Firewalling bei IPsec-Einsatz unter Kernel 2.6

Sicherer Brandstifter

Seit Entwicklerkernel 2.5.45 ist IPsec fester Bestandteil von Linux. Dabei löste eine neue Implementierung das bisherige Freeswan ab. Das neues Design verzichtet aber auf virtuelle Interfaces und bereitet dadurch dem Zusammenspiel mit Netfilter-Firewalls große Probleme. Michael Becker, Sebastian Claßen

Dass IPsec viele Firewall-Techniken untergräbt, ist den meisten Admins bekannt. Die Firewall kann nicht in die Innereien eines IPsec-verschlüsselten Pakets blicken und darum nicht nach Dienst und Port filtern.

IPsec gefiltert

Es bleiben in der Praxis dennoch Szenarien, die den kombinierten Einsatz von IPsec und Firewalls erfordern. Sinnvoll ist dieses Paar zum Beispiel in einem Endsystem, dessen Firewall sowohl Klartext-Pakete als auch IPsec-Verkehr empfängt. Nach dem Entschlüsseln und Auspacken soll die Firewall noch unterscheiden können, ob das Paket ursprünglich mit IPsec-Schutz versehen war. Überraschenderweise ist das unter Kernel 2.6 aber nur mit wesentlich größeren Anstrengungen möglich als noch unter Kernel 2.4.

Bei den IPsec-Implementierungen für Kernel 2.4 (Freeswan [3], Openswan [4] und Strongswan [5]) stellt die Kernelkomponente KLIPS virtuelle Interfaces zur Verfügung. Nachdem der IPsec-Stack die empfangenen Pakete ausgepackt und entschlüsselt hat (Decapsulation) und noch bevor er zu sendende Pakete verschlüsselt und verpackt (Encapsulation), durchlaufen die Pakete die Netfilter-Hooks des virtuellen Interface im Klartext.

Dank dieses Interface ist das Filtern vergleichsweise einfach. Die Regeln können zwischen IPsec- und Nicht-IPsec-Verkehr unterscheiden und die IPsec-Pakete im Klartext analysieren. **Abbildung 1** zeigt, durch welche Netfilter-Hooks die Pakete in welchem Stadium laufen.

Als Beispiel soll ein Rechner nur IPsec-geschützte Daten akzeptieren. Die Firewall-Policy darf auf dem physikalischen WAN-Interface »eth1« dazu ausschließlich ESP-Pakete sowie UDP-Pakete auf Port 500 (IKE-Daemon) und Port 4500 (NAT-Traversal) gestatten. Die Paketfilter sind in der »INPUT«- und »OUTPUT«-Chain des virtuellen Interface zu platzieren. Arbeitet der Rechner als Gateway und schützt ein privates LAN, dann kommen zusätzlich Regeln in der »FORWARD«-Chain zwischen virtuellem Interface »ipsec0« und LAN-Interface »eth0« zum Einsatz.

Der Weg der Pakete durch die einzelnen Netfilter-Hooks ist für Tunnel- und Transportmodus identisch. Im Transportmodus entfallen jedoch die Regeln für die »FORWARD«-Chain, da bei der direkten IPsec-Verbindung die Endpunkte selbst das Entschlüsseln der Pakete übernehmen. **Listing 1** zeigt den Auszug eines typischen Firewallskripts, passend für IPsec unter Kernel 2.4.

Die Befehle in den Zeilen 15 bis 17 setzen die Default-Policys der Filter-Chains auf »DROP«. Danach definieren die Kommandos ab Zeile 34 die Regeln für ESP, IKE und NAT-Traversal auf dem WAN-Interface. Die Regeln in Zeile 57 und 58 erlauben einen Echo-Request über IPsec an das Gateway und das entsprechende Echo-Reply. Der geschützte Zugriff auf einen POP3-Server im LAN, etwa durch einen Roadwarrior, wird von den Regeln ab Zeile 74 gestattet.

Die Regeln aus **Listing 1** zeigen auszugsweise, dass das Filtern ein-, aus- und weitergehender Pakete über das »ipsec0«-Interface recht übersichtlich erfolgt und sich nicht von Regeln für ein physikalisches Interface unterscheidet.

Kernel 2.6

Die IPsec-Implementierung in Kernel 2.6 [6] verzichtet leider auf virtuelle Interfaces. Darüber hinaus ist der Weg der Pakete durch die Netfilter-Hooks im Tunnelmodus anders als im Transportmodus. Kommt auch noch NAT-Traversal zum Einsatz, dann ist das Verkehrschaos perfekt. Es sind komplizierte Zusatzfeatures aus dem Netfilter-Framework nötig (Mark-Target oder IPsec-Policy), um wenigstens einige der Probleme in den Griff zu bekommen. Andere lassen sich selbst damit nicht lösen und erfordern experimentelle Kernelpatches, die

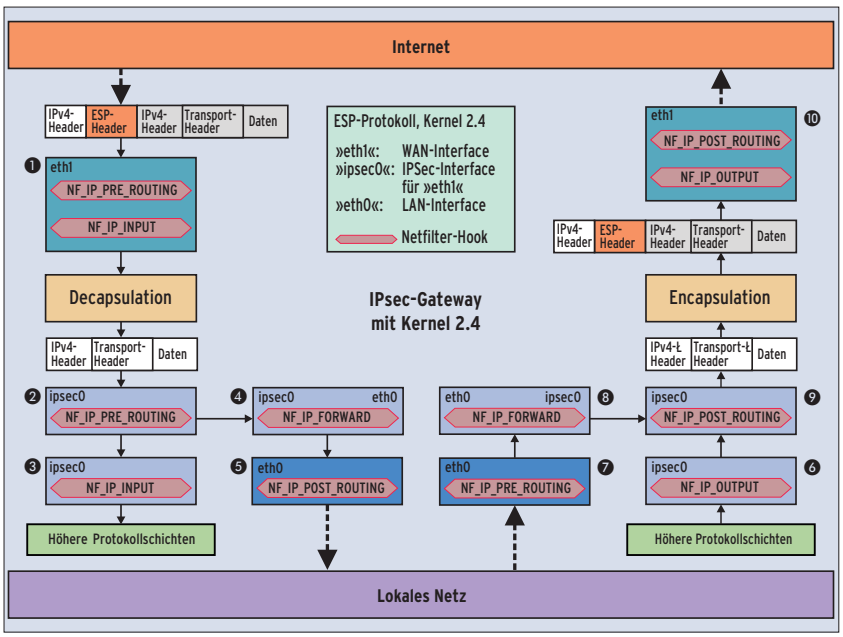

Abbildung 1: Die ESP- und Klartext-Pakete durchlaufen bei Freeswan (Kernel 2.4) mehrere Netfilter-Hooks. Das WAN-Interface »eth1« nimmt ein ESP-Paket aus dem Internet entgegen (❶). Unabhängig vom Ziel leitet der IPsec-Stack das ausgepackte Klartext-Paket zunächst an »ipsec0« (❷). Pakete für den lokalen Rechner bleiben in »ipsec0« (❸), alle anderen gehen über einen weiteren Hook in »ipsec0« (❹) an das LAN-Interface »eth0« (❺). Beim Versenden ist der Ablauf umgekehrt (❻ bis ❿).

Netfilter-Core-Mitglied Patrick McHardy entwickelt hat (IPsec-Hooks).

Im Tunnelmodus durchläuft ein ESP-verschlüsseltes Paket die »PREROUTING«- und »INPUT«-Chains des WAN-Interface »eth1«. Nach der Decapsulation passiert das Paket im Klartext erneut die »PREROUTING«-Chain des WAN-Interface »eth1« und, abhängig von der Zieladresse, ein zweites Mal die »INPUT«-Chain oder die »FORWARD«-Chain von »eth1« (siehe Abbildung 2). Das Problem ist: Ausgepackte Pakete sind nicht mehr von einem neu eingetroffenen Paket gleichen Typs zu unterscheiden. Ob das Paket ursprünglich als ESP-Paket eintraf, lässt sich beim Durchlaufen der Netfilter-Hooks im unverschlüsselten Zustand nicht mehr feststellen.

Als Ausweg muss der Kernel ESP-verschlüsselte Pakete markieren und die unverschlüsselte Fassung beim erneuten Netfilter-Durchlauf auf diese Markierung hin überprüfen. Das Markieren der Pakete erledigt das »MARK«-Target; ein IPtables-Match-Filter prüft, ob ein Paket markiert ist. Eine Markierung für ausgehende Daten ist nur dann nötig, wenn die Firewall-Policy bestimmte Pakettypen ausschließlich unverschlüsselt, nicht aber im Tunnel erlauben soll. Eine »DROP«-Regel für ESP-Pakete mit gesetzter Marke verwirft in diesem Sonderfall unerwünschte Pakete.

Egal ob ein Dienst nur im Tunnel oder auch unverschlüsselt erlaubt sein soll, die »FORWARD«- oder »OUTPUT«-Chain muss ihn zunächst akzeptieren. Das relevante Filtern findet in der »POSTROUTING«-Chain statt, die auf »DROP«-Policy gesetzt ist. Erlaubt sind in dieser Chain nur ESP- sowie jene Pakete, die den Rechner auch unverschlüsselt verlassen dürfen.

In Abbildung 2 sind die Stellen gekennzeichnet, an denen Netfilter-Regeln die Markierungen setzen oder prüfen müssen. Listing 2 setzt die Theorie in die Praxis um und erreicht die gleiche Funktion wie Listing 1. Auch hier handelt es sich nur um einen kleinen Auszug.

Schutzmarke

Der Kernel verwaltet Netzwerkpakete in der Datenstruktur »sk_buff«, die neben einem Zeiger auf die Paketdaten viele weitere Variablen für paketspezifische Informationen enthält. Eine dieser Varia-

Listing 1: IPsec-Firewalling auf Kernel 2.4

```
10 # eth1   - physikalisches WAN-Interface (öffentliche IP-Adresse)
11 # ipsec0 - virtuelles Interface gebunden an eth1
12 # eth0   - physikalisches LAN-Interface (private IP-Adresse)
13
14 # Standardmäßig alle Pakete verwerfen
15 iptables -P INPUT   DROP
16 iptables -P OUTPUT  DROP
17 iptables -P FORWARD DROP
[...]
34 ### ESP und optional IKE und NAT-Traversal auf WAN-Interface erlauben
35 # ESP
36 iptables -A INPUT  -p esp -i eth1 -s 0.0.0.0/0 -d ${IP_eth1} -j ACCEPT
37 iptables -A OUTPUT -p esp -o eth1 -s ${IP_eth1} -j ACCEPT
38
39 # IKE (udp 500)
40 iptables -A INPUT  -p udp -i eth1 -s 0.0.0.0/0  --sport 500 -d ${IP_eth1}
   --dport 500 -j ACCEPT
41 iptables -A OUTPUT -p udp -o eth1 -s ${IP_eth1} --sport 500 -d 0.0.0.0/0
   --dport 500 -j ACCEPT
42
43 # NAT-Traversal
44 iptables -A INPUT  -p udp -i eth1 -s 0.0.0.0/0  --sport 4500 -d ${IP_eth1}
   --dport 4500 -j ACCEPT
45 iptables -A OUTPUT -p udp -o eth1 -s ${IP_eth1} --sport 4500 -d 0.0.0.0/0
   --dport 4500 -j ACCEPT
[...]
55 ### via IPsec <--> Gateway: PING
56 # Ping über IPsec auf das Gateway von jeder IP-Adresse erlauben
57 iptables -A INPUT  -p icmp --icmp-type 8 -i ipsec0 -s 0.0.0.0/0 -j ACCEPT
58 iptables -A OUPTUT -p icmp --icmp-type 0 -o ipsec0 -d 0.0.0.0/0 -j ACCEPT
[...]
73 ### via IPsec <--> LAN: POP3
74 iptables -A FORWARD -i ipsec0 -o eth0 -p tcp -s 0.0.0.0/0 --sport
   ${UNPRIV_PORTS} -d ${IP_LAN_POP3} --dport 110 -m state --state NEW -j
   ACCEPT
75 # Aufgebaute Verbindungen und damit zusammenhängenden
76 # Verkehr in FORWARD-Chain erlauben
77 iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

blen heißt »nfmark«, sie speichert die Netfilter-Markierungen als Unsigned-Long-Variable:

```
struct sk_buff {
        ...
# ifdef CONFIG_NETFILTER
        unsigned long   nfmark;
        ...
};
```

Beim Ver- und Entschlüsseln ändert das IPsec-Modul nur das Paket, nicht aber die Struktur, durch die es im Kernel repräsentiert wird. Damit bleibt der Wert von »nfmark« erhalten, IPtables kann ihn setzen und später wieder prüfen. Anders als bei den Targets »ACCEPT« oder »DROP« bricht eine Verzweigung »-j« zum Target »MARK« die Bearbeitung der folgenden Firewallregeln nicht ab. Das Setzen einer Netfilter-Markierung ist nur in den Chains der Mangle-Tabelle möglich: »iptables -t mangle -A *Kette* .... -j MARK --set-mark *Wert*«. Zum Beispiel:

```
iptables -t mangle -A PREROUTING -p esp ➥
  -j MARK --set-mark 0x00000001
```

Überprüfen lässt sich die Markierung in beliebigen Netfilter-Tabellen und allen Chains, die das zu prüfende Paket durchläuft. Eine Regel kann den Wert der Markierung prüfen: »-m mark --mark *Wert*«. Ein Beispiel:

```
iptables -A INPUT -p icmp -m mark ➥
  --mark 1 -j DROP
```

Statt eines einfachen Werts ist es sogar möglich, zusätzlich eine Maske zu übergeben: »--mark 1/1«. In diesem Fall verknüpft (AND) Netfilter die Maske Bit für Bit mit der Markierung, bevor es das Ergebnis mit dem Sollwert vergleicht und über das weitere Schicksal des Pakets entscheidet (Abbildung 3).

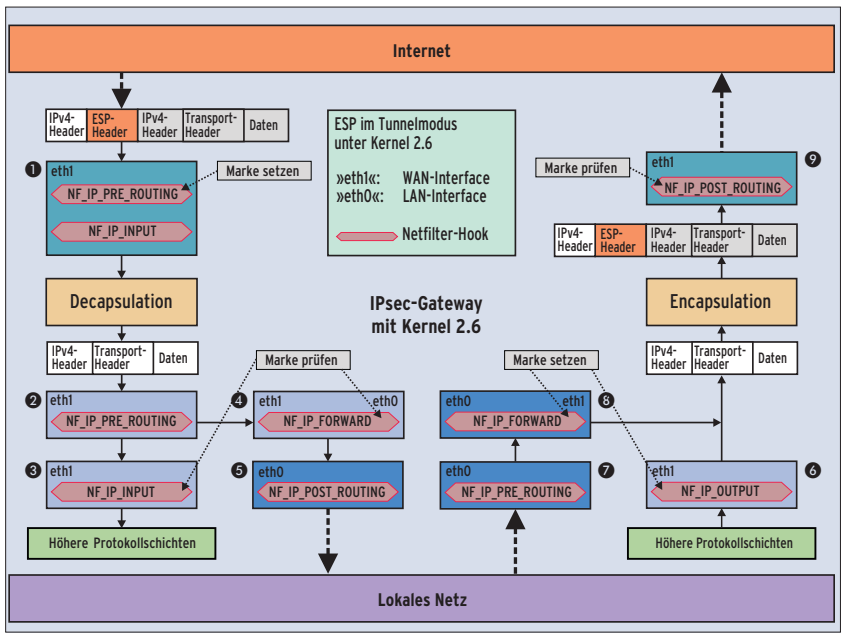

Abbildung 2: Im Tunnelmodus von Kernel 2.6 durchläuft ein IPsec-Paket die Eingangsfilter von »eth1« doppelt. Erst nimmt das WAN-Interface »eth1« ein ESP-Paket an (❶). Unabhängig vom Ziel leitet der IPsec-Stack das ausgepackte Klartext-Paket erneut an »eth1« (❷). Pakete für den lokalen Rechner bleiben in »eth1« (❸), die anderen gehen über einen weiteren Hook in »eth1« (❹) an das LAN-Interface »eth0« (❺). Am Klartext-Paket ist nur noch durch spezielle Markierungen erkennbar, ob es über IPsec eingetroffen ist.

Eleganter mit Policy-Modul

Neben dem recht primitiven Markierungsverfahren führt auch das neue Policy-Modul zum Ziel. Dazu ist allerdings der Netfilter-Code im Kernel auf den neuesten Stand zu bringen (siehe Kasten „Kurzanleitung Patch-o-matic-NG“). Ähnlich der Paketmarkierung greift das Policy-Modul auf eine Variable in der Struktur »sk_buff« zurück, diesmal ist es jedoch eine Struktur »struct sec_path« und keine einfache Zahl.

Der IPsec-Stack des Kernels setzt den Inhalt dieser Variablen selbst. Damit müssen die IPtables-Regeln sich nicht mehr um das Setzen und Prüfen der Markierungen kümmern. Die Regeln für das

Listing 2: IPsec-Firewalling auf Kernel 2.6

```
09 # eth1 - physikalisches WAN-Interface (öffentliche IP-Adresse)
10 # ipsec0 - virtuelles Interface gebunden an eth1
11 # eth0 - physikalisches LAN-Interface (private IP-Adresse)
12
13 # Policies standardmäßig auf DROP setzen
14 iptables -P INPUT   DROP
15 iptables -P OUTPUT  DROP
16 iptables -P FORWARD DROP
17 iptables -t mangle -P POSTROUTING DROP
[...]
29 ### ESP
30 # Eintreffende ESP-Pakete auf eth1 in PREROUTING markieren
31 iptables -t mangle -A PREROUTING -i eth1 -p esp -j MARK --set-mark 1
32
33 # Eintreffende ESP-Pakete auf eth1 in INPUT erlauben
34 iptables -t mangle -A INPUT -i eth1 -p esp -j ACCEPT
35
36 # Ausgehende ESP-Pakete auf eth1 erlauben
37 iptables -t mangle -A POSTROUTING -o eth1 -p esp -j ACCEPT
38
39 ### IKE und NAT-Traversal (identisch zu Listing 1, Zeile 40-45)
[...]
75 ### via IPsec <--> Gateway: PING
76 # Eingehenden Echo-Request mit Markierung erlauben
77 iptables -A INPUT -p icmp -i eth1 --icmp-type 8 -m mark --mark 1 -s
   0.0.0.0/0 -j ACCEPT
78 # Ausgehenden Echo-Reply erlauben
79 iptables -A OUPTUT -o eth1 -p icmp --icmp-type 0 -d 0.0.0.0/0 -j ACCEPT
[...]
93 ### via IPsec <--> LAN: POP3
94 iptables -A FORWARD -i eth1 -o eth0 -m mark --mark 1 -p tcp -s 0.0.0.0/0
   --sport ${UNPRIV_PORTS} -d ${IP_LAN_POP3} -dport 110 -m state --state
   NEW -j ACCEPT
95 # Aufgebaute Verbindungen und damit zusammenhängenden
96 # Verkehr in FORWARD-Chain erlauben
97 iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

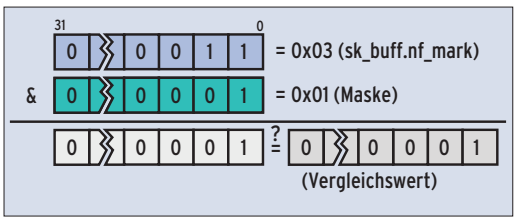

Abbildung 3: Bevor Netfilter die Markierung mit dem Wert vergleicht, maskiert das Modul die Marke Bit für Bit per AND-Verknüpfung.

Ping-Beispiel aus Listing 2 lassen sich wie folgt neu schreiben:

```
iptables -A INPUT -i eth1 -m policy ➥
  --dir in --pol ipsec --mode tunnel ➥
  --proto esp -p icmp --icmp-type 8 ➥
  -j ACCEPT
iptables -A OUTPUT -o eth1 -m policy ➥
  --dir out --pol ipsec --mode tunnel ➥
  --proto esp -p icmp --icmp-type 0 ➥
  -j ACCEPT
```

Befindet sich zwischen zwei IPsec-Endpunkten ein NAT-Router, kommt es wegen der Architektur der IPsec-Protokolle zu einem Problem. Im Gegensatz zu TCP oder UDP arbeitet ESP ohne Portnummern. Der NAT-Router ist aber auf Ports angewiesen, um die Pakete eindeutig ihrer Verbindung zuzuordnen. Wenn mehrere Clients gleichzeitig über das NAT-Gateway mit dem gleichen Server kommunizieren, sind Portnummern das einzige Kriterium, um die Antworten zu unterscheiden.

Eine Technik namens NAT-Traversal schafft Abhilfe und sorgt dafür, dass mehr als ein Client eine IPsec-Verbindung durch ein NAT-Gateway aufbauen kann. Der IPsec-Stack verpackt das ESP-Paket in UDP (ESP-in-UDP-Encapsulation genannt). Das UDP-Pro-

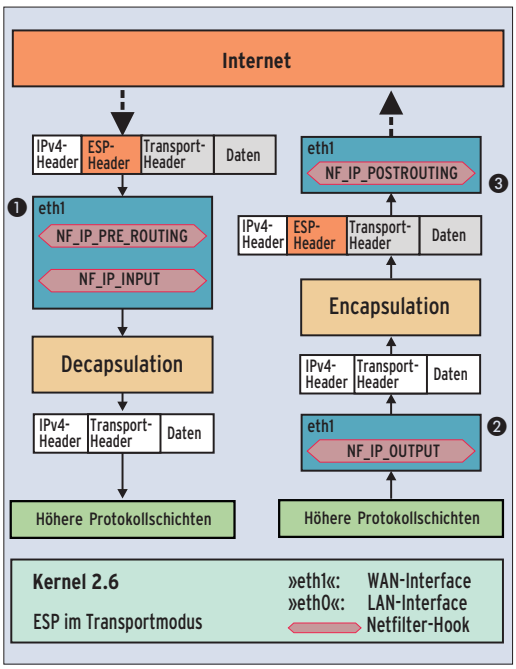

Abbildung 4: Im Transportmodus von Kernel 2.6 sind eintreffende Pakete nur verschlüsselt zu sehen (❶). Nach dem Entschlüsseln leitet der Kernel das Paket direkt an höhere Protokollschichten weiter, es durchläuft keine weiteren Netfilter-Hooks mehr.

Kurzanleitung Patch-o-matic-NG

Das Netfilter-Projekt hat mit Patch-o-matic-NG ein komfortables Patch-Programm entwickeln. Neue Kernelfeatures sind damit schnell nachgerüstet: Die tagesaktuelle Version von IPtables und Patch-o-matic-NG (kurz »pomng-*Datum*«) herunterladen [7], mit Root-Rechten nach »/usr/src« entpacken und einen symbolischen Link anlegen:

```
tar xvjf patch-o-matig-ng-Datum.tar.bz2
tar xvjf iptables-Version.tar.bz2
ln -s iptables-Version iptables
```

In das Patch-o-matic-NG-Verzeichnis wechseln, die Umgebungsvariablen »KERNEL_DIR« und »IPTABLES_DIR« setzen und das Patch-o-matic-NG-Utility aufrufen:

```
cd /usr/src/patch-o-matic-ng-Datum
export KERNEL_DIR="/usr/src/linux"
export IPTABLES_DIR="/usr/src/iptables"
./runme Suite
```

Der Suite-Parameter wählt einen Patch-Zweig, derzeit aus »pending«, »base« und »extra«. Statt einer ganzen Suite kann man auch das Patch-Verzeichnis angeben, um ein einzelnes Patch zu benennen. Beispielsweise installiert »./runme policy« das Policy-Patch; es fügt in IPtables ein IPsec-Policy-Match ein.

Anschließend den Kernel konfigurieren und übersetzen (»make menuconfig; make«). Das eben eingefügte Policy-Modul steht im Konfigurationsprogramm unter »Device Drivers | Networking support | Networking options | Network packet filtering | IP: Netfilter Configuration | IPSec policy match support«.
Zum Abschluss die aktuelle Version von IPtables neu kompilieren. Dazu in das IPtables-Verzeichnis wechseln und Make mit den korrekten Verzeichnissen aufrufen:

```
make BINDIR=Dir LIBDIR=Dir MANDIR=Dir
make BINDIR=Dir LIBDIR=Dir MANDIR=Dir ➥
  install
```

tokoll arbeitet mit Ports und löst so das Zuordnungsproblem. Der Austausch der UDP-Pakete zwischen den beiden IPsec-Endpunkten erfolgt über Port 4500, den die Firewall auch freigeben muss (siehe Listing 1, Zeilen 44 und 45).
Der Nachteil für die Firewall: Bei NAT-Traversal sehen die Netfilter-Hooks im Kernel nur noch UDP-Pakete statt der ESP-Pakete (Abbildung 2). Für IPtables bleibt der besondere Inhalt des UDP-Pakets verborgen, das ESP-Paket passiert nach dem Entfernen des UDP-Headers keine Netfilter-Chains mehr. Erst nach dem Entschlüsseln durchläuft das Klartextpaket die gleichen Filter, die auch im Tunnelmodus ohne NAT-Traversal zur Anwendung kommen. Im Transportmodus ist der Einsatz von NAT-Traversal allerdings sowieso unsicher.
Im Transportmodus stellt der IPsec-Stack von Kernel 2.6 den Firewall-Designer vor beinahe unlösbare Probleme. Nachdem eintreffende ESP-Pakete die »PREROUTING«- und »INPUT«-Chain hinter sich gelassen haben, werden sie entpackt und direkt an die höheren Protokollschichten weitergereicht. IPtables erhält keinen Zugriff auf die Klartext-Pakete, wie Abbildung 4 zeigt. In dieser Situation bleiben NF-Mark und Policy-Patch wirkungslos. Hier helfen nur noch experimentelle Patches von Netfilter-Core-Mitglied Patric McHardy.

Transportmodus

Ausgehende Pakete durchlaufen genau wie im Tunnelmodus zuerst die »OUTPUT«-Chain, werden danach verschlüsselt und verlassen als ESP-Pakete das WAN-Interface über die »POSTROUTING«-Chain. Die »FORWARD«-Chain spielt keine Rolle, da im Transportmodus zwei Endpunkte direkt miteinander kommunizieren.
Die McHardy-Patches setzen einige Erfahrung im Umgang mit Kernelpatches voraus. Unter Umständen bedarf es manueller Nachhilfe, bis sie sauber im Kernel integriert sind. Einfacher ist es, den Tunnelmodus statt des Transportmodus einzusetzen. Das zieht zwar als kleinen, aber meist verschmerzbaren Nachteil einen größeren Protokoll-Overhead nach sich. Dafür bietet dieser Modus mehr Sicherheit, da ESP das gesamte innere IP-Paket verschlüsselt.

Neue IPsec-Hooks

Die McHardy-Patches sind im »extra«-Repository von Patch-o-matic-NG zu finden. Folgendes Kommando installiert die Input- und Output-Hooks:

```
./runme ipsec-01-output-hooks
./runme ipsec-02-input-hooks
```

Das experimentelle Stadium der Patches bringt einige Seiteneffekte mit sich – nachzulesen in der Zusammenfassung des Netfilter-Workshops 2004 [8]. Sind die Patches sauber in den Kernel integriert, stellt sich fast das aus Kernel 2.4 gewohnte Bild ein. Es fehlen nur noch die virtuellen Interfaces: »ipsec0« aus Abbildung 1 ist durch das physikalische Interface »eth1« zu ersetzen. Der durch

IPsec-Protokolle und -Modi

IPsec standardisiert zwei Protokolle, die IP-Pakete schützen: AH (Authentication Header) und ESP (Encapsulating Security Protocol). Beide gehören zur IP-Protokollfamilie (Nummern 50 und 51). AH schützt die Integrität des ganzen IP-Pakets einschließlich IP-Header, verschlüsselt aber nichts. ESP verschlüsselt den Paketinhalt und schützt die Integrität des inneren Pakets – ohne den äußeren IP-Header.

Transport- und Tunnelmodus

Außerdem sind Transport- und Tunnelmodus zu unterscheiden. Ersterer eignet sich nur für die direkte Kommunikation zweier Rechner, die dabei die IPsec-Endpunkte bilden. Im Transportmodus befindet sich der ESP- oder AH-Header zwischen IP-Header und Payload. Die Payload umfasst die Header der höheren Protokollschichten und Daten.
Anders verhält es sich im Tunnelmodus. AH- oder ESP-Header stehen hier vor dem IP-Header des Orignalpakets. Vor dem IPsec-Header wird ein weiterer IP-Header eingefügt, der die Adressen der Tunnelendpunkte enthält. Die Payload besteht im Tunnelmodus aus einem kompletten IP-Paket. Beim ESP-Protokoll ist damit das gesamte innere IP-Paket verschlüsselt. Ein frühere Artikel [6] erklärt den Aufbau des AH- und ESP-Headers und die Zusammensetzung der Pakete.

IPsec in der Kritik

Die Krypto-Experten Ferguson und Schneier empfehlen in ihrer Analyse des IPsec-Protokolls [2] unter anderem, das AH-Protokoll und den Transportmodus aus dem IPsec-Standard zu entfernen. Probleme, die sich aus AH für das Firewall-Design ergeben, berücksichtigt der vorliegende Artikel daher nicht.

die Patches korrigierte Weg der Pakete ist nun für Tunnel- und Transportmodus gleich. Trotz dieser Modifikation sind Paketmarkierung oder das Policy-Modul weiter erforderlich.

Die Autoren dieses Artikels arbeiten an einem Patch [10], das auch im Kernel 2.6 virtuelle Devices implementieren soll. Der Code befindet sich aber noch in der frühen Entwicklungsphase. Das Virtual-Device-Patch modifiziert das XFRM-Framework des Linux Kernels. Die IPsec-Protokolle benutzen XFRM fürs Ver- und Entschlüsseln der Pakete. Zum Framework gehört die »xfrm_state«-Struktur, sie repräsentiert die IPsec-SA (Security Association) im Kernel. Der IKE-Daemon soll künftig beim Anlegen dieser Struktur den Namen eines virtuellen Device für diese Verbindung hinterlegen. Derzeit legt in »net/xfrm/xfrm_state.c« die Funktion »xfrm_state_alloc()« das Device noch selbst fest.

Der Entwurf von Firewallregeln für IPsec gestaltet sich unter Kernel 2.6 unnötig kompliziert. Dass Sicherheit mit steigender Komplexität abnimmt, ist allgemein bekannt. Der Verzicht auf ein virtuelles Interface ist nach Meinung der Autoren daher ein Fehler; die Regeln für Kernel 2.4 waren wesentlich übersichtlicher.

Guter alter 2.4

Es bleibt zu hoffen, dass die Kernelentwickler den bisherigen Ansatz auch in Kernel 2.6 übernehmen. Viele der Probleme mit der IPsec/IPtables-Kombination lassen sich mit den beschriebenen Verfahren und Patches aber auch jetzt schon lösen. *(fjl)* ■

Infos

[1] USAGI: [http://www.linux-ipv6.org]
[2] Bruce Schneier, „A Cryptographic Evaluation of IPsec“: [http://www.schneier.com/paper-ipsec.html]
[3] Freeswan: [http://www.freeswan.org]
[4] Strongswan: [http://www.strongswan.org]
[5] Openswan: [http://www.openswan.org]
[6] Ralf Spenneberg, „Sicherer Transport – Virtuelle Private Netze mit Linux 2.6 und IPsec“: Linux-Magazin 05/03, S. 60
[7] Netfilter-Projekt mit IPtables und Patch-o-matic-NG: [http://www.netfilter.org]
[8] Harald Welte, „Summary of the netfilter developer workshop 2004“: [http://www.netfilter.org/documentation/conferences/nf-workshop-2004-summary.html]
[9] Achim Leitner, „Transport-Sicherheit – VPN mit verschiedenen Protokollen und Programmen“: Linux-Magazin 10/03, S. 23
[10] Experimentelles Virtual-Device-Patch: [http://netfilter.basti79.de]
[11] Quellen zum Artikel: [ftp://ftp.linux-magazin.de/pub/listings/magazin/2004/12/IPsec-IPtables/]

Die Autoren

Sebastian Claßen und Michael Becker studieren Technische Informatik an der Hochschule Niederrhein und beschäftigen sich in ihren Abschlussarbeiten mit der Absicherung eines WLAN mittels IPsec.