Eine modulare Netfilter-basierte IPSec-Implementierung

Michael Conrad, *Universität Karlsruhe*
Ulrich Mohr, *Univeristät Karlsruhe*
Stefan Mink, *Schlund+Partner AG Karlsruhe*
Frank Pählke, *Universität Karlsruhe*
Marcus Schöller, *Universität Karlsruhe*

Abstract

In diesem Paper wird das Konzept und die Realisierung einer RFC-konformen IPSec-Implementierung unter Linux, basierend auf vorhandenen Standardschnittstellen vorgestellt.
Das Konzept basiert auf einem modularen Ansatz. Durch die Modularität soll es ermöglicht werden, verschiedene Implementierungen von Verschlüsselungsverfahren in Hard- und Software einfach zu integrieren und parallel zu nutzen. Alle Teile der Implementierung sind als ladbare Kernel-Module realisiert, so dass sie zur Laufzeit dem Kern hinzugefügt werden können.
Als Zugangspunkt zur IP-Schicht wurde Netfilter gewählt. Dadurch ist es möglich, an allen notwendigen Stellen in Netzwerkstack einzugreifen und dort eine Verarbeitung der IP-Pakete zu veranlassen. Durch die Nutzung von Netfilter konnte gleichzeitig das Regelwerk der Security Policy von IPSec durch IP-Tables einfach realisiert werden.
Um eventuell vorhandene Parallelität durch Mehrprozessorsysteme oder Verschlüssungshardware besser nutzen zu können, wurde bei der Implementierung ein mehrfädiger Ansatz gewählt. Dadurch wird eine Entkoppelung der verschiedenen Phasen der IPSec-Verarbeitung hergestellt.
Als eine Beispielimplementierung einer Softwareverschlüsselung wurde die International Crypto API benutzt. Diese stellt alle notwendigen Verschlüsselungs- und Prüfsummenverfahren zur Verfügung, die für eine IPSec-Implementierung notwendig sind.

1 Grundlagen

1.1 IPSec Grundlagen

IPSec steht für *IP Security* und wurde entworfen, um den sicheren Transport von Daten durch ein unsicheres Netzwerk zu ermöglichen.
IPSec erweitert das IP-Protokoll um Sicherheitsaspekte. Es enthält Funktionalität zur Wahrung der Vertraulichkeit von Paket- und Verbindungsdaten, Schutz gegen Veränderungen von IP-Paketen sowie Mechanismen zur Erkennung und Abwehr von Replay-Attacken.

1.1.1 AH und ESP

Die Erweiterungen der IP-Implementierung durch IPSec umfasst zwei neuen Protokolle: Das AH-Protokoll und das ESP-Protokoll. AH steht für *Authentication Header* und stellt die Authentizität und Integrität der Daten durch die Berechnung einer kryptographischen Prüfsumme über das Paket sicher. ESP bedeutet *Encapsulated Security Payload* und bietet eine Verschlüsselung sowie Authentisierung der Paketdaten.
Das AH Protokoll fügt dem IP-Paket in einem zusätzlichen Kopf eine Prüfsumme hinzu, über die der Empfänger des Paketes feststellen kann, ob die Daten auf dem Weg vom Sender bis zum Ziel mutwillig verändert worden sind.
ESP wird verwendet um die zu versendenden Daten zu verschlüsseln und so für Dritte unlesbar zu machen. Zusätzlich bietet es ebenfalls eine Prüfsumme, um Veränderungen der Paketdaten feststellen zu können. Diese Prüfsumme wird an das Paket angehängt. Wie bei AH wird ein zusätzlicher Protokollkopf eingefügt.

1.1.2 Betriebsarten von IPSec

IPSec kennt zwei Betriebsarten: Tunnel- und Transport-Modus.
Der Tunnel-Modus betrachtet das ursprüngliche IP-Paket als Daten und packt es in ein neues Paket mit einem neuen Paketkopf ein. Die Daten im neu hinzugefügten IP-Kopf können (bzw. müssen, z.B. hat das neu entstandene Paket eine andere Länge) sich von den Werten im Original-Kopf entscheiden.
Der Transport-Modus dagegen baut das ursprüngliche Paket in ein neues Paket um und nutzt dabei schon vorhandene Teile wie z.B. den IP-Kopf für das veränderte Paket. Dabei wird der Original-Kopf geändert.
Beide Betriebsarten sind protokolltransparent: Die durch den Sender vorgenommenen Veränderungen des IP-Verkehrs können beim Empfänger vollständig rückgängig

gemacht werden. Erzeuger und Konsument der Daten bemerken nichts vom Einsatz von IPSec.

1.1.3 Security Associations

Um Schutz gewährleisten zu können, setzen sowohl das ESP- als auch das AH-Protokoll Schlüsselmaterial ein. Dieses Schlüsselmaterial bildet zusammen mit der Kennung des zu verwendenden Verfahrens und den Kennungen der Endpunkte der Kommunikation eine so genannte *Security Association (SA)*. Diese enthält außer den schon genannten Informationen auch Informationen darüber, wie lange ein Schlüssel gültig ist (eine *Lifetime*), und welches Protokoll (AH oder ESP) zu verwenden ist. Eine SA enthält also alle Informationen, die nötig sind, um für ein bestimmtes Paket eine Verschlüsselung oder Authentisierung durchzuführen.
Eine einzelne SA dient immer nur der Verschlüsselung für Pakete einer "Art" für einen bestimmten Empfänger. Die Klassifizierung der "Art" der Pakete wird durch die Einträge in der *Security Policy Database* (s. 1.1.4) festgelegt. Für verschiedene Empfänger werden verschiedene SAs benötigt. Für bidirektionale Kommunikation sind also zwei SAs nötig.
Sollen auf ein Paket mehrere Verfahren kaskadierend angewendet werden, so ist für jede Verarbeitungsstufe eine eigenen SA nötig. Der dadurch gebildete Vektor von SAs bildet ein SA-Bundle.

1.1.4 Die Security Policy Database (SPD)

Die *Security Policy Database (SPD)* ist die Gesamtheit aller Regeln, welche die Verarbeitung von Paketen durch IPSec bestimmen.
So muss für jedes Paket festgestellt werden, ob es durch IPSec verarbeitet (*process*), ohne Anwendung von IPSec weiterverarbeitet (*bypass*) oder sofort verworfen werden soll (*deny*).
Beim Senden von Paketen durch IPSec müssen aus der SPD die Selektoren für die Folge von SAs (SA-Bundle) entnommen werden, mit denen die Verschlüsselung bzw. Signierung durchgeführt werden soll.
Beim Empfangen von Paketen durch IPSec muss nach der Entschlüsselung bzw. Authentisierung wiederum in der SPD überprüft werden, ob es sich um eine gültige Folge von SAs handelt, die für die Verarbeitung des Paketes genutzt worden sind.

1.1.5 Die Security Association Database (SAD)

Die *Security Association Database (SAD)* ist die Datenbasis aller momentan verfügbarer SAs. Sie übernimmt die Verwaltung und Speicherung der SAs im Kern.
Die Schnittstelle zur IPSec-Implementierung im Kern ermöglicht es, die zum Paket passende SA oder das passende SA-Bundle auszuwählen. Die SAD kennt jedoch nur SAs, keine SA-Bundle. Ein SA-Bundle muss daher durch mehrere Anfragen in die SAD zusammengestellt werden. Als Schlüssel zum Zugriff auf die SAD dienen dabei das benutzte IPSec-Protokoll (AH oder ESP), die Zieladresse und der *Security Parameter Index (SPI)*. Der SPI dient dabei beim Senden zur Unterscheidung mehrerer SAs, deren andere Selektoren keine Unterscheidung zulassen. Er wird in den jeweiligen Protokollkopf (AH oder ESP) eingefügt und ermöglicht es beim Empfänger die SA wiederzufinden, die zur Entschlüsselung benutzt wird. Das Tripel <`Protokoll, Zieladresse, SPI`> beschreibt eine SA in der SAD immer eindeutig.
Intern muss die SAD die Speicherung der SAs übernehmen und den Lebenszyklus der einzelnen SAs überwachen.

1.1.6 Verschlüsselungsimplementierung

In der Verschlüsselungsimplementierung findet die eigentliche Verarbeitung der Pakete statt. Dies beinhaltet die Berechnung von Prüfsummen, Ent- oder Verschlüsselung, Hinzufügen oder Entfernen von Protokollköpfen und Überprüfung der Replay-Zähler.

1.1.7 PF_KEY API

Die PF_KEY API ist eine Schnittstelle zur Kommunikation zwischen der SAD im Kern und den Schlüsselaustauschdiensten im Benutzer-Raum. Die für die Implementierung verwendete Version 2 ist in [McDo98] spezifiziert.
Die Schnittstelle definiert eine Kommunikation über Nachrichten. Über die PF_KEY API kann ein Benutzerprogramm eine SA setzen, verändern, oder Informationen über SAs in Erfahrung bringen. Der Kern kann über PF_KEY API die Erzeugung neuer Schlüssel veranlassen oder den Anwendungen mitteilen, dass eine SA demnächst oder sofort ausläuft.

1.2 Linux-Grundlagen

In diesem Abschnitt werden kurz die wichtigsten Kernschnittstellen und Erweiterungen vorgestellt, die bei Entwurf und Implementierung berücksichtigt bzw. verwendet wurden.

1.2.1 Netfilter

Netfilter ist seit der Version 2.4.0 fester Bestandteil des Linux–Kernels und stellt einen Rahmenwerk für die Paketbehandlung dar. Dadurch ist es möglich, in die Paketverarbeitung innerhalb des Protokollstacks einzugreifen.
Dazu definiert Netfilter für jedes Protokoll (z.B. IPv4, IPv6, DECNet) fünf so genannte Hooks (in Abb: 1, grau unterlegt) welche festgelegte Punkte auf der Reise eines Paketes durch den Protokollstack darstellen. Zusätzlich stellt Netfilter Funktionen bereit, die es beliebigen Teilen des Kerns

ermöglichen, sich an diesen Hooks zu registrieren und in die Paketverarbeitung einzugreifen.
Hat man eine entsprechende Funktion an einem der Hooks registriert und passiert ein Paket im Protokollstack diesen Hook, so werden alle an diesen Hook registrierten Funktionen nacheinander mit dem passierenden Paket als Parameter aufgerufen und können eine Bearbeitung des Paketes vornehmen.
Nach der Bearbeitung des Paketes durch eine Funktion muss das weitere Vorgehen durch Netfilter durch die Funktion signalisiert werden. Möglich sind dabei eine Weiterverarbeitung des Paketes (`NF_ACCEPT`), das sofortige Verwerfen (`NF_DROP`) oder das zeitweilige Entfernen aus dem Protokollstack (`NF_STOLEN`).

1.2.2 IP-Tables

IP-Tables ist eine Erweiterung des Netfilter Rahmenwerkes und ermöglicht eine tabellen- und regelbasierte Verarbeitung von IP-Paketen. Zusätzlich zur bisherigen Funktionalität von Netfilter können nun komplexe Regelwerke im Kern abgebildet werden.
Im allgemeinen besteht eine Regel aus einer Menge von Match-Funktionen und einer Target-Funktion. Die Match-Funktionen testen einfache Parameter oder protokollspezifische Eigenschaften des zu prüfenden Paketes und die Target-Funktion beschreibt die auszuführende Aktion.
Mögliche Aktionen sind Akzeptieren (*accept*), Verwerfen (*deny*), Zurückweisen (*reject*). Für eigene Vorhaben ist es möglich, zusätzliche Tabellen im Kern zu erzeugen, außerdem können leicht eigene Match- und Target-Funktionen registriert und verwendet werden.

1.2.3 International Crypto API

Die Crypto API ist eine Bibliothek, die Verschlüsselungs- und Prüfsummenverfahren für die Benutzung im Linux-Kern bereitstellt. Die bereitgestellten Verschlüsselungsverfahren und Prüfsummenverfahren sind jeweils über eine einheitliche Schnittstelle ansprechbar.
Die Bibliothek ist modular aufgebaut. Die Schnittstellen-Funktionalität wird als Modul in den Kern eingebracht. Jedes Prüfsummen- oder Verschlüsselungsverfahren wird in einem eigenen Modul implementiert. Implementiert sind Verschlüsselungs-Module für AES, Blowfish, DES, Triple DES, DFC, IDEA, MARS, RC5, RC6, SERPENT, TWOFISH und eine NULL-Verschlüsselung sowie Prüfsummen-Module für MD5 und SHA1. Damit sind alle Verfahren vorhanden, die für IPSec als verpflichtend vorgesehen sind (s. [Kent98b, Abschnitt 5], [Kent98a, Abschnitt 5])
Die International Crypto API bietet eine eigene synchrone Schnittstelle für Verschlüsselungsverfahren und eine für Prüfsummenverfahren. Um ein Verfahren einzusetzen erfragt man von der Schnittstelle mit Hilfe des Namens des Verfahrens ein Verwaltungsobjekt, über das die bereitgestellte Funktionalität aufgerufen werden kann.
Die Bibliothek ist im Internet frei erhältlich, zur Zeit in Version 2.4.7.0, unter `http://sourceforge.net/projects/cryptoapi`.

2 Konzept

2.1 Problemfelder

Für eine IPSec-Lösung wurden drei Problemfelder identifiziert, auf die besonderes Augenmerk gelegt wurde:

- *Einsatz von spezieller Hardware*: Um die rechenintensive Verarbeitung durch IPSec schnell durchzuführen ist der Einsatz von spezieller Hardware sinnvoll. Diese kann in unterschiedlichem Umfang Aufgaben von IPSec übernehmen und verlangt daher Abstraktionen (Schnittstellen) auf verschiedenen Ebenen der IPSec-Implementierung.

- *Nutzung von Parallelität*: Der Netzwerk-Stack in Linux ist blockierend. Ein vom Benutzer zu versendendes IP-Paket kann erst dann verarbeitet werden, wenn die Abarbeitung des vorherigen Paketes abgeschlossen ist. Dies ist ein sinnvoller Ansatz, so lange die Abarbeitung der Pakete schnell und in einer gleichbleibenden Zeit gewährleistet werden kann. Da die Verschlüsselung rechenintensiv ist und zudem nur ein Teil der Pakete eine Verarbeitung durch IPSec benötigen, sind beide Voraussetzungen nicht mehr gegeben.

  Dadurch entsteht eine Reihe von Problemen:

  1. Während ein zu versendendes Paket durch IPSec bearbeitet wird müssen nachfolgende Pakete, die keine Verarbeitung durch IPSec benötigen, warten, obwohl sie schon verschickt werden könnten.
  2. Sind im System mehrere Funktionseinheiten vorhanden, die parallel Pakete verarbeiten können, tritt ein ähnliches Problem auf: Eine Einheit ist mit der Berechnung einer Prüfsumme oder einer Verschlüsselung beschäftigt. Währenddessen wartet ein zweites Paket auf die Verarbeitung durch IPSec. Diese Verarbeitung kann nicht angestoßen werden, da der Netzwerkstack auf die Fertigstellung des ersten Paketes wartet.

- *Stabilität bezüglich Änderungen und Erweiterungen* Als weiteres Problem wurde erkannt, dass die Netzwerkimplementierung von Linux immer wieder geändert wird. Damit muss eine Lösung wie FreeS/WAN immer wieder an die neuen Gegebenheiten im Linux Kern angepasst werden.

2.2 Lösungsansatz: Entkoppelung und Modularität

Als Lösungsansatz wurde deshalb einen mehrfach entkoppelte Lösung gewählt. Einzelne Aufgaben der Implementierung werden jeweils von einem eigenen Thread übernommen. Die Kommunikation zwischen den Threads wird über Warteschlangen realisiert.
Dadurch werden zwei Dinge erreicht: Eine Entkoppelung der verschiedenen Verarbeitungsstufen und damit das Ermöglichen von Parallelität, und eine starke Modularisierung der einzelnen Teile. Hardwareunterstützung kann einfach integriert werden, indem unterschiedlich große Untermengen von Aufgaben durch die Hardware durchgeführt werden.
Um eine Loslösung von der internen Struktur des Linux-Netzwerkstacks zu erreichen, wurde als Schnittstelle das *Netfilter*-Rahmenwerk verwendet (siehe 1.2.1). Diese Schnittstelle ist Grundbestandteil des Kerns und ermöglicht einen relativ einfachen Eingriff in die Paketverarbeitung.
Um eine Adaption neuer kryptographischer Verfahren zu erleichtern und dem Grundsatz der konsequenten Wiederverwendung schon bestehender Linux-Lösungen zu verwirklichen, wurde als kryptographische Schnittstelle die *International Crypto API* verwendet. Diese wurde in Abschnitt 1.2.3 vorgestellt.

2.3 Probleme des Lösungsansatzes

Der gewählte Lösungsansatz bringt auch Probleme mit sich. Durch die Entkoppelung lockert sich der zeitliche Zusammenhang von Paketen. Dies kann Probleme auf höheren Protokollschichten verursachen.
Ein weiteres Problem tritt auf, wenn zu viele Pakete auf ihre Abarbeitung warten. In diesem Fall kommt die Verarbeitung zu spät – das Paket wird verworfen. In diesem Fall sollten neue Pakete gar nicht erst in die Warteschlangen eingefügt, sondern weiter unten abgeblockt werden. In der Implementierung der Netzwerkfunktionalität ohne IPSec erledigt dies die Netzwerkkarte, sobald sie zu viele Pakete nicht mehr los wird. In einem entkoppelten System muss eine künstliche Blockade in das System eingefügt werden.
Zusätzlich verursacht die Entkoppelung mit mehreren Threads natürlich auch Geschwindigkeitsverluste. Diese entstehen durch die zusätzlichen Kontextwechsel.

2.4 Datenaustausch zwischen Modulen

Der Datenaustausch zwischen den einzelnen Einheiten in der kryptographischen Verarbeitung sowie deren Einbindung in die umgebende IPSec-Implementierung ist über *Paketwarteschlangen* realisiert. Paketwarteschlangen sind einfach verkettete Listen mit einem Kopf- und einem Endzeiger.
Soll ein Paket von einer aktiven Einheit an die nächste weitergeleitet werden, so hängt der Produzent das Paket an das Ende der Liste an. Dieses neue Paket wird dem Konsumenten über ein Ereignis signalisiert; ist der Thread des Konsumenten gerade nicht aktiv, so wird er durch dieses Ereignis aufgeweckt.

2.5 Anbindung an den Linux-Netzwerkstack

Aus dem entwickelten Lösungsansatz wurde ein Design entwickelt, das wie in Abb. 1 dargestellt in den Linux-Netzwerkstack integriert wird.
Die Integration in den IP-Netzwerkstack erfolgt ausschließlich über die fünf bestehenden Netfilter-Hooks. Dort können die eingehenden und ausgehenden Pakete der Paketverarbeitung entnommen werden, wenn sie einer IPSec-Verarbeitung unterzogen werden sollen.
Wie man der Abbildung entnehmen kann, wird an zwei Hooks (`PRE_ROUTING, LOCAL_IN`) die Empfangsfunktion (`IPSec_recv`) und an drei Hooks (`PRE_ROUTING, LOCAL_OUT, POST_ROUTING`) die Sendefunktion (`IPSec_send`) registriert. Diese mehrfache Registrierung von Empfangs- und Sendefunktion ist notwendig, um die verschiedenen Anforderungen an eine IPSec-Implementierung zu erfüllen.
So ist es beispielsweise notwendig, die jeweiligen Funktionen am Hook `PRE_ROUTING` zu registrieren, weil es möglich sein muss, einkommende Pakete aus einem Tunnel sofort wieder in einen neuen Tunnel zu verpacken. Da für die zwischenzeitlich ausgepackten Pakete keine Route auf dem Rechner existieren muss, muss die Eintunnelung vor dem Routing passieren, also im gleichen Hook.
Zusätzlich muss die Sendefunktion am Hook `POST_ROUTING` registriert werden, um mögliche netzwerkgerätespezifische SAs zu realisieren. Dabei muss die IPSec-Verarbeitung nach dem Routing erfolgen, da sonst unter Umständen nicht feststeht, über welches Gerät das Paket den Rechner verlässt.
Die Empfangs- und Sendefunktionen an den lokalen Hooks (`LOCAL_IN`, `LOCAL_OUT`) dienen dem Empfang und Senden lokal zu verarbeitender oder erzeugter Pakete.

2.6 IPSec-Paketverarbeitung

Wird eine der registrierten IPSec-Funktionen (Empfangs- oder Sendefunktion) durch Netfilter aufgerufen, so muss zuerst überprüft werden, ob durch sie eine IPSec-Verarbeitung für das aktuelle Paket durchgeführt werden soll.
Die Überprüfung beinhaltet in der Empfangsfunktion unter anderem eine Kontrolle des IP-Protokolles (`IPPROTO_AH, IPPROTO_ESP`), in der Sendefunktion einen Aufruf in die SPD. Ist die jeweilige Überprüfung positiv, wird die eigentliche IPSec-Verarbeitung durchgeführt.

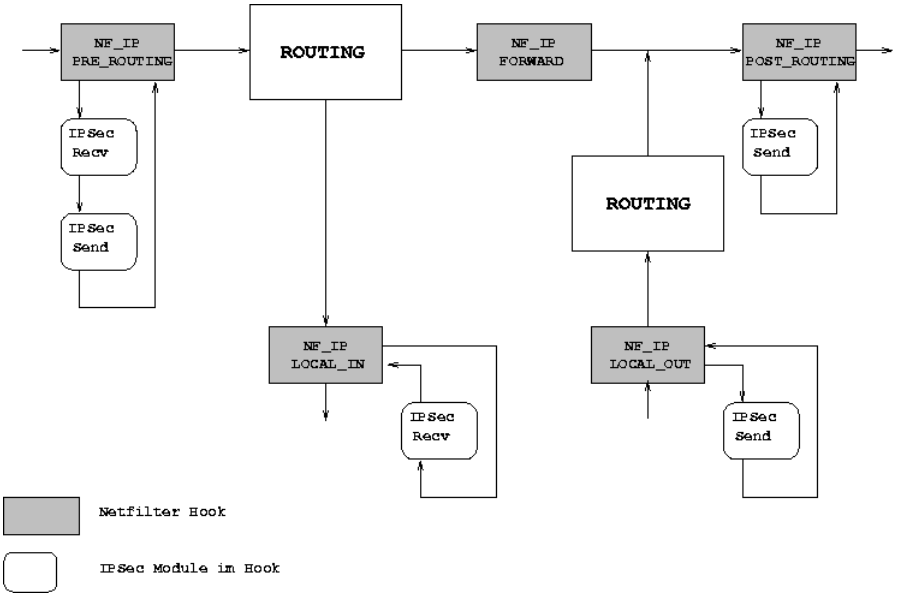

Abbildung 1: Anbindung an den Linux IP-Netzwerkstack

Die darauf folgende IPSec-Verarbeitung ist in der Abb. 2 schematisch dargestellt. Da diese sich für das Empfangen und Senden von Paketen sehr ähnelt, sind beide Darstellungen in der Abbildung zusammengefasst. Zur Unterscheidung sind die empfangs- und sende-spezifischen Teile durch einen extra Rahmen mit der Angabe (*Empfangen/Senden*) gekennzeichnet.
Im Groben wird die IPSec-Verarbeitung eines Paketes in folgende Abschnitte unterteilt.

- Annahme des Paketes am Netfilter Hook
  In diesem Teil der Verarbeitung wird das Paket angenommen und in der Eingangswarteschlange abgelegt. Danach wird das Paket aus dem Netzwerkstack entfernt (`NF_STOLEN`).

  Im Unterschied zum Empfangen wird beim Senden von Paketen an dieser Stelle zusätzlich der sendespezifische Teil in die Verarbeitungabfolge aufgenommen. Dies bedeutet, dass für das Paket eine Anfrage in der SPD durchgeführt wird, um festzustellen, ob die IPSec-Verarbeitung weitergeführt werden soll.

- Paket-Scheduler
  Dieser zentrale Teil entnimmt ein Paket der Eingangswarteschlange und führt die eigentliche Verarbeitung des Paketes durch. Dazu legt der Paket-Scheduler fest, wie weiter mit dem Paket zu verfahren ist (Ent-/Verschlüsseln, Verwerfen, Beenden).

  Soll eine kryptographische Operation durchgeführt werden, so werden zuerst die benötigte Daten (SA) aus der SAD abgefragt. Danach wird das Paket über eine Warteschlange an das Modul `Kryptographische Verarbeitung` übergeben. Das Paket wird nach der Bearbeitung über eine andere Warteschlange an den `Paket Scheduler` zurückgeliefert. Dieser legt danach wieder die weitere Verarbeitung fest.

  Ist die Verarbeitung des Paketes abgeschlossen, wird das Paket in der Ausgangswarteschlange abgelegt.

- Kryptographische Verarbeitung
  Dieser Teil führt alle notwendigen kryptographischen Operationen durch. Dazu wird das jeweilige Paket aus der Eingangswarteschlange entnommen, die gewünschte Operation wird durchgeführt und das Paket in die Ausgangswarteschlange dieses Modul eingefügt.

- Rückgabe des Paketes an Netfilter
  Im letzten Teil (`Netfilter - reinject`) wird das Paket aus der Ausgangswarteschlange entnommen und wieder in den Netzwerkstack eingefügt. Damit ist die Verarbeitung des Paketes durch IPSec abgeschlossen.

  Im Unterschied zum Senden wird beim Empfangen von Paketen an dieser Stelle zusätzlich der empfangsspezifische Teil in die Verarbeitungabfolge aufgenommen. Dies bedeutet, dass für das Paket eine Anfrage in der SPD durchgeführt wird, um festzustellen, ob das verarbeitete Paket angenommen werden soll.

Eine detaillierte Übersicht über das Modul `Kryptographische Verarbeitung` wird in Abb. 3 gegeben, eine ausführliche Beschreibung erfolgt in Abschnitt 2.7.

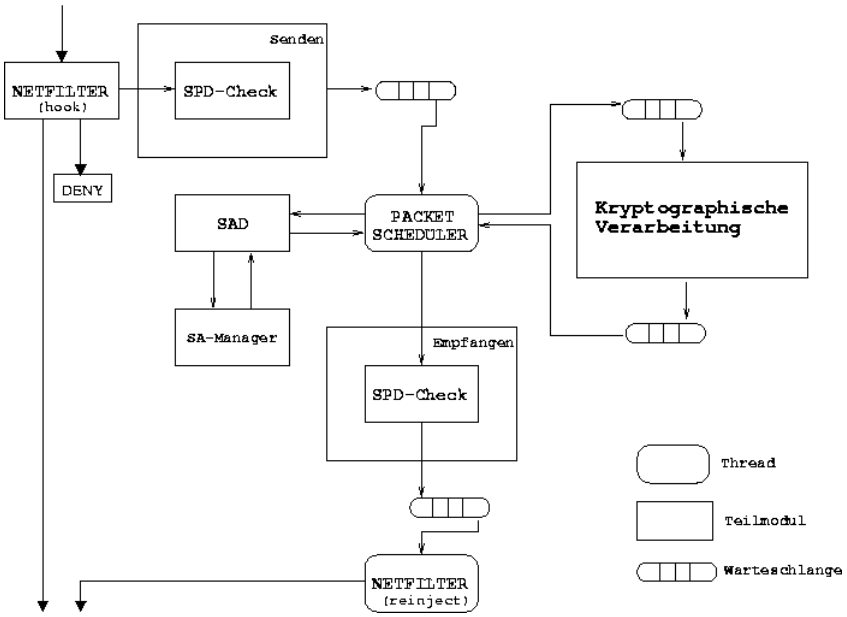

Abbildung 2: IPSec Paketverarbeitung

2.6.1 SPD-Überprüfung bei zu sendenden Paketen

Der spezifische Teil für das Senden von IPSec-Paketen befindet sich weit am Anfang der IPSec-Verarbeitung (siehe Abb. 2, Rahmen *Senden*). Dies ist deshalb notwendig, da für jedes zu versendende Paket bestimmt werden muss, ob eine IPSec-Verarbeitung durchgeführt werden soll oder nicht.
Die Überprüfung erfolgt durch das Suchen einer passenden Regel (Policy) in der SPD. Existiert in der SPD eine Regel, der das Paket entspricht, und legt diese eine IPSec-Verarbeitung fest, so kann aus der Regel die Folge von SAs bestimmt werden, die auf das Paket nacheinander angewendet werden sollen. Mit dieser Menge von SAs (SA-Bundle) als Eingabe wird die eigentliche Verarbeitung des Paketes durch den Paket-Scheduler veranlasst.
Die SPD selbst ist jeweils als separate Tabelle für IP-Tables realisiert. Die Regeln enthalten neben anderen Daten (Quelladresse, Zieladresse) auch die Folge von SAs, die bei der IPSec-Verarbeitung angewendet werden soll.

2.6.2 SPD-Überprüfung bei empfangenen Paketen

Der spezifische Teil für das Empfangen von IPSec-Paketen befindet sich weit am Schluss der eigentlichen IPSec-Verarbeitung (siehe Abb. 2, Rahmen *Empfangen*). Dies kommt daher, da erst nach der kompletten Entschlüsselung bzw. Prüfsummenbildung überprüft werden kann, ob es sich um ein gültiges Paket handelt.
Im Netfilter Hook werden nur Pakete mit der IP-Protokollkennung `IPPROTO_AH` oder `IPPROTO_ESP` aus dem Netzwerkstack entnommen und über die Warteschlange an den Paket-Scheduler übergeben. Dieser ermittelt aus dem Paket die notwendigen Daten, um aus der SAD die entsprechende SA für die Entschlüsselung bzw. Prüfsummenbildung abzufragen. Danach wird das Paket mit der jeweiligen SA kryptographisch behandelt. War die kryptographische Operation erfolgreich, wird die verwendete SA gespeichert und somit eine Folge von verwendeten SA aufgebaut.
Ist die IPSec-Verarbeitung durch den Paket-Scheduler abgeschlossen, muss nun anhand der SPD überprüft werden, ob dieses Paket mit einer gültigen Folge von SAs bearbeitet wurde und damit angenommen werden kann (`NF_ACCEPT`). Gibt es keine passende Regel zu einem Paket in der SPD, so muss das Paket sofort verworfen werden (`NF_DROP`).
Auch hier ist die SPD wieder als eine separate IP-Tables Tabelle realisiert. Die Regeln enthalten neben anderen Daten (Quelladresse) auch die Folge von SAs, die bei der IPSec-Verarbeitung verwendet werden musste. Diese muss der tatsächlich verwendeten Folge von SAs entsprechen.

2.7 Module der kryptographische Verarbeitung

Der konzeptuelle Aufbau der kryptographischen Verarbeitung ist in Abb. 3 dargestellt.
Die kryptographische Verarbeitung besteht aus *aktiven* und *passiven* Einheiten. Aktive Einheiten sind eigene Threads, die Daten selbsttätig verarbeiten können. Passive Einheiten sind Module, die durch aktive Einheiten benutzt werden, jedoch keine eigene Verarbeitung anstoßen.
Die Implementierung der kryptographischen Verarbeitung benutzt mehrere aktive Einheiten: einen Scheduler und ein oder mehrere kryptographische Einheiten (s. 2.7.1). Passi-

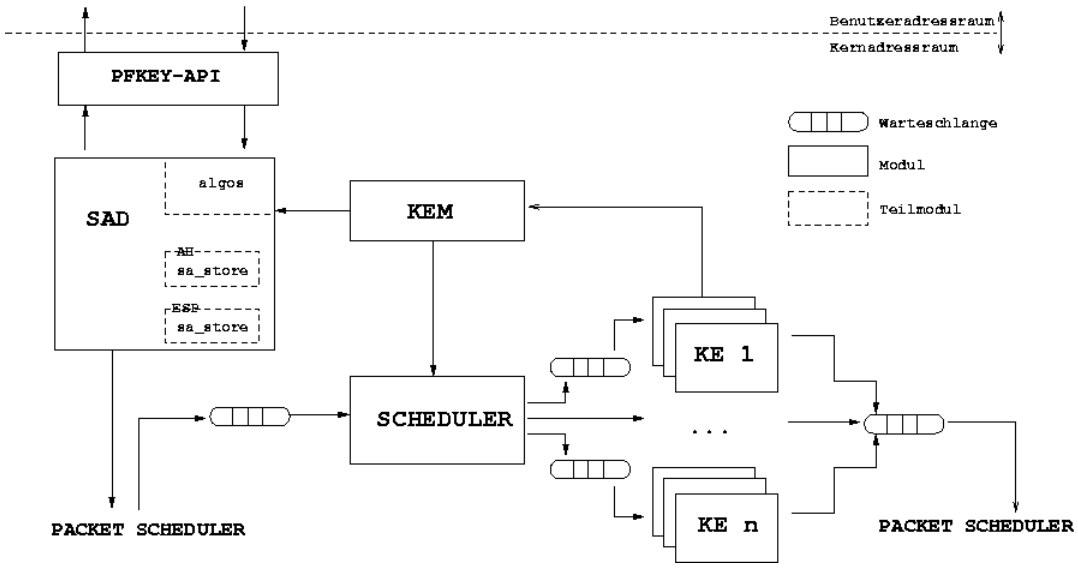

Abbildung 3: Module der kryptographische Verarbeitung

ve Einheiten sind die SAD, die PF_KEY API und die verschiedenen Paketschlangen.

2.7.1 Kryptographische Einheiten

Eine *Kryptographische Einheit (KE)* führt den Zusammenbau und das Zerteilen von IPSec-Paketen durch. Dazu gehört auch die die Verschlüsselung der Paketdaten oder die Bildung einer Prüfsumme. Alle kryptographischen Einheiten müssen in der Lage sein, das ESP und das AH-Protokoll sowohl im Tunnel- als auch im Transportmodus durchzuführen. Beherrscht eine kryptographische Einheit nur Prüfsummenverfahren, so besitzt es nur Unterstützung für AH und für ESP mit NULL-Verschlüsselung. Beherrscht sie nur Verschlüsselungsverfahren, so wird nur Unterstützung für AH mit NULL-Prüfsumme und für ESP mit NULL-Prüfsumme angeboten.

Eine kryptographische Einheit erhält vom Restsystem nur Pakete, deren Verschlüsselung und/oder Prüfsummenbildung es auch ausführen kann. Welche Verfahren dies sind teilt die kryptographische Einheit bei der Anmeldung bei IPSec mit.

Die Einheiten sind ebenfalls als nachladbare Module realisiert.

Für die Beispielimplementierung wurde eine kryptographisch Einheit auf Basis des International Crypto Patch realisiert.

Die kryptographischen Einheiten sind an den Scheduler für kryptographische Einheiten angebunden, der sie mit Paketen versorgt. Die kryptographischen Einheiten sind in Abb. 3 als `KE 1` bis `KE n` dargestellt.

2.7.2 Verwaltung kryptographischer Einheiten

Da kryptographische Einheiten währen der Laufzeit des System hinzugefügt oder entfernt werden können gibt es eine Verwaltungsinstanz, die die Verwaltung der kryptographischen Einheiten übernimmt (*Manager für Kryptographische Einheiten (KEM)*). Diese bietet eine Schnittstelle, an der sich kryptographische Einheiten an- oder abmelden können.
Meldet sich eine kryptographische Einheit am System an, so wird sie in eine Hashtabelle eingetragen. Als Schlüssel wird das unterstützte Verfahren verwendet. Beherrscht eine kryptographische Einheit mehrere Verfahren, so wird für jedes Verfahren ein Eintrag in der Tabelle erzeugt.
Meldet sich eine kryptographische Einheit vom System ab, so werden alle Einträge aus der Tabelle entfernt, die auf diese Einheit verweisen.
Bei An- oder Abmeldung wird zusätzlich die SAD über die von dem Modul unterstützten Verfahren unterrichtet, damit diese nun auch Schlüssel für diese Verfahren annimmt und verwaltet.

2.7.3 Scheduler für kryptographische Einheiten

Der Scheduler für kryptographische Einheiten (in Abb. 3 als `Scheduler` gekennzeichnet) übernimmt die Verteilung der durch kryptographische Verfahren zu bearbeitende Pakete. Der Scheduler sucht für jedes Paket eine Verarbeitungseinheit aus, die in der Lage ist sowohl die für das Paket angeforderte Verschlüsselung als auch die Prüfsummenbildung durchzuführen.
Der Scheduler wird als nachladbares Kern-Modul integriert. Für die beschriebene Implementierung wurde ein einfacher Scheduler realisiert, der die Pakete immer an die am längsten nicht benutzte Verschlüsselungseinheit abgibt.

2.7.4 SAD

Die SAD besteht im wesentlichen aus drei Teilen: Jeweils ein Kontainer für AH- und einem Kontainer für ESP-SAs (in Abb.3 als `sa_store` referenziert). Des Weiteren ist ein Objekt vorhanden, das festlegt, welche Verfahren von der IPSec- Implementierung unterstützt werden. Dieses Objekt implementiert eine Schnittstelle über die ihm angezeigt werden kann, wenn ein Verfahren neu oder nicht mehr unterstützt wird. In Abbildung 3 ist dieses Objekt mit `algos` betitelt.

Als Anbindung zum Benutzeradressraum wurde die PF_KEY API verwendet. Sie besteht aus zwei Teilaspekten:

- Der Realisierung der PF_KEY-Schnittstelle, Version 2. Dies umschließt die Teilaspekte des Zusammenbaus der Nachrichten aus den Nutzdaten, das Zerteilen der einkommenden Nachrichten und das Extrahieren der Nutzdaten aus den Nachrichten. Hierfür wurden Teile der BSD- und der FreeS/WAN -Implementierung genutzt.

- Die Kommunikation mit dem Benutzerraum: Versenden und Empfangen der Nachrichten an der Socket-Schnittstelle und die Bereitstellung von Puffern für diese Kommunikation.

Wird eine SA aus der IPSec-Implementierung angefragt, so wird die SAD mit dem Selektor aufgefordert, die entsprechende SA zu suchen. Ist die SA schon vorhanden, so wird sie mit erhöhtem Referenzzähler zurückgegeben. Das Paket wird normal weiterverarbeitet.
Da die SA aber jederzeit ungültig werden kann, z.B. weil ihre Lebensdauer abgelaufen ist, muss vor dem Versenden noch getestet werden, ob die SA noch gültig ist. Deshalb findet hier noch einmal eine Anfrage in die SAD statt. Ist die SA nicht mehr gültig, so wird das Paket verworfen.
Ist die SA noch nicht vorhanden, so wird von der SAD über die PF_KEY API eine SA angefordert, was einen Schlüsselaustausch anstößt.

2.7.5 Konzeption einer kryptographischen Einheit basierend auf der Crypto API

Als Referenzimplementierung für eine kryptographische Einheit wurde eine Implementierung auf Basis der Crypto API erstellt. Diese besteht aus zwei Teilen:

1. Einem Modul, das den Zusammenbau eines IPSec-Paketes und das Wiederherstellen des Original-Paketes aus einem IPSec-Paket übernimmt.

2. Einem Modul, das die eigentliche Verschlüsselung der Daten sowie die Bildung von Prüfsummen übernimmt.

Alle Softwareimplementierungen von Verschlüsselungseinheiten benötigen bestimmte Funktionalität um IPSec-Pakete aus IP-Paketen zu erzeugen bzw. um aus empfangenen IPSec-Paketen die Original-IP-Pakete zu restaurieren.
Dieser Teil ist generisch. Für die Durchführung der Verschlüsselung und der Prüfsummenbildung greift dieses über eine definierte Schnittstelle auf ein Verschlüsselungsobjekt zu, das dem Modul mit übergeben wurde.
Diese Schnittstelle stellt eine Methode zur Verschlüsselung eines Datenblockes und eine Methode zur Bildung einer Prüfsumme über einen bestimmten Block zur Verfügung. Um die Behandlung und den Zusammenbau der Pakete zu erleichtern, können an dieser Schnittstelle auch Parameter erfragt werden, die vom verwendeten Verfahren abhängen.
Dieses Modul wurde zwar im Rahmen einer kryptographischen Einheit mit der Crypto API erstellt, ist aber nicht von dieser abhängig. Genauso gut könnte über diese Schnittstelle eine andere Bibliothek kryptographischer Verfahren oder ein selbst erstellte Implementierung angebunden werden.
Um eine Verschlüsselung oder Prüfsummenbildung mit der Crypto API durchzuführen muss das Crypto API Modul nur noch wenig Funktionalität realisieren. Es implementiert die Schnittstelle zum Scheduler und meldet sich mit dieser am System an (s. 2.7.2). Es bindet das beschriebene generische Modul ein. Mit Hilfe der Crypto API realisiert es ein Verschlüsselungsobjekt, das an das generische Modul übergeben wird, und das die Verschlüsselung und / oder Prüfsummenbildung durchführt. Die Funktionalität dieses Objekts reduziert sich darauf, die von IPSec in [Kr97] beschriebenen Anforderungen an eine Prüfsummenbildung durchzusetzen und die Daten an die Crypto API weiterzugeben.

3 Ausblick

Mit der vorgestellten Lösung wurde eine IPSec-Implementierung realisiert, die durch ihre Modularität neue Einsatzgebiete für IPSec eröffnet.
So kann durch die Implementierung neuer kryptographischer Einheiten einfach neue Hardware ins System eingebracht werden. Neue Verfahren können schnell in das System integriert werden, ohne tiefgreifend in die eigentliche IPSec-Implementierung eingreifen zu müssen.
Auf Basis dieser Implementierung können IPSec-Lösungen entwickelt werden, die Anforderungen von Multicast oder QoS erfüllen. Dazu muß ein kryptographischer Scheduler entwickelt werden, der mit Hilfe spezielle kryptographischer Einheiten bestimmte Dienstgütemerkmale durchsetzt. Ebenfalls durch Realisierung neuer Scheduler können Strategien entwickelt und getestet werden, wie Hardware möglichst effizient ausgenutzt werden kann.
Damit steht eine IPSec-Lösung zur Verfügung, die Sicherheit auf IP-Ebene zu mehr verhelfen kann, als zur Realisierung eines einfachen Punkt-zu-Punkt VPN und den

Einsatz von IPSec in verschiedensten Kontexten fördert: Durch einfache Erweiterbarkeit und unter Nutzung bewährter Schnittstellen zum Linux-Kern.

Literatur

[Kent98b] S.Kent, BBN Corp, R.Atkinson. Security Architecture for the Internet Protocol. *RFC 2401*, November 1998.

[Kent98a] R.Atkinson, S.Kent. IP Authentication Header. *RFC 2402*, November 1998.

[Kent98b] R.Atkinson, S.Kent. IP Encapsulated Security Payload. *RFC 2406*, November 1998.

[Kr97] H.Krawczyk, M.Bellare, R.Canetti. HMAC: Keyed-Hashing for Message Authentication. *RFC 2104*, Februar 1997.

[McDo98] D. McDonald, C.Metz, B.Phan. PF_KEY Key Management API, Version 2. *RFC 2367*, July 1998.

[DeCi98] S. Deering, Cisco, R. Hinden, Nokia. Internet Protocol, Version 6 (IPv6) Specification. *RFC 2460*, December 1998.

[WePa02] Wehrle, Pählke, Ritter, Müller, Bechler. Linux Netzwerkarchitektur. Muenchen - Addison Wesley Verlag, 2002. ISBN 3-8237-1509-3.

[BeBo97] Beck, Boehme, Dziadzka, Kunitz, Magnus, Verworner. Linux-Kernel-Programmierung. 4. akt. u. erw. Auflage - Bonn: Addison-Wesley-Longmann, 1997. ISBN 3-8273-1144-6.