VDMFA, eine verteilte dynamische Firewallarchitektur für Multimedia-Dienste

Christoph Rensing[1], Utz Rödig[1], Ralf Ackerman[1], Lars Wolf[1], Ralf Steinmetz[1,2]

1
Technische Universität Darmstadt
Merckstr. 25 • D-64283 Darmstadt

2
GMD IPSI
Dolivostr. 15 • 64293 Darmstadt

Email:{Christoph.Rensing, Utz.Roedig, Ralf.Ackermann, Lars.Wolf, Ralf.Steinmetz}
@KOM.tu-darmstadt.de

Kurzfassung Im Rahmen einer umfassenden Security Policy stellen Firewalls eine wichtige Maßnahme zum Schutz eines privaten Netzes vor Angriffen aus dem Internet dar. Sie basieren in der Regel auf IP-Filtern und Proxies. Filter selektieren an den Netzgrenzen Datenströme nach definierten Regeln, zumeist über TCP- oder UDP-Portnummern, die einen Dienst identifizieren, leiten sie weiter oder blocken sie ab. Die Selektion über bestimmte Portnummern ist bei vielen Protokollen nicht statisch möglich, da diese erst zur Verbindungszeit dynamisch bestimmt werden. Daher kommen - sollen solche Dienste die Firewall passieren - sogenannte Proxies zum Einsatz. Proxies stellen den Endpunkt der Kommunikation zu beiden Seiten (lokales Netz und Internet) dar und leiten die Daten auf Anwendungsebene weiter. Sie müssen für jedes Protokoll der Anwendungsebene neu entwickelt werden. Eine Alternative bildet eine dynamische, vom Protokollstatus abhängige Erweiterung der Regeln einer filterbasierten Firewall während der Verbindungszeit. Bestehende kommerzielle Ansätze realisieren diese Dynamik heute im Kern der Firewall selbst, indem in diesen Kenntnisse über die Semantik der Protokolle auf Anwendungsebene integriert werden. Wünschenswert ist aber eine allgemeinere Architektur, wie die Verteilte Dynamische Multimedia Firewallarchitektur VDMFA, die es erlaubt, einfache filterbasierte Firewalls flexibel für neue, insbesondere multimediale Protokolle zu erweitern. Die Funktionsweise der VDMFA basiert auf einer dynamischen Anpassung von Filterregeln über die intelligente Komponente VDMFA-Core, welche wiederum per Skriptsprache oder ein benutzerfreundliches Front-end gesteuert wird. In diesem Beitrag werden die VDMFA vorgestellt und Einsatzmöglichkeiten der Firewallarchitektur für Internet Telefonie Anwendungen aufgezeigt.

1 Motivation

Durch die Anbindung von privaten Netzen an das globale Internet sind diese zunehmend auch potentiellen Angriffen ausgesetzt. Um ein privates Netz zu schützen, errichtet man sogenannte Firewalls an den Grenzen zum Internet, die nur ausgewählten Datenströmen von und zum privaten Netz den Transfer ermöglichen. Dies geschieht zum einen über IP-Paketfilter, die zumeist statisch definiert werden, und über Proxies, die auf Anwendungsebene arbeiten. Dazu müssen diese weite Teile der Protokolle dieser Ebene implementieren.

Die Entwicklung der im Internet verfügbaren Dienste ist heute durch eine enorme Dynamik gekennzeichnet. Die einzusetzenden Protokolle werden dabei teilweise sehr umfangreich, viele Anwendungen erfordern auch mehrere parallele Netzverbindungen. Der Anteil dynamischer Komponenten (z.B. zur Laufzeit ausgehandelter Ports) nimmt gerade bei komplexen Protokollen und Anwendungen zu. Heute gängige Firewallkon-

zepte sind nur bedingt geeignet, um die Forderung nach einer kurzen Zeitspanne bis zur Unterstützung eines Dienstes zu erfüllen und der Komplexität und Dynamik gerecht zu werden. Wünschenswert ist daher eine allgemeinere Architektur, die es ermöglicht, in bestehende, nicht spezifisch ausgeprägte Firewalls mittels eines einheitlichen Werkzeugs auf systemunabhängiger Ebene schnell die Unterstützung neuer oder veränderter Protokolle zu integrieren. Die VDMFA bietet die Grundlage für eine solche Lösung, dabei werden bestehende Firewalls um zusätzliche Komponenten erweitert und die vorhandenen IP-Filter dynamisch konfiguriert.

Am Beispiel des ITU Standards H.323 für Internet Telefonie werden zunächst die von uns zu betrachtenden typischen Szenarien, Anforderungen und Schwierigkeiten der Behandlung von Datenströmen in einer Firewall vorgestellt. In diesem Rahmen erfolgt eine Beschreibung der grundlegenden Firewallkomponenten, ihrer Arbeitsweise und Interaktion. Anschließend stellen wir in Abschnitt 4 die prinzipielle Architektur und die zusätzlichen Module und die Arbeitsweise der VDMFA vor. Die konkreten Einsatzmöglichkeiten der VDMFA werden im Beispiel der Internet Telefonie nach H.323 im Abschnitt 5 erläutert, bevor zum Abschluß eine kritische Betrachtung der Gefahren und des weiteren Potentials des Ansatzes, sowie eine Einordnung in Bezug auf Arbeiten mit gleicher oder ähnlicher Ausrichtung erfolgt.

2 Verwendung dynamischer zugewiesener Ports bei Internetprotokollen

Internetanwendungen verwenden die Protokolle IP, TCP und UDP, die der Anwendung einen gesicherten oder ungesicherten Datentransport zur Verfügung stellen. Die Adressierung eines Dienstes auf einem Endsystem erfolgt durch die Spezifizierung von TCP oder UDP Ports, die im Paket-Header angegeben werden. Viele, insbesondere neuere Dienste beschränken sich nicht auf die Verwendung nur einer Verbindung. Für die initiale Verbindung einer Session wird in der Regel ein fest definierter Port (well-known-port) verwendet. Die Anzahl der folgenden Verbindungen sowie die für sie verwendeten Ports werden erst während der Nutzung des Dienstes durch die Kommunikationspartner bestimmt. Ein solches Vorgehen ist insbesondere dann oftmals vorgesehen, wenn mehrere Verbindungen für den Nutzdatentransfer getrennt von Steuerungskanälen aufgebaut werden müssen. Typische Beispiele finden sich gerade bei Protokollen zum Transport multimedialer Daten aber auch bei einfachen Anwendungen wie z.B. nach dem File Transfer Protokoll [1].

Exemplarisch soll an dieser Stelle der Verbindungsaufbau für Internet-Telefonie nach dem ITU Standard H.323 erläutert werden, da an ihm in Abschnitt 5 die Funktionsweise der Komponenten der VDMFA genauer beschrieben werden. H.323 in der Version 1 [3] bezeichnet eine Protokollfamilie für multimediale Kommunikation über lokale Netze, die keine Dienstgüte garantieren. Mit der zweiten Version von H.323 [4] erfolgte eine Erweiterung auf alle IP-basierten Netze. H.323 stellt somit neben dem primär vorgesehenen Einsatzfall für lokale Netze einen wichtigen Standard für die Internet-Telefonie dar und wird heute von vielen Produktanbietern zumindest in der Version 1 implementiert.

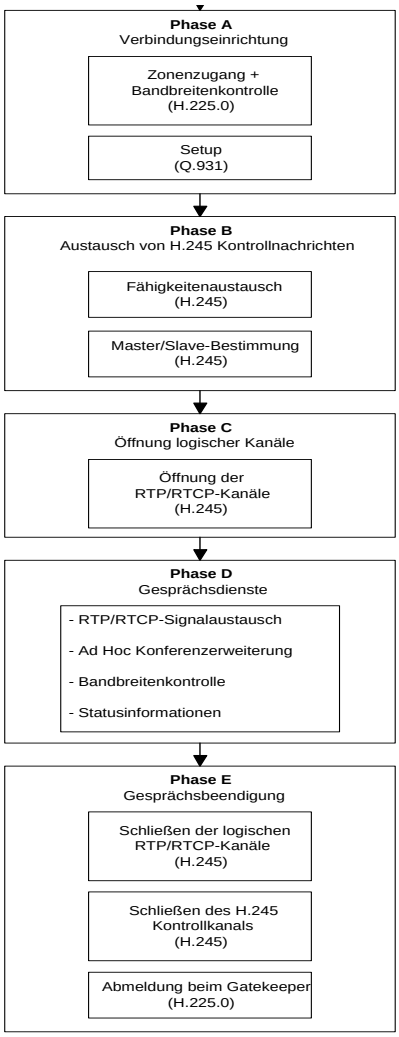

Abbildung 1: Internet-Telefonie nach H.323

Die H.323 Empfehlung beschreibt alle Komponenten eines H.323 Systems und die Kommunikation dieser Komponenten untereinander durch Spezifizikation der Kontrollnachrichten und Prozeduren.

Für Rufaufbau und Nutzdatentransfer sind eine Vielzahl von Protokollen, wie das von der IETF spezifizierte und zum Transport von Echtzeitdaten vorgesehene Realtime Transport Protocol RTP[5], weitere ITU Protokolle wie H.225.0 und H.245 sowie mehrere obligat oder alternativ zu unterstützende Audio- und Videokodierungsverfahren einbezogen.

Eine Internet-Telefonie Sitzung nach dem Standard H.323 verläuft in verschiedenen Phasen, die zusammenfassend in Abbildung 1 dargestellt werden. In dieser Arbeit werden die Phasen A und C genauer betrachtet (vgl. Abbildung 2: Dynamische Ports sind kursiv dargestellt.), da nur diese für das betrachtete Zusammenwirken mit Firewallkomponenten relevant sind.

In der Phase A der Verbindungseinrichtung wird eine TCP-Verbindung zwischen den Endpunkten der Verbindung initiiert. Der Verbindungsaufbau erfolgt dabei stets zum Port 1720.

Über die Verbindung wird eine Q.931 Setup Nachricht an den Gesprächspartner versandt. Der gerufene Partner zeigt den einkommenden Ruf durch ein Alerting an, bestimmt einen dynamischen Port für die nachfolgende H.245 Kommunikation und sendet dessen Nummer in einer Connect Nachricht zurück.

In der zweiten Phase wird eine neue TCP-Verbindung, der sogenannte H.245 Kontrollkanal unter Benutzung des dynamisch bestimmten Ports eröffnet. Über diesen Kanal werden nachfolgend eine Vielzahl von Kontrolldaten ausgetauscht. In Phase C werden über den H.245 Kanal die logischen Kanäle für den Nutzdatentransfer ausgehandelt und aufgebaut. Für diesen verwendet man zwei unidirektionale RTP Kanäle und einen zur Kontrolle dienenden bidirektionalen RTCP Kanal. Die von den Verbindungen zu nutzenden Ports werden wiederum dynamisch ermittelt und ausgetauscht.

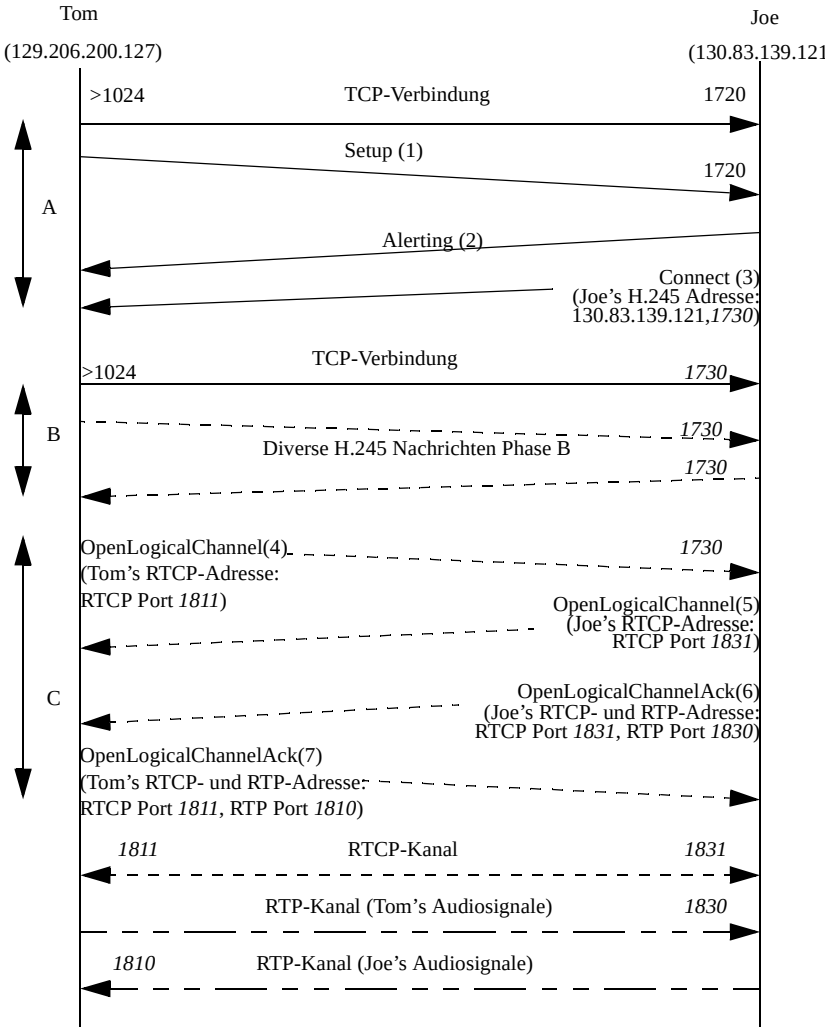

Abbildung 2: Verbindungsaufbau nach H.323

3 IP-Filter und Proxies als wichtigste Firewallkomponenten

Firewalls haben im Rahmen einer umfassenden Security Policy die Aufgabe, ein privates Netz vor unerlaubten Zugriffen aus dem Internet zu schützen, aber auch unerwünschte Zugriffe der Anwender innerhalb des privaten Netzes auf Dienste des Internets zu unterbinden. In diesem Abschnitt werden die wesentlichen Komponenten einer Firewall beschrieben und die Probleme im Zusammenspiel mit den zuvor vorgestellten Szenarien aufgezeigt.

3.1 Statische und dynamische IP-Filter

IP-Filter [2] verwenden ein sehr einfach zu realisierendes, zumeist statisches Verfahren zur Sicherung eines privaten Netzes. Ein IP-Filter untersucht die IP- oder TCP- bzw. UDP-Header eines Datenpaketes und entscheidet anhand der Liste seiner Regeln, ob das betreffende Paket blockiert oder weitergeleitet wird. Damit selektiert er Datenpakete nach - konkreten Bytefolgen an einem speziellen Offset zugeordneten - inhaltlichen Merkmalen wie Protokolltyp (ICMP, TCP, UDP), IP-Quell- und Zieladresse, TCP/ UDP-Quell- und Zielport oder auch ICMP-Nachrichtentyp. Ein einfacher Paketfilter arbeitet nur auf der Transportebene.

Für eine Vielzahl von Protokollen, insbesondere auch für multimediale Anwendungen sind einfache Paketfilter nicht unmittelbar verwendbar, da die Selektionskriterien nicht festzulegen sind. So können z.B. dynamisch zugewiesene Ports in den Filterregeln nicht angegeben werden. Für eine FTP Anfrage aus dem privaten Netz an einen Server im Internet müßten für alle Verbindungen mit Quellport 20 die eingehenden Portnummern > 1023 freigeschaltet werden, da der Server eine Verbindung zu einem dieser Ports aufbaut. Grundsätzlich besitzen statische IP-Filter keine Kenntnisse der Protokolle auf Anwendungsebene und können daher den Status eines Protokolles nicht berücksichtigen.

Eine erste Erweiterung der IP-Filter stellt die Realisierung sogenannter Keep-State Mechanismen dar. Keep-State Mechanismen erweitern nach einmaligem, aufgrund einer statischen Regel erlaubten Weiterleiten eines TCP-Paketes in einer Richtung das Regelwerk der Firewall dynamisch in der Weise, daß auch nachfolgende TCP-Pakete und in Gegenrichtung versandte Pakete, welche die identischen Ports und IP-Adressen verwenden, bis zum Abbau der TCP-Verbindung erlaubt werden. Somit kann die TCP-Verbindung durchgängig ermöglicht werden, ohne daß dazu manuell eine Regel erzeugt werden muß.

Um die Problematik der nicht vorhersehbaren Bestimmung und Nutzung von Ports zu lösen, gibt es aktuell weitergehende Anstrengungen, IP-Filter um eine sogenannte dynamische Komponente zu erweitern [6]. Diese führt eine protokollspezifische Analyse des Datenanteils eines TCP-Paketes durch. Gelingt es, die Daten zu interpretieren und die ausgehandelten Kommunikationsparameter zu bestimmen, dann können dynamisch neue Regeln erzeugt werden. Dieses Verfahren wird auch Stateful Inspection Firewall Technology [7] oder Stateful Filtering Firewall [8] genannt. Zur Realisierung der dynamischen Filter ist eine genaue Kenntnis der Anwendungsprotokolle notwendig; die dynamische Komponente wird in uns bekannten Ansätzen für jede Anwendung spezifisch im Code der Firewall implementiert.

3.2 Application Level Gateways - Proxies

Im Gegensatz zu einem Paketfilter, greift ein Proxy* oder Application Level Ga-

---

* Ein Proxy wird hier als Application Level Proxy verstanden. Auf den Einsatz von Circuit Level Proxies wird hier nicht eingegangen, da sie den Datenstrom nur zu einem anderen, gesicherten Rechner umleiten und die Probleme der Verwendung von dynamischen Ports nicht lösen.

teway [6] auf Anwendungsebene in eine Verbindung ein. Bei seinem Einsatz wird die direkte Verbindung zwischen den angeschlossenen Netzen grundsätzlich oder für den vom Proxy umzusetzenden Dienst unterbunden. Dies kann man z.B. durch einen entsprechend konfigurierten Paketfilter erreichen. Eintreffende Pakete werden durch diesen nicht an die eigentliche Zieladresse versandt, sondern dem Proxy übergeben. Dieser kennt das jeweilige Anwendungsprotokoll, interpretiert die Daten und simuliert den eigentlichen Zielrechner, zu dem er eine eigene Verbindung aufbaut. Dabei ist er auch in der Lage, selektiv in den Datenstrom einzugreifen und Teile zu unterdrücken, zu verändern oder hinzuzufügen. Durch die Kenntnis der Semantik der übertragenen Daten ist es möglich, einzelne Interaktionen auf Anwendungsebene zu erlauben oder zu verbieten (z.B. Sperren des FTP “get“-Befehls). Des weiteren können vom Proxy zusätzliche Funktionen ausgeführt werden, so kann er z.B. eine Authentifizierung des Benutzers anfordern. Auch die Protokollierung und Auswertung von Aktivitäten zur Entdeckung von Angriffsversuchen ist realisierbar.

Wertend gegenüber der Vielzahl der genannten Vorteile ist zu beachten, daß jeder neue bzw. geänderte Dienst einen Neuimplementierung oder Veränderung eines Proxies erfordert und es durch das Durchlaufen des gesamten Anwendungsstacks zu einer nicht unerheblichen und gerade für Echtzeitanwendungen möglicherweise nicht tolerierbaren zusätzlichen Verzögerung kommen kann.

3.3 Bewertung existierender Lösungen

Die Leistungsfähigkeit der beschriebenen Paketfilter und Proxies hängt von der Ebene ab, in der die Kommunikation unterbrochen und beeinflußt wird. Proxies müssen jeweils spezifisch zur Verfügung gestellt werden und führen zu größeren Verzögerungszeiten, da die Pakete bis in die Anwendungsschicht verarbeitet werden. Paketfilter arbeiten dagegen schneller, sind aber weniger flexibel.

In der Regel ist es wünschenswert, aber oftmals nicht möglich, das Vorhandensein bzw. Verhalten eines Proxies transparent für die betroffenen Anwendungen zu gestalten. Dann ist es notwendig, Programme entsprechend zu konfigurieren bzw. zu modifizieren, was ebenfalls nicht in jedem Falle einfach realisierbar ist.

Ein weiterer Vorteil von Proxies stellt die Realisierungsmöglichkeit von zentralen Logging- und Warnsystemen dar, wohingegen ein einfacher und nicht wie nachfolgend beschrieben erweiterter Paketfilter nur Debuginformationen schreiben kann, die sehr schwer auszuwerten sind.

4 Die Verteilte Dynamische Multimedia Firewallarchitektur - VDMFA

4.1 Lösungsansatz und VDMFA Konzept

Die bisher beschriebenen Probleme können teilweise gelöst werden, wenn es gelingt, aufbauend auf die Analyse des Datenstromes dynamisch neue Regeln zu erzeugen und diese in den Firewalls durch Parametrisierung der vorhandenen IP-Filter entsprechend umzusetzen. Bisher dazu vorhandene Lösungen sind in der Regel wenig flexibel und werden vom Anbieter von Firewall-Software fest im Programmcode eingebettet.

Unser Ansatz versucht:

- die für die Konfiguration der Firewall auszuwertenden Datenströme geeignet zu einer Auswertekomponente hinzuleiten.
- eine Auswertung der Daten in flexibler Art und Weise durchzuführen und auf dieser Grundlage Kommandos für Paketfilter auf einem hohen und zunächst implementierungsunabhängigen Niveau zu generieren.
- die generierten Kommandos in einer heterogenen Landschaft von Firewalls zu verteilen und zu deren Konfiguration zu benutzen.
- völlige Transparenz für die kommunizierenden Teilnehmer zu schaffen.

Die Steuerung dieser Aktivitäten soll in einer durch den Anwender zur Laufzeit beeinfluß- und erweiterbaren Weise erfolgen. Durch ein modulares Design wird sichergestellt, daß neue Protokolle einfach und schnell in das System integriert werden können und mehrere Firewalls (bzw. die einzelnen Systemkomponenten einer Firewall) gemeinschaftlich konfiguriert und in ihrem Zusammenwirken koordiniert werden können.

Neben der eigentlichen Aufgabe, die Filtermöglichkeiten eines Firewall-Systems zu erweitern, kann die VDMFA für weitere Funktionen wie die Überwachung von Aktivitäten an den Netzübergängen und die gezielte Auslösung von Maßnahmen im Angriffs- oder Fehlerfall genutzt werden. Die Gesamtarchitektur ist in Abbildung 3 aufgezeigt und wird nachfolgend beschrieben.

4.2 Komponenten der VDMFA

Die VDMFA wird neben der oder den verwendeten auf IP-Filtern basierenden Firewalls, an die keine speziellen Anforderungen gestellt werden müssen, aus mehreren zusätzlichen Interface-Komponenten, einer Core-Komponente und einer Client-Komponente gebildet.

Für jede Firewall, die in das System integriert werden soll, wird eine Interface-Komponente benötigt. Diese stellt die Schnittstelle zur VDMFA dar, und ist daher für jeden Firewall-/Filtertyp neu zu entwickeln. Sie hat die Aufgabe, die von der Core-Komponente benötigten Daten zu liefern und die von ihr generierten Konfigurationsanweisungen entsprechend der Spezifika (Kommandoschnittstelle/Sprachumfang) der Firewall umzusetzen und auszuführen.

Die Core-Komponente realisiert die Kernfunktionalität, die nötig ist, um Datenströme auf der Anwendungsebene in für die Konfiguration der Firewall relevanter Hinsicht zu verarbeiten und entsprechende Aktionen auszulösen. Die Konfiguration der VDMFA Komponenten erfolgt über eine Multi-User fähige Netz-Schnittstelle. In der jetzigen experimentellen Implementierung ist der Client als Texteingabemodul gestaltet. Für eine automatische Initialisierung des Systems, aber auch zur Automatisierung sich wiederholender Abläufe ist eine Stapelverarbeitungsschnittstelle sinnvoll.

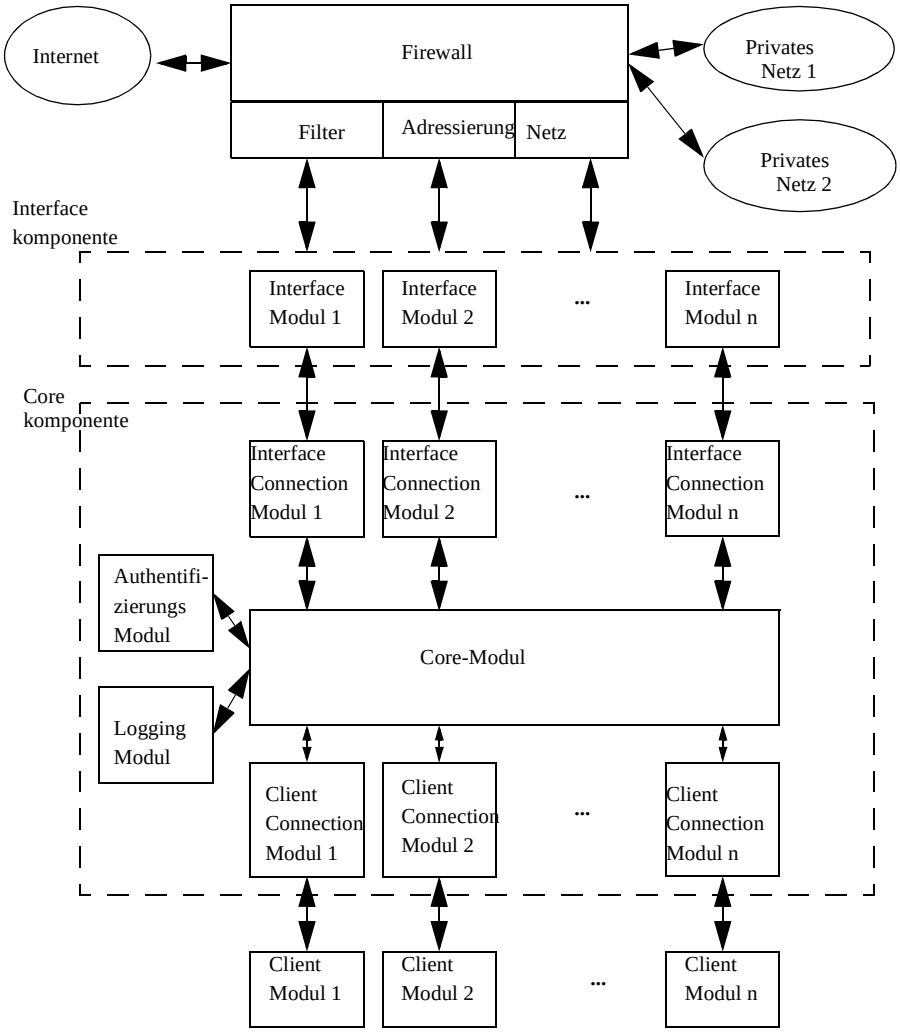

Abbildung 3: Komponenten der VDMFA

4.3 Beschreibung der Funktionalität durch Regeln

Die für die Beschreibung der Arbeitsweise notwendigen Regeln können vom Anwender mit Hilfe einer einfachen regulären Sprache notiert werden, die durch das System zur Programmlaufzeit entsprechend ausgewertet wird. Die Formulierung der Sprache, deren Regeln nachfolgend auszugsweise und teilweise informal beschrieben werden, erfolgte unter dem Aspekt der einfachen und kompakten Beschreibbarkeit der gewünschten Funktionalität und der einfachen Implementierbarkeit bei gleichzeitiger hinreichender Mächtigkeit. Im Abschnitt 5 werden wir die formale Beschreibung an einem Anwendungsszenario aus der IP-Telefonie praktisch darstellen.

Startsymbol der Sprache ist *Regelsystem*.

Regelsystem	→	Regel; Regelsystem
	|	ε
Regel	→	if Prädikat { Regel } else { Regel **}**
	|	Aktionsfolge

Als Prüfoperationen über den Daten des aktuellen Paketes ist ein Test auf Vorhandensein einer bestimmten Bit-Folge im Paket und zusätzlich an einem speziellen Offset, aber auch eine komplexere Operation wie das Auswerten des Paketes durch einen aus z.B. der ASN.1 Notation des zu behandelnden Protokolls abgeleiteten Parser möglich.

Prädikat	→	True
	|	False
	|	„Prüfoperation über Paketdaten"
	|	TEST_MODULE Modulbezeichner „Vektor zur Parametrisierung"

Zu diesem Zweck kann ein Programmodul mit Hilfe seines Klassennamens spezifiziert werden. Dieses besitzt eine Methode mit folgender Signatur (in Java Notation)

```
boolean TestMethod(byte[] data, Vector para_in, Vector
para_out)
```

und kann in der von uns gewählten Java-Realisierung zur Laufzeit dynamisch anhand des angegebenen Klassennamens (Modulbezeichners) geladen werden. Parameter werden als Kombination aus einem String als Identifikator und einem Wert in Vektoren sowohl über- als auch zurückgegeben. Damit ist eine weitestgehende Unabhängigkeit und Flexibilität möglich, entsprechende Module können einem Anwender auch für seinen individuellen Einsatzzweck als Primitive zur Formulierung von Regeln zur Verfügung gestellt werden, ohne daß er die genauen Spezifika des zu behandelnden Datenstromes kennt. Zusätzlich ist es möglich, bestimmte Prüfaktionen auch zustandsbehaftet auszuführen. Damit ermöglicht man, geeignet auf fragmentierte Daten oder eine bekannte logische Abfolge von Paketen zu reagieren.

Eine Aktionsfolge beschreibt - in einer Aufzählung von Einzelaktionen - Aktionen zur Steuerung des Gesamtsystems wie die Generierung und Parametrisierung neuer Module und Konfigurationskommandos für die IP-Filter in einer von der konkret anzusprechenden Firewall unabhängigen Meta-Notation. Dabei ist die Bezugnahme auf die von einem Testmodul gelieferten Rückgabeparameter über deren Namen möglich.

Aktionsfolge	→	Aktion; Aktionsfolge
	|	ε
Aktion	→	"Konfigurationsanweisung für IP-Filter"
	|	"Anweisung zur Steuerung des Gesamtsystems"

4.4 Core-Modul

Das Core-Modul ist die wichtigste Komponente der Gesamtarchitektur. Es hat die zentrale Aufgabe, die formalen Regeln zu verwalten und abzuarbeiten. Des weiteren werden von ihm Aufgaben wie Logging, Authentifizierung und die Bereitstellung der Konfigurationsschnittstelle übernommen.

Für jeden formalen Regelsatz der an das Core-Modul übergeben wird, wird ein Interface Connection Modul erzeugt. Dieses realisiert eine Verbindung zur entsprechenden Firewall, die das Modul mit Daten versorgt, einen Parser, der die Regeln umsetzt sowie einen Paket Decoder und einen Paket-Encoder, die für die Kommunikation mit dem Interface-Modul benötigt werden. Wird ein Paket von einem Interface-Modul an das Interface Connection Modul geschickt, so klassifiziert der Paket Decoder, um was für eine Information es sich handelt. Erkennt er eine Instruktion, dann wird entweder eine Anweisung an das Core-Module oder direkt an den Paket-Coder weitergegeben. Handelt es sich um ein Daten-Paket, so wird es an den Parser weitergereicht. Dieser analysiert das Paket anhand der zuvor eingegebenen Befehle (siehe Sprach-Syntax), erzeugt wenn nötig neue Interface Module und die gewünschten Konfigurationsanweisungen für die Firewall oder überführt das Gesamtsystem in einen neuen Zustand.

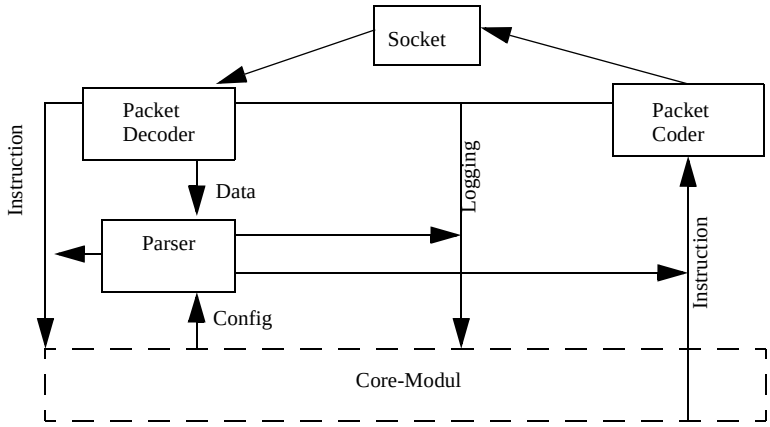

Abbildung 4: Interface Connection Modul

4.5 Interface-Modul

Diese Komponente ist das Verbindungsglied zwischen der Firewall und dem Core-Modul. Sie muß auf der Firewall Zugang zu den Interfaces haben, um den Netzverkehr abzuhören. Wird im Core-Modul ein neuer Regelsatz definiert, so erzeugt dieses ein Interface Connection Modul, welches dann das Interface-Modul auf der Firewall kontaktiert und es konfiguriert. Dabei wird dem Interface-Modul mitgeteilt, nach welchen Kriterien es die Pakete, die es an das Core-Modul weiterleiten muß, selektieren soll. Zur Entscheidung über die Selektion stehen dem Core-Modul die Header der IP-, TCP- und UDP-Pakete zur Verfügung. Wurde ein Paket selektiert, so wird es an das Core-Modul übergeben und dort wird die Semantik der Daten bis auf Anwendungsebene analysiert. Gegebenenfalls erhält das Interface-Modul über die Verbindung zum Core-Modul eine Anweisung, die Firewall neu zu konfigurieren.Unter Kontrolle des Interface-Moduls werden, falls notwendig, auch Daten in einer Richtung verzögert, um sicherzustellen, daß zunächst ein Kanal für die Antwort geöffnet werden kann.

Wenn das Core-Modul und das Interface-Modul nicht zwingend auf dem gleichen Host lokalisiert sind, besteht die Notwendigkeit, die zu untersuchenden Pakete über das

Netz zu transportieren. Da es sich bei den selektierten Paketen in den von uns zu behandelnden Szenarien um Kontrolldaten handelt, ist die daraus resultierende Mehrbelastung des Netzes schwach. Die in der Regel volumenintensiven Nutzdaten einer Verbindung, wie z.B. die RTP-Pakete einer Sprachübertragung werden nicht an das Core-Module weitergegeben.

4.6 Implementierungsspezifische Umsetzung

Jede einzelne Komponente wird als Dämon implementiert und kann auf verschiedene Hosts lokalisiert werden. Zwischen den einzelnen Systemen und dem Firewallhost werden Netzverbindung aufgebaut, die unter Nutzung einer Authentifizierungs- und Verschlüsselungskomponente kryptographisch abgesichert werden, um eine Manipulation zu verhindern. Um das System möglichst portabel zu gestalten, wurde die Core-Komponente in Java realisiert. Die systemnahe Interface-Komponente ist in C implementiert, da sie auf die Netz-Interfaces des Systems zugreift; Aufgaben die ein Java-Programm nur schwer elegant und performant lösen kann.

5 Die VDMFA für den H.323 IP-Telefonie Verbindungsaufbau

Nachfolgend werden das Zusammenspiel der einzelnen Komponenten, die zu behandelnden Datenströme und die für die Konfiguration der Paketfilter der Firewall generierten Regeln am Beispiel des Verbindungsaufbaus bei einer IP-Telefonie-Verbindung nach dem Standard H.323 erläutert. Die Darstellung erfolgt zur besseren Verständlichkeit informal und nimmt Bezug auf die in Abschnitt 2 gezeigte Abbildung 2.

Für den Q.931 Verbindungsaufbau wird eine TCP-Verbindung an den “well-known” Port 1720 benutzt. Soll von der Firewall ein Internet-Telefonie-Verbindungsaufbau erlaubt werden, so sind die Regeln des statischen IP-Filters wie in Tabelle 1 beschrieben zu konfigurieren. Die Firewallregeln werden dabei von uns in einer leicht verständlichen, allgemeinen und nicht filterspezifischen Notation angegeben. Dabei wird von Netzdevices abstrahiert, indem die Richtung der Verbindung (Incoming: aus dem Internet in das private Netz; Outgoing: aus dem privaten Netz in das Internet) angegeben wird.

Action	Direction	Source	Port	Destination	Port
Allow	Incoming	*	> 1024	<our host>	1720
Allow	Outgoing	<our host>	> 1024	*	1720

Tabelle 1: IP-Filter Ausgangskonfiguration

Mittels eines Clients wird ein Interface Modul “Q.931-Outgoing” mit der entsprechenden Regel erzeugt.

```
Int.Mod-Q.931-Outgoing:Select all TCP where(Outgoing,<our
hosts>,*,*,1720)
```

Dies bedeutet, daß alle TCP Pakete, die von einem internen Host von einem beliebigen Port an einen beliebigen externen Host an den Port 1720 gesendet werden, selektiert und an das Interface Connection Modul “Q.931-Outgoing” geschickt werden. Analog

muß über einen Client ein Interface Modul “Q.931-Incoming” erzeugt werden.

```
Int.Mod-Q.931-Incoming:Select all TCP where (Incoming, *, *, <our
hosts>, 1720)
```

Sendet nun Tom (siehe Abbildung 2) über die TCP-Verbindung an Port 1720 die Setup Nachricht, so wird die Regel des Interface Moduls “Q.931-Incoming” aktiviert und das TCP Paket wird von diesem an die Socket Schnittstelle des zugehörigen Interface Connection Modul geschickt. Der Paket Decoder erkennt, daß es sich um ein TCP Datenpaket handelt und übergibt dies an den Parser. Dieser besitzt die Fähigkeit, die Inhalte des Paketes gemäß der ASN.1 Protokollspezifikation zu interpretieren. Im betrachteten Fall muß er folgende Werte aus dem TCP Paket ermitteln:

```
SI:=Source IP-Adresse
DI:=Destination IP-Adresse
PT:=PDU-Type
```

sowie im Falle einer Connect-PDU:

```
H245P:=Cnnct_UUIE_h245Adress:port
```

Der Regelsatz für die Q.931 spezifische Auswertung läßt sich wie folgt beschreiben:

```
if PT=Setup {;}
if PT=Alerting {;}
if PT=Connect { create on firewall IP-Filter-Rule(Allow, incoming,
DI, *, SI, H245P);
create Int.Mod-H.245-Incoming(Incoming, DI, *, SI, H245P);
create Int.Mod-H.245-Outgoing(Outgoing, SI, H245P, SI, *);}
```

Bei einer Setup- und Alerting-Nachricht sind keine neuen Instruktionen notwendig. Bei einer Connect-Nachricht müssen die IP-Filterregeln um eine Regel, welche die TCP-Verbindung für den H.245 Kanal an den dynamisch bestimmten Port erlaubt, erweitert werden. Zusätzlich sind für diesen H.245 Kanal zwei neue Interface Module, mit den nachfolgend aufgeführten Regeln, dynamisch zu erzeugen, welche die ein- und ausgehenden Datenpakete auf dieser TCP-Verbindung kontrollieren.

```
if PT=Connect {
create IP-Filter-Rule(Allow, incoming, 129.206.200.127, *,
130.83.139.121, 1730);
create Int.Mod-H.245-Incoming(Incoming, 129.206.200.127, *,
130.83.139.121, 1730);
create Int.Mod-H.245-Outgoing(Outgoing, 130.83.139.121, 1730,
129.206.200,127, *);}
```

Die vom Parser erzeugten Interface Module haben folgende Spezifika:

```
Int.Mod-H.245-Incoming:Select all TCP where (Incoming, DI, *, SI,
H245P)
Int.Mod-H.245-Outgoing:Select all TCP where (Outgoing, SI, H245P,
SI, *)
```

Der Parser des H.245 Interface Connection Moduls analysiert nun die Datenpakete:

```
SI-245:=Source IP-Adresse
SP-245:=Source Port
DI-245:=Destination IP-Adresse
DP-245:=Destination Port
PT-245:=PDU-Type
```

sowie im Falle einer OpenLogicalChannel-PDU:

```
RTCPP:=fLCPs_mPs_h2250LCPs:H2250LCPs_mdCntrlChnnl:TSAPIdentifier
```

bzw. im Falle einer OpenLogicalChannelAck-PDU:

```
RTCPP:=fLCPs_mPs_h2250LCPs:H2250LCPs_mdCntrlChnnl:TSAPIdentifier
RTPP:=fLCPs_mPs_h2250LCPs:H2250LCPs_mdChnnl:TSAPIdentifier
```

Der Befehlssatz des H.245-Parsers kann wie folgt beschrieben werden:

```
if PT=TerminalCapabilitySet {;}
if PT=TerminalCapabilitySetAck {;}
if PT=MasterSlaveDetermination {;}
if PT=MasterSlaveDeterminationAck {;}
if PT=OpenLogicalChannel and Direction=Received {
create IP-Filter-Rule(Allow, Outgoing, SI-245, RTCPP, DI-245, *);}
if PT=OpenLogicalChannel and Direction=Sent {
create IP-Filter-Rule(Allow, Outgoing, SI-245, RTCPP, DI-245, *);}
if PT=OpenLogicalChannelAck and Direction=Received {
create IP-Filter-Rule(Allow, Outgoing, SI-245, RTPP, DI-245, *);}
if PT=OpenLogicalChannelAck and Direction=Sent {
create IP-Filter-Rule(Allow, Incoming, DI-245, *, SI-245, RTPP);}
if PT=CloseLogicalChannel and Direction=Received {
delete IP-Filter-Rule(Allow, Outgoing, SI-245, RTCPP, DI-245, *);}
if PT=CloseLogicalChannel and Direction=Sent {
delete IP-Filter-Rule(Allow, Outgoing, SI-245, RTCPP, DI-245, *);}
if PT=CloseLogicalChannelAck and Direction=Received {
delete IP-Filter-Rule(Allow, Outgoing, SI-245, RTPP, DI-245, *);}
if PT=CloseLogicalChannelAck and Direction=Sent {
delete IP-Filter-Rule(Allow, Outgoing, SI-245, RTPP, DI-245, *);}
if PT=EndSessionCommand {
delete IP-Filter-Rule(Allow, incoming, DI, DP-245, SI, SP-245);
exit;}
```

Bei den Nachrichten der Phase B des Verbindungsaufbaues sind keine neuen Instruktionen notwendig. Mit einer OpenLogicalChannel-Nachricht werden die RTCP Ports mitgeteilt, mit einer OpenLogicalChannelAck-Nachricht zusätzlich die für die Audiodatenübertragung verwendeten RTP Ports. Mit deren Kenntnis können die IP-Filterregeln dynamisch erweitert werden, um nachfolgend die RTP und RTCP Kanäle freizugeben. Beim Verbindungsabbau werden wiederum H.245 Kontrollnachrichten gesendet, in deren Folge die RTP und RTCP Kanäle geschlossen werden, weshalb der Parser Befehle zum Löschen der IP-Filterregeln erzeugt. Als Reaktion auf das EndSessionCommand zum Beenden der H.245-Verbindung werden die entsprechende Filterregel gelöscht und zuletzt das H.245 InterfaceConnectionModul selbst beendet.

6 Zusammenfassung

Beim Design unserer VDMFA lag der Schwerpunkt auf einem modularen Aufbau und der Forderung, eine Architektur zu schaffen, die es ermöglicht, die vorhandenen, bewährten und schnellen IP-Filter auch für komplexe Protokolle zu verwenden. Für neue Protokolle müssen neue Interface Modul Regeln beschrieben und ein zugehöriges In-

terface Connection Modul realisiert werden. Dies ist mit relativ geringem Aufwand und als dynamische Ergänzung des bestehenden Gesamtsystems möglich.

Damit unterscheidet sich unser Ansatz von anderen uns bekannten dynamischen Erweiterungen [7][8], bei denen für jede neue Anwendung ein Softwaremodul entwikkelt und statisch in die Firewall eingebettet werden muß. Er zeichnet sich auch durch eine kürzere mögliche Reaktionszeit bis zur Unterstützung eines neuen Dienstes und eine hohe Flexibilität bei veränderten Anforderungen aus. Im nächsten Schritt beabsichtigen wir, ein Authentifizierungs- und ein Warningmodul zu realisieren. Das Warningmodul stellt eine für IP-Filter sehr wichtige Erweiterung dar, da seine Funktionalität bei reinen IP-Filtern nur sehr schlecht realisierbar ist. Auch ist durch die Modularisierung und Verwendung von formalen Regeln die Möglichkeit gegeben, Firewalls mehrerer Hersteller mit nur einem Client konsistent zu konfigurieren; es muß nur das entsprechende Interface-Modul implementiert werden.

IP-Filter zeichnen sich gegenüber Proxies durch eine geringere Verzögerung aus, da die Daten nicht auf Anwendungsebene analysiert werden. Diesen Vorteil macht sich auch die VDMFA zunutze, das Interface Module analysiert nur Kontrolldatenströme. Die Nutzdatenströme hingegen durchlaufen nur IP-Filter, wodurch die Verzögerung gering gehalten wird, was gerade für multimediale Audio- und Videodaten von großer Bedeutung ist.

Bei einer Erweiterung sicherheitsrelevanter Systeme um neue Komponenten muß auch stets deren eigene Absicherung hinterfragt werden. So ist in unserer Architektur zum Beispiel zu gewährleisten, daß die Komponente Core-Modul auf einem sicheren Rechner lokalisiert ist, damit ein Angreifer die Regeln und Befehlssätze nicht modifizieren kann. Inwieweit mit den neuen Komponente der VDMFA neue Angriffspunkte für Angreifer geschaffen werden, wird von uns aktuell im Detail untersucht. Der Entwurf erfolgte unter Beachtung uns bekannter Erfahrungen zum Aufbau von Sicherungsinfrastrukturen.

Literatur

[1] D. E. Comer: Internetworking with TCP/IP Volume I, 2nd Edition, Prentice Hall, 1991

[2] W. R. Cheswick, S. Bellovin: Firewalls and Internet Security, Addison Wesley, 1994.

[3] ITU-T Recommendation H.323 “Visual telephone Systems and Equipment for Local Area Networks Which Provide A Non-Gurararanteed Quality of Service.”, Genf, 1996

[4] ITU-T Recommendation H.323 V.2 “Packet-Based Multimedia Communication Systems”, Genf, 1998

[5] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson: RTP: A Transport Protocol for Real-Time Applications, RFC 1889, Internet Engineering Task Force, January 1996

[6] S. Strobel: Firewalls für das Netz der Netze, dpunkt-Verlag, 1997

[7] Anonymous, Stateful Inspection Firewall Technology Tech Note, Whitepaper, http://www.checkpoint.com/products/technology/stateful1.html

[8] Anonymous, Cisco’s PIX Firewall Series an Stateful Firewall Security, Whitepaper, http://www.cisco.com/warp/public/751/pix/nat_wp.htm

[9] D. Chouinard, J. Richardson, M. Khare, “H.323 and firewalls: The problems and pitfalls of getting H.323 safely through firewalls.” Revision 2, White Paper, Intel Corporation, October 1997