Zürcher
Hochschule
Winterthur

Hochschule
für Architektur,
Technik, Wirtschaft
und Sprache

Mitglied
der Fachhochschule
Zürich

H.323 Firewall-Proxy

Projektarbeit SS2000 – Sna02
Fach Kommunikationssysteme (KSy)
Dozent: Dr. Andreas Steffen
Studierende: Roman Buser (IT3a), Michael Stolz (IT3a)
Termin: 14.3. – 19.5.2000
Labor: E523
Arbeitsplätze: KSY007, KSY008, KSY123

Inhalt

1. Zusammenfassung

Wir haben eine Firewallerweiterung geschrieben, die es ermöglicht, eine H.323 Verbindung durch einen Firewall hindurch zu benutzen, ohne das sämtliche Ports offen sein müssen. Da erst während der Verbindung vereinbart wird, welche Ports verwendet werden, ist das normalerweise nicht möglich. Um das zu erreichen überwachen wir den Verkehr auf den Verbindungen und geben bei Bedarf weitere Ports im Firewall frei.
Zur Zeit werden noch nicht alle verwendeten Ports überwacht, sondern nur jener, auf dem die Verbindung initialisiert wird. Das zeigt aber bereits, dass unser Ansatz funktioniert.

Abstract

We implemented a firewall extension making it possible to use an H.323 connection through a firewall without opening every port. An ongoing connection will negotiate which ports are going to be used. A traditional firewall can't cope with this behaviour. Therefore we monitor traffic on the connections and open further firewall ports if needed. Until now we do not monitor all used ports, but only the one which is used to initialise the connection. This alone shows that our approach is a working solution.

2. Aufgabenstellung Sna02

Kommunikationssysteme (KSy)

Projektarbeiten SS 2000 - Sna02

H.323 Firewall-Proxy

Studierende:

- Roman Buser, IT3a
- Michael Stolz, IT3a

Termine:

- Ausgabe: Dienstag, 14.03.2000 10:00 - 11:00 im E509
- Abgabe: Freitag, 19.05.2000

Beschreibung:

H.323 basierte Multimediaanwendungen, wie zum Beispiel Microsoft Netmeeting, vertragen sich schlecht mit sicheren Firewalls, da im Verlauf der Kommunikation dynamisch eine ganze Anzahl von im voraus unbekannten UDP und TCP Ports für die Übertragung von Audio/Video-Kanälen, respektive Anwenderdaten geöffnet werden müssen.

In dieser Arbeit soll auf der Basis des Open H.323 Stacks ein Firewall-Proxy erstellt werden, der das Q.931 Call Setup Protokoll auf dem TCP-Port 1720, sowie das anschliessende H.245 Control Protokoll analysiert und die darüber ausgehandelten Multimedia Ports für die Dauer der Kommunikation im Firewall öffnet und nachher wieder schliesst. Alle zu einer H.323 Verbindung gehörenden IP-Pakete sollen durch den Firewall unverändert durchgeschalten werden, d.h. IP Masquerading ist im Rahmen dieser Arbeit nicht vorgesehen. Jede Verbindung soll in einem Logfile protokolliert werden.

Aufgaben:

- Einarbeiten in den Open H.323 Source Code
- Einarbeiten in die Linux 2.2.x Firewall Regeln basierend auf ipchains
- Erstellen einer kurzen Software-Spezifikation für das H.323 Proxy-Modul
- Codieren des H.323 Proxy-Moduls unter Linux (GNU C++ Compiler / Debugger)
- Austesten des H.323 Proxy-Moduls mit folgenden Konstellationen:
  - Netmeeting - Netmeeting (Direct Mode)
  - Netmeeting - Netmeeting oder Hicom Xpress (via RADVision Gatekeeper)
  - Netmeeting - RADVision ISDN Gateway (via RADVision Gatekeeper)
- Dokumentation der Projektarbeit

Infrastruktur / Tools:

- Raum: E523
- Rechner: 3 PCs mit Dual-Boot: SuSE Linux 6.3 / Windows NT 4.0
- Telefone: 1 ISDN-Telefon
- VoIP: RADVision L2W-323 Gateway / Gatekeeper, MS Netmeeting 3.01
- SW-Tools: GNU C++ Compiler / Debugger, Open H.323 Source Code

Literatur / Links:

- Open H.323 Website:
  http://www.de.openh323.org
- Open H.323 Mailing List:

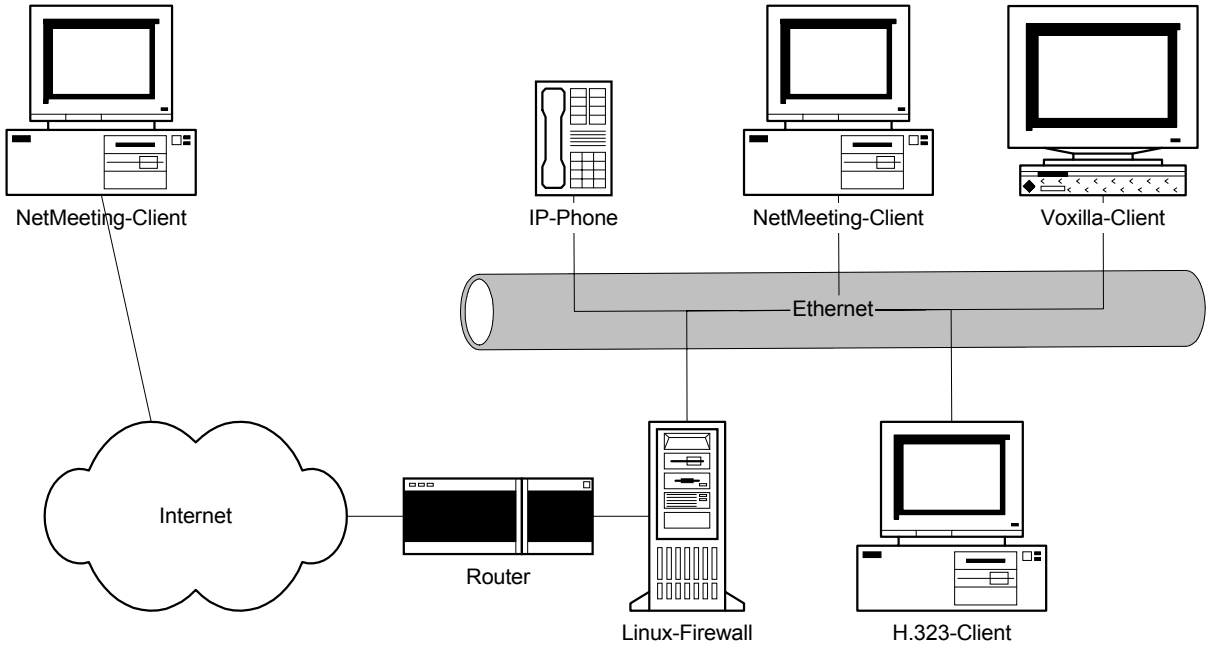
http://www.de.openh323.org/list.html

- Open H.323 Tips zur Installation
  http://fbi.zhwin.ch/PA/openh323tips.txt
- ITU-T Recommendation H.323
  Packet-Based Multimedia Communications Systems
- ITU-T Recommendation H.225.0
  Call Signalling Protocols and Media Stream Packetization
- ITU-T Recommendation H.245
  Control Protocol for Multimedia Communication
- ITU-T Recommendation X.680
  Abstract Syntax Notation One (ASN.1): Specification of Basic Notation
- ITU-T Recommendation X.691
  ASN.1 Encoding Rules: Specification of Packed Encoding Rules (PER)
- Linux Firewall and Proxy Server HOWTO
  http://metalab.unc.edu/mdw/HOWTO/Firewall-HOWTO.html
- Linux Ipchains HOWTO
  http://www.rustcorp.com/linux/ipchains/HOWTO.html
- Linux IP Masquerade HOWTO
  http://members.home.net/ipmasq/ipmasq-HOWTO-1.82.html
- IP_Masq Module für Netmeeting 2 und Linux 2.0.x Kernel
  http://members.home.net/ipmasq/patches/ip_masq_h3231.c.tgz

3. Einleitung

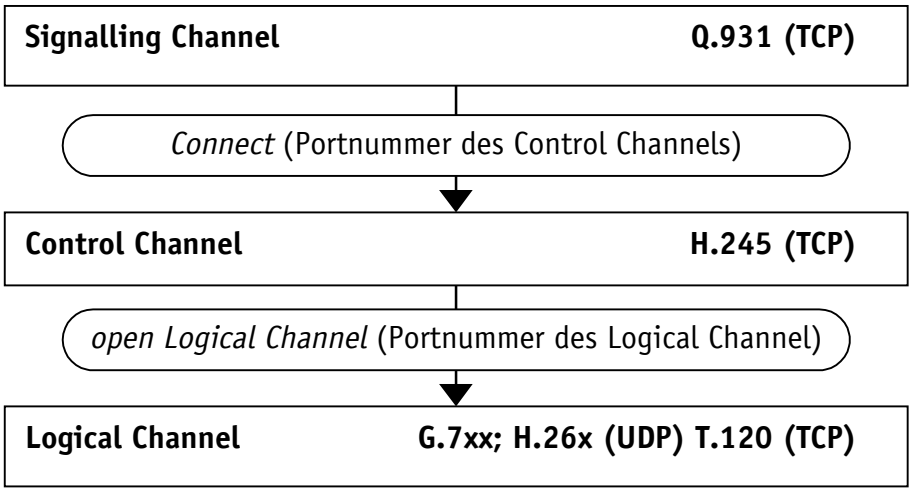
Ausgangssituation

Stellen wir uns folgende Situation vor: Ein Betrieb hat ein geschütztes Intranet, das durch einen Firewall vom Internet getrennt wird. Um eine maximale Sicherheit vor Attacken aus dem Internet zu gewährleisten, schliesst der Firewall-Administrator alle nicht für ein spezifisches Protokoll notwendigen Ports und öffnet nur jene, für die ein Benutzer berechtigten Nutzen anmeldet.
Probleme ergeben sich bei dieser Konfiguration sobald eine Anwendung verwendet werden soll, die im Betrieb dynamisch neue Ports verwendet, welche zwischen den Beteiligten Kommunikationspartnern ausgehandelt werden. Zum Beispiel H.323-Clients wie NetMeeting und Voxilla fallen in diese Kategorie.
An dieser Stelle setzt unsere Projektarbeit an: Der Firewall soll sich automatisch so rekonfigurieren, dass H.323-Verbindungen möglich werden.



Überblick H.323

Um was es bei H.323 geht kann in [6] nachgelesen werden. Wie es genau funktioniert steht in [7]. Im folgenden gehe ich nur auf den Ablauf einer Verbindung ein.



Ein Anruf erfolgt auf dem Signalling Channel, der auf TCP-Port 1720 ist. Er kann von einem beliebigen Port aus erfolgen. Wenn der Anruf angenommen wird, folgt die Vereinbarung der Portnummer des Control Channels, in dem diese Verbindung verwaltet wird. Dabei wird nur der Port des Angerufenen bestimmt. Der Port des Anrufers ist einfach eins nach dem Signalling Channel (ein Beispiel ist im Kapitel Testresultate). Im Control Channel wiederum werden die Portnummern sämtlicher benötigter Logical Channels vereinbart, wenn diese geöffnet werden. Wenn ein Logical Channel nicht mehr gebraucht wird, wird er hier auch wieder geschlossen. Die Daten im Signalling und Control Channel werden nach ASN.1 [10] codiert. Was innerhalb der Logical Channels abläuft, ist für diese Projektarbeit uninteressant.

Infrastruktur

Es standen uns drei PC-Arbeitsplätze im KSy-Labor mit folgender Konfiguration zur Verfügung.

KSY007, KSY008:

Hardware:

- Pentium III (450Mhz), 128MB RAM
- Siemens Multimedia-Keyboard
- 2 Ethernet-Karten (Digital DE500)

Software:

- Windows NT 4.0 auf interner SCSI-Disk
- SuSE Linux 6.3 auf Wechselharddisk
- Netmeeting 2.11
- Open H.323 Project [12]

KSY123:

Hardware:

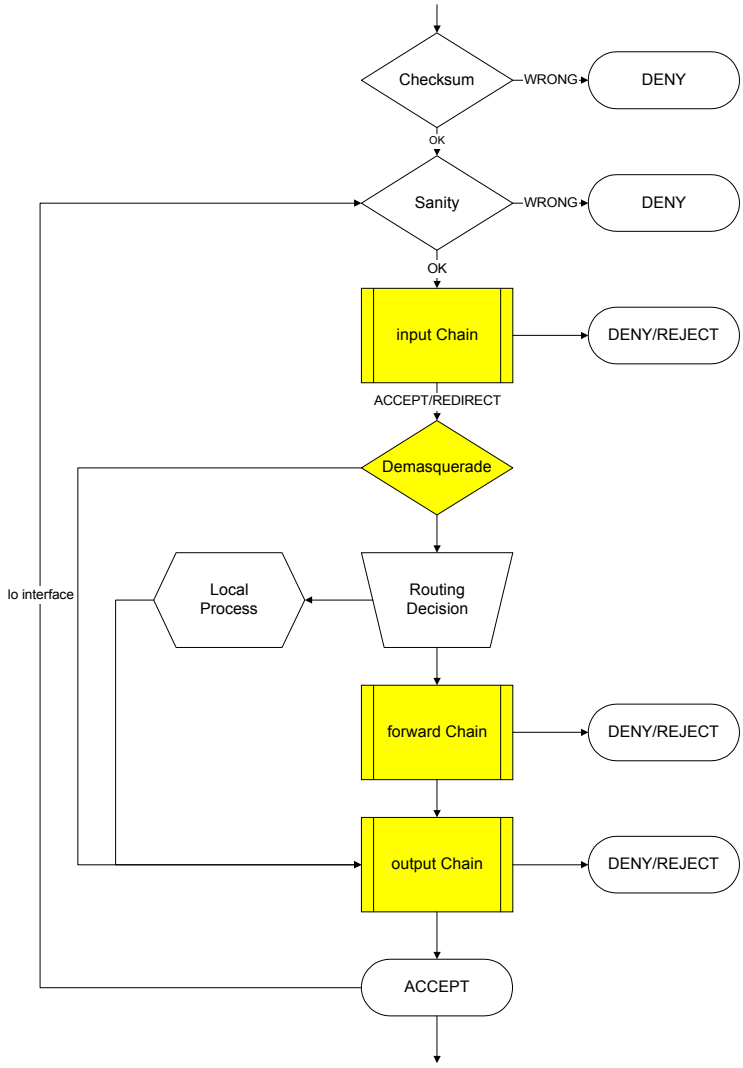
- Pentium III (550Mhz), 128MB RAM
- 2 Ethernet-Karten (3Com 3c905C)

Software:

- Windows NT 4.0 auf interner SCSI-Disk
- SuSE Linux 6.3 auf Wechselharddisk
- Kernel 2.2.13 (mit IP Chains Firewall und Firewall Netlink Device)

Der Linux Firewall (ipchains)

Der Linux-Kernel kann mit Unterstützung für einen Packet-Filtering-Firewall kompiliert werden. In Version 2.2 des Kernels nennt sich dieser Firewall *IP Firewall Chains* oder kurz *ipchains*. Der Pfad, den ein Paket beim Passieren des Firewalls durchläuft, ist recht kompliziert:



Es existieren 3 eingebaute Chains (input, forward, output), welche das Paket durchlaufen muss. Innerhalb jeder Chain wird anhand von Regeln (rules) entschieden, zu welchem Ziel (target) ein Paket als nächstes befördert wird. Sechs Ziele (ACCEPT, REJECT, DENY, MASQ, REDIRECT, RETURN) sind vordefiniert, es kann aber auch eine benutzerdefinierte Chain erzeugt und als Ziel angegeben werden. Die genaue Funktionsweise der ipchains kann in [1] nachgelesen werden.

4. Analyse

Der Titel der Projektarbeit ist leider etwas verwirrend. Aus der Aufgabenstellung geht hervor, dass kein eigentlicher Proxy-Server zu realisieren ist, sondern eine Art dynamisches Firewalling.

Proxy-Server versus dynamisches Firewalling

Ein richtiger H.323 Proxy-Server tritt gegenüber den beiden kommunizierenden Clients als ihr Kommunikationspartner auf und stellt intern die Verbindung zwischen den beiden her. Damit das funktioniert, muss allerdings ein grosser Aufwand betrieben werden: Analyse des gesamten Verkehrs auf allen beteiligten Channels, Anpassen der IP-Adressen und Portnummern in diesen Datenströmen, Weiterreichen der Daten. Somit stellt ein Proxy-Server im Grunde genommen 2 komplette Clients dar. Einen richtigen Proxy-Server zu schreiben, wäre also zu aufwändig für eine 7-wöchige Projektarbeit.
Unter dynamischem Firewalling mit H.323-Unterstüzung verstehen wir hingegen die Fähigkeit des Firewall, über ihn laufende H.323-Verbindungen zu analysieren und sich so zu rekonfigurieren, dass von den H.323-Clients neu vereinbarte Channels auf den entsprechenden TCP- resp. UDP-Ports freigegeben werden. Im Gegensatz dazu wird ein konventioneller Firewall von der Systemadministratorin aufgesetzt und ändert ohne ihr Zutun seine Konfiguration nicht.
Recherchen im WWW ergaben, dass auch andernorts auf diesem Gebiet entwickelt wird [2][3].

Lauschangriff

Als erstes stellt sich die Frage, wie sich das Q.931 Call Setup Protokoll auf Port 1720 mitverfolgen lässt. Offensichtlich der einzige Punkt, an dem angesetzt werden kann, ist der Linux-Firewall. Nur dort sind die Pakete zugänglich, welche den Firewall-Server passieren sollen.
Die naheliegendste Lösung wäre also, ein Firewallmodul zu schreiben, welches die gewünschte Funktionalität bietet, wie es beispielsweise für den Kernel 2.0 existiert [4]. Dieses Modul müsste den hohen Anforderungen genügen, die an Kernelcode gestellt werden. Die Verwendung des umfangreichen, in C++ geschriebenen, Open H.323 Stacks würde damit wohl unmöglich werden.

Firewall Netlink Device

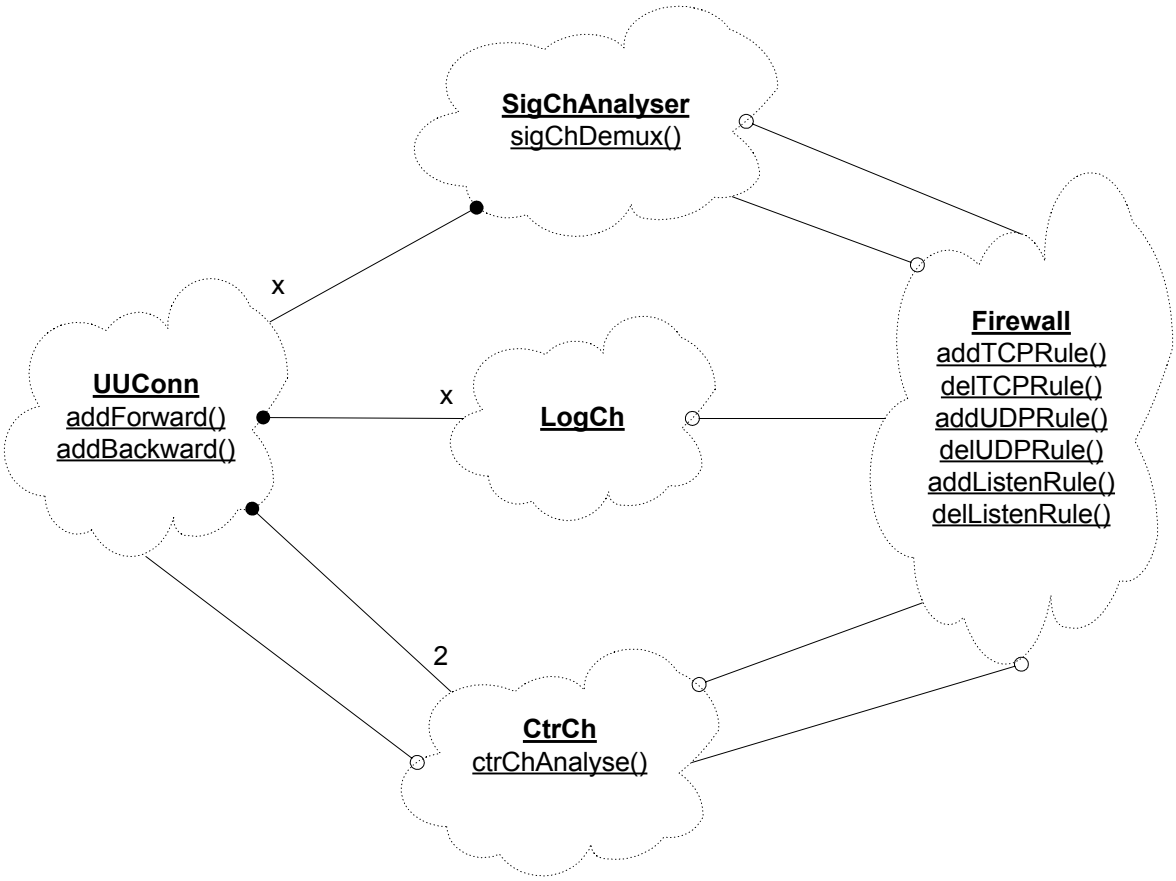
Eine zweite Möglichkeit ergibt sich durch die Firewalloption CONFIG_IP_FIREWALL_NETLINK: Man kann dann eine Firewallrule (Option –o von ipchains) erzeugen, welche die übereinstimmenden Pakete auf ein spezielles Character Device (firewall netlink device) kopiert. Dort kann ein Programm aus dem Userspace auf die Pakete zugreifen.
Nach der Analyse/Überprüfung der Pakete können diese mittels Ethertap Device (/dev/tap0) wieder in den Kernel eingespiesen werden. Dieses Device wird mit der Kerneloption CONFIG_ETHERTAP aktiviert.
Die Library libfw-0.2 stellt ein Interface mit dieser Funktionalität zur Verfügung. Dadurch wird der Programmierer von der Verwendung der oben genannten Devices verschont.
Diese Lösung hat zudem den Vorteil der Zukunftssicherheit, da aller Voraussicht nach im Kernel 2.4 eine ähnliche Funktion enthalten sein wird [5].

H.323-Analyse

Um die so erhaltenen Pakete zu analysieren, greifen wir auf das Open H.323 Project [12] zurück. Es stellt einen H.323-Stack zur Verfügung. Dieser hat Funktionen, welche die ASN.1 [10] Struktur in eine C++ Klassenstruktur abbilden. Darin werden die Meldungen auf dem Signalling und Control Channel (definiert in [8][9]) übertragen. Wie diese Struktur verwendet wird, kann dem Testprogramm DumpH323 entnommen werden, welches auch zu [12] gehört (dump323/main.cxx). Um mit den ASN.1 Daten umzugehen, benötigen wir auch noch die PWLib [13], welche die ASN.1 Typen definiert.
Die Meldungen sind in einzelne TPKT's [11] eingepackt und werden in einem TCP-Stream übertragen. TPKT's geben dem TCP-Stream eine Paketstruktur. Dadurch kann das Ende einer Meldung problemlos erkannt und die Analyse des Inhaltes gestartet werden.

5. Design

Unser Programm hat folgende Struktur:



- H323fwadm (H323 Firewall Administration):
  - Wrapper-class für die Funktionen der libfw-0.2.
  - Es können Rules gesetzt und gelöscht werden. Das wird von allen Channel Objekten gemacht.
  - Wenn die Rules neue Daten liefern, werden Callback-Funktionen der Channel Objekte aufgerufen.
  - Sollte als Singleton implementiert werden
- SigChAnalyser (Signalling Channel Analyser):
  - Gibt die Nutzdaten der Pakete vom Signalling Channel an das richtige UUConn Objekt weiter. Welches das ist, wird anhand der Sender- und Empfänger-IP-Adresse bestimmt. Falls es noch nicht existiert, wird es erzeugt.
  - Verwaltet die Rules des Signalling Channels (beide Richtungen; mehrere Verbindungen)
- UUConn (User to User Connection):
  - Für jede Verbindung die über den Firewall geht, existiert ein Objekt dieser Klasse. Es analysiert die Daten vom SigChAnalyser und erstellt wenn gefordert die zwei zugehörigen CtrCh Objekte. Für jede Richtung eines.
  - Verwaltet eine Liste mit den Logical Channel Objekten
- CtrCh (Control Channel):
  - Sammelt und analysiert die Daten des Control Channels. Es erhält nur die Daten einer Richtung. Die Daten erhält es vom H323fwadm. Wenn gefordert werden die LogCh Objekte erstellt oder gelöscht.
  - Verwaltet die Rule eines Control Channel (eine Richtung)
- LogCh (Logical Channel):
  - Verwaltet nur die Rules eines Logical Channel (beide Richtungen). Sonst passiert hier nichts.

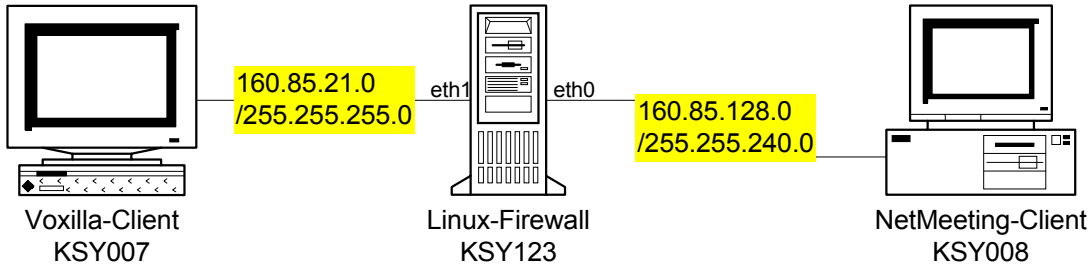
Um diese Klassenorganisation zu verwenden, wird zuerst je ein Objekt der Klasse H323fwadm und SigChAnalyser erzeugt. Letzteres erzeugt dann die weiteren Objekte.

6. Implementation

Die Analyse der Pakete erfolgt mit Funktionen aus dem Open H.323 Project [12]. Verwendet werden die Klassen q931, h225, h245. Des weiteren greifen wir auf die Pwlib [13] zurück. Diese stellt diverse Containerklassen und die ASN.1 Typen zur Verfügung.
Im momentanen Stand des Projektes wird nur der Signalling Channel überwacht. Da H323fwadm auf einer C-Library (libfw-0.2) basiert, konnten die Callback-functions nicht als Klassen-Memberfunctions ausgeführt werden. Deshalb ist die Callback-function des SigChAnalyser's als friend-function implementiert. Das dient nur zu Demonstrationszwecken. In einer weiteren Version muss eine Object-Verwaltung integriert werden, damit die Callback-functions Member-functions sein können. Am besten wäre es den Code der ipchains Library anzupassen (fwinterface.c).
Der Code zur Analyse des Control Channel ist bereits vorhanden und getestet, wird aber momentan nicht benutzt.
Die Portnummer des Control Channels wird in den Meldungen setup, connect, alerting und facility gesucht. Die Portnummern der Logical Channels werden vorerst nur aus der open Logical Channel Meldung gelesen. Bei Bedarf müssen weitere Meldungen analysiert werden. Die vollständigen ASN.1 Spezifikationen dieser Meldungen sind in [8] und [9].
Wenn jetzt H323fwadm ein neues Paket an einen Channel übergibt, überprüft dieser zuerst ob ein ganzes TPKT [11] vorliegt. Sobald das der Fall ist, werden die darin enthaltenen Daten ausgewertet und anschliessend gelöscht (Das löschen ist noch nicht implementiert).

7. Test

Testanordnung



Die Routingtables der beteiligten PCs werden so konfiguriert, dass aller Verkehr von KSY008 zu KSY007 und umgekehrt über KSY123 laufen muss. Auf KSY123 kommen die Daten jeweils auf dem einem Netzwerkinterface hinein und verlassen KSY123 auf dem Anderen. Somit muss der Firewall gezwungenermassen passiert werden.

Testablauf

1. Starten unserer Firewall-Extension (analyse > analyse.log 2>&1)
2. Anruf von Netmeeting zu Voxilla
3. Anruf annehmen
4. Verbindung in Voxilla beenden.
5. Anruf von Netmeeting zu Voxilla
6. Anruf ablehnen

Testresultate

Die Ausgaben vom Firewall wurden in der Datei analyse.log abgelegt. Sie enthält den gesamten Verkehr des Signalling Channels. Entscheidend ist jedoch nur Zeile 277:
Analyse.log(277): H245 Port : 1034 #
Das ist der eine Port der für den Control Channel verwendet wird. Der andere entspricht der Portnummer des Signalling Channels plus eins. In unserem Fall 1405+1 = 1406. Diese Nummer wird noch nicht aus den Daten extrahiert.
Wir sehen, dass die Firewall-Extension sämtliche Anrufe, die durch den Firewall erfolgen, erkennen und analysieren kann. Damit ist das erste Ziel unserer Arbeit erfüllt. Der gefundene Control Channel könnte nun auf analoge Weise behandelt werden.

8. Installation

a) Als erstes muss ein geeigneter Kernel kompiliert werden, da der bei SuSE 6.3 mitgelieferte modulare Kernel kein Firewall Netlink Device beinhaltet. Es müssen die folgenden Optionen gesetzt werden:
   - CONFIG_FIREWALL=y
   - CONFIG_IP_FIREWALL=y
   - CONFIG_IP_FIREWALL_NETLINK=y
   - CONFIG_ETHERTAP=y

   Wie man den Linux-Kernel kompiliert, ist in [14] detailliert beschrieben.

b) Danach muss das Firewall Netlink Device im /dev-Verzeichnis angelegt werden:
   ```
   mknod –m 600 /dev/fwmonitor c 36 3
   ```

c) Als nächstes werden die Dateien unserer Projektarbeit in einem frei wählbaren Verzeichnis ausgepackt (z.B. ~/src):
   ```
   cd ~/src
   tar xvIf /cdrom/H323fwext_0.1.tar.bz2
   ```

d) Der Pfad muss in der Datei `~/src/analyzer/setpath` folgendermassen angepasst werden:
   ```
   FWEXT_ROOT_PATH=~/src
   ```

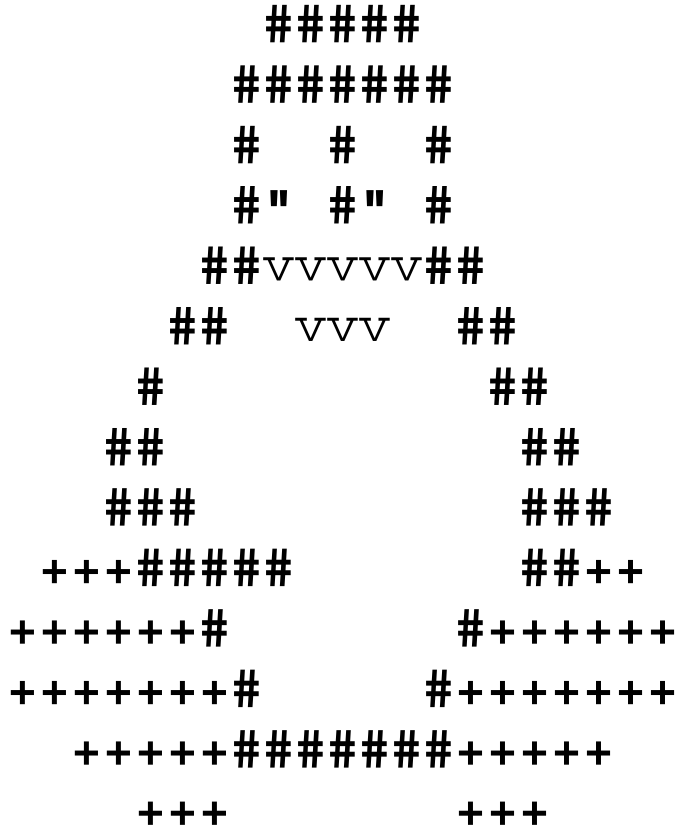
e) Pfad setzen:
   ```
   . analyzer/setpath
   ```

f) Programm starten:
   ```
   analyzer/analyse
   ```

g) Das Programm kann mit `cd analyzer; make` neu compiliert werden, falls gewünscht (Achtung: dauert etwas)

9. Schlusswort

Die vorliegende Version unseres Programms überwacht erst den Signalling Channel. Der ganze Verkehr darin wird ausgegeben. Die Portnummern werden herausgefiltert und nochmals separat ausgegeben. Die übrigen Channels werden weder überwacht, noch werden die benötigten Ports geöffnet.
Der Hauptgrund für den geringen Funktionsumfang liegt an der grossen Einarbeitungszeit in die vorhandenen Sourcecodes, die wir benötigten. Deren Dokumentation reicht von knapp bis nicht existent. Das Open H.323 Projekt ist sehr umfangreich und man muss zuerst herausfinden, welchen Teil davon man braucht. Die Verbindung von C und C++ war auch nicht problemlos. So ist momentan noch nicht entschieden, wie die Callback-functions von der C-Library aus aufgerufen werden sollen. Auch das linken bereitet Probleme. Es ist zwingend erforderlich, dass in der Kommandozeile zuerst die Objectfiles und dann die Libraryfiles aufgeführt werden. Wenn die Libraryfiles voneinander abhängen, ist auch hier die Reihenfolge entscheidend.
In einer Fortsetzung gibt es also noch viel zu tun.



10. Anhänge

Verzeichnis der Dateien auf CDROM

- H323_Firewall-Proxy_Sna02_V01.doc
  Projektdokumentation
- H323fwext_0.1.tar.bz2
  Sourcecode- und Binaries-Archiv des gesamten H323fwext-Projekts (im bzip2-tar-Format)
- H323fwadm-dev.tar.bz2
  Sourcecode- und Binaries-Archiv des H323fwadm-Teils (im bzip2-tar-Format)
- ./archiv
  Diverse tar-Archive zu den Themen H.323 und Firewalls: u.a. ipchains, tcpdump, libfw-0.2, Open H.323 Stack, pwlib

Literaturverzeichnis

[1] Linux IPCHAINS-HOWTO, v1.0.7, March 1999, Paul Russell.
http://www.linuxdoc.org/HOWTO/IPCHAINS-HOWTO.html

[2] SPF: Stateful Packet Filtering.
ftp://ftp.interlinx.bc.ca/pub/spf

[3] THE SINUS FIREWALL PAGE.
http://www.ifi.unizh.ch/ikm/SINUS/firewall/

[4] IP Masq module for Net Meeting 2.
http://members.home.net/ipmasq/patches/ip_masq_h3231.c.tgz

[5] The Netfilter Project: Packet Mangling for Linux 2.3+
http://antarctica.penguincomputing.com/~netfilter/

[6] A Primer on the H.323 Series Standard
http://www.databeam.com/standards/index.html

[7] ITU-T Recommendation H.323:
Packet-Based Multimedia Communications Systems

[8] ITU-T Recommendation H.225.0:
Call Signalling Protocols and Media stream Packetization

[9] ITU-T Recommendation H.245:
Control Protocol for Multimedia Communication

[10] ITU-T Recommendation X.680:
Abstract Syntax Notation One (ASN.1). Specification of Basic Notation

[11] RFC 1006
ISO Transport Service on top of the TCP

[12] The Open H.323 Project
http://www.openh323.org

[13] Portable Windows Library
http://www.equival.net/pwlib/

[14] The Linux Kernel HOWTO
http://www.linuxdoc.org/HOWTO/Kernel-HOWTO.html

Logfiles

- ./analyzer/analyse.log

Sourcecode

- ./analyzer/include
  - crtCh.h
  - H323fwadm.h
  - logCh.h
  - sigChAnalyser.h
  - tcpstream.h
  - uuConn.h
  - fwinterface.hpp
- ./analyzer/src
  - crtCh.cxx
  - H323fwadm.cxx
  - logCh.cxx
  - main.cxx
  - sigChAnalyser.cxx
  - tcpstream.cxx
  - uuConn.cxx