LinuX Enterprise

LinuX Enterprise

Deutschland € 5,80
Österreich € 6,90
Schweiz sFr 10,90

S&S Magazin für Professional Linux & Open Source Solutions 4.05

Auf CD

- Intrexx Xtreme 2.5
- VTune Performance Analyzer v3.0 for Linux
- Cluster Math Kernel Library v7.2 for Linux
- bitkit/MAIL 1.0
- Kernel & Software

Linux-Kernel 2.6.11.10
ActiveState Perl 5.8.6.811
GIMP 2.2.7
Python 2.4.1
Ruby 1.8.2

CPAN Perl-Module
Python-Tools
Ruby-Tools

CD-Inhalt auf Seite 3

Linux-Desktops

Profi-Schreibtische für den Business-Einsatz

CMS-Trends

Web Content Management heute

OpenOffice.org 2.0

Was gibt es Neues?

Virtual Private Networks

Technologien für Linux-Systeme

SuSE Linux Professional 9.3

Neuigkeiten

www.linuxenterprise.de

Apple Mac mini

Alternative für Linux-Anwender

VMware

Neue Funktionen in Version 5

Website Engine

Modulares Engine Design mit PHP

Simple Security Policy Editor (SSPE): Dezentrale Firewalls ankl. IPSec-VPN

Der praktische Ansatz

Als mein Arbeitgeber mich Ende 2001 beauftragte, die aus der Teilung hervorgehende neue Firma mit sechs deutschen Standorten ausschließlich mit Linux auf Standard-PCs abzusichern und ans Internet anzuschließen, war neben Kostenüberlegungen sicherlich auch meine ihm bekannte Erfahrung mit kommerziellen Firewalls und Open-Source-Lösungen mit ausschlaggebend.

von Johannes Hubertz

Spätestens seit meiner Lektüre von Fachzeitschriften [1], [2] war ich ab ca. 1996 davon überzeugt, mit Linux [3] professionell und punktgenau IT-Sicherheit herstellen und überprüfbar gestalten zu können. Die damalige Goldgräberstimmung mit E-Commerce-Boom und Webseiten-Euphorie ließ den noch kleinen Tux gegenüber herkömmlichen Unices noch nicht so gut dastehen, sie ermöglichte mir jedoch in bis dahin unbekannter Weise Quellcode-Einblicke in vollständig überzeugender Qualität. Schnell war zuerst nur eine Linux-Maschine ausschließlich dazu da, Dinge zu testen und nachzuvollziehen, zunächst im Netzwerkbereich mit ipfwadm und später im Umfeld von Routing, DNS, E-Mail und Webservern. Alle Erfahrung sammelte sich in Shell-Scripts und Kommentaren darin, die Einsatzmöglichkeiten waren jedoch aufgrund mangelnder Verbreitung (in der Firma) gering. Erst mein Wechsel in die interne Netzwerkadministration ließ meinem Tatendrang freieren Lauf. Eine erste Firewalladministration auf Basis der Erfahrungen mit der kommerziellen Internet-Firewall entstand, wurde jedoch nie veröffentlicht. Zwei Dateien waren zentraler Bestandteil: eine Definitionsdatei, in der alle Netz- und Hostadressen mit Netzmaske erfasst wurden, und eine Regeldatei, in der die Verkehrsbeziehungen der definierten Quellen und Ziele mit Protokoll, Port und eventuell mit weiteren Optionen in geordneter, sinnhafter Reihenfolge abgelegt wurden.

Die 2001 entstandene Aufgabe, das Firmennetzwerk in zwei Hälften zu teilen, überstieg schnell die Forderung, einfach nachvollziehbar und überschaubar zu bleiben. Also machte ich mir ein paar Gedanken und SSPE entstand im stillen Kämmerlein zunächst in klinisch reiner Laboratmosphäre.

Konzeption

Der erste und wichtigste Grundgedanke war, dass eine zentrale, kompakte Administration mir das Leben als Administrator deutlich erleichtern würde. Der Wunsch, automatisiert die Konsistenz und konsequente Anwendung der Filterregeln über alle beteiligten Maschinen zu sichern, war treibender Gedanke. Aufgrund der Anzahl Kunden, Mitarbeiter, Standorte und Projekte war auch weiterhin mit täglichen Änderungsanforderungen in den Firewalls zu rechnen. Der zweite, ebenso wichtige war, die Netzwerksicherheit an möglichst vielen Stellen zu unterstützen. Also sollte auf vielleicht jedem Serversystem des Unternehmens eine Firewall genau die Ports öffnen, die notwendigerweise die Applikation braucht, alle anderen jedoch vor den neugierigen Benutzeraugen und der bösen restlichen Welt abschotten. Eine Art Selbstregulierung auf jeder Maschine schied aus verschiedenen Gründen aus, zentral sollten die notwendigen iptables-Kommandos pro Zielsystem generiert, verteilt und endlich zentral gesteuert in den Kernels aktiviert werden. Jedoch waren nicht nur Linux-Maschinen im Spiel, auch Router können per Access-Listen die Ports ähnlich einer Firewall regulieren. Also wurde die Hardware bzw. das OS als Kriterium herangezogen, was zu generieren war. Um mit Kernels der 2.4-er-Serie zu filtern, ist es nötig zu wissen, ob ein IP-Paket für die Maschine selbst bestimmt ist oder nur durchgeleitet werden soll. Ergo waren zur Generierung auch die jeweils eigenen IP-Adressen zu betrachten. Da ein beispielhaft betrachtetes Paket von A nach B die Standorte C bis F gar nicht berührt, ist weiterhin die Routingtabelle der Maschine mit zu Rate zu ziehen, um zu entscheiden, wie eine Maschine ein beliebiges Paket handhaben soll. Der letzte Gedanke dient hier sowohl der Korrektheit als auch der Minimierung von Regeln, die in iptables umzusetzen sind. Ein aktueller Kernel aus der Vanilla-Serie

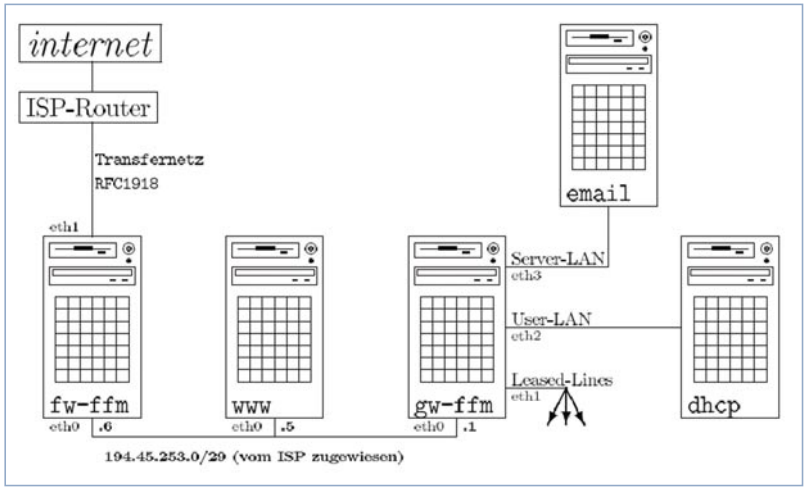

Abb. 1: Typische Architektur eines Firmenstandortes

erschien mir der einzig sinnvolle, um proprietäre Einflüsse zu vermeiden und die Möglichkeit zu haben, die Standorte per VPN zu vernetzen. FreeSwan [4] als IPSec-VPN-Mechanismus schien bestens geeignet, die Belange der Firma in allen Richtungen abdecken zu können. Blieb noch die Frage, welche Geschmacksrichtung die Linux installation haben sollte. Mein Wunsch, Debian wegen seiner überragenden Administratorfreundlichkeit und bekannten Stabilität einzusetzen, wurde vom verantwortlichen Manager aus Mangel an vertraglich abzusicherndem Support verworfen. Er bestand auf einer kommerziellen Version. Schlussendlich war Red Hat die Wahl, welche sich nicht an allen Stellen bewährte. Bei verschiedenen Systemen sollten zur Erhöhung der Stabilität IDE-Raids eingebaut werden, der gewählte Controller verstand sich jedoch nicht so gut mit dem FreeSwan-gepatchten Kernel. Auf meine Anfrage war die Antwort des Supports wenig erhellend und trug nichts zur Lösung bei. In der Folge wurde auf den Raid-Controller ganz verzichtet. Die spätere Erfahrung zeigte, dass dies die richtige Entscheidung war. „Keep it simple" hatte sich wieder bestens bewährt.

Listing 1

hostnet-Beispiel

```
#name      networkaddress    # comment
##########################################
any      0.0.0.0/0       # the whole Internet!
many     0.0.0.0/1       # lower half, not any!
many     128.0.0.0/1     # upper half, not any!
# our admins need access to noc via ssh, noone else!
admin    172.16.1.1/32     # Sysadmin joe smith
admin    172.17.0.9/32     # Sysadmin peter nobody
#
# on noc resides sspe to admin all the gateways
noc      172.16.0.1/32     # Network Operation Center
#
# internet gateways follow
gw-munich-e  192.168.111.1/32   # Munich gateway
                                          Internet
gw-berlin-e  192.168.112.1/32   # Berlin gateway
                                          Internet
gw-tokio-e   192.168.119.1/32   # Tokio gateway
                                          Internet
gw-all    192.168.111.1/32   # Munich gateway
                                       Internet group
gw-all    192.168.112.1/32   # Berlin gateway
                                       Internet group
gw-all    192.168.119.1/32   # Tokio gateway
                                       Internet group
gw-munich-i  172.16.255.254/32  # Munich gateway
                                          internal
```

Für jeden Standort sollte die Architektur gleich aussehen: Eine Firewall mit RFC1918-Transfernetz zum Router des Internet-Providers, daran angeschlossen ein LAN mit den offiziellen Adressen, die der Provider zu diesem Standort routet, daran je nach Bedarf des Standortes Server (Web, E-Mail, DNS, ...) und ein Gateway, welches nach außen das VPN realisiert und nach innen die internen RFC- 1918-Adressen der Firma auf mehrere LANs in vereinbarten Subnetzen routet. Pro Standort wurde ein /21 festgelegt, 8.192 Adressen sollten für die nächsten Jahre ausreichen. Eine interne Subnettierung in Server-, User- und Leased-LAN ergab sich sinnvoll aus der Tatsache einiger Standleitungen zum Headquarter und zu anderen ausländischen Standorten. Alle Benutzer bzw. Mitarbeiter wurden nach Möglichkeit per DHCP ans User-LAN angeschlossen. Durch registrierte MAC-Adressen und jeweils zugeordneten, festen IP-Adressen wurde dem Missbrauch teilweise vorgebeugt.

Programmierung

Bei der Auswahl, wie zu programmieren sei, spielte Erfahrung die wichtigste Rolle. Shell-Scripts und Perl boten die nötige Flexibilität und insbesondere die Änderungsfreudigkeit, die von einer solchen Sache erwartet wurde und wird. Einige Wochen Tüftelei und Fleißarbeit, vor allem am Wochenende, sicherten erste Testergebnisse, die wiederum in die weitere Programmierung einflossen. Aufgrund leidvoller Erfahrung wurde der Regelsatz auf mehrere Dateien gesplittet, insbesondere die administrativen Regeln, die den Zugang in das jeweilige System herstellen, sollten nach Möglichkeit nur einmal editiert und dann vergessen werden, da hier Fehler schnell zur Abnabelung und damit zu lästigen Telefonaten oder gar Dienstreisen führen. Der Rest des Regelwerks wurde weiter gegliedert, die VPN-Regeln, welche den Verkehr zwischen den VPN-Gateways ermöglichen, wurden ebenso abgekoppelt wie die Benutzerregeln. Eine SSPE-Verzeichnisstruktur unterhalb des Wurzelverzeichnisses des Benutzers entstand:

```
/home/adm/
/home/adm/bin
/home/adm/desc/machine-x
/home/adm/desc/machine-y
/home/adm/desc/machine-z
/home/adm/etc
/home/adm/software
```

Alle ausführbaren Programme sind im *bin*-Verzeichnis, die Konfiguration ist je nach Gültigkeit verteilt in *etc* und den Maschinen-Verzeichnissen unterhalb von *desc*. In *software* sind einige Skripts angesiedelt, die auf den Zeilmaschinen zum Einsatz kommen. Die Definition aller am Regelwerk beteiligter Hosts und Netzwerke ist in *etc/hostnet* abgelegt; eine strikte CIDR-Notation [5] ergab sich von selbst aus den vormaligen Erfahrungen. Gleiche Namen ergeben eine Gruppe, dabei werden für jedes Mitglied entsprechende iptables-Kommandos erzeugt. Auch diese Maßnahme trägt zur Minimierung der Anzahl nötiger Regeln und damit zur Übersichtlichkeit bei.

Eine Besonderheit in der „hostnet“ bzw. den Programmen: Da dem Eintrag *any* in den Programmen spezielle Bedeutung zugeordnet ist, ergibt *many* als Gruppe das, was intuitiv mit *any* verstanden wird. *any* erzeugt neben Forwarding- auch Input- und Output-Kommandos, da jede Firewall mit *any* eben auch gemeint ist. *many* erzeugt im Allgemeinen nur Forwarding-Kommandos. IP-technisch betrachtet ist *many* natürlich identisch mit *any*.

Die Regeldateien werden beim Anlegen einer neuen Maschine als symbolic Link auf die Regeldateien in „etc“ angelegt, der Administrator kann diese jedoch ganz nach seinem Belieben anders verlinken oder für jede Maschine eigene Dateien vorsehen. Dies gilt selbstverständlich auch für die „hostnet“. Bei der Abarbeitung des Regelwerks werden diese Dateien zuerst im Maschinenverzeichnis und danach bei Nichtvorhandensein in „etc“ gesucht. Dies hat eine hohe Flexibilität zur Folge, der Administrator kann seiner Phantasie hier freien Lauf lassen zur effektiven Erreichung seiner Ziele. Die Regeldateien haben alle die gleiche Syntax, jedoch unterschiedlich beabsichtigte Inhalte, die sich aus den Namen ergeben: admin, head, ipsecs, local, users, tail. Beim Anlegen einer neuen Maschine werden nur admin, ipsecs und users als Links im Maschinenverzeichnis auf die entsprechende Datei in „etc“ angelegt, der Administrator hat freie Hand, dies in den Quellen von SSPE zu ändern. In folgendem Listing ist ein Beispiel für eine Regeldatei *rules.admin* zu sehen:

```
# source destination Dir  protocol port policy options
################################################
#
admin   noc       1    tcp      ssh accept
noc     gw-all    1    tcp      ssh accept
noc     any       1    icmp     echo-request accept
any     noc       1    icmp     echo-reply  accept
```

Die Datei *nathosts* gibt Auskunft darüber, welche IP-Adresse zum NAT bei ausgehenden Paketen für welche Sourceadressen nutzbar ist. *privates* enthält die Netzadressen, die zur Überprüfung herangezogen werden, ob NAT stattfindet oder nicht. Private Quelle zu privatem Ziel wird so nie per NAT übertragen. Das *nathosts*-Beispiel:

```
#local net          gateways external ip    # comment
################################################
172.16.0.0/16       192.168.111.1           # Munich
172.17.0.0/16       192.168.112.1           # Berlin
172.23.0.0/16       192.168.119.1           # Tokio
```

Das *privates*-Beispiel:

```
#Network address    # comment
################################################
172.16.0.0/16       # Munich
172.17.0.0/16       # Berlin
172.23.0.0/16       # Tokio
```

Die Konfiguration des VPN mit FreeSwan erfolgt aufgrund der Datei *ipsecs*, diese enthält die beteiligten Standort-Gateways, die dazugehörigen Router und Subnetze. Daraus wird mit SSPE aus dem Menü eine Konfiguration erstellt, die eine voll vermaschte Vernetzung ergeben. Diese kann durch trickreiche Skripte um nicht vermaschte Elemente automatisch erweitert werden, wenn z.B. ein Standort nur per ISDN von genau einem anderen Standort erreicht werden kann. In folgendem Listing ist ein Beispiel für eine *ipsecs*-Datei zu sehen:

```
#Location Gateway      Nexthop          protected-subnet
################################################
Munich  192.168.111.1  192.168.111.254  172.16.0.0/16
Berlin  192.168.112.1  192.168.112.254  172.17.0.0/16
Tokio   192.168.119.1  192.168.119.254  172.23.0.0/16
```

Erweiterte Beispiele zu den Dateien befinden sich bei den Quellen bei Sourceforge [6].

Fähigkeiten des Administrators gefragt

Von vornherein wurde ein Administrations-PC vorgesehen, dessen einziger Nutzen das Beherbergen von SSPE ist. Regelmäßiges Backup versteht sich von selbst, die Qualität in Bezug auf die Filterwirkung bzw. Effektivität der Firewalls hängt hauptsächlich von der des Administrators ab. Die Reihenfolge der Regeln wird durch die verwendeten Programme nicht beeinflusst, hier hat der Administrator seine Kreativität im Sinne der Performanz auszuleben. Nach reiflicher Überlegung während des SSPE-Designs wurde von einer automatisierten Sortierung der Regeln, z.B. absteigend nach dem Produkt der Netzmasken oder ähnlichen Verfahrensgrundlagen, abgesehen zugunsten klarer Durchsichtigkeit des Regelsatzes. Der Sinngehalt des Regelsatzes ist nur durch das fundierte Fachwissen des Administrators zu gewährleisten. Jedoch spielt die Flexibilität des Programms eine Rolle, der Administrator muss in der Lage sein, mit dem Hilfsmittel SSPE seine Vorgaben genau und einfach umsetzen zu können; z.B. mag es wichtig sein, icmp nicht symmetrisch, allerdings nach Typen sortiert zu filtern; einfach abzuschalten ist nicht immer die beste Lösung. Grundsätzlich findet durch die Voreinstellungen ein default-Logging aller nicht erlaubten Pakete statt, für unerwünschte, nicht zu loggende Pakete (z.B. netbios-broadcasts) bieten sich Regeln mit der Aktion „drop“ frei nach dem Geschmack des Administrators an.

Hürden

Wenn aus einem privaten Netz [7] Zugriffe ins Internet erfolgen sollen, sind Proxies, NAT oder eine Kombination aus beiden angesagt. Falls die Proxy-Dienste auf den Firewalls bzw. IPSec-Gateways laufen, wird bei der Initiierung einer TCP-Session automatisch die offizielle Adresse des abgehenden Interfaces in das IP-Paket gesetzt, bei NAT wird das durchlaufende Paket in seiner Absendeadresse modifiziert und eine Tabelle angelegt. Dies kann mit geeigneten iptables-Kommandos gezielt beeinflusst werden. Wenn jedoch aus einem Firmenstandort in einen anderen Pakete geschickt werden sollen, ist dieser Vorgang kontraproduktiv. Ergo ergibt sich daraus:

- Auf dem sendenden Gateway darf kein NAT gemacht werden.
- Entsprechend muss auf dem empfangenden Gateway die absendende Originaladresse gefiltert werden und nicht die externe IP des absendenden Gateway.
- IPSec muss völlig transparent sein.

Eine weitere Einschränkung, die sich im Laufe der Programmierung ergab, war die, dass eine Zuordnung der IPSec-Schnittstelle zu einem Ethernet-Interface nur in der IPSec-Konfiguration eingetragen wird. Innerhalb von SSPE ist implizit vereinbart, dass dies immer eth0 des IPSec-Gateway ist, was in der Folge erst zu Komplikationen führte, als IPSec auch innerhalb der

Firma gefragt wurde. Diese und ähnliche andere Feinheiten galt es in dem Perlscript zu regeln, sodass der Administrator sich um solche Details keine weiteren Gedanken machen muss. Letztlich war es ein einfaches XOR, welches die Lösung in sich barg, der Weg dahin war etwas mühsam. Die Generierung der kompletten FreeSwan-Konfiguration einschließlich dazugehöriger „PreSharedKeys“ und die Verteilung der neuen Konfiguration und der Schlüssel erfolgt aus dem Hauptmenü, hierzu läuft auf den IPSec-Gateways per cron jede Minute ein winziges Shellscript, welches auf neue Konfiguration hin diese aktiviert. Also sollte eine solche neue Konfiguration jeweils in den ersten Sekunden einer neuen Minute erfolgen, um dem Verteilungsmechanismus die Chance zu geben, synchron an allen Standorten die FreeSwan „plutodaemons“ neu zu starten. Damit minimiert sich nur die Ausfallzeit, die Sache funktioniert selbstverständlich zu jedem beliebigen Zeitpunkt. Der Neustart von FreeSwan dauert im Falle korrekter Konfigurationen und Erreichbarkeit aller Standorte nur wenige Sekunden. Hierbei ist zu erwähnen, dass die amtliche Uhrzeit auf den Geräten selbstverständlich ist, nicht nur um IPSec synchron zu starten. Auch die Logdateien wollen evtl. nebeneinander gelegt werden, um Vergleiche anzustellen, dies funktioniert in der Sicherheit verbindlicher Zeitstempel deutlich besser.

> Einfaches XOR barg die Lösung in sich.

Die komplette Gestaltung des verteilten Paketfilterns basierte auf der meist zutreffenden Annahme, dass die Geräte per ssh erreichbar sind. Um eine automatisierte Verteilung zu ermöglichen, ist ein Einloggen per RSA-Keys unerlässlich. Dies hat den Vorteil, von der IPSec-Tunnelfunktionalität völlig unabhängig die Maschinen zu verwalten, die für das IPSec-VPN zuständig sind. Dabei hatten auch schon Tests in der Laborumgebung gezeigt, dass eine Maßnahme in Verbindung mit der voll vermaschten IPSec-Tunnelung sinnvoll ist: die manuelle Deaktivierung einer Maschine innerhalb der Administrationsoberfläche und deren Reaktivierung. Im Internet kann es jederzeit geschehen, dass ein Standort aufgrund von Problemen eines ISP nicht erreichbar ist. Wenn zeitgleich Änderungen an den Regeln erfolgen sollen, muss es einfach möglich sein, die immensen Timeouts der ssh-sessions abzuschalten bzw. zu vermeiden. Dies wird in SSPE dadurch erreicht, dass das betreffende maschinenspezifische Unterverzeichnis in „desc“ einfach durch Voranstellung eines Punktes „unsichtbar“ gemacht wird. Damit wird die Anwendung der Regeländerung auf einen späteren Zeitpunkt verschoben, die erreichbaren Standorte sind zeitnah administrierbar. Bei erneuter Erreichbarkeit kann der Punkt im Verzeichnisnamen per Menü entfernt werden und via „Apply“-Button die Verteilung erneut einfach für alle Maschinen gestartet werden. Automatisch wird dabei überprüft, ob Änderungen für eine Maschine stattgefunden haben, und, wenn ja, wird der neue iptables-Kommandosatz übertragen und ausgeführt.

Fehlermeldungen

Eines der Grundprinzipien wurde während der Entwicklung klar eingehalten: Fehlermeldungen von verschiedenen Maschinen sollten nicht dazu führen, dass der Vorgang der Regelaktivierung unterbrochen oder abgebrochen wird. Ergo werden Fehlermeldungen weitestgehend ignoriert und gesammelt, um am Ende, wenn alle Maschinen fertig abgearbeitet sind, angezeigt zu werden. Ein Fehler in einer Maschine durfte keinesfalls die Gesamtinstallation behindern oder gefährden. Lediglich im Setup der SSPE-Umgebung dürfen Fehler zum sofortigen Abbruch des Dialog-Skripts führen. Wenn man alles richtig macht, geht's auch.

Real Life

Nach einigen Monaten war es dann im April 2002 soweit: Die Mühe sollte erst anfangen durch die Aufnahme des Wirkbetriebs. Die Verkehrsbeziehungen aller beteiligten Rechner zu erfassen und in Regeln umzusetzen dauerte aus verschiedenen Gründen etwa anderthalb Monate. Dabei wurden die Anforderungen meist in Form von Hilferufen und Unterstützungsanfragen gestellt, vieles konnte unmittelbar durch Einsetzen einer Regel und *Apply* erledigt werden. Lediglich Außenbeziehungen gestalteten sich teils etwas schwieriger, da nicht nur unsere Firewallstruktur sich geändert hatte, sondern auch unsere abgehenden Adressen bei den jeweiligen Gegenüber in deren Firewalls eingetragen werden mussten.

Aber die fortwährenden Änderungen brachten einigen Ärger mit sich: Durch Neueinsetzung der Regeln werden selbstverständlich auch die Tabellen im Kernel bereinigt. Dies führt bei bestehenden TCP-Sessions u.U. dazu, dass sie nicht weiter durchgelassen werden. Dies führt zu hässlichen kleinen Fenstern nicht nur bei Oracle-Anwendern, die meist als letzte einer Anwendung deren Ende andeuten. Was tun? Ein einfacher, aber sehr wirkungsvoller Trick half, während der Neueinbringung der iptables-Kommandos nicht auf die aus der Sicherheitsperspektive einzig sinnvolle Policy Drop, sondern auf Accept umzustellen. Die Arbeit des Administrators konnte daraufhin ungestört weiterfließen. Die bewusst in Kauf genommene verminderte Wirksamkeit der Paket-Filter durch diese Entscheidung des Administrators hatte als wesentliche Folge eine stark gestiegene Akzeptanz bei allen Benutzern. Das eingegangene Risiko wurde und wird aufgrund der anderen verwendeten Mechanismen als gering erachtet, da ein Zugriff von außen durch die doppelte Absicherung (Firewall und Gateway) nicht erfolgen kann, wenn die gegenseitigen Abhängigkeiten (apply-options) der Geräte untereinander korrekt eingestellt sind. Damit kann die Anwendung eines neuen Regelsatzes davon abhängig sein, ob der Vorgang bei einem anderen Gerät bereits abgeschlossen ist. Die wechselseitigen Beziehungen unterliegen keinen Restriktionen, sollten jedoch nicht zirkulär sein. ;-)

Betriebserfahrungen

SSPE ist seit 2002 im Einsatz und zeigt auch beim Kunden, dass mit einfacher ASCII-Oberfläche effektiv und zeitnah auf alle Anforderungen aus dem Kreis der Anwender eingegangen werden kann. Durch den Einsatz von speziellen, alten Verschlüsselungsgeräten, die bis in die TCP-Header hinein die Daten verschlüsseln, ergab sich dort bald die Notwendigkeit, IP-Pakete nur in Bezug auf Quelle und Ziel zu filtern, da

die „stateful-inspection-filter“ der Netfilter-Architektur fehlerhafte TCP-Pakete nur verwerfen können. Eine entsprechende Modifikation des Kernels scheint wenig angebracht angesichts proprietärer Verschlüsselungsmechanismen [8], die für sich alleine schon eine ausreichende Sicherheit gegenüber verschiedenen Angriffen suggerieren. Ergo ist das Passierenlassen der korrupten Pakete ein vertretbares Risiko, durch das die Sicherheit des Unternehmens nicht beeinflusst wird. Die Migration dieser antiken Verfahren steht kurz vor dem Abschluss.

In verschiedenen anderen Einzelfällen ergab sich in den letzten beiden Jahren immer wieder, dass neue, bis dahin nicht notwendige Dinge in SSPE eingebaut werden mussten, z.B. die Unterdrückung der Interface-Angabe im iptables-Kommando, der Stateful-Inspection oder die Unterdrückung von NAT aus besonderen Gründen. Dank der Implementierung in Perl und Shell sind die Änderungen jedes Mal einfach und schnell möglich gewesen und in allen Fällen zur Zufriedenheit des Kunden erledigt worden. Einfache Optionen am Ende einer Regel ergeben eine Flexibilität in der Administration, die manche kommerzielle Lösung nicht bieten kann. Anfang 2004 sollten bei einem Kunden bestehende proprietäre Zweigstellenanbindungen und Standleitungen durch IPSec über preisgünstige DSL-Anbindungen abgelöst werden. Dank der Erweiterbarkeit von SSPE konnte eine Maßanfertigung gestaltet werden, die mit vorhandener Hardware und minimierten Linux-Systemen eine bis dahin unbekannte Funktionalität und Transparenz in das erweiterte Kundennetz bringt. Nun sind mehr als 30 Zweigstellen angeschlossen, das Ende der Migration steht bevor. Eine weitere, sinnvolle Anwendung in Verbindung mit Hochverfügbarkeit ist in [9] beschrieben.

Aussichten

Da SSPE freie Software ist und der GNU General Public License unterliegt, wird sowohl jeder Verbesserungsvorschlag als auch jede Kritik gerne angenommen. Von verschiedenen Seiten wurde der Wunsch nach einer grafischen Benutzerschnittstelle laut, ein direkter Nutzen ist gegenüber mangelnder Transparenz der ausgeführten Aktionen nicht erkennbar. Es sähe vielleicht hübscher aus, die Inhalte werden jedoch gleich bleiben müssen. Die simplen Operationen auf Zeichenebene sind nachvollziehbar, einfach und schnell durchführbar, ohne unnötige Bandbreite zu verschwenden. Eine mögliche Erweiterung wäre denkbar in Richtung einer Revisionsfähigkeit, also einem Ablegen der Dateien mit Zeitstempeln und Änderungshistorie in einer Datenbank. Inwieweit sich solche Vorgehensweise mit den Sicherheitsanforderungen verträgt, sei dahingestellt. Dass jede Firewall nur in dem Maße sicher ist, wie der steuernde Kopf paranoid und kenntnisreich zugleich ist, versteht sich von selbst. Paranoid zu sein bedeutet nicht, dass keiner hinter einem her wäre. ;-) Daher ist es selbstverständlich, dass alle beteiligten Geräte nur mit gehärteten Betriebssystemen und monolithischen Kernels betrieben werden. Wer bei seiner Sicherheit ausschließlich auf Paketfilterung setzt, sollte vielleicht doch noch das eine oder andere Buch lesen; mir hat seinerzeit der „Klassiker“ von Bellovin und Cheswick [10] nicht nur sehr gut gefallen, sondern auch die Augen geöffnet. In diesem Sinne: Keep it simple!

Johannes Hubertz kennt Linux seit 0.99..2 und schätzt es, weil er jedes Bit so genau ansehen kann, wie es ihm behagt. Daten-, Host- und Netzwerksicherheit sind seine Leidenschaft, die er bald in selbstständiger Tätigkeit entfalten wird, damit Segeln möglich bleibt.

Links & Literatur

[1] Georg Basse: Linux in heterogenen Netzwerken, in *Linux Magazin* 4/1996, Seite 7

[2] Ralf Burger: Professioneller Einsatz von Linux, in *Linux Magazin* 1/1997, Seite 51

[3] www.kernel.org/pub/linux/kernel/v2.0/linux-2.0.1.tar.gz

[4] www.freeswan.ca/code/super-freeswan

[5] www.rfc-editor.org/rfc/rfc1517.txt, www.rfc-editor.org/rfc/rfc1518.txt und andere

[6] sspe.sourceforge.net

[7] www.rfc-editor.org/rfc/rfc1918.txt

[8] Bruce Schneier: Angewandte Kryptographie. Addison Wesley, 2000

[9] Johannes Hubertz: Wege sind das Ziel, in *Linux Magazin* 4/2005, Seite 70

[10] Bellovin/Cheswick: Firewalls and Internet Security, Addison-Wesley, 1994