simple security policy editor

Der Unterschied zwischen Information und Desinformation ist nicht virtuell.

Johannes Hubertz

20. Juli 2005

Zusammenfassung

simple security policy editor, im Folgenden *sspe* genannt, ist eine verteilte Firewall mit FreeSwan IPSec-VPN für Linux ab 2.4.x Kerneln. Aus Text-Steuerdateien werden Netfilter-Kommandos generiert, BSD-Filter und Router-Accesslisten sind noch zu integrieren. Mehrere Jahre Betriebserfahrung zeigten Effektivität und Stabilität. *sspe* gibts unter GNU General Public License hier:

http://sspe.sourceforge.net/

Inhaltsverzeichnis

1 Hintergrund

Diese kleine Vorgeschichte soll das Verständnis der komplexen Thematik erhöhen. Will der geneigte Leser direkt zum Kern kommen, möge er bitte sofort zum zweiten Abschnitt vorblättern.

1.1 ipfwadm

Als ich 1998 in die interne DV-Abteilung der deutschen Filiale eines europäischen IT-Herstellers wechselte, hatte ich meinen Bellovin [2] als wichtigstes Buch zum Thema Sicherheit verinnerlicht und kannte Linux[1] seit einigen Jahren im Zusammenhang mit DNS- und Webservern. Die benutzten Geräte wurden stets durch ipfwadm-Kommandos in Shellscripts geschützt, Flexibilität war aufgrund geringer Änderungshäufigkeit nicht notwendig. RFCs, Bücher und die Linux-Kernel-Quellen hatten mir etwas Verständnis von IP ermöglicht. Meine Ansicht, daß dies alles elektrisch sei, wurde dennoch nicht von allen Kollegen geteilt. ;-)

Die wichtigste Forderung an Linux-Firewalls war neben der zu erbringenden Sicherheit deren einfache Handhabung im Sinne von Änderungen des Regelwerks, um die vielfältigen Erfordernisse des alltäglichen IT-Betriebes zu erfüllen. Mein Ansatz, alle erforderlichen Daten in einfachen ASCII-Dateien zu sammeln, ergab mit Dialog[2] und Shellscripts eine LPFC[3] genannte Lösung, die den harten Anforderungen des Alltags gewachsen war und vereinzelt bis heute im Einsatz ist. Sie wurde nicht veröffentlicht. Ausgehend von einer Datei

[1] http://www.kernel.org/pub/linux/kernel/v1.0/
[2] http://www.freeos.com/guides/lsst/ch04sec7.html
[3] Linux Packet Firewall Configurator

hostnet[4] mit Definitionen der beteiligten Netz- und Hostadressen und einer anderen Datei mit dem Regelwerk mit einfacher Syntax[5] wurde ein Shellscript erstellt, welches die ipfwadm-Kommandos sofort und bei jedem Boot absetzte. Insbesondere die Möglichkeit, Pakete aufgrund einer solchen Regeloption zu loggen, erwies sich als Entstörungsmaßnahme sehr nützlich. Die Einschränkungen aufgrund der Verwendung von ipfwadm wurden durch die Kompatibiltätsmodi der späteren Kernel und entsprechende Wrapper aufgehoben.

1.2 ipchains

1999 war die neue stabile Kernelversion 2.2[6] da. Neue Funktionalitäten brachte das Userlandprogramm ipchains. Aus konservativen Überlegungen heraus verzichtete ich lange Zeit auf Anpassungen in meinen Scripts. Als ich dann wenig später las, daß ipchains sowieso nur als Übergangslösung gedacht sei, bis eine weitere, nochmals neue Version des Filterns von IP-Paketen als Ablösung fertig sei, entschied ich entgültig, auf ipchains bei meiner Arbeit und in meinen Entwicklungen zu verzichten und bis zu nächsten stabilen Kernelversion 2.4 zu warten. Bei dem Versuch, mittels Gibraltar[7] eine VPN-Verbindung aufzubauen, hatte ich 2001 erneut Kontakt mit ipchains und dies jedoch schnell aufgrund von Zeitmangel aufgegeben, um mich ganz der bis heute aktuellen Netfilter-Architektur auf Linux zu widmen ...

1.3 iptables

Mein Arbeitgeber verkaufte Ende 2001 einen Teil seiner Firma. Hieraus ergab sich die für mich aufregende Aufgabenstellung, das lokale Netzwerk der Zentrale ebenfalls zu teilen, etwa die Hälfte der Serverlandschaft sollte die IP-Nummern wechseln und das Ganze physikalisch getrennt werden. Lediglich ein einziger Netzwerkübergang wurde für eine Übergangszeit gestattet. Dieser war einfach durch die oben beschriebene Architektur zu bewerkstelligen. Ein anderer Aspekt der Teilung war, an den sechs bestehenden Standorten die jeweilig vorhandenen Strukturen unter allzeitiger Wahrung der Konnektivität ebenfalls zu teilen, um anschließend diese wieder miteinander zu verbinden.

Es existierten bereits kommerzielle Produkte mit zentraler Administration, die Filtermechanismen für beliebig viele Firewalls erzeugen und verteilen konnten. Eine unternehmensweite und damit konsistente Filterung dank automatischer Erzeugung faszinierte mich, die Kosten sprachen sehr dagegen. Daher machte ich den Vorschlag, ein ähnliches Konzept mit Linux in Eigenleistung zu erbringen. Einige Diskussionen später waren meine Vorgesetzten überzeugt, auf dem richtigen Weg zu sein: die Kosten der kommerziellen Lösung verglichen mit den avisierten Einsparungen bereiteten auch dem Finanzverantwortlichen Vergnügen.

Die Kernelversion bewegte sich etwas oberhalb von 2.4, stabil genug, um in produktiver Umgebung mit Sicherheitsbedürfnissen eingesetzt zu werden. Netfilter mit iptables war angesagt; mir unbekanntes Neuland, aber ich war bereit, Über- und Lesestunden auch an Wochenden zu leisten. Schnell füllte sich ein Ordner mit HOWTOs, READMEs und anderen Dokumenten zum Thema. Einige Dutzend PC wurden gekauft, zwecks Hardware-Backup fast doppelt so viele wie nötig. Da Manager ungern ohne Netz und doppelten Boden arbeiten, musste eine Linuxversion mit kommerziellem Support zum Einsatz gebracht werden. Also wurde die Geschmacksrichtung *Roter Hut* gewählt. Die Güte bzw. Qualität der Supportleistungen sollte sich erst später erweisen.

1.4 gleiche Anordnung

Um die Sache nicht unnötig zu verkomplizieren, sollte an allen Standorten die gleiche Netzwerk-Anordnung der PC stattfinden. In Abbildung 1 wird ein typischer Standort dargestellt. Eine erste Firewall sollte die extern erreichbaren Server vom Internet trennen, eine weitere Firewall, im folgenden immer als Gateway bezeichnet, die internen Server, Benutzer und Standleitungen jeweils voneinander trennen. Zwischen der Firewall und dem Gateway wurden die vom jeweiligen Provider zugewiesenen Adressen genutzt. Nicht an allen Standorten waren alle Komponenten vorhanden, jedoch konzeptionell vorgesehen. Das Design war so aber für alle Standortanforderungen flexibel genug ausgelegt. Um auch im Kollegenkreis eine verständliche Sprachregelung zu haben, war fortan nur noch von *USER*-, *SERVER*-, *LEASED*-, *RFU*[8]- und *EXTERN-LAN* die Rede. Das *RFU-LAN* wurde nie physikalisch.

Der deutschen Dependence war von der Konzernzentrale ein /16 zugewiesen, bei sechs Standorten und starkem Expansionswillen wollten wir fürs erste mit /19 auskommen. Pro Standort wurde ein /21 für den internen Bedarf vereinbart, davon sollte ein /22 den Anwendern, je ein /24 für lokale Server und lokale Router zu bestehenden Kunden und anderen Nutzern ausreichen. Das restliche /23 sollte für zukünftige andere Zwecke reserviert bleiben.

Die PC wurden in einem abgeschlossenen Raum ohne externe Netzwerkverbindung installiert. Einige Router mit unterschiedlichen Anschlußbandbreiten simulierten die späteren Internetprovider, unsere zugewiesenen Adressen waren schon bekannt. So war das zukünftige Netz der Firma komplett in

[4] Format: **name IP/Netzmaske**
[5] Format: **source destination direction protocol port action options**
[6] http://www.kernel.org/pub/linux/kernel/v2.2/
[7] http://www.gibraltar.at/
[8] Reserved for Future Use

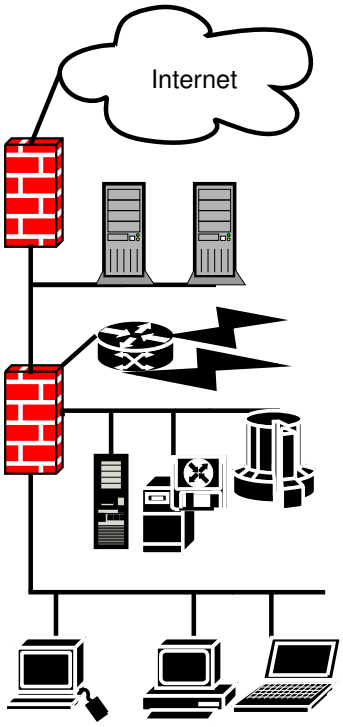

Abbildung 1: Standortdesign

einem Raum simuliert. Nun galt es also nur noch, ein paar konstruktive Ideen zu haben und in die Tat umzusetzen. Die Dokumentation zu Netfilter bzw. Iptables ließ kaum Wünsche offen, eher verwirrend schien die Anzahl möglicher Kommandozeilenparameter. Der Ansatz von LPFC, die Kommandos durch ein Shellscript zu erzeugen, schied wegen der erhöhten Komplexität durch die Anzahl der Maschinen und deren jeweilig speziellen Konfiguration aus. Perl[9] sollte das schneller und beherrschbar machen. Letztlich kam eine Mischung aus Bash[10] und Perl heraus.

2 neue Ideen

Die erneute Verwendung des fantastischen Tools Dialog stand für mich außer Frage. Eine graphische Oberfläche schien mir als Entwickler abwegig, da dies eklatant gegen das oberste Prinzip '*Keep it simple!*' aus Bellovin [2] verstossen hätte. Mein Werkzeug zur täglichen Benutzung sollte so einfach wie möglich und nur so kompliziert wie unbedingt nötig werden. Klar war mir, dass die erzeugten Kommandos von den jeweiligen IP-Adressen und dem eingestellten Routing abhängen. Ergo mussten diese Daten bei der Generierung für jede Maschine vorliegen. Da ich bei Firewalls aus Sicherheitsgründen auf dynamisches Routing generell verzichten wollte, reichte eine einmalige Übertragung dieser Daten nach jeder Änderung aus.

Es gab einige, teilweise auch unangenehme, Erfahrungen mit LPFC. Bei einem Tippfehler in der Definitionsdatei *hostnet* blieb Apply[11] hängen, in der Folge mußte ich zum Gerät laufen oder jemanden anrufen und bitten, auf den Resetknopf zu drücken. Das war peinlich und sollte nach einem Redesign der Vergangenheit angehören.

2.1 teile und herrsche

Der rettende Gedanke war der, die administrativen Regeln, also insbesondere jene, die den Zugang zum Gerät erlauben, vom Rest zu trennen. Da dieser Teil der Regeln erfahrungsgemäß nur selten geändert werden musste, schieden durch die Separierung o.a. Fehler in Zukunft zwar nicht grundsätzlich, aber weitestgehend aus. Konsequenterweise schnitt ich das Restregelwerk in weitere Teile. Die Definitionen aller Netzwerke und Hosts, die am Regelwerk beteiligt waren, sollten in eine Datei, diese sollte zunächst für alle Zielmaschinen gelten. Da aufgrund der unterschiedlichen Konfigurationen der Ziele sich jeweils ein Verzeichnis anbot, konnte darin sehr einfach mittels *Symbolic-Link* auf die zentral abgelegte Datei verwiesen werden. Genau dieses Konzept erwies sich auch für fast alle anderen Dateien als hilfreich. So ergaben sich pro Zielverzeichnis die Regeldateien *rules.admin, rules.head, rules.ipsec, rules.local, rules.users, rules.tail*, diese konnten, mußten aber nicht alle in jedem Verzeichnis vorhanden sein. Das Fehlen sollte nur nur dann zum Abbruch des Apply führen, wenn keine Regeldatei im Zielverzeichnis vorhanden ist, da dann und nur dann ein Fehler des Administrators mit Sicherheit vor-

[9] http://www.perl.org/
[10] http://www.gnu.org/software/bash/bash.html
[11] Anwenden der Firewallregeln, d.h. Einbringen der Kommandos in die Kernel-Tabellen

liegt.

2.2 schaffe Struktur

Eine weitere unschöne Eigenschaft von LPFC war die inkonsistente Schreibweise, in der Gruppen von Hosts und Netzwerken notiert wurden. Aufgrund der völligen Neuschaffung war eine Lösung einfach: CIDR[12] bietet eine einheitliche Schreibweise[13], eine Gruppierung ist über Perl-Arrays einfach herstellbar. Ein Beispiel für *hostnet* zeigt Abbildung 3.

Um gleiche Dinge für mehrere Maschinen zu verwalten, ergab sich eine Verzeichnisstruktur wie in Abbildung 2 gezeigt. Auf den Zielmaschinen sollte so wenig wie nötig installiert sein, die Maschinen sollten gleichartig ausgestattet sein und gleichartig administriert werden. Diese Forderungen waren mit der gewählten Geschmacksrichtung nicht alle zu erfüllen, insbesondere gefiel mir die Installationsvariante mit der vollen X11-Funktionalität nicht. Auf bestimmten Maschinen sollten Services wie Webseiten und DNS präsentiert werden, daher sollten diese Geräte mit preiswerten IDE-Raidcontrollern ausgerüstet werden.

Verzeichnis	**Inhalt**
`~/.`	Arbeitsverzeichnis des Benutzers
`~/bin`	*sspe* shell- und perlscripts
`~/etc`	globale Einstellungen, hostnet, ipsecs, nathosts, privates
`~/desc`	Zielmaschinen
`~/desc/gw-cgn`	Zielmaschine gw-cgn: Regeln (evtl. als sym-links), Routingtabelle, evtl. hostnet, privates
`~/desc/admin`	Zielmaschine admin
`~/software`	Scripts zur Verteilung auf alle Ziele
`~/software.gws`	auf IPSec-Gateways zu verteilende Scripts, z.B. ipsec-supervisor
`~/hardware`	Kernelversionen Backup für alle Ziele Versionen

Abbildung 2: sspe-Verzeichnisse

2.3 professioneller Support

Der Rote Hut ließ sich einfach installieren; meine Idee vom universellen, monolithischen Kernel für alle Maschinen erwies sich jedoch als Falle. Ein auf den Roten Hut neu aufgebrachter, ansonsten fehlerfrei funktionierender Kernel zeigte mit Lilo keinerlei Probleme, bootete jedoch nicht. Der vorherige Kernel aus dem Hut zeigte munter weiter sein /proc. Die Ursache für dieses Verhalten war nicht festzustellen. Daher wurde eine passende Frage mit den Daten zu unserem selbstgebackenen Kernel an den bis dahin unbekannten Support des Roten Hutes geschickt. Da unsere Angaben wahrheitsgemäß den Hinweis auf den FreeSwan-Patch enthielten, war man dort sehr einfach die Sache los: man verwies uns an den Hersteller des FreeSwan-Patches und war schon fertig. Diesem in unseren Augen abwegigen Ansinnen folgten wir nicht. Von Seiten des Controller-Herstellers war kein Support zu erwarten, da dieser seinen Angaben zufolge ausschließlich Systeme aus Redmond unterstützte.

Am Ende wurde entschieden, auf die RAID-Kontroller zu verzichten. Betrachtet im Schatten dreier Betriebsjahre, war dies die einzig richtige Vorgehensweise. Ohne Hochverfügbarkeitsanforderungen sind einfache IDE-Platten ausreichend.

2.4 ignoriere Fehler

Während des Apply können die verschiedensten Fehler auf jeder der beteiligten Maschinen geschehen. Trotzdem muß der Mechanismus für alle anderen weiterlaufen, ein Anhalten, um dem Administrator eine Eingriffsmöglichkeit zu geben, ist nicht vorgesehen. Da jedoch an fast allen Stellen Fehler geschehen können, werden diese bei Auftreten erst gesammelt, um am Ende des Apply dem Administrator in Summe gezeigt zu werden.

Er hat sodann die Pflicht, geeignet für Abhilfe im Sinne des handelnden Unternehmens zu sorgen. Seiner Kompetenz obliegt es, die Maßnahmen wenig betriebsunterbrechend oder besser gleitend so zu gestalten, daß der Geschäftsablauf wenig bis garnicht gestört wird. Bei Betriebsbeginn war den jeweiligen ssh-sessions noch ein ping vorgeschaltet, der die Erreichbarkeit der Zielsysteme aktuell überprüft, dieses Prüfung wurde zugunsten schnellerer Abläufe schnell gestrichen. Infolge wird bei Fehlschlagen der ssh-session durch die TCP-Retries eine erhebliche Verzögerung auftreten. Da der Administrator jedoch den Apply manuell initiiert, wird er auch den Bildschirm beobachten und eingreifen können.

[12] Common Inter Domain Routing: RFC1517, RFC 1518, RFC1519, RFC1520, RFC1817
[13] Adresse/Netzmaske, z.B. 192.168.1.0/24

```
#################################################################
# File: hostnet
# Demonstration of CIDR-noatation
#name                address             # comment
#################################################################
any                  0.0.0.0/0           # all the net
many                 0.0.0.0/1           # lower half
many                 128.0.0.0/1         # upper half
#
admin                192.168.1.195/32    # sspe host
alice                192.168.1.11/32     # admin alice
#
gw-cgn-s             192.168.1.222/32    # server-LAN
gw-cgn-u             192.168.1.126/32    # user-LAN
gw-cgn-e             194.120.12.96/32    # extern
cgn-user             192.168.1.0/25      # dhcp here
cgn-leased           192.168.1.128/26    #
cgn-server           192.168.1.192/27    #
cgn-rfu              192.168.1.224/27    #
cgn-net              192.168.1.0/24      # cgn complete
ptb1                 192.52.103.103/32 # NTP-Source
#################################################################
```

Abbildung 3: *hostnet*

```
# File: rules.admin
# deonstration of sspe-rules
#####################################################################
#src        dst         dir     proto   port    action  options
#####################################################################
alice       admin       1       tcp     ssh     accept
many        admin       1       ip      all     drop    LOG
gw-cgn-e    ptb1        1       udp     ntp     accept  NTP
cgn-user    gw-cgn-u    1       udp     135:139 drop NETBIOS
#####################################################################
```

Abbildung 4: *rules.admin*

2.5 zeige Fortschritt

Bei LPFC wurde das Aktivieren der Regeln durch 'echo'-Kommandos begleitet, die Punkte als Fortschrittsanzeige aufs Display des Administrators ausgaben. Dies diente der Kontrolle, ob das Gerät ordnungsgemäß arbeitet oder zum Telefon gegriffenwerden musste. Da nun aber viele Maschinen gleichzeitig zu überwachen waren, konnte dieses nur anderweitig realisiert werden. Eine der letzten Änderungen[14] ergab nun eine zufriedenstellende Darstellung des Fortschritts. Es werden nun pro Maschine in je einer Zeile der aktuelle Status[15] sekundengenau dargestellt. Die Werte '*sleeping, waiting*' werden gezielt über die Datei *apply-options* im jeweiligen Verzeichnis gesetzt. Als Konsequenz muß evtl. das darstellende Fenster in ausreichender Höhe in Abhängigkeit zur Maschinenanzahl eingestellt werden.

3 ernste Umsetzung

Ab Dezember 2001 waren dann einige Überstunden fällig, die Sechstagewoche war für etwa ein halbes jahr angesagt. Da der normale Wahnsinn des alltäglichen Lebens weiterging, kam nur die Zeit nach Büroschluß oder das Wochenende für eine ruhige, besinnliche Art der Entwicklung in Frage. Alle 5, spätestens 10 Minuten per Telefonklingel aus den notwendigen Gedanken gerissen zu werden, macht keine Freude und auch keinen Fortschritt bei der Arbeit.

Als Entwicklungsumgebung für die neue Administrationszentrale diente ein Intel-PC mit 1GHz Takt, 512kByte Speicher und ausreichend SCSI-Plattenplatz. Hierauf wurde die Geschmacksrichtung Debian stable installiert, um keinerlei Risiken bzgl. der unbekannten Variante einzugehen. Zur bequemen Arbeit diente ein normaler Arbeitsplatz-PC mit *Rotem Hut*, auch um eben dieses kennenzulernen. Die KDE[16]-Oberfläche erwies sich als angenehm, zuvor hatte ich nur von Systemen aus Redmond per telnet und putty, sowie von Debian (80x25 ASCII) aus mit ssh meine Systeme administriert. Die in der Grafik möglichen breiten Fenster überzeugten mich schnell, meinen Code nur so eingeben zu wollen. Per Dialog ein Hauptmenü darzustellen und anschließend mit q&d scripts zur ersten Funktion zu erwecken, dauerte dann nur wenige Tage. Die ersten Übungen mit iptables sollten bald danach die Hürden der Geschichte zeigen –

3.1 zuerst Firewall

Das war gar nicht so einfach, aus dem Wust an Dokumentation genau die Dinge herauszufinden, die gebraucht wurden. Es sollte ja 'so einfach wie möglich und nicht ein-

[14] Version 0.2.5

[15] S ∈ {sleeping, waiting, generating, diff, transferring, executing}

[16] http://www.kde.org/

facher werden'[17]. Top-Down-Design war Wunschkonzept, leider habe ich es nicht immer eingehalten. Zuerst wurde die Hülle in Form eines Hauptprogramms geschrieben und getestet. In seiner Gliederung enthält es Unterprogramme, die wiederum sooft untergliedert werden, bis eine tatsächliche Funktion stattfindet. Im Ergebnis war es Dialog, welches auf Tastenauswahl und 'Enter' ein Shellscript aufruft, dieses verteilt anhand in Dateien vorgefundener Werte auf Linux- oder Routerscripts[18], die wiederum die spezifische Perl-Routine aufrufen, welche die Filter generiert. In genau diesem letzten Schritt müssen die Interface- und Routingtabellen berücksichtigt werden. Ebenso findet hier die Entscheidung über NAT[19] statt. Dies ist bei einer automatischen Regelung nicht ganz trivial, da einige Aspekte diese Entscheidung beeinflussen.

Zur Veranschaulichung diene wieder Abbildung 1, man stelle sich jedoch die Zeichnung an der Internetwolke gespiegelt vor, d.h. im Folgenden werden zwei Standorte A und B betrachtet. Für das Verständnis reicht die Betrachtung zweier Standorte aus, die reale Anzahl trägt nur zur Verwirrung bei.

3.1.1 input oder forward

Anhand der jeweils vorliegenden IP-Konfiguration des Zielgerätes kann für jede Regel einfach entschieden werden, ob Source oder Destination hierfür eine lokale Adresse sind oder ob es sich bei betrachteter Regel um durchgehenden Verkehr handelt. Vier Fälle sind im Allgemeinen zu unterscheiden:

1. Ankommend: INPUT und korrespondierende OUTPUT-Statements
2. Abgehend: OUTPUT und dazugehörige INPUT-Statements
3. durchgehend: FORWARD-Statements
4. vorbeigehend: keine iptables-Kommandos

Die Entscheidung für jede Regel wird anhand der Geräte-IP-Adressen und seiner Routingtabelle gefällt. Hierbei kommen alle IP-Nummern, also auch Aliasadressen, die im Gerät konfiguriert sind, in Betracht. Wenn Quelle und Ziel einer Regel auf unterschiedlichen Zeilen der Routingtabelle liegen, handelt es sich um durchgehenden Verkehr. Sind beide in der letzten Zeile (default-route) enthalten, wird der Verkehr sehr wahrscheinlich am Gerät vorbeigehen, mindestens wird kein iptables-Kommando erzeugt. Dies dient lediglich einem Versuch, die erzeugten Kommandos zu minimieren.

3.1.2 any vs. many

Eine sehr spezielle Betrachtung erhielt die Host-Definition *any* bzw. die eng verwandte Gruppen-Definition *many*. Wenn eine Regel als Source oder Destination *any* enthält, so ist der Traffic in jeder Maschine zu berücksichtigen. In den Perlprogrammen ist hart codiert, dass dann INPUT und OUTPUT und FORWARD generiert werden. Natürlich **muß** hierzu *any* in der *hostnet* mit *0.0.0.0/0* stehen. Dann repräsentiert es das Internet in Gänze. Soll die allumfassende Generierung unterbunden werden, ist stattdessen *many* einzusezten, dies steht als Gruppe in der *hostnet* mit zwei Einträgen: *0.0.0.0/1* und *128.0.0.0/1*. Dies führt nicht zu unerwünschten Nebenwirkungen, außer daß zwei iptables-Kommandos erzeugt werden. Mehr dazu steht selbstverständlich im Quelltext.

3.1.3 case intern-extern

Als erstes konkretes Beispiel sei der Traffic eines internen Hosts am Standort A ins Internet betrachtet. Dieser muß beim Verlassen des Gateway an A NAT erfahren, da nur dann Antwortpakete korrekt eintreffen werden. Dies ist einfach mit der Datei *nathosts* gelöst, welche die internen Adressen am Standort A und die externe Adresse des betreffenden Gateway enthält. Die gleiche Datei mag alle Standorte in gleicher Weise abdecken, so kann anhand der Gleichheit der externen Adresse des Gateways in der *nathosts* und der Interface-Adresse der gerade bearbeiteten Maschine[20] einfach festgestellt werden, daß NAT stattfindet. Der gemeine Trick, die hart codierte Bezeichnung 'gw-XYZ-e' in der *hostnet* ebenfalls in die Entscheidung mit einzubeziehen, sei hier nicht unterschlagen. Er machte diese Sache wirklich einfach durch die zusätzliche Einschränkung, daß NAT überhaupt nur an Maschinen stattfinden kann[21], deren Name in der *hostnet* mit 'gw-' beginnt. Erschwerend kommt hinzu, daß dies ebenfalls wegen harter Codierung nur am Interface *eth0* funktioniert. Dies ist eine unschöne, jedoch konsequenzenlose Eigenschaft, die auch bei der VPN-Konfiguration keine negativen Folgen zeigte. (*ipsec0* ist auch hart an *eth0* gekoppelt) Allerdings sei nicht verschwiegen, daß ich während dieser aufregenden Tage einige Male Auszeit brauchte, um darüber nachzudenken. Die nette Kollegin, mit der ich solche Dinge besprechen konnte, war nicht immer zur Stelle.

3.1.4 case intern-vpn-intern

Diese Methode eignete sich nicht, Traffic von Standort A intern zu Standort B intern zu regulieren. Dieser sollte ja durch IPSec-VPN von einem Gateway zum anderen getunnelt werden und durfte daher in keinem Falle NAT erfahren. Eine weitere Prüf-Logik mit einer weiteren Datei *privates*, die nur die privat genutzten Adressen aller Standorte enthält, kommt ins Spiel. Sofern die in der Regel beteiligten Quell- und Zieladressen beide in einem der in der *privates* beschriebenen Netze ganz oder als Subnetz enthalten sind, wird keinesfalls NAT beim Forwarding generiert. Da der ganze Mechanismus ausschließlich SNAT, also

[17] Albert Einstein

[18] hier ist die Möglichkeit vorgesehen, z.B. BSD einzubeziehen. Die Programmierung für Router wurde aufgrund von Zeit- und Speichermangel nicht beendet.

[19] Network Address Translation

[20] wird aus *parameters* im Maschinenverzeichnis geholt

[21] dies wurde späeter mit den mangling-Dateien aufgeweicht

das Ändern der Quelladresse kennt, ist diese einfache Entscheidungsfindung ausreichend.

3.1.5 case intern-intern

Dieser Fall war durch die *privates* gleich mit erschlagen. Für die Erzeugung der iptables-kommandos spielt die VPN-Tunnelung keine Rolle.

Dennoch ist eine wesentliche Erkenntnis aus dem FreeSwan-IPSec Betrieb zu entnehmen: Die forwarding-Kommandos dürfen keine Interface-Optionen enthalten, da bei Down/Up des VPN die Routing-Tabelle heftig durcheinandergeworfen wird mit der Folge, dass Traffic u.U. nicht weitegeleitet wird. Bei den ssh-sessions zur Verwaltung der Firewalls ein fatales Fehlverhalten, welches ohne Konsequenzen einfach zu vermeiden war. Da stets *ipsec0* auf *eth0* abgebildet wird, deren Stati jedoch unabhängig voneinander sind, gibt es manchmal Probleme. FreeSwan bzw. *pluto*[22] stellt Routen im Kernel der Maschine ein, falls *eth0* in den Status 'Down' versetzt wird, fallen die vorherigen Routen von *eth0* auf *ipsec0* und verbleiben dort. Dies war nur durch Beenden des VPN, 'Down/Up' des *eth0* und anschliessendem Neustart des VPN zu beheben. Eine wirkliche Erklärung würde ein sehr intensives, tiefes Hinabtauchen in die gepatchten Quellen des Kernels erfordern, die hierzu nötige Zeit wollte mir niemand geben. So blieb das seltene, aber mehrfach beobachtete Phänomen stets etwas nebulös.

3.2 dann VPN

Mit FreeSwan oder basierenden Implementierungen wie OpenSwan oder StrongSwan ist ein IPSec-VPN mit einfachen Mitteln erstellbar, lediglich der Kernel muß einen Patch erhalten. Im Zuge der Kernelerzeugung werden die notwendigen Userlandprogramme an die richtige Stelle gebracht, um dem Administrator zur Verfügung zu stehen. Mit einem zusätzlichen X.509-Patch kann FreeSwan um Routinen zur Handhabung von X.509-Zertifikaten zur Authentisierung erweitert werden.

3.2.1 preshared keys

Damit stehen zur gegenseitgen sicheren Erkennung zwei Mechanismen zur Wahl: **preshared Keys** und **X.509-Zertifikate**. Zur Verbindung der Standorte untereinander wurde der Einfachheit halber die erste Variante gewählt. Zur Verteilung der Schlüssel dient ssh. Die automatisch erzeugten Schlüssel bestehen im wesentlichen aus einer MD5-Summe. In diese gehen Datum, Uhrzeit und ein Zufallswert, also vermutlich ausreichend Entropie aufgrund der ebenfalls zufälligen Uhrzeit der Erzeugung ein.

```
# FILE: ipsecs.internet
# loc.  gateway          next-Hop         subnet
bln     172.22.0.41      172.22.0.46      10.11.48.0/21
cgn     172.22.0.25      172.22.0.30      10.11.40.0/21
nyc     172.22.0.65      172.22.0.70      10.11.4.0/22
sdy     172.22.0.17      172.22.0.22      10.0.0.0/8
kap     172.22.0.9       172.22.0.14      10.11.56.0/21
tok     172.22.0.1       172.22.0.6       10.11.16.0/21
to2     172.22.0.1       172.22.0.6       10.11.80.0/21
```

Abbildung 5: private-subnets

3.2.2 ipsec.conf

Die IPSec-Konfigurationsdatei bestand aus einem allgemeinen Teil und verbindungsspezifischen Abschnitten; mit Perl war das einfach durch *print*-Statements herstellbar. Aus verschiedenen Gründen stand fest, ausschließlich *eth0* für IPSec zu nutzen, daher wurde an dieser Stelle zugunsten der Einfachheit auf jede Flexibilität verzichtet. Da alle beteiligten Geräte die gleichen, eigenen Mechanismen verwenden sollten, waren auch die definierenden Elemente wie in Abbildung 5 gezeigt aufs Minimum zu beschränken. Zur Konfiguration der beteiligten Außendienstlerverbindungen reichte eine konstante anzuhängende Verbindungsdefinition, die auf die benutzten Zertifikate verweist.

3.2.3 roadwarrior

Für Außendienstler, die an wechselnden Standorten mit Wählverbindungen, also stets wechselnden IP-Adressen arbeiten, ist ein vordefinierter Schlüssel unbrauchbar, da in der FreeSwan- Konfiguration die IP-Adressen zum Schlüssel anzugeben ist. Also kommt hier ein X.509-Zertifikat zum Zuge, dieses kann aufgrund der verwendeten Schlüssellänge (2048Bit) und der vertrauenswürdigen Unterschrift der eigenen Zertifizierungsstelle zur Authentisierung ausreichen. Im Sinne der verwendeten Firewall ist dies jedoch mit einem gravierenden Nachteil verbunden: Der Mitarbeiter arbeitet mit einer nicht vorhersehbaren IP-Adresse. daraus folgt, daß in den Filtern allen Adressen die jeweiligen Dienste freigeschaltet sein müssen. Dies scheint wenig sinnvoll, auch das interne Routing muß nicht alle Adressen umfassen. Ein Trick erschien wie Licht am Ende eines dunklen Tunnels.

3.2.4 L2TP durch IPSec

Mit L2TP[23], also PPP über IP, kann einem entfernten System eine zusätzliche IP-Nummer incl. routing verpasst werden. Die Authentisierung kann mit einfachem Usernamen/Passwort auch automatisch er-

[22] pluto ist das Userlandprogramm für den Schlüsselaustausch bei FreeSwan
[23] layer two tunneling protocol

folgen. Damit besteht im Unternehmen die Chance, allen Mitarbeitern interne wohlbekannte IP-Adressen zuzuordnen.

3.2.5 private PKI

Selbstverständlich sind zur Erzeugung von X.509-Zertifikaten Routinen Erforderlich, die Mechanismen sind Dank ssleay[24], später als OpenSSL[25] bekannt, frei verfügbar. Mit OpenCA[26] oder TinyCA[27] sind Oberflächen hierzu vorhanden, die Wahl muß nach anderen, unternehmensweiten Kriterien und Richtlinien erfolgen. Eine Verwaltung der Zertifikate ist unumgänglich, ausscheidenden Mitarbeitern wird man den Zugang zu internen Systemen vorenthalten wollen. Die CRL[28] der benutzten CA[29] ist hierzu das geeignet Mittel, welches von FreeSwan berücksichtigt wird. Sie zu verteilen ist Aufgabe des CA-Administratrs, er hat ein geeignetes Script, um diese Datei an alle erforderlichen Ziele zu verteilen. Sinnvollerweise wird dieser Vorgang weitgehend automatisiert, nach Möglichkeit ausschließlic mit ssh unter Umgehung der IPSec-Tunnel.

4 lockerer Betrieb

Als die ersten Gehversuche in der Laborumgebung erfolgreich Angriffe mit nmap überstanden hatten, konnte der Sprung ins kalte Wasser gewagt werden. Anfang April 2002 hatten wir die ersten beiden Standorte mit 2M-Standleitungen ausgestattet, die Infrastruktur wurde innerhalb von wenigen Stunden dupliziert[30]. Die Maschinen waren schnell aufgestellt, ssh funktionierte auf Anhieb. Die IPSec-Verbindung zum anderen Standort auch. Damit war die erste Hürde geschafft. Aufmerksame Beobachtung der Logdateien und derer statistischen Eigenschaften folgte, die in Zeitungen berichteten Wurm- und Virenwellen waren direkt nachvollziehbar.

4.1 startup (2002)

Im Laufe weniger Wochen waren die anderen vier Standorte in das Szenario eingebunden, in Köln entstand eine spezielle Konstruktion zur Vorbereitung: Ein zusätzliches Interface wurde an der Firewall in Gang gesetzt, welches über einen Router die jeweiligen Adressen des nächsten Standortes direkt auslieferte. So konnte der Verkehr mit echen Adressen getestet werden, wenn auch nur zwischen Köln und dem neuen Ort. So vorgetestet konnten die Geräte per Spedition ausgeliefert werden und am endgültigen Standort angekommen direkt in Betrieb gehen.

4.1.1 default accept

Bei mehr als 400 über mehrere Standorte verteilten Arbeitsplätzen ergab sich sehr bald eine Änderungsfrequenz in den Firewallregeln von mehreren Ereignissen pro Arbeitstag. Die Anwender merkten üblicherweise nichts von dem. Nur einige wenige Orakelbenutzer aus der Buchhaltung klagten bald über heftige Unterbrechungen. Jeweils die Anmeldeprozedur und Suche nach Fragmenten des Absturzes mußte wiederholt werden. Nach Bekanntwerden der Ursache wollte man mir meine Arbeit auf die Nacht verschieben. Ich wollte lieber tagsüber arbeiten und sann auf Abhilfe. Ein Versuch, die default-policy während des Einbringens der Filter-Regeln auf Accept zu stellen, zeigte Wirkung: die Oracle-Sessions überstanden den Apply ohne Schaden. Um nun sicherheitstechnisch keinen Durchzug zu erzeugen, sollten Firewall und Gateway nicht mehr gleichzeitig, sondern stets zeitversetzt mit neuen Regeln versorgt werden. Erste, schnelle Lösung hierzu war ein einfaches *sleep* vor der Verteilung zu den Gateways. Die Zeit mußte nur länger sein, als das Einbringen in die Firewalls dauert.

4.1.2 ISP-Besonderheiten

Wenn der ISP nicht so kann, wie der Kunde möchte, ist Improvisation angesagt. Am Standort Berlin (Stadtrandlage) war noch kein DSL verfügbar, erst Monate später sollte ein solcher Anschluß gelegt werden können. Ein ISDN-Basisanschluß konnte sofort geschaltet werden, mit einem passenden Router wurde mangels offizieller Adressen eine Standort-Struktur mit RFC1918-Adressen aufgesetzt und per ISDN-Einwahl mit dem zentralen Standort Köln verbunden. Dies hatte den Vorteil, zur Umschaltung auf DSL nur wenige rechner umnumerieren zu müssen, alle Verkabelung konnte schon realisiert werden. Weiterer Vorteil war, intern nichts mehr ändern zu müssen. IPSec läßt sich auch über ISDN transportieren, die einzige Einschränkung gegenüber den anderen Standorten war, keine volle Vermaschung zu haben. Von den anderen Standorten wurde ein zusätzlicher IPSec-Tunnel mit den Berliner Adressen nach Köln gelegt, damit war interner Verkehr von und zu allen Standorten gegeben ...

4.2 und Kundenwünsche

Alle kundenseitigen Anforderungen, die im Laufe des Betriebs nicht sofort erfüllt werden konnten, führten zu Änderungen an den Programmen. Diese sind im Changelog rudimentär dokumentiert, welches den Quellen beiliegt. Mein Augenmerk galt dabei stets der Funktionalität im Sinne des Auftraggebers, als alleiniger Entwickler spielte für mich weitergehende Dokumentation keine Rolle. Andererseits steht sowieso nur im Quelltext, was ein Programm macht, wenn man von unbeabsichtigten Nebeneffekten absieht. Was ein Programm machen soll, mag zwar den Vermarkter interessieren, ist letztlich aber nicht von Belang.

[24] Erste, frei verfügbare Implementierung von SSL, heute nicht mehr aktuell
[25] http://www.OpenSSL.org
[26] http://www.OpenCA.org
[27] http://tinyca.sm-zone.net
[28] certificate revokation list, Liste zurückgewiesener Zertifikate
[29] certificate authority, Zertifizierungsstelle
[30] ein Backup-Konzept existierte bereits

Es mag Kunden geben, die bereits eigene, möglicherweise proprietäre Verschlüsselungsmechanismen benutzen. Diese verunstalten die IP-Pakete, z.B. stimmen die TCP-Checksummen der verschlüsselten Pakete nicht mehr. Damit ist jede Firewall vollständig überfordert. Selbst die einfache Filterung nach Portnummern wird unmöglich. Als Konsequenz bleibt, die IP-Pakete nur aufgrund der IP-Nummer in der Zieladresse passieren zu lassen und ausschließlich auf die Sicherheit des Verschlüsselungsgerätes zu vertrauen.

Mit der eigenen Programmstruktur war diese Anforderng schnell in Perl geschrieben, wo sonst in der Regel die Portnummer zu stehen hat, kann das Schlüsselwort *all* gesetzt werden. Da die Verschlüsselungsgeräte zwischen Firewall und Gateway standen, musste IP zu diesen nur die Firewall queren.

4.2.1 zweiter Kunde

Später führte ein etwas kniffliger Fehler zu einer Verbesserung, die auf eine konfigurierte Abhängigkeit hinauslief. Neben meiner Tätigkeit für die hausinteren Dienste war auch Geschäft mit der Kundschaft abzuwickeln. Ein Vertrag war neu entstanden, demzufolge die Rechner eines Kunden bei meinem Arbeitgeber aufgestellt und betrieben werden sollten. Hierzu war zunächst je ein Internetanschluß mit 2Mbit/s vorgesehen, diese sollten nach wenigen Monaten Testbetrieb um je einen weiteren mit 4Mbit/s Bandbreite ergänzt werden. Und so kam es, daß *sspe* in einer zweiten Instanz aufgesetzt wurde und fortan fürs Kundengeschäft seine Tauglichkeit ebenfalls unter Beweis stellen konnte. Zunächst wurde die erste Verbindung als Backup beibehalten und bei Bedarf umgestöpselt. Das war nicht nur fachlich unelegant gelöst, sondern mir auch unangenehm; manuelle Eingriffe setzen stets physische Anwesenheit voraus...

```
# FILE: ipsecs.internet
# loc.  gateway          next-Hop         subnet
bln     172.22.0.41      172.22.0.46      10.11.48.0/21
cgn     172.22.0.25      172.22.0.30      10.11.40.0/21
nyc     172.22.0.65      172.22.0.70      10.11.4.0/22
sdy     172.22.0.17      172.22.0.22      10.0.0.0/8
kap     172.22.0.9       172.22.0.14      10.11.56.0/21
tok     172.22.0.1       172.22.0.6       10.11.16.0/21
to2     172.22.0.1       172.22.0.6       10.11.80.0/21
I01     172.22.0.25      172.22.0.30      0.0.0.0/1
I02     172.22.0.25      172.22.0.30      128.0.0.0/3
I03     172.22.0.25      172.22.0.30      160.0.0.0/5
I04     172.22.0.25      172.22.0.30      168.0.0.0/6
I05     172.22.0.25      172.22.0.30      172.0.0.0/12
### !!! never open next line or gateways will be lost !!!!
### !!!Ixx 172.22.0.25  172.22.0.30      172.16.0.0/12 !!!
I06     172.22.0.25      172.22.0.30      172.32.0.0/11
I07     172.22.0.25      172.22.0.30      172.64.0.0/10
I08     172.22.0.25      172.22.0.30      172.128.0.0/9
I09     172.22.0.25      172.22.0.30      173.0.0.0/8
I10     172.22.0.25      172.22.0.30      174.0.0.0/7
I11     172.22.0.25      172.22.0.30      176.0.0.0/4
I12     172.22.0.25      172.22.0.30      192.0.0.0/3
```

Abbildung 6: public-subnets

4.2.2 ungwöhnliche Ports

Ein Dienstleister sollte sinnvollerweise auf die Belange seiner Kunden eingehen. Daher kamen einige Kollegen auf die Idee, beim Kundenaufenthalt die dort aussschließlich erlaubten Webzugriffe auf Port 80 zum Eingang in unsere Firma zu nutzen. Einen Openssh-Server auf Port 80 mag zwar unüblich, jedoch nützlich sein. Da dieser nicht nur von festen Kundenadressen, sondern auch von Heimarbeitsplätzen genutzt werden sollte, schied jede Nutzungs-Einschränkung durch IP-Filter aus. Dank der Möglichkeit des OpenSSH-Dämons, einzig mit RSA-Keys zu authentisieren, können sich nur Bekannte[31] anmelden und den Service nutzen. Allerdings sollten hier wie sonst auch immer stets die jeweils neuesten Patches eingespielt werden. Vielleicht nicht ganz uninteressant ist in diesem Zusammenhang, daß für diesen Zweck ein eigener Rechner aufgesetzt wurde.

4.3 das zweite Jahr

Nichttechnische Gründe führten Anfang 2003 zu einem Wechsel des Internetproviders. Eine Umnummerierung später waren einige Nerven verloren und die Erkenntnis gewonnen, daß mit *sspe* nicht alles, aber fast alles zu machen ist. Eine wesentliche Einschränkung hatte sich gezeigt: NAT und IPSec nur auf eth0 zuzulassen war für diesen Anwendungsfall nicht unmittelbar geeignet. Schließlich sollte es nur noch einen Internetanschluß für die ganze Firma geben und die Standorte durch ein MPLS-VPN des Providers verbunden werden. Als negative Folge waren am zentralen Standort zwei meiner IPSec-Gateways aufzustellen, damit konnten Außendienstler weiterhin per IPSec zugreifen und die Standorte nach innen mit IPSec transparent verbunden werden. Der besseren Handhabbarkeit halber wurde die Installation mit je

[31] RSA-Keys sind aufgrund der Anzahl Bits in jedem Falle deutlich schwieriger zu raten als achtbuchstabige Passwörter

einer sspe-Installation fürs Internet und interne VPN weiterbetrieben. Dies hat sich bewährt, wenngleich durch die Flexibilität symbolischer Links eine Instanz ausreichend gewesen wäre.

4.3.1 Nichts ist unmöglich

In RFC 2917 ist nachzulesen, das MPLS zwar ein Übersprechen[32] des einen auf das andere VPN zu verhindern hat, eine Verschlüsselung ist jedoch nicht zwingend vorgesehen. Daraus folgerte ich unwidersprochen, daß dann auch nicht verschlüsselt werde und infolgedessen unser bestehendes VPN diese weiter zu leisten habe. Nur wie geht ein Internet durchs Nadelöhr?

IP-Routing ist einfach: eine Maske bestimmt, wie viele Nummern zu einer Netz-Adresse gebündelt werden. Internet wird normalerweise als *0.0.0.0/0* geschrieben. Doch, wie bereits erwähnt, kann es auch anders dargestellt werden. Meine Gateways sollten weiterhin direkt per ssh durch NAT erreicht werden. Also reichte die Halbierung zur Tunnelung nicht aus, eine Lücke für die Gateway-Adressen durfte nicht in IPSec-Tunnel eingepackt werden. Die Folge: eine Subnettierung wie in Abbildung 6 gezeigt. Jenes **172.16.0.0/12** wird vom MPLS-VPN genutzt. Die Gateways tunneln den Rest, um am zentralen Standort alle Internet-Dienste per NAT erreichbar und dennoch effektiv gefiltert zu gestalten. Die von DSL bekannten problematischen MTU-Werte waren dank FreeSwan stets unkritisch, selbst ein weiteres, gekapseltes IPSec-VPN für spezielle Projekte durch das IPSec-VPN im MPLS-VPN war funktional und ausreichend performant.

4.3.2 VPN und HA

Die Sicherheit eines VPN basiert auf der Geheimhaltung der verwendeten Schlüssel. Einmal erzeugte, primäre Schlüssel[33] oder X.509 Zertifikate dienen dazu, regelmässig neue Session-Keys auszuhandeln, das Verfahren ist ausreichend kompliziert und muß hier nicht dargestellt werden. Voraussetzung ist, daß die primären Schlüssel auf sicherem Wege in die Geräte kommen. Mit FreeSwan wird der Austausch von Parametern[34] und die Verhandlung sekundärer Schlüssel vom Dämon *pluto* arrangert. Hochverfügbarkeit eines VPN unter Beibehaltung zeitlich wechselnder Session-Keys mit zwei oder mehr Geräten erfordert den Transfer der primären und sekundären Schlüssel; mit der Paranoia eines Administrators ist das nur schwer zu vereinabren. Wie dennoch eine Hochverfügbarkeit unter Wahrung aller kryptographischen Finessen hergestellt werden kann, ist bereits in [5] beschrieben worden.

4.4 drei Jahre

Der Betrieb ging ins dritte Jahr, durch zahlreiche lange Telefonate war die nette Kollegin immer wieder willkommene Gesprächspartnerin, wenns um anstehende Lösungen neuer Aufgabenstellungen ging. Auf der Suche nach einem passenden Denkansatz für viele gleichartige Gateways war *sspe* sicher gut, mir aber aufgrund der mit DSL täglich wechslenden IP-Nummern nicht gut genug. Eine Dynamik in die *hostnet* einzubauen, schien mir zu abenteuerlich. Nie wollte ich die Sicherheit eines Unternehmens davon abhängig machen, daß ein externer Dienst, wie z.B. dyn-dns.org, funktioniert. Der Einsatz von FreeSwan stand außer Frage, die zentrale Seite war mit festen IP-Adressen ausgestattet. Zertifikatsbasierte Authentisierung war mittels X.509-Patch einfach machbar, also galt es nur, eine geeignete Struktur aufzubauen. Die vorhandene PKI sollte weiterhin genutzt werden.

4.4.1 kleine Filialen

Ende 2003 wollte ein Kunde seine Perspektiven aufgezeigt haben, wie mit den noch verwendeten proprietären Mechanismen (Firewall und Verschlüsselungsgeräte) gleitend in eine offene Welt zu migrieren sei. Es stand der Anschluß von einigen dutzend Außenstellen an, am liebsten wollte man das mit „alter“PC-Harware realisieren. Jeweils zwei bis zwanzig Arbeitsplätze sollten mit den zentralen Rechnern arbeiten.

Ich empfahl den Wechsel zu Debian GNU/Linux, darauf SSPE als Firewall und IPSec-VPN sowie den Kauf neuer Maschinen hierzu für die Zentrale. In Hardware zu investieren, leuchtete dem Kunden unmittelbar ein. Die alte Firewall einschließlich der Umgebung war abgeschrieben. Die notwendige Dienstleistung dazu einzukaufen, war er etwas weniger schnell bereit. Meine Argumentation, auch bei kommerzieller Software sei Dienstleistung zur Installation und insbesondere auch zum Betrieb nötig, überzeugte dann auch den Vorstand, allerdings sollte noch bis zum nächsten Jahr gewartet werden. Ein weiterer Kunde wollte ebenfalls eine kleine Zweigstelle mit zwei Arbeitsplätzen an seinen Host anschliessen.

4.4.2 Prototypenbau

Das traf sich günstig, die Entwicklung war nur einmal zu tun. Debian verfügt über Mechanismen, das System zu verkleinern und diesen Zustand stabil zu halten, debfoster ist das passende Debianpaket. Zur Installation eines kleinen, neuen Systems bietet sich die rescueCD[35] von Timo Blenk bestens an. Um das prototypenhaft gebaute Debiansystem zu seiner spezifischen Aufgabe fertig zu konfiurieren, waren einige Handgriffe in ein Script zu gießen. Die Konfigurationsdateien wurden an den entscheidenden Stellen mit Platzhaltern versehen, die dann *sed* ersetzt. Ein ganzes System eingepackt in einem gezippten tar war nur einige 10 Mbyte klein, auf der rescueCD ausreichend Platz für Scripts zur automatischen Partitionierung, Formatierung und Anpassung an die IP-Konfiguration des endgültigen standortes. DSL-Kennung und Passwort sind bei Ablauf des Scripts einmal in eine

[32] Der Provider setzt MPLS-VPNs auf seinen Routern für mehrere Kunden gleichzeitig ein, dies ist sein Kostenvorteil
[33] Preshared Keys
[34] IP-Adressen der Tunnel, Timer und mehr
[35] http://rescuecd.sourceforge.net

dialog-Maske einzugeben, sie werden nur im Endgerät an der passenden Stelle gespeichert. Natürlich sollte das Gateway über Firewallfunktionalität verfügen, sspe schied aufgrund der dynamischen IP-Nummern aus. Jedoch kann ein Satz iptables-Kommandos auch durch ein einfaches Script erzeugt werden. Der Verkehr zur Zentrale wird transparent durchgelassen, dieser wird mit sspe dort feinreguliert. Verkehr mit dem Internet (Email, Webseiten) wird ebenfalls über die Zentrale abgewickelt, so daß auch alle zentralen Virenschutzmaßnahmen greifen.

5 heitere Aussichten

5.1 Zukunft

Ab August wird sspe in der hubertz-it-consulting GmbH, Sitz in Köln, weiterentwickelt. Wichtigstes Ziel erscheint zur Zeit die Einbindung von BSD bzw. Solaris-Systemen mit IPF-Kommandos zu sein. Herr Ney wünscht eine einfachere Installtaion, dies ist mir angesichts der Zeit, die mit dem Werkzeug verbracht wird, nicht wichtig. Die Installation ist absolut unwichtig gegenüber der Nutzungsdauer.

5.2 Dank

Mein Dank gilt meinem ehemaligen Vorgesetzten, der mir erlaubte, sspe zu veröffentlichen. Die aufgrund meiner Initiative entstandene Lösung kann so auch anderen nutzen. Herrn Georg Greve[36] danke ich für seine Ermutigung, in der Sache unbeirrt fortzufahren und für seine Vorstellung von SSPE in der Brave GNU World[37]. Ebenso danke ich Herrn Achim Leitner vom LinuxNewMedia Verlag. Er unterstützte mich nicht nur bei meiner Veröffentlichung[38], sondern machte mich auch auf eine Dissertation zum Thema aufmerksam. Er regte Herrn Christian Ney zu seinem Artikel [6] über SSPE an, dem ich auch herzlich wegen seiner positven Beurteilung danke.

5.3 Persönliches

Angesichts zunehmender Werte bzw. schützenswerter Güter, die auch weiterhin in Unternehmen wie Behörden dargestellt und per Informationstechnologie verfügbar gemacht werden, parallel damit überproportional ansteigenden Verlust-, Verfälschungs- und Mißbrauchsrisiken sollte IT-Sicherheit bestmögliche Minimierung all dieser und anderer Risiken leisten. Vertraulichkeit und Integrität von Daten kann mit moderner Verschlüsselungstechnick hergestellt werden. Ebenso kann diese Technik zur Erhöhung der Verfügbarkeit von IT beitragen, also die Profitabilität steigern. Hierbei muß es vielleicht nicht immer der allerletzte modische Schrei sein, wenigstens erste Exploits sollten mittels Patches der Erzeuger wirkungslos bleiben. Eine in diesem Sinne konservative Herangehensweise ist die vielleicht beste Art, die unternehmerischen Risiken des IT-Betreibers zu handhaben. Auch Releasewechsel sind nicht immer fortschrittlich im Sinne der IT-Sicherheit ...

Einem überprüfbaren endlichen Automaten steht meinerseits mehr Vertauen entgegen als einer Blackbox, deren Hersteller in seinen Lieferbedingnungen alle Garantien ausschließt. Eine Appliance, die vor Aufnahme der Firewalltätigkeit erst eine tcp-session mit verschlüsseltem Datentransfer zu ihrem Hersteller aufbaut, ist mir zutiefst suspekt. Das Mystische aus der Welt zu schaffen, sollte im Bereich der IT-Sicherheit selbstverständlich sein. Dieses Ziel ist offensichtlich noch lange nicht erreicht. Offene Quelltexte können und sollen einen wesentlichen Beitrag dazu leisten, Bereiche von IT-Sicherheit prüfbar und beweisbar zu gestalten. Der Einbau von Hintertüren wird durch die Möglichkeit der Einsichtnahme deutlich erschwert. Die verbreitete Einsicht ins Kerkhoffsche Prinzip wird dabei vielleicht gleichzeitig weitervermittelt. Zur Geheimhaltung von Inhalten sollte die Geheimhaltung von Algorithmen nichts, die Geheimhaltung der verwendeten Schlüssel jedoch alles beitragen. So bleibt nur anzumerken:

Wovon man nicht sprechen kann, darüber muß man schweigen.[39]

Literatur

[1] M. D. Bauer. *Building SECURE SERVERS with LINUX*. O'Reilly, 2002.

[2] Bellovin and Cheswick. *Firewalls and Internet Security*. Addison Wesley, AT&T, 1994.

[3] C. S. . P. W. . M. Erwin. *Virtuelle Private Netzwerke*. O'Reilly Verlag, 1999.

[4] FreeSwan.org. http://www.freeswan.org, 2002.

[5] J. Hubertz. Wege sind das ziel. *Linux Magazin*, pages 70–72, 04 2005.

[6] C. Ney. Simple security policy editor. *Linux Magazin*, pages 70–73, 07 2005.

[36] Free Software Foundation Europe
[37] Ausgabe 54, Linux-Magazin 10/2003
[38] Linux-Magazin 4/2005, Wege sind das Ziel
[39] Tractatus Logicus Philosophicus, Ludwig Wittgenstein