INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Fortgeschrittenenpraktikum

Neugestaltung des Firewall Versuchs für das Rechnernetzepraktikum

Leonhard Bär

Aufgabensteller:	Prof. Dr. Heinz-Gerd Hegering
Betreuer:	Markus Garschhammer Annette Kostelesky
Abgabetermin:	

Inhaltsverzeichnis

Kapitel 1

Einleitung

Das am Lehrstuhl angebotene Rechnernetzepraktikum (RNP) beinhaltet im Block „IP-Netze“ einen Versuch, welcher sich mit der Realisierung einer Firewall, befasst. Hauptaufgabe dieses Versuchnachmittages ist die Absicherung eines Teilnetzes des RNP mittels eines Paketfilters. Bisher kam hierbei die Software *Firewall-1* von checkpoint auf einem Windows NT System zum Einsatz.

Die große Zahl an Informatik Studenten der letzten Semester führte zu einem erheblichen Engpass an Praktikumsplätzen. Aus diesem Grund wurde im Wintersemester 01/02 die Entscheidung getroffen, die Versuchsplätze im RNP zu verdoppeln. Im Zuge dieser Verdoppelung fiel auch die Entscheidung, den Firewall Versuch zu überarbeiten. Hierbei sollten unter anderem folgende Punkte beachtet werden:

- Das erstellte System sollte ohne Probleme und zusätzliche Software geklont werden können. Dies bietet die Möglichkeit eines einfachen backups und der einfachen Verdoppelung der Versuchsplätze.
- Sowohl für das Betriebssystem, als auch für die Firewall Software sollte eine lizenzfreie Lösung gefunden werden.
- Die Praktischen Versuche müssen an die neue Platform angepasst werden. Desweiteren sollten aktuelle Entwicklungen bei der Erstellung der Filterregeln berücksichtigt werden.
- Falls möglich sollte der Versuch auf einer schon im RNP verwendeten Plattform laufen, um weitere Einarbeitungszeit der Administratoren zu minimieren.

Diese Neugestaltung führte zu einem Linux System mit Kernel 2.4, welches auf einer SUSE 7.2 Distribution basiert. Die Versuche wurden entsprechend für die Verwendung von *iptables* (dem Paketfilter von Linux 2.4) angepasst und weitere Aufgaben wurden hinzugefügt. Der Versuch wurde anschließend ein Semester lang getestet und die Aufgaben, aufgrund von Verbesserungsvorschlägen der Praktikumsteilnehmer und Praktikumsbetreuer, überarbeitet.

Kapitel 2

Ermitteln einer neuen Firewall Platform

Eine der Hauptaufgaben der Arbeit bestand darin, eine neue Platform für die Firewallarchitektur zu finden. Hierbei sollten die schon in der Einleitung angesprochenen Punkte, wie Anschaffungskosten, Modernität und Wartungsarbeit berücksichtigt werden. Im Folgenden werden nun kurz die betrachteten Platformen vorgestellt und die Kriterien angegeben, die zum Ausschluss dieser Systeme führten.

2.1 Kommerzielle Produkte

Diese Produkte widersprechen ganz klar der Prämisse, eine lizenzfreie Lösung zu finden. Trotzdem ist es sinnvoll, auch diese Gruppe kurz zu betrachten.

Für eine kommerzielle Firewalllösung spricht vor allem die einfache Bedienung und die Unterstützung bei der Erstellung der Sicherheits Regeln. So werden kommerzielle Firewalls in der Regel mit einem leistungsstarken grafischen Interface ausgeliefert, welches die Erstellung der Regeln mittels drag & drop erlaubt. Die hierfür nötige, oft objektorientierte, Repräsentation der einzelnen Netzwerkressourcen wird von der Software automatisch ermittelt und zu Verfügung gestellt. Desweiteren werden sogenannte „wizards“ angeboten, welche mittels eines Frage und Antwort Vorgangs die nötigen Informationen ermitteln und daraufhin einen Regelsatz für die Firewall erstellen, bzw. diesen erweitern.

Der größte Nachteil dieser Lösungen sind aber nicht die Anschaffungskosten, sondern die Plattform, auf welcher sie laufen. Im Normalfall ist dies Windows 2000 oder Windows NT. Für diese sind zum einen oft wesentlich höhere Lizenzkosten, als für die Firewallsoftware selbst, zu zahlen. Zum anderen gestaltet sich bei diesen Betriebssystemen ein Wechsel der zu Grunde liegenden Hardware (und damit z.B. das einfache Klonen der Versuchsplätze) als äußerst schwer, bzw. ist nur mit zusätzlicher Software durchführbar. Ähnliches gilt auch für die Wiederherstellung eines nicht mehr funktionierenden oder „jungfräulichen“ Systems. Ohne zusätzliche Software ist hier normalerweise eine Neuinstallation des Betriebssystems notwendig, bevor eine evtl. mitgelieferte Backup Software verwendet werden kann.

2.2 Hardware Lösungen

Für eine Hardwarelösung spricht vor allem deren einfach Installation. Das Gerät ist einfach in die vorhandene Netzwerk Infrastruktur einzubinden und nach Angabe der entsprechenden Adressbereiche kann damit gearbeitet werden. Ähnlich den kommerziellen Lösungen, wird neben der Konfiguration mittels Konsole, in der Regel auch eine grafische Konfigurationsmöglichkeit (z.B. mittels Browser) zu Verfügung gestellt. Hauptvorteil einer Hardwarelösung ist deren hoher Durchsatz in der Verarbeitung der Datenpakete. Dieser kann aber in der Praktikumsumgebung nicht ausgespielt werden, da hier der Schwerpunkt in der Konfiguration der Firewall und nicht in deren Performance liegt.

Neben den hohen Anschaffungskosten, ist vor allem der kurze Produkt Lebenszyklus für das nicht Verwenden einer hardware Lösung ausschlaggebend. Ein Produkt ist selten länger als 2 Jahre auf dem Markt. Bei einem Defekt, den es nicht mehr „lohnt“ zu reparieren bzw. der nicht repariert werden kann, ist ein neues Gerät unumgänglich, welches einige Probleme mit sich bringt. Zwar wird bei einer Neuanschaffung wahrscheinlich die gleiche Funktionalität wie beim alten Produkt geboten werden, aber mit ziemlicher Sicherheit wird sich die Art der Konfiguration ändern. Eine Überarbeitung der Praktikumsunterlagen wird daher unumgänglich.

2.3 Open Source Lösungen

Solche Systeme sind frei Verfügbar und erfüllen somit die Prämisse einer lizenzfreien Lösung. Desweiteren sind sie auf handelsüblicher PC-Hardware lauffähig. Diese ist zum einen einfach und kostengünstig zu erwerben, zum anderen kann man davon ausgehen, dass die entsprechende Software auch in einigen Jahren auf einer dann aktuellen PC-hardware lauffähig ist.

Abstriche müssen hingegen bei der Bedienung akzeptiert werden. So wird bei diesen Lösungen der Paketfilter grundsätzlich per Kommandozeile bzw. per Shell-Skript konfiguriert. Jede weitere Funktionalität, wie eine grafische Oberfläche oder das automatischen Ermitteln der Netzwerkresourcen, muss also mittels zusätzlicher Software realisiert werden.

2.3.1 BSD Derivate

Hier wurden die zwei bekanntesten Ableger freeBSD und openBSD nur kurz untersucht. Eigentlich wären diese zwei Systeme für das Erstellen einer Firewalllösung geeignet, so wird z.B. openBSD extra dafür entwickelt, dass keine Sicherheitslücken in der herausgegebenen Distribution vorhanden sind.

Für den Einsatz im Praktikum waren und sind sie aber nicht empfehlenswert. Dies liegt daran, dass die zwei Betriebssysteme, sowohl in der Forschung, als auch in der Lehre, an der LMU und TU-München kaum bis gar nicht genutzt werden. Für die Praktikumsteilnehmer ist dies nicht besonders hinderlich, da sie sich meist mit dem Thema Firewall Konfigurieren zum ersten Mal auseinander setzen. Welches Programm sie hierfür erlernen ist im Prinzip egal, wichtig für sie ist das Erlernen der Konzepte. Ein den Studenten vertrautes System wäre aber natürlich vorzuziehen.

Problematisch hingegen ist der Einsatz für die Praktikumsleiter und die Tutoren. Zum einen sind sie mit der Administration des Betriebssystems nicht vertraut, wodurch das Beheben von eventuell auftretenden Fehlern erheblich länger dauern würde als bei einem vertrauten System. Zum anderen würde die Verwendung von BSD die Heterogenität der Betriebssysteme im Praktikum weiter erhöhen. Da normalerweise aber zwei Gruppen gleichzeitig das Praktikum an verschiedenen Versuchen durchführen und diese wöchentlich wechseln, führt dies dazu, dass sich die Tutoren jedesmal auf ein neues Betriebssystem einstellen müssen. Der hierfür benötigte Zeitaufwand fehlt dann natürlich bei der Betreuung der Studenten.

2.3.2 Linux Systeme

Eine auf einem Linux System basierende Lösung wurde von Anfang an in Betracht gezogen. Zum einen bietet diese natürlich die oben angesprochenen Vor- und Nachteile einer Open-Source-Lösung. Viel wichtiger ist aber die Tatsache, dass Linux das bevorzugte Betriebssystem am Institut für Informatik ist. Hierdurch sollte sich die Einarbeitungszeit sowohl für die Tutoren und Studenten, als auch der Administrationsaufwand für den Praktikumsleiter auf ein Minimum reduzieren.

Zum Zeitpunkt der Evaluation waren zwei verschiedene Linux Kernel aktuell, welche unterschiedliche Funktionalität für den Aufbau eines Paketfilters bieten. Zum einen Kernel 2.2, der mittels *ipchains* erlaubt, statische Filterregeln zu erzeugen. Zum anderen Kernel 2.4, welcher *iptables* verwendet und unter anderem die Möglichkeit bietet, auch dynamische Filterregeln zu erzeugen. Dies erleichtert die Entwicklung von Filterregeln um einiges und zusätzlich sind bestimmte Regelsätze erst mit dieser Technik möglich.

Desweiteren wurde im Design von *iptables* das Konzept der *Network Address Translation* (NAT) mitberücksichtigt, so dass die Erstellung der Regeln hierfür, übersichtlicher und damit um einiges leichter wurden. Realisiert wurde dies zum einen durch die Aufteilung der verarbeitenden Pakete in zwei Teilströme. Der eine beinhaltet alle Pakete, die direkt an die Firewall gehen bzw. von ihr kommen, der andere alle Pakete, welche nur weitergeleitet werden. Zum anderen wurde die Möglichkeit geschaffen, alle Pakete, kurz nachdem sie die Firewall erreicht haben bzw. kurz bevor sie diese verlassen, noch ein zweites Mal zu manipulieren. Hierdurch ist es möglich, die Filterregeln völlig unabhängig von den evtl. benötigten NAT Regeln zu entwickeln.

Neben diesen zwei großen Vorteilen, der dynamischen Filterung und der Unabhängigkeit von NAT, bietet die Verwendung eines 2.4 Kernels mit *iptables* noch weitere Vorteile. So existiert die Möglichkeit mittels Kernelmodulen z.B. syn-flood Attacken zu erkennen und zu verhindern oder IP-Pakete, deren Adresse nicht korrekt aufgebaut ist, zu ignorieren. Desweiteren bietet *iptables* die Möglichkeit auf *Denial of Service* Attacken zu reagieren. Dies wird z.B. dadurch realisiert, dass nur noch eine bestimmte Anzahl von Paketen pro Zeiteinheit akzeptiert werden, falls ein entsprechendes Ereignis auftritt, welches auf einen Angriff schließen lässt.

Wegen den oben genannten Eigenschaften wird *iptables* bei neu aufgesetzten Systemen immer häufiger zum Einsatz kommen und „ältere“ Systeme, welche auf *ipchains* basieren, nach und nach ersetzen. Es bietet sich daher an, auch im Praktikum diese Linux Version zu verwenden, wodurch sich die Studenten gleich mit einer aktuellen Technik vertraut machen.

Kapitel 3

Aufbau des Firewall Rechners

Im folgenden wird der Aufbau des Firewall Rechners beschrieben, wie sie zur Zeit im RNP im Einsatz ist. Hierbei wird insbesondere auf die Installation des *fwbuilders* eingegangen, da sich diese zum Zeitpunkt der Erstellung (Frühjahr 2002) noch etwas komplizierter gestaltete. Anschließend wird dargelegt, welche Vorgehensweise sich für einen aktuellen (Frühjahr 2003) bzw. zukünftigen Neuaufbau der Firewall anbietet.

3.1 Aktuelle Konfiguration

Das aktuelle Firewall System basiert auf einer SUSE 7.2 Distribution und *fwbuilder* in der Version 1.0.1. Auf eine zu dieser Zeit aktuellen SUSE 7.3 Distribution wurde aufgrund einer Hardware bzw. Software Inkompatibilität der Grafikkarte nicht zurückgegriffen.

3.1.1 Installation der Komponenten

Für die Installation wurde die Option *minimales graphisches System* gewählt. Desweiteren sind noch folgende Pakete einzeln anzuwählen, welche mit Ausnahme von iptables, für die *fwbuilder* software nötig sind.

- iptables
- gdk-pixbuf
- libsigc++
- libxml2
- ucd-snmp
- gktmm

Da die meisten Studenten nur mit der Bedienung eines Editors vertraut sind und oft wenig Motivation bestand, die Bedienung eines weiteren Editors zu erlernen, empfiehlt es sich, einige weitere Editoren, wie *Pico, Joe* oder *Emacs* zu installieren.

Da der fwbuilder ein noch relativ neues Projekt ist, mussten vor dessen Installation noch folgende Pakete neu eingespielt bzw. auf einen aktuelleren Stand gebracht werden:

Paket	mindestens benötigte Version	verwendete Version
libstdc	2.9	3.0.4-1
libxml2	2.4.10	2.4.18-1
libxslt	1.0.7	1.0.14-1

Für die endgültige Installation des *fwbuilder* wurden folgende *rpm-Pakete* verwendet, welche von der *fwbuilder* Homepage [fwb] stammen und zur schnellen Wiederfindung unter `/baer03/sourcen/` abgelegt wurden.

- libfwbuilder-0.10.5-2suse73.i386.rpm
- fwbuilder-pf-1.0.1-2suse73.i386.rpm
- fwbuilder-ipf-1.0.1-2suse73.i386.rpm
- fwbuilder-iptables-1.0.1-2suse73.i386.rpm
- fwbuilder-1.0.1-2suse73.i386.rpm

Es kamen suse73 Pakete zum Einsatz, da aktuelle *fwbuilder* Pakete nur für diese Version zu Verfügung standen. Dies hatte aber keine weiteren Auswirkungen, da sich die zwei SUSE Distributionen nur durch aktuellere Software Versionen unterscheiden.

Der dritte praktische Versuch (siehe Seite 19) befasst sich mit Kerneloptionen, welche den Betrieb als Firwall erleichtern. Diese Optionen sind in der Datei *ip-sysctrl.txt* beschrieben. Leider wird sie bei der SUSE Distribution nicht automatisch mitinstalliert. Es ist daher notwendig, diese Datei aus dem Internet (z.B. unter ..) zu beschaffen und entsprechend unter (`/usr/src/linux/Documentation/networking/`) abzulegen.

3.1.2 Konfiguration des Systems

Für eine korrekte Arbeitsweise von *iptables* ist es notwendig, einige Kernel Module zu laden. Hierzu wurde das shell-skript *grundkonf* (siehe Seite 16) erstellt und in `/etc/init.d/` abgelegt. Damit dieses Skript bei jedem Neustart der Firewall ausgeführt wird, ist in `/etc/init.d/rc5/` noch ein symbolischer Link darauf anzulegen.

Um jeder Praktikumsgruppe den gleichen Ausgangszustand der Firewall zu bieten wurde dieses Skript noch um einige Funktionen erweitert. Zum einen werden alle evtl. noch vorhandenen *iptables* Regeln gelöscht und die Standard policy auf accept gestellt. Hiermit wird sichergestellt, dass die Firewall nach dem booten als „normaler“ Router arbeitet und den sonstigen Praktikumsverlauf nicht weiter beeinflusst. Zum anderen wird das Skript-Skelett *firewall* (siehe Seite 17), in welches die Studenten ihre Regeln schreiben sollten, neu angelegt.

3.2 Zukünftige Neugestalltung

Aktuell stehen den Praktikumsteilnehmern zwei Firewall Systeme (Versuchsaufbauten) zu Verfügung. Desweiteren liegt ein Image der Festplatte als Backup auf dem Praktikumsserver *pcrnp10*. Falls also eines der beiden Systeme wider erwartend nicht funktionieren sollte, ist durch Klonen des anderen Systems, bzw. durch das Rückspielen des Backups, ein „einfaches“ Wiederherstellen eines funktionierenden Versuches möglich.

Fallen diese Wiederherstellungsmaßnahmen aus, oder ist aus anderen Gründen ein Neuaufbau / eine Überarbeitung des Versuches nötig, so bietet sich ein genaues Wiederherstellen der Konfiguration auf Basis der SUSE 7.2 Distribution nicht an.

So stehen mittlerweile auf der *fwbuilder*Homepage [fwb] einige Anleitungen zu Verfügung, welche die Installation auf aktuellen Distributionen Schritt für Schritt beschreiben. Desweiteren dürften bei aktuellen Distributionen diejenigen Pakete auf dem neuesten Stand sein, welche für die Installation des *fwbuilder* notwendig sind. Somit würde das manuelle Neueinspielen der entsprechenden Pakete entfallen und eine weitere Fehlerquelle beseitigt werden. Eventuell wird die *fwbuilder*-Software sogar mit einer der zukünftigen Distributionen mitgeliefert, womit sich der Aufwand einer Installation auf ein Minimum beschränken würde.

Kapitel 4

Praktische Aufgaben

Nachdem die Entscheidung für Linux 2.4 als neue Platform feststand, war es notwendig, die praktischen Aufgaben zu überarbeiten und insbesondere die Lösungen an das neue System anzupassen. Hierbei war zu beachten, dass die *Firewall-1* von checkpoint eher einer kompletten Sicherheitsarchitektur entspricht, *iptables* hingegen ein reiner Paketfilter ist. Ein Teil der Aufgaben, wie zum Beispiel die Authorisierung von Benutzern an der Firewall oder zeitabhängige Filterregeln, war deshalb nicht wiederzuverwenden.

Um den Versuch weiterhin anspruchsvoll zu halten, aber auch um die Möglichkeiten einer Linux / *iptables* Kombination auszuschöpfen wurde deshalb der Versuch in 4 Bereiche aufgeteilt. Im ersten beschäftigen sich die Studenten mit dem Firewallrechner und ermitteln, welche Möglichkeiten ihnen Linux zum Realisieren einer Firewall bietet. Der zweite Teil beinhaltet eine Teilmenge der Aufgaben aus den „alten" Aufgabenstellungen und beschäftigt sich mit statischen Filterregeln. Im dritten Teil werden dann die Möglichkeiten von *iptables* genutzt, dynamische Regeln zu verwenden. Der letzte Teil bietet dann noch eine kleine Einführung in die grafisches Benutzeroberfläche *fwbuilder* [fwb].

Während diese vier Teilbereiche, welche in den folgenden Kapiteln noch etwas genauer beschrieben werden, die Thematik des Versuches wiederspiegeln, wurde bei der Erstellung der Aufgaben versucht, die folgenden Prämissen einzuhalten. Zum einen wurden die Aufgaben so gestaltet, dass deren Komplexität im Laufe des Versuchtages kontinuierlich ansteigt. Hierdurch sollte gewährleistet sein, dass die Aufgaben auch „Interessant" bleiben und nicht zu einem „einfachen" abarbeiten verkommen. Zum anderen wurden die Aufgaben in jedem der vier Bereiche so gewählt, dass zunächst eine relativ einfache Aufgabe die Thematik des aktuellen Bereiches kurz vermittelt. Anschließend wird mit etwas komplizierteren Aufgaben versucht, diese Thematik noch zu vertiefen.

4.1 Der Firewall Rechner

In diesem ersten Versuchsblock setzen sich die Studenten mit dem Firewall Rechner und dem Linux System an sich auseinander. Hierzu dienen zunächst die Befehle *lsof* und *netstat*, welche genaue Auskünfte über die aktuell laufenden Prozesse liefern, welche im Zusammenhang mit dem Netzwerk stehen. Anschließend wird mit Hilfe der Datei *ip-sysctl.txt* ermittelt, welche Möglichkeiten der Linux Kernel zur Unterstützung einer firewall bietet.

Bevor die die Studenten im nächsten Versuchsblock einfache Filterregeln erstellen, wird in einer etwas umfangreicheren Aufgabe der *iptables* Paketfilter noch einmal genauer untersucht. Hierzu wir mit Hilfe der *iptables manpage* [ipt] und/oder des *Paket Filter Tutorial* [Rus] der genaue Weg eines Paketes beim passieren der Firewall ermittelt. Desweiteren werden die Konzepte der *chain* (Kette von Regeln) und einer *table* (Gruppierung von Ketten) erarbeitet.

4.2 Statische Filterregeln

In diesem Teil des Praktikums ist es Aufgabe der Studenten, zunächst leichte, dann immer kompliziertere statische Filterregeln zu erstellen. Hierzu wird ein rudimentäres Skript (siehe Anhang Seite 17) zu Verfügung gestellt, welches den Studenten erlaubt, die Firewall mit einen Aufruf zu aktivieren bzw. zu deaktivieren.

Die Aufgabenstellung reicht vom einfachen abschotten der Firewall, über das Protokollieren und Verwerfen von gespooften Paketen, bis hin zur eingeschränkten Nutzung des ICMP-Protokolls. Ideen für diese Aufgaben stammen zu einem großen Teil von den Aufgaben des früheren Firewall Versuchs. Neben der nötigen Anpassung der Lösung an *iptables*, wurde hierbei insbesondere darauf geachtet, den Sinn der Aufgabe in der Fragestellung klar zu verdeutlichen, bzw. zum weiteren Nachdenken über dieses Teilgebiet aufzufordern.

4.3 Dynamsiche Filterregeln

Dieser Aufgabenbereich vertieft die Kenntnisse in *iptables* und macht intensiven Gebrauch von dessen *connection tracking* Fähigkeit. Hierbei wird nur vorgegeben, welche Instanzen eine neue Verbindung aufbauen dürfen und wie diese beschaffen sein sollten (Verwendetes Protokoll, korrekter Verbindungsaufbau). *Iptables* erstellt dann dynamisch entsprechende Regeln um eine korrekte Verbindung zu ermöglichen.

Die Aufgaben erstrecken sich hierbei über das Ermöglichen der Konfiguration der Firewall mittels SSH und der Einschränkung der Nutzung auf wichtige Dienste wie WWW und E-Mail. Desweiteren wird den Studenten angeboten, mit eigenen Ideen zu experimentieren, um die Firewall evtl. noch mehr abzusichern.

4.4 Die grafische Benutzeroberfläche fwbuilder

Dieser letzte Teil des Versuchs unterscheidet sich ziemlich stark von den vorherigen. Galt es in den ersten drei Teilen vor allem, auf textueller Basis zu arbeiten (Ermittlung und Konfiguration von angebotenen Diensten, Erstellen der Regeln in einem Skript), wird hier eine leistungsfähige GUI bereitgestellt, welche in der Lage ist, die erstellten Regeln als *iptables* Skript abzuspeichern. Die Entscheidung, auch eine GUI zu verwenden, beruht darauf, dass kommerzielle Produkte eigentlich immer mit solch einer Oberfläche ausgestattet sind. Es ist daher sinnvoll, die Studenten mit der Bedienung einer solchen vertraut zu machen.

Neben dem Kennenlernen der GUI war es Aufgabe, alle Regeln aus Teil zwei und drei noch einmal zu erstellen. Hierbei sollten die Fähigkeiten und somit die Vorteile einer GUI erarbeitet werden. Dazu gehört z.B. die Fähigkeit, die meisten Resourcen im Netz automatisch zu erkennen oder aber auch mittels eines Assistenten relativ schnell einen umfangreichen Regelsatz zu erstellen.

Insbesondere sollten aber auch die Nachteile einer solchen „aufgesetzen" GUI dargestellt werden. Hierzu gehört zum einen die Tatsache, dass nicht alle Regeln, welche in fwbuilder erstellt wurden, korrekt in iptables Regeln übersetzt werden können. Zum anderen aber auch das falsche Gefühl von vermeintlicher Sicherheit, wenn die Regeln mittels eines Assistenten erstellt und im Nachhinein nicht verifiziert wurden.

Kapitel 5

Zusammenfassung

Im Zuge dieses Fortgeschrittenen Praktikums wurde der Aufgabenblock des Rechnernetze Praktikums, welcher sich mit dem Aufbau einer Firewall beschäftigt, überarbeitet. Hierbei wurde zunächst eine geeignete Platform ermittelt. Anschließend wurde ein Teil der schon verwendeten Aufgaben an das aktuelle System angepasst und neue, systemspezifische Aufgaben, so erstellt, dass die Studenten in einem Versuchsnachmittag an das Thema Firewall herangeführt werden.

Als Platform wurde hierzu ein Linux System gewählt, welches auf einem Kernel der Version 2.4 basiert. Für die Konfiguration des Paketfilters wurde das Programm *iptables* verwendet, als grafische Interface die Software *fwbuilder*.

Nach Fertigstellung der Aufgaben wurden diese von den Studenten im Praktikum ein Semester lang bearbeitet. Auf Grund deren Rückmeldungen und den Kritiken der Praktikumsbetreuer wurden die Aufgaben noch einmal überarbeitet. Hierzu gehörte die Präzisierung der Aufgaben, indem Mehrdeutigkeiten in der Fragestellung beseitigt wurden. Desweiteren wurden die umfangreicheren Aufgaben um Hinweise erweitert, welche die Studenten bei der Bearbeitung in die „richtige Richtung“ leiten sollen.

Anhang A

Verwendete Skripten

A.1 /etc/init.d/grundkonf

```
#!/bin/sh
# Stellt die Grundkonfiguration für den RNP Firewall Versuch wieder her

echo "Grundkonfiguration herstellen"
/usr/sbin/iptables -F
/usr/sbin/iptables -X
/usr/sbin/iptables -F INPUT
/usr/sbin/iptables -F OUTPUT
/usr/sbin/iptables -F FORWARD
/usr/sbin/iptables -P INPUT ACCEPT
/usr/sbin/iptables -P OUTPUT ACCEPT
/usr/sbin/iptables -P FORWARD ACCEPT
rm -r /tmp/firewall
mkdir /tmp/firewall
cp /restore_dir/firewall /tmp/firewall/firewall
insmod /lib/modules/2.4.4-4GB/kernel/net/ipv4/netfilter/ip_conntrack.o
insmod /lib/modules/2.4.4-4GB/kernel/net/ipv4/netfilter/ipt_state.o
insmod /lib/modules/2.4.4-4GB/kernel/net/ipv4/netfilter/ip_queue.o
insmod /lib/modules/2.4.4-4GB/kernel/net/ipv4/netfilter/ipt_LOG.o
insmod /lib/modules/2.4.4-4GB/kernel/net/ipv4/netfilter/ipt_TCPMSS.o
insmod /lib/modules/2.4.4-4GB/kernel/net/ipv4/netfilter/ipt_tcpmss.o
```

A.2 /restoredir/firewall

```
#!/bin/sh

# Einige Grundlegende Definitionen um die Skripte
# übersichtlicher zu machen

IPTABLES="/usr/sbin/iptables"
LO="lo"
DEVEXTERN="eth0"
DEVINTERN="eth1"

#Einschränkung auf pcfwsub1
NETZINTERN="192.168.215.80/28"


case "$1" in
  start)
   echo "Firewall wird aktiviert"

    exit 1
    ;;
  stop)
# Rechner arbeitet als normaler Router
   echo "Firewall wird deaktiviert"

    exit 1
    ;;
  close)
# Rechner routet nicht und blockt jeden Datenverkehr
   echo "Firewall abschotten"
    exit 1
    ;;
  *)
    echo "Usage: $0 start|stop|close"
    exit 1
esac

exit 0
```

Anhang B

Aufgaben

Im folgenden werden kurz die Aufgaben und deren Lösungen presentient, wie sie zur Zeit im Praktikum zum Einsatz kommen. Dabei sollen die Versuche nur eine Einführung in das Thema Firewall bieten, da die Ausarbeitung und Implementierung eines vollständigen Firewall Konzepts den Rahmen des Praktikums sprengen würde. Näheres zur Erstellung und der Struktur der Aufgaben findet sich im Kapitel 4 auf Seite 12.

B.1 Die Packet Filter Firewall

1. Wie schon erwähnt kann im Praktikum leider kein Minimalsystem realisiert werden. Als erstes sollte man daher feststellen, welche Dienste laufen. Hierzu bieten sich z.B. die Programmaufrufe `lsof -i, netstat -tu` und `netstat -lp` an. Welche Informationen liefern diese Aufrufe? Interpretieren Sie kurz die angezeigten Werte.

   Lösungshilfe für den Betreuer

   Der Aufruf `lsof -i` listet alle files auf, die im Zusammenhang mit dem Netzwerk und der Internet Familie stehen. Dies sind also alle Programme, die momentan auf irgendeinem Port lauschen und ihren Dienst anbieten wollen. In unserem Fall sind dies der X-server, der SSH Dämon und portmap.

   `netstat -tu` zeigt alle aktiven Kommunikationsstränge / Verbindungen die über diesen Rechner laufen. Da eigentlich keine Verbindungen existieren sollten, werden hier auch keine angezeigt.

   `netstat -lp` zeigt die gerade laufenden server an. Neben den schon mit `lsof` ermittelten Diensten, wird hier noch der ncsd (name service cache daemon) server angezeigt.

   Auf einem eigenständigen Paketfilter sollten hier natürlich nur die Dienste laufen, welche wirklich gebraucht werden, da jeder Dienst eine potentielle Angriffsfläche darstellt.

2. Der Kernel (bzw. das netfilter Modul) ist für die Filterung der einzelnen Datenpakete verantwortlich. Konfiguriert wird dieser Filter mit Hilfe des `iptables` Programmes. Verdeutlichen Sie sich noch einmal kurz das Konzept von `netfilter/iptables` z.B. mittels der manpage oder dem ausliegenden Paket Filter Tutorial.

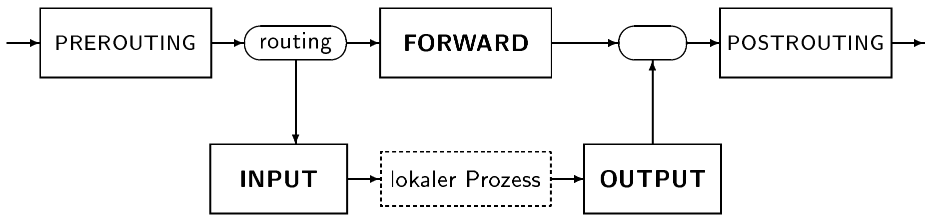
   Was versteht man unter einer "chain"? Erklären Sie kurz die Aufgaben der 5 vorgegebenen chains und stellen sie den Weg eines Datenpaketes durch den Kernel (die einzelnen chains) grafisch da. Was ist die Aufgabe einer "table" und welche stellt das System zu Verfügung? Welche Zuordnung kann zwischen den vorgegebenen chains und den 3 standard tables gemacht werden? Welche table ist für unseren einfachen Paketfilter interessant?

Lösungshilfe für den Betreuer

Eine chain (Kette) ist eine Liste von (Filter) Regeln, welche der Reihe nach abgearbeitet werden. Folgende 5 Ketten sind vom System definiert:

INPUT	Hier landen alle Pakete, die an einen lokalen Prozess (also an den Paket Filter selber) gerichtet sind.
OUTPUT	Hier laufen alle Pakete durch, die von einem lokalen Prozess stammen.
FORWARD	Alle zu routenden Pakete passieren diese chain.
PREROUTING	Unmittelbar bevor eine Routing Entscheidung getroffen wird, müssen alle Pakete (welche in irgendeiner Art beim Firewall Rechner ankommen) diese Stelle passieren. Hier erfolgt z.B. bei NAT die Adress-Übersetzung in den Headern.
POSTROUTING	Entsprechend PREROUTING nur für ausgehende Datenpakete.

Der Weg eines Datenpaketes durch den Kernel kann folgendermaßen dargestellt werden:



In der Netfilter Architektur existieren 3 tables (filter, nat, mangle). Diese sind von gruppierendem Charakter (sie fassen verschieden chains zusammen) und haben den Zweck, die verschiedenen Arten der Paketbehandlung auf Module zu verteilen. Damit ist sichergestellt, dass nur die chains geladen werden, die auch wirklich gebraucht werden.

filter	Standard Tabelle, welche auch für unseren Paket Filter benötigt wird. Sie besteht aus den INPUT, FORWARD und OUTPUT chains, welche oben etwas hervorgehoben wurden.
nat	Ist für alle Arten von Adress-Umwandlung (NAT) oder Port-Forwarding zuständig. Sie beinhaltet die PREROUTING, OUTPUT und POSTROUTING chain und wird für den Versuch nicht benötigt.
mangel	Hiermit werden spezielle Änderungen an den Paketen vorgenommen (z.B. QoS Parameter). Sie beinhaltet PREROUTING und OUTPUT und wird im Versuch auch nicht verwendet.

3. Sobald das netfilter Modul geladen ist, bietet der Kernel einige Optionen, welche den Betrieb als Firewall erleichtern bzw. die Sicherheit des Systemes erhöhen können. Diese Optionen sind in der Datei
`/usr/src/linux/Documentation/networking/ip-sysctl.txt` beschrieben und können unter `/proc/sys/net/ipv4/` bzw. unter `/proc/sys/net/ipv4/conf/*` aktiviert werden[1].
   Wählen Sie einige Parameter aus, welche für eine Firewall evtl. interessant sein könnten und geben sie eine knappe Beschreibung.
   Welche Parameter am Ende angebracht sind, ist natürlich vom jeweiligen Sicherheitskonzept abhängig. Teilweise kann der gleiche Effekt auch mittels iptables Re-

[1] Dies geschieht mittels `echo WERT > PARAMETER`, also z.B. `echo 1 > ip_forward` um das forwarding einzuschalten

geln erreicht werden (siehe unten).

Lösungshilfe für den Betreuer

Bei den Kernelparametern wären z.B. folgende Parameter nützlich:

```
ip_forward                    # schaltet das forwarding ein und aus
tcp_syncookies                # erkennt und meldet syn flood attacks
icmp_echo_ignore_broadcasts   # verhindert ausspionieren des Netzes
icmp_*_rate                   # erschwert DoS Attacken da nur auf eine
                              # bestimmte Anzahl von Paketen reagiert wird
conf/*/log_martians           # unmögliche Pakete aufzeichnen
conf/*/rp_filter              # aktivieren (anti spoofing)

conf/*/accept_redirects       # Diese 2 Werte sollte man deaktivieren
conf/*/accept_source_route    # da diese Verfahren für Angriffe
                              # verwendet werden können, bzw. damit die
                              # Firewall umgangen werden kann da eigene
                              # Routen gesetzt werden
```

B.2 Einfache Firewall Skripts

1. Für alle folgenden Aufgaben sollten Sie das Skript `/tmp/firewall/firewall` verwenden, da hier schon einige rudimentäre Definitionen gemacht wurden.

```
#!/bin/sh

# Einige Grundlegende Variablen auf die mittels $VARIABLE
# zugegriffen werden kann.
IPTABLES="/usr/sbin/iptables"
LO="lo"
DEVEXTERN="eth0"
DEVINTERN="eth1"
#Einschränkung auf Subnetz von pcfwsub1
NETZINTERN="192.168.215.80\28"

case "$1" in
  start)
    echo "Firewall wird aktiviert"

    exit 1
    ;;
  stop)
    echo "Firewall wird deaktiviert"
    # Rechner arbeitet als normaler Router

    exit1
    ;;
  close)
    echo "Firewall abschotten"
    # Rechner routet nicht und blockt jeden Datenverkehr

    exit1
    ;;
  *)
```

```
    echo Üsage: $0 start|stop|close"
    exit 1
esac

exit 0
```

Wir wollen von einer pessimistischen Vorgehensweise bei der Konfiguration unseres Paketfilters ausgehen. Das Skript sollte so erweitert werden, dass beim aktivieren der Firewall (`start)` vorerst jeglicher Datenverkehr verhindert wird und somit das interne Netz abgeschottet ist. Verwenden sie hierzu das Konzept der "policies".

Es bietet sich natürlich an, diese Regeln auch gleich in die hierfür vorgesehene Abteilung `close)` zu schreiben, welche genau für diesen Zweck (der Abschottung) angelegt wurde. Der Abschnitt unter `stop)` sollte so erweitert werden, dass alle Filterregeln deaktiviert werden. Die Firewall sollte dann also wieder normal routen und alle Datenpakete akzeptieren.

Die Funktionstüchtigkeit der Regeln sollte jetzt, wie auch nach allen anderen Versuchen, überprüft werden.

Hinweis:

Um Inkonsistenzen durch evtl. sich noch im System befindliche Regeln zu vermeiden, sollte darauf geachtet werden, die chains am Anfang immer zu leeren.

Lösungshilfe für den Betreuer

Im Gegensatz zum vorherigen Versuchstag, können die folgenden Aufgaben auf verschiedene Weise gelöst werden. Die "Lösungshilfe" bietet hier also immer nur einen Vorschlag wie die Aufgabe bearbeitet werden kann. Wichtig ist, dass sich die Studenten mit iptables auseinander setzen und dessen Möglichkeiten kennen lernen.

```
# für start) und close)
# routing ausschalten
echo 0 > /proc/sys/net/ipv4/ip_forward
# löscht alle Regeln der filter table
$IPTABLES -F
# löscht alle benutzerdefinierten Ketten (falls vorhanden)
$IPTABLES -X
# Mittels policies alle Pakete verwerfen
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

# Firewall deaktivieren
stop)
$IPTABLES -F
$IPTABLES -X
# Alle Pakete akzeptieren
$IPTABLES -P INPUT ACCEPT
$IPTABLES -P OUTPUT ACCEPT
$IPTABLES -P FORWARD ACCEPT
echo 1 > /proc/sys/net/ipv4/ip_forward
```

2. Um auf dem Rechner lokal installierte Dienste (z.B. DNS, NIS oder HTTP) unabhängig von der Konfiguration der Netzwerkkarten zu benutzen, steht normalerweise ein loopback Interface zu Verfügung (Auf einem reinen Paketfilter hätten diese natürlich nichts verloren). Dieses kann aber auch für Testzwecke (z.B. des Protokollstacks oder der obigen Dienste) nötig sein. Das firewall Skript soll nun so erweitert werden, dass bei aktivierter Firewall ein uneingeschränkter Zugriff auf

das loopback Interface möglich ist.

Lösungshilfe für den Betreuer

```
# Erlaubt vollständigen Zugriff auf das Loopback Interface
$IPTABLES -A OUTPUT -o $LO -j ACCEPT
$IPTABLES -A INPUT -i $LO -j ACCEPT
```

3. Bevor in den folgenden Abschnitten der Paketfilter immer weiter geöffnet wird um bestimmte Dienste anzubieten, sollte noch eine Einschränkung vorgenommen werden. Da das zu schützende Netz einen festen IP Bereich besitzt, sollten alle Datenpakete an die interne Schnittstelle auch aus diesem stammen. Adressen die diese Eigenschafften nicht besitzen ("gespoofte" Adressen) kommen nicht von einem regulären Rechner und stammen mit ziemlicher Sicherheit von einem Angriffsversuch (oder falsch konfigurierten Rechnern die es eigentlich nicht geben sollte).

   Schreiben Sie Regeln, welche diese offensichtlich gefälschten Adressen (spoofing) protokollieren und anschließend verwerfen. Sorgen Sie dafür, dass die entsprechenden Meldungen unter `/var/log/messages` eindeutig auffindbar sind und überprüfen Sie dies, indem Sie sich diese Datei anzeigen lassen und nach entsprechenden Meldungen suchen.

   Hierbei kann die Möglichkeit genutzt werden, eigene (sub)chains zu erstellen. Diese können das Skript übersichtlicher machen, erleichtern die Wartung und sparen Schreibarbeit.

   Zur Überprüfung der Konfiguration steht auf dem Rechner `pcfwsub1` das tool **spoof** zu Verfügung. Beim Aufrufen dieses Programmes wird eine Befehlssyntax ausgegeben, wobei darauf zu achten ist, dass nur der Zielrechner 192.168.215.3 erlaubt ist.

   Hinweis

   Würde unser Paketfilter am Internet hängen, könnten auch an dessen externen Interface nach diesem Schema gespoofte Adressen auftauchen. Es wäre dann also notwendig Regeln aufzustellen, die alle Pakete verwerfen, welche aus einem reservierten Adressbereich stammen und am externen Interface ankommen.

   Lösungshilfe für den Betreuer

   Hier ist darauf zu achten, sowohl die INPUT auch als die FORWARD chain zu berücksichtigen und die Pakete nach dem protokollieren zu löschen.

```
# neue Benutzerdefinierte chain anlegen
$IPTABLES -N logspoofdrop
# übergebene Pakete protokollieren und verwerfen.
$IPTABLES -A logspoofdrop -j LOG --log-prefix "$IPTABLES: spoofing Versuch"
$IPTABLES -A logspoofdrop -j DROP

# fremde/falsche Adressen an interne Schnittstelle verarbeiten
$IPTABLES -A INPUT -i $DEVINTERN ! -s $NETZINTERN -j logspoofdrop
$IPTABLES -A FORWARD -i $DEVINTERN ! -s $NETZINTERN -j logspoofdrop
```

4. Für administrative Zwecke ist das ICMP Protokoll eine große Hilfe. Schreiben Sie einen Regelsatz, welcher die Verwendung der häufigsten ICMP Dienste (ping, traceroute) in jegliche Richtung erlaubt (auch wenn dies sicherheitstechnisch (z.B. ping an Broadcastadressen zum ausspionieren des Netzes) bedenklich ist! Wenn genügend Zeit vorhanden ist, steht es ihnen natürlich frei auch einen sehr fein granularen Filter zu entwerfen). Die in Aufgabe 3 untersuchten Kernelparameter könnten hier sehr nützlich sein und sollten bei Bedarf auch verwendet werden. Um die Regeln testen zu können, sollte nun auch das ip forwarding eingeschaltet werden.

Hinweise:

Hierbei ist zu beachten, dass mit diesen ICMP Regeln alleine, der traceroute Dienst noch nicht funktioniert (siehe auch die manpage). Dies liegt daran, dass der Dienst auf UDP Testpaketen (probes) basiert, welche natürlich (noch) von der Firewall blockiert werden.

Teilweise können sich hier die Funktionen der Kernelparamter mit denen der Regeln überschneiden. Dies ist bewusst so gewählt. Zum einen könnten Situationen auftreten in denen die Parameter nicht zu Verfügung stehen (z.B. stand alone Lösungen die auf iptables basieren), zum anderen ist es schwierig die Korrektheit der Funktionsweise der Kernelparameter zu überprüfen.

Lösungshilfe für den Betreuer

```
# einige Einstellungen im Kernel (siehe Aufgabe 3)
echo 1 > /proc/sys/net/ipv4/conf/all/rp_filter
echo 1 > /proc/sys/net/ipv4/conf/all/log_martians
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
echo 1 > /proc/sys/net/ipv4/tcp_syncookies

# ALLOW ICMP
# Eigene chain erstellen welche ICMP erlaubt
$IPTABLES -N allowICMP

# verhindert unnötige Wartezeiten da wir ja eh pings zulassen
$IPTABLES -A allowICMP -p icmp --icmp-type destination-unreachable -j ACCEPT
# ermöglicht den Dienst traceroute
$IPTABLES -A allowICMP -p icmp --icmp-type time-exceeded -j ACCEPT
# ermöglicht ping und pong
$IPTABLES -A allowICMP -p icmp --icmp-type echo-request -j ACCEPT
$IPTABLES -A allowICMP -p icmp --icmp-type echo-reply -j ACCEPT
# Alle anderen ICMP Dienste zurückweisen
# (Drop wird nicht so gerne gesehen, mehr in der firewall FAQ
$IPTABLES -A allowICMP -p icmp -j REJECT

# aktivieren unserer ICMP chain
$IPTABLES -A INPUT -p -icmp -j allowICMP
$IPTABLES -A OUTPUT -p -icmp -j allowICMP
$IPTABLES -A FORWARD -p -icmp -j allowICMP

# IP-Forwarding aktivieren
# Dies sollte erst hier (am Ende des Skriptes) passieren,
# damit keine Pakete durch die evtl.
# noch lückenhafte Firewall gelangen können.
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Alternativ können auch die `icmp-type` 0, 3, 5 und 11 zugelassen werden.

B.3 Erweiterte Filterregeln

Netfilter/iptables bietet mit dem Modul **ip_state**[2] die Möglichkeit des "connection tracking" (Das Prinzip ist auch unter *stateful inspection* bekannt). Hiermit werden die Zustände aller Kommunikationsverbindungen, welche durch die Firewall aktiv bestehen, mitverfolgt und es kann entsprechend darauf reagiert werden. Dies geschieht in der Datei `/proc/net/ip_conntrack` und funktioniert auch mit zustandslosen Verbindungen wie bei UDP oder ICMP. Verifi-

[2] In den Regeln mittels `-m state`

zieren Sie dies, indem Sie sich diese Datei während eines laufenden Pings anzeigen lassen. Falls keine Verbindung angezeigt wurde, führen Sie einen Ping an eine nicht existierende Adresse durch, um eine noch nicht beendete Verbindung zu erhalten. Wiederholen Sie dies später, wenn weitere Verbindungen möglich sind.

1. Wie schon erwähnt, sollten auf einem reinen Paketfilter eigentlich keine Dienste laufen. Evtl. ist es aber manchmal nötig, die Firewall aus der Ferne zu administrieren. Richten Sie zu diesem Zweck eine Regel ein, die es erlaubt, aus dem internen Netz mittels SSH auf den Paketfilter zuzugreifen.

   Machen Sie dabei so viele Einschränkungen wie möglich und achten sie darauf, dass weder vom externen Netz, noch vom Firewall Rechner selber, *neue* SSH Verbindungen in das interne Netz aufgebaut werden dürfen.

   Weiterhin ist der korrekte Aufbau einer neuen TCP Verbindung zu beachten. Sorgen Sie dafür, dass nur solche Pakete eine neue TCP Verbindung aufbauen dürfen und alle "nicht korrekten" TCP Verbindungsversuche verworfen werden.

   Hinweis:
   Für einen korrekte Nutzung des SSH Dienstes ist natürlich auch eine Regel für den entsprechenden Rückkanal erforderlich. Es sollte darauf geachtet werden, dass nur solche Rückverbindungen akzeptiert werden, die zu einer korrespondieren SSH Verbindung gehören.

   Lösungshilfe für den Betreuer

```
# verwirft alle neuen TCP Verbindungen (--state NEW), bei denen das SYN
Bit nicht
# gesetzt ist und das ACK und FIN Bit nicht ungesetzt sind. (! --syn)
$IPTABLES -A INPUT -i $DEVINTERN -o $DEVEXTERN -s $NETZINTERN
    -p tcp ! --syn -m state --state NEW -j DROP

# erlaubt das aufbauen einer SSH Verbindung zur Firewall
iptables -A INPUT -i $DEVINTERN -s $NETZINTERN -p TCP --dport 22
    -m state --state NEW,ESTABLISHED -j ACCEPT
# erlaubt den Rückkanal von der Firewall aus
iptables -A OUTPUT -o $DEVINTERN -d $NETZINTERN -p TCP --sport 22
    -m state --state ESTABLISHED,RELATED -j ACCEPT
```

   Regel eins sorgt für einen korrekten Verbindungsaufbau. Die zweite Regel erlaubt (mit den gegebenen Einschränkungen) eine neue Verbindung zum Firewall Rechner. Die dritte Regel öffnet dann einen entsprechenden Rückkanal, falls eine passende Anfrage erfolgt ist.

2. Nun wird es Zeit, dass wir den Zugriff aus dem inneren Netz auf das äußere erlauben. Schreiben sie eine einfache Regel, die alle TCP/UDP Verbindungen aus dem internen Netz und die zugehörigen Rückverbindungen ins Netz zulässt.

   Hinewise:
   In der vorliegenden iptables Version können die zu verwendenden Protokolle leider nicht (wie in der manpage angegeben) als durch Komma getrennte Liste angegeben werden. Es ist daher leider erforderlich für TCP und UDP eigene Regeln zu schreiben.

   Die Funktionsweise seiner Regeln prüft man am besten mittels *telnet*, falls auf den internen Rechnern ein entsprechender Dienst nicht vorhanden ist.

   Lösungshilfe für den Betreuer

```
# verwirft ungültige Verbindungsversuche $IPTABLES -A FORWARD -i $DEVINTERN
-o $DEVEXTERN -s $NETZINTERN
    -p tcp ! --syn -m state --state NEW -j DROP
```

```
# erlaubt alle sonstigen Verbindungen nach außen
$IPTABLES -A FORWARD -i $DEVINTERN -o $DEVEXTERN -s $NETZINTERN
    -p tcp -m state --state NEW,ESTABLISHED -j ACCEPT
$IPTABLES -A FORWARD -i $DEVINTERN -o $DEVEXTERN -s $NETZINTERN
    -p udp -m state --state NEW,ESTABLISHED -j ACCEPT
# erlaubt die zurückkommenden Verbindungen
$IPTABLES -A FORWARD -i $DEVEXTERN -o $DEVINTERN -p tcp
    -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -i $DEVEXTERN -o $DEVINTERN -p udp
    -m state --state ESTABLISHED,RELATED -j ACCEPT
```

3. Die vorhergegangene Regel geht natürlich davon aus, dass sich die Benutzer einigermaßen vernünftig verhalten und keine unerlaubte Software ausführen. Hier ist zum einen bösartiger Code zu nennen. Dieser kann sowohl von internen Rechnern Angriffe auf andere Netze ausführen, als auch Verbindungen zu fremden Rechnern öffnen und interne Daten verschicken. Zum anderen sind hier aber auch Programme zu nennen (z.B. file sharing), deren Verwendung von einer Administrativen Stelle untersagt wurde.

   Aus diesem Grund soll nun der Zugang zum äußeren Netz auf WWW (http, https) und EMail (pop3, smtp) Dienste eingeschränkt werden. Falls gewünscht können natürlich auch noch weitere Dienste wie in `/etc/services` beschrieben verwendet werden. Vergessen Sie nicht die Namensauflösung zu erlauben, da ohne sie ein normales arbeiten kaum möglich ist.

   Lösungshilfe für den Betreuer

```
# DNS
$IPTABLES -A FORWARD -i $DEVINTERN -s $NETZINTERN -p udp
  --sport 1024:65535 --dport 53 -m state --state NEW, ESTABLISHED -j ACCEPT
# SMTP
$IPTABLES -A FORWARD -i $DEVINTERN -s $NETZINTERN -p tcp
  --sport 1024:65535 --dport smtp -m state --state NEW, ESTABLISHED -j ACCEPT
# POP3
$IPTABLES -A FORWARD -i $DEVINTERN -s $NETZINTERN -p tcp
  --sport 1024:65535 --dport pop3 -m state --state NEW, ESTABLISHED -j ACCEPT
# HTTP
$IPTABLES -A FORWARD -i $DEVINTERN -s $NETZINTERN -p tcp
  --sport 1024:65535 --dport http -m state --state NEW, ESTABLISHED -j ACCEPT
# HTTPS
$IPTABLES -A FORWARD -i $DEVINTERN -s $NETZINTERN -p tcp
  --sport 1024:65535 --dport https -m state --state NEW, ESTABLISHED -j
ACCEPT
```

B.4 FWBUILDER

FIREWALL BUILDER ist ein objetktorientiertes grafisches Frontend für die Erstellung von Firewall Skripten. Unter anderem wird auch iptables unterstützt. Objektorientiert heist in diesem Fall, dass alle Elemente, welche später in den Filterregeln Verwendung finden, durch Objekte repräsentiert werden und zunächst definiert werden müssen.

Die Vor und Nachteile einer GUI werden weiter unten innerhalb einer Aufgabe geklärt. Weiterhin ist es von Vorteil einmal mit einer GUI zu arbeiten, da die meisten kommerziellen Firewall Produkte (z.B. firewall-1) mit einer ähnlichen GUI ausgeliefert werden. Teilweise sind sie sogar nur über eine solche zu konfigurieren.

Hinweis:

Leider ist der fwbuilder noch ein wenig instabil. So führt das Verwenden von *cut* und *paste* fast immer zu einem Absturz. Für das Löschen von Objekten sollte man daher das Kommando *delete* im linken Objektbaum verwenden. *Copy & past* sollte ohne Probleme funktionieren.

1. Starten Sie den fwbuilder und machen Sie sich zunächst mit der Bedienung der GUI vertraut (Die GUI sollte recht intuitiv zu benutzen sein. Es steht aber auch ein Tutorial in gedruckter Form zu Verfügung, welches gleichzeitig als Anleitung dient). Das wichtigste Objekt ist natürlich die Firewall selber. Dieses sollte als erstes erzeugt und so weit wie möglich konfiguriert werden (Die Reiter *Sysinfo* und *compile/install* können so gelassen werden wie sie sind). Aus welchem Grund bietet das Firewall Objekt zwei verschiedene Stellen, an denen Regeln erstellt werden können? Welche Auswirkungen hat die Einstellung `Assume firewall object is part of any` in den FIREWALL Optionen des Firewall Objekts?

   Hinweis:

   Bei der Konfiguration ist darauf zu achten, dass das erzeugte Firewall Skript den gleichen Namen trägt wie der unter Optionen angegebene. Achten Sie also darauf, ihr altes Skript nicht aus versehen zu löschen.

   Lösungshilfe für den Betreuer

   Das Firewall Objekt muss mit INSERT/FIREWALL erzeugt werden. Das HOST Objekt ist hier nicht sinnvoll, da dann alle Einstellungsmöglichkeiten für die Firewall fehlen. Die Konfiguration sollte eigentlich Eindeutig sein, da die Einstellungen entweder selbst erklärend sind oder mit den Definitionen übereinstimmen, welche in den Aufgaben weiter oben per Hand vorgenommen/untersucht worden sind.

   Das Firewall Objekt bietet die Möglichkeit *global policy rules* und *interface policy rules* zu erstellen. Der Unterschied besteht darin, dass die global policy rules auf alle Pakete angewandt werden, welche die Firewall passieren, wobei nicht darauf geachtet wird, auf welchem Interface die Pakete ankommen bzw. abgehen. Dies entspricht also der FORWARD chain, ohne Angabe der Parameter `-i` und `-o`. Bei den *interface policy rules* hingegen kann noch das verwendete Interface angegeben werden, was z.B. für die anti-spoofing Regeln notwendig ist.

   Von der Einstellung `firewall object is part of any` hängt ab, ob gewisse Regeln in der FORWARD und INPUT (bzw. OUTPUT) chain abgelegt werden (Firewall gehört zu Subnetz (any)) oder nur in einer von beiden.

2. Bevor nun überhaupt irgendwelche Regeln verwaltet werden können, müssen erst einmal alle benötigten Objekte erzeugt werden. Dies kann entweder von Hand gemacht werden, oder man benutzt das Tool `Discover Objects`. Probieren sie hier alle Möglichkeiten der Erkennung aus. Für die Erkennung mittels SNMP verwenden sie den community string *public* oder *rnp*. Um Gruppen zu füllen, werden Objekte aus dem linken Objektbaum einfach in das offene Gruppen Fenster gezogen.

   Erstellen Sie nun ein paar einfache Regeln und verwenden Sie auch den eingebauten *policy building druid* unter `RULES / HELP ME BUILD FIREWALL POLICY`. Wie ist dieser zu bewerten? Es sollte mindestens ein `time` Object erstellt und in einer Regel verwendet werden. Dieses dient dazu, Filterregeln nur für einen bestimmten Zeitraum zu aktivieren. Was passiert mit diesen Zeiteinstellungen?

   Bevor mittels `RULES / COMPILE` von fwbuilder ein iptables Skript erzeugt werden kann, ist es notwendig das Firewall Objekt zu sichern (z.B. in `/tmp/firewall/`). Das erzeugte Skript sollte nun genauer untersucht und die Unterschiede zwischen den Regeln im fwbuilder und dem dazu erzeugten Skript diskutiert werden. Was fällt ihnen auf?

Lösungshilfe für den Betreuer

Die Bewertung des druid ist natürlich rein subjektiv. Zum einen kann er natürlich helfen, in kurzer Zeit eine einfache Basis von Regelsätzen aufzustellen. Da diese aber mit ziemlicher Sicherheit überarbeitet werden müssen, ist der Zeitvorteil evtl. nur marginal. Die Möglichkeit, dass mit diesem wizard auch unerfahrene Benutzer eine evtl. undichte Firewall erstellen können und sich mit dieser dann in Sicherheit wiegen, ist natürlich bedenklich.

Die Beantwortung der letzten Frage ist natürlich abhängig von den benutzten Regeln. Sie dient eigentlich mehr dazu die Studenten darauf aufmerksam zu machen, dass eine 1:1 Übersetzung und damit ein vollständiger Funktionstransfer der Filterregeln (von der GUI zu iptables) nicht immer möglich ist.

Dies sieht man sehr deutlich an dem erzeugten `time` Objekt. In fwbuilder wird einem hiermit die Möglichkeit gegeben, Regeln nur für einen bestimmten Zeitraum zu aktivieren. Iptables als zu darunter liegende Architektur bietet hierfür aber keine Möglichkeit, außer vielleicht über aufwendige shell Programmierung oder mittels cron-jobs.

Eventuell wäre auch noch zu beanstanden, dass das erzeugte Skript wahrscheinlich nicht so übersichtlich ist wie das selbst geschriebene. Dies liegt zum einen an den nicht sehr aussagekräftigen Namen der benutzerdefinierten Regeln, als auch an den fehlenden Kommentaren.

3. Erstellen Sie mit dem fwbuilder (soweit möglich) nun noch einmal alle Regeln, die Sie oben per Hand erstellt haben. Benutzen Sie auch hier alle Möglichkeiten, welche ihnen fwbuilder bietet um einen übersichtlichen Aufbau zu gewährleisten.

   Auch hier sollte wieder eine Gegenüberstellung zwischen compilierten Regeln und dem von ihnen per Hand erstelltem Skript erfolgen. Sind die zwei Skripten vom Funktionsumfang äquivalent? Diskutieren Sie die Vor- und Nachteile einer GUI.

   Um diesen Vergleich auch später noch nachvollziehen zu können ist es notwendig, auch die Regeln der GUI in die Ausarbeitung aufzunehmen. Entweder von Hand in Form von Tabellen wie in den Theorieaufgaben, oder durch Screenshots z.B. mittels `xv`.

Lösungshilfe für den Betreuer

Das nochmalige Erstellen aller Regeln sollte kein Problem darstellen. Mit ziemlicher Sicherheit werden aber die von der GUI erzeugten Regeln anders aufgebaut sein als die von Hand erstellten. Trotzdem sollte aber der gleiche Schutz gewährleistet werden, was natürlich von den Studenten zu verifizieren ist.

Genau wie bei der vorhergegangenen Aufgabe sind die Vor und Nachteile teilweise wieder rein subjektiv zu bewerten. Als Vorteile wären z.B.:

- Übersichtlichkeit (in der GUI)
- schnelle Erzeugung von Regeln
- einfachere Administration / Wartung
- geringerer Wissensstand notwendig, um Regeln zu erstellen (Regel-wiazards)

Nachteile:

- geringerer Wissensstand notwendig um eine vermeintlich sichere Firewall zu erzeugen
- kein 100% Funktionstransfer, abhängig von der zu Grunde liegenden Plattform
- unübersichtliches iptables Skript
- Skript muss irgendwie auf die Firewall gebracht werden (evtl. Sicherheitslücke)

Literaturverzeichnis

[And] Oskar Andreasson. *iptables Tutorial.*
Available from World Wide Web:
`http://iptables-tutorial.frozentux.net/index.html`.

[Don] Lutz Donnerhacke. *Firewall FAQ aus de.comp.security.firewall.*
Available from World Wide Web:
`http://www.iks-jena.de/mitarb/lutz/usenet/Firewall.html`.

[Frü] Jörg Frühbrodt. *Bessere Linux-Firewalls mit Netfilter und Iptables.*
Available from World Wide Web:
`http://joerg.fruehbrodt.bei.t-online.de/netfilter.html`.

[fwb] fwbuilder webpage.
Available from World Wide Web:
`http://www.fwbuilder.org/`.

[ipt] *Linux manpage zu iptables.*
Available from World Wide Web:
`http://www.netadmintools.com/html/8iptables.man.html`.

[Kin] Wolfgang Kinkeldei. *Firewall selbst entwickeln.*
Available from World Wide Web:
`http://www.pl-berichte.de/t_netzwerk/print/iptables.html`.

[Kur] Vadim Kurland. *Firewall Builder Tutorial.*
Available from World Wide Web:
`http://www.fwbuilder.org/pages/Documents/Tutorial/index.html`.

[rpm] rpmfind webpage.
Available from World Wide Web:
`http://www.rpmfind.net/`.

[Rus] Rusty Russel. *Linux 2.4 Packet Filtering HOWTO.*
Available from World Wide Web:
`http://www.netfilter.org/unreliable-guides/packet-filtering-HOWTO/`.

[Wel] Harald Welte. *netfilter/iptables FAQ.*
Available from World Wide Web:
`http://www.netfilter.org/documentation/FAQ/netfilter-faq.html`.