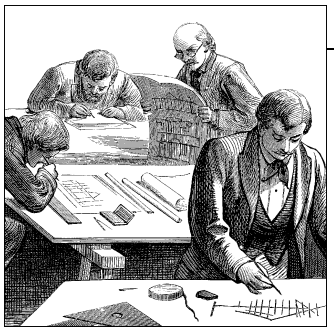

KAPITEL 5

Firewall-Architekturen

Kommen wir nun zum Kern des Themas. Nachdem wir uns in den vorigen Kapiteln mit Netzwerkprotokollen und Angriffen beschäftigt haben, wird es nun Zeit, uns etwas genauer anzusehen, wie Firewalls funktionieren. Wir werden sehen, daß es verschiedene Varianten von Firewalls gibt.

Während für einige Anwendungsfälle schon Router ausreichen, die es erlauben, Zugriffe auf bestimmte Ports zu unterbinden, ist für andere eine Firewall nötig, die die vermittelten Daten »versteht«. Solch eine Firewall wäre dann sogar in der Lage zu erkennen, daß eine HTML-Seite JavaScript-Funktionen enthält, und diese zu filtern. Die folgenden Abschnitte werden daher die unterschiedlichen Bausteine und Architekturen betrachten, mit denen Firewalls realisiert werden können.

Paketfilter

Abbildung 5-1 verdeutlicht die Arbeitsweise eines Paketfilters. Hierbei handelt es sich um einen modifizierten Router, der einzelne Pakete nach einem vorher festgelegten Satz von Regeln weiterleitet oder verwirft. Die Regeln beziehen sich dabei nur auf die Paketheader. Der Inhalt der Pakete wird nicht beachtet. Auf diese Weise kann nach folgenden Informationen gefiltert werden:

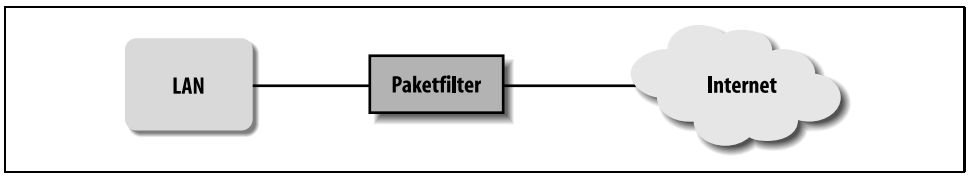

Abbildung 5-1: Einfacher Paketfilter

- Protokoll (ICMP, UDP, TCP . . .)
- Quelladresse
- Quellport

- Zieladresse
- Zielport
- Flags (insbesondere TCP-Verbindungsaufbau)

Einige Paketfilter erlauben es sogar, Regeln aufzustellen, die davon abhängen, welche Pakete zuvor empfangen wurden (*Stateful Packet Filtering*).

Linux unterstützt dies erst seit dem Kernel 2.4.0. Hier wird über bestehende Verbindungen Buch geführt und jedem Paket die Eigenschaft »öffnet eine neue Verbindung«, »gehört zu einer bestehenden Verbindung«, »öffnet eine Verbindung, die in Beziehung zu einer bestehenden Verbindung steht« (z. B. FTP-Datenverbindung) oder »ungültig« zugeordnet. Diese Zuordnung kann in den Firewallregeln abgefragt werden, so daß es z. B. möglich ist, eingehende Verbindungen zu verbieten, die nicht zu einer FTP-Kontrollverbindung gehören.

Für viele Anwendungen bieten Paketfilter einen guten Schutz bei geringem Aufwand und zu vertretbaren Kosten. Oft kann ein sowieso schon vorhandener Router entsprechend umgerüstet werden. Da die vermittelten Pakete zwar unter bestimmten Umständen unterdrückt, ansonsten aber nicht verändert werden, bleibt die Firewall für die Anwendungen transparent. Weder Server noch Klienten müssen an die Firewall angepaßt werden. Schließlich sind Paketfilter auch noch effizient. Die durch die Überprüfung entstehende Verzögerung ist gering, so daß Paketfilter deutlich höhere Durchsatzraten erzielen als andere Systeme.

Trotz der Vorteile werden auch andere Konstruktionen verwendet. Dies liegt daran, daß Paketfilter zwar schnell sind, aber diese Geschwindigkeit durch Verzicht auf Kontrolle erkauft wird. Da Paketfilter den Inhalt der weitergeleiteten Pakete nicht verändern, ist es z. B. nicht möglich, mit ihnen aktive Inhalte oder Cookies zu filtern. Benutzer können nicht unterschieden werden, wodurch Systemprotokolle weniger aussagekräftig und Zugriffe nur anhand der Rechneradressen, nicht aber anhand der Benutzer erlaubt werden.

Schließlich können sich auch Filterregeln gegenseitig beeinflussen. Bei FTP wird z. B. eine Rückverbindung von Port 20 des Servers zu einem hohen Port des lokalen Rechners geöffnet. Nun kann man dafür eine Filterregel aufstellen. Dies bedeutet dann aber, daß beliebige Zugriffe von Port 20 auf hohe Ports der Klienten zulässig sind. Die Entscheidung wird ausschließlich anhand der Portnummern getroffen, ein tieferes Verständnis des FTP-Protokolls fehlt dem Paketfilter.

Eine Ausnahme bilden hier nur zustandsbehaftete Filter, mit denen man Regeln der Art aufstellen kann:

»Falls eine Verbindung von einem lokalen Rechner geöffnet wird, die zu einem Protokoll gehört, bei dem auch Rückverbindungen geöffnet werden, so erlaube solche Rückverbindungen von Port 20 an hohe Ports des lokalen Rechners.«

Dies erlaubt Linux allerdings erst seit dem Kernel 2.4. Vorher konnte man nur versuchen, das Problem mit Masquerading zu entschärfen. Dazu aber später mehr in Kapitel 5, Unterabschnitt *Masquerading*, ab Seite 69.

Proxies

Die Abbildung 5-2 zeigt ein anderes Konzept. Hier wird ein Rechner mit zwei Netzwerkkarten eingesetzt, der keine Netzwerkpakete routet. Statt dessen sind auf ihm spezielle Programme, sogenannte *Proxies*, installiert, die Anfragen entgegennehmen, als wären sie Server. Statt die ankommenden Anfragen aber selbst zu bearbeiten, reichen sie sie an einen Rechner im Internet weiter. Gegenüber diesem treten sie wie normale Klienten auf.

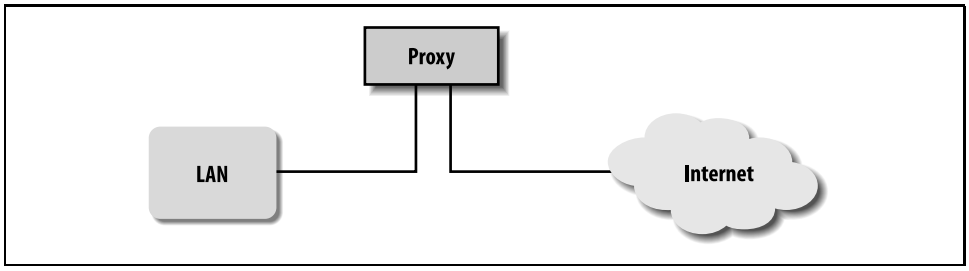

Abbildung 5-2: Einfacher Proxy

Dieses Vorgehen hat deutliche Vorteile gegenüber einfachen Paketfiltern. Diese liegen in der Tatsache begründet, daß ein guter Proxy Anfragen interpretiert[1]. Dadurch kann gezielt zwischen »guten« und »schlechten« Anfragen bzw. Antworten unterschieden und können »schlechte« gezielt herausgefiltert werden. Dies produziert aussagekräftigere Systemprotokolle und erlaubt eine feinere Kontrolle der übertragenen Inhalte. Auch wird es einfacher, Netzwerkpakete einem Protokoll zuzuordnen. So ist es im Fall von FTP nicht nötig, Pakete von Port 20 ohne nähere Kontrolle ins innere Netz zu lassen. Statt dessen öffnet der Server im Internet eine Verbindung zum Proxy der Firewall. Nur wenn hier auch eine solche Verbindung erwartet wird, werden auch tatsächlich Pakete entgegengenommen.

Mehr Kontrolle bedeutet andererseits aber auch, daß die Kontrollen länger dauern. Je gründlicher die Daten untersucht werden, um so mehr sinkt die Übertragungsgeschwindigkeit. Darüber hinaus kann auf einen Dienst nur dann zugegriffen werden, wenn ein entsprechender Proxy für das jeweilige Übertragungsprotokoll vorhanden ist. Dies kann zu Problemen führen, wenn das Übertragungsprotokoll neu, für den Gebrauch mit einem Proxy ungeeignet oder zu wenig verbreitet ist.

Während einige Protokolle (z. B. DNS, SMTP, HTTP) relativ einfach mit einem Proxy zu betreiben sind, besteht bei anderen das Problem, daß im Protokoll der Weg über einen Zwischenrechner nicht wirklich vorgesehen ist. Ein gutes Beispiel dafür ist FTP. Das Protokoll sieht nicht vor anzugeben, an welchen Zielrechner die Anfrage gerichtet ist. Dies ist eigentlich auch nicht nötig, da Anfragen direkt an den Zielrechner gestellt werden. In unserem Fall ist dieser aber nicht erreichbar. Wir müssen statt dessen unseren Proxy bitten, für uns eine Verbindung zum Zielrechner aufzubauen.

1 Für spezielle Anwendungen existieren auch Proxies, die Pakete, ohne sie auszuwerten, direkt an einen bestimmten Rechner weiterleiten. Diese sind aber nicht Gegenstand der Ausführungen.

Hierzu bieten sich uns zwei Möglichkeiten. Wir können erstens einen modifizierten Klienten benutzen, der eine spezielle Methode kennt, dem Proxy die nötige Information mitzuteilen. Im Falle gängiger Webbrowser ist dies so gelöst, daß der Browser die Anfrage an den Proxy mittels HTTP stellt und eine URL der Art *ftp://some.server.com/...* angibt. Erst der Proxy benutzt dann FTP, um die gewünschte Datei vom Zielrechner zu holen.

Eine andere Variante benutzt das SOCKS-Protokoll. Mit diesem teilt der Klient dem Proxy mit, zu welchem Server er sich verbinden will. Falls der zu kontaktierende Server und Port in der Proxy-Konfiguration gestattet wurde, öffnet der Proxy eine entsprechende Verbindung und leitet dann die Daten des Klienten transparent weiter. Es besteht auch die Möglichkeit, den Server anzuweisen, auf eingehende Verbindungen aus dem Internet zu warten (z. B. für FTP-Datenverbindungen). Seit SOCKS Version 5 wird auch die Authentisierung des Benutzers und die Weiterleitung von UDP unterstützt. Hier handelt es sich um einen generischen Proxy, der Daten einfach nur weiterleitet, ohne sie zu verstehen. Dies bedeutet, daß viele der Vorteile, die der Einsatz eines Proxys bieten kann, nicht genutzt werden können. Hinzu kommt, daß man Klienten benötigt, die das SOCKS-Protokoll nutzen können. Ist man nicht darauf angewiesen, seine Benutzer zu authentisieren, so kann man denselben Grad an Sicherheit bei geringerem Aufwand durch Paketfilter in Verbindung mit Masquerading (siehe Kapitel 5, Unterabschnitt *Masquerading*, ab Seite 69) erreichen.

Besteht keine Möglichkeit, spezielle Klienten zu nutzen, so muß der Benutzer selbst der Firewall mitteilen, mit welchem Rechner er sich zu verbinden wünscht. Eine gängige Lösung für FTP sieht so aus, daß der Benutzer mit einem unmodifizierten Klienten eine FTP-Verbindung zum Proxy öffnet. Bei der Anmeldung gibt er dann statt der normalen Kennung (z. B. `anonymous`) seine Kennung gefolgt von einem »@« und dem Rechnernamen an (z. B. `anonymous@some.server.com`).

Network Address Translation

Unter *Network Address Translation* (NAT) versteht man die regelgesteuerte Manipulation von Absender- und/oder Empfängeradresse und -port.

Obwohl NAT strenggenommen nichts mit Firewalling zu tun hat, wird es doch oft zur Unterstützung des Firewalling eingesetzt. Im Linux-Kernel wird es außerdem spätestens seit Version 2.4 durch dieselben Mechanismen wie die Paketfilterung realisiert und auch mit denselben Werkzeugen (`iptables`) konfiguriert.

Grundsätzlich sind verschiedene Varianten der Network Address Translation denkbar. Man unterscheidet normalerweise *Destination NAT* und *Source NAT*.

Beim Source NAT wird die Quelladresse verändert. Dies geschieht, wenn man z. B. mehr Rechner als im Internet gültige Adressen besitzt. In diesem Fall ändert man die Quelladressen jedes gesendeten Paketes, so daß es von einer der wenigen legalen IP-Adressen zu stammen scheint. Um auch die Antwort wieder zurückübersetzen zu können, wird normalerweise auch der Quellport verändert. Die veränderten Quellangaben bilden da-

bei einen eindeutigen Schlüssel, anhand dessen die ursprünglichen Angaben in einer Tabelle gefunden und wiederhergestellt werden können. Ein Spezialfall ist dabei das Masquerading, das im folgenden in einem eigenen Abschnitt beschrieben wird.

Beim Destination NAT wird auf die gleiche Weise die Zieladresse und u. U. der Zielport eines Paketes verändert. Auf diese Weise ist es möglich, Nachrichten, die für einen Rechner bestimmt sind, auf einen anderen umzuleiten. Einen Spezialfall stellt dabei die Redirection dar, der ebenfalls ein eigener Abschnitt gewidmet ist.

Masquerading

Das in Abbildung 5-3 dargestellte Masquerading eignet sich recht gut als Ergänzung zur einfachen Paketfilterung. Bei dieser Technik verändert der maskierende Router die Header der weiterzuleitenden Pakete aus dem lokalen Netz derart, daß als Absender der Router und nicht mehr der Rechner im lokalen Netz angegeben ist. Um trotzdem Antwortpakete noch zuordnen zu können, verändert der Router auch den Absenderport auf einen zufälligen Wert größer 1023. Kommt nun ein Antwortpaket, braucht er diesen Port nur in einer internen Tabelle nachzuschlagen und kann es dann dem eigentlichen Empfänger zustellen.

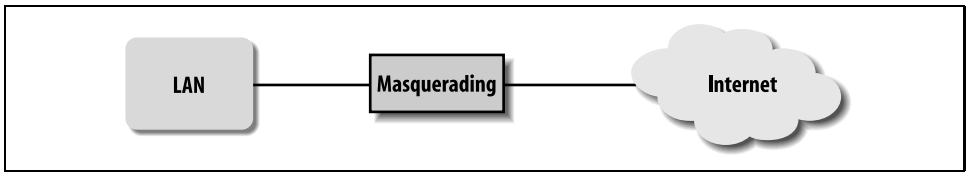

Abbildung 5-3: Masquerading

Hierdurch sind die Rechner im lokalen Netz aus dem Internet wie bei einer Proxy-basierten Firewall nicht mehr sichtbar. Alle Pakete scheinen von der Firewall zu kommen. Leitet der Router darüber hinaus keine Pakete weiter, die direkt an die Rechner im LAN gerichtet sind, so brauchen letztere auch keine gültigen IP-Adressen. Statt dessen kann man ihnen private Adressen (siehe Kapitel 3, Abschnitt *IP*, ab Seite 20) zuweisen, was neben der Ersparnis der Registrierungskosten auch den Vorteil hat, daß Versuche, Pakete zu senden, die direkt an diese Rechner gerichtet sind, schon am ersten Router des Übertragungsweges scheitern.

Ein maskierender Paketfilter verbindet damit die Vorteile eines Paketfilters mit denen eines Proxys. Wie bei einem Proxy sind die Rechner im lokalen Netz nicht sichtbar. Angriffe sind damit nur gegen die Firewall möglich. Da es sich um einen Paketfilter handelt, werden bei geringerem Ressourcenbedarf höhere Durchsatzraten als bei der Verwendung von Proxies erreicht. Auch ist ein maskierender Paketfilter für die Klienten im lokalen Netz völlig transparent, so daß die Benutzer normale, nicht modifizierte Klienten benutzen können, ohne spezielle Vorgehensweisen einhalten zu müssen.

Allerdings werden auch Nachteile aus beiden Welten übernommen. So ist auch ein maskierender Paketfilter letztlich nur ein Paketfilter, der nicht dieselben Kontroll- und Proto-

kollierungsmöglichkeiten wie ein Proxy aufweist. Protokolle wie FTP verlangen darüber hinaus zumindest, daß das Masquerading weiß, wann eine zusätzliche Datenverbindung zu erwarten ist. Hier ist eine Unterstützung für das Protokoll nötig, die über das generische Umsetzen von Adressen deutlich hinausgeht. Zwar existiert für FTP in der Regel sehr wohl eine solche Unterstützung, es existieren aber Protokolle, für die ein Masquerading nicht möglich ist oder nicht unterstützt wird. Da das Masquerading außerdem eine Funktion des Kernels und kein eigenständiges Programm ist, ist es schwerer, eine nicht unterstützte Funktionalität nachzurüsten.

Redirection

Die in Abbildung 5-4 dargestellte Redirection bewirkt, daß eine Anfrage, die für einen Server im Internet bestimmt ist, auf einen Server auf der Firewall umgeleitet wird. Die wichtigste Anwendung im Firewalling stellt dabei sicherlich die Möglichkeit dar, *transparente Proxies* zu realisieren. Hierbei wird jeder Versuch, aus dem lokalen Netz auf einen Server im Internet zuzugreifen, abgefangen und auf einen Proxy auf der Firewall umgeleitet, der dann den eigentlichen Zugriff durchführt. Auf diesem Wege wird verhindert, daß Benutzer im Proxy realisierte Schutzmaßnahmen umgehen, indem sie einfach am Proxy vorbei direkt auf Rechner im Internet zugreifen.

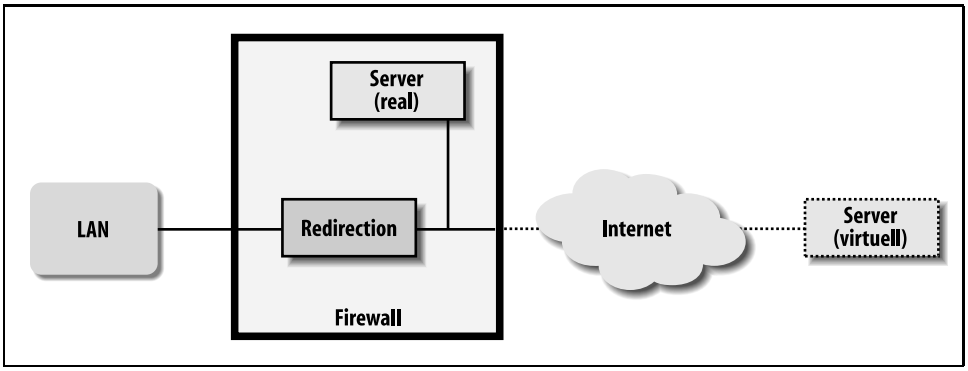

Abbildung 5-4: Redirection

Der Hauptvorteil transparenter Proxies liegt in dem geringeren Administrationsaufwand für die Rechner im lokalen Netz. Es müssen nicht auf jedem Rechner die aktuellen Proxy-Einstellungen eingetragen sein. Die Benutzung erfolgt vielmehr automatisch und für den Benutzer unsichtbar. Dies ist insbesondere dann wichtig, wenn der Betreiber der Firewall nur geringen Einfluß auf die Personen hat, welche die Rechner im lokalen Netz konfigurieren. Ein Internet-Provider wird seinen Kunden zum Beispiel nur wenige Konfigurationsmaßnahmen zumuten können. Je mehr Aufwand er von seinen Kunden verlangt, bevor diese surfen können, je eher werden diese sich nach einem Konkurrenten umsehen, bei dem das alles nicht so kompliziert ist.

Kombinierte Konzepte

Heutige Firewalls sind in der Regel weder reine Proxies noch reine Paketfilter. Wir wollen daher im folgenden zwei Konzepte betrachten, die sowohl Paketfilter als auch Proxies benutzen, um die Stärken beider Firewall-Konzepte zu verbinden.

Screened Host

Bei einer *Screened Host Firewall*, wie sie in Abbildung 5-5 zu sehen ist, werden Anfragen aus dem LAN entweder an den Proxyserver gestellt, der sie dann weitervermittelt, sie können aber auch direkt gestellt werden.

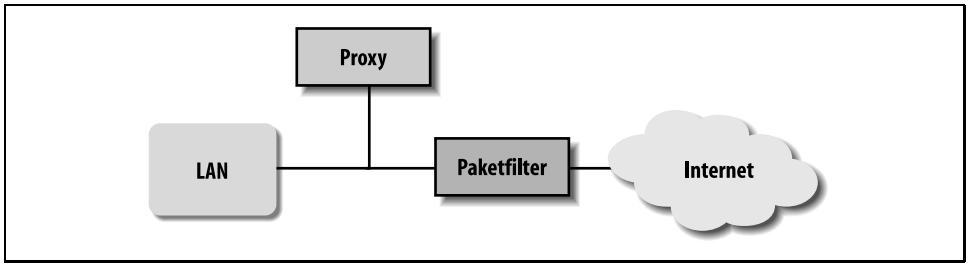

Abbildung 5-5: Screened Host

Der Paketfilter stellt sicher, daß

- nur als »sicher« definierte Protokolle benutzt werden,
- Verbindungen aus dem Internet (wenn überhaupt) nur an den Proxy gestellt werden können,
- der Verbindungsaufbau für bestimmte Protokolle nur vom Proxyserver aus möglich ist und
- nur für bestimmte definierte Protokolle ein direkter Verbindungsaufbau aus dem lokalen Netz möglich ist.

Vorzugsweise wird der direkte Verkehr von Paketen aus dem LAN ins Internet auf ein Minimum von als »sicher« angesehenen Protokollen beschränkt. In diesem Fall ist der Proxy der Rechner, der am auffälligsten ist und am besten abgesichert werden muß. Während ein gut konfigurierter Router schwer angreifbar ist, muß ein Proxyserver Proxies für diverse Netzwerkdienste bereithalten. Ist einer dieser Proxies angreifbar, weil er zum Beispiel anfällig für Speicherüberläufe ist, so besteht die Gefahr, daß der Rechner von einem Angreifer übernommen wird. In diesem Fall wäre die Firewall wertlos, da der Angreifer nun den Proxyserver kontrolliert und von diesem aus Angriffe auf die Rechner im lokalen Netz starten kann.

Bei Proxyservern ist diese Gefahr relativ gering. Diese Anordnung ist daher durchaus gebräuchlich, um kleinere LANs mit dem Internet zu verbinden. Dabei werden in vielen Fällen die Funktionen von Paketfilter und Proxyserver in einem Rechner zusammenge-

legt. Dies entspricht auch der Grundkonfiguration, die im nächsten Abschnitt vorgestellt werden soll.

Problematischer wird es, wenn nicht nur Klienten aus dem LAN auf Server im Internet zugreifen sollen, sondern auch ein eigener Server Dienste für das Internet anbieten soll, beispielsweise ein Webserver. Solche Server sind oft viel komplexer und damit deutlich anfälliger für Angriffe. Noch dazu ist es hier normal, daß die Zugriffe von außen nach innen erfolgen anstatt wie bisher von innen nach außen. Damit kann auch kein Masquerading im Paketfilter eingesetzt werden. Der Server ist im Internet deutlich sichtbar. Ihn im lokalen Netz zu positionieren, würde bedeuten, dem Angreifer einen Brückenkopf mitten im eigenen System auf dem Silbertablett zu präsentieren. Die Alternative bestünde darin, ihn außerhalb des geschützten Bereichs aufzustellen, womit dieser sensible Rechner aber völlig ungeschützt wäre.

Screened Subnet

Um die Gefahren durch eine Kompromittierung des Proxyservers zu verringern, kann ein zweiter Paketfilter zum Einsatz kommen, der zwischen Proxyserver und lokalem Netz plaziert wird. Abbildung 5-6 veranschaulicht dies. So entsteht ein zusätzliches Teilnetz, in dem Server plaziert werden können, die zwar geschützt werden müssen, denen die Rechner im lokalen Netz aber nicht »vertrauen« können. In der Fachliteratur wird der Bereich zwischen den Paketfiltern häufig als *demilitarisierte Zone*, kurz *DMZ*, bezeichnet. Hier können neben einem oder mehreren Proxyservern auch Web-, Mail- und FTP-Server untergebracht werden.

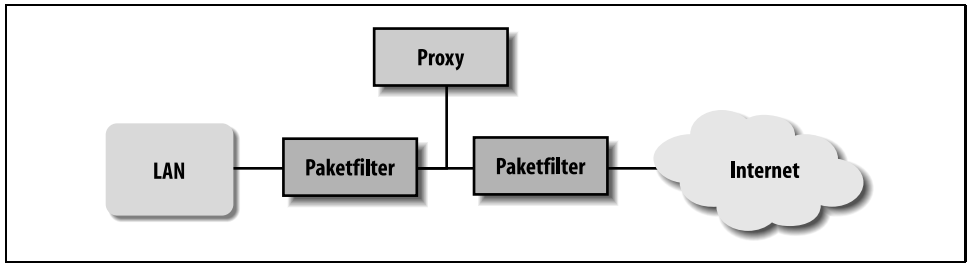

Abbildung 5-6: Screened Subnet

Grundsätzlich gilt, daß Verbindungen jeweils nur aus dem lokalen Netz in die DMZ aufgebaut werden dürfen, nicht aber aus der DMZ in das lokale Netz. Für die Rechner im lokalen Netz besteht prinzipiell kein Unterschied zwischen den Rechnern in der DMZ und denen im Internet. Auch ist es nicht verkehrt, auf dem inneren Paketfilter Masquerading einzusetzen.

Man kann das Prinzip noch weiter treiben, indem man bestimmte Bereiche des lokalen Netzes gegen den Rest des Netzes durch weitere Firewalls abschirmt. Dies kommt in der Realität aber nur in großen Organisationen vor, wenn z. B. ein weltweit operierender Konzern seine Forschungsabteilungen vor allzu neugierigen Angestellten schützen will.

Eine Variante, die in der Praxis häufiger anzutreffen ist, besteht in der Zusammenlegung von äußerem und innerem Paketfilter. Dies geschieht, indem als Paketfilter ein Rechner mit drei Netzwerkkarten eingesetzt wird. Je eine Karte ist dabei mit dem Internet, dem lokalen Netz und der DMZ verbunden. Dies kompliziert zwar die Aufstellung der Paketfilterregeln, wenn diese aber sorgfältig durchdacht wird, ist die in Abbildung 5-7 dargestellte Architektur der in Abbildung 5-6 skizzierten durchaus ebenbürtig.

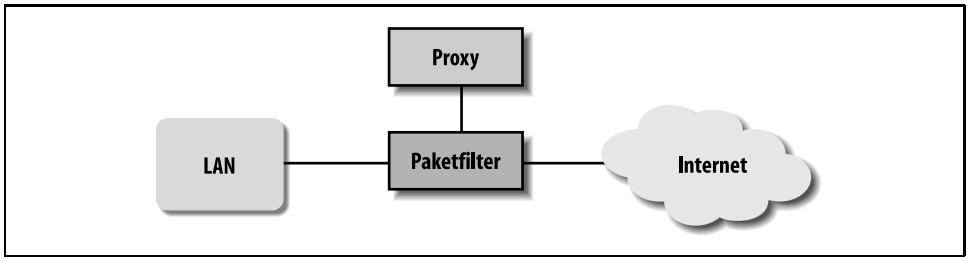

Abbildung 5-7: Screened Subnet mit einem Paketfilter

Dies ist allerdings nicht dasselbe wie die im vorigen Abschnitt erwähnte Zusammenlegung von Router und Proxyserver auf einem Rechner. Wir haben es hier immer noch mit einem Screened Subnet zu tun, während wir ansonsten einen Screened Host vor uns hätten. Man geht bei dieser Unterscheidung davon aus, daß ein reiner Paketfilter ohne zugreifbare Serverdienste deutlich schwerer anzugreifen ist als ein Proxyserver, der diverse Dienste anbietet. Darum befindet sich in einer Screened-Subnet-Architektur immer ein reiner Paketfilter zwischen Proxyserver und lokalem Netz.