FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Fachhochschule Aachen

Analyse und Konzeption einer Datentransformation von Firewall-Regeln

RWTH AACHEN UNIVERSITY

Georgios Diamantis

Matr.-Nr.: 3000664

1. Prüfer: Prof. Dr. Philipp Rohde
2. Prüfer: Marcel Crolla

08. Januar 2023

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die Seminararbeit mit dem Thema

Analyse und Konzeption einer Datentransformation von Firewall-Regeln

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, kenntlich gemacht sind und die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war. Ich verpflichte mich, ein Exemplar der Seminararbeit fünf Jahre aufzubewahren und auf Verlangen dem Prüfungsamt des Fachbereiches Medizintechnik und Technomathematik auszuhändigen.

Name: **Georgios Diamantis**

Aachen, den 08.01.2023

Georgios Diamantis

Digital unterschrieben
von Georgios Diamantis
Datum: 2023.01.08
10:00:32 +01'00'

Unterschrift der Studentin / des Studenten

Abstract

In dieser Seminararbeit wird die Anwendung „Firewall-View“ modelliert und konzipiert, welche als eine Ansichtsseite für die Firewall-Regeln der RWTH Aachen agieren soll. Die Anwendung soll dazu beitragen, dass Institutsadministratoren einen visuellen Überblick über Firewall-Regeln ihrer eigenen Anwendungen und Netzwerke bekommen, da sie aktuell noch keine einfache Möglichkeit dazu haben.

Da der Umgang mit den Daten den größten Einfluss auf den Erfolg der Anwendung hat, liegt der Hauptfokus der Arbeit bei der Datentransformation, die genutzt wird, um die Informationen aus der Firewall zu verarbeiten und in eine Zieldatenbank zu speichern, um diese visualisieren zu können. Dabei wird auch das verwendete Datenbankschema analysiert.

Zusätzlich wird auf Sicherheitsaspekte eingegangen, um zu garantieren, dass die sensiblen Daten der Firewall nur berechtigten Personen angezeigt werden.

Inhaltsverzeichnis

1 Einleitung

In der heutigen Zeit ist die Sicherheit von Computern und Netzwerken von großer Bedeutung, wodurch auch Firewall-Systeme eine wichtige und entscheidende Rolle einnehmen. Die RWTH Aachen nutzt eine Firewall mit mehreren Firewall-Regeln, um unerwünschte Verbindungen einzuschränken und nur erlaubte Verbindungen zuzulassen.

Da es viele verschiedene Institute mit verschiedensten Netzwerken, Servern und ihren Anwendungen innerhalb der RWTH Aachen gibt, muss es eine Schnittstelle für die Institutsadministratoren geben, um Freischaltungen in der Firewall einsehen zu können, da es aktuell dafür keine effiziente Möglichkeit gibt.

Mit Hilfe der Anwendung „Firewall-View“ soll diesem Problem entgegengewirkt werden. Die vorliegende Seminararbeit beschäftigt sich daher mit der Modellierung und Analyse der Anwendung „Firewall-View“, die in der Lage sein soll, die Firewall-Regeln der RWTH Aachen entsprechenden Nutzern anzuzeigen, die keinen direkten Zugriff auf die RWTH Firewall haben, jedoch trotzdem auf bestimmte Firewall-Regeln Lesezugriff bekommen sollen.

Zunächst wird es einen Überblick über Firewall-Systeme und ihre Funktionsweisen geben, bevor die Modellierung der Anwendung im Detail thematisiert werden kann.

2 Die Firewall und ihre Funktionen

Eine Firewall ist ein Sicherheitsmechanismus, der wie eine Art Schranke zwischen einem zu schützenden Netz und einem unsicheren Netz fungiert. Sie kontrolliert die gesamte Kommunikation und blockiert unerwünschte Verbindungen oder erlaubt zulässige Verbindungen. Firewalls werden verwendet, um das Netzwerk, Anwendungen und Services vor Angriffen von Hackern, Malware und anderen Bedrohungen zu schützen. Kommunikationsdaten werden analysiert, Kommunikationsbeziehungen und Kommunikationspartner werden kontrolliert und die Kommunikation wird nach einer Sicherheitspolitik reglementiert. Zusätzlich werden alle relevanten

Ereignisse protokolliert und entsprechende Warnungen dem Netz-Administrator vorgelegt. [3]

Die Funktionsweise einer Firewall basiert auf Protokollen und Regeln, die den Datenverkehr im Netzwerk blockieren oder passieren lassen. Die Regeln werden dabei durch den Absender der Daten, den Empfänger, das Protokoll, welches für den Datenverkehr verwendet wird, und den Port festgelegt.

Firewalls können auf verschiedene Arten arbeiten, zum Beispiel auf der Anwendungsebene, auf der Transportebene oder auf der Netzwerkebene. Diese Ebenen sind Teil des OSI-Schichtenmodells, welches von der „International Organization for Standardization“ (ISO) entworfen wurde, um eine Grundlage für Kommunikationsstandards zu legen. **(siehe Abbildung 1)**

	ISO/OSI Schicht	TCP/IP Schicht	Protokolle
7	Application Layer (Anwendungsschicht)	Application Layer	HTTP, SMTP, FTP, DHCP, Telnet
6	Presentation Layer (Darstellungsschicht)		
5	Session Layer (Sitzungsschicht)		
4	Transport Layer (Transportschicht)	Transport Layer	TCP, UDP
3	Network Layer (Vermittlungsschicht)	Internet Layer	IP, IPsec, IPv6, ICMP
2	Data Layer (Sicherungsschicht)	Network Access Layer	Ethernet
1	Physical Layer (Bitübertragungsschicht)		

Abbildung 1: OSI-Schichtenmodell [1]

Auf der Anwendungsschicht werden Protokolle wie zum Beispiel HTTP, FTP oder SMTP von der Firewall kontrolliert. In der Transportschicht kontrolliert die Firewall Protokolle TCP oder UDP. Auf der Netzwerkschicht werden Protokolle kontrolliert wie zum Beispiel IP oder IPv6. [5]

Die verschiedenen Schichten des OSI-Schichtenmodells bieten der Firewall eine strukturierte Möglichkeit den Datenverkehr zu überwachen und zu steuern, indem die Firewall Kommunikationsregeln in den unterschiedlichen Schichten festlegen kann.

Um sicherzustellen, dass die Firewall das Netzwerk vor Bedrohungen schützt, muss diese regelmäßig konfiguriert, erweitert und auf den neusten Stand gebracht werden. Institutsadministratoren können Änderungen an ihren Anwendungen, Netzen oder Servern vornehmen, welche eine neue Firewall-Regel oder eine Anpassung einer bereits vorhandenen Firewall-Regel zur Folge haben. Deswegen ist eine Anwendung zur Ansicht der Firewall-Regeln für Institutsadministratoren unerlässlich, da diese keine andere Möglichkeit haben, sich die vorhandenen Firewall-Regeln anzeigen zu lassen.

3 Die Anforderungen an die Anwendung „Firewall-View“

Um auf die Anwendung „Firewall-View“, die in der vorliegenden Seminararbeit thematisiert wird, näher einzugehen, muss zunächst auf die Anforderungen eingegangen werden. Diese beschreiben Eigenschaften und Funktionen, die die geplante Anwendung besitzen soll.

In der Anwendung können die Anforderungen in drei Bereiche unterteilt werden:

- Anforderungen an die Benutzeroberfläche
- Anforderungen an die Sicherheit
- Anforderungen an die Beschaffung der Daten

Die Benutzeroberfläche soll jeweils eine Ansichtsseite für den Nutzer und für den Administrator der Anwendung besitzen. Es soll jeweils eine Tabelle mit den Firewall-Regeln der RWTH Firewall angezeigt werden, welche die Details dieser Regeln anzeigen soll. Angezeigt werden sollen hier:

- der Absender
- der Empfänger mitsamt seiner IP-Adresse
- der genutzte Service mit Protokoll und Portnummer

Des Weiteren soll eine weitere Tabellenspalte Options, zusätzliche Funktionen der Regeln darstellen.

Dabei sollen einem Administrator der Anwendung alle Regeln angezeigt werden. Einem normalen Nutzer hingegen werden nur diese angezeigt, die zu

Netzwerken oder Anwendungen gehören, in denen der Nutzer als Ansprechpartner eingetragen ist und die Berechtigungen besitzt.

Um den Nutzer zu authentifizieren und um seine Berechtigungen rausfinden zu können, wird ein Login Verfahren namens Shibboleth genutzt, welches zur verteilten Authentifizierung und Autorisierung für Webanwendungen verwendet wird. Mithilfe von Shibboleth muss sich ein Nutzer nur einmal authentisieren, um auf mehrere Web-Anwendungen der RWTH Aachen Zugriff zu bekommen. **[10]**

Bei der Beschaffung der Daten sollen die Firewall-Regeln aus der Firewall der RWTH Aachen mittels Skripts extrahiert und in eine strukturierte Datenbank eingetragen werden.

4 Sicherheit

Da Firewall-Regeln sensible Daten sind, die nicht in die falschen Hände geraten sollten, ist das Thema der Sicherheit ein sehr großer Faktor bei dem Konzept dieser Anwendung. Wie schon durch die Anforderungen deutlich geworden ist, wird hierfür das Single Sign-On Verfahren genutzt, welches in allen Webanwendungen der RWTH Aachen genutzt wird. Dadurch werden der Anwendung mehrere Attribute des eingeloggten Nutzers mitgeteilt, wie zum Beispiel Vorname, Nachname, E-Mail-Adresse und RWTH-Kennung. Die RWTH-Kennung ist ein Benutzername, welcher innerhalb der RWTH Aachen genutzt wird, um den Nutzer eindeutig zu identifizieren.

Außerdem werden dem Nutzer über die bereits vorhandene API „itc-gem_rights-manager“ Zugehörigkeiten zugeteilt, wie zum Beispiel der Studenten-, Mitarbeiter- oder Administratorenstatus der Anwendung. Dadurch kann die zweite Ansichtsseite, die nur für Administratoren vorgesehen ist, für Nutzer, die diese Zugehörigkeit nicht vorweisen können, ausgeblendet werden. Es werden somit nur ausgewählte Mitarbeiter die Möglichkeit haben, alle Firewall-Regeln angezeigt zu bekommen.

Um eingrenzen zu können, welche Firewall-Regeln dem Nutzer angezeigt werden sollen, wird wieder auf die API „itc-gem_rights-manager“ zurückgegriffen, welche Zugriffsrechte auf die jeweilige Anwendung sowie Netzwerke, Domains und Gruppen des jeweiligen Nutzers zurückgibt. Die „itc-

gem_rights-manager“ API bildet somit eine Schnittstelle zwischen den Nutzerdaten und der Anwendung. Alle Netzwerke sind mit Gruppen der einzelnen RWTH Aachen Organisationen verknüpft, wodurch man einen Nutzer als Ansprechpartner der einzelnen Anwendungen bestimmen kann. **(siehe Abbildung 2)**

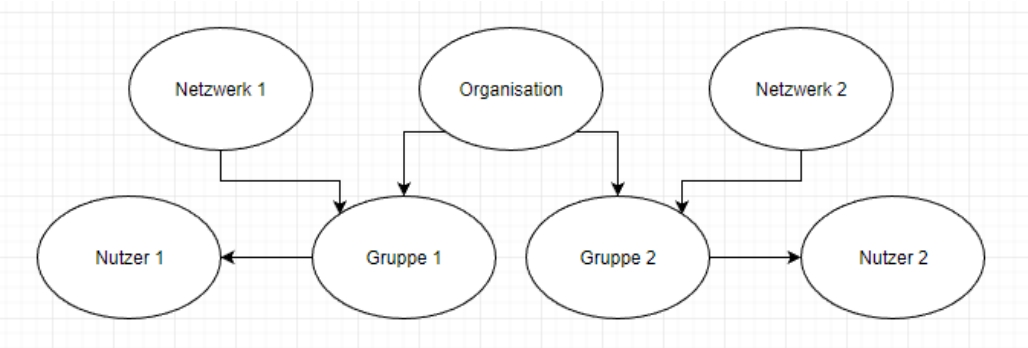


Abbildung 2: Rights-Manager Struktur einer RWTH Aachen Organisation

Durch diese Zugehörigkeit zwischen Ansprechpartner und Anwendung, kann gefiltert werden, welche Firewall-Regeln dem über Shibboleth angemeldeten Nutzer zugeordnet sind, um diese anzeigen zu können.

Somit sieht jeder Nutzer auf der ersten Ansichtsseite nur jene Firewall-Regeln, für die er als Ansprechpartner eingetragen ist. Für Personen, die eine Administrator-Berechtigung besitzen, gibt es eine zusätzliche Ansichtsseite, auf der alle Firewall-Regeln angezeigt werden.

5 Das Datenbankschema

Eine der wichtigsten Bestandteile einer Anwendung ist die Datenbank. In einem Datenbanksystem werden Daten, die für die Anwendung relevant sind, langfristig gespeichert. Sie bietet unter anderem auch einige Operationen an, wie zum Beispiel das Suchen eines bestimmten Datensatzes, das Einfügen neuer Datensätze und das Löschen bzw. Aktualisieren von bestehenden Datensätzen. [2]

Um Anforderungen und Struktur der Datenbank visuell darzustellen, wird ein sogenanntes Entity-Relationship-Modell genutzt. Hierbei handelt es sich um ein grafisches Modell zwischen Entitäten und deren Beziehungen, um eine Grundlage für die Entwicklung der Datenbank zu beschreiben. Eine Entität ist hier ein Objekt, das eindeutig identifiziert werden kann. Es besitzt Attribute, welche genaue Informationen über die Entität speichern. Die Beziehungen zwischen diesen Entitäten werden Assoziation genannt. Entitäten und Assoziationen werden durch Kanten miteinander verbunden, an denen die Kardinalitäten eingetragen werden, welche darstellen, wie viele Entitäten eines Entitätstyps mit genau einer Entität des anderen Entitätstyps in Beziehung stehen. [6]

Wie bereits in Kapitel 2 „Die Firewall und ihre Funktionen“ beschrieben, setzt sich eine Firewall-Regel aus mehreren Komponenten zusammen, welche in der Datenbank getrennt gespeichert werden wollen. Die wichtigsten Bestandteile einer solchen Regel sind der Absender, hier als „Source“ dargestellt, und der Empfänger, hier als „Destination“ dargestellt, sowie der genutzte Service und das Protokoll.

Die zwei Komponenten „Source“ und „Destination“ bestehen aus jeweils drei möglichen IP-Address Konstrukten, denn es kann sich bei der „Destination“ und bei der „Source“ um jeweils einen Host, eine IP-Address-Range oder ein ganzes Netzwerk handeln. Die verschiedenen Entitätstypen unterscheiden sich hier anhand der Anzahl der IP-Adressen, die dargestellt werden. Ein Host ist durch eine einzige IP-Adresse definiert, während eine IP-Address-Range und ein Netzwerk aus mehreren IP-Adressen bestehen. Hier wird auch zwischen IPv4 und IPv6 unterschieden.

Zusätzlich zu dem Empfänger und dem Absender wird eine Firewall-Regel durch den Service definiert, welcher darstellt, für welchen Zweck der

Absender die Verbindung zum Empfänger aufbaut. Dieser Service beinhaltet Daten zu dem genutzten Protokoll und dem verwendeten Port.

Um die einzelnen Firewall-Regeln darzustellen, werden in der Datenbank Kreuztabellen zwischen den zuvor beschriebenen Entitäten erzeugt.

Einen genauen visuellen Einblick in die Beziehungen des Datenbankschemas bietet hier das Entity-Relationship-Modell. **(siehe Abbildung 3)**

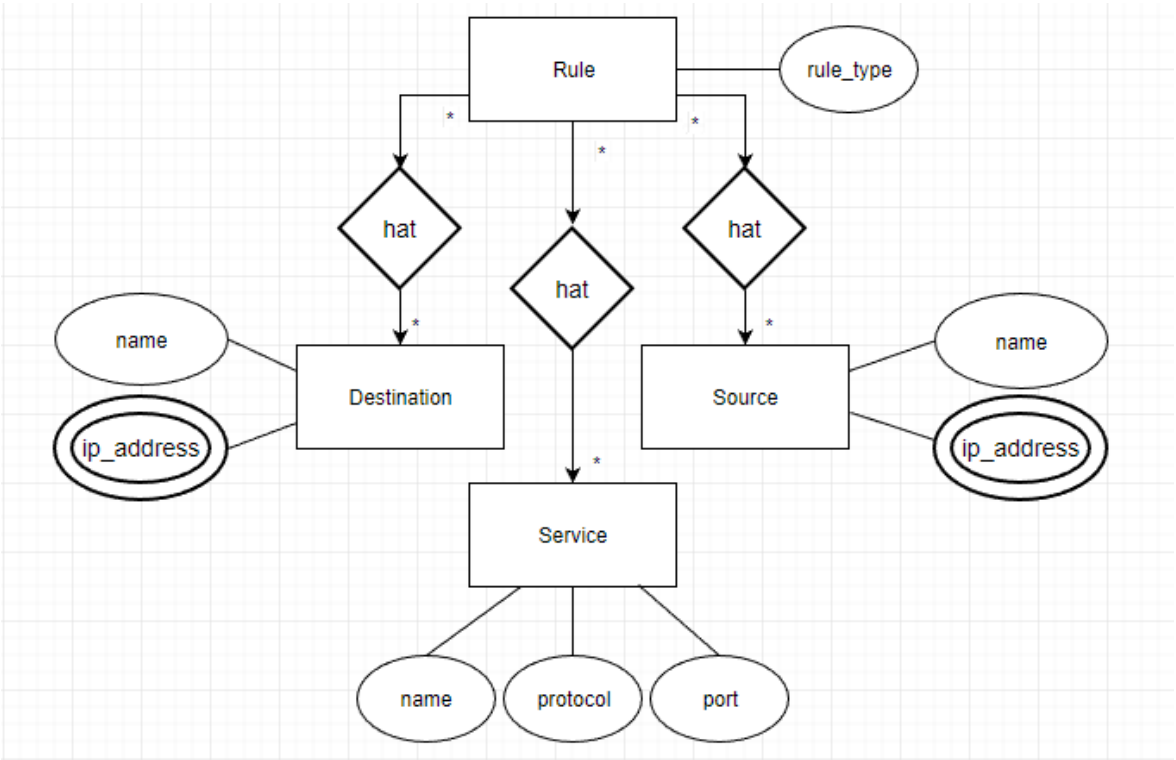


Abbildung 3: Entity-Relationship-Diagramm der Anwendung „Firewall-View“

6 Die Prozessphasen der Datenbeschaffung

Da das Grundgerüst der Anwendung geklärt ist, ist der nächste Schritt die Frage der Datenbeschaffung. Um die Daten zu bekommen, die in der Datenbank gespeichert werden sollen, wird der ETL-Prozess verwendet, da dieser die Datenbeschaffung in drei Prozessphasen einteilt. Die Abkürzung ETL steht für die drei englischen Wörter „Extract“, „Transform“ und „Load“ und jedes dieser Wörter bezeichnet einen Einzelschritt des Prozesses. **(siehe Abbildung 4)**

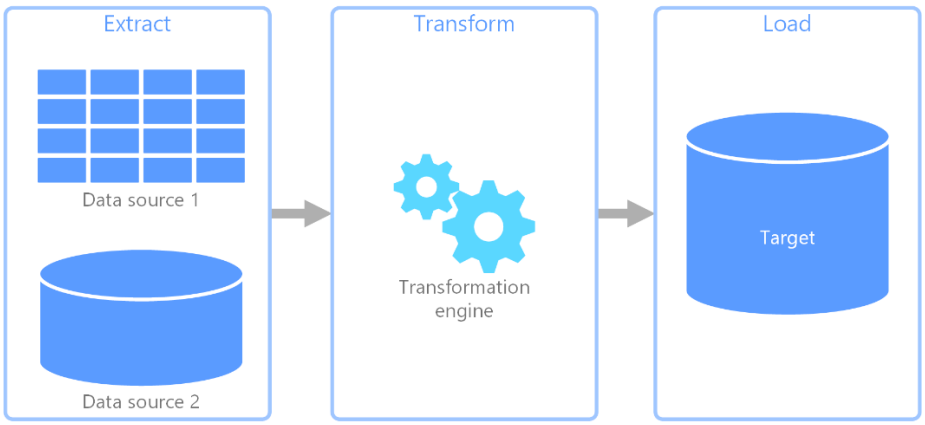


Abbildung 4: ETL-Prozess [7]

In der ersten Prozessphase „Extract“ werden Daten aus einer oder mehrerer Datenquellen extrahiert, um diese für den weiteren Verlauf bereitzustellen. Die zweite Prozessphase „Transform“ dient zur Prüfung und Transformation der extrahierten Daten in das Format und Schema der Zieldatenbank. Zuletzt werden die vorbereiteten Daten in der Prozessphase „Load“ in die Zieldatenbank übertragen. Durch den ETL-Prozess wird garantiert, dass die Daten in der Datenbank der Anwendung in einem geeigneten Format vorliegen. **[8]**

Für den ETL-Prozess wird in der Anwendung „Firewall-View“ ein Python-Skript genutzt. Dieses wird in regelmäßigen Abständen einmal täglich ausgeführt, um die Datenbank zu aktualisieren. Dadurch werden Änderungen an Firewall-Regeln und neue Firewall-Regeln innerhalb der RWTH Firewall stets in die Datenbank übertragen.

6.1 Die „Extract“- Phase

Da die Firewall-Regeln aus der Firewall der RWTH Aachen die einzigen Daten sind, die in der Zieldatenbank gespeichert werden sollen, müssen Daten nur aus dieser einzelnen Datenquelle extrahiert werden.

Zuerst muss eine Verbindung zu der Firewall aufgebaut werden. Dies geschieht mit Hilfe des Python Session Moduls, welches eine persistente Sitzung mit der Firewall Schnittstelle SMC-Python liefert.

Nach erfolgreicher Verbindung können die Firewall-Regeln von dem Python-Skript ausgelesen werden. Um genau bestimmen zu können, wie in der zweiten Phase des ETL-Prozesses mit dem Datensatz umgegangen werden soll, muss die Datenstruktur des extrahierten Datensatzes deutlich gemacht werden.

6.2 Die „Transform“- Phase

Die gewonnenen Daten bestehen aus miteinander verketteten Objekten, die verschiedenen Gruppen zugeordnet werden. Diese Gruppen müssen iterativ durchlaufen werden, um an die Firewall-Regeln zu kommen. Die Gruppen verweisen auf verschiedene Objektklassen, jedoch ist für die Anwendung „Firewall-View“ nur die Objektklasse „Rule“ relevant. In dieser Klasse werden die Firewall-Regeln dargestellt, die in der Zieldatenbank gespeichert werden sollen. Um diese Regeln in eine Form zu bringen, die mit der Zieldatenbank übereinstimmen und von der Anwendung genutzt werden können, müssen auf drei grundlegendende Probleme eingegangen werden, die bei der Transform-Phase vorkommen können.

6.2.1 Die „Schema-level“ Probleme

Die „Schema-level“ Probleme sind die offensichtlichsten Probleme. Sie beziehen sich auf strukturelle Widersprüche zwischen den gewonnenen Daten aus der ersten Phase und der Zieldatenbank der Anwendung. Außerdem gehören Namenskonflikte, wie zum Beispiel verschiedene Objekte mit dem gleichen Namen, zu dieser Problemklasse. [4]

Um die strukturellen Probleme zu verhindern, werden in dem Python-Skript mehrere Datenstrukturen angelegt, die in der Programmiersprache Python Dictionaries genannt werden. Sie speichern mehrere Key-Value Paare und können so dem Datensatz eine ähnliche Struktur geben wie in der Zieldatenbank. Für jede Tabelle der Datenbank wurde ein Dictionary erzeugt, in dem die gewonnen Daten eingetragen werden, um den späteren „Load“ Prozess zu vereinfachen. Zum Beispiel werden Destinations und Sources nach Host, Netzwerk und IP-Address-Ranges gegliedert und diese wiederum nach IPv4- und IPv6-Adressen. Das Dictionary, in dem Services gespeichert werden, wird nach Protokoll und Portnummer gegliedert. Diese Gliederungen innerhalb der Dictionaries entsprechen der Struktur der Zieldatenbank.

Beim iterativen Durchlaufen der Gruppen und derer Firewall-Regeln werden die gesuchten Informationen, die für die Anwendung relevant sind, in die vorhandenen Dictionaries gefüllt.

6.2.2 Die „Record-level“ Probleme

Die „Record-level“ Probleme beziehen sich auf widersprüchliche bzw. falsche Daten und Duplikate. Es muss verhindert werden, dass solche Daten in die Zieldatenbank überführt werden, um die Datenbank vollständig korrekt zu halten. [4]

Um widersprüchliche bzw. falsche Daten zu finden führt das Python-Skript viele Überprüfungen durch. Bevor die Informationen der Firewall-Regel in das passende Dictionary überführt werden, wird zum Beispiel jede Firewall-Regel auf ihre Korrektheit und Vollständigkeit überprüft, indem die Existenz der einzelnen Firewall-Regel Komponenten, die in die Datenbank überführt werden sollen, geprüft wird. Außerdem wird jedes Mal darauf geachtet keine

Duplikate in die Dictionaries aufzunehmen, auch wenn diese in der RWTH Firewall öfter vorkommen. Dieses Szenario kann durchaus auftreten, da mehrere Gruppen die gleiche Firewall-Regel zugeordnet haben können, was in der Struktur der RWTH Firewall Sinn macht, jedoch für Zieldatenbank und die Anwendung nicht.

6.2.3 Die „Value-level“ Probleme

Die dritte und letzte Problemklasse „Value-level“ Probleme bezieht sich auf unterschiedliche Wertedarstellungen oder verschiedene Interpretationen von Werten der Datenquelle. Um diese Art von Problemen zu verhindern, müssen die Daten normalisiert werden und entsprechend umgewandelt werden. **[4]**

Die Normalisierung der Daten findet vor dem Eintragen in die Dictionaries des Python-Skripts statt, damit diese Dictionaries vollständig normalisiert und einheitlich sind. Es spielt hier keine Rolle wie die Daten in der RWTH Firewall formatiert sind, da gesuchte Informationen immer gleich in die Dictionaries gespeichert werden, um „Value-level“ Probleme zu verhindern. Hosts, Netzwerke und IP-Address-Ranges haben stets als Key den Namen und als Value die IP-Adresse, welche als hexadezimal Wert dargestellt wird. Services haben auch als Key ihren Namen, jedoch als Value ein Tupel aus Protokollnamen, Protokollnummer und einer Portnummer, die entweder einen einzelnen Wert oder eine Reichweite angibt.

6.3 Die „Load“- Phase

Um nun die organisierten und vollständigen Datensätze, welche in den Dictionaries des Python-Skripts vorliegen, in die Zieldatenbank zu übertragen, wird die SQL Sprache verwendet. Die Abkürzung SQL steht für „Structured Query Language“. Mit SQL ist es möglich mit einer relationalen Datenbank zu kommunizieren und Datensätze zu verändern, einzufügen und zu löschen. **[9]**

Damit beim Einfügen der Daten keine Komplikationen entstehen, wie zum Beispiel Duplikate oder veraltete Datensätze in der Datenbank, werden zunächst alle bereits existierende Datensätze der Datenbank gelöscht. Das Löschen von Datensätzen wird in der SQL Sprache mit dem Befehl „DELETE“

umgesetzt. Um zu verhindern, dass beim produktiven Gebrauch der Anwendung „Firewall-View“ die Datenbank leer ist und kein Inhalt auf den Ansichtsseiten der Anwendung angezeigt werden kann, wird die Löschung der alten Datensätze und das Einfügen der neuen Datensätze direkt nacheinander ausgeführt. Damit zwischen diesen beiden Aktionen keine Zeit verloren geht, werden alle SQL-Statements zuvor gebildet und vorbereitet.

Das Hinzufügen neuer Datensätze in die Zieldatenbank wird über den SQL Befehl „INSERT“ erreicht. Beim Einfügen der Datensätze in die Zieldatenbank kann man auf zwei verschiedene Weisen vorgehen. Die erste und deutlich ineffizientere Weise ist es, jeden einzelnen Datensatz nacheinander in die Datenbank hinzuzufügen. Bei einer derart großen Menge an Daten kann dieses Verfahren sehr lange dauern, da für jeden Datensatz ein neuer Befehl an die Datenbank gesendet werden muss und die Daten einzeln eingetragen werden. Die zweite Möglichkeit eine große Datenmenge in eine Zieldatenbank einzusetzen, ist es, eine sogenannte Bulk-Insertion durchzuführen. Bei diesem Prozess werden größere Mengen an Daten auf einmal in eine Datenbank geladen, wodurch man die Schnelligkeit der Datenübertragung und die Anzahl an Zugriffen deutlich verbessert.

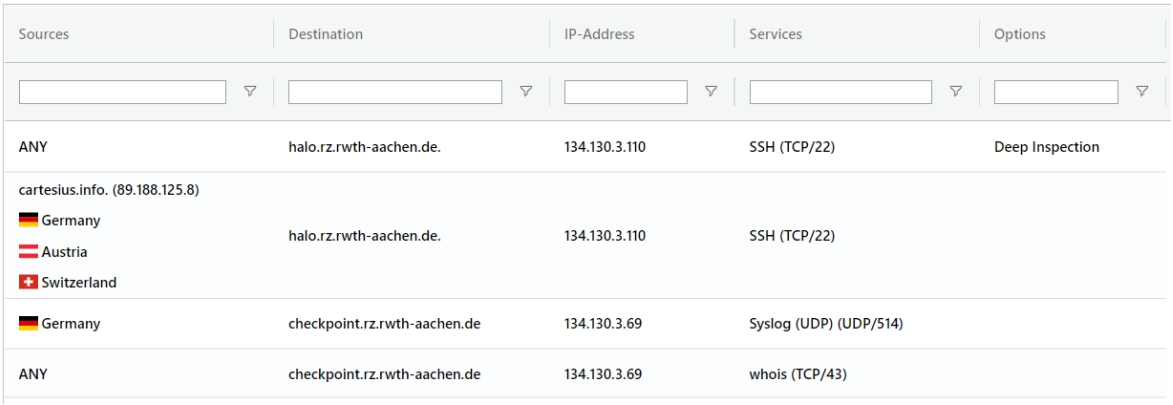
Das Python-Skript der Anwendung „Firewall-View“ verwendet Bulk-Insertion, um die Datenbank so schnell wie möglich zu füllen, damit die Anwendung bei einem produktiven Gebrauch nie funktionsunfähig ist. Zusätzlich wird das Python-Skript, welches zur Aktualisierung der Datenbank gedacht ist, nur nachts ausgeführt, da die Anwendung zu dieser Zeit höchstwahrscheinlich nicht genutzt wird.

Um die organisierten Daten für das Einfügen in die Datenbank vorzubereiten, müssen diese sortiert und mit Indizes versehen werden, da Daten in einer Datenbank geordnet und über eine Identifikationsnummer eindeutig gekennzeichnet werden. Diese Indizes werden beim Einfügen als Identifikationsnummer mitgegeben.

Beziehungen zwischen den Objekten einer Datenbank werden in der Anwendung „Firewall-View“ über Kreuztabellen organisiert, die jeweils die Identifikationsnummer der zwei in Verbindung stehender Objekte speichert. Damit auch diese Kreuztabellen erfolgreich eingetragen werden können, ist auch hier die Organisation der Identifikationsnummer innerhalb des Python-Skripts sehr wichtig.

7 Datentransformation zur Filterung der Daten

Wie in Kapitel 4 „Sicherheit“ bereits beschrieben, sollen auf der Ansichtsseite, die für normale Nutzer der Anwendung „Firewall-View“ vorgesehen ist, nur bestimmte Firewall-Regeln aus der Datenbank angezeigt werden. **(siehe Abbildung 5)**



Sources	Destination	IP-Address	Services	Options
ANY	halo.rz.rwth-aachen.de.	134.130.3.110	SSH (TCP/22)	Deep Inspection
cartesius.info. (89.188.125.8) Germany Austria Switzerland	halo.rz.rwth-aachen.de.	134.130.3.110	SSH (TCP/22)	
Germany	checkpoint.rz.rwth-aachen.de	134.130.3.69	Syslog (UDP) (UDP/514)	
ANY	checkpoint.rz.rwth-aachen.de	134.130.3.69	whois (TCP/43)	

Abbildung 5: Ansicht der Firewall-Regeln in „Firewall-View“

Um diese Filterung der Daten zu bewerkstelligen, kann man die Grundoperationen der relationalen Algebra verwenden. Die wichtigste mengentheoretische Operation für das Filtern der angezeigten Daten, ist die Projektionsfunktion. Diese vermindert die Inputdaten um einige Attributwerte. Es werden also aus einer großen Datenmenge nur bestimmte Daten projiziert, die gesucht werden. **[11]**

Mit dem SQL Befehl „SELECT“ können bestimmte Datensätze aus der Datenbank abgefragt werden. Dieser SQL Befehl entspricht der Projektionsfunktion der relationalen Algebra, welche schon vorher erwähnt wurde. **[12]**

Durch eine Verknüpfung der Berechtigungen eines Nutzers, die über die API „itc-gem_rights-manager“ erlangt wurden, und der vollständigen Daten der Datenbank können, unter Benutzung des SQL Befehls „SELECT“, genau die Daten gefiltert werden, die dem Nutzer angezeigt werden sollen.

8 Fazit und Ausblick

Die Anwendung „Firewall-View“ erweist sich als großer Vorteil für Institutsadministratoren der RWTH Aachen, da diesen eine effiziente Möglichkeit geboten wird, Firewall-Regeln, die für ihre Netze und Anwendungen konfiguriert wurden, einsehen zu können. Dadurch wird eine Organisation der Firewall-Regeln für die Institutsadministratoren vereinfacht und es können einfacher Fehler gefunden werden, falls eine gewollte Kommunikationsverbindung nicht durch die Firewall kommt.

Außerdem werden dem Nutzer durch die Datentransformation der Firewall-Regeln, nur bestimmte Informationen angezeigt, wodurch die Ansicht auf die Firewall-Regeln übersichtlicher und verständlicher ist.

Um den Umgang mit den Firewall-Regeln für Institutsadministratoren noch weiter zu verbessern, kann in der Zukunft die Anwendung noch weiter ausgebaut werden. Zum Beispiel könnte eine weitere Funktion der Anwendung „Firewall-View“, mit der Institutsadministratoren Anpassungen von Firewall-Regeln oder das Hinzufügen neuer Firewall-Regeln veranlassen könnten, ergänzt werden. Diese Funktion könnte direkt aus der Anwendung genutzt werden, um die Arbeitsweise mit der Firewall effizienter zu gestalten.

9 Quellenverzeichnis

[1] *„ISO/OSI Referenzmodell“*, (zuletzt aufgerufen: 13. Dezember, 2022). URL: https://dev-supp.de/netzwerk-anonymitaet/iso-osi-referenzmodell

[2] Prof. Dr. Karl Friedrich Gebhardt: *„Datenbanken“*, 10.10.2019. URL: https://wwwlehre.dhbw-stuttgart.de/~kfg/db/db.pdf

[3] Norbert Pohlmann: *„Firewall-Systeme“*, 5. Auflage, 2003.

[4] Panos Vassiliadis, Alkis Simitsis: *„Extract, Transformation, and Loading“*, In: *„Encyclopedia of Database Systems“*, 2009, S. 1095-1101.

[5] Patrick Schnabel: *„Netzwerktechnik-Fibel“*, 2004.

[6] Qing Li, Yu-Liu Chen: *„Modeling and Analysis of Enterprise and Information Systems“*, 16.04.2009.

[7] Raunak Jhawar, Zoiner Tejeda: *„Extract, transform, and load (ETL)“*, (zuletzt aufgerufen: 20. Dezember, 2022). URL: https://learn.microsoft.com/en-us/azure/architecture/data-guide/relational-data/etl

[8] Dipl.-Ing. (FH) Stefan Luber: *„Was ist ETL (Extract, Transform, Load)?“*, 16.11.2018. URL: https://www.bigdata-insider.de/was-ist-etl-extract-transform-load-a-776549/

[9] Dipl.-Ing. (FH) Stefan Luber: *„Was ist SQL?“*, 17.03.2017. URL: https://www.dev-insider.de/was-ist-sql-a-586264/

[10] *„The Shibboleth Project“,* (zuletzt aufgerufen: 15. Dezember, 2022). URL: https://www.shibboleth.net/about-us/the-shibboleth-project/

[11] Thomas Studer: *„Die relationale Algebra“*, In: *„Relationale Datenbanken“*, 2016, S. 41-68.

[12] Thomas Studer: *„SQL Abfragen“*, In: *„Relationale Datenbanken“*, 2016, S. 69-100.