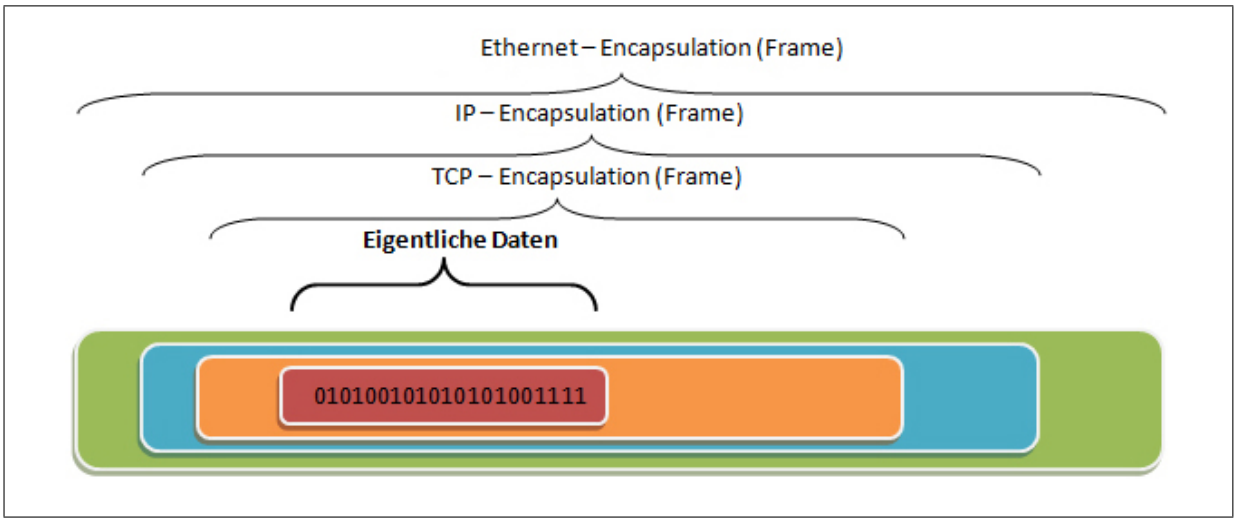

Abbildung 3.2.: Schematische Daten und Protokollkapselung

Aus der Abbildung wird auch ersichtlich, dass eine Firewall in der Lage sein muss, den Aufbau sowie die Einkapselung der einzelnen Protokolle zu kennen, um diese durchsuchen zu können.

Weiters ist ersichtlich, dass die ursprünglichen Daten nur einen Bruchteil der gesamten Daten beziehungsweise des Frames darstellen. Zudem ist dieser verfügbare Bereich (für die eigentlichen Daten) stark limitiert wodurch bei größeren Datenmengen mehrere Frames erstellt werden müssen. Diese, quasi Fragmentierung ihrerseits, hat direkten Einfluss auf die Arbeit einer Firewall, da die integrierte „Intelligenz“ (zum Beispiel Deep Inspection) darüber entscheidet, ob die Frames nur einzeln oder im Verbund inspiziert werden.

3.2. Internet Protocol (IP)

Das *Internet Protocol* (kurz IP) findet zum Datentransport, von aus höheren Schichten übermittelten Daten, Anwendung und stellt daher eine Art Grundbaustein der Kommunikation dar.

Das IP stellt ein verbindungsloses Protokoll dar, welches keinerlei eigene Korrekturtechniken besitzt, was in weiterer Folge bedeutet, dass sich Protokolle höherer Schichten um diese Funktionen und somit der korrekten Auslieferung der Daten kümmern müssen.

Das IP-Datagram besteht generell gesehen aus einem Header und den sogenannten Payload (beziehungsweise Daten). Das Datagramm enthält dabei im Header die Absender- sowie Zieladresse, welche für das Finden der Routen durch das Internet notwendig sind. Da jedoch das IP verbindungslos ist, ist nicht gewährleistet, dass alle Pakete den selben Weg zum Ziel vermittelt bekommen. So kann es auch vorkommen, dass ein zu einem späteren Zeitpunkt versendetes Paket zuerst ankommt. Das Internet Protokoll übernimmt somit die „Vermittlung“ der (Fragment-)Pakete vom Absender zum Ziel.

Ein wesentliches Merkmal des Internet Protocol liegt daher in der Möglichkeit der Fragmentierung von großen Datagrammen, um den Maximalgrenzen (zum Beispiel Ethernet 1500 Byte) Rechnung zu tragen. Dabei wird das Datagramm in mehrere IP-Datagramme zerlegt und mittels Angabe des ID-Feldes in Verbindung mit dem Fragment-Offset beim Empfänger in der richtigen Reihenfolge wieder zusammengesetzt. Somit ist auch klar, dass ein Reassembling nicht an beliebigen Punkten (im Internet) möglich ist, da ja nicht

alle Fragment-Teile dieselben Knotenpunkte passieren müssen, weshalb das Zusammensetzten vorwiegend am Ende (Ziel-Host) stattfindet [Tan02].

Die folgende Abbildung soll einen Überblick über die Zusammensetzung des IP-Headers geben und die einzelnen, darin enthaltenen Elemente aufzeigen:

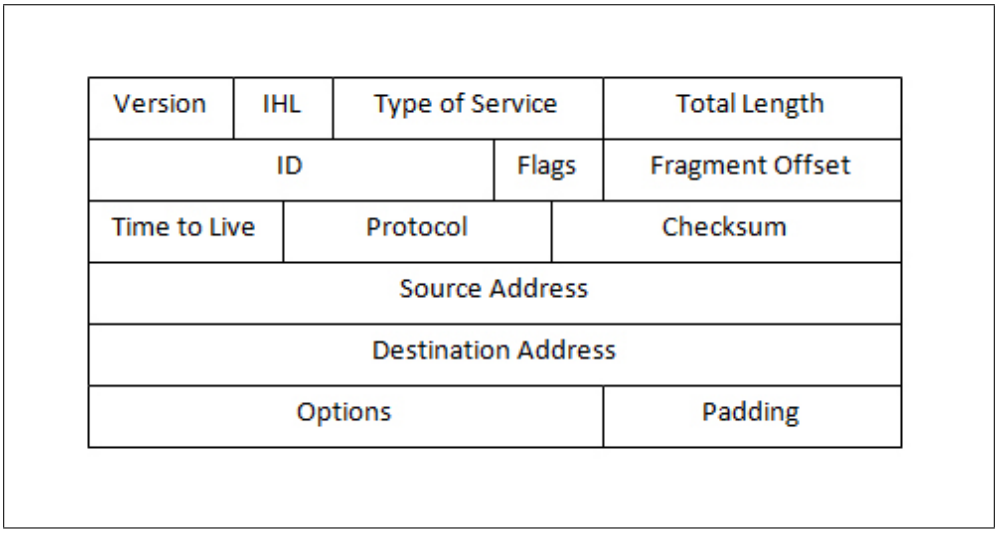

Abbildung 3.3.: Detailansicht des IP-Headers [vgl. WP07, S. 101]

Auf eine detailierte Erläuterung aller Felder wird aus Platzgründen verzichtet, da diese den Rahmen dieser Arbeit sprengen würde. Ebenso soll eine Unterscheidung der Protokollversionen an dieser Stelle ausgeklammert werden. Auch hier sei auf die einschlägige (Fach-)Literatur verwiesen[3].

IP-Sicherheitsaspekte In Hinsicht auf die sicherheitstechnischen Aspekte des Protokolls sei vermerkt, dass dieses vorwiegend zur Realisierung von Spoofing-Attacken Verwendung findet beziehungsweise ausgenutzt wird. Dabei handelt es sich um das Versenden von Pakten mit gefälschter Absender-Adresse, wie es zum Beispiel für Reflektor-Angriffe (Flooding) Anwendung findet.

Weitere Schwachstellen des Internet Protocols bestehen in der Anfälligkeit für Angriffe,

[3] zum Beispiel: [Tan02] – Tanenbaum, Andrew S.: *Computer Networks*. Prentice Hall International, 4. Auflage, August 2002.

welche auf der Fragmentierung, der Angabe von beliebigen Protokollen (Protokollscan) aber auch auf anderen Optionen des Protokolls basieren [WP07].

IP-Firewall-Aspekte In Hinblick auf die Verwendung von IP-Paketen in Verbindung mit Firewalls sei festgehalten, dass zumeist eine Filterung basierend auf den Adressen (Paketfilter) durchgeführt wird.

Zusätzlich sollte das gewählte Firewall-Produkt über die technische Möglichkeit verfügen, ein Reassembling von fragmentierten Paketen durchzuführen. Diese Fragmentierung ermöglicht es mit einfachsten Mitteln, die Funktionsweise von einfachen Paketfiltern zu umgehen beziehungsweise führt im schlimmsten Fall zu einem Ausfall der Firewall-Komponente, wenn diese ressourcenbegrenzt beziehungsweise technisch nicht ausgereift ist.

Um die Wahrscheinlichkeit eines erfolgreichen DoS-Angriffes, welcher auf IP-Spoofing basiert zu minimieren, besteht die Möglichkeit, nur jene Anfragen gewähren zu lassen, welche aus dem eigenen (beziehungsweise privaten) Netzwerk kommen. Dies wird dadurch ermöglicht, dass bestimmte IP-Adress-Bereiche für private Netzwerke reserviert sind (zum Beispiel 192.168.0.0 - 192.168.255.255).

Zusätzlich soll an dieser Stelle festgehalten werden, dass eine ausschließliche Betrachtung des IP-Datenverkehrs, durch die Firewall, nicht als ausreichend angesehen wird, wenngleich sie dennoch eine Vielzahl unterschiedlicher Angriffsszenarien im Voraus unterbinden kann [Rue07].

3.3. Transmission Control Protocol (TCP)

Das *Transmission Control Protocol* (kurz TCP) wird verwendet, um Pakete zwischen Anwendungen, basierend auf dem IP, zu transferieren und gleichsam sicherzustellen, dass die Pakete in richtiger Reihenfolge beim Ziel eintreffen. Darüber hinaus stellt TCP sicher, dass keine Pakete ausgelassen, doppelt gesendet oder verloren werden.

Es muss eine Adressierung der Anwendungen realisiert werden, denn auf einem Rechner können eine Vielzahl von Anwendungen (Dienste) das Netzwerk zur Kommunikation

verwendet werden. Für diese Adressierung finden die sogenannten *Portnummern* Verwendung, welche durch die Anwendung an das Betriebssystem übergeben und somit reserviert werden. Durch die Verwendung des Portnummern-Konzepts ist eine Möglichkeit zur Verbindung zweier Anwendungen sichergestellt. Wobei hier von einer eindeutig beschriebenen Verbindung gesprochen wird, wenn zusätzlich zu den Portadressen (des Absenders und Ziels) auch deren Internet Adressen (IP-Adressen) angegeben sind. Eine solch eindeutig beschriebene Adresse (nicht Verbindung) wäre etwa 127.0.0.1:80, wobei die Angabe der Portnummer durch einen Doppelpunkt von der IP-Adresse abgegrenzt wird. Die Verwendung von Portnummern wird erst durch die fixe Vergabe solcher sinnvoll, wobei die Verwendung dieser Zuordnung nicht als obligatorisch gilt und somit zu Problemen führen kann.

Zu Verdeutlichung soll folgende Abbildung dienen:

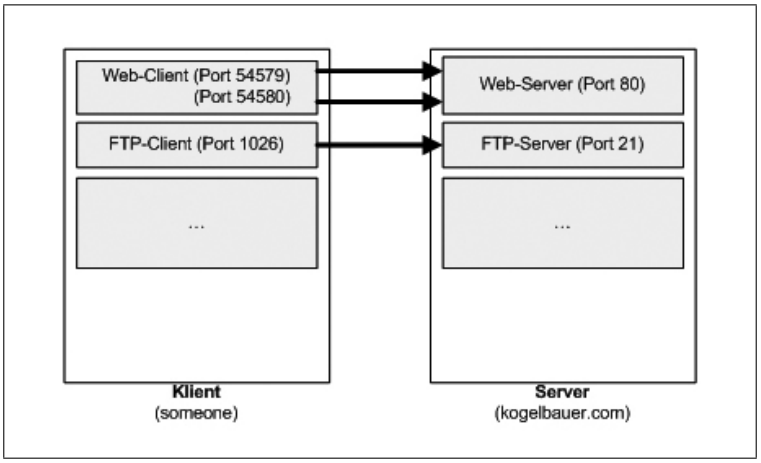

Abbildung 3.4.: Eindeutig beschriebene Verbindung

Neben dieser genau determinierten Verbindung zwischen den Applikationen liegt die zweite Aufgabe von TCP in der Bereitstellung einer gewissen Verlässlichkeit in Hinblick auf die Übertragung. Dies wird durch die Herstellung einer Verbindung zwischen den beiden Anwendungen und die Verwendung von Folgenummern gewährleistet. Trifft zum Beispiel innerhalb einer Verbindung ein Paket mit einer niedrigeren als der aktuellen Nummer ein, kann dieses verworfen werden, da es schon empfangen wurde. Kommt jedoch ein Paket mit einer zu hohen Nummerierung (im Vergleich zur korrekten Folgenummer), so kann auf den Verlust eines Zwischenpakets geschlossen werden und dieses in weiterer Folge erneut beantragt werden [Les06].

TCP-Sicherheitsaspekte Es gibt eine Vielzahl von Attacken, welche sich gegen TCP richten [WP07]. Das vorwiegende Problem des Protokolls ist in der Berechenbarkeit beziehungsweise leichten Erkennbarkeit (zum Beispiel mittels Sniffing) der Folgenummern zu sehen.

Als *TCP-Spoofing* bezeichnet man das Fälschen von TCP-Paketen, wobei die korrekte Folgenummer bekannt sein muss. Darauf aufbauend kann ein Hijacking-Angriff erfolgen.

Als *Hijacking-Angriff* wird die mutwillige Übernahme einer bestehenden TCP-Verbindung bezeichnet. Der Sinn eines solchen Angriffs ist die unbemerkte Injektion beziehungsweise das Auslesen von Daten. Des weiteren kann TCP-Hijacking zum Umgehen von schwachen Authentifizierungsmaßnahmen eingesetzt werden.

Das sogenannte *TCP-Syn-Flooding* zielt darauf ab, die Kommunikationsfähigkeit des Opfers einzuschränken. Hierbei wird eine Vielzahl an Verbindungsanfragen zum Opfer geleitet, welches unter der Last zusammenbrechen soll, da es diese zu merken und zu beantworten versucht.

Bei der *Reset-Attacke* wird ein manipuliertes Paket (mit richtiger Folgenummer) und gesetztem RST-Flag an das Ziel gesendet. Dies hat eine Verbindungsunterbrechung sowie einen Neuaufbau dieser zur Folge.

TCP-Firewall-Aspekte Da TCP ein statusorientiertes Protokoll darstellt und dementsprechend Sequenznummern und TCP-Flags nutzt um dies zu ermöglichen, gilt es im Vergleich zu UDP als „relativ“ sicher.

Dennoch ist zu beachten, dass eine Firewall, die als Paketfilter ausgeführt ist, zumindest über Stateful Inspection[4] verfügen sollte, da sonst eine unzureichende Analyse basierend auf den TCP-Flags durchzuführen wäre. Da das zufällige Erraten der komplexer werdenden Sequenznummer mittlerweile als relativ schwer anzusehen ist, verliert das TCP-Hijacking zunehmend an Bedeutung.

Eine Verwendung von Paketen mit manipulierten TCP-Flags ist dennoch in zweierlei Hinsicht möglich (ohne Stateful Inspection) [Rue07]:

[4] Technologie zur Unterstützung von Analysetätigkeiten basierend auf dem Verbindungsstatus, um zum Beispiel ein Hijacking zu unterbinden.

1. Provozieren einer Reaktion – Verwendung von ACK-Segmenten zur Provokation von Reaktionen, um zum Beispiel ein ACK-Mapping beziehungsweise einen Window-Scans durchzuführen.
2. Tunneling - Aufbau einer Verbindung zwischen den beiden Endpunkten ohne Syn-Flagge behaftetem Drei-Wege-Handschlag, wobei der Inhalt- beziehungsweise Datenteil zur Synchronisierung verwendet wird.

Die zuvor genannten Missstände können durch den Einsatz von Stateful Inspection unterbunden werden, weswegen dessen Einsatz empfohlen wird.

3.4. User Datagram Protocol (UDP)

Das *User Datagram Protocol* (kurz UDP) befindet sich ebenso wie TCP in der Transportschicht und ermöglicht so Applikationen einen Verbindungsaufbau. Wenngleich auch die Addressierung zu TCP ähnlich ist, so fehlt dem UDP die Folgenummer, als auch die Bestätigungsmeldungen. UDP-Datagramme werden demnach abgesendet, wobei eine Ankunft dieser nicht explizit gewährleistet wird.

Dieser Umstand, dass die Ankunft von Paketen nicht hundertprozentig gewährleistet werden kann, führt oft zu dem Trugschluss, dass es sich bei UDP um ein veraltetes und somit überflüssiges Relikt aus vergangenen Zeiten handelt. Das Einsatzgebiet von UDP jedoch zeigt, dass vor allem Applikationen aus dem Multimedia-Bereich, in Verbindung mit komplexen Algorithmen zu Fehlerkorrektur, auch ohne den Erhalt jedes Paketes annehmbare Ergebnisse erzielen. Eine Verwendung von TCP würde in diesem Zusammenhang (Verlust eines Paketes) eine erneute Nachfrage nach sich ziehen, wodurch im Normalfall Pausen entstehen würden. Die Reduktion des Protokoll-Overheads stellt somit den wesentlichen Grund für die Verwendung dieses Protokolls dar, wenngleich auch die Anwendungsfälle stark limitiert sind[5].

Ein weiterer Anwendungsfall, bei dem es als legitim gilt, UDP zu verwenden, stellt den Sonderfall dar, in dem die Anfrage als auch dessen Antwort in ein Paket passen. Dies ist zum Beispiel bei DNS-Anfragen der Fall und würde unter der Verwendung von TCP wieder zu einem unerwünschten Daten-Overhead führen.

[5] Zum Beispiel im Bereich des Tunneling (VPN), da eine TCP Kapselung in einem TCP Paket zu Problemen führen kann.

Weiters gibt es Fälle, in denen mit keiner Antwort gerechnet werden kann. Hier würde die Verwendung von TCP eine überflüssige Sperre der (Netzwerk-)Ressourcen verursachen. Zu diesen Anwendungsfällen zählt zum Beispiel das Schreiben von Systemprotokollen auf einen Protokollierungsrechner [Les06].

UDP-Sicherheitsaspekte Aufgrund der fehlenden Kontrollmechanismen in UDP ist dieses besonders für Spoofing-Angriffe anfällig, wobei in diesem Zusammenhang meistens DNS-Anfragen gefälscht werden [WP07]. Das Fehlen der Sequenznummer erleichtert zudem wesentlich die mutwillige Übernahme (Hijacking) von Verbindungen. Abhilfe schaffte in diesem Zusammenhang der Einsatz von Software zur Unterbindung solcher Spoofing-Attacken von Seiten der Provider (zum Beispiel T-Online) [WP07].

UDP-Firewall-Aspekte In Verbindung mit der Verwendung von Firewalls (insb. Paketfilter) stellt die Unterstützung von UDP einen Problemfall dar, da eine Unterscheidung von Verbindungsanfragen sowie Antworten auf diese aus dem zeitlichen Kontext hergeleitet werden müssen. Dieses Problem wird anhand des Merkens der Socket-Eigenschaften von ausgehenden Paketen gelöst, da nach einer bestimmten Zeitspanne auf ein Antwortpaket geschlossen werden kann [WP07].

Ein Erkennen von UDP-Portscans wird durch die verhältnismäßig langsame Rückmeldung an den Client zu einer zeitaufwendigen Angelegenheit. Dies wiederum erhöht die Wahrscheinlichkeit einen Angriff unterbinden zu können, bevor der Angreifer einen offenen UDP-Port erkennen konnte. So kann zum Beispiel ein Scan der ersten 1024 Ports bis über vier Stunden Zeit in Anspruch nehmen [Rue07].

3.5. Adress Resolution Protocol (ARP)

Das *Adress Resolution Protocol* (kurz ARP) stellt eine Verbindung zwischen dem IP und den untergelagerten Protokollen (zumeist Ethernet) her, indem es die IP-Adressen auf sogenannte MAC-Adressen umwandelt. Dies ist notwendig, da niedrigere Protokolle als das IP, keine IP-Adressen kennen und somit keine Weiterleitung der Pakete möglich wäre. Deshalb verwendet das Internet Protocol ARP zur Umwandlung ihrer 4 Byte IP-Adresse (IPv4) in die 6 Byte lange MAC-Adresse (zum Beispiel 00:19:D2:9B:38:AD). Hierbei wird auf Ethernet-Ebene eine Anfrage an alle Rechner gestellt (beinhaltet MAC- sowie IP-Adresse des Senders sowie IP-Adresse des Empfängers), wobei der Rechner mit

der entsprechenden IP-Adresse antwortet und im Antwortpaket seine MAC-Adresse vermerkt.

Netzwerkkarten nehmen für gewöhnlich nur an sie per MAC-Adresse adressierte Pakete an, wobei alle anders adressierten verworfen werden. Die MAC-Adresse wird vom Hersteller eindeutig in das Interface integriert, wobei auch Netzwerkkarten sowie Betriebssysteme (zum Beispiel Linux) existieren, welche diese Adresse im Betrieb per Softwareeinstellungen ändern können.

Darüber hinaus ist es möglich gewisse Netzwerkkarten in den *Promiscuous Mode* zu schalten, was den Empfang aller Netzwerkpakete ermöglicht und ursprünglich für die Analyse des Netzwerkverkehrs mittels spezielle Diagnosetools (sog. Sniffer) Verwendung fand beziehungsweise findet. Dies stellt verständlicherweise ein gewisses Sicherheitsrisiko dar, da dabei auch die Schwächen von übergeordneten Protokollen zum Beispiel durch das Auslesen von (Plain-Text)-Passwörtern ausgenutzt werden können [Tan02].

ARP-Sicherheitsaspekte ARP ist im Speziellen für Spoofing anfällig, wobei hierbei der Angreifer versucht einen ARP-Buffer-Eintrag am Ziel zu erreichen. Damit würde sich der Angreifer als wichtiger Knotenpunkt (zum Beispiel Router oder aber auch Datenbankserver) ausgeben, was zur Folge hätte, dass alle entsprechenden Anfragen an diesen weitergeleitet werden würden. Dies würde zum Beispiel ein Auslesen von Passwörtern und Ähnlichem erleichtern [WP07].

ARP-Firewall-Aspekte Als eines der Probleme von ARP kann der Umstand gewertet werden, dass es so gut wie keine (integrierte) Logik zur Überprüfung von ARP-Nachrichten sowie deren Richtigkeit gibt.

In der Praxis verarbeiten die meisten Betriebssysteme ARP-Nachrichten ohne diese auf Richtigkeit beziehungsweise Inkonsistenzen zu überprüfen [Rue07]. Eine Überprüfung des ARP-Verkehrs wird zumeist Softwarekomponenten höherer Intelligenz (zum Beispiel arpwatch[6]) überlassen und ist somit für die Betrachtung unter dem Aspekt der Firewall ausgeschlossen.

[6] Populäre quelloffene Lösung zur Überwachung der lokalen ARP-Tabelle von Craig Leres (Universität von Berkeley, Kalifornien).

3.6. Internet Control Message Protocol (ICMP)

Das *Internet Control Message Protocol* (kurz ICMP) stellt die Funktionalität zur Rückmeldung von Statusmeldungen (zum Beispiel Fehlermeldungen) sowie zur Informationsbeschaffung (zum Beispiel PING-Request) im Netzwerk bereit.

Das ICMP ist ebenso, wie das dazugehörige IP, bereits in der zweiten Version (Version 6) verfügbar, wird jedoch wegen der (noch) spärlichen Verwendung hier ausgeklammert.

Die ICMP Nachrichten Typen werden zur Übermittlung in IP-Pakete eingekapselt. Obwohl eine Vielzahl an Nachrichten Typen definiert sind, sollen in der nachfolgenden Tabelle nur die wichtigsten dargestellt werden.

Message Type	Beschreibung
Destination unreachable	Paket konnte nicht ausgeliefert werden
Time exceeded	Time to Live Feld hat Wert 0 erreicht
Parameter problem	Header Felder ungültig
Source quench	Geschwindigkeit drosseln
Redirect	Vermutlichen Routingfehler an Host melden
Echo	Nachfrage ob Rechner erreichbar ist
Echo reply	Positive Rückmeldung auf Echo
Timestamp request	Prinzipielle wie Echo-Request nur mit Zeitstempel
Timestamp reply	Prinzipielle wie Echo-Reply nur mit Zeitstempel

Tabelle 3.1.: Die wichtigsten ICMP Nachrichten Typen [vgl. Tan02, S. 449]

Aufgrund der Möglichkeiten, welche ICMP bereitstellt, können spezielle Nachrichten Typen zumeist schon auf Betriebssystembasis (zum Beispiel Echo-Reply) deaktiviert werden.

ICMP-Sicherheitsaspekte ICMP eignet sich aufgrund der bereitgestellten Kontrollfunktionalität besonders für DoS Angriffe [WP07]. Darüber hinaus kann es auch zum Fingerprinting (Informationsbeschaffung) von Hosts eingesetzt werden. Eine weitere Möglichkeit ICMP zu missbrauchen, liegt in der mutwilligen Umleitung von Verbindungen (Redirects).

ICMP-Firewall-Aspekte Bei ICMP-Paketen beziehungsweise -Nachrichten ist eine spezifische Filterung von Nachrichten aus dem externen Netz (sprich Internet) sinnvoll. Die Kommunikation im internen Netzwerk wird aufgrund der Firewall Definition davon nicht betroffen sein, wobei eine Erhöhung des Sicherheitsstandards realisierbar ist. Zu den üblicherweise gefilterten Nachrichtentypen zählen etwa der Echo-Request, ICMP-Timestamp-Request als auch die ICMP-Adress-Mask-Request-Pakete, welche eine Analyse des Netzwerks als auch eine Flooding ermöglichen würden [Rue07].

3.7. Domain Name System (DNS)

Als *Domain Name System* (kurz DNS) bezeichnet man den Dienst zur Umwandlung von logischen Adressen (zum Beispiel `www.kogelbauer.com`) in IP-Adressen (zum Beispiel 127.0.0.1), was durch die fehlende Interpretationsfähigkeit des IP notwendig wird. Die Verwendung von IP-Adressen durch Menschen wird durch deren mangelnde Fähigkeit, eine Vielzahl unterschiedlicher Zahlen zu behalten, nahezu ausgeschlossen, weshalb für den Menschen die leicht interpretierbaren und merkbaren logischen Adressen eingesetzt werden. Dem DNS kommt dabei die Funktion eines dezentralen Telefonbuchs zu [Tan02].

Aus Gründen der Verwaltbarkeit wird zur Namesauflösung das Internet in Zonen eingeteilt, welche wiederum in Unterzonen eingeteilt werden. Diese Schachtelung setzt sich weiter fort und ergibt bei genauer Betrachtung beziehungsweise grafischer Darstellung eine Baumstruktur. Festzuhalten ist, dass für jede Zone meist zumindest zwei Rechner, aus Gründen der Verfügbarkeit, redundant diesen Dienst anbieten. Ausgehend von der TLD (Top Level Domain) werden die einzelnen Ebenen bei der Umwandlung durchschritten, wobei der Server, welcher sich in der Zone der nachgefragten Domäne befindet,

die IP-Adresse zurückliefert.

Das DNS wird üblicherweise über UDP auf dem Port 53 angesprochen, wobei eine Verwendung von TCP auf demselben Port auch möglich ist. Die Verwendung von TCP wird vorwiegend beim Überschreiten eines gewissen Umfangs, zum Beispiel *Zone Transfer Requests*[7] eingesetzt.

Folgende Grafik soll zur Verständlichkeit beitragen:

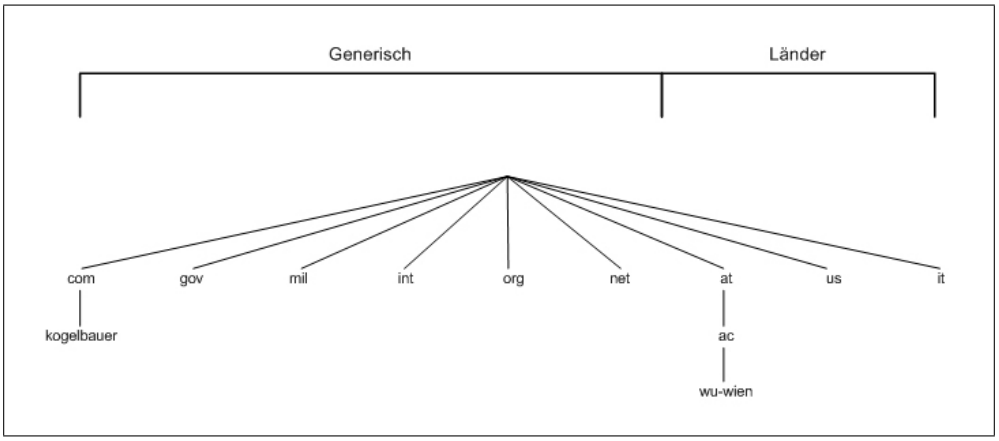

Abbildung 3.5.: Der Internet Domain Name Space [vgl. Tan02, S. 581]

DNS-Sicherheitsaspekte Die größte Gefahr bei der (mehr oder weniger unumgänglichen) Verwendung von DNS liegt in der Tatsache, dass es durch die Verwendung von UDP möglich ist, den Cache eines DNS-Servers zu manipulieren (poisoned cache) [Tan02]. Beim sogenannten DNS-Spoofing wird grob vereinfacht dargestellt, eine Anfrage an den DNS-Server zur Umwandlung einer (ihm) unbekannten Adresse gestellt, dieser stellt dann seinerseits eine Anfrage an die nächst höhere Ebene, wobei die Antwort von dieser (Ebene) vom Angreifer injiziert wird. So ist es möglich, jede beliebige IP-Adresse zu hinterlegen, was in weiterer Folge dazu verwendet werden kann, um eine *Man-in-the-Middle-Attacke*, für das Opfer unbemerkt, durchzuführen. Eine Weiterentwicklung und

[7] Ermöglicht den Transfer des gesamten DNS Datenbankinhalts einer Ressource (DNS-Server) und wird meistens zur Synchronisierung der redundanten DNS-Server eingesetzt.

Sicherung des DNS ist unter dem DNSSec Projekt derzeit im Gange, wenngleich noch nicht vollständig umgesetzt.

DNS-Firewall-Aspekte Da DNS-Anfragen vorwiegend Ressourcen im Internet betreffen, und die interne Adressstruktur im Netzwerk ja bekannt ist, ist eine Kontrolle der Anfragen mittels Firewalls als schwierig anzusehen. Eine Möglichkeit die Sicherheit dennoch zu erhöhen, ist die Installation eines Caching-DNS-Servers am Firewall-System, welcher jedoch von außen nicht zugänglich sein darf [Rue07]. Der Einsatz geeigneter Software ermöglicht eine Kontrolle der DNS-Ergebnisse in Form einer Gegenabfrage auf anderen DNS-Servern. Ein Gefahrenpotential, zum Beispiel durch vergiftete Caches bei den DNS-Servern, ist jedoch nicht ausschließbar, wenngleich es unwahrscheinlich ist, dass mehrere DNS-Server gleichzeitig betroffen sind.

4. Angriffe und Angriffsarten

Das Agieren innerhalb vernetzter Umgebungen birgt eine Vielzahl von Gefahren, sei es im weltweiten Internet oder nur im firmeneigenen Netzwerk. Dezidiert sei darauf hingewiesen, dass Angriffe, welche durch Insider verübt werden, statistisch gesehen eine höhere Erfolgswahrscheinlichkeit haben, da diese mit dem Netzwerk vertraut sind, somit also die Serverinfrastruktur zumeist kennen und diesen Umstand ausnutzen können [Tan02].

Das nachfolgende Kapitel soll ein Art Fundament an Wissen über die gängigsten Angriffe und Angriffsarten vermitteln. Es soll zeigen, wie Schwächen, zum Beispiel der Protokolle oder aber auch von fehlerhafter Hardware, mutwillig ausgenutzt werden können, um an vertrauliche Information zu gelangen beziehungsweise um den Kommunikationsfluss einer Stelle zu unterbinden. Weiters soll hervorgehoben werden, dass dieses Kapitel keinen Anspruch auf Vollständigkeit erhebt, da die Anzahl sowie die Variationen der Angriffe stetig steigen und somit eine statische Behandlung dieser Thematik unmöglich macht. An dieser Stelle sei auch darauf hingewiesen, dass eine hundertprozentige Sicherheit niemals erreicht werden kann, wobei eine zuverlässige Weiterentwicklung beziehungsweise zuverlässiges Aktualisieren der Systeme das Risiko beträchtlich reduziert.

Dieser Umstand gewinnt vorwiegend dadurch an Bedeutung, dass die Vielzahl der Angriffe von „ungeübten“ Amateur-Crackern durchgeführt wird. Diese Praktik, wobei vorgefertigte Programme, welche gezielt auf bekannte Exploits ausgerichtet sind, eingesetzt werden, ist generell bekannt als “Script Kidding“ [FG05].

4.1. Denial-of-Service (DoS) Attacken

Denial-of-Service (kurz DoS) Attacken sind die wohl meist bekanntesten Angriffe der Fragmentierungsattacken. Ihr Ziel ist es, ein System beziehungsweise einen Server „lahm“ zu legen beziehungsweise dessen Kommunikationsfähigkeit stark beziehungsweise ganz

einzuschränken. Bekannte Vertreter dieser Gattung sind zum Beispiel „Ping of Death“ (besonders unter Win95) sowie die aktuellere Teardrop-Attacke [FG05].

Kurzzeitig in den Hintergrund gerückt, kommt DoS-Angriffen erneut besondere Bedeutung zu, da die Verbreitung von WLAN (Wireless Lan) Funknetzwerken selbst bei verschlüsselter Ausprägung (insb. WEP) diese ermöglichen. Das einfache Auslesen der benötigten Daten mittels eines Sniffers verstärkt diesen Effekt, wobei hier zum Beispiel von „Deauthentication Flooding“ gesprochen wird [WP07].

Nachfolgend sollen die unterschiedlichen Techniken zur Durchführung solcher Attacken erläutert werden, wobei diese in „DoS mittels Flooding“ und „DoS mittels ICMP“ untereilt werden.

4.1.1. DoS mittels Flooding

Unter dem Begriff „Flooding“ werden Praktiken zusammengefasste, welche das Überlasten eines Rechners mittels des Sendens einer Vielzahl von Netzwerkpakten herbeiführen beziehungsweise zum Ziel haben.
Als eine der vielversprechendsten Techniken hat sich das Versenden von (möglicherweise gefälschten) TCP-Paketen mit gesetztem SYN-Flag etabliert. In diesem Zusammenhang wird deshalb auch von SYN-Flooding gesprochen. Ein derart modifiziertes TCP-Paket hat zur Folge, dass der Empfänger (beziehungsweise das Opfer) versucht eine Verbindung zum Anfragenden herzustellen. Ist dies jedoch unmöglich, so werden dennoch die benötigten Ressourcen (zum Verbindungsaufbau) für eine bestimmte Zeit (zum Beispiel 75 Sekunden) belegt und erst nach Ablauf dieser Frist wieder freigegeben. Ist demnach eine ausreichende Anzahl an Paketen an das Opfer geschickt, ist es möglich, den angesprochenen Port, des Hosts zu blockieren (zumindest für die Timeout-Zeit).

Eine andere Möglichkeit des Floodings besteht darin, direkt die Netzwerkanbindung des Opfers durch eine Vielzahl von Paketen zu blockieren beziehungsweise lahmzulegen. Ziel solcher Attacken sind vorwiegend Web-Server großer Internetfirmen. Das ein solcher Angriff, ausgehend von nur einem Angriffsrechner nahezu unmöglich ist, ist dadurch begründet, dass die anzugreifenden Server zumeist über eine höhere Übertragungskapazität als die des Angreifers verfügen. Selbst wenn die Übertragungskapazitäten gleich wären (am Punkt der Anknüpfung), ist es als sehr unwahrscheinlich anzusehen, dass am Übertagungsweg keinerlei Engpässe auftreten, welche diesen Angriff zunichte machen.

Die Lösung besteht in der Bildung von regelrechten Angriffsnetzwerken beziehungsweise eines Verbundes von Rechnern. Diese Art des DoS wird auch als DDos (Distributet Denial of Service) bezeichnet und kann zum Beispiel mittels eines Refektor-Angriffes realisiert werden. Diese Methode zielt darauf ab, einem Rechner (oder im Idealfall ein ganzes Subnetz) durch das Zuspielen eines Paketes mit gefälschter Absenderadresse dazu zu bewegen, eine Antwort an das Opfer zu senden. Beim Erreichen der „kritischen“ Masse an Paketen bricht die Verbindungsfähigkeit des Kommunikationsnetzes beim Opfer zusammen.
Eine weitere Möglichkeit besteht darin, eine ausreichend große Anzahl an Rechnern zu „kapern“, um auf Befehl hin, eine solchen Angriff zu starten [Rue07].

Eine Möglichkeit sich gegen SYN-Flooding zu schützen besteht, zumindest unter Linux-Systemen darin, die sogenannten Syncookies zu aktivieren, welche dazu führen, dass, obwohl die Verbindungstabelle gefüllt ist, dennoch Verbindungen angenommen werden können.

Das Abwehren von Angriffen, welche sich direkt gegen das Kommunikationsvermögen des Opfers richten, ist bis dato nicht beziehungsweise nur bedingt möglich. Angriffe bei denen das eigene Netz als quasi „Verstärker“ dienen soll, sprich Reflektor-Angriffe, können durch den Einsatz einer Firewall verhindert werden [Les06]. So können zum Beispiel Pakete mit einer gefälschten Absenderadresse, die an eine interne Broadcast-Adresse weitergeleitet werden sollen, herausgefiltert werden. Somit kann der kritische Verstärkungseffekt, zumindest aus dem eigenen (Sub-)Netzwerk, unterbunden werden und eine unbeabsichtigte Mittäterschaft vermieden werden.

4.1.2. DoS mittels ICMP

ICMP eignet sich aufgrund seiner mangelnden Sicherheitsoptionen und der im Protokoll hinterlegten Funktionen zu einer ganzen Reihe von Flooding Angriffen.
Als klassisches Beispiel sei der Angriff mittels ICMP-ECHO-Paketen genannt, welche ursprünglich als Ping Befehl Verwendung finden, und als bekannter Ping-of-Death Angriff zum Flooding missbraucht werden kann.
Bei bekannten Ziel- und Quelladressen kann mittels des ICMP-Befehls *Destination Unreachable* ein DoS-Angriff geführt werden, welcher dem Quellrechner ein schlichtes Nichtvorhandensein des Zielrechners vorgaukelt.
Weiters kann mittels ICMP einem Opfer durch Manipulation des Routings ein nicht existenter Rechner als Router zugewiesen werden, was eine Trennung vom Netzwerk zur

Folge hat. Dies kann zum Beispiel mittels *Redirect* des ICMP-Befehlssatzes realisiert werden. Auch ein mutwilliges Antworten auf eine ICMP-RDP-Anfrage (Router Discovery Protocol) mit verfälschten Parametern führt zum selben Ergebnis, nämlich der Isolation des Zielrechners.

Die Verwendung einer Firewall bringt in diesem Zusammenhang für Nachrichten aus dem internen Netz (mit dem Ziel des internen Netzes) keinen wünschenswerten Effekt [Les06]. Zur Abschirmung gegenüber des Internets hat die Verwendung einer FW dennoch Sinn, da ICMP-RDP-Pakete zum einen komplett herausgefiltert werden können und zum anderen restriktive Regeln für den Umgang mit ICMP-Paketen realisiert werden können.

4.2. Cracking

Unter dem Begriff *Cracking* versteht man das Erlangen von Rechten zum Ausführen von Funktionen beziehungsweise Programmen, obwohl man dazu nicht berechtig ist. Ziele können das Erlangen von Schreibrechten auf Dateien, bis hin zur Möglichkeit Fremdcode auszuführen sein. Die schlimmste Ausprägung stellt die Erlangung der Root-Rechte dar, wobei in diesem Fall der Angreifer in der Lage ist, jedwede Aktion auf dem fremden System auszuführen.

Da der Prozess des Cracking um ein vielfaches schwieriger und komplizierter ist als das Skript-Kidding, ist es nur sehr geübten (IT-)Spezialisten vorbehalten. Weiters gibt es kein allgemeines „Kochbuch“ beziehungsweise Rezept um einen fremden Rechner zu kompromittieren, wohl aber eine Verfahrensbeschreibung. Diese soll hier kurz unter dem Bergriff des Crackings zusammengefasst dargestellt werden.

Die einzelnen Schritte sind [Fyo98]:

1. Auswahl des Ziels
2. Informationsbeschaffung
3. Einbruch in den Rechner
4. Rootrechte erlangen
5. Sicherungsmaßnahmen (für wiederholte Angriffe)

6. Vorbereitung des nächsten Angriffs
7. (Tieferes Eindringen in das fremde Netzwerk)

Für eine detailierte Aufschlüsselung der Prozessschritte, sowie deren Inhalte, verweist der Autor auf die einschlägige Literatur[1].

Die wohl wichtigsten Teilschritte liegen in der Bestimmung des am Ziel installierten Betriebssystems und der laufenden Dienste. Diese können darüber Aufschluss geben, welche Exploits vorhanden und ausgenutzt werden können. Die Feststellung des Remote-Betriebssystems kann zum Beispiel mittels TCP/IP Stack Fingerprinting bewerkstelligt werden.

Die Feststellung des Betriebssystems erlaubt Rückschlüsse auf die möglichen verwendeten Dienste, welche über die Analyse der geöffneten Ports ermittelt werden können [Fyo97].

Ein Auftreten solcher Angriffe kann durch den Einsatz von Firewalls beziehungsweise -systemen zumindest verzögert werden. Generell auszuschließen sind sie aufgrund des fortgeschrittenen Wissenstandes der Angreifer dennoch nicht, wobei aus Sicht des Administrators fehlerhafte Protokoll(-implementierungen) sowie Anwendungen eine wesentliche Gefahr darstellen.

4.3. Würmer und Trojaner

Unter dem Begriff „Wurm“ verstehen wir eine Weiterentwicklung von Angriffsskripten in der Hinsicht zur selbstständigen Verbreitung. Wenn also ein System erfolgreich angegriffen wurde, wird dieses sogleich als Ausgangspunkt für die selbstständig initiierte Verbreitung im Netzwerk verwendet. Die Gefahr beziehungsweise der Erfolg dieser Würmer ist in homogenen Systemen besonders hoch, jedoch mit der Charakteristik, dass diese für eine genau spezifizierte Anzahl an Exploits entwickelt wurden, was den Effekt bei einer Immunsierung (zum Beispiel durch Updaten) abschwächt [WP07].
Würmer haben in den letzten Jahren vor allem für das Anlegen von Botnetzen (zum Beispiel zur Verbreitung von SPAM) sowie für die Durchführung von DDoS Angriffen

[1] zum Beispiel: [Les06] – Lessig, Andreas G.: *Linux Firewalls- Ein praktischer Einstieg.* O'Reilly, 2. Auflage, 2006.

an Interesse gewonnen, wobei auch das Ausspionieren der befallenen Systeme eine beliebte Tätigkeit darstellt.
Besteht bereits ein Wurmbefall im Netzwerk, so ist die Firewall generell nicht in der Lage die Verbreitung in diesem einzudämmen. Die FW kann jedoch zur Unterbindung der Kommunikation des Wurmes mit seinem „Mutterschiff“ eingesetzt werden, wodurch ein Nachladen von Fremdcode verhindert werden kann [Les06].

Als *Trojaner* werden generell Programme beziehungsweise -teile bezeichnet, die sich nach außen hin als harmlos darstellen in der Realität jedoch Schadcode enthalten und es einem Angreifer ermöglichen über eine Hintertür in das fremde System einzusteigen [WP07]. Sie operieren dabei meist im Verborgenen, um einem Entfernen beziehungsweise Erkennen entgegen zu wirken.
Es gibt eine Vielzahl von Möglichkeiten wie ein solches Programm ins System kommen kann.

Die wichtigsten beziehungsweise gefährlichsten sind [Les06]:

- E-Mail Anhang
- Programmdownload von infiziertem Server
- Website mit bösartigem ActiveX-Control Element
- Verwendung eines unsicheren Web-Browsers
- Herunterladen von Spyware (mit Nachladefunktion)

Ein Rechner der mit einem Trojaner infiziert ist, stellt ein ernst zu nehmendes Sicherheitsrisiko dar, da auch in diesem Fall der Angreifer in der Lage ist schädliche Aktionen mit diesem Rechner durchzuführen.

Generell gilt, dass der Schutz vor Trojanern durch technische Maßnahmen nur bedingt möglich ist, wobei die Verwendung eines (aktuellen) Virenscanners eine Vielzahl dieser Schädlinge erkennen und auch beseitigen kann. Jedoch ist die Umstellung beziehungsweise Neuerstellung eines Trojaners nicht mit viel technischem Aufwand verbunden, wobei dies ein nicht Erkennen durch den Scanner zur Folge hätte. Dieser Umstand bedingt eine verhaltenstechnische Unterweisung der Mitarbeiter für diese Problematik.
Die Verwendung einer Firewall hat in der Regel nur sehr wenig Einfluss auf die schädliche Wirkung von Trojanern. Sie vermeidet zum einen nicht das Herunterladen von schädlichen Dateien aus dem Internet, zum anderen kann sie jedoch zumindest unter

bestimmten Umständen die Kommunikation dieser Schadsoftware einschränken beziehungsweise unterbinden [Les06].

4.4. Schädliche Inhalte in HTML-Seiten

Um die Interaktivität von HTML-Seiten zu erhöhen, werden mehrere Technologien zur Erstellung sogenannter „aktiver“ Inhalte bereitgestellt. Diese Bereitstellung von aktiven Inhalten beruht zumeist auf der Ausführung kleiner Programme, welche in die Webseite eingebunden werden können.

Bekannte Vertreter, der für aktive Inhalte konzipierten Technologien sind etwa *Java-Applets*, *Skriptsprachen* (wie JavaScript, VB-Script) sowie ActiveX-Controls. Da diese Programmteile für gewöhnlich direkt auf dem Rechner, welcher die Seite anzeigt beziehungsweise abruft, ausgeführt werden, besteht die Gefahr, dass schädlicher Code ausgeführt werden kann. Dieser kann unter Umständen dazu Verwendung finden um Würmer, Viren oder Trojaner in den Rechner zu injizieren.

Eine Vielzahl der schädlichen Programmteile wird durch die, zum Download notwendigen Signatur bereits durch die damit verbundene zeitliche Anstrengung sowie der Bekanntmachung der eigenen Person unterbunden. Dieses Zertifikat wird jedoch nicht von jedem Browser als zwingend angesehen beziehungsweise für jede Technologie angeboten, weshalb erneut die Verwendung eines als „sicher“ beziehungsweise „bugfrei“ geltenden Web-Browser empfohlen wird. Zudem ermöglichen diverse Optionen (im Browser) die Unterbindung des Downloads solcher Elemente, was einer Verbesserung der Sicherheitslage entspricht. Jedoch gelingt es findigen Programmierern immer wieder, Bugs in den Browsern als Angriffspunkt zu nutzen und somit den Rechner zu infizieren [Jan07].

Beim Einsatz eines entsprechenden Firewallsystems kann eine Filterung etwaiger aktiver Elemente zum gewünschten Erfolg des Unterbindens dieser eingesetzt werden. Dies ist jedoch auch immer mit einer Einschränkung der Funktionalität beziehungsweise Interaktivität, die zentral bedingt ist, verbunden. Es wird daher empfohlen, Systeme zu nutzen, welche den Zugang zu etwaigen schädlichen Seiten im Voraus unterbinden, ohne die Funktionalität der „restlichen“ Seiten zu beeinträchtigen. Diese agieren zumeist über (regelmäßig aktualisierte) Listen von schädlichen Seiten, die aus dem internen Netz nicht besucht werden dürfen und die Anfragen zu diesen somit herausfiltern [Les06].

4.5. Sonstige

Zusätzlich zu den zuvor genannten Angriffsarten beziehungsweise -techniken gibt es eine Vielzahl derer mehr, von denen einige in diesem Abschnitt behandelt werden sollen. Festzuhalten gilt, dass es für ein Netzwerk beziehungsweise eine Firmeninfrastruktur durchaus Bedrohungen gibt, welche nicht technischer Natur sind und somit auch nicht mit technischen Hilfsmitteln unterbunden werden können (zum Beispiel Social Engineering).

Angriffe auf die Privatsphäre Unter *Angriffen auf die Privatsphäre* werden Praktiken zum Ausspionieren von Benutzern sowie deren Verhalten zusammengefasst, welche auf die Zuhilfenahme technischer Methoden beruhen [Les06].

Vielen Akteuren im World Wide Web als auch im Internet ist nicht bewusst, dass eine Verfolgung ihrer Schritte zur Erstellung eines Profils verwendet werden kann. So ist es zum Beispiel Werbeagenturen möglich zielgerichtete Werbungsinhalte an den so entstandenen gläsernen Menschen zu übermitteln.

Der unvermeidliche Ursprung der Datenweitergabe liegt beim Bereitsteller des Internetanschlusses (auch Provider genannt), welcher jeglichen Verbindungsaufbau sowie die übermittelten Daten mitlesen kann. In der Realität werden mittlerweile aufgrund juristischer Vorgaben alle Daten, welche in Verbindung mit den Verbindungsanfragen stehen mitgeloggt und gegebenenfalls (bei begründetem Verdacht) weitergegeben.

Die direkt nachgeschaltete Instanz, die von unseren Schritten informiert ist, ist natürlich der besuchte Server, der wohlgleich auch weniger Information über uns erhält. Zu diesen Informationen zählen etwa die IP - Adresse, welche aufgrund der vorherrschenden Knappheit jedoch zumeist dynamisch von den Providern an die Nutzer vergeben wird. Dieser Problematik wurde durch die Vergabe von *Cookies* entgegengewirkt. Welche technisch gesehen, kleine Textdateien darstellen und ein Identifizierungskriterium enthalten. Beim Besuch des selben Servers wird dieses dann automatisch (beim Request) mit übertragen und gibt somit unsere Identität preis. Diese Methodik funktioniert jedoch nur auf dem ausstellenden Server, wobei die Information vor anderen Interessenten (vermeintlich) verborgen bleibt.

Eine Server übergreifende Lösung stellt die Verwendung von zentral verwalteten Werbe-Bannern (dazu werden auch 1x1 Pixel großen Web Bugs gezählt) dar, wobei der Aufruf

dieser Bilder(-chen) einem erneutem Request entspricht. Dies ermöglicht seinerseits wieder das Setzen eines Cookies, wobei mittels HTTP-Referer-Tag eine Nachverfolgung der Benutzer ober Webseitengrenzen hinweg möglich ist. Dieser Umstand wird dann zur Erstellung des Benutzerprofils ausgenutzt und ist mittlerweile nicht mehr nur auf HTML-Seiten beschränkt. Die sogenannten *Web Bugs*, welche 1x1 Pixel große weiße Grafiken sind, machen ein Erkennen von Seiten des Benutzers besonders schwer und können mittlerweile auch in Office-Dokumente eingebaut werden. Dies ermöglicht, obwohl der Benutzer vermeintlich glaubt offline zu agieren, Information über das Lesen der Dokumente zu erlangen.

Abhilfe kann durch das Abschalten der Cookies im Browser geschaffen werden, jedoch ist auch dies wieder mit (unbeabsichtigten) funktionalen Einschränkungen anderer Seiten verbunden. Eine bessere Lösung stellt der Einsatz eines Proxy - Servers dar, welcher die Header-Informationen gegebenenfalls herausfiltern kann (insb. den http-Referer-Tag). Diese Funktionalität kann natürlich auch von einer Firewall wahrgenommen werden, wobei sich in der Praxis jedoch spezialisierte Produkte durchgesetzt haben, wie etwa Squid.

Eine drastischere und technisch aufwendigere Technik zur Informationsgewinnung stellt die Installation eines Sniffer-Programms auf dem Kommunikationskanal dar [Erb01]. Unter Umständen können so bis hin zum Tastenanschlag alle Bewegungen der Benutzer verfolgt und mitgelesen werden.

Social Engineering Der mythisierte Begriff des *Social Engineering* stellt eine Gefahr nicht technischer Natur dar und kann auf Grund dieses Umstandes auch nur schwer mit technischen Hilfsmitteln unterbunden werden [Gra01]. Im Falle des Social Engineerings versucht eine unberechtigte Person durch geschickte Manipulation von Personen(-kreisen) unauthorisierten Zugriff auf Daten beziehungsweise Systeme zu bekommen. Dabei macht sich der Angreifer „geschickt“ den Umstand zu Nutze, dass Menschen veranlagt sind, Vertrauensnetze zu bilden und private Information zu teilen. Ebenso ist die Ausnutzung einer vermeintlichen Machtposition eine mögliche Strategie um an die notwendigen Daten zu kommen. Dem Leser wird einleuchten, dass das beste System sowie dessen Sicherheitsmechanismen „wertlos“ sind, wenn das schwächste Glied – der Mensch – „bricht“.

Um eine Vorstellung von der Vorgangsweise zu bekommen sei hier ein Beispiel genannt: „Während der Arbeit ruft eine Person bei Ihnen an und gibt sich als Leiter der Verrechnungsabteilung aus, wobei die zornige Stimme bereits ein Fehlverhalten Ihrerseits vermuten lässt. Diese Person behauptet sogleich, dass ihr Telefonkonto (beziehungsweise

Limit) hoffnungslos überzogen sein. Während des Gesprächs erläutert dieser, dass soeben wieder ein Anruf nach Ägypten geführt werde, da er jedoch gerade mit Ihnen spreche, sie jedoch nicht der Schuldige sein könnten. Freundlicher Weise bietet er Ihnen an – gegen Bekanntgabe des Benutzernamens und PINs - diesen Missstand aus der Rechnung zu tilgen und den Schuldigen zu finden."

Phishing *Phishing* als „Sonderform" des Social Engineering stellt eine Attacke dar, welche erst durch das fehlende Sicherheitsbewusstsein der Benutzer sowie der veralteten Sicherheitsstandards der Dienstanbieter (zumeist Banken) ermöglicht wird. Sinn des Angriffes ist es, sensible Daten (zum Beispiel Bankdaten) durch mutwillige Täuschung des Anwenders zu bekommen, wobei zumeist vermieden werden soll, dass der Benutzer Verdacht schöpft. Dies spricht für die technische und betrügerische Ausgereiftheit solcher Angriffe und Angriffstechniken.

Der vereinfachte Ablauf eines Phishing-Angriffs ist [Jan07]:

1. Vorbereitung – In der Phase der Vorbereitung wird sowohl das Ziel ausgewählt, als auch die Anmeldeseite des Ziels bis ins kleinste Detail (womöglich mit Links zu Originalseite) gefälscht. Das dies natürlich keinerlei Probleme macht, da sowohl der Quellcode als auch die Bilder der Seite publik sind, sollte dem Leser verständlich sein. Zumeist werden solche gefälschten Seiten, welche den Benutzer dazu bewegen sollen seine Zugangsdaten Preis zu geben, so aufgebaut, dass die eingegebenen Daten zur Anmeldung an die Originalseite versandt werden, um einen etwaig aufkommenden Verdacht zu unterbinden. Weiters benötigt der Angreifer einen Web-Server, auf der seine Seite veröffentlicht werden kann. Zum Verbergen der wahren Adresse kommen unterschiedlichste Mechanismen zum Einsatz, welche von einfacher Angabe der IP-Adresse, bis hin zum Verstecken der Adressleiste im Browser reichen. Diese Maßnahmen zielen darauf ab, dass der Benutzer - welcher die ursprüngliche Seite genau kennt - durch seine Unaufmerksamkeit den Unterschied nicht bemerkt.

2. Angriff - Der Angriff beginnt mit dem Versenden von Massenmails an Adressen, die im Vorfeld gesammelt wurden. Zum Vertuschen des wahren Absenders wird die Absender-Adresse gefälscht und möglicherweise ein Bot-Netz zu Verbreitung genutzt. Die Aufmachung des E-Mails muss dabei natürlich im Corporate Design des anzugreifenden Dienstes gestaltet sein und beinhaltet den Link zur gefälschten Seite. Durch Folgen des Links durch den Benutzer, wird in weiterer Folge eine Preisgabe der vertraulichen Information angestoßen. Mit diesen Daten ist es

dem Angreifer dann in weiterer Folge möglich, zu einem späteren Zeitpunkt, zum Beispiel Geld vom Konto zu überweisen (für den Fall, dass das Ziel eine Online-Banking Applikation war).

3. Beutezug - Um nun die Beute zu erhalten, ohne Gefahr zu laufen sein eigenes Konto Preis zu geben, könnte zum Beispiel eine Scheinfirma im Internet angepriesen werden, welche freie Mitarbeiter mit Höchstbeträgen entlohnt. Nach der Überweisung des Betrages auf das Konto der so gewonnenen „freien“ Mitarbeiter folgt ein Bargeldtransfer (exklusive Profision) an die vermeintliche Firma (den Angreifer).

Da bei solch einem Angriff wieder der Mensch beziehungsweise seine Gutgläubigkeit das Ziel ist, ist der Schutz vor diesen technisch aufwendig und wird von spezialisierten Produkten realisiert. Hier sei nur festgehalten, dass dies nicht Aufgabe einer Firewall ist.

5. Firewallgrundkonzepte

Dieser Abschnitt behandelt Konzepte, wie Firewalls ausgelegt beziehungsweise implementiert werden können. Zusätzlich zu den beiden Grundkonzepten der Filter und Proxies werden auch Praktiken wie NAT (Network Address Translation) und kombinierte Konzepte erläutert. Der Leser soll darüber in Kenntnis gesetzt werden, welche praktischen Umsetzungstechniken es gibt und wie diese sinnvoll kombiniert werden können.

5.1. Filter (Paketfilter)

Die älteste und auch verbreitetste Implementierung von Firewalls, sind die Paketfilter, welche auf der TCP/IP Ebene fungieren und zumindest folgende Eigenschaften haben müssen [FG05]:

- Definierte Regeln zu Filterung des TCP/IP-Verkehrs (auf Paket-Ebene)
- Router-Funktionalität (daher Umsetztung einer IP-Weiterleitung auch Forwarding genannt)

Wenngleich diese Eigenschaften allen Paketfiltern als Basis dienen, so ist dennoch eine Unterscheidung in *Statische Paketfilter* und *Dynamische Paketfilter* sinnvoll und wird somit in weiterer Folge durchgeführt.

Paketfilter wirken daher vorwiegend auf der dritten beziehungsweise vierten Ebene des ISO/OSI - Referenzmodells und sind für übergelagerte Schichten und Protokolle praktisch „unsichtbar“. Daher wird ein Verbindungsaufbau erlaubt und weder Client noch Server wissen von der Existenz der Firewall, wohingegen ein Abweisen als ein schlichtes „Down“ des Servers interpretiert wird.

Nachfolgende Grafik soll dies verdeutlichen:

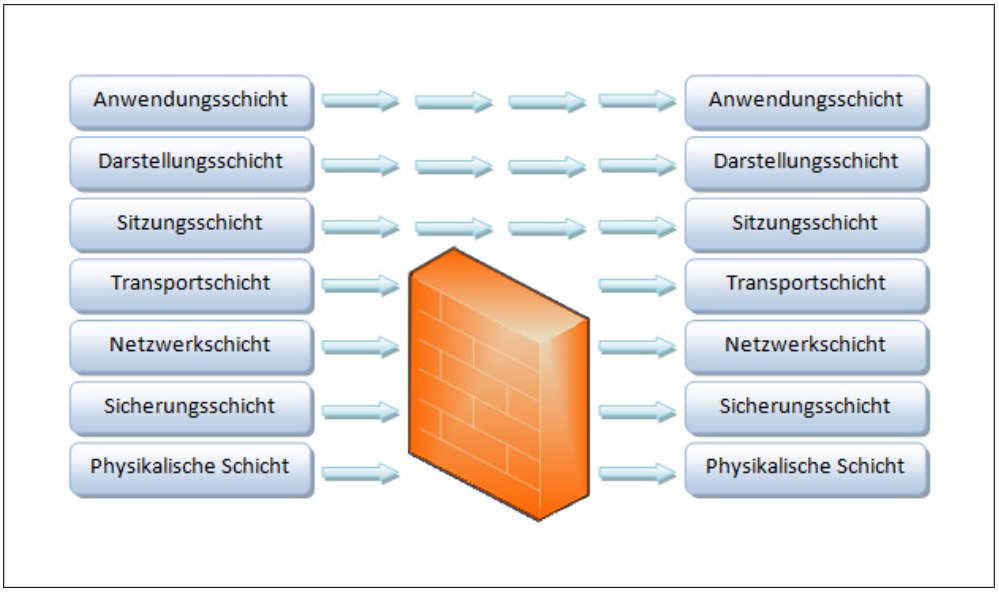

Abbildung 5.1.: (Dynamischer) Paketfilter im ISO/OSI - Modell [vgl. FG05, S. 50]

5.1.1. Statische Paketfilter

Statische Paketfilter, oder auch aufgrund ihrer geringen Schutzfunktion im Vergleich zu dynamischen Filtern *Access Control Lists* genannt, akzeptieren oder verwerfen IP-Pakete basierend auf ihrer IP-Adressen (Quell- als auch Zieladresse). Zudem können als weitere Filterkriterien die TCP/UDP-Ports, als auch der ICMP-Nachrichtentyp als Kriterien herangezogen werden.

Zumeist sind statische Filter auf Routern implementiert und verfügen über eine genau determinierte Liste von Regeln (die Access Control List), welche bei der Untersuchung sequenziell durchlaufen wird und damit in weiterer Folge zum Passieren beziehungsweise Verwerfen der Pakete dient.

Als die wohl größte Schwäche der Paketfilter statischer Natur ist die Statuslosigkeit anzusehen. Dies bedeutet, dass unbeachtet der vor- beziehungsweise nachkommenden Pakete, jedes Paket für sich, anhand des Regelsatzes überprüft wird und somit zum Beispiel mittels Fragmentierung leicht umgangen werden kann.

Nachfolgende Grafik soll den schematischen Aufbau, als auch die Funktionsweise ver-

sinnbildlichen:

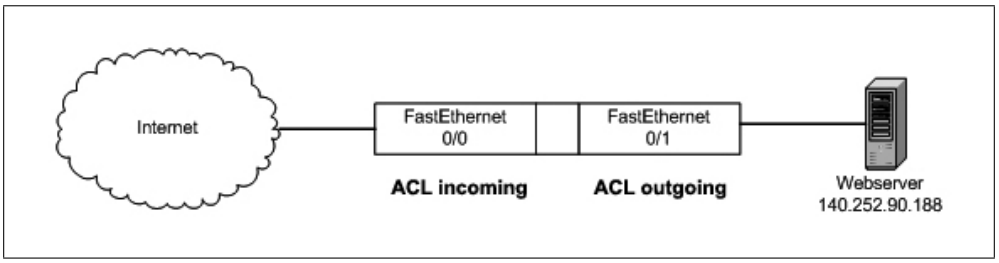

Abbildung 5.2.: Statische Filterung des Webserver-Zugriffs [vgl. FG05, S. 39]

Schwachstellen der Statischen Paketfilter Wie zuvor schon beschrieben, liegt der Hauptnachteil von statischen Filtern in der dezidierten Betrachtung jedes einzelnen Pakets, ohne Verknüpfung zu anderen herzustellen. Diese Statuslosigkeit hat nun folgende „negative“ Auswirkungen [FG05]:

- Minimaler Schutz vor Angriffen, welche auf der Fragmentierung beruhen
- Notwendigkeit für jeweils zwei Regeln der Verbindung (einmal incoming und einmal outgoing)
- Quellporteinschränkungen (outgoing) sind aufgrund der dynamischen Vergabe nicht realisierbar, was zu einem „ungewollten“ Zulassen aller Zielports führt

Die Einfachheit der Sperre von IP-Adressen beziehungsweise Bereichen sowie der minimale notwendige Ressourceneinsatz sprechen jedoch klar für den fortwährenden Einsatz dieser Filtertechnologie.

5.1.2. Dynamische Paketfilter

Dynamische Paketfilter erweitern den Funktionsumfang der statischen Filter um das Speichern der Statusinformationen einer Verbindung um diese zur dynamischen (auch „stateful“ genannten) Filterung zu verwenden.

Die Speicherung der Verbindungsinformation erfolgt in der sogenannten *Statustabelle* wobei der entsprechende Eintrag bis zur Beendigung der Verbindung aktualisiert und danach gelöscht wird. Zu den gespeicherten Informationen zählen der Protokolltyp (zum

Beispiel TCP, UDP, ...), etwaige Zusatzinformationen wie TCP-Flags und Sequenznummern, IP-Adressen (Quelle und Ziel) und bei TCP/UDP die anzusprechenden Ports (Quelle und Ziel). Bei ICMP-Nachrichten wird zusätzlich der Nachrichtentyp gespeichert.

Basierend auf diesen Informationen und der Gewissheit, dass Antwortpakete einen (bis auf den Datentyp) reziproken (Ziel ist nun Quelle und so weiter) Aufbau besitzen, kann der Paketfilter ein Paket einer Verbindung zuordnen und nur folgerichtige Antwortpakete passieren lassen, wobei die restlichen verworfen werden.

Vorteile gegenüber statischen Filtern Durch die Speicherung der Statusinformation ergeben sich folgende Vorteile von dynamischen Paketfiltern gegenüber denen statischer Natur [FG05]:

- Eine Regel pro Verbindung genügt
- Portbereiche müssen nicht mehr freigeschalten werden, da der Ziel- als auch der Quellport bekannt sind
- Antwortpakete sind nur innerhalb der Verbindung sowie deren Dauer zulässig und werden falls diese beendet ist verworfen (zum Beispiel gegen Replay-Attacken)

Das Beenden der Verbindung wird dabei entweder durch das gesetzte FIN-Flag oder einen sicherheitsbedingten Timeout (zum Beispiel wenn der Client abgestürzt ist), welcher in Verbindung mit dem zuletzt gesendeten Paket gestellt wird, eingeleitet.

Die als *Reflexive Access Control Lists* bezeichneten Regellisten werden dynamisch bei der Verbindungsherstellung erstellt und nach deren Beendigung wieder aus der Liste entfernt und ermöglichen so eine dynamische Filterung, die zudem auf nur einer Regel basiert.

Als eine Erweiterung beziehungsweise Verbesserung von dynamischen Paketfiltern kann die *Stateful Inspection* gesehen werden, welche bis in die Anwendungsprotokolle hinein filterrelevante Informationen extrahieren kann [FG05]. Diese tiefere Betrachtung von „höheren“ Protokollen ist notwendig, um das Freischalten von größeren Portbereichen (selbst bei dynamischen Filtern) unter der Verwendung von bestimmten Protokollen (zum Beispiel FTP) vermeiden zu können. Diese Funktionalität kann dann in weiterer Folge auch zur Unterscheidung von einfachen Routern mit reflexiven ACLs und den technisch aufwendigeren Stateful Inspection Firewallsystemen verwendet werden.

5.2. Proxies

Anders als beim Paketfilter (da auf gewissen Ebenen unsichtbar), stellen Proxy-Firewalls einen Aufbruch der Server/Client-Verbindung dar, weil sie als eine Art „Mittelsmann“ funktionieren. Dies bedeutet zum Beispiel, dass eine Verbindung vom Client zum Server als Verbindung zum Proxy aufgebaut wird, wobei dieser wiederum eine Verbindung zum Server herstellt, ohne das der Client in „direktem“ Kontakt zu diesem stehen würde.

Da dies auf allen Ebenen des ISO/OSI - Referenzmodells geschieht wird von einer „sichtbaren“ Firewall gesprochen, welche für das jeweilige (zu verwendende) Protokoll einen eigenen Proxy zur Verfügung stellen muss.

Folgende Grafik soll das Konzept der Proxy-Firewalls erläutern:

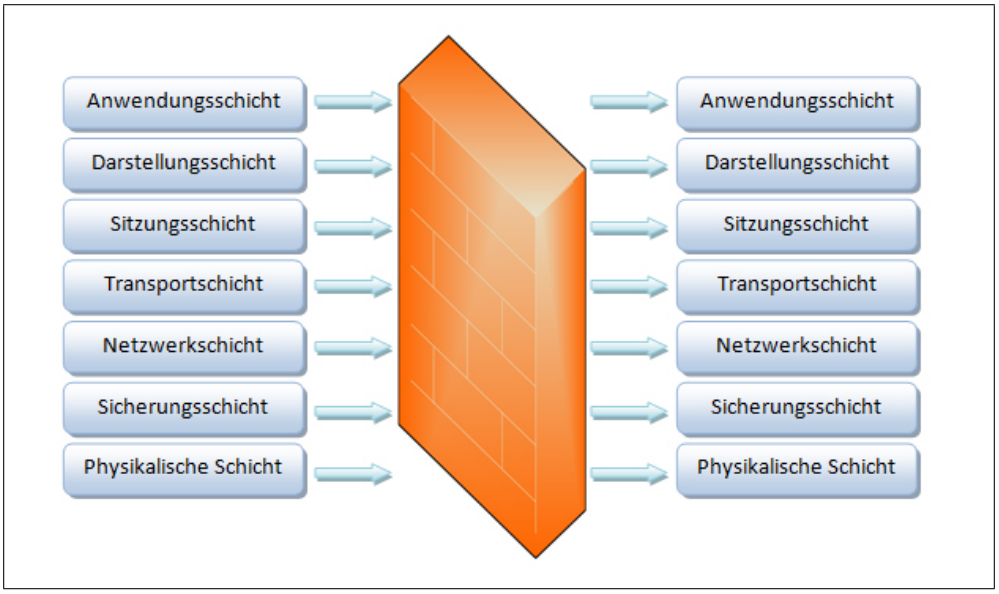

Abbildung 5.3.: Die nicht transparente Proxy-Firewall [vgl. FG05, S. 51]

Dieser Ansatz determiniert [FG05]:

- Für jedes zu unterstützende Protokoll muss ein Proxy zur Verfügung stehen welche dieses interpretieren und verstehen kann
- Die Proxy-Firewall ist für den Client sichtbar, da sich dieser mit dem Proxy verbindet, welcher eine andere IP-Adresse und zumeist auch unter einem anderen Port

zu Verfügung steht

- Korrekt konfigurierte Proxy-Firewalls benötigen keine Router Funktionalität (IP-Forwarding sollte deaktiviert sein), da diese nur Endpunkte beziehungsweise -knoten kennen und diese als Initiator (Verbindungsanfragender) ausgelegt sind

Vorteile von Proxy-Firewalls Der Hauptvorteil von Proxy-Firewalls gegenüber von Paketfiltern besteht in der vollständigen Implementierung von Filtermöglichkeiten der Protokolle bis hin zur Applikationsebene.

So können zum Beispiel bestimmte FTP-Benutzer gesperrt werden oder aber auch bösartige E-Mail Absender aufgrund des Verständnisses des SMTP-Protokolls abgeblockt werden.

Ein weiterer Vorteil liegt in der Protokollkonformität begründet, welche eine Verwendung von eingeschränkten Diensten auf dem Port eines anderen zu verhindern weiß (da die Struktur und der Aufbau bekannt sind).

Darüber hinaus besteht bei der Verwendungen einer Proxy-Firewall niemals eine direkte Verbindung vom externen Netz in das interne Netz, was bei der Übernahme (Hijacking) einer Verbindung bedeutet, dass sich der Angreifer noch nicht im internen Netz befindet und einen Sicherheitsvorteil birgt.

Probleme von Proxy-Firewalls Den zuvor beschriebenen Vorteilen stehen in der Realität aber auch eine Reihe von Problemen bei der Verwendung von Proxy-Firewalls gegenüber.

Zu diesen zählen etwa [FG05]:

- Hinzufügen eines zusätzlichen Dienstes erfordert die Bereitstellung eines neuen Proxies. Ist für diesen Dienst kein Proxy vom Hersteller vorhanden, muss auf ein Relay zurückgegriffen werden, welches etwaige Unregelmäßigkeiten im Datenverkehr bereinigt und aus sicherheitstechnischen Gründen nicht empfehlenswert ist.
- Proxies stehen vorwiegend für den Internetzugangsbereich bereit, jedoch nicht für

VPN[1]-Verbindungen was bei zunehmender (datentechnischer) Vernetzung zu Problemen führen kann.

- Hoher Ressourcenaufwand sowie die Entfernung zur Kernel-Ebene sprechen gegen den Einsatz.
- Proxy-Firewalls können aufgrund ihrer Implementierung als Netzwerkdienst selbst zum Ziel eines Angriffes werden.
- Zumeist führen Proxy-Firewalls nur grundlegende sicherheitstechnische Prüfungen auf protokollspezifische Eigenschaften durch, welche für den Unternehmenseinsatz teilweise zu wenig sind. Spezielle Anforderungen können es notwendig machen, Produkte von Drittanbietern zusätzlich zu installieren, um den Ansprüchen zu genügen.

5.3. Kombinationen

Aufgrund der Vor- und Nachteile beider zuvor genannter Technologien ist es nicht weiter verwunderlich, dass in der Praxis zumeist Kombinationen dieser Anwendung finden. Zu diesen zählen unter anderen Screened Hosts und Screened Subnets, aber auch eine Kaskadierung der Technologien ermöglicht einen sinnvollen und nutzbringenden Einsatz.

5.3.1. Hybrid-Firewalls auf Basis von Paketfiltern

Die als Hybrid-Firewalls, auf Basis von Paketfiltern, bezeichneten Firewalls, stellen generell gesprochen eine Erweiterung von dynamischen Filtertechnologien um Proxies für die gängigsten Protokolle dar und können somit auch einfache Aufgaben, wie Virus-Kontrollen oder URL-Filteraufgaben durchführen [FG05]. Sie werden somit quasi um eine Content-Security-Funktionalität erweitert und ermöglichen eine zentrale Administration dieser Belange.

Ein bekannter Vertreter dieser integrierten Lösungen, ist zum Beispiel die Firewall-1 von Check Point.

[1] Virtuelle Private Netzwerke

5.3.2. Hybrid-Firewalls auf Basis von Proxies

Anders als bei den Hybrid-Firewalls auf Paketfilterbasis findet sich im Segment der Hybriden-Firewalls auf Proxybasis kein nennenswertes Produkt auf dem Markt, was nicht zuletzt wegen der Vielzahl an Problemen bei den Proxies liegt, sondern auch darin begründet ist, dass Proxy-Firewalls vielfach bereits integrierte Funktionalität zur dynamischen Filterung bieten [FG05].

5.3.3. Hybrid-Firewalls in der Praxis

Wie aus den Vorabschnitten bereits hervorgegangen, werden vorwiegend kombinierte Konzepte auf Basis der Paketfiltertechnologie verwendet. Ob dabei eine integrierte Hybrid-Lösung verwendet wird oder einzelne Komponenten (von Drittanbietern) zur Erweiterung des dynamsichen Paketfilters verwendet werden, hängt zum einen von der Größe des Budgets und zum anderen von den Sicherheitsansprüchen ab [FG05].

Screened Host Beim Screened Host werden lokale Anfragen entweder an den Proxy gestellt, welcher sie in weiterer Folge weiterleitet oder aber sie werden direkt gestellt, wobei hier auf als sicher geltende Protokolle aufgebaut wird. Daher wird der Verkehr aus dem LAN ins Internet auf ein Minimum von Protokollen beschränkt, welche zudem gewissen Sicherheitsaspekten genügen müssen.

Folgende Grafik soll die Funktionalität des Screened Hosts verdeutlichen:

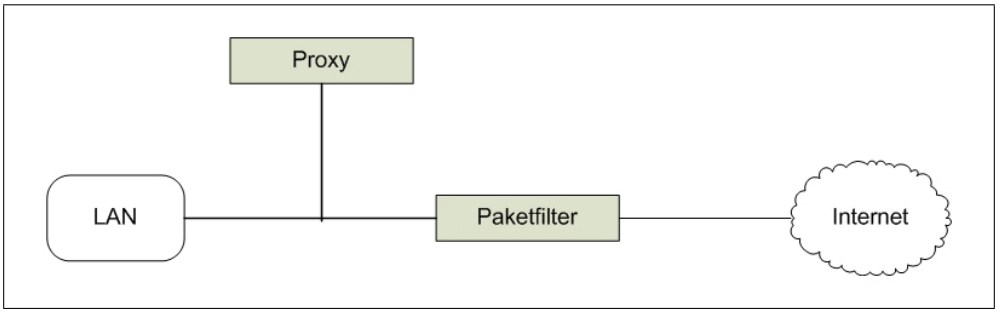

Abbildung 5.4.: Aufbau Screened Host [vgl. Les06, S. 71]

Die Aufgaben des Paketfilters bestehen hierbei aus [Les06]:

- Restriktion auf sichere Protokolle
- Verbindungen aus dem Internet dürfen nur an den Proxy weitergeleitet werden
- Unterscheidung der Protokolle und ob die entsprechenden Anfragen über den Proxy gehen müssen oder direkt passieren dürfen

Die Praktik des Screend Hosts ist vorwiegend für kleine Netzwerke gedacht, die zudem auf den Einsatz von eigenen (im Internet verfügbaren) Servern verzichten [Les06]. Dies würde bedeuten, dass ein Masquerading nicht möglich wäre, was zu einer Vielzahl von Gefahren führen würde, wie zum Beispiel ein durchaus sichtbares lokales Netzwerk (von Seiten des Internets).

Secure Server Net (SSN) beziehungsweise Screened Subnet Bei den Screened Subnets geht es um die Schaffung von Teilnetzen für bestimmte Aufgaben [Les06]. So werden zumeist (Teil-)Netze definiert die öffentlich über das Internet verfügbar sein sollen (zum Beispiel DMZ mit Web-Server und Mail-Server) sowie ein privates Netz, welches nicht von außen zugänglich ist. Der Sinn dahinter besteht in dem Vertrauensausschluss des öffentlich zugänglichen (Teil-)Netzes, welches ja durch Angriffe kompromittiert worden sein könnte. Ein Zugriff vom privaten Netz auf das öffentliche Teilnetz ist hierbei möglich, wobei aus Sicherheitsgründen vom öffentlich zugänglichen Teilnetz nicht auf das private Netz zugegriffen werden kann. Diese unterschiedlichen Zonen werden auch DMZ, also demilitarisierte Zonen genannt, wobei Screened Subnet und Secure Server Net als Synonyme verwendet werden können.

Für den Zugriff aus dem lokalen Netzwerk (DMZ 2) auf den öffentlich zugänglichen Teil (DMZ 1) unterscheidet sich die Anfrage nicht gegenüber jener einen Anfrage ins Internet.

Diese Teilung bringt noch weitere sicherheitstechnische Vorteile gegenüber den Screened Hosts, zumal es für größere Unternehmen gedacht ist und somit auch anderen Ansprüchen genügen muss. Einer dieser Vorteile liegt in der Verwendbarkeit beziehungsweise dem Einsatz von Masquerading, welches nun für den internen Teil Verwendung findet. Diese Unkenntlichmachung der internen Netzwerkstruktur erhöht die Sicherheit, wobei das zweite Teilnetz, in dem die Serverlandschaft angesiedelt ist, unberührt bleibt und des Weiteren von außen erreichbar ist.

Folgende Grafik soll dies verdeutlichen:

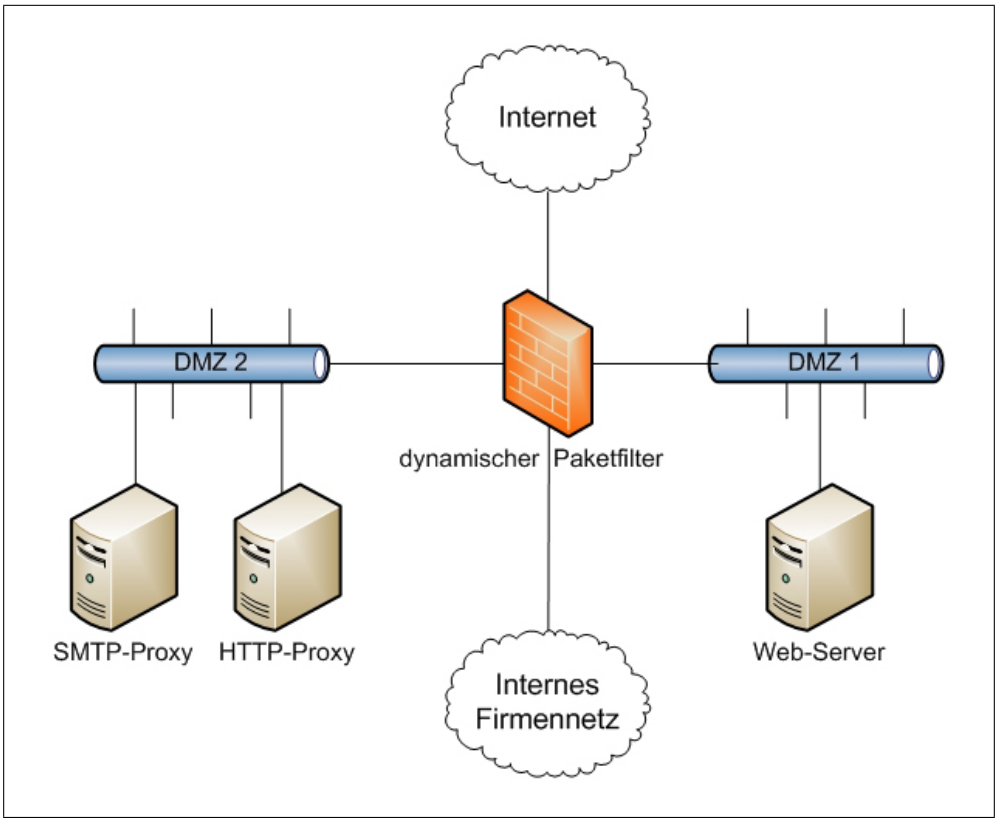

Abbildung 5.5.: Aufbau Screened Subnet [vgl. FG05, S. 56]

5.4. Exkurs: Network Address Translation (NAT)

Als Network Address Translation (kurz NAT) bezeichnet man die Umsetzung der IP-Adressen / Ports aus einem (lokalen) Netzwerk zum Internet und wurde bereits in Abschnitt 2.2 auf Seite 18 beschrieben. Ursprünglich wurde diese Funktionalität aus der Ressourcenknappheit des IPv4-Adressraums heraus entwickelt und ermöglicht es so, ein Netz beziehungsweise eine Vielzahl an Rechnern mit lokalen IP-Adressen unter einer offiziell registrierten und damit auch öffentlichen Adresse mit dem Internet zu vernetzten. Die Grundfunktionalität besteht demnach in der Verbindung verschiedener Netze

wobei, dennoch NAT oftmals eine Schutzwirkung zugesprochen wird. Dies basiert auf dem Gedanken, dass die Infrastruktur hinter der im Router beziehungsweise Firewall implementierten NAT für Angreifer nicht direkt einsehbar ist.

Die derzeit am häufigsten eingesetzte Variante des NAT stellt das Network Address Port Translation (kurz NAPT) dar, wobei hier neben den IP-Adressen auch die Portnummer umgewandelt beziehungsweise angepasst wird [Tan02]. Dieses „Maskieren“ wird in der Fachwelt auch als *Masquerading* bezeichnet und findet sich auch in preisgünstigen Hardwarerealisierungen im Privatbereich wieder.

Da NAT eine Umwandlung des Paketinhalts darstellt (Abänderung der Adresse und/oder Ports), kann der Einsatz bei Protokollen, welche Hash-Werte beziehungsweise Prüfsummen bilden, zu Problemen führen [Les06].

Gemeinhin gilt NAT nicht als Hochsicherheitslösung, wenngleich sie den Sicherheitsstandard erhöhen kann und wird vorwiegend zum Verbinden zweier Subnetzte in der Praxis eingesetzt [FG05]. Ein korrekter und sinnvoller Einsatz wird erst in Kombination mit einem *Application Level Gateway*[2] (kurz ALG) erzielt.

[2]System welches Pakete auf Applikationsebene untersucht und vorwiegend vor Angriffen basierend auf dem HTTP-Protokoll schützen soll.

6. (Paket-)filter mit iptables

Dieses Kapitel beschäftigt sich mit Paketfilterfunktionalität, welche mittels iptables implementiert werden kann und soll als Basis beziehungsweise Grundlage für die später vorgestellte Software (Firewall Builder siehe Kapitel 13) dienen und stellt ein Realisierungsbeispiel aus der Linux Umgebung dar.

6.1. Grundlagen von iptables

Grundidee Seit der Einführung des Linux-Kernels 2.4.0 wurden die (Paket-)Filterung, als auch NAT zu einem gemeinsamen Konzept zusammengefasst und können unter Linux mit dem Befehl *iptables* konfiguriert beziehungsweise erweitert werden [Les06]. Die Regeln haben direkten Einfluss darauf, wie Pakete basierend auf deren Headerinformationen zu behandeln sind.

Eine Zusammenfassung solcher Regeln zu Chains ermöglicht eine dezidierte Verwendung dieser bei Gebrauch (zum Beispiel bei der Weiterleitung von Paketen). Zusätzlich zu den selbstdefinierten Chains steht eine Vielzahl an vordefinierten zur Verfügung, welche aus eigens definierten angesprochen beziehungsweise aktiviert werden können.

Die aufgabenbasierte Trennung ermöglicht eine Strukturierung der Chains in unterschiedlichen *Tables* und kann somit als Zusammenfassung dieser verstanden werden. Beispiele hierfür sind etwa die NAT- und Filter-Tables.

Beim Eintreffen eines beliebigen Paketes wird zunächst ein Prerouting der NAT-Table durchgeführt, was zumeist zu einer Änderung der Zieladresse sowie des -ports verwendet wird und besondere Bedeutung für die Implementierung eines transparenten Proxies hat.

Danach wird die Routing Entscheidung getroffen, wobei zu unterscheiden gilt, ob ein Paket für den Rechner selbst oder für einen anderen (Weiterleitung) bestimmt ist. Ist

das Paket für den Rechner selbst bestimmt, so werden die Regeln der Input-Chain aus dem Table Filter verwendet und das Paket gegebenenfalls an den entsprechenden Prozess weitergeleitet.

Pakte für andere Rechner im nachfolgenden Netzwerk werden stattdessen anhand der Forward-Chain des Tables Filter untersucht und gegebenenfalls erlaubt oder verworfen.

Pakete ausgehend von lokalen Prozessen werden mittels der Output-Chain des Tables Filter abgeglichen und können sogleich mittels Output-NAT übersetzt werden, was jedoch selten zur Anwendung kommt.

Abschließend können alle ausgehenden Pakete mittels Postrouting-NAT abgeändert werden, wodurch zum Beispiel die Realisierung von Masquerading ermöglicht wird. Für das nachgelagerte Netzwerk scheinen somit alle Pakete von der Firewall zu stammen. Dies ist besonders wichtig, falls nur eine gültige beziehungsweise öffentliche IP-Adresse im Unternehmen zur Verfügung steht.

Der Vorgang wie Pakete beim Eintreffen behandelt werden, soll anhand der nachfolgenden Grafik erläutert werden:

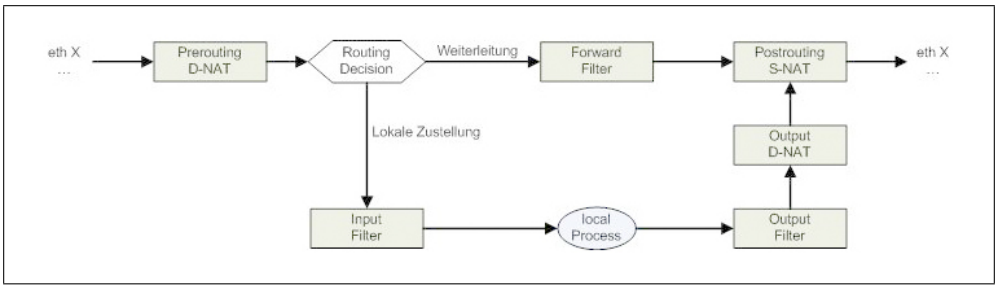

Abbildung 6.1.: NAT und Filterung von Paketen (Kernel 2.4) [vgl. Les06, S. 327]

Policies Als Policy bezeichnet man die Grundregel, welche einer Chain mitgegeben werden kann und Anwendung findet falls kein adäquater Eintrag gefunden wird.

Es existieren grundsätzlich zwei Policies:

ACCEPT Passieren des Paketes

DROP Verwerfen des Paketes

Wobei grundsätzlich dem Leitspruch zu folgen ist „Was nicht explizit erlaubt ist soll verboten sein!“ und dementsprechend der Einsatz der Accept-Policy nur selten sinnvoll ist.

Policies sind wie folgt aufgebaut:

```
iptables -P <Chain> <Policy>
```

Regeln beziehungsweise Rules Generell bestehen Regeln aus den zwei Komponenten, Muster und Aktion, wobei mit dem Muster definiert ist, auf welche Pakete eine Regel anzuwenden ist und die Aktion determiniert, was mit den musterkonformen Pakten zu geschehen hat.

Die Verwaltungsfunktionen sehen folgendermaßen aus:

```
iptables [-t Table] -option chain [pos] Muster Aktion
```

Wobei folgende Optionen zur Verfügung stehen [Les06]:

- **A** Die Regel wird am Ende der Chain angehängt. (Append)
- **I** Die Regel wird an einer bestimmten Position (pos) eingefügt. (Insert)
- **R** Ersetzen einer Regel an einer definierten Position (pos). (Replace)
- **D** Löschen einer Regel an einer bestimmten Position (pos) beziehungsweise eine genau determinierte Regel. (Delete)

Für eine detailierte Darstellung des Aufbaus von Mustern sowie Aktionen sei auf die nachfolgenden Unterpunkte verwiesen.

– Muster Da die Mehrzahl der Muster protokollspezifisch verfügbar ist, sollen hier nur einige wenige Standardmuster vorgestellt werden. Es sei darauf verwiesen, dass darüber hinaus noch ein Vielzahl anderer Muster zur Verfügung stehen. Für einen detailierten Einblick sei auf die Manpage zu den iptables verwiesen.

Folgende Tabelle soll einen Überblick über die Standardmuster geben, wobei mittels „*!*“ eine Negation dieses erwirkt wird:

Muster	Beschreibung
–p [!] Protokoll	Bezeichung des Protokolls (TCP, ICMP, ...)
–s [!] Adresse[/Maske]	Quelladresse mit Maske oder Bitanzahl
–d [!] Adresse[/Maske]	Zieladresse
–i [!] interface[+]	Empfangsinterface, wobei + als Wildcard verwendet werden kann.
–o [!] interface[+]	Sendeinterface
[!] –f	Folgefragmentspezifikation, da diese keinen Quell-, Zielport als auch ICMP-Typ enthalten

Tabelle 6.1.: iptables - Standardmuster [vgl. Les06, S. 329]

Zudem ist die Verwendung von Erweiterungen (mittels -m Erweiterung) möglich, wodurch zum Beispiel ein *stateful packet filtering* ermöglicht wird. Die entsprechende Erweiterung ist *state*, welche in der Basiskonfiguration vier Zustände kennt [Les06].

– Aktionen Trifft eines der zuvor spezifizierten Muster zu, so stößt dieses die Ausführung einer Aktion an, welche folgendermaßen definiert werden kann:

```
-j Target [Option]
```

Als Ziel (Target) können hierbei im einfachsten Fall die zuvor gezeigten Policies (Accept, Drop) zum Einsatz kommen. Darüber hinaus gibt es jedoch noch eine Vielzahl anderer Ziele, wovon nachfolgend die wichtigsten erläutert werden sollen.

Target	Beschreibung
RETURN	Paket wird an die übergeordnete Chain zurückgeleitet oder aber, wenn bereits Top-Chain erreicht ist, die Policy tritt in Kraft
LOG	Paket wird vermerkt, zum Beispiel im Systemprotokoll
REJECT	Verwerfen des Pakets und Rückmeldung des Fehlers (ICMP-Fehlermeldung)
SNAT	Paket wird mit veränderter Quelladresse weitergeleitet
DNAT	Paket wird mit veränderter Zieladresse weitergeleitet
MASQUERADE	Quelladresse des gegenwärtigen Pakets sowie der Nachfolgepakete der Verbindung werden geändert
REDIRECT	Selbe Funktionalität wie DNAT, jedoch mit geänderter Ziel–adresse (auf den Firewall-Rechner selbst - Proxyfunktionalität)

Tabelle 6.2.: iptables - Aktionen Targets [vgl. Les06, S. 330 - 332]

Viele der in der Tabelle erläuterten Targets, haben ein breites Spektrum an Parametern, welche jedoch aus Gründen der Übersichtlichkeit als auch Platzgründen ausgeblendet werden sollen.

Tables Tables stellen quasi die Zusammenfassung von Chains dar, wobei das Konzept neben den vordefinierten Tables auf Erweiterung abzielt und das Hinzufügen demnach ermöglicht.

6.2. Beispiele zu iptables

Nachfolgend sollen einige Beispiele zur Herstellung des praktischen Bezuges herangezogen werden [Les06]. Zu Beginn soll die oftmals in der Literatur gezeigte *absolut sichere Firewall* besprochen werden. Danach wird auf das *Loopback-Interface* sowie die *Vermeidung von Zugriffen auf lokale Serverdienste* eingegangen.

Absolut sichere Firewall Als absolut sichere Firewall kann eine derart reglementierte Firewall angesehen werden, welche jeden Netzwerkzugriff verbietet. Um dies zu realisieren müssen zuerst die bereits definierten Regeln sowie Chains gelöscht und in weiterer Folge durch neue restriktive ersetzt werden.

Folgende Eingaben realisieren eine solche absolut sichere Firewall:

```
# Löschen aller Regeln beziehungsweise Chains
iptables -F
iptables -X
iptables -t nat -F
iptables -t nat -X

# Verwendung von Policies zum Abweisen aller Pakete
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP
```

Die Table „nat“ bleibt dabei unverändert, könnte jedoch generell auf die ACCEPT-Policie gesetzt werden, da ja die Filterung ohnehin zuvor durchgeführt wird.

Loopback-Interface Um den Zugriff der Firewall auf Dienste, die auf dem Firewall-System laufen zu ermöglichen, ist es notwendig, ein spezielles Netzwerk-Interface (lo) zu definieren beziehungsweise zu reglementieren. Da dieses Interface ausschließlich für interne Zwecke Verwendung findet, spricht generell nichts dagegen, das lo-Interface in den INPUT sowie OUTPUT Chains freizuschalten.

```
# Lokale Paketvermittlung über das lo-Interface
```

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -i lo -j ACCEPT
```

Die Chain „forward“ wird nicht zur internen Vermittlung verwendet und kann daher unberücksichtigt bleiben.

Vermeidung von Zugriffen auf lokale Serverdienste Obwohl durch den Einsatz des „Stateful Packet Filtering“ beliebige unaufgeforderte Verbindungen prinzipiell abgewiesen werden, ist eine Betrachtung der verfügbaren Dienste dennoch sinnvoll, da Ports jenseits der üblichen Grenze von 1024 oftmals nicht gesondert reglementiert werden, um FTP-Zugriffe zu ermöglichen. Um spezielle Dienste, welche diesen Bereich dennoch nutzen, explizit von der Außenwelt abzuschotten ist eine Analysetätigkeit somit unumgänglich.

Auf der Suche nach den aktiven Internet-Verbindungen wird gerne auf das Kommandozeilentool *netstat [-options]* zurückgegriffen. Dieses Tool informiert uns, sofern richtig parametriert, über aktive Verbindungen sowie die dazugehörigen Ports und deren Zustände (zum Beispiel wartend).

Gehen wir nun davon aus, dass auf unserem Firewall-System zusätzlich noch ein Proxy-Server (squid) installiert ist, welcher standardmäßig auf dem Port 3128 TCP operiert und somit nicht reglementiert ist. Da jedoch ein externer Zugriff auf diesen nicht erwünscht ist, müssen wir den Zugriff auf den internen Gebrauch beschränken.

Dies können wir mittels folgenden Regeln realisieren:

```
# Vermeidung externer (aus dem Internet) Zugriff auf Server
# Proxyserver - squid

iptables -A ext-in -p tcp -dport 3128 -j DROP
```

Es erklärt sich von selbst, dass eine solche Reglementierung erst nach Installation der entsprechenden Serverkomponenten sinnvoll ist, da Updates sowie unterschiedliche Releases der Software verschiedene Ports verwenden können. Auch eine regelmäßige Kontrolle (zum Beispiel nach Updates) sei hier vorgeschlagen [Les06].

Aufgrund der Einfachheit der Verwendung, als auch der Erstellung von Regeln mittels iptables, gibt es neben der standardmäßigen Implementierung in Linux-Systemen, eine Vielzahl weiterer Lösungen, welche diese Technologie einsetzen. Zu diesen zählen etwa der *Packet Filter* (kurz pf) unter OpenBSD [Ope09a]. Nähere Informationen zu pf können unter der URL `http://www.openbsd.org/faq/pf/` eingesehen werden.

Teil II.

Virtuelle Maschinen

7. Virtuelle Maschinen - Grundlagen

Die nachfolgenden Kapitel sollen über die Grundlagen sowie die Funktionalität von Virtuellen Maschinen (nachfolgend kurz VMs) Aufschluss geben. Darüber hinaus werden die geschichtliche Entwicklung, als auch die zugrunde liegenden Techniken beschrieben. Weiters wird sowohl das Thema des Bridgings, als auch die damit verbundene Netzwerkanbindung von VMs beschrieben.

7.1. Aufgaben der Virtuellen Maschinen

Virtuelle Maschinen basieren wie der Name schon vermuten lässt auf dem Prinzip der *Virtualisierung*. Unter diesem Oberbegriff der Virtualisierung wird die Abstraktion von verfügbarer Hardware(-Ressourcen) hin zur virtuellen Maschine verstanden, wobei die Abstraktion eine Zuteilung einzelner Anteile der Hardware zu den VMs ermöglichen soll und so die Eigenschaften wirklich verfügbarer Maschinen beziehungsweise Computer replizieren soll [Tho08].

Zusätzlich zu der in den letzten Jahren stark zunehmenden Verbreitung, ist auch eine Erweiterung des Einsatzgebietes, ausgehend von den Rechenzentren hin bis zu Privatanwendern festzustellen. Konzeptionell haben beide Bereiche eine bereits vor Jahrzehnten beginnende Entwicklungsarbeit gemeinsam, welche sich in der heute verfügbaren Technologie widerspiegelt. So ist bekannt, dass IBM bereits in den 1960ern den Grundstein für diese Technik legte um die stark verteilten und zumeist unerschwinglichen Ressourcen effizient nutzen zu können [Fis08a].

Generell kann Virtualisierung in zwei entgegengesetzte Richtungen betrieben werden. Die in dieser Arbeit gegenständliche Ausprägung beschäftigt sich mit der Aufteilung vorhandener Ressourcen zur Schaffung unabhängiger logischer Systeme. Dem entgegen steht die Ausprägung der Vernetzung einzelner logischer Einheiten (zum Beispiel physikalische Rechner) zur Schaffung eines überdurchschnittlich performanten (virtuellen)

Systems welche in dieser Arbeit nur nebensächlich behandelt werden soll.

Folgende Grafik soll diesen Umstand verdeutlichen:

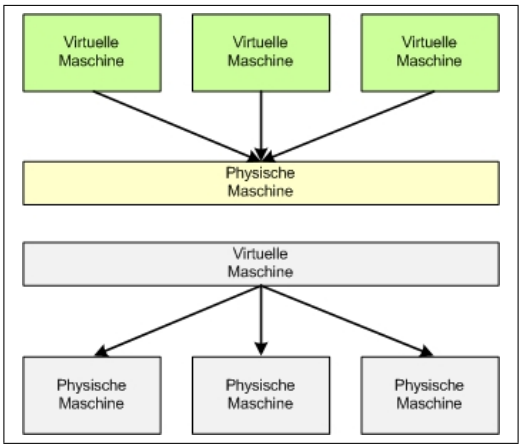

Abbildung 7.1.: Die unterschiedlichen VM-Konzepte [vgl. Tho08, S. 20]

Das Betriebssystem benötigt grundsätzlich zur Durchführung seiner Aufgaben eine Arbeitsumgebung, welche durch den Computer repräsentiert ist. Die obligatorischen Elemente dieser Arbeitsumgebung werden in der Literatur auch als *Core Four* bezeichnet [Tho08] und entsprechen der CPU, dem Massenspeicher, dem Arbeitsspeicher sowie dem Netzwerk-Interface. Durch Einsatz von Virtualisierungsverfahren kann die ursprüngliche Bindung zwischen Betriebssystem und Arbeits- sowie Hardware-Umgebung aufgehoben werden, wobei dies die Erstellung mehrerer virtueller Systemumgebungen, welche einer echten Systemumgebung gleich kommen, ermöglicht. Diese können in weiterer Folge parallel und isoliert voneinander existieren und können entweder innerhalb eines Betriebssystems oder auf einer Hardware-Komponente ausgeführt werden.

Das Ziel beziehungsweise die zentrale Aufgabe der Virtualisierung ist somit die Schaffung virtueller Systemumgebungen, welche einen eigenen Software- sowie Datenbestand besitzen und unabhängig voneinander agieren können [Tho08].

7.2. Geschichtliche Entwicklung

Die geschichtliche Entwicklung der Virtualisierung beginnt im Zeitalter der Großrechner, genau gesagt während der Zeit, als deren Verbreitung stetig zunahm. Die markantesten Merkmale dieser Großrechner waren ihre enormen räumlichen Dimensionen, als auch die kostspielige Anschaffung, welche eine Verwendung durch eine Vielzahl von Benutzern erforderte. Diese parallele Verwendung in den 1950er Jahren kann bereits als Grundstein beziehungsweise Konzept zur Virtualisierung angesehen werden und erwuchs verständlicherweise aus der Not, diese Systeme der „Öffentlichkeit“ zugänglich zu machen [Tho08].

Nachfolgend soll eine chronologische Übersicht der geschichtlichen Entwicklung der Virtualisierung, als auch der Methodiken gegeben werden. Als zeitlicher Horizont wird dabei die Entwicklung von den 1950er Jahren bis hin zu heutigen Anwendungen festgelegt.

7.2.1. Klassische Virtualisierung

Chronologisch kann die Entwicklung der klassischen Virtualisierung (1951-2003) folgendermaßen entlang des Zeitstrahls dargestellt werden, wobei auch die technologischen Hintergründe nicht vernachlässigt werden sollen [Fis08b]:

1951 - 1960 Dieser Zeit wird die Verbreitung der Großrechner vor allem im universitären Umfeld zugeschrieben, wobei die vorherrschende Meinung von nur zwei weltweit benötigten Großrechnern obsolet wurde. Die Verwendung solcher Systeme durch eine Vielzahl unterschiedlicher Personen ließ die Nachteile dieses Konstrukts offensichtlich werden. Diesem Umstand zu verdanken, veröffentlichte 1959 der damalige Professor der Informatik *Christopher Strachey* seine Abhandlung über die Technik der *Multiprogrammierung* mit dem Titel *Time Sharing in large fast computers*. Diese Veröffentlichung wird nachträglich als die Geburtsstunde der VMs angesehen, da die darin veröffentlichten Grundlagen zum erstenmal Thematiken wie *Time Sharing* und *Multiprogramming* näher spezifizierten.

1961 - 1970 In den 60er Jahren wurde die Entwicklung der Großrechner stetig vorangetrieben, wobei der damalige Marktführer (IBM) beträchtlichem Druck von Seiten der Konkurrenz ausgesetzt war. Mit dem Erscheinen des *Atlas - Systems* bewies die kleinerer Konkurrenz eindeutig ihre Flexibilität, als auch die technische Fertigkeit System „höchster“ Geschwindigkeiten erstellen zu können.

Basierend auf der Publikation von Christopher Strachey wurden erstmals von allen Anbietern Geräte angeboten, welche Time Sharing unterstützen und somit simple Multitasking-Operationen ermöglichten. Systeme aus diesem Zeitalter sind etwa das Grorechnermodell 360-67 als auch das VM/370 von IBM, welche rudimentäre Unterstützung für die Virtualisierung boten.

Auch von Seiten der Betriebssystementwicklung ist dieser Zeit besonderes Augenmerk zu schenken, da es zur erstmaligen Veröffentlichung eines systemübergreifenden Betriebssystems (OS/360) von IBM kam. Auch die Entwicklung am UNIX-System begann zum Ende dieses Zeitraums.

1971 - 1980 Während der 1970er Jahre begann UNIX seinen Siegeszug und somit die Verdrängung anderer Betriebssysteme auf Großrechnerbasis. Dennoch wurde diese Zeit auch durch das Aufkommen der sogenannten Mikrocomputer geprägt, welche in weiterer Folge den Durchbruch im Privatbereich (PC, Personal Computer) einläuteten. Die Technologieführerschaft des UNIX - Systems wurde zumeist durch eine unattraktive Bedienbarkeit erkauft, was zum Aufkommen benutzerfreundlicherer Systeme, wie zum Beispiel aus dem Hause Microsoft führte. Zeitlich ist auch die Erscheinung der 16 Bit-Prozessoren und somit der x86-Architektur sowie des ersten GUI (Graphical User Interface) an dieser Stelle zu erwähnen.

Die absehbare Verbreitung der Mikrocomputer führte zu einer Konzentration der Entwicklungsarbeit auf diesem Gebiet, weshalb keine herausragenden Ereignisse in Hinblick auf die Virtualisierung in der Literatur genannt sind.

1981 - 1990 Maßgeblich für die Zeit sind die rasante Verbreitung der PCs (mit x86-Architektur von Intel) als auch der Aufstieg der Software-Firma *Microsoft*.

Aber auch die Entwicklungen in Richtung der Virtualisierung wurden wieder vorangetrieben. So wurde 1985 die Entwicklung des *Mach Microkernels* für paralleles und

virtualisiertes Computing, eingeleitet. Im selben Jahr wurde auch das 32 Bit Virtualisierungssystem VM/XA von IBM veröffentlicht.

Im Jahr 1990 veröffentlichte IBM ein Großrechnersystem, welches den neuen Maschinenbefehl *SIE* (Start Interpretive Execution) unterstützt und somit zu einer erheblichen Beschleunigung der Emulation führte. Weiters kam die LPAR[1]-Technologie auf den Markt, welche bis in die heutige Zeit Verwendung findet.

1991 - 2000 Die 1990er Jahre waren der Weiterentwicklung der x86-Struktur gewidmet und führten somit zu enormen Leistungssteigerungen, welche auch in den privaten Sektor durchschlugen. Die mittlerweile erzielte Monopolstellung der Microsoft Produkte, sowohl als OS als auch im Bereich der Bürosoftware, motivierte und beflügelte die Linux - Entwicklungsgemeinschaft zur Erstellung ihres freien Betriebssystems.

Bezüglich der Virtualisierungsthematik machte ein Unternehmen mit seinen Produkten besonders auf sich aufmerksam. Es handelte sich um das Unternehmen VMware, welches sich den Linux - Kernel - Patch (UML = User Mode Linux) zur Ausführung eines zweiten Systems zunutze machte und somit schnell zum Technologieführer aufstieg.

2001 - 2003 Im Jahr 2001 wird UML in den offiziellen Linux-Kernel aufgenommen und dient vorwiegend dem Testen von neuen Kernel-Versionen. Firmen wie SWsoft veröffentlichen kostenpflichtige Virtualisierungslösungen, welche jedoch in weiterer Folge auch als OpenSource-Derivate verfügbar werden. Ebenso wurde an der Universität von Cambridge das XenoServer-Projekt initiiert, welches 2003 zum eigenständigen Projekt beziehungsweise Lösung XEN wird und fortan weite Verbreitung findet.

Darüber hinaus wurde eine Vielzahl anderer Virtualisierungslösungen geplant und veröffentlicht, welche aufgrund der vereinfachten Bedienung schnell Anklang finden.

7.2.2. Moderne Virtualisierung

Obwohl die Grenzen zwischen der klassischen sowie der modernen Virtualisierung keinesfalls scharfkantig anzusehen sind, so wird diese, in dieser Arbeit durch den Übergang

[1] Logical Partitioning - Unterteilung der verfügbaren Rechner-Ressourcen zum Zwecke der Virtualisierung

von der Architektur-Emulation zum direkten Zugriff auf die virtualisierte Hardware verstanden. Dies führt im Generellen zu Performancegewinnen, welche durch das Wegfallen der Befehls-Übersetzungen (für den Hardware-Zugriff) erklärt werden können.

Beide Techniken, sowohl Emulation als auch Interpretation, verwenden die verfügbare Hardware beziehungsweise Ressourcen jedoch auf unterschiedliche Art und Weise. Wohingegen es bei der Emulation möglich ist, eine völlig, ausgehend von der verfügbaren Hardware, unterschiedliche Konfiguration zu präsentieren beziehungsweise verfügbar zu machen. So greift die Virtualisierung mittels Interpretation direkt auf die virtuell zur Verfügung gestellten Komponenten zu. Dies führt bei adäquatem Einsatz zu einem Entfallen der Übersetzung der Befehle, woraus sich die Leistungsfähigkeit enorm steigern lässt und allgemeinhin von moderner Virtualisierung gesprochen wird [Xen08].

Der Grundstein zur Anwendung moderner Virtualisierungslösungen wurde durch das bereits zuvor beschriebene UML gesetzt und wird derzeit auch auf Linux-fremden Systemen erfolgreich angewendet. Die Nennung zu diesem Zeitpunkt entfällt dennoch aus chronologisch bedingten Gründen.

7.3. Einsatzbereiche und -gründe

Die Einsatzbereiche von VMs haben sich abgesehen von den geschichtlich abgeleiteten stark diversifiziert und unterliegen unterschiedlichen Prämissen und Motivationen. Von simplen Test-Betriebssystem-Installationen auf Heim PCs über System-Konsolidierung bis hin zu Green-IT gibt es eine Reihe von Anwendungsfällen die nachfolgend beschrieben werden sollen. Aufgrund des Potentials der vorgestellten Technologie ist es nicht auszuschließen, dass neben den beschriebenen Use Cases auch noch weitere sinnvolle Anwendungsgebiete vorhanden sind beziehungsweise gefunden werden können.

Folgende Aufzählung behandelt die gebräuchlichsten Anwendungsfälle [Tho08, Fis08b]:

- **Schulungen**
  - Flexible und schnelle Bereitstellung von Schulungssytemen
  - Physische Rechner benötigen keine Neuinstallation, sondern dienen als Terminals für das virtuelle System
  - Parallelinstallationen möglich

- **Virtuelle Desktops**
  - Zentrale Verwaltung der Software
  - Beschleunigte Systembereitstellung und -Recovery
  - Zugriff auch von externen Standorten möglich
  - Datenschutz-Policy kann wesentlich einfacher umgesetzt werden
- **Anwendungsintegration**
  - Anwendungen verschiedener Plattformen beziehungsweise Systeme können integriert werden
  - Portierbarkeit von Legacy-Anwendungen (zum Beispiel bei Systemupgrades)
- **Softwareentwicklung/-testing und SOA**
  - Fremde und vormals nicht verfügbare Plattformen können getestet werden
  - Unterschiedliche Plattformen können getestet werden
  - Testumgebungen können repliziert werden
  - Virtuelle Geräte können getestet werden (Emulation von Ressourcen)
  - Hardwarefehleremulation ist möglich
  - Automatisierung der Tests gestaltet sich komfortabler
- **Support und Wartung**
  - Nachvollziehbarkeit von Fehlern unter Realbedingungen
  - Replikation von Produktivsystemen zu Analysezwecken möglich
  - Kundenkonfigurationen können simuliert und fehlerbereinigt werden
- **Systemkonsolidierung**
  - Nutzung von Synergie-Effekten
  - Effizienzsteigerung sowie Weiterführung von Legacy-Systemen möglich
  - Gesamtkostenreduktion bei gleichbleibender Leistung möglich

- **Administrationsdomänen, vertikale/operationale Isolation**
  - Dezidierte Umgebungen für diverse Administratoren
  - Datenschutz zwischen den Parteien
- **Sicherheitsdomänen, horizontale/funktionale Isolation**
  - Isolation der unterschiedlichen Dienste möglich
  - Entkoppelung von Wechselwirkungen und unbeabsichtigter Beeinflussung
  - Einschränkung der Auswirkungen einer Sicherheislücke
- **Hochverfügbarkeit und Service-Level**
  - Wartung der Hardwarekomponenten und damit verbundene Ausfälle können vermieden werden
  - Desaster-Recovery durch Portierbarkeit erleichtert
  - Ressourcenbindung im Standby-Betrieb reduzierbar
  - Dynamische Verteilung der Ressourcen kann bei Spitzenwerten zur Einhaltung eines bestimmten Quality of Service führen
- **Green-IT**
  - Höhere Energieeffizienz, da erhöhter Wirkungsgrad, bedingt durch die Reduktion von Stillstands-Zeiten
  - Anteilsmäßig reduzierter Leistungsverbrauch für ein und die selbe Leistung

Den zuvor genannten Einsatzgebieten und der damit verbundenen Vorteile stehen natürlich auch Nachteile gegenüber, welche teils sehr spezifisch sind, andere hingegen betreffen jedweden Einsatz von Virtualisierungstechniken.

Diese Nachteile sind unter anderen zum Beispiel [Fis08b]:

- Systeme, welche eine hohe hardwarebedingte Ein-/Ausgabe Leistung erfordern, laufen aufgrund der Aufteilung dieser Ressourcen langsamer als explizite Realisierungen

- Der Ausfall eines Servers zur Virtualisierung kann zum Ausfall mehrerer Systeme fürhen (*SPoF* = Single Point of Failure)
- Hotplug-Hardware sollte Verwendung finden, da sonst unter Umständen, alle Instanzen neugestartet werden müssen
- Zunehmende Komplexität von Sicherheitsaspekten, wo vormals die Herstellung einer galvanischen Verbindung notwendig war, entscheiden nun Programmparameter über Verbindung oder Trennung von (Teil-)Systemen

7.4. Virtualisierungstechniken

Im nachfolgenden Abschnitt sollen die grundlegenden Virtualisierungstechniken vorgestellt und beschrieben werden. Zusätzlich sollen die mit den konkreten Techniken verbundenen Vor- sowie Nachteile aufgezeigt und besprochen werden. Desweiteren sollen die Verwendungsgebiete der einzelnen Methoden in der Praxis beleuchtet werden.

Die Einteilung der Virtualisierungstechniken basiert dabei auf der Kategorisierung [Tho08]:

- **Echte Virtualisierung:** Im Host-System vorhandene Hardware wird auf die vorhandenen VMs aufgeteilt. Als Beispiele seien hier die Hardware- sowie Paravirtualisierung genannt.
- **Emulatoren:** Ein dezidierter Rechner wird mittels Software vorgetäuscht, wobei dessen Eigenschaften beliebig abänderbar sind. Als konkrete Implementierungen seien hier der *Hercules*-[2] sowie *Qemu-Emulator*[3] genannt [QEM08].
- **API-Emulatoren:** Sie stellen einen Spezialfall der Virtualisierung dar, da sie nicht ein komplettes System emulieren sondern nur die Schnittstellen bereitstellen, um Software im Kontext lauffähig zu machen. Bekannte Implementierungen dieser Gattung sind zum Beispiel *Wine*[4], *Qemu Usermode* sowie *Cygwin*[5] [Fis08a].

Sowohl Punkt zwei als auch Punkt drei stellen nicht den Gegenstand dieser Arbeit dar und werden deswegen nachfolgend nicht oder nur bedingt detailiert behandelt.

[2] Hercules: Open Source Software Implementierung für Mainframe Systeme. URL: `http://www.hercules-390.org/`
[3] Qemu: PC System Emulator. URL: `http://www.nongnu.org/qemu/`
[4] Wine: API-Emulator für Windows Anwendungen. URL: `http://www.winehq.org/`
[5] Cygwin: API-Emulator für Windows Anwendungen. URL: `http://www.cygwin.com/`

7.4.1. Hardware-Virtualisierung

Die *Hardware-Virtualisierung*, wie sie hier beschrieben wird, stellt im Desktop-Bereich eine der neueren Entwicklungen dar. Sie basiert auf der Fähigkeit der Prozessoren (sowohl Intel als auch AMD), Virtualisierungsfunktionen erkennen und interpretieren (beziehungsweise weitergeben an den Virtual Machine Monitor) zu können und soll demzufolge den Overhead sowie die Komplexität der vormals notwendigen *Binary-Translation* entgegen wirken [Tho08]. Als Kernelement ist das Erkennen von kritischen Operationen für den *Hypervisor*[6] und das Informieren des *Virtual Machine Monitor*[7] (kurz VMM) zu nennen.

Als Resultat dieser hardwareseitigen Erweiterung kann jedes beliebige Betriebssystem ohne weitere Modifikation als VM betrieben werden, da die Interaktionen dieser mit dem Hypervisor durch den Prozessor selbst gewährleistet werden.

Die (Hardware-)Erweiterungen selbst sind einerseits die Abänderung des Privilegiensystems um ausreichend Platz für den VMM bereit zu stellen und andererseits die Integration von Mechanismen zum Erkennen von virtualisierungsspezifischen Operationen.

Als Vorteile dieser Virtualisierungstechnik sind der stark vereinfachte Aufbau der VMM sowie der damit verbundende Wegfall der Binary Translation anzusehen. Weiters stellt AMD eine Möglichkeit zur Abbildung des virtuellen Speichers auf den tatsächlichen (mittels zweistufiger Page-Tables) bereit, was in weiterer Folge zu einem reduzierten Aufwand bei der Verwaltung dieses Speichers führt. Dadurch ergeben sich im Wesentlichen die Vorteile der höheren Perfomance als auch der Stabilität.

Als Nachteil kann die fehlende Kompatibilität der beiden Realisierungen genannt werden, als auch das Fehlen der Implementierung der zweistufigen Page-Tables von Seiten Intels. Weiters ist eine Portierbarkeit der VMs nicht ohneweiteres gewährleistet und erfordert eine zeitaufwändige Anpassung dieser.

[6] Der *Hypervisor* verteilt beziehungsweise bildet die Elemente (Prozesse, I/O - Operationen, und ähnliches) aller virtueller Systeme auf die vorhandene Hardware ab und regelt dementsprechend auch den Zugriff auf diese. Des Weiteren ist der Hypervisor dafür verantwortlich, grenzwertige Befehle abzufangen und gegebenenfalls abzuändern um einen reibungslosen Betrieb der VMs sicherzustellen. Er kann daher als ein der Hardware übergeordnetes minimalistisches Betriebssystem angesehen werden.

[7] Dient als Synonym für den Hypervisor.

7.4.2. Para-Visualisierung

Entgegen der Hardware-basierten-Virtualisierung ist bei der Para-Virtualisierung der Umstand festzustellen, dass nicht die vorhandene Hardware, sondern ein Software-System zur Regelung des Zugriffs auf den VMM beziehungsweise den Hypervisor verantwortlich ist. Die bedingt jedoch ein Gastsystem, welches von sich aus einen Interrupt auslöst, wenn auf den Hypervisor (mittels kritischen Befehlen) zugegriffen werden soll. Diese Erweiterung des VM-Befehlssatzes wird allgemein als Hypervisor-Call bezeichnet.

Der wesentliche Vorteil der Para-Virtualisierung ist daher in dem passiven Status des Hypervisors begründet, der erst in Aktion tritt, wenn dieser gezielt aufgerufen wird. Dieser Wegfall zahlreicher Exceptions, welche abgefangen und bearbeitet werden müssten, führt zu der Tatsache, dass paravirtualisierte Systeme generell leistungsstärker sind als jene, welche vollständig virtualisiert (zum Beispiel mittels Emulation) sind [Tho08]. Auch die Portierbarkeit der eingesetzten VMs auf eine andere Hardware-Konstellation (zum Beispiel bedingt durch Server-Crash) ohne intensive Anpassung, stellt einen wesentlichen Vorteil dieser Technologie dar.

Als schwerwiegendster Nachteil dieser höchst effizienten Technologie ist die notwendige Modifikation des Gastsystems anzusehen, da diese zumeist nur für quelloffene Systeme (wie zum Beispiel Linux) mittels Patches zur Verfügung steht [Fis08a].

Zu den bekanntesten beziehungsweise populärsten Lösungen dieser Sparte zählen Xen, welches in Abschnitt 8.1 auf Seite 89 genauer betrachtet wird, als auch diverse VMware Produkte [Tho08].

7.4.3. Virtualisierung auf Kernel-Ebene

Unter *Virtualisierung auf Kernel-Ebene* beziehungsweise *Kernel-basierte-Virtualisierung* wird die Schaffung einer Linux-Kernel-Infrastruktur zu Zwecken der Virtualisierung verstanden [Fis08a].

Der expliziten Nennung liegt die Verwendung beziehungsweise Kombination der zuvor genannten Virtualisierungstechniken zu Grunde. Die Kernel-basierte-Virtualisierung nutzt demnach sowohl hardwarebasierte als auch software-emulierte Elemente zur Gewährleistung der Funktionalität.

So stellt die Erweiterung des Kernels einen leistungsstarken Hypervisor für die rechnereigene CPU zur Verfügung, wobei die anderen I/O-Gerätschaften mittels Emulation (zumeist Qemu) an die VM angebunden werden.

Dies bedingt, durch die Reduktion der durchzuführenden Aufgaben des Hypervisors, eine Leistungssteigerung der virtualisierten Umgebungen im Vergleich zur herkömmlichen Para-Virtualisierung. Gleichsam bedingt dies jedoch auch die Notwendigkeit zur Verfügbarkeit der CPU-Erweiterungen Vanderpool beziehungsweise Pacifica [Tho08].

Geschichtlich gesehen ist die Kernel-basierte-Virtualisierung eine Erweiterung beziehungsweise Weiterentwicklung der Para-Virtualisierung und gewann vor allem durch KVM, welches in Abschnitt 8.2 auf Seite 91 genauer betrachtet wird, an zunehmender Bedeutung.

Praxisbedingte Vorteile bringt der Einsatz vor allem durch die gesteigerte Leistungsfähigkeit, als auch durch die Verfügbarkeit von stabilen Lösungen.

Als Nachteile sind die Betriebssystem-spezifische Verfügbarkeit, als auch die Mehrzahl an Software-Komponenten, welche für den Einsatz benötigt werden, zu nennen.

Folgende Grafik soll der Verständlichkeit dienen:

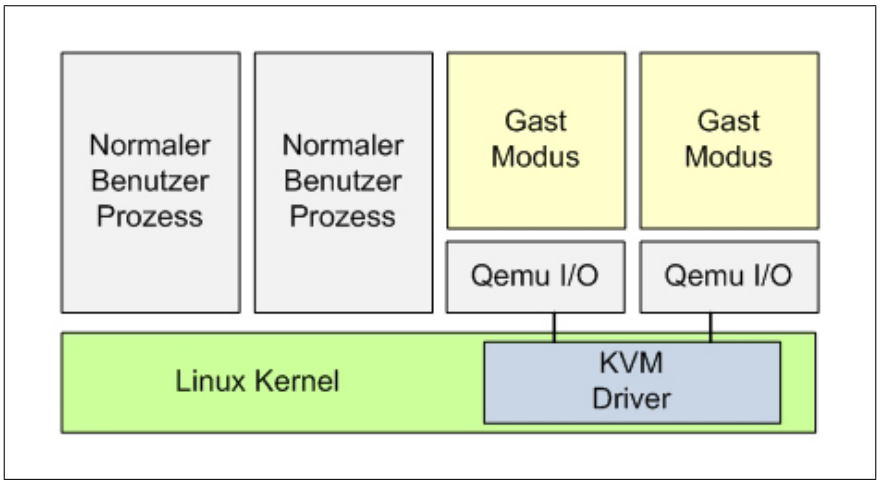

Abbildung 7.2.: Kernel basierte Virtualisierung [Fis08a]

7.4.4. Exkurs: OS-API-Emulation

Unter *OS*[8]*-API*[9]*-Emulation* beziehungsweise *API-Emulation* wird die Bereitstellung einer Betriebssystem-Laufzeitumgebung zu Zwecken der Virtualisierung beziehungsweise Portierung von Anwendungsprogrammen verstanden [Tho08].

Die Gründe einer solchen Virtualisierung sind vielfältig. So kann es zum Beispiel die Umstellung beziehungsweise das Upgrade eines Betriebssystems notwendig machen, die vormaligen Bedingungen weiterhin bereit zu stellen, um Legacy-Software lauffähig zu halten. Weiters kann der Betriebssystemumstieg zu einer Notwendigkeit dieser Emulation führen, wenn zum Beispiel ein Windows-Programm auch unter Linux ausführbar sein soll beziehungsweise muss.

Als *API* (Application Programming Interface) bezeichnet man die Schnittstelle zur Anwendungsprogrammierung. Der Umstand, dass diese API, je Betriebssystem unterschiedlich implementiert ist, führt zu dem Problem der Nicht-Ausführbarkeit von Programmen unter dem besagten Systemen (selbst wenn die Hardware dies zulassen würde).

Die Emulation einer solchen API führt zur gewohnten Bereitstellung der Schnittstelle zwischen Anwenderprogramm und Betriebssystem, wobei eine Umwandlung in die zugrunde liegende (systemspezifische) API notwendig wird.

Um den Einsatz solcher OS-API-Emulatoren zu gewährleisten, muss generell die zugrundeliegende Hardware zur Ausführung des portierten Anwenderprogramms ausgelegt sein. Da dies bei den meisten x86-Systemen gewährleistet ist, entspricht dieser Einsatzzweck dem pragmatisch häufigsten.

Als Vorteile dieser Technik zur Virtualisierung sind die Einfachheit der Installation sowie der Wartung als auch die hohe Verfügbarkeit der Windows-API-Emulatoren zu nennen (zum Beispiel Wine). Ebenso kann es zu Einsparungen im Lizenzbereich kommen, da das emulierte OS nicht erworben werden muss um Anwendungen dieser Plattform ausführen zu können [Tho08].

Folgende Grafik soll diesen Umstand verdeutlichen:

[8] OS = Operating System = Betriebssystem
[9] API = Application Programming Interface = Schnittstelle zur Anwendungsprogrammierung

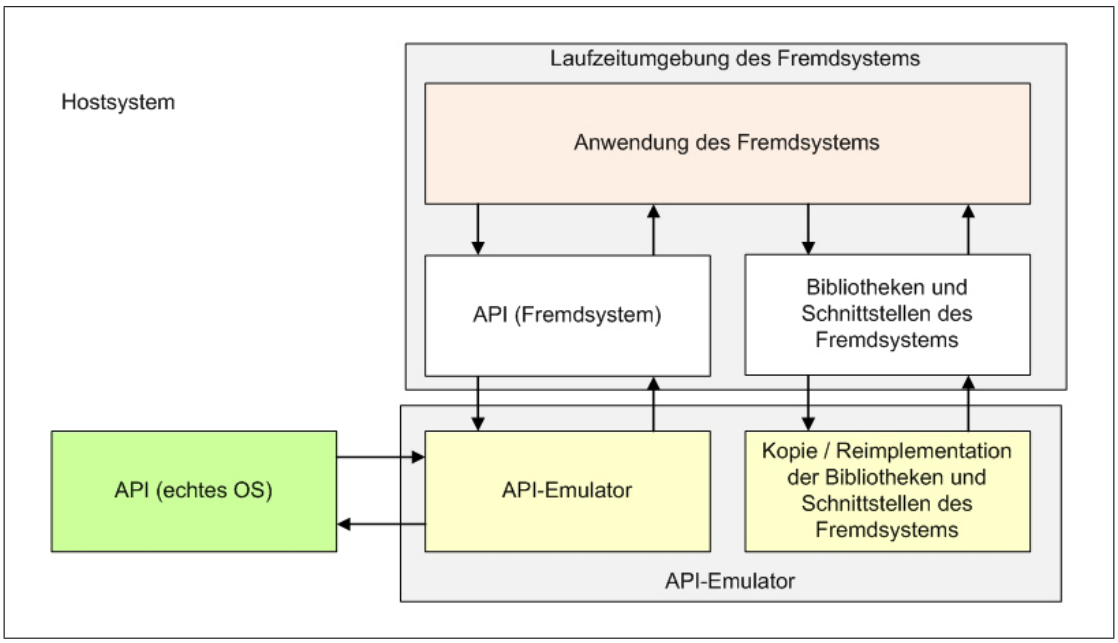

Abbildung 7.3.: API-Emulation zur Abbildung der Schnittstellen und Bibliotheken eines OS [vgl. Tho08, S. 37]

7.5. Netzwerkanbindung von Virtuellen Maschinen

Bridging Als *Bridging* (in Kontext zu VMs) wird die Schaffung einer Brücke zur gemeinsamen Nutzung einer physikalischen Netzwerkschnittstelle verstanden, welche auf dem Host- beziehungsweise Wirt-System einzurichten ist [Tan02]. Nach erfolgter Einrichtung kann diese in der VM initialisiert und verwendet werden.

Das Bridging ermöglicht somit die Netzwerkkommunikation der einzelnen VMs in gebündelter Art und Weise (als virtuelle Schnittstelle) auf der vorhandenen Hardware und kann mit starken Restriktionen (beziehungsweise bei Nicht-Einrichtung) diese von der Außenwelt abschotten.

Diese virtuelle Schnittstelle wird von der zugewiesenen VM wie eine echte physikalische Schnittstelle angesprochen und präsentiert sich auch nach außen hin (restliches Netzwerk) als eigener Host.

Unter einem Linux-Betriebssystem (wie zum Beispiel Ubuntu) ist die Erstellung ei-

ner Brücke einfach durch das Editieren beziehungsweise Anpassen der *interfaces*-Datei (Pfad: /etc/network/interfaces) folgendermaßen möglich [Fis08a]:

```
auto br0
iface br0 inet static
        address 192.168.0.10
        network 192.168.0.0
        netmask 255.255.255.0
        broadcast 192.168.0.255
        gateway 192.168.0.1
        bridge_ports eth0
        bridge_fd 9
        bridge_hello 2
        bridge_maxage 12
        bridge_stp off
```

Um diese bereitgestellte Bridge im Gast-System zu verwenden beziehungsweise einzurichten, stehen mehrere, teils unabhängige Möglichkeiten zur Verfügung, welche aufgrund des eingesetzten Virtualisierungsprodukts unterschiedlich sein können [Fis08a]. Die Möglichkeiten reichen von der Parametrierung der Installationsroutine bis hin zur grafisch unterstützen Drop-Down-Auswahl im Virtualisierungsprodukt selbst und werden aufgrund der Vielzahl dieser, nicht näher erläutert.

VLANs Als *VLANs* (Virtuelle LANs) bezeichnen wir die Schaffung virtueller Netzwerke, welche sich zur Vernetzung von VMs einsetzen lassen. Der prinzipielle Aufbau unterscheidet sich im Wesentlichen nicht von normalen LANs bis auf die Tatsache, dass VLANs über LAN-Grenzen hinweg existieren können und einen restriktiven Einsatz zur ausschließlichen Vernetzung unserer VMs ermöglichen [Fis08a].

Näheres zum Thema der Vernetzung von VMs wird im Abschnitt 11 auf Seite 119 erläutert.

8. Virtualisierungslösungen

Im nachfolgenden Kapitel werden konkrete Virtualisierungslösungen vorgestellt. Bei der Auswahl wurde auf die Aktualität, als auch auf die Verbreitung Rücksicht genommen. Neben einer Beschreibung der geschichtlichen Wurzeln sollen vor allem auch die Grundlagen, als auch die eingesetzte Technologie erläutert werden. Ebenso soll ein Überblick über die damit verbundene Einsatzgebiete gegeben werden. Weiters soll zu Übersichtszwecken (im Exkurs) auf die Hardwarevirtualisierungsbefehle eingegangen sowie mögliche Schwachstellen besprochen werden.

8.1. Xen

Beginnend mit der Namensgebung lässt sich feststellen, dass der Begriff Xen ursprünglich mit größter Wahrscheinlichkeit aus der Kombination von Linu**x** und **Z**en gebildet wurde [Fis08b]. Wenig verwunderlich ist die Komponente Linux, welche als Entwicklungsplattform für die Virtualisierungstechnologie Verwendung findet. Zen hingegen ist ein Derivat des Buddhismus, welches einem Schüler ermöglichen soll, die Erkenntnis der absoluten Realität zu erfahren. Zusammengeführt als Xen, soll so eine virtuelle (Beinahe-)Realität für die Virtualisierung geschaffen werden. Dieser Vorsatz wird durch den geringen Overhead, in Hinblick auf die Performance, von etwa 0,1% bis 3,5% (lt. Herstellerangaben) Folge geleistet [Xen08].

8.1.1. Einsatzbereiche

Xen als Open Source Virtual Machine Monitor (VMM) auf Basis der Para-Virtualisierung kann für folgende Zwecke eingesetzt werden. Da die Einsatzbereiche bereits in Abschnitt 7.3 auf Seite 79 beschrieben wurden, wird auf eine Erläuterung an dieser Stelle verzichtet.

Folgende Einsatzbereiche werden offiziell angeführt [Xen08]:

- Server Konsolidierung
- Hardware Abstraktion
- Kernel Entwicklung
- Multiple Betriebssystem Konfiguration zu Testzwecken
- Systememulation zur OS-Entwicklung
- Cluster Computing

8.1.2. Geschichte

Geschichtlich gesehen ist die Geburtsstätte von Xen die Universität von Cambridge, wo dieses 2001 im Computer-Labor erstmals erdacht beziehungsweise angedacht wurde. Damals, Teil des XenoServer-Projekts, wurde es 2003 unter der Leitung von Ian Pratt ausgegliedert und selbstständig weiterentwickelt.

Im Oktober 2003 wurde Xen das erste Mal auf dem Symposium on Operating Systems (SOSP) in einem Artikel mit dem Titel „Xen and the Art of Virtualization“ vorgestellt und ist seither zu einem der erfolgreichsten Open Source Projekte geworden. Zur gleichen Zeit wurde auch das erste Release zum Download bereitgestellt und überzeugte seither vorwiegend durch seine außerordentliche Stabilität. Davon zeugt auch die eigens gegründete Firma *XenSource*, welche kommerzielle Abwandlungen sowie Support für diese anbietet und vertreibt, und das virtuelle Zuhause des Projektes darstellt. Im Jahr 2007 wurde XenSource und die damit verbundenen XenServer-Produkte durch die Firma Citrix übernommen.

Neben diesen kommerziellen Ablegern existieren jedoch auch die frei verfügbaren Xen-Pakete, welche Bestandteil vieler Repositories diverser (Linux-)Distributionen sind und so zur rigorosen Verbreitung beitragen.

Obwohl die feste Integration in den Linux-Kernel derweilen noch nicht geschafft ist, so wird dies mit der zukünftigen Version 3.4 angestrebt. Dieser Aspekt ist vor allem in Hinsicht auf die Verbreitung dieser Software zu berücksichtigen, da ähnliche Lösungen, wie zum Beispiel KVM, bereits diesen Schritt überwunden haben.

Gegenwärtig ist die Version 3.3 zum Download auf `http://www.xen.org/` frei verfügbar.

Aufgrund der regen Popularität von Xen zählen eine Vielzahl an Firmen, wie zum Beispiel Intel, IBM, AMD und Novell, zu den Unterstützern beziehungsweise Sponsoren. Eine Verwendung der Basisdistribution in anderen Projekten (zum Beispiel VirtualIron) wird vielfach durch die Konzentration auf die Funktionalität und der damit verbundenen mangelnden Management-Funktionen bedingt [Tho08].

Diese Schwächen sowie die Schaffung einer GUI für diese Zwecke sind bereit seit Version 3.x aktiver Bestandteil der Roadmap [Fis08b].

8.1.3. Technik

Obwohl Xen´s Wurzel in der Para-Virtualisierung liegen, wurde bedingt durch die hardwareseitige Virtualierungsunterstützung der Prozessoren, eine Erweiterung zur Nutzung dieser Technologie in das Projekt integriert.

Wie bereits zu einem früheren Zeitpunkt erwähnt, bedingt die Verwendung der Para-Virtualisierung die Integration einer Zwischenschicht, dem Hypervisor, welcher aufgrund der fehlenden Aufnahme in den Standard-Linux-Kernel derweilen noch per Hand, unter Berücksichtigung der Version mittels Patch eingespielt werden kann.

Bei der hardwarebasierten Virtualisierung baut Xen vor allem auf die Reduktion der Funktionalität des Hypervisors was vor allem durch den Wegfall der CPU-Adressierung zu einer besseren Leistungsfähigkeit dieser führt [Fis08b]. Die restliche Hardware bedingt dennoch einer Emulation wobei zu diesem Zwecke aus dem Qemu-Projekt die entsprechenden Teile entnommen wurden beziehungsweise werden.

8.2. KVM

KVM als Abkürzung für *Kernel-based Virtual Machine* ist eine auf den Linux-Kernel basierende Virtualisierungslösung, welche vorwiegend auf die Funktionalität der Hardware-Virtualisierung zurückgreift. Die restliche Peripherie wird mittels Emulation an die VMs bereitgestellt, wobei KVM hier auf das „reife“ Qemu-Projekt zurückgereift [KVM09].

Ähnlich wie Xen, erfreut es sich zunehmender Beliebtheit, hat jedoch im Unterschied zu

diesem bereits den Einzug in den Linux-Kernel geschafft und steht somit dem breiten Publikum in diversen Standard-Distributionen zur Verfügung.

Das Zusammenspiel von Hardware-Virtualisierungsmethodiken mit der Emulation der restlichen Hardware-Komponenten ermöglicht den Einsatz eines breiten Spektrums an Gastsystemen [Tho08].

Im Effektivbetrieb lässt sich dabei, abhängig von der Anwendung, von einem effektiven Overhead von 2,5 bis 5% ausgehen[1] [Pho08].

8.2.1. Einsatzbereiche

Aufgrund der Vielzahl an unterstützten Gast-Betriebssystemen erscheint eine Limitierung der Einsatzgebiete als wenig sinnvoll, da davon ausgegangen werden kann, dass alle Aufgaben, welche auf der Emulation eines PC-Systems beziehungsweise dessen Infrastruktur basieren, wahrgenommen werden können.

Dies mag auch der Grund sein, warum die offizielle Dokumentation keinerlei Anwendungsfälle beziehungsweise Einsatzbereiche gesondert listet oder beschreibt [KVM09]. Als Restriktionen seien dennoch die Verfügbarkeit von Linux-Drivern für div. Hardware-Komponenten sowie die in Arbeit befindliche Para-Virtualisierung dieser zu nennen.

8.2.2. Geschichte

Ursprünglich wurde und wird KVM von dem israelischen Unternehmen *Qumranet* entwickelt, welches jedoch im Jahr 2008 von Red Hat[2] übernommen wurde.

Bereits im Oktober 2006 wurde ein Patch für den Linux-Kernel per Mailingliste angekündigt, welche die Nutzung und somit die Implementierung der Schnittstelle zu KVM, unter der Voraussetzung der virtualisierungsfähigen Prozessoren ermöglichte. Dieser Umstand ist umso interessanter, zumal zu bedenken ist, dass KVM auch 2006 der Öffentlichkeit

[1] Der Benchmark wurde unter Ubuntu 8.04 (mit Linux Kernel 2.6.24) als Host-System, ohne die Verwendung von paravirtualisierten Treibern, durchgeführt, da diese eine Kernelverion größer gleich 2.6.25 voraussetzen. Als Anwendungen wurden Audiokompressionen als auch eine Gzip-Kompressionen durchgeführt und verglichen.

[2] Linux-Distributor: `http://www.redhat.com/`

zugänglich gemacht wurde.

KVM ist unter der GPL v2 Lizenz verfügbar und bietet daher für jedermann die Möglichkeit, es frei und ohne Einschränkungen einzusetzen, wobei in Fachkreisen spekuliert wird, ob dies der (gewichtigste) Grund für die Integration in den Linux-Kernel darstellte. Neben den Community-politischen Beweggründen muss dennoch hervorgehoben werden, dass es bis zu diesem Zeitpunkt (Februar 2007) kein vergleichbares Virtualisierungsprodukt in die Mainstream Kernel Distribution geschafft hat, was von Seiten der Kernel-Entwickler durch die Qualität des Quellcodes sowie der Implementierung begründet wird.

So stellt KVM die Standardvirtualisierungslösung für den Long-Term-Release Hardy Heron[3] dar [Fis08a].

Von Seiten Red Hats wurde bekanntgeben, dass bereits 2009 erste Produkte auf den Markt kommen werden, welche KVM als Virtualisierungsgrundlage einsetzten [Fis08b]. Aufgrund der Vielfalt an Distributionsversionen (bedingt durch die Kernel-Versionen und der Prozessor-Architektur) soll auf eine Listung dieser, zu diesem Zeitpunkt verzichtet werden. KVM steht unter `http://www.linux-kvm.org/` frei zum Download bereit.

Die Aufnahme in den Mainstream-Kernel trägt laut Meinung des Autors, bedeutend zur Verbreitung und Akzeptanz der Technologie bei. Dies wird auch ersichtlich, wenn die Hauptakteure, welche das Projekt fördern, betrachtet werden. Zu ihnen zählen etwa AMD, IBM, Intel sowie Red Hat selbst.

Gegenständlich wird (lt. Qumranet) an der Verbesserung des GUI sowie der Integrationsfähigkeit für Xen-Clients gearbeitet. Weiters wird an der Hardwareunterstützung sowie dem Management dieser gearbeitet (zum Beispiel PCI-Weiterreichung an Clients). Darüber hinaus stehen vor allem Funktionalitäten, welche für Unternehmenslösungen benötigt werden, als auch das Aufheben bestehende Limitationen im Vordergrund.

8.2.3. Technik

KVM stellt einen klassischen Ansatz der Hardware-(CPU-)Virtualisierung auf Basis einer Linux-Kernel-Komponente dar [KVM09]. Dazu benötigt die Kernel-based Virtual

[3] Ubuntu 8.04 LTS welches am 24. April 2008 erschienen ist und bis 2013 (Server) unterstützt wird.

Machine zumindest einen virtualisierungstauglichen Prozessor von Intel oder AMD (Intel VTx oder AMD-V).

Von Seiten der KVM-Entwickler ist die Implementierung zur Nutzung von Para-Virtualisieurngsmethoden geplant, bedingt jedoch in der gegenwärtigen Ausprägung dennoch das Vorhandensein eines virtualisierungstauglichen Prozessors, weshalb dessen Einsatzgebiet sehr eingeschränkt ist [Fis08a].

Der Grund für diese Erweiterung dürfte die oftmalige Deaktivierung der Virtualisierungsfunktionalität von Seiten der Hardware-Hersteller sein, was eine Nutzung unmöglich machen würde, da diese teilweise nicht durch den Benutzer reaktiviert werden kann.

KVM stellt im Produktivbetrieb einen Hypervisor vom Typ 2 zur Verfügung, da dieser auf einen proprietären Mikrokernel angewiesen ist. Durch das Fehlen des eigenen Mikrokernels wird der Einsatz des Linux-Kernels unter Verwendung einer Schnittstellenerweiterung nötig.

KVM besteht grundsätzlich aus zwei funktionalen Elementen [Fis08a]:

- Gerätetreiber mit Schnittstelle zur virtuellen Hardware
- *Qemu* basierter virtueller PC

Der Aufbau sowie das Zusammenspiel dieser Komponenten wurde bereits in Abbildung 7.2 auf Seite 85 skizziert und ist dieser zu entnehmen.

Qemu kommt dabei die Aufgabe des Prozess-Emulator zu und stellt somit die Umgebung (virtuelle Maschine) für eine Vielzahl an (Gast-)Betriebssystemen dar (sofern x86 kompatibel) [QEM08].

KVM übernimmt in dieser Situation keine Emulation, stattdessen stellt es die Infrastruktur für diese bereit. Um diese Schnittstelle nutzen zu können ist derzeit die Verwendungen eines modifizierten Qemu unerlässlich. Nach dem Starten beziehungsweise Laden von KVM beginnt der Linux-Kernel die Aufgabe des Hypervisors zu übernehmen beziehungsweise durchzuführen, was gemeinhin als Hypervisor vom Typ 2 bezeichnet wird [Fis08a].

8.3. Sonstige

Abseits der beiden zuvor genannten populären Lösungen gibt es eine Vielzahl weiterer Virtualisierungslösungen, wovon nachfolgend die wichtigsten, ohne Anspruch auf Vollständigkeit, gelistet werden sollen.

andLinux *andLinux* stellt eine Virtualisierungslösung zur Anwendungsintegration dar und ermöglicht den täglichen Einsatz von Linux-Systemen direkt auf einem Windows-Desktop. Technisch gesehen wird der Linux-Kernel sowie eine modifizierte Version von Ubuntu in die Windows-Umgebung portiert, was einen dual-Boot der Systeme überflüssig macht. Für weitere Informationen wird auf die URL: `http://www.andlinux.org/` verwiesen.

Cygwin *Cygwin* stellt eine API-Emulation zur Anwendungsintegration dar und ermöglicht eine Verwendung von Unix-Programmen in einer Windows-Umgebung. Cygwin stellt den wahrscheinlich bekanntesten Vertreter der POSIX[4]-Implementierungen dar und integriert technisch gesehen eine Kompatibilitätsschicht für Unix-Programme, welche auf die Windows-Bibliotheken zugreift. Für weitere Informationen wird auf die URL: `http://www.cygwin.com/` verwiesen.

Microsoft Virtual PC *Microsoft Virtual PC* stellt eine Virtualisierungslösung zur Anwendungsintegration dar und ermöglicht ein Ausführen diverser (nicht aller) Betriebssysteme unter einer Windows-Umgebung. Voraussetzung ist zumindest eine Windows 2000 Installation, wobei die Software kostenfrei bezogen werden kann. Für weitere Informationen wird auf die URL: `http://www.microsoft.com/windows/products/winfamily/virtualpc/default.mspx` verwiesen.

Microsoft Hyper-V *Microsoft Hyper-V* stellt eine Virtualisierungslösung zur Serverkonsolidierung dar und ermöglicht das parallele (virtualisierte) Ausführen unterschiedlicher Betriebssysteme auf einem Windows Server 2008 (64 Bit) Hostsystem. Neben der Hardwareunterstützung moderner CPUs wartet dieses Produkt mit einem hohen Grad an Para-Virtualisierung auf. Interessanterweise ist dieses Produkt im Preis des Windows Severs 2008 inbegriffen und kann so ohne zusätzliche Kosten zum Einsatz kommen.

[4]POSIX stellt eine gemeinsame Plattform für Binärprogramme unter Unix dar.

Für weitere Informationen wird auf die URL: `http://www.microsoft.com/germany/windowsserver2008/virtualisierung.mspx` verwiesen.

Sun VirtualBox *Sun VirtualBox* stellt eine Virtualisierungslösung zur Anwendungsintegration dar und ermöglicht ein Ausführen vieler Gast-Betriebssysteme auf unterschiedlichen Host-Systemen (Windows, Solaris, OpenSolaris, ...). VirtualBox stellt eine ausgesprochen schlanke Implementierung dar und kann als eingeschränkte Open Source Edition sowie als Vollversion zum persönlichen Gebrauch kostenfrei bezogen werden. Von der Verwendung der Open Source Version wird aufgrund der fehlenden USB-Unterstützung jedoch abgeraten. Für weitere Informationen wird auf die URL: `http://www.virtualbox.org/` verwiesen.

VMware Player *VMware Player* stellt eine Virtualisierungslösung zur Anwendungsintegration dar und ermöglicht das Abspielen von virtuellen Maschinen unterschiedlicher Natur (auch fremder Images). In der nicht-servicierten Version kann der Player frei heruntergeladen werden, wobei auch eine Version (ACE-System) mit Support (kostenpflichtig) angeboten wird. Als Hostplattformen können sowohl Windows als auch Linux-Systeme genutzt werden, wobei gleiches für die Gast-VMs gilt und zusätzlich von einer generellen Unterstützung von diversen 32- und 64-Bit Systemen ausgegangen werden kann. Für weitere Informationen wird auf die URL: `http://www.vmware.com/de/products/player/` verwiesen.

VMware Server 2 *VMware Server 2* stellt eine Virtualisierungslösung zur Serverkonsolidierung dar, welche mittels Web-Tools aktiv gesteuert werden kann. Die Funktionalität ist ähnlich des VMware Workstation Produkt, wobei der Server kostenfrei bezogen werden kann. Die zweite Version des Produkts wurde speziell wegen der IIS-basierten Exploit-Gefährdung auch für andere Web-Server optimiert. Für weitere Informationen wird auf die URL: `http://www.vmware.com/de/products/server/` verwiesen.

8.4. Exkurs: Hardwarevirtualisierungstechnologien

Prinzipiell lässt sich festhalten, dass Virtualisierungstechnologien wie Intel's VT oder AMD's IOMMU (I/O Memory Management Unit) darauf abzielen, Virtualisierung auf

einer Plattform-unabhängigen Basis bereitzustellen [Int08]. Zusätzlich zu der Hardwareunterstützung zur Prozessorvirtualisierung werden auch Mechanismen zur Kontrolle beziehungsweise zum Management des I/O-Systems beziehungsweise dessen Peripherie auf virtueller Ebene unterstützt.

Hardwareunterstützung zur Prozessorvirtualisierung Der Vorteil der Hardwareseitigen Unterstützung der Prozessorvirtualisierung liegt in dem Wegfall des entsprechenden Funktionsumfanges im VMM[5].

Dies bewirkt zum einen eine mögliche Leistungssteigerung beziehungsweise Ressourcenschonung und zum anderen eine Reduktion des VMM-eigenen Quellcodes. Diese Reduktion des Quellcodes ihrerseits kann sich somit positiv auf die Zuverlässigkeit und Beständigkeit des VMM auswirken, da eine (eigene) Schnittstelle zur Behandlung von Exceptions und Events entfallen kann.

Um einen sicheren Betrieb gewährleisten zu können, muss eine CPU Speicher Virtualisierung vorgenommen werden. Diese soll ungewollten Zugriff vermeiden und eine Isolation der einzelnen VMs sicherstellen.

Folgende Grafik soll einen Überblick zur Thematik bieten:

I/O Virtualisierung Eine weitere wichtige Aufgabe des VMM liegt in der Bereitstellung von Mechanismen, um virtualisierte I/O Anfragen von Seiten der Gäste beziehungsweise dessen Software behandeln zu können.

Um dies gewährleisten zu können, gibt es im Allgemeinen vier Modelle [Int08]:

- **Emulation** des I/O Gerätes unabhängig von der tatsächlich vorhandenen Hardware (zum Beispiel Kompatibilität gegenüber der Legacy-Systeme).
- **Software Interfaces** werden zur Distribution der I/O-Geräte für die Gäste bereitgestellt und entsprechen einem synthetischem Geräte Interface.
- **Abgrenzung beziehungsweise Abtretung** (auch Assignment) des I/O Gerätes bei einer Weiterreichung des physischen Gerätes an eine dezidierte VM. Dies

[5] Virtual Machine Monitor

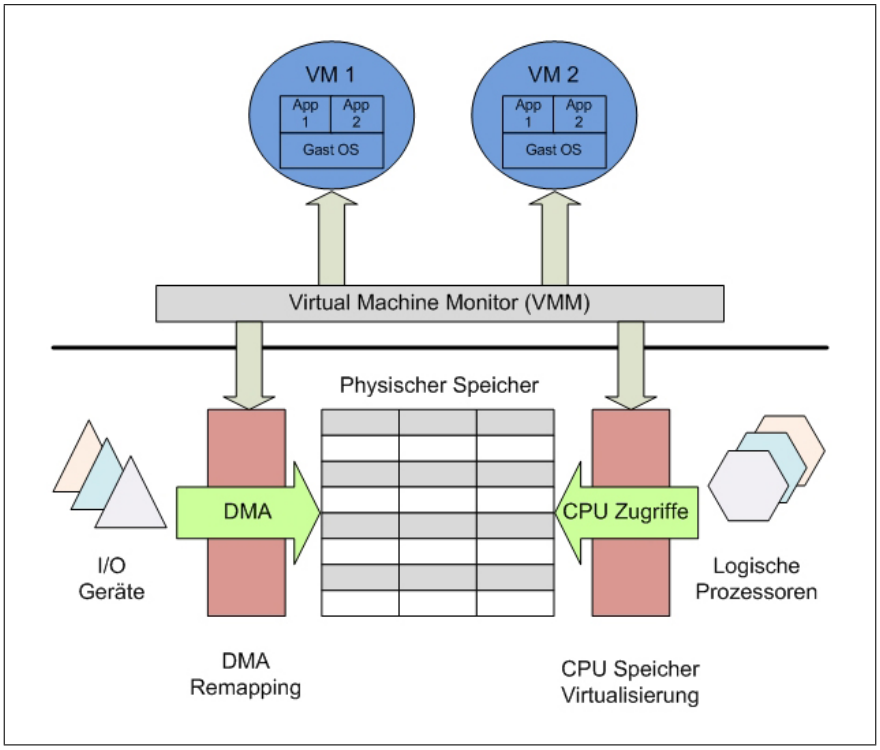

Abbildung 8.1.: Interaktionsschema von I/O und Prozessor Virtualisierung [vgl. Int08, S. 16]

bedingt zur sicheren Verwendung eine (Hardware-)Unterstützung zur Isolation und Begrenzung der eingesetzten Ressourcen und hat den Vorteil eines minimalen Interaktionsgrades mit dem VMM. Dieser Ansatz wird zumeist auch als I/O-Container basierter Ansatz bezeichnet, wobei die Installation des Hardware-Drivers als privilegierte Software im VMM entfallen kann und somit die Fehleranfälligkeit des VMM reduziert werden kann.

- **Gemeinsame Nutzung** beziehungsweise Sharing des I/O-Gerätes, welches als Erweiterung der Abtretung angesehen werden kann und das Ansprechen der Hardware von unterschiedlichen VMs erlaubt. Diese Distribution von multiplen funktionalen Schnittstellen erlaubt eine gemeinsame Nutzung des Host-Gerätes und kann (sofern unterstützt) dynamisch die verfügbaren Ressourcen aufteilen.

Abhängig vom Einsatzgebiet kann es notwendig sein, dass der VMM all diese Modelle unterstützen muss. Als klassische Anwendungsbeispiele gelten etwa I/O intensive Prozesse, welche am besten mittels Abgrenzung (direkte Zuweisung) realisiert werden können, aber auch die Weiterverwendbarkeit von Alt-Systemen unter der Verwendung vom Emulation zur Bereitstellung der notwendigen Infrastruktur.

Hardwareunterstützung für I/O Virtualisierung Um die sichere Verwendung, insbesondere die Isolation und Restriktion der I/O-Geräte gewährleisten zu können, ist es notwendig, dem Hostsystem beziehungsweise bei genauerer Betrachtung dem VMM ein entsprechendes Rüstzeug (unabhängig vom I/O-Virtualisierungsmodell[6]) mitzugeben.

Intel's VT stellt für diese Zwecke folgende (Hardware-basierte) Funktionalitäten zur Verfügung [Int08]:

- I/O Device Assignment: Zur Gerätebereitstellung an VMs.
- Interrupt Remapping: Zur Isolation und Weiterleitung der Interrupts an die entsprechenden VMs.
- Reliability: Protokoll- und Meldesystem um DMA Interrupt Errors auswertbar zu machen.
- **DMA Remapping** Sorgt für die nötige unabhängige Adressübersetzung für DMA (Direct Memory Access) von Seiten der Geräte. Näheres zum Thema DMA Remapping ist im entsprechenden Abschnitt auf Seite 102 nachzulesen.

Ähnlich Intel's DMA Remapping bietet auch AMD's IOMMU Funktionen um die Adressübersetztung der DMA Transaktionen durchzuführen und so einen Schutz des Speicherbereiches vor nicht privilegiertem Zugriffs zu realisieren [AMD07]. Auch AMD stellt diese Funktionalität auf Chipset-Funktions-Basis bereit.

Als mögliche Anwendungsfälle der IOMMU-Technologie, unter dem Aspekt der Virtualisierung, gibt AMD (unter anderen) folgende Szenarien an [AMD07]:

- **32 Bit zu 64 Bit I/O-Mapping** Um Legacy-Geräte, welche auf einer 32 Bit Architektur aufbauen, auch unter 64 Bit-Systemen einsetzten zu können. Vorwiegendes Problemfeld, ohne des Einsatzes von IOMMU, wäre die Speicher-

[6] Nicht alle VMMs unterstützten alle oben genannten I/O-Virtualisierungsmodelle.

adressierung unter der Verwendung von DMA, da unter 32 Bit eine vorzeitige Begrenzung unvermeidbar wäre.

- **Gerätezugriff im Benutzermodus** Zentrale Regelung und Beschränkung des Zugriffs auf I/O Geräte auf Basis einer Legitimationstabelle.
- **VM Gastzugriff auf Geräte** Direkter I/O-Zugriff unter VM-Gastmaschinen. Dies hat den Vorteil, dass die Gastmaschine nicht modifiziert werden muss und der Gast I/O-Bereich direkt in den Host-I/O-Bereich übernommen wird. Dies führt in den meisten Fällen zu einer Performancesteigerung bei der Benutzung der unterschiedlichen Geräte von Seiten des Gastes.

VMM I/O Sicherheit Basierend auf der Tatsache, dass moderne I/O Architekturen höchst komplexe Ausmaße angenommen haben, ergibt sich das Problem, dass der VMM simpel und möglichst klein gehalten werden soll. Denn nur der übersichtliche Umfang des VMM (oder auch Hypervisors) kann die Sicherheit sowie strenge Isolation der VMs untereinander sowie der sensiblen Daten sicherstellen.

Empirisch lässt sich festhalten, dass die Virtualisierung der I/O-Geräte generell als problematischer angesehen werden kann als jene des Prozessors beziehungsweise der Prozessoren. Dieser Umstand ist bedingt durch die Vielzahl der verschiedenen Geräte, welche unterstützt werden sollen, wohingegen die Anzahl der Prozessoren wesentlich übersichtlicher ist.

Ausgehend von dem Gedanken, dass der VMM möglichst simpel gehalten werden soll, haben sich im Laufe der Zeit unterschiedliche Techniken herauskristallisiert wie die I/O-Virtualisierung realisiert und gleichsam klassifiziert werden kann. Diese VMM-Klassen sollen nachfolgend dargestellt werden und sicherheitstechnisch betrachtet werden. Folgende Hypervisor-Klassen, basierend auf der verwendeten I/O-Virtualisierungsmethode, können unterschieden werden [KS08]:

- **Pure Isolation** ist die einfachste als auch die sicherste Klasse der Hypervisors, welche auf einem Server-System laufen können. Als Beispiele für diese Klasse seien der PR/SM auf dem IBM System z Server sowie Xen mit IOMMU-Unterstützung genannt. Hierbei werden die Geräte nicht von mehreren VMs verwendet und es kommt somit zu keinem *sharing* beziehungsweise keiner gemeinsamen Nutzung. In dieser Klasse sind die Geräte-Driver direkt in den

Gast-Systemen angesiedelt und haben aus Sicht des VMM keine Vertrauensstellung. Da die Gäste untereinander und unabhängig voneinander agieren, können Timing-Probleme und dergleichen eliminiert werden und der VMM als Minimal-Code implementiert werden. Der größte Nachteil dieser Methode liegt in der unbehandelten Verwundbarkeit gegenüber von Prozessor-Cache-Angriffen [KS08].

- **Sharing Hypervisor am Server** bedingen eine gesonderte Behandlung der gemeinsam verwendeten Netzwerk- und Speichergeräte, welche als kritische Sicherheitslücken angesehen werden können. Eine Verschlüsselung der Datenströme kann zur Anwendung kommen, hat jedoch den wesentlichen Nachteil des Performance-Overheads. Darüber hinaus können mittels nicht vertrauenswürdiger Speicher-Driver dennoch umfangreiche Analysen, basierend auf dem Traffic-Flow, durchgeführt werden, selbst wenn die Daten selbst nicht lesbar beziehungsweise verwertbar sind. Bei dieser Klasse von VMMs können die gemeinsam verwendeten Driver entweder direkt im VMM, als auch in einer speziell privilegierten I/O Partition abgelegt sein.

- **Sharing Hypervisor im Client** stellen den schwierigsten Anwendungsfall für den VMM dar, weil alle Geräte mit dem Gast geteilt werden müssen. Zum Problem wird dies vor allem bei der Verwendung von gemeinsam genutzten Anzeige-Geräten, welche keine Unterstützung für verschlüsselte Datenübertragung bieten. Da die Driver für solche Geräte zumeist proprietär und groß sind, kommt es zum direkten Interessenskonflikt bei der Integration im VMM. In diesem Anwendungsfall bestehen im Vergleich zu den zuvor genannten, wesentlich mehr Möglichkeiten zum direkten Durchsickern von kritischer Information, weshalb der Einsatz als bedenklich angesehen werden kann.

- **Exkurs: Spezielle privilegierte I/O Partitionen** stellen Umgebungen zur Bereitstellung von umfangreichen I/O-Support bereit. Hierbei wird die Virtualisierung der I/O-Driver auf eine spezielle Partition verlagert, wobei der VMM in diesem Fall zumeist als komplettes Linux-System darstellt und somit weit von der simplen und überschaubaren Implementierung entfernt ist. Im Falle der privilegierten I/O Partitionen werden die I/O-Requests der Gäste auf diese Partition umgeleitet und verarbeitet. Diese beliebte Art der Implementierung bietet eine simple, kostengünstige als auch schnelle Unterstütztung von diversen I/O-Gerätschaften. Einher gehen jedoch auch sicherheitstechnische Nachteile, wie die Möglichkeit des direkten Angriffs, als auch die Auflösung der Ring-Struktur, welche die Sicherheit gewähren soll. Diese Auflösung der

Ring-Struktur kann jedoch durch die Verwendung von IOMMU und somit durch direktes Zuweisen der Hardware zu den einzelnen Gästen entgegengewirkt werden. Darüber hinaus sind auch die Performance-Werte schlechter als bei anderen Realisierungen.

DMA Remapping Hardwareunterstütztes DMA Remapping stellt die Grundlage für die Isolation der Gerätezugriffe auf den Speicher dar [Int08]. Es ermöglicht eine Zuweisung von einzelnen Geräten zu Speicherbereichen und wird über sogenannte I/O Page Tables kontrolliert und parametrisiert. Durch die Zwischenschaltung von DMA-remapping fähiger Hardware wird es ermöglicht, eine Abfrage auf Gültigkeit beziehungsweise Zulässigkeit einer Geräte-(Speicher-)Anfrage durchzuführen und diese entweder zu verwerfen oder weiterzuleiten.

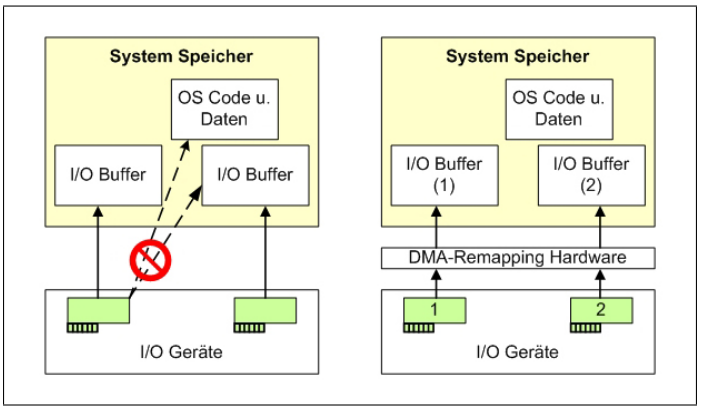

Abbildung 8.2.: DMA Remapping Hardware sowie Einsatz [vgl. Int08, S. 14]

Ebenso ist es möglich, etwaige Speicherbereiche auf anderen abzubilden beziehungsweise auf diese zu verweisen. Dieser Vorgang wird in der Fachsprache als *Mapping* bezeichnet. Neben einer Möglichkeit zum Cachen von häufig benötigten Page-Tables, können die unterschiedlichen Devices entweder einzeln oder aber auch im Verbund parametriert werden.

Eine Übersicht über das Zusammenspiel der einzelnen Komponenten kann der Abbildung 8.1 auf Seite 98 entnommen werden.

Die Einsatzbereiche beziehungsweise -szenarien von DMA Remapping sind mannigfaltig und von der Anwendungsebene abhängig. So kann DMAR, aus Sicht des Betriebssystems

zum Schutz dessen, zur Bereitstellung von Legacy-Features (32 Bit zu 64 Bit) als auch zur Geräteisolation, Verwendung finden. In Verbindung mit dem VMM kann DMAR zum Beispiel zur Weitergabe eines I/O-Geräts an die VM dienen (Device Assignment). Aus Sicht eines VM-Gasts kann DMAR zum Beispiel dazu verwendet werden, um ein Abbild des DMA im Host zu erreichen, was in weiterer Folge zu Leistungssteigerungen auf Gastebene führen kann.

9. VM-Sicherheit

Im nachfolgenden Kapitel werden grundlegende Angriffsszenarien, als auch deren Vermeidung besprochen werden. Ausgehend von der Problemstellung sowie der Definition der gewichtigsten Bedrohungen, soll der Leser einen Einblick in die Thematik erhalten. Weiters wird die gegenwärtige Affinität der IT-Branche gemessen an Befragungsergebnissen reflektiert und dargebracht.

9.1. Grundlagen und Problemdefinition

Das grundlegende Problem beim Einsatz von Virtualisierungstechnologie ist jenes der übereilten Umsetzung zum Zwecke der Kostenersparnis. Die mögliche Kostenreduktion beziehungsweise Effizienzsteigerung führte somit vielfach zu einer überhasteten Umsetzung, ohne Bedacht auf die Schaffung von Sicherheitslücken zu nehmen. Generell scheint Virtualisierung vielfach als generelle IT-Strategie Verwendung zu finden, ohne Rücksicht auf die Zentralisierung der Sicherheitsaspekte zu nehmen [Dub08].

Um die Ausmaße dieses Trends quantifizierbar zu machen, soll hier von einer prognostizierten Anzahl von 45%[1] an virtualisierten Server-Systemen im Unternehmensumfeld ausgegangen werden [Bab08].

Ein vielfach diskutiertes Problem der Virtualisierung liegt im Hinzukommen einer zusätzlichen Schicht zur Sicherung dieser. Dabei unterscheiden sich die Sicherungsmaßnahmen nur minimal von jener der physischen Gerätschaften mit dem Unterschied, dass ein tieferes Verständnis über die Implementierung und Funktionsweise dieser Systeme vorhanden sein muss beziehungsweise etwaige Kompetenzen diesbezüglich gefördert oder aufgebaut werden müssen [Dub08].

[1] lt. Frank Gillett (Forrester Research) im Jahr 2008 [Bab08]

Das erste, und aus sicherheitstechnischer Sicht gravierendste Problem bei der Verwendung von VMs stellt die Konnektivität dieser untereinander dar [Cum08]. So sind sich Sicherheitsexperten einig, dass es eine sehr begrenzte Anzahl an Möglichkeiten gibt diese zu beschränken, da eine Kommunikation schon alleine über den Hypervisor, sprich ohne Verlassen des physischen Hosts, möglich ist.

Dies bedeutet in weiterer Folge jedoch auch, dass traditionelle Tools wie Intrusion-Detection-Systeme, Netzwerk-Zugangskontrollsysteme, Traffic-Analyzer und der gleichen, den fortschreitenden Anforderungen im virtuellen Umfeld nicht gerecht werden und demzufolge nur nach außen hin wirksam sind und bleiben [Her08]. Der Host-Server entspricht demnach einem einzelnen physischen Server, zumindest aus der Sicht dieser Tools.

Somit lässt sich zusammenfassend festhalten, dass die Interaktion beziehungsweise der Netzwerkverkehr zwischen den Gästen (Instanzen der VMs) äußerst schwierig angezeigt und überwacht werden kann. Dies gilt vor allem für die Host-System-eigenen Boardmittel und Analyseprogramme, welche um proprietäre Software erweitert werden müssen [Gre08].

Ein weiteres Problem stellt der Auswuchs an Legacy-Systemen beziehungsweise individualisierten Systemen dar, da das Aufsetzten im Vergleich zur herkömmlichen Installation ein beträchtliches Maß an Zeitersparnis bieten kann. Dies jedoch bedingt wiederum ein Vielfaches an Arbeitsaufwand (Updates und ähnliches), welcher zur Wartung dieser Systeme investiert werden muss und kompensiert somit einen Großteil an Einsparungspotential. So stellte sich heraus, dass bei Umfragen bereits 15% der Systemadministratoren diese Situation des „Patchings“ als undurchführbar ansahen [Her08].

Diese Situation stellt natürlich das klassische Problem der fehlenden Standardisierung in Unternehmen dar und soll deswegen in dieser Arbeit nicht weiter behandelt beziehungsweise erläutert werden.

9.2. Wechselwirkung Host- und Gastmaschine

Hinsichtlich des operativen Betriebes von Virtuellen Maschinen kann davon ausgegangen werden, dass es prinzipiell zwei denkbare Möglichkeiten von Angriffen auf ein virtualisiertes System durchgeführt werden können. Zur Unterscheidung dient die Richtung, in welcher ein Angriff ausgeführt wird beziehungsweise von welchem System dieser ausgeht.

Den systematischen Aufbau einer lauffähigen Umgebung für VMs soll folgende Grafik verdeutlichen:

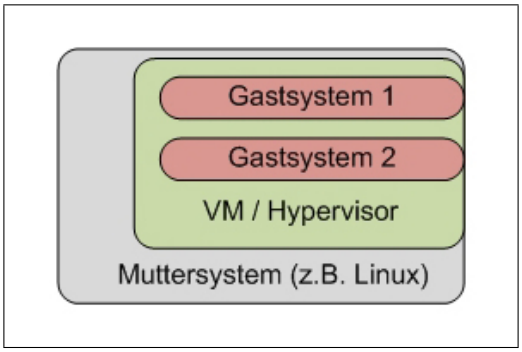

Abbildung 9.1.: Schematische VM Infrastruktur [vgl. Rue07, S. 653]

Basierend auf der vorliegenden Grafik können wir die grundsätzlichen Angriffsmöglichkeiten näher spezifizieren als [Rue07]:

- Ausbruch aus einem Gastsystem
  - Übergriffe auf andere Gastsysteme (**Gast zu Gast**)
  - Übergriffe auf das Hostsystem beziehungsweise den Hypervisor (**Gast zu Mutter, Gast zu Hypervisor**)
- Mutter- beziehungsweise Hostsystem als (Angriffs-)Ausgangspunkt (**Mutter zu Gast**)[2]

Da der zweite Fall, in dem das Host-System als Ausgangspunkt dient, dem einer ursprünglichen Serverinstallation beziehungsweise -sicherung entspricht, wird gemeinhin angenommen, dass dieser Fall als relativ unwahrscheinlich einzuschätzen ist. Dies ist vor allem dadurch begründet, dass dieses System nur die Infrastruktur zur Betreibung der VMs bereitstellt, darüber hinaus jedoch keinerlei Dienste ausführen sollte. Im Extremfall kann dies sogar bedeuten, dass das Netzwerkinterface für den Host deaktiviert wird und somit nur noch die Gast-Systeme mit der Außenwelt kommunizieren können. Eine Übernahme des Host-Systems würde gleichsam einer Übernahme beziehungsweise Kontrolle aller auf dem Hostsystem installierter VMs entsprechen, da diese vom Host-System verwaltet und betrieben werden.

[2]Kann als vollständige Übernahme der darin laufenden VMs angesehen werden.

Aufgrund der obigen Umstände sowie der damit verbundenen niedrigen Wahrscheinlichkeit, wird auf eine Darstellung dieses Zustandes im folgenden verzichtet, sie wird jedoch aus Gründen der Vollständigkeit dennoch genannt.

Folgende Grafik soll die verbleibenden Angriffsszenarien skizzieren:

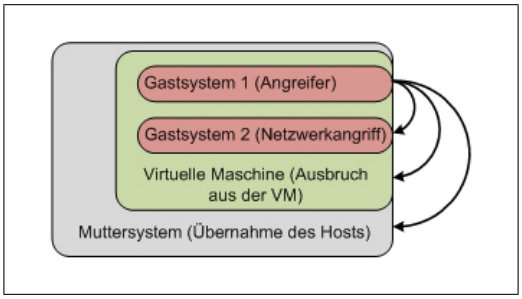

Abbildung 9.2.: Mögliche Angriffe vom Gastsystem ausgehend [vgl. Rue07, S. 654]

Somit verbleibt die Möglichkeit eines Ausbruchs aus der Gast-VM, wobei dessen Ausprägungen nachfolgend gesondert behandelt werden.

9.3. Angriffe auf andere Gäste

Ausgehend von der Situation, dass ein Gastsystem kompromittiert ist, kann ein Angreifer versuchen andere Gast-Systeme anzugreifen.

Diese Praktik wird auch als *Intrahost Threath* bezeichnet und unterscheidet sich im Wesentlichen nicht von konventionellen Angriffen (div. Ausprägung) mit dem Unterschied, dass dieser Angriff nach außen hin nicht beziehungsweise nur schwer von den traditionellen Tools erkannt werden kann [Her08].

Zumeist finden Betriebssystem- beziehungsweise Applikations-Exploits Anwendung um den Gast zu befallen und darüber hinaus eine Ausbreitung im System zu erlangen.

9.4. Angriffe auf den Hypervisor beziehungsweise die Mutter

Der erfolgreiche Angriff der Software-Komponente, welche die Verwaltung über die unterschiedlichen virtuellen Maschinen hat, kann gleichgesetzt werden mit der Übernahme aller virtuellen Maschinen [Gre08].

Angriffe über das Netzwerk Der Angriff vom Netzwerk aus kann relativ einfach über die Sicherheitsinfrastruktur (zum Beispiel Firewall) unterbunden werden, vorausgesetzt alle nicht benötigten Dienste sind deaktiviert. Die verbleibenden (aktiven) Dienste sollten natürlich ständig auf dem aktuellsten Stand gehalten werden.

Obwohl derzeit noch keine Sicherheitslücken zur direkten Übernahme des Hypervisors (aus Netzwerksicht) bekannt sind, kann dennoch davon ausgegangen werden, dass dies lediglich eine Frage der Zeit ist [Gre08].

Aufgrund der Einfachheit, diese Angriffe abwehren zu können, welche ja über das Netzwerk und nicht von einem Gast-System ausgeführt werden, werden diese nachfolgend in dieser Arbeit nicht näher betrachtet.

Angriffe, welche von Gast-System ausgehen Ausgehend von der Situation, dass ein Gastsystem kompromittiert ist, kann ein Angreifer versuchen, den Hypervisor zu infizieren, um an sensible Daten zu kommen beziehungsweise Zugriff auf das gesamte System zu bekommen.

Core Security Technologies[3] zeigte bereits im Jahr 2008 durch einen Laborversuch (unter Verwendung von VMware-Produkten), wie dies möglich ist[4] [Bab08].

Diese als *Hyperjacking* bekannte Methodik resultiert, ähnlich wie bei einem Angriff von außen auf den Hypervisor, in einer Situation in welcher ein Angreifer alle Gäste der betroffenen Host-Maschine unter Kontrolle hat.

Dies kann zum Beispiel dazu verwendet werden, um alle Pakete, welche den Hypervisor

[3] Softwarefirma mit der Spezialisierung auf Netzwerk Sicherheit
[4] Nähere Information über den Versuch sowie -aufbau in [Bab08] nachzulesen.

durchlaufen, mitzulesen und gegebenenfalls auch zu manipulieren, falls dies erforderlich beziehungsweise gewünscht ist [Her08].

Technisch kann dieser Zustand am besten mit einem Rechner, welcher mit einem Rootkit versehen ist, verglichen werden, was zu der Möglichkeit der Schaffung einer Transparenz des schadhaften Programms beziehungsweise Systems ausgenutzt werden kann.

Diese Möglichkeit ist durchaus auch dadurch bedingt, dass Rootkits (in vielen Fällen) auf der Kernel-Ebene eines Systems arbeiten und die zur Erkennung dieser eingesetzte Software auf einer höheren Ebene angesiedelt sind [Rad08]. Dieser Umstand bedeutet jedoch auch eine drastische Reduktion des Handlungsspielraums sowie der Privilegien der Erkennungssoftware.

Obwohl dieser Umstand bereits seit Jahren bekannt ist und auch diskutiert wird, werden diese Mängel beziehungsweise Gefahren oftmals unterschätzt beziehungsweise falls vorhanden mangelhaft publiziert. Was in weiterer Folge dazu führte, dass bei einer Umfrage (unter IT-Administratoren) dennoch 36% der Beteiligten noch nicht einmal von Hyperjacking gehört hatten [Her08].

9.5. Lösungsansätze

Prinzipiell gibt es drei Möglichkeiten die zuvor genannten Problematiken handzuhaben, von denen jedoch nur zwei von praktischer Relevanz sind.

Diese drei Möglichkeiten werden repräsentiert durch [Gre08]:

- **Externalisierung des Traffics** Der gesamte anfallende Traffic jeder einzelnen Gast-VM wird aus dem physischen Host herausgeführt, gescannt und anschließend an sein Ziel (andere Gast-VM) weitergeleitet. Dies bedingt die Verfügbarkeit hochskalierbarer Scannerinstrumentarien, als auch eine hohe Belastung des vorhandenen I/O-Systems und stellt aufgrund dessen eine wenig praktikable Lösung dar.
- **Gast VM-Firewalling** Hierbei werden alle Gast-VMs mit Software-Firewalls ausgestattet, um den Traffic direkt filtern zu können. Dies bedingt durch die Wartung, als auch durch die Lizenzierung einen gewissen Overhead an Wartungs- und Instandhaltungsaufwand.

- **VM spezifische Lösungen** Hierbei handelt es sich um Softwarekomponenten, welche speziell für den Einsatz in beziehungsweise auf virtualisierten Systemen erstellt wurden. Aufgrund der Vielfalt dieser werden sie nachfolgend gesondert behandelt.

Aufgrund der Vielzahl der verfügbaren Softwarelösungen werden wir uns auf die Konzepte sowie einige wenige konkrete Implementierungen konzentrieren. Die Vielzahl der verfügbaren Produkte dürfte durch das (vormals) mangelnde Sicherheitsbewusstsein der VM-Hersteller begründet sein, was zur Entstehung eines Nischenmarktes geführt hat. Dennoch sei allgemein darauf hingewiesen, dass der Einsatz von Hersteller-spezifischer Sicherheitstechnologie schon alleine durch die Privilegien-Erweiterung als Vorteil angesehen werden kann. Andererseits stellen bestimmte Hersteller auch „nur“ Schnittstellen zur Implementierung von Sicherheitssystemen bereit und überlassen deren Entwicklung den spezialisierten Firmen.

Ohne Anspruch auf Vollständigkeit zu erheben, lassen sich die unterschiedlichen Konzepte in folgende kategorische Klassen einteilen:

Hypervisor API Ausgehend von dem Gedanken, dass der Hypervisor gewisse Aktivitäten vorzeitig erkennen kann, ohne kurzfristig selbst erkannt zu werden, kann dieser um Sicherheitsfunktionalitäten erweitert werden. Geschieht diese Erweiterung jedoch nicht durch den Hersteller selbst, sondern stellt dieser nur die Schnittstelle zur Implementierung solcher Funktionalitäten bereit, so spricht man von der Bereitstellung einer *Hypervior API Lösung*.

Diese Schnittstelle ermöglicht eine Erstellung von Seiten der spezialisierten Drittanbieter zum Schutz beziehungsweise zur Überwachung des Hypervisor sowie dessen Aktivitäten. Dies hat den enormen Vorteil, dass Sicherheitsprodukte nicht wie gewöhnlich auf der Anwenderschicht operieren, sondern direkt auf der „Kernel“-Ebene ausgeführt werden können [Bab08]. Dies geht gleichsam mit einer Steigerung der Privilegien solcher Softwarelösungen einher.

Als konkrete Realisierung dieses Konzepts kann das VMware VMSafe (Security Technology) angesehen werden [Bro08]. Dieses soll Programmierern die Möglichkeiten zur Kontrolle des Hypervisors zur Verfügung stellen und so das Abfangen von Viren und anderem Schadcode ermöglichen. Eine Vielzahl an namhaften Softwarehäusern hat sich

offiziell zur Implementierung von Sicherheitslösungen für diese Schnittstelle angemeldet.

VM Selbstprüfung Als *VM Selbstprüfung* oder *VM Introspection* (kurz VMI) bezeichnet man die Überprüfung von VMs zur Laufzeit [NH08]. Teil der Analysen sind die Überwachung der Verhaltensweisen von VM und gegebenenfalls das Anhalten dieser zur Untersuchung der gegenständlichen Inhalte, wie zum Beispiel des Hauptspeichers und Ähnlichem. Diese Überwachung findet außerhalb der eigentlichen VM statt und birgt dem zu Folge einige Vorteile gegenüber jenen Sicherheitslösungen, welche direkt in der VM arbeiten.

Prinzipiell lassen sich zwei strukturelle Stellen identifizieren, welche für die Integration von VMI adäquat erscheinen. Zum einen ist dies direkt im VMM und zum anderen die Integration in einem privilegierten Bereich (zum Beispiel Dom0 bei Xen). Die Integration im VMM ist dabei als kritisch anzusehen, da dies eine Erweiterung des VMM-Quellcodes erfordern würde und somit eine hohe Abhängigkeit der Version dieser generieren würde. Die Integration im privilegierten Bereich hingegen basiert auf der gemeinsamen Verwendung der zur Verfügung gestellten API, welche zumindest mittelfristig als kompatibel angesehen werden kann. Ein Beispiel für diese Art der Implementierung stellt die Introspektions-Lösung VIX von Kara Nance und Brian Hay dar[5]. Dabei ist VIX in der Lage die angehaltene VM sowohl im Festplattenspeicher- als auch im Hauptspeicher-Bereich, zum Zwecke der forensischen Analyse, auszulesen, ohne das die VM davon Notiz nehmen kann.

Generell lässt sich die VMI-Technologie anhand folgender Kriterien klassifizieren [NH08]:

- Behandlung des Angriffs - nur Aufzeichnung und Meldung versus aktive Gegenwirkung
- Semantisches Wissen über die VM - kein Wissen versus Möglichkeit zur Integritätsprüfung
- Event Replay - Überprüfung nur zur Laufzeit versus Wiederherstellung zu Analysezwecken

Der wesentlichste Vorteil vom VMI liegt in der Möglichkeit die VMs außerhalb der eigentlichen OS überprüfen zu können und somit ein vorzeitiges Aushebeln der Sicherheits-

[5] siehe dazu zum Beispiel [NH08]

mechanismen dennoch erkennen zu können. Dies wäre zum Beispiel durch eine Rootkit-Infektion im Gast-OS möglich und würde somit ein Erkennen im selben System schwierig gestalten. Ähnlich zu anderen Lösungen stellt auch hier die direkte Angreifbarkeit des (Software basierten) VMM, bedingt durch Bugs im Quellcode, die größte Gefahr dar.

Vertrauenswürdigkeit Ein weiteres Konzept zur Realisierung von Sicherheitsfunktionalität stellt die dezidierte Betrachtung der Vertrauenswürdigkeit der VMs dar. Wobei die Vertrauensstellung durch den Vergleich eines zuvor erstellten, einmaligen Fingerprints determiniert wird [Bab08]. Dieser einmalige Ausschnitt enthält einen Vergleichswert, welcher auf der Konfiguration der VM basiert. Solange dieser Wert unverändert bleibt, kann davon ausgegangen werden, dass es sich um einen vertrauenswürdigen Gast handelt. Ändert sich dieser Wert jedoch, so ist von einer Manipulation auszugehen und die entsprechende VM wird isoliert.

Diese Funktionalität und die Möglichkeit zur Isolation ihrerseits, bedingen eine Ansiedelung im Hypervisor, da nur dieser dementsprechende Rechte auf dem System hat.

Als konkretes Realisierungsbeispiel dieser Technik ist der *sHype*-ausgestattete Xen Hypervisor zu nennen. Wobei die sHype-Technologie ursprünglich von IBMs Virtualisierungserfahrungen abgeleitet beziehungsweise übernommen wurde.

Sicherheits VMs Als Sicherheits-VMs im Kontext bezeichnet man virtuelle Firewalls mit Intrusion-Prevention-Systemen, welche als Virtuell Maschine ihre Ausprägung finden und so die Kontrolle und Sicherheit der Kommunikation im physischen Host sicherstellen. Dies bringt den Vorteil, dass eine Externalisierung wegfallen kann und so Einsparungen eingefahren werden können (zumindest theoretisch).

Konkrete Realisierungen dieses Konzepts (zum Beispiel Reflex Systems) erwiesen sich jedoch in der Praxis, als auch in Testszenarien als Engpass, da diese die Weiterleitung der Pakete nicht schnell genug sicherstellen konnten [Cum08].

Teil III.

Firewalls unter dem Aspekt der Virtuellen Maschinen

10. Firewalls unter VMs

Bei der Verwendung von Firewalls in VMs auf Gast- sowie Hostseite können bestimmte Ansätze verfolgt aber auch kombiniert werden. Dieses Kapitel soll darüber Aufschluss geben, welche speziellen Anforderungen an Firewalls unter den Bedingungen eines Einsatzes im VM-Umfeld erwachsen sowie als Zusammenfassung der bereits zuvor genannten (speziellen) Eigenschaften dienen.

Neben einer Betrachtung der Netzwerkzonen, welche im Praxisbetrieb vorhanden sein können, sollen auch die Auflösung beziehungsweise Aufweichung dieser bei Verwendung von VM-Technologie aufgezeigt werden. Ebenso sollen die Punkte, welche für eine Filterung des Netzwerkverkehrs in Frage kommen, vorgestellt werden. Darüber hinaus wird ein kohärenter Ansatz der kollaborativen Filterung vorgestellt.

10.1. Netzwerkzonen

Basierend auf dem Gedanken, dass Firewalls zur Erstellung von unterschiedlichen Sicherheitszonen beziehungsweise -bereichen eingesetzt werden können, ergibt sich folglich ein Abbild des Einsatzes, wie zum Beispiel in der nachfolgenden Abbildung schemenhaft zu sehen ist.

Wie aus der Abbildung ersichtlich ist, ist ein dezidiertes Firewall-Device für die Vernetzung beziehungsweise Filterung und Abschottung der unterschiedlichen Neztwerke beziehungsweise Teilsegmente dieser zuständig. So ist ein dezidiertes Interface für die Verbindung mit dem Internet abgestellt. Ebenso eines zur Weiterleitung der Verbindungen in die Subnetze (A, B, C) und zusätzlich eines für das Anbinden der DMZ. In diesem Fall werden die Interfaces durch physikalische Netzwerkadapter repräsentiert und können auf einfache Art und Weise den unterschiedlichen Bereichen zugewiesen werden. In einer virtualisierten Umgebung entspricht dies nicht dem Regelfall, da die

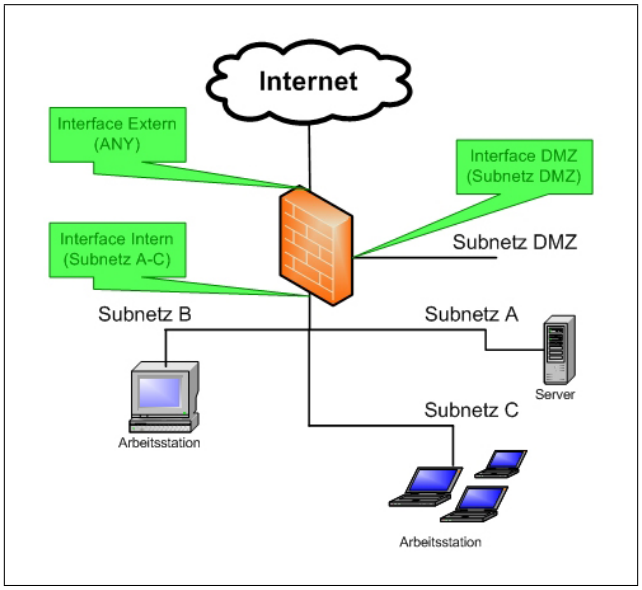

Abbildung 10.1.: Zonen des Netzwerks mit Adaptern [vgl. Net09, S. 88]

Anzahl der vorhandenen Adapter (teils virtualisiert) nicht der vorhandenen physikalischen Verbindungen entspricht. Hierbei kann die Funktion des Firewall-Devices durch den Host-Rechner wahrgenommen werden.

10.2. Elimination der Zonen

Durch die Verlegung des Netzwerkverkehrs nach innen (basierend auf der Virtualisierung) kann demzufolge eine Störung der ursprünglichen Sicherheitsmechanismen eintreten, welchen durch eine Kontrolle von Seiten des Hosts entgegengewirkt werden kann. Somit ist eine teilweise Übergabe der FW-Aufgaben an den Host zu beobachten beziehungsweise kann in diesem wahrgenommen werden. Dies wird zum Beispiel durch den Einsatz von virtuellen Netzwerkadaptern (mit den Interface-typischen Parametern) ermöglicht. Eine mögliche Ausprägung wäre durch die Verwendung von Bridges unter Linux zu nennen, da diese entweder direkt (Xen) oder nachträglich (KVM) initialisiert, parametrisiert und vernetzt werden. Diesen Bridges (xen-br0 unter Xen, br0 unter KVM) können dabei Parameter echter beziehungsweise physisch vorhandener Netzwerkinterfaces zugewiesen werden und haben die Aufgabe, das (beziehungsweise mehrere)

verfügbare Interface auch für die installierten Gäste bereitzustellen und somit eine Kommunikation nach außenhin zu ermöglichen [Fis08b]. Diese Brücken können in weiterer Folge durch spezielle Parameter[1] (zum Beispiel bei Paketfiltern basierend auf iptables mittels „-m“-Parameter) ebenso wie das hardwaremäßig vorhandene Interface einer Kontrolle unterzogen werden. Die aufwendige und resourcenintensive Externalisierung des Traffics kann somit zu einem Großteil entfallen.

Folgende Abbildung soll darüber Auskunft geben:

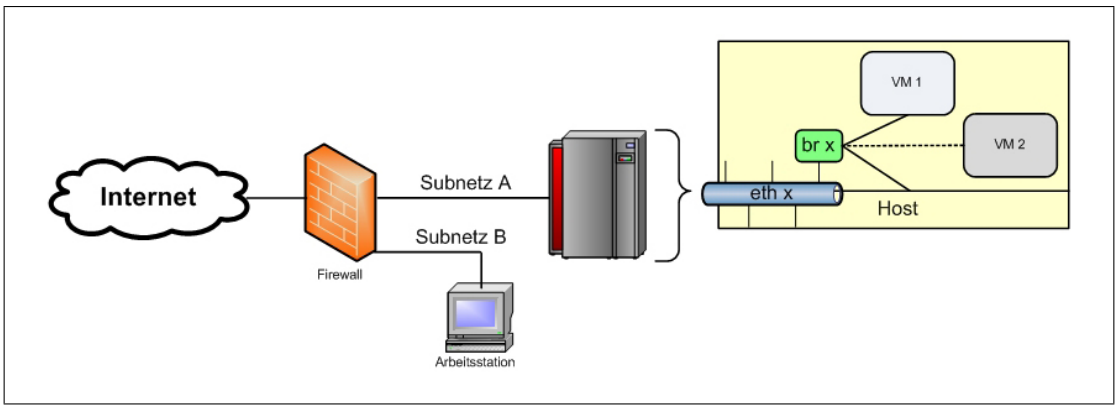

Abbildung 10.2.: (Teilweise) Aufhebung der Netzwerkzonen

Unter Berücksichtigung der Gegebenheiten (FW + VMs) können nun zwei praktikable (eigenständige) Möglichkeiten zur Filterung des Netzwerk-Verkehrs aufgezeigt werden:

Clientseitige Filterung im Gast-OS, da jede IP-Adresse einen eigenständigen (zwar virtualisierten) Rechner darstellt und eine Installation von FW-Lösungen somit nicht ausschließt.

Hostseitige Filterung im Host-OS, sofern virtuelle Netzwerkadapter beziehungsweise Bridges im Betriebssystem zur Verfügung stehen und zudem eine Filterung dieser erlauben.

Zuzüglich zu diesen Varianten bestünde die Möglichkeit, den Netzwerkverkehr nach außenhin auf ein Firewall-Device umzuleiten, da dies jedoch ein spezielles Umfeld beziehungsweise eine besondere Infrastruktur als auch einen entsprechenden Use Case voraussetzt, wird diese Möglichkeit hier nicht weiter behandelt.

[1] Wird zum Beispiel durch den FirewallBuilder, welcher im nächsten Abschnitt erläutert wird, unterstützt.

10.3. Kohärenter Ansatz

Unter dem Synonym „Kohärenter Ansatz“ soll hier die folgerichtige Abstimmung der einzelnen FW-Parameter beziehungsweise Filterregeln bezeichnet werden. Basierend auf unserer Unterscheidung in Client- beziehungsweise Host-seitige Filterung stellt dieser Ansatz eine Kombination dieser beiden Konzepte dar. Da dieser Ansatz vergleichsweise zeitaufwendig ist, zumindest in Hinblick auf die Erst-Konfiguration, kann davon ausgegangen werden, dass dieser ohne entsprechende Infrastruktur sowie Softwaretechnische Unterstützung kaum Anklang findet. Das Auffinden der richtigen Netzwerkteile, welche mittels Routing in den VMs initialisiert werden müssen, wird an dieser Stelle ausgeblendet.

Da prinzipiell eine Kommunikation der einzelnen virtuellen Netzwerkadapter untereinander möglich ist, erscheint es sinnvoll, diese Kommunikation im Host (fungierend als FW-Hub) zu reglementieren.

Die nachfolgende Grafik soll diesen Umstand der kohärenten Konfiguration der Firewall-Komponenten hervorheben beziehungsweise verdeutlichen:

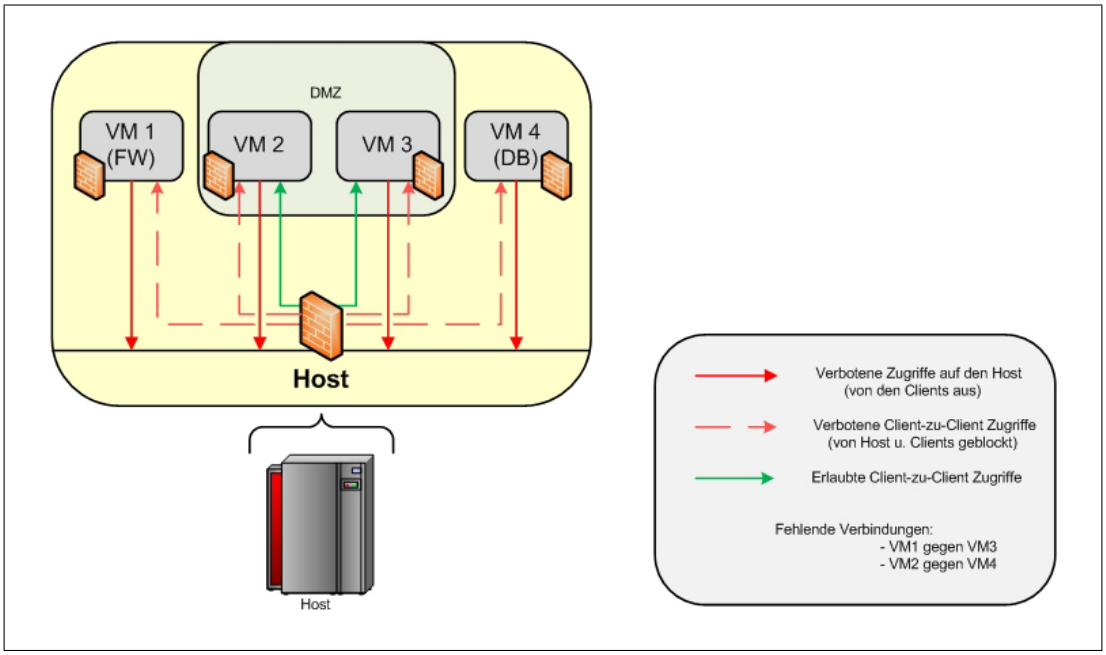

Abbildung 10.3.: Kohärente Bestimmung der FW-Regeln

Basierend auf dem Szenario, welches in der obigen Abbildung zu sehen ist, sind jedwede Zugriffe ausgehend von den Clients auf das Host-System untersagt. Dies betrifft jedoch nicht die gemeinsame Nutzung der Netzwerkschnittstelle beziehungsweise des virtuellen Interfaces, nur die Zugriffe auf das Host-OS.

Wie aus der Abbildung ersichtlich ist, ist es den virtuellen Maschinen 2 und 3 möglich miteinander zu kommunizieren, da sich diese in einer gemeinsamen DMZ befinden.

Zugriffe auf VM1, einer zu Testzwecken aufgesetzten Firewall, sind den anderen Clients untersagt. Ebenso ist der Zugriff auf VM4, welche einen Datenbank-Server repräsentiert, untersagt, da dieser nur aus dem internen Netzwerk heraus angesprochen werden kann, nicht jedoch vom Test-Firewall als auch vom DMZ-System heraus.

Alle erstellten Regeln, sei es client- aber auch hostseitig, werden auf die konkrete Situation abgestimmt und stehen somit im Einklang mit den anderen Akteuren.

Als konkretes Beispiel sei hier das Verhältnis zwischen VM1 und VM2 dargestellt. Die Firewall-Policy der Clients enthält dabei folgende Regeln:

- Geblockte Verbindung zum Host
- Geblockte Verbindung zum jeweils anderen Client
- Geblockte Verbindung zum internen Netz

Die Firewall-Policy des Hosts enthält dabei folgende Regeln:

- Geblockte Verbindung zu den Clients
- Geblockte Verbindung zwischen den konkreten Clients
- Geblockte Verbindung zum internen Netz

Ähnlich verhält es sich mit den Beziehungen der anderen Clients mit der Ausnahme, dass VM2 und VM3 in einer gemeinsamen Zone (der DMZ) miteinander kommunzieren können beziehungsweise dürfen. Aus Gründen der Übersichtlichkeit wird auf eine weitere Darstellung dieser verzichtet.

11. Firewalls und VM-Clusters

Als *VM-Cluster* werden wir in weiterer Folge den Verbund von mehreren VMs zur Erledigung einer gemeinsamen Aufgabe bezeichnen. Die zu erledigende Aufgabe kann unterschiedlicher Natur sein und unterliegt somit diversifizierten Motiven ebenso wie deren Einsatz. So können Cluster gebildet werden um eine Leistungssteigerung zu erzielen als auch um eine hohe Verfügbarkeit gewisser Dienste beziehungsweise Applikationen sicherzustellen.

Eine genaue Darstellung der Aufgaben sowie der Funktionsweise von Cluster soll hier ausgeklammert werden, da die Vernetzung der einzelnen VMs sowie dessen technische Hintergründe den Inhalt dieses Kapitels darstellen sollen. Zusätzlich soll der Fokus auf den in der VM einzustellenden Parameter liegen, wobei die Bereitstellung der Infrastruktur vernachlässigt wird.

11.1. Art der Vernetzung

Obwohl eine Vernetzung auf der von der Host-Maschine zugewiesenen IP-Adresse im Intranet möglich wäre, so wollen wir dennoch davon ausgehen, dass die Ausgangslage eine verteilte Infrastruktur mit unterschiedlichen Netzen und IP-Bereichen darstellt. Die Verbindung sowie deren -aufbau soll über das öffentliche Internet erfolgen können, weshalb eine Verwendung eines VPNs[1] als sinnvoll erscheint. Zusätzlich zu diesen Kriterien definieren wir die Prämisse, dass eine Integration beziehungsweise vollständige Vernetzung der unterschiedlichen Teilnetze nicht angestrebt wird.

Dieses Szenario entspricht im Wesentlichen den Anforderungen beziehungsweise dem Schemata des Extranet-VPNs, welches auf der Basis von normaler VPN-Technologie aufgebaut wird. Der Unterschied zu anderen VPN-Typen, wie zum Beispiel Remote-Access oder Branch-Office-VPNs, liegt in der unterschiedlichen Natur der Teilnehmer

[1] Virtuelles Privates Netzwerk

im VPN begründet. Wohingegen die zuvor genannten Typen von VPNs auf eine dezidierte private Vernetzung der Standorte aufbauen und abzielen. So wird es durch den Einsatz eines Extranet-VPNs möglich, dieses private Netzwerk zu öffnen und somit anderen Teilnehmern verfügbar zu machen.

Dies bedingt natürlich eine besondere beziehungsweise genauere Behandlung der nicht vertrauenswürdigen Pakete, welche entweder durch einen VPN-Gateway oder aber durch eine Firewall überprüft werden müssen. Zumeist werden Firewalls zur Termination der VPN-Verbindungen verwendet, da diese eine hinreichend sichere Filterung der Pakte erlaubt und gleichsam für die Zugriffsbeschränkung als auch das Logging zuständig ist.

Sicherheitstechnisch wird durch den Einsatz dieser Variante häufig auf den Verzicht des *gehärteten Stacks*[2] der VPN-Router hingewiesen, was in weiterer Folge zu Risiken führen kann [Lip07].

11.2. SSL-VPNs

Obwohl es eine Vielzahl an Möglichkeiten gibt, das zuvor beschriebene Szenario zu implementieren und in die Realität umzusetzen, wird hier nur die Variante der SSL-VPNs beschrieben. Dies ist dadurch begründet, dass zum einen frei verfügbare (Client-)Lösungen[3] kostenfrei bezogen werden können, als auch zum anderen eine ausreichende Sicherheit gewährleistet werden kann.

Auf eine Erläuterung der technischen Hintergründe von SSL wird bewusst verzichtet und auf die einschlägige Literatur verwiesen[4] da dies den Umfang der Arbeit sprengen würde.

Obwohl SSL-VPN in unterschiedlichen Modi betrieben werden können, konzentrieren wir uns auf die Client-based SSL-VPNs, welche unter gewissen Voraussetzungen eine

[2] Gehärteter Netzwerk-Stack des öffentlichen Interfaces, welches nur wenige Protokolle unterstützt, diese aber sehr gut vor Angriffen schützen kann.

[3] zum Beispiel: OpenVPN welches auf SSL-Basis arbeitet und aus dem entsprechenden Repository zum Beispiel unter Ubuntu installiert werden kann. [Fis08a] (Online Bezugsquelle: `http://openvpn.net/`)

[4] zum Beispiel: [Tan02] - Tanenbaum, Andrew S.: Computer Networks. Prentice Hall International, 4. Auflage, August 2002.

Transparenz gegenüber von Applikationen ermöglichen. Für gewöhnlich handelt es sich hierbei um Programme, welche auf der Ebene von Sockets oder Adapter arbeiten und somit eine Tunnelung von TCP, UDP und teils auch IP-Paketen ermöglichen.

Besonders interessant ist die Variante, welche auf Adapter-Ebene fungiert und dadurch auch eine Kapselung von IP-Paketen ermöglicht, ohne die Applikationsebene zu beeinträchtigen beziehungsweise für diese sichtbar zu sein. Der Umstand, dass hierbei nur eine SSL-Session mit einer Connection zum SSL-VPN-Gateway verfügbar ist, wird jedoch vielfach als Einbuße an Flexibilität wahrgenommen.

Folgende Grafik soll dies verdeutlichen:

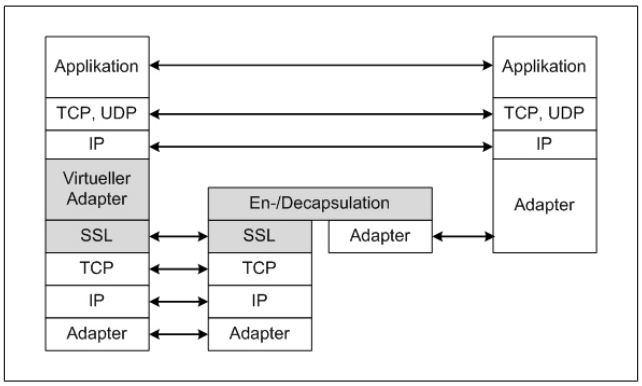

Abbildung 11.1.: Transparenter SSL-Client auf Adapterbasis [vgl. Lip07, S. 287]

Hinsichtlich der Performance sei festgehalten, dass die Ein-/Entkapselung einen gewissen Overhead darstellt. Dieser wird jedoch erst kritisch bei der Verwendung von Protokollen, wie UDP als auch das darauf aufbauende RTP (Real Time Protocol) und damit verbundenen Anwendungen, welche auf einen schnellstmöglichen Datentransfer abzielen. Zu solchen Anwendungen zählen etwa das Voice over IP (VoIP) als auch das Video Conferencing.

Sicherheitstechnisch soll an dieser Stelle darauf hingewiesen werden, dass SSL an sich eine Reihe von Sicherheitsschwächen oder auch protokollimmanente Angriffspunkte aufweist, welche jedoch in den meisten Fällen in den konkreten Implementierungen behoben wurde, ohne die Kompatibilität zu anderen Lösungen zu beeinträchtigen. Zu diesen zählen etwa der Bleichenbacher-Angriff, Cipher-Suite-Rollback-Angriff, Version-Rollback-Angriff als auch der Key-Exchange-Algorithm-Rollback-Angriff [Lip07].

Darüber hinaus sind auch konkrete Fehler und damit verbundene Schwachstellen der einzelnen Implementierungen zu erwähnen, welche bei der Auswahl der einzusetzenden Technologie bewusst abgewogen werden sollten.

11.3. Firewallspezifische Einstellungen

In diesem eher pragmatischen Subkapitel wenden wir uns zu Anschauungszwecken der frei verfügbaren VPN-(Software-)Lösung OpenVPN zu, welche auf unserer VM zum Einsatz kommen soll [PW07]. Somit ist eine Betrachtung aus der Sicht des Clients vorherrschend. Die Erstellung der Zertifikate und ähnliches, welches nicht in Zusammenhang mit der Firewallkonfiguration steht, wird ausgeblendet. Die Konfiguration anderer Lösungen ist dementsprechend zu modifizieren (Portnummern, Hosts, ...).

Ausgehend von einer korrekten Parametrierung beziehungsweise Konfiguration der Gegenstellen, sprich des Clients und des Servers[5], wird ohne die entsprechenden Erweiterungen der Firewall-Parameter eine Verbindung beziehungsweise -versuch fehlschlagen. Dies kann mehrere Gründe haben, wobei eine Vielzahl der Fehlerquellen durch die Analyse der Infrastruktur leicht ersichtlich gemacht werden können. So kann das Fehlen einer entsprechenden Konfiguration eines zwischengeschalteten NAT bereits dramatische Auswirkungen haben und eine korrekte Weiterleitung der Pakte verhindern.

Ausgehend von dem Wissen, dass OpenVPN in der Standardkonfiguration UDP zur Datenvermittlung einsetzt und hierzu der Port 1194 verwendet wird, können die entsprechenden Forwardings durch die Infrastruktur bis zur Gegenstelle weitergeleitet werden [Ope09b].

Ist dies vollbracht, so müssen die Firewalleinstellungen der einzelnen VMs an die Verbindungseinstellungen (Port, Protokoll) der konkreten Implementierung (OpenVPN) angepasst werden.

Ebenso müssen etwaige installierte Firewall-Rules der Host-Maschine an die geänderten Umstände angepasst werden. Prinzipiell ist jedoch davon auszugehen, dass die einzelnen VMs gegenüber dem Netzwerk als eigenständige Rechner fungieren.

[5] Nach der Herstellung der Verbindung sind die Gegenstellen gleichberechtigt.

Ausgehend von den zuvor genannten Ports und Protokolldaten lässt sich die Verbindungsfreigabe folgendermaßen als iptables anschreiben:

- Zuerst soll es einmal möglich sein, die Verbindung mittels PING testen zu können.

```
# Zulassen von PING-Requests
iptables -A INPUT -p icmp -icmp-type echo-request -j ACCEPT
```

- Nun soll auf UDP Port 1194 ein Listening möglich sein, um ggf. einen Tunnel aufbauen zu können.

```
# Freischalten des UDP Ports zur Tunnelherstellung
iptables -A INPUT -p udp -dport 1194 -j ACCEPT
```

Falls bekannt, könnte hier mit der Option -s noch die IP-Adresse der Gegenstelle determiniert werden, was jedoch bei der Verwendung von OpenVPN im *Secure Mode* (zum Beispiel mittels HMAC-based Authentication) entfallen kann.

Ein Vervielfältigen dieses Eintrags unter Eintragung anderer Portnummern, ermöglicht gleichsam den Aufbau mehrerer Tunnel.

- Abschließend muss nun noch eine Kommunikation (IP Verkehr) über die virtuellen Netzwerk-(Kernel-)Driver, wie sie OpenVPN nutzt, freigeschalten werden.

```
# Freischalten der TUN/TAP Geräte
iptables -A INPUT -i tun+ -j ACCEPT #tun input
iptables -A FORWARD -i tun+ -j ACCEPT #tun forward
iptables -A INPUT -i tap+ -j ACCEPT #tap input
iptables -A FORWARD -i tap+ -j ACCEPT #tap forward
```

Die vorliegenden iptabel-Einträge stellen jedoch eine minimale Einstellung dar und können zum Beispiel um Stateful Package Inspection oder Ähnliches erweitert werden, falls dies benötigt wird.

12. Aktuelle Forschungsthemen

Im nachfolgenden Kapitel werden aktuelle Forschungsthemen sowie deren Ergebnisse vorgestellt. Anfangs werden Methoden zur Erkennung von virtuellen Umgebungen durch Programme, welche in dieser agieren, vorgestellt. Weiters werden aktuelle Probleme bei der Verwendung von Prozessorvirtualisierungserweiterungen in Verbindung mit VM-basierten Rootkits behandelt. Anschließend wird eine Möglichkeit zur Verlagerung der Intrusion Detection Funktionalität vom Gast- in die Host-Umgebung vorgestellt. Abschließend wird der Sinn sowie Nutzen von transparenten Netzwerkdiensten mittels einer virtuellen Traffic Schicht für VMs erläutert.

12.1. Erkennung virtueller Umgebungen

Dieses Kapitel behandelt spezialisierte Möglichkeiten zur Erkennung von virtuellen Maschinen Umgebungen (VMEs[1]) aus Gast-Systemumgebungen. Ausgehend von der Motivation um eine Erkennung durchzuführen, sollen darüber hinaus zwei konkrete Möglichkeiten zur Realisierung von dieser vorgestellt werden.

12.1.1. Motivation

Die Möglichkeit eines Programmes, festzustellen, ob dieses in einer virtuellen Umgebung arbeitet oder nicht, kann unter Umständen dazu benutzt werden, um dessen Laufzeiteigenschaften maßgeblich zu verändern [CLS07]. Dies ist zum Beispiel für Schadsoftware (sogenannte Malware) von Bedeutung, welche sich der Erkennung zu entziehen versucht. Die Verwendung von virtuellen Maschinen zur Analyse von Schadcode ist weithin bekannt, weshalb eine Anpassung des Verhaltens zu einem fehlerhaften Analyseergebnis führen kann und somit ein Erkennen der Malware beziehungsweise von deren Ausprägungen erschwert.

[1] Virtual Machine Environments

Ebenso kann sich ein Angreifer diesen Umstand zu Nutze machen und ein „fingerprinting“ der Umgebung durchführen. Die Ergebnisse dieser Analyse können in weiterer Folge zur gezielten Suche nach Exploits für die verwendete Plattform angewandt werden. Dies kann im Extremfall zu einem erweiterten Angriff gegen andere Gäste oder aber auch gegen den Host selbst führen.

12.1.2. Erkennungsmethoden

Die nachfolgend vorgestellten Methoden setzen den Einsatz von VMware Produkten voraus und können nur zur Erkennung von VMEs dieses Typs eingesetzt werden.

VMware Kommunikationskanal

Der VMware Kommunikationskanal stellt ein Instrument zur Verständigung zwischen dem Host- und Gast-OS dar und ist in den meisten VMware Produkten fest implementiert [CLS07]. Das Vorhandensein dieses Kanals beziehungsweise des damit verbundenen Wertes („VMXh“) im Speicher lässt somit Rückschlüsse auf die Verwendung in einem VME zu.

Folgendes Assembler Snippet soll die Einfachheit der Analyse verdeutlichen [CLS07]:

```
MOV EAX, 564D5868 <- entspricht dem Wert „VMXh“
MOV EBX, 0
MOV ECX, 0A
mov EDX, 5658 <- entspricht dem Wert „VX“
in EAX, DX
CMP EBX, 564D5868 <- Vergleich ob VMWare eingesetzt wird
```

Im ersten Schritt wird der Variable EAX der hexadezimale Wert 564D5868 zugewiesen (entspricht VMXh in ASCII), welcher für die Adressierung des Kanals verantwortlich ist. Nachfolgend wird EBX auf den Wert null gesetzt, da dies nachfolgend das Ergebnis übergeben bekommt. Im dritten Schritt wird der Variable ECX der hexadezimale Wert 0A (entspricht 10 in ASCII) zugewiesen, welcher die durchzuführende Aktion bestimmt. Dieser Wert signalisiert, dass eine Versionskontrolle ausgeführt werden soll. Im

vierten Schritt wird mittels Setzen des Wertes EDX der zu verwendende Port festgelegt. Anschließend wird mittels parametrisiertem IN-Kommando ein I/O-Aufruf initiiert. Bei Vorhandensein eines VMware Produkts wird jedoch nicht wirklich eine I/O-Aufruf gestartet, sondern nur der uns bereits bekannte Wert 564D5868 ins Register EBX geschrieben. Im letzten Schritt muss somit nur mehr überprüft werden, ob diese Werte identisch sind. Ist dies der Fall, ist somit erwiesen, dass der Aufruf des Programms aus einem VMware Gast getätigt wurde.

Globale Speicherbelegung

Eine weitere Möglichkeit zur Feststellung einer (Programm-)Ausführung in einer virtuellen Gastumgebung stellt die Analyse der Position von globalen Variablen im Speicherbereich dar. Dieser Umstand wird durch die Aufteilung des vorhandenen Speichers auf den Host, als auch auf die Gäste möglich. Eine ungewöhnlich hohe Speicheradresse für spezielle Beschreibungstabellen[2] lässt somit entsprechende Rückschlüsse zu.

Eines der ersten Tools, welche sich diesen Umstand zur Analyse zu nutze machte war das von Joanna Rutkowska veröffentlichte Programm „Red Pill“ [CLS07].

Das Aufkommen von Multikern-Prozessoren und der damit verbundenen Speicheraufteilung führt jedoch zu einer hohen Fehlerrate dieser Methode, weshalb diese hier nicht näher erläutert werden soll. Für weitere Informationen zu dieser Thematik wird an dieser Stelle auf die einschlägige Literatur verwiesen[3].

12.2. VM-basierte Rootkits (VMBRs)

Die zunehmende Verbreitung von Virtualisierungstechnologien sowie insbesondere von Prozessoren, welche Hardwareunterstützung für diese zur Verfügung stellen, hat neben der erwünschten Verbesserung der Performance auch die Möglichkeit zur Implementierung einer neuen Generation von Rootkits eingeleitet. In diesem Zusammenhang wird von sogenannten VM-basierten Rootkits (kurz VMBRs) gesprochen.

[2] Zum Beispiel der Interrupt Descriptor Tabelle, Global Descriptor Tabelle und Local Descriptor Tabelle.

[3] Zum Beispiel: [CLS07] – Carpenter, Matthew, Tom Liston und Ed Skoudis: Hiding Virtualization from Attackers and Malware. IEEE Security and Privacy, 5(3):62-65, 2007.

Das Grundprinzip der VMBRs besteht im Einfügen eines „bösartigen“ Hypervisors unterhalb des Betriebssystems, um mittels Ausnutzung der Virtualisierung ein Erkennen unmöglich zu machen [CZL08]. Die minimale Größe des Hypervisors, sowie die Möglichkeit, das Einfügen „on the fly“ (auch ohne Reboot) durchführen zu können, ermöglicht ein unbemerktes Eindringen. Gegenwärtige Prototypen wie zum Beispiel SubVirt, Vitriol und Blue Pill beweisen die Durchführbarkeit eines solchen Angriffes.

Als Grundlagen für solche VMBRs seien folgende Kriterien genannt [CZL08]:

- Vorhandensein eines Prozessors, welcher Hardwaresupport für Virtualisierung bietet.
- Nichtverwendung der Hypervisorschicht zur Virtualisierung.

Ebenso muss eine Möglichkeit bestehen, feststellen zu können, ob ein Betriebssystem in einer virtualisierten Umgebung läuft oder nicht. Dies stellt jedoch, wie aus dem vorigen Kapitel ersichtlich, eine nicht allzu große Hürde dar.

Vorschläge zur Lösung dieses Problems reichen vom Laden eines minimalen präventiven Hypervisors, bis hin zur Integration eines voll ausgestatteten VMM beim Systemstart. Da beide Ansätze einen relativ simplen Zugang zum Problem darstellen und zudem mit teils enormen Nachteilen verbunden sind, werden diese hier außer Acht gelassen[4]. Eine dritte Möglichkeit würde die Integration eines Sicherheitshypervisors unterhalb der Schicht des normalen Hypervisors (von Drittanbietern) darstellen. Die Realisierung dieser zusätzlichen Sicherheitsschicht könnte zum Beipsiel mittels Emulation der CPU eigenen Virtualisierungserweiterung erreicht werden. Eine solche Realisierung hätte zudem den großen Vorteil, dass die von Drittanbietern bereitgestellten VMMs unmodifiziert zur Anwendung kommen könnten.

Für weitere Informationen zu diesem Thema sowie einer konzeptionellen Lösung (GuardHype) des Problems sei auf den Artikel von Carbon und Lee aus dem Jahr 2008 (siehe [CZL08]) verwiesen.

[4] Zum Beispiel würde die Verwendung des Minimal-Hypervisors eine Verwendung der Virtualisierung, aufbauend auf der Hardwareunterstützung, komplett unterbinden. Die Verwendung des voll ausgestatteten Hypervisors hingegen würde eine Vertrauensstellung von Seiten der Drittanbieter, welche darauf aufbauende Komponenten bereitstellen müssten, implizieren, welche im Widerspruch zu den eigenen Entwicklungen steht.

12.3. Verlagerung von Intrusion Detection Systemen

Ähnlich zu anderen sicherheitstechnischen Applikationen, wird die Realisierung von Intrusion Detection Systemen (kurz IDS) im Gastsystem aufgrund ihrer direkten (lokalen) Angreifbarkeit im Falle eines erfolgreichen Angriffes nicht als Ideallösung angesehen. Die Grundfunktionalität der IDS, nämlich die Analyse von Netzwerkdaten um auffällige Aktivitäten zu erkennen, wäre somit nur bedingt gewährleistet. Die nachfolgenden Erläuterungen beziehen sich auf die Host-basierten Intrusion Detection Systeme.

12.3.1. Motivation

Ursprünglich gibt es zwei Einsatzorte für die Implementierung von IDS. Zum einen auf einer eigens dafür präparierten und speziell abgesicherten phyiskalischen Maschine[5] und zum anderen den Host-basierten Ansatz[6], bei dem das IDS lokal in der zu sichernden Plattform arbeitet. Mit dem Einsatz von HIDS sind generell die Nachteile der Notwendigkeit zur Installation in jedem Host sowie die direkte Einflussnahme eines Angreifers bei erfolgreicher Penetration zu nennen. Die Verwendung von VME kann die Sicherheit solcher Systeme maßgeblich erhöhen, da diese keine, beziehungsweise eine geringere Angriffsfläche bieten [LMJ07]. Durch die Auslagerung dieser Systeme kann somit die Möglichkeit einer direkten Manipulation ausgeschlossen werden. Ebenso wird die sonst dezentrale Ausprägung (in den Gästen) zentralisiert, wodurch ein enormer Informationsgewinn, aufbauend auf der kollaborativen Betrachtung aller Gäste, ermöglicht wird [LMJ07].

12.3.2. Realisierungsvorschlag

Die von Laureano, Maziero und Jamhour in [LMJ07] vorgeschlagene Architektur zur Realisierung dieser zentralen IDS bedingt eine Anpassung des VMM. Die vorzunehmenden Anpassungen betreffen vorwiegend die fehlende Funktionalität zur Überwachung der Aktivitäten der einzelnen Gäste. Die Vorteile bei der Anpassung des VMM liegen in der für die Gäste transparenten Integration des HIDS (da auf Host-Ebene) sowie das

[5] Diese sogenannten Netzwerk-basierenden IDS (kurz NIDS) überwachen den Netzwerkverkehr und versuchen Anomalien zu erkennen.

[6] Auch Host-basierte IDS (kurz HIDS) überwachen die lokalen Aktivitäten wie zum Beispiel Prozesse, Netzwerkverbindungen und Systemaufrufe.

Wegfallen der direkten Angreifbarkeit dieses Überwachungsinstruments.

Folgende Abbildung soll dieses Konzept verdeutlichen:

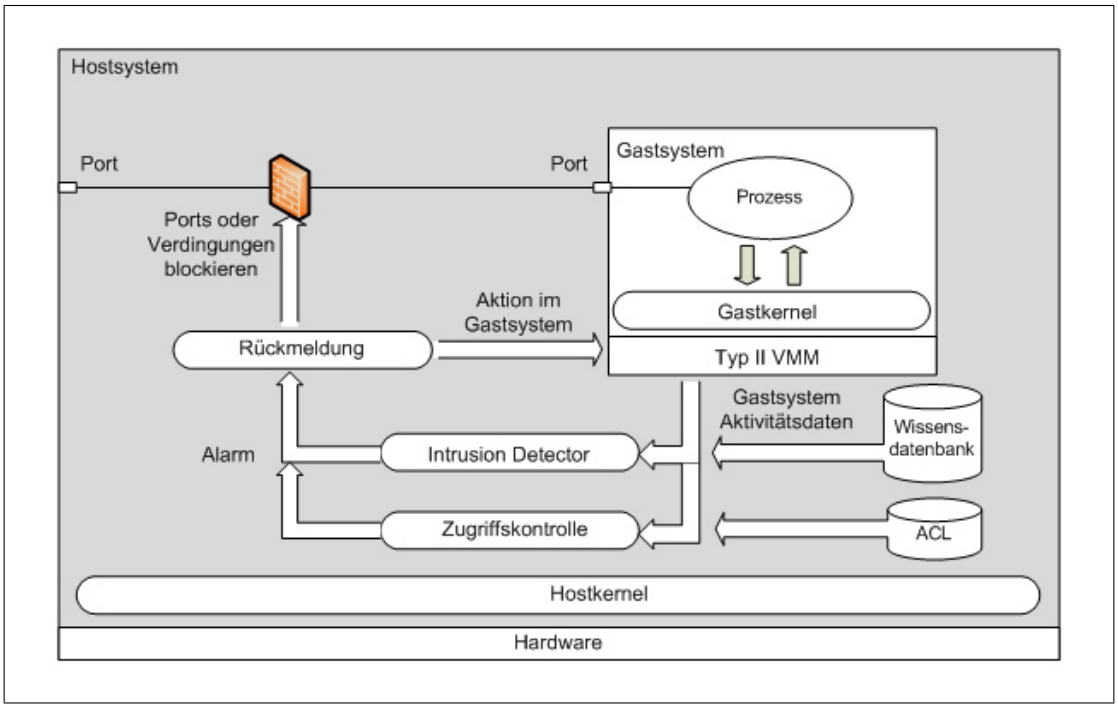

Abbildung 12.1.: Konzeptionelle Architektur [LMJ07]

Wie aus der Abbildung ersichtlich, sind die Hauptmodule des Konzepts das Intrusion Detection- , Zugriffskontroll- sowie Rückmeldemodul. Dem Intrusion Detection Modul kommt die Aufgabe zu, die gesammelten Daten des Gastsystems mit der verfügbaren Wissensdatenbank zu vergleichen. Das Zugriffskontrollmodul hat die Aufgabe, sicherzustellen, dass die Prozesse sowie die Benutzer bekannt und legitimiert sind, bestimmte Aktionen durchzuführen. Dem Rückmeldemodul kommt bei entsprechender Alarmmeldung von einem der anderen beiden Module die Aufgabe zu, entsprechende Aktionen auf dem Gastsystem durchzuführen. Ebenso ist es dafür verantwortlich, gegebenenfalls die Regeln der Hostfirewall[7] anzupassen, um die Sicherheit der Systeme gewähren zu können.

Die Verknüpfung zwischen dem Gastsystem und den beiden Kontrollmodulen wird durch den modifizierten VMM hergestellt, welcher die Informationen über die Aktivitäten des

[7] Zum Beispiel die Anpassung des iptables Regelwerks der Hostmaschine.

Gastsystems weiterreicht und gegebenenfalls die Aktionen des Rückmeldemoduls umsetzt.

Sowohl das Intrusion Detector Modul als auch das Zugriffskontrollmodul können als einfache Prozesse im Userspace des Hosts implementiert werden. Dies hat den enormen Vorteil, dass zusätzliche Funktionalität auf einfache Weise nachgerüstet werden kann [LMJ07].

Laut Meinung des Autors ist die Modifikation des VMM bedingt für die zu erzielende Funktionalität geeignet, da die Veröffentlichung einer neuen Version dessen, eine erneute Modifikation bedingen würde. Deshalb wäre, wo möglich, die Erstellung einer Sicherheits-API für diese Zwecke eine adäquate Lösung und hätte zudem den Vorteil des niedrigen Performanceverlustes im Gegensatz zu anderen Lösungen.

Für weitere Informationen zu diesem Thema sowie für die Darstellung eines Prototypen sei auf den Artikel von Laureano, Maziero und Jamhour [LMJ07] verwiesen.

12.4. Transparente Netzwerkdienste

Die Forschungs- und Entwicklungsarbeit auf dem Gebiet der transparenten Netzwerkdienste beschäftigen sich mit der Erstellung von Netzwerkdiensten, welche ein unmodifiziertes Verwenden von Applikationen (aufbauend auf diesen) innerhalb einer VME ermöglichen. Zusätzlich soll das Vorhandensein dieser Dienste den entsprechenden virtuellen Instanzen (Gästen) nicht bekannt sein. Das nachfolgende Kapitel soll über den derzeitigen Entwicklungsstand sowie den Nutzen solcher Dienste informieren.

12.4.1. Motivation

Wie bereits aus der vorliegenden Arbeit hervorgegangen, sind Programme und Dienste, welche sich im Gast-OS befinden, durch einen Angriff unmittelbar zu beeinflussen. Der Ansatz von transparenten Netzwerkdiensten versucht, eine zusätzliche Sicherungsschicht („Virtual Traffic Layer“) für den Netzwerkverkehr zu schaffen. Diese virtuelle Schicht kann trotz einer Kompromitierung des Gastsystems eine effektive Kontrolle gewährleisten, da diese im Hostsystem implementiert ist [LD07].

Durch die Verwendung einer solchen virtuellen Schicht können zum Beispiel folgende Funktionalitäten realisiert werden:

- Traffic Monitoring
- Kontrolle des Routings
- Datenmanipulation

Diese Punkte implizieren die Fähigkeit zur (Paket-)Manipulation auf den Ebenen der Sicherungs-, Netzwerk-, Transport- sowie teilweise Anwendungsschicht [LD07]. Des weiteren werden durch diese Manipulationsfähigkeit sowohl die eingehenden, als auch die ausgehenden Pakete einer VM kontrollierbar beziehungsweise manipulierbar. Dies kann in weiterer Folge zur Implementierung einer Host-basierten Firewall Verwendung finden.

12.4.2. Realisierung

Zur Realisierung der zuvor genannten Funktionen kann zum Beispiel das „Virtual Traffic Layer"-Framework (kurz VTL) zur Hilfe genommen werden [LD07]. Dieses wurde ursprünglich zur einfachen Erstellung von transparenten Netzwerkdiensten implementiert und stellt grundsätzlich APIs zur Implementierung dieser zur Verfügung. Diese Programmierschnittstellen werden in weiterer Folge zur Erstellung von Modulen, welche bestimmte Dienste abbilden, weiterverwendet. Entgegen der Vielzahl erläuterter Dienste[8] wird sich dieses Kapitel auf die von Lange und Dina (siehe [LD07]) vorgestellte VTL basierende Stateful Firewall beschränken.

Die Verwendung eines transparenten Netzwerkdienstes zur Realisierung einer Firewall hat den enormen Vorteil, dass diese außerhalb der VM zur Anwendung kommt und somit von dieser unabhängig agieren kann. Die Bereitstellung der Firewall als Dienst hat zudem den Vorteil, dass diese unabhängig von den systemeigenen Sicherheitsmechanismen, welche primäre Angriffsziele darstellen, agieren und zudem beliebig erweitert werden kann, ohne zukünftige Kompatibilitätsprobleme befürchten zu müssen. Um dies zu ermöglichen, muss es prinzipiell möglich sein, sowohl Netzwerkpakete abzufangen, als auch wieder in den Paketstrom einzufügen. Zusätzlich muss es möglich sein, Pakete zu inspizieren, als auch zu modifizieren. Diese vier elementaren Operation werden durch die

[8] Zum Beispiel ist auch ein Dienst zur Erkennung von Angriffsmethoden, basierend auf einer kollaborativen Basis des VM-Netzwerkverkehrs, beschrieben (Cooperative Selective Wormholing - CSW).

Netzwerkinterface- und die Paketzugriffs-API des VTL Frameworks bereitgestellt.

Die Einfachheit einer möglichen Implementierung soll anhand folgenden Codebeispiels, welches auf API-Funktionen zurückgreift, veranschaulicht werden [LD07]:

```
RawEthernetPacket pkt;
unsigned long dst, new_dst;

dst = *(uint32 *)IP_DST(pkt.data);
*(uint32 *)IP_DST(pkt.data) = new_dst;

dst = GET_IP_DST(&pkt);
SET_IP_DST(&pkt, new_dst);
```

Das vorliegende Beispiel zeigt den Zugriff auf grundlegende Attribute des Paketkopfes (hier die IP-Adresse des Empfängers).

12.4.3. Leistungsmerkmale

Basierend auf dem technischen Charakter der Erweiterung (sie läuft im User Space von Xen) ist mit Performanceverlusten zu rechnen. Die nachfolgende Abbildung soll diesen Umstand verdeutlichen:

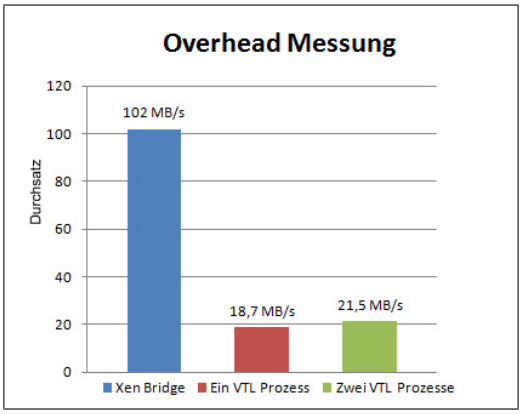

Abbildung 12.2.: Leistungsmessung VTL Einsatz [LD07]

Die Abbildung beinhaltet zum einen den Netzwerkdurchsatz der Xen-Bridge in der Standardkonfiguration sowie zwei Messergebnisse unterschiedlicher VTL Module (Testmodule). Das erste VTL Modul bildet einen Prozess im Halbduplex-Betrieb ab, wohingegen das zweite Modul zwei Prozesse im Vollduplex-Betrieb zeigt (beide im User Space ausgeführt).Wie aus der Abbildung ersichtlich, ist mit einem hohen Bandbreitenverlust zu rechnen. Um diese Verluste einzudämmen, wäre es zum Beispiel möglich, die Prozesse in den Dom0 Kernel von Xen zu übernehmen. Darüber hinaus kann davon ausgegangen werden, dass die gegenwärtig (ohne Optimierung der Prozesse) gezeigte Geschwindigkeit für WAN Umgebungen ausreichend ist [LD07].

12.4.4. Voraussetzungen

Der Einsatz von VTL erfordert die Konfiguration von „host-only“ Netzwerkinterfaces innerhalb der virtuellen Maschinen. Nur so kann sichergestellt werden, dass VTL den gesamten Netzwerkverkehr empfängt und diesen kontrollieren sowie manipulieren kann. Dies bedingt auch die Möglichkeit, dass VTL den virtuellen Netzwerkadaptern des VMM angefügt werden kann. Dies kann zum Beispiel mittels der libpcap sowie der libnet Bibliothek (unter Unix) realisiert werden[9].

Für weitere Informationen zu diesem Thema sei auf den Artikel von Lange und Dinda aus dem Jahr 2007 (siehe [LD07]) verwiesen.

[9]Für den Einsatz unter Windows wäre dementsprechend Winpcap zu verwenden.

13. Firewall Policies mit dem FirewallBuilder

In diesem abschließenden Kapitel soll eine konkrete Software-Realisierung eines Firewall-Policie-Tools, welches zudem die Aufgaben eines Firewall-Management-Tools wahrnimmt, dargestellt werden. Ebenso ist eine kritische Betrachtung der Unterstützung von virtuellen Maschinen durch die Software Teil dieses Kapitels. Unter der Betrachtung der Verwendbarkeit der Software soll zudem dargestellt werden, wie eine externe (experimentelle) Erweiterung die vorhandenen Problembereiche beziehungsweise Defizite lösen könnte.

13.1. Grundlagen

Der *Firewall Builder* kann als Software-Lösung (mit GUI) zur Firewall-Konfiguration, und als Management-Instrument auf breiter Basis beschrieben werden. Das Management von Firewalls mittels eigener Policies kann als Zentralaufgabe der Software verstanden werden, wobei auf eine Unabhängigkeit gegenüber diverser Lieferanten geachtet wird. Eine Remote-Wartung beziehungsweise Aktualisierung der Clients mittels SSL-Verbindung ermöglicht somit ein komfortables Eingreifen in die Netzwerk-Struktur beziehungsweise deren Filterung und Sicherung.

Zu den unterstützen Technologien zählen etwa [Kur08]:

- iptables (netfilter)
- ipfilter
- pf
- ipfw

- Cisco PIX (FWSM, ASA)
- Cisco routers extended access lists

Der Firewall Builder ist unter zwei verschiedenen Lizenzmodellen (Dual-Licensing-Modell) zu beziehen, wobei die Lizenz durch den Einsatz des jeweiligen Betriebssystems determiniert wird. Für freie Betriebssysteme, welche ihrerseits unter der GPL[1]-Lizenz verfügbar sind (zum Beispiel Linux), ist auch der Firewall Builder unter dieser Lizenz erhältlich. Bei kommerziellen Betriebssystemen (zum Beispiel Microsoft Windows) fällt der Einsatz des Firewall Builders unter das NetCitadel EULA[2].

Der FWBuilder (Version 3.0.5) kann unter folgender URL für diverse Betriebssystem-Plattformen bezogen werden: `http://www.fwbuilder.org/`.

Zu den unterstützten OS, auf denen der Firewall Builder eingesetzt werden kann, zählen etwa FreeBSD, Linux, Mac OS X, OpenBSD und Windows.

Ohne Anspruch auf Vollständigkeit zu erheben, seien noch die IPv6 Unterstützung sowie die Möglichkeit zur zentralen Verwaltung und Steuerung von Firewalls als interessante Funktionen der Implementierung hervorzuheben. Der objektorientierte Ansatz soll zudem ein einfaches Handling von Policies und ähnliches sicherstellen. Auch ist die Erstellung von Policies auf einer abstrakten Ebene vom Anwender durchzuführen und wird erst durch das Compilieren (mittels der Policy Compiler) in die entsprechende Form übertragen, welche für die entsprechende Plattform beziehungsweise Technologie benötigt wird. Anschließend können die so erstellten Regeln „remote“ an die entsprechende Firewall übertragen werden.

Für eine ausführlichere Darstellung der Funktionalität, als auch der vorhandenen Features sei an dieser Stelle auf den offiziellen User´s Guide verwiesen, welcher unter `http://www.fwbuilder.org/docs/UsersGuide3.pdf` zur Verfügung steht.

13.2. Unterstützung von VMs

Generell sieht die auf Objekte (Server, Interface, ...) ausgelegte Umgebung des FW-Builders keine Unterstützung von virtuellen Maschinen vor. Dies bedeutet auch, dass

[1] GNU Public License
[2] End User License Agreement

für diese Einsatzzwecke kein gesondertes beziehungsweise spezielles Template zur Verfügung steht. Ebenso sucht man ein Integrations- beziehungsweise Erstellungstool für VM-Umgebungen vergebens.

Bei genauerer Betrachtung der Objekte stellt sich heraus, dass im Konkreten zwei dieser Objekte beziehungsweise Methodiken, zur Realisierung von VM Infrastrukturkomponenten in Betracht gezogen werden könnten. Dies jedoch auch nur, falls eine manuelle Erstellung der Policy für jeden beteiligten Client/Host ausgeschlossen wird. Diese sind [Net09]:

- Host Objekt
- Erweitertes Firwall Objekt

Beim Host Objekt handelt es sich um ein Template zur Erstellung einer Policy für einen Host, welcher zumeist nur mit einer sichtbaren Netzwerkschnittstelle ausgestattet ist. Diesem Interface sind dabei mehrerer (virtuelle) Adressen zugewiesen, welche jedoch alle dem selben Interface angehören (zum Beispiel eth0). Das Host Objekt kann somit als eine Art Abstraktion beziehungsweise Generalisierung angesehen werden. Bei Verwendung dieses Objektes agiert dieses als Gruppe, was bedeutet, dass die Erstellung der Regeln für alle vorhandenen Adressen in kollaborativer Art und Weise vonstatten geht. Dies bedeutet, dass bei einer Erstellung des Regelsets, welches aufbauend auf diesem Objekt erstellt wird, für alle (virtuellen) Interfaces die selben Parameter oder aber auch Regeln erstellt werden.

Ähnlich verhält es sich mit dem erweiterten Firewall Objekt, wobei einem Interface mehrere Adressen zugeordnet werden können. Somit ist es möglich, einen Server zu reglementieren und auch einzelne Dienste/Server von einander abzukoppeln. Weiters ist es möglich, die einzelnen Interfaces eigens anzusprechen und zu parametrisieren. Leider entspricht dieses Element eher einem konkreten Use Case als einem Template, weshalb eine nicht manuelle Erstellung dessen ausgeschlossen ist.

13.3. Problembereiche

Beide zuvor vorgestellten Objekte sind nicht oder nur bedingt geeignet um eine virtuelle Umgebung einzubetten. Zum einen sei die summative Behandlung des IP-Adresspool

des Host-Objekts genannt, welche für unsere Anwendung nicht adäquat ist. Zum anderen fehlt beiden Ansätzen gänzlich die kohärente Konfiguration der unterschiedlichen virtuellen Adapter. Zudem ist eine Blockade des Zugriffs auf den Host nicht von Haus aus realisiert und muss händisch nachgetragen werden.

Basierend auf diesen „Schwachstellen“ beziehungsweise der unzureichenden Unterstützung erscheint es sinnvoll, ein Tool zu entwickeln, mit welchem ein (vordefiniertes) virtuelles Szenario exemplarisch bearbeitet und gelöst werden kann.

13.4. Erweiterung des Firewall Builders

Die Erweiterung der Funktionalität des FWBuilders betrifft die Erstellung eines externen Tools (nachfolgend FWBPlus genannt), um die zuvor genannte fehlende Unterstützung für die Einbindung von virtualisierten Umgebungen handhaben zu können. Dabei wird von einem konkreten Anwendungsfall beziehungsweise vorhandener Infrastruktur ausgegangen, wobei das erstellte Tool einem schematischen Problemlösungsansatz entspricht.

Der Anwendungsfall sieht folgendermaßen aus:

- Host mit einem physischen Interface
- Drei Clientkategorien (nachfolgend auch Zonen genannt) mit unterschiedlichen Funktionen
- Keine Zugriffe der Clients auf den Host
- Reglementierter Zugriff der Clients untereinander

Aufgrund der summativen Behandlung von Adressen des Host Objektes, welches ein Erben von dem übergeordneten Element impliziert, wird dieser Ansatz ausgeschlossen. Stattdessen wird auf die Methodik des erweiterten Firewall Objekts aufgebaut und dieses zur Lösung des Problems verwendet. Dem Host Objekt wird dementsprechend eine Art Platzhalterfunktion zugewiesen, welche die konkreten VMs in den Firewalls (im Speziellen iptables) referenziert.

13.4.1. Externes Tool FWBPlus

Das Zusatztool FWBPlus ermöglicht prinzipiell die Erstellung von Dateien, welche aufgrund ihres Aufbaus direkt in den FWBuilder (als Basis) übernommen werden können und einer virtualisierten Infrastruktur Rechnung tragen sollen. Zur Erfüllung der Hauptaufgabe, der externen Erweiterung des FWBuilders um den kohärenten Ansatz zu verfolgen, wird folgende Funktionalität im GUI bereitgestellt:

- Erstellung von FWBuilder-Dateien (*.fwb)
- Parametrisierung des Hosts (IP- sowie Netmask-Adresse)
- Parametrisierung von bis zu 7 VMs (IP-, Netmask-Adressen, Zone)
- Auswahl, ob Standardregeln inkludiert werden sollen

13.4.2. Analyse und Design

Die Analyse- und Designphase bildet die Basis einer guten Implementierung, weshalb nachfolgend die Ergebnisse (Klassen- sowie Use-Case-Diagramm) von dieser vorgestellt werden sollen.

Use-Case

Da es die Komplexität des Tools erlaubt, soll an dieser Stelle eine grafische Darstellung der Abläufe hinreichend sein, weswegen auf eine textuelle Beschreibung dieser verzichtet wird.

Die nachfolgende Abbildung soll über das Interaktionsvermögen des Benutzers Auskunft geben sowie die Möglichkeiten des Tools aufzeigen.

Wie aus der Abbildung ersichtlich, stehen dem Benutzer diverse Standardinteraktionen, wie zum Beispiel Speichern, Rücksetzten und Ähnliches zur Verfügung. Diese Funktionen wurden als übliche Menüeinträge realisiert und zudem mit Shortcuts belegt.

Will nun ein Benutzer seine Konfiguration speichern, so muss dieser folgende Reihenfolge einhalten:

1. Eingabe der Hostdaten
2. Auswahl der VMs
3. Eingabe der VM-Daten
4. Auswahl der VM-Zone(n)
5. (Eingabe des Dateinamens) bei „Speichern unter“

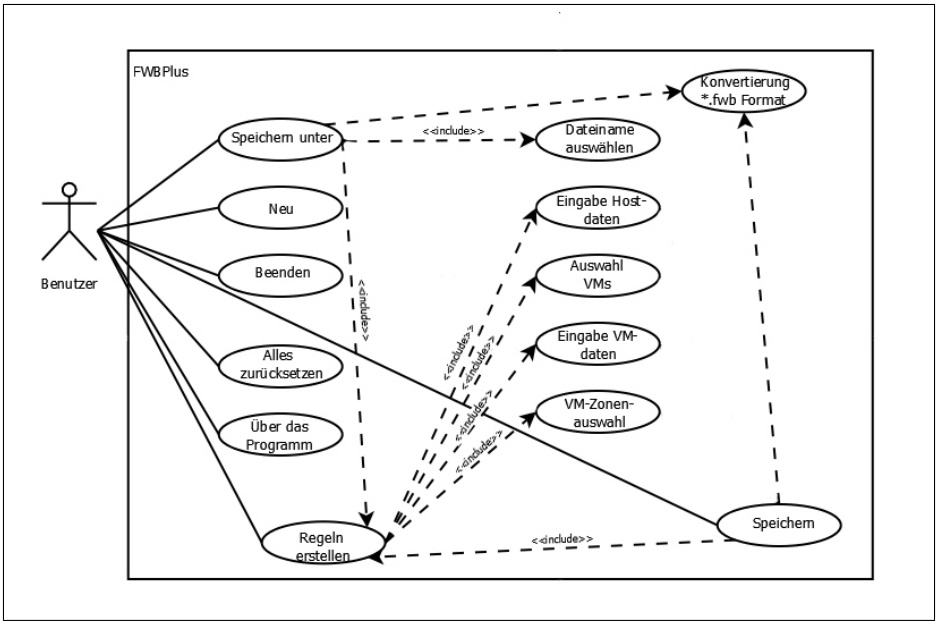

Abbildung 13.1.: Use-Case-Diagramm des FWBPlus-Tools

Beim Aufruf der Speicherroutine werden die eingegebenen Daten für IP- und Netmask-Adressen (für alle markierten Elemente) sowie die Eingabe einer Zone überprüft. Bei einem negativen Befund, oder aber auch einem Fehlen einer Eingabe dieser Überprüfungen, wird der Speichervorgang folglich abgebrochen und eine entsprechende Fehlermeldung ausgegeben. Die „Speichern unter“-Routine bedingt zusätzlich eine Eingabe des zukünftigen Dateinamens beziehungsweise eines Pfades.

Klassendiagramm des FWBPlus

Ähnlich dem Use-Case-Diagramm ist auch das Klassendiagramm der externen Erweiterung leicht lesbar beziehungsweise übersichtlich gehalten. Die anschließende Abbildung

soll die Klassen sowie Methoden der Erweiterung sowie deren Verbindung zueinander verdeutlichen.

Wie aus der Abbildung ersichtlich, besteht das FWBPlus-Tool aus fünf Klassen, welche zur Bereitstellung der Funktionalität notwendig sind. Ausgehend von der „FWBPlusMain"-Klasse, welche das Benutzerinterface bereitstellt sowie die Ablaufkontrolle und Interaktion mit dem Benutzer steuert, sind noch vier weitere Klassen, welche Hilfsfunktionalitäten bereitstellen, vorhanden. Der Hauptklasse kommt neben der Bereitstellung des grafischen Interface, noch die Aufgabe der sequenziellen Erstellung der Ausgabedatei in Abhängigkeit der Benutzereingaben zu. Ebenso beinhaltet sie die Menüfunktionalität als auch die Steuermechanismen zur Kontrolle der eingegebenen Daten. Darüber hinaus kann auch ausgewählt werden, ob die Standardregeln inkludiert werden sollen oder ob eine schematische beziehungsweise exemplarische Darstellung gewünscht ist.

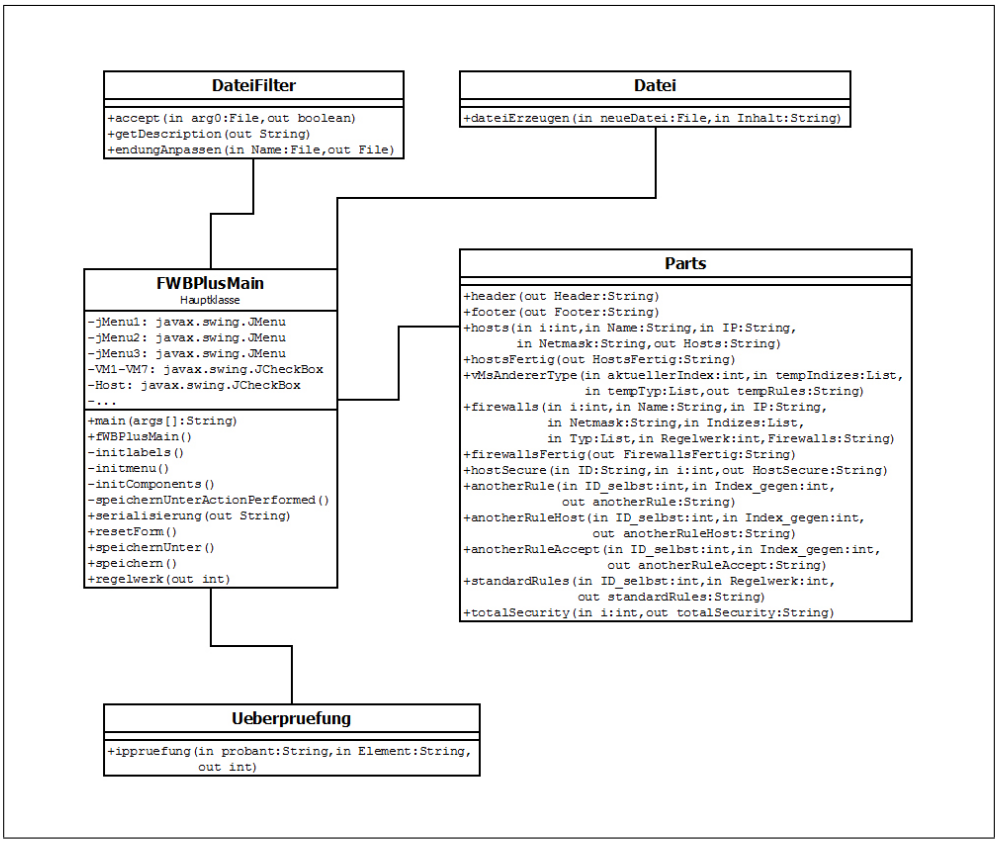

Abbildung 13.2.: Klassendiagramm des FWBPlus-Tools

Der „DateiFilter“-Klasse kommt die Aufgabe der Filterung der Dateien sowie Ordner im Auswahldialog des „Speichern unter“ zu. Ebenso stellt sie für den eingegebenen Dateinamen eine Anpassung der Endung (*.fwb) zur Verfügung.

Die Klasse „Ueberpruefung“ kontrolliert die Eingabedaten der IP- als auch Netmask-Adressen hinsichtlich deren Korrektheit. Zur Kontrolle dieser Daten finden die entsprechenden Regular Expressions Anwendung, welche den Wertebereich als auch die Struktur überprüfen.

Die „Datei“-Klasse hat wie der Name bereits vermuten lässt, die Aufgabe, die konkreten Strukturen sowie die damit verbundenen Daten in eine konkrete Datei zu schreiben. Bei der Verwendung des „Speichern unter“-Dialogs kann sowohl der entsprechende Dateiname als auch der Pfad ausgewählt werden, welcher in weiterer Folge an diese Klasse zur Erstellung übergeben wird. Bei der Verwendung des einfachen „Speicher“-Befehls wird die Datei „test.fwb“ im Verzeichnis erstellt, in welchem sich auch die Anwendung selbst befindet.

Der „Parts“-Klasse kommt die Funktion des Bereitstellens der Dateibestandteile zu, welche über speziell parametrisierte Aufrufe durch die „FWBPlusMain“-Methoden zu einer interpretierbaren FWBuilder Datei zusammengefügt werden. Zusätzlich wird das Vorhandensein von VMs anderer Klassen (DMZ, Intern, Test) überprüft und gegebenenfalls eine Anpassung der Firewall-Regeln durchgeführt. Der Rückgabewert besteht dabei immer aus einer Zeichenkette, welche die mühelose Weiterverarbeitung in der Hauptklasse sicherstellt.

13.4.3. Implementierung

Bei der FWBPlus-Erweiterung handelt es sich um ein Java Tool welches mittels Swing-Komponenten ein GUI zur Verfügung stellt. Es wurde unter Ubuntu (9.04) in der NetBeans-Umgebung (IDE v6.5) entwickelt und mittels des Java-Compilers (v1.6.0_0) in den Bytecode übersetzt. Die Verwendung von Java sowie des Swing-Frameworks ermöglichen eine problemlose Verwendung des Tools auf anderen OS-Plattformen (zum Beispiel MS Windows). Nachfolgend sollen die Hauptelemente sowie deren besondere Behandlung bei der Erstellung von FWBuilder-Dateien erläutert werden. Der prinzipielle Ablauf der Erstellung der Dateisubelemente, welche nachfolgend in die konkrete Datei geschrieben werden, entspricht:

1. (Auslesen und Überprüfen der eingegebenen Daten)
2. Einfügen des Headers
3. Einfügen der konkreten Host-Elemente (Host, VM1, ...)
4. Abschluss des Host-Elementes
5. Erstellung der Firewall-Elemente (Host, VM1, ...)
   a) Erstellung der Regeln basierend auf der Kategorie
   b) Erstellung der Absicherung des Hosts
   c) Erstellung der Standard-Regeln
6. Abschluss des Firewall-Elementes
7. Einfügen des Footers

Zur Einsicht des Quellcodes wird auf `http://www.semanticlab.net/index.php/FWBPlus` hingewiesen, welche unter anderem auf die entsprechenden Quellen verweist.

Anschließend sollen die einzelnen FWBuilder Elemente sowie deren Behandlung im Tool beschrieben werden.

Host-Elemente

Die Host-Elemente entsprechen eingenständigen Hosts, welche ein Netzwerk-Interface untergeordnet haben. Dieses Interface wird entsprechend der Benutzereingabe parametriesiert und unterstützt standardmäßig in der Implementierung keine dynamische Adresszuweisung. In weiterer Folge dienen die Host-Elemente als Platzhalter für die zu sperrenden, beziehungsweise freizugebenden Adressen der virtuellen Maschinen. Für jedes aktive Element, welches in der GUI ausgewählt wurde, wird somit ein eigener Eintrag erstellt und im Programm weiterverwendet.

Firewall-Elemente

Das Firewall-Element entspricht den verfügbaren Firewalls welche unterschiedlicher Ausprägung sein können. Als Standard wurde die Verwendung von Linux-Betriebssystemen mit einer Kernelversion von 2.4 oder 2.6 vorausgesetzt. Ebenso ist die Verwendung von iptables als Instrument vordefiniert, da dies üblicherweise Teil der gängigen Distributionen ist.

Wie bei den Host-Elementen wird für jedes selektierte Element des GUIs ein Firewall-Element angelegt. Dieses hat eine fix zugewiesene IP- und Netmask-Adresse und ist für die zweckmäige beziehungsweise automatisierte Verteilung der FW-Policy notwendig. Diese Verteilung wird mittels SSL-Verbindung aus dem FWBuilder heraus initiiert.

Das Firewall-Element enthält ebenso die anzuwendenden Regeln, welche beim Versand beziehungsweise Ankunft von Paketen zu Kontrollzwecken herangezogen werden. Entsprechend der gewählten Zone des jeweils selektierten Elements werden folgende unterschiedliche Regeln erstellt:

Test-Zone: VMs, welche dieser Zone angehören, werden gegen Zugriff auf/von anderen Elementen reglementiert.

Interne-Zone: VMs dieses Typs werden jeweils gegen die VMs anderer Zonen (DMZ und Test) mittels DENY-Regel abgesichert und enthalten explizite ACCEPT-Regeln für VMs des selben Typs.

DMZ-Zone: Entspricht dem Modell der internen Zone mit einer Absicherung gegenüber der Internen als auch Test-Zone.

Zu dieser kohärenten Konfiguration werden Regeln zur Absicherung (DENY) des Hosts für ausgehende Pakete unabhängig zu der gewählten Zone erstellt. Zusätzlich werden, sofern nicht explizit angegeben, die Standardregeln für die entsprechenden Elemente angelegt. Zu diesen Standardregeln zählen etwa eine Anti-Spoofing, als auch eine SSH-Zugangs-Regel. Auch werden spezielle Typen von ICMP-Nachrichten[3], welche gemeinhin als notwendig angesehen werden, explizit freigeschaltet. Darüber hinaus werden auch PING-Anfragen an die jeweiligen Elemente erlaubt.

Die oben genannte Regelerstellungsmethodik betrifft die unterschiedlichen VMs. Für den Host, welcher diese beherbergt, gelten unterschiedliche Kriterien. So wird etwa der

[3] Time Exceeded, Time Exceeded in Transit, PING-Reply und alle ICMP-Unreachable Nachrichten.

Zugriff der VMs auf den Host (Inbound-Nachrichten) verboten. Die Standardregeln hingegen werden ebenso wie bei den VMs angewandt.

Zusätzliche Informationen über die Erstellung von FWBPlus sowie der Funktionalität können dem Testszenario auf Seite 145 entnommen werden.

13.4.4. Benutzeroberfläche

Die Benutzeroberfläche beziehungsweise das GUI der experimentellen Erweiterung wurde mit Hilfe des Swing-Frameworks in Java erstellt.

Prinzipiell kann diese in zwei funktionale Teile eingeteilt werden. Zum Einen das Menü und zum Anderen der Teil zur Eingabe der Daten durch den Benutzer. Nachfolgende Abbildung soll die grafische Benutzeroberfläche verdeutlichen.

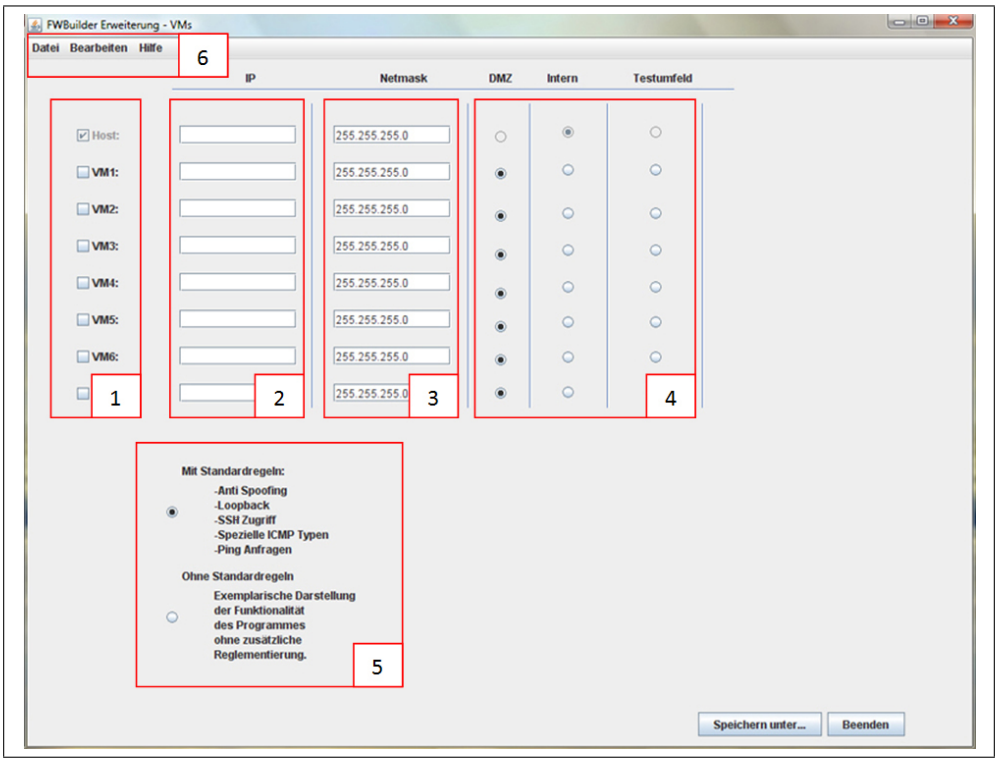

Abbildung 13.3.: Benutzeroberfläche des FWBPlus-Tools

Wie aus der Abbildung ersichtlich, ist zu Präsentationszwecken das GUI nochmals in sechs Teile aufgeteilt worden.

Im linken Bereich (Nummer 1) ist die Auswahl der VMs ersichtlich, welche von VM1 bis VM7 reicht. Der Host-Eintrag kann nicht manipuliert werden, da dieser als Grundvoraussetzung für eine virtualisierte Umgebung angesehen wird. Direkt neben der Auswahl der aktiven VMs ist die Eingabe der IP- (Nummer 2) sowie Netmask-Adressen (Nummer 3) realisiert. Im Weiteren (Nummer 4) kann die Zonenzugehörigkeit der selektierten virtuellen Maschinen festgelegt werden (nur ein Eintrag pro VM). Zusätzlich zu diesen spezifischen Angaben kann ausgewählt werden, ob ein exemplarischer Output gewünscht ist oder ob zusätzlich diverse Standardregeln inkludiert werden sollen (Nummer 5). Im Menü (Nummer 6) stehen schlussendlich die üblichen Einträge zur Steuerung des Programms bereit[4].

Anhand der in der Benutzeroberfläche eingegebenen VM-Konfigurationsdaten, können nun die FWBuilder Regeln erstellt werden. Diese Regeln können mittels des „Speichern"-Befehls exportiert werden und nachfolgend in die FWBuilder-Software übernommen werden. Die vom FWBPlus-Tool erstellten Regeln dienen somit als Ausgangspunkt für die Erstellung einer Host-Konfigurationsdatei, welche einem virtualisierten Umfeld Rechnung trägt.

FWBPlus Szenario Das nachfolgende Szenario soll zu Testzwecken die Funktionsweise des FWBPlus-Tools verdeutlichen.

Folgende Kriterien sollen erfüllt werden:

- Host mit der IP-Adresse 192.168.1.10
- Vier virtuelle Maschinen
  - VM1: IP-Adresse 192.168.1.11 (Intern)
  - VM2: IP-Adresse 192.168.1.12 (Intern)
  - VM3: IP-Adresse 192.168.2.10 (DMZ)

[4] Auf die Realisierung einer Öffnen-Routine wurde aufgrund der geringen Anzahl an Programmparametern verzichtet.

  - VM4: IP-Adresse 192.168.3.1 (Test)
- Standardregeln sollen hinzugefügt werden
- (Netmask jeweils 255.255.255.0)

Die Eingabe der Szenariowerte in das FWBPlus-Tool ergibt das Ergebnis wie in Abbildung 13.4 ersichtlich. Diese Eingaben werden mittels „Speichern Unter“-Befehl in die Datei „Szenario.fwb“ gespeichert.

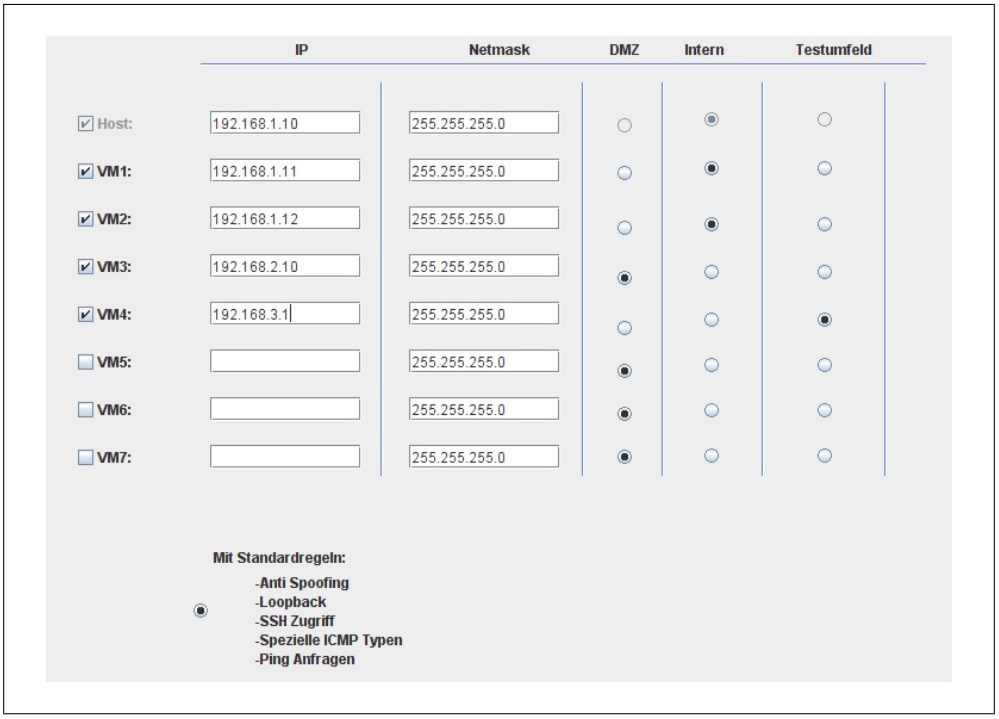

Abbildung 13.4.: Szenariodaten im FWBPlus-Tools

Die erstellte Datei wird in weiterer Folge im Firewall Builder Programm geöffnet und enthält sowohl Host- als auch Firewall-Elemente für alle selektierten Einträge (Host, VM1-VM4). Dies können wir in Abbildung A.1 auf Seite 156 sehen[5]. Ebenso ist in dieser Abbildung die Host-Policy auf der rechten Seite zu sehen. Diese enthält, wie beabsichtigt, Einträge um (eintreffende) Zugriffe von den einzelnen VMs zu unterbinden.

[5]Aus Gründen der Übersichtlichkeit sowie um die Lesbarkeit der Abbildungen zu erhalten, sind die Abbildungen A.1 bis A.5 Anhang A zu entnehmen.

Darüber hinaus sind auch die Standardregeln (siehe Kapitel 13.4.3), welche ab Index 4 beginnen, zu sehen.

Die Abbidlungen A.2 auf Seite 157 und A.3 auf Seite 158 zeigen die Policies von VM1 und VM2, welche sich in der selben Zone (Intern) befinden und dementsprechend Regeln enthalten, welche einen gegenseitiger Zugriff erlauben. Die VMs anderer Zonen (VM3 und VM4) werden hingegen mittels DENY-Regel vor Zugriffen von VM1 und VM2 geschützt. Ebenso enthalten die Policies die standardmäßige Regel zum Schutz des Hosts.

Da sich VM3 in einer DMZ befindet und keine andere virtuelle Maschine in selbiger eingesetzt wird, wird der Zugriff auf alle anderen Rechner unterbunden. Standardmäßig ist auch der Zugriff auf den Host unterbunden. Dies ist in Abbildung A.4 auf Seite 159 ersichtlich.

Da es sich bei der VM4 um einen Rechner im Testumfeld handelt, werden sämtliche Verbindungen ausgehend von diesem prinzipiell unterbunden. Dieser Umstand bleibt ebenso erhalten, falls sich eine andere VM in selbiger Kategorie befinden würde. Die entsprechende Policy ist Abbildung A.5 auf Seite 160 zu entnehmen.

14. Zusammenfassung und Ausblick

Zusammenfassend lässt sich festhalten, dass die schnelle funktionelle Umsetzung von VM-Technologie scheinbar zu Lasten sicherheitstechnischer Aspekte durchgeführt wurde. Dennoch kann der sinnvolle Einsatz von Firewalls auf virtualisierter Infrastruktur die Sicherheit nachhaltig erhöhen. Dies ist insbesondere durch kohärente Konfiguration der einzelnen Komponenten (Host und Clients) zu realisieren.

Die Einfachheit der Vervielfältigung von VM-Clients (basierend auf VM-Images) fördert die Verbreitung dieser. Dies bedingt bei dezentraler Sicherheitspolitik[1] den Einsatz leistungsstarker Lösungen, um die Sicherheitsrichtlinien durchsetzen zu können (zum Beispiel FWBuilder).

Die großteils mangelnde Unterstützung von virtualisierten Anwendungsfällen, dieser und ähnlicher Produkte, wird laut Sicht des Autors vorwiegend durch zwei Punkte beeinflusst. Zum einen, dass VM-Clients als eigene Hosts angesehen werden, wobei fälschlicherweise auf die Interaktion und mögliche Angreifbarkeit des VM-Hosts keine Rücksicht genommen wird. Andererseits dürften die gegenwärtigen (sicherheitstechnischen) Weiterentwicklungen von Seiten der VM-Technologie-Hersteller eine Art Warteposition unter den Herstellern von Sicherheitslösungen provozieren.

Der Verlagerung von Sicherheitsanwendungen, weg von den einzelnen VM-Clients auf den Host, können dabei einige gewichtige Vorteile zugeschrieben werden. Die Implementierung der Sicherheitsfunktionalität außerhalb der einzelnen Gast-VM verhindert zu einem gewissen Grad deren Manipulierbarkeit, zumindest ausgehend von kompromitierten Gastmaschinen. Darüber hinaus ist bei adäquater Implementierung (zum Beispiel im VMM) ein Performancegewinn gegenüber herkömmlicher Lösungen zu erwarten. Als weiterer nicht zu unterschätzender Punkt sei die Möglichkeit der zentralen Verwaltung

[1] Im Host sowie in allen verfügbaren Clients werden die entsprechenden Richtlinien umgesetzt. Demnach existiert keine zentrale Kontrolle, sondern nur ein zentrales Verwaltungstool. Die Nachteile dieser Umsetzung liegen im hohen Wartungsaufwand sowie der Ausführung auf Applikationsebene begründet.

und Kontrolle der Aktivitäten (zum Beispiel Protokollierung) zu nennen.

Den zuvor genannten Vorteilen steht der gegenwärtige, experimentelle Entwicklungsstand der Sicherheitstools auf Host-Ebene[2], in diversen Ausprägungen gegenüber, welcher einen Einsatz derzeit nicht rechtfertigt. Ebenso erfordern die derzeit erzielten Leistungsmerkmale der unterschiedlichen Ansätze eine ausgeprägte Optimierungsphase um einen effizienten Einsatz überhaupt erst ermöglichen zu können.

[2]Diese können zum Beispiel direkt im VMM als Sicherheits-API oder aber auch als virtuelle Netzwerkschicht implementiert sein.

Literaturverzeichnis

[AMD07] AMD: *IOMMU Architectural Specification*, 2007. `http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/34434.pdf`, Abruf am 2009-02-02.

[Arc07] Arce, Iván: *Ghost in the Virtual Machine*. IEEE Security and Privacy, 5(4):68–71, 2007.

[Bab08] Babcock, Charles: *Virtualization's Tipping Point*. InformationWeek, (1186):14, Mai 2008.

[Bro08] Brodkin, Jon: *VMware partners demonstrate VMsafe virtual security prototypes*. Network World (Online), September 2008.

[CLS07] Carpenter, Matthew, Tom Liston und Ed Skoudis: *Hiding Virtualization from Attackers and Malware*. IEEE Security and Privacy, 5(3):62–65, 2007.

[Cum08] Cummings, Joanne: *How to segregate virtual servers*. Network World, 25(11):40, März 2008.

[CZL08] Carbone, Martim, Diego Zamboni und Wenke Lee: *Taming Virtualization*. IEEE Security and Privacy, 6(1):65–67, 2008.

[DPW09] Dalton, Chris I., David Plaquin und Wolfgang Weidner: *Trusted virtual platforms: a key enabler for converged client devices*. ACM SIGOPS Operating Systems Review, 43(1):36–43, 2009.

[Dub08] Dubie, Denise: *Vendors tackle virtual security*. Network World, 25(36):20, September 2008.

[EM06] England, Paul und John Manferdelli: *Virtual machines for enterprise desktop security*. Information Security Technical Report, 11(4):193–202, 2006.

[Erb01] Erb, Hubert: *Die Cyberspace-Fallen des FBI*, 2001. `http://www.heise.de/tp/r4/artikel/7/7634/1.html`, Abruf am 2008-12-30.

[FG05] Fritsch, Jörg und Steffen Gundel: *Firewalls im Unternehmenseinsatz: Grundlagen, Betrieb und Produkte*. Dpunkt Verlag, 2. überarb. und aktualis. a. Auflage, August 2005.

[Fis08a] Fischer, Marcus: *Ubuntu GNU/Linux: Aktuell zu "Hardy Heron"*. Galileo Press, 3., aktualisierte a. Auflage, Juni 2008. `http://openbook.galileocomputing.de/ubuntu/`.

[Fis08b] Fischer, Marcus: *Xen: Von den Grundlagen bis zur Administration*. Galileo Press, 1. Auflage, November 2008.

[Fyo97] Fyodor: *Port Scanning Techniques*, 1997. `http://nmap.org/book/man-port-scanning-techniques.html`, Abruf am 2008-12-30.

[Fyo98] Fyodor: *Remote OS detection via TCP/IP Stack FingerPrinting*. Phrack Magazine(8), 1998. `http://www.phrack.org/issues.html?issue=54&id=9#article`, Abruf am 2008-12-30.

[Gla07] Gladewitz, Robert: *Angriffserkennung in Firewalls: Implementierung einer schwellwertbasierten Net- und Portscananalyse*. Vdm Verlag Dr. Müller, 1. Auflage, Juli 2007.

[GR05] Garfinkel, Tal und Mendel Rosenblum: *When virtual is harder than real: security challenges in virtual machine based computing environments*. In: *Proceedings of the 10th conference on Hot Topics in Operating Systems*, Band 10, Santa Fe, NM, 2005. USENIX Association.

[Gra01] Granger, Sarah: *Social Engineering Fundamentals, Part I: Hacker Tactics*, 2001. `http://www.securityfocus.com/infocus/1527`, Abruf am 2008-12-30.

[Gre00] Grennan, Mark: *Firewall and Proxy Server HOWTO*, 2000. `http://www.grennan.com/Firewall-HOWTO.html`, Abruf am 2008-12-30.

[Gre08] Greene, Tim: *10 security threats to watch for*. Network World, 25(15):30, April 2008.

[Her08] Hernick, Joe: *Securing VMware*. InformationWeek, (1193):31, Juli 2008.

[Int08] Intel: *Intel(r) VT for Direct IO*, 2008. `http://download.intel.com/technology/computing/vptech/Intel(r)_VT_for_Direct_IO.pdf`, Abruf am 2009-02-02.

[Jan07] Janowicz, Krzysztof: *Sicherheit im Internet*. O'Reilly, 3. Auflage, Juli 2007. `http://www.oreilly.de/german/freebooks/sii3ger/`.

[Kni05] Knight, William: *Firewalls ring changes*. Infosecurity Today, 2(2):18–21, April 2005.

[Kog07] Kogelbauer, Florian: *Elektronische Dokumentenverwaltung und -bereitstellung auf einer Website durch Implementierung eines Zusatzmoduls DokuSERVE*. Wirtschaftsuniversität Wien, 2007.

[Kog08] Kogelbauer, Florian: *Virtual Private Networks - Grundlagen, Geschichte sowie aktuelle Entwicklungen*. Wirtschaftsuniversität Wien, 2008.

[KS08] Karger, Paul A. und David R. Safford: *I/O for Virtual Machine Monitors*. IEEE Security & Privacy, 6(5):16–23, Oktober 2008.

[Kur08] Kurland, Vadim: *Firewall Builder FAQ*, Juli 2008. `http://www.fwbuilder.org/docs/firewall_builder_faq.html`, Abruf am 2008-12-30.

[KVM08] KVM: *Kvm Forum 2008*, 2008. `http://www.linux-kvm.org/page/KvmForum2008`, Abruf am 2009-04-06.

[KVM09] KVM: *KVM - FAQ*, 2009. `http://www.linux-kvm.org/page/FAQ#FAQ`, Abruf am 2009-03-30.

[LD07] Lange, John R. und Peter A. Dinda: *Transparent network services via a virtual traffic layer for virtual machines*. In: *Proceedings of the 16th international symposium on High performance distributed computing*, Seiten 23–32, Monterey, California, USA, 2007. ACM.

[Les06] Lessig, Andreas G.: *Linux Firewalls- Ein praktischer Einstieg*. O'Reilly, 2. Auflage, 2006.

[Lip07] Lipp, Manfred: *VPN - Virtuelle Private Netzwerke: Aufbau und Sicherheit*. Addison-Wesley, München, 1. Auflage, September 2007.

[LMJ07] Laureano, M., C. Maziero und E. Jamhour: *Protecting host-based intrusion detectors through virtual machines*. Computer Networks, 51(5):1275–1283, April 2007.

[Net09] NetCitadel, LLC: *Firewall Builder User´s Guide*, 2009. `http://www.fwbuilder.org/docs/UsersGuide3.pdf`, Abruf am 2008-06-25.

[NH08] Nance, Kara und Brian Hay: *Virtual Machine Introspection*. IEEE Security & Privacy, 6(5):32–37, Oktober 2008.

[Ope09a] OpenBSD: *PF: Der OpenBSD Packet Filter*, 2009. `http://openbsd.org/faq/pf/de/index.html`, Abruf am 2009-02-02.

[Ope09b] OpenVPN: *OpenVPN 2.0.x - ManPage*, 2009. `http://openvpn.net/index.php/documentation/manuals/openvpn-20x-manpage.html#lbAV`, Abruf am 2009-04-17.

[Pho08] Phoronix: *Ubuntu 8.04 KVM Benchmarks*, 2008. `http://www.phoronix.com/scan.php?page=article&item=ubuntu_virt_benchmarks&num=1`, Abruf am 2009-04-01.

[PW07] Plötner, Johannes und Steffen Wendzel: *Linux: Das distributionsunabhängige Handbuch*. Galileo Press, 1. Auflage, Oktober 2007. `http://openbook.galileocomputing.de/linux/`.

[QEM08] QEMU: *QEMU Emulator User Documentation*, Jänner 2008. `http://bellard.org/qemu/qemu-doc.html`, Abruf am 2008-12-30.

[Rad08] Radcliff, Deb: *HOW TO ROOT OUT ROOTKITS*. Network World, 25(31):28, August 2008.

[RS09] Ray, Edward und Eugene Schultz: *Virtualization security*. In: *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, Seiten 1–5, Oak Ridge, Tennessee, 2009. ACM.

[Rue07] Ruef, Marc: *Die Kunst des Penetration Testing - Handbuch für professionelle Hacker: Sicherheitslücken finden, Gefahrenquellen schließen*. C & l Computer-U. Literaturverlag, 1. auflage Auflage, Juni 2007.

[Rus00] Russell, Rusty: *Linux IP Firewalling Chains*, 2000. `http://people.netfilter.org/~rusty/ipchains/`, Abruf am 2008-12-30.

[Spe06] Spenneberg, Ralf: *Linux-Firewalls mit iptables & Co.* Addison-Wesley, München, 1. Auflage, 2006.

[Spe07] Spenneberg, Ralf: *Sicherheit für virtuelle Systeme mit Xen*, Februar 2007. `http://www.linux-magazin.de/videos/sicherheit_fuer_virtuelle_systeme_mit_xen`, Abruf am 2008-12-30.

[Tan02] Tanenbaum, Andrew S.: *Computer Networks.* Prentice Hall International, 4. Auflage, August 2002.

[Tho08] Thorns, Fabian: *Das Virtualisierungs-Buch.* C & l Computer- U. Literaturverlag, 2., aktualisierte und erweiterte auflage. Auflage, September 2008.

[Vir08] VirtualBox: *Sun xVM VirtualBox - User Manual*, Dezember 2008. `http://dlc-cdn-rd.sun.com/c1/virtualbox/2.1.0/UserManual.pdf?e=1230900701&h=59dd6233e80ec9b7ad2a52666075bc07`, Abruf am 2008-12-30.

[WDC08] Wang, Xiaoying, Zhihui Du und Yinong Chen: *Virtualization-based autonomic resource management for multi-tier Web applications in shared data center.* Journal of Systems and Software, 81(9):1591–1608, September 2008.

[WHD09] Williams, Daniel, Wei Hu und Jack W. Davidson: *Security through Diversity: Leveraging Virtual Machine Technology.* IEEE Security and Privacy, 7(1):26–33, 2009.

[Won08] Wong, Bill: *Will Virtualization Save The Day?* Electronic Design, 56(9):47, Mai 2008.

[WP07] Wendzel, Steffen und Johannes Plötner: *Netzwerk-Sicherheit: Risikoanalyse, Methoden und Umsetzung.* Galileo Press, 2., aktualis. und erw. a. Auflage, Februar 2007.

[WSV09] Wood, Timothy, Prashant Shenoy und Arun Venkataramani: *Sandpiper: Black-box and Gray-box Resource Management for Virtual Machines.* Computer Networks, July 2009.

[Xen08] Xen: *Xen v3.3 - Users' Manual*, 2008. `http://bits.xensource.com/Xen/docs/user.pdf`, Abruf am 2008-12-30.

Appendix

A. FWBPlus Szenario Abbildungen

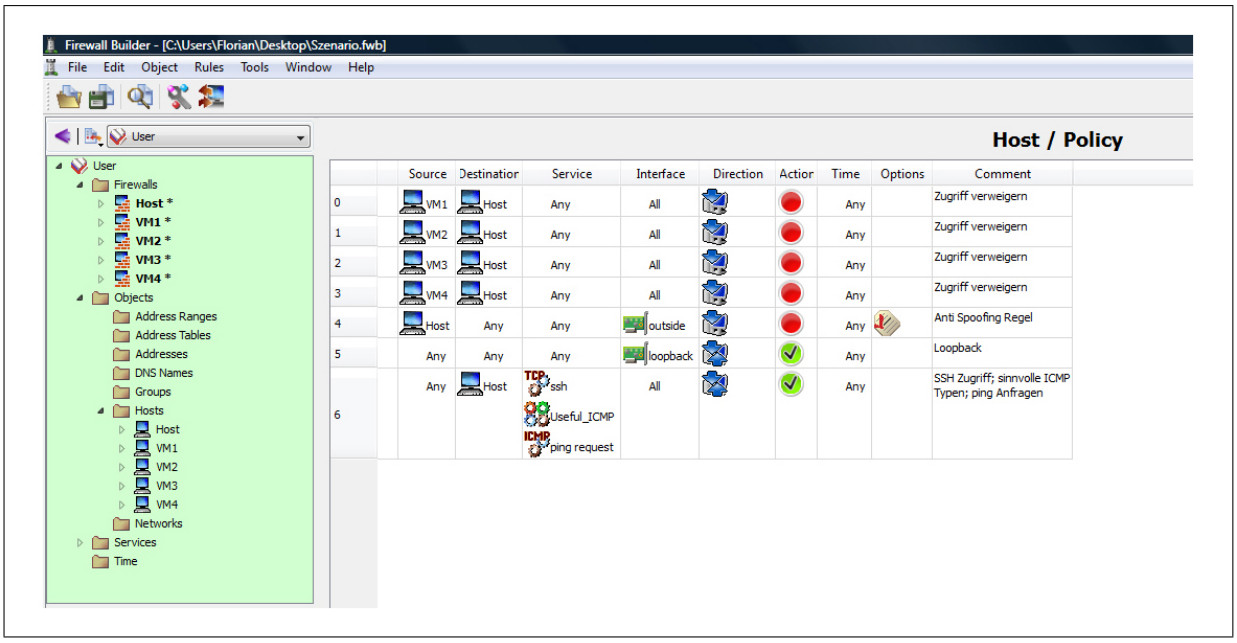

Abbildung A.1.: Host- und Firewall-Elemente im FWBuilder mit Host-Policy

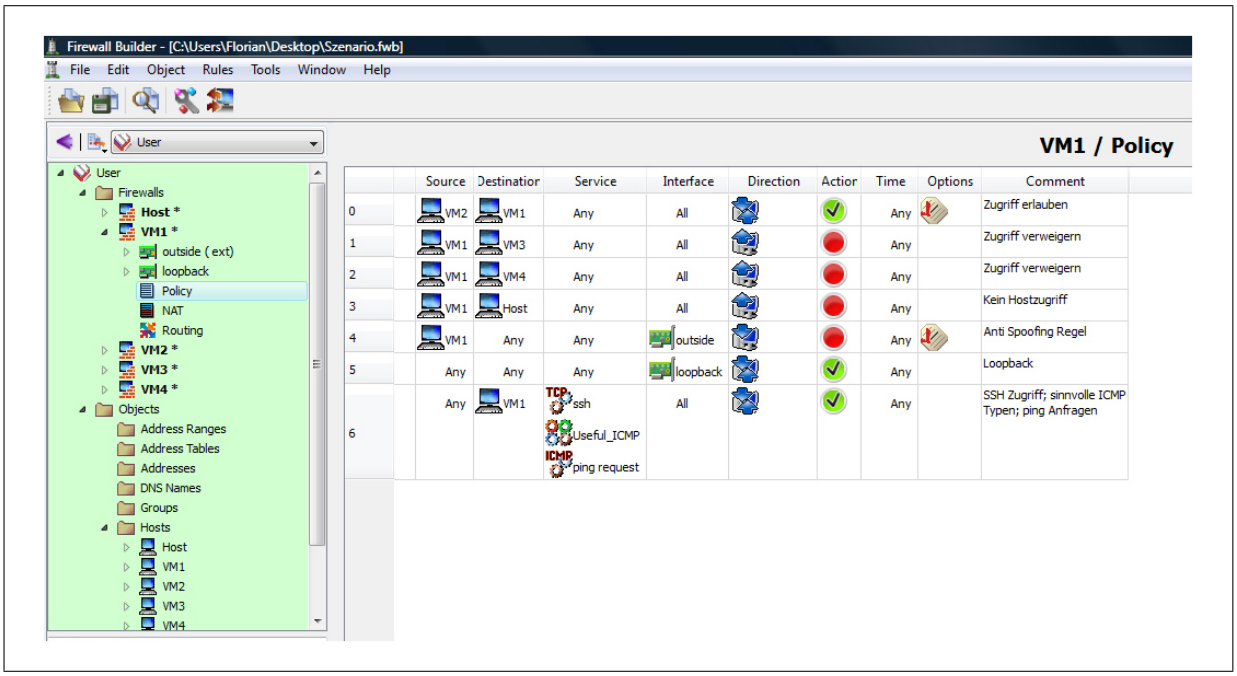

Abbildung A.2.: Erzeugte VM1-Policy im FWBuilder

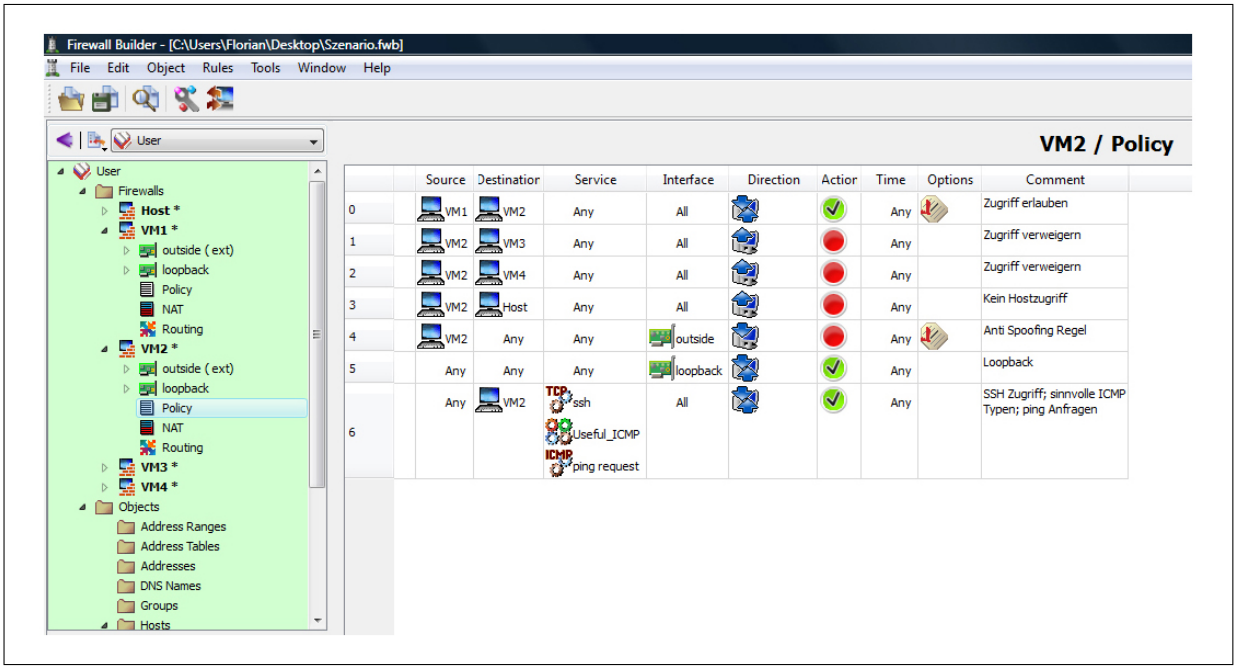

Abbildung A.3.: Erzeugte VM2-Policy im FWBuilder

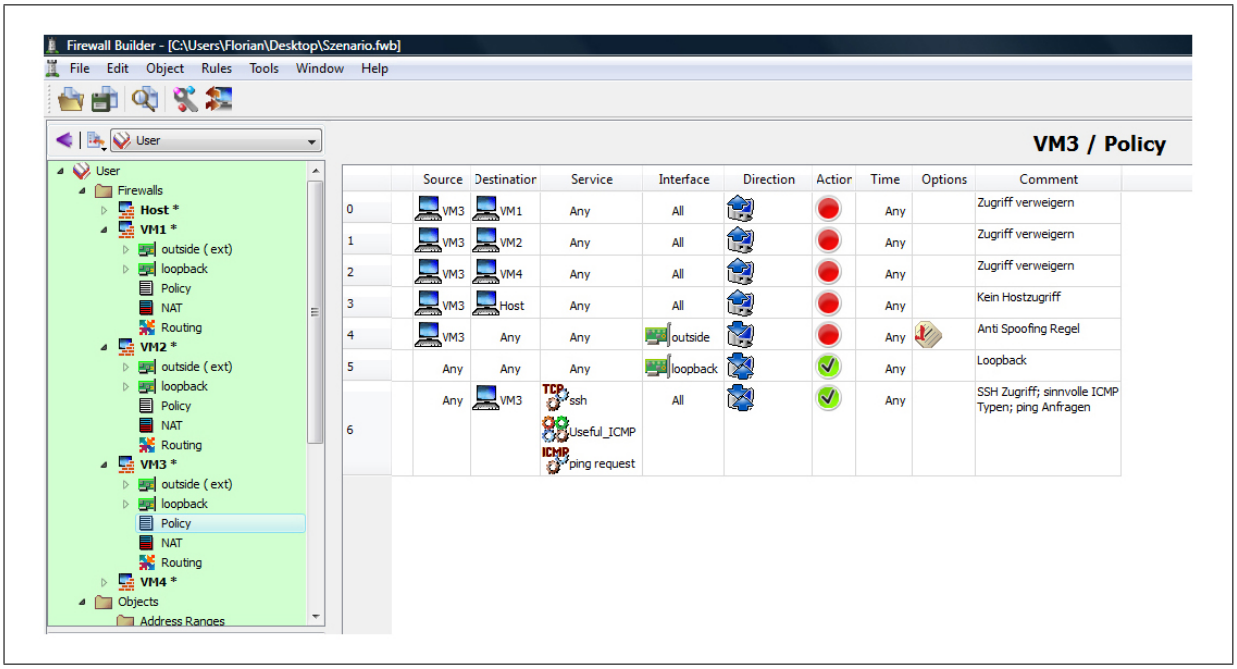

Abbildung A.4.: Erzeugte VM3-Policy im FWBuilder

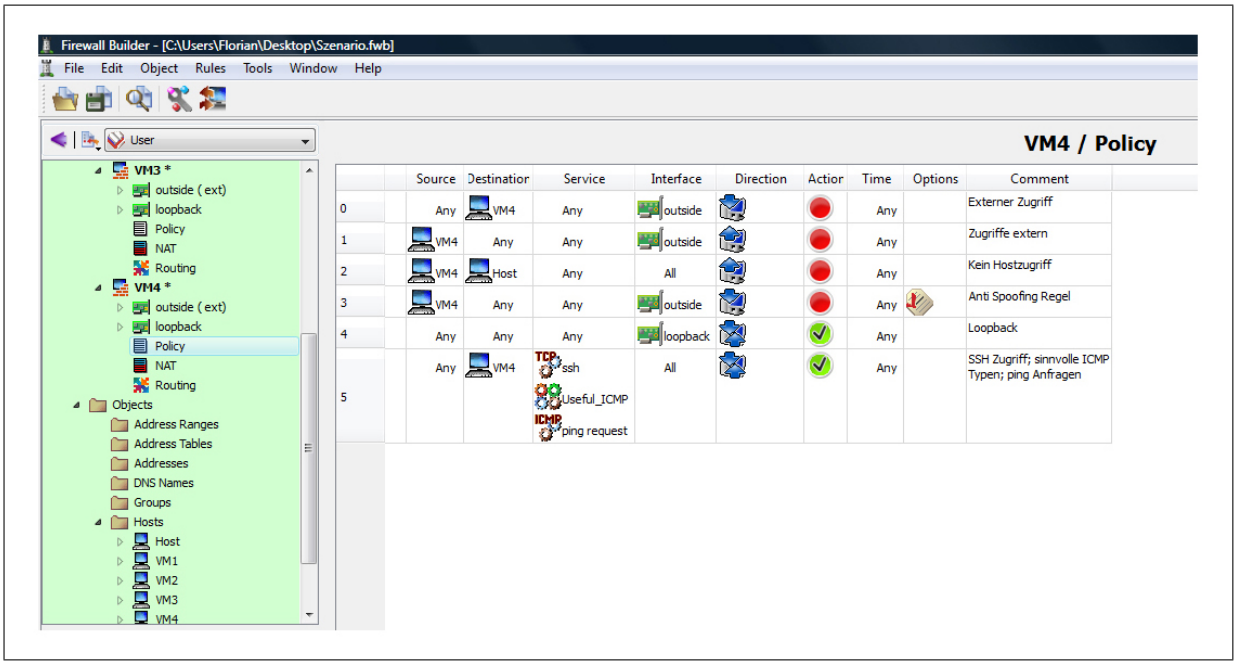

Abbildung A.5.: Erzeugte VM4-Policy im FWBuilder