Grafische Oberfläche für die Konfiguration eines verteilten Firewallsystems

Diplomarbeit

im Studiengang Elektronik/Technische Informatik
der
Fachhochschule Aalen

Michael Hübner

Betreuer: Prof. Dr.-Ing. Manfred Strahnen

Durchgeführt im WS 2002 / 2003

Kurzfassung

Gegenstand der hier vorgestellten Arbeit ist die Konzeption und Erstellung eines Tools zur Konfiguration eines verteilten Firewallsystems. Ein solches Firewallsystem ist aus mehreren, mit dem Betriebssystem Linux ausgestatteten Rechnern aufgebaut, die die Funktionen von Paketfiltern und Proxys übernehmen. Die Aufgabe eines Firewallsystems besteht darin, den Zugriff auf ein Netzwerk zu beschränken und das Netzwerk vor Angriffen zu schützen. Es kann ein- und ausgehende Daten auf Protokoll- und Anwendungsebene überprüfen, manipulieren und zurückweisen. Jeder Rechner dieses Firewallsystems wird im Allgemeinen mittels Scripten konfiguriert. Da die Erstellung von Konfigurationsscripten jedoch alles andere als trivial ist, soll das Konfigurationstool dem Administrator des Firewallsystems Hilfestellung geben. Es soll die Komplexität von Filterregeln vor dem Anwender verbergen und ihn bei deren Erstellung unterstützen. Ein verteiltes Firewallsystem wird bereits im Netzwerklabor sowie im Fachbereichsnetz des Fachbereichs Elektronik/Technische Informatik eingesetzt.

Schlagwörter: Firewall, Linux, Paketfilter, Proxy, Scripte

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Vorwort

Die Arbeit basiert auf Vorleistungen, die im Rahmen von Diplomarbeiten von Bernd Müller, Jochen Zeller und Gregor Domhan an der Fachhochschule Aalen erbracht wurden.

Die Notwendigkeit, den Zugriff auf Hochschul- /Firmennetze zu beschränken bzw. zu reglementieren hat bereits in der Anfangszeit des Internet dazu geführt, ein System zu entwickeln, das die Filterung und Manipulation von Paketen ermöglicht. Seit der 1.1er Serie ist der Paketfilter Bestandteil des Linuxkernel. Die erste Version, basierend auf ‚ipfw' von BSD[1], wurde von Alan Cox 1994 portiert. Das Tool ‚ipfwadm' kontrollierte die Filterregeln des Kernels. 1998 wurde der Kernel von Rusty Russell und Michael Neuling überarbeitet und das Tool ‚ipchains' eingeführt. 1999 wurde der Kernel komplett überarbeitet und die vierte Generation, ‚iptables', eingeführt.

Auch wenn sich mit ‚iptables' die Konfiguration des Filters im Kernel vereinfacht hat, ist sie jedoch alles andere als trivial. Ein Regelsatz für ein kleines Netzwerk kann so leicht zu einer beinahe undurchschaubaren Flut von voneinander abhängigen Regeln werden. Um den Administrator einer Firewall bei deren Konfiguration zu unterstützen, wurde im Rahmen dieser Diplomarbeit das Konfigurationstool „FirewallConfig" entwickelt. Es ist eine Weiterentwicklung des Tools „Firewall 1.0" von Bernd Müller, welches im Sommersemester 2002 im Rahmen einer Diplomarbeit an der Fachhochschule Aalen entwickelt wurde. Der Ansatz zur Lösung des Problems der Handhabbarkeit von Filterregeln wurde dahingehend geändert, dass vor Beginn der Konfiguration eines Firewallrechners die Topologie des Netzwerkes in Form von Stammdaten erfasst wird. Durch die Trennung von Datenerfassung und Regelerstellung wird erreicht, dass die Daten bei ihrer Erfassung überprüft werden können und Änderungen der Topologie direkt Änderungen der Regeln auslösen.

[1] BSD: Berkeley Software Design, Inc

1 Überblick

Gegenstand dieser Diplomarbeit ist die Entwicklung und Umsetzung eines Konzeptes, Filterregeln für einen Paketfilter möglichst einfach und effizient zu erstellen. Dazu ist es notwendig, die Abhängigkeiten von Filterregeln von der gegebenen Topologie, den Protokollen und Anwendungen aufzuzeigen. Änderungen an der Topologie sollen sich direkt in Änderungen der Filterregeln niederschlagen. Die Erstellung der Regeln soll möglichst unabhängig von der verwendeten Filtersoftware sein, sich leicht an neue Technologien anpassen lassen und in einem hohen Maße die verwendete Filtersoftware abstrahieren. Der Lösungsansatz basiert auf Objektorientierter Analyse und Objektorientierter Programmierung, die sozusagen „state of the art“ sind. Dadurch wird die Wiederverwendbarkeit des Codes erhöht sowie ein hohes Maß an Wartbarkeit und Erweiterbarkeit gewährleistet. Die entstandene Softwarelösung ist modular aufgebaut, so dass Änderungen im Design, der Logik oder der Datenhaltung relativ problemlos durchführbar sind. Die Minimierung von Redundanz in den Datensätzen gewährleistet die Integrität der Daten. Eine Dreiteilung der Software in Benutzeroberfläche (GUI[2]), Progammlogik und Datenhaltung vereinfacht die Programmierung und das Verständnis für den Programmablauf. Die Entscheidung, die Programmiersprache PHP4 zu verwenden, wurde bereits im Vorfeld der Diplomarbeit getroffen. PHP4 hat den Vorteil, dass es sich um eine relativ leicht zu erlernende Scriptsprache handelt, von Hause aus netzwerkfähig ist und die erzeugte Bedienoberfläche über einen Webserver plattformunabhängig an Clients verteilt.

[2] GUI: graphical user interface

2 Ziele

Ziel dieser Diplomarbeit ist es, ein schlüssiges Konzept zu entwickeln, mit dessen Hilfe ein Prototyp eines Konfigurationstools für ein verteiltes Firewallsystem erstellt wird. An diesem Prototyp können in folgenden Arbeiten Untersuchungen vorgenommen werden, um den erforderlichen Funktionsumfang und den Grad der Automatisierbarkeit der Erstellung komplexer Filterregeln zu ermitteln. Das folgende Pflichtenheft konkretisiert die Anforderungen an das Konfigurationstool.

2.1 Pflichtenheft

2.1.1 Zielbestimmung

Der Fachbereich Elektronik/Technische Informatik der Fachhochschule Aalen soll durch das Produkt in die Lage versetzt werden, das vorhandene verteilte Firewallsystem zu administrieren.

2.1.1.1 Musskriterien

- Verwalten der Konfigurationen der Firewallrechner
- Erstellung von Scipten für die Firewallrechner

2.1.1.2 Wunschkriterien

- Mehrbenutzerbetrieb
- Sichere Übertragung der Startscripte auf die Firewallrechner
- Starten und Stoppen von Prozessen auf den Firewallrechnern

2.1.1.3 Abgrenzungskriterien

- keine

2.1.2 Produkteinsatz

Das Produkt dient zur Paketfilteradministration des Fachbereichs Elektronik/Technische Informatik der Fachhochschule Aalen.

2.1.2.1 Anwendungsbereiche

- Administration der Firewall im Fachbereich Elektronik/Technische Informatik der Fachhochschule Aalen

2.1.2.2 Zielgruppen

- Netzwerk-Administratoren der Fachhochschule Aalen

2.1.2.3 Betriebsbedingungen

- Büroumgebung

2.1.3 Produktumgebung

- Das Serverprogramm läuft auf einem dedizierten Rechner.
- Das Clientprogramm läuft auf einem beliebigen Rechner.

2.1.3.1 Software

- Serversoftware mit Apache Webserver und PHP4
- Plattformunabhängige Clientsoftware

2.1.3.2 3.2 Hardware

- PC

2.1.3.3 Orgware

- Netzwerkverbindung zum Webserver

2.1.3.4 Produkt Schnittstellen

- Die Konfigurationsdaten werden in einer MySQL Datenbank gespeichert.
- Die Konfigurationsdaten können auf einen beliebigen Datenträger exportiert werden.
- Die Startscripte werden auf einen beliebigen Datenträger exportiert.

2.1.4 Produktfunktionen

- Erfassung der Topologie des Netzwerkes (Rechner, Netzwerkkarten, Teilnetze)
- Erstellung von Filterregeln mittels der erfassten Topologie
- Erstellung von Startscripten mittels der erfassten Filterregeln
- Export und Import der Daten eines Projekts

3 Stand der Technik

Der Stand der Technik bei der Erstellung von Filterregeln für Paketfilter ist:

1. Einarbeitung in die Technik eines Paketfilters (tables, chains, targets, …).
2. Bestimmung der Charakteristika des zu entwerfenden Paketfilters (nur bestimmte Pakete sperren oder nur bestimmte Pakete durchlassen).
3. Analyse der Teilnehmer des zu schützenden Netzes (Rechner, benötigte Dienste).
4. Erfassung der Daten der Teilnehmer (Rechnernamen, IP Adresse, Netzmaske, …).
5. Erstellung der Filterregeln aus den Daten der Teilnehmer in Form eines Scriptes.
6. Test der Syntax der erstellten Filterscripte.
7. Start des Paketfilters und Test der Filterregeln.

Die entscheidenden Nachteile bei dieser Art der Vorgehensweise sind:

1. Die Daten der Teilnehmer müssen gegebenenfalls mehrfach erfasst werden.
2. Bei Änderungen müssen die Scripte komplett von Hand überarbeitet werden, wobei die Gefahr besteht, dass etwas übersehen wird.
3. Die Syntax der jeweiligen Filtersoftware muss geläufig sein.

4 Eingeschlagener Realisierungsweg

4.1 Vorüberlegungen

Um die unter „3 Stand der Technik“ aufgeführten Nachteile der konservativen Erstellung von Paketfilterregeln aufzuheben, sind an dieser Stelle einige Vorüberlegungen angebracht.

Betrachtet man ein Firewallsystem, sei es nun ein einzelner Rechner oder ein verteiltes System, so besteht seine primäre Aufgabe darin, Pakete, die an einen Rechner im Netz adressiert sind oder von einem Rechner im Netz kommen, auf Absender und Empfänger zu untersuchen und entweder passieren zu lassen oder zu verwerfen. Sekundär kann ein solches Firewallsystem dazu eingesetzt werden, den Datenverkehr ganz oder teilweise zu protokollieren und/oder zu manipulieren. Wird darüber hinaus eine Proxy-Software verwendet, so kann auch auf Anwendungsebene der Inhalt des Datenverkehrs untersucht und manipuliert werden.

4.2 Analyse

Um die Komplexität eines solchen Firewallsystems aufzubrechen, muss nach dem Grundsatz „divide et impere[3]“ das Problem in handhabbare Teile zerlegt werden. Dabei bietet sich als erste Maßnahme an, die Topologie, d.h. alle Teilnetze und Rechner, zu erfassen [Abbildung 1: Topologie]. Die Topologie besteht demnach aus Rechnern, die Teile von Netzwerken sind. Noch weiter heruntergebrochen besteht eine Topologie aus Rechnern, die eine oder mehrere Netzwerkkarten besitzen, wobei die Menge der Netzwerkkarten, die die gleiche Netzadresse haben, ein Netzwerk bilden. In „Abbildung 2: Statisches Modell der Topologie“ ist die Toplogie mit Hilfe von Klassendiagrammen dargestellt. Ein weiterer Punkt ist die Erfassung aller relevanten Attribute, die die Topologie charakterisieren.

Ist die Topologie erst einmal abgebildet, kann ein anderer Teil der Software die verifizierten Daten dazu benutzen, Filterregeln zu erstellen. Diese Aufgabe übernimmt das Firewallsystem. Das Firewallsystem erhält von der Topologie eine Liste aller Rechner, eine Liste aller Netzwerkkarten sowie eine Liste aller Netzwerke. Da die Elemente der Listen untereinander verlinkt sind, entsteht eine Struktur ähnlich der eines Baumes. Im Firewallsystem besteht die Möglichkeit, Services zu definieren. Ein Service ist quasi die Zuordnung einer Kombination aus Port und Protokoll zu einem Keyword. Ein solcher Service kann dazu benutzt werden, eine Filterregel mit einer Anwendung zu verknüpfen (siehe 4.2.2.2 Klassendiagramm Service).

[3] divide et impere: Teile und Herrsche

Werden nun bei der Erstellung von Filterregeln, aus den Daten der Topologie und den Services, Links auf diese Daten gesetzt, so ist gewährleistet, dass sich Änderungen in der Topologie oder den Services unmittelbar auf die Filterregeln auswirken. Damit wird ein weiterer Nachteil der konventionellen Methode der Erstellung von Filterregeln eliminiert. Ein weiterer Vorteil dieses Ansatzes ist, dass sich das Modul Firewallsystem relativ leicht an veränderte Gegebenheiten anpassen lässt.

Anmerkung: Die Klassendiagramme wurden mit der Dokumentationsfunktion von Together® erstellt. In Together® ist es leider nicht möglich, or- und subset-Restriktionen darzustellen.

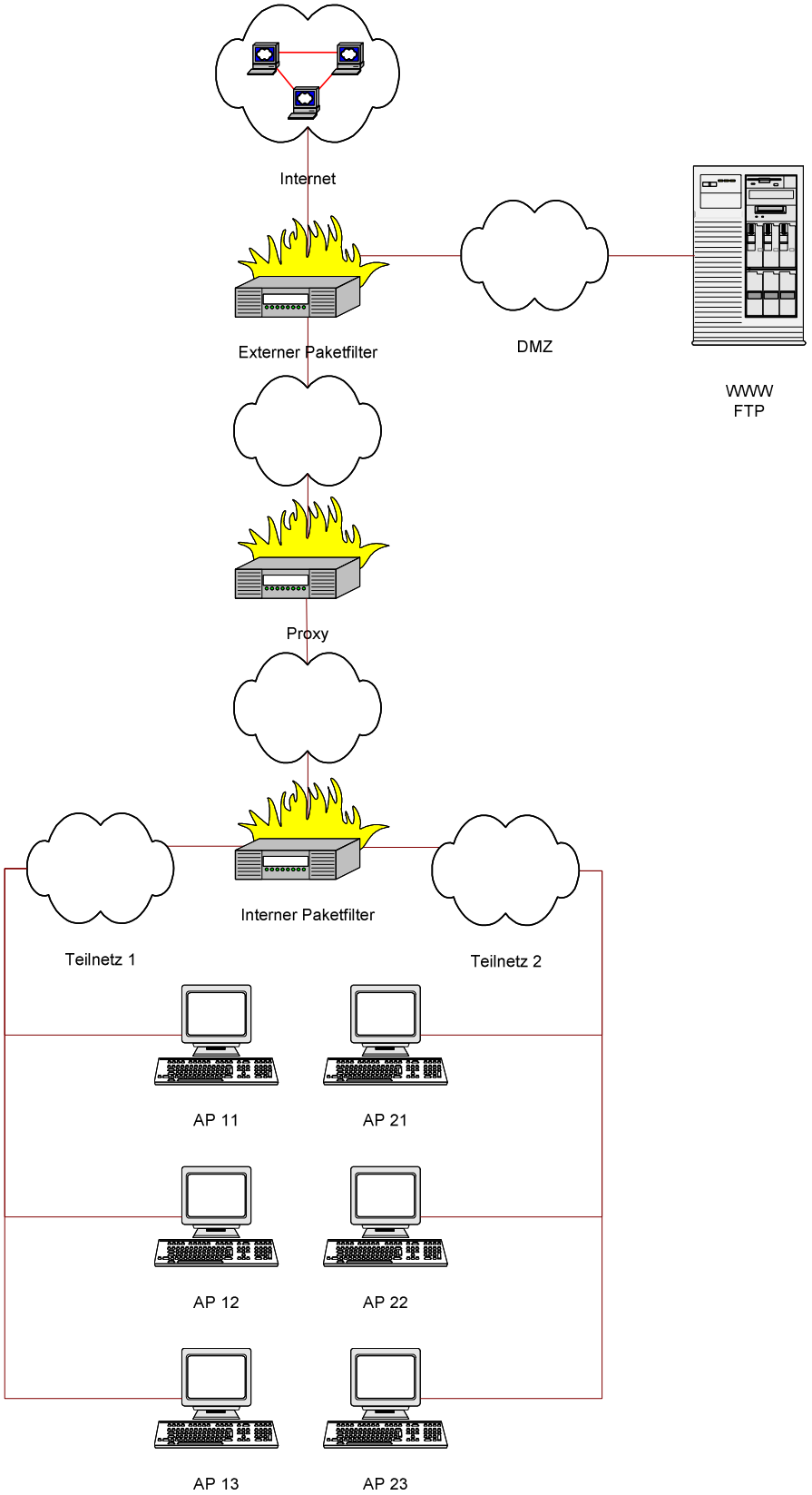

Abbildung 1: Topologie

4.2.1 Topologie

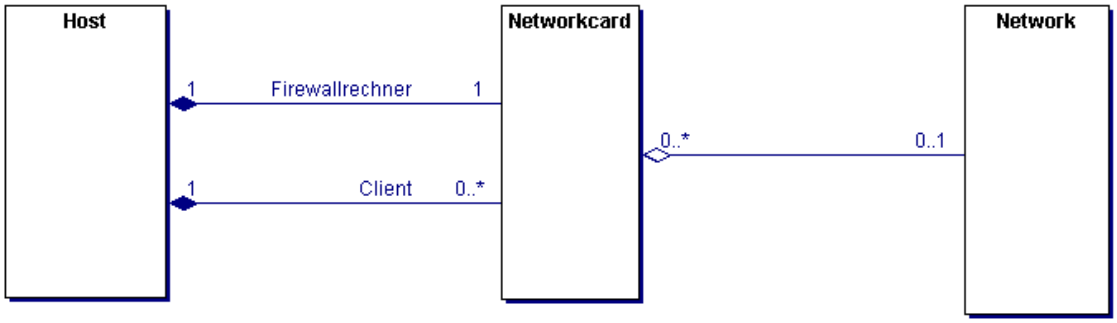

Abbildung 2: Statisches Modell der Topologie

4.2.1.1 Statisches Modell der Topologie

Im statischen Modell besteht die Topologie aus „Hosts“, „Networkcards“ und „Networks“. Ist ein Host Teil des Firewallsystems, so hat er 0…* Networkcards. Ist der Host ein Client oder Server, hat er genau eine Networkcard. Eine Networkcard ist genau dann Teil eines Networks, wenn eine IP Adresse und eine Netzmaske angegeben sind. Ein Network kann 0…* Networkcards enthalten. Ein Host hat die volle Kontrolle über seine Networkcards, d.h. er kann Networkcards anlegen und löschen. Eine Networkcard kann ein Network erstellen, jedoch nicht löschen. Ein Network kann nur gelöscht werden, wenn es leer ist, d.h. keine Networkcards enthält.

4.2.1.2 Klassendiagramm Host

In „Abbildung 3: Klasse Host“ ist die Klasse Host mit Attributen dargestellt. Ein Host ist für die Erstellung von Filterregeln eigentlich vollkommen irrelevant und dient nur zur Unterstützung der Abstraktion der Topologie gegenüber dem User.

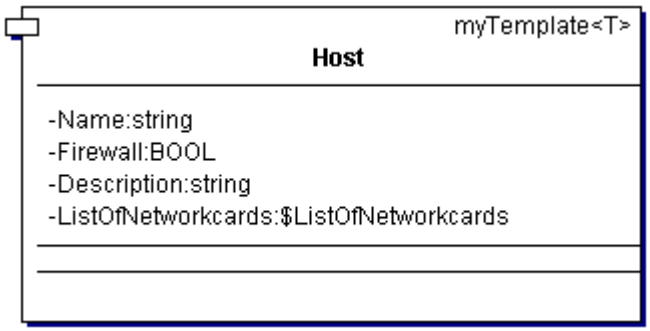

Abbildung 3: Klasse Host

- Der **Name** gibt dem Anwender die Möglichkeit, einen Rechner zu identifizieren.
- Das Attribut **Firewall** gibt Auskunft darüber, ob der Rechner Mitglied eines Firewallsystems ist.

- In **Description** kann ein beschreibender Text zu einem Rechner abgelegt werden.
- Die **ListOfNetworkcards** ist eine Liste, in der die Links zu den Netzwerkkarten des Rechners eingetragen werden. Ist der Rechner ein Client, so enthält diese Liste genau eine Netzwerkkarte.

4.2.1.3 Klassendiagramm Networkcard

In „Abbildung 4: Klasse Networkcard“ ist die Klasse Networkcard mit Attributen dargestellt.

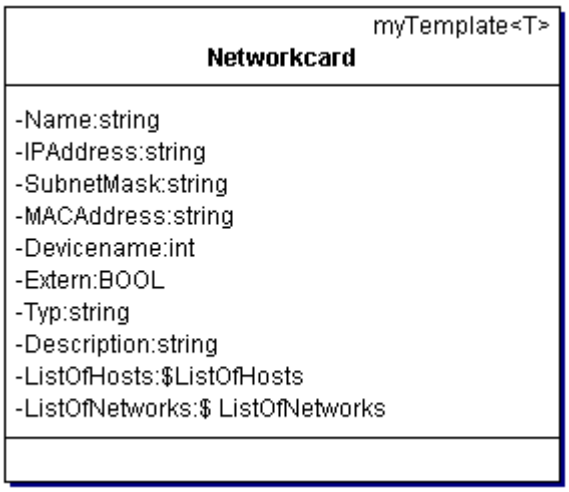

Abbildung 4: Klasse Networkcard

- Der **Name** gibt dem Anwender die Möglichkeit, für die Netzwerkkarte einen Alias zu vergeben.
- Die **IPAddress** ist die IP Adresse der Netzwerkkarte. Dieses Attribut ist in der Topologie eindeutig mit einer Ausnahme: der Adresse des Localloop Device.
- Die **MACAddress** ist die Hardwareadresse einer Ethernet-Karte.
- Die **SubnetMask** ist die Netzmaske des Netzes, in der sich die Karte befindet. Dieses Attribut wird benötigt, um die Netzadresse des zur Netzwerkkarte gehörenden Netzes auszurechnen.
- Der **Devicename** ist der Name, den Linux zur Kennzeichnung seiner Devices verwendet (eth0 für das erste Ethernet-Interface).
- Das **Extern** Attribut gibt an, ob eine IP Adresse eine offizielle oder eine private Adresse ist.
- Der **Typ** einer Netzwerkkarte bezeichnet die Technologie des Netzwerkes (Ethernet, Token Ring, …).

- In **Description** kann ein beschreibender Text zu einer Netzwerkkarte abgelegt werden.
- Die **ListOfHosts** ist eine Liste, in der der Link zum Rechner der Netzwerkkarte eingetragen wird.
- Die **ListOfNetworks** ist eine Liste, in der der Link zum Netzwerk eingetragen wird. Ist eine IP Adresse und eine Netzmaske angegeben, so enthält diese Liste genau einen Link, sonst keinen.

4.2.1.4 Klassendiagramm Network

In „Abbildung 5: Klasse Network“ ist die Klasse Network mit Attributen dargestellt.

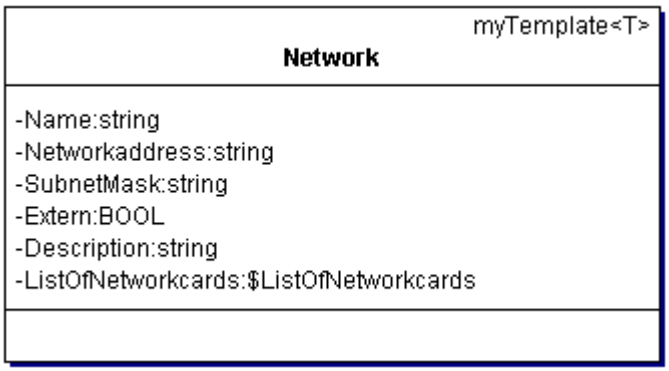

Abbildung 5: Klasse Network

- Der **Name** gibt dem Anwender die Möglichkeit, für das Netzwerk einen Namen zu vergeben.
- Die **Networkaddress** ist die Netzadresse des Netzwerkes.
- Die **SubnetMask** ist die Netzmaske des Netzes.
- Das **Extern** Attribut gibt an, ob eine Netzdresse eine öffentliche oder eine private Adresse ist.
- In **Description** kann ein beschreibender Text zu einem Netzwerk abgelegt werden.
- Die **ListOfNetworkkards** ist eine Liste, in der Links zu den Netzwerkkarten des Netzwerkes eingetragen werden. Diese Liste kann beliebig viele Einträge haben.

4.2.2 Firewallsystem

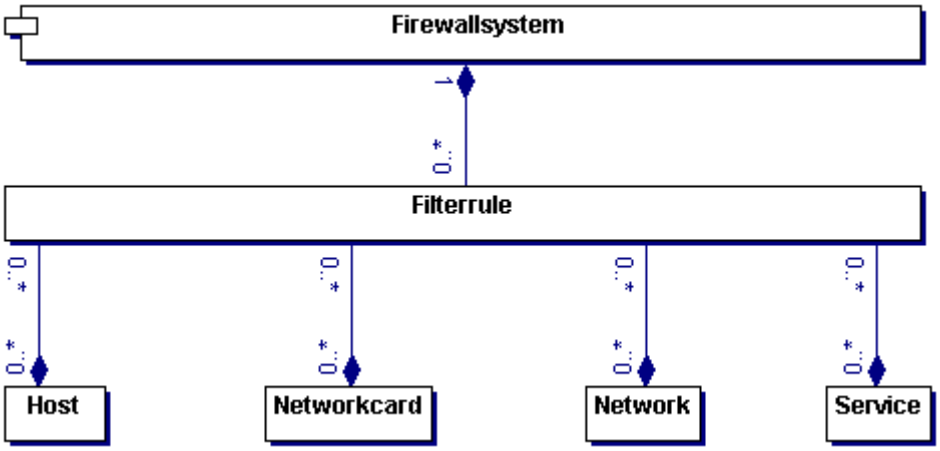

Abbildung 6: Statisches Modell des Firewallsystems

4.2.2.1 Statisches Modell der Firewallsystems

Wie im statischen Modell dargestellt [Abbildung 6: Statisches Modell des Firewallsystems], besteht das Firewallsystem aus einer Liste von Filterregeln, wobei sich jede Filterregel auf 0…* Services, 0…* Hosts, 0…* Networkcards und 0…* Networks bezieht. Auch ist es die Aufgabe eines Services, eines Hosts, einer Networkcard oder eines Networks, bei seiner Entfernung aus dem System alle Filterregeln zu löschen, die sich auf ihn beziehen. Daher benötigen die oben genannten Klassen eine zusätzliche Liste, um diese Links zu speichern.

4.2.2.2 Klassendiagramm Service

Ein Service ist eine Kombination aus einem Port und optional einem Protokoll. Mit Hilfe eines Service lassen sich Aufgaben bzw. Eigenschaften einer Verbindung definieren. Dabei ist es keineswegs notwendig, dass ein Service eindeutig ist. In „Abbildung 7: Klasse Service“ ist die Klasse Service mit Attributen dargestellt.

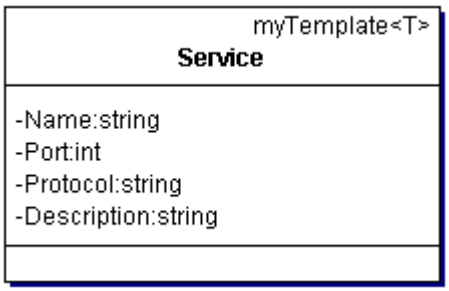

Abbildung 7: Klasse Service

- Der **Name** gibt dem Anwender die Möglichkeit, dem Service einen Namen zu vergeben.

- Der **Port** des Service spezifiziert den Service.
- Das **Protokoll** des Service dient gemeinsam mit dem Port zu seiner Spezifikation.
- In **Description** kann ein beschreibender Text zu einem Service abgelegt werden.

4.2.2.3 Klassendiagramm Filterrule

Die Filterregel wird aus bereits verifizierten Daten der Topologie und den Services erstellt. In „Abbildung 8: Klasse Filterrule“ ist die Klasse Filterrule mit Attributen dargestellt.

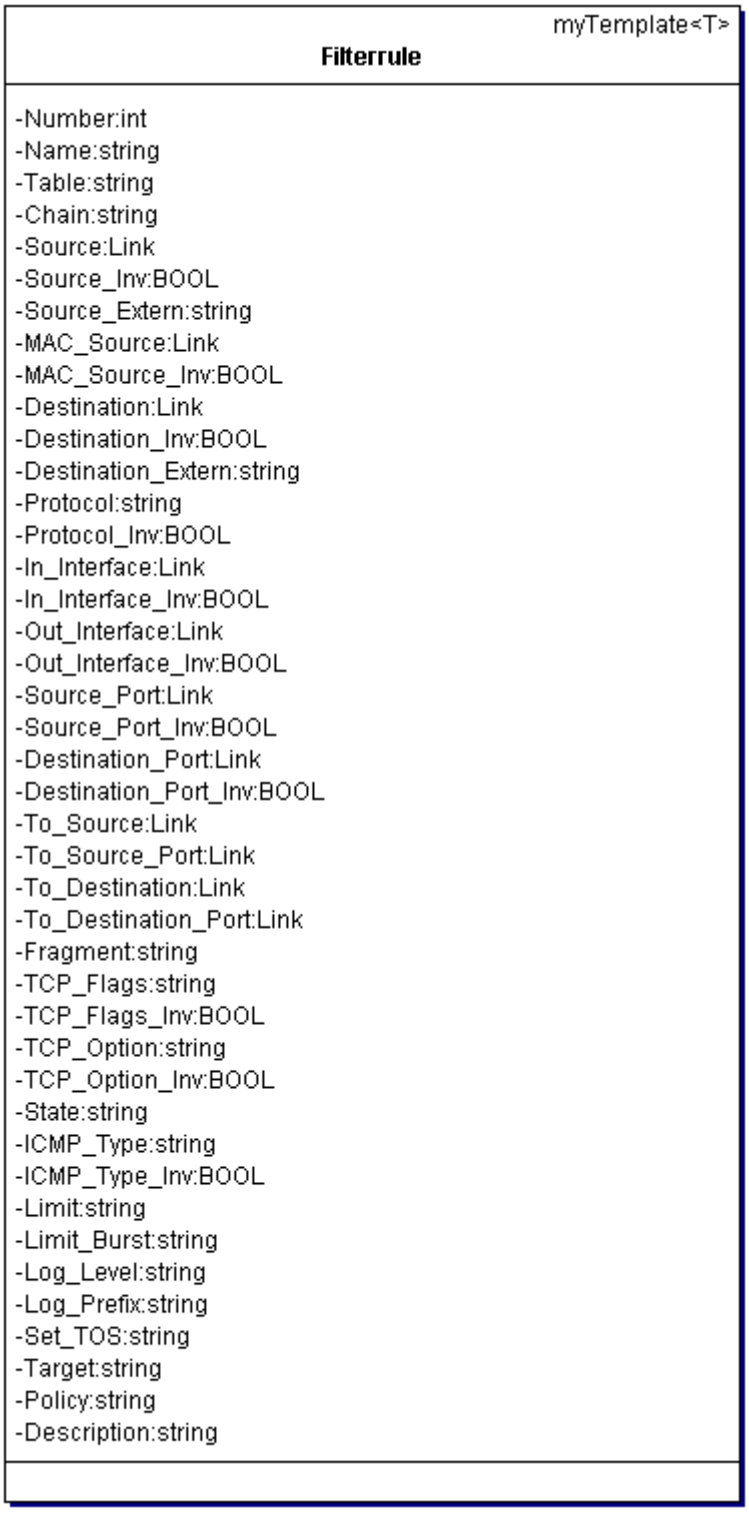

Abbildung 8: Klasse Filterrule

- Die **Number** ermöglicht es, die Reihenfolge von Filterregeln festzulegen.
- Der **Name** ermöglicht es dem Anwender, der Regel einen Namen zu vergeben.
- Der **Table** gibt an, welcher Art (Filter, NAT) die Regel ist.

- Die **Chain** gibt Auskunft über den Ort der Manipulation innerhalb des Paketfilters (INPUT, OUTPUT, FORWARD, PREROUTING, POSTROUTING).
- Die **Source** gibt Auskunft über den Absender eines Paketes und ist ein Link zu einer Netzwerkkarte oder einem Netzwerk in der Topologie.
- Mit **Source_Inv** kann die Source invertiert werden.
- **Source_Extern** gibt an, ob der Absender eines Paketes eine öffentliche Adresse hat.
- Die **MAC_Source** gibt Auskunft über die Hardwareadresse des Absenders eines Paketes.
- Mit **MAC_Source_Inv** kann die MAC_Source invertiert werden.
- Die **Destination** gibt Auskunft über den Empfänger eines Paketes und ist ein Link zu einer Netzwerkkarte oder einem Netzwerk in der Topologie.
- Mit **Destination_Inv** kann die Destination invertiert werden.
- **Destination_Extern** gibt an, ob der Empfänger eines Paketes eine öffentliche Adresse hat.
- Das **Protocol** gibt an, welches Protokoll die Filterregel untersuchen soll.
- Das **In_Interface** ermöglicht es anzugeben, welches Interface für eintreffende Pakete zu überwachen ist und ist ein Link zu einer Netzwerkkarte des Firewallrechners.
- Mit **In_Interface_Inv** kann das In_Interface invertiert werden.
- Das Out**_Interface** ermöglicht es anzugeben, welches Interface für ausgehende Pakete zu überwachen ist und ist ein Link zu einer Netzwerkkarte des Firewallrechners.
- Mit **Out_Interface_Inv** kann das Out_Interface invertiert werden.
- Der **Source_Port** gibt Auskunft über den Quellport eines Paketes und ist ein Link zu einem Service.
- Mit **Source_Port_Inv** kann der Source_Port invertiert werden.
- Der **Destination_Port** gibt Auskunft über den Zielport eines Paketes und ist ein Link zu einem Service.
- Mit **Destination_Port_Inv** kann der Destination_Port invertiert werden.
- **To_Source** ermöglicht es, die Quelladresse eines Paketes zu verändern.
- **To_Source_Port** ermöglicht es, den Quellport eines Paketes zu verändern.
- **To_Destination** ermöglicht es, die Zieladresse eines Paketes zu verändern.
- **To_Destination_Port** ermöglicht es, den Zielport eines Paketes zu verändern.

- **Fragment** ermöglicht es, Regeln für fragmentierte Teile eines Paketes zu erstellen.
- **TCP_Flags** ermöglicht es, Regeln für Pakete einer TCP-Verbindung zu erstellen, bei der eine bestimmte Kombination der Flags gesetzt ist.
- Mit **TCP_Flags_Inv** kann TCP_Flags invertiert werden.
- **TCP_Option** ermöglicht es, bei TCP-Verbindungen das Option Feld auszuwerten.
- Mit **TCP_Option_Inv** kann TCP_Option invertiert werden.
- **State** ermöglicht es, die mit connection tracking gewonnenen Informationen für stateful inspection zu nutzen.
- **ICMP_Type** ermöglicht es, bei ICMP-Paketen die Felder Type und Code auszuwerten.
- Mit **ICMP_Type_Inv** kann ICMP_Type invertiert werden.
- **Limit** ermöglicht es, die Anzahl der Treffer einer Regel zu begrenzen.
- **Limit_Burst** siehe Limit.
- **Log_Level** ermöglicht es, den Kernel-Logging Modus einzustellen.
- **Log_Prefix** ermöglicht es, einen Text am Anfang der Logmeldung anzufügen.
- **Set_TOS** ermöglicht es, den Type of Service im IP Header zu setzen.
- Im Attribut **Target** wird angegeben, was passieren soll, wenn eine Regel zutrifft.
- Im Attribut **Policy** wird angegeben, was passieren soll, wenn keine Regel zutrifft.
- In **Description** kann ein beschreibender Text zu einer Filterregel abgelegt werden.

4.3 Implementation

Nachdem in der Analysephase die Struktur der Daten festgelegt wurde, wird in der Phase der Implementation das Modell um Verwaltungs- und Kontrollstrukturen erweitert.

4.3.1 Klassendiagramm der Konfigurationssoftware

In „Abbildung 9: Klassendiagramm der Konfigurationssoftware FirewallConfig“ ist die Schichtung der Klassen dargestellt. In der ListOfUsers werden Benutzer erfasst, die die Berechtigung zum Benutzen des Systems haben. Jeder User hat eine ListOfProjects, in der Daten zu den von ihm erstellten Projekten gespeichert sind. Jedes Projekt besteht aus der Topologie und dem Firewallsystem. Die Topologie ist eine Verwaltungsklasse,

die für die Integrität der Daten innerhalb der Topologie sorgt. Das Firewallsystem erhält die Daten der Topologie und gibt sie ihrerseits an die Firewallsoftware (hier IPTables) weiter. IPTables enthält die Logik zur Erstellung der Filterrule.

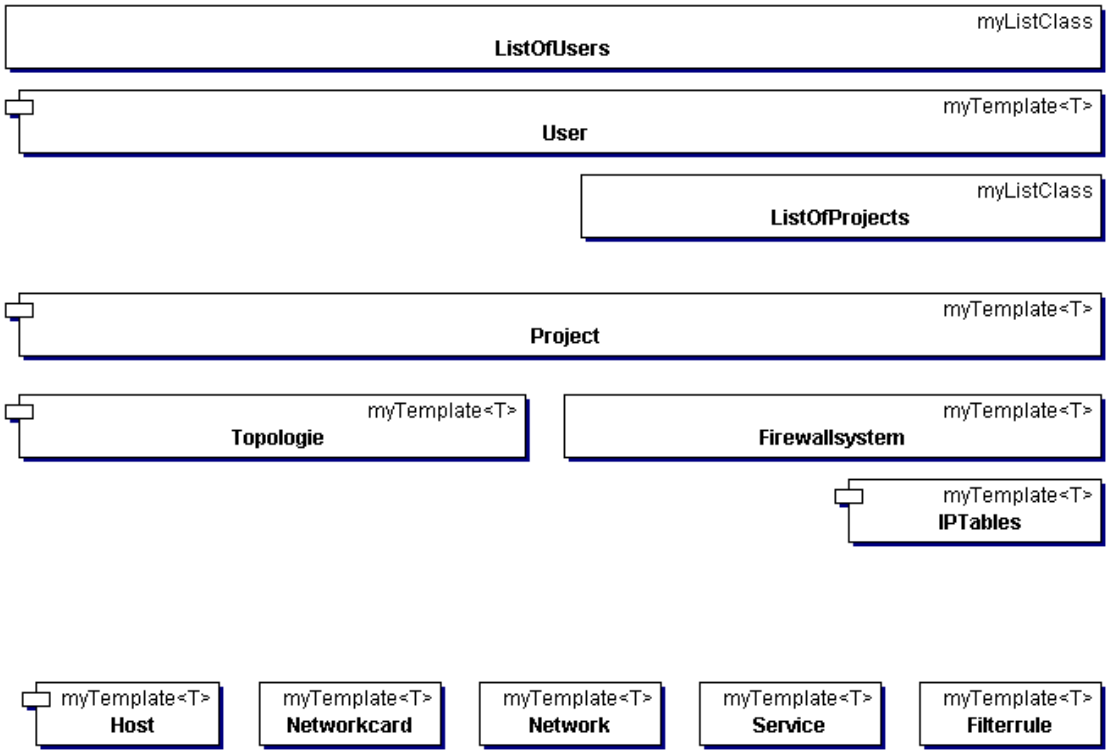

Abbildung 9: Klassendiagramm der Konfigurationssoftware FirewallConfig

4.3.2 Generierung von Tabellen, Logik und Datenhaltung

In „Abbildung 10: Schichtenmodell“ ist das Schichtenmodell dargestellt, das dem Konfigurationstool zugrunde liegt. Aus Anwendersicht übernimmt der view_stub die Anzeige der Daten. Die program_logic übernimmt die Verarbeitung der Daten. Der data_stub ist für die permanente Speicherung der Daten verantwortlich.

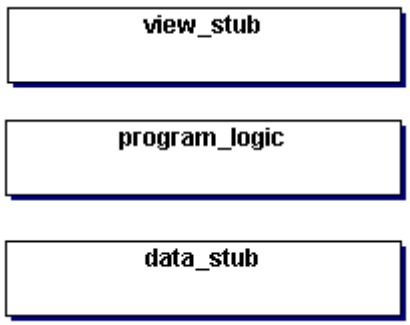

Abbildung 10: Schichtenmodell

5 Installation und Inbetriebnahme

In diesem Kapitel wird die Installation und Inbetriebnahme des Konfigurationstools „Firewallconfig“ auf einem Linuxsystem mit SuSE 7.1 beschrieben.

5.1 Hard- und Softwarevoraussetzungen

Entwickelt wurde das Konfigurationstool „FirewallConfig“ auf einem Rechner mit einem Pentium Prozessor mit 133MHz Taktfrequenz und 64MB Arbeitsspeicher. Als Betriebssystem wurde SuSE Linux 7.1 mit Kernel 2.4.16 eingesetzt. Der Webserver ist ein Apache in der Version 1.3.14-6 mit PHP4 als Modul in der Version 4.0.4pl1-2. MySQL ist in Version 3.23.33-16 installiert. Apache, MySQL und PHP4 sind rpm’s und stammen von der SuSE 7.1 Installations-CD.

5.2 Installation

Voraussetzung für die Installation des Konfigurationstools „FirewallConfig“ ist ein installierter und gestarteter Apache-Webserver sowie eine installierte und gestartete MySQL-Datenbank. In der Datei /etc/http/php.ini muss die Einstellung memory_limit=15M vorgenommen werden. In der Datei httpd.conf muss der Einsatz von .htaccess Dateien in Verzeichnissen erlaubt werden. Danach muss der Apache neu gestartet werden.

```
rcapache restart
```

Um das Konfigurationstool „FirewallConfig“ zu installieren, muss das Verzeichnis „FirewallConfig“ mit allen Dateien und Unterordnern in ein beliebiges Verzeichnis innerhalb des Document-Root des Webservers kopiert werden. Um die Zugriffsrechte der Scripte zu beschränken, sollten Besitzer und Gruppe sowie die Rechte geändert werden.

```
chown –R wwwrun:nogroup FirewallConfig
chmod –R 0500 Firewallconfig
chmod –R 0700 Firewallconfig/conf
```

In der Datei FirewallConfig/conf/dbconf.cfg.php werden die benötigten Daten für den Zugriff auf die MySQL Datenbank gespeichert. Die Variablen $db_server für die Adresse des Rechners, $db_user für den Usernamen und $db_pass für das Passwort müssen an eine bestehende Datenbankkonfiguration angepasst werden. Der User muss das Recht zur Erstellung einer Datenbank haben. Um die Installation abzuschließen, muss mit einem Webbrowser das Verzeichnis Firewallconfig/secure aufgerufen werden. Dadurch wird automatisch ein Script gestartet [Abbildung 11: Installationsscript], das die Datenbank „FirewallConfig“ und die Tabellen „users“, “currentsesion“, „projects“, „networks“, „hosts“, „networkcards“, „services“ sowie „filterrules“ erstellt. Das Verzeichnis FirewallConfig/secure sollte nach Möglichkeit durch eine Datei „.htpasswd“

vor fremdem Zugriff geschützt werden, da in diesem Verzeichnis die Benutzerverwaltung vorgenommen wird.

Mit der Management-Konsole werden die Benutzer verwaltet. Hier können User angemeldet und gelöscht werden [Abbildung 12: Benutzer anlegen]. Damit ist die Installation abgeschlossen.

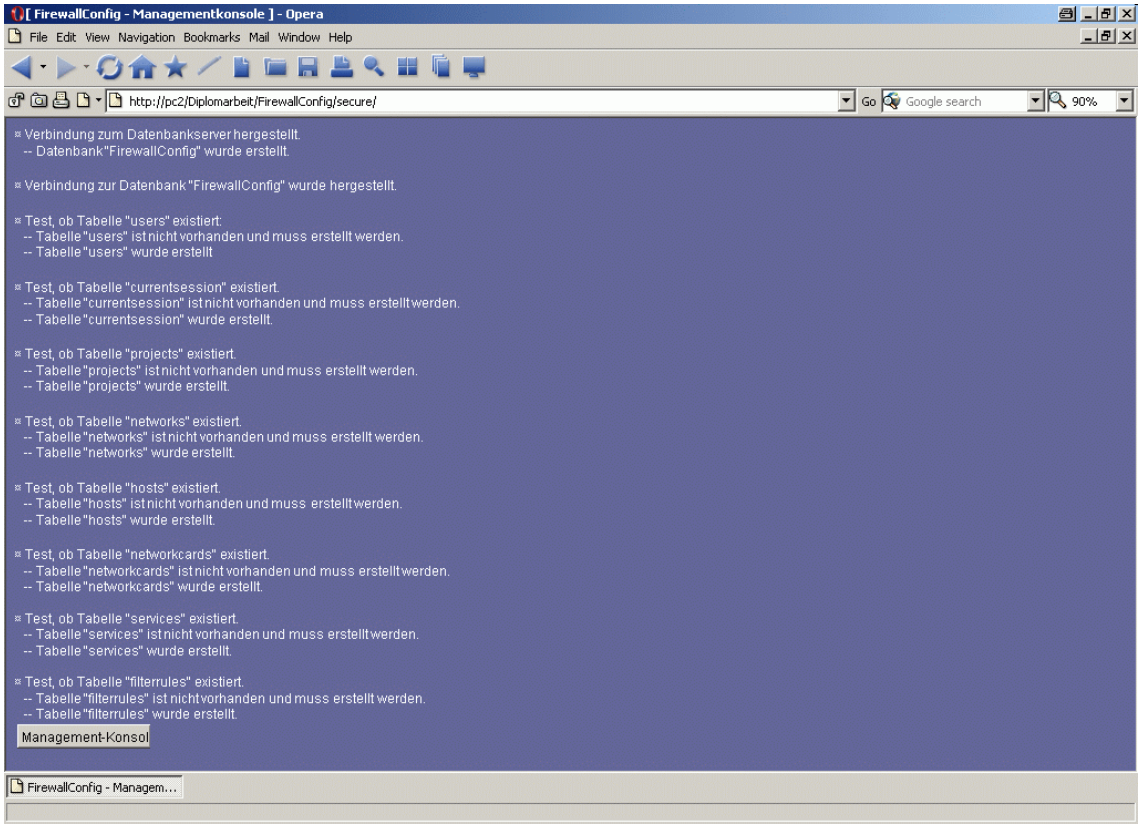

Abbildung 11: Installationsscript

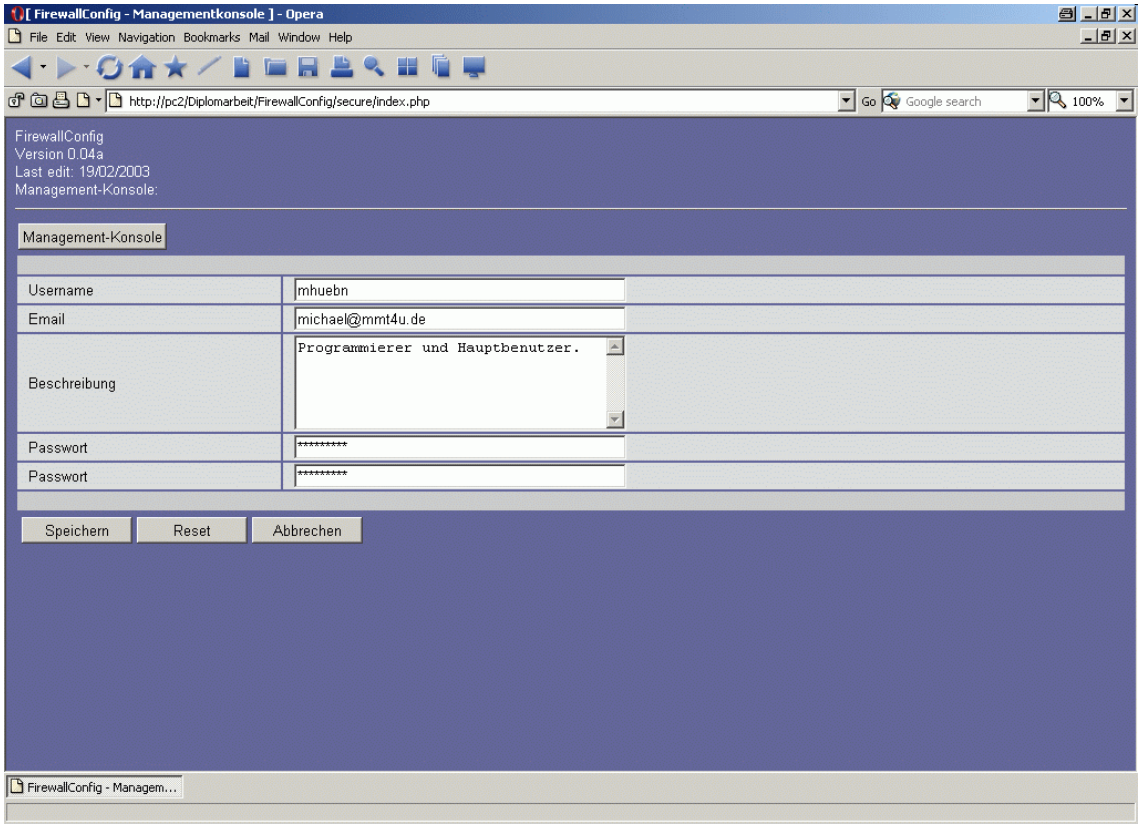

Abbildung 12: Benutzer anlegen

6 Handbuch

In diesem Kapitel wird die Benutzung des Konfigurationstools „FirewallConfig“ erläutert. Voraussetzung ist die korrekte Installation und ein gültiger Benutzername (siehe 5.2 Installation).

6.1 Start der Software

Um die Software zu starten, muss lediglich deren Internetadresse in einen Browser eingegeben werden. Als nächstes muss eine gültige Kombination aus Benutzername und Passwort eingegeben werden. Nach erfolgreicher Anmeldung kann ein Projekt erstellt, geladen, gelöscht, kopiert, exportiert oder importiert werden. Nach einer Unterbrechung von mehr als 30 Minuten muss man sich erneut anmelden.

6.2 Projektverwaltung

Die Projektverwaltung ermöglicht es dem Benutzer, Projekte zu erstellen, zu laden, zu löschen, zu kopieren, zu ändern, zu importieren oder zu exportieren [Abbildung 13: Projektverwaltung].

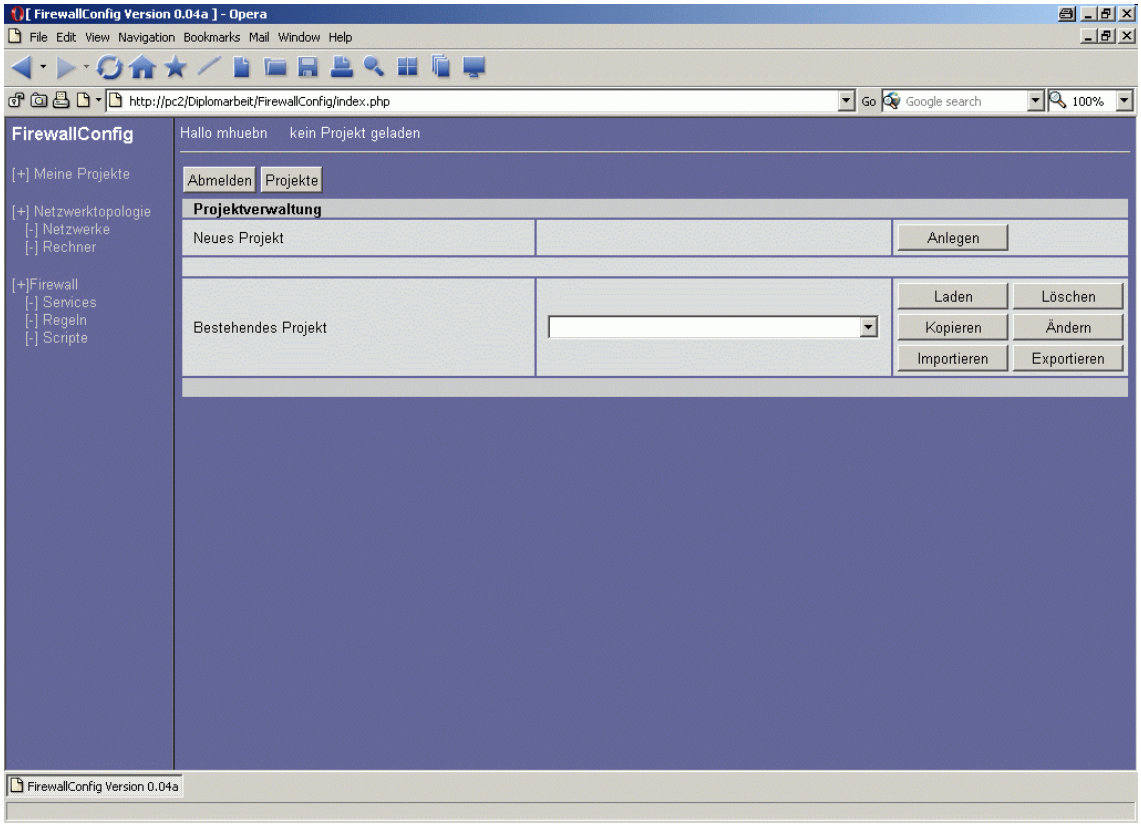

Abbildung 13: Projektverwaltung

6.2.1 Ein neues Projekt anlegen

Nach der Auswahl „Anlegen“ können der Name und die Beschreibung für ein neu zu erstellendes Projekt eingegeben werden [Abbildung 14: Projekt anlegen].

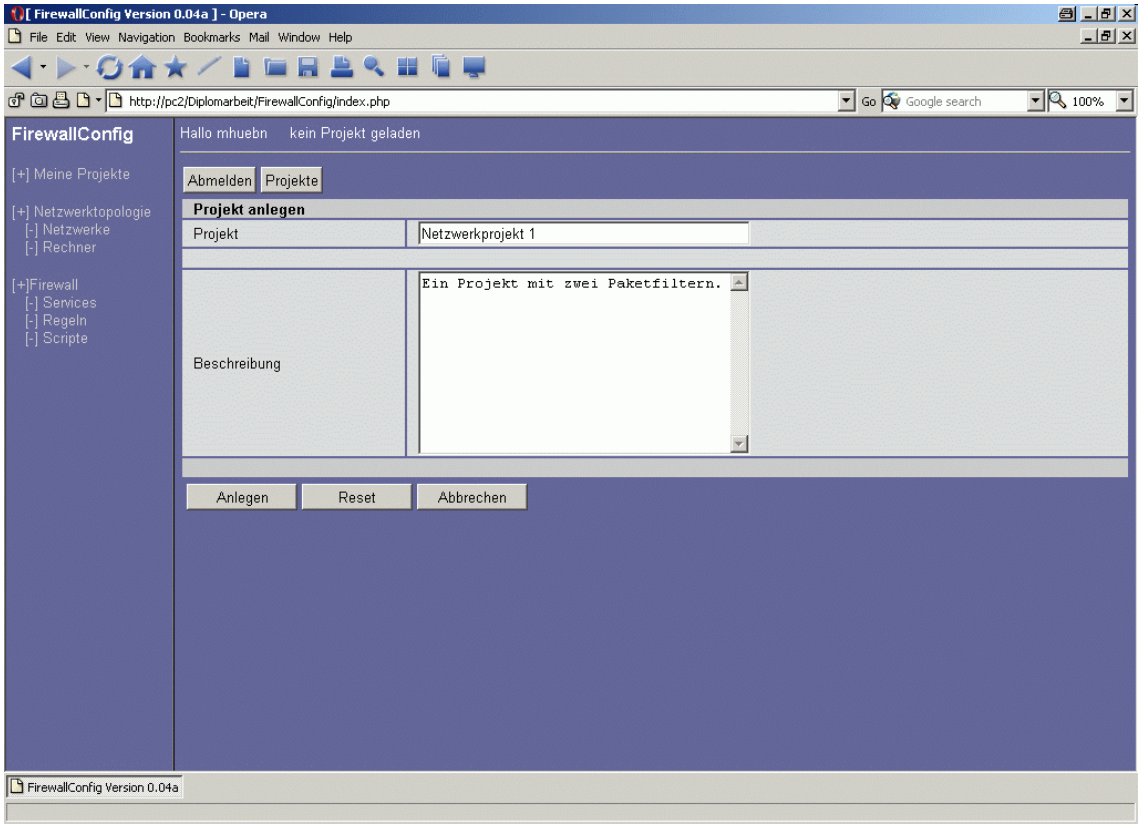

Abbildung 14: Projekt anlegen

6.2.2 Ein Projekt laden

Nach der Auswahl „Laden“ kann ein Projekt geladen werden.

6.2.3 Ein Projekt ändern

Nach der Auswahl „Ändern“ können der Name und die Beschreibung des Projektes geändert werden.

6.2.4 Ein Projekt importieren

Nach der Auswahl „Importieren“ kann ein exportiertes Projekt importiert werden [Abbildung 15: Projekt importieren].

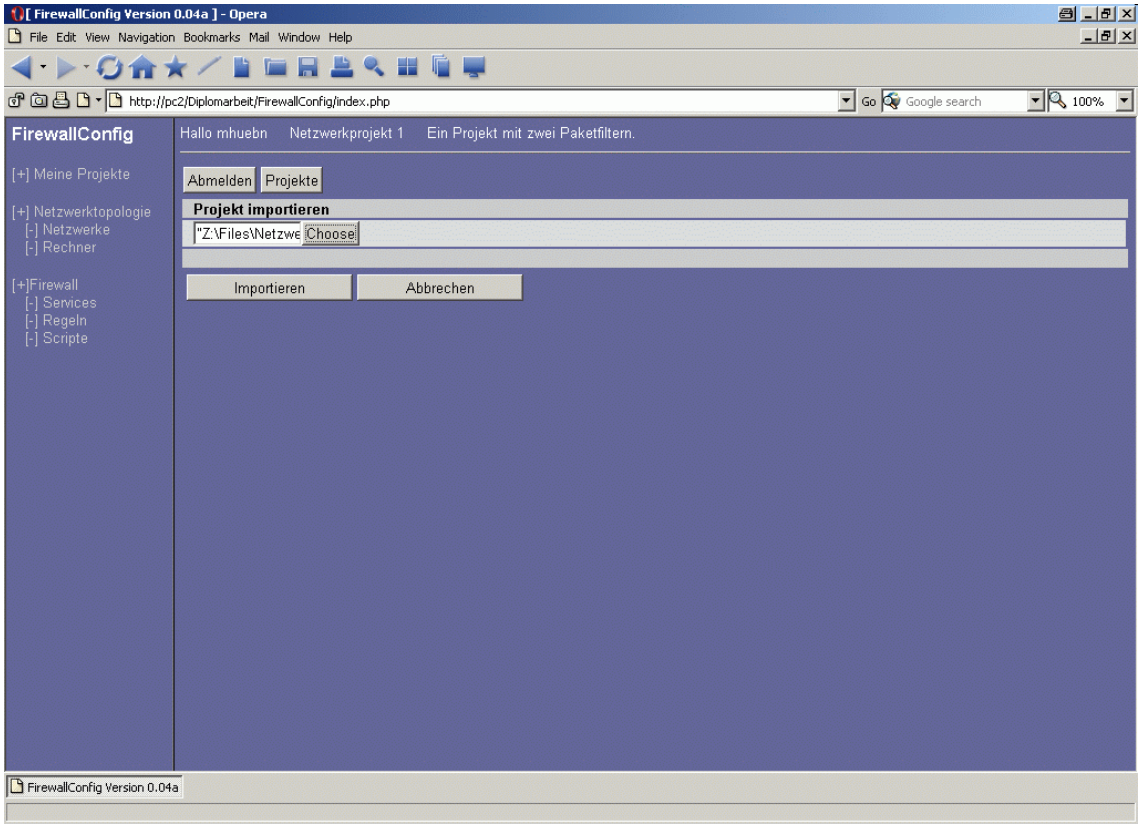

Abbildung 15: Projekt importieren

6.2.5 Ein Projekt exportieren

Um ein Projekt zu exportieren, das Projekt selektieren und dann „Exportieren“ wählen. Das exportierte Projekt wird als Textfile im Unix-Standard an den Browser gesendet und kann lokal gespeichert werden [Abbildung 16: Projekt exportieren].

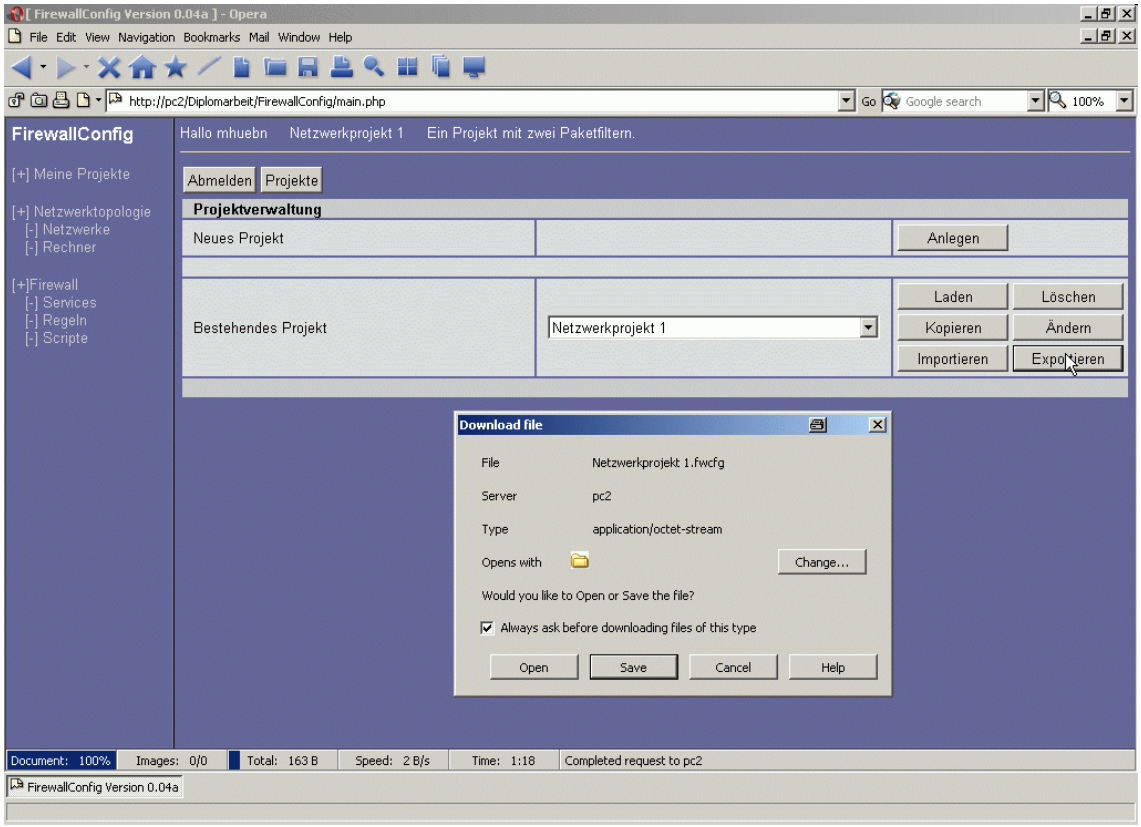

Abbildung 16: Projekt exportieren

6.2.6 Ein Projekt kopieren

Um ein Projekt zu kopieren, das Projekt selektieren und dann „Kopieren“ auswählen. Es wird eine Kopie des Projektes mit dem Namen „Kopie_von_*Originalname*“ erzeugt. Der Name und die Beschreibung können unter „Ändern“ weiter bearbeitet werden.

6.2.7 Ein Projekt löschen

Um ein Projekt zu löschen. das Projekt unter „Bestehendes Projekt“ auswählen und dann mit „Löschen“ löschen.

6.3 Netzwerktopologie

Ist ein Projekt geladen, kann die Netzwerktopologie aufgebaut werden. Dazu werden alle vorhandenen Rechner und/oder Netzwerke erfasst.

6.3.1 Netzwerke

Netzwerke sind definiert durch ihren Adressbereich (Netzadresse) und die Netzmaske. Jedes Interface/Netzwerkkarte, das über eine IP Adresse und eine Netzmaske verfügt, wird automatisch Teil eines Netzwerkes.

6.3.1.1 Netzwerk erstellen

Unter „Netzwerktopologie/Netzwerke/neues Netzwerk“ kann ein neues Netzwerk erstellt werden [Abbildung 17: Netzwerk erstellen].

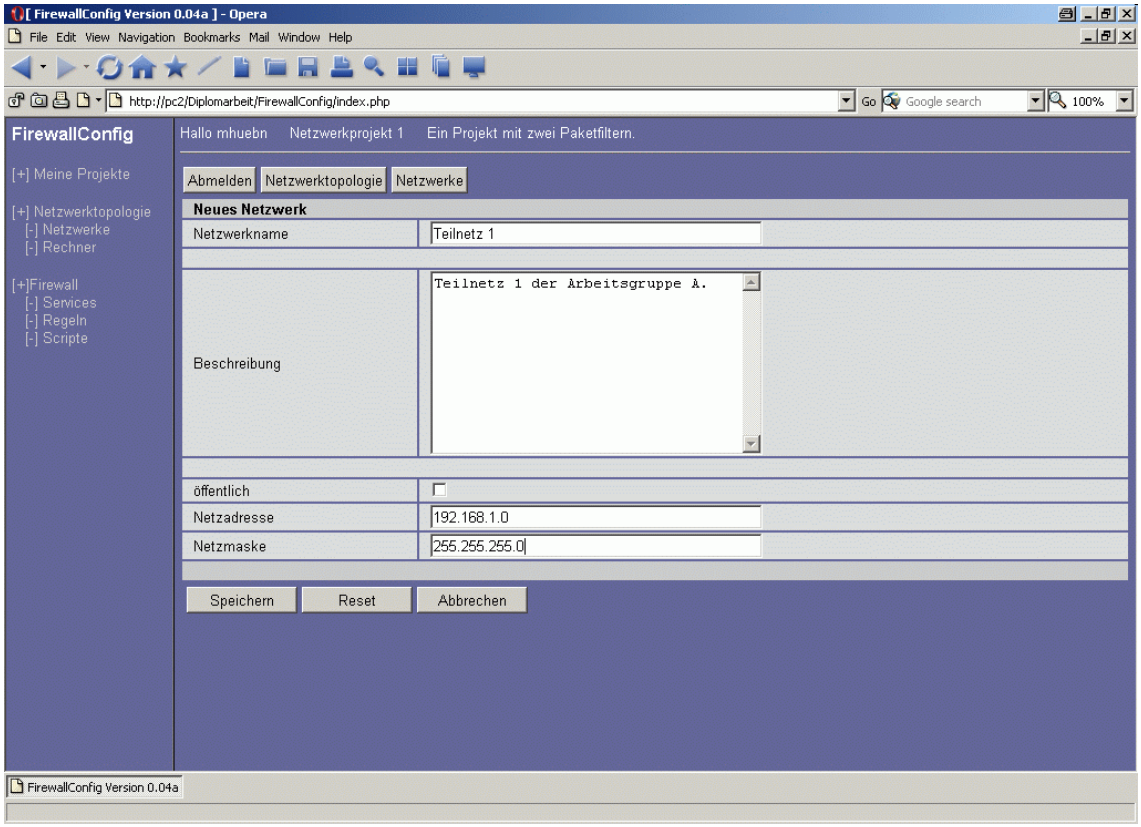

Abbildung 17: Netzwerk erstellen

6.3.1.2 Netzwerk ändern

Um ein Netzwerk zu ändern, muss es unter „Netzwerktopologie/Netzwerke“ mit „Ändern“ ausgewählt werden. Es können nur der Name und die Beschreibung geändert werden.

6.3.1.3 Netzwerk löschen

Um ein Netzwerk zu löschen, unter „Netzwerktopologie/Netzwerke“ „Löschen“ für das betreffende Netzwerk auswählen.

Hinweis: Sind noch Clients oder Netzwerkkarten von Firewallrechnern mit diesem Netzwerk verbunden, so kann es nicht gelöscht werden.

6.3.2 Rechner

In der Netzwerktopologie wird zwischen Firewallrechnern und Clients unterschieden. Firewallrechner sind Teil des Firewallsystems und besitzen im Allgemeinen mehrere Netzwerkkarten. Clients sind Arbeitsplatzrechner, dedizierte Server oder ähnliche

Rechner, die mit der Regelung des Datentransfers im Netzwerk nichts zu tun haben. Sie haben lediglich eine Netzwerkkarte.

6.3.2.1 Firewallrechner erfassen

Unter „Netzwerktopologie/Rechner/neuer Rechner“ kann ein neuer Firewallrechner angelegt werden. Dazu muss als Rechnertyp „Firewallrechner“ ausgewählt werden.

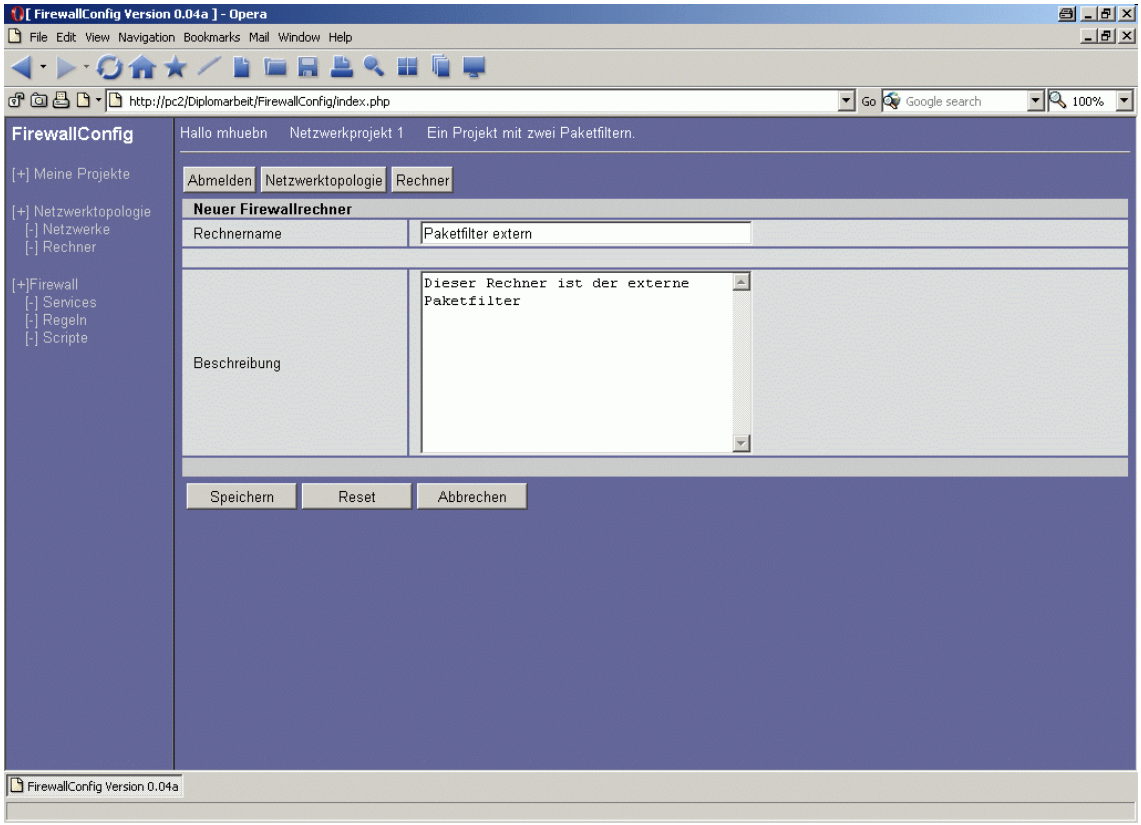

Abbildung 18: Firewallrechner erstellen

6.3.2.2 Firewallrechner ändern

Um einen Firewallrechner zu ändern, wird dieser in der Tabelle „Netzwerktopologie/Rechner“ ausgewählt.

6.3.2.2.1 Interface/Netzwerkkarte erstellen

Um ein Interface zu erstellen, den betreffenden Rechner unter „Netzwerktopologie/Rechner/Ändern“ auswählen. Mit dem Button „Interfaces“ kann zur Interfaceverwaltung gewechselt werden. Mit dem Button „neues Interface“ wird die Eingabemaske für ein neues Interface aufgerufen [Abbildung 19: Interface erstellen]. Bei Alias sollte ein Name eingegeben werden, der das Interface beschreibt. Das Feld Device akzeptiert die Eingabe von eth[0-9] für ein Ethernet-Device, ipp[0-9] für ein ISDN-Device, ppp[0-9] für ein DSL-Device sowie lo für das Loopback-Device. Die Checkbox „öffentliche Adresse“ muss aktiviert werden, wenn das Device eine offiziell zugewiesene IP Adresse hat. Unter IP Adresse kann eine IP Adresse eingetragen werden. Sie muss das Format

148.16.7.8 haben. Unter Netzmaske kann die Subnetz Maske eingetragen werden. Sie muss das Format 255.255.0.0 haben. Unter HWaddr kann die Hardwareadresse eines Ethernet-Device eingetragen werden. Sie muss das Format 11:44:aa:ff:0f:33 haben. Werden eine IP Adresse und eine Netzmaske eingetragen, so wird die Netzwerkkarte automatisch dem zugehörigen Netzwerk zugeordnet. Existiert das Netzwerk noch nicht, so wird es erzeugt.

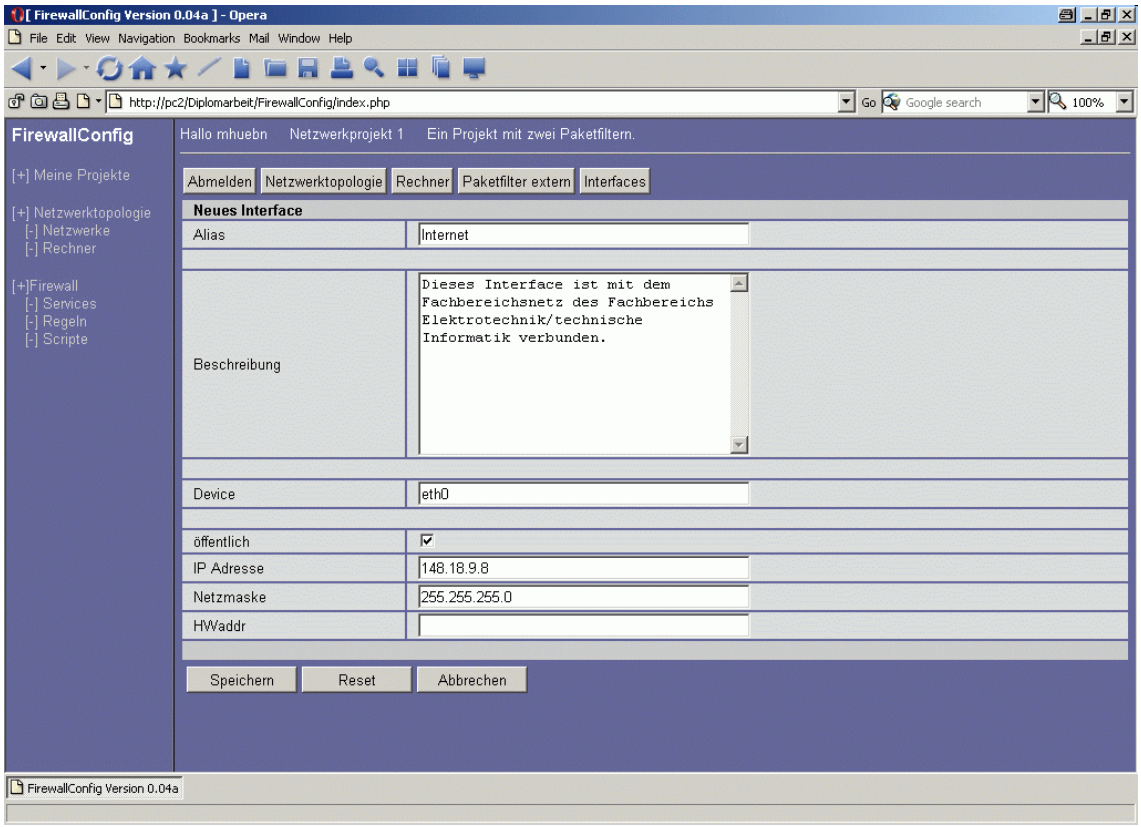

Abbildung 19: Interface erstellen

6.3.2.2.2 Interface/Netzwerkkarte ändern

Um ein Interface eines Firewallrechners zu ändern, muss es aus der Liste seiner Interfaces ausgewählt werden. Danach können die Einstellungen geändert werden.

6.3.2.2.3 Interface/Netzwerkkarte löschen

Um ein Interface eines Firewallrechners zu löschen, muss es aus der Liste seiner Interfaces ausgewählt werden. Danach kann es gelöscht werden. Zu beachten ist, dass ein eventuell von diesem Interface erzeugtes Netzwerk nicht automatisch gelöscht wird.

6.3.2.3 Firewallrechner löschen

Um einen Firewallrechner aus der Topologie zu löschen, muss er unter „Topologie/Rechner“ mit „Löschen“ gelöscht werden. Gleichzeitig werden alle seine Interfaces gelöscht.

6.3.2.4 Client erstellen

Unter „Netzwerktopologie/Rechner/neuer Rechner“ kann ein neuer Client angelegt werden. Dazu muss als Rechnertyp „Client“ ausgewählt werden [Abbildung 20: Client erstellen]. Die Checkbox öffentliche Adresse muss aktiviert werden, wenn das Device eine offiziell zugewiesene IP Adresse hat. Unter IP Adresse muss eine IP Adresse eingetragen werden. Sie muss das Format 148.16.7.8 haben. Unter Netzmaske kann die Subnetz Maske eingetragen werden. Sie muss das Format 255.255.0.0 haben. Unter HWaddr kann die Hardwareadresse eines Ethernet-Device eingetragen werden. Sie muss das Format 11:44:aa:ff:0f:33 haben. Werden eine IP Adresse und eine Netzmaske eingetragen, so wird der Client automatisch dem zugehörigen Netzwerk zugeordnet. Existiert das Netzwerk noch nicht, so wird es erzeugt.

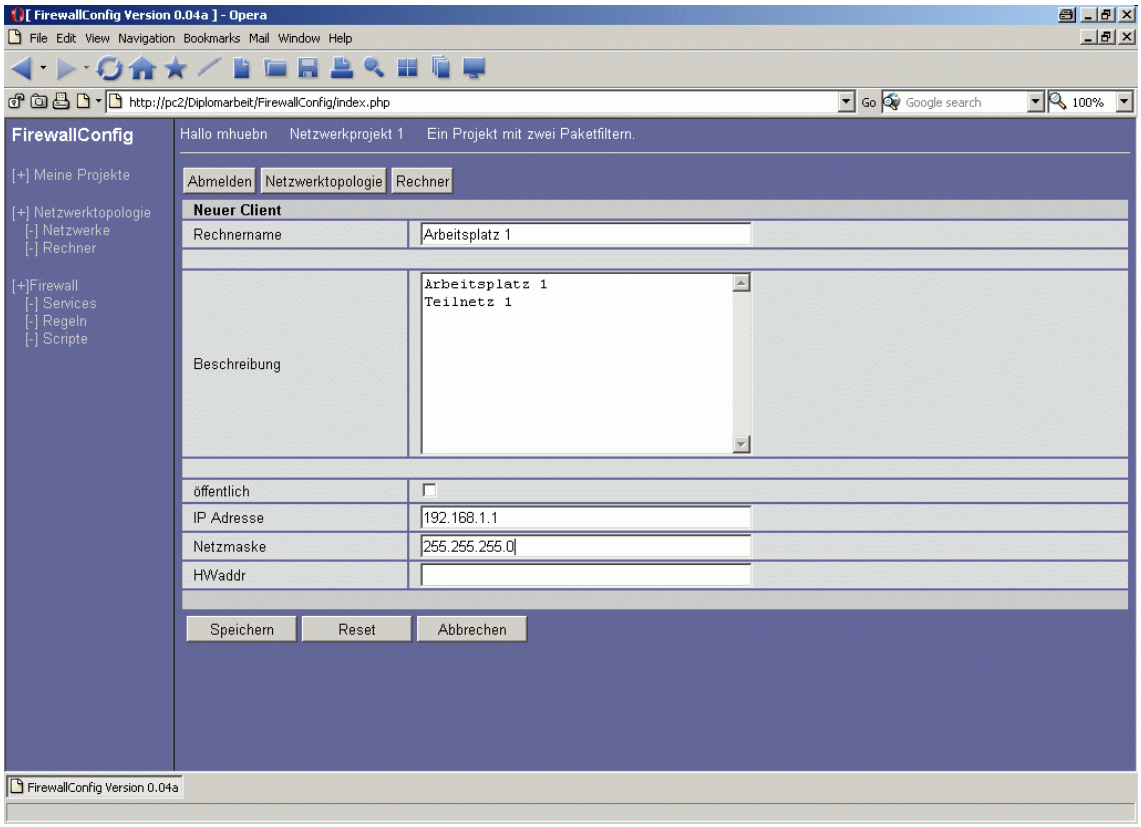

Abbildung 20: Client erstellen

6.3.2.5 Client ändern

Um einen Client zu ändern, muss er unter „Netzwerktopologie/Rechner“ mit „Ändern“ ausgewählt werden.

6.3.2.6 Client löschen

Um einen Client zu löschen, muss er unter „Topologie/Rechner“ mit „Löschen“ gelöscht werden. Eventuell vom Client erstellte Netzwerke werden nicht gelöscht.

6.4 Firewallsystem

Das Firewallsystem beinhaltet alle zur Erstellung von Filterregeln notwendigen Angaben. Es greift dabei auch auf die in der Topologie angelegten Rechner und Netzwerke zurück. Änderungen, die in der Netzwerktopologie vorgenommen werden, verändern automatisch die Filterregeln, so dass die Integrität des Firewallsystems gewährleistet bleibt.

6.4.1 Services

Unter Services werden Dienste, Protokolle, Ports und Portgruppen verstanden. Dabei müssen die Attribute nicht eindeutig sein. Es ist z.B. möglich, für Port 80 tcp zwei verschiedene Services anzulegen. Einen unter dem Keyword „HTTP Anfragen“ und einen unter „Umleitung Webserver“ [Abbildung 21: Services].

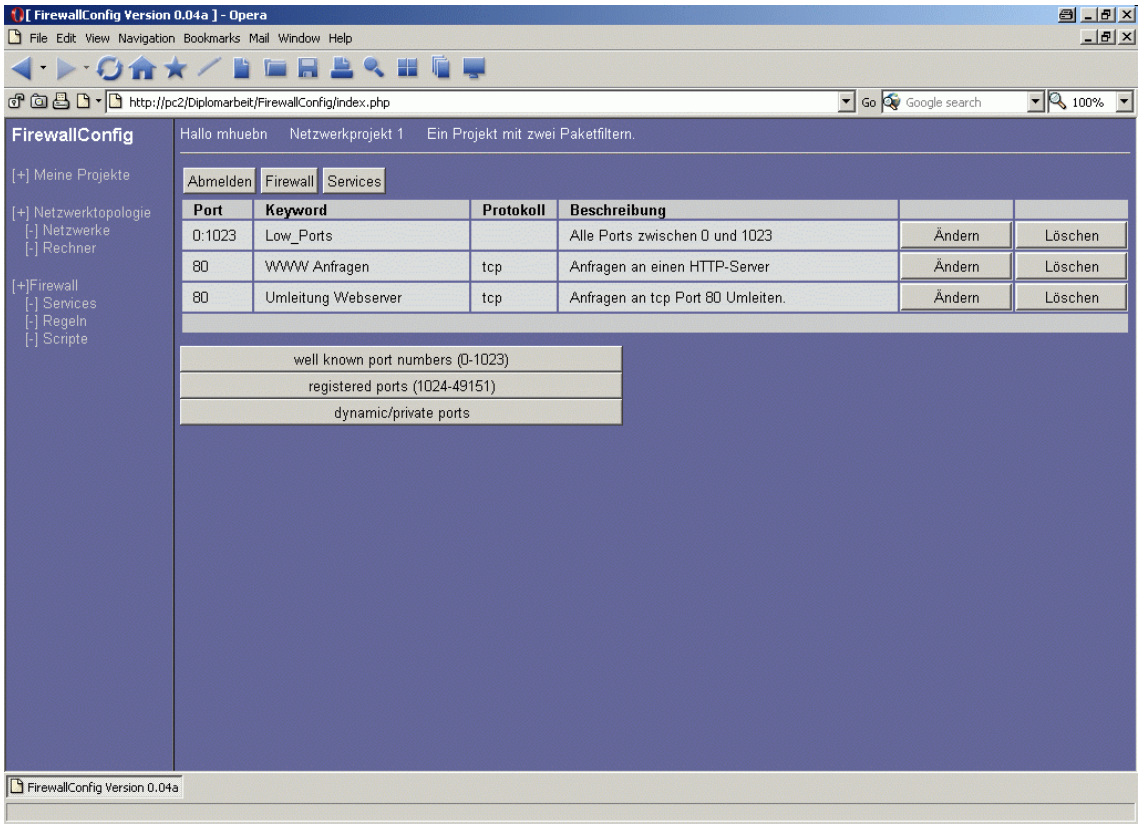

Abbildung 21: Services

6.4.1.1 Service anlegen

Ein Service kann auf zwei verschiedene Arten angelegt werden.

1. Einen oder mehrere Services aus den Listen „well known port numbers“ oder „registered ports“ auswählen. In diesen Listen sind alle definierten Ports enthalten (Stand 20/11/2002) [Abbildung 22: Services - well known port numbers]. Da die Listen sehr umfangreich sind, ist je nach Internetanbindung mit erhöhten Ladezeiten zu rechnen.

2. Unter „dynamic/private ports“ einen Service definieren [Abbildung 23: Services - selbstdefinierte Services]. Das Protokoll kann aus dem Drop Down Menü ausgewählt werden. Für den Port kann entweder ein einzelner Port oder ein Bereich in der Form 0:1024 angegeben werden.

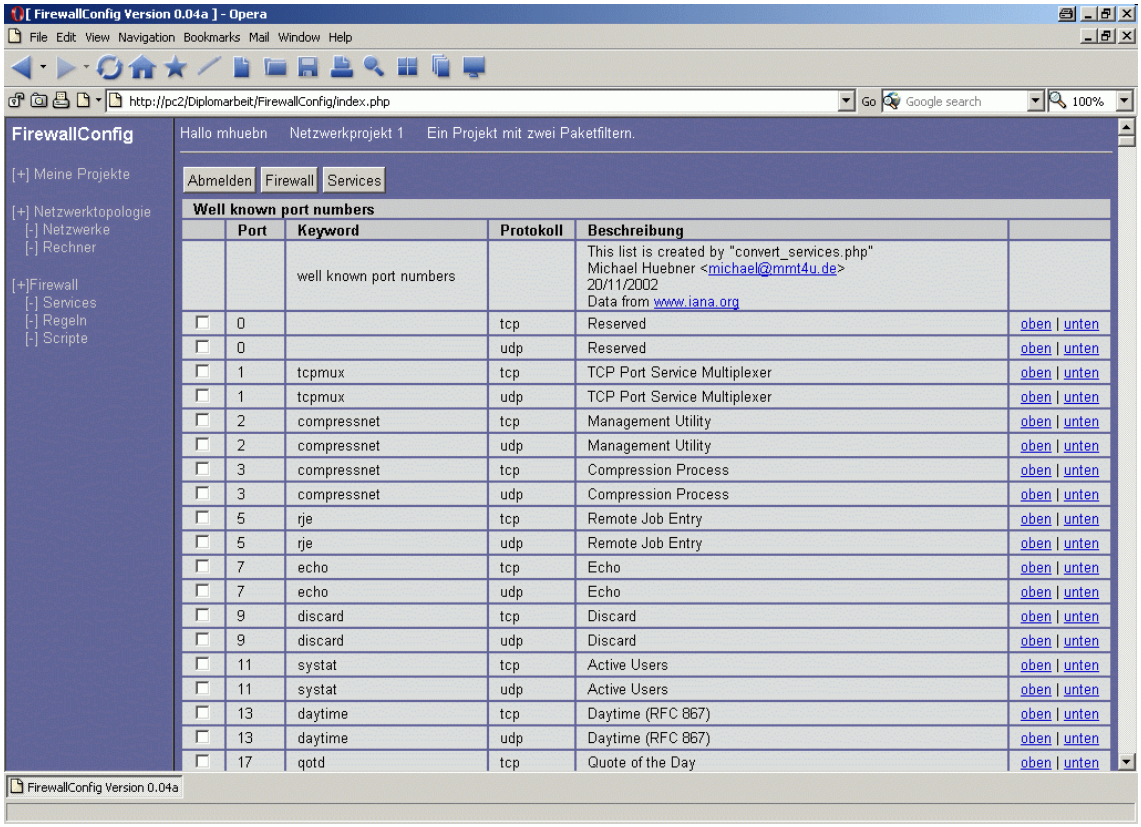

Abbildung 22: Services - well known port numbers

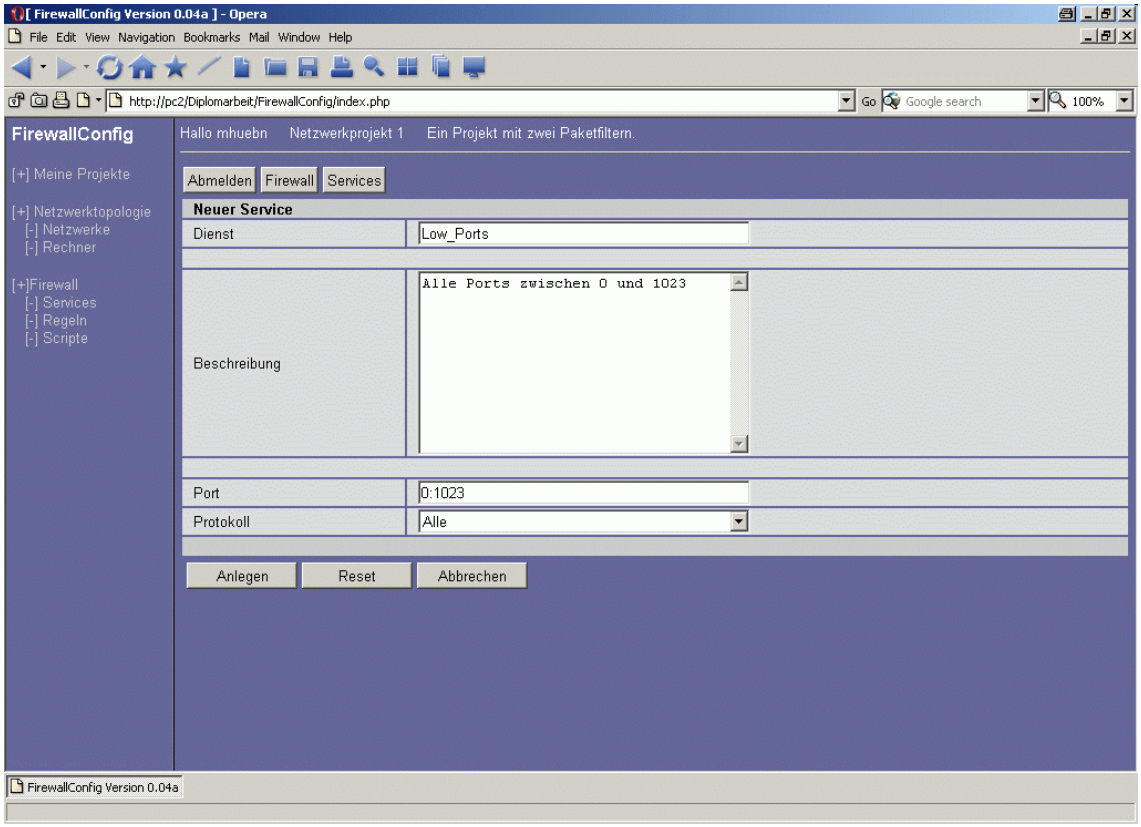

Abbildung 23: Services - selbstdefinierte Services

6.4.1.2 Service ändern

Jeder Service, auch ein durch die Listen „well known port numbers“ oder „registered ports“ angelegter, kann durch Auswahl von „Firewall/Services/Ändern“ verändert werden. Dabei bleiben eventuelle Verknüpfungen zu Filterregeln erhalten.

6.4.1.3 Service löschen

Ein Service kann unter „Firewall/Services/Löschen“ wieder gelöscht werden.

6.4.2 Regeln

Sind in der Netzwerktopologie Firewallrechner angelegt worden, so können für diese unter „Firewall/Regeln/Filter Regeln“ Filter- oder NAT-Regeln Regeln angelegt werden. Filterregeln geben an, welches Paket einen Firewallrechner passieren darf. NAT-Regeln können die Herkunft oder das Ziel eines Paketes verändern. Standard Filterregeln sind vordefinierte Regeln, die unabhängig von der Topologie oder definierten Services wirken. Filter-Regeln und NAT-Regeln werden mit den Daten erstellt, die unter „Rechner“, „Netzwerke“ und „Services“ eingegeben wurden. Werden diese Daten geändert, z.B. die IP Adresse eines Clients, so werden automatisch alle Regeln, die sich auf diesen Client beziehen, geändert. Ebenso werden, falls ein Client gelöscht wird, alle Regeln, die sich auf diesen Client beziehen, gelöscht.

6.4.2.1 Standard Filterregeln erstellen

Unter „Firewall/Regeln/Filter Regeln/Standard Filterregeln“ können Standard Filterregeln erstellt werden. Möglich sind zurzeit folgende Regeln [Abbildung 24: Regeln - Standard Filterregeln]:

1. Die **Policy** gibt an, was mit einem Paket passieren soll, falls keine Regel des Filters zutrifft. Mögliche Ziele sind **ACCEPT** und **DROP**. Wird keine Policy ausgewählt, so gilt die Policy DROP.
2. **Fragmente** sind Teile eines Paketes, das durch ein fragmentation request aufgeteilt wurde. Da in einem Fragment die Protokoll-Header-Informationen fehlen, kann keine Regel, die auf Header-Informationen basiert, auf ein Fragment zutreffen. Im Allgemeinen wird angenommen, das Header sicher sind. Zusätzlich bewirkt diese Option, das sogenannte *short fragments*, d.h. Fragmente, die kürzer als 8Byte sind, automatisch verworfen werden. Mögliche Ziele sind **ACCEPT** und **DROP**.
3. **SMTP** bezieht sich auf Port 25, Protokoll tcp für alle Interfaces. Diese Regel wird für den Betrieb eines Email-Servers benötigt. Mögliche Ziele sind **ACCEPT** und **DROP**.
4. **HTTP** bezieht sich auf Port 80, Protokoll tcp für alle Interfaces. Diese Regel wird für den Betrieb eines HTTP-Servers benötigt. Mögliche Ziele sind **ACCEPT** und **DROP**.
5. **SSH** bezieht sich auf Port 22, Protokoll tcp für alle Interfaces. Diese Regel wird für den Zugang zu diesem Rechner via Secure Shell benötigt. Mögliche Ziele sind **ACCEPT** und **DROP**.
6. **Telnet** bezieht sich auf Port 23, Protokoll tcp für alle Interfaces. Diese Regel wird für den Zugang zu diesem Rechner via Telnet benötigt. Mögliche Ziele sind **ACCEPT** und **DROP**. Es wird dringend empfohlen, statt Telnet SSH zu verwenden.
7. **FTP** bezieht sich auf Port 20 und 21, Protokoll tcp für alle Interfaces. Diese Regel wird für den Betrieb eines FTP-Servers benötigt. Mögliche Ziele sind **ACCEPT** und **DROP**.
8. **ICMP 'unreachable' messages** ermöglicht den Empfang der Nachricht, dass ein Ziel nicht erreichbar ist. Mögliche Ziele sind **ACCEPT** und **DROP**.
9. Die Regel für das **Loopback** Interface erlaubt oder verbietet alle Pakete von und zu diesem Interface. Mögliche Ziele sind **ACCEPT** und **DROP**.
10. Die Regel für Pakete, die von diesem Rechner stammen, erlaubt oder verbietet alle lokal erzeugten Pakete. Mögliche Ziele sind **ACCEPT** und **DROP**.

11. **Catch and log** zeichnet alle Pakete auf, auf die bis zu dieser Regel keine andere Regel gepasst hat. Diese Regel wird vorzugsweise die letzte Regel sein, bevor die Policy das Paket übernimmt.

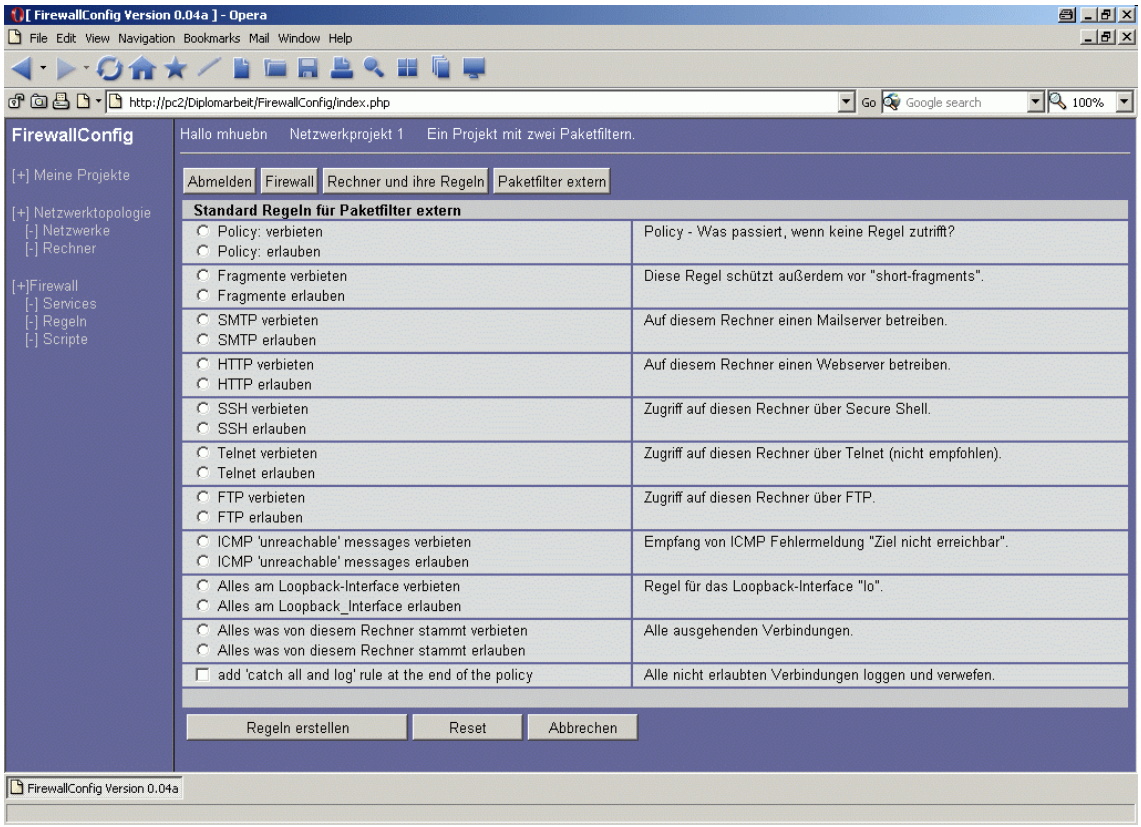

Abbildung 24: Regeln - Standard Filterregeln

6.4.2.2 Eine Filterregel erstellen

Unter „Firewall/Regeln/Filter Regeln/neue Filterregel“ können Filterregeln erstellt werden [Abbildung 25: Filterregel erstellen]. Einige Attribute einer Regel können durch makieren des Feldes [☑ !] invertiert werden.

Die **Verbindung** ordnet die Regel intern zu:

- **Eingehende Verbindung:** Es kann ein In_Interface und ein eine Source angegeben werden. Das In_Interface ist das Interface, an dem die Pakete eintreffen. Die **Source** ist der Absender der Pakete.
- **Weitergehende Verbindungen:** Es kann ein In_Interface, ein Out_Interface, eine Source und eine Destination angegeben werden. Das In_Interface ist das Interface, an dem die Pakete eintreffen. Das Out_Interface ist das Interface, an dem die Pakete den Rechner wieder verlassen. Die Source ist der Absender der Pakete. Die Destination ist der Empfänger der Pakete.

- **Ausgehende Verbindungen:** Es kann ein Out_Interface und eine Destination angegeben werden. Das Out_Interface ist das Interface, an dem die Pakete den Rechner verlassen. Die Destination ist der Empfänger der Pakete.

Das **Protokoll** kann ein zu filterndes Paket näher bestimmen.

- **tcp:** Es kann ein Source-Port und ein Destination-Port angegeben werden. Es stehen die Ports zur Verfügung, die unter „Services“ mit dem Protokol „tcp“ angelegt wurden.
- **udp:** Es kann ein Source-Port und ein Destination-Port angegeben werden. Es stehen die Ports zur Verfügung, die unter „Services“ mit dem Protokol „udp“ angelegt wurden.
- **icmp:** Es kann ein icmp Type ausgewählt werden.

Die **Art der Verbindung** schränkt die Auswahl der Pakete weiter ein.

- **Verbindungsaufbau:** Das sind Pakete, die eine neue Verbindung aufbauen.
- **Bestehende Verbindungen:** Das sind Pakete, die zu einer bereits bestehenden Verbindung gehören.
- **Hat etwas mit anderen Verbindungen zu tun:** Das sind Pakete, die zu einer anderen Verbindung gehören (ftp-Kontrollverbindung, echo reply nach einem echo request).
- **Ungültiger Zustand:** Das sind Pakete, die keines der Kriterien 1-3 erfüllen.

Was ist zu tun? An dieser Stelle wird das Ziel des Paketes bestimmt. Es gibt zurzeit drei Möglichkeiten.

1. **Annehmen:** Das Paket wird akzeptiert und an seinen Bestimmungsort weitergeleitet.
2. **Ablehnen:** Das Paket wird verworfen.
3. **Loggen:** Es wird ein Eintrag in das Log-File gemacht und das Paket wird zur weiteren Untersuchung an die nächste Regel weitergeleitet. Unter „Setup“ kann eingestellt werden, wie geloggt werden soll. Dabei bedeuten:
    - **Log-Präfix:** Ein Text, der im Logfile erscheint. Der Text darf bis zu 29 Zeichen lang sein und aus Groß-/Kleinbuchstaben, Ziffern sowie den Zeichen ‚Unterstrich’, ‚Punkt’ und ‚Leerzeichen’ bestehen.
    - **Log-Level:** Die Priorität der Meldung.

Die nächste Einstellung kann eine **Regel limitieren.** Limit bedeutet, dass eine Regel nur für eine bestimmte Anzahl pro Zeiteinheit angewendet wird. Das ist z.B. bei einem LOG sinnvoll, um das Logfile nicht zu überfluten. Mit „Setup“ kann das Limit eingestellt werden.

- Die **Anzahl** gibt an, in welchem Intervall der Zähler zurückgesetzt wird.

- Die **Obergrenze** gibt an, wie viele Treffer erreicht werden dürfen, bevor das Limit wirksam wird. Die Standardobergrenze ist 5.
- Die **Zeiteinheit** gibt an, in welcher Größenordnung der Zähler zurückgesetzt wird.

Ein Beispiel soll dies verdeutlichen:

Anzahl = 20; Obergrenze = 5; Zeiteinheit = pro Stunde;

Die Obergrenze ist 5. Daher trifft diese Regel, unabhängig von Anzahl und Zeiteinheit, fünfmal zu. Das bedeutet, dass ein interner Zähler bis zum Zählerstand fünf erhöht werden kann, bevor das Limit erreicht ist. Anzahl und Zeiteinheit bewirken, dass der Zählerstand um zwanzig pro Stunde dekrementiert wird, also alle drei Minuten um eins. Trifft kein Paket mehr ein, so könnte die Regel nach drei Minuten einmal angewendet werden, nach sechs Minuten zweimal, … und nach fünfzehn Minuten fünfmal. Dann währe der Zähler wieder bei Null und die Regel könnte bis zur Obergrenze von fünf angewendet werden.

Das Auswahlfeld **Fertig** erscheint, nachdem alle notwendigen Eingaben gemacht wurden. Durch die Auswahl von Fertig wird die Regel noch einmal überprüft und ein Button zum Speichern angezeigt.

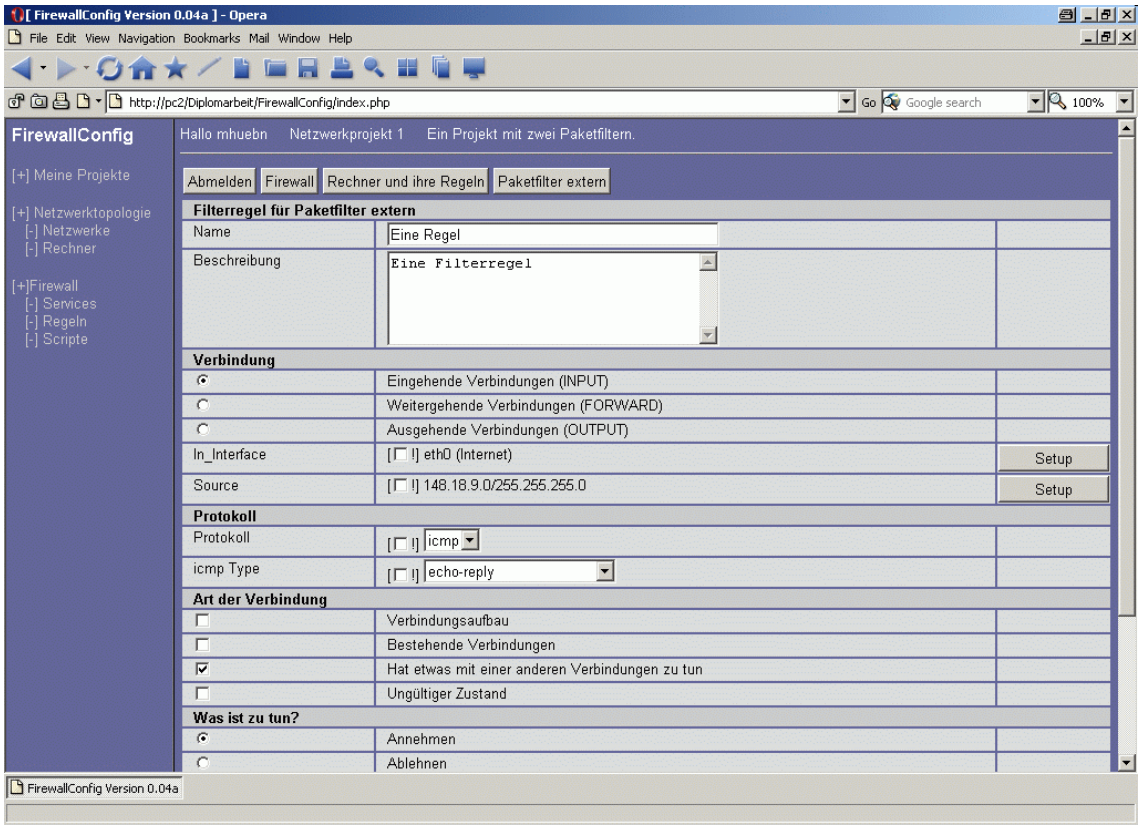

Abbildung 25: Filterregel erstellen

6.4.2.3 Eine Filterregel ändern

Unter „Firewall/Regeln/Filter Regeln“ werden die bereits erstellten Filterregeln angezeigt. Selbsterstellte Filterregeln können hier geändert werden. Das Ändern einer Filterregel geschieht analog zum Erstellen einer Filterregel. Eine geänderte Filterregel kann entweder als Änderung oder als neue Regel gespeichert werden.

6.4.2.4 Die Position einer Filterregel ändern.

Unter „Firewall/Regeln/Filter Regeln“ werden die bereits erstellten Filterregeln angezeigt. Die Position einer Filterregel (Nummer) kann hier geändert werden. Zum Ändern der Position den Button mit der Nummer anklicken und anschließend eine neue Nummer eingeben.

6.4.2.5 Eine Filterregel löschen

Unter „Firewall/Regeln/Filter Regeln“ werden die bereits erstellten Filterregeln angezeigt. Die Filterregeln können hier gelöscht werden.

6.4.2.6 Eine NAT Regel erstellen

Unter „Firewall/Regeln/Filter Regeln“ können NAT Regeln erstellt werden [Abbildung 26: NAT Regel erstellen]. Einige Attribute einer Regel können durch markieren des Feldes  invertiert werden.

Was wollen sie ändern?

- **Die Quelladresse (feste IP):** Es wird die Quelladresse eines Paketes verändert (SNAT). Diese Option wird gewählt, wenn die Adresse eine feste Adresse ist.
- **Die Quelladresse (dynamische IP):** Es wird die Quelladresse eines Paketes verändert (MASQUERADE). Diese Otion wird gewählt, wenn die Adresse unbekannt ist (dial up Verbindung).
- **Die Zieladresse:** Es wird die Zieladresse eines Paketes geändert (das Paket wird umgeleitet).

Welches Interface?

- Abhängig davon, was geändert werden soll, kann hier ein Out_Interface, das spezifiziert, wo die Pakete den Rechner verlassen, oder ein In_Interface, das spezifiziert, wo die Pakete beim Rechner eintreffen, ausgewählt werden.

Auswahl der betreffenden Pakete

- Es kann ein Filter gesetzt werden, auf welche Pakete die Änderung zutrifft. Nach Auswahl eines Protokolls können für den Source-Port und den Destination-Port Dienste ausgewählt werden, die unter „6.4.1.1 Service anlegen“ für das entsprechende Protokoll definiert wurden.

Neue Quelladresse

- Es wird automatisch die Quelladresse des Out_Interface übernommen. Der Quellport kann geändert werden, indem ein neuer Quellport aus allen unter „6.4.1.1 Service anlegen“ definierten Services ausgewählt wird.

Neue Zieladresse

- Es muss ein unter „6.3.2 Rechner“ angelegter Rechner ausgewählt werden. Der Ziellport kann geändert werden, indem ein neuer Ziellport aus allen unter „6.4.1.1 Service anlegen“ definierten Services ausgewählt wird.

Das Auswahlfeld **Fertig** erscheint, nachdem alle notwendigen Eingaben gemacht wurden. Durch die Auswahl von Fertig wird die Regel noch einmal überprüft und ein Button zum Speichern angezeigt.

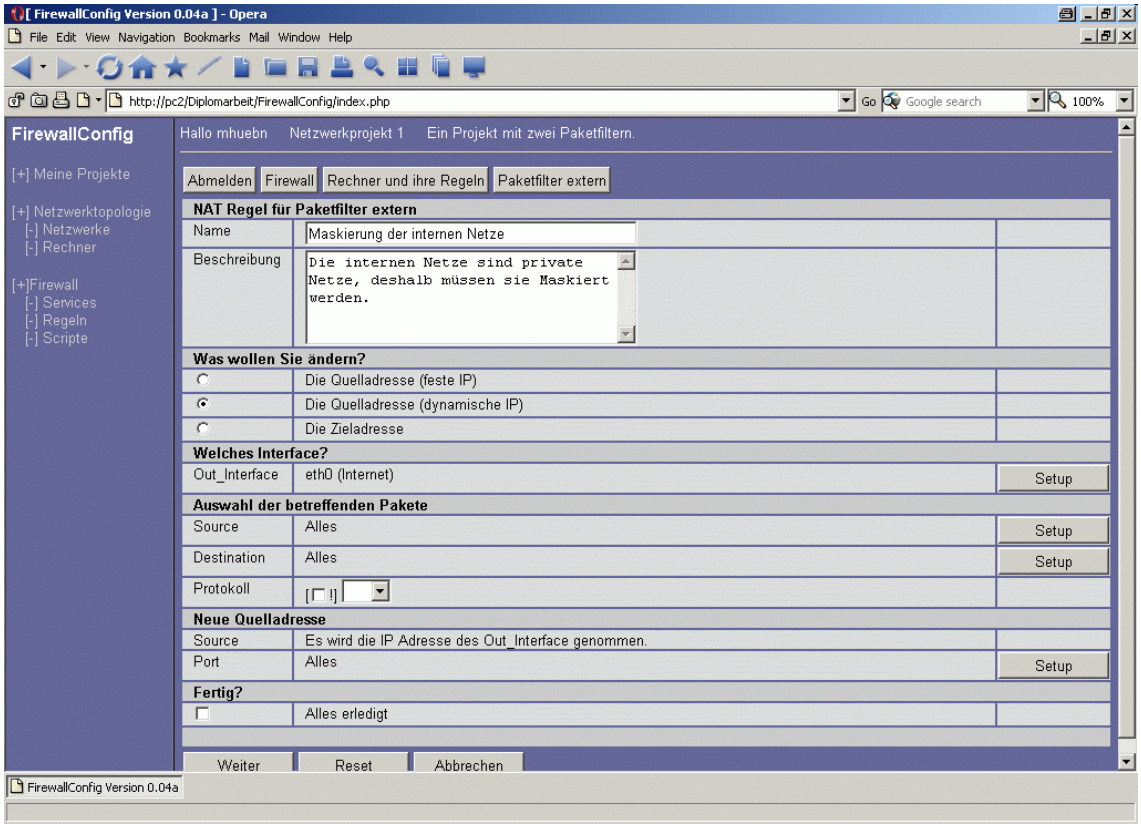

Abbildung 26: NAT Regel erstellen

6.4.2.7 NAT Regel ändern

Unter „Firewall/Regeln/Filter Regeln“ werden die bereits erstellten NAT Regeln angezeigt. Die NAT Regeln können hier geändert werden. Das Ändern einer NAT Regel geschieht analog zum Erstellen einer NAT Regel. Eine geänderte NAT Regel kann entweder als Änderung oder als neue Regel gespeichert werden.

6.4.2.8 Die Position einer NAT Regel ändern.

Unter „Firewall/Regeln/Filter Regeln“ werden die bereits erstellten NAT Regeln angezeigt. Die Position einer NAT Regel (Nummer) kann hier geändert werden. Zum Än-

dern der Position den Button mit der Nummer anklicken und anschließend eine neue Nummer eingeben.

6.4.2.9 Eine NAT Regel löschen

Unter „Firewall/Regeln/Filter Regeln“ werden die bereits erstellten NAT Regeln angezeigt. Die NAT Regeln können hier gelöscht werden.

6.4.3 Scripte

Unter „Firewall/Scripte“ können die Startscripte für die Firewallrechner erzeugt werden. Die automatisch generierten Scripte werden in einem Fenster angezeigt, in dem sie noch von Hand bearbeitet werden können [Abbildung 27: Scripte erstellen]. Der Button „Download Script“ öffnet den Download Dialog des Browsers, mit dem das Script lokal gespeichert werden kann. Das Script ist ein Textfile im UNIX Standard.

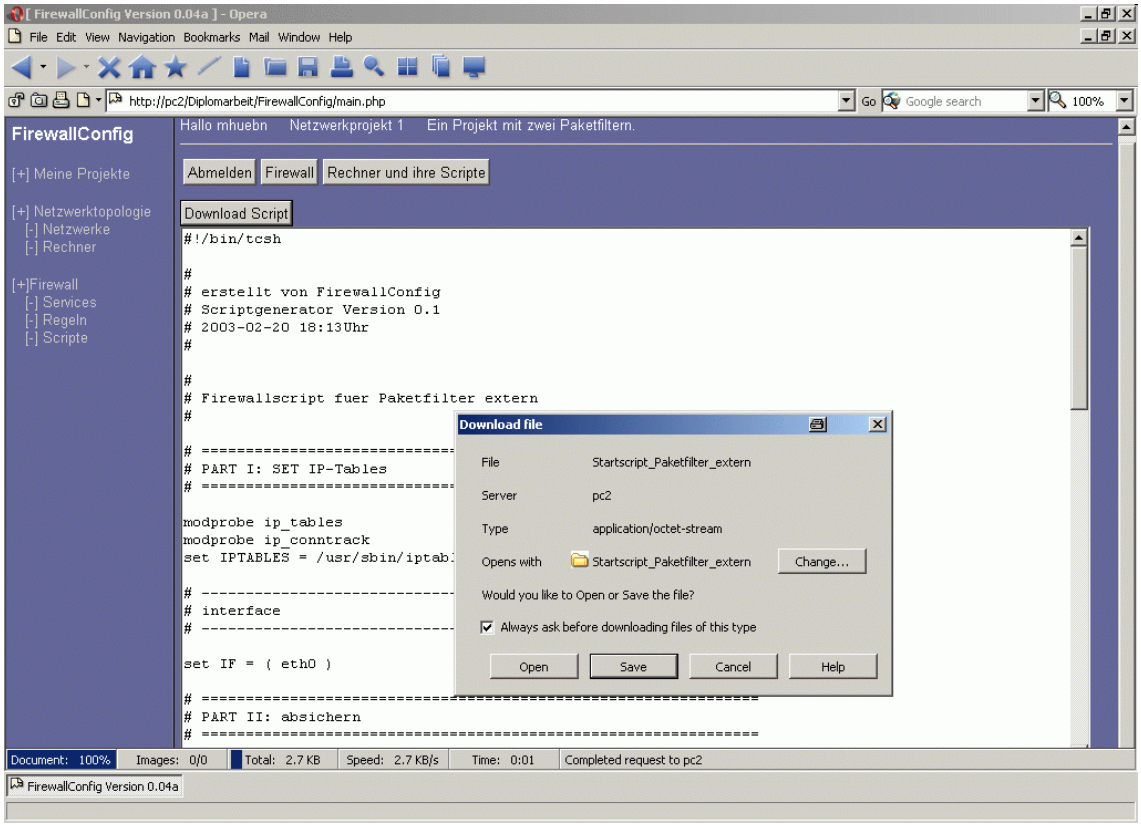

Abbildung 27: Scripte erstellen

7 Erläuterungen zum Quellcode

Dieses Kapitel richtet sich in erster Linie an Programmierer, die Änderungen am Quelltext des Konfigurationstools „FirewallConfig“ vornehmen wollen.

7.1 Programmstruktur

Die Startdatei der Software ist die Datei index.php welche die Aufgabe hat, das Frameset zu erstellen.

Das Programm wird über die Datei main.php kontrolliert. Als erstes wird in dieser Datei das Sessionmanagement gestartet. Die Session wird in der Tabelle currentsession der MySQL-Datenbank Firewallconfig gespeichert. Meldet sich ein Benutzer an, so wird der Benutzername als Sessonvariable in das Feld variables der Tabelle currentsession geschrieben. In das Feld laccess wird ein Zeitstempel eingetragen. Ist der Unterschied zwischen Zeitstempel und aktueller Zeit größer als 30min, wird die Session zerstört.

Ist ein Benutzer angemeldet, muss er ein Projekt auswählen. Die ProjektID wir ebenfalls als Sessionvariable im Feld variables der Tabelle currentsession gespeichert.

Existiert eine gültige Sessionvariable ProjektID, kann im Projekt navigiert werden. Die Navigation innerhalb des Projektes ist identisch zur Navigation auf der Bedienoberfläche aufgebaut.

Zwischen den Hauptprogrammteilen Projektmanagement, Netzwerktopologie und Firewall wird mit der Variable $decision gewechselt. Innerhalb dieser Hauptprogrammteile wird mit der Variable $part weiter verzweigt.

Die Ausgabe der Daten an den Browser erfolgt erst am Ende der Datei main.php. Um die auszugebenden Daten (Statusmeldungen, Formulare, Tabellen, …) während der Abarbeitung des Programms zwischenzuspeichern, existiert das globale Array $messages. Dieses Array ist zweidimensional und ermöglicht die Zwischenspeicherungen von Fehlermeldungen, Warnmeldungen, Inhalt, http-Header, Daten zum Download, Statusmeldungen sowie von Navigationsbuttons.

Das Laden der Listen erfolgt aus Performancegründen nicht rekursiv. Jede Liste wird separat geladen und die Verknüpfung der Objekte erfolgt, nachdem die Listen geladen wurden.

7.2 Verzeichnisstruktur

Im Wurzelverzeichnis des Programms Firewallconfig stehen die Dateien index.php, main.php und nav.php. Die Datei index.php erzeugt das Frameset. Die Datei nav.php enthält die Navigationsleiste. Die Datei main.php kontrolliert den Programmablauf. Das

Unterverzeichnis classes enthält die Definitionen der verwendeten Klassen. Dieses Verzeichnis hat als weitere Unterverzeichnisse die Verzeichnisse mysqlstub für die Klassen der Datenspeicherung sowie viewstub für die Darstellung der Listen. Das Unterverzeichnis conf enthält Konfigurationsdateien. Das Unterverzeichnis form enthält Formulare, die die Benutzereingaben entgegen nehmen. Das Unterverzeichnis include enthält Dateien, die zur Verzweigung des Programmablaufes benötigt werden. Das Unterverzeichnis inf enthält Informationen zum Programm. Das Unterverzeichnis libraries enthält Dateien, die zur Verarbeitung der Daten benötigt werden. Das Unterverzeichnis secure enthält die Installationsdatei/Benutzerverwaltung.

7.3 Informationsdateien

7.3.1 Datei firewallconfig.inf.php

In der Datei firewallconfig.inf.php sind Informationen zum Programmnamen, zur Programmversion sowie zum Datum der letzten Änderung hinterlegt.

7.4 Konfigurationsdateien

7.4.1 Datei dbconf.cfg.php

In der Datei dbconf.cfg.php werden die Variablen definiert, die für den Zugriff auf eine MySQL Datenbank erforderlich sind.

7.4.2 Datei firewallconfig.conf.php

In der Datei firewallconfig.conf.php werden die Variablen definiert, die die Software intern benötigt. Die Variable $programm_root enthält den absoluten Pfad, der beim Aufruf der Datei /secure/index.php automatisch gesetzt wird. Die Variable $timeout gibt an, nach welcher Zeit der Inaktivität man sich neu anmelden muss. Die Variablen $data_stub und $view_stub enthalten den Präfix der Klassen, die automatisch geladen werden (siehe Datei /classes/mytemplate.class.php).

7.5 Formulare

7.5.1 Datei authenticate.frm.php

Die Datei autenticate.frm.php erzeugt das Anmeldeformular.

7.5.2 Datei projectmanagement.frm.php

Die Datei projectmanagement.frm.php erzeugt das Formular zur Projektverwaltung.

7.5.3 Datei edit_projectmanagement.php

Die Datei edit_projectmanagement.frm.php erzeugt das Formular zum Anlegen und Ändern eines Projekts.

7.5.4 Datei import_project.frm.php

Die Datei import_project.frm.php erzeugt das Formular zur Auswahl des zu importierenden Projekts.

7.5.5 Datei edit_host.frm.php

Die Datei edit_host.frm.php erzeugt das Formular zum Anlegen und Ändern von Rechnern.

7.5.6 Datei edit_networkcard.frm.php

Die Datei edit_networkcard.frm.php erzeugt das Formular zum Anlegen und Ändern der Netzwerkkarten eines Firewallrechners.

7.5.7 Datei edit_network.frm.php

Die Datei edit_network.frm.php erzeugt das Formular zum Anlegen und Ändern eines Netzwerkes.

7.5.8 Datei edit_service.frm.php

Die Datei edit_service.frm.php erzeugt das Formular zum Anlegen oder Ändern eines Service.

7.5.9 Datei edit_filter.frm.php

Die Datei edit_filter.frm.php erzeugt das Formular zum Anlegen oder Ändern einer Filterregel. Dieses Formular sorgt gleichzeitig für die Integrität der Daten, indem es mit bereits verifizierten Daten arbeitet und nur gültige Kombinationen akzeptiert. Außerdem erzeugt diese Datei das Formular zum Ändern der Filterregelnummer.

7.5.10 Datei edit_nat.frm.php

Die Datei edit_nat.frm.php erzeugt das Formular zum Anlegen oder Ändern einer NAT-Regel. Dieses Formular sorgt gleichzeitig für die Integrität der Daten, indem es mit bereits verifizierten Daten arbeitet und nur gültige Kombinationen akzeptiert. Außerdem erzeugt diese Datei das Formular zum Ändern der NAT-Regelnummer.

7.5.11 Datei edit_user.frm.php

Die Datei edit_user.frm.php erzeugt das Formular zum Anlegen, Ändern und Löschen eines Benutzers.

7.6 Include Dateien

7.6.1 Datei projectmanagement.inc.php

Die Datei projectmanagement.inc.php ist die Schnittstelle zwischen GUI und Projektverwaltung.

7.6.2 Datei topologie.inc.php

Die Datei topologie.inc.php stellt Verzweigungen zur Rechnerverwaltung sowie zur Netzwerkverwaltung zur Verfügung.

7.6.3 Datei hosts.inc.php

Die Datei hosts.inc.php ist die Schnittstelle zwischen GUI und Rechnerverwaltung.

7.6.4 Datei networks.inc.php

Die Datei networks.inc.php ist die Schnittstelle zwischen GUI und Netzwerkverwaltung.

7.6.5 Datei firewall.inc.php

Die Datei firewall.inc.php stellt Verzweigungen zur Serviceverwaltung, Regelerstellung sowie Scripterzeugung zur Verfügung.

7.6.6 Datei services.inc.php

Die Datei services.inc.php ist die Schnittstelle zwischen GUI und Serviceverwaltung.

7.6.7 Datei rules.inc.php

Die Datei rules.inc.php ist die Schnittstelle zwischen GUI und Regelverwaltung.

7.6.8 Datei scripts.inc.php

Die Datei scripts.inc.php ist die Schnittstelle zwischen GUI und Scripterzeugung.

7.7 Libraries

7.7.1 Datei config_prog.lib.php

Die Datei config_prog.lib.php ist das Konfigurationsscript für das Konfigurationstool FirewallConfig. Es schreibt den Pfad des Programms in die Datei /conf/firewallconfig.conf.php, testet die Verbindung zu MySQL und erstellt die Datenbank FirewallConfig und die Tabellen.

7.7.2 Datei mysqlstuff.lib.php

In der Datei mysqlstuff.lib.php stehen die SQL Statements, die die Datenbank sowie die Tabellen erzeugen.

7.7.3 Datei mysessionmanagement.lib.php

Die Aufgabe der Datei mysessionmanagement.lib.php ist es, einen Benutzer anzumelden. In dieser Datei werden die Sessionfunktionen von PHP überschrieben, so dass die Session nun in der Tabelle currentsession der MySQL Datenbank FirewallConfig gespeichert wird. Jeder Zugriff auf das Programm startet die Sessionfunktion. Meldet sich ein Benutzer an, d.h. wird in der Tabelle users der Benutzername gefunden und stimmt das Passwort überein, so wird die Sessionvariable $sess_Username in der Session gespeichert. Ist die Sessionvariable $sess_Username vorhanden, so ist der Benutzer angemeldet. Bei jedem Zugriff auf die Session wird der Eintrag im Feld laccess ausgewertet und mit der Variable $timeout aus der Datei /conf/firewallconfig.conf.php verglichen. War der letzte Zugriff auf die Session nicht im von $timeout vorgegebenen Intervall, so wird die Session zerstört und der Benutzer muss sich erneut anmelden.

7.7.4 Datei grab_globals.lib.php

Die Datei grab_globals.lib.php stellt Funktionen zur Verarbeitung von $_POST und $_GET Variablen zur Verfügung.

7.7.5 Datei checkstuff.lib.php

Die Datei checkstuff.lib.php stellt Funktionen zur Verfügung, mit deren Hilfe sich Benutzereingaben überprüfen lassen.

7.7.6 Datei networkstuff.lib.php

Die Datei networkstuff.lib.php stellt Funktionen zur Verfügung, mit deren Hilfe die Netzadresse, die Netzmaske und die IP Adresse in das Binärformat umgewandelt werden können, sowie eine Funktion, mit deren Hilfe die Netzadresse aus der IP Adresse und der Netzmaske errechnet werden kann.

7.7.7 Datei sortstuff.lib.php

Die Datei sortstuff.lib.php stellt Funktionen zum Sortieren der Listen zur Verfügung.

7.7.8 Datei welknownportnumbers.lib.php

Die Datei welknownportnumbers.lib.php enthält ein mehrdimensionales Array, welches Informationen über die Ports 0-1023 enthält.

7.7.9 Datei registeredports.lib.php

Die Datei registeredports.lib.php enthält ein mehrdimensionales Array, welches Informationen über die Ports 1024-48556 enthält.

7.7.10 Datei convert_services.lib.php

Die Datei convert_services.lib.php kann dazu benutzt werden, die Arrays wellknownportnumbers und registeredports zu erstellen.

7.7.11 Datei icmp_type.lib.php

Die Datei icmp_type.lib.php enthält das Array $array_icmp_type, in welchem die Namen der ICMP Typen gespeichert sind.

7.7.12 Datei create_filterscript.lib.php

Die Datei create_filter_script.lib.php enthält den „Scriptgenerator“ für IPTables.

7.7.13 Datei viewstuff.lib.php

Die Datei viewstuff.lib.php enthält Funktionen, um das globale Array $messages auszulesen und die Daten an den Browser zu senden.

7.7.14 Datei style.css

Die Datei style.css enthält die verwendeten Style Sheets.

7.8 Klassen

7.8.1 Klasse myTemplate

Die Klasse myTemplate [Abbildung 28: Klasse myTemplate] dient als Vorlage für die anderen Klassen. In ihr werden Variablen und Funktionen definiert, die auch in anderen Klassen benötigt werden.

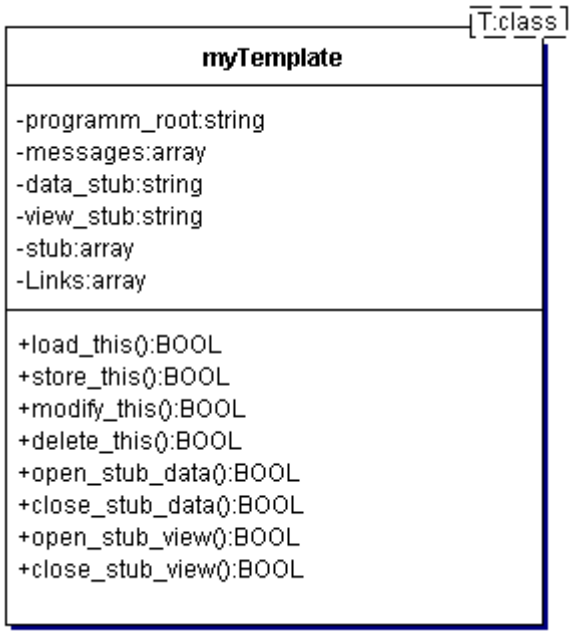

Abbildung 28: Klasse myTemplate

Attribute:

- **programm_root** ist der absolute Pfad zum Wurzelverzeichnis des Programms.
- **messages** ist ein Array, in das die Ausgaben an den Browser geschrieben werden.
- **data_stub** enthält ein Schlüsselwort, welches den Präfix des Unterverzeichnisses und den Präfix des Dateinamens der für die Datenspeicherung zuständigen Klassen darstellt.
- **view_stub** enthält ein Schlüsselwort, welches den Präfix des Unterverzeichnisses und den Präfix des Dateinamens der für die Generierung von Tabellen zuständigen Klassen darstellt.
- **stub_array** ist ein Array, das Verweise auf die aktuellen View- und Data-Klassen enthält.
- **Links** ist ein Array, in das Verweise auf Objekte eingetragen werden, die beim Löschen mit zerstört werden sollen.

Methoden:

- **load_this** lädt die Instanz der Klasse.
- **store_this** speichert die Instanz der Klasse.
- **modify_this** ändert die Instanz der Klasse.
- **delete_this** löscht die Instanz der Klasse.
- **open_stub_data** öffnet eine Verbindung zur Data-Klasse.

- **close_stub_data** schließt die Verbindung zur Data-Klasse.
- **open_stub_view** öffnet eine Verbindung zur View-Klasse.
- **close_stub_view** schließt die Verbindung zur View-Klasse.

7.8.2 Klasse myListClass

Die Klasse MyListClass erweitert die Klasse MyTemplate. Sie dient als Vorlage für die listenverwaltenden Klassen.

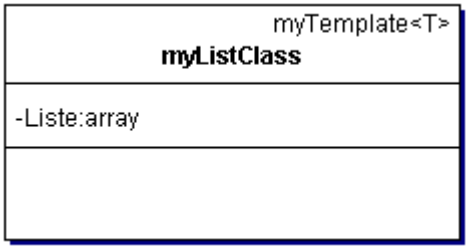

Abbildung 29: Klasse myListClass

Attribute:

- **Liste** ist ein Array, in das Verweise auf Instanzen einer Klasse eingetragen werden.

7.8.3 Klasse MySQL_stub

MySQL_stub ist Vorlage für alle Data-Klassen, die über MySQL auf ihre Daten zugreifen. Im Konstruktor der Klasse wird überprüft, ob bereits eine Verbindung zur Datenbank besteht. Falls das nicht der Fall ist, wird eine Verbindung hergestellt.

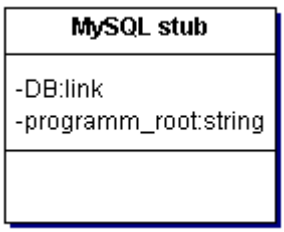

Abbildung 30: Klasse MySQLstub

Attribute:

- **DB** ist ein Handle zur MySQL Datenbank.
- **programm_root** ist der absolute Pfad zum Wurzelverzeichnis des Programms.

7.8.4 Klasse view_stub

Die Klasse view_stub erweitert die Klasse myTemplate. Sie ist Vorlage für die View-Klassen.

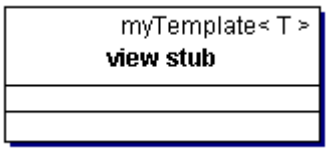

Abbildung 31: Klasse view_stub

7.8.5 Klasse ListOfUsers

Die Klasse ListOfUsers erweitert die Klassen myListClass und myTemplate. Sie dient als Kollektions- und Verwaltungsklasse für Instanzen der Klasse User.

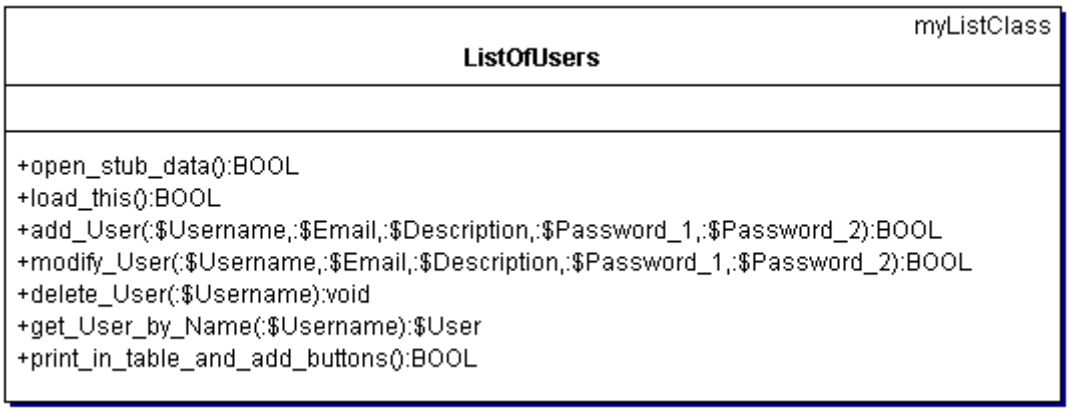

Abbildung 32: Klasse ListOfUsers

Methoden:

- **open_stub_data** überschreibt die Methode der Klasse myTemplate, da die User zurzeit nur in einer MySQL Datenbank gespeichert werden können.
- **load_this** überschreibt die Methode der Klasse myTemplate und lädt die User über die Klasse Data.
- **add_User** überprüft die Gültigkeit der Attribute, erzeugt eine neue Instanz der Klasse User, ruft die Methode store_this des Users auf und trägt den User in die Liste ein.
- **modify_User** überprüft die Gültigkeit der Attribute, ändert die Attribute und ruft die Methode modify_this des Users auf.

- **delete_User** lädt die ListOfProjects des Users, ruft die Methode delete_this der ListOfProjects auf, ruft die Methode delete_this() des Users auf und entfernt ihn aus der Liste.
- **get_User_by_Name** gibt den User zurück.
- **print_in_Table_and_add_Buttons** ruft die Methode print_this_in_table_and_add_buttons der View-Klasse auf.

7.8.6 Klasse MySQL_ListOfUsers

Die Klasse MySQL_ListOfUsers erweitert die Klasse MySQL_stub. Sie ist die Klasse Data der ListOfUsers.

Abbildung 33: Klasse MySQL_ListOfUsers

Methoden:

- **load_this** lädt die User aus einer MySQL Datenbank.

7.8.7 Klasse view_ListOfUsers

Die Klasse view_ListOfUsers erweitert die Klasse view_stub. Sie ist die Klasse View für die ListOfUsers..

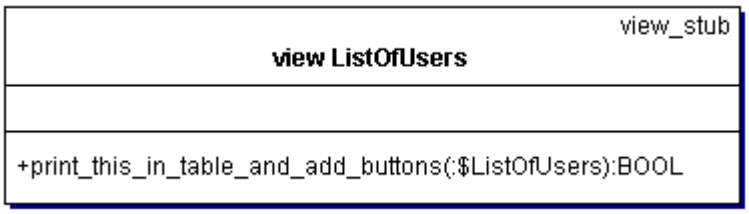

Abbildung 34: Klasse view_ListOfUsers

Methoden:

- **print_this_in_Table_and_add_Buttons** erstellt eine Tabelle aus der ListOfUsers.

7.8.8 Klasse User

Die Klasse User erweitert die Klasse myTemplate.

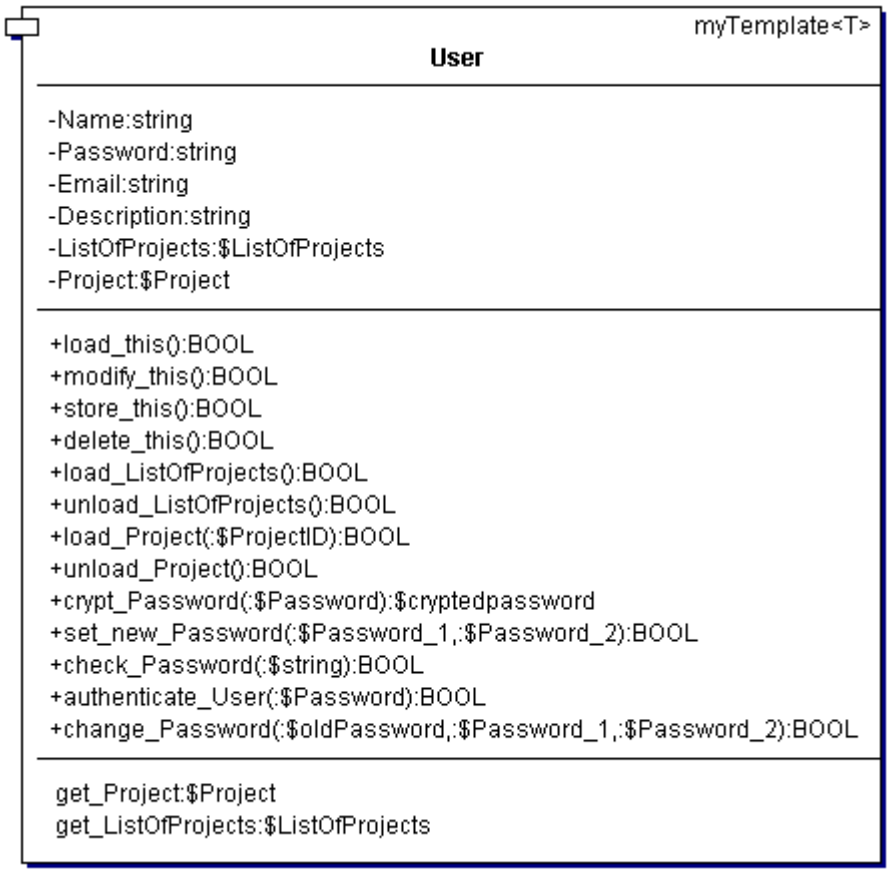

Abbildung 35: Klasse User

Attribute:

- Der **Name** des Users kann aus bis zu 128 Zeichen bestehen. Erlaubt sind die Zeichen [-1-9a-zA-Z._].
- **Password** ist das Passwort des Users. Es wird auf acht Zeichen gekürzt und kann die Zeichen [0-9a-zA-Z_] enthalten.
- **Email** ist die Email Adresse des Users. Die Email Adresse kann aus bis zu 128 Zeichen bestehen.
- **Description** ist ein Text, der den User beschreibt. Das Zeichen ‚|' darf in der Beschreibung nicht enthalten sein, da es beim Export die Datensätze trennt.
- **ListOfProjects** ist eine Instanz der Klasse ListOfProjects.
- **Project** ist eine Instanz der Klasse Projekt.

Methoden:

- **load_this** überschreibt die Methode der Klasse myTemplate. Sie lädt einen User über die Klasse Data.
- **modify_this** überschreibt die Methode der Klasse myTemplate. Sie ändert die Attribute eines Users über die Klasse Data.

- **store_this** überschreibt die Methode der Klasse myTemplate. Sie speichert die Attribute eines Users über die Klasse Data.
- **delete_this** überschreibt die Methode der Klasse myTemplate. Sie lädt die ListOfProjects, ruft deren Methode delete_this auf und löscht den User über die Klasse Data.
- **load_ListOfProjects** ruft die Methode load_this der ListOfProjects auf.
- **unload_ListOfProjects** gibt den Speicher frei.
- **load_Project** ruft die Methode load_this eines Projektes auf.
- **unload_Project** gibt den Speicher
- **crypt_Password** kürzt eine Passworteingabe auf acht Zeichen und verschlüsselt sie mit md5.
- **set_new_Password** vergleicht zwei Passworteingaben miteinander. Bei Erfolg wird das Passwort mit crypt_Passwort verschlüsselt und in das Attribut Password übernommen.
- **check_Password** überprüft, ob ein eingegebenes Passwort dem gespeicherten Passwort entspricht.
- **autenticate_User** überprüft das Passwort und speichert den User bei Erfolg in der Sessionvariablen $sess_Username. Damit ist der User angemeldet.
- **change_Password** überprüft, ob das $oldPassword dem gespeicherten Passwort entspricht. Bei Erfolg wird ein neues Passwort gesetzt.
- **get_ListOfProjects** lädt die ListOfProjects und gibt sie zurück.
- **get_Project** gibt ein geladenes Projekt zurück.

7.8.9 Klasse MySQL_User

Die Klasse MySQL_User erweitert die Klasse MySQL_stub. Sie ist die Klasse Data des User.

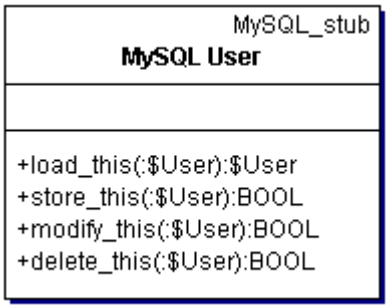

Abbildung 36: Klasse MySQL_User

Methoden:

- **load_this** enthält die SQL Statements, um den User aus der MySQL Datenbank zu laden.
- **store_this** enthält die SQL Statements, um den User in der MySQL Datenbank zu speichern.
- **modify_this** enthält die SQL Statements, um den User in der MySQL Datenbank zu ändern.
- **delete_this** enthält die SQL Statements, um den User aus der MySQL Datenbank zu löschen.

7.8.10 Klasse ListOfProjects

Die Klasse ListOfProjects erweitert die Klassen myListClass und myTemplate. Sie ist die Kollektions- und Verwaltungsklasse für die Projekte.

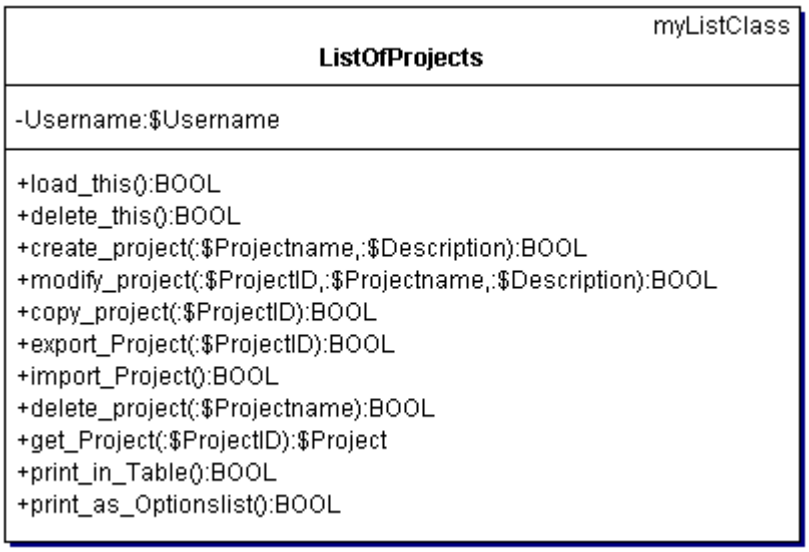

Abbildung 37: Klasse ListOfProjects

Attribute:

- **Username** ist eine Referenz auf das Attribut Name der Klasse User.

Methoden:

- **load_this** überschreibt die Methode der Klasse myTemplate. Sie lädt alle Projekte eines Users über die Klasse Data.
- **delete_this** überschreibt die Methode der Klasse myTemplate. Sie lädt jedes Projekt eines Users, ruft die Methoden load_this und delete_this des Projekts auf und entfernt es anschließend aus der Liste.

- **create_Project** überprüft die übergebenen Attribute, erzeugt ein neues Projekt, ruft die Methode store_this des Projekts auf, trägt das Projekt in die Liste ein und lädt das Projekt.
- **modify_Project** überprüft die übergebenen Attribute, ändert die Attribute des Projekts und ruft die Methode modify_this des Projekts auf.
- **copy_Project** lädt ein Projekt, erzeugt eine Kopie, ändert deren Projektnamen, ruft die Methode copy_this des kopierten Projekts auf und lädt es anschließend.
- **export_Project** lädt ein Projekt und ruft die Methode export_this des Projekts auf. Der Rückgabewert der Methode ist eine Variable, die das exportierte Projekt enthält.
- **import_Project** erzeugt ein leeres Projekt, füllt die Tabellen des Projekts, ruft die Methode copy_this des Projekts auf und lädt es anschließend.
- **delete_Project** lädt ein Projekt und ruft dessen Methode delete_this auf.
- **get_Project** gibt ein Projekt zurück.
- **print_in_Table** ruft die Methode print_this_in_Table der Klasse View auf.
- **print_as_Optionslist** ruft die Methode print_this_as_Optionslist der Klasse View auf.

7.8.11 Klasse MySQL_ListOfProjects

Die Klasse MySQL_ListOfProjects erweitert die Klasse MySQL_stub. Sie ist die Klasse Data der ListOfProjects.

Abbildung 38: Klasse MySQL_ListOfProjects

Methoden:

- **load_this** enthält die SQL Statements, um die Projekte aus der MySQL Datenbank zu laden.

7.8.12 Klasse view_ListOfProjects

Die Klasse view_ListOfProjects erweitert die Klasse view_stub. Sie ist die View Klasse der ListOfProjects.

Abbildung 39: Klasse view_ListOfProjects

Methoden:

- **print_this_in_Table** erstellt eine Tabelle mit den Projekten der ListOfProjects.
- **print_this_as_Optionslist** erstellt eine Auswahlliste mit den Projekten der ListOfProjects.

7.8.13 Klasse Project

Die Klasse Project erweitert die Klasse myTemplate. Sie ist die Verwaltungs- und Kollektionsklasse für das eigentliche Projekt.

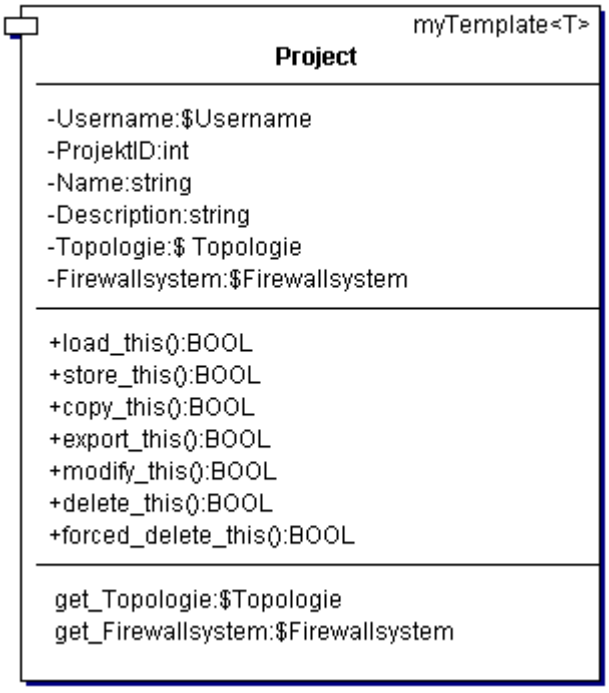

Abbildung 40: Klasse Project

Attribute:

- **Username** ist eine Referenz auf das Attibut Name eines Users.
- **ProjectID** ist ein Integer und dient zur eindeutigen Identifizierung eines Projekts.

- Der **Name** des Projekts kann aus bis zu 128 Zeichen bestehen. Erlaubt sind die Zeichen [-1-9a-zA-Z._].
- **Description** ist ein Text, der das Projekt beschreibt. Das Zeichen ‚|' darf in der Beschreibung nicht enthalten sein, da es beim Export die Datensätze trennt.
- **Topologie** ist eine Instanz der Klasse Topologie.
- **Firewallsystem** ist eine Instanz der Klasse Firewallsystem.

Methoden:

- **load_this** überschreibt die Methode der Klasse myTemplate. Sie lädt die Attribute eines Projekts über die Klasse Data, registriert die ProjektID in der Session, lädt die Topologie und das Firewallsystem und stellt das Projekt als Globale zur Verfügung.
- **store_this** überschreibt die Methode der Klasse myTemplate. Sie speichert die Attribute eines Projekts über die Klasse Data und erzeugt dabei die ProjektID.
- **copy_this** speichert die Attribute eines Projekts über die Klasse Data und erzeugt dabei eine neue ProjektID. Diese neue ProjektID ist über Referenzen im gesamten Projekt verfügbar. Dann werden die Methoden store_this des Firewallsystems und der Topologie aufgerufen.
- **export_this** schreibt die Inhalte der Tabellen des Projekts in eine Variable und gibt diese zurück.
- **modify_this** überschreibt die Methode der Klasse myTemplate. Sie ändert das Projekt über die Klasse Data.
- **delete_this** überschreibt die Methode der Klasse myTemplate. Sie löscht die ProjectID in der Session, ruft die Methoden delete_this des Firewallsystems und der Topologie auf und löscht das Projekt über die Klasse Data.
- **forced_delete_this** löscht im Gegensatz zu delete_this das Projekt nicht rekursiv, sondern ruft die Methode delete_entire_Project der Klasse Data auf.
- **get_Topologie** gibt die Topologie zurück.
- **get_Firewallsystem** gibt das Firewallsystem zurück.

7.8.14 Klasse MySQL_Project

Die Klasse MySQL_Project erweiter die Klasse MySQL_stub. Sie ist die Data Klasse des Projects.

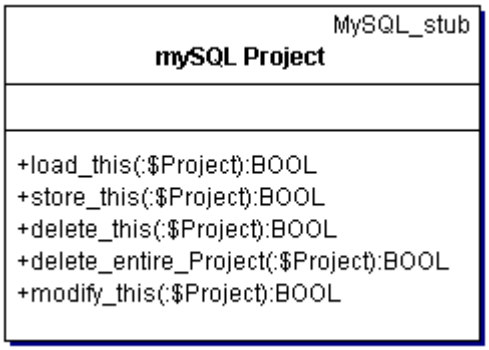

Abbildung 41: Klasse MySQL_Project

Methoden:

- **load_this** enthält die SQL Statements, um das Projekt aus der MySQL Datenbank zu laden.
- **store_this** enthält die SQL Statements, um das Projekt in der MySQL Datenbank zu speichern.
- **modify_this** enthält die SQL Statements, um das Projekt in der MySQL Datenbank zu ändern.
- **delete_this** enthält die SQL Statements, um das Projekt aus der MySQL Datenbank zu löschen.
- **delete_entire_Project** enthält die SQL Statements, um das komplette Projekt aus allen Tabellen zu löschen.

7.8.15 Klasse Topologie

Die Klasse Topologie erweitert die Klasse myTemplate. Sie ist eine Verwaltungsklasse für Rechner, Netzwerkkarten und Netzwerke.

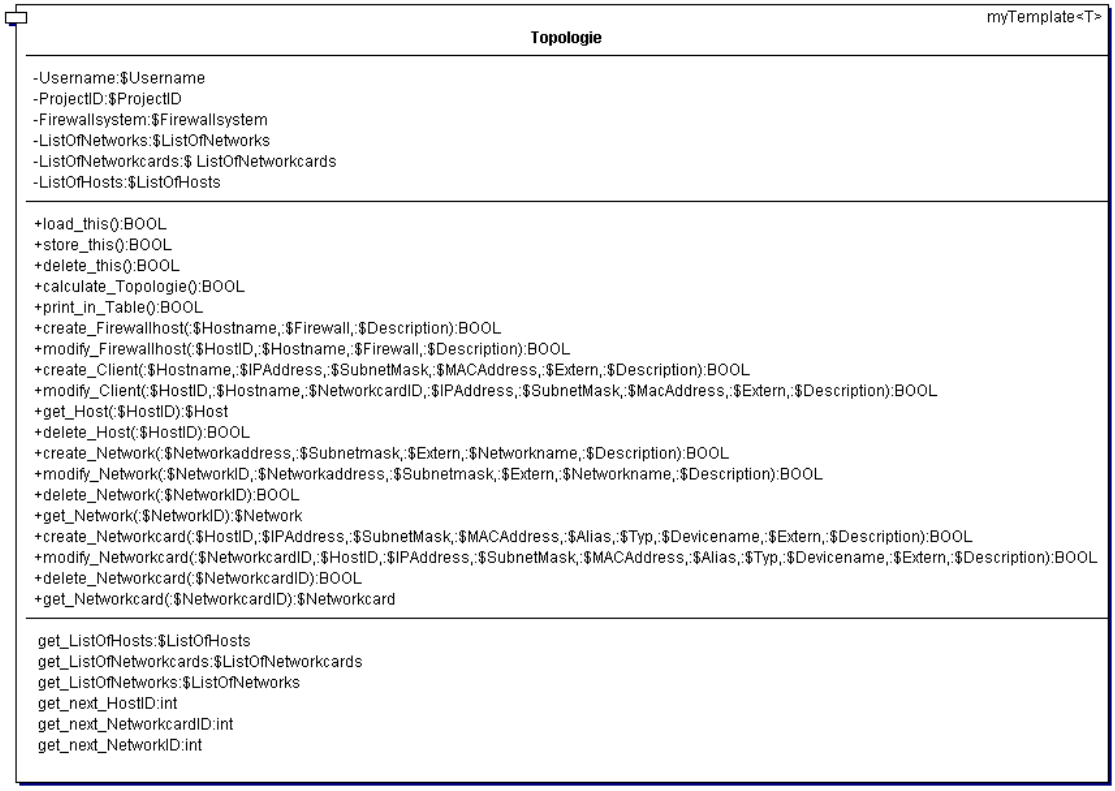

Abbildung 42: Klasse Topologie

Attribute:

- **Username** ist eine Referenz auf das Attribut Name des Users
- **ProjectID** ist eine Referenz auf das Attribut ProjectID des Projekts.
- **Firewallsystem** ist eine Referenz auf eine Instanz der Klasse Firewallsystem.
- **ListOfNetworks** ist eine Instanz der Klasse ListOfNetworks.
- **ListOfHosts** ist eine Instanz der Klasse ListOfHosts.
- **ListOfNetworkcards** ist eine Instanz der Klasse ListOfNetworkcards.

Methoden:

- **load_this** überschreibt die Methode der Klasse myTemplate. Sie erzeugt eine ListOfNetworks, eine ListOfHosts und eine ListOfNetworkcards und ruft deren Methoden load_this auf. Bei Erfolg ruft sie die Methode calculate_Topologie auf.
- **store_this** überschreibt die Methode der Klasse myTemplate. Sie ruft die Methoden store_this der Klassen ListOfNetworks, ListOfNetworkcards und ListOfHosts auf.

- **delete_this** überschreibt die Methode der Klasse myTemplate. Sie ruft die Methoden delete_this der Klassen ListOfHosts und ListOfNetworks auf. Die Netzwerkkarten werden von ihren Rechnern gelöscht.
- **calculate_Topologie** setzt Links zwischen Host, Networkcard und Network.
- **print_in_Table** ruft die Methode print_this_in_Table der Klasse View auf.
- **create_Firewallhost** überprüft die übermittelten Attribute und ruft die Methode create_Host der ListOfHosts auf.
- **modify_Firewallhost** überprüft die übermittelten Attribute und ruft die Methode modify_Host der ListOfHosts auf.
- **create_Client** überprüft die übermittelten Attribute, ruft die Methode create_Client der ListOfHosts und die Methode create_Networkcard auf.
- **modify_Client** überprüft die übermittelten Attribute, ruft die Methode modify_Client der ListOfHosts und die Methode modify_Networkcard auf.
- **delete_Host** ruft die Methode delete_Host der ListOfHosts auf.
- **get_Host** gibt den Host zurück.
- **create_Network** überprüft die übermittelten Attribute und ruft die Methode create_Network der ListOfNetworks auf.
- **modify_Network** überprüft die übermittelten Attribute und ruft die Methode modify_Network der ListOfNetworks auf.
- **delete_Network** ruft die Methode delete_Network der ListOfNetworks auf.
- **get_Network** gibt das Netzwerk zurück.
- **create_Networkcard** überprüft die übermittelten Attribute, ruft die Methode create_Networkcard der ListOfNetworkcards auf, setzt Links zum Rechner, ruft wenn nötig die Methode create_Network auf und setzt Links zum Netzwerk.
- **modify_Networkcard** überprüft die übermittelten Attribute, ruft die Methode modify_Networkcard der ListOfNetworkcards auf und ruft wenn nötig die Methode create_Network auf.
- **delete_Networkcard** ruft die Methode delete_Networkcard der ListOfNetworkcards auf.
- **get_Networkcard** gibt die Netzwerkkarte zurück.
- **get_ListOfHosts** gibt die ListOfHosts zurück.
- **get_ListOfNetworkcards** gibt die ListOfNetworkcards zurück.
- **get_ListOfNetworks** gibt die ListOfNetworks zurück.
- **get_next_HostID** gibt die nächste HostID des Projekts zurück.

- **get_next_NetworkcardID** gibt die nächste NetworkcardID des Projekts zurück.
- **get_Next_NetworkID** gibt die nächste NetworkID des Projekts zurück.

7.8.16 Klasse view_Topologie

Die Klasse view_Topologie erweitert die Klasse view_stub. Sie ist die Klasse View der Topologie.

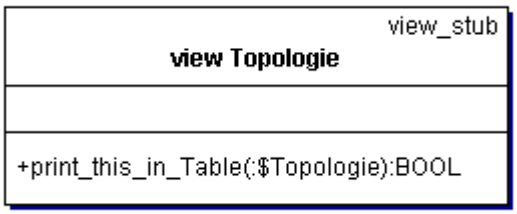

Abbildung 43: Klasse view_Topologie

Methoden:

- **print_this_in_Table** erstellt eine Tabelle mit allen Netzwerken und Rechnern des Projekts.

7.8.17 Klasse ListOfHosts

Die Klasse ListOfHosts erweitert die Klassen myListClass und myTemplate. Sie ist die Kollektions- und Verwaltungsklasse für die Hosts.

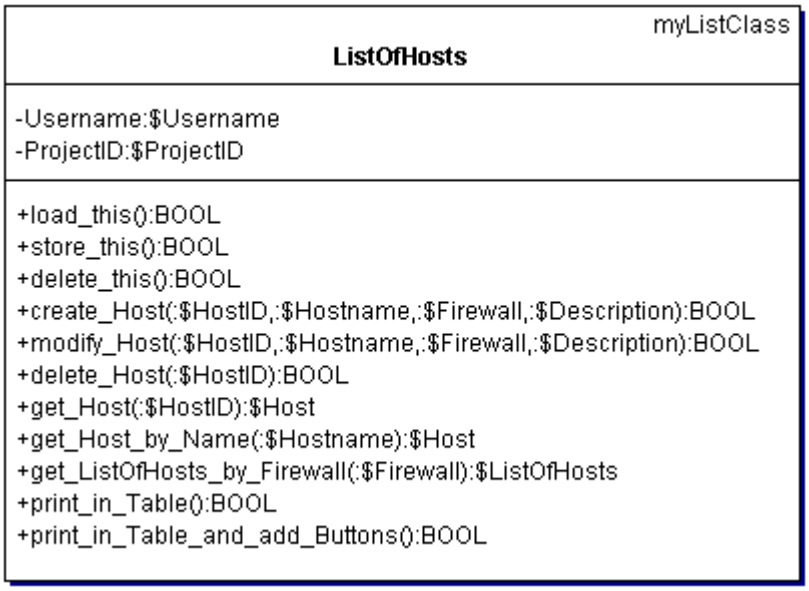

Abbildung 44: Klasse ListOfHosts

Attribute:

- **Username** ist eine Referenz auf das Attribut Name des Users
- **ProjectID** ist eine Referenz auf das Attribut ProjectID des Projekts.

Methoden:

- **load_this** überschreibt die Methode der Klasse myTemplate. Sie lädt die Rechner über die Klasse Data.
- **store_this** ruft für jeden Rechner in der Liste dessen Methode store_this auf.
- **delete_this** ruft für jeden Rechner in der Liste dessen Methode delete_this auf und entfernt ihn aus der Liste.
- **create_Host** erzeugt eine Instanz der Klasse Host, ruft deren Methode store_this auf und speichert sie in der Liste.
- **modify_Host** ändert die Attribute des Host und ruft dessen Methode modify_this auf.
- **delete_Host** ruft die Methode delete_this des Hosts auf und entfernt ihn aus der Liste.
- **get_Host** gibt den Host zurück.
- **get_Host_by_Name** sucht einen Host anhand seines Namens und gibt ihn zurück.
- **print_in_Table** ruft die Methode print_this_in_Table der Klasse View auf.
- **print_in_Table_and_add_Buttons** ruft die Methode print_this_in_Table_and_add_Buttons der Klasse View auf.

7.8.18 Klasse MySQL_ListOfHosts

Die Klasse MySQL_ListOfHosts erweitert die Klasse MySQL_stub. Sie ist die Klasse Data der ListOfHosts.

Abbildung 45: Klasse MySQL_ListOfHosts

Methoden:

- **load_this** enthält die SQL Statements, um die Rechner aus der MySQL Datenbank zu laden.

7.8.19 Klasse view_ListOfHosts

Die Klasse view_ListOfHosts erweitert die Klasse view_stub. Sie ist die Klasse View der ListOfHosts.

Abbildung 46: Klasse view_ListOfHosts

Methoden:

- **print_this_in_Table** erstellt eine Tabelle mit den Hosts der Liste.
- **print_this_in_Table_and_add_Buttons** erstellt eine Tabelle mit den Hosts der Liste und fügt Schaltflächen hinzu.

7.8.20 Klasse Host

Die Klasse Host erweitert die Klasse myTemplate.

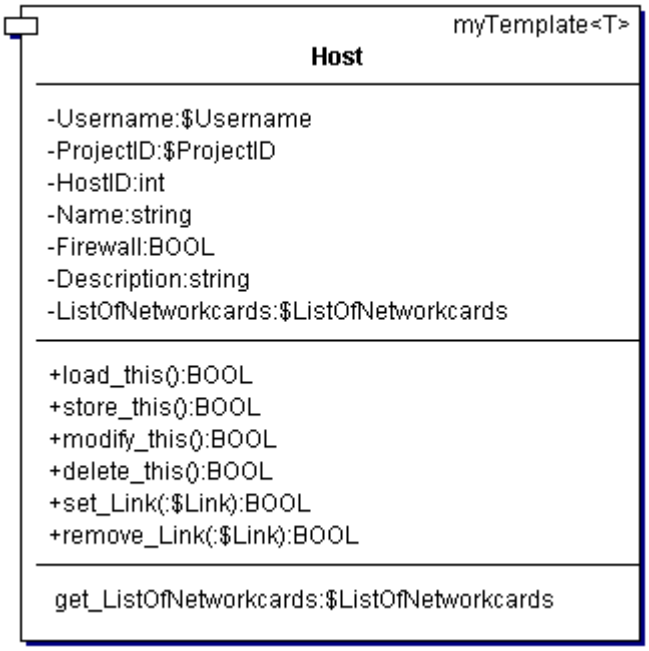

Abbildung 47: Klasse Host

Attribute:

- **Username** ist eine Referenz auf das Attibut Name des Users

- **ProjectID** ist eine Referenz auf das Attribut ProjectID des Projekts.
- **HostID** ist ein Integer und dient zur eindeutigen Identifizierung eines Host.
- Der **Name** des Hosts kann aus bis zu 128 Zeichen bestehen. Erlaubt sind die Zeichen [-1-9a-zA-Z._].
- **Firewall** gibt an, ob der Rechner Teil des Firewallsystems oder eine Client ist.
- **Description** ist ein Text, der den Host beschreibt. Das Zeichen ‚|' darf in der Beschreibung nicht enthalten sein, da es beim Export die Datensätze trennt.
- **ListOfNetworkcards** ist eine Instanz der Klasse ListOfNetworkcards, die Referenzen auf die Netzwerkkarten des Rechners enthält.

Methoden:

- **load_this** überschreibt die Methode der Klasse myTemplate. Sie lädt den Host über die Klasse View.
- **store_this** überschreibt die Methode der Klasse myTemplate. Sie speichert den Host über die Klasse View.
- **modify_this** überschreibt die Methode der Klasse myTemplate. Sie ändert den Host über die Klasse View.
- **delete_this** überschreibt die Methode der Klasse myTemplate. Sie ruft für alle Objekte, deren Referenzen im Array Links gespeichert sind, deren Methoden delete_this auf und löscht den Host über den stub_data.
- **set_Link** trägt eine übermittelte Referenz in das Array Links ein.
- **remove_Link** löscht eine übermittelte Referenz aus dem Array Links.
- **get_ListOfNetworkcards** gibt die ListOfNetworkcards zurück.

7.8.21 Klasse MySQL_Host

Die Klasse MySQL_Host erweitert die Klasse MySQL_stub. Sie ist die Klasse Data des Host.

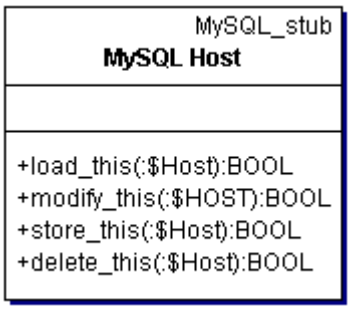

Abbildung 48: Klasse MySQL_Host

Methoden:

- **load_this** enthält die SQL Statements, um den Host aus der MySQL Datenbank zu laden.
- **store_this** enthält die SQL Statements, um den Host in der MySQL Datenbank zu speichern.
- **modify_this** enthält die SQL Statements, um den Host in der MySQL Datenbank zu ändern.
- **delete_this** enthält die SQL Statements, um den Host aus der MySQL Datenbank zu löschen.

7.8.22 Klasse ListOfNetworkcards

Die Klasse ListOfNetworkcards erweitert die Klassen myListClass und myTemplate. Sie ist die Verwaltungs- und Kollektionsklasse für die Networkcards.

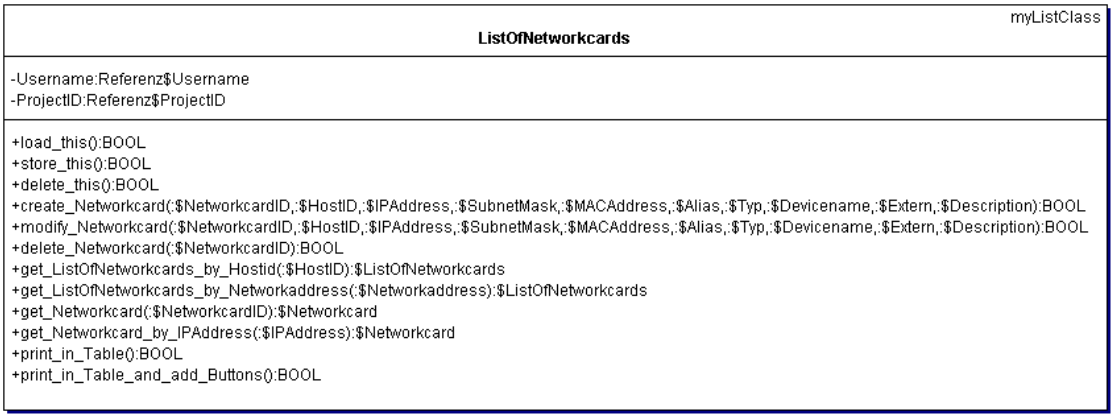

Abbildung 49: Klasse ListOfNetworkcards

Attribute:

- **Username** ist eine Referenz auf das Attibut Name des Users
- **ProjectID** ist eine Referenz auf das Attribut ProjectID des Projekts.

Methoden:

- **load_this** überschreibt die Methode der Klasse myTemplate. Sie lädt die Networkcards über die Klasse Data.
- **store_this** ruft für jede Networkcard in der Liste deren Methode store_this auf.
- **delete_this** ruft für jede Networkcard in der Liste deren Methode delete_this auf und entfernt sie aus der Liste.
- **create_Networkcard** erzeugt eine Instanz der Klasse Networkcard, ruft deren Methode store_this auf und trägt sie in die Liste ein.

- **modify_Networkcard** ändert die Networkcard und ruft deren Methode modify_this auf.
- **delete_Networkcard** ruft die Methode delete_this der Networkcard auf.
- **get_ListOfNetworkcards_by_HostID** erstellt eine Instanz der Klasse ListOfNetworkcards, trägt die Referenzen auf Networkcards mit gleicher HostID ein und gibt sie zurück.
- **get_ListOfNetworkcards_by_Networkaddress** erstellt eine Instanz der Klasse ListOfNetworkcards, trägt die Referenzen auf Netzwerkkarten mit gleicher Netzadresse (IPAddress & SubnetMask) ein und gibt sie zurück.
- **get_Networkcard** gibt die Networkcard zurück.
- **get_Networkcard_by_IPAddress** sucht eine Networkcard anhand ihrer IP Adresse und gibt sie zurück.
- **print_in_Table** ruft die Methode print_this_in_Table der View Klasse auf.
- **print_in_Table_and_add_Buttons** ruft die Methode print_this_in_Table_and_add_Buttons der View Klasse auf.

7.8.23 Klasse MySQL_ListOfNetworkcards

Die Klasse MySQL_ListOfNetworkcards erweitert die Klasse MySQL_stub. Sie ist die Klasse Data der ListOfNetworkcards.

Abbildung 50: Klasse MySQL_ListOfNetworkcards

Methoden:

- **load_this** enthält die SQL Statements, um die Networkcards aus der MySQL Datenbank zu laden.

7.8.24 Klasse view_ListOfNetworkcards

Die Klasse view_ListOfNetworkcards erweitert die Klasse view_stub. Sie ist die Klasse View der ListOfNetworkcards.

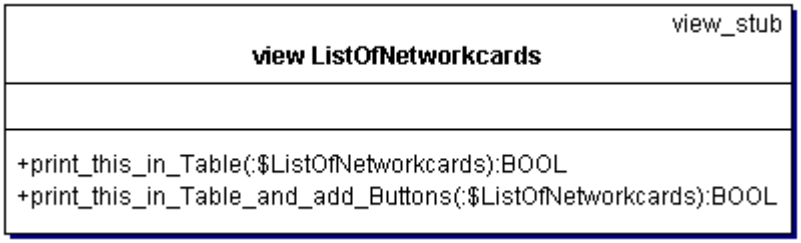

Abbildung 51: Klasse view_ListOfNetworkcards

Methoden:

- **print_this_in_Table** erstellt eine Tabelle der Networkcards der Liste.
- **print_this_in_Table_and_add_Buttons** erstellt eine Tabelle der Networkcards der Liste und fügt Schaltflächen hinzu.

7.8.25 Klasse Networkcard

Die Klasse Networkcard erweitert die Klasse myTemplate.

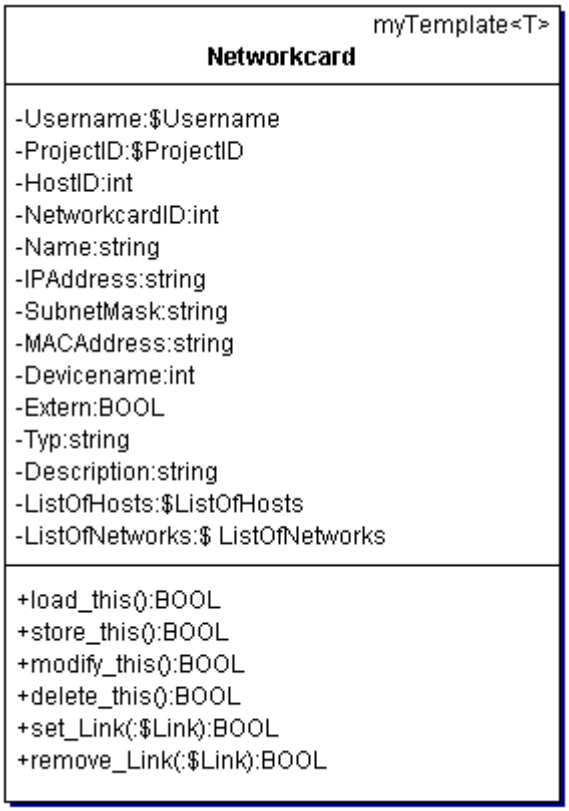

Abbildung 52: Klasse Networkcard

Attribute:

- **Username** ist eine Referenz auf das Attribut Name des Users

- **ProjectID** ist eine Referenz auf das Attribut ProjectID des Projekts.
- **NetworkcardID** ist ein Integer und dient zur eindeutigen Identifizierung einer Netzwerkkarte.
- **HostID** ist ein Integer und die ID des zugehörigen Rechners.
- Der **Name(Alias)** der Netzwerkkarte kann aus bis zu 128 Zeichen bestehen. Erlaubt sind die Zeichen [-1-9a-zA-Z._].
- Die **IPAddress** der Netzwerkkarte wird in der Form 192.186.5.33 angegeben.
- Die **SubnetMask** der Netzwerkkarte wird in der Form 255.255.255.0 angegeben.
- Die **MACAddress** der Netzwerkkarte wird in der Form AA:FF:67:F4:77:0A angegeben.
- Gültige **Devicename** sind eth[0-9], ipp[0-9], ppp[0-9] sowie lo.
- **Extern** gibt an, ob die IP Adresse eine offizielle Adresse ist.
- Der **Typ** der Netzwerkkarte gibt deren Technologie an (Ethernet, Token Ring).
- **Description** ist ein Text, der die Netzwerkkarte beschreibt. Das Zeichen ‚|' darf in der Beschreibung nicht enthalten sein, da es beim Export die Datensätze trennt.
- **ListOfHosts** ist eine Instanz der Klasse ListOfHosts, in der eine Referenz auf den Rechner der Netzwerkkarte gespeichert ist.
- **ListOfNetworks** ist eine Instanz der Klasse ListOfNetworks, in der eine Referenz auf das Netzwerk der Netzwerkkarte gespeichert ist.

Methoden:

- **load_this** überschreibt die Methode der Klasse myTemplate. Sie lädt die Networkcard über die Klasse Data.
- **store_this** überschreibt die Methode der Klasse myTemplate. Sie speichert die Networkcard über die Klasse Data.
- **modify_this** überschreibt die Methode der Klasse myTemplate. Sie ändert die Networkcard über die Klasse Data.
- **delete_this** überschreibt die Methode der Klasse myTemplate. Sie ruft für alle Objekte, deren Referenzen im Array Links gespeichert sind, deren Methoden delete_this auf und löscht die Networkcard über die Klasse Data.
- **set_Link** trägt eine übermittelte Referenz in das Array Links ein.
- **remove_Link** löscht eine übermittelte Referenz aus dem Array Links.

7.8.26 Klasse MySQL_Networkcard

Die Klasse MySQL_Networkcard erweitert die Klasse MySQL_stub. Sie ist die Klasse Data der Networkcard.

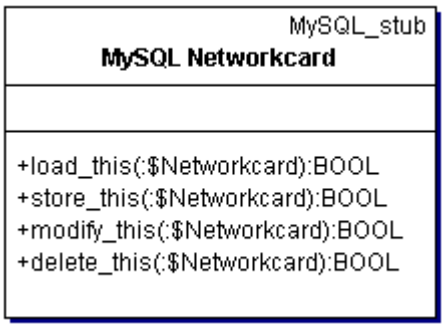

Abbildung 53: Klasse MySQL_Networkcard

Methoden:

- **load_this** enthält die SQL Statements, um die Networkcard aus der MySQL Datenbank zu laden.
- **store_this** enthält die SQL Statements, um die Networkcard in der MySQL Datenbank zu speichern.
- **modify_this** enthält die SQL Statements, um die Networkcard in der MySQL Datenbank zu ändern.
- **delete_this** enthält die SQL Statements, um die Networkcard aus der MySQL Datenbank zu löschen.

7.8.27 Klasse ListOfNetworks

Die Klasse ListOfNetworks erweitert die Klassen myListClass und myTemplate. Sie ist eine Verwaltungs- und Kollektionsklasse für die Networks.

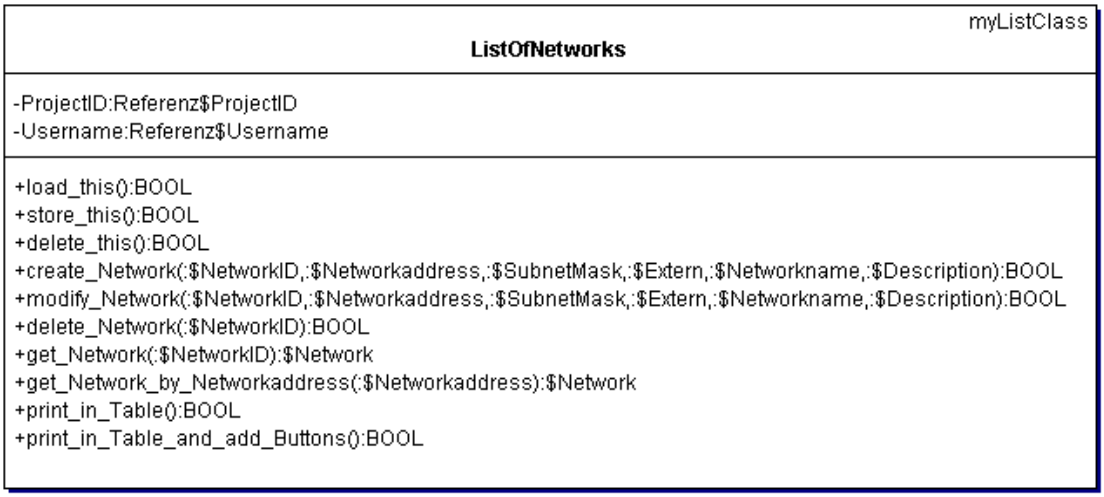

Abbildung 54: Klasse ListOfNetworks

Attribute:

- **Username** ist eine Referenz auf das Attribut Name des Users
- **ProjectID** ist eine Referenz auf das Attribut ProjectID des Projekts.

Methoden:

- **load_this** überschreibt die Methode der Klasse myTemplate. Sie lädt die Networks über die Klasse View.
- **store_this** ruft für jedes Network dessen Methode store_this auf.
- **delete_this** ruft für jedes Network dessen Methode delete_this auf und entfernt es aus der Liste.
- **create_Network** überprüft die übermittelten Attribute, erzeugt eine Instanz der Klasse Network, ruft dessen Methode store_this auf und trägt es in die Liste ein.
- **modify_Network** überprüft die übermittelten Attribute, ändert das Network und ruft dessen Methode modify_this auf.
- **delete_Network** ruft die Methode delete_this des Networks auf.
- **get_Network** gibt das Network zurück.
- **get_Network_by_Networkaddress** gibt das Network zurück.
- **print_in_Table** ruft die Methode print_this_in_Table der Klasse View auf.
- **print_in_Table_and_add_Buttons** ruft die Methode print_this_in_Table_and_add_Buttons der Klasse View auf.

7.8.28 Klasse MySQL_ListOfNetworks

Die Klasse MySQL_ListOfNetworks lädt die Daten aller Netzwerke aus einer MySQL Datenbank. Sie erweitert die Klasse MySQL_stub.

Abbildung 55: Klasse MySQL_ListOfNetworks

Methoden:

- **load_this** enthält die SQL Statements, um die Networks aus der MySQL Datenbank zu laden.

7.8.29 Klasse view_ListOfNetworks

Die Klasse view_ListOfNetworks stellt die in der ListOfNetworks gespeicherten Netzwerkkarten in Tabellenform dar. Sie erweitert die Klasse view_stub.

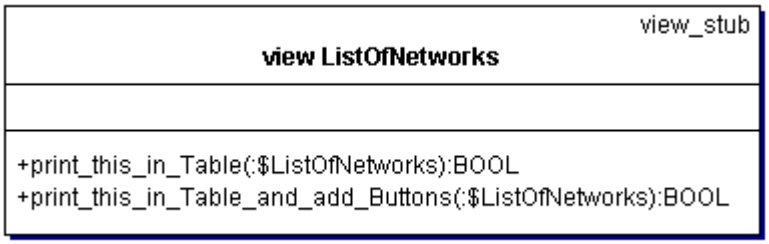

Abbildung 56: Klasse view_ListOfNetworks

Methoden:

- **print_this_in_Table** erstellt eine Tabelle der Networks.
- **print_this_in_Table_and_add_Buttons** erstellt eine Tabelle der Networks und fügt Schaltflächen hinzu.

7.8.30 Klasse Network

Die Klasse Network erweitert die Klasse myTemplate.

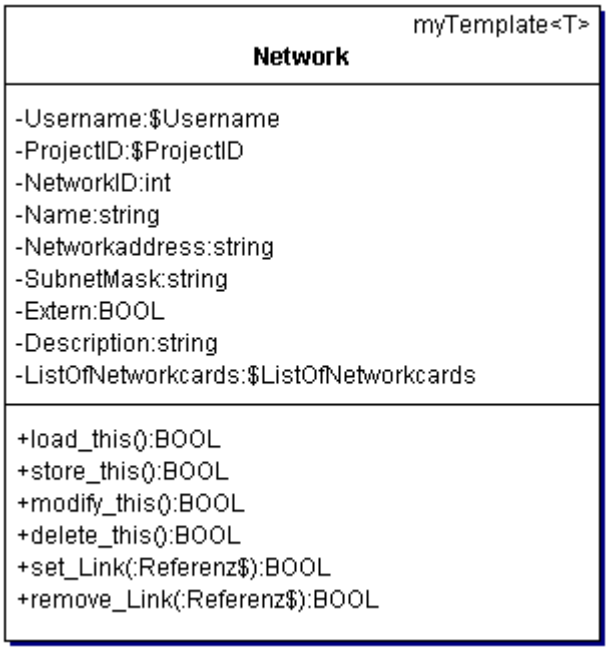

Abbildung 57: Klasse Network

Attribute:

- **Username** ist eine Referenz auf das Attribut Name des Users
- **ProjectID** ist eine Referenz auf das Attribut ProjectID des Projekts.
- **NetworkID** ist ein Integer und dient zur eindeutigen Identifizierung eines Netzwerks.
- Der **Name** des Netzwerks kann aus bis zu 128 Zeichen bestehen. Erlaubt sind die Zeichen [-1-9a-zA-Z._].
- Die **Networkaddress** des Netzwerks wird in der Form 192.186.5.0 angegeben.
- Die **SubnetMask** des Netzwerks wird in der Form 255.255.255.0 angegeben.
- **Extern** gibt an, ob die Netzadresse eine offizielle Adresse ist.
- **Description** ist ein Text, der die Netzwerkkarte beschreibt. Das Zeichen ‚|' darf in der Beschreibung nicht enthalten sein, da es beim Export die Datensätze trennt.
- **ListOfNetworkcards** ist eine Instanz der Klasse ListOfNetworkcards, in der Referenzen auf die in diesem Netzwerk befindlichen Netzwerkkarten gespeichert sind.

Methoden:

- **load_this** überschreibt die Methode der Klasse myTemplate. Sie lädt das Network über die Klasse Data.

- **store_this** überschreibt die Methode der Klasse myTemplate. Sie speichert das Network über die Klasse Data.
- **modify_this** überschreibt die Methode der Klasse myTemplate. Sie ändert das Network über die Klasse Data.
- **delete_this** überschreibt die Methode der Klasse myTemplate. Falls das Network keine Networkcards enthält, ruft sie für alle Objekte, deren Referenzen im Array Links gespeichert sind, deren Methoden delete_this auf und löscht das Network über die Klasse Data.
- **set_Link** trägt eine übermittelte Referenz in das Array Links ein.
- **remove_Link** löscht eine übermittelte Referenz aus dem Array Links.

7.8.31 Klasse MySQL_Network

Die Klasse MySQL_Network repräsentiert die Daten eines Netzwerks in einer MySQL Datenbank. Sie erweitert die Klasse MySQL_Stub.

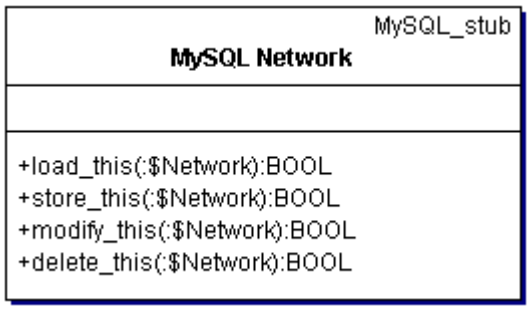

Abbildung 58: Klasse MySQL_Network

Methoden:

- **load_this** enthält die SQL Statements, um das Network aus der MySQL Datenbank zu laden.
- **store_this** enthält die SQL Statements, um das Network in der MySQL Datenbank zu speichern.
- **modify_this** enthält die SQL Statements, um das Network in der MySQL Datenbank zu ändern.
- **delete_this** enthält die SQL Statements, um das Network aus der MySQL Datenbank zu löschen.

7.8.32 Klasse Firewallsystem

Die Klasse Firewallsystem erweitert die Klasse myTemplate und ist eine Verwaltungsklasse.

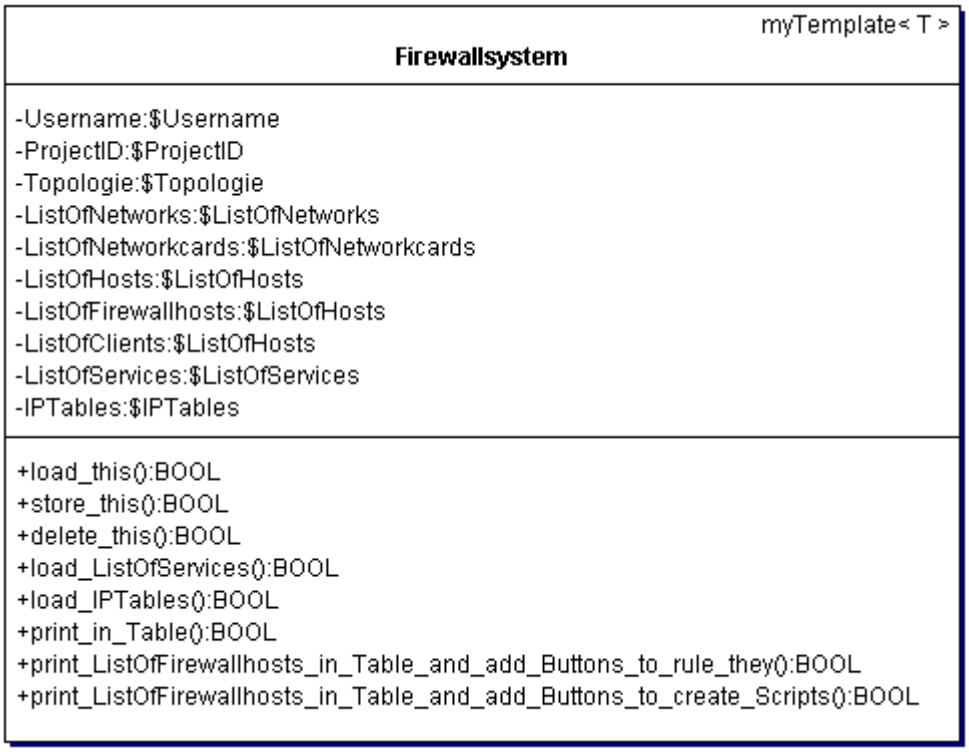

Abbildung 59: Klasse Firewallsystem

Attribute:

- **Username** ist eine Referenz auf das Attribut Name des Users
- **ProjectID** ist eine Referenz auf das Attribut ProjectID des Projekts.
- **Topologie** ist eine Referenz auf die Topologie.
- **ListOfNetworks** ist eine Referenz auf die ListOfNetworks der Topologie.
- **ListOfNetworkcards** ist eine Referenz auf die ListOfNetworkcards der Topologie.
- **ListOfHosts** ist eine Referenz auf die ListOfHosts der Topologie.
- **ListOfFirewallhosts** ist eine Instanz der Klasse ListOfHosts mit Referenzen auf alle Rechner, deren Attribut Firewall den Wert TRUE hat.
- **ListOfClients** ist eine Instanz der Klasse ListOfHosts mit Referenzen auf alle Rechner, deren Attribut Firewall nicht den Wert TRUE hat.
- **ListOfServices** ist eine Instanz der Klasse ListOfServices.
- **IPTables** ist eine Instanz der Klasse IPTables.

Methoden:

- **load_this** überschreibt die Methode der Klasse myTemplate. Sie setzt die Attribute ListOfNetworks, ListOfNetworkcards, ListOfHosts, ListOfFirewallhosta,

ListOfClients und ruft die Methoden load_ListOfServices und load_IPTables auf.

- **store_this** überschreibt die Methode der Klasse myTemplate und ruft die Methoden store_this der ListOfServices und von IPTables auf.
- **delete_this** überschreibt die Methode der Klasse myTemplate und ruft die Methoden delete_this der ListOfServices und von IPTables auf.
- **load_ListOfServices** erzeugt eine Instanz der Klasse ListOfServices und ruft deren Methode load_this auf.
- **load_IPTables** erzeugt eine Instanz der Klasse IPTables und ruft deren Methode load_this auf.
- **print_in_Table** ruft die Methode print_Firewallsystem der Klasse View auf.
- **print_ListOfFirewallhosts_in_Table_and_add_Buttons_to_rule_they** ruft die Methode print_ListOfFirewallhosts_in_Table_and_add_Buttons_to_rule_they der Klasse View auf.
- **print_ListOfFirewallhosts_in_Table_and_add_Buttons_to_create_Scripts** ruft die Methode print_ListOfFirewallhosts_in_Table_and_add_Buttons_to_create_Scripts der Klasse View auf.

7.8.33 Klasse view_Firewallsystem

Die Klasse view_Firewallsystem erweitert die Klasse view_stub und ist die Klasse View des Firewallsystems.

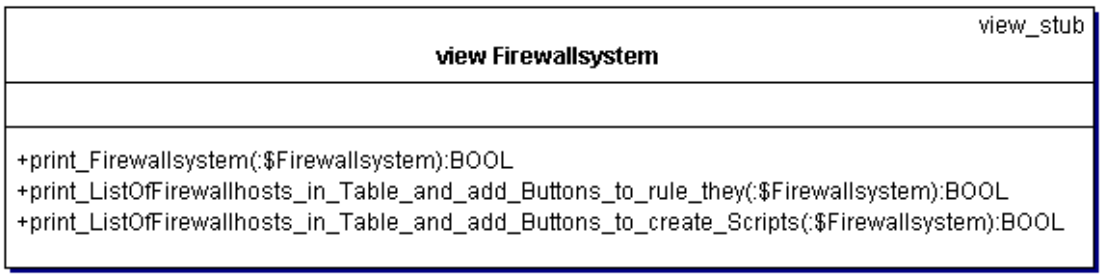

Abbildung 60: Klasse view_Firewallsystem

Methoden:

- **print_Firewallsystem** erstellt eine Tabelle der Rechner, deren Attribut Firewall TRUE ist und gibt deren Netzwerkkarten aus.

- **print_ListOfFirewallhoste_in_Table_and_add_Buttons_to_rule_they** gibt die ListOfFirewallhosts als Tabelle aus und fügt Buttons hinzu, um Regeln zu erstellen.
- **print_ListOfFirewallhosts_in_Table_and_add_Buttons_to_create_Scripts** gibt die ListOfFirewallhosts als Tabelle aus und fügt Buttons hinzu, um Startscripte zu erstellen.

7.8.34 Klasse ListOfServices

Die Klasse ListOfServices ist eine Kollektionsklasse für Instanzen der Klasse Service. Sie erweitert die Klassen myTemplate und myListClass.

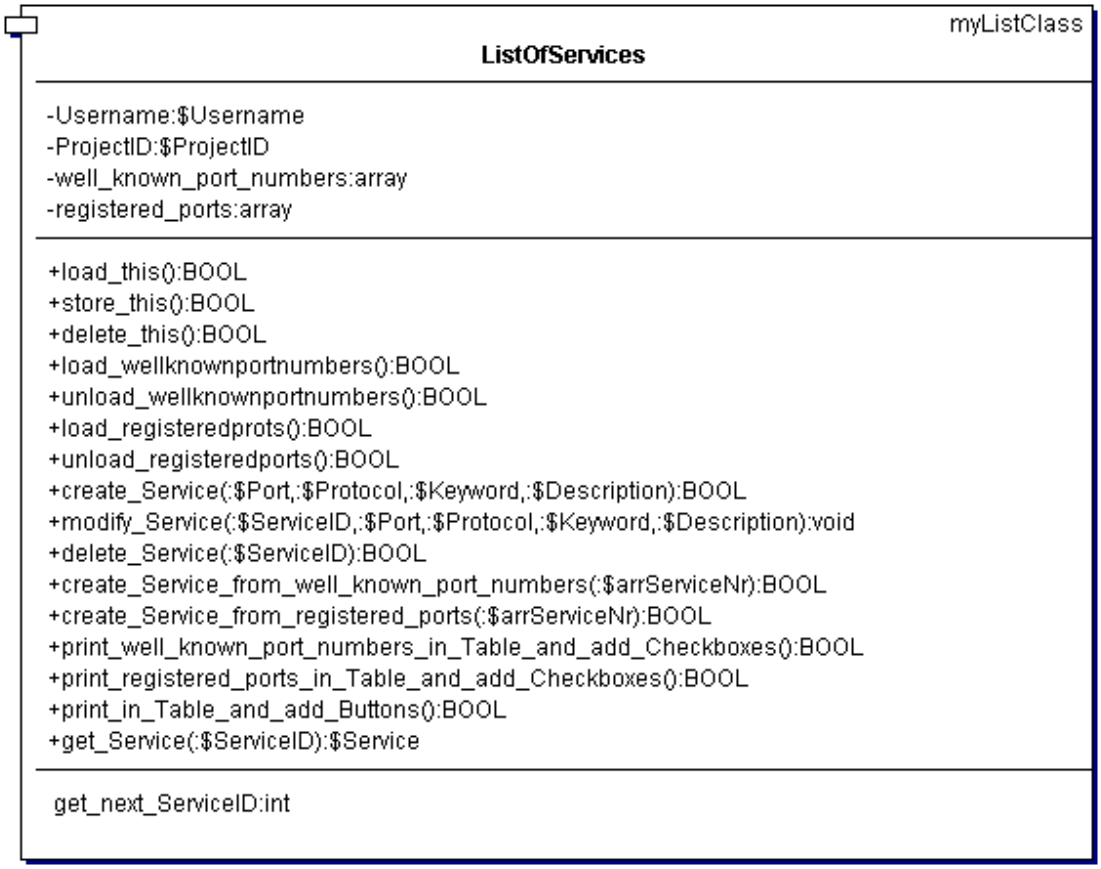

Abbildung 61: Klasse ListOfServices

Attribute:

- **Username** ist eine Referenz auf das Attribut Name des Users
- **ProjectID** ist eine Referenz auf das Attribut ProjectID des Projekts.
- **well_known_port_numbers** ist ein Array, welches die Ports von 0-1023 incl. Protokoll, Beschreibung und Keyword enthält.

- **registered_port_numbers** ist ein Array, welches die Ports von 1024-49151 incl. Protokoll, Beschreibung und Keyword enthält.

Methoden:

- **load_this** überschreibt die Methode der Klasse myTemplate. Sie lädt die Services über die Klasse View.
- **store_this** ruft für jeden Service dessen Methode store_this auf.
- **delete_this** ruft für jeden Service dessen Methode delete_this auf und entfernt ihn aus der Liste.
- **load_wellknownportnumbers** bindet die Datei /libraries/wellknownportnumbers.lib.php ein, in welcher das Array well_known_port_numbers definiert ist.
- **unload_wellknownportnumbers** gibt den Speicher des Arrays well_known_port_numbers frei.
- **load_registeredports** bindet die Datei /libraries/registeredports.lib.php ein, in welcher das Array registered_ports definiert ist.
- **unload_registeredports** gibt den Speicher des Arrays registered_ports frei.
- **create_Service** überprüft die übermittelten Attribute, ruft die Methode get_next_ServiceID auf, erzeugt eine Instanz der Klasse Service, ruft deren Methode store_this auf und trägt sie in die Liste ein.
- **modify_Service** überprüft die übermittelten Attribute, ändert den Service und ruft dessen Methode modify_this auf.
- **delete_Service** ruft dessen Methode delete_this auf und entfernt ihn aus der Liste.
- **create_Service_from_well_known_port_numbers** ruft die Methode load_wellknownportnumbers auf, entpackt die Servicenummern aus dem übermittelten Array, holt die zugehörigen Services aus dem Array well_known_port_numbers, ruft die Methode create_Service auf und gibt den Speicher mit unload_wellknownportnumbers wieder frei.
- **create_Service_from_registered_ports** ruft die Methode load_registeredports auf, entpackt die Servicenummern aus dem übermittelten Array, holt die zugehörigen Services aus dem Array registered_ports, ruft die Methode create_Service auf und gibt den Speicher mit unload_registeredports wieder frei.
- **print_well_known_port_numbers_in_Table_and_add_Checkboxes** ruft die Methode load_welknownportnumbers auf, ruft die Methode print_well_known_port_numbers_in_Table_and_add_Checkboxes der View Klasse auf und gibt den Speicher mit unload_wellknownportnumbers wieder frei.

- **print_registered_ports_in_Table_and_add_Checkboxes** ruft die Methode load_registeredports auf, ruft die Methode print_registered_ports_in_Table_and_add_Checkboxes der View Klasse auf und gibt den Speicher mit unload_registeredports wieder frei.
- **print_in_Table_and_add_Buttons** ruft die Methode print_this_in_Table_and_add_Buttons der View Klasse auf.
- **get_Service** gibt der Service zurück.
- **get_next_ServiceID** ermittelt die nächste ServiceID und gibt sie zurück.

7.8.35 Klasse MySQL_ListOfServices

Die Klasse MySQL_ListOfServices erweitert die Klasse MySQL_stub. Sie ist die Klasse Data der ListOfServices.

Abbildung 62: Klasse MySQL_ListOfServices

Methoden:

- **load_this** enthält die SQL Statements, um die Services aus der MySQL Datenbank zu laden.

7.8.36 Klasse view_ListOfServices

Die Klasse view_ListOfServices erweitert die Klasse view_stub. Sie ist Die Klasse view der ListOfServices.

Abbildung 63: Klasse view_ListOfServices

Methoden:

- **print_well_known_port_numbers_in_Table_and_add_Checkboxes** erstellt eine Tabelle, aus der Services im Bereich Port 0-1023 ausgewählt werden können.
- **print_registered_ports_in_Table_and_add_Checkboxes** erstellt eine Tabelle, aus der Services im Bereich Port 1024-49151 ausgewählt werden können.
- **print_this_in_Table_and_add_Buttons** erstellt eine Tabelle der Services und fügt Schaltflächen hinzu.

7.8.37 Klasse Service

Die Klasse Service erweitert die Klasse myTemplate.

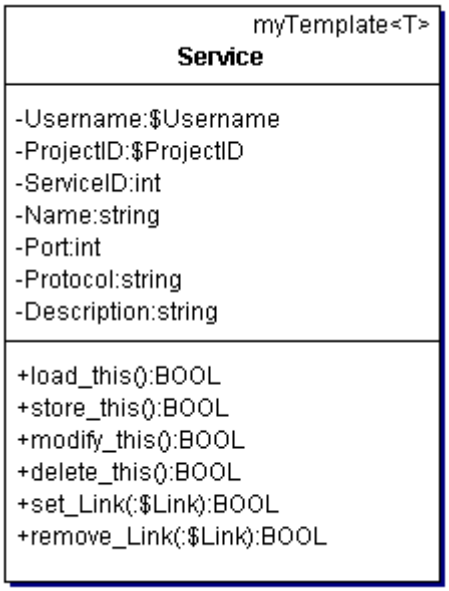

Abbildung 64: Klasse Service

Attribute:

- **Username** ist eine Referenz auf das Attribut Name des Users
- **ProjectID** ist eine Referenz auf das Attribut ProjectID des Projekts.
- **ServiceID** ist ein Integer und dient zur eindeutigen Identifizierung eines Service.
- Der **Name(Keyword)** des Service kann aus bis zu 128 Zeichen bestehen. Erlaubt sind die Zeichen [-1-9a-zA-Z._].
- Der **Port** kann ein Integer oder Bereich sein (Port[:Port]).
- **Protocol** ist optional tcp oder udp.
- **Description** ist ein Text, der den Service beschreibt. Das Zeichen ‚|' darf in der Beschreibung nicht enthalten sein, da es beim Export die Datensätze trennt.

Methoden:

- **load_this** überschreibt die Methode der Klasse myTemplate. Sie lädt die Attribute eines Service über die Klasse Data.
- **modify_this** überschreibt die Methode der Klasse myTemplate. Sie ändert die Attribute eines Service über die Klasse Data.
- **store_this** überschreibt die Methode der Klasse myTemplate. Sie speichert die Attribute eines Service über die Klasse Data.
- **delete_this** überschreibt die Methode der Klasse myTemplate. Sie ruft für alle Objekte, deren Referenzen im Array Links gespeichert sind, deren Methoden delete_this auf und löscht den Service anschließend über die Klasse Data.
- **set_Link** trägt eine übermittelte Referenz in das Array Links ein.
- **remove_Link** löscht eine übermittelte Referenz aus dem Array Links.

7.8.38 Klasse MySQL_Service

Die Klasse MySQL_Service erweitert die Klasse MySQL_stub. Sie ist die Klasse Data der Services.

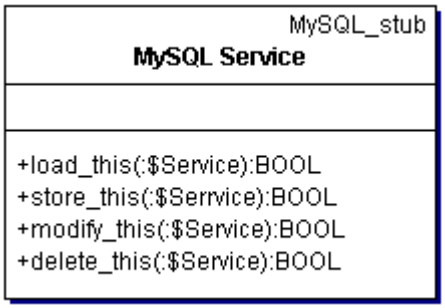

Abbildung 65: Klasse MySQL_Service

Methoden:

- **load_this** enthält die SQL Statements, um den Service aus der MySQL Datenbank zu laden.
- **store_this** enthält die SQL Statements, um den Service in der MySQL Datenbank zu speichern.
- **modify_this** enthält die SQL Statements, um den Service in der MySQL Datenbank zu ändern.
- **delete_this** enthält die SQL Statements, um den Service aus der MySQL Datenbank zu löschen.

7.8.39 Klasse IPTables

Die Klasse IPTables erweitert die Klasse myTemplate und ist eine Verwaltungsklasse für die Filterregeln.

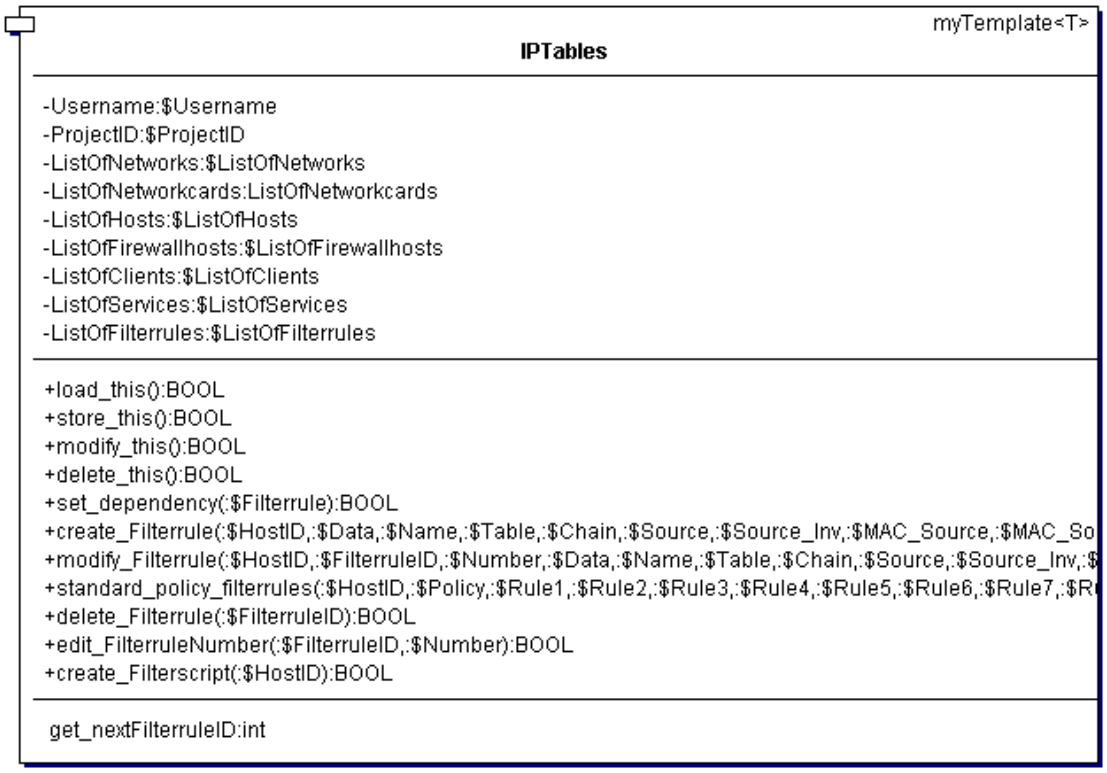

Abbildung 66: Klasse IPTables

- **Username** ist eine Referenz auf das Attribut Name des Users
- **ProjectID** ist eine Referenz auf das Attribut ProjectID des Projekts.
- **ListOfNetworks** ist eine Referenz auf das Attribut ListOfNetworks des Firewallsystems.
- **ListOfNetworkcards** ist eine Referenz auf das Attribut ListOfNetworkcards des Firewallsystems.
- **ListOfHosts** ist eine Referenz auf das Attribut ListOfHosts des Firewallsystems.
- **ListOfFirewallhosts** ist eine Referenz auf das Attribut ListOfFirewallhosts des Firewallsystems.
- **ListOfClients** ist eine Referenz auf das Attribut ListOfClients des Firewallsystems.
- **ListOfServices** ist eine Referenz auf das Attribut ListOfServices des Firewallsystems.
- **ListOfFilterrules** ist eine Instanz der Klasse ListOfFilterrules.

Methoden:

- **load_this** überschreibt die Methode der Klasse myTemplate. Sie erzeugt eine Instanz der Klasse ListOfFilterrules, ruft deren Methode load_this auf und ruft die Methode set_dependency auf.
- **store_this** überschreibt die Methode der Klasse myTemplate. Sie ruft der Methode store_this der ListOfFilterrules auf.
- **delete_this** überschreibt die Methode der Klasse myTemplate. Sie ruft die Methode delete_this der ListOfFilterrules auf.
- **set_dependency** ruft die Methoden set_Link der mit den Regeln zu verknüpfenden Objekte auf.
- **create_Filterrule** überprüft die übermittelten Attribute, ruft die Methode get_nextFilterruleID auf, ruft die Methode create_Filterrule der ListOfFilterrules auf und ruft die Methode set_dependency auf.
- **modify_Filterrule** überprüft die übermittelten Attribute und ruft die Methode modify_Filterrule der ListOfFilterrules auf.
- **standard_policy_filterrules** ruft die Methode create_Filterrule mit vordefinierten Attributen auf.
- **delete_Filterrule** ruft die Methode delete_Filterrule der ListOfFilterrules auf.
- **edit_FilterruleNumber** ändert die Filterregel-Nummer.
- **get_nextFilterruleID** gibt die nächste FilterruleID zurück.
- **create_Filterscript** ruf die Funktion create_Filterscript der Datei /libraries/create_filter_script.lib.php auf.

7.8.40 Klasse ListOfFilterrules

Die Klasse ListOfFilterrules erweitert die Klassen myListClass und myTemplate und ist eine Verwaltungs- und Kollektionsklasse für die Filterrules.

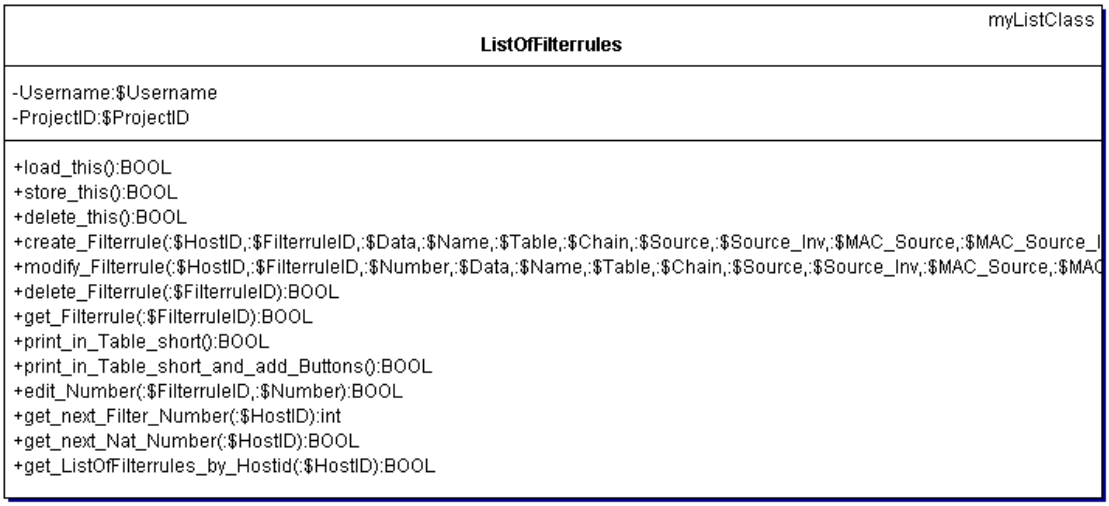

Abbildung 67: Klasse ListOfFilterrules

Attribute:

- **Username** ist eine Referenz auf das Attribut Name des Users
- **ProjectID** ist eine Referenz auf das Attribut ProjectID des Projekts.

Methoden:

- **load_this** überschreibt die Methode der Klasse myTemplate. Sie lädt die Filterregeln über die Klasse View.
- **store_this** ruft für jede Filterregel deren Methode store_this auf.
- **delete_this** ruft für jede Filterregel deren Methode delete_this auf und entfernt sie aus der Liste.
- **create_Filterrule** überprüft die übermittelten Attribute, erzeugt eine Instanz der Klasse Filterrule, ruft deren Methode store_this auf und trägt sie in die Liste ein.
- **modify_Filterrule** überprüft die übermittelten Attribute, ändert die Filterregel und ruft deren Methode modify_this auf.
- **delete_Filterrule** ruft die Methode delete_this der Filterregel auf.
- **get_Filterrule** gibt die Filterregel zurück.
- **print_in_Table_short** ruft die Methode print_this_in_Table_short der Klasse View auf.
- **print_in_Table_short_and_add_Buttons** ruft die Methode print_this_in_Table_short_and_add_Buttons der Klasse View auf.
- **edit_Number** ändert die durch Number gegebene Position einer Regel.
- **get_next_Filter_Number** gibt die nächste Number für eine Filterregel zurück.
- **get_next_NAT_Number** gibt die nächste Number für eine NAT Regel zurück.

- **get_ListOfFilterrules_by_HostID** erstellt eine Instanz der Klasse ListOfFilterrules, trägt die Referenzen auf Filterrules mit gleicher HostID darin ein und gibt sie zurück.

7.8.41 Klasse MySQL_ListOfFilterrules

Die Klasse MySQL_ListOfFilterrules erweitert die Klasse MySQL_stub. Sie ist die Klasse Data der ListOfFilterrules.

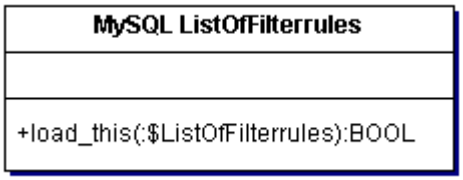

Abbildung 68: Klasse MySQL_ListOfFilterrules

Methoden:

- **load_this** enthält die SQL Statements, um die Filterrules aus der MySQL Datenbank zu laden.

7.8.42 Klasse view_ListOfFilterrules

Die Klasse view_ListOfFilterrules erweitert die Klasse view_stub. Sie ist die Klasse View der ListOfFilterrules.

Abbildung 69: Klasse view_ListOfServices

Methoden:

- **print_this_in_Table_short** erstellt eine Tabelle mit den Regeln der Liste.
- **print_this_in_Table_short_and_add_Buttons** erstellt eine Tabelle mit den Regeln der Liste und fügt Buttons hinzu.

7.8.43 Klasse Filterrule

Die Klasse Filterrule erweitert die Klasse myTemplate.

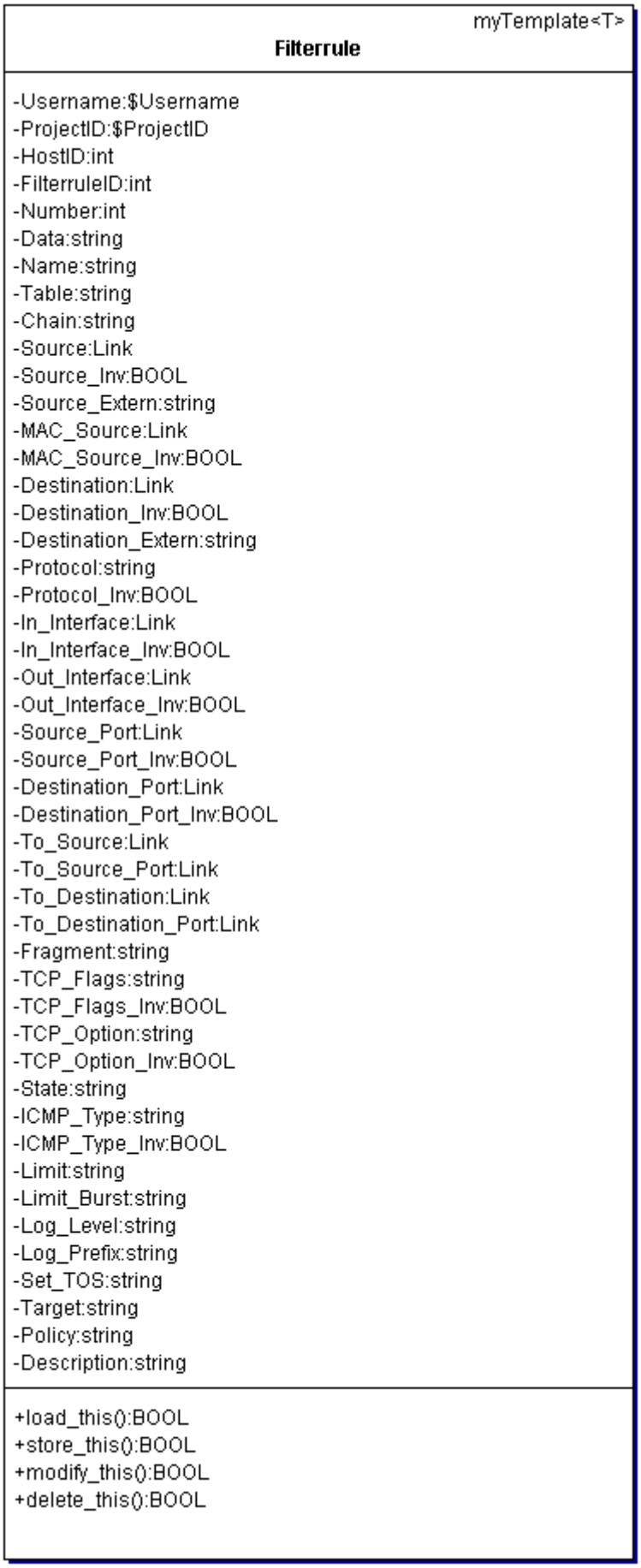
myTemplate<T>
Filterrule
-Username:$Username
-ProjectID:$ProjectID
-HostID:int
-FilterruleID:int
-Number:int
-Data:string
-Name:string
-Table:string
-Chain:string
-Source:Link
-Source_Inv:BOOL
-Source_Extern:string
-MAC_Source:Link
-MAC_Source_Inv:BOOL
-Destination:Link
-Destination_Inv:BOOL
-Destination_Extern:string
-Protocol:string
-Protocol_Inv:BOOL
-In_Interface:Link
-In_Interface_Inv:BOOL
-Out_Interface:Link
-Out_Interface_Inv:BOOL
-Source_Port:Link
-Source_Port_Inv:BOOL
-Destination_Port:Link
-Destination_Port_Inv:BOOL
-To_Source:Link
-To_Source_Port:Link
-To_Destination:Link
-To_Destination_Port:Link
-Fragment:string
-TCP_Flags:string
-TCP_Flags_Inv:BOOL
-TCP_Option:string
-TCP_Option_Inv:BOOL
-State:string
-ICMP_Type:string
-ICMP_Type_Inv:BOOL
-Limit:string
-Limit_Burst:string
-Log_Level:string
-Log_Prefix:string
-Set_TOS:string
-Target:string
-Policy:string
-Description:string
+load_this():BOOL
+store_this():BOOL
+modify_this():BOOL
+delete_this():BOOL

Abbildung 70: Klasse Filterrule

Attribute:

- **Username** ist eine Referenz auf das Attribut Name des Users
- **ProjectID** ist eine Referenz auf das Attribut ProjectID des Projekts.
- **HostID** ist ein Integer und dient der Zuordnung zu einem Firewallrechner.
- **FilterruleID** ist ein Integer und dient zur eindeutigen Identifizierung der Regel.
- Die **Number** ermöglicht es, die Reihenfolge von Filterregeln festzulegen.
- **Data** gibt die Art der gespeicherten Datan an (verified – die Regel enthält Links, spr – Standard Policy Rule mit Daten im Klartext).
- Der **Name** der Filterrule kann aus bis zu 128 Zeichen bestehen. Erlaubt sind die Zeichen [-1-9a-zA-Z._].
- **Table** ordnet die Regel einer Software zu. Es ist möglich, eine Regel für einen Filter (filter) oder eine Adressänderung (nat) zu erstellen.
- **Chain** gibt Auskunft über den Ort der Manipulation innerhalb des Paketfilters (INPUT, OUTPUT, FORWARD, PREROUTING, POSTROUTING).
- **Source** gibt Auskunft über den Absender eines Paketes und ist ein Link zu einer Netzwerkkarte oder einem Netzwerk der Topologie.
- **Source_Inv** invertiert die Source (TRUE).
- **Source_Extern** gibt an, dass die Source eine öffentliche Adresse hat (TRUE). Dieses Attribut wird benötigt, wenn die Source im Klartext angegeben wird.
- In **MAC_Source** kann als Source die MAC Adresse einer Networkkard angegeben werden (noch nicht implementiert).
- **Destination** gibt Auskunft über den Empfänger eines Paketes und ist ein Link zu einer Netzwerkkarte oder einem Netzwerk der Topologie.
- **Destination_Inv** invertiert die Destination (TRUE).
- **Destination_Extern** gibt an, dass die Destination eine öffentliche Adresse hat (TRUE). Dieses Attribut wird benötigt, wenn die Destination im Klartext angegeben wird.
- **Protocol** akzeptiert tcp, udp oder icmp.
- **In_Interface** ermöglicht es anzugeben, welches Interface für eintreffende Pakete zu überwachen ist und ist ein Link zu einer Netzwerkkarte des Firewallrechners.
- **In_Interface_Inv** invertiert das In_Interface (TRUE).

- **Out_Interface** ermöglicht es anzugeben, welches Interface für ausgehende Pakete zu überwachen ist und ist ein Link zu einer Netzwerkkarte des Firewallrechners.
- **Out_Interface_Inv** invertiert das Out_Interface (TRUE).
- **Source_Port** gibt Auskunft über den Quellport eines Paketes und ist ein Link zu einem Service.
- **Source_Port_Inv** invertiert den Source_Port (TRUE).
- **Destination_Port** gibt Auskunft über den Zielport eines Paketes und ist ein Link zu einem Service.
- **Destination_Port_Inv** invertiert den Destination_Port (TRUE).
- **To_Source** ermöglicht es, die Quelladresse eines Paketes zu verändern und ist ein Link zu einer Networkcard.
- **To_Source_Port** ermöglicht es, den Quellport eines Paketes zu verändern und ist ein Link zu einem Service.
- **To_Destination** ermöglicht es, die Zieladresse eines Paketes zu verändern und ist ein Link zu einer Networkcard.
- **To_Destination_Port** ermöglicht es, den Zielport eines Paketes zu verändern und ist ein Link zu einem Service.
- **Fragment** ermöglicht es, Regeln für fragmentierte Teile eines Paketes zu erstellen (TRUE).
- **TCP_Flags** ermöglicht es, Regeln für Pakete einer TCP-Verbindung zu erstellen, bei der eine bestimmte Kombination der Flags gesetzt ist (SYN, ACK, FIN, RST, URG, PSH, ALL, NONE).
- **TCP_Flags_Inv** invertiert TCP_Flags (TRUE).
- **TCP_Option** ermöglicht es, bei TCP-Verbindungen das Option Feld auszuwerten.
- **TCP_Option_Inv** invertiert TCP_Option (TRUE).
- **State** ermöglicht es, die mit connection tracking gewonnenen Informationen für stateful inspection zu nutzen (NEW, ESTABLISHED, RELATED, INVALID).
- **ICMP_Type** ermöglicht es, bei ICMP-Paketen die Felder Type und Code auszuwerten.
- **ICMP_Type_Inv** invertiert ICMP_Type (TRUE).
- **Limit** ermöglicht es, die Anzahl der Treffer einer Regel zu begrenzen.
- **Limit_Burst** siehe Limit.

- **Log_Level** ermöglicht es, den Kernel-Logging Modus einzustellen.
- **Log_Prefix** ermöglicht es, einen Text (29 Zeichen) vor die Logmeldung zu schreiben. Das Zeichen ‚|' darf im Log_Prefix nicht enthalten sein, da es beim Export die Datensätze trennt.
- **Set_TOS** ermöglicht es, den Type of Service im IP Header zu setzen.
- Im Attribut **Target** wird angegeben, was passieren soll, wenn eine Regel zutrifft (ACCEPT, DROP).
- Im Attribut **Policy** wird angegeben, was passieren soll, wenn keine Regel zutrifft (ACCEPT, DROP).
- In **Description** kann ein beschreibender Text zu einer Filterregel abgelegt werden. Das Zeichen ‚|' darf in der Beschreibung nicht enthalten sein, da es beim Export die Datensätze trennt.

Methoden:

- **load_this** überschreibt die Methode der Klasse myTemplate. Sie lädt die Attribute einer Filterrule über die Klasse Data.
- **modify_this** überschreibt die Methode der Klasse myTemplate. Sie ändert die Attribute einer Filterrule über die Klasse Data.
- **store_this** überschreibt die Methode der Klasse myTemplate. Sie speichert die Attribute einer Filterrule über die Klasse Data.
- **delete_this** überschreibt die Methode der Klasse myTemplate. Sie löscht die Filterrule über die Klasse Data.

7.8.44 Klasse MySQL_Filterrule

Die Klasse MySQL_Filterrule erweitert die Klasse MySQL_stub. Sie ist die Klasse data der Filterrule.

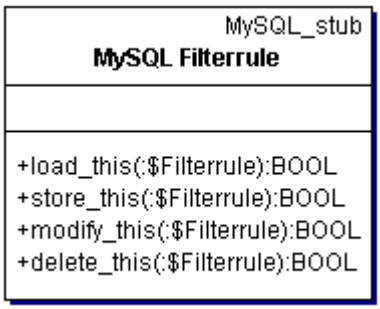

Abbildung 71: Klasse MySQL_Filterrule

Methoden:

- **load_this** enthält die SQL Statements, um die Filterrule aus der MySQL Datenbank zu laden.
- **store_this** enthält die SQL Statements, um die Filterrule in der MySQL Datenbank zu speichern.
- **modify_this** enthält die SQL Statements, um die Filterrule in der MySQL Datenbank zu ändern.
- **delete_this** enthält die SQL Statements, um die Filterrule aus der MySQL Datenbank zu löschen.

7.9 Besonderheiten PHP4

Bei der Programmierung mit PHP4 sind einige Besonderheiten zu beachten.

1. In PHP4 können Variable zu jedem Zeitpunkt ohne vorherige Deklaration und Initialisierung verwendet werden.
2. PHP4 unterstützt keine Datenkapselung.
3. PHP4 unterstützt keine Polymorphie und dynamisches Binden.
4. PHP4 unterstützt keine Zeiger. Zuweisungen mittels Adressoperator bewirken lediglich, das eine Referenz (ähnlich eines Links im UNIX Filesystem) erzeugt wird. Es besteht keine Möglichkeit, ein Attribut zu löschen, auf das noch Referenzen bestehen.

   Workaround:

```
$Attribut_1 = 1;
$Attribut_2 =& $Attribut_1;
$Attribut_1 = NULL;
```

   Weiterhin besteht keine Möglichkeit festzustellen, ob zwei Referenzen auf das gleiche Objekt zeigen (mit Variablen funktioniert es).

   Workaround:

```
$Objekt_1->ein_garantiert_nicht_verwendetes_Attribut = TRUE;
if($Objekt_2->ein_garantiert_nicht_verwendetes_Attribut)
{
 // dann sind Objekt_1 und Objekt_2 die gleiche Referenz
} // endif
else
{
 // Objekt_1 und Objekt_2 sind zwei verschiedene Instanzen
} // endelse
```

```
unset($Objekt_1->ein_garantiert_nicht_verwendetes_Attribut)
```

Referenzen: siehe http://www.phpcenter.de/en-html-manual/language.references.html

5. PHP4 unterstützt keinen Destruktor.

   Workaraound:

```
// Eine explizite Methode zum Löschen eines Objekts
class Demo
{
…
  function delete_this()
  {
    // tue, was noch getan werden muss
    $this = NULL; // das Objekt für alle Referenzen als nicht
                  // existent markieren
    unset($this); // falls nur eine Referenz auf das Objekt
                  // existiert, wird es aus dem Speicher entfernt
  } // end delete_this
} // end Class Demo
```

6. Das Konstrukt foreach zum Durchlaufen eines Arrays arbeitet mit Kopien der Einträge.

```
$Testarray = array();
$Testarray[0] = „erster Eintrag“;
$Testarray[1] = „zweiter Eintrag“;
$Testarray[0] = „dritter Eintrag“;

// Beispiel mit Kopien
foreach($Testarray as $Value)
{
 $Value = „Test“;  // Dieser Eintrag wird das Originalarray nicht
                   // verändern
} // end foreach

// Beispiel mit Original
foreach($Testarray as $Key=>$Value)
```

```
{
 $Testarray[$Key] = „Test“  // Dieser Eintrag verändert das
                            //Originalarrayy
} // end foreach
```

7. Return gibt im Normalfall eine Kopie zurück. Soll das Original zurückgegeben werden, muss die Funktion mit dem Adressoperator definiert werden.

```
funktion &Testfunktion()
{
   $Attribut = „Test“;
   return $Attribut;
} // end Testfunktion
```

8 Zusammenfassung und Ausblick

Mit der hier vorgestellten Diplomarbeit steht ein Werkzeug zur Verfügung, das es ermöglicht, die Topologie von Rechnernetzen zu erfassen und auf dieser Basis Filterregeln für ein verteiltes Firewallsystem zu erstellen. Dies spart Arbeitszeit und verringert die Fehlerträchtigkeit der Erstellung von Filterscripten.

Die Implementation einer Klasse „Delegate“ zur Erstellung von Proxy-Regeln konnte aus Zeitgründen nicht mehr durchgeführt werden, kann aber analog zu Klasse „IP-Tables“ erfolgen.

Des Weiteren würde sich anbieten, das Tool „FirewallConfig“ dahingehend zu erweitern, Routingtabellen zu erzeugen.

Als Nachteile von PHP4 stellten sich im Verlauf der Arbeit die ungenügende Unterstützung der Objektorientierung sowie die aufwändige Erstellung des GUI heraus. Da PHP4 noch eine junge Sprache ist, besteht jedoch die Hoffnung, dass in naher Zukunft die Objektorientierung gewissenhafter umgesetzt und von Modellierungssoftware wie Together® oder RationalRose® direkt unterstützt wird. Auch sind Entwicklungen von komfortablen IDE’s[4] für PHP4 zu beobachten, so dass sich die Unterstützung bei der Programmentwicklung unter PHP4 stetig verbessert.

[4] DIE: intergrated development environment

Anhang A: MySQL Tabellen

Tabelle 1: MySQL Tabelle „users“

Feld	**Datentyp**
Username	varchar(128) NOT NULL
Password	varchar(128) NOT NULL
Email	varchar(128)
Description	TEXT

Tabelle 2: MySQL Tabelle „currentsession“

Feld	**Datentyp**
sessionID	varchar(32) NOT NULL
variables	TEXT NOT NULL
laccess	int(14)

Tabelle 3: MySQL Tabelle „projects“

Feld	**Datentyp**
ProjectID	int AUTO_INCREMENT
Projectname	varchar(128) NOT NULL
Username	varchar(128) NOT NULL
Description	TEXT

Tabelle 4: MySQL Tabelle „hosts“

Feld	**Datentyp**
ProjectID	int NOT NULL
Username	varchar(128) NOT NULL
HostID	int NOT NULL
Hostname	varchar(128)
Firewall	varchar(5)
Description	TEXT

Tabelle 5: MySQL Tabelle „networks“

Feld	**Datentyp**
ProjectID	int NOT NULL
Username	varchar(128) NOT NULL
NetworkID	int NOT NULL
Networkname	varchar(128)
Networkaddress	varchar(15)
SubnetMask	varchar(15)
Extern	varchar(5)
Description	TEXT

Tabelle 6: MySQL Tabelle „networkcards“

Feld	**Datentyp**
ProjectID	int NOT NULL
Username	varchar(128)
HostID	int NOT NULL
NetworkcardID	int NOT NULL
Alias	varchar(128)
IPAddress	varchar(15)
SubnetMask	varchar(15)
MACAddress	varchar(128)
Devicename	varchar(128)
Extern	varchar(5)
Typ	varchar(12)
Description	TEXT

Tabelle 7: MySQL Tabelle „services“

Feld	**Datentyp**
ProjectID	int NOT NULL
Username	varchar(128)
ServiceID	int NOT NULL

Port	varchar(128)
Protocol	varchar(5)
Feldword	varchar(128)
Description	TEXT

Tabelle 8: MySQL Tabelle „filterrules“

Feld	**Datentyp**
ProjectID	int NOT NULL
Username	varchar (128) NOT NULL
HostID	int NOT NULL
FilterruleID	int NOT NULL
RNumber	int NOT NULL
RData	varchar(8)
RName	varchar(128)
RTable	varchar(6)
RChain	varchar(128)
RSource	varchar(128)
RSource_Inv	varchar(5)
RSource_Extern	varchar(5)
RMAC_Source	varchar(128)
RMAC_Source_Inv	varchar(5)
RDestination	varchar(128)
RDestination_Inv	varchar(5)
RDestination_Extern	varchar(5)
RProtocol	varchar(5)
RProtocol_Inv	varchar(5)
RIN_Interface	varchar(128)
RIn_Interface_Inv	varchar(5)
ROut_Interface	varchar(128)
RIn_Interface_Inv	varchar(5)
RSource_Port	varchar(128)

RSource_Port_Inv	varchar(5)
RDestination_Port	varchar(128)
RDestination_Port_Inv	varchar(5)
RTo_Source	varchar(128)
RTo_Source_Port	varchar(128)
RTo_Destination	varchar(128)
RTo_Source_Port	varchar(128)
RFragment	varchar(5)
RTCP_Flags	varchar(128)
RTCP_Flags_Inv	varchar(5)
RTCP_Option	varchar(128)
RTCP_Option_Inv	varchar(5)
RState	varchar(128)
RICMP_Type	varchar(128)
RICMP_Type_Inv	varchar(5)
RLimit	varchchar(128)
RLimit_Burst	varchar(128)
RLog_Level	varchar(128)
RLog_Prefix	varchar(128)
RSet_TOS	varchar(128)
RTarget	varchar(128)
RPolicy	varchar(6)
RDescription	TEXT

Quellenverzeichnis

Penguin Computing http://netfilter.filewatcher.org (Datum des Zugriffs: 20.Oktober 2002).

The Samba Team http://netfilter.samba.org (Datum des Zugriffs: 20.Oktober 2002).

Harald Welte http://netfilter.gnumonks.org (Datum des Zugriffs: 20.Oktober 2002).

Linux 2.4 Packet Filtering HOWTO v1.0.2 Mon May 1 2000.

Linux 2.4 NAT HOWTO v1.0.1 Mon May 1 2000.

man IPTABLES Aug 11,2000.

Krause,Jörg (2001): Programmieren lernen in PHP4. Hanser.

Krause,Jörg (2001): PHP 4. Die Referenz. Hanser.

Reeg, Christoph und Hatlag, Jens (2002): Datenbank, MySQL und PHP http://ffm.junetz.de/members/reeg/DSP/ (Datum des Zugriffs: 11.November 2002).

PHP Handbuch http://www.php-center.de/de-html-manual/ (Datum des Zugriffs: 28.Januar 2003).

Allgemeine deutschsprachige PHP-Liste
Informationen: http://www.phpcenter.de/php-de/
Listenadresse: php@phpcenter.de.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Diplomarbeit selbständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Ort, Datum

Unterschrift