HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

ins institute for networked solutions

Firewall-Analyse-Tool

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2009

Autoren:	Christoph Rebsamen, Gieri Kohler
Betreuer:	Roman Ammann
Projektpartner:	HSR, Hochschule für Technik Rapperswil
Verantwortlicher:	Prof. Beat Stettler
Gegenleser:	Thomas Letsch

Abstract

Abteilung	Informatik
Namen der Studierenden	Christoph Rebsamen Gieri Kohler
Studienjahr	HS 2009
Titel der Studienarbeit	Firewall-Analyse-Tool
Themengebiet	Internet-Technologien und –Anwendungen
Projektpartner	HSR, Hochschule für Technik Rapperswil
Institut	INS, Institute for Networked Solutions

Kurzfassung der Studienarbeit:

Aufgrund der stetig steigenden Sicherheitsanforderungen und der Komplexität heutiger Firewallumgebungen wird es für Netzwerkadministratoren immer schwieriger, verlässliche Aussagen zu treffen, ob ein bestimmter Netzwerkstrom sein Ziel erreicht oder nicht.

Diese Studienarbeit befasst sich deshalb mit der Entwicklung einer Software für das Einlesen von Firewallkonfigurationen verschiedener Hersteller und das Übersetzen in ein einheitliches, herstellerunabhängiges Format. Für die Weiterverwendung der eingelesenen Konfigurationen werden diese einheitlich in einer relationalen Datenbank abgespeichert. Ein weiterer Teil der Software befasst sich mit dem Auswerten von benutzerdefinierten Datenströmen, welche mit den in der Datenbank abgelegten Filterregeln verglichen werden. Dafür wurden Matching-Algorithmen implementiert, welche den Matching-Prozess einer Firewall simulieren.

Die gesamte Software wurde mittels Java 6 implementiert. Für das Einlesen der unterschiedlichen Konfigurationen wurde pro Format jeweils ein eigener Parser implementiert. Bei der Implementation der Parser kam der Scannergenerator JFlex zum Einsatz, der aus einer Grammatik einen Java-Parser generieren kann. Die Persistenzschicht wurde mittels JPA und Hibernate als ORM Framework realisiert. Als relationale Datenbank kommt SQLite zum Einsatz.

Inhaltsverzeichnis

I. Übersicht

1 Einführung

1.1 Zweck

Dieses Dokument beschreibt die an der Hochschule für Technik (HSR) entwickelte Studienarbeit "Firewall-Analyse-Tool" und deren Ergebnisse.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments beschränkt sich auf die Studienarbeit "Firewall-Analyse-Tool", welche im Herbstsemester 2009, vom 14.09.2009 bis am 18.12.2009, durchgeführt wurde.

1.3 Beteiligte Personen

Die Studienarbeit wurde durch die Informatikstudenten Gieri Kohler und Christoph Rebsamen bearbeitet. Betreut wurde die Arbeit durch Roman Ammann vom Institute for Networked Solutions der HSR.

1.4 Übersicht

Im ersten Teil des Dokuments wird die Aufgabenstellung erläutert und eine allgemeine Beschreibung der Arbeit und deren Resultate gegeben. Im zweiten Teil wird auf die technischen Ergebnisse detailliert eingegangen. Im dritten Teil finden sich die die Artefakte des Software-Projekts. Im vierten und letzten Teil befinden sich die persönlichen Berichte und weiterführende Informationen zum Thema.

2 Aufgabenstellung

Heutige Firewallsetups zeichnen sich durch zunehmende Komplexität aus. Netzwerkverkehr muss zwischen dem Absender und dem Empfänger oft mehrere Firewalls passieren. Diese sind manchmal noch von verschiedenen Herstellern und werden zum Teil sogar durch unterschiedliche Personen betreut. Was der Netzwerksicherheit dienlich ist, führt in der Praxis aber zum Problem, dass nicht mehr einfach bestimmt werden kann, ob ein fraglicher Netzwerkstrom zugelassen oder blockiert wird. Das führt zu einem hohen administrativen Aufwand für die Netzwerkadministratoren, welche diese Frage beantworten müssen.

Die Semesterarbeit soll eine Applikation entwicklen, welche diese Aufgabe weitestgehend automatisiert. Der erste Teil der Arbeit befasst sich mit der Aufgabe, Firewallkonfigurationen verschiedener Hersteller einzulesen und in ein herstellerunabhängiges Regelset abzubilden. Die Kernapplikation soll dieses Regelset auf bestimmte Verkehrsmuster überprüfen können.
Der zweite Teil befasst sich mit dem topologischen Setup der verschiedenen Firewalls. Welche Regeln werden an welchen Schnittstellen verwendet, wie stehen die Firewalls zueinander, wie passiert Netzwerkverkehr von einer Quelle zu einem bestimmten Ziel und welche Firewallregeln kommen dabei zur Anwendung. Die topologischen Informationen können aus dem statischen oder dynamischen Routing der Firewalls gezogen werden oder können manuell durch einen Benutzer erfasst werden. Hier ist eine Analyse der Möglichkeiten und Grenzen einer automatischen Erfassung zu erstellen.

Die beiden Teilaufgaben sollen anschliessend zusammengeführt werden und das gesamte Analyse-Tool soll über eine Webschnittstelle abfragbar sein.

Die Arbeit wird in Zusammenarbeit mit der IT der HSR erstellt. Die verwendeten Firewallprodukte und die Anforderungen an die Software werden daher in einem ersten Schritt durch die Architektur der HSR bestimmt. Die Lösung soll aber allgemein verwendbar sein und als Open Source veröffentlicht werden.

Unterschrift des Betreuers:

Roman Ammann

3 Management Summary

3.1 Ausgangslage

Da heutzutage der Anspruch an die Sicherheit ständig wächst, wächst auch die Komplexität von Firewallumgebungen in Unternehmen. Für den zuständigen Techniker wird es ab der stetig wachsenden Anzahl Firewalls in einer Netzwerkumgebung immer schwieriger, verlässliche Aussagen zu treffen, ob ein bestimmter Netzwerkstrom sein Ziel erreicht oder nicht.

Deswegen wurde ein Werkzeug entwickelt, welches es dem Techniker erlaubt, Firewallkonfigurationen verschiedener Hersteller in ein einheitliches Format zu übersetzen. Anschliessend erlaubt es die Software, Netzwerkströme zu spezifizieren und ihren Weg durch eine einzelne Firewall zu simulieren. Schlussendlich lassen sich Aussagen darüber treffen, durch welche Filterregeln der Verkehr erlaubt oder allenfalls blockiert wird.

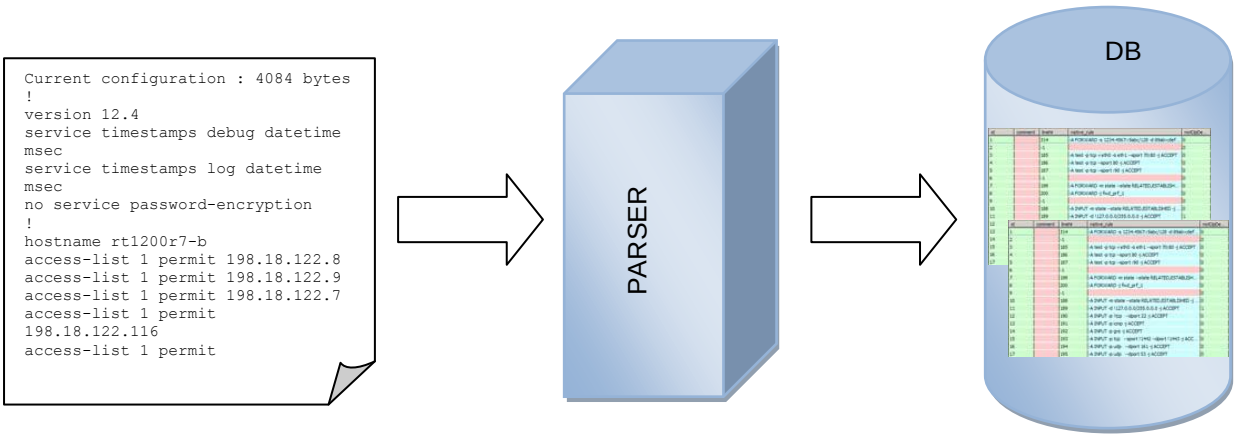
3.2 Vorgehen/Technologien

Die gesamte Software wurde unter Java 6 realisiert. Für die verschiedenen Teilaufgaben wurden einzelne Werkzeuge entwickelt, welche im Folgenden kurz beschrieben werden.

Übersetzungswerkzeug

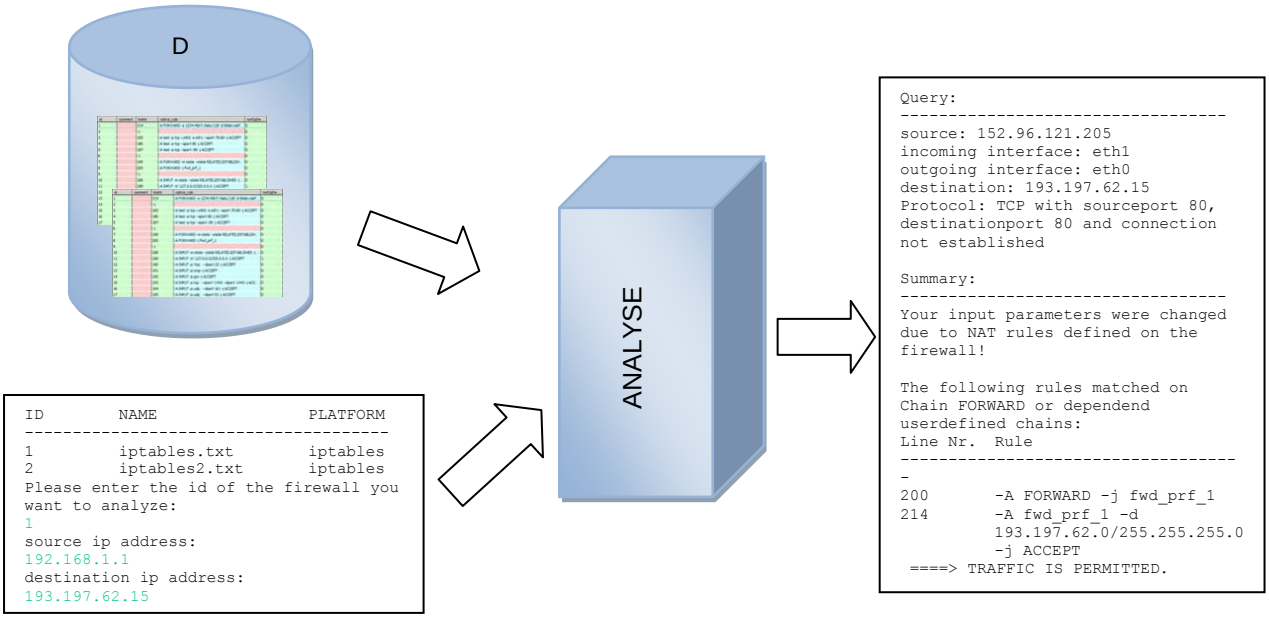
Um Firewallkonfigurationen verschiedener Hersteller, vorliegend als Textdatei, in ein unabhängiges Format innerhalb der Java-Applikation übersetzen zu können, wurde ein Parser für jedes Format entwickelt. Zur Entwicklung der Parser wurde JFlex eingesetzt, eine mächtige Unterstützung für die lexikalische Analyse und das Generieren eines Parsers.

Um die gewonnenen Daten anschliessend weiteren Werkzeugen zur Verfügung zu stellen, werden sie in einer relationalen Datenbank abgespeichert. Hierbei wird auf die Java Persistence API gesetzt, einem Teil der Enterprise JavaBeans 3.0 (EJB 3.0), welche wiederum ein Teil der JavaEE 5 Plattform ist.



Analysewerkzeug

Das Analysewerkzeug besteht aus einem Command Line Interface, über welches Netzwerkströme definiert werden können. Die Software wertet die Eingaben anschliessend gegen die gewünschte Firewallkonfiguration aus, welche aus der relationalen Datenbank geladen wird. Das Analysewerkzeug sucht in der Datenbank nach Filterregeln, welche dem spezifizierten Verkehr entsprechen und präsentiert ein Endergebnis mit den zutreffenden Regeln. Allfällige auf der Firewall konfigurierten NAT-Regeln[1] werden berücksichtigt. Das heisst, die spezifizierten Netzwerkströme werden allenfalls den NAT-Regeln entsprechend angepasst.



3.3 Ergebnisse

Als Ergebnis der Arbeit steht eine Java-Software zur Verfügung, die in der Lage ist, Firewall-konfigurationen von Cisco Geräten unter Cisco IOS[2] Version 12.4 sowie iptables- und ip6tables-Konfigurationen[3] in ein herstellerunabhängiges Format zu übersetzen. Als Eingabe für den Parser (Übersetzer) dient eine Textdatei, die entweder eine running-config eines Cisco IOS Gerätes oder die Ausgabe des iptables-save Kommandos unter Linux beinhalten kann.
Unterstützt wird neben IPv4 auch das in Zukunft immer wichtiger werdende IPv6.

Weiter kann die Software dazu eingesetzt werden, die durch den Parser in das allgemeine Format übersetzten Konfigurationen auszuwerten. Dazu wurden Matching-Algorithmen implementiert, welche der Funktionalität der Original-Firewall entsprechen.

3.4 Ausblick

Die Software erfüllt noch nicht alle gewünschten Funktionen. Folgende Punkte sind noch erstrebenswert.

[1] Network Address Translation: Übersetzung von IP-Adressen oder Ports durch ein Netzwerkgerät, in diesem Fall durch eine Firewall
[2] Internetwork Operating System ist ein Betriebssystem von Cisco-Routern und Switches
[3] Iptables kontrolliert das Paket-Filtering im Linux-Kernel.

GUI

Das GUI (Graphical User Interface) könnte die gesamte Funktionalität des CLI (Command Line Interface) ersetzen. Anstatt einer textbasierten Eingabe würde die Bedienung dann grafisch über ein Webinterface mittels eines Webbrowsers erfolgen.

Topologisches Setup

Die Software wird so ausgebaut, dass ein topologisches Setup mehrerer Firewalls abgespeichert und analysiert werden kann. So könnte Netzwerkverkehr spezifiert werden, der mehrere Firewalls durchläuft. Das Analysewerkzeug würde dann die Konfigurationen aller betroffenen Firewalls auswerten und ein Ergebnis zurückliefern, ob der Verkehr zugelassen wird oder nicht. Eine genaue Auflistung aller angewandten Regeln auf allen passierten Firewalls würde ebenfalls angezeigt.

Weitere Formate

Die Unterstützung weiterer Formate und weiterer Firewallhersteller könnte implementiert werden. Als wichtigster wäre hier Cisco ASA[4] 8.0 zu nennen.

[4] Cisco Adaptive Security Appliance

II. Technischer Bericht

4 Einleitung und Übersicht

4.1 Ähnliche Arbeiten

Eine erwähnenswerte Arbeit ist sicherlich die Software FirewallBuilder [1]. Der FirewallBuilder bietet die Möglichkeit, Firewallregeln in einem allgemeinen Format zu definieren und diese dann in ein spezifisches Format zu übersetzen. Die Software unterstützt Formate wie IOS ACL, ipfilter, iptables, PIX und mehr. Die Funktionalität des FirewallBuilder ist sozusagen genau die Umkehrung der Funktionalität des Parsers im Firewall-Analyse-Tool.

Der FirewallBuilder hatte einen Einfluss auf unsere Arbeit, da das allgemeine Format der Firewallregeln nicht neu erfunden werden musste. Nachdem das vom FirewallBuilder verwendete Format gründlich untersucht worden ist, stand fest, dass grosse Teile davon in gleicher oder ähnlicher Weise verwendet werden können.

4.2 Übersicht

Das Firewall-Analyse-Tool besteht hauptsächlich aus drei unterschiedlich umfangreichen Werkzeugen, welche alle über ein Command Line Interface bedient werden:

4.2.1 Übersetzungswerkzeug

Das Übersetzungswerkzeug besteht aus je einem Parser für Cisco IOS und iptables bzw. ip6tables. Für die Entwicklung der Parser wurde JFlex verwendet. Bei JFlex handelt es sich um einen Parsergenerator, der aus einer Grammatik einen Parser in Java generieren kann. Die Funktionsweise von JFlex wird im Abschnitt 5.1 unter anderem auch durch ein Beispiel beschrieben.
Der Funktionsumfang des Parsers ist ein Thema im Abschnitt 6.2 (Cisco) und 6.3 (iptables)

Das Übersetzungswerkzeug speichert die durch die Parser übersetzten Konfigurationen in einer Datenbank. Hierbei wird die Java Persistence API eingesetzt, ein Teil der Enterprise JavaBeans 3.0 (EJB 3.0), welche wiederum ein Teil der JavaEE 5 Plattform ist. Die Umsetzung der Persistenz wird in den Abschnitten 5.2 und 5.3 beschrieben.

4.2.2 Analysewerkzeug

Das Analysewerkzeug erlaubt es dem Benutzer, einen Netzwerkstrom zu spezifizieren, der über eine Firewall laufen soll. Der spezifizierte Netzwerkstrom wird gegen die bereits vorhandenen Daten in der Datenbank verglichen und ein Ergebnis wird ausgegeben. Das Ergebnis beinhaltet, ob der Datenstrom zugelassen oder blockiert wird und durch welche Regeln diese Entscheidung zustande kommt, sowie weitere interessante Details. Für das Analysewerkzeug wurden komplexe Matching-Algorithmen implementiert, welche in Abschnitt 6.5 beschrieben werden.

4.2.3 Datenbankverwaltung

Das Werkzeug für die Datenbankverwaltung ist das kleinste der drei Werkzeuge. Es wird im Abschnitt 6.4.3 bei der Beschreibung aller CLI's nochmals etwas genauer erklärt.

5 Technologien

5.1 JFlex - The Fast Scanner Generator for Java[5]

Jflex ist ein Hilfsmittel, um Scanner zu generieren, welche eine bestimmte Grammatik erkennen. JFlex wurde für das Projekt Firewall-Analyse-Tool verwendet, um eine running-config eines Cisco-Geräts oder die Ausgabe eines iptables-save Befehls unter Linux zu verarbeiten. Die Grammatik wird innerhalb einer Datei mit der Endung .jflex beschrieben. Aus dieser Datei lässt sich mit JFlex automatisch eine Java-Klasse generieren, welche eine Inputdatei als Parameter erwartet.

Eine *.jflex Datei ist in drei Sektionen unterteilt, welche durch Zeilen mit dem Inhalt „%%“ abgetrennt werden.

5.1.1 Erste Sektion: Code to include

Der Inhalt dieser Sektion wird unverändert in die generierte *.java Datei kopiert, genau genommen wird er direkt vor der eigentlichen Deklaration der generierten Lexerklasse eingefügt. Ausser package- und import-Anweisungen kommt hier in der Regel nichts rein.

5.1.2 Zweite Sektion: Options and Macros

Diese Sektion kann

- eine Menge von Optionen
- Code, der in die generierte Scannerklasse inkludiert wird
- Code, der in den Konstruktor der generierten Klasse inkludiert wird
- lexikalische Zustände
- Macrodeklarationen

beinhalten.

Wichtige Optionen:

`%class Lexer`
weist JFlex an, die generierte Klasse „Lexer“ zu nennen.

`%public`
Die generierte Klasse wird public.

`%standalone`
erstellt eine main-Funktion in der generierten Klasse, welche den Namen einer Inputdatei auf der Kommandozeile erwartet und dann den Scanner auf dieser Inputdatei ausführt.

Diese Liste ist keinesfalls vollständig. Die Auswahl zeigt nur ein paar Beispiele. Für weitere Informationen gibt es eine ausführliche Dokumentation von JFlex auf http://jflex.de/manual.html.

Code, der in die generierte Klasse inkludiert wird

Dieser Code muss durch `%{` und `%}` eingeschlossen werden.

Code, der in den Konstruktor der generierten Klasse inkludiert wird

Dieser Code muss durch `%init{` und `%init}` eingeschlossen werden.

[5] Quelle: [2]

Lexikalische Zustände

Ein lexikalischer Zustand wird definiert durch `%state` gefolgt von seinem Namen, z.B.
```
%state YYINITIAL
```

Macros

Ein Macro wird durch einen regulären Ausdruck definiert, z.B.
```
LineTerminator = \r|\n|\r\n
```

5.1.3 Dritte Sektion: Rules and Actions

Diese Sektion beinhaltet reguläre Ausdrücke und Aktionen (Javacode), der ausgeführt wird, wenn der Scanner den dazugehörigen regulären Ausdruck liest. Macros aus der zweiten Sektion können hier verwendet werden. Sie müssen dazu in `{` und `}` eingeschlossen werden. Folgender Code gibt beispielsweise einen Text aus, wenn ein Zeilenumbruch gelesen wird.
```
{LineTerminator}        { System.out.println("Zeilenumbruch gelesen"); }
```

Es gibt auch die Möglichkeit, reguläre Ausdrücke nur in bestimmten Zuständen abzufangen und in anderen zu ignorieren. Die Zustände werden in spitze Klammern gesetzt. Der betreffende Zustand muss jedoch in der zweiten Sektion definiert werden.
```
<YYINITIAL> {
{LineTerminator}        { System.out.println("Zeilenumbruch in Zustand
                        YYINITIAL gelesen"); }
}
```

Zustandswechsel erfolgen durch den Javacode
```
yybegin(ZustandsName);
```

Den aktuell getroffenen regulären Ausdruck erhält man durch den Javacode
```
yytext();
```

5.1.4 Beispiel

Der folgende Code zeigt einen kleinen Auszug aus dem Parser für Cisco IOS running-configs in vereinfacher Form und stellt an einem Beispiel dar, wie bei der Implementierung des Parsers vorgegangen wurde. Die Erklärung folgt nach dem Code.

Datei ios.jflex:
```
1    package ch.hsr.fwtool.parser;
2    import ch.hsr.fwtool.pd.*;
3    import ch.hsr.fwtool.parser.helper.cisco.AclInformation.ACLType;
4    import ch.hsr.fwtool.parser.helper.cisco.*;
5    import org.apache.log4j.*;
6
7    %% /*section 2 begins here*/
8
9    %public
10   %class ParserIOS
11   %standalone
12   %unicode
13
14   %init{
15     yybegin(YYINITIAL);
16     fw = new Firewall();
```

```
17      logger = Logger.getRootLogger();
18      logger.setLevel(Level.INFO);
19      try {
20        logger.addAppender(new FileAppender(new SimpleLayout(),
21          "logfile"));
22      } catch (IOException e) {
23        System.err.println("error adding FileAppender");
24        e.printStackTrace();
25      }
26    %init}
27
28    %{
29    private AclInformation aclInf;
30    private InterfaceInformation ifaceInf;
31
32    //returns the firewall object with all attached information
33    public Firewall getFirewall() {
34      return fw;
35    }
36
37    //add rule to ruleset, if ruleset doesn't exist, create it first
38    private void addIPv4ACL() {
39      addAccessPolicyRuleSet(aclInf.getName(), RuleSetFamily.IPV4);
40      addIPv4AccessPolicyRule(aclInf.getName());
41    }
42
43    //add an AccessPolicyRuleSet to the firewall
44    private void addAccessPolicyRuleSet(String aclName,
45      RuleSetFamily family) {
46      //code to create and add the access policy ruleset
47    }
48
49    //add an ipv4 access policy rule to an access policy ruleset
50    private void addIPv4AccessPolicyRule(String aclName) {
51      //code to create and add the access policy rule
52    }
53
54    //add interface to the firewall
55    private void addInterface() {
56      //code to create and add the interface
57    }
58    %}
59
60    //states
61    %state YYINITIAL
62    %state ACL
63    %state STDACLSRCADR
64    %state IFACE
65
66    //macros
67    stdAclNr = ([1-9][0-9]?)|(1[3-9][0-9][0-9])
68    extAclNr = (1[0-9][0-9])|(2[0-6][0-9][0-9])
69    aclNr = {stdAclNr}|{extAclNr}
70    whitespace = {lineTerminator} | [ \t\f]
71    whitespaces = {whitespace}*
```

```
72  permitOrDeny = permit|deny
73
74  %% /*section 3 begins here*/
75
76  <YYINITIAL> {
77    "access-list "
78    +{aclNr}+" "       { yybegin(ACL);
79                         String[] s = yytext().split(" ");
80                         aclInf = new AclInformation(s[1],
81                         ACLType.NUMBERED); }
82
83    "interface "
84    +{interfaceName}   { yybegin(IFACE);
85                         ifaceInf = new InterfaceInformation();
86                         ifaceInf.setName(yytext().substring(10));
87                         addInterface(); }
88
89    .                  { /*ignore unknown*/ }
90  }
91
92  <ACL> {
93    {permitOrDeny}
94    +" "               { yybegin(STDACLSRCADR);
95                         String[] s = yytext().split(" ");
96                         aclInf.setPermitOrDeny(s[0]);
97                         aclInf.setProtocol("IP");
98                         aclInf.setIPProtocolName("IP"); }
99
100   {whitespace}       { /*ignore*/ }
101
102   .                  { logger.error("<ACL> unknown input: "+yytext());}
103 }
104
105 <STDACLSRCADR> {
106   {ip}+" "
107   +{wildcard}        { yybegin(YYINITIAL);
108                        String[] s = yytext().split(" ");
109                        aclInf.setSourceIP(s[0]);
110                        aclInf.setSourceWildcard(s[1]);
111                        aclInf.setDestinationIP("0.0.0.0");
112                        aclInf.setDestinationWildcard("255.255.255.255");
113                        addIPv4ACL(); }
114
115   {ip}               { yybegin(YYINITIAL);
116                        aclInf.setSourceIP(yytext());
117                        aclInf.setSourceWildcard("0.0.0.0");
118                        aclInf.setDestinationIP("0.0.0.0");
119                        aclInf.setDestinationWildcard("255.255.255.255.");
120                        addIPv4ACL(); }
121
122   "host "+{ip}       { yybegin((YYINITIAL);
123                        aclInf.setSourceIP(yytext().substring(5));
124                        aclInf.setSourceWildcard("0.0.0.0");
125                        aclInf.setDestinationIP("0.0.0.0");
126                        aclInf.setDestinationWildcard("255.255.255.255");
```

```
127                     addIPv4ACL(); }
128
129    "any"            { yybegin(YYINITIAL);
130                     aclInf.setSourceIP("0.0.0.0");
131                     aclInf.setSourceWildcard("255.255.255.255");
132                     aclInf.setDestinationIP("0.0.0.0");
133                     aclInf.setDestinationWildcard("255.255.255.255");
134                     addIPv4ACL(); }
135
136    " "              { /*ignore*/ }
137
138    .                logger.error("<STDACLSRCADR>: unknown input "
139                     +yytext()); }
140  }
141
142  <IFACE> {
143    "ip address "
144    +{ip}+" "
145    +{netmask}       { String[] s = yytext().split(" ");
146                     ifaceInf.setIPAddress(s[2]);
147                     ifaceInf.setNetMask(s[3]);
148                     addIpv4ToInterface(); }
149
150    .                { /*ignore*/ }
151  }
```

Das gezeigte Beispiel ist vereinfacht und unvollständig. Es parst nummerierte Standard-ACLs der folgenden Form:

```
access-list access-list-number {permit|deny}
{host|source source-wildcard|any}
```

Die Funktionsweise des Parsers lässt sich folgendermassen zusammenfassen: Im Konstruktor des Parsers wird ein Firewallobjekt der Problem Domain erstellt. An dieses Objekt werden während des Parsingvorgangs die gelesenen Interfaces, ACLs sowie anderen relevanten Informationen angehängt. Am Ende des Parsingvorgangs kann dann das vollständig erstellte Firewall-Objekt über die Methode `getFirewall()` abgeholt werden. Ebenfalls besitzt der Parser jeweils eine Referenz auf die Klassen `AclInformation` sowie `InterfaceInformation`. Diese dienen zur Zwischenspeicherung der gelesenen Parameter, bevor diese endgültig an das Firewallobjekt angehängt werden. Eine Regel einer ACL wird beispielsweise immer dann an das Firewallobjekt angehängt, wenn diese komplett eingelesen wurde. Die Klasse `AclInformation` enthält also immer nur die Informationen über die Regel einer ACL, welche im Moment gerade eingelesen wird. Wurde die Regel komplett eingelesen, wird über eine Hilfsfunktion eine AccessPolicyRule erstellt und am Firewallobjekt angehängt.

Anhand des vorangegangenen Beispielcodes wird dieser Vorgang aufgezeigt. Es wird folgende ACL als Input angenommen:
`access-list 1 deny host 172.16.3.10`

Im Init-Bereich, welcher in der generierten Scannerklasse in den Konstruktor eingefügt wird, wird das Firewallobjekt erzeugt (Zeile 16). Weiter wird in den Initialzustand `<YYINITIAL>` gewechselt (Zeile 15). Der Parsingvorgang startet also in diesem Zustand (Zeile 76). Der Input wird nun zeichenweise gelesen und mit den regulären Ausdrücken auf der linken Seite der JFlex-Datei verglichen. Mit obigem Input trifft der erste reguläre Ausdruck `"access-list " +{aclNr}+" "` auf die Eingabe zu, wobei die Zahl 1 mit dem regulären Ausdruck `{aclNr}` übereinstimmt. Der passende Javacode wird ausgeführt (Zeilen 78-81). Konkret wird ein neues Objekt der Klasse `AclInformation` erstellt und der ACL-Name (1) und

der Typ der ACL (Numbered) darin zwischengespeichert. Zusätzlich wird in den Zustand <ACL> (Zeile 92) gewechselt. Hier stimmt der reguläre Ausdruck {permitOrDeny}+" " mit der weiteren Eingabe überein. Im zugehören Javacode wird in das Objekt der Klasse AclInformation die Aktion (deny) und das Protokoll (IP) zwischengespeichert. Es wird in den Zustand <STDACLSRCADR> (Zeile 105) gewechselt. Hier wird die Sourceadresse erwartet, die in vier unterschiedlichen Varianten angegeben sein kann. Mit dem gegebenen Input trifft der reguläre Ausdruck "host "+{ip} auf die Eingabe. Die restlichen Parameter wie SourceIP/Wildcard und DestinationIP/Wildcard werden zwischengespeichert. Danach wird die Hilfsmethode addIPv4ACL() aufgerufen (Zeile 127). Diese Methode holt sich die Parameter aus dem Objekt der Klasse AclInformation und erstellt eine neue AccessPolicyRule der Problem Domain anhand dieser Parameter. Danach wird diese AccessPolicyRule dem Firewallobjekt hinzugefügt.

Logging mit log4j [3]

Wie im Beispiel zu sehen ist, wird log4j für das Logging verwendet (Zeilen 5, 17-25). In den jeweiligen lexikalischen Zuständen wird wenn möglich eine ungültige Eingabe abgefangen. Der Punkt ist ein regulärer Ausdruck, der für ein beliebiges Zeichen steht. Wird ein unbekanntes Zeichen abgefangen, werden der aktuelle jeweilige Zustand sowie die aktuelle Eingabe in ein Logfile gespeichert. Beispiele finden sich in der Zeile 101 für den Zustand <ACL> und in der Zeile 137 für den Zustand <STDACLSRCADR>. In einigen Zuständen, wie beispielsweise im initialen Zustand <YYINITAL> oder im Zustand <IFACE> kann diese Fehlerüberprüfung jedoch nicht durchgeführt werden. Der Grund ist, dass in einer running-config nicht alle Informationen relevant sind für das Firewall-Analyse-Tool und daher unbekannter Input ignoriert wird. Es kann daher nicht entschieden werden, ob der Input tatsächlich ungültig ist oder vom Parser nur als nicht gültig erkannt wird.

5.2 SQLite[6]

SQLite ist eine Bibliothek, welche ein relationales Datenbanksystem beinhaltet. Im Gegensatz zu anderen relationalen Datenbanken benötigt SQLite keinen Datenbank-Server, was vor allem zu einer grossen Verbreitung im Embedded-Bereich geführt hat. SQLite unterstützt einen grossen Teil des SQL-92-Standards.

Der Entscheid für den Einsatz von SQLite im Firewall-Analyse-Tool anstatt einer anderen relationalen Datenbank wurde aufgrund der gestellten Anforderung einer einfachen Verbreitung des Tools getroffen. Durch den Einsatz von SQLite muss der Benutzer keinen Datenbank-Server betreiben und gegen eine Datenhaltung in Files (z.B. im XML-Format) sprechen die Features, welche eine relationale Datenbank bietet.

Obwohl SQLite weitverbreitet ist, wird sie von Java nicht wirklich unterstützt. Es gibt keinen offiziellen JDBC-Treiber und auch keine Unterstützung durch JPA/Hibernate. Glücklicherweise finden sich im Internet sowohl eine Implementation für den JDBC-Treiber (http://www.zentus.com/sqlitejdbc/) wie auch ein Projekt, welches sich der Kooperation von Hibernate mit SQLite widmet (http://code.google.com/p/hibernate-sqlite/).

5.2.1 Limitierungen

Da SQLite relativ leichtgewichtig ist, hat es leider einige Limitierungen. Im Folgenden werden die für das Projekt relevanten Limitierungen aufgelistet und erklärt.

SQLite unterstützt maximal 64 Tabellen in einem JOIN-Statement. Das heisst, in einem einzigen SELECT-Statement können nicht mehr als 64 JOIN-Statements verwendet werden. Dies führte anfangs zu einem Problem, als über JPA eine gesamte Firewall inklusive der gesamten Konfiguration ausgelesen werden sollte.

[6] Quelle: [4]

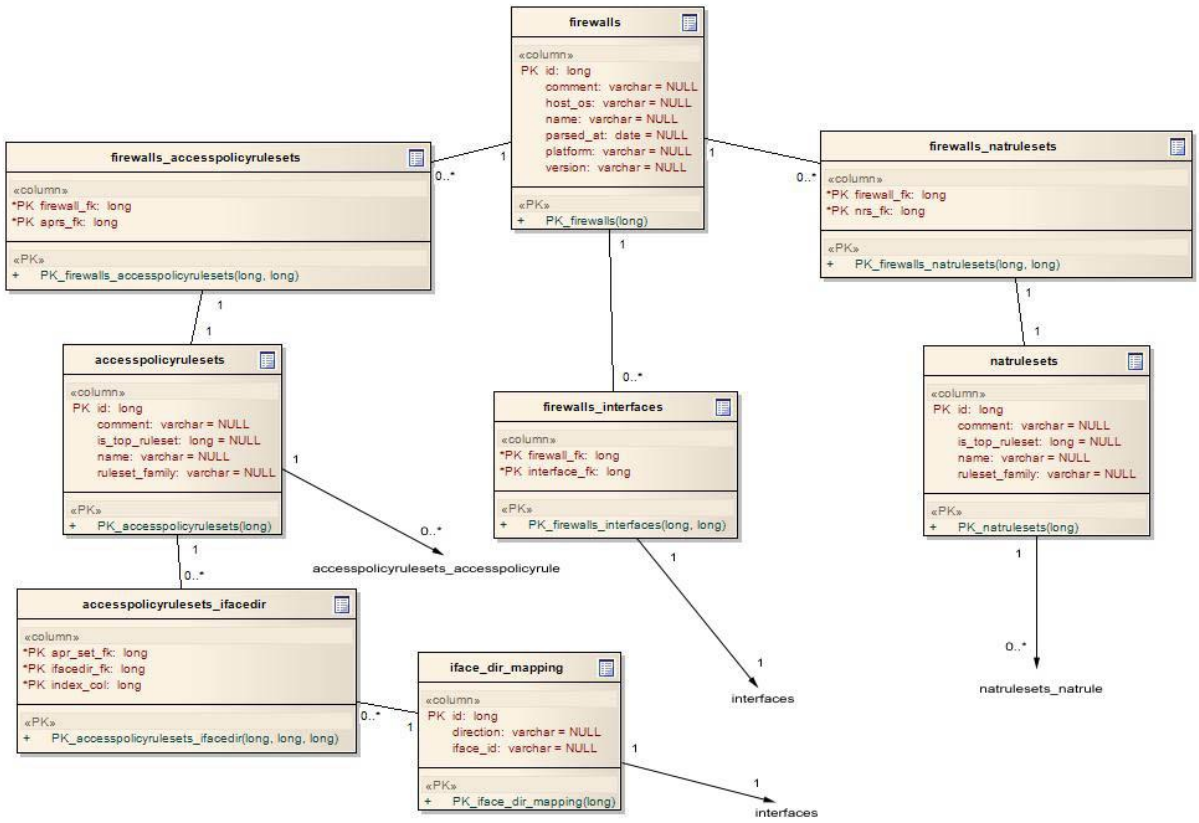
SQLite unterstützt keine Fremdschlüssel. Dies ist grundsätzlich mit der Verwendung von JPA/Hibernate kein Problem, kann aber zu ungewolltem Verhalten führen. So wäre es z.B. möglich, über ein DELETE-Statement eine Firewall zu löschen und alle angehängten Konfigurationen bleiben bestehen, da keine Fremdschlüssel die referentielle Integrität sicherstellen.

5.2.2 Struktur der fwtool-Datenbank

Die Datenbank, welche für das fwtool verwendet wurde, besteht aus insgesamt 27 Tabellen. Da das Schema für eine einzelne Grafik zu gross und unübersichtlich wurde, ist die Grafik in mehrere Teile zerlegt worden.

Jede Tabelle, welche eine Klasse aus der Problem Domain der fwtool-Applikation repräsentiert, besitzt eine eindeutige ID, welche über die SQLite Autoindex-Funktion erstellt werden. Diese ID's werden als Primärschlüssel für die einzelnen Datensätze verwendet.

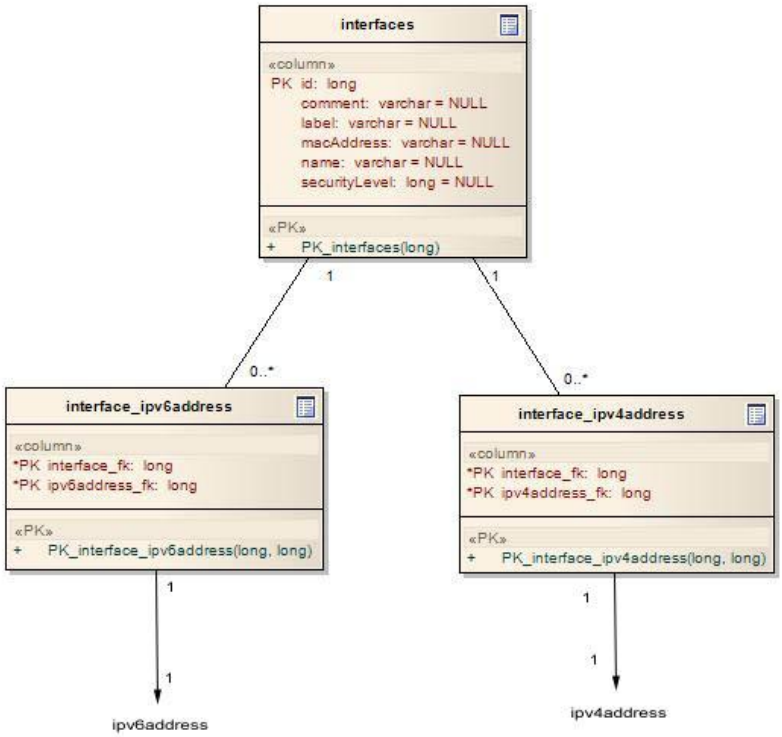
Die Wahl der Datenbank-Datentypen wurde durch JPA/Hibernate, das verwendete ORM-Framework, welches im folgenden Kapitel genauer beschrieben wird, getroffen. Zu beachten ist, dass boolean-Werte als long mit dem Wert 0 oder 1 abgelegt werden.



Datenbankdiagramm Teil Firewall

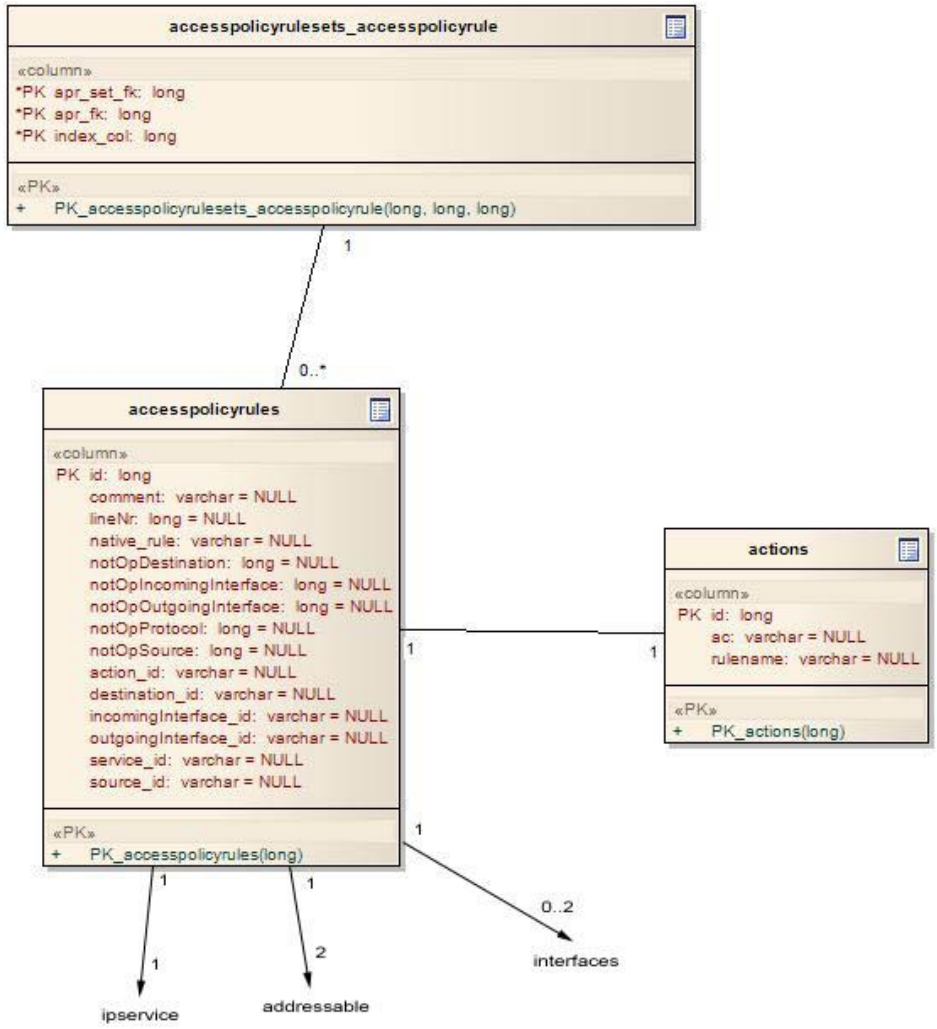
Dieser Teil des Diagramms beschreibt die Tabellen, in welchen die Daten zur Firewall, den definierten NAT-Regelsets und den Access Policy Regelsets abgelegt werden. In der Tabelle "iface_dir_mapping" wird zudem festgelegt, welche Access Policy Regeln auf welches Interface und in welche Richtung angewendet werden. Diese Tabelle wurde parallel zur entsprechenden Klasse in der Problem Domain für die Repräsentation der Zuordnung von Regelsets auf Interfaces bei Cisco IOS eingeführt.

Die restlichen ersichtlichen Tabellen sind Zuordnungs-Tabellen, welche 1:m und m:n-Beziehungen von Entitäten auflösen. Der Grund, warum teilweise auch 1:m-Beziehungen über eine solche Tabelle aufgelöst wurden, ist eine bessere Lesbarkeit der Datenbank.



Datenbankdiagramm Teil Interfaces

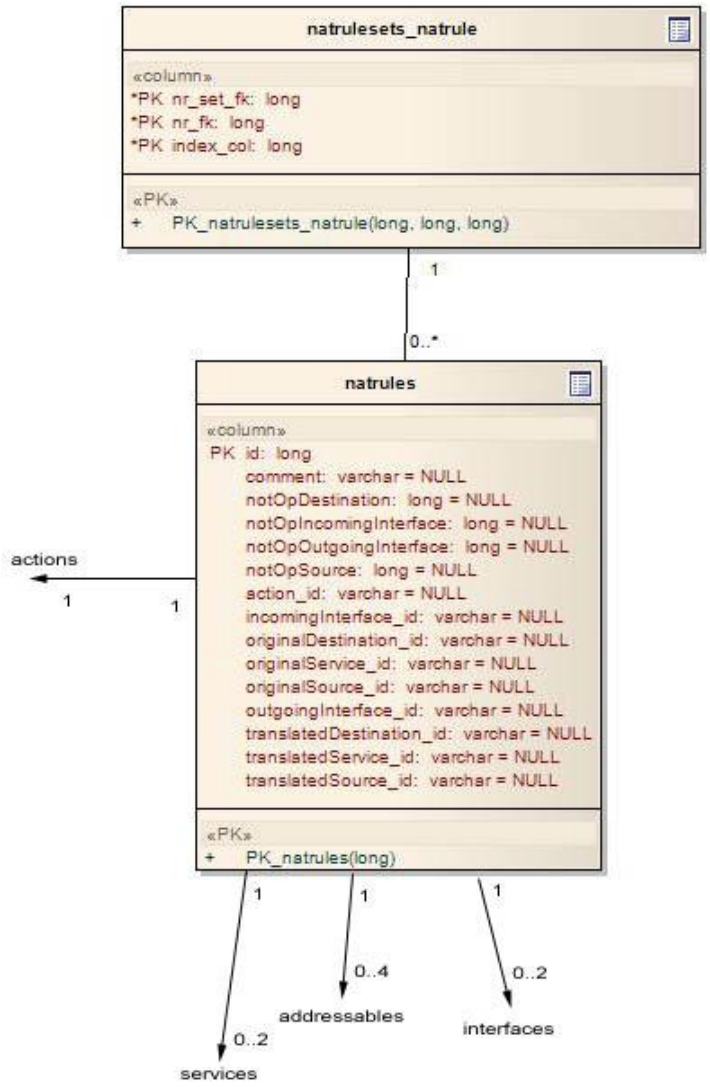
Dieser Teil des Diagramms beschreibt die Tabelle "interfaces", in welcher die Informationen über die Interfaces der Firewall abgelegt sind. Die beiden anderen Tabellen sind die Mapping-Tabellen auf die dem Interface zugeordneten IP-Adressen.



Datenbankdiagramm Teil Access Policy Rule

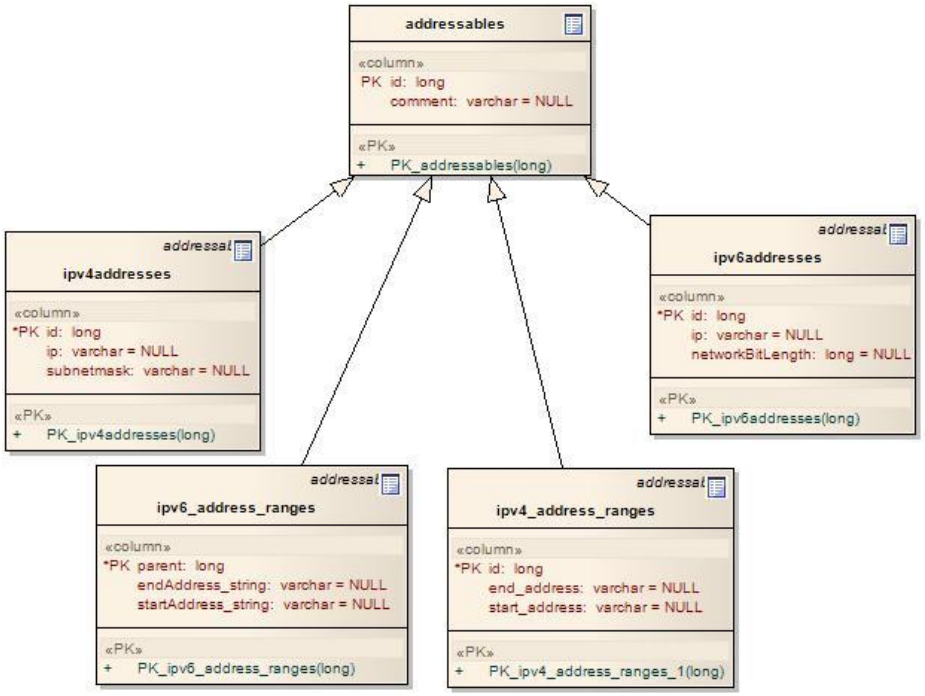
Im obigen Diagramm ist die Struktur der Tabellen "accesspolicyrules" und "actions" ersichtlich. Sie repräsentieren die Klassen `AccessPolicyRule` und `Action` aus der Problem Domain der Applikation.

Interessant hier sind die Beziehungen zur Tabelle "addressables" und zur Tabelle "interfaces". Da es bei iptables die Möglichkeit gibt, einer Access Policy Regel ein Source- und/oder ein Destination-Interface anzugeben, besteht eine Beziehung zwischen der Regel und dem Interface der Firewall. Falls die Access Policy Regel auf einem Cisco IOS Gerät definiert wurde, wird diese Beziehung nie aktiv sein. Die Kardinalität der Beziehung zur Tabelle "addressables" lässt sich auch genau beziffern, da jede Access Policy Regel genau ein Source-Addressable und genau ein Destination-Addressable besitzt.



Datenbankdiagramm Teil NAT Rule

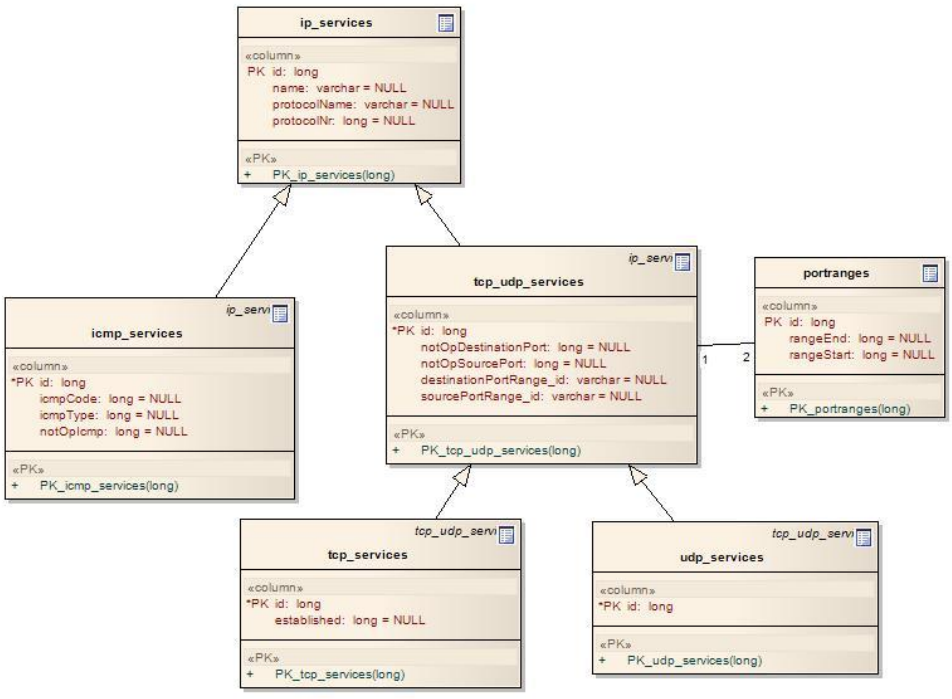
Die Tabelle "natrules" wird dafür verwendet, Objekte vom Typ `NatRule` zu persistieren. Eine NAT-Regel hat bis zu 4 angehängte Addressables, nämlich die Originalsource und die Originaldestination sowie die beiden übersetzten resp. veränderten Addressables. Eine NAT-Regel kann, muss aber nicht einem Interface zugeordnet sind und kann sowohl auf der Incoming- wie auch auf der Outgoing-Seite definiert sein. Zu guter Letzt sind der Regel auch noch ein Original-Service (z.B. TCP-Service auf Port 80) sowie ein übersetzter Service zugeordnet.



Datenbankdiagramm Teil Addressables

Dieser Teil des Diagramms zeigt das Addressable-Konstrukt. Da das fwtool mit IPv4-Addressen wie auch mit IPv6-Addressen umgehen kann, braucht es in der Problem-Domain eine gemeinsame Oberklasse, nämlich die Klasse `Addressable`. Diese deckt zudem die Möglichkeit ab, dass als Source wie auch als Destination nicht nur eine spezifische IP-Adresse, also z.B. 192.168.1.10/32, oder ein Netz, also z.B. 192.168.1.0/24, sein kann, sondern auch ein Adressbereich mit Start- und End-Adresse.

Bei der Abbildung der Vererbung wurde für jede Super- und Subklasse eine eigene Tabelle erstellt. Damit wird Redundanzfreiheit erreicht und weil dabei nicht die 3. Normalform verletzt wird, ist es die "reinste" Lösung für das Problem der Vererbung.



Datenbankdiagramm Teil Services

Der letzte Teil des Diagramms zeigt die verschiedenen Tabellen welche für die verschiedenen Services angelegt wurden. Auch hier zeigt sich eine Vererbungshierarchie, welche mittels der Methode "Eine Tabelle je Super- sowie Subklasse" auf das relationale Modell abgebildet wurde.

5.3 JPA/Hibernate[7]

JPA sowie Hibernate sind Persistenz-Provider für Java, welche das Mapping von Java-Klassen auf eine relationale Datenbank erleichtern. JPA ist dabei eine Schnittstelle, welche von einer Java Expert Group entwickelt wurde und im Jahre 2006 erstmals veröffentlicht wurde. Hibernate ist ein Framework, welches von JBoss entwickelt wurde und dessen Hauptaufgabe das Object-Relational-Mapping ist. Der Grund, warum neben JPA Hibernate und nicht die Referenz-Implementation Toplink Essentials eingesetzt wurde, ist der, dass Hibernate gewisse Funktionen speziell im Umgang mit Listen bietet, die für dieses Projekt essentiell sind. Zudem wurde im parallel zur Studienarbeit besuchten Modul "Enterprise Computing" ebenfalls mit dieser Kombination gearbeitet.

5.3.1 Umsetzung ORM im fwtool

Um die für das ORM nötigen Meta-Informationen den Klassen hinzuzufügen, gibt es die Möglichkeit, entweder mit Java-Annotationen zu arbeiten oder die Meta-Informationen über ein zusätzliches XML-File hinzuzufügen.

Im Firewall-Analyse-Tool wird die Möglichkeit genutzt, diese Informationen per Annotation hinzuzufügen. So können die relevanten Informationen gleich bei der betreffenden Klasse und den betreffenden Attributen gefunden werden und müssen nicht in einem separaten File gesucht werden. Für dieses Projekt sollten nur Annotationen aus dem JPA-Namespace verwendet werden. Da aber JPA im Bereich der Abbildung von Listen noch gewisse Defizite hat, waren wir gezwungen, an wenigen Orten auf Hibernate-Annotationen zurückzugreifen.

Als Beispiel für die Umsetzung des ORM soll die Klasse `Firewall` dienen:

```
@Entity
@Table(name = "firewalls")
public class Firewall extends AbstractEntity {

   @Column
   private String comment;
   @Column(name="host_os")
   private String hostOs;
   @Column
   private String name;
   @Column
   private String platform;
   @Column
   private String version;
   @Column(name="parsed_at")
   @Temporal(TemporalType.DATE)
   private Date parsedAt;

     @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
     @JoinTable(name = "firewalls_interfaces",
        joinColumns = {@JoinColumn(name = "firewall_fk")},
        inverseJoinColumns = {@JoinColumn(name = "interface_fk")})
     @MapKey(name="name")
   private Map<String, Interface> interfaces;

     @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.LAZY)
     @JoinTable(name = "firewalls_accesspolicyrulesets",
```

[7] Quelle: [5]

```
      joinColumns = {@JoinColumn(name = "firewall_fk")},
      inverseJoinColumns = {@JoinColumn(name = "aprs_fk")})
    @MapKey(name="name")
  private Map<String, AccessPolicyRuleSet> accessPolicyRuleSets;

    @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.LAZY)
    @JoinTable(name = "firewalls_natrulesets",
      joinColumns = {@JoinColumn(name = "firewall_fk")},
      inverseJoinColumns = {@JoinColumn(name = "nrs_fk")})
    @MapKey(name="name")
  private Map<String, NatRuleSet> natRuleSets;
  ...
}
```

Mit `@Entity` wird festgelegt, dass diese Klasse ein Persistenz Domain Objekt ist. In der Regel entspricht ein solches Objekt einer Tabelle in der Datenbank. Um den Namen der Tabelle selber festzulegen wurde mittels der Annotation `@Table(name = "firewalls")` der Tabellenname händisch definiert.

Mit `@Column` wird festgelegt, welche Attribute eine eigene Spalte in der Datenbank erhalten. Es kann dabei ähnlich wie bei `@Table` ein Name angegeben werden. Wird dies unterlassen, benutzt das Persistenz-Framework den Attribut-Namen als Spaltenname. Die `@Temporal`-Annotation teilt dem Persistenz-Framework mit, dass das folgende Attribut als Datum, als Timestamp oder als Zeit behandelt werden soll.

Der wichtigste und auch herausforderndste Punkt war die Definition der Beziehungen zwischen den Klassen. Diese Definitionen geschehen mit den Annotationen `@OneToOne`, `@OneToMany`, `@ManyToOne` oder `@ManyToMany`. Mittels `cascade` wird festgelegt, welche Operationen auf die untergeordneten Entities kaskadiert werden. In unserem Fall werden jegliche Operationen weitergegeben. Mit `fetch` wird angegeben, ob die untergeordneten Entities zusammen mit dem übergeordneten Element geladen werden sollen (`FetchType.EAGER`) oder ob die untergeordneten Elemente erst beim ersten Zugriff nachgeladen werden (`FetchType.LAZY`). Da Usus ist, alle Daten in einem Datenbankzugriff zu laden, es sei denn, man muss grössere Datenmengen (BLOB/CLOB) oder riesige Strings laden, haben wir zu Beginn alle unsere angehängten Entitäten mit `EAGER` geladen. Leider generierte Hibernate darauf ein SELECT-Statement welches mehr als 64 JOIN's enthielt. Da sqlite wie oben besprochen auf max. 64 JOIN's pro SELECT-Statement limitiert ist, waren wir gezwungen, die Access Policy Regelsets und die NAT Regelsets mittels Lazy-Loading zu laden um die Menge an JOIN's signifikant zu reduzieren.

Mittels der Annotation `@JoinTable` kann eine Beziehungstabelle zwischen den Entitäten erstellt werden. Dies ist bei Many-to-Many-Beziehungen Pflicht und kann für eine bessere Lesbarkeit auch bei One-to-Many- oder sogar bei One-to-One-Beziehungen eingesetzt werden.

Mittels `@MapKey` wird festgelegt, welches Attribut der untergeordneten Klasse als Map-Key dient.

AbstractEntity

Damit eine Entität persistiert werden kann muss sie zwingend ein mit `@Id` annotiertes Attribut besitzen. Dieses Attribut wird in der Tabelle der Datenbank zum Primary Key. Um diese Id nicht in jeder Klasse der Problem-Domain implementieren zu müssen, wurde eine `AbstractEntity` genannte Klasse geschaffen, welche nur das Attribut id besitzt. Alle Klassen, welche persistiert werden sollen, müssen demnach von dieser Klasse erben. Damit die Informationen dieser Klasse zu den einzelnen definierten Klassen gepackt werden, wird diese AbstractEntity-Klasse mit der Annotation `@MappedSuperclass` versehen.

```
@MappedSuperclass
public abstract class AbstractEntity {

   @Id
   @GeneratedValue
   private long id;
   ...
}
```

Mittels der Annotation `@GeneratedValue` kann dem Persistenz-Manager und in letzter Konsequenz der Datenbank die Verantwortung für die Generierung der Surrogatschlüssel übergeben werden.

Hibernate-Annotation für Listen

Da bei den Access Policy Regeln in den Sets die Reihenfolge der Regeln von grösster Wichtigkeit für den späteren Matching-Prozess ist, wird in der Datenbank zusätzlich der Index der Regel im Set abgelegt. Damit wird sichergestellt, dass beim Laden der Daten aus der Datenbank die Reihenfolge der Regeln wieder exakt der Reihenfolge entspricht, in der sie zu Beginn aus dem Konfigurations-File geparst und anschliessend persistiert wurden.

Da JPA leider diese Möglichkeit nicht vorsieht, musste in diesem speziellen Fall auf die Hibernate-Annotation `@IndexColumn` aus dem Package `org.hibernate.annotations` zurückgegriffen werden.

Beispiel aus der Klasse `AccessPolicyRule`:

```
   ...
@OneToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
   @JoinTable(name = "accesspolicyrulesets_accesspolicyrule",
       joinColumns = {@JoinColumn(name = "apr_set_fk")},
       inverseJoinColumns = {@JoinColumn(name = "apr_fk")})
   @IndexColumn(name="index_col")
   private List<AccessPolicyRule> rules;
   ...
```

Vererbung

Auch das korrekte Handling der Vererbungshierarchie aus der Problem Domain in eine Tabellen-Struktur in der Datenbank kann über Annotations an den Klassen gemacht werden. Die dafür relevante Annotation ist die `@Inheritance`-Annotation. Sie wird an der Vater-Klasse angebracht. Mittels dem Enum `InheritanceType` wird festgelegt nach welcher Strategie die Vererbung auf das relationale Modell übertragen wird. Wie bereits im Kapitel zu SQLite beschrieben, wurde im fwtool auf die Strategie `JOINED` gesetzt, welche für jede Super- wie Subklasse eine eigene Tabelle erstellt.

Beispiel aus der Klasse `IPService`:

```
@Entity
@Table ( name = "ip_services" )
@Inheritance(strategy=InheritanceType.JOINED)
public class IPService extends AbstractEntity {

...
}
```

Persistence Context und EntityManager

Persistence Context wird der Kontext genannt, welcher die verschiedenen Entitäts-Instanzen enthält. Er funktioniert als Cache, bevor die Daten aus der Applikation in die Datenbank geschrieben werden. Diese Schreiboperation wird in der Regel erst mit Abschluss der Transaktion ausgelöst. Um Zugriff auf

diesen Persistence Context zu erhalten, bietet JPA das Interface `EntityManager` an. Über eine Factory kann ein Entity-Manager für eine zuvor definierte Persistenz-Unit kreiert werden. Mit Hilfe des Entity-Managers können Entitäten dem Persistenzkontext hinzugefügt oder daraus gelöscht werden. Es können ebenfalls Entitäten mithilfe ihrer ID gefunden werden oder Queries, sei es in Java Persistence Query Language oder in Native SQL, abgesetzt werden.

Als Beispiel für die Java Persistence Query Language diene folgende Named Query, definiert im orm.xml:

```
<named-query name="get_firewall_with_name">
  <query><![CDATA[
       SELECT fw
       FROM Firewall fw
       WHERE fw.name = :name]]>
  </query>
</named-query>
```

Wie zu sehen ist, werden anstelle von Tabellennamen die Namen der Entitäten verwendet, sprich die Klassennamen. Zudem kann mittels der Deklaration von Parametern (: + Identifier) die Query parametrisiert werden.

Named Queries könnten auch als Annotation in einer Klasse definiert werden. Um aber ein Vermischen der verschiedenen Sprachen zu vermeiden, werden sie im fwtool in einem separaten File definiert.

Persistence.xml

Die Konfiguration der Persistenz Unit geschieht im persistence.xml, welches zwingend im META-INF-Ordner im CLASSPATH abgelegt werden muss.

Persistence.xml aus dem Projekt fwtool:

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
  version="1.0">

  <persistence-unit name="fwtoolPersistency" transaction-
        type="RESOURCE_LOCAL">
    <properties>
      <!-- Properties for the Usage of a SQLite-Database -->
      <property name="hibernate.connection.driver_class"
        value="org.sqlite.JDBC" />
      <property name="hibernate.connection.url"
        value="jdbc:sqlite:test.db" />
      <property name="hibernate.connection.username" value=""/>
      <property name="hibernate.connection.password" value=""/>
      <property name="hibernate.dialect"
        value="ch.hsr.fwtool.persistence.SQLiteDialect"/>
      <!-- Echo all executed SQL to stdout -->
      <!—<property name="hibernate.show_sql" value="false" />-->

      <!-- Drop and recreate the database schema on startup -->
      <!--<property name="hibernate.hbm2ddl.auto" value="create"/>-->

      <property name="hibernate.cache.provider_class"
            value="org.hibernate.cache.HashtableCacheProvider" />
    </properties>
  </persistence-unit>
</persistence>
```

In diesem File werden die grundlegenden Konfigurationen für Hibernate als Persistenz-Framework festgelegt. Fürs Debugging kann man sich mittels des Properties `hibernate.show_sql` alle SQL-Statements, welche auf die Datenbank gehen, ausgeben lassen.

Um das Datenbank-Schema nicht jedesmal bei einer Änderung händisch neu kreieren zu müssen, kann man das Property `hibernate.hbm2ddl.auto` auf den Wert `create` setzen. Damit wird bei jedem Erstellen des Entity-Managers die Struktur der Datenbank neu gebildet und allfällige Änderungen am Mapping automatisch übernommen. Für den produktiven Einsatz ist dieses Property natürlich unter Anderem aus Sicherheitsgründen zu deaktivieren.

6 Ergebnisse

6.1 Architektur

Das Projekt Firewall-Analyse-Tool wurde in folgende Packages aufgeteilt:

src/ch.hsr.fwtool.cli
Dieses Paket enthält das Command Line Interface, bestehend aus den Klassen `AnalyzerCLI.java`, `DBManagerCLI.java` sowie `ParserCLI.java`. Das Command Line Interface (CLI) wird in einem eigenen Kapitel beschrieben.

src/ch.hsr.fwtool.forms
In diesem Package werden die sogenannten Forms-Klassen abgelegt. In diesen Forms-Klassen werden z.B. die Benutzereingaben welche über das Analyzer-CLI eingegeben wurden, abgelegt. Auch die Resultate der Abfragen werden in den entsprechenden Forms-Klassen abgelegt.

src/ch.hsr.fwtool.matcher
Im Package matcher ist die statische Klasse `Matcher.java` enthalten, welche das gesamte Matching der Access Policy Regeln gegen die Benutzereingaben macht. Dazu finden sich in diesem Package die Klassen, welche des Standard-Verhalten der Firewall-Technologien beinhaltet. Diese Klassen verwenden den Matcher, um die Anfragen gemäss dem Standard-Verhalten zu bearbeiten.

src/ch.hsr.fwtool.parser
Dieses Paket enthält die beiden Parser `ParserIOS.java` und `ParserIPTables.java`, welche aus den Dateien cisco.flex bzw. iptables.flex generiert wurden.

src/ch.hsr.fwtool.parser.helper
Dieses Paket enthält die Klasse `IPCalculator.java`, welche dazu dient, verschiedene Berechnungen durchzuführen, die mit IP-Adressen zu tun haben. Zum Beispiel kann eine Wildcard in eine Netzmaske oder auch ein Prefix in eine Netmaske umgerechnet werden.

src/ch.hsr.fwtool.parser.helper.cisco
Dieses Paket enthält Hilfsklassen, die vom Parser für Cisco IOS benötigt werden. Beispielsweise enthält das Paket eine Klasse `CiscoKeywords.java`, welche Keywords für Protokolle, Ports und ICMP-Message-Typen enthält, die auf einem Cisco IOS Gerät erlaubt sind. Die weiteren Klassen in diesem Paket werden vor allem dazu verwendet, die gelesenen Informationen während dem Parsen zwischenzuspeichern, bevor sie endgültig der Problem Domain hinzugefügt werden.

src/ch.hsr.fwtool.parser.helper.iptables
Dieses Paket enthält Hilfsklassen, die vom Parser für iptables benötigt werden. Beispielsweise enthält das Paket eine Klasse `IptablesKeywords.java`, welche Keywords für Protokolle und ICMP-Message-Typen enthält, die für iptables verwendet werden können. Die weiteren Klassen in diesem Paket werden vor allem dazu verwendet, die gelesenen Informationen während dem Parsen zwischenzuspeichern, bevor sie endgültig der Problem Domain hinzugefügt werden.

src/ch.hsr.fwtool.pd
Dieses Paket enthält die Problem Domain, wie sie in dem Dokument „Datenmodell“ beschrieben wird.

src/ch.hsr.fwtool.exception
Dieses Paket enthält einige Exceptions, die für das Projekt benötigt werden.

src/ch.hsr.fwtool.persistence
Im Package persistence sind jegliche Klassen abgelegt, welche für die Persistierung der Objekte aus der Problem Domain per ORM nötig sind. Im Moment ist dies nur eine einzige Klasse, welche den SQLite-Dialekt beschreibt. Diese Dialekt-Klasse benötigt das ORM-Framework, um die Umsetzung der Java-Datentypen in die Datentypen der spezifischen Datenbank korrekt vorzunehmen. Diese Klasse wurde von uns nicht selber erstellt, sondern aus dem Projekt hibernate-sqlite (http://code.google.com/p/hibernate-sqlite/) kopiert.

test/ch.hsr.fwtool.parser.test
Dieses Paket enthält Tests für die Parser und deren Hilfsklassen.

test/ch.hsr.fwtool.pd.test
Dieses Paket enthält Tests für die Klassen der Problem Domain.

test/ch.hsr.fwtool.persistence.test
Dieses Paket enthält Tests für alle Klassen, die mit der Persistenz und Datenhaltung zu tun haben.

test/ch.hsr.fwtool.test
Dieses Paket enthält eine TestSuite, die alle anderen Tests beinhaltet.

6.2 Cisco IOS Parser

Der Parser für Cisco IOS Version 12 verarbeitet eine running-config eines Cisco-IOS-Geräts und übersetzt sie ins unabhängige Format. Nicht der gesamte Inhalt eines running-configs ist relevant für das Firewallanalyetool. Der Parser versteht somit auch nicht den gesamten Inhalt und ignoriert sämtlichen unbekannten Input. Er muss jedoch die cisco-spezifische Syntax verstehen, welche relevant für das Firewall-Analyse-Tool ist. Was der Parser im Detail genau verstehen und verarbeiten kann, wird in den nächsten Abschnitten beschrieben.

6.2.1 Syntax

Hostname und Version

Hostname: `hostname` *`hostname`*
Der Name der Firewall. Dieser Name wird auch in der Datenbank für den Firewallnamen verwendet.

IOS Version: `version` *`version`*

ACLs

Der Parser für Cisco IOS kann folgende ACLs auswerten und ins unabhängige Format konvertieren:

Standard-ACLs: `access-list` *`access-list-number`* `{permit|deny}` *`source`*

Erweiterte ACLs: `access-list` *`access-list-number`* `{permit|deny}` *`protocol source destination`*

Erweitere TCP ACLs: `access-list` *`access-list-number`* `{permit|deny} tcp` *`source`* `[operator [`*`port`*`]]` *`destination`* `[operator[`*`port`*`]]`

Erweiterte UDP ACLs: `access-list` *`access-list-number`* `{permit|deny} udp` *`source`* `[operator [`*`port`*`]] destination [operator[`*`port`*`]]`

Erweiterte ICMP ACLs: `access-list access-list-number {permit|deny} icmp source destination [icmp-type]|[[icmp-type icmp-code]|[icmp-message]]`

Benannte ACLs: `ip access-list {standard|extended} access-list-name`
`//IPv4 Rules`

IPv6 ACLs: `ipv6 access-list access-list-name`
`//IPv6 Rules`

IPv6 Rule: `{permit|deny} protocol v6source v6destination`

IPv6 TCP Rule: `{permit|deny} tcp v6source [operator [port]] v6destination [operator [port]]`

IPv6 UDP Rule: `{permit|deny} udp v6source [operator [port]] v6destination [operator [port]]`

IPv6 ICMP Rule: `{permit|deny} icmp v6source v6destination [icmp-type]|[[icmp-type icmp-code]|[icmp-message]]`

access-list-number:
Gültige ACL-Nummern sind abhängig vom Typ der ACL:
Standard-ACL: 1-99, 1300-1999
Erweiterte ACL: 100-199, 2000-2699

source/destination:
Die Source/Destination kann auf unterschiedliche Art und Weise dargestellt sein:
`host hostIP`
`IPv4 wildcard`
`any`

v6source/v6destination:
`IPv6/prefix`
`host IPv6`
`any`

protocol:
Das Protokoll muss eine Zahl zwischen 0 und 255 oder ein von Cisco definiertes Keyword sein, welches eine dieser Zahlen repräsentiert.

```
R1(config-ext-nacl)#permit ?
  <0-255>     An IP protocol number
  ahp         Authentication Header Protocol
  eigrp       Cisco's EIGRP routing protocol
  esp         Encapsulation Security Payload
  gre         Cisco's GRE tunneling
  icmp        Internet Control Message Protocol
  igmp        Internet Gateway Message Protocol
  ip          Any Internet Protocol
  ipinip      IP in IP tunneling
  nos         KA9Q NOS compatible IP over IP tunneling
  ospf        OSPF routing protocol
  pcp         Payload Compression Protocol
  pim         Protocol Independent Multicast
  tcp         Transmission Control Protocol
  udp         User Datagram Protocol
```

Keywords für Protokolle auf einem Cisco IOS Version 12 Router

Beachten Sie zu diesem Thema auch die „Assigned Internet Protocol Numbers“ im Anhang.

operator:
Gültige Operatoren sind

`eq`	equal
`neq`	not equal
`lt`	less than
`gt`	greater than
`range`	range

Die ersten 4 Operatoren erwarten jeweils einen Port bzw. ein entsprechendes Keyword. Der Operator range erwartet zwei davon.

port:
Der Port muss eine Zahl zwischen 1 und 65535 oder ein von Cisco definiertes Keyword sein, welches eine dieser Zahlen repräsentiert.

```
R1(config-ext-nacl)#permit tcp any eq ?
  <0-65535>    Port number
  bgp          Border Gateway Protocol (179)
  chargen      Character generator (19)
  cmd          Remote commands (rcmd, 514)
  daytime      Daytime (13)
  discard      Discard (9)
  domain       Domain Name Service (53)
  drip         Dynamic Routing Information Protocol (3949)
  echo         Echo (7)
  exec         Exec (rsh, 512)
  finger       Finger (79)
  ftpFile      Transfer Protocol (21)
  ftp-data     FTP data connections (20)
  gopher       Gopher (70)
  hostname     NIC hostname server (101)
  ident        Ident Protocol (113)
  irc          Internet Relay Chat (194)
  klogin       Kerberos login (543)
  kshell       Kerberos shell (544)
  login        Login (rlogin, 513)
  lpd          Printer service (515)
  nntp         Network News Transport Protocol (119)
  pim-auto-rp  PIM Auto-RP (496)
  pop2         Post Office Protocol v2 (109)
  pop3         Post Office Protocol v3 (110)
  smtp         Simple Mail Transport Protocol (25)
  sunrpc       Sun Remote Procedure Call (111)
  tacacs       TAC Access Control System (49)
  talk         Talk (517)
  telnet       Telnet (23)
  time         Time (37)
  uucp         Unix-to-Unix Copy Program (540)
  whois        Nicname (43)
  www          World Wide Web (HTTP, 80)
```

Keywords für Ports auf einem Cisco IOS Version 12 Router

Beachten Sie zu diesem Thema auch die „Well Known Ports“ im Anhang.

icmp:
ICMP-Verkehr kann auf unterschiedliche Weise ausgewählt werden.

- `icmp-type`
- `icmp-type icmp-code`
- `icmp-message`

ICMP-Typ und ICMP-Code können jeweils eine Zahl zwischen 0 und 255 sein. Alternativ kann auch ein gültiges Keyword für einen ICMP message type eingegeben werden. Bei der ersten Variante fehlt der ICMP-Code. Dieser wird darum auf ‘ANY‘ gesetzt. Die beiden anderen Varianten setzen ICMP-Typ und ICMP-Code entsprechend der Eingabe.

```
R1(config-ext-nacl)#permit icmp any any ?
  <0-255>                     ICMP message type
  administratively-prohibited Administratively prohibited
  alternate-address           Alternate address
  conversion-error            Datagram conversion
  dod-host-prohibited         Host prohibited
  dod-net-prohibited          Net prohibited
  echo                        Echo (ping)
  echo-reply                  Echo reply
  general-parameter-problem   Parameter problem
  host-isolated               Host isolated
  host-precedence-unreachable Host unreachable for precedence
  host-redirect               Host redirect
  host-tos-redirect           Host redirect for TOS
  host-tos-unreachable        Host unreachable for TOS
  host-unknown                Host unknown
  host-unreachable            Host unreachable
  information-reply           Information replies
  information-request         Information requests
  mask-reply                  Mask replies
  mask-request                Mask requests
  mobile-redirect             Mobile host redirect
  net-redirect                Network redirect
  net-tos-redirect            Net redirect for TOS
  net-tos-unreachable         Network unreachable for TOS
  net-unreachable             Net unreachable
  network-unknown             Network unknown
  no-room-for-option          Parameter required but no room
  option-missing              Parameter required but not present
  packet-too-big              Fragmentation needed and DF set
  parameter-problem           All parameter problems
  port-unreachable            Port unreachable
  precedence-unreachable      Precedence cutoff
  protocol-unreachable        Protocol unreachable
  reassembly-timeout          Reassembly timeout
  redirect                    All redirects
  router-advertisement        Router discovery advertisements
  router-solicitation         Router discovery solicitations
  source-quench               Source quenches
  source-route-failed         Source route failed
  time-exceeded               All time exceededs
  time-range                  Specify a time-range
  timestamp-reply             Timestamp replies
  timestamp-request           Timestamp requests
  traceroute                  Traceroute
```

```
  ttl-exceeded                 TTL exceeded
  unreachable                  All unreachables
```

Keywords für ICMP Message Types auf einem Cisco IOS Version 12 Router

Beachten Sie zu diesem Thema auch die „ICMP Type Numbers“ im Anhang.

NAT

Der Parser für Cisco IOS kann folgende Eingaben in NatRules der Problem Domain übersetzen:

Pool erstellen:
`ip nat pool poolname ipv4 ipv4 {netmask netmask|prefix-length prefix-length}`

statisches NAT:
`ip nat {inside source|inside destination|outside source} static ip ip`

dynamisches NAT :
`ip nat {inside source|inside destination|outside source} list access-list pool pool`

Interfaces

Der Parser für Cisco IOS erkennt alle Interfaces und dazugehörigen relevanten Informationen:

Interface:
`interface interfaceName`

IP-Adressen:
```
ip address ip netmask
ipv6 address IPv6/prefix
```

ACL Zuordnungen:
```
ip access-group access-list {in|out}
ipv6 traffic-filter access-list {in|out}
```

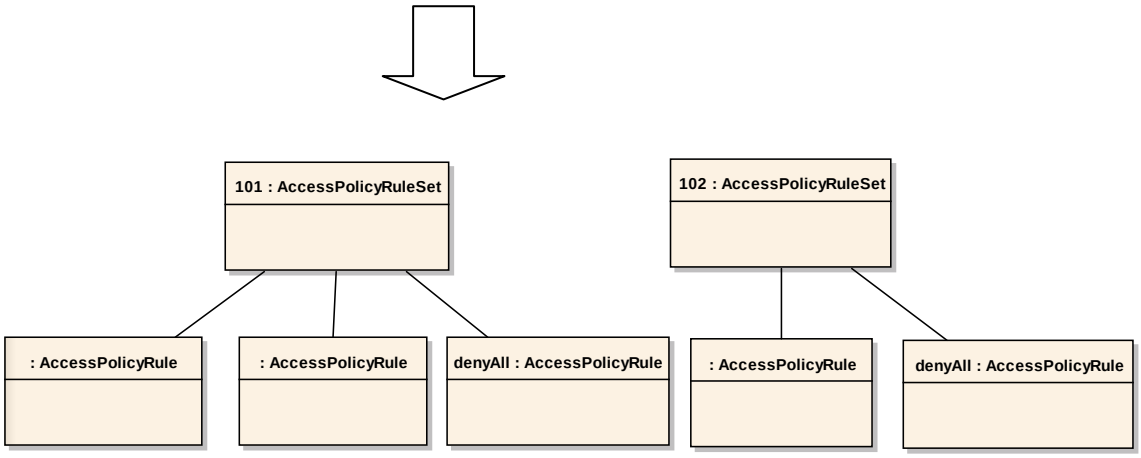
NAT Inside/Outside Interfaces:
`ip nat {inside|outside}`

6.2.2 Übersetzung der ACLs in das unabhängige Format

Dieser und der nächste Abschnitt beschreibt die Übersetzung von ACLs bzw. NAT-Regeln in das unabhängige Format. Dabei werden UML-Diagramme gezeigt, welche Ausschnitte aus der Problem Domain zeigen. Beachten Sie bitte, dass die gesamte Problem Domain in einem eigenen Dokument beschrieben wird.

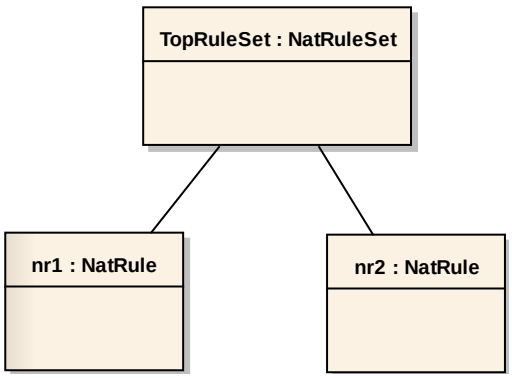
Das Firewall-Analyse-Tool übersetzt eine ACL in ein AccessPolicyRuleSet. Jede Regel einer ACL wird als AccessPolicyRule an das entsprechende AccessPolicyRuleSet angehängt. Die AccessPolicyRules innerhalb eines AccessPolicyRuleSets sind geordnet. Die Reihenfolge der AccessPolicyRules entspricht der Reihenfolge, wie diese aus der running-config gelesen werden. Die letzte AccessPolicyRule innerhalb jedes AccessPolicyRuleSets entspricht dem „implicit deny all“, welches bei Cisco-Geräten üblich ist. Die folgende Darstellung verdeutlicht dies:

```
access-list 101 deny 172.16.3.10 0.0.0.0
access-list 101 permit 0.0.0.0 255.255.255.255
access-list 102 permit host 10.1.2.1
```



6.2.3 Übersetzung der NAT-Regeln in das unabhängige Format

Die NAT-Regeln werden alle in ein einziges NatRuleSet namens „TopRuleSet“ übersetzt, unabhängig davon, ob es sich um statisches oder dynamisches Mapping handelt:

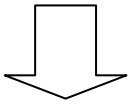


Das folgende Beispiel zeigt, mit welchen Werten eine NatRule initialisert wird. Hierbei wird auch berücksichtigt, welche Interfaces mittels `ip nat inside` oder `ip nat outside` als intern bzw extern definiert sind. Für das Beispiel sei angenommen, dass folgende Interfaces definiert sind.

```
interface FastEthernet0/0
  description Inside Interface
  ip nat inside
  ip address 192.168.1.0 255.255.255.0
!
interface FastEthernet0/1
  description Inside Interface
  ip nat inside
  ip address 192.168.2.0 255.255.255.0
!
interface Serial0/0/0
  description Outside Interface
  ip nat outside
  ip address 198.18.5.0 255.255.255.0
!
```

statisches NAT

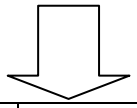
```
ip nat inside source static 192.168.1.10 198.18.5.10
```



Incoming Interface	Outgoing Interface	Original Source	Original Destination	Translated Source	Translated Destination
FastEthernet0/0	ANY	192.168.1.10	0.0.0.0	198.18.5.10	0.0.0.0
FastEthernet0/1	ANY	192.168.1.10	0.0.0.0	198.18.5.10	0.0.0.0

dynamisches NAT

```
ip access-list standard natACL
  deny   192.168.1.0 0.0.0.255
  permit 192.168.2.0 0.0.0.255

ip nat pool natpool 192.18.120.11 192.18.120.20 netmask 255.255.255.224
ip nat inside source list natACL pool natpool
```



Incoming Interface	Outgoing Interface	Original Source	Original Destination	Translated Source	Translated Destination
FastEthernet0/0	ANY	192.168.1.0/24	0.0.0.0/0	0.0.0.0/0	0.0.0.0/0
FastEthernet0/0	ANY	192.168.2.0/24	0.0.0.0/0	192.18.120.11-192.18.120.20	0.0.0.0/0
FastEthernet0/1	ANY	192.168.1.0/24	0.0.0.0/0	0.0.0.0/0	0.0.0.0/0
FastEthernet0/1	ANY	192.168.2.0/24	0.0.0.0/0	192.18.120.11-192.18.120.20	0.0.0.0/0

6.3 iptables Parser

Der Parser für iptables verarbeitet die Ausgaben des `iptables-save` bzw. `ip6tables-save` Kommandos sowie des `ip` Kommandos unter Linux. Das `iptables-save/ip6tables-save` Kommando generiert eine Ausgabe mit allen auf einem Rechner konfigurierten ip(6)tables Regeln, wohingegen das `ip` Kommando eine Liste aller Interfaces und die dazugehörigen relevanten Informationen ausgeben kann.

Für weitere Informationen über die Kommandos `iptables-save` [6] und `ip6tables-save` [7] lesen Sie bitte die entsprechenden Manuals oder weiterführende Literatur.

6.3.1 Anforderungen an den Input

Wie schon erwähnt, verarbeitet der Parser die Ausgaben der Kommandos `iptables-save/ip6tables-save` und `ip`.

Die Ausgaben der Befehle müssen für den Parser in einer einzigen Inputdatei zusammengefasst werden. Wichtig ist hier auch die Reihenfolge. Zu Beginn der Inputdatei muss die Ausgabe des `ip` Kommandos stehen, da der Parser die Interfaces kennen muss, bevor er irgendwelche iptables Regeln verarbeiten kann, in welchen Interfaces vorkommen. Danach können dann die Ausgaben von `iptables-save`/`ip6tables-save` folgen. Ob hier zuerst der Output von `iptables-save` oder von `ip6tables-save` eingefügt wird oder umgekehrt, ist nicht relevant. Es kann auch eines von beidem fehlen. Wichtig

ist jedoch, dass folgende Kopfzeile des `iptables-save`/`ip6tables-save` Kommandos nicht weggelassen oder gelöscht wird:

```
# Generated by iptables-save v1.3.6 on Fri Oct 30 13:04:34 2009
```

Durch den fett markierten Teil erkennt der Parser nämlich, ob es sich um den Output von `iptables-save` oder `ip6tables-save` handelt. Abhängig davon erwartet der Parser danach IPv4- oder IPv6-iptables Regeln. Folgender Codeausschnitt zeigt das Grundgerüst einer Inputdatei für den iptables Parser:

```
linux:~# ip a l
1:  lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
                  //more interfaces

# Generated by iptables-save v1.4.1.1 on Thu Oct  8 16:01:48 2009
*filter
:INPUT ACCEPT [72:13717]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [10:1156]
-A INPUT -i lo -j ACCEPT
-A FORWARD -i eth4 -o eth6 -p tcp -m tcp --dport 80 -j ACCEPT
                  //more iptables-rules
COMMIT
# Completed on Thu Oct  8 16:01:48 2009

# Generated by ip6tables-save v1.4.2 on Sat Dec  5 20:35:49 2009
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [6:384]

-A INPUT -j ACCEPT
-A INPUT -i eth0 -j ACCEPT
-A FORWARD -s 1234:4567::9abc/128 -d 89ab:cdef:1234::1212/128 –j ACCEPT
-A OUTPUT -o eth0 -j ACCEPT
                  //more ip6tables-rules
COMMIT # Completed on Sat Dec  5 20:35:49 2009
```

Wichtig im Zusammenhang mit dem Input ist noch die Tatsache, dass der Dateiname der Inputdatei als Firewallname in der Datenbank übernommen wird. Wenn zwei Inputdateien denselben Dateinamen haben und beim ParserCLI das update-Flag gesetzt ist, wird also einfach die zuerst geparste Firewall in der Datenbank überschrieben.

6.3.2 Syntax

Der Parser für iptables kennt nicht den gesamten Umfang aller Möglichkeiten mit iptables. Er versteht folgenden Input, der durch den iptables-save Befehl generiert wird:

`*nat ..... COMMIT`	Der gesamte Input zwischen `*nat` und `COMMIT` gehört zur NAT Table.

`*filter ..... COMMIT`	Der gesamte Input zwischen `*filter` und `COMMIT` gehört zur Filter Table.
`:chain target`	Das Target gibt an, welche Aktion am Ende der gewählten Chain wirkt, wenn keine andere Regel zutrifft. Zum Beispiel bewirkt `:INPUT ACCEPT`, dass die Regel am Ende der INPUT chain allen Verkehr akzeptiert.
`-A chain rule-specification`	fügt eine neue Regel an das Ende der selektierten Chain an.

Rule Specification

`-p [!]protocol`	gibt das Protokoll an. Das Ausrufezeichen negiert das Protokoll.
`-s [!]source`	gibt die Source-Adresse an. Das Ausrufezeichen negiert die Source.
`-d [!]destination`	gibt die Destination-Adresse an. Das Ausrufezeichen negiert die Destination.
`--sport [!]port`	gibt den Source-Port an. Das Ausrufezeichen negiert den Source-Port.
`--dport [!]port`	gibt den Destination-Port an. Das Ausrufezeichen negiert den Destination-Port.
`-i [!]interfaceName`	gibt das Incoming Interface an. Das Ausrufezeichen negiert das Incoming Interface.
`-o [!]interfaceName`	gibt das Outgoing Interface an. Das Ausrufezeichen negiert das Outgoing Interface.
`--icmp-type [!]icmpType`	gibt einen ICMP-Typ an und einen ICMP-Code an. Das Ausrufezeichen negiert den ICMP-Typ und den ICMP-Code zusammengenommen.
`-j target`	gibt das Target an.
`-m state --state states`	gibt einen oder mehrere States an.
`-j SNAT --to-source ip[-ip][:port-port]`	führt ein NAT für die Source Adresse durch. Mittels `--to-source` kann eine IP-Adresse bzw. ein IP-Adressrange und optional auch ein Port-Range angegeben werden, in welche die Source-Adresse übersetzt wird.
`-j MASQUERADE [--to-ports port[-port]]`	führt ein NAT für die Source Adresse durch. Die übersetzte Source entspricht immer der IP-Adresse des ausgehenden Interfaces. Optional kann ein Port-Range angegeben werden, um die Ports zu bestimmen, in welche übersetzt wird.

`-j DNAT --to-destination ip[-ip][:port-port]`	führt ein NAT für die Destination Adresse durch. Mittels `--to-destination` kann eine IP-Adresse bzw. ein IP-Adressrange und optional auch ein Port-Range angegeben werden, in welche die Destination-Adresse übersetzt wird.

chain:

Bei *:chain target* sind nur sogenannte built-in chains erlaubt. Dies sind in der NAT-Table die Chains `PREROUTING`, `OUTPUT` und `POSTROUTING`. In der Filter-Table handelt es sich um die Chains `INPUT`, `FORWARD` und `OUTPUT`. Beim Befehl `-A` *chain rule-specification* allerdings können auch sogenannte user-defined chains angegeben werden.

target:

Das Target muss entweder `ACCEPT`, `DROP` oder `RETURN` sein. In einer rule specification kann mittels `-j target` auch eine weitere user-defined chain angegeben werden.

protocol:

Das Protokoll muss eine Zahl zwischen 0 und 255 oder ein für iptables gültiges Keyword sein, welches eine dieser Zahlen repräsentiert. Erlaubt sind alle Keywords in der Datei `/etc/protocols`.

```
ip           0   IP           # internet protocol, pseudo protocol number
#hopopt      0   HOPOPT       # IPv6 Hop-by-Hop Option [RFC1883]
icmp         1   ICMP         # internet control message protocol
igmp         2   IGMP         # Internet Group Management
ggp          3   GGP          # gateway-gateway protocol
ipencap      4   IP-ENCAP     # IP encapsulated in IP (officially ``IP'')
st           5   ST           # ST datagram mode
tcp          6   TCP          # transmission control protocol
egp          8   EGP          # exterior gateway protocol
igp          9   IGP          # any private interior gateway (Cisco)
pup          12  PUP          # PARC universal packet protocol
udp          17  UDP          # user datagram protocol
hmp          20  HMP          # host monitoring protocol
xns-idp      22  XNS-IDP      # Xerox NS IDP
rdp          27  RDP          # "reliable datagram" protocol
iso-tp4      29  ISO-TP4      # ISO Transport Protocol class 4 [RFC905]
xtp          36  XTP          # Xpress Transfer Protocol
ddp          37  DDP          # Datagram Delivery Protocol
idpr-cmtp    38  IDPR-CMTP    # IDPR Control Message Transport
ipv6         41  IPv6         # Internet Protocol, version 6
ipv6-route   43  IPv6-Route   # Routing Header for IPv6
ipv6-frag    44  IPv6-Frag    # Fragment Header for IPv6
idrp         45  IDRP         # Inter-Domain Routing Protocol
rsvp         46  RSVP         # Reservation Protocol
gre          47  GRE          # General Routing Encapsulation
esp          50  IPSEC-ESP    # Encap Security Payload [RFC2406]
ah           51  IPSEC-AH     # Authentication Header [RFC2402]
skip         57  SKIP         # SKIP
ipv6-icmp    58  IPv6-ICMP    # ICMP for IPv6
ipv6-nonxt   59  IPv6-NoNxt   # No Next Header for IPv6
ipv6-opts    60  IPv6-Opts    # Destination Options for IPv6
rspf         73  RSPF CPHB    # Radio Shortest Path First (officially CPHB)
vmtp         81  VMTP         # Versatile Message Transport
eigrp        88  EIGRP        # Enhanced Interior Routing Protocol (Cisco)
ospf         89  OSPFIGP      # Open Shortest Path First IGP
```

```
ax.25       93  AX.25         # AX.25 frames
ipip        94  IPIP          # IP-within-IP Encapsulation Protocol
etherip     97  ETHERIP       # Ethernet-within-IP Encapsulation [RFC3378]
encap       98  ENCAP         # Yet Another IP encapsulation [RFC1241]
#           99                # any private encryption scheme
pim         103 PIM           # Protocol Independent Multicast
ipcomp      108 IPCOMP        # IP Payload Compression Protocol
vrrp        112 VRRP          # Virtual Router Redundancy Protocol
l2tp        115 L2TP          # Layer Two Tunneling Protocol [RFC2661]
isis        124 ISIS          # IS-IS over IPv4
sctp        132 SCTP          # Stream Control Transmission Protocol
fc          133 FC            # Fibre Channel
udplite     136 UDPLite       # UDP-Lite
mpls-in-ip  137 MPLS-in-IP    # MPLS-in-IP [RFC4023]
manet       138               # MANET Protocols
hip         139 HIP           # Host Identity Protocol
```

Datei /etc/protocols auf einem Knoppix 2009

Beachten Sie zu diesem Thema auch die „Assigned Internet Protocol Numbers“ im Anhang.

source/destination:
Die Source/Destination kann auf unterschiedliche Art und Weise dargestellt sein:

```
ip
ip/netmask
ip/prefix
```

port:
Der Port muss eine Zahl zwischen 1 und 65535 sein.

interfaceName:
Der interfaceName muss ein gültiger Interfacename auf dem entspechenden Rechner sein.

icmpType
Der ICMP-Typ kann eine Zahl zwischen 0 und 255 sein oder ein ICMP-Typname, der beim Komando `iptables –p icmp –h` ausgegeben wird.

```
linux:~# Iptables –p icmp -h
Valid ICMP Types:
   any
   echo-reply (pong)
   destination-unreachable
   network-unreachable
   host-unreachable
   protocol-unreachable
   port-unreachable
   fragmentation-needed
   source-route-failed
   network-unknown
   host-unknown
   network-prohibited
   host-prohibited
   TOS-network-unreachable
   TOS-host-unreachable
   communication-prohibited
   host-precedence-violation
   precedence-cutoff
```

```
source-quench
redirect
   network-redirect
   host-redirect
   TOS-network-redirect
   TOS-host-redirect
echo-request (ping)
router-advertisement
router-solicitation
time-exceeded (ttl-exceeded)
   ttl-zero-during-transit
   ttl-zero-during-reassembly
parameter-problem
   ip-header-bad
   required-option-missing
timestamp-request
timestamp-reply
address-mask-request
address-mask-reply
```

Ausgabe des Befehls `iptables -p icmp -h` auf einem Knoppix 2009

Beachten Sie zu diesem Thema auch die „ICMP Type Numbers“ im Anhang.

`states:`
Einer oder mehrere States sind möglich. Mehrere States werden durch Kommas abgetrennt. Die möglichen States sind ESTABLISHED und NEW.

`SNAT, DNAT` und `MASQUERADE`
Diese Targets sind nur in der NAT Table gültig.

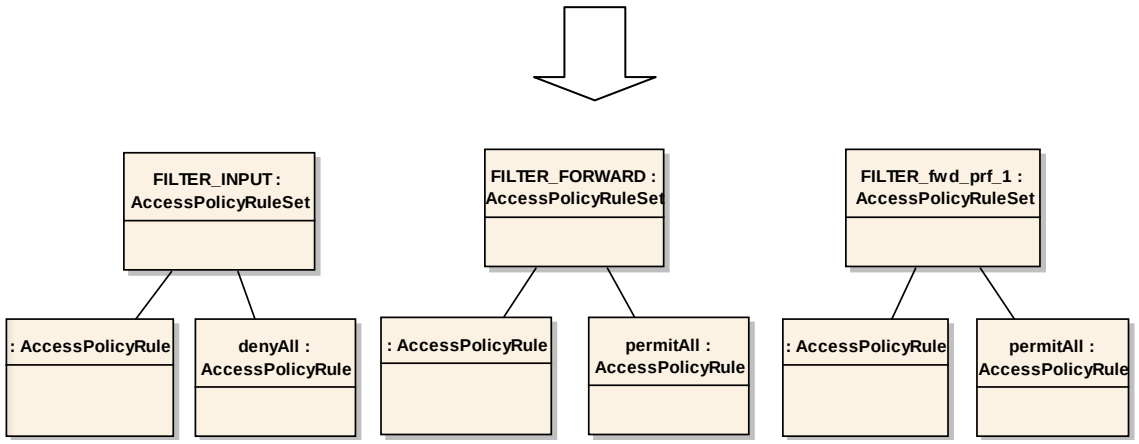
Der Parser für iptables kann weiter auch Informationen über interfaces einlesen. Hierbei dient die Ausgabe des `ip` Kommandos unter Linux als Input.

`number: interfaceName`	Dieser Input am Zeilenanfang bedeutet, dass ab hier das Interface mit dem angegebenen Namen beschrieben wird.
`inet ip/prefix`	Die IP-Adresse wird dem aktuellen Interface zugeordnet.
`inet6 ipv6/prefix`	Die IPv6-Adresse wird dem aktuellen Interface zugeordnet.

6.3.3 Übersetzung von Filter-Regeln in das unabhängige Format

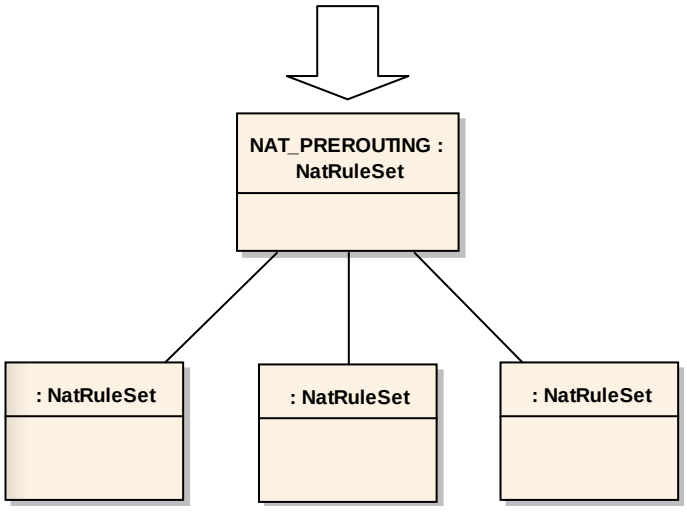
Der Parser für iptables erstellt aus jeder chain ein eigenes AccessPolicyRuleSet. Die einzelnen Regeln einer Chain werden dann jeweils in eine AccessPolicyRule übersetzt, die dann dem entsprechenden AccessPolicyRuleSet hinzugefügt wird. Die folgende Abbildung verdeutlicht dies:

```
*filter
:INPUT DROP [72:13717]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [10:1156]

-A INPUT -d !127.0.0.0/255.0.0.0 -j ACCEPT
-A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
-A fwd_prf_1 -d 152.96.37.60 -j ACCEPT
COMMIT
```



6.3.4 Übersetzung von NAT-Regeln in das unabhängige Format

```
linux:~# ip a l
0: eth0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast qlen 1000
   link/ether 00:16:34:00:0a:00 brd ff:ff:ff:ff:ff:ff
   inet 152.96.121.205/29 brd 152.96.121.207 scope global eth0

# Generated by iptables-save v1.3.6 on Fri Oct 30 13:04:34 2009
*nat
:PREROUTING ACCEPT [47769:3967164]
:POSTROUTING ACCEPT [8151:554145]
:OUTPUT ACCEPT [4586:345580]
-A PREROUTING -p tcp -m tcp --dport 80 -o eth0 -j MASQUERADE
-A PREROUTING -p tcp -m tcp --dport 80 -j SNAT --to-source 123.123.123.123
-A PREROUTING -p tcp -m tcp --dport 443 -j DNAT --to-destination 152.96.8.65
COMMIT
```



Incoming Interface	Outgoing Interface	Original Source	Original Destination	Translated Source	Translated Destination
ANY	eth0	0.0.0.0/0	0.0.0.0/0	152.96.121.205 /32	0.0.0.0/0
ANY	ANY	0.0.0.0/0	0.0.0.0/0	123.123.123.123 /32	0.0.0.0/0
ANY	ANY	0.0.0.0/0	0.0.0.0/0	0.0.0.0/0	152.96.8.65/32

6.4 CLI

Für dieses Projekt wurden drei unterschiedliche CLI's erstellt. Dies sind das ParserCLI, das AnalyzerCLI und das DBManagerCLI. Was die Aufgaben der einzelnen CLI's im Einzelnen sind, wird in diesem Abschnitt beschrieben.

6.4.1 ParserCLI

Zweck

Das ParserCLI wählt anhand der übergebenen Parameter eine Inputdatei und übergibt diese dem entsprechenden Parser. Der Parser liest die gesamte Inputdatei ein und übersetzt den gelesenen Input in das allgemeine Format. Die generierte Firewall der Problem Domain wird danach in die Datenbank eingefügt. Das ParserCLI gibt nach dem Parsevorgang eine Zusammenfassung über die Anzahl Interfaces der Firewall sowie Anzahl geparster Rules und RuleSets.

Parameter

Das ParserCLI erwartet folgende Parameter in der hier angegebenen Reihenfolge:

`format:` Format des Inputs. Mögliche Formate sind IOS und IPTABLES.

`filename:` Dateiname der Inputdatei. Es muss sich dabei um eine Textdatei handeln.

`update:` optionaler Parameter. Gibt an, dass eine eventuell bestehende Firewall mit gleichem Namen in der Datenbank kommentarlos überschrieben wird.

Das ParserCLI kann also folgendermassen aufgerufen werden:
`ParserCLI format filename [update]`

6.4.2 AnalyzerCLI

Zweck

Das AnalyzerCLI erlaubt es, Traffic zu spezifieren, welcher über eine bereits persistierte Firewall laufen soll. Nachdem alle Parameter spezifiziert wurden, berechnet die Software, ob der Traffic erlaubt oder verworfen wird. Zusätzlich wird ausgegeben, ob NAT-Regeln angewendet werden. Weitere Informationen über die Regeln, welche angewendet wurden und wo innerhalb der Inputdatei diese zu finden sind, werden ausgegeben.

Das AnalyzerCLI fordert den Benutzer auf, nacheinander folgende Angaben zu machen:

- Auswahl der Firewall (aus einer Liste aller Firewalls in der DB)
- Source IP Address
- Destination IP Address
- Incoming Interface (Auswahl aus einer Liste)
- Outgoing Interface (Auswahl aus einer Liste)

- Protocol
- protokollspezifische Parameter

Parameter

Das AnalyzerCLI wird ohne Parameter aufgerufen.

6.4.3 DBManagerCLI

Zweck

Das DBManagerCLI bietet die Möglichkeit, alle Firewalls in der Datenbank aufzulisten oder aber auch einzelne oder alle Firewalls auf einmal zu löschen.

Parameter

Das DBManagerCLI wird ohne Parameter aufgerufen.

6.5 Matching Prozess

Im folgenden Kapitel wird auf die Algorithmen eingegangen, welche zum Nachbilden des Matching-Vorgangs auf einer Firewall implementiert wurden.

6.5.1 AnalyzerForm

In der Klasse AnalyzerForm werden die Eingaben des Benutzers, welche über das Command Line Interface getätigt wurden und welche den Netzwerkstrom, der simuliert werden möchte, definiert, gespeichert. Dieses Objekt dient im gesamten Matching-Prozess als Repräsentation der Benutzereingaben.

Falls NAT-Regeln, welche auf der Firewall definiert sind, auf den definierten Netzwerkstrom angewendet werden, dann werden die Änderungen (Neue Source-Adresse, Änderung des Services, etc.) auf das AnalyzerForm-Objekt angewendet.

6.5.2 Matcher

Der Matcher ist die statische Klasse, welche den Matching Prozess für ein gegebenes `AccessPolicyRuleSet` und ein ausgefülltes `AnalyzerForm` durchführt. Mittels der Methode `processRule()` wird durch das gesamte Regelset iteriert. Grundsätzlich ist es so, dass bei einem positiven Fund, sprich sowohl Source-Adresse, wie auch Destination-Adresse und der Service aus den Benutzereingaben stimmen mit den in der Filter-Regel definierten Werten überein, die Behandlung dieses Regelsets sofort abgebrochen und die matchende Regel angewendet wird.

```
public static boolean processRules(AccessPolicyRuleSet aprs,
  AnalyzerForm af) {

  boolean permitDenyMatchFound = false;

  for (AccessPolicyRule apr : aprs.getRules()) {
    if(permitDenyMatchFound)
      return true;
    boolean sourceMatch = matchSource(apr, af.getSourceIP());
    boolean destinationMatch = matchDestination(apr,
      af.getDestinationIP());
    boolean serviceMatch = matchService(apr, af);

    if (sourceMatch == true && destinationMatch == true
```

```
            && serviceMatch == true) {
          result.add(apr);
          if(apr.getAction().getAc() == ActionCommand.REDIRECT){
            String rulesetname = apr.getAction().getApr_name();
            Firewall fw = AnalyzerCLI.getFirewall();
            AccessPolicyRuleSet temp =
              fw.getAccessPolicyRuleSets().get(rulesetname);
            if(temp != null){
              permitDenyMatchFound = processRules(temp, af);
            }
          }
          else if(apr.getAction().getAc() == ActionCommand.RETURN){
            return false;
          }
          else
            return true;
        }
      }
      return false;
    }
```

In dieser Methode wird auch die Möglichkeit von iptables behandelt, aus einer Chain in eine andere, selber definierte Chain zu springen `(-j –user-defined-chain)`. Um diese Verhalten zu implementieren, wird die Methode `processRules()` rekursiv mit dem neuen `AccessPolicyRuleSet`, welches als Sprung definiert ist, aufgerufen.

Da aus einer solchen Chain bei einem Match auch wieder zurück zur Eltern-Chain gesprungen werden kann `(-j RETURN)` oder aber ein "endgültiger" Match gefunden werden kann `(-j ACCEPT` oder `–j DROP)` welcher zum Abbruch der Behandlung auf dieser built-in-Chain führt, wird der boolean `permitDenyMatchFound` verwendet, um diese beiden Möglichkeit zu unterscheiden.

Matchende Regeln, egal welche Aktion sie anschliessend auslösen, werden in einer Collection abgespeichert und können anschliessend für das Feedback an den User verwendet werden.
Als Beispiel für eine Methode, welche das konkrete Matching durchführt, wird hier die Methode matchSource() aufgeführt und kurz besprochen:

```
private static boolean matchSource(AccessPolicyRule apr,
  Addressable address) {
  boolean isAMatch = true;
  Addressable src = apr.getSource();
  if (src instanceof IPv4Address && address instanceof IPv4Address) {
       if (!((IPv4Address) src).isInNet(
          ((IPv4Address)address).getIpString())) ^ apr.getNotOpSource()) {
          isAMatch = false;
       }
    } else if (src instanceof IPv6Address
       && address instanceof IPv6Address) {
       if (!((IPv6Address) src).isInNet(
          ((IPv6Address)address).getIpString())) ^ apr.getNotOpSource()) {
          isAMatch = false;
       }
    } else if (src instanceof IPv4AddressRange
       && address instanceof IPv4Address) {
       if (!((IPv4AddressRange) src).addressIsInRange(
          ((IPv4Address)address).getIpString())) ^ apr.getNotOpSource()) {
          isAMatch = false;
       }
    } else if (src instanceof IPv6AddressRange
       && address instanceof IPv6Address) {
       if (!((IPv6AddressRange) src).addressIsInRange(
          ((IPv6Address)address).getIpString())) ^ apr.getNotOpSource()) {
          isAMatch = false;
```

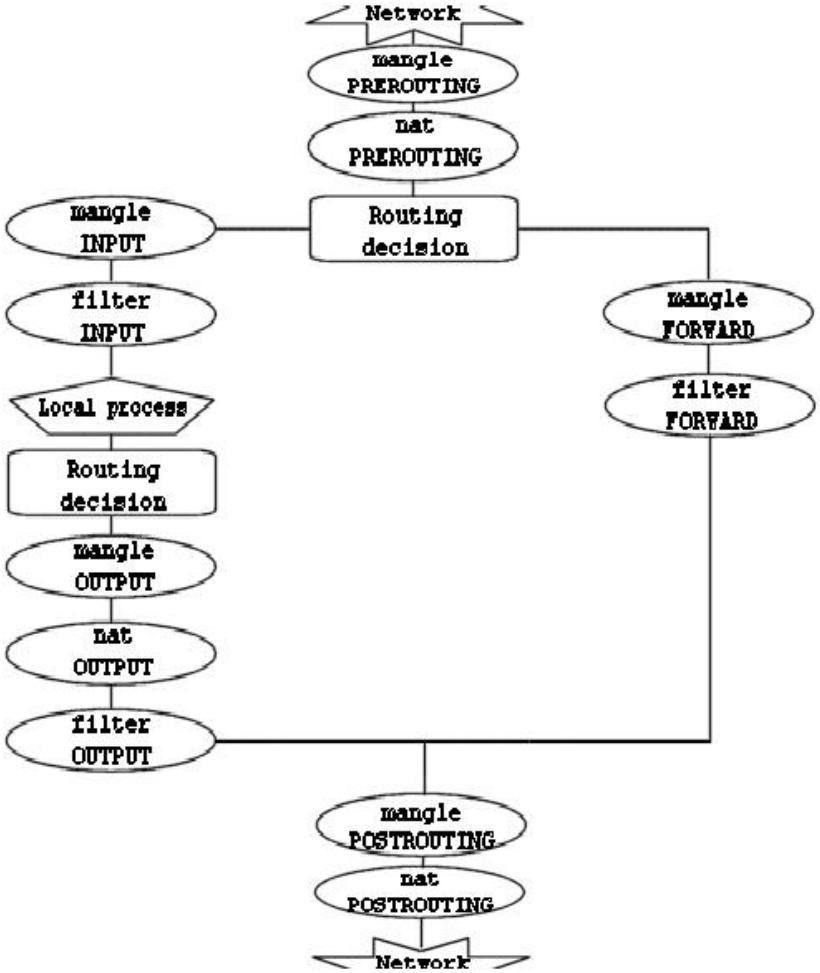
```
            }
         }
         else
            isAMatch = false;
      return isAMatch;
   }
```

Da ein Source-Addressable in einer Access Policy Regel entweder eine IPv4-Addresse, eine IPv6-Addresse oder ein Adressbereich der beiden Typen sein kann, muss die match-Methode diese Möglichkeiten unterscheiden. Auch die Adresse, welche vom Benutzer für die Definition des Netzwerkstroms eingegeben wurde, kann entweder vom Typ IPv4 oder IPv6 sein. Sobald festgestellt wurde, mit welchen Typen gearbeitet wird, wird überprüft, ob die vom User definierte Adresse auf die in der Regel definierte Adresse zutrifft. Hierfür werden die Methoden `isInNet()` der Address-Klassen oder die `addressIsInRange()`–Methode der Range-Klassen verwendet.

Da es bei der Definition von Regeln die Möglichkeit gibt, Teile der Regel zu negieren (beispielsweise mit dem Operator "!" in iptables oder mit dem Operator „neq" in Cisco IOS), muss auch dies noch in die Schlussentscheidung einfliessen. Deshalb wird die vorher gewonnene Entscheidung mittels dem XOR-Operator (^) mit dem allenfalls gesetzten Not-Operator verknüpft.

iptables Standard Verhalten

Um das Verhalten einer iptables-Firewall nachzubilden, wurde die Klasse IptablesStandardBehaviour implementiert. Darin ist die Reihenfolge der Bearbeitung der einzelnen Schritte, sprich die Reihenfolge der NAT- und Filter-Entscheidungen gemäss folgender Grafik festgelegt:



Darstellung der Reihenfolge der Traversierung von iptables-Chains[8]

[8] Quelle: http://www.faqs.org/docs/iptables/traversingoftables.html, letzter Zugriff am 15.12.2009

Aus dieser Darstellung ist ersichtlich, dass es eigentlich drei Typen von Verkehr, in den eine iptables-Firewall involviert sein kann, gibt:

1. Verkehr auf die Firewall selber (Destination = IP eines Interfaces der Firewall)
2. Verkehr weg von der Firewall (Source = IP eines Interfaces der Firewall)
3. Verkehr, welcher durch die Firewall weitergeleitet, sprich über die Firewall gerouted wird.

Für diese drei Arten wurde jeweils eine Methode implementiert, welche den Ablauf festlegt. Als Beispiel sie hier die Methode `processIPv4TrafficFromFw()` aufgeführt:

```
public IptablesResultForm processIPv4TrafficFromFw(AnalyzerForm af){
   NatRule matchingOutRule = processNat(ipv4Nat_OUTPUT, af);
   irf.setAppliedPreroutingNatRule(matchingOutRule);
   Matcher.processRules(ipv4_OUTPUT, af);
   irf.setMatchingOutputRules(Matcher.getResult());
   NatRule matchingPostRule = processNat(ipv4Nat_POSTROUTING, af);
   irf.setAppliedPreroutingNatRule(matchingPostRule);
   return irf;
}
```

Das Feld `irf` ist vom Typ `IptablesResultForm`, welches die erzielten Resultate (Matches) für die spätere Ausgabe zuhanden des Users speichert. Ebenfalls ersichtlich aus dem Code-Fragment ist die Tatsache, dass die Behandlung der NAT-Regeln im Standard-Verhalten für die Technologie festgelegt ist, und nicht wie die Behandlung der Filterregeln im Matcher geschieht. Dazu mehr im später folgenden Kapitel über NAT.

6.5.3 Cisco IOS Standard Verhalten

Das Standard-Verhalten von Cisco IOS ist im Vergleich zum iptables-Verhalten relativ simpel. Es kann auf jedem Interface maximal ein `AccessPolicyRuleSet` pro Richtung (in/out) definiert werden und zwar für IPv4 und für IPv6. Das heisst, dass beim Matching-Prozess genau zwei `AccessPolicyRuleSets` relevant sind, nämlich dasjenige auf dem Incoming-Interface in der Richtung IN und dasjenige auf dem Outgoing-Interface in Richtung OUT.

Der Algorithmus aus der `IOSStandardBehviour`-Klasse, um die relevanten `AccessPolicyRuleSets` zu definieren, sieht folgendermassen aus:

```
private void populateFields(Firewall fw, AnalyzerForm af) {

   ipv4Nat = fw.getNatRuleSets().get("TopRuleSet");

   for (AccessPolicyRuleSet aprs:fw.getAccessPolicyRuleSets().values()){
      if (aprs.isTopRuleSet() == true) {
         List<InterfaceDirectionMapping> mappings = aprs.getIfDirMapping();
         Iterator<InterfaceDirectionMapping> iter = mappings.iterator();
         while (iter.hasNext()) {
            InterfaceDirectionMapping m = iter.next();
            if (m.getIface().getName().equals(af.getIncomingInterface())) {
               if (m.getDirection() == Direction.IN
                  || m.getDirection() == Direction.BOTH) {
                  if (aprs.getRuleSetFamily() == RuleSetFamily.IPV4)
                     ipv4In = aprs;
                  else if (aprs.getRuleSetFamily() == RuleSetFamily.IPV6)
                     ipv6In = aprs;
                  break;
               }
            }
            else if (m.getIface().getName().equals(af.getOutgoingInterface()))
            {
               if (m.getDirection() == Direction.OUT
```

```
                || m.getDirection() == Direction.BOTH) {
                if (aprs.getRuleSetFamily() == RuleSetFamily.IPV4)
                    ipv4Out = aprs;
                else if (aprs.getRuleSetFamily() == RuleSetFamily.IPV6)
                    ipv6Out = aprs;
                break;
            }
        }
    }
  }
}
```

Die NAT-Regelbehandlung findet bei Cisco IOS zwischen dem Filtering auf dem Incoming und dem Filtering auf dem Outgoing-Interface statt [8]. Dazu mehr im nächsten Kapitel. Der folgende Code-Ausschnitt zeigt den Ablauf der Traffic-Behandlung auf einem IOS-Gerät:

```
public IOSResultForm processIPv4Traffic(AnalyzerForm af) {
  if (ipv4In != null) {
    Matcher.processRules(ipv4In, af);
    AccessPolicyRule matchingRule =
      (Matcher.getResult().size()>=1)?Matcher.getResult().get(0):null;
    isf.setMatchingInRule(matchingRule);
    isf.setMatchingInRuleSetName(ipv4In.getName());
  }
  if(ipv4Nat != null){
    af = processNat(ipv4Nat, af);
  }
  if (ipv4Out != null) {
    Matcher.processRules(ipv4Out, af);
    AccessPolicyRule matchingRule =
      (Matcher.getResult().size()>=1)?Matcher.getResult().get(1):null;
    isf.setMatchingOutRule(matchingRule);
    isf.setMatchingOutRuleSetName(ipv4Out.getName());
  }
  return isf;
}
```

Was hierbei speziell noch auffällt, ist die ständige Überprüfung auf null. Dies ist sehr unschön und macht den gesamten Algorithmus auch fehleranfälliger. Es würde sich sicher lohnen, hierbei das Null-Object-Pattern einzuführen. Leider reichte auch hier die Zeit nicht mehr dazu, dieses doch etwas grössere Refactoring durchzuführen.

6.5.4 NAT

Grund für die Behandlung der NAT-Regeln in der jeweiligen StandardBehaviour-Klasse ist die Tatsache, dass die Definition von NAT-Regeln bei den Technologien etwas verschieden ist und deshalb auch andere Matching-Algorithmen verwendet werden. Bei IOS müssen die NAT-Regeln z.B. zwingend einem Interface zugeordnet werden, ansonsten werden sie nicht angewendet, während es bei iptables zwar möglich ist, sie einem Interface zuzuordnen, aber nicht notwendig. Bei iptables ist es wiederum möglich, eigene Chains mit NAT-Regeln zu definieren und aus einer built-in-Chain (PREROUTING, OUTPUT, POSTROUTING) bei einem Match in diese zu springen `(-j user-defined-chain)`.

Was uns bei iptables ebenfalls ziemliche Kopfschmerzen bereitete, ist die Möglichkeit in einer NAT-Chain eine Filter-Regel zu definieren. Diese Funktionalität soll es einem Firewall-Administrator erlauben, unerwünschten Traffic so früh wie möglich zu behandeln, insbesondere zu droppen, um die Arbeitsbelastung auf den nachfolgenden Chains zu reduzieren. Leider wurden wir auf diese Möglichkeit erst sehr spät aufmerksam und mussten grössere Anpassungen an der Problem Domain gegen Ende

des Projekts durchführen, um die Funktionalität noch zu implementieren. Zudem wurde dadurch die bereits ziemlich komplexe NAT-Behandlung noch weiter verkompliziert.
Um diese Unterschiede korrekt zu implementieren wurde darum jeweils eine eigene Methode geschrieben, welche die Anwendung der NAT-Regeln für die konkrete Technologie übernimmt.

Behandlung von Filter-Regeln in NAT-Chains aus der Klasse `IptablesStandardBehaviour`:

```
...
if(nr.getAction().getAc() == ActionCommand.DENY ||
nr.getAction().getAc()== ActionCommand.PERMIT)
{
   boolean match = processFilterRuleInNatChain(nr, af);
   if(match){
      irf.setMatchingSpecialRules(Matcher.getResult());
      Matcher.clearResult();
      break;
   }
}
...
private boolean processFilterRuleInNatChain(NatRule nr, AnalyzerForm af) {
   AccessPolicyRule apr = new AccessPolicyRule();
   apr.setAction(nr.getAction());
   apr.setDestination(nr.getOriginalDestination());
   apr.setSource(nr.getOriginalSource());
   apr.setService(nr.getOriginalService());

   AccessPolicyRuleSet aprs = new AccessPolicyRuleSet();
   aprs.addAccessPolicyRule(apr);
   aprs.setRuleSetFamily(RuleSetFamily.IPV4);

   return Matcher.processRules(aprs, af);
}
```

Aus diesem Codefragment geht hervor, wie mit einer Filter-Regel, welche in einer NAT-Chain definiert wurde, umgegangen wird. Vom Parser her wird die Regel zusammen mit den NAT-Regeln in einem NAT-Regelset abgelegt. Wenn die Logik in der NAT-Behandlung feststellt, dass die Action der Regel nicht NAT, sondern PERMIT oder DENY lautet, wird ein neues AccessPolicyRuleSet gebildet, die gefundene Regel angehängt und anschliessend mit dem bekannten Algorithmus aus der Matcher-Klasse behandelt. Wird dabei ein Match gefunden, wird die gefundene Regel als SpecialMatchingRule für die weitere Verwendung abgelegt und die Behandlung der aktuellen NAT-Chain abgebrochen. Wird allerdings kein Match auf die Filterregel gefunden, wird mit der Behandlung der folgenden NAT-Regeln weitergefahren, bis entweder das Ende der Chain erreicht wird oder ein Match erfolgt.

7 Schwierigkeiten

Dieser Abschnitt beschreibt Schwierigkeiten und Probleme, die während des Projekts auftraten.

7.1.1 Unterschiede der verschiedenen Technologien

Eine der grössten Schwierigkeiten bei diesem Projekt waren die Unterschiede zwischen den verschiedenen Technologien, namentlich Cisco IOS, Cisco ASA und iptables. Beispielsweise werden bei Cisco die Regeln (ACLs) den Interfaces zugeordnet und wirken nur auf den jeweiligen Interfaces in der eingestellten Richtung (in, out). Bei iptables ist es freiwillig, ein Incoming- bzw. ein Outgoing Interface für eine Regel auszuwählen und eine Regel ist grundsätzlich auf allen Interfaces wirksam.
Die Frage war dann, wie wir dies in der Problem Domain umsetzen sollen. Hier haben wir jetzt eine Mischform gewählt. Es ist somit möglich, ein AccessPolicyRuleSet der PD an ein Interface zuzuordnen. Es kann aber auch wie bei iptables ein Incoming- bzw. Outgoing-Interface angegeben werden. Dieser Umstand wiederum warf die Frage auf, wie die Auswertung der Regeln dann erfolgen soll. Wir haben dann festgestellt, dass eine richtige Auswertung nur möglich ist, wenn diese an die verschiedenen Technologien angepasst wird und somit eine eigene Auswertung für jede Technologie programmiert wird.

7.1.2 Umfang der Möglichkeiten

Eine weitere Schwierigkeit waren die unglaublich vielfältigen Möglichkeiten der verschiedenen Technologien, vor allem aber von iptables. Da beide Mitarbeiter dieser Projektarbeit vor Beginn der Arbeit noch eher wenig über die verwendeten Technologien wussten, war es sehr schwer oder fast unmöglich, diese grosse Menge an Möglichkeiten in der gegebenen Zeit kennenzulernen. Irgendwann mussten wir uns dann entscheiden, mit der Programmierung zu beginnen, obwohl wir nicht alles bis ins letzte Detail verstanden haben. So kam es dann auch, dass wir während der Programmierung immer wieder feststellten, dass wir bestimmte Aspekte nicht berücksichtigt haben und mussten dann bis beinahe zum Schluss des Projekts immer wieder Änderungen an der Problem Domain vornehmen. Leider hat uns dann auch die Zeit zum Schluss nicht mehr gereicht, um alle Möglichkeiten aller drei Technologien (Cisco IOS, Cisco ASA und iptables) umzusetzen und wir mussten uns auf die wichtigsten Konzepte beschränken. Für die Umsetzung des Parsers für Cisco ASA running-configs hat die Projektdauer von 14 Wochen dann nicht mehr gereicht.

7.1.3 JPA/Hibernate mit SQLite

Das Problem mit SQLite im gesamten Java-Umfeld besteht darin, dass SQLite nicht offiziell unterstützt wird, das heisst es gibt keinen offiziellen JDBC-Treiber für den Zugriff auf SQLite. Man findet im Internet zwar einen JDBC-Treiber für SQLite (http://www.zentus.com/sqlitejdbc/index.html), doch leider mussten wir im Verlauf des Projekts feststellen, dass gewisse Funktionalitäten (noch) nicht implementiert sind. So war es z.B. nicht möglich, ein BLOB zu speichern, da jeweils eine NotYetImplementedException geworfen wurde.

Dazu kommt, dass durch die fehlende unterstützung durch Java auch Hibernate als ORM-Framework SQLite nicht unterstützt. Glücklicherweise haben auch hier findige Leute im Internet Lösungen zur Verfügung gestellt, sprich den SQLiteDialect, welcher von Hibernate benötigt wird, um die SQL-Queries zu formulieren, zum Download bereitgestellt (http://code.google.com/p/hibernate-sqlite/). Wenn man dieses File betrachtet, fällt einem aber bald auf, dass auch hier einige Operationen nicht unterstützt werden.

Es wäre wünschenswert, wenn Sun SQLite in die offizielle Liste der unterstützten relationalen Datenbanken aufnehmen würde und dafür auch einen JDBC-Treiber zur Verfügung stellen würde.

Dasselbe gilt für JBoss, die Entwickler von Hibernate, für die Unterstützung von SQLite als relationale Datenbank im ORM-Framework.

7.1.4 BigInteger

Ein spezifisches SQLite-Problem, welches sich ergab, war das Speichern eines BigIntegers, welcher dazu verwendet wurde, eine IPv6-Addresse zu speichern. Der Versuch, den BigInteger direkt persistieren zu lassen schlug fehl, da der JDBC-Treiber oder Hibernate nicht in der Lage war, einen passenden SQL-Datentypen zu finden. Der nächste Versuch, ihn als ein BLOB zu speichern schlug fehl, weil der JDBC-Treiber eine NotYetImplemented-Exception geworfen hat.

Zu guter Letzt haben wir uns entschieden, die IPv6-Addresse als String zu speichern und jeweils nur wenn benötigt in eine Zahl umzuwandeln.

Leider finden sich allgemein zu SQLite-Problematiken mit Java relativ wenige Lösungen im Internet. Auch Bücher zum Thema sind Mangelware und deshalb gestaltete sich die Lösungssuche in diesem Bereich relativ schwierig.

7.1.5 IPv4/IPv6 Probleme

Bei den IP-Adressen stiessen wir vor allem bei der Validierung auf Probleme. Wenn die Frage, ob eine Benutzereingabe eine gültige IPv4-Adresse ist, noch relativ einfach zu beantworten war, so war es bei IPv6-Adressen schon etwas schwieriger zu beantworten. Wird eine simple Zahl als Eingabe akzeptiert? Wie werden die fehlenden Teile ergänzt? Wird beispielsweise aus der Eingabe 16 die IPv6-Adresse 0000:0000:0000:0000:0000:0000:0000:0016 oder 0016:0000:0000:0000:0000:0000:0000:0000 erstellt?

Zurzeit ist diese Überprüfung auf Eis gelegt. Hier muss in einer späteren Arbeit definitiv noch ein Übereinkommen gefunden werden, welche Eingabeformate als gültig erkannt werden.

Allgemein kann man sagen, dass die Unterstützung von IPv4 wie auch IPv6 einen grossen Aufwand ergeben hat, aber absolut notwendig war, weil in absehbarer Zeit IPv6 zum Standard erhoben wird und deshalb das heute weltweit verbreitete IPv4-Format ablöst.

7.1.6 Tests

Als Schwierigkeit stellten sich auch automatische Tests heraus. Schwierig hierbei ist beispielsweise, dass es theoretisch unendlich viele unterschiedliche Inputdateien geben kann und somit eine nicht endliche Menge an möglichen Testcases. Auch das Testen von Funktionen, welche prüfen, ob sich eine IP-Adresse in einem bestimmten Netzwerk befindet oder nicht, würde das Testen von sehr vielen Kombinationen erfordern, was leistungsstarke Rechner erfordern würde. Hier wurden bisher Tests für einige Grenzfälle geschrieben, um die wahrscheinlichsten Fehler zu verhindern. Da ausführliche Tests des Matching-Algorithmus aus zeitlichen Gründen nicht mehr durchgeführt werden konnten, wird dieses Thema im nächsten Kapitel, „Ausblick“ nochmals angesprochen.

8 Ausblick

Dieses Kapitel gibt einen Überblick, was bisher noch nicht erreicht worden ist und in weiterführenden Projekten denkbar wäre.

8.1.1 iptables

Das Firewall-Analyse-Tool deckt aufgrund des Umfangs der Möglichkeiten von iptables noch nicht die gesamte Funktionalität ab. Weiterentwicklungen sind in diesem Bereich also denkbar. Ein Beispiel sind Markierungen von Paketen in der Mangle-Table und das dazugehörige Filtern nach Markierungen mittels `--mark`.

8.1.2 Phase 2

Aufgrund der Komplexität von Phase 1 und 2 des Projekts sind wir nicht mehr dazu gekommen, die Arbeit in den Bereich Netzwerk-Topologie weiterzuentwickeln, was aber schon in den ersten Wochen des Projekts ersichtlich wurde. In Phase 3 des Projekts geht es um das Erfassen der Topologie mehrerer Firewalls und um das Bestimmen, ob ein bestimmter Netzwerkstrom von A nach B sein Ziel erreicht oder nicht und welche Regeln dabei angewendet werden.

8.1.3 Weitere Formate

Denkbar wäre es, noch weitere Formate zu unterstützen. Allen voran ist hier ASA 8.0 zu nennen, was eigentlich noch zu Phase 1 gehört hätte, aber aufgrund von mangelnder Zeit nicht mehr umgesetzt werden konnte.

8.1.4 GUI/Webinterface

Denkbar und auch bereits an Meetings angesprochen wurde auch ein Webinterface, über welches configs geparst und ausgewertet werden können.

8.1.5 Automatische Tests

Um das Testen der Applikation, insbesondere des Matching-Prozesses, zu vereinfachen und vor allem, um eine grössere Testabdeckung zu erreichen, wäre es sinnvoll, möglichst bald automatisierte Tests, allenfalls mit Unterstützung von JUnit oder einem ähnlichen Testframework, zu schreiben. Das Problem bei diesen automatisierten Tests liegt einfach darin, dass eine Menge Testdaten von Hand erstellt werden müssen. Der Code ist problemlos testbar, aber insbesondere die erwarteten Resultate (IOSResultForm/IptablesResultForm) müssten für jeden Test-Case manuell erfasst werden.

Eine automatisierte Generierung der erwarteten Resultate ist mit unserem jetzigen Wissenstand nicht möglich. Allenfalls wäre die Idee, die erwarteten Resultate aus den Logs auf dem Firewall zu generieren, zu prüfen.

9 Schlussfolgerungen

Unser Tool beherrscht bis jetzt den gesamten relevanten Funktionsumfang von Access Lists auf IOS und einen Grossteil der Funktionen von iptables. Es erlaubt den Import von Konfigurationen von IOS-Firewalls und iptables-Firewalls und damit die darin enthaltenen Daten in ein unabhängiges Format zu bringen.

Die eingelesenen Daten können dazu verwendet werden, vom Benutzer definierte Netzwerkströme zu prozessieren und schlussendlich festzustellen, ob die Firewall den spezifischen Netzwerkstrom erlauben würde oder nicht. Dazu werden die relevanten Filter-Regeln herausgesucht und dem Benutzer für ein Nachvollziehen der Entscheidungen präsentiert. Mit dieser Ausgabe lässt sich eine Aussage machen, durch welche Filter-Regeln ein Netzwerkstrom erlaubt oder verweigert wird.

Was nicht erreicht wurde, ist die Unterstützung für das ASA-Format und es werden auch noch nicht alle Funktioalitäten von iptables, speziell diejenige der Mangle-Tables, unterstützt. Dies liegt insbesondere daran, dass wir den Funktionsumfang der beiden implementierten Formate massiv unterschätzt haben. Auch der Einsatz von uns wenig bis gar nicht bekannten Technologien (JPA/Hibernate und JFlex) war ein Wagnis und benötigte einige Einarbeitungszeit, hat sich schlussendlich aber trotzdem gelohnt, weil damit ein Grundstein für die weitere Entwicklung hin zu einem noch komplexeren Werkzeug gelegt wurde.

Rückblickend würden wir auch einiges anders anpacken. Es wäre wohl sinnvoller gewesen, sich zuerst auf ein unterstütztes Format zu beschränken und die Applikation für dieses Format sauber zu implementieren und vor allem auch zu testen und erst dann weitere Formate hinzuzufügen. Nun existiert zwar ein Tool für IOS und iptables, aber die Korrektheit ist in unseren Augen noch zu wenig getestet. Hier muss in einer Folgearbeit ganz sicher ein grosser Effort geleistet werden.

Trotz allem betrachten wir die erreichten Ergebnisse dieser Semesterarbeit insgesamt als gelungen. Sie bilden eine gute Grundlage, um die Arbeit an einem umfassenden Firewall-Analyse-Tool, welches weitere Formate sowie eine Unterstützung für mehr als eine Firewall im Netzwerk-Setup, fortzusetzen. Für uns war die Arbeit extrem lehrreich und gibt uns viele wichtige Erkenntnisse mit auf den Weg.

III. Projektdokumentation

10 Anforderungsspezifikation

10.1 Allgemeine Beschreibung

10.1.1 Produktfunktion

Das Produkt liest Firewallregeln aus einer Textdatei und speichert diese herstellerunabhängig in einer SQLite-Datenbank (Version 3.6.18) ab. Die Firewallregeln in der Quelldatei können mit Linux iptables 1.4.1.1, der Cisco ASA 8.0- oder der Cisco IOS-12.4 Syntax (ACLs) definiert sein.

10.1.2 Benutzercharakteristik

Die Applikation richtet sich an Netzwerk-Profis, welche grössere Netzwerke mit mehreren oder sehr vielen Firewalls administrieren.

10.1.3 Einschränkungen

- Für die Ausführung der Applikation ist die Installation einer Java Virtual Machine erforderlich.
- In Phase 1 ist die Applikation insofern eingeschränkt, dass noch keine Netzwerk-Topologie abgebildet, sondern nur einzelne Konfigurationen in Form einer Textdatei analysiert werden können.
- Ein GUI wird in Phase 1 noch nicht realisiert.

10.1.4 Abhängigkeiten

Das Projekt wird mit Java realisiert. Somit muss auf dem Rechner, auf welchem das Programm laufen soll, eine Java Virtual Machine installiert sein.

10.2 Spezifische Anforderungen

10.2.1 Effizienz

Verbrauchsverhalten

Die Applikation ist mindestens auf folgenden Betriebssystemen lauffähig, sofern eine Java Virtual Machine installiert ist.

- Windows XP oder neuer
- Linux Distributionen, Ubuntu 9.04, SuSE 11.0, Debian 4.0, Fedora 10 oder neuer
- Mac OS X oder neuer

10.2.2 Benutzbarkeit

Bedienbarkeit

In Phase 1 des Projekts wird das Tool über ein Kommandozeileninterface verfügen. Ein GUI ist nicht vorgesehen.

10.2.3 Änderbarkeit

Prüfbarkeit

Alle Use Cases werden durch automatische Unit-Tests abgedeckt.

Analysierbarkeit

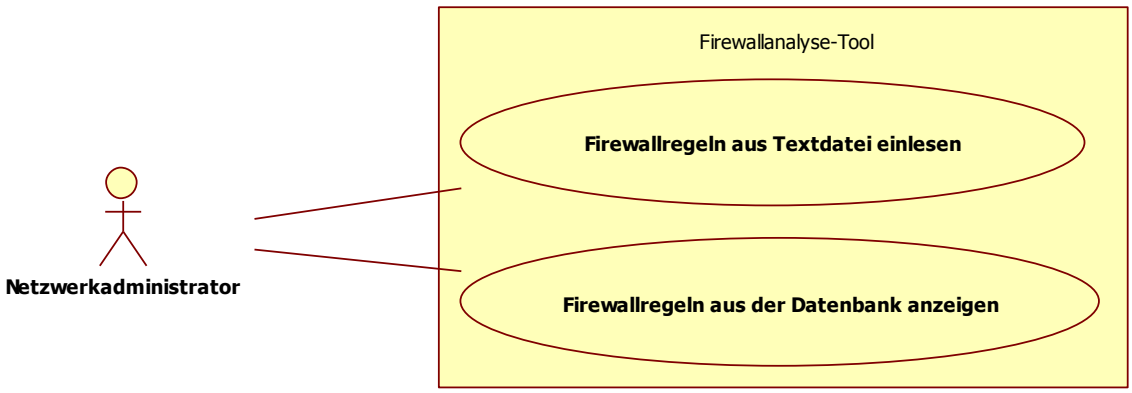
Ergebnisse beim Parsen werden in einer Log-Datei festgehalten.

10.2.4 Lizenz

Die Software ist allgemein verwendbar und soll als Open Source unter der GNU General Public License (GPL) Version 3 veröffentlicht werden.

10.3 Use Cases

Use Case Diagramm



10.3.1 Aktoren

Der primäre Aktor ist der Netzwerkadministrator.

10.3.2 Use Cases (Brief)

UC 1: Firewallregeln aus Textdatei einlesen

Der Netzwerkadministrator bestimmt, welche Datei die einzulesenden Firewallregeln enthält. Nachdem die Datei bekannt ist, liest das System die Datei und parst den Inhalt in ein herstellerunabhängiges Format. Danach werden die Regeln in der Datenbank abgelegt.

UC 2: Analyse

Der Netzwerkadministrator definiert einen Netzwerkstrom. Dieser wird dann mit den in der Datenbank gespeicherten Regeln verglichen. Das System gibt zum Schluss ein Ergebnis aus, ob der spezifizierte Netzwerkverkehr erlaubt oder blockiert wird und ebenfalls, durch welche Regeln dies geschieht.

10.3.3 Use Case 1 (Fully dressed): Firewallregeln einlesen

Scope	The system under design
Level	user-goal
Primary Actor	Netzwerkadministrator
Stakeholders and Interests	Der Netzwerkadministrator möchte Firewallregeln zur späteren einfachen Analyse aus einer Textdatei in ein herstellerunabhängiges

	Format umwandeln.
Preconditions	1. Eine gültige Textdatei mit eventuell vorhandenen Firewallregeln muss zugriffsbereit sein (z.B. auf lokaler Festplatte oder Netzlaufwerk).
Success Guarantee	Die in der angegebenen Textdatei definierten Firewallregeln müssen korrekt in der Datenbank gespeichert sein.
Main Success Scenario	1. Der Netzwerkadministrator übergibt den Dateinamen der Datei mit den Firewallregeln sowie das Format, in welchem die Regeln vorliegen, per Parameter an das Programm ParserCLI. 2. Das System durchsucht die ganze Datei nach Firewallregeln im per Parameter übergebenen Format. 3. Das System ordnet die erkannten Regeln einem Gerät zu. 4. Das System speichert die Regeln in der Datenbank ab.
Alternative Flows	2. Beim Speichern in die Datenbank tritt ein Fehler auf. a. Der aufgetretene Fehler wird in ein Logfile geschrieben. b. Der Benutzer wird informiert und der Vorgang abgebrochen.
Special requirements	-
Technology and data variations	-
Frequency of occurence	häufig, nämlich dann wenn Netzwerkprobleme auftreten, die möglicherweise auf Firewallregeln zurückzuführen sind
Open Issues	-

10.3.4 Use Case 2 (Fully dressed): Analyse

Scope	The system under design
Level	user-goal
Primary Actor	Netzwerkadministrator
Stakeholders and Interests	Der Netzwerkadministrator möchte wissen, ob ein bestimmter Netzwerkstrom über eine Firewall erlaubt oder blockiert wird.
Preconditions	
Success Guarantee	Das System zeigt das korrekte Ergebnis des Vergleichs zwischen dem vom Netzwerkadministrator spezifizierten Netzwerkstrom und den Regeln in der Datenbank an.
Main Success Scenario	1. Der Netzwerkadministrator startet das Programm AnalyzerCLI. 2. Das System erwartet als Eingabe den zu spezifizierenden Datenstrom. Konkret fragt das System folgende Parameter ab: - Firewall, über welche der Verkehr laufen soll - Source IP Address - Destination IP Address - Incoming Interface - Outgoing Interface - Protokoll 3. Das System vergleicht den spezifizierten Netzwerkstrom mit den Regeln in der Datenbank und berechnet das Ergebnis. 4. Das System zeigt als Ergebnis an, ob der spezifizierte Verkehr zugelassen oder blockiert wird und durch welche Regeln.

Extensions	2a. Der Netzwerkadministrator wählt das Protokoll IP. 1. Das System erwartet als Eingabe folgende Parameter: - IP Protokoll Nr. 2b. Der Netzwerkadministrator wählt das Protokoll TCP. 1. Das System erwartet als Eingabe folgende Parameter: - Source Port - Destination Port - Established Flag 2c. Der Netzwerkadministrator wählt das Protokoll UDP. 1. Das System erwartet als Eingabe folgende Parameter: - Source Port - Destination Port 2d. Der Netzwerkadministrator wählt das Protokoll ICMP. 1. Das System erwartet als Eingabe folgende Parameter: - ICMP-Typ - ICMP-Code
special requirements	-
Technology and data variations	-
Frequency of occurence	Je nach Grösse des Netzwerks auch mehrmals täglich
Open Issues	-

11 Projektplan

11.1 Projektübersicht

Es geht darum, einen Proof-of-Concept einer Applikation zu entwickeln, die es einem Netzwerkadministrator erlaubt, seine Firewall-Konfigurationen zu analysieren und den Weg eines Datenpaketes durch einen Firewall zu simulieren.

In einer ersten Phase wird die Materie von Firewall-Konfigurationen erarbeitet. Dies geschieht anhand von konkreten Szenarios, welche ausgearbeitet und anschliessend auf Test-Firewalls durchgespielt werden. Im Anschluss daran soll eine Applikation realisiert werden, welche es erlaubt, die Firewall-Konfigurationen verschiedener Standards und Hersteller in ein formatunabhängiges Regelset zu parsen. Ist dieser Schritt realisiert, geht es in einer zweiten Phase darum, den Matching-Prozess von einem Test-Netzwerkstrom auf die Firewall zu implementieren.

In einem letzten Schritt wären die beiden Werkzeuge aus den Phasen 1 und 2 zu einem Werkzeug zusammenzufassen, um die gesamte gewünschte Funktionalität zu erreichen.

Die Erkenntnisse bezüglich einheitlichem Regelset und und Matching-Prozess sollen in einem technischen Bericht zum Schluss des Projekts festgehalten werdenFalls noch Zeit vorhanden ist, wäre ein Vorabklärung zum Thema "Automatische Erfassung einer Netzwerktopologie" der nächste Schritt der Arbeit.

11.1.1 Zweck und Ziel

Zweck dieser Arbeit ist es, die Grundlage zu schaffen, um eine mächtiges Werkzeug im Bereich Firewall-Analyse zu entwickeln. Dieses Werkzeug soll es Netzwerk-Administratoren erlauben, Netzwerkverkehr in ihrem Netzwerk oder in ihr Netzwerk zu simulieren und mögliche Probleme, sprich Firewalls, die den Verkehr eventuell blockieren, zu detektieren.

Konkretes Ziel der Arbeit ist es, die Abbildung von Firewall-Regeln verschiedenen Ursprungs auf ein formatunabhängiges Regelset und die Implementation der Matching-Algorithmen für die jeweiligen Formate.

11.1.2 Annahmen und Einschränkungen

Zur Zeit wird davon ausgegangen, dass die Firewall-Konfigurationen (Regeln, etc.) in Form einer lesbaren Datei zur Verfügung stehen. Dies kann z.B. eine in ein Text-File abgelegte running-config eines Cisco-Firewalls sein.Desweiteren einigte man sich im Verlauf der ersten Sitzung darauf, den Fokus in der ersten Phase auf das Verarbeiten von Konfigurationen von iptables-, von ASA 8.0- und von IOS-basierenden Geräten zu legen.

11.2 Projektorganisation

Gieri Kohler und Christoph Rebsamen sind beide gleichberechtigte Teampartner in diesem Projekt. Gemeinsame Arbeitszeit ist jeweils dienstags den ganzen Tag sowie Freitagmorgen zwischen 10:00 und 12:00 Uhr.

11.2.1 Externe Schnittstellen

Die Arbeit wird direkt durch Roman Ammann vom Institute for Networked Solutions (INS) betreut. Prof. Beat Stettler vom INS zeichnet für diese Studienarbeit verantwortlich.

11.3 Verantwortlichkeiten

Im Verlauf des Projekts konzentriert sich Gieri Kohler auf die Entwicklung der Parser für die verschiedenen Formate. Christoph Rebsamen implementiert die Persistenz-Schicht und die Matching-Algorithmen für die Filter und NAT-Regeln.Die Problem-Domain und die Command-Line-Interfaces werden gemeinsam entwickelt.

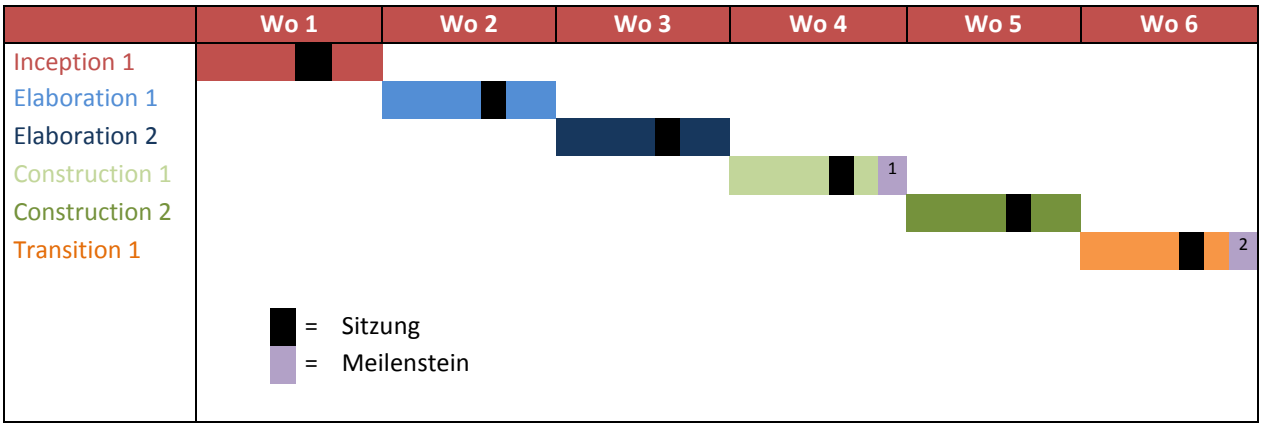
11.4 Managementabläufe

11.4.1 Zeitvoranschlag

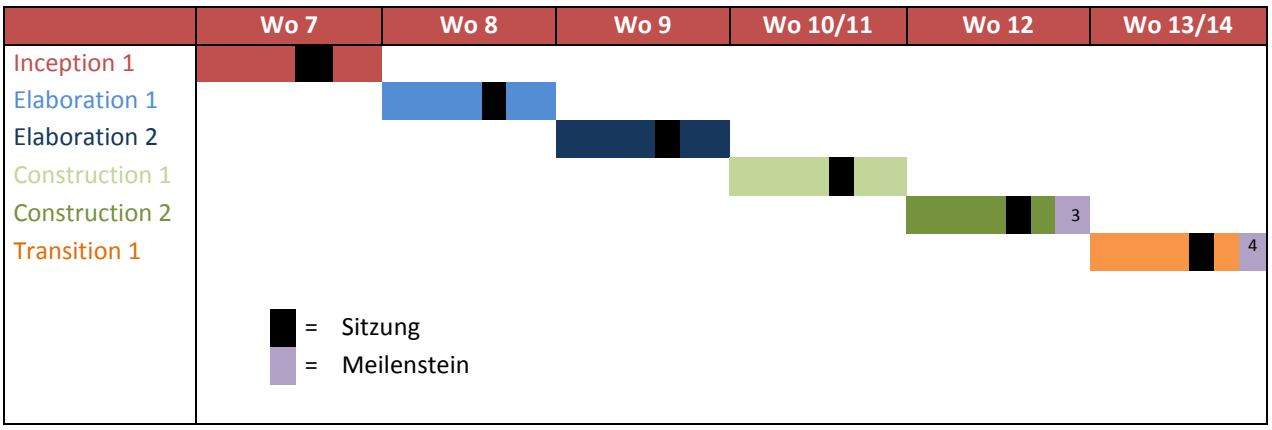
Für die Bearbeitung der Aufgabe steht die Zeit bis zum 18. Dezember 2009 zur Verfügung. Dies ist durch die Bestimmungen zur Studienarbeit festgelegt und kann nicht angepasst werden.

11.4.2 Projektplan

Zeitplan Phase 1



Zeitplan Phase 2



Meilensteine

Meilenstein	Beschreibung	Arbeitsresultate	Zeitpunkt
MS 1	Präsentation „Formatunabhängiges Regelset“	Präsentation zu den Erkenntnissen aus den durchgespielten Firewall-Szenarios und zum formatunabhängigen Regelset,	Wo 3/4

		welches entwickelt werden soll	
MS 2	Prototyp Parser	Lauffähiger Prototyp zum Verarbeiten von verschiedenen Firewall-Konfigurationen	Woche 6
MS 3	Prototyp Topologie	Lauffähiger Prototyp zur Erfassung der relevanten Netzwerk-Topologie	Woche 12
MS 4	Abgabe	Komplette Studienarbeit-Dokumentation	Woche 14

Iterationsplanung Phase 1

Iteration	Arbeitspakete	Dauer in Wochen
Inception 1		1
Elaboration 1	Projektplan Anforderungspezifikation Phase 1 Test-Szenarios	1
Elaboration 2	Domainanalyse Architektur Risikomanagement Projekt-Automation	1
Construction 1	Datenhaltung Unit-Tests	1
Construction 2	Proof-of-Concept Parsing Unit-Tests	1
Transition 1	Dokumentation Parser Systemtest	1

Iterationsplanung Phase 2

Iteration	Arbeitspakete	Dauer in Wochen
Inception 1		0
Elaboration 1	Anforderungspezifikation Phase 2	1
Elaboration 2	Domainanalyse Architektur Projekt-Automation	1
Construction 1	Datenhaltung Unit-Tests	1
Construction 2	Proof-of-Concept Topologie Unit-Tests	1
Transition 1	Dokumentation Topologie Semesterarbeit-Dokumentation Systemtest	2

Besprechungen

Das wöchentliche Meeting mit dem Betreuer und dem Projektverantwortlichen seitens des INS findet jeweils am Freitag um 11:15 statt. In der Regel trifft man sich im Raum 6.001.

11.5 Arbeitspakete

11.5.1 Phase 1

Nr.	Name	Beschreibung	Iteration	Wer
0x	**Projektmanagement**			
01	Projektplan	Umfassender Plan zum zeitlichen Ablauf des Projekts, zu Zuständigkeiten und zu Meilensteinen.	Elaboration 1	CR
02	Risikomanagement	Risikomanagement für das Projekt erstellen.	Elaboration 1	Team
03	Sitzungen	Wöchentlich findet eine Sitzung des Projektteams mit den Betreuern statt	Inception 1 – Transition 1	Team
1x	**Requirements**			
11	Anforderungspezifikation	Spezifikation zu den Anforderungen des „Kunden“ an die fertige Applikation. Dokument wird vom Kunden abgenommen.	Elaboration 1	GK
2x	**Analyse**			
21	Test-Szenarios	Es sollen Szenarios zur Firewall-Konfiguration entwickelt werden, die in einem Lab angewendet und konfiguriert werden können. Damit soll ein Verständnis für Firewall-Konfigurationen erreicht werden.	Elaboration 1	Team
22	Domain-Analyse	Die Erkenntnisse zur Umsetzung des formatunabhängigen Regelsets soll in einem Dokument niedergeschrieben werden. Es dient als Grundlage für die Entwicklung der Problem-Domain	Elaboration 2	Team
3x	**Implementation**			
31	Projektautomation	Der Build-Prozess der Applikation ist mittels eines Build-Tools (in unserem Fall Apache ANT) zu automatisieren	Elaboration 2	CR
32	Datenhaltung	Implementieren der Datenhaltung	Construction 1	Team
33	Unit-Tests DH	Implementierung der Unit-Tests zur Datenhaltung. Vorgehen gemäss test-driven Development	Construction 1	Team
34	Proof-of-Concept Parsing	Implementierung des Parsens der Firewall-Konfigurationen und der Abbildung in eine formatunabhängiges Regelset im Stile eines Proof-of-Concept.	Construction 2	Team
35	Unit-Tests Parsing	Implementierung der Unit-Tests zu Parsing-Teil der Applikation. Vorgehen gemäss test-	Construction 2	Team

		driven Development		
4x	**Testing**			
41	Systemtests	Testen der Applikation als Ganzes	Transition 1	CR
5x	**Dokumentation**			
51	Dokumentation Parser	Der Applikationsteil Parsing (Resultat Phase 1) ist für die Semesterarbeit vollständig und ausführlich zu dokumentieren.	Transition 1	Team

11.5.2 Phase 2

Nr.	Name	Beschreibung	Iteration	Wer
0x	**Projektmanagement**			
03	Sitzungen	Wöchentlich findet eine Sitzung des Projektteams mit den Betreuern statt	Inception 1 – Transition 1	Team
2x	**Analyse**			
22	Domain-Analyse	Die Erkenntnisse zur Umsetzung des formatunabhängigen Regelsets soll in einem Dokument niedergeschrieben werden. Es dient als Grundlage für die Entwicklung der Problem-Domain	Elaboration 2	Team
3x	**Implementation**			
34	Proof-of-Concept Matcher	Implementierung der Matching-Prozesse auf den Firewalls im Stile eines Proof-of-Concept.	Construction 2	Team
35	Unit-Tests Matcher	Implementierung der Unit-Tests zu Mather-Teil der Applikation. Vorgehen gemäss test-driven Development	Construction 2	Team
4x	**Testing**			
41	Systemtests	Testen der Applikation als Ganzes	Transition 1	Team
5x	**Dokumentation**			
51	SA-Dokumentation	Die gemäss Vorlagen verlangten Dokumente für die Semesterarbeit sind zu erstellen	Transition 1	Team

11.6 Infrastruktur

11.6.1 Räumlichkeiten

Zur Verfügung steht in erster Linie der Studienarbeits-Raum 1.258.

Zum Einarbeiten in die Materie der Firewall-Konfiguration steht nach momentaner Information ein Netzwerk-Labor des Institute for Networked Solutions (INS) zur Verfügung.

Desweiteren können selbstverständlich alle öffentlich verfügbaren Räume der HSR genutzt werden.

11.6.2 Hardware

- Firewalls
  - Ubuntu 9.04 mit Iptables 1.4.1.1 in einer VirtualBox
  - Cisco 2811 Integrated Services Router
  - Cisco 2821 Integrated Services Router
  - Cisco ASA 5520
  - Cisco PIX 515e Security Appliance
- Persönlicher Laptop
- Labor-Rechner der HSR (\\vf3.hsr.ch\skripte\Informatik\Fachbereich\SA-DA-BA\Labor Image Guide.pdf)

11.6.3 Software

- Betriebssysteme
  - Microsoft Windows XP SP3
  - Microsoft Windows 7
- Entwicklungsumgebung
  - Eclipse Ganymede
  - JUnit
  - Ant
- Versionsverwaltung
  - TortoiseSVN
- Dokumentation
  - Microsoft Office 2007

11.6.4 Kommunikation

Kommunikation innerhalb des Projekt-Teams findet mündlich oder per E-Mail/Instant-Messaging-Service statt.

Die Kommunikation mit dem Betreuer oder dem Verantwortlichen findet ebenfalls mündlich oder per E-Mail statt.

Name	E-Mail	Telefon
Gieri Kohler	gkohler@hsr.ch	078 809 38 94
Christoph Rebsamen	crebsame@hsr.ch	076 502 88 45
Roman Ammann (Betreuer)	rammann@ins.hsr.ch	078 825 00 53
Prof. Beat Stettler (Verantwortlicher)	bstettler@hsr.ch	079 218 79 15

11.6.5 Backup

Der verwendete SVN-Server der HSR wird zwischen Montag und Freitag täglich durch die Informatik-Dienste der HSR gesichert.

Das Backup des persönlichen Laptops und des persönlichen Labor-Rechners an der HSR ist Sache des einzelnen Projekt-Mitarbeiters und geschieht durch manuelles Kopieren der relevanten Daten auf ein transportables Medium (USB-Stick/-Festplatte o. ä.). Das persönliche Backup soll im Minimum einmal wöchentlich durchgeführt werden.

11.7 Qualitätsmassnahmen

11.7.1 Dokumentation

Die Dokumentation wird durch die Team-Mitglieder in gegenseitiger Absprache stets aktuell gehalten. Änderungen an einem Dokument sind in der Änderungsgeschichte des Dokumentes festzuhalten. Folgende Regeln gelten für das Versionsmanagement:

- Das Dokument wird mit Version 1.0 erstellt
- Wird das Dokument mit einem neuen Text oder Kapitel ergänzt, erhöht sich die Versionsnummer auf 1.x
- Werden in einem Dokument lediglich Rechtschreibefehler oder Formulierungen korrigiert, d.h. es werden dem Dokument keine neuen Informationen hinzugefügt, erhöht sich die Versionsnummer auf 1.0.x
- Wird ein Dokument von Grund auf neu erstellt, erhöht sich die Versionsnummer auf 2.0, resp. x.0.
- Die Dokumentation des Codes erfolgt durch Javadoc-Kommentare direkt im jeweiligen Java-File. Dies erlaubt am Schluss ein Exportieren der Code-Dokumentation in eine HTML-Dokumentation.

11.7.2 Code

Grundsätzlich ist der Code verständlich und selbsterklärend zu erstellen. Durch stetiges Refactoring wird erreicht, dass der Code leicht zu verstehen bleibt. Das Kommentieren von Code wird nur sparsam eingesetzt.

Der erstellte Java-Code soll den Java Code Conventions[9] entsprechen.

11.7.3 Sitzungsprotokolle

Nach jeder Sitzung wird durch ein Team-Mitglied ein Sitzungsprotokoll verfasst. Darin werden insbesondere getroffene Abmachungen und Entscheide sowie Tasks bis zur nächsten Sitzung festgehalten. Das Sitzungsprotokoll wird bis spätestens zwei Tage nach der Sitzung per E-Mail an alle Beteiligten des Projektes gesendet.

Somit ist gewährleistet, dass getroffenen Entscheide und aktuelle Tasks für alle Beteiligten verbindlich festgehalten und jederzeit verfügbar sind.

11.7.4 Reviews

Reviews Dokumente

Jedes Dokument wird neben dem Autor vom anderen Teammitglied im Sinne eines Peer-Reviews durchgelesen. Allfällige Vorschläge zur Verbesserung werden mit dem Autor besprochen und anschliessend in das Dokument eingebaut.

Im Dokument wird mit Datum und Kürzel vermerkt, wer wann das Review durchgeführt hat.

[9] http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html

Reviews Code

Mindestens vor Beendigung des Projektes sollte der gesamte Code durch die Team-Mitglieder angeschaut und gegebenenfalls überarbeitet werden. Es geht insbesondere darum, die Einhaltung der gängigen Codierungsrichtlinien und die Verständlichkeit des Codes zu überprüfen.

Selbstverständlich sollte der Code auch während der Entwicklung durch stetiges Refactoring eine hohe Qualität besitzen.

11.7.5 Tests

Unit-Tests

Gemäss dem Vorgehen im Test-Driven-Development werden die Unit-Tests vor dem Entwickeln des eigentlichen Programmcodes geschrieben. Unit-Tests sind als Blackbox-Tests zu implementieren, sprich sie testen die Auswirkungen (Rückgabewerte, Veränderung an Dateien, Datenbanken, o.ä.) von Methoden und nicht die Internas einer Methode.

Code sollte nur in das Versionsverwaltungssystem eingecheckt werden, wenn nach erfolgter Änderungen oder Neuerstellung noch alle Unit-Tests erfolgreich verlaufen (Green Bar).

Allenfalls ist der Einsatz von Mock-Objekten zur Simulation von Modulen zu prüfen um die Unit-Tests durchführen zu können

System-Tests

Basierend auf den Use-Cases für das System werden Systemtests durchgeführt. Diese finden jeweils kurz vor einem Release statt und sollen sicherstellen, dass das System den funktionalen Anforderungen gemäss Anforderungsspezifikation genügt.

Usability-Tests

Usability-Tests werden nicht stark forciert, da es in diesem Projekt in erster Linie um einen Proof-of-Concept geht.

11.7.6 Versionsverwaltungssystem

Dokumente sowie Quellcode werden mit Subversion auf dieselbe Version gebracht und verwaltet. Es sollen nur Dokumente in sinnvollem Zwischenstand und Quellcodedateien in lauffähigem und getestetem Zustand in das Versionsverwaltungssystem eingecheckt werden. Die Verwendung eines Versionsverwaltungssystems ermöglicht allen Teammitgliedern den Zugriff auf die jeweils aktuellste Dateiversion unabhängig von Ort und Zeit. Ältere Versionen können bei Bedarf, z.B. nach versehentlichem Löschen oder fälschlichem Ändern einer Datei wiederhergestellt werden.

11.7.7 Automatisierung

Zur Automatisierung des Projektes, insbesondere das automatische Erstellen von installierbaren Software-Paketen wird ein Build-Management-Tool, in unserem Fall Apache Ant, eingesetzt. Damit wollen wir erreichen, dass immer gleiche Schritte zur Erstellung der Software nicht manuell vorgenommen werden müssen und erreichen damit, dass die Fehlerquote bei den genannten Schritten tief bleibt. Zudem vereinfacht der Einsatz eines solchen Tools das Erstellen von CRISP-Builds[10] welche im Rahmen des Pragmatic Programming angestrebt werden.

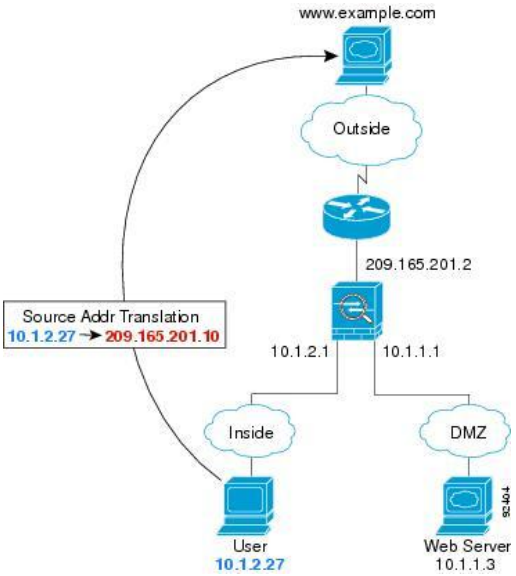
[10] **C**omplete, **R**epeatable, **I**nformative, **S**chedulable, **P**ortable

12 Szenarios

12.1 Einführung

Folgende fünf Fälle stellen realistische Szenarios dar, welche auf dem CCIE Rental Rack durch Cisco IOS 12.4 und Cisco ASA 8.0 simuliert werden. Die Szenarios werden zusätzlich durch iptables innerhalb einer VirtualBox abgebildet. Zweck dieser Szenarios ist die Einarbeitung der Teammitglieder in die verschiedenen Technologien.

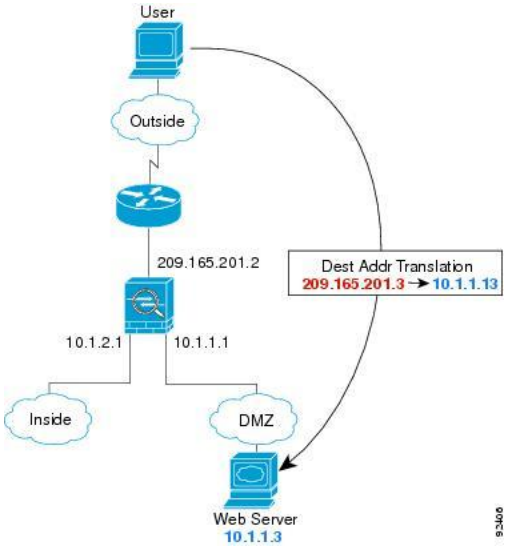
12.2 Szenario 1: Ein interner Benutzer besucht einen Webserver



Der interne Benutzer „User“ besucht den externen Webserver „www.example.com“. Dabei wird die interne IP-Adresse des Users (Inside Local) in eine externe (Inside Global) übersetzt. Der Zugriff ist erlaubt.

to / from	in	dmz	out
In		permit all	permit all
Dmz	deny all*		permit all
Out	deny all*	deny all*	

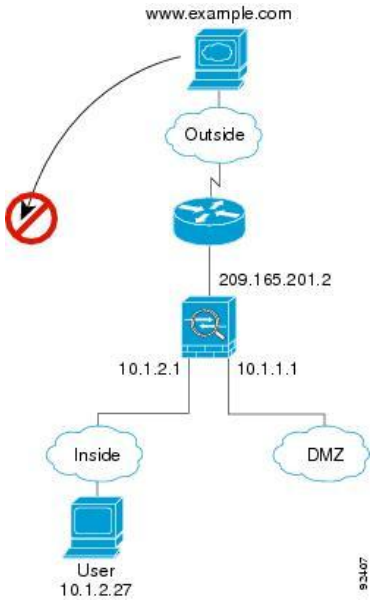
12.3 Szenario 2: Ein externer Benutzer besucht einen Webserver in der DMZ



Der externe Benutzer „User“ besucht den Webserver in der DMZ mit der IP-Adresse 10.1.1.3. Der User spricht den Webserver über seine Inside Global-Adresse 209.165.201.3 an. Der Zugriff ist erlaubt.

to / from	in	dmz	Out
In		permit all	permit all
Dmz	deny all*		permit all
Out	deny all*	permit tcp port 80	

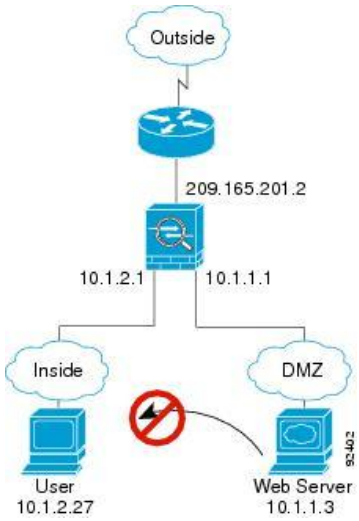
12.4 Szenario 3: Zugriffsversuch von aussen auf einen Host im internen Bereich



Ein Benutzer versucht von aussen auf das interne Netzwerk zuzugreifen zuzugreifen. Der Zugriff muss geblockt werden.

to / from	In	dmz	out
In		permit all	permit all
Dmz	deny all*		permit all
Out	deny all*	deny all*	

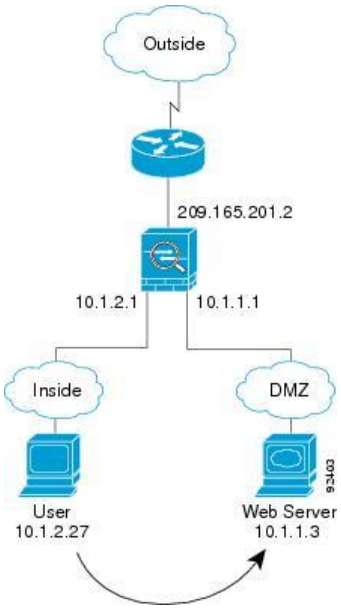
12.5 Szenario 4: Zugriffsversuch aus der DMZ auf einen internen Host



Der Zugriff aus der DMZ ins interne Netz muss geblockt werden.

to / from	in	dmz	out
In		permit all	permit all
Dmz	deny all*		permit all
Out	deny all*	deny all*	

12.6 Szenario 5: Ein interner Benutzer besucht einen Webserver in der DMZ



Ein Benutzer aus dem internen Netz muss auf einen Webserver in der DMZ zugreifen können.

to / from	in	dmz	out
In		permit all	permit all
Dmz	deny all*		permit all
Out	deny all*	deny all*	

*neue Verbindungen werden geblockt. Wurde eine Verbindung von innen initiiert, werden die Antworten erlaubt. Beispielsweise kann ein Client von innen auf einen Webserver zugreifen. Es wird eine Verbindung gespeichert und die Antwort des Webservers wird erlaubt.

Quelle der Bilder im Kapitel Szenarios: http://www.cisco.com/en/US/docs/security/asa/asa81/config/guide/fwmode.html (24.09.2009)

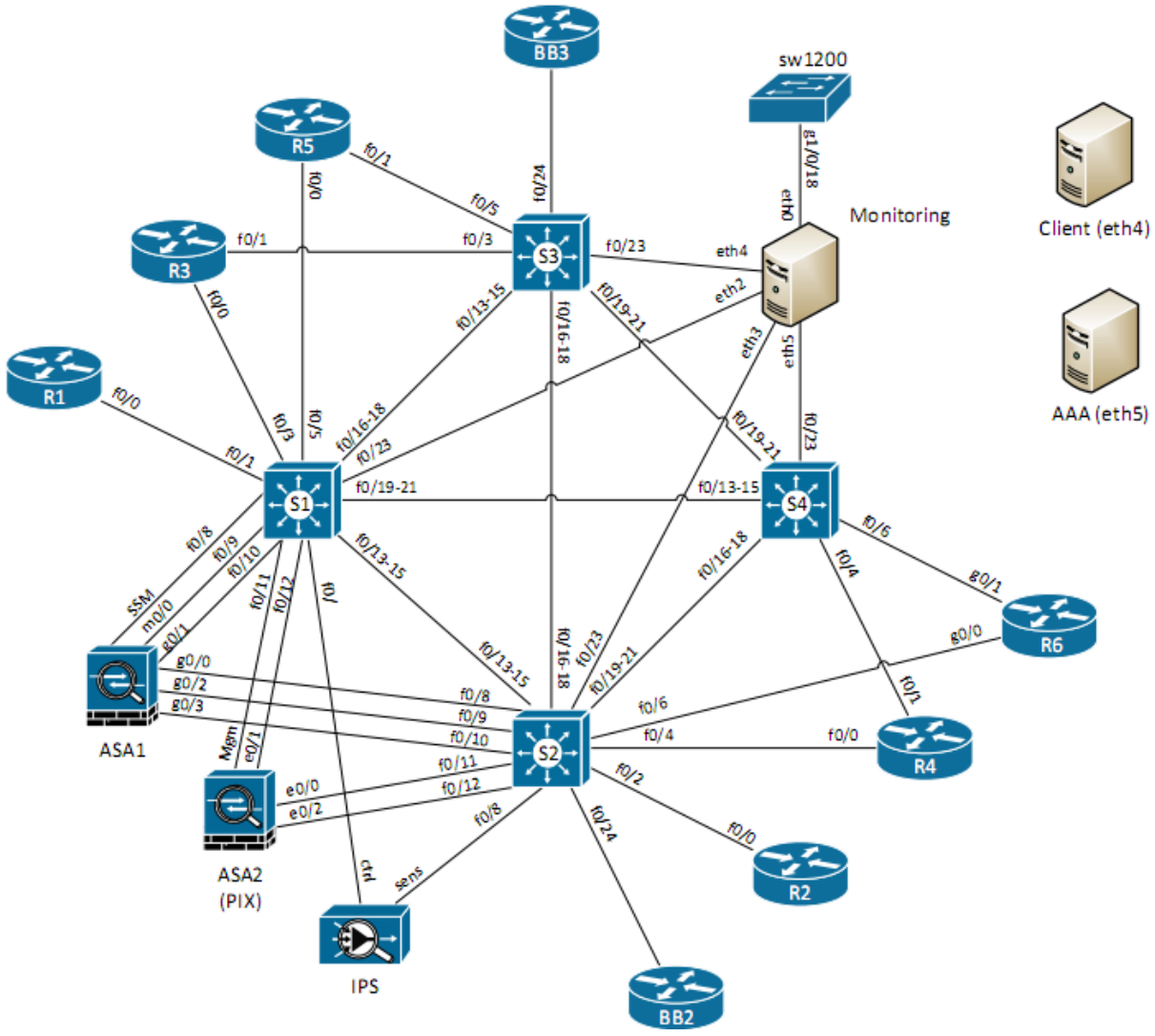
13 Realisation der Szenarios

13.1 Übersicht

Dieses Dokument beschreibt die Realisation der aufgestellten Szenarios (siehe Kapitel 12 Szenarios) in der Übungsumgebung, bestehend aus dem CCIE Rental Rack und einer mittels VirtualBox virtualisierten Umgebung. Inhalt des Dokuments sind hauptsächlich alle Konfigurationsdateien der verwendeten Geräte sowie für jede Technologie jeweils eine bildliche Übersicht über die Anordnung aller Geräte.

13.2 CCIE Rental Rack-Umgebung

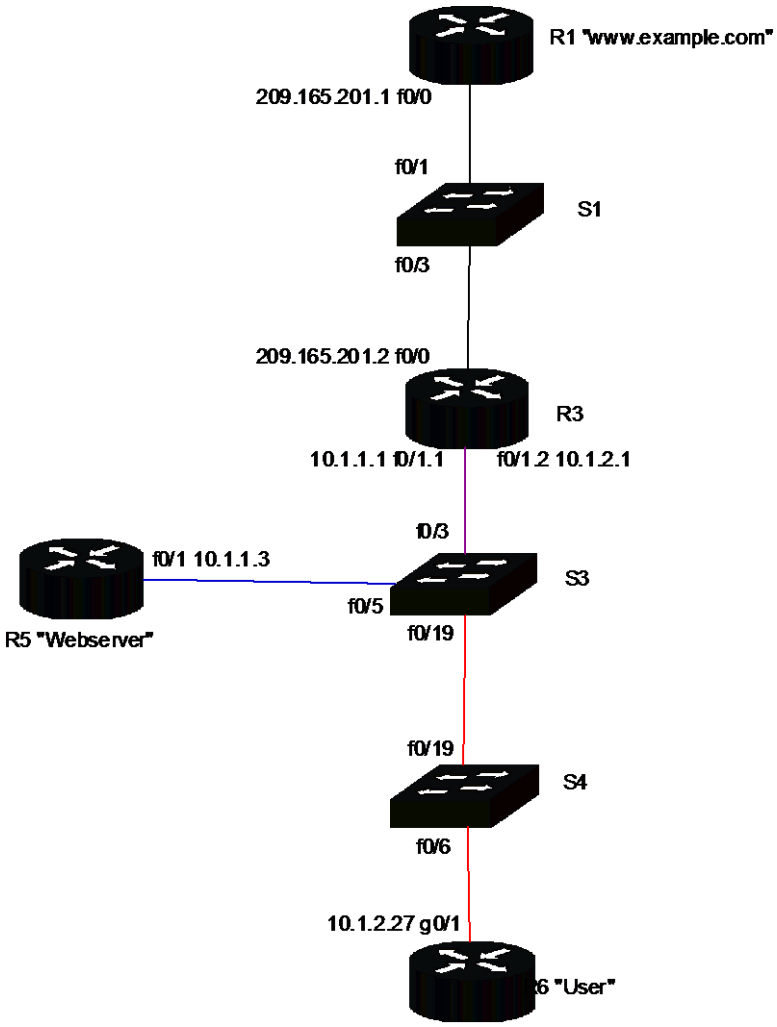
Das CCIE Rental Rack wird uns für die Studienarbeit als Übungsumgebung zur Verfügung gestellt und kann nach Bedarf reserviert werden. Folgende Abbildung zeigt das Rack mit seinen Komponenten und die Anordnung derselben. Die Umgebung wurde verwendet, um die Szenarios mittels Cisco IOS und Cisco ASA zu realisieren. Die in Abbildung 3.1 und 4.1 verwendeten Portnummern und Gerätenamen beziehen sich auf die Abbildung auf dieser Seite.



CCIE Rental Rack

13.3 Cisco IOS

13.3.1 Übersicht



Übersicht der Realisiation mittels Cisco IOS

VLAN 10 (DMZ)

VLAN 20 (IN)

Trunk

13.3.2 Initialkonfiguration

R1

```
hostname R1
!
interface FastEthernet0/0
  ip address 209.165.201.1 255.255.255.0
  no shutdown
!
router ospf 1
  network 209.165.201.0 0.0.0.255 area 0
!
ip http server
```

R3

```
hostname R3
!
interface FastEthernet0/0
  ip address 209.165.201.2 255.255.255.0
  no shutdown
!
interface FastEthernet0/1
  no shutdown
!
interface FastEthernet0/1.1
  encapsulation dot1q 10
  ip address 10.1.1.1 255.255.255.0
  no shutdown
!
interface FastEthernet 0/1.2
  encapsulation dot1q 20
  ip address 10.1.2.1 255.255.255.0
  no shutdown
!
router ospf 1
  network 10.0.0.0 0.255.255.255 area 0
  network 209.0.0.0 0.255.255.255 area 0
!
```

R5

```
hostname R5
!
interface FastEthernet0/1
  ip address 10.1.1.3 255.255.255.0
  no shutdown
!
router ospf 1
  network 10.0.0.0 0.255.255.255 area 0
!
ip http server
```

R6

```
hostname R6
!
interface GigabitEthernet0/1
  ip address 10.1.2.27 255.255.255.0
  no shutdown
!
router ospf 1
  network 10.0.0.0 0.255.255.255 area 0
```

S1

```
hostname S1
interface range FastEthernet0/1-24
  shutdown
!
interface FastEthernet0/1
  no shutdown
!
interface FastEthernet0/3
  no shutdown
```

S3

```
hostname S3
interface range FastEthernet 0/1-24
   shutdown
!
interface FastEthernet0/3
   switchport trunk encapsulation dot1q
   switchport mode trunk
   no shutdown
!
interface FastEthernet0/5
   switchport access vlan 10
   no shutdown
!
interface FastEthernet0/19
   switchport access vlan 20
   no shutdown
```

S4

```
hostname S4
!
interface FastEthern0/1-24
   shutdown
!
interface FastEthernet0/6
   switchport access vlan 20
   no shutdown
!
interface FastEthernet0/19
   switchport access vlan 20
   no shutdown
```

13.3.3 Konfiguration Szenario 1

Ein interner Benutzer besucht einen externen Webserver.

R3

```
interface FastEthernet 0/0
   ip access-group 100 in
   ip nat outside

interface FastEthernet0/1.2
   ip nat inside

access-list 100 permit ospf any any
access-list 100 permit tcp 209.165.201.1 0.0.0.0 eq www any
access-list 1 permit 10.1.2.0 0.0.0.255
ip nat inside source list 1 interface FastEthernet0/0 overload
```

13.3.4 Konfiguration Szenario 2

Ein externer Benutzer besucht einen Webserver in der DMZ.

R3

```
interface FastEthernet 0/0
   ip access-group 101 in
```

```
  ip nat outside

interface FastEthernet0/1.1
  ip nat inside

access-list 101 permit ospf any any
access-list 101 permit tcp any 10.1.1.3 0.0.0.0 eq www
ip nat outside source static 209.165.201.1 10.1.1.13
```

13.3.5 Konfiguration Szenario 3

Blockiert Zugriffsversuch von aussen auf einen Host im internen Bereich

R3

```
interface FastEthernet 0/0
  ip access-group 102 in

access-list 102 permit ospf any any
access-list 102 deny any 10.1.2.0 0.0.0.255
```

13.3.6 Konfiguration Szenario 4

Blockiert Zugriffsversuch aus der DMZ auf einen internen Host

R3

```
interface FastEthernet0/1.1
  ip access-group 103 in

access-list 103 permit ospf any any
access-list 103 deny 10.1.1.0 0.0.0.255 10.1.2.0 0.0.0.255
```

13.3.7 Konfiguration Szenario 5

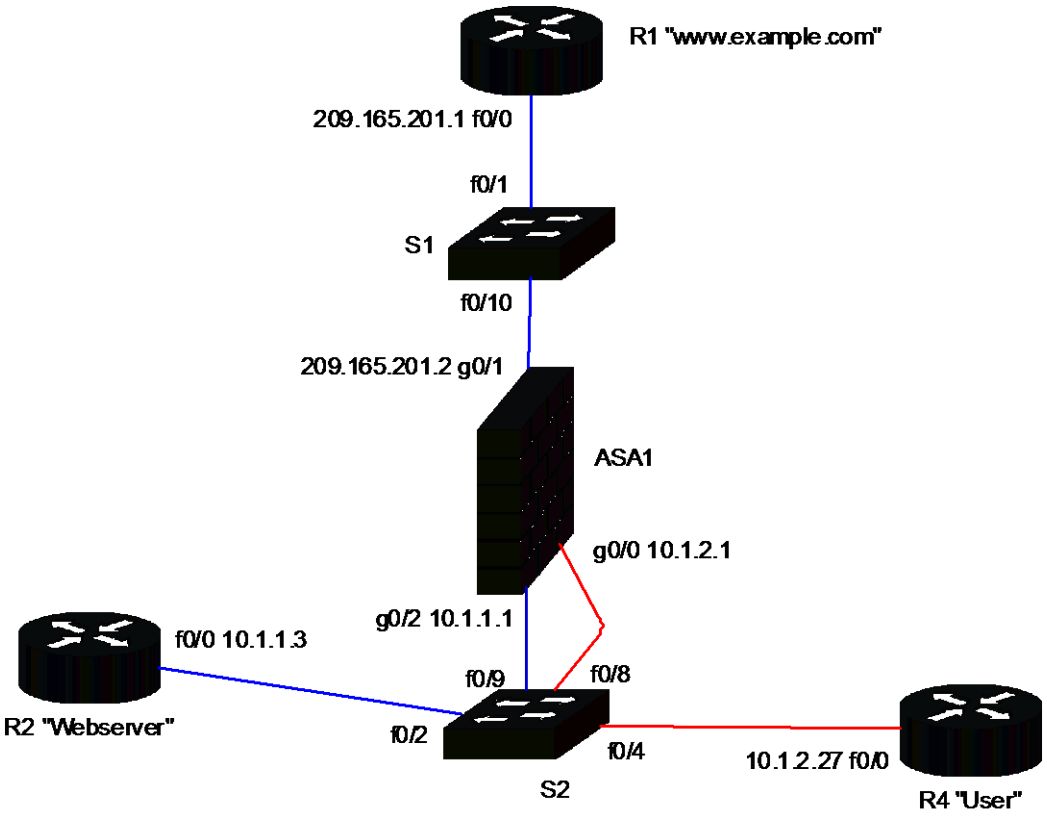
Ein interner Benutzer besucht einen Webserver in der DMZ

R3

```
interface FastEthernet0/1.2
  ip access-group 104 in

access-list 104 permit ospf any any
access-list 104 permit tcp any eq www any
```

13.4 Cisco ASA

13.4.1 Übersicht



Übersicht der Realisation mittels Cisco ASA

VLAN 10 (DMZ)

VLAN 20 (IN)

13.4.2 Initialkonfiguration

ASA1

```
hostname ASA1
!
interface GigabitEthernet0/0
   nameif inside
   security-level 100
   ip address 10.1.2.1 255.255.255.0
   no shutdown
!
interface GigabitEthernet0/1
   nameif outside
   security-level 0
   ip address 209.165.201.2 255.255.255.0
   no shutdown
!
interface GigabitEthernet0/2
   nameif dmz
   security-l§evel 50
   ip address 10.1.1.1 255.255.255.0
   no shutdown
```

```
!
access-list 100 extended permit ip 10.1.1.0 255.255.255.0 any
access-list 100 extended permit ip 10.1.2.0 255.255.255.0 any
nat (dmz) 0 access-list 100
nat (inside) 0 access-list 100
!
router ospf 1
  network 0.0.0.0 0.0.0.0 area 0
```

R1

```
hostname R1
!
interface FastEthernet0/0
  ip address 209.165.201.1 255.255.255.0
  no shutdown
!
router ospf 1
  network 209.165.201.0 0.0.0.255 area 0
  no shutdown
!
ip http server
```

R2

```
hostname R2
!
interface FastEthernet0/0
  ip address 10.1.1.3 255.255.255.0
  no shutdown
!
router ospf 1
  network 10.0.0.0 0.255.255.255 area 0
!
ip http server
```

R4

```
hostname R4
!
interface FastEthernet0/0
  ip address 10.1.2.27 255.255.255.0
  no shutdown
!
router ospf 1
  network 10.0.0.0 0.255.255.255 area 0
```

S1

```
hostname S1
!
interface range FastEthernet0/1-24
  shutdown

interface FastEthernet0/1
  no shutdown
!
interface FastEthernet0/10
no shutdown
!
```

S2

```
hostname S2
interface range FastEthernet0/1-24
interface FastEthernet0/2
  switchport access vlan 10
  no shutdown
!
interface FastEthernet0/4
  switchport access vlan 20
  no shutdown
!
interface FastEthernet0/8
  switchport access vlan 20
  no shutdown
!
interface FastEthernet0/9
  switchport access vlan 10
  no shutdown
```

13.4.3 Konfiguration Szenario 1

Ein interner Benutzer besucht einen externen Webserver. Seine interne IP-Adresse (Inside Local) wird in eine externe abgebildet (Inside Global).

ASA1

```
access-list 100 extended permit ip 10.1.2.0 255.255.255.0 209.165.201.0
255.255.255.0
nat (inside) 1 access-list 100
global (outside) 1 209.165.201.2
```

13.4.4 Konfiguration Szenario 2

Ein externer Benutzer besucht einen Webserver in der DMZ. Seine Global Outside-Adresse wird auf eine Global Inside (DMZ) Adresse übersetzt.

ASA1

```
access-group 200 in interface outside
access-list 200 extended permit tcp any host 10.1.1.3 eq www
nat (outside) 1 209.165.201.1
global (dmz) 1 10.1.1.13 255.255.255.0
```

13.4.5 Konfiguration Szenario 3

Blockiert Zugriffsversuch von aussen auf einen Host im internen Bereich

ASA1

Für dieses Szenario ist keine weitere Konfiguration nötig. Dies wird durch die Securitylevel automatisch umgesetzt. Wichtig ist hier, dass der Security-Level auf dem inside Interface grösser ist als auf dem outside interface (siehe 4.2 Initialkonfiguration).

13.4.6 Konfiguration Szenario 4

Blockiert Zugriffsversuch aus der DMZ auf einen internen Host

ASA1

Für dieses Szenario ist keine weitere Konfiguration nötig. Dies wird durch die Securitylevel automatisch umgesetzt. Wichtig ist hier, dass der Security-Level auf dem inside Interface grösser ist als auf dem dmz interface (siehe 4.2 Initialkonfiguration)

13.4.7 Konfiguration Szenario 5

Ein interner Benutzer besucht einen Webserver in der DMZ

ASA1

```
access-list 100 extended permit ip 10.1.2.0 255.255.255.0 any
nat (inside) 0 access-list 100
```

13.5 VirtualBox

VirtualBox ist eine Virtualisierungs-Software von Sun Microsystems (http://www.virtualbox.org). Sie diente für das Iptables-Testszenario zur Virtualisierung des Ubuntu 9.04 "Jaunty Jackalope" auf einer Windows XP-Umgebung. Insgesamt wurden vier virtualisierte Ubuntu's aufgesetzt, wovon eines schlussendlich als Firewall mittels Iptables konfiguriert wurde. Die anderen drei dienten als Test-Hosts.

13.5.1 Internes Networking

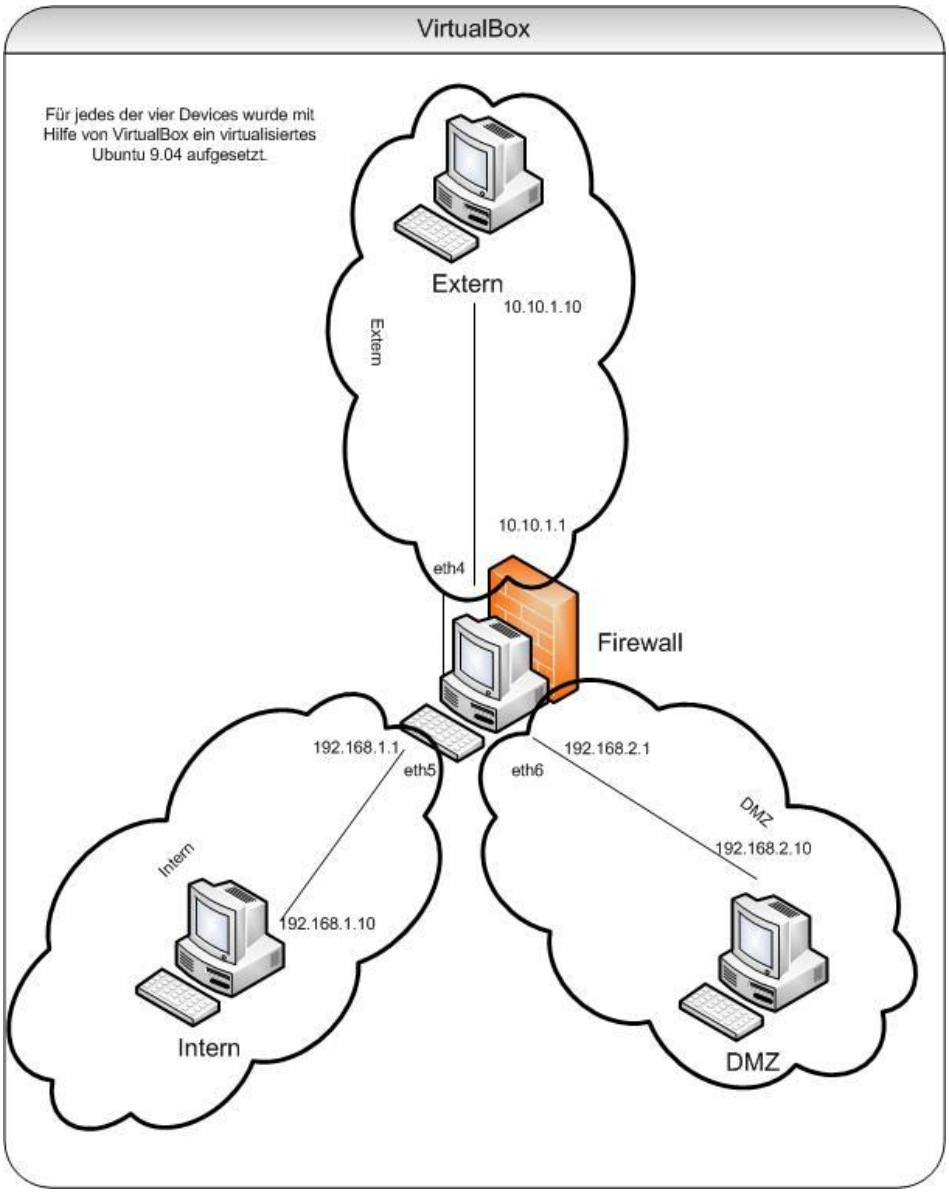
VirtualBox bietet die Möglichkeit, für jedes Gastbetriebssystem bis zu 8 Netzwerkkarten zu virtualisieren. Dabei werden Karten wie PCnet-PCI II Am79C970A, PCnet-FAST III Am79C973 oder Intel PRO/1000 MT Desktop 8254 OEM verwendet, deren Treiber in den meisten Betriebssystemen, welche als Gast-OS in Frage kommen, vorhanden sind.

Desweiteren bietet VirtualBox die Möglichkeit, für das Vernetzen von verschiedenen Gastsystemen sogenannte interne Netzwerke zu erstellen. Dabei funktioniert die VirtualBox-Laufzeitumgebung als Switch zwischen den einzelnen Gastsystemen.

Für die Realisation unserer Szenarios wurde jeweils für jede Zone (Intern, DMZ, Extern) ein eigenes virtuelles Netzwerk erstellt, an welchem der Host der Zone und die entsprechende Netzwerkkarte des Firewalls hängt. **Da in den internen Netzwerken von VirtualBox nur IP's aus privaten IP-Ranges zugelassen sind, wurde bei der Vergabe der IP-Adressen von der Vorgabe aus den Szenarios abgewichen!**

13.6 iptables

13.6.1 Übersicht



Übersicht der Realisation mittels iptables 1.4.1.1 unter Ubuntu 9.04

Da der Host "Firewall" als Router funktionieren muss, wurde im Kernel das Forwarding mittels folgendem Befehl aktiviert:

```
echo 1 >> /proc/sys/net/ipv4/ip_forward
```

Anschliessend wurden die folgenden Konfigurationen ebenfalls auf dem Host "Firewall" vorgenommen. Die gesicherte komplette Konfiguration befindet sich im Anhang.

13.6.2 Initialkonfiguration

```
-A INPUT -i lo -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
-t nat -A POSTROUTING -o eth4 -j MASQUERADE
```

Konfiguration des Loopback-Interfaces, welches jeglichen Traffic über sich erlauben soll sowie die NAT-Einstellung für das externe Interface (eth4) welches die Source-Adresse aus unseren internen Netzen mit der Adresse des externen Interfaces ersetzt.

13.6.3 Konfiguration Szenario 1

Ein interner Benutzer besucht einen externen Webserver.

```
-A FORWARD -i eth4 -o eth5 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i eth5 -o eth4 -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -j REJECT --reject-with icmp-port-unreachable
```

13.6.4 Konfiguration Szenario 2

Ein externer Benutzer besucht einen Webserver in der DMZ. Diese Szenario wurde extra stateless konfiguriert, um einen zusätzlichen Aspekt in die Rules zu bekommen.

```
-t nat -A PREROUTING -i eth4 -p tcp --dport 80 -j DNAT --to 192.168.2.10:80
-A FORWARD -i eth4 -o eth6 -p tcp -m tcp --dport 80 -j ACCEPT
-A FORWARD -i eth6 -p tcp -m tcp --sport 80 -j ACCEPT
-A FORWARD -j REJECT --reject-with icmp-port-unreachable
```

13.6.5 Konfiguration Szenario 3

Blockiert Zugriffsversuch von aussen auf einen Host im internen Bereich.

```
-A FORWARD -i eth4 -o eth5 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -j REJECT --reject-with icmp-port-unreachable
```

Blockiert Zugriffsversuch von aussen, sofern nicht von intern bereits eine Session eröffnet wurde (stateful firewalling)

13.6.6 Konfiguration Szenario 4

Blockiert Zugriffsversuch aus der DMZ auf einen internen Host.

```
-A FORWARD -i eth6 -o eth5 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -j REJECT --reject-with icmp-port-unreachable
```

Blockiert Zugriffsversuch aus der DMZ, sofern nicht von intern bereits eine Session eröffnet wurde (stateful firewalling)

13.6.7 Konfiguration Szenario 5

Ein interner Benutzer besucht einen Webserver in der DMZ.

```
-A FORWARD -i eth5 -o eth6 -m state --state NEW,RELATED,ESTABLISHED -j
ACCEPT
-A FORWARD -i eth6 -o eth5 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -j REJECT --reject-with icmp-port-unreachable
```

13.6.8 Gesamte iptables Konfiguration

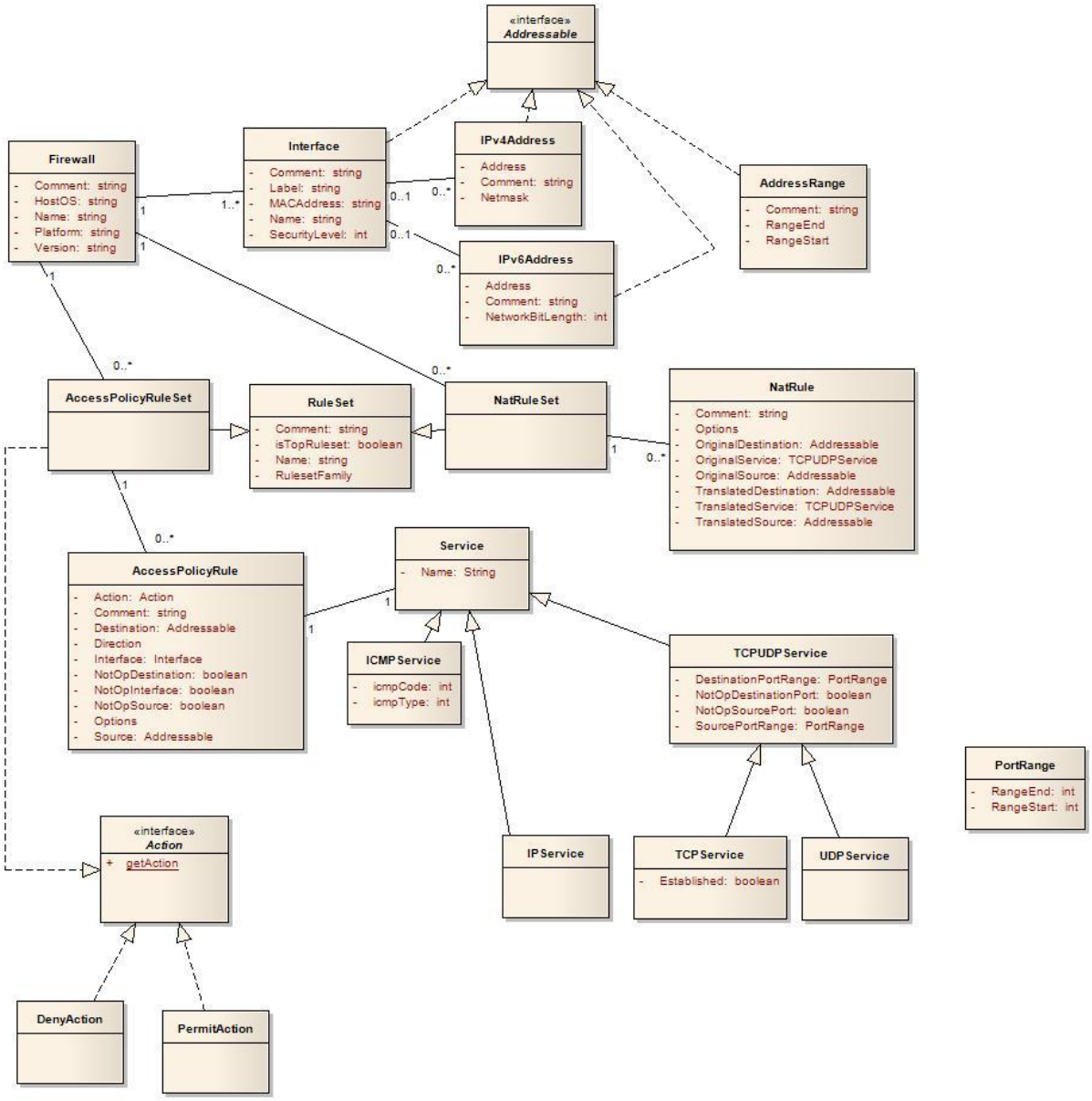
```
# Generated by iptables-save v1.4.1.1 on Thu Oct  8 16:01:48 2009
*nat
-t nat -A POSTROUTING -o eth4 -j MASQUERADE
-t nat -A PREROUTING -i eth4 -p tcp --dport 80 -j DNAT --to 192.168.2.10:80

*filter
:INPUT ACCEPT [72:13717]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [10:1156]
```

```
-A INPUT -i lo -j ACCEPT
-A FORWARD -i eth4 -o eth6 -p tcp -m tcp --dport 80 -j ACCEPT
-A FORWARD -i eth6 -p tcp -m tcp --sport 80 -j ACCEPT
-A FORWARD -i eth6 -o eth4 -m state --state NEW,RELATED,ESTABLISHED -j
ACCEPT
-A FORWARD -i eth4 -o eth6 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i eth4 -o eth5 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i eth5 -o eth4 -m state --state NEW,RELATED,ESTABLISHED -j
ACCEPT
-A FORWARD -i eth5 -o eth6 -m state --state NEW,RELATED,ESTABLISHED -j
ACCEPT
-A FORWARD -i eth6 -o eth5 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -j REJECT --reject-with icmp-port-unreachable
-A OUTPUT -o lo -j ACCEPT
COMMIT
# Completed on Thu Oct  8 16:01:48 2009
```

14 Datenmodell

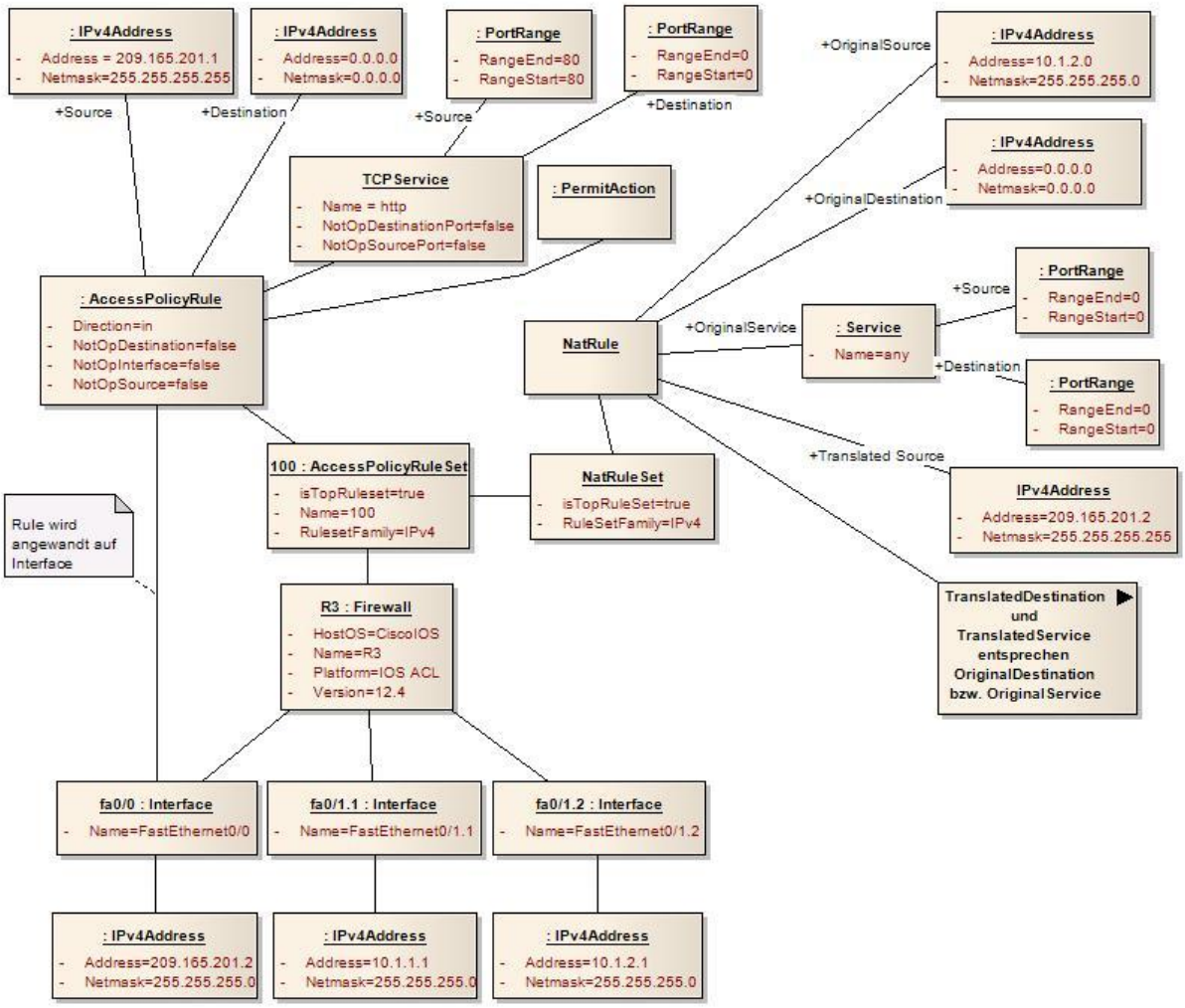
14.1 Erster konzeptueller Entwurf

Die folgende Abbildung zeigt einen ersten konzeptuellen Entwurf der Problem Domain, der allerdings während des Projekts teilweise noch angepasst wurde, um auftauchende Probleme zu beheben.



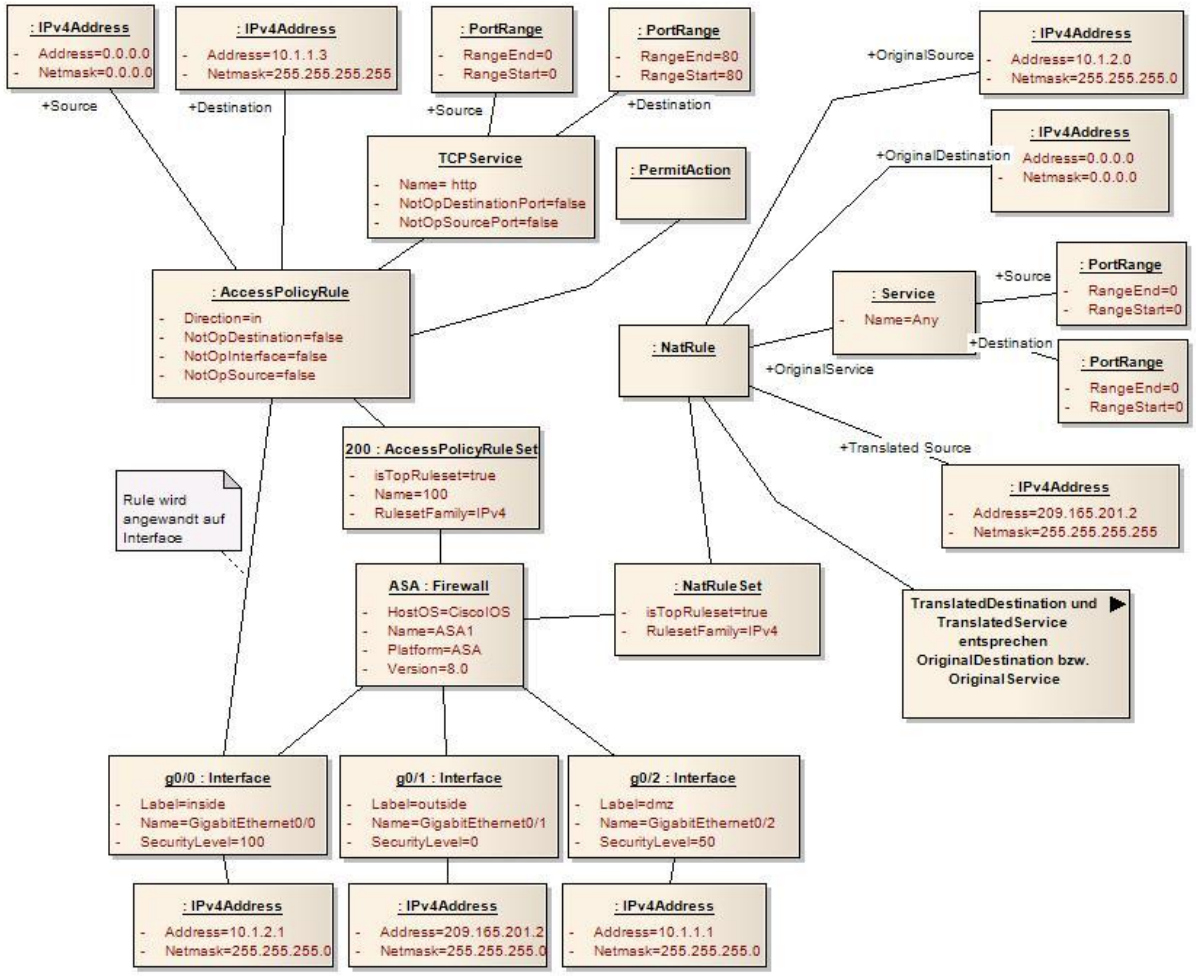
14.2 Verifizierung (Objektdiagramme)

14.2.1 Cisco IOS



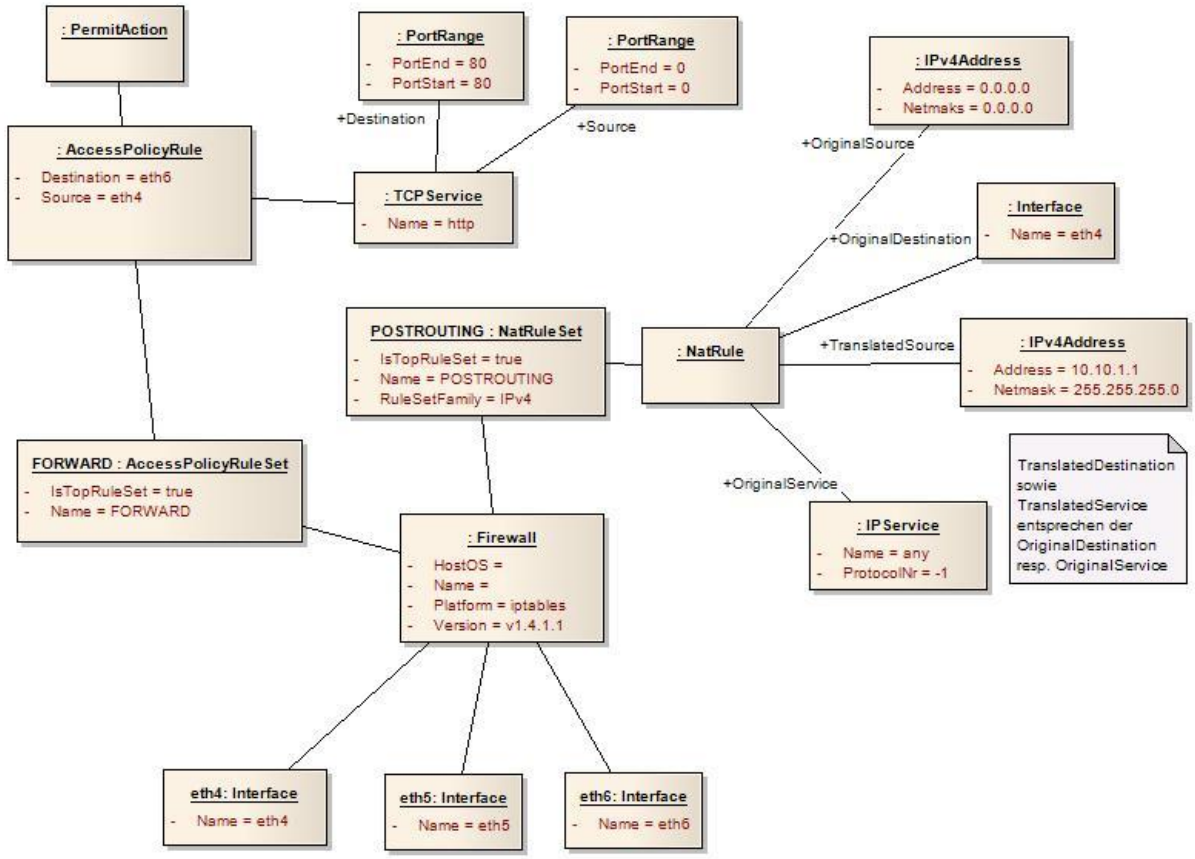
Die Abbildung beschreibt einen Teil der Konfiguration des Szenario Nr. 1, konfiguriert auf einem Cisco IOS Gerät. Die Regel (AccessPolicyRule) bewirkt das Zulassen des Verkehrs vom Webserver über Port 80 in das interne Netz. Zusätzlich müsste hier noch eine Regel definiert werden, welche den restlichen, einkommenden Verkehr blockiert.

14.2.2 Cisco ASA



Diese Konfiguration wurde von einem Cisco ASA Gerät eingelesen. Es zeigt eine AccessPolicyRule aus dem Szenario 2. Die Policy bewirkt das Zulassen des Verkehrs von einem Host im externen Netz auf den Webserver in der DMZ über Port 80. Weiter sieht man rechts in der Abbildung eine NatRule aus dem Szenario 1. Die NatRule übersetzt alle IP-Adressen im internen Netz 10.1.2.0 255.255.255.0 (Inside Local) in eine der Adressen im Range 209.165.201.0 255.255.255.0 im externen Netz (Inside Global).

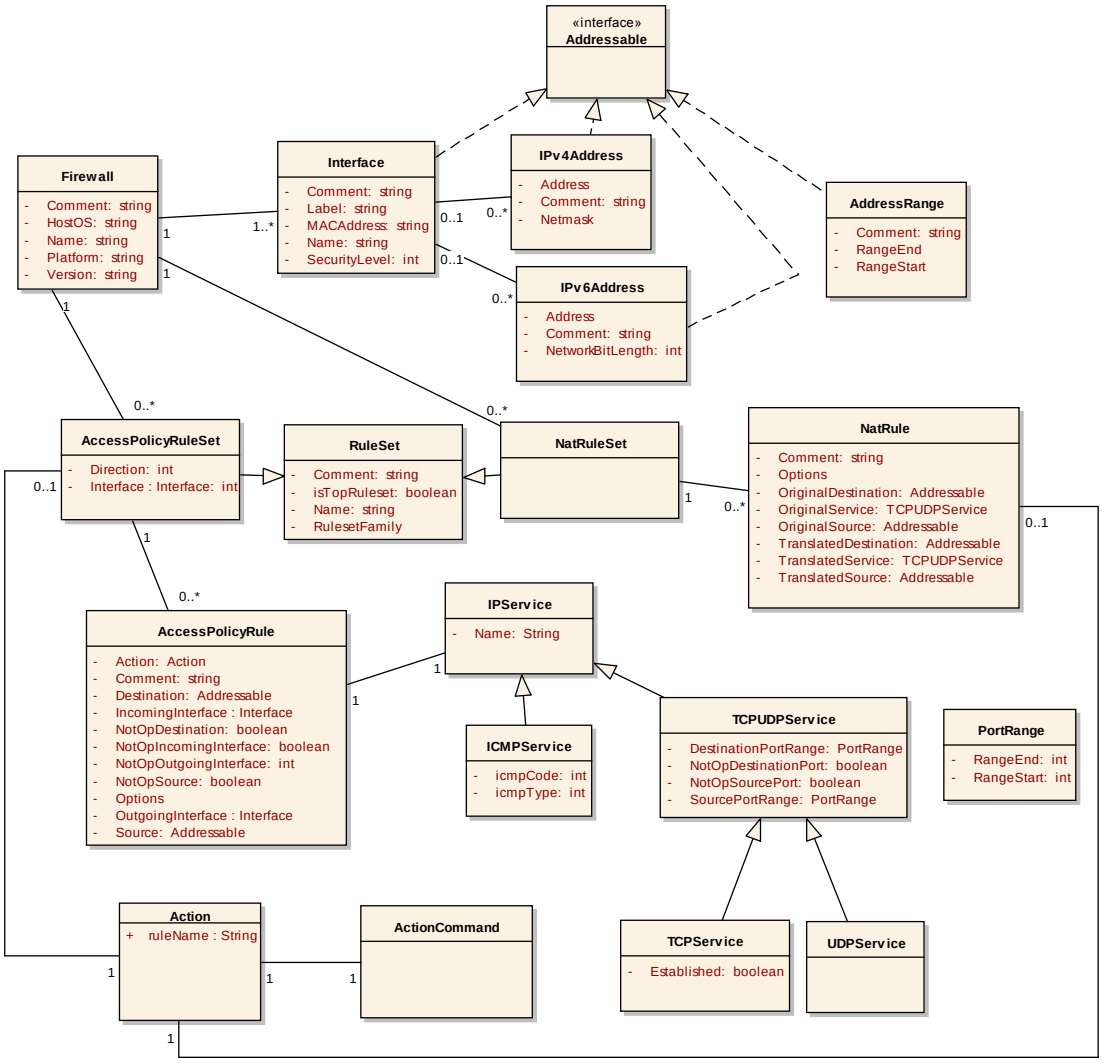
14.2.3 iptables



Dieses Diagramm zeigt eine eingelesene Konfiguration eines iptables-Firewalls. Es zeigt einen Teil des Szenarios 2, in welchem es darum geht, einem externen Host (Besucher) den Zugriff über Port 80 auf einen Webserver innerhalb der DMZ zu erlauben. Um das Szenario 2 zu komplettieren fehlt selbstverständlich einerseits die Regel, den nach dem Zugriff erzeugten "Gegenverkehr" vom Webserver zum Host zu erlauben und andererseits die Regel, den restlichen Verkehr zu unterbinden (iptables ist bekannterweise per default so konfiguriert, dass am Schluss implizit "allow all" angewendet wird).

14.3 Finaler konzeptueller Entwurf

Die folgende Abbildung zeigt den finalen konzeptuellen Entwurf der Problem Domain. Im Vergleich zum ersten Entwurf wurden einige Details angepasst, die anschliessend kurz beschrieben werden.



- Die Zuordnung eines Interfaces sowie einer Direction geschieht nicht mehr bei einer einzelnen Rule, sondern bei einem gesamten AccessPolicyRuleSet. Die zwei Attribute wurden also von der Klasse AccessPolicyRule in die Klasse AccessPolicyRuleSet verschoben. Diese Attribute werden zur Abspeicherung von cisco-spezifischen Konfigurationen verwendet.
- Gleichzeitig musste es aber bei iptables möglich sein, Incoming- sowie Outgoing-Interfaces zu definieren. Diese beiden Attribute und die dazugehörigen NotOperatoren, welche das Interface negieren, wurden also bei der Klasse AccessPolicyRule wieder eingeführt.
- Da das Firewall-Analyse-Tool nur mit IP-Services umgehen kann, wurde die Klasse Service entfernt und die Klasse IPService als oberste Klasse für alle anderen Services definiert.
- Die verschiedenen Action Klassen wurden entfernt und dafür die Klasse ActionCommand eingeführt, welche die verschiedenen möglichen Actions als Enum enthält.
- Die Klasse Action erhielt ein neues Feld ruleName, das eine eventuell vorhandene weitere Rule als String enthalten kann. So können Rules verschachtelt werden.
- NatRules können nun auch verschachtelt werden.

15 Sitzungsprotokolle

15.1 18. September 2009

Datum:	18. September 2009
Zeit:	11:15 Uhr bis 11:45 Uhr
Ort:	Hochschule Rapperswil, Raum 6.001
Anwesende:	Prof. Beat Stettler
	Roman Ammann
	Christoph Rebsamen
	Gieri Kohler
Protokoll:	Gieri Kohler
Nächste Sitzung:	25. September 2009, 11:15 Uhr

Als erstes wurden kurz einige administrative Dinge (Arbeitsaufwand, Dokumentation und Präsentation) besprochen und die Telefonnummern getauscht.

Danach ging es in der heutigen Sitzung unter Anderem um die grobe Projektzeitplanung. Das Projekt wird in folgende zwei Phasen aufgeteilt:

1 Auslesen von Firewallkonfigurationen
   a. Die verschiedenen Konfigurationen kennenlernen, indem Szenarien ausgedacht und diese dann innerhalb einer Übungsumgebung umgesetzt werden.
   b. Firewallkonfigurationen verschiedener Hersteller in ein herstellerunabhängiges Regelset abbilden
2 Topologisches Setup der verschiedenen Firewalls darstellen

Es wurde entschieden, dass wir uns vor allem mit der Syntax von iptables, ASA 8.0 und Cisco IOS auseinandersetzen und weitere Technologien vorerst noch nicht in Betracht ziehen sollen. Diese Formate sollen dann in Phase 1b eben auch in das unabhängige Regelset abgebildet werden können.

Weiteres Vorgehen
Bis zum nächsten Treffen sollen folgende Arbeiten erledigt werden:

- Einlesen in die Thematik
- Anforderungsspezifikation schreiben
- Projektplan (Zeitplanung inkl. Meilensteine) erstellen
- Szenarios ausdenken

15.2 25. September 2009

Datum:	25. September 2009
Zeit:	11:15 Uhr bis 11:45 Uhr
Ort:	Hochschule Rapperswil, Raum 6.001
Anwesende:	Prof. Beat Stettler
	Roman Ammann
	Christoph Rebsamen
	Gieri Kohler
Protokoll:	Christoph Rebsamen
Nächste Sitzung:	2. Oktober 2009, 11:15 Uhr

Als erstes fand ein kurzer Rückblick auf die vergangene Sitzung statt. Anschliessend wurden die erbrachten Arbeitsresultate (Anforderungsspezifikation/Projektplan) angeschaut und besprochen.
Ergebnis:

- Anforderungsspezifikation muss noch klarer formuliert werden
- Im Projektplan müssen noch die Workpackages definiert werden
- Dokumente, welche in der Sitzung besprochen werden sollen, sind bis Donnerstagabend vor der Sitzung per E-Mail einzureichen

Daraufhin konzentrierte sich die Diskussion auf die anstehende Lab-Session auf den Rental Rack. Der Zugang zu diesem wird uns von Roman Ammann ermöglicht.
Im Anschluss an die Sitzung erhalten wir von Roman gesammelte, aber nicht gefilterte Literatur zum Thema.

Weiteres Vorgehen
Bis zum nächsten Treffen sollen folgende Arbeiten erledigt werden:

- Änderung/Anpassung der Anforderungsspezifikation/Projektplan
- Durchspielen der Szenarios im Lab

Kommende Abwesenheiten:
10.10. - 17.10.2009 Christoph Rebsamen (J&S-Lager)
15.10. - 15.11.2009 Christoph Rebsamen (2-3-tägige Alarmübung Militär; Zeitpunkt unklar)

15.3 2. Oktober 2009

Datum:	2. Oktober 2009
Zeit:	11:15 Uhr bis 11:50 Uhr
Ort:	Hochschule Rapperswil, Raum 6.001
Anwesende:	Prof. Beat Stettler
	Roman Ammann
	Christoph Rebsamen
	Gieri Kohler
Protokoll:	Gieri Kohler
Nächste Sitzung:	9. Oktober 2009, 11:15 Uhr

Zu Beginn der Sitzung klärten wir, was in der Vorwoche erreicht wurde. Das Durchspielen der Szenarien in der Übungsumgebung stellte sich als schwieriger heraus als zuerst angenommen. Auf die Clients konnte nicht zugegriffen werden und das Testen der Konfigurationen war somit nicht richtig möglich. Roman Ammann gab uns den Tipp, die Router als Ersatz für die Clients und Server einzusetzen. Ausserdem wies er uns auf die 2 Programme dynamips und pemu hin, welche zum Testen eventuell nützlich wären.

Zu den Anforderungen an die Software wurden ein paar genauere Spezifikationen gemacht. Es sollen folgende Informationen aus einer ASA 8.0 oder Cisco Firewall in das unabhängige Regelset abgebildet werden können:

- Standard/Extended/Named/Numbered ACLs
- Reflexive ACLs
- Time based ACLs
- Cisco ASA 8.0 security levels
- ASA 8.0 Connections/NAT

Folgende Technologien und Möglichkeiten müssen nicht abgebildet werden können:

- Lock and Key ACLs
- Content Filtering
- Wenn nötig sollen aber Warnungen ausgegeben werden, dass der Verkehr aufgrund dieser Technologien blockiert werden könnte, um den Benutzer der Applikation zu informieren.

Ideen für die Abbildung in ein einheitliches Regelset wurden besprochen. Es war noch nicht klar, ob der Ansatz, alle Formate in ACLs abzubilden, funktionieren würde. Vor allem bei der Abbildung von

iptables in ACLs waren einige Zweifel vorhanden. Roman Ammann hat vorgeschlagen, alle Formate in iptables abzubilden. Was hier besser funktioniert, muss noch abgeklärt werden.

Beim Review der Dokumente meinte Prof B. Stettler, dass noch einige Szenarios fehlen.

Weiteres Vorgehen
Bis zum nächsten Treffen sollen folgende Arbeiten erledigt werden:

- Überarbeiten der Dokumente Anforderungsspezifikation, Projektplan und Szenarios
- Durchführen weiterer Tests in der Übungsumgebung
- Erstellen aller config-Files der fünf Szenarien in den Varianten ASA, IOS und iptables
- Ev. Domainanalyse

Kommende Abwesenheiten
09.10.2009 Prof. B. Stettler
09.10.2009 Gieri Kohler

15.4 9. Oktober 2009

Datum:	9. Oktober 2009
Zeit:	11:15 Uhr bis 11:35 Uhr
Ort:	Hochschule Rapperswil, Raum 6.001
Anwesende:	Roman Ammann
	Christoph Rebsamen
Protokoll:	Christoph Rebsamen
Nächste Sitzung:	16. Oktober 2009, 11:15 Uhr

Rückblick
Während der vergangenen Woche wurden die Szenarios mit allen drei Firewall-Technologien (ASA, IOS, iptables) in einer Testumgebung umgesetzt. Die cisco-spezifischen Technologien wurden auf dem CCIE-Rental-Rack umgesetzt und die iptables-Technologie in einem virtualisierten Netzwerk mittels Virtualbox.
Die Dokumente wurden gemäss letzter Sitzung angepasst und teilweise ergänz. Einzig bei den Szenarios war unklar, was Herr Stettler mit der Unvollständigkeit gemeint hat. Dies wird in der nächsten Sitzung geklärt und anschliessend ergänzt.
Zudem wurde mit jflex (http://jflex.de/) erste Versuche zum Parsen gemacht.

Zu erledigen
Neues Dokument erstellen zu den umgesetzten Szenarios. Ersichtlich sollten Test-Aufbau, Konfigurationen, etc. sein. Dazu soll noch für jede Technologie eine Kommunikationsmatrix erstellt werden.
Desweiteren ist die Entscheidung zu treffen, in welches Format geparst wird. Der Entscheid ist mit Argumenten zu stützen. Als Unterstützung zur Entscheidungsfindung kann hier eventuell das Tool FirewallBuilder (http://www.fwbuilder.org/) dienen.

Kommende Abwesenheiten
10. -17.10.09 Christoph Rebsamen

15.5 16. Oktober 2009

Datum:	16. Oktober 2009
Zeit:	11:15 Uhr bis 11:30 Uhr
Ort:	Hochschule Rapperswil, Raum 6.001
Anwesende:	Roman Ammann

	Gieri Kohler
Protokoll:	Gieri Kohler
Nächste Sitzung:	23. Oktober 2009, 11:15 Uhr

Als erstes diskutierten wir die in der vergangenen Woche erstellten Dokumente. Da die Kommunikation zwischen den Teammitgliedern seit der letzten Sitzung nicht mehr möglich war, kam es zu ein paar kleinen Missverständnissen bezüglich des Inhalts der Dokumente. Insbesondere, was unter einer Kommunikationsmatrix zu verstehen ist, war nicht klar. Einige Dokumente waren ausserdem noch unvollständig. Dies soll bis zur nächsten Sitzung behoben werden.

Weiteres Vorgehen
Bis zum nächsten Treffen sollen folgende Arbeiten erledigt werden:

- Abklärung, ob das Datenmodell des Tools FirewallBuilder für unsere Zwecke geeignet ist (ev. Anpassen des entworfenen Datenmodells)
- Überarbeitung des Dokuments „Realisation Szenarios“
  - Ergänzung der fehlenden Kapitel über iptables
  - Überflüssige Konfigurationen in den jeweiligen Running Configs entfernen. Die Grundkonfiguration kann angenommen werden. Das Dokument muss nur noch die zusätzlich von Hand eingestellten Konfigurationen enthalten.
  - Konfigurationen aufteilen in die jeweiligen Szenarios. Das Dokument soll eine Initialkonfiguration und 15 Codeschnipsel enthalten (ein Codeschnipsel pro Senzario und Technologie)
  - Die Konfigurationen müssen abgeändert werden. Alle Firewallkonfigurationen sollen auf einem Gerät eingestellt sein. Outside, DMZ und Inside sollen an 3 Interfaces des selben Geräts angeschlossen sein.
- Überarbeitung des Dokuments „Szenarios“
  - Hinzufügen einer Kommunikationsmatrix für jedes Szenario.
- Überarbeitung des Dokuments „Datenmodell“
  - UML-Diagramm hinzufügen

15.6 27. Oktober 2009

Datum:	27. Oktober 2009
Zeit:	09:30 bis 10:00 Uhr
Ort:	Hochschule Rapperswil, Raum 6.001
Anwesende:	Beat Stettler
	Roman Ammann
	Gieri Kohler
	Christoph Rebsamen
Protokoll:	Christoph Rebsamen
Nächste Sitzung:	30. Oktober 2009, 11:15 Uhr

Zu Beginn wurde kurz geklärt, was zwischen vorletzter und letzter Sitzung schief gelaufen war (Abwesenheit Gieri Kohler, welche sich mit Abwesenheit von Christoph Rebsamen überschnitten hat -> ungenügende Kommunikation).
Anschliessend wurden die Dokumente besprochen. Folgende Punkte sind zu korrigieren/ergänzen:

- In den Dokumenten keine cisco-spezifischen Terms verwenden oder erklären ("inside local" u. ä.)
- Da sich die Kommunikationsmatrix der einzelnen Szenarien nicht unterscheidet, nur eine gemeinsame Matrix im Dokument belassen
- Szenario-Dokument: Fehler ausmerzen
- Permit All für die Kommunikation von intern zur DMZ ist zu grosszügig, nur einzelne Services zulassen
- iptables-Diagramm: Interfaces ergänzen

- Szenario 1 bei iptables: Speziell erwähnen, dass absichtlich stateless konfiguriert wurde um zusätzlichen Aspekt einzubringen
- iptables-config und Cisco IOS-config um NAT ergänzen
- Domainmodel: Interface kann mehrere IP's haben und nicht wie definiert nur eine

Die Frage ob die Teammitglieder selber festlegen können, was als Input für den Parser geliefert werden muss, wurde bejaht. Das heisst, dass neu zusätzlich zur iptables-config auch noch die Konfiguration der Interfaces des zugrundeliegenden Linux erforderlich ist. Dies deshalb, da ansonsten möglicherweise die IP's der Interfaces nicht bekannt sind, da sie nicht in der iptables-config vorhanden sein müssen.
Desweiteren liefert uns Roman zusätzliche grosse Konfigurations-Files von Firewalls. Mithilfe dieser soll die Problem Domain nochmals verifiziert werden.

Folgende Dinge sind bis zur nächsten Sitzung zu erledigen:

- Idee zum Persistence-Teil der Applikation entwickeln. Ev. kann im selben Format wie der Firewall-Builder gespeichert werden
- Problem Domain implementiert
- Dokumente gemäss obiger Vorgaben ergänzt

15.7 30. Oktober 2009

Datum:	30. Oktober 2009
Zeit:	11:15 bis 11:30 Uhr
Ort:	Hochschule Rapperswil, Raum 6.001
Anwesende:	Roman Ammann
	Gieri Kohler
	Christoph Rebsamen
Protokoll:	Gieri Kohler
Nächste Sitzung:	6. November 2009, 11:15 Uhr

Am Anfang der Sitzung haben wir die überarbeiteten Dokumente besprochen. Beim Datenmodell müssen noch die NAT-Regeln bei den Cisco-IOS- und iptables-Objektdiagrammen ergänzt werden.

Danach haben wir verschiedene Möglichkeiten für die Umsetzung der Persistence betrachtet. Die Variante, alles im Format des Firewallbuilders abzuspeichern (XML), wäre wünschenswert und „nice to have“, ist aber eher kompliziert umzusetzen. Wir haben uns darauf geeinigt, dieses Problem anzugehen, falls am Ende noch Zeit dafür ist und vorerst noch mit einer SQLite Datenbank oder einem einfachen XML-Format zu arbeiten. Die Entscheidung für eine dieser beiden Varianten ist uns überlassen. Am Ende soll die Applikation aber eine einfache Installation erlauben.

Weiteres Vorgehen
Bis zur nächsten Sitzung sollen wir folgendermassen weiter vorgehen:

- Christoph Rebsamen: Umsetzung der Persistence
- Gieri Kohler: Weiterentwicklung des Parsers für Cisco IOS
- Entwicklung eines Prototyps für Cisco IOS Parsing, welcher an der nächsten Sitzung kurz demonstriert wird

15.8 6. November 2009

Datum:	06. November 2009
Zeit:	11:15 bis 11:40 Uhr
Ort:	Hochschule Rapperswil, Raum 6.001
Anwesende:	Roman Ammann

	Gieri Kohler
	Christoph Rebsamen
Protokoll:	Christoph Rebsamen
Nächste Sitzung:	13. November 2009, 11:15 Uhr

Zu Beginn erklärte Christoph Rebsamen, warum die Persistenz für das Firewall-Tool noch nicht soweit gediehen ist, wie geplant. Probleme mit dem Mapping der Problem-Domain-Klassen benötigen noch eine vertiefte Analyse. Deshalb funktioniert im Prototyp auch erst das Parsen des Config-Files während das Ablegen der Problem-Domain noch weggelassen werden musste.
Die Demo des Parser hingegen war erfreulich. Was im Parser noch fehlt, ist eine sinnvolle Fehlerbehandlung. Momentan werden alle unbekannten Eingaben einfach ignoriert. Dies sollte noch verbessert werden.

Zu erledigen:

- Es sollte eine Liste erstellt werden, aus der ersichtlich ist, welche Inputs bereits geparst werden und welche noch nicht implementiert sind
- Bis zur nächstens Sitzung sollte der vertikale Prototyp, inklusive einer Komponente für die Abfrage an die Firewall (Wird dieser Verkehr erlaubt oder nicht?), erstellt werden.

15.9 17. November 2009

Datum:	17. November 2009
Zeit:	09:30 bis 09:50 Uhr
Ort:	Hochschule Rapperswil, Raum 1.258
Anwesende:	Roman Ammann
	Gieri Kohler
	Christoph Rebsamen
Protokoll:	Gieri Kohler
Nächste Sitzung:	20. November 2009, 11:15 Uhr

Christoph Rebsamen erklärte am Anfang einige noch bestehende Probleme mit der Persistence mittels JPA auf einer SQLite-Datenbank. JPA versucht momentan noch beim Auslesen eines gesamten Firewall-Objekts aus der Datenbank, einen Join über mehr als 64 Tables zu machen, was aber eine Limitierung bei SQLite darstellt. Er wird dieses Problem bis zur nächsten Sitzung noch unter die Lupe nehmen und eine Entscheidung für oder gegen SQLite fällen.

Beim Parser wurden in der vorigen Woche verschiedene Erweiterungen vorgenommen. Einerseits werden jetzt ACLs mit ICMP unterstützt (wird gespeichert mittels eines Codes und eines Typs) und andererseits auch ACLs mit anderen IP-Protokollen (ahp, eigrp, esp, igmp, ospf, usw.) Abgespeichert werden diese als IP-Service mit einer IP-Protokoll-Nummer.

Beim CLI sind bis zum nächsten Mal noch einige Änderungen zu machen. Es soll jeweils ein eigenes CLI für das Parsen und für das Abfragen der Datenbank erstellt werden. Momentan sind diese beiden Aufgaben noch mit einem einzigen CLI zu erledigen. Weiter müssen noch einige Vergleichsoperatoren implementiert werden. Zum Beispiel können noch keine Interfaces als Source und Destination und noch keine IPv6-Adressen verglichen werden. Roman Ammann merkte ausserdem noch an, dass es besser wäre, wenn bei der Auswahl der incoming- und outgoing-Interfaces neben dem Interface-Namen die IP-Adresse des Interfaces ebenfalls sichtbar wäre.

15.10 20. November 2009

Datum:	20. November 2009
Zeit:	11:15 bis 11:50 Uhr
Ort:	Hochschule Rapperswil, Raum 6.001
Anwesende:	Roman Ammann
	Gieri Kohler
	Christoph Rebsamen
Protokoll:	Christoph Rebsamen
Nächste Sitzung:	27. November 2009, 11:15 Uhr

Rückblick:
Christoph Rebsamen legte zuerst dar, dass die Probleme betreffend der Persistenz mit sqlite behoben werden konnten. Das einzige zur Zeit noch bestehende Problem mit sqlite stellt die Speicherung von BigInteger-Werten (verwendet für die Speicherung von IPv6-Adressen) dar. Dieses kann aber nach Einschätzung von CR gelöst werden.
Am Analyzer wurden in der letzten Woche diverse Verbesserungen vorgenommen, unter anderem wird jetzt dem Benutzer eine abschliessende Entscheidung (Deny/Permit) mitgeteilt.
Die Demo des Prototypen funktionierte soweit gut, einzig ein Fehler im Parser mit Sub-Interfaces von IOS wurde festgestellt und sollte so schnell als möglich behoben werden.

Ausblick:
Folgende zusätzlichen oder ergänzenden Funktionen sind noch gewünscht:

- Der Parser soll mit Aufruf-Parametern funktionieren. Dabei soll der Dateiname für das zu parsende Konfigurationsfile sowie der Name der zu parsenden Firewall angegeben werden. (Der Name wird dazu verwendet, eine Entscheidung zu treffen, ob ein bestehender Firewall aktualisiert wird oder ob ein neues Firewall-Objekt angelegt wird.
- Für den Komfort soll die Zeilennummer der Rule-Definition aus dem Config-File im Objekt abgelegt werden und gegebenenfalls dem Benutzer beim analysieren angegeben werden.
- Der Parser soll um ein Statistik-Summary, welches zum Schluss ausgegeben wird, ergänzt werden. Dies dient der Kontrolle, ob alles relevante aus dem Konfigurations-File gelesen wurde.
- Die Fehlerbehandlung im Parser sollte ausgebaut werden. Beispielsweise, wenn ein Interface doppelt erstellt wird.
- Das Datum, wann ein Konfigurations-File geparst wurde, sollte als Ergänzung zum Firewall-Objekt hinzugefügt werden.
- Im Anschluss an eine Abfrage sollte dem Benutzer ein Summary präsentiert werden mit folgendem Inhalt:
  - Die gemachte Anfrage
  - Die Regeln die matchen
  - Die konkrete Linie aus dem Config-File, die gematcht hat

15.11 27. November 2009

Datum:	27. November 2009
Zeit:	14:00 bis 14:30 Uhr
Ort:	Hochschule Rapperswil, Raum 6.001
Anwesende:	Roman Ammann
	Gieri Kohler
	Christoph Rebsamen
Protokoll:	Gieri Kohler
Nächste Sitzung:	4. Dezember 2009, 11:15 Uhr

In der vorigen Woche wurden erneut einige Änderungen am CLI vorgenommen. Es ist jetzt möglich, den Namen der Inputdatei als Parameter anzugeben. Ebenfalls per Parameter kann angegeben werden, ob eine eventuell schon bestehende Firewall mit gleichem Namen überschrieben werden soll. Beim CLI wird nach dem Parsen zur Kontrolle eine kleine Statistik mit Angaben über Anzahl geparster Rulesets, Rules und Interfaces ausgegeben. Beim Analyzer CLI werden die eingebenen Parameter zur einfacheren Kontrolle nochmals ausgegeben. Nach dem Auswerten der Parameter werden die matchenden Rules im allgemeinen sowie im „native“ Format angezeigt. Die Zeilennummer der gematchten Rule anzuzeigen, wäre noch wünschenswert, konnte aber wegen Zeitmangel noch nicht umgesetzt werden.
Das Problem, das Subinterfaces in einer IOS Inputdatei nicht erkannt werden, wurde behoben.

Weiter traten in der Woche noch einige Probleme auf, wenn IPv4 und IPv6 kombiniert wurde. Hier hatten wir uns bisher zu wenige Gedanken gemacht. Das Standardverhalten hier muss noch richtig implementiert werden, so dass dann auch das richtige Resultat bei der Auswertung angezeigt wird.

Ausserdem haben wir uns mit der Implementierung des Iptables-Parsers beschäftigt. Hier ist es bisher möglich, die Ausgabe des ip commands unter Linux auszuwerten und die Interfaces mitsamt IPv4 und IPv6 Adressen einzulesen. Mehrere Adressen pro Interface sind möglich. Weiter können Teile eines iptables-save commands bereits ausgewertet und ins allgemeine Format umgewandelt werden, andere Teile fehlen noch. Konkret werden bereits IPv4 Regeln in der Filter-Tabelle erkannt. Das Parsen der Nat-Tabelle sowie IPv6 Regeln soll nächste Woche in Angriff genommen werden.

Roman Ammann hat uns noch nahe gelegt, uns Gedanken zu automatischen Tests der Software zu machen.

Die Dokumentation wurde kurz angesprochen. Es wurde geklärt, ob es spezielle Vorstellungen gibt, was Inhalt der technischen Dokumentation sein sollte. Wir sollen hier vor allem die Umsetzung beschreiben, das Wie und Wieso. Auch Probleme, auf die wir gestossen sind, können ein Thema sein. Die technische Dokumentation soll für einen Techniker auf dem Gebiet verständlich sein. Ansonsten können wir uns an den Vorschlägen und dem Muster auf der HSR-Website orientieren.

15.12 4. Dezember 2009

Datum:	04. Dezember 2009
Zeit:	11:15 bis 11:55 Uhr
Ort:	Hochschule Rapperswil, Raum 6.001
Anwesende:	Roman Ammann
	Gieri Kohler
	Christoph Rebsamen
Protokoll:	Christoph Rebsamen
Nächste Sitzung:	11. Dezember 2009, 11:15 Uhr

Rückblick:
Die Arbeit von letzter Woche beinhaltete insbesondere die Implementation der iptables-Behandlung im Parser wie auch im Analyzer. Dabei tauchten einige Probleme, welche sich teilweise bis auf die Problem Domain auswirken, auf. Insbesondere wurde nicht berücksichtigt, dass in iptables NAT-Regeln gleich wie Filter-Regeln "verschachtelt" (Als Target wird eine weitere Chain eingegeben) werden können. Diese Möglichkeit lässt unser Design der Problem Domain gar nicht zu.
Durch die knappen zeitlichen Verhältnisse bis zur Abgabe baten wir, diese Änderungen an der Problem Domain nicht mehr durchführen zu müssen, um uns auf das Bugfixing und die Dokumentation zu konzentrieren. Diese Bitte wurde abgelehnt und die Änderung wird in der nächsten Woche implementiert.
Zudem wird auch noch die Möglichkeit behandelt, dass in einer NAT-Chain (PREROUTING, etc.) auch Filter-Regeln stehen können.

Ausblick:
In der noch folgenden zur Verfügung stehenden Zeit, wird insbesondere der Dokumentation einen grossen Teil der Zeit gewidmet. Alle Funktionen, sowie nicht erreichte Funktionalitäten müssen noch im Detail beschrieben werden.
Zudem ist für den Zeitraum zwischen dem 21. und 23.12.2009 eine kurze Präsentation (15 min) vorzubereiten, in welcher die geleistete Arbeit dem Betreuer und allenfalls anderen Interessierten präsentiert wird. Die Präsentation sollte in etwa folgende Punkte beinhalten:

- Aufgabe
- Was wurde gemacht
- Probleme / Nicht erreicht
- Wie weiter
- Demo

Anschliessend an die Präsentation werden während ca. 15 min Fragen beantwortet.

15.13 11. Dezember 2009

Datum:	11. Dezember 2009
Zeit:	11:15 bis 11:30 Uhr
Ort:	Hochschule Rapperswil, Raum 6.001
Anwesende:	Roman Ammann
	Gieri Kohler
	Christoph Rebsamen
Protokoll:	Gieri Kohler

Als erstes stimmten alle Sitzungsteilnehmer dem Termin für die Abschlusspräsentation am Dienstag, dem 22. Dezember, zu.

Als nächstes besprachen wir kurz die letzten Änderungen an der Problem Domain. Die Verschachtelung von NAT-Regeln bei iptables kann jetzt richtig auf die Problem Domain abgebildet werden. Ansonsten wurde in dieser Woche vor allem an der Dokumentation gearbeitet.

Weiter wurden an der Sitzung noch einige Details besprochen, welche die Dokumentation betreffen, wie beispielsweise die Anzahl abzugebender Exemplare.

IV. Anhang

16 Persönliche Berichte

16.1 Christoph Rebsamen

Nachdem Gieri Kohler und ich zu unserer grossen Freude den Zuschlag für die Semesterarbeit "Analyse-Tool für komplexe Firewallumgebungen", unserer 1. Priorität in der Vergabe der Semesterarbeiten, gingen wir voller Elan, aber ohne spezifische Kenntnisse im Bereich von Firewalls, an die Arbeit. Nach einigen Schwierigkeiten bezüglich der genauen Aufgabenstellung stand bald einmal die Anforderungsspezifikation an die zukünftige Software.

Nun ging es darum, sich in die Thematik einzuarbeiten. Diese Arbeit war sehr zeitaufwendig und trotz vieler investierter Stunden erreichte ich zu meiner Frustration nie den Punkt, an dem ich sagen konnte, ich habe alle Technologien bis in ihren letzten Winkel und mit allen Funktionen erfassen können. Trotzdem war es eine sehr lehrreiche und spannende Zeit.

Da aber die Zeit bereits stark fortgeschritten war, waren wir gezwungen, trotzdem mit der Implementation der Applikation zu beginnen. Da Gieri sich im Bereich von Cisco IOS durch die Absolvierung des CCNA-Kurses relativ gut auskannte, übernahm er die Entwicklung des Parsers für IOS-Konfigurationen. Ich widmete mich in dieser Zeit der Entwicklung der Persistenzschicht mittels JPA und Hibernate als OR-Mapper. Hierbei konnte ich die Kenntnisse, welche ich im parallel zur SA stattfindenden Modul "EnCom" erworben habe, gleich praktisch einsetzen. Auftretende Problem, insbesondere im Zusammenspiel von JPA/Hibernate mit der leider nicht ganz ausgewachsenen relationalen Datenbank liessen sich nach intensiven Online-Recherchen zum Schluss doch alle beseitigen. Diese Anwendungen von früher erworbenem Wissen machen für mich diese Semester-Arbeit auch so wertvoll. Ich konnte viel theoretisch vorhandenes Wissen in der Praxis in einem echten Projekt anwenden und traf erstmals auch auf "echte" Probleme, welche es zu lösen galt.

Auf die gesamte Arbeit blicke ich mit gemischten Gefühlen zurück. Einerseits bin ich stolz, was wir trotz einiger Widrigkeiten erreicht haben, andererseits sehe ich überall noch Dinge die verbessert werden könnten, wofür aber die Zeit nicht mehr gereicht hat. Manchmal denke ich, es wäre besser gewesen, früher mit der Implementation zu beginnen, dann hätten wir mehr Zeit gehabt, Dinge noch fertigzustellen oder zu implementieren. Andererseits haben wir jetzt schon mit der Implementation begonnen, als wir noch nicht zu hundert Prozent sattelfest im Bereich Firewall-Technologien waren, was wir teilweise teuer bezahlten, wenn wir Fehler, welche entstanden sind, weil wir die Technologie zu wenig gekannt haben, ausmerzen mussten.

Als Fazit kann ich sagen, dass ich die Semesterarbeit als eine sehr spannende, lehrreiche, aber auch anstrengende und manchmal nervenaufreibende Zeit empfunden habe. Gewisse Dinge würde ich sicherlich das nächste Mal anders angehen. Ich bedanke mich bei allen Personen, die mich in dieser Zeit unterstützt haben, insbesondere Roman Ammann, welcher uns während der gesamten Semesterarbeit betreut und mit guten Ratschlägen unterstützt hat.

16.2 Gieri Kohler

Das Projekt „Firewall-Analyse-Tool“ war insgesamt ein interessantes Projekt, bei dem Netzwerk- sowie Programmierkenntnisse gleichermassen eingesetzt und ausgebaut werden konnten. Gerade aufgrund der Vielfältigkeit war es meiner Meinung nach aber nicht ganz einfach und eher zeitaufwändig, sich in die gesamte Thematik einzuarbeiten. Für die Lösung der Aufgabe war es notwendig, dass wir uns in Themen wie iptables, Cisco IOS und Cisco ASA einarbeiten und nebenbei noch Technologien wie JFlex kennenlernen. Viele Technologien waren für mich vor dem Projektstart noch beinahe unbekannt. Einzig Cisco IOS kannte ich ein wenig, da ich kurz vor dem Projekt die CCNA-Prüfung erfolgreich abgelegt hatte. So war es schwierig, den richtigen Zeitpunkt zu bestimmen, mit dem Einarbeiten in die Thematik aufzuhören und mit dem Programmieren zu beginnen. Als wir uns dann entschieden, mit dem Implementieren zu beginnen, da auch schon einige Wochen vergangen

waren, waren sicherlich noch einige Dinge nicht komplett durchdacht, was sich später noch einige Mal rächte. Immer wieder mussten Änderungen an der Problem Domain durchgeführt werden, weil sich beispielsweise noch niemand Gedanken darüber gemacht hatte, wie denn ein bestimmtes Feature von iptables in das allgemeine Format umgesetzt werden soll.
Aufgrund der unterschiedlichen, auszuführenden Arbeiten war es aber wiederum einfach, die Aufgabe aufzuteilen, wobei ich die Implementierung der Parser übernahm und mein Projektpartner sich mit der Implementierung der Persistenz und der Matching-Algorithmen befasste. Als Hilfsmittel für die Implementierung der Parser verwendete ich den Scannergenerator JFlex. Trotzdem hat das Schreiben der Parser einen beträchlichen Zeitanteil eingenommen. Deshalb bin ich mir jetzt am Ende des Projekts auch nicht mehr ganz sicher, ob JFlex die beste Lösung für diese Aufgabe war, oder ob es sich gelohnt hätte, weitere Parsergeneratoren für Java unter die Lupe zu nehmen.
Umso mehr wir uns mit Cisco IOS und iptables beschäftigten, desto mehr merkten wir, wie umfangreich die Möglichkeiten sind. Vor allem bei iptables ist dies der Fall. So haben wir es in der gegebenen Zeit leider nicht geschafft, die gesamte Funktionalität umzusetzen und auch der Parser für Cisco ASA konnte nicht mehr implementiert werden.
Als Fazit könnte man sagen, dass wir eine noch nicht ganz ausgereifte Software als gute Grundlage entwickelt haben, die noch Potential für Weiterentwicklungen hat.

17 Glossar

ACL:	Access Control List ist ein Begriff, welcher Cisco für eine geordnete Liste von Filterregeln verwendet. Bei Cisco-Geräten kann eine ACL einem Interface zugeordnet und eine Richtung festgelegt werden, um festzulegen, wo diese wirken soll.
API	Als Application Programming Interface wird eine Schnittstelle bezeichnet, welche die Anbindung eines Systems hinter der APIüber diese Schnittstelle ermöglicht.
BLOB	Binary Large Object sind grosse binäre Objekte wie Audiodaten oder Bilder.
CLI	Als Command Line Interface wird ein Eingabebereich für die Steuerung von Software bezeichnet, der textbasiert funktioniert. Dies im Gegensatz zu einem Graphical-User-Interface (GUI).
JPA	Die Java Persistence API ist eine Schnittstelle aus der Java EE-Umgebung für Java-Programme, welche die Zuordnung von Objekten zu Einträgen in eine Datenbank vereinfacht.
NAT	Network Address Translation bezeichnet das Übersetzen von IP-Adressen oder Ports durch ein Netzwerkgerät, z.B. eine Firewall.
Netmask/Netzwerkmaske:	Die Netzmaske, Netzwerkmaske oder Subnetzmaske ist eine Bitmaske, die im Netzwerkprotokoll IPv4 bei der Beschreibung von IP-Netzen angibt, wie viele Bits am Anfang der dargestellten IP-Adresse das Netzpräfix ausmachen. Für IPv6 spricht man von der Präfixlänge. (aus Wikipedia-Artikel: Netzmaske, am 03.12.2009)
ORM	Object-Relational-Mapping bezeichnet eine Technik in der Software-Entwicklung mit welcher Objekt aus einer objektorientierten Programmiersprache in eine relationale Datenbank abgelegt werden.
Wildcard:	Die Wildcard wird bei Cisco beispielsweise bei ACLs verwendet. Wie die Netzwerkmaske ist sie eine Bitmaske, die bei IPv4 angibt, wieviele Bits am Anfang der IP-Adresse das Netzpräfix ausmachen. Im Unterschied zur Netzwerkmaske geben allerdings führende Nullen die relevanten Bits an, gefolgt von Einsen. Bei der Netzwerkmaske ist dies genau umgekehrt.

18 Definitionen

18.1 Assigned Internet Protocol Numbers[11]

Decimal	Keyword	Protocol
0	HOPOPT	IPv6 Hop-by-Hop Option
1	ICMP	Internet Control Message
2	IGMP	Internet Group Management
3	GGP	Gateway-to-Gateway]
4	IP	IP in IP (encapsulation)
5	ST	Stream
6	TCP	Transmission Control
7	CBT	CBT
8	EGP	Exterior Gateway Protocol
9	IGP	any private interior gateway (used by Cisco for their IGRP)
10	BBN-RCC-MON	BBN RCC Monitoring
11	NVP-II	Network Voice Protocol
12	PUP	PUP
13	ARGUS	ARGUS
14	EMCON	EMCON
15	XNET	Cross Net Debugger
16	CHAOS	Chaos
17	UDP	User Datagram
18	MUX	Multiplexing
19	DCN-MEAS	DCN Measurement Subsystems
20	HMP	Host Monitoring
21	PRM	Packet Radio Measurement
22	XNS-IDP	XEROX NS IDP
23	TRUNK-1	Trunk-1
24	TRUNK-2	Trunk-2
25	LEAF-1	Leaf-1
26	LEAF-2	Leaf-2
27	RDP	Reliable Data Protocol
28	IRTP	Internet Reliable Transaction
29	ISO-TP4	ISO Transport Protocol Class 4
30	NETBLT	Bulk Data Transfer Protocol
31	MFE-NSP	MFE Network Services Protocol
32	MERIT-INP	MERIT Internodal Protocol
33	DCCP	Datagram Congestion Control Protocol
34	3PC	Third Party Connect Protocol
35	IDPR	Inter-Domain Policy Routing Protocol
36	XTP	XTP
37	DDP	Datagram Delivery Protocol
38	IDPR-CMTP	IDPR Control Message Transport Protocol
39	TP++	TP++ Transport Protocol
40	IL	IL Transport Protocol
41	IPv6	Ipv6
42	SDRP	Source Demand Routing Protocol
43	IPv6-Route	Routing Header for IPv6
44	IPv6-Frag	Fragment Header for IPv6
45	IDRP	Inter-Domain Routing Protocol
46	RSVP	Reservation Protocol
47	GRE	General Routing Encapsulation
48	DSR	Dynamic Source Routing Protocol
49	BNA	BNA

[11] Quelle: [9]

50	ESP	Encap Security Payload
51	AH	Authentication Header
52	I-NLSP	Integrated Net Layer Security TUBA
53	SWIPE	IP with Encryption
54	NARP	NBMA Address Resolution Protocol
55	MOBILE	IP Mobility
56	TLSP	Transport Layer Security Protocol using Kryptonet key management
57	SKIP	SKIP
58	IPv6-ICMP	ICMP for IPv6
59	IPv6-NoNxt	No Next Header for IPv6
60	IPv6-Opts	Destination Options for IPv6
61		any host internal protocol
62	CFTP	CFTP
63		any local network
64	SAT-EXPAK	SATNET and Backroom EXPAK
65	KRYPTOLAN	Kryptolan
66	RVD	MIT Remote Virtual Disk Protocol
67	IPPC	Internet Pluribus Packet Core
68		any distributed file system
69	SAT-MON	SATNET Monitoring
70	VISA	VISA Protocol
71	IPCV	Internet Packet Core Utility
72	CPNX	Computer Protocol Network Executive
73	CPHB	Computer Protocol Heart Beat
74	WSN	Wang Span Network
75	PVP	Packet Video Protocol
76	BR-SAT-MON	Backroom SATNET Monitoring
77	SUN-ND	SUN ND PROTOCOL-Temporary
78	WB-MON	WIDEBAND Monitoring
79	WB-EXPAK	WIDEBAND EXPAK
80	ISO-IP	ISO Internet Protocol
81	VMTP	VMTP
82	SECURE-VMTP	SECURE-VMTP
83	VINES	VINES
84	TTP	TTP
85	NSFNET-IGP	NSFNET-IGP
86	DGP	Dissimilar Gateway Protocol
87	TCF	TCF
88	EIGRP	EIGRP
89	OSPFIGP	OSPFIGP
90	Sprite-RPC	Sprite RPC Protocol
91	LARP	Locus Address Resolution Protocol
92	MTP	Multicast Transport Protocol
93	AX.25	AX.25 Frames
94	IPIP	IP-within-IP Encapsulation Protocol
95	MICP	Mobile Internetworking Control Protocol
96	SCC-SP	Semaphore Communications Sec. Protocol
97	ETHERIP	Ethernet-within-IP Encapsulation
98	ENCAP	Encapsulation Header
99		any private encryption scheme
100	GMTP	GMTP
101	IFMP	Ipsilon Flow Management Protocol
102	PNNI	PNNI over IP
103	PIM	Protocol Independent Multicast
104	ARIS	ARIS
105	SCPS	SCPS
106	QNX	QNX
107	A/N	Active Networks
108	IPComp	IP Payload Compression Protocol
109	SNP	Sitara Networks Protocol
110	Compaq-Peer	Compaq Peer Protocol
111	IPX-in-IP	IPX in IP
112	VRRP	Virtual Router Redundancy Protocol

```
113         PGM                 PGM Reliable Transport Protocol
114                             any 0-hop protocol
115         L2TP                Layer Two Tunneling Protocol
116         DDX                 D-II Data Exchange (DDX)
117         IATP                Interactive Agent Transfer Protocol
118         STP                 Schedule Transfer Protocol
119         SRP                 SpectraLink Radio Protocol
120         UTI                 UTI
121         SMP                 Simple Message Protocol
122         SM                  SM
123         PTP                 Performance Transparency Protocol
124         ISIS over IPv4
125         FIRE
126         CRTP                Combat Radio Transport Protocol
127         CRUDP               Combat Radio User Datagram
128         SSCOPMCE
129         IPLT
130         SPS                 Secure Packet Shield
131         PIPE                Private IP Encapsulation within IP
132         SCTP                Stream Control Transmission Protocol
133         FC                  Fibre Channel
134         RSVP-E2E-IGNORE
135         Mobility Header
136         UDPLite
137         MPLS-in-IP
138         manet               MANET Protocols
139         HIP                 Host Identity Protocol
140         Shim6               Shim6 Protocol
141-252                         Unassigned
253                             Use for experimentation and testing
254                             Use for experimentation and testing
255                             Reserved
```

18.2 Port Numbers[12]

```
Keyword     Decimal             Description
------------------------------------------------------------------------
ftp-data    20/TCP              File Transfer [Default Data]
ftp         21/TCP              File Transfer [Control]
telnet      23/TCP              Telnet
smtp        25/TCP              Simple Mail Transfer
tftp        69/UDP              Trivial File Transfer
www-http    80/TCP              World Wide Web HTTP
pop3        110/TCP             Post Office Protocol Version 3
snmp        161/UDP             SNMP
https       443/TCP,UDP         HTTPS
```

18.3 ICMP Type Numbers[13]

```
Type   Name
------------------------------------------------------------------------------
0      Echo Reply
         Codes
           0  No Code
1      Unassigned
2      Unassigned
3      Destination Unreachable
         Codes
           0  Net Unreachable
```

[12] Quelle: [10]
[13] Quelle: [11]

```
            1 Host Unreachable
            2 Protocol Unreachable
            3 Port Unreachable
            4 Fragmentation Needed and Don't Fragment was Set
            5 Source Route Failed
            6 Destination Network Unknown
            7 Destination Host Unknown
            8 Source Host Isolated
          9 Communication with Destination Network is Administratively
            Prohibited
            10 Communication with Destination Host is Administratively
              Prohibited
            11 Destination Network Unreachable for Type of Service
            12 Destination Host Unreachable for Type of Service
            13 Communication Administratively Prohibited
            14 Host Precedence Violation
            15 Precedence cutoff in effect
4       Source Quench
          Codes
            0 No Code
5       Redirect
        Codes
          0 Redirect Datagram for the Network (or subnet)
          1 Redirect Datagram for the Host
          2 Redirect Datagram for the Type of Service and Network
          3 Redirect Datagram for the Type of Service and Host
6       Alternate Host Address
        Codes
          0 Alternate Address for Host
7       Unassigned
8       Echo
          Codes
            0 No Code
9       Router Advertisement
          Codes
          0 Normal router advertisement
          16 Does not route common traffic
10      Router Selection
          Codes
          0 No Code
11      Time Exceeded
          Codes
            0 Time to Live exceeded in Transit
            1 Fragment Reassembly Time Exceeded
12      Parameter Problem
          Codes
          0 Pointer indicates the error
          1 Missing a Required Option
          2 Bad Length
13      Timestamp
          Codes
            0 No Code
14      Timestamp Reply
          Codes
            0 No Code
15      Information Request
          Codes
            0 No Code
16      Information Reply
          Codes
            0 No Code
17      Address Mask Request
          Codes
            0 No Code
```

```
18      Address Mask Reply
          Codes
             0  No Code
19      Reserved (for Security)
20-29   Reserved (for Robustness Experiment)
30      Traceroute
31      Datagram Conversion Error
32      Mobile Host Redirect
33      IPv6 Where-Are-You
34      IPv6 I-Am-Here
35      Mobile Registration Request
36      Mobile Registration Reply
39      SKIP
40      Photuris
          Codes
             0 Bad SPI
             1 Authentication Failed
             2 Decompression Failed
             3 Decryption Failed
             4 Need Authentication
             5 Need Authorization
41-252  Unassigned
253     RFC3692-style Experiment 1
254     RFC3692-style Experiment 2
```

19 Literaturverzeichnis

[1] FirewallBuilder http://www.fwbuilder.org/, letzter Zugriff am 15.12.2009
[2] JFLex Download: http://jflex.de/download.html, letzter Zugriff am 15.12.2009
JFlex Manual: http://jflex.de/manual.html, letzter Zugriff am 15.12.2009
[3] Log4j Download: http://logging.apache.org/log4j/1.2/index.html, letzter Zugriff am 15.12.2009
Log4j Manual/Short Introduction: http://logging.apache.org/log4j/1.2/manual.html, letzter Zugriff am 15.12.2009
[4] SQLite Dokumentation: http://www.sqlite.org/docs.html, letzter Zugriff am 15.12.2009
[5] a. JPA Dokumentation:
http://java.sun.com/javaee/5/docs/tutorial/doc/bnbpz.html, letzter Zugriff am 15.12.2009

b. Präsentationsfolien Vorlesung EnCom von Wolfgang Giersche:
http://skripte.hsr.ch/Informatik/Fachbereich/Enterprise_Computing/EnCom/HS09/Vorlesungen/06_JPA_Basics/JPA_Basics.pdf
http://skripte.hsr.ch/Informatik/Fachbereich/Enterprise_Computing/EnCom/HS09/Vorlesungen/07_JPA_II/JPA_ORM.pdf
http://skripte.hsr.ch/Informatik/Fachbereich/Enterprise_Computing/EnCom/HS09/Vorlesungen/08_JPA_III_EJBQL/JPA_III.pdf

c. Hibernate Dokumentation:
https://www.hibernate.org/5.html, letzter Zugriff am 15.12.2009
[6] iptables-save(8) - Linux man page: http://linux.die.net/man/8/iptables-save, letzter Zugriff am 15.12.2009
[7] ip6tables-save(8) - Linux man page: http://linux.die.net/man/8/ip6tables-save, letzter Zugriff am 15.12.2009
[8] Cisco, NAT Order Of Operation http://www.cisco.com/en/US/tech/tk648/tk361/technologies_tech_note09186a0080133ddd.shtml, letzter Zugriff am 15.12.2009
[9] IANA Protocol Numbers: http://www.iana.org/assignments/protocol-numbers/, letzter Zugriff am 15.12.2009
[10] IANA Port Numbers: http://www.iana.org/assignments/port-numbers, letzter Zugriff am 15.12.2009
[11] IANA ICMP Type Numbers: http://www.iana.org/assignments/icmp-parameters, letzter Zugriff am 15.12.2009

19.1 Weiterführende Literatur

- CCNA ICND2 Prüfungshandbuch, 2. Auflage von Wendell Odom, deutsche Übersetzung von Christian Alkemper, Addison Wesley Verlag, ISBN 978-3-8273-2635-5
Kapitel 6 über IP-ACLs, Seite 297-341
Kapitel 16 über NAT, Seite 653-688

- Cisco Pix Firewall, mitp Verlag, ISBN 3-8266-1305-8,
Firewall-Schnittstellen: innen, aussen und DMZ, Seite 46-50
Adaptive Security Algorithm, Seite 67-78

- Cisco IOS Release 12.0 Security Configuration Guide: http://www.cisco.com/en/US/docs/ios/12_0/security/configuration/guide/secur_c.html, letzter Zugriff am 15.12.2009

- Cisco Security Appliance Command Reference Version 8.0:
http://www.cisco.com/en/US/docs/security/asa/asa80/command/reference/cmd_ref.html, letzter Zugriff am 15.12.2009

- Cisco, Configuring IP Access Lists: http://www.cisco.com/en/US/products/sw/secursw/ps1018/products_tech_note09186a00800a5b9a.shtml#receiveacl, letzter Zugriff am 15.12.2009

- Implementing Traffic Filters and Firewalls for IPv6 Security: http://www.cisco.com/en/US/docs/ios/ipv6/configuration/guide/ip6-sec_trfltr_fw.html#wp1072424, letzter Zugriff am 15.12.2009

- Configuring Network Address Translation: Getting Started: http://www.cisco.com/en/US/tech/tk648/tk361/technologies_tech_note09186a0080094e77.shtml, letzter Zugriff am 15.12.2009

- Tutorial, NAT mit iptables: http://www.karlrupp.net/de/computer/nat_tutorial, letzter Zugriff am 15.12.2009

- iptables manual: http://linux.die.net/man/8/iptables, letzter Zugriff am 15.12.2009

- ip6tables manual: http://linux.die.net/man/8/ip6tables

20 Erklärung über die eigenständige Arbeit

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt haben, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde.
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.

Ort, Datum:

Namen, Unterschriften:

Gieri Kohler

Christoph Rebsamen