

Ralf Spenneberg

Linux-Firewalls mit iptables & Co.

Sicherheit mit Kernel 2.4 und 2.6
für Linux-Server und -Netzwerke



 ADDISON-WESLEY

An imprint of Pearson Education

München • Boston • San Francisco • Harlow, England
Don Mills, Ontario • Sydney • Mexico City
Madrid • Amsterdam



2 Firewall-Technologien

Es gibt viele verschiedene Technologien, die beim Aufbau einer Firewall eingesetzt werden können. Dieses Kapitel stellt die wichtigsten vor und beschreibt ihre Funktionsweise sowie ihre Vor- und Nachteile.

2.1 Paketfilter

Der Paketfilter ist die einfachste Technologie für die Implementierung einer Firewall. Die erste Implementierung geht auf Jeffrey C. Mogul im Jahr 1989 zurück (Simple and flexible datagram access controls for Unix-based gateways: <ftp://ftp.digital.com/pub/Digital/WRL/research-reports/WRL-TR-89.4.pdf>). Mogul stellte auf der USENIX Summer Conference im März 1989 mit dem Programm `screend` einen ersten Paketfilter vor. Der `screend` ist ein Userspace-Programm, das die Entscheidung trifft, ob ein Paket passieren darf oder nicht.

Im Grunde ist ein Paketfilter ein intelligenter Router. Während ein normaler Router lediglich entscheidet, wohin ein Paket geschickt wird, analysiert ein Paketfilter zusätzlich das Paket und bestimmt, ob ein Paket überhaupt weitergesendet werden darf oder ob es verworfen werden muss.

Da diese beiden Aufgaben eng miteinander verwandt sind, werden seit damals fast alle Router mit zusätzlichen Paketfilterfunktionalitäten ausgestattet. Auch die meisten Netzwerkbetriebssysteme verfügen seit Jahren über derartige Paketfilter.

Die meisten Paketfilter sind IP-Paketfilter. Das bedeutet, sie beschränken ihre Betrachtung auf IP-Pakete. Dort analysieren sie den IP-Header. Die meisten Paketfilter können zusätzlich im Falle eines UDP-, TCP- oder ICMP-Paketes auch deren Header analysieren. Hierzu nutzen die Paketfilter üblicherweise ein Regelwerk, das sie Regel für Regel bei jedem Paket testen. Trifft eine Regel zu, so wird die mit der Regel verbundene Aktion ausgeführt. Die üblicherweise unterstützten Aktionen sind das Annehmen und Routen des Pakets, das Verwerfen des Pakets und das Ablehnen des Pakets (Senden einer Fehlermeldung an den Absender).

Ein klassischer zustandsloser Paketfilter kann nicht erkennen, ob ein Paket zu einer aufgebauten Verbindung gehört oder nicht. Jedes Paket wird einzeln für sich betrachtet und analysiert. Der Zusammenhang, in dem das Paket auftritt, ist belanglos. Der Administrator eines Paketfilters muss daher sowohl die Pakete des Clients als auch die Pakete des potenziellen Servers in seinem Regelwerk berücksichtigen.

2.2 Zustandsorientierte Paketfilter

Die ersten zustandsorientierten kommerziellen Paketfilter wurden 1993 verfügbar. Das bekannteste, heute noch im Einsatz befindliche Produkt ist die Check Point Firewall-1. Dieses Produkt besaß eine einfache Verbindungstabelle, in der sich die Firewall sämtliche Verbindungen merkte, die von innen aufgebaut worden waren, und automatisch sämtliche Antworten zuließ.

Die erste frei verfügbare Variante eines zustandsorientierten Paketfilters war IP-Filter (IPF) 3.0 (1996) von Darren Reed. Ebenfalls 1996 erschien mit der SF Firewall (<http://www.ifi.unizh.ch/ikm/SINUS/sf-doc/impl.htm>) eine erste Implementierung für Linux. Diese wurde in den weiteren Jahren zur Sinus Firewall mit grafischer Oberfläche weiterentwickelt.

Heute besitzt Linux mit Iptables/Netfilter einen der mächtigsten zustandsorientierten Paketfilter, die verfügbar sind.

Allen zustandsorientierten Paketfiltern ist gemeinsam, dass sie sich in einer Tabelle die aufgebauten und von dem Regelsatz erlaubten Verbindungen merken und alle weiteren Pakete dieser Verbindungen mehr oder weniger automatisch erkennen und zulassen. Die explizite Erkennung dieser Pakete durch einzelne Regeln ist nicht erforderlich. Dies vereinfacht zum einen die Definition und die Wartung der Regeln erheblich, denn die Zahl der Regeln schrumpft mindestens um die Hälfte. Zum anderen erhöht es auch sehr stark die Sicherheit der Firewall. Es ist nun nicht erforderlich, grundsätzlich jedes mögliche Antwortpaket eines jeden möglichen Kommunikationspartners durchzulassen. Die zustandsorientierte Firewall beschränkt dies auf die Pakete, die sie als Teil der Verbindung tatsächlich erkennt.

Eine Weiterentwicklung ist die Inspection- oder auch Deep-Inspection-Firewall. Hierbei handelt es sich um Systeme, die die Pakete noch genauer analysieren und – besonders als Deep-Inspection-Systeme – eine Kombination aus Paketfilter und Intrusion-Detection-System oder Virens Scanner darstellen. Hierzu prüfen die Systeme, ob über die Verbindung ein Angriff oder ein Virus übertragen wird. Auf Grund der Natur eines Paketfilters und der hohen Anforderung an die Geschwindigkeit, sind die Deep-Inspection-Systeme jedoch meist nicht so gut wie eine Kombination aus einem reinen Paketfilter und einem Virusscanner oder einem IDS.

Dennoch haben sich die Paketfilter in vielen Bereichen gegenüber den Proxys durchgesetzt, da die Implementierung preiswert und einfach möglich ist, der Paketfilter auch im Bereich der Gigabit-Netzwerke eine leistungsgerechte Filterung durchführen kann und als zustandsorientierter Paketfilter mit Inspection oder sogar Deep-Inspection-Analyse für viele Anwendungsfälle ausreichend Sicherheit bietet.

2.3 Curcuit Relay

Das Curcuit Relay ist ein Proxy auf der Transport-Schicht, der das verwendete Applikationsprotokoll nicht versteht, sondern die Daten auf der Transport-Schicht austauscht. Dieses Prinzip wurde 1996 in dem SOCKS-Protokoll weiterentwickelt und

verbessert. Das SOCKSv5-Protokoll wurde in dem RFC 1928 definiert und festgelegt. Heute sind noch zwei Versionen des Protokolls im Einsatz: SOCKSv4 und SOCKSv5. SOCKSv5 unterstützt auch Applikationen, die das UDP-Protokoll nutzen.

Der besondere Pluspunkt für den Einsatz des SOCKS-Proxys ist die einfache Implementierung. Wenn der Quelltext einer Applikation verfügbar ist, kann diese sehr einfach mit der Fähigkeit ausgestattet werden, den SOCKS-Proxy zu nutzen. Der einfache Socket-Aufruf muss lediglich durch den SOCKS-Socket-Aufruf ersetzt werden. Viele kommerzielle Programme, wie der Microsoft Internet Explorer, enthalten bereits die hierfür notwendige Logik.

2.4 Application-Level-Gateway (Proxy)

Dies ist die klassische Firewall-Technologie, wie sie von Marcus Ranum (siehe Kapitel 1) in der ersten Firewall implementiert wurde. Hierbei handelt es sich um Proxys, die auf der Schicht des Applikationsprotokolls arbeiten. Daher benötigt ein derartiges Gateway für jedes Applikationsprotokoll einen eigenen Proxy. Diese Application-Level-Gateways können daher meist nur eine beschränkte Anzahl von Protokollen filtern. Obwohl Sie dies als einen Nachteil dieser Technologie ansehen können, bietet diese Technologie auch Vorteile. Die Application-Level-Gateways implementieren das Applikationsprotokoll und können daher auch die korrekte Verwendung prüfen. Speziell bei komplizierten und gefährlichen Protokollen wie FTP oder HTTP können diese Proxys durch selektive Unterstützung des Protokolls die gefährlichen Befehle herausfiltern und deren Verwendung unterbinden. Ein reiner Paketfilter kann dies nicht. Hier sind zusätzliche Funktionen in Form eines Inspection- oder Deep-Inspection-Paketfilters erforderlich. Während jedoch das Application-Level-Gateway nur die erlaubten Funktionen implementiert (Positiv-Liste) und alle weiteren Verwendungen des Protokolls automatisch unterbindet, muss ein Deep-Inspection-Paketfilter die bösartige Verwendung des Protokolls erkennen und verhindern (Negativ-Liste). Der Deep-Inspection-Paketfilter kann daher nur bekannte Angriffe erkennen und verhindern, während ein Application-Level-Gateway weiter reichende Sicherheit bietet.

Dennoch haben sich in den letzten Jahren die Paketfilter in vielen Bereichen durchgesetzt. Dies hängt mit den vielen verschiedenen, teilweise sehr komplizierten modernen Protokollen zusammen. Für jedes dieser Protokolle ist ein eigener Proxy erforderlich. Deren Implementierung ist häufig so aufwendig und kompliziert, dass sie entweder gar nicht vorgenommen wird oder der Proxy anschließend selbst zu einem Sicherheitsrisiko wird, da die Implementierung Fehler aufweist.

2.5 Network Address Translation

Die Network Address Translation (NAT) wurde zunächst als Antwort auf die beschränkte Anzahl von IP-Adressen im Internet entwickelt. Mit Hilfe von NAT können viele Rechner über dieselbe IP-Adresse auf Dienste im Internet zugreifen. Hier-

für wird NAT üblicherweise auf einem Router oder der Firewall implementiert. Die IP-Adresse aller in das Internet ausgehenden Pakete wird dann von dem Router oder der Firewall durch die IP-Adresse des Routers/der Firewall selbst ersetzt. Dabei verwendet der Router/die Firewall für jede Verbindung einen eindeutigen ungenutzten lokalen Port. Die Information über die Verbindung und die verwendeten IP-Adressen und Ports speichert das Gerät in einer Tabelle ab. Sobald ein Antwortpaket das Gerät erreicht, liest das Gerät das echte Ziel aus der Tabelle ab und leitet das Paket an den entsprechenden Rechner weiter. Nach Beendigung der Verbindung oder nach Ablauf eines Zeitgebers wird die Verbindung aus der Tabelle entfernt. Dies ist ein Source-NAT, bei dem die Quelle der Verbindung *genattet*, also die Quelladresse ausgetauscht, wird.

Das Source-NAT wird häufig als Sicherheitsfunktion eingesetzt, da es auch wirksam den Zugriff auf Systeme hinter dem Router/der Firewall unterbindet. Da deren IP-Adresse unbekannt ist, ist kein Zugriff möglich, außer es existiert ein entsprechender Eintrag in der NAT-Tabelle, der Pakete, die von außen auf einer bestimmten IP/Port-Kombination ankommen, nach innen zu einem System weiterleitet. Dies ist das Destination-NAT. Dabei wird die Zieladresse einer Verbindung ausgetauscht.

Grundsätzlich ist ein Destination-NAT unter Sicherheitsaspekten kritischer zu betrachten, da es durch die Firewall den Zugriff auf interne Systeme erlaubt. Bei dem Einsatz des Destination-NAT sollte daher das interne System, auf das der Zugriff freigegeben wird, zusätzlich isoliert sein. Dies erfolgt in den meisten Umgebungen durch die Platzierung dieser Systeme in einer DMZ (siehe Abschnitt 3.2).