Höhere Technische Bundeslehranstalt Wien 3, Rennweg
IT & Mechatronik

HTL Rennweg :: Rennweg 89b
A-1030 Wien :: Tel +43 1 24215-10 :: Fax DW 18

Diplomarbeit

Netzwerkanalyse eines Enterprise-Netzwerks (Argos)

ausgeführt an der
Höheren Abteilung für Informationstechnologie/Netzwerktechnik
der Höheren Technischen Lehranstalt Wien 3 Rennweg

im Schuljahr 2019/2020

durch

Lukas Bandion
Thomas Hilscher
Harald Moritz
Lorenz Stechauner

unter der Anleitung von

DI Christian Schöndorfer
DI August Hörandl

Wien, 20. Juni 2020

Kurzfassung

Durch die immer weiter fortschreitende Digitalisierung versuchen Schulen sowie kleine und mittlere Unternehmen (KMUs) ihre Netzwerke vor Angriffen und Fehlern zu schützen. Security Information and Event Management (SIEM) Systeme können das Netzwerk und dessen Komponenten überwachen, analysieren und die Administratoren dieser Netzwerke über Zwischenfälle informieren. Am Markt werden einige wenige SIEM Systeme angeboten. Die Kosten dieser Systeme können oft nur von großen Unternehmen getragen werden. Schulen und KMUs haben meist nicht die finanziellen Mittel, sich diese Systeme anzuschaffen.

Das Ziel dieser Diplomarbeit ist es zu evaluieren, inwiefern es möglich ist, mit möglichst geringem finanziellen Aufwand ein SIEM System zu entwickeln. Teil dessen ist auch die Entwicklung eines eigenen SIEM Systems und die Einbindung dieses Systems in ein Netzwerk. Dazu wird ein Testnetzwerk aufgesetzt, welches im Kern denselben Aufbau mit dem eines durchschnittlichen Unternehmensnetzwerkes teilt. Die Daten dieses Netzwerks werden durch das entwickelte System gesammelt, gespeichert, verarbeitet, analysiert, interpretiert und visualisiert.

Im Laufe dieser Diplomarbeit wurde herausgefunden, dass das Sammeln der wichtigsten Daten eines Netzwerks ohne finanzielle Mittel, aber mit intensivem Auseinandersetzen mit den jeweiligen Netzwerkgeräten möglich ist. Das automatisierte Interpretieren dieser Daten stellt eine größere Hürde dar. Einfache Angriffe können zwar erkannt werden, komplexere auch mit großem Zeitaufwand nicht. Bei jeder Verarbeitung von personenbezogenen Daten, wie u. a. in dieser Diplomarbeit, ist der Datenschutz, respektive zutreffende Datenschutzgesetze, zu beachten.

Aufgrund der vorliegenden Ergebnisse wird Schulen und KMUs nicht empfohlen, eigene SIEM Systeme zu entwickeln oder entwickeln zu lassen. Es ist auch anzumerken, dass teure, komplexe SIEM Systeme nur bedingten Mehrwert für Schulen und KMUs bieten, da die durch das SIEM System zur Verfügung gestellten Informationen in so kleinen Netzwerken nur geringen Nutzen haben und dadurch nicht die Kosten rechtfertigen.

Abstract

As digitalization continues to advance, schools, small and medium-sized enterprises (SMEs) are trying to protect their networks from attacks and errors. Security Information and Event Management (SIEM) systems can monitor and analyze the network and its components and inform the administrators of these networks about incidents. A few SIEM systems are available on the market. The cost of these systems can often only be borne by large companies. Schools and SMEs usually do not have the financial means to purchase these systems.

The aim of this diploma thesis is to evaluate the extent to which it is possible to develop a SIEM system with as little financial expenditure as possible. Part of this is the development of an own SIEM system and the integration of this system into a network. For this purpose, a test network is set up, which essentially shares the same structure with that of an average enterprise network. The data of this network is collected, saved, processed, analyzed, interpreted and visualized by the developed system.

In the course of this diploma thesis it was found that the collection of the most important data of a network is possible without any financial means, but with intensive examination of the respective network devices. The automated interpretation of this data represents a greater hurdle. Simple attacks can be detected, but more complex ones cannot within an reasonable amount of time. Whenever personal data is processed, such as in this diploma thesis, data protection or applicable data protection laws must be considered.

Based on the available results, schools and SMEs are not recommended to develop their own SIEM systems. It should also be noted that expensive, complex SIEM systems only offer limited added value for schools and SMEs, since the information provided by the SIEM system has only little use in such small networks and this does not justify the costs.

Ehrenwörtliche Erklärung

Ich erkläre an Eides statt, dass ich die individuelle Themenstellung selbstständig und ohne fremde Hilfe verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

Wien, am 20. Juni 2020

Lukas Bandion

Thomas Hilscher

Harald Moritz

Lorenz Stechauner

Danksagung

An dieser Stelle möchten wir uns bei allen Personen und Organisationen bedanken, die uns bei der Durchführung und Anfertigung dieser Diplomarbeit unterstützt haben. Allen voran geht der Dank an usere Lehrer und Betreuer Christian Schöndorfer und August Hörandl, ohne deren Hilfe und Anleitung diese Diplomarbeit nicht zustande gekommen wäre. Außerdem wurde durch sie veranlasst, dass uns die nötige Hardware für unser Beispiel-Netzwerk zur Verfüung gestellt wurde.

Weiters möchten wir uns bei Harry Neumayer und Philip Wölfel von der NTS Netzwerk Telekom Service AG bedanken, welche dankenswerterweise einen Workshop zum Thema SIEM-Systeme insbesondere *Splunk* mit uns durchgeführt haben. Dieser Workshop hat uns einen wichtigen ersten Einblick in die Funktionsweise eines SIEM-Systems gegeben.

Inhaltsverzeichnis

1 Ziele **1**
1.1 Einleitung . . . 1
1.2 Individuelle Zielsetzung . . . 1
1.3 Ergebnisse . . . 3

2 Diplomarbeit Argos **5**
2.1 Module von Argos . . . 5
2.2 Argos als SIEM . . . 6

3 SIEM-Lösungen **7**
3.1 Aufgaben eines SIEMs . . . 7
3.2 Splunk . . . 8
3.3 FortiSIEM . . . 10
3.4 Fazit . . . 12

4 Aufbau eines Beispiel-Netzwerks **13**
4.1 Einleitung zum Netzwerkaufbau . . . 13
4.2 Aufbau des Netzwerkes bei Argos . . . 13

5 Aggregieren von Logdaten **15**
5.1 Logging Übersicht . . . 15
5.2 Protokolle zur Logdatenübermittlung . . . 15
5.2.1 Syslog-Protokoll . . . 16
5.2.2 Simple Network Management Protocol . . . 18
5.3 Logging auf Cisco Geräten . . . 22
5.3.1 Logging auf einer Cisco ASA . . . 23
5.3.2 Logging auf einem Cisco WLC . . . 24
5.4 Logging auf Fortinet Geräten . . . 26
5.4.1 Logging auf einer FortiGate Firewall . . . 26
5.5 Konfiguration von Windows Server . . . 27
5.5.1 Einleitung zu Windows Server . . . 27
5.5.2 Windows-zu-Windows Log-Weiterleitung . . . 28
5.5.3 Windows-zu-Linux Log-Weiterleitung . . . 30

6 Deep Packet Inspection **33**
6.1 Einleitung zu Deep Packet Inspection . . . 33
6.2 Grundvoraussetzungen für DPI . . . 33
6.3 DPI auf Firewalls . . . 34

6.4 DPI auf einer Cisco ASA . . . 34
6.4.1 Einleitung zur Cisco ASA . . . 34
6.4.2 Inspection Engine der Cisco ASA . . . 35
6.4.3 DNS-Doctoring auf einer Cisco ASA . . . 35
6.5 DPI auf einer FortiGate . . . 36
6.5.1 Einleitung zur FortiGate . . . 36
6.5.2 SSL-Inspection auf einer FortiGate . . . 37
6.6 DPI in der Diplomarbeit . . . 37
6.6.1 Einleitung zu DPI in der Diplomarbeit . . . 37
6.6.2 Konfiguration eines Port-Mirrors auf einem HP Switch . . . 38
6.6.3 Konfiguration eines Port-Mirrors auf einem Cisco Switch . . . 39
6.6.4 DPI-Server in der Diplomarbeit . . . 39

7 Speichern von Logdaten 43
7.1 Einleitung in die Speicherung von Logdaten . . . 43
7.2 Optimierung von MongoDB . . . 44
7.2.1 Überblick zur Optimierung von MongoDB . . . 44
7.2.2 Indizes in MongoDB . . . 44
7.2.3 Aufrechterhalten der Datenbankverbindung . . . 45
7.3 Datenbankstruktur in der Diplomarbeit . . . 46
7.3.1 Collection der Rohdaten . . . 46
7.3.2 Collection der verarbeiteten Daten . . . 47
7.3.3 Collection der Anmeldungen (Logins) . . . 49
7.3.4 Collection der Zwischenfälle (Incidents) . . . 50
7.3.5 Collection der historischen DNS-Daten . . . 53
7.4 Collector . . . 57

8 Verarbeiten von Logdaten 59
8.1 Indexer . . . 59
8.2 Verarbeiten von Syslog-Nachrichten . . . 60
8.2.1 Verarbeiten von Cisco-Logs im Allgemeinen . . . 61
8.2.2 Verarbeiten von Cisco-ASA-Logs . . . 62
8.2.3 Verarbeiten von Fortinet-Logs . . . 72
8.2.4 Verarbeiten von Solarwinds-Logs (Windows) . . . 72
8.3 Verarbeiten von Cisco-WLC-Nachrichten in der Diplomarbeit . . . 74
8.4 Sniffer . . . 75
8.4.1 Einleitung zum Sniffer . . . 75
8.4.2 Der Entpackungsprozess des Sniffers . . . 76
8.4.3 Die Analyse eines DNS-Pakets . . . 77
8.4.4 Verarbeitung von Echtdaten einer Schulklasse . . . 78
8.5 Enricher . . . 79
8.5.1 Einleitung zum Enricher . . . 79
8.5.2 WebUntis Schnittstelle . . . 79
8.5.3 Verarbeitung von MAC-Adressen . . . 80
8.5.4 Auflösen von IP-Adressen zu Domain-Namen . . . 80
8.5.5 Geolokalisieren von IP-Adressen . . . 80

8.5.6 Erzeugung von Angriffen . . . 81
8.5.7 Portscan . . . 81
8.5.8 DNS-Tunnel . . . 82
8.5.9 DoS-Angriff . . . 82
8.5.10 Abnormales Loginverhalten . . . 83
8.5.11 Nicht erkennbare Anomalien . . . 84
8.6 Anonymizer . . . 85

9 Auswerten von Logdaten 87
9.1 Searcher . . . 87
9.2 Query-Language . . . 88
9.3 Dashboard . . . 89
9.3.1 Landing-Page . . . 90
9.4 Query-Language im Dashboard . . . 91
9.5 User-Pages . . . 92
9.5.1 Users . . . 92
9.5.2 User . . . 93
9.6 Incidents . . . 94
9.6.1 Incidents-Übersicht . . . 94
9.6.2 DNS-Tunnel-Incident . . . 95
9.6.3 Portscan-Incident . . . 96

10 Künstliche Intelligenz 97
10.1 Einleitung zu Künstlicher Intelligenz . . . 97
10.2 Unterschied zwischen Mensch und KI erkennen . . . 97
10.3 Machine Learning . . . 98
10.4 Neuronales Netzwerk . . . 98
10.4.1 Das Netzwerk . . . 98
10.4.2 Das Neuron . . . 99
10.4.3 Aktivierungsfunktionen . . . 99
10.4.4 Gewichte . . . 101
10.4.5 Optimizer . . . 101
10.5 Deep Learning . . . 102
10.6 Inputdaten . . . 102
10.6.1 Inputdatentypen . . . 102
10.6.2 Die Intervallskala . . . 102
10.6.3 Die Ordinalskala . . . 103
10.6.4 Die Nominalskala . . . 103
10.6.5 One-Hot-Codierung . . . 103
10.7 Arten von Problemstellungen . . . 104
10.7.1 Klassifikation . . . 104
10.7.2 Clustering . . . 104
10.8 Verschiedene KI-Systeme . . . 105
10.8.1 Supervised Learning . . . 105
10.8.2 Unsupervised Learning . . . 105
10.8.3 Reinforcement Learning . . . 105

10.9 Deep Learning mit einem IDS/IPS . . . 106
10.9.1 TensorFlow . . . 106
10.9.2 Multiseries Forecasting . . . 106
10.9.3 Time Series Forecasting . . . 112
10.9.4 Ergebnis . . . 116
10.10 Fazit . . . 117
10.10.1 Künstliche Intelligenz in der Netzwerktechnik . . . 117
10.10.2 Unterschied zwischen Theorie und Praxis . . . 117
10.10.3 Modell der Diplomarbeit . . . 118
10.10.4 Rechenleistung und Dauer . . . 118
10.10.5 Datenmenge und Inputdaten . . . 118
10.10.6 Clustering . . . 118

11 Datenschutz 119
11.1 Datenschutz in Österreich . . . 119
11.2 Grundsätze der DSGVO . . . 120
11.3 Vorgehensweise in der Diplomarbeit . . . 121

Abbildungsverzeichnis 123

Abkürzungsverzeichnis 125

Glossar 131

Literaturverzeichnis 133

1 Ziele

1.1 Einleitung

Mit zunehmender Größe von Netzwerken wird es für den Administrator dieser immer schwieriger, den Überblick über den Datenverkehr zu behalten. Dadurch ist es für Benutzer innerhalb und außerhalb des Netzes bzw. Angreifern einfacher möglich, unerlaubterweise Unternehmensrichtlinien zu umgehen bzw. dem Netzwerk zu schaden. Betreiber von Netzwerken mit hohen Sicherheitsanforderungen beanspruchen oft Dienstleistungen von diversen Security-Consulting-Unternehmen, um dieser Probleme Herr zu werden. Diese sind meistens mit hohem finanziellem Aufwand verbunden. Vor allem die HTL Rennweg sieht sich aufgrund ihres IT-Schwerpunktes massiv mit diesen Problemen konfrontiert, doch für Schulen ist der enorme finanzielle Aufwand in den meisten Fällen nicht tragbar.

Das Ziel des Projektes ist es zu verstehen, wie solche Richtlinienumgehungen bzw. Angriffe für den Administrator überprüfbar gemacht und im besten Fall grafisch dargestellt werden könnten. Darunter fällt unter anderem das Zurückverfolgen einer Verbindung auf einen gewissen Benutzer oder Computer. Hierbei gilt es natürlich die DSGVO zu beachten. Weiters ist es für Administratoren praktisch das Datenaufkommen pro Benutzer oder Raum festzustellen oder die meistbesuchte Website eines Tages einsehen zu können. Um all dies zu bewerkstelligen, benötigt man auswertbare Daten. Um an diese Daten zu gelangen, werden Logdaten von diversen Intermediate-Devices (wie z. B. Firewalls, Router, WLAN-Controller, etc.) gesammelt.

1.2 Individuelle Zielsetzung

Da es bei einer Diplomarbeit wichtig ist, dass jedes Ziel einem Teammitglied zugeordnet ist, sind unterhalb alle Ziele aufgelistet, wie sie im Antrag der Diplomarbeit formuliert sind, und das jeweils zugeordnete Teammitglied zu finden.

Sammlung der Logdaten (Ziel H1, Stechauner)
: Es wird eine geeignete Syslog-Server-Lösung ausgewählt und aufgesetzt, welche Syslog-Nachrichten von diversen Geräten sammelt und an einem zentralisierten Ort speichert (siehe Kapitel 7 auf Seite 43).

Implementierung einer Packet Capture Lösung (Ziel H2, Bandion)
Ein Programm wird erstellt, welches Pakete aus einem Netzwerk (z. B. Port-Mirror auf einem Switch) mittels Deep Packet Inspection verarbeitet und gegebenenfalls an eine Speicherlösung weiterleitet (siehe Kapitel 8.4 auf Seite 75).

Aufbereitung der ASA Logdaten (Ziel H3, Moritz)
Ein Programm wird erstellt, welches bereits gesammelte Logdaten einer Cisco ASA (Firewall) entgegennimmt und aus diesen Daten technisch nicht relevante Informationen entfernt. Dies ist notwendig, da Logs von Cisco darauf ausgelegt sind, von einem Menschen gelesen zu werden, wodurch die Datenspeicherung und -verarbeitung erschwert wird (siehe Kapitel 8.2.2 auf Seite 62).

Speicherung und einfache Verarbeitung der Logdaten (Ziel H4, Stechauner)
Eine für alle anfallenden Logdaten geeignete Speicherlösung wird ausgewählt und implementiert. Die Speicherlösung muss auf Datenmengen, die in Netzen, wie z. B. einem EDV-Saal der Schule anfallen, ausgelegt sein, ohne die Leistungsfähigkeit zu vermindern. Die Daten sollen in eine für möglichst simple Abfragen geeignete Form (z. B. JSON) gebracht und gespeichert werden (siehe Kapitel 7 auf Seite 43).

Query Language (Ziel H5, Stechauner)
Es wird ein Konzept für eine Query Language ausgearbeitet und implementiert. So soll es möglich werden, mit einfachen Abfragen die gespeicherten Daten in passender Form auszugeben (siehe Kapitel 9 auf Seite 87).

Visualisierung der Daten (Ziel H6, Hilscher)
Daten, die mithilfe der Query Language abgerufen werden, werden in einem Webinterface grafisch dargestellt, um dem Administrator einen einfachen und schnellen Überblick über die gespeicherten Daten zu ermöglichen (siehe Kapitel 9.3 auf Seite 89).

Ergänzen der Daten (Ziel H7, Bandion)
Ein Programm wird erstellt, welches auf Anfrage hin die Daten mit Informationen aus externen, u. a. öffentlich zugänglichen Datenquellen (z. B. MAC-Adresse wird in Hersteller übersetzt) ergänzt (siehe Kapitel 8.5 auf Seite 79).

Datenschutzkonforme Behandlung von Nutzerdaten (Ziel H8, Hilscher)
Es werden Überlegungen zur datenschutzkonformen Behandlung der Nutzerdaten betreffend der DSGVO getroffen, welche rechtlichen Probleme dadurch entstehen und wie diese für dieses Projekt gelöst werden könnten. Diese Lösungen werden in allen diesbezüglich relevanten Programmen umgesetzt (siehe Kapitel 11 auf Seite 119).

Vergleichen verschiedener SIEM-Lösungen (Ziel H9, Hilscher)
Es werden verschiedene, bereits bestehende SIEM Systeme miteinander verglichen und evaluiert, welches am besten für den Anwendungsfall dieses Projektes geeignet wäre (siehe Kapitel 3 auf Seite 7).

Evaluierung von Deep Packet Inspection auf Firewalls (Ziel H10, Bandion)
Es wird überprüft, ob der Einsatz von Deep Packet Inspection auf Firewalls ausreichend sinnvolle Informationen liefern kann. Zusätzlich wird auch überprüft, ob und wie man verschlüsselte Verbindungen (TLS, SSL, ...) mitverfolgen kann (siehe Kapitel 6 auf Seite 33).

Evaluierung des Einsatzes von Deep Learning (Ziel H11, Moritz)
Es wird überprüft, ob und wie der Einsatz von Deep Learning bzw. Machine Learning dabei helfen kann, aussagekräftige Voraussagen über den Netzwerkverkehr treffen zu können (siehe Kapitel 10.5 auf Seite 102).

Vergleichen verschiedener KI-Systeme (Ziel H12, Moritz)
Mögliche Deep Learning Modelle (z. B. Supervised, Unsupervised) werden verglichen, und es wird für das Projekt ein geeignetes Modell evaluiert (siehe Kapitel 10.8 auf Seite 105).

1.3 Ergebnisse

Das Endprodukt der Arbeit ist ein zentrales Dashboard (siehe Kapitel 9.3 auf Seite 89), welches den Status eines Netzwerks anhand von empfangenen Logdaten darstellt. Weiters werden auch alle gesammelten Logdaten in Form eines Datenbank-Exports zur Verfügung gestellt. Diese Exporte sollen es weiteren Arbeiten ermöglichen, sich auf die Verarbeitung und Darstellung der Logdaten zu konzentrieren, ohne im Vorhinein selbst Logdaten im großen Stil zu sammeln. Zusammengefasst sind die Ergebnisse dieser Diplomarbeit:

- dieses Diplomarbeitsbuch selbst beinhaltet theoretische und praktische Ausarbeitungen zu allen beschriebenen Bereichen,
- das Git-Repository `Log-Manager`: beinhaltet fünf der sechs Module (siehe Kapitel 2.1 auf Seite 5):

  1. Collector (siehe Kapitel 7.4 auf Seite 57)
  2. Indexer (siehe Kapitel 8.1 auf Seite 59)
  3. Sniffer (siehe Kapitel 8.4 auf Seite 75)
  4. Enricher (siehe Kapitel 8.5 auf Seite 79)
  5. Searcher (siehe Kapitel 9.1 auf Seite 87)

- das Git-Repository `Dashboard`: beinhaltet das letzte der sechs Module in Form einer PHP-Web-Application (siehe Kapitel 9.3 auf Seite 89),

- Datenbank-Exporte der folgenden Collections:
  - `raw` (siehe Kapitel 7.3.1 auf Seite 46)
  - `logs` (siehe Kapitel 7.3.2 auf Seite 47)
  - `logins` (siehe Kapitel 7.3.3 auf Seite 49)
  - `incidents` (siehe Kapitel 7.3.4 auf Seite 50)
  - `dns_raw` (siehe Kapitel 7.3.5.1 auf Seite 53)
  - `dns` (siehe Kapitel 7.3.5.2 auf Seite 53)
  - `dns_records` (siehe Kapitel 7.3.5.3 auf Seite 56)

2 Diplomarbeit Argos

2.1 Module von Argos

Zur Erhöhung der Übersichtlichkeit wird die Diplomarbeit in sechs Module unterteilt. Jedes Modul ist selbst programmiert und quasi unabhängig von den anderen Modulen. Dadurch, dass die Module in der Programmiersprache Python geschrieben sind, ist eine schnelle und einfache Änderung bzw. Ergänzung der Module leicht möglich. Diverse Abhängigkeiten der Module untereinander sind in Abbildung 2.1 zu sehen. Auf der nächsten Seite ist eine kurze Auflistung und Erklärung der einzelnen Module zu finden.

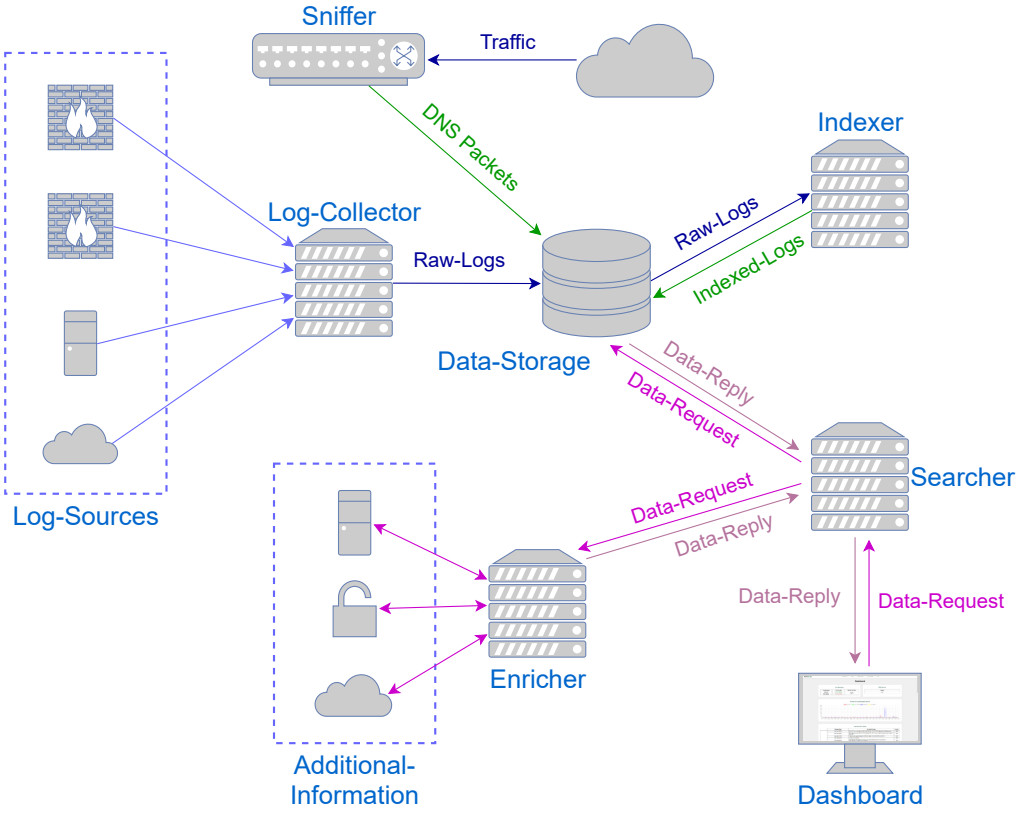

Abbildung 2.1: Modul-Architektur bei Argos

Collector (siehe Kapitel 7.4 auf Seite 57)
Der *Collector* hat die Aufgabe Syslog- und SNMP-Trap-Nachrichten zu empfangen und zu speichern. Dieses Modul hat eine ungefähre Länge von 80 Python-Codezeilen.

Indexer (siehe Kapitel 8.1 auf Seite 59)
Der *Indexer* hat die Aufgabe die gespeicherten Logdaten in ein maschinenlesbares Format umzuwandeln. Dieses Modul hat eine ungefähre Länge von 750 Python-Codezeilen.

Sniffer (siehe Kapitel 8.4 auf Seite 75)
Der *Sniffer* hat die Aufgabe DNS-Pakete auszulesen, zu verarbeiten und zu speichern. Dieses Modul hat eine ungefähre Länge von 280 Python-Codezeilen

Enricher (siehe Kapitel 8.5 auf Seite 79)
Der *Enricher* wird dazu verwendet, bereits gewonnene Informationen durch zusätzliche Informationsquellen aussagekräftiger zu gestalten. Dieses Modul hat eine ungefähre Länge von 80 Python-Codezeilen

Searcher (siehe Kapitel 9.1 auf Seite 87)
Der *Searcher* bietet ein einfaches Datenbank-Interface für das Dashboard. Dieses Modul hat eine ungefähre Länge von 850 Python-Codezeilen.

Dashboard (siehe Kapitel 9.3 auf Seite 89)
Im Dashboard werden alle gewonnenen Informationen zentral und übersichtlich dargestellt.

2.2 Argos als SIEM

Seit Beginn der Diplomarbeit war klar, dass ähnliche Systeme bereits von verschiedensten Unternehmen entwickelt wurden. Es stellte sich durch verschiedenste Recherchen heraus, dass alle diese Systeme sogenannte SIEMs sind und die Diplomarbeit also auch zu einem gewissen Maße in diese Kategorie einzuordnen ist. Daher war es sinnvoll, sich einmal anzusehen wie sich so ein Security Information and Event Management (SIEM) eigentlich definiert und wie so ein industrieübliches System aufgebaut ist. Weiters war es interessant, den Funktionsumfang der bereits bestehenden SIEM Systeme genauer zu betrachten und zu analysieren, welche Parallelen diese mit der Diplomarbeit genau haben.

Deshalb wird im Kapitel 3 auf der nächsten Seite genauer darauf eingegangen, welche Aufgaben ein SIEM hat und weiters werden zwei bereits bestehende SIEM Systeme genauer erklärt.

3 SIEM-Lösungen

3.1 Aufgaben eines SIEMs

Ein SIEM ist ein System, das die beiden Konzepte Security Information Management (SIM) und Security Event Management (SEM) kombiniert. Diese Konzepte werden von Unternehmen zu Unternehmen unterschiedlich verstanden, haben aber immer gewisse Gemeinsamkeiten. Grundsätzlich handelt es sich immer um die Analyse des Netzwerks im Bezug auf dessen Sicherheit und Angriffe, die auf es stattfinden.

Die Begriffe SIM, SEM und SIEM noch einmal etwas genauer erläutert:

Security Information Management (SIM)
Der Begriff Security Information Management (SIM) entspricht grundsätzlich dem *Log Management*. Bei diesem handelt es sich um das zentrale Sammeln, Übertragen, Speichern, Analysieren und Weiterleiten von Logdaten, die von Netzwerkgeräten stammen. Mit diesen Daten werden im Zuge des SIM meist Analysen zu Trends durchgeführt oder Berichte zu den Funktionen des Netzwerks erstellt. [11]

Security Event Management (SEM)
Das Security Event Management (SEM) befasst sich mit dem Korrelieren der Logdaten und im Unternehmen definierten Richtlinien wie IT Infrastructure Library (ITIL), Control Objectives for Information and Related Technologies (COBIT), Sarbanes-Oxley Act (SOX) oder Internationale Organisation für Normung (ISO). Meist steht eine Art Echtzeit-Alarmfunktion zur Verfügung, die im Fall eines Richtlinienverstoßes Alarm schlägt. Teilweise deckt sich der Funktionsumfang von SEM auch mit den klassischen IDS und IPS Systemen. Es überwacht daher die Auslastung und Kommunikation innerhalb des Netzwerks auf bestimmte Muster bzw. Abweichungen im Netzwerkverkehr. [11]

Security Information and Event Management (SIEM)
Security Information and Event Management (SIEM) kombiniert die Funktionen des SIM und SEM in einer einzigen Management-Lösung. Diese ist meist sehr unternehmensspezifisch gestaltet. In dem Sinne, dass für unterschiedliche Unternehmen verschiedene Ereignisse im Netzwerk verschieden wichtig sind. [11]

Um solch ein SIEM System nun selbst zu implementieren, war es von Vorteil sich anzusehen, wie Hersteller von bereits bestehenden SIEM-Lösungen ihre Produkte aufgebaut haben. Dafür wurde in Zusammenarbeit mit der *NTS Netzwerk Telekom Service AG* ein Workshop abgehalten. In diesem wurde dem Diplomarbeitsteam der Aufbau und die Funktionsweise von Splunk Enterprise, der SIEM-Lösung der Firma Splunk, näher gebracht. Im Zuge dieses Workshops wurden außerdem die Module von Argos (siehe Kapitel 2.1 auf Seite 5) erarbeitet, die dem Aufbau von Splunk (siehe Kapitel 3.2) sehr ähneln.

Nach genauerer Überlegung, welche Anforderungen so ein SIEM System eigentlich erfüllen muss, sind immer wieder zwei wichtige Kriterien aufgekommen:

- Effizienz
- Skalierbarkeit

Einerseits muss ein SIEM System sehr effizient sein. In größeren Netzwerken passiert es sehr schnell, dass hunderte Log-Nachrichten pro Sekunde generiert werden und daraufhin auch verarbeitet werden müssen. Ein solches System muss also mit dieser Flut an Nachrichten umgehen können, ohne Leistungseinbußen zu vermerken. Dies sollte natürlich mit einem möglichst geringem Aufwand an Hardware möglich sein, um die Kosten des ganzen Systems niedrig zu halten.

Andererseits muss ein solches System aber auch so konzipiert sein, dass wenn neue Hardware angeschafft werden muss, dies möglichst einfach geschehen kann. Falls es notwendig ist, das momentane System auszubauen, sollte es möglich sein, die zusätzlichen Ressourcen möglichst einfach zu integrieren. Weiters sollten die hinzugefügten Ressourcen mit der gleichen Effizienz arbeiten wie die initiale Konfiguration und nicht an Leistung einbüßen.

3.2 Splunk

Splunk ist eine Plattform, die es erlaubt, Log-Nachrichten und Monitoring-Daten zu sammeln und daraus individuell angepasste Berichte zu erstellen. Dabei beschreibt sich Splunk als *The Data-to-Everything Platform* und möchte damit ausdrücken, dass Daten von allen möglichen Quellen gesammelt und analysiert werden können. Dies erreicht Splunk damit, dass die Verarbeitung von speziellen Nachrichten vom Nutzer selbst programmierbar sind und dieser das System daher immer auf seine eigenen Wünsche anpassen kann.

Splunk als Plattform lässt sich grob in drei verschiedene Produkte einteilen:

Splunk Enterprise
Splunk Enterprise ist die Lösung von Splunk, die auf große Unternehmen, mit einer sehr umfassenden Informationstechnologie (IT), abzielt. Sie kann einfach auf verschiedensten Betriebssystemen, wie Windows, Linux oder MacOS, installiert werden und ist in der Lage Daten von verschiedensten Quellen zu sammeln. Einige Beispiele für solche Datenquellen wären Netzwerkgeräte, Serversysteme oder andere Sensoren, die angesteuert werden können.

Splunk Cloud
Splunk Cloud ist im Grunde genommen eine in der Cloud betriebene Version von Splunk Enterprise. Sie kann entweder von Splunk selbst oder auf der Amazon Web Services (AWS) Cloud-Plattform betrieben werden.

Splunk Light
Splunk Light erlaubt es die grundlegenden Funktionalitäten, wie das Speichern von Logdaten und das Suchen in diesen, von Splunk Enterprise zu nutzen, schränkt die erweiterten Funktionalitäten aber ein.

Speziell für den Bereich *Security* gibt es für Splunk Enterprise bzw. Splunk Cloud eine Erweiterung, die sich speziell mit der Erkennung, der Analyse und dem Blockieren von Angriffen auf das Netzwerk befasst. Diese nennt sich Splunk Enterprise Security (ES) und erzwingt das Verwenden des Splunk eigenen Splunk Common Information Model (CIM). Dieses erlaubt es, die erhaltenen Logdaten so weit wie möglich zu vereinfachen und zu normalisieren, was die weitere Verarbeitung einfacher gestaltet.

Die wichtigsten Funktionen von Splunk Enterprise waren natürlich auch für die Verarbeitung der Daten in der Diplomarbeit interessant und lassen sich grob in sechs verschiedene Bereiche einteilen:

Daten einspielen
Beim Einspielen der Daten ist es wichtig, möglichst viele verschiedene Typen von Daten verarbeiten zu können. So erlaubt es Splunk Enterprise, dass verschiedenst strukturierte Formate, wie Extensible Markup Language (XML) oder JavaScript Object Notation (JSON) verwendet werden können. Es können aber auch Daten verarbeitet werden, die nicht so klar strukturiert sind. Beispiele für diese wären verschiedenste Arten von Logdaten.

Daten modellieren
Die Daten, die in das System eingespielt wurden, können in Form von verschiedensten Modellen dargestellt werden, um auf bestimmte Anforderungen genauer eingehen zu können. So können in Splunk Enterprise vom Nutzer verschiedenste Modelle für die Daten erstellt werden oder von Splunk vorgefertigte verwendet

werden. Modelle erleichtern die spätere Analyse und Auswertung der Daten, da diese dadurch für Mensch und Maschine einfacher zu verarbeiten werden.

Daten indizieren

Um die Daten möglichst schnell und effizient zu durchsuchen, ist es sinnvoll Indizes anzulegen. So werden eingespielte Daten in Splunk Enterprise mit einem Index versehen, damit das spätere Weiterverarbeiten schneller erfolgen kann.

Daten suchen

Nach den zuvor gespeicherten und indizierten Daten muss gesucht werden. Splunk Enterprise ermöglicht dies über die Splunk eigene Search Processing Language (SPL), die es erlaubt, diverse Suchen durchzuführen. Die Daten können mittels dieser auch in verschiedenste Formen, wie Tabellen oder unterschiedlichen Diagrammen, dargestellt werden.

Dashboards

Splunk Enterprise ermöglicht es nun, aus vom Nutzer beschriebenen Suchen, *Dashboards* zu erstellen. Diese können vom Nutzer komplett selbst gestaltet werden, um genau die Informationen darzustellen, die für den Anwendungsfall wichtig sind.

Alerts

Sollte es im Netzwerk zu einer Anomalie kommen, die Splunk Enterprise durch das Verarbeiten der Daten erkennt, sollte der Nutzer direkt kontaktiert werden. Hier werden die gespeicherten Suchen verwendet, um das abnormale Verhalten zu erkennen.

3.3 FortiSIEM

Fortinet, die Firma hinter FortiSIEM, begann ihre Geschichte mit dem Entwickeln von Firewall-Lösungen. Danach wurde das Produktportfolio immer weiter vergrößert, vor allem im Bereich der Sicherheitslösungen und unter anderem eine SIEM-Lösung hinzugefügt. Dadurch, dass Fortinet selbst Firewalls und andere Netzwerkgeräte herstellt, ist die Integration dieser natürlich sehr gut. FortiSIEM ist aber auch in der Lage, mit Geräten von anderen Herstellern zu arbeiten.

FortiSIEM kann auf verschiedene Weisen aufgesetzt werden. So kann es für kleine bis mittlere Unternehmen in Form von einer kombinierten Hardware bzw. Virtual Machine (VM)-Lösung installiert werden. Um in größeren Unternehmen sehr hohe Datenmengen verarbeiten zu können, kann FortiSIEM auch in seine einzelnen Komponenten zerteilt werden:

Collector

Der *Collector* ist dafür zuständig, das Ziel für die Logdaten der Netzwerkgeräte zu sein. Es können mehrere *Collectors* auf verschiedene physische Standorte verteilt werden, damit diese direkt auf ihrem Standort weiterverarbeitet werden können. Der *Collector* sammelt, analysiert und komprimiert die Logdaten. Diese Daten werden dann über eine Hypertext Transfer Protocol (HTTP) oder Hypertext Transfer Protocol Secure (HTTPS) Verbindung an die *Worker* und *Supervisor* Geräte gesendet. Dabei werden die Daten gleichmäßig über diese verteilt.

Worker und Supervisor

Die *Worker* und *Supervisor* sind die Geräte, die im *Datacenter* betrieben werden und die eigentliche Analyse der Logdaten vornehmen. Dabei wird die Analyse in mehrere Unterschritte geteilt, die entweder von *Worker* Geräten oder von *Supervisor* Geräten durchgeführt werden. Dies wird getan, um die Skalierbarkeit des ganzen Systems sicherzustellen, indem flexibel neue Geräte hinzugefügt werden können.

Ein weiteres großes Feature von FortiSIEM ist die automatische Geräteerkennung. So erkennt FortiSIEM von alleine, um welches Netzwerkgerät es sich bei der Log-Quelle handelt und kann diese dann entsprechend verarbeiten. Die Informationen beschränken sich aber nicht nur darauf, um welches Gerät es sich handelt, sondern es kann auch erkannt werden, um welche Betriebssystemversion oder um welche Hardware es sich handelt. Diese Informationen werden dann in der Configuration Management Database (CMDB) gespeichert, in der es dem Nutzer dann möglich ist, die verschiedensten Geräte zu gruppieren.

Weiters ist es bei FortiSIEM, ähnlich wie bei Splunk Enterprise (siehe Kapitel 3.2 auf Seite 8), möglich, verschiedenste *Dashboards* zu erstellen.

So stehen grundsätzlich fünf verschiedene Typen von *Dasboards* zur Verfügung:

Summary Dashboards

Hier werden fast in Echtzeit wichtige Leistungsmetriken des Netzwerks, wie die Betriebszeit oder Vorfälle, angezeigt. Dieses *Dashboard* bietet einen generellen Überblick über den momentanen Status des Netzwerks und kann vom Nutzer individuell gestaltet werden. So kann dieser beispielsweise konfigurieren, welche Geräte und Metriken hier gezeigt werden sollen.

Widget Dashboards

Dieser *Dashboard* Typ bietet einen eher klassischen Überblick über eine einzelne Metrik. Die Metriken werden in Form von verschiedensten Graphen dargestellt.

Business Service Dashboards

Hier werden Informationen zu momentanen Vorfällen im Netzwerk angezeigt, die direkt die Tätigkeit des Unternehmens gefährden könnten. Von diesen Vorfällen

kann dann genau auf ein einzelnes Gerät zurückverfolgt werden, wo dieser aufgetreten ist.

Identity and Location Dashboards
In diesem *Dashboard* kann genau aufgeschlüsselt werden, welche Benutzer momentan im Netzwerk angemeldet bzw. aktiv sind.

Incident Dashboards
Hier können alle Vorfälle im Netzwerk angezeigt und nach dem dargestellten Risiko sortiert werden. Weiters können auch einzelne Geräte und Benutzer nach dem von ihnen ausgehenden Risiko sortiert werden.

3.4 Fazit

Alle SIEM Systeme haben grundsätzlich sehr ähnliche Features. Sie können alle, von Haus aus, sehr viele unterschiedliche Typen von Logdaten, von verschiedensten Herstellern, verarbeiten. Für die Verarbeitung besteht meist die Möglichkeit, diese auf mehrere physische Geräte zu verteilen und eine Art Loadbalancing zu betreiben. Weiters gibt es immer viele Arten von *Dashboards*, die es den Nutzern ermöglichen, genau ihre wichtigsten Komponenten zu überwachen. Dazu kommt noch eine Form von *Query Language* (siehe Kapitel 9.2 auf Seite 88), die es erlaubt, eigene Abfragen zu erstellen und diese in die *Dashboards* einzubinden. Dazu kommen meist Alarmsysteme, die den Nutzer alarmieren, falls eine Metrik einen der konfigurierten Schwellenwerte überschreitet.

Für eine genaue Analyse, wann welche SIEM-Lösung eine bessere Wahl wäre, müssten die einzelnen Lösungen aufgesetzt und dann evaluiert werden. Die grundlegende Funktionalität, wie sie die Diplomarbeit Argos bietet, ist bei beiden SIEM Systemen gegeben. Daher kommt es bei diesem Anwendungsfall weniger darauf an, welche zusätzlichen Funktionen noch zur Verfügung stehen, sondern wie es um den finanziellen Aspekt steht bzw. zu welchem Unternehmen bereits Kontakte bestehen.

4 Aufbau eines Beispiel-Netzwerks

4.1 Einleitung zum Netzwerkaufbau

Viele moderne Netzwerke haben einen ähnlichen logischen Aufbau wie in Abbildung 4.1 zu sehen ist. Dabei ist unschwer zu erkennen, dass die einzige Verbindung zwischen den Routern und den ISPs über die Firewall führt. Das entspricht auch dem IT-Grundsatz der Angriffsflächenminimierung. Je weniger Geräte das Firmennetzwerk nach außen hin dem Internet Service Provider (ISP) gegenüber repräsentieren, desto geringer ist die Anzahl der Sicherheitslücken, die wiederum für Angriffe ausgenutzt werden können. Aufgrund ihrer Lage hat die Firewall also in diesem Fallbeispiel die Möglichkeit, den gesamten Datenverkehr, der vom internen Netzwerk ins Externe oder umgekehrt fließt, zu inspizieren, da dieser das Gerät dafür passieren muss. Die Firewall stellt hier also nicht nur den kritischen Punkt, sondern auch den *Single Point of Failure* für das gesamte Netzwerk dar.

4.2 Aufbau des Netzwerkes bei Argos

Der Netzwerkaufbau bei Argos orientiert sich am Aufbau eines Enterprise Netzwerkes und ist in Abbildung 4.2 auf der nächsten Seite zu sehen. Allgemein werden die Server von einer Firewall getrennt und somit vor den Mitarbeitercomputer und umgekehrt

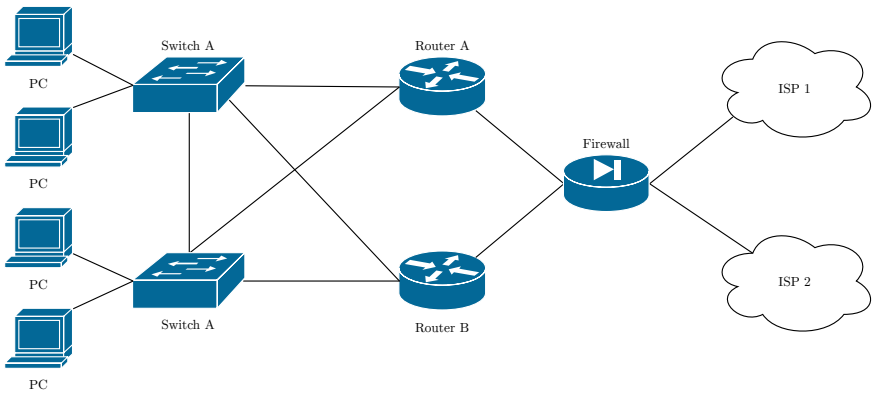

Abbildung 4.1: Der schematisch logische Netzwerkaufbau eines modernen Netzwerkes

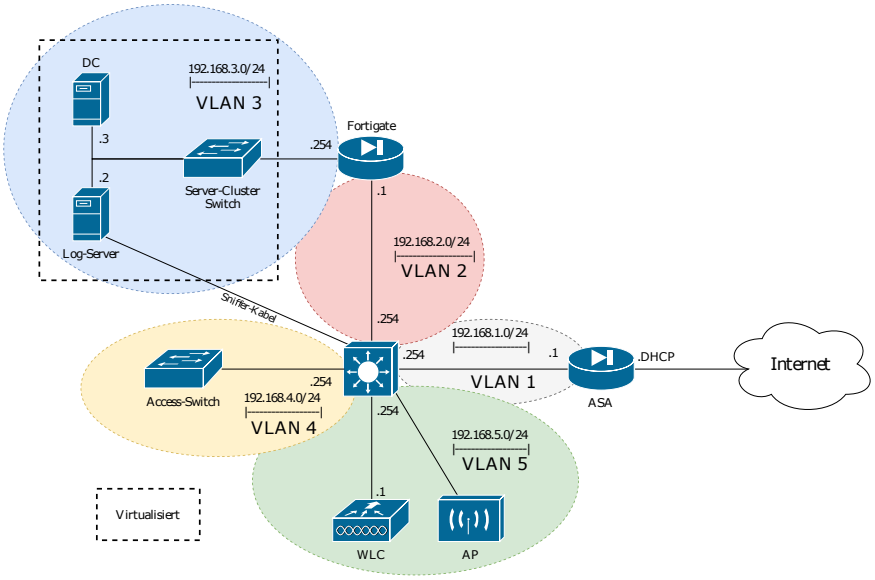

Abbildung 4.2: Der Aufbau des Netzwerkes bei Argos

geschützt. Eine weitere Firewall repräsentiert das Netzwerk nach außen hin. Das gesamte Netzwerk ist in 5 VLANs unterteilt, und alle terminieren auf dem zentralen Core-Switch.

VLAN 1

Dieses Virtual LAN (VLAN) verbindet den Core-Switch mit der Firewall am Rand des Netzwerkes. Es bietet somit die Konnektivität zur Firewall und weiterführend zum Internet.

VLAN 2

Dieses VLAN verbindet den Core-Switch mit der Firewall zum Servernetzwerk und schafft somit die Erreichbarkeit der laufenden Dienste.

VLAN 3

Dieses VLAN verbindet die virtualisierten Server untereinander und mit der Firewall.

VLAN 4

Dieses VLAN simuliert das Local Area Network (LAN) eines typischen Netzwerks. Dort können Computer per Kabel am Access-Switch angeschlossen und verwendet werden.

VLAN 5

Dieses VLAN simuliert das Wireless LAN (WLAN) eines typischen Netzwerks. Mobile Geräte können sich mit dem WLAN verbinden und gegen die Anmeldedaten am Windowsserver authentifizieren.

5 Aggregieren von Logdaten

5.1 Logging Übersicht

In der Diplomarbeit wurden unterschiedliche Protokolle zur Übermittlung von Log-Nachrichten verwendet. Genauer eingegangen wird, aus im Kapitel 5.2 ausgeführten Gründen, auf die für die Diplomarbeit wichtigsten Protokolle:

- Syslog
- SNMP

Diese Protokolle wurden auf den verwendeten Netzwerkkomponenten aktiviert und konfiguriert. Zu diesen zählen:

- Cisco Adaptive Security Appliance (ASA)
- Cisco WLAN-Controller (WLC)
- FortiGate
- Windows Server

5.2 Protokolle zur Logdatenübermittlung

Die Logdaten, welche für die Auswertung benötigt werden, können über verschiedenste Protokolle und Standards von den einzelnen Netzwerkkomponenten (wie Firewalls, Cisco WLC oder Serversysteme) zur weiteren Verarbeitung an den Log-Server übermittelt werden. Die am weitesten verbreiteten Techniken dafür sind Syslog, Simple Network Monitoring Protocol (SNMP), IP Flow Information Export (IPFIX) und das Cisco proprietäre Netflow. Alle diese Techniken bringen sowohl Vorteile als auch Nachteile mit sich. Sie unterscheiden sich hauptsächlich in den folgenden Punkten:

Dem Auslöser für das Senden der Logdaten:
Die Protokolle unterscheiden sich darin, aus welchen Gründen die Logdaten an den Server geschickt werden. So ist Syslog darauf ausgelegt, auf Basis von Ereignissen, welche auf den Netzwerkkomponenten auftreten, zu arbeiten. Falls ein Ereignis eintreten sollte, wird die entsprechende Nachricht an den Server gesendet. So

würde Syslog z. B. eine Log-Nachricht senden, wenn sich ein Benutzer auf dem Gerät angemeldet hat oder eine neue Transmission Control Protocol (TCP)-Verbindung aufgemacht wurde. SNMP ist auch in der Lage, Log-Nachrichten aufgrund von Ereignissen am Netzwerkgerät zu versenden. Es ist aber auch möglich, Anfragen für bestimmte Daten an das Netzwerkgerät zu senden, welches daraufhin mit diesen antwortet.

Der Aufbereitung der Logdaten:
Weitere Unterschiede sind in der Aufbereitung der verschickten Daten zu finden. Während bei Syslog die Informationen zu den einzelnen Ereignissen verschickt werden, werden bei Netflow Daten zu Verbindungen innerhalb des Netzwerks aufgezeichnet und verschickt. Es ist also wichtig, im Vorhinein zu wissen, welche Informationen zur weiteren Verarbeitung benötigt werden, damit dahingehend das richtige Protokoll gewählt werden kann.

Den unterschiedlichen Informationen, die gesendet werden können:
Ein weiterer wichtiger Unterschied liegt darin, dass nicht jedes Gerät jede Information über alle Protokolle übertragen kann. So können gewisse Informationen zu einem Event, in dem Maße, in dem sie benötigt werden, nur über Syslog übermittelt werden, während SNMP diese Informationen nicht zur Verfügung stellt. Dies variiert sehr nach Hersteller, Art des Netzwerkgeräts und verwendeter Betriebssystem-Version.

Aufgrund dieser Tatsachen haben wir uns in der Diplomarbeit auf die Protokolle Syslog und SNMP konzentriert. Denn diese Protokolle übermitteln jene Daten, die für die Auswertung im Rahmen dieser Diplomarbeit interessant sind. Beide Protokolle sind von sehr vielen Herstellern auf so gut wie allen Geräten implementiert. Wie die beiden Protokolle funktionieren, wird in den jeweiligen Kapiteln zum Syslog-Protokoll (siehe Kapitel 5.2.1) und SNMP Protokoll (siehe Kapitel 5.2.2 auf Seite 18) beschrieben.

5.2.1 Syslog-Protokoll

Syslog wurde bereits im Jahr 2001 als Standard zur Übertragung von Log-Nachrichten in der Request for Comments (RFC) 3164 definiert. Diese beschreibt, wie Netzwerkgeräte Ereignisse an eine zentrale Stelle, also einen Server, melden können, damit diese von Netzwerkadministratoren interpretiert und ausgewertet werden können. Der Begriff Syslog bezieht sich meist auf das Syslog-Protokoll selbst, wird aber auch in Bezug auf die Anwendungen, die die Syslog-Nachrichten versenden, empfangen und verarbeiten, verwendet. Eine solche Syslog-Anwendung kann in drei verschiedenen Modi operieren:

Device
Die Quelle der Syslog-Nachricht.

Relay
Ein Gerät, welches Syslog-Nachricht empfängt und gleich wieder weiterleitet.

Collector
Das Gerät, welches die Syslog-Nachricht schlussendlich verarbeitet.

In der ersten Veröffentlichung des Syslog-Standards wurde als Transportprotokoll User Datagram Protocol (UDP) auf Port 514 gewählt. Dies hält die Belastung auf das Netzwerk durch die zusätzlich versendeten Log-Nachrichten möglichst gering. Es stellte sich jedoch heraus, dass einige wichtige Log-Nachrichten, die systemkritische Ereignisse betreffen, doch verlässlich zugestellt werden sollten. Deshalb wurde in der späteren RFC 5424, die die ältere RFC 3164 ersetzt, die Anwendung von Syslog über das Transportprotokoll TCP beschrieben. In dieser wird jedoch kein zu verwendender Port angegeben, da der Port 514, wie er bei UDP verwendet wird, für TCP schon vergeben ist, deshalb muss dieser vom Netzwerkadministrator komplett frei gewählt werden. Ein wichtiges Feature von Syslog über TCP ist die Unterstützung von Secure Socket Layer (SSL), wodurch die Log-Nachrichten zusätzlich verschlüsselt werden können.

Eine Syslog-Nachricht besteht nach der neueren RFC 5424 aus drei wesentlichen Teilen:

HEADER
Dieser Teil enthält allgemeine Daten zum Event, wie den Gerätenamen des Netzwerkgeräts, das die Log-Nachricht verschickt oder den Zeitstempel, der angibt, zu welchem Zeitpunkt das Event eingetreten ist. Weiters enthält der *HEADER* das *PRI*-Feld, welches dafür da ist, die *Facility*- und *Severity*-Werte der Syslog-Nachricht anzugeben.

Die *Facility*, die die ersten fünf Bit des *PRI*-Felds ausmacht, gibt an, welcher Dienst oder welche Komponente innerhalb des Geräts die Nachricht hervorgerufen hat. Hierfür stehen laut der RFC 5424, die diese Version von Syslog beschreibt, 24 verschiedene Werte zur Verfügung. Die ersten 16 haben vordefinierte Bedeutungen und die restlichen 8 sind frei vom Hersteller oder Systemadministrator zu verwenden.

Die restlichen drei Bit des *PRI*-Felds beinhalten das *Severity-Level*, welches mittels eines Wertes zwischen 0 und 7 angibt, wie systemkritisch das aufgetretene Event ist. Hier gilt: während ein *Severity-Level* von 0 (Debug) eine einfache Debug-Nachricht sein kann und daher meistens eher unwichtig ist, ist ein *Severity-Level* von 7 (Emergency) ein für das Gerät kritisches Event und sollte auf jeden Fall nicht ignoriert werden.

STRUCTURED-DATA
Hier wird ein Mechanismus geboten, um Daten in einfachen Schlüssel-Wert-Paaren anzugeben. Er kann verwendet werden, muss aber nicht. In dem Fall,

dass dieser Teil nicht verwendet wird, muss er den Wert *NILVALUE* enthalten. Der Vorteil dieser Paare ist es, dass diese einfach von Maschinen verarbeitet werden können und daher sehr gut Metadaten zu den Syslog-Nachrichten angeben können.

Es können kein, ein oder mehrere Elemente vorhanden sein. Diese Elemente werden *SD-ELEMENT* genannt und setzen sich aus einem Namen (*SD-ID*) und einem oder mehreren Schlüssel-Wert Paaren (*SD-PARAM*) zusammen.

MSG

Hier werden in Form eines vom Hersteller frei wählbaren Textes die Informationen zum Ereignis angegeben. Es gibt keine Vorlagen vom Syslog-Standard wie dieser Text aussehen sollte. Es gibt Hersteller, die versuchen, diesen Text möglichst maschinenlesbar zu halten, während andere diesen Text so formulieren, dass er einfacher von Menschen gelesen werden kann. Beispiele hierfür sind im Kapitel 8.2 auf Seite 60 zu finden.

Vom Syslog-Protokoll selbst wird keine maximale Paketlänge angegeben. Diese ist rein durch die Implementierung mit dem verwendeten Transportprotokoll bestimmt. Empfohlen wird jedoch, dass Syslog-Applikationen Nachrichten mit einer Länge bis zu 2048 Oktetten verarbeiten können. Eine Syslog-Applikation muss Nachrichten verarbeiten können, die mindestens 480 Oktetten lang sind.

5.2.2 Simple Network Management Protocol

SNMP ist ein Protokoll, das es erlaubt, Netzwerkgeräte, wie Router, Switches oder Firewalls, aber auch Serversysteme von einem zentralen Punkt im Netzwerk aus zu verwalten. Hierbei kann, im Gegensatz zu Syslog, eine bidirektionale Kommunikation zwischen den Netzwerkgeräten und der zentralen Managementstation stattfinden. Um diese Kommunikation zu gewährleisten, benötigt SNMP zwei Komponenten:

SNMP-Agent

Der SNMP-Agent ist eine Software, die auf den einzelnen Netzwerkgeräten läuft. Sie ist in der Lage Systemzustände zu erfassen, Konfigurationen am Gerät zu ändern oder andere, vom Hersteller definierte Aktionen, durchzuführen.

Die meisten modernen Betriebssysteme von Netzwerkgeräten haben bereits eine Art SNMP-Agent integriert. Man kann einen SNMP-Agent aber auch auf anderen Betriebssystemen finden, wie z. B. auf Linux-basierten Betriebssystemen in Form des *snmpd*.

SNMP-Manager

Der SNMP-Manger ist die Software, die auf einem zentralen Server installiert

wird und Anfragen an die SNMP-Agents der Netzwerkgeräte sendet, die von diesen dann beantwortet werden. Meist ist der SNMP-Manager jedoch sehr eingeschränkt, da verschiedene Hersteller unterschiedliche Aktionen unterstützen und der Funktionsumfang daher sehr schwankt.

Die Informationen, wie die Daten der Netzwerkgeräte von den SNMP-Managern abgefragt werden können, werden in Form von MOs in der sogenannten Management Information Base (MIB) gespeichert. Es handelt sich bei einer MIB um keine Datenbank, in der Daten gespeichert werden, sondern um eine Beschreibung, wo welche Daten zu finden sind und wie diese aussehen. Um die MIB zu definieren, wurde die Spezifikationssprache Abstract Syntax Notation One (ASN.1) verwendet, um eine einem Baum ähnliche Struktur zu erstellen (siehe Abbildung 5.1).

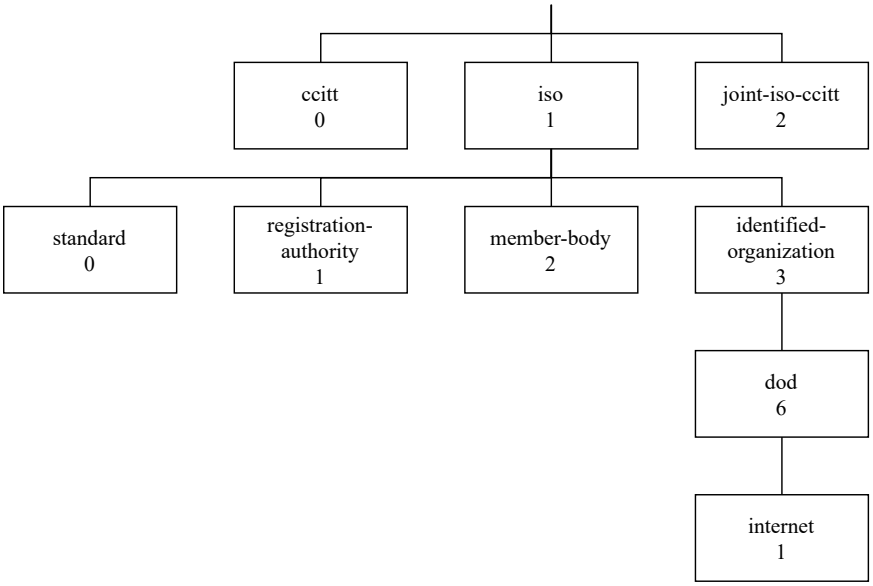

Abbildung 5.1: Auszug des SNMP MIB Baums

In der RFC 1155 wurde eine global gültige MIB definiert, die zum Ziel hat, alle möglichen Geräte von allen möglichen Herstellern abzubilden. Dieser wurde in der Vergangenheit bereits öfters erweitert, wobei diese Erweiterungen immer daraus bestanden, dass neue Verzweigungen im Baum ergänzt wurden. Diese werden auch MIB-Module genannt.

Die einzelnen MOs werden nun durch sogenannte OIDs identifiziert, wobei jeder Object Identifier (OID) im Baum eindeutig ist. Um einen OID darzustellen, gibt es zwei verschiedene Methoden (am Beispiel des *sysUpTimeInstance*-OID):

- Als Zahlenkette: 1.3.6.1.2.1.1.3.0 oder
- Als ASCII-Punktnotation:
  iso.identified-organization.dod.internet.mgmt.mib-2.system.sysUpTime.sysUpTimeInstance

Bei den obigen Notationen gilt immer, dass der eigentliche Wert, der ausgelesen werden soll, bei der Zahlenkette durch eine *0* am Schluss und bei der ASCII-Punktnotation durch das Wort *Instance* identifiziert wird.

Damit ein SNMP-Manager nun aber einen gewissen OID auslesen oder verändern kann, muss dieser die entsprechenden Pakete an den SNMP-Agent schicken:

GET-REQUEST
: Mit Hilfe dieses Befehls fordert der SNMP-Manager den Datensatz einer bestimmten OID an, welche im Paket angegeben sein muss. Dieses Paket wird dann vom SNMP-Agent durch eine *GET-RESPONSE*-Nachricht beantwortet. Weiters ist das Iterieren von mehreren OIDs mittels des *GETNEXT-REQUEST* Befehls möglich, ebenso das Auslesen mehrerer Datensätze auf einmal mit dem *GETBULK* Befehl.

SET-REQUEST
: Mit diesem Befehl versucht der SNMP-Manager, einen oder mehrere Datensätze am SNMP-Agent zu setzen.

Für beide dieser Nachrichten wird der UDP-Port 161 verwendet.

Der SNMP-Agent, der auf dem Netzwerkgerät läuft, ist auch von sich aus in der Lage, ohne das vorherige Empfangen eines Befehls vom SNMP-Manager, Nachrichten an diesen zu senden. Dafür stehen zwei Nachrichtentypen zur Verfügung:

TRAP
: Mit dieser Nachricht werden unaufgefordert Nachrichten vom SNMP-Agent an den SNMP-Manager gesendet. *TRAP-Nachrichten* werden meist verwendet, um auf dem Netzwerkgerät aufgetretene Ereignisse möglichst schnell an den SNMP-Manager zu melden. Dies geschieht in Form einer einzigen Nachricht, deren Empfang nicht bestätigt wird.

INFORM
: Diese Nachricht ähnelt sehr der *TRAP*-Nachricht, der große Unterschied ist aber, dass der Erhalt dieser Nachricht vom SNMP-Manager bestätigt wird. Dies ermöglicht es, sicherzustellen, dass systemkritische Nachrichten garantiert ihr Ziel erreichen.

Sowohl die *TRAP*- als auch die *INFORM*-Nachricht verwendet den UDP-Port 162.

So wie viele andere Protokolle auch, wird SNMP stetig weiterentwickelt, und neue Versionen spezifiziert. Während sich auf Seiten der verwendeten Nachrichten in den verschiedenen Versionen, bis auf das Hinzufügen der *INFORM*-Nachricht und *GETBULK*-Nachricht, eher wenig getan hat, hat sich auf Seiten der Authentifizierung der Nachrichten Einiges getan:

SNMPv1

Die erste Version von SNMP hat die grundsätzlichen Protokollerwartungen erfüllt.

Es ist möglich, Netzwerkgeräte von einer zentralen Managementstation aus zu überwachen. Ein großes Problem stellt hier nur die Implementierung der Authentifizierung zwischen SNMP-Agent und SNMP-Manager dar. Diese wird mittels eines *Community-Strings* realisiert, der von der Funktionsweise her sehr mit dem eines Passworts zu vergleichen ist. Das Problem hierbei ist, dass dieser Teil jeder Nachricht ist um diese zu authentifizieren. Da er jedoch in Form eines Klartext-Strings übertragen wird, kann er einfach ausgelesen werden.

SNMPv2

SNMPv2 existiert wiederum in drei verschiedenen Versionen: SNMPv2p, SNMPv2u und SNMPv2c.

SNMPv2p hebt sich von der ersten Version in punkto Sicherheit ab: Der *Community-String* wird nun nicht mehr in Klartext, sondern verschlüsselt übertragen. Weiters ist es nun möglich, dass sich mehrere SNMP-Manager untereinander austauschen und diese mehrere OIDs gleichzeitig mittels des *GETBULK*-Befehls auslesen.

SNMPv2u löst das Sicherheitsproblem der ersten Version, indem zur Authentifizierung nun zusätzlich Benutzernamen verwendet werden.

SNMPv2c ist jedoch die einzige der drei Versionen, die heute noch anzutreffen ist und wenn von SNMPv2 gesprochen wird, ist meist diese Version gemeint. Die Sicherheit der Authentifizierung ist bei dieser Version jedoch immer noch auf dem gleichen Level wie bei der ersten und verwendet keine verschlüsselte *Community-Strings*. Das Verwenden des *GETBULK*-Befehls und die Kommunikation zwischen SNMP-Managern ist in dieser Version ebenfalls möglich.

SNMPv3

SNMPv3 ist die momentan neueste Version und bringt einige Verbesserungen mit sich, vor allem im Bereich der Sicherheit. So können Nachrichten nun einerseits verschlüsselt übertragen werden und andererseits kann der Empfänger der Nachricht authentifizert werden, damit dieser sicherstellen kann, dass diese Nachricht auch an ihn gerichtet war.

Diese zwei Features können in unterschiedlichen Kombinationen zusammen angewandt werden:

- *noAuthNoPriv*: Die Daten werden nicht verschlüsselt, und der Empfänger wird nicht authentifiziert. Es muss nur ein Benutzername angegeben werden.
- *authNoPriv*: Die Daten werden nicht verschlüsselt, der Empfänger wird aber authentifiziert. Es werden ein Benutzername und Passwort angegeben.
- *authPriv*: Die Daten werden verschlüsselt, und der Empfänger wird authentifiziert. Es werden ein Benutzername und ein Passwort zur Authentifizierung und ein Passwort für die Verschlüsselung angegeben.

Durch die in SNMPv3 verbesserten Sicherheitsmaßnahmen fällt die Konfiguration dieser Version im Vergleich zu der der anderen Versionen komplexer aus. Dies hat SNMPv3 bis jetzt daran gehindert, den gleichen Verbreitungsgrat wie SNMPv2 zu erreichen.

5.3 Logging auf Cisco Geräten

Der Hersteller *Cisco Systems, Inc.* bietet im Bereich der Netzwerkgeräte ein sehr breit gefächertes Produktportfolio an. So sind viele verschiedene Typen an klassischen Netzwerkgeräten wie Router, Switches oder Firewalls am Markt. Weiters streckt sich die Produktpalette bis hin zu APs und WLCs, die kabellose Verbindungen ermöglichen. Alle diese Geräte sind wichtige Bestandteile eines Netzwerks und müssen daher auch vom Systemadministrator überwacht werden. Dafür stehen verschiedene Möglichkeiten bereit, wie Log-Nachrichten auf Cisco Geräten gehandhabt werden.

Möglichkeiten, die auf sehr vielen Geräten zur Verfügung stehen, sind:

Console-Logging
Besteht eine physische Verbindung auf den Konsolen-Port des Geräts, können die Log-Nachrichten im Terminal in Echtzeit ausgelesen werden. Ein Problem hierbei ist, dass für jede Nachricht ein CPU-Interrupt gesendet wird, was jedoch die Performance des Geräts beeinträchtigen kann.

Terminal-Logging
Diese Möglichkeit ist der ersten sehr ähnlich, indem die Log-Nachrichten in Echtzeit an das verbundene Gerät sendet. Der Unterschied liegt darin, dass die Verbindung hier via der *Virtual Terminal Line (VTY)* des Geräts hergestellt wird und nicht physisch über den Konsolen-Port.

Buffered-Logging
Diese Funktion ermöglicht es, die Log-Nachrichten im Random Access Memory (RAM) des Geräts zwischenzuspeichern und diese dann später abzurufen. Dieser Zwischenspeicher hat eine konfigurierbare Größe. Sollten mehr Nachrichten eintreffen als gespeichert werden können, werden die ältesten Nachrichten wieder verworfen. Die Größe des Zwischenspeichers sollte so gewählt werden, dass das Netzwerkgerät noch genug RAM zur Verfügung hat, um seine normalen Tätigkeiten durchführen zu können.

Syslog-Logging
Wie im Kapitel 5.2.1 auf Seite 16 beschrieben, werden die Log-Nachrichten hier an einen zentralen Syslog-Server gesendet, der diese dann speichert und auswertet.

SNMP-Trap-Logging
Wie im Kapitel 5.2.2 auf Seite 18 beschrieben, erlauben es SNMP-Traps, einem Netzwerkgerät proaktiv SNMP-Nachrichten zu Ereignissen zu senden.

Netflow
Mit Hilfe von Netflow können Informationen zu sogenannten Flows gewonnen werden, die jeweils aus einer Quell-Adresse, einer Ziel-Adresse und dem verwendeten Protokoll bestehen. Diese Daten werden dann zur Auswertung wiederum an einen zentralen Server gesendet.

Mail
Einige Geräte sind in der Lage, über Simple Mail Transfer Protocol (SMTP) E-Mails an den Systemadministrator zu senden und diesen dadurch auf ein aufgetretenes Ereignis aufmerksam zu machen.

5.3.1 Logging auf einer Cisco ASA

In der Diplomarbeit wurde als eine der Firewalls konkret eine Cisco ASA 5506-X verwendet. Die Log-Nachrichten dieser sind besonders interessant, da diese Firewall die Grenze zwischen internen und externen Netzwerken darstellt. Aus diesem Grund werden jegliche Nachrichten, die an einen Empfänger in einem externen Netzwerk gerichtet sind, über diese geleitet (nähere Informationen zur Topologie Abbildung 4.2 auf Seite 14). Daher fallen auf diesem Gerät sehr viele Log-Nachrichten an, die Aufschluss darüber geben, mit welchen externen Zielen unsere Geräte kommunizieren, wie sie das tun und wie oft.

Daher wurde auf der Cisco ASA das Syslog-Logging konfiguriert. Da die Syslog-Nachrichten genauen Aufschluss über die einzelnen Verbindungen geben, die über die Cisco ASA laufen, wurde dieses Protokoll gewählt. Konfiguriert wurde das Logging über das Command Line Interface (CLI) mit dem Syslog-Server der Topologie als Ziel.

Die für das Logging wichtige Konfiguration sieht wie folgt aus:

```
logging enable
logging host INSIDE 192.168.3.3 udp 514
logging trap debug
```

Zuerst wird Logging aktiviert. Danach wird das Ziel für die Log-Nachrichten definiert. *INSIDE* ist der Alias der Schnittstelle auf der Cisco ASA, die in Richtung des Syslog-Servers zeigt, also die Schnittstelle, die in unser Netzwerk hinein führt. *192.168.3.3* ist die Internet Protocol (IP)-Adresse des Syslog-Servers und *udp 514* ist das Transportprotokoll mit dem dazugehörigen Port, auf den die Nachricht gesendet werden soll.

Die dritte Zeile konfiguriert, ab welchem *Severity-Level* eine Log-Nachricht an den Syslog-Server gesendet werden soll. Das konfigurierte Level von *DEBUG* bedeutet, dass alle auftretenden Nachrichten über Syslog an den Server gesendet werden sollen.

Weiters wäre es noch möglich, die *Syslog-Facility* manuell auf einen der frei verwendbaren Level zu setzen. Bei einem der frei wählbaren Werte, in diesem Fall wurde die *Facility* 21 gewählt, ist dies mit dem folgenden Befehl möglich:

```
logging facility 21
```

5.3.2 Logging auf einem Cisco WLC

Der WLAN-Controller (WLC), der in der Diplomarbeit verwendet wurde, ist der Cisco WLC 4402. Dieser wird verwendet, um es Access Points (APs) zu erlauben, als Light Weight Access Point (AP) (LWAP) zu fungieren. Dies bedeutet, dass APs zentral am WLC verwaltet werden können und die Logik, wie Pakete weitergeleitet werden, komplett vom WLC übernommen wird. Daher übernimmt dieser auch die Authentifizierung der Benutzer im WLAN, wobei er dafür in dieser Diplomarbeit die Anfragen an einen Radius-Server sendet, der die eigentliche Überprüfung der Benutzerdaten vornimmt. Dadurch, dass der WLC diesen Anmeldeprozess von Benutzern bzw. Geräten zumindest als Mittelsmann dient, können von diesem wichtige Informationen, wie welche Geräte und Benutzer sich wann angemeldet haben, gewonnen werden. Diese Informationen sind essenziell, um zurückverfolgen zu können, welche Benutzer sich wann, mit welchem Gerät und an welchem AP angemeldet haben.

Zu diesem Zwecke wurden auf dem Cisco WLC sowohl Syslog, als auch SNMP konfiguriert. Es wurden beide Protokolle verwendet, da keines der beiden Protokolle alleine einen vollständigen Überblick über die Anmeldung eines Benutzers im WLAN gibt.

Dadurch, dass beide konfiguriert wurden, können die Informationen später am Server zusammengefügt werden, um alle wichtigen Details zu speichern. Da die Konfiguration so um Einiges leichter ist, als über das CLI, wurden die Protokolle mittels der Weboberfläche des WLC konfiguriert.

In der Abbildung 5.2 wird die SNMP-Trap Konfiguration dargestellt. Konfiguriert wurde ein frei wählbarer Name für die Community, in diesem Fall *Argos* und die IP-Adresse des Log-Servers. Schlussendlich wurde diese Konfiguration noch aktiviert, damit aktiv SNMP-Traps gesendet werden können.

Abbildung 5.2: Konfiguration von SNMP-Traps auf einem Cisco WLC

Zum Vergleich: Die Abbildung 5.3 enthält die Konfiguration, um Syslog-Nachrichten vom Cisco WLC an den Log-Server zu senden. Hier muss die IP-Adresse des Log-Servers, das *Syslog-Level* (in diesem Fall *Debugging*, damit wirklich alle Log-Nachrichten über Syslog versandt werden) und die *Syslog-Facility* konfiguriert werden.

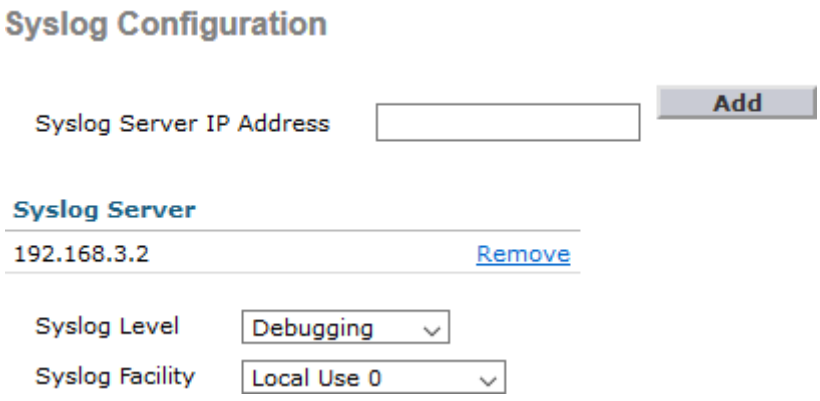

Abbildung 5.3: Konfiguration von Syslog auf einem Cisco WLC

Beispiele zu Log-Nachrichten des Cisco WLC und wie diese weiterverarbeitet werden, werden im Kapitel 8.3 auf Seite 74 beschrieben.

5.4 Logging auf Fortinet Geräten

Ähnlich wie Cisco hat der Hersteller Fortinet eine sehr breite Produktpalette. Die von Fortinet beschränkt sich aber eher auf den *Security*-Bereich. Dies bedeutet, dass hier der Fokus auf verschiedensten Firewalls (FortiGate), SIEM-Lösungen oder Intrusion Prevention System (IPS)-Systemen liegt.

Genauso wie bei Cisco ist es hier auch wieder möglich, die auf den Geräten anfallenden Log-Nachrichten unterschiedlich zu behandeln. Als Log-Quellen kommen hier hauptsächlich nur Firewalls in Frage, da die restlichen Geräte eher als Ziele für Log-Nachrichten dienen und diese eher selten selbst generieren. Die wichtigsten Möglichkeiten sind:

Lokal
: Die Log-Nachrichten können lokal am Gerät selbst gespeichert werden.

Syslog
: Um die Log-Nachrichten zu einem zentralen Syslog-Server zu senden, kann Syslog genutzt werden.

SNMP
: Mittels SNMP-Traps können die Log-Nachrichten an einen SNMP-Manager geschickt werden.

FortiAnalyzer
: Die Log-Nachrichten können an ein Fortinet eigenes Serversystem gesendet werden, welches speziell für die Auswertung von Fortinet-Logs ausgelegt ist.

5.4.1 Logging auf einer FortiGate Firewall

In der Diplomarbeit wurde als weitere Firewall konkret eine FortiGate 60D Firewall verwendet. Sie dient als Schutzmaßnahme zwischen dem Servernetzwerk und dem restlichen Netzwerk. Da dadurch alle Nachrichten, die an einen Server in unserem Netzwerk gerichtet sind, über diese Firewall geleitet werden, ist dies die ideale Stelle im Netzwerk, um Log-Nachrichten in Bezug auf Zugriffe auf die Server zu sammeln.

Auf der FortiGate wurde das Logging via Syslog auf einen Syslog-Server konfiguriert. Hier wurde das Syslog Protokoll gewählt, da dieses Aufschluss darüber gibt, welche Nachrichten über die FortiGate aufgebaut werden. Um Syslog auf einer FortiGate zu konfigurieren, müssen zwei verschiedene Konfigurationsmodi genutzt werden. Zuerst müssen die allgemeinen Einstellungen getroffen werden:

```
config log syslogd setting
    set server 192.168.3.2
    set status enable
end
```

Hierfür wird zuerst der *setting* Konfigurationsmodus von Syslog betreten, in dem die IP-Adresse des Syslog-Servers konfiguriert wird. Weiters wird der Syslog-Server als Ziel auch noch aktiviert. Es wäre hier auch möglich, einen eigenen Port zu konfigurieren: `set port <PORT>` oder selbst eine *Facility* zu wählen: `set facility <FACILITY>`.

Im zweiten Konfigurationsmodus wird der *filter* konfiguriert. Hier wird das *Severity*-Level gesetzt:

```
config log syslogd filter
    set severity debug
end
```

Es ist möglich, bis zu vier verschiedene Syslog-Server gleichzeitig zu konfigurieren. Hierfür wird einfach statt *syslogd* *syslogd2*, *syslogd3* oder *syslogd4* eingesetzt. Beispiele zu Log-Nachrichten der Fortigate und wie diese weiterverarbeitet werden, werden im Kapitel 8.2.3 auf Seite 72 beschrieben.

5.5 Konfiguration von Windows Server

5.5.1 Einleitung zu Windows Server

Ein Active Directory (AD) mit Windows Servern ist heutzutage standardmäßig Teil von beinahe jedem modernen Firmennetz. Es vereinfacht die Benutzer- und Computer-Verwaltung, indem eine zentrale Verwaltungsumgebung zur Verfügung gestellt wird. Dadurch wird ein Benutzer nur einmalig erstellt, dieser hat aber die Möglichkeit, sich im gesamten Firmennetz zu authentifizieren. Weiters können einmalig Regeln erstellt werden, welche wiederum auf alle Computer und Benutzer des AD-integrierten Firmennetzes angewandt werden. Zusätzlich dazu kann den Windows Servern auf sehr intuitiver Weise gewisse Rollen hinzugefügt werden, welche wiederum den Funktionsumfang des Servers erweitern. Natürlich wird innerhalb des Systems auch überprüft und protokolliert, ob es Fehler- und/oder Error-Meldungen gibt bzw. gab. Diese Mitprotokollierung ist in der Windows Ereignisanzeige einsehbar.

Im Grunde wird der Windows Server in der Diplomarbeit für die Authentifizierung von Benutzern für das WLAN herangezogen. Dafür muss auf dem Gerät die Rolle

Network Policy Server (NPS) hinzugefügt und im Konfigurationsfenster für diese das Beantworten von Radiusanfragen[1] konfiguriert werden. Natürlich protokolliert die Rolle mit, welcher Benutzer zu welcher Zeit entweder einen erfolgreichen oder einen nicht erfolgreichen Authentifizierungsversuch durchgeführt hat. Das Logging kann zwar auch manuell unterbunden werden, dies ist aber in den meisten Fällen nicht sinnvoll. Für eine zentrale Echtzeitüberwachung müssen diese Informationen nun zu einem entfernten Sammelpunkt gelangen – die Lognachrichten müssen weitergeleitet werden.

5.5.2 Windows-zu-Windows Log-Weiterleitung

5.5.2.1 Übersicht über verschiedene Weiterleitungstechnologien

AD-integrierte Windows Geräte bieten grundsätzlich zwei Möglichkeiten, damit Lognachrichten an einem entfernten Windows Gerät verarbeitet werden können. Bei der Technologie *Remote Event Viewer* werden die Nachrichten nicht physikalisch auf der entfernten Maschine abgelegt und gespeichert, sondern lediglich angezeigt. Im Gegensatz dazu können die Logs auch automatisiert und physikalisch übertragen werden. Hierbei handelt es sich um die Funktion *Weiterleitung der Log Nachrichten.*

5.5.2.2 Remote Event Viewer

Erstere Technologie lässt sich in einem bereits aufgesetzten AD sehr einfach implementieren. Zwischen dem Sender und Empfänger muss natürlich IP Konnektivität bestehen. Zusätzlich muss der TCP-Port 135 mittels der vordefinierten Firewallregel *COM+ Network Access (DCOM-In)* für eintreffende Anfragen am Punkt des Einsehens geöffnet werden. Danach kann in der Windows Ereignisanzeige des Empfängers eine Verbindung mit der Ereignisanzeige der Quelle hergestellt werden. Dadurch erhält der Empfänger nun einen zweiten Tab mit der Ereignisanzeige und somit mit den mitprotokollierten Ereignissen der Quelle. Diese Technologie eignet sich somit für das manuelle Auslesen, nicht aber für eine tatsächliche Weiterleitung und dadurch auch nicht für eine automatisierte und zentrale Echtzeitübersicht der Ereignisse im Netzwerk.

[1] Radiusanfragen werden durchgeführt, wenn einem Benutzer die Möglichkeit gegeben werden soll, dass er seine Windows Accountdaten auch für andere Wege der Authentifizierung verwenden können soll. Zum Beispiel kommt dies bei der Anmeldung im WLAN vor.

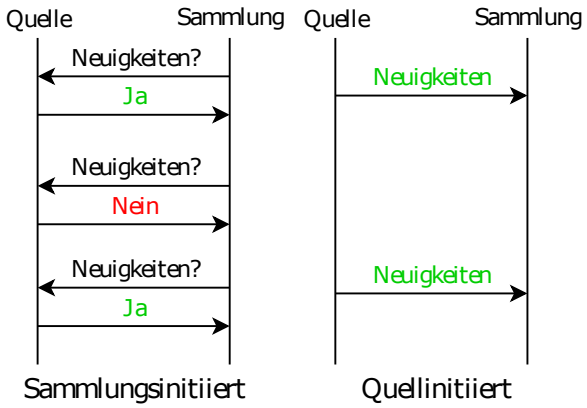

Abbildung 5.4: Der Ablauf von beiden verschiedenen Weiterleitungen für Syslog

5.5.2.3 Sammlungsinitiierte Weiterleitung

Die sammlungsinitiierte Weiterleitung stellt die Konfiguration einer tatsächlichen Logweiterleitung mit weniger administrativem Aufwand dar. Die Konfiguration dieser Technologie wird – wie der Name schon sagt – am Empfänger, dem Sammelpunkt, konfiguriert. Dabei muss am Empfänger in der Ereignisanzeige ein sammlungsinitiiertes Abonnement erstellt und der/die Quellcomputer ausgewählt werden. Hier müssen ebenfalls die Ereignisse, welche repliziert werden sollen und der Anzeigeort der weitergeleiteten Ereignisse, ausgewählt werden. Bei dieser Art der Weiterleitung initiiert der Empfänger also periodisch Anfragen, ob beim Quellcomputer neue Ereignisse vorliegen. Mit diesen periodischen Anfragen geht somit immer eine gewisse Weiterleitungsverzögerung Hand in Hand. Deswegen eignet sich diese nur bedingt für das Sammeln von kritischen Systembenachrichtigungen, wie zum Beispiel solche, die bei einem versuchten Login auftreten. In Abbildung 5.4 ist auf der linken Seite diese Technologie veranschaulicht. [12]

5.5.2.4 Quellcomputerinitiierte Weiterleitung

Auch die quellcomputerinitiierte Weiterleitung wird in der Ereignisanzeige des Empfängers konfiguriert. Dabei unterscheiden sich die beiden Konfigurationen erstmal nur anhand einer Auswahl, welche die gewünschte Funktion durch ein Kontrollkästchen identifiziert. Zusätzlich dazu muss bei dieser Konfiguration noch ein Group Policy Object (GPO) konfiguriert werden, welches den Empfänger eindeutig erfasst. Der Wert des GPOs

```
Computerkonfiguration/Administrative Vorlagen/Windows-Komponenten/
    Ereignisweiterleitung/Ziel-Abonnement-Manager
```

muss dem Syntax

```
Server=https://<FQDN>:5986/wsman/SubscriptionManager/WEC,
    Refresh=<Sekunden>,IssuerCA=<Fingerprint>
```

entsprechen. Es kann aber auch eine anderes Übertragungsprotokoll und nicht HTTPS verwendet werden. Hier bietet sich zum Beispiel das unverschlüsselte Übertragungsprotokoll HTTP an, und weil bei diesem Protokoll keine Zertifikate zum Einsatz kommen, kann in dem Fall das Feld der *IssuerCA* weggelassen werden. Diese Weiterleitung eignet sich schon sehr gut für das skalierende Sammeln von kritischen Ereignissen, wenn der Empfänger ein Windows Gerät ist. Dies ist bei der Diplomarbeit nicht der Fall. In Abbildung 5.4 auf der vorherigen Seite ist auf der rechten Seite diese Technologie veranschaulicht. [12]

5.5.3 Windows-zu-Linux Log-Weiterleitung

Die Weiterleitung von Windows zu anderen Windows Geräten ist sehr gut im Grundbetriebssystem implementiert. Sollte aber das Ziel sein, die Nachrichten zu einem Linux Gerät weiterzuleiten – wie es bei der Diplomarbeit der Fall ist –, dann bedarf es entweder Drittanbietersoftware oder aber Skripte. In der Diplomarbeit kommt ersteres in Form der Ereignisweiterleitung von SolarWinds zum Einsatz. Die Konfiguration ähnelt die der quellcomputerinitiierten Weiterleitung sehr stark. Allerdings werden von dem Programm die ausgewählten Ereignisse in Form von Syslognachrichten an einen Syslogserver weitergeleitet, und das manuelle Konfigurieren des GPOs muss nicht durchgeführt werden.

Die Konfiguration der Weiterleitung lässt sich am besten in der Abbildung 5.5 auf der nächsten Seite betrachten, welche die Oberfläche für die Konfiguration darstellt. Zuerst muss ein Syslog-Server und eine Subscription erstellt werden. Der Syslog-Server identifiziert anhand der IP-Adresse den Sammelpunkt der Log-Nachrichten und die Subscription die Ereignisse, welche weitergeleitet werden sollen.

In der Diplomarbeit werden ausschließlich sicherheitsrelevante Ereignisse herangezogen, um Angriffe zu erkennen. Diese umfassen unter anderem alle Anmeldungen und Authentifizierungsversuche gegenüber dem Windows Server. Für jeden unterschiedlichen Authentifizierungsversuch existiert ein eigener Typ. Das grundlegende XML Format dieser unterschiedlichen Typen unterscheidet sich nicht. Allerdings werden je nach Typ verschiedene Felder hinzufügt, respektive weggelassen. Die wichtigsten Felder allerdings sind:

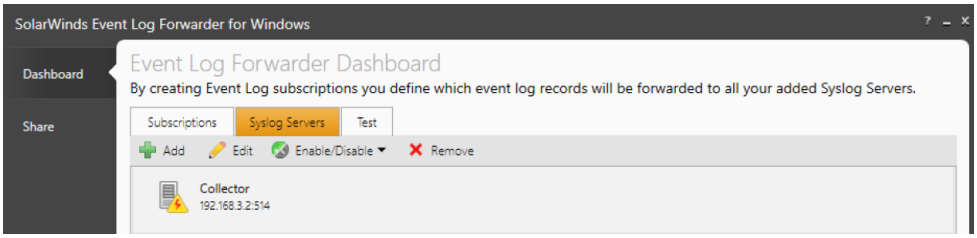

Abbildung 5.5: Die Oberfläche für die Log-Weiterleitung

AccountInformation
: Beinhaltet Information, die einen Account eindeutig identifiziert. Darunter fällt unter anderem der Security Identifier (SID) und der Accountname.

EventId
: Zeigt an, um welches Event es sich handelt.

Origin
: Gibt an, von welchem Gerät das Event stammt.

6 Deep Packet Inspection

6.1 Einleitung zu Deep Packet Inspection

Deep Packet Inspection (DPI) beschreibt die Informationsgewinnung aus der siebten und somit höchsten OSI-Schicht des OSI-Modells. Dies bedeutet, dass man bei DPI nicht auf die Informationen aus dem Kopfbereich eines Paketes beschränkt ist, sondern dass man zusätzlich dazu auch Informationen aus dem Datenbereich eines Paketes extrahieren kann. In diesem Kaptel kann ein Paket allerdings nicht mit einer Layer 3 Protocol Data Unit (PDU) im OSI-Modell gleichgesetzt werden. Vielmehr wird hier mit dem Begriff *Paket* das gesamte PDU beschrieben. Wenn man nun DPI mit dem Lesen eines Buches vergleichen würde, so hätte man im Gegensatz dazu mit herkömmlichen Überwachungstechnologien maximal eine Inhaltsangabe, wenn nicht überhaupt nur den Titel. Aufgrund der neugewonnenen Informationen kann das PDU somit viel granularer eingeteilt und individueller behandelt werden. Unter diese individuelle Behandlung fällt unter anderem das Umleiten, Verwerfen oder das spezielle Markieren mittels Quality of Service.

Dass die Pakete nun granularer eingeteilt werden können, ist der grundlegende Vorteil von DPI gegenüber herkömmlichen Überwachungstechnologien. Wegen dieser großen Menge an Neuinformationen und der hohen Individualität kann erst effektiv erkannt werden, ob aktuell ein komplex zu erkennender Angriff auf das Netzwerk stattfindet. Dies ist erst durch DPI möglich, da manche Angriffsvektoren aus der Sicht von niedrigeren OSI-Schichten als normaler Netzwerkverkehr missverstanden werden können. Erst durch das Inspizieren der Nutzdaten – Schicht sieben – wird ein solcher Angriff aufgedeckt, und es kann dagegen vorgegangen werden. [3]

6.2 Grundvoraussetzungen für DPI

Einige Rahmenbedingungen müssen gegeben sein, damit DPI zum Einsatz kommen kann. Es muss erstmal evaluiert werden, an welchen Punkten im Netzwerk DPI benötigt wird, um damit die Sicherheit des Netzes zu verbessern. Diese werden oft als kritische Punkte[1] beschrieben. An diesen Punkten müssen dann DPI-fähige Geräte

[1] Der kritische Punkt ist ein Punkt im Netzwerk, welcher sich aufgrund der logischen Netzwerktopologie besonders gut dazu eignet, um dort eine Überwachungstechnologie zu implementieren.

den Netzwerkverkehr einsehen. Weiters kann DPI keine Aussagen über verschlüsselte Daten, die von dem bearbeitenden Gerät nicht entschlüsselt werden können, treffen. In diesem Fall kann die DPI lediglich herausfinden, dass die Daten verschlüsselt sind, nicht aber die wirklichen Nutzdaten inspizieren oder analysieren. [3]

6.3 DPI auf Firewalls

Dass sich Firewalls in den meisten Netzaufbauten ähneln, wie in Kapitel 4 auf Seite 13 zu sehen ist, ist natürlich auch diversen Herstellern bewusst, und deswegen haben viele Unternehmen wie Cisco oder Fortinet eigene DPI-fähige Produktlinien eröffnet. Die Firewalls daraus werden NGFWs genannt, und sie beschränken sich dabei nicht nur auf DPI, sondern sie fusionieren auch zahlreiche andere Funktionen, wie zum Beispiel einen VPN-Endpoint, Antispam/Antivirus Technologien oder aber auch ein automatisiertes Verhindern von Angriffen, miteinander. Allerdings wird ein Großteil dieser Funktionen zumeist individuell lizenziert, und deswegen steigt der Verkaufspreis eines solchen Produktes massiv, und das sprengt das Budget einer Schule bei weitem. [8]

Der einzig wirklich große Nachteil, den DPI – im Vergleich zu herkömmlichen Paketfiltertechnologien – auf Firewalls mit sich bringt, ist, dass bei einem zu großen Datenverkehr das Gerät überlastet wird. In dem Szenario fallen dann nicht nur die DPI-Funktionalität, sondern auch andere Prozesse weg. Das kann im schlimmsten Fall bis zum Konnektivitätsverlust der Benutzer führen.

Einige NGFWs können nicht nur Daten in Schicht sieben analysieren, sondern sogar den Datenbereich im PDU verändern. Dies ist allerdings nochmal bei weitem aufwendiger als das ohnehin schon sehr ressourcenintensive Auswerten des Datenbereichs und wirkt sich nochmals stärker auf die Auslastung der Firewall aus.

6.4 DPI auf einer Cisco ASA

6.4.1 Einleitung zur Cisco ASA

Die Cisco ASA ist eine von Cisco entwickelte und 2005 eingeführte Firewall. Seit der Übernahme von Sourcefire im Jahr 2014 wurden immer mehr Features von besagter Firma in die Firewall Technologie von Cisco übernommen und auf das Namensschema

Zusätzlich dazu kann der kritische Punkt auf dem *Single Point of Failure* liegen, da dieser das einzige Verbindungsglied zweier Netzwerksektoren ist und somit der gesamte Netzverkehr dieses Gerät passieren muss.

von Cisco angepasst. Somit wurde auch die Cisco ASA zumindest zum Teil eine Next Generation Firewall (NGFW). 2016 stellte Cisco dann seine eigene komplette NGFW – die Cisco ASA mit Firepower Diensten – vor. In Fachkreisen wird diese Firewall auch nur „Cisco Firepower“ genannt. Die „Cisco Firepower“ umfasst nun alle Cisco ASA Funktionalitäten und zusätzlich noch Cisco Advanced Malware Protection (AMP). Je nach Lizenzierung werden noch weitere Funktionen hinzugefügt, welche zwar die Sicherheit weiter anheben, aber individuell den Verkaufspreis stark steigern. Dadurch kann die „Cisco Firepower“ noch schneller und gründlicher Angriffe auf das Netzwerk erkennen. Die Ziele dieser Diplomarbeit sahen nicht zwingend die Konfiguration einer „Cisco Firepower“ vor, und das Budget ließ den Kauf eines physischen Gerätes auch nicht zu. Deswegen wurde von Cisco nur die ASA analysiert und konfiguriert.

6.4.2 Inspection Engine der Cisco ASA

Das Inspizieren des Netzverkehrs bei einer Cisco ASA funktioniert mittels eines objektorientierten Richtlinienaufbaus. Das bedeutet, dass eine konfigurierte Richtlinie an mehreren Stellen unterschiedlich angewandt werden kann. Dadurch reduziert sich der Konfigurationsumfang für eine gut durchplante Richtlinienstruktur drastisch. Standardmäßig gibt es eine global angewandte Policy Map, welche verschiedene Class Maps miteinander vereint. In diesen Class Maps ist dann enthalten, welche Form von Netzverkehr inwiefern individuell behandelt werden soll. Hier kann ein PDU basierend auf dessen Protokoll, einer greifenden Access Control List (ACL) oder aber auch aufgrund von Informationen aus OSI-Schicht sieben, abgefertigt werden.

Für die Informationsgewinnung durch Nutzdaten können zum Beispiel bei Domain Name System (DNS) PDUs bestimmte FQDNs oder Typen als Kriterium konfiguriert werden. DNS PDUs, die entweder alle oder zumindest eines – basierend auf der Konfiguration – der konfigurierten Merkmale aufweisen, werden dann je nach der jeweiligen Aktion gehandhabt. Anders gesagt, DNS PDUs, die den konfigurierten Fully-Qualified Domain Name (FQDN) aufzulösen versuchen oder einen bestimmten DNS Typen anfragen, werden individuell behandelt.

6.4.3 DNS-Doctoring auf einer Cisco ASA

Die Cisco ASA kann mit der richtigen Lizenzierung nicht nur den Datenbereich inspizieren, sondern sogar Änderungen darin vornehmen. Der Netzplan für die Erklärung der Funktion ist in Abbildung 6.1 auf der nächsten Seite zu sehen:

- Im privaten Netz steht ein DNS Server, der den FQDN eines internen Webservers auf eine private IP-Adresse auflöst.
- Die Server und die internen Geräte befinden sich im selben Netz.

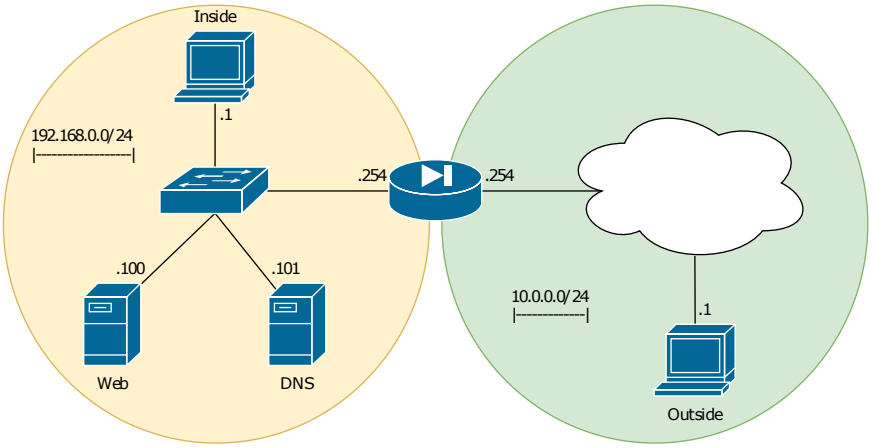

Abbildung 6.1: Der Aufbau für DNS Doctoring auf der Cisco ASA

- Auf der Firewall findet eine statische Netzwerkadressübersetzung des Web- und des DNS-Servers statt.

Sollte jetzt ein DNS Request aus dem privaten Netz an den DNS Server gelangen, dann transversiert dieser nicht die Firewall, und der FQDN wird normal auf die private IP-Adresse des Webservers aufgelöst. Sollte allerdings ein Gerät von außerhalb versuchen, den FQDN des Webservers aufzulösen, so wird diese Anfrage auf dem Hinweg durch die Firewall hindurchgelassen, und auf dem Rückweg wird die aufgelöste Adresse im Datenbereich des DNS-Pakets auf die öffentliche geändert. Die Funktion, welche dies ermöglicht, nennt sich DNS Doctoring und ist optional konfigurierbar.

Um DNS Doctoring zu konfigurieren, muss lediglich bei der statischen Adressübersetzung des Webservers das Schlüsselwort *dns* angehängt werden. Zusätzlich muss in den vordefinierten Richtlinien dns inspiziert werden. Beides wird durch folgende Befehle ermöglicht:

```
nat (inside,outside) source static 192.168.0.100 10.0.0.100 dns
policy-map global_policy
  class inspection_default
    inspect dns
```

6.5 DPI auf einer FortiGate

6.5.1 Einleitung zur FortiGate

Die FortiGate ist wie schon die Cisco ASA oder „Cisco Firepower“ eine NGFW. Sie kann somit natürlich auch den Netzverkehr basierend auf Informationen aus der OSI-

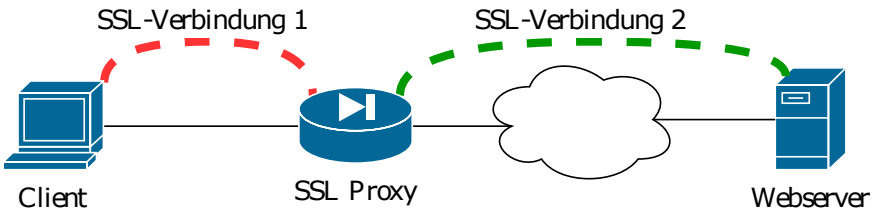

Abbildung 6.2: Die Funktionsweise von SSL-Inspection

Schicht sieben klassifizieren und individuell behandeln. Weiters ist auch diese Firewall in der Lage, Daten im Nutzbereich der PDUs umzuschreiben. Allerdings können auch mit der FortiGate keine statistischen Aussagen über das Datenaufkommen getroffen werden.

6.5.2 SSL-Inspection auf einer FortiGate

Die FortiGate Firewall von Fortinet ist unter anderem auf ihre trivial zu konfigurierende SSL-Inspection spezialisiert. Dazu stellt die Firewall, wie in Abbildung 6.2 zu sehen, einen SSL Proxy dar. Zur Konfiguration dieser Funktion wird im Webinterface entweder das offizielle Fortinet Zertifikat für diesen Zweck heruntergeladen oder aber ein firmeneigenes importiert. Allerdings hat das gewonnene Wissen maßgeblich die Funktionsweise der DPI Argos beeinflusst. Wenn der Client jetzt zum Beispiel mit HTTPS auf eine Webseite zugreifen will, dann baut er die SSL Verbindung nicht direkt mit dem Webserver, sondern nur mit der FortiGate auf. Dafür muss der Client natürlich dem Zertifikat der Firewall vertrauen – es muss auch auf ihm importiert werden. Die FortiGate wiederum baut eigens eine SSL Verbindung mit dem Webserver auf und besitzt nun bei der Datenübertragung Einblick über die kompletten Nutzdaten, da der Webserver meint, dass die FortiGate der Client sei. Die Firewall stellt hier also einen Man-in-the-Middle dar. Diese Funktion ist natürlich aufgrund der selbstständig durchführenden Verschlüsselung massiv ressourcenaufwendig und sollte deswegen nur wenn unbedingt notwendig zum Einsatz kommen.

6.6 DPI in der Diplomarbeit

6.6.1 Einleitung zu DPI in der Diplomarbeit

Bei der Diplomarbeit Argos wird DPI nicht auf den Firewalls eingesetzt, da diese zwar sehr effektiv filtern, aber keine statistischen Auswertungen erheben können. Allerdings wurde die grundlegende Logik, die durch DPI auf Firewalls zum Einsatz kommt, auf die Anwendung in der Diplomarbeit analog dazu implementiert. Deswegen benötigt man ein anderes DPI-fähiges Gerät an ebenfalls einem kritischen Punkt im Netz. Dafür

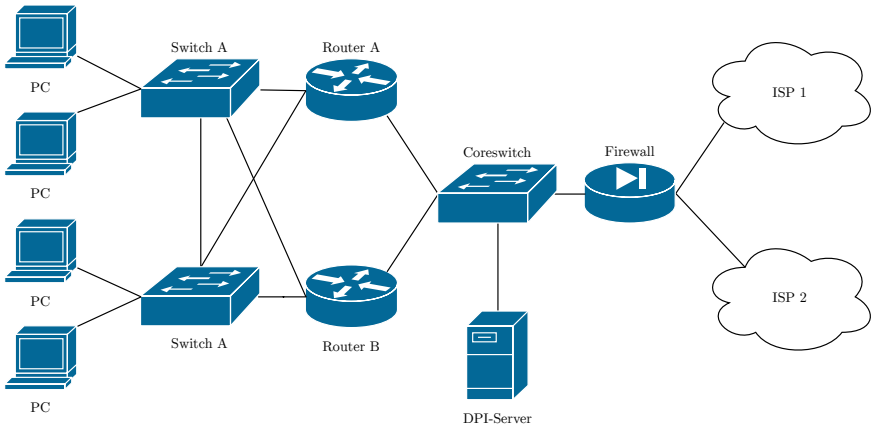

Abbildung 6.3: Der schematisch logische Netzaufbau des Netzes bei Argos

kann zum Beispiel ein Switch vor die Firewall eingebaut werden, welcher den gesamten Netzwerkverkehr an einen DPI-Server weiterleitet. Somit stellt der Switch den kritischen Punkt dar, und der Server erhält ebenso Einblick in den gesamten Netzverkehr. Im Falle der Diplomarbeit werden die Pakte von einem der Verarbeitungsmodule – dem Sniffer – aufgenommen und verarbeitet (siehe Kapitel 8.4 auf Seite 75).

Dieser Prozess wird im Fachkreis als *Port-Mirroring* bezeichnet. Eine visuelle Darstellung des zusätzlich eingebauten kritischen Punktes und des Servers kann in Abbildung 6.3 betrachtet werden. Im Laufe der Diplomarbeit wurde sich mit der Konfiguration von Switches zweier verschiedener Hersteller befasst, und zwar mit Cisco und HP Switches.

Im Kapitel 6.3 auf Seite 34 wurde erwähnt, dass DPI auf Firewalls die Leistung des Geräts massiv beeinträchtigt und im schlimmsten Fall zu einem Konnektivitätsverlust führen kann. Durch das Implementieren eines separaten Servers wird diese Last von der Firewall abgenommen. Dadurch wird die Netzkonnektivität der übrigen Geräte bei Überlastung des Servers nicht beeinträchtigt. Der einzige Nachteil hierbei ist nun, dass bei einem Angriff nicht automatisiert reagiert werden kann. Der Administrator muss nach einer erhaltenen Meldung selbstständig Aktionen setzen, um den Angriff zu stoppen.

6.6.2 Konfiguration eines Port-Mirrors auf einem HP Switch

Bei HP Switches tragen die Eingangsinterfaces die Bezeichnung Monitorport und das Ausgangsinterface die Bezeichnung Mirrorport. Der Datenverkehr wird also von jedem Monitorport auf den Mirrorport gespiegelt. Als Monitorports können entweder physikalische Schnittstellen oder aber VLANs konfiguriert, nicht aber können die beiden Technologien bei ein und demselben Spiegelungsvorgang verwendet werden. Als Mirrorport kann pro Spiegelungsvorgang lediglich ein physikalisches Interface verwendet werden und die Anzahl der unterschiedlichen möglichen Spiegelungsvorgänge sind von

der Version des Geräts und vom Gerät selbst begrenzt. Die Konfiguration eines Monitor- und Mirrorports lautet bei einem Gerät, das nur einen Spiegelungsvorgang gleichzeitig unterstützt, wie folgt:

- Monitorport: `interface VLAN-ID | INTERFACE-ID vlan monitor`
- Mirrorport: `mirror-port INTERFACE-ID`
- Überprüfung: `show monitor`

6.6.3 Konfiguration eines Port-Mirrors auf einem Cisco Switch

Bei Cisco Switches werden die Eingangsinterfaces als Sourceports und die Ausgangsinterfaces als Destinationports bezeichnet. Die Terminologie ändert allerdings kaum die Einschränkungen von Monitor- oder Mirrorports im Vergleich zu HP Geräten. Allerdings unterstützen beinahe alle verbreiteten Cisco Switches unterschiedliche Spiegelungsvorgänge gleichzeitig – hier wird von unterschiedlichen Monitorsessions gesprochen. Außerdem kann auf Cisco Geräten optional definiert werden, ob nur eingehende (RX-), ausgehende (TX-) oder aber bidirektionale (BOTH-) Frames gespiegelt werden sollen. Die Konfiguration ändert sich durch die variable Anzahl an Monitorsessions und durch die zusätzlich konfigurierbaren Möglichkeiten geringfügig:

- Sourceport: `monitor session SESSION-ID source INTERFACE-ID [both | rx | tx]`
- Mirrorport: `monitor session SESSION-ID destination INTERFACE-ID`
- Uberprüfung: `show monitor session SESSION-ID`

6.6.4 DPI-Server in der Diplomarbeit

Der DPI-Server von Argos ermöglicht es nun, statistische Auswertungen über das Netzverhalten durchzuführen und die Ergebnisse zu analysieren. Somit kann der DPI-Server statistische Kenngrößen über das Netzwerk sammeln und in der Datenbank speichern (siehe Kapitel 8.4 auf Seite 75). Dafür akzeptiert dieser alle Frames, die an seine Netzwerkschnittstelle gelangen, auch jene, die nicht für ihn bestimmt sind. Um das zu ermöglichen, muss das Interface in den sogenannten *Promicious-Mode* versetzt werden.

Die genaue Konfiguration und Beschreibung zur Programmierung und Durchführung von DPI in der Diplomarbeit findet sich im Kapitel 8.4 auf Seite 75. Hier wird lediglich ein Überblick darüber vermittelt, welche Informationen aus Deep Packet Inspection gewonnen werden können. Vorrangig beschränkt sich die Anwendung von DPI darauf, den Inhalt von DNS Anfragen und DNS Antworten zu extrahieren und zu interpretieren.

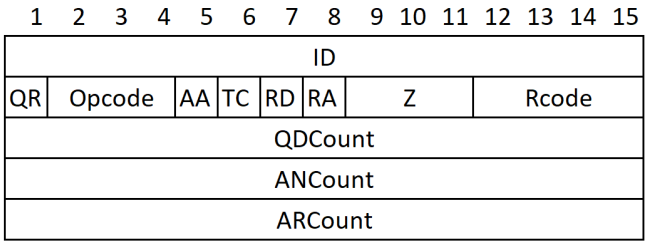

Abbildung 6.4: Der Aufbau eines DNS Headers

Um verstehen zu können, welche Informationen aus DNS Paketen gewonnen werden können, muss zuerst einmal das Paketformat von DNS bekannt und verstanden sein. Grundsätzlich wird jede DNS Anfrage oder Antwort mit einer geräteindeutigen Identifikator (ID) versehen, damit eine Antwort eindeutig einer Anfrage zuordnet werden kann. Weiters sind zwei Bytes im DNS-Kopf für diverse Markierungen und für eine einfachere Bedienung reserviert. Zudem werden im Kopfbereich auch die Anzahl an Anfragen und Antworten festgelegt.

In der Technik wird nicht zwischen einer DNS Anfrage oder einer DNS Antwort unterschieden. Das bedeutet, dass eine DNS Antwort lediglich an die DNS Anfrage angehängt und somit wieder in einem PDU übertragen wird. Ein DNS Antwort PDU unterschiedet sich somit nur durch andere Markierungen im Paketkopf und durch die zusätzlichen Nutzdaten. Der Datenbereich beinhaltet dann die im Paketkopf angegebene Anzahl an Anfragen, beziehungsweise Antworten. Mithilfe der DPI von DNS PDUs kann nun unter Zuhilfenahme von OSI-Schicht drei Informationen bestimmt werden, welche IP-Adresse, zu welchem Zeitpunkt, welchen FQDNs auflösen wollte und welche Antwort zu der Anfrage retourniert wurde. Mithilfe dieser Informationen kann nun auch eine zeitliche Änderung der IP Adressen zu den jeweiligen FQDNs aufgeschlüsselt werden. Man erhält somit Einsicht über die DNS Historie des Internets.

Das genaue Header Format von DNS PDUs findet sich in Abbildung 6.4 wieder und wird folglich erklärt:

ID
: Eindeutige Transaktionsidentifikator

Flags
: Die nächsten zwei Bytes sind für die automatische Verarbeitung Kennzeichnungen. Am wichtigsten hier ist der RCODE, welcher angibt, ob die Antwort erfolgreich war und falls nicht, den Grund dafür einfügt.

QDCount
: Der *QueryDomainCount* gibt an, wie viele FQDNs aufgelöst werden sollen.

ANCount

Der *AnswerCount* gibt an, wie viele Antworten in dieser Transaktion enthalten sind.

NSCount

Der *NameServerCount* gibt an, wie viele andere DNS Server für diese Domain verantwortlich sind.

ARCount

Der *AdditionalRessourceCount* gibt an, wie viele Antworten zusätzliche Informationen beinhalten und somit von den tatsächlichen Antworten getrennt sind.

7 Speichern von Logdaten

7.1 Einleitung in die Speicherung von Logdaten

Um enormer unstrukturierter Datenmengen, wie jenen dieser Diplomarbeit (siehe Kapitel 8.2 auf Seite 60) Herr zu werden, wird ein geeignetes Datenbankmanagementsystem (DBMS) benötigt. Zu beachten ist, dass im laufenden Betrieb mehrere hundert Einträge pro Sekunde von der Datenbank (DB) verarbeitet werden müssen, weil in großen Netzwerken durchaus mit einem solchen – auf den ersten Blick übertriebenen – Daten- und Nachrichtenaufkommen zu rechnen ist. Bei Abfragen aus der DB ist es wichtig, dass vor allem nach Zeitpunkt der Nachrichten gefiltert bzw. gruppiert werden muss. Eine für eine DB eines SIEMs typische Abfrage wäre z. B. alle Nachrichten eines gewissen Typs der letzten sieben Tage anzeigen zu lassen.

Da die Daten selbst keine geeigneten Felder zur Bildung eines Primärschlüssels besitzen, muss dieser künstlich erzeugt werden. Die einzigen Felder, die bei allen Einträgen der DB vorhanden sind, sind die folgenden:

- Der Zeitpunkt des Eintreffens der Nachricht (*Receive Timestamp*)
- Die IP-Adresse des Gerätes, welches die Nachricht gesendet hat (*Origin*)
- Die Nachricht selbst (*Message*)

Bis auf diese Felder gibt es rein informationstheoretisch sonst keine Gemeinsamkeiten. Natürlich lassen sich durch die Analyse dieser drei Felder weitere Felder erzeugen:

- Die Art des Gerätes, welches die Nachricht gesendet hat (*Platform*) durch
  (a) Abfragen der IP-Adresse (*Origin*) in einer vordefinierten Datenstruktur, welche die *Platform* sicher zurückgibt oder
  (b) Analyse der Syntax der Nachricht und Erraten der passenden *Platform*
- Der Typ der Nachricht (*Type*) durch
  (a) Analyse der Semantik der Nachricht[1] und
  (b) Verknüpfen mit der *Platform*
- Der Zeitpunkt des Erzeugens der Nachricht (*Timestamp*) durch
  (a) Analyse der Semantik der Nachricht

[1] Wird in der gesamten Diplomarbeit nicht genauer behandelt. Durch die Analyse einer Nachricht, ob gewisse Phrasen vorkommen, etc. könnte auch der Typ festgestellt werden.

In der Diplomarbeit wurde letztendlich das DBMS MongoDB zur Datenspeicherung ausgewählt. MongoDB ist eine NoSQL DB und genau das ermöglicht es, JSON Dokumente[2] sehr effizient zu speichern. Der Zugriff auf die DB mittels Python ist sehr einfach, und das war ein ausschlaggebender Punkt dafür, dass MongoDB verwendet wurde.

7.2 Optimierung von MongoDB

7.2.1 Überblick zur Optimierung von MongoDB

Da dokumentenorientierte Datenbanken im Allgemeinen nicht auf hohe Effizienz ausgelegt sind, ist es besonders wichtig, die DB mit allen zur Verfügung stehenden Mitteln zu unterstützen, die Daten zu ordnen und zu speichern. In der Diplomarbeit werden grundsätzlich zwei Möglichkeiten genutzt, um die DB zu unterstützen:

- Das Erstellen von Indizes
- Die Aufrechterhaltung der Verbindung (Session) zur DB

7.2.2 Indizes in MongoDB

Der einfachste Weg der DB überflüssige Arbeit abzunehmen, ist anzugeben, nach welchen Feldern im Dokument man die Abfragen orientieren möchte – im Allgemeinen bekannt als Index (Abbildung 7.1 auf der nächsten Seite). Wie in der Abbildung zu erkennen ist, ist ein Index ein Verzeichnis bzw. Nachschlagewerk für das DBMS, in dem es schneller nach den geforderten Informationen suchen kann. Bei einem Index, welcher nur auf einen Zeitstempel angewendet wird, werden die Dokumente in der DB dem Zeitstempel nach geordnet und der Primärschlüssel im Index vermerkt. So muss die DB bei einer Abfrage, welche alle Ereignisse vor einem gewissen Zeitpunkt abfragt, nur diesen einen Zeitpunkt im Index finden und alle Dokumente vor diesem Punkt zurückgeben – Es muss nicht jedes einzelne Dokument auf den Zeitstempel überprüft werden, und so kann die Antwortzeit der DB erheblich verkürzt werden. [13]

In dem Fall der Diplomarbeit war es wichtig, für folgende Felder einen Index zu erstellen:

- Dem Zeitpunkt des Erzeugens der Nachricht (*Timestamp*)
- Dem Typ der Nachricht (*Type*/*Platform*)
- Welches Gerät die Nachricht gesendet hat (*Origin*)

[2] hier: Eintrag in der DB; Synonyme: Row, Zeile

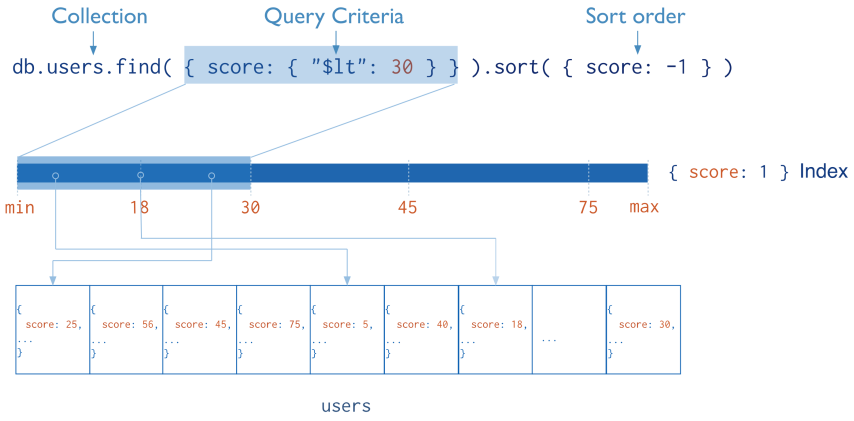

Abbildung 7.1: Veranschaulichung eines Indizes in MongoDB [13]

Die zu indizierenden Felder sind quasi dieselben Felder, die durch einfache Analysen der Nachricht erzeugt werden können. Dies liegt unter anderem daran, dass jede Nachricht diese Felder hat und deswegen leicht danach sortiert werden kann und daran, dass diese Felder ohne großen Aufwand aus der Nachricht ausgelesen werden können und so vom Zeitpunkt des Einfügens in die DB an vorhanden sind.

Das Erstellen eines Index funktioniert in MongoDB recht einfach. Im folgenden Beispiel werden die Indizes für die Collection[3] der verarbeiteten Daten erstellt:

```
db.logs.createIndex({timestamp: 1})
db.logs.createIndex({origin.addr: 1, type: 1})
db.logs.createIndex({type: 1})
```

- Der erste Index wird nur auf den *Timestamp* angewendet,
- der zweite auf primär *Origin* und sekundär *Type*,
- der dritte nur auf den *Type*.

Mit diesen drei Indizes kann die Collection der verarbeiteten Daten schnell und effizient auf zeitlich sortierte und vom Typ abhängige Dokumente durchsucht werden.

7.2.3 Aufrechterhalten der Datenbankverbindung

Jedes Mal, wenn sich ein Programm oder Python-Skript mit MongoDB verbindet, dauert es ungefähr 1–2 Sekunden bis die Initiierung der Verbindung abgeschlossen wurde. Man würde nun denken, dass diese wenigen Sekunden nicht der Rede wert wären, doch wenn die Abfrage der DB selbst nur einige wenige Millisekunden dauert, verlängert sich die Gesamtdauer des Prozesses um ein Hundertfaches. Die Lösung für

[3] in MongoDB: Tabelle

dieses Problem ist, einen Dienst zu erstellen, welcher dauerhaft mit der DB verbunden ist – daher die Initiierung nur ein einziges Mal durchführen muss – und andere Prozesse nur mehr über den Dienst auf die DB zugreifen.

Dieser Dienst heißt in der Diplomarbeit *Searcherd* (*Searcher Daemon*) und bietet über einen geöffneten TCP-Port anderen Programmen die Möglichkeit, Daten aus der DB abzufragen. Das ist aber nicht die einzige Funktion des *Searchers* – er bietet auch die Möglichkeit, die Daten, die die DB zurückgibt, etwas zu vereinfachen, um den aufrufenden Programmen diese repetitive Arbeit abzunehmen (siehe Kapitel 9.1 auf Seite 87).

7.3 Datenbankstruktur in der Diplomarbeit

In diesem Teil werden aus Dokumentationszwecken alle erstellten Collections aufgelistet und beschrieben. Für die Diplomarbeit wurden für die folgen Bereiche Collections in MongoDB erstellt:

- Rohdaten
- Verarbeitete Daten
- Anmeldungen (*Logins*)
- Zwischenfälle (*Incidents*)
- Historische DNS Daten

7.3.1 Collection der Rohdaten

In dieser Collection werden ohne weitere Verarbeitungsschritte alle empfangenen Nachrichten sofort gespeichert. Durch diese Vorgehensweise sollen Datenverluste, die bei Fehlern des verarbeitenden Prozesses auftreten, vermieden werden, indem das Programm, das für die Speicherung zuständig ist (siehe Kapitel 7.4 auf Seite 57), möglichst klein gehalten wird. Durch die Minimierung der Programmlänge kann auch die Anzahl an potentiellen Fehlern in diesem Programm möglichst geringgehalten werden. Es folgen alle Felder, welche in der Collection der Rohdaten in jedem Dokument enthalten sind.

timestamp (Date Objekt)
: Der Zeitpunkt, an dem die Nachricht empfangen wurde.

raw (Binärdaten)
: Die Nachricht, wie sie Byte für Byte empfangen wurde.

proto (String)
Das Protokoll, mit dem die Nachricht empfangen wurde. Dieses Feld kann die Werte `syslog` oder `snmp` annehmen.

type (String)
Typ oder ID der Nachricht in der Form `vendor:platform/msgId`.

origin.addr (IP-Adresse, String)
Die IP-Adresse des sendenden Gerätes.

origin.vendor (String)
Der Hersteller des sendenden Gerätes.

origin.platform (String)
Die Plattform des Herstellers des sendenden Gerätes.

tried (Boolean)
True, wenn der *Indexer* bereits versucht hat, diese Nachricht zu verarbeiten.

indexed (Boolean)
True, wenn der *Indexer* die Nachricht bereits verarbeitet hat.

Ein Auszug eines Eintrags der Collection `raw`:

```
{
    "timestamp" : ISODate("2020-01-01T12:34:56.789Z"),
    "raw" : BinData(0, "<BASE64>"),
    "proto" : "syslog",
    "origin" : {
        "addr" : "192.168.1.1",
        "vendor" : "cisco",
        "platform" : "ASA"
    },
    "tried" : true,
    "indexed" : true,
    "type" : "cisco:asa/609002"
}
```

7.3.2 Collection der verarbeiteten Daten

In dieser Collection werden nach der Verarbeitung der Daten aus der Collection der Rohdaten die ausgewerteten Felder gespeichert. Viele der Felder in dieser Collection sind ähnlich zu denen der Collection der Rohdaten. Das liegt daran, dass die Information

dieser Felder für beide Collections wichtig ist. Das *data* Feld beinhaltet je nach Typ der Nachricht weitere Sub-Felder, Sub-Sub-Felder usw., welche nicht standardisierbar sind und deswegen in dieser Auflistung nicht aufscheinen.

timestamp (Date Objekt)
: Entweder ein Zeitstempel, der aus der Nachricht selbst ausgelesen wurde, oder, wenn kein Zeitstempel gefunden wurde, oder das Feature deaktiviert wurde, der Zeitpunkt, an dem die Nachricht empfangen wurde.

proto (String)
: Das Protokoll, mit dem die Nachricht empfangen wurde. Dieses Feld kann die Werte `syslog` oder `snmp` annehmen.

origin.addr (IP-Adresse)
: Die IP-Adresse des sendenden Gerätes.

origin.vendor (String)
: Der Hersteller des sendenden Gerätes.

origin.platform (String)
: Die Plattform des Herstellers des sendenden Gerätes.

origin.hostname (String)
: Der Hostname des sendenden Gerätes, falls dieser aus der Nachricht ausgelesen werden kann.

msg (String)
: Eine kurze beschreibende Nachricht des Events, falls eine aus der Nachricht ausgelesen werden kann. Dieses Feld beinhaltet **nicht** die originale Nachricht.

data (Objekt)
: Aus der Nachricht ausgelesene Felder. Diese Felder unterscheiden sich von Nachrichtentyp zu Nachrichtentyp.

severity (Integer)
: Das Severity-Feld aus dem Syslog-Protokoll.

facility (Integer)
: Das Facility-Feld aus dem Syslog-Protokoll.

type (String)
: Typ oder ID der Nachricht in der Form `vendor:platform/msgId`.

flags.linked (Boolean)
True, wenn die Nachricht bereits durch den *Linker* auf Anmeldeversuche hin analysiert wurde.

Ein Auszug eines Eintrags der Collection `logs`:

```
{
    "timestamp" : ISODate("2019-09-18T21:13:23.873Z"),
    "proto" : "syslog",
    "origin" : {
        "addr" : "192.168.1.1",
        "vendor" : "cisco",
        "platform" : "ASA",
        "hostname" : "ASA-Edge-Firewall"
    },
    "msg" : "Deny TCP (no connection) from 208.91.113.103/443 to
             10.70.4.1/3654 flags RST  on interface outside",
    "data" : {
        "ipp_in" : {
            "addr" : "208.91.113.103",
            "port" : 443
        },
        "ipp_out" : {
            "addr" : "10.70.4.1",
            "port" : 3654
        },
        "tcp_flags" : "RST",
        "interface_name" : "outside"
    },
    "severity" : 20,
    "facility" : 6,
    "type" : "cisco:asa/106015"
}
```

7.3.3 Collection der Anmeldungen (Logins)

In dieser Collection werden die verschiedensten Anmeldeversuche der Benutzer im Netzwerk (z. B. Authentifizierung im WLAN oder AD) festgehalten. Durch regelmäßiges Durchsuchen der neuesten Nachrichten der Collection der verarbeiteten Daten und Verknüpfen dieser kann eine zeitliche Abfolge aller Anmeldeversuche erstellt werden. Anzumerken ist, dass diese Collection im Produktivbetrieb wesentlich kleiner ausfällt als die vorhergehenden – Logins geschehen zwar oft, aber bei weitem nicht so oft, wie Nachrichten über neue TCP Verbindungen erzeugt werden.

_id.username (String)
Der Nutzername des Benutzers, der versucht hat sich anzumelden.

_id.timestamp (Date Objekt)
Der Zeitpunkt, an dem der Anmeldeversuch stattgefunden hat.

mac (MAC-Adresse, String)
Die Media Access Control (MAC)-Adresse des Gerätes, an dem versucht wurde sich anzumelden.

ip (IP-Adresse, String)
Die IP-Adresse des Gerätes, an dem versucht wurde sich anzumelden.

apHostname (String)
Der Hostname des APs, an dem versucht wurde sich anzumelden.

apMac (MAC-Adresse, String)
Die MAC-Adresse des APs, an dem versucht wurde sich anzumelden.

ssid (String)
Die SSID des WLANs, in dem versucht wurde sich anzumelden.

Ein Auszug eines Eintrags der Collection `logins`:

```
{
    "_id" : {
        "timestamp" : ISODate("2019-09-16T11:28:18.730Z"),
        "username" : "lorenz.stechauner"
    },
    "mac" : "94:B8:6D:DB:DA:17",
    "ip" : "192.168.5.16",
    "apHostname" : "WLAN_ARGOS_1",
    "apMac" : "CC:D5:39:27:EA:B0",
    "ssid" : "ARGOSNET"
}
```

7.3.4 Collection der Zwischenfälle (Incidents)

In dieser Collection werden alle erkannten Zwischenfälle bzw. Angriffe festgehalten. Durch regelmäßiges Durchsuchen und Analysieren der neuesten verarbeiteten Nachrichten können einfache Angriffe auf oder kleinere Zwischenfälle im Netzwerk erkannt werden.

timestamp (Date Objekt)
Der Zeitpunkt, an dem der Zwischenfall stattgefunden hat.

type (String)
Die Art des Zwischenfalls. Dieses Feld kann die Werte `dns_tunnel`, `portscan` oder `login` annehmen.

Weitere Felder in dieser Collection sind vom Feld `type` abhängig.

(a) Falls `type` den Wert `dns_tunnel` annimmt:

bytes (Integer)
Die Anzahl der Bytes, die durch den DNS-Tunnel gewandert sind.

src (IP-Adresse, String)
Die Adresse des Gerätes, welches den DNS-Tunnel initiiert hat, also eine lokale Adresse.

dest (IP-Adresse, String)
Die Adresse, zu der der Tunnel aufgebaut wurde, also eine öffentliche Adresse.

(b) Falls `type` den Wert `portscan` annimmt:

attempts (Integer)
Die Anzahl an Ports, die abgetastet wurden.

src (IP-Adresse, String)
Die Adresse, von der aus der Portscan stattgefunden hat. Diese Adresse kann sowohl innerhalb als auch außerhalb des eigenen Netzwerks sein.

dest (IP-Adresse, String)
Die Adresse, auf der die Ports gescannt werden. Diese Adresse kann sowohl innerhalb als auch außerhalb des eigenen Netzwerks sein.

(c) Falls `type` den Wert `login` annimmt:

user (String/Objekt)
Identifiziert den Benutzer, der versucht sich anzumelden. Falls vorhanden, ein Objekt mit den Feldern `sid`, `username` und `domain`, wenn nicht, ist dieses Feld nur ein einfacher String.

attempts (Integer)
Die Anzahl an Anmeldeversuchen.

identifier (String)
Identifiziert das genaue Ereignis und ob der Anmeldeversuch erfolgreich war, oder nicht.

clientmachines (Array)
Ein Array aus entweder MAC-Adressen oder IP-Adressen, von welchen aus die Anmeldeversuche durchgeführt wurden.

Ein Auszug eines Eintrags des Typs `dns_tunnel` der Collection `incidents`:

```
{
    "timestamp" : ISODate("2020-01-05T15:42:45Z"),
    "type" : "dns_tunnel",
    "bytes" : 1320,
    "src" : "192.168.2.1",
    "dest" : "209.222.147.38"
}
```

Ein Auszug eines Eintrags des Typs `portscan` der Collection `incidents`:

```
{
    "timestamp" : ISODate("2019-09-25T15:28:29.066Z"),
    "type" : "portscan",
    "src" : "10.70.4.1",
    "dest" : "52.142.84.61",
    "attempts" : 8
}
```

Ein Auszug eines Eintrags des Typs `login` der Collection `incidents`:

```
{
    "user" : {
        "sid" : "S-1-5-21-4216098271-1549864735-2282191126-1683",
        "username" : "lorenz.stechauner",
        "domain" : "ARGOS"
    },
    "timestamp" : ISODate("2020-03-10T15:27:22.451Z"),
    "clientmachines" : [
        "94:B8:6D:DB:DA:17"
    ],
    "attemps" : 12,
    "type" : "login",
    "identifier" : "WLAN_WRONG_PASSWORD"
}
```

7.3.5 Collection der historischen DNS-Daten

7.3.5.1 Roh-DNS-Collection

Wie in der Collection für die Rohdaten der Nachrichten werden in dieser Collection alle empfangenen Daten Byte für Byte gespeichert. Falls es in Zukunft einmal nötig werden sollte, können anhand der Daten dieser Collection alle DNS-Pakete erneut verarbeitet werden.

timestamp (Date Objekt)
: Der Zeitpunkt, zu dem das DNS-Paket empfangen wurde.

msg (Binärdaten)
: Das gesamte IP-Paket, wie es Byte für Byte empfangen wurde.

Ein Auszug eines Eintrags der Collection `dns_raw`:

```
{
    "timestamp" : ISODate("2019-11-02T09:09:56.773Z"),
    "msg" : BinData(0,"<BASE64>")
}
```

7.3.5.2 DNS-Collection

In dieser Collection werden alle DNS-Anfragen und -Antworten gespeichert. Mithilfe dieser Collection kann jede einzelne DNS-Anfrage und -Antwort auf einen gewissen Host zurückgeführt werden.

timestamp (Date Objekt)
: Der Zeitpunkt, zu dem das DNS-Paket empfangen wurde.

src_ip (IP-Adresse, String)
: Die Adresse des Geräts, welches die DNS-Anfrage gestellt hat.

dest_ip (IP-Adresse, String)
: Die Adresse des Geräts, zu welchem die DNS-Anfrage gestellt wurde.

type (String)
: Der Typ der Anfrage bzw. der Antwort. Dieses Feld kann entweder die Werte `DNS-Query`, oder `FULL-DNS-REPLY` annehmen.

src_port (Integer)
Der Quell-Port, von dem die Anfrage aus gestellt wurde.

dest_port (Integer)
Der Ziel-Port, zu dem die Anfrage geschickt wurde. Dieses Feld hat üblicherweise den Wert `53`.

transaction_id (Integer)
Die Transaktions-ID der Abfrage.

requests (Array)
Ein Array, in dem alle abgefragten FQDNs gespeichert werden:

fqdn (FQDN, String)
Der Fully-Qualified Domain Name (FQDN), der abgefragt wird.

record (String)
Der DNS-Record-Typ, der abgefragt wird. Dieses Feld kann `A`, `AAAA`, `TXT` usw. annehmen.

class (Integer)
Die Klasse bzw. Protokollgruppe, zu der die Abfrage gehört. Dieses Feld nimmt immer den Wert `1` (Internet) an.

replies (Array)
Ein Array, in dem alle zur Anfrage zugehörigen Antworten gespeichert werden:

answer_record (String)
Der DNS-Record-Typ, der abgefragt wird. Dieses Feld kann `A`, `AAAA`, `TXT` usw. annehmen.

answer_class (Integer)
Die Klasse bzw. Protokollgruppe, zu der die Abfrage gehört. Dieses Feld nimmt immer den Wert `1` (Internet) an.

ttl (Integer)
Die Time to Live (TTL) des DNS-Records.

length (Integer)
Die Länge der Antwort in Byte. Bei A-Records wird die Länge der rohen Integer Bytes gezählt.

answer (String)
Die letztendliche Antwort des DNS-Servers.

Ein Auszug eines Eintrags aus der Collection `dns`:


```
{
    "timestamp" : ISODate("2019-10-30T13:04:55.344Z"),
    "src_ip" : "192.168.2.1",
    "dest_ip" : "208.91.112.53",
    "type" : "DNS-Query",
    "src_port" : 3067,
    "dest_port" : 53,
    "transaction_id" : 19535,
    "requests" : [
        {
            "fqdn" : "softwareupdate.vmware.com",
            "record" : "A",
            "class" : 1
        }
    ],
    "replies" : [
        {
            "answer_record" : "CNAME",
            "answer_class" : 1,
            "ttl" : 3350,
            "length" : 34,
            "answer" : "esd751.vmware.com.ds.edgekey.net"
        },
        {
            "answer_record" : "CNAME",
            "answer_class" : 1,
            "ttl" : 8,
            "length" : 23,
            "answer" : "e751.dscd.akamaiedge.net"
        },
        {
            "answer_record" : "A",
            "answer_class" : 1,
            "ttl" : 10,
            "length" : 4,
            "answer" : "23.59.156.48"
        }
    ]
}
```

7.3.5.3 DNS-Records-Collection

Mithilfe der DNS-Records-Collection kann ein historischer Verlauf des gesamten Domain Name System gebildet werden. Zu beachten ist, dass nur Einträge verarbeitet werden können, die zumindest einmal von einem Host angefragt wurden, da sonst keine Daten zur Verfügung stehen.

timestamp (Date Objekt)
: Der Zeitpunkt, zu dem die Antwort des Servers empfangen wurde.

request_fqdn (FQDN, String)
: Der Fully-Qualified Domain Name (FQDN), welcher angefragt wurde.

server (IP-Adresse, String)
: Die Adresse des Servers, die Anfrage verarbeitet hat und eine Antwort zurückgeliefert hat.

client (IP-Adresse, String)
: Die Adresse des Hosts, der die Anfrage geschickt hat.

answers (Array)
: Ein Array, in dem alle zur Antwort gehörenden Records gespeichert werden:

  answer_record (String)
  : Der DNS-Record-Typ, der abgefragt wird. Dieses Feld kann `A`, `AAAA`, `TXT` usw. annehmen.

  answer_class (Integer)
  : Die Klasse bzw. Protokollgruppe, zu der die Abfrage gehört. Dieses Feld nimmt immer den Wert `1` (Internet) an.

  ttl (Integer)
  : Die Time to Live (TTL) des DNS-Records.

  length (Integer)
  : Die Länge der Antwort in Byte. Bei A-Records wird die Länge der rohen Integer Bytes gezählt.

  answer (String)
  : Die letztendliche Antwort des DNS-Servers.

Ein Auszug eines Eintrags aus der Collection `dns_records`:

```
{
    "timestamp" : ISODate("2019-10-30T13:04:40.018Z"),
    "request_fqdn" : "softwareupdate.vmware.com",
    "server" : "208.91.112.52",
    "client" : "192.168.2.1",
    "answers" : [
        {
            "answer_record" : "CNAME",
            "answer_class" : 1,
            "ttl" : 3369,
            "length" : 34,
            "answer" : "esd751.vmware.com.ds.edgekey.net"
        }
    ]
}
```

7.4 Collector

Der *Collector* ist das Modul, welches auf eintreffende Syslog- oder SNMP-Trap-Nachrichten wartet und diese in der Collection der Rohdaten abspeichert. Seine einzige Aufgabe ist es, die empfangenen Nachrichten Byte für Byte in der Rohdaten-Collection abzulegen, dazu den Zeitstempel, das Protokoll und die IP-Adresse. Durch eine beiliegende Konfigurationsdatei (`collector.conf`) kann der *Collector* anhand der IP-Adresse auch die Plattform und den Hersteller des Gerätes zum Eintrag in die Collection hinzufügen. Um als Programm auf zwei oder mehr Ports gleichzeitig und asynchron Daten empfangen zu können, muss für jeden Port ein einzelner Thread samt eigenen Socket verwendet werden. Das dadurch entstehende Multithreading stellt auch hinsichtlich der Modularität einen großen Vorteil dar.

Das Programm `collector.py` wird ohne weitere Optionen ausgeführt. Sofort nach dem Start wird die Verbindung zur DB aufgebaut, danach werden die Threads der jeweiligen Ports gestartet und warten auf eingehende Verbindungen.

8 Verarbeiten von Logdaten

8.1 Indexer

Der *Indexer* ist ein modular aufgebautes Python-Programm, welches periodisch die Collection der Rohdaten nach noch nicht verarbeiteten Daten durchsucht und anschließend verarbeitet. Um eine Minimierung des Datenverlustes zu bewirken, werden die Daten nicht sofort beim Eintreffen verarbeitet, sondern als erstes gespeichert, denn durch die Komplexität des *Indexers* kann ein Programmabsturz nicht ausgeschlossen werden. Die Aufgabe des *Indexers* ist es, das `data`-Feld zu befüllen. Im `data` Feld werden alle Informationen gespeichert, die bei verschiedenen Formaten anders strukturiert sind und deshalb nicht vereinheitlicht werden können.

Das Programm `indexer.py` wird folgendermaßen verwendet:

```
./indexer.py [{-r | -u}] [-i] [-A] [-s ADDRESS] [-t TYPE[,...]] [-T
TYPE[,...]]
```

Ohne `-u` oder `-r`
: Im Normalfall wird versucht, alle Einträge zu verarbeiten, bei welchen eine Verarbeitung noch nicht versucht wurde.

`-r, --retry`
: Es wird versucht, alle Einträge zu verarbeiten, bei welchen die Verarbeitung bereits fehlgeschlagen hat.

`-u, --update-all`
: Alle Einträge der Rohdaten-Collection werden nochmals bzw. erstmals verarbeitet. Die Verarbeitung aller Einträge kann eine sehr lange Zeit in Anspruch nehmen.

`-i, --ignore-timestamps`
: Es werden alle Zeitstempel, die in den Nachrichten selbst vorkommen, ignoriert. Nur der Zeitpunkt, zu dem die Nachricht eingetroffen ist, wird weiter verwendet. Ohne diesen Parameter werden zum Setzen von `timestamp` Zeitstempel aus den Nachrichten verwendet. Wenn auf den Netzwerkgeräten kein einheitlicher Network Time Protocol (NTP) Server verwendet wird, können die Zeitstempel der verschiedenen Geräte Inkonsistenzen aufweisen. Falls in den Nachrichten

keine Zeitstempel mitgeschickt wurden, wird der Zeitstempel des Eintreffens der Nachrichten verwendet.

`-A, --no-anonymization`
Der *Anonymizer* wird nicht aufgerufen und dadurch werden Benutzer- und Hostnamen sowie IP- und MAC-Adressen nicht anonymisiert. Dieser Parameter ist v. a. zu Testzwecken sehr hilfreich.

`-s, --source`
Es werden nur Einträge verarbeitet, die als Quell-Adresse (`source`) die angegebene IP-Adresse eingetragen haben. Dieser Parameter wird überwiegend dann eingesetzt, wenn einzelne Module des *Indexers* verbessert oder überarbeitet werden und die Änderungen nur gewisse Log-Typen bzw. Geräte betreffen.

`-t, --types`
Es werden nur Einträge verarbeitet, die einen der angegebenen Typen, Plattformen oder Herstellern haben. Es können mehrere Typen kommasepariert angegeben werden. Der Typ, die Plattform oder der Hersteller wird erst während des Verarbeitens festgestellt. Je nachdem, ob diese/r zu der angegebenen Liste passt oder nicht, wird in der Logdaten-Collection der dazugehörende Eintrag aktualisiert.

`-T, --indexed-types`
Es werden nur Einträge verarbeitet, dessen Typ in der angegebenen kommaseparierten Liste vorkommt. Im Gegensatz zum Parameter `-t` wird hier das `type`-Feld in der Rohdaten-Collection verwendet, daher ist nach Möglichkeit dieser Parameter gegenüber `-t` zu bevorzugen.

Auch der *Indexer* ist multithreading-fähig, d. h. dass immer pro Verarbeitung einer Nachricht ein eigener Thread gestartet wird. So soll v. a. auf Multicore-Prozessoren die Gesamtgeschwindigkeit des *Indexers* erhöht werden.

8.2 Verarbeiten von Syslog-Nachrichten

Die wichtigste Aufgabe des *Indexer* ist die Verarbeitung von Syslog-Nachrichten in eine maschinenlesbare Form – im Falle dieser Diplomarbeit JSON. Bei Syslog-Nachrichten ist dieser Schritt von besonderer Wichtigkeit, da in den Spezifikationen des Protokolls (siehe Kapitel 5.2.1 auf Seite 16) keine einheitliche Handhabung diverser Informationen vorzufinden ist. Aus diesem Grund müssen Syslog-Formate verschiedenster Hersteller erst in die Module des *Indexers* eingearbeitet werden, wenn sie noch nicht vorhanden sind.

In der Diplomarbeit ist es vorgesehen, dass der *Indexer* Nachrichten von folgenden Geräten bzw. Herstellern verarbeiten kann:

- Cisco ASA (Syslog), siehe Kapitel 8.2.2 auf der nächsten Seite
- Cisco WLC (Syslog und SNMP-Trap), siehe Kapitel 8.3 auf Seite 74
- Fortinet FortiGate (Syslog), siehe Kapitel 8.2.3 auf Seite 72
- Microsoft Windows Server (Syslog), siehe Kapitel 8.2.4 auf Seite 72

Wie unschwer im nachstehenden Vergleich zu erkennen ist, weichen die praktischen Implementierungen der Hersteller bis zu einem gewissen Grad vom Standard ab. Vor allem die unterschiedlichen Auffassungen der Zeitstempel (deutsches oder englisches Format, etc.) führen bei der automatisierten Verarbeitung zu erhöhtem Aufwand.

RFC 5424 [6]
`<`*Prio*`>` *Version Timestamp Hostname App-Name ProcId MsgId* ...

Solarwinds-Log-Format (Windows Server, ...)
`<5>Jän 01 12:34:56 Srv01.local MSWinEventLog ...`

Fortinet-Log-Format (ForitGate, ...)
`<189>date=2020-01-01 time=12:34:56 devname=DMZ-Firewall ...`

Cisco-Log-Format (ASA)
`<166>Jan 01 2020 12:34:56 Edge-Firewall : %ASA-6-106015: ...`

Cisco-Log-Format (WLC)
`<131>ARGOS-WLC: *spamReceiveTask: Jan 01 12:34:56.789:`
`%DTLS-3-HANDSHAKE_FAILURE: ...`

8.2.1 Verarbeiten von Cisco-Logs im Allgemeinen

Vor der eigentlichen Nachricht in einem Cisco-Syslog-Paket ist noch eine Art Header vorzufinden. Dieser weist Ähnlichkeiten mit dem Header des Standards auf, vertauscht allerdings einige Felder oder lässt manche sogar komplett weg. Um den Verarbeitungsprozess besser zu veranschaulichen, folgt ein simpler Prozessablauf des Cisco-Moduls des *Indexers*.

Basierend auf den gesammelten Logdaten der Cisco-Geräte kann gesagt werden, dass eine Cisco-Syslog-Nachricht im Allgemeinen wie folgt aufgebaut ist:

```
"<" PRIO ">" [ HOSTNAME ": *" PROCESS ": " ] TIMESTAMP [ " " HOSTNAME
" " ] ": %" PLATTFORM "-" SEVERITY "-" MSGID ": " MSG}
```

1. Die Priorität wird ausgelesen und die Felder `facility` und `severity` befüllt.
2. Zwischen `"\%"` und `"-"` werden Nachrichten-Plattform und -Typ ausgelesen und das `type`-Feld befüllt.
3. Es wird geprüft, ob `"*"` vor dem `"\%"` vorkommt:
   (a) Falls ja, wird der Hostname und der Prozess ausgelesen und die Felder `hostname` und `provider` befüllt.
   (b) Falls nein, wird der Hostname ausgelesen und das dementsprechende Feld befüllt.
4. Der Zeitstempel wird ausgelesen und das dementsprechende Feld befüllt.

Das letzte Feld (`MSG`) wird durch den *Indexer* nur verarbeitet, wenn die Nachricht von einer Cisco ASA kommt – Genaueres wird im nächsten Kapitel erläutert (siehe Kapitel 8.2.2). Falls die Plattform eine andere ist, wird die genauere Nachricht nicht weiter bearbeitet und nur als Text in die DB abgelegt.

8.2.2 Verarbeiten von Cisco-ASA-Logs

8.2.2.1 Problem

Die Cisco ASA implementiert einen möglichst human-readable (für den Menschen verständlichen) Ansatz des Syslog Formates, was als größter Vorteil dieses Ansatzes gesehen werden kann. Aufgrund der überwältigenden Menge der Syslog Daten bei einem IPS ist die Interpretation mittels Maschine unumgänglich. Für Maschinen stellt jedoch ein human-readable Text eine große Herausforderung dar, weil die Informationsdichte des Textes sehr gering ist.

8.2.2.2 Informationsdichte

Unter Informationsdichte versteht man den Informationsgrad eines Textes in Bezug auf Redundanzen sowie die Verwendung von für die Informationsvermittlung unwichtiger Worte.

Beispiel unterschiedlicher Informationsdichten im Vergleich zwischen ASA und Fortigate:

Geringe Informationsdichte anhand einer ASA Log Message:

```
Teardown TCP connection 893 for outside:10.2.61.60/40056 to
inside:192.168.3.2/443 duration 0:00:11 bytes 1233 TCP Reset-O
```

Hohe Informationsdichte anhand einer Fortigate Log Message:

```
devname=Fortigate-DMZ-Firewall devid=FGT60D4Q16016151
logid="0000000015" type="traffic" subtype="forward" srcip=192.168.3.3
srcport=63313 srcintf="internal1" dstip=8.8.8.8 dstport=53
dstintf="wan1" sessionid=1204304 proto=17 action="start" service="DNS"
duration=0 sentbyte=0 rcvdbyte=0 sentpkt=0
```

Während Menschen Sätze mit geringer Informationsdichte verständlicher finden, stellt diese für Maschinen, aufgrund der Notwendigkeit, unwichtige Informationen herauszufiltern, eine große Herausforderung dar. Dies begründet sich vor allem darin, dass die Fähigkeit zwischen wichtiger und unwichtiger Information zu unterscheiden der Maschine erst implementiert werden muss.

8.2.2.3 Lösungsansatz

Man kann das am besten anhand von Regular-Expressions implementieren. Diese sind jedoch umständlich zu formulieren und bei komplexeren Ausdrücken werden diese schnell unübersichtlich und unlesbar. Hier brachte Lark Abhilfe, weil Lark eine Python-Library für komplexere Regular-Expressions ist. [5]

8.2.2.4 Lark

Lark ist eine moderne Grammatikübersetzungsbibliothek für Python, die eine Erweiterung zu den weit verbreiteten Regular Expressions darstellt. Folgendes Beispiel zeigt, warum Lark für die Anwendung der Indizierung von ASA Log Daten verwendet wird und welche Vorteile Lark bringt.

Das „Hello World“ von Lark:

```
from lark import Lark
l = Lark('''start: WORD "," WORD "!"

            %import common.WORD   // Importiert den Variablentypen WORD
            %ignore " "           // Ignoriert Leerzeichen
         ''')
print( l.parse("Hello, World!") )
```

Ergebnis:

```
Tree(start, [Token(WORD, 'Hello'), Token(WORD, 'World')])
```

In diesem Beispiel wird aus dem String `Hello, World!` der Syntaxbaum `(WORD, "Hello"), (WORD, "World")` gebildet, wobei WORD die Art der Variable darstellt und Hello bzw. World den Inhalt der Variable. WORD ist eine vordefinierte Variablenart, die sich aus einer beliebigen Menge an Klein- und Großbuchstaben zusammensetzt. Der Aufbau von Variablen wird in einem späteren Kapitel behandelt.

Der Ausdruck `"start"` benennt die Regel, die in diesem Fall `start: WORD "," WORD "!"` implementiert. Ausdrücke, die nicht als Variable definiert werden, werden während des Verarbeitens weggefiltert, wie in diesem Beispiel der Beistrich und das Rufzeichen.

Abschließend ist die Ableitbarkeit der Variablentypen von großer Bedeutung, da somit erstens komplexe Regular Expressions klein und übersichtlich aussehen und zweitens die Variablentypen auch gleichzeitig deren Benennung darstellen, obwohl es sich um denselben Grundtyp handelt.

Wie bereits angesprochen, setzt sich Word aus einer beliebigen Anzahl an Klein- und Großbuchstaben zusammen. Hierzu kann eine Regular Expression oder eine Variablendefinition in Lark erstellt werden.

Regular Expression Variablendefinition: `pattern = re.compile('([a-zA-Z]+)')`

Lark Variablendefinition:

```
LCASE_LETTER: "a".."z"
UCASE_LETTER: "A".."Z"

WORD: LETTER+

GREETING: WORD
ENTITY: WORD
```

In diesem Beispiel wird `Hello, World!` sowohl mit Regular Expressions als auch mit Lark interpretiert.

```
from lark import Lark
import re
string = "Hello, World!"
parser = Lark('''start: GREETING "," ENTITY "!"
            %import common.WORD
            GREETING: WORD
```

```
            ENTITY: WORD
            %ignore " "
            ''', parser='earley')
tree = parser.parse(string)
print(tree)
print(tree.children[0], tree.children[1])
string = "Hello, World!"
pattern = re.compile('([a-zA-Z]+), ([a-zA-Z]+)!')
match = re.match(pattern, string)
print(match)
print(match.group(1), match.group(2))
```

Auch wenn sich die zwei Outputs bezüglich vieler Faktoren hier sehr ähnlich sind, kann schon die übersichtlichere Struktur von Lark erkannt werden:

```
Tree(start, [Token(GREETING, 'Hello'), Token(ENTITY, 'World')])
Hello World
<_sre.SRE_Match object; span=(0, 13), match='Hello, World!'>
Hello World
```

Im folgenden Beispiel liegt ein wichtiges Augenmerk auf den Variablennamen, die der Indizierung dienen, die später für die verschiedenen Felder der ASA Logdaten wichtig werden.

```
print(tree.children[0].type, tree.children[1].type)
GREETING ENTITY
```

Werden optionale Felder einer Nachricht hinzugefügt, kommt der wahre Vorteil von Lark zum Vorschein, weil diese Felder mit Namen immer noch vollständig indiziert werden können. Ein weiterer Vorteil ist, dass Lark mit der `%ignore` Funktion automatisch auf Leerzeichen eingehen kann, die der Regular Expression auch Probleme bereiten kann. Um dies beispielhaft darzustellen, fehlt das Leerzeichen zwischen Beistrich und `beautiful`.

```
from lark import Lark
import re
string = "Hello,beautiful World!"
parser = Lark('''start: GREETING "," ADJECTIVE? ENTITY "!"
            %import common.WORD
            GREETING: WORD
            ENTITY: WORD
            ADJECTIVE: WORD
            %ignore " "
```

```
            ''', parser='earley')
tree = parser.parse(string)
print(tree)
for token in tree.children:
    print(f'{token.type}: {token.value}')
pattern = re.compile('([a-zA-Z]+),([a-zA-Z]*) ([a-zA-Z]+)!')
match = re.match(pattern, string)
print(match)
for group in match.groups():
    print(group)
```

Output ohne `beautiful`:

```
Tree(start, [Token(GREETING, 'Hello'), Token(ENTITY, 'World')])
GREETING: Hello
ENTITY: World
<_sre.SRE_Match object; span=(0, 13), match='Hello, World!'>
Hello

World
```

Output mit `beautiful`:

```
Tree(start, [Token(GREETING, 'Hello'), Token(ADJECTIVE, 'beautiful'),
    Token(ENTITY, 'World')])
GREETING: Hello
ADJECTIVE: beautiful
ENTITY: World
<_sre.SRE_Match object; span=(0, 22), match='Hello,beautiful World!'>
Hello
beautiful
World
```

Aufgrund der oben beschriebenen Vorteile von Lark wird im Zuge der Erläuterung von ASA-Syslog Nachrichten auf Lark eingegangen.

8.2.2.5 Verarbeitung der ASA-Syslog-Nachrichten mittels Lark

Um genauer zu verstehen, wie Lark bei der Analyse von ASA Syslog Nachrichten hilft, wird ein Beispiel anhand der ASA Syslog Nachricht 302010 gegeben.

Auszug aus der offiziellen ASA Dokumentation von Cisco. [4]

ASA-6-302010

Error Message

%ASA-6-302010: *connections* in use, *connections* most used

Explanation

Provides information on the number of connections that are in use and most used.

connections

The number of connections.

Recommended Action

None required.

Die Syslog Nachricht, die das System aufnimmt, sieht wie folgt aus. Eine Erklärung des Syslog Formats findet sich unter siehe Kapitel 5.2.1 auf Seite 16.

```
<166>Sep 12 2019 03:42:06 ASA-Edge-Firewall : %ASA-6-302010: 34 in
use, 896 most used
```

Bei Lark kommt folgende Zeile an:

```
{"msg":"34 in use, 896 most used", "type":"cisco:asa/302010"}
```

Definition der Variablen für Lark:

```
DIGIT: "0".."9"

INT: DIGIT+

CONNECTIONS_USE: INT
CONNECTIONS_MAX: INT
```

Die Beschreibung der Lark Regel sieht wie folgt aus:

```
302010 start: CONNECTIONS_USE "in use," CONNECTIONS_MAX "most used"
```

Daraus kann das Script die Felder der Syslog Nachricht exportieren und den restlichen human-readable Text verwerfen:

```
cisco:asa/302010 ({'msg': '34 in use, 896 most used', 'type':
'cisco:asa/302010', 'data': {'connections_use': '34',
'connections_max': '896'}}, True)
```

Der Interpreter kann somit die Variablen aus dem Fließtext herausfiltern und diese in die Datenbank speichern, siehe Kapitel 7 auf Seite 43.

Es gibt auch weitaus schwierigere Syslog Nachrichten, wie etwa die ASA Syslog Nachricht 302014. Dazu ein Auszug aus der offiziellen Referenz:

ASA-6-302014

Error Message

`%ASA-6-302014: connections Teardown TCP connection id for interface:real-address/real-port [(idfw_user)] to interface:real-address/real-port [(idfw_user)] duration hh:mm:ss bytes bytes [reason [from teardown-initiator]] [(user)]`

Explanation

A TCP connection between two hosts was deleted. The following list describes the message values:

id
: A unique identifier

interface, real-address, real-port
: The actual socket

duration
: The lifetime of the connection

bytes
: The data transfer of the connection

User
: The AAA name of the user

idfw_user
: The name of the identitiy firewall user

reason
: The action that causes the connection to terminate.

teardown-initiator
: Interface name of the side that initiated the teardown.

Recommended Action

None required.

Die Implementation dieser Nachricht ist eine Herausforderung, weil die Nachricht viele optionale Felder enthält. Dennoch spielt die Nachricht 302014 für das Herausfinden der Anzahl der Bytes, die mit TCP über das Netzwerk übertragen werden, eine wesentliche Rolle.

Nachdem bereits in einem vorherigen Kapitel geklärt wurde, dass der Header des Syslog Formates bereits abgeschnitten wurde, wird im Weiteren der für die Diplomarbeit relevante Teil dargestellt:

```
Teardown TCP connection 893 for outside:10.2.61.60/40056 to
inside:192.168.3.2/443 duration 0:00:11 bytes 1233 TCP Reset-O
```

Aus der Referenz und der vorgegebenen Syslog Nachricht kann eine Lark Regel erstellt werden, wobei im Sinne der Übersichtlichkeit auf nicht relevante, optionale Felder verzichtet wird.

```
Teardown TCP connection" ID "for" INTERFACE_NAME_IN ":" IPP_IN "to"
INTERFACE_NAME_OUT ":" IPP_OUT "duration" TIME "bytes" BYTES REASON?
("from" INITIATOR)? USER?
```

Dazu müssen noch Variablen definiert werden. Die Vererbung wird in der Abbildung 8.1 auf der nächsten Seite veranschaulicht. In der Abbildung stehen Ovale für Regular Expressions, Rechtecke für vererbte Objekte, die mit Regular Expressions erweitert worden sind und abgerundete Rechtecke für deren Benennung. Unter Benennug versteht man den Namen, mit dem auf sie zugegriffen werden kann.

```
DIGIT: "0".."9"
LCASE_LETTER: "a".."z"
UCASE_LETTER: "A".."Z"

INT: DIGIT+
LETTER: UCASE_LETTER | LCASE_LETTER
CNAME: ("_"|LETTER) ("_"|LETTER|DIGIT)*

IP: INT "." INT "." INT "." INT
IPP: (IP "/" INT | IP)

IPP_IN: IPP
IPP_OUT: IPP
INTERFACE_NAME_IN: CNAME
INTERFACE_NAME_OUT: CNAME
TIME: DIGIT~1..2 ":" DIGIT~1..2 ":" DIGIT~1..2
BYTES: INT
REASON: ("Connection timeout"|"Deny Terminate"|
    "Failover primary closed"|"FIN Timeout"|
```

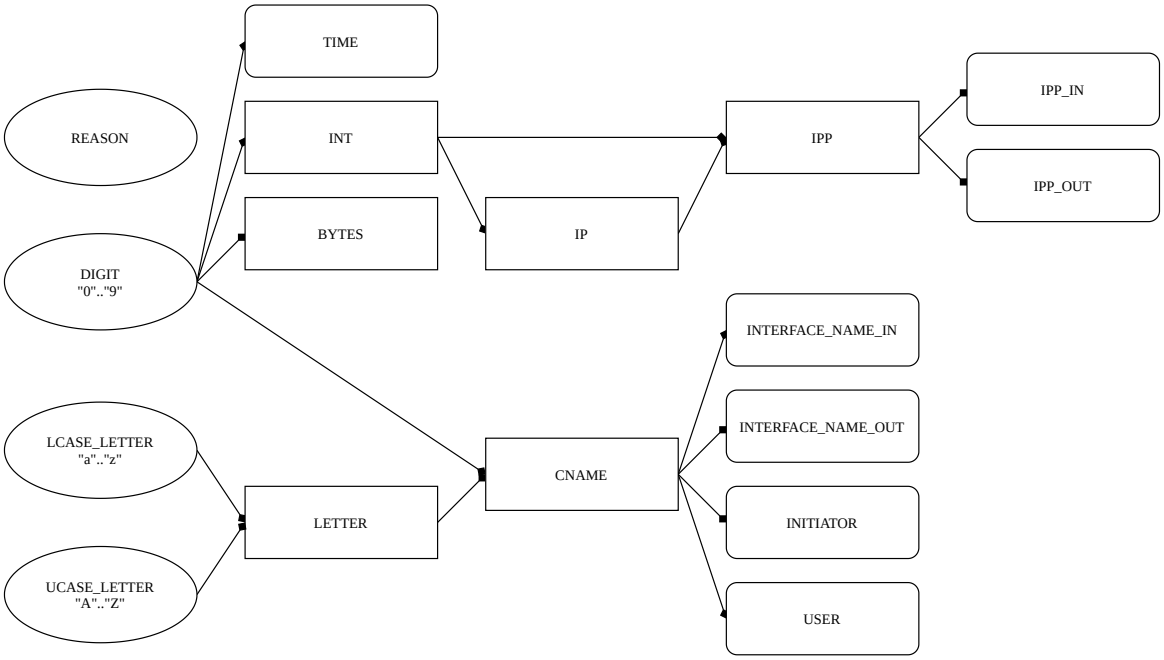

Abbildung 8.1: Die Variablenvererbung in Lark

```
    "Flow closed by inspection"|"Flow terminated by IPS"|
    "Flow reset by IPS"|"Flow terminated by TCP Intercept"|
    "Flow timed out"|"Flow timed out with reset"|"Flow is a loopback"|
    "Free the flow created as result of packet injection"|
    "Invalid SYN"|"IPS fail-close"|
    "No interfaces associated with zone"|"No valid adjacency"|
    "Pinhole Timeout"|"Route change"|"SYN Control"|"SYN Timeout"|
    "TCP bad retransmission"|"TCP FINs"|"TCP Invalid SYN"|
    "TCP Reset-APPLIANCE"|"TCP Reset-I"|"TCP Reset-O"|
    "TCP segment partial overlap"|
    "TCP unexpected window size variation"|
    "Tunnel has been torn down"|"Unauth Deny"|
    "Unknows"|"Xlate Clear")
INITIATOR: CNAME
USER: CNAME
```

Die korrekte Funktion der Regel ist abhängig vom zuvor ausgewählten Parser, weil Earley und Look-ahead LR Parser (LALR)(1) diese Regel unterschiedlich interpretieren.

Vergleich Earley und LALR(1) LALR(1) erkennt in diesem und in anderen Beispielen die Felder falsch. Earley hingegen erkennt das `TCP Reset-O` richtig als ein Reason und kann damit gegebenenfalls auf, in diesem Beispiel nicht vorhandene, `INITIATOR` und `USER` Einträge eingehen. Dieses Fehlverhalten von LALR(1) lässt sich aufgrund

seiner Architektur begründen, da LALR(1) mit dieser 1 beschreibt, dass es einen einzigen vorausschauenden Schritt macht. Im Gegensatz dazu bildet Earley einen Syntaxbaum, um alle Zustände durchzuprobieren und damit auch alle abzudecken. Dieser Syntaxbaum bringt den Vorteil, dass Earley die erwarteten Ergebnisse liefert, dafür aber Performance einbußt, während LALR(1) zwar schneller ist, jedoch mit dieser komplexen Grammatik einen Fehler liefert und abbricht.

Feld	Earley	LALR(1)
ID	893	893
INTERFACE_NAME_IN	outside	outside
IPP_IN	10.2.61.60/40056	10.2.61.60/40056
INTERFACE_NAME_OUT	inside	inside
IPP_OUT	192.168.3.2/443	192.168.3.2/443
TIME	0:00:11	0:00:11
BYTES	1233	1233
REASON	TCP Reset-O	
INITIATOR		
USER		TCP

8.2.2.6 Ergebnis der ASA-Logdatenverarbeitung

Mit Lark kann theoretisch jede ASA Syslog Nachricht interpretiert und indexiert werden. Es wurden von der ASA insgesamt 6.212.109 Millionen Syslog Nachrichten aufgezeichnet und davon 6.203.505 indiziert, was 99,8 Prozent der gespeicherten ASA Log Nachrichten entspricht. Damit wurde auch das anfänglich gesetzte Ziel, von einer mindestens 95-prozentigen Indizierung aller anfallenden ASA Syslog Nachrichten erreicht, weil lediglich 8604 Nachrichten nicht indizierbar waren. Um weitere Nachrichten zu implementieren, bedarf es zusätzliche Regeln zu implementieren und gegebenenfalls auch neue Variablen zu definieren.

8.2.2.7 Fazit

Zusammenfassend kann man sagen, dass nur wenige Syslog Nachrichtentypen für diese Diplomarbeit von Bedeutung sind, denn es waren primär die Summe der Bytes und Hosts, die außerhalb des Netzwerkes kontaktiert wurden, von Interesse.

Auch wenn man bei einem Indizierungsgrad von 99,8 Prozent davon ausgehen kann, dass die meisten relevanten Informationen erfasst werden können, ist es doch möglich, dass in den 0,2 Prozent der nicht indizierten Daten wichtige Informationen stecken. Diese Informationen werden zwar nicht ausgewertet, bleiben jedoch in der Datenbank gespeichert.

8.2.3 Verarbeiten von Fortinet-Logs

Das Format, in dem die FortiGate Firewall ihre Syslog-Nachrichten sendet, ist recht simpel: *Key-Value-Pairs* (Schlüssel-Wert-Paare). Zum automatisierten Verarbeiten gibt es fast kein Format, das geeigneter wäre, weil es sehr schnell und einfach verarbeitet werden kann.

```
<189>date=2020-01-01 time=12:34:56 devname=DMZ-Firewall
logid="0000000015" type="traffic" ...
```

Es gibt grundsätzlich zwei Möglichkeiten Schlüssel-Wert-Paare einzulesen: per RegEx, oder per Statemachine. In der Diplomarbeit wurde die Möglichkeit der Statemachine ausgewählt, da diese einen weniger rechenintensiven Overhead hat und tendenziell weniger Speicher zur Laufzeit verbraucht. Es gilt auch zu beachten, dass die Schlüssel-Wert-Paare entweder ohne weitere Elemente funktionieren (`KEY=VALUE`) oder die Werte von Anführungszeichen umgeben sind und dadurch auch Leerzeichen enthalten können (`KEY="VAL UE"`).

Spezielle Schlüssel werden aus dem `data` Feld genommen und im DB-Eintrag selbst weiterverwendet:

- `data.devname` wird zu `origin.hostname`
- `data.date` und `data.time` werden zu `timestamp` (falls beim *Indexer* nicht der Parameter `-i` angegeben wurde)
- `data.logid` wird als `type`-Feld verwendet (`type = fortinet:fortigate/LOGID`)

8.2.4 Verarbeiten von Solarwinds-Logs (Windows)

Da ein Windows Gerät von alleine aus keine Syslog-Nachrichten senden kann, muss der Syslog-Client *Solarwinds* verwendet werden (siehe Kapitel 5.5.3 auf Seite 30). Bei Syslog-Nachrichten von *Solarwinds* (einem Syslog-Client auf Windows) ist die Verarbeitung etwas komplexer als bei denen von Fortinet-Geräten. Vor der eigentlichen Nachricht fügt *Solarwinds* noch einen eigenen Header hinzu, welcher Zeitstempel und Hostname des abschickenden Gerätes beinhaltet. Danach folgen Tab-getrennte Felder, die das Windows-Event genauer beschreiben und zuletzt die Details des Events (hier: `DATA`). Wie an den `"?"` unterhalb zu sehen ist, konnte im Laufe der Recherchearbeiten der Diplomarbeit nicht allen Feldern ein eindeutiger Sinn oder Nutzen zugeordnet werden. Der Prefix `SW_` steht in diesem Fall für *Solarwinds*.

```
"<" PRIO ">" SW_TIMESTAMP " " SW_HOSTNAME " MSWinEventLog" tab PRIO
tab LOG_GROUP tab ? tab TIMESTAMP tab EVENT_ID tab PROVIDER tab ? tab
? KEYWORD tab HOSTNAME tab TASK tab MSG tab DATA
```

Wobei `DATA` wie folgt aufgebaut ist:

```
Group A:
    Key 1: Value 1
    Key 2: Value 2

Group B:
    Key 3: Value 3
    Key 4: Value 4
```

Ein Beispiel:

```
<5>Mär 01 17:06:51 Logging_Source.argos.local MSWinEventLog     5
    Security        2481391 So. Mär 01 17:06:46 2020        4624
    Microsoft-Windows-Security-Auditing             N/A
    Audit Success   Logging_Source.argos.local  12544
    An account was successfully logged on.

Subject:
        Security ID:            S-1-0-0
        Account Name:           -
        Account Domain:         -
        Logon ID:               0x0

Logon Information:
        Logon Type:             3
        Restricted Admin Mode:  -
        Virtual Account:                No
        Elevated Token:         Yes

Impersonation Level:            Impersonation

New Logon:
        Security ID:            S-1-5-18
        Account Name:           LOGGING_SOURCE$
        Account Domain:         ARGOS.LOCAL
        Logon ID:               0xE0B0D0B6
        Linked Logon ID:                0x0
        Network Account Name:   -
        Network Account Domain: -
        Logon GUID:             {915926E6-87FD-FFAA-8F94-E77FFEA12090}


...
```

Im Gegensatz zu Fortinet-Logs sind Solarwinds-Logs wesentlich komplexer, aber immer noch mit relativ einfachen Mitteln zu verarbeiten. Das allergrößte Problem, welches das Solarwinds-Log-Format hat, ist, dass das Format des Zeitstempels in der Windows-Systemsprache ist. Wenn – wie in der Diplomarbeit – die Systemsprache auf Deutsch gestellt ist, dann werden Monate bzw. Wochentage statt mit ihren englischen Namen mit ihren deutschen Namen abgekürzt. Der *Indexer* ist daher nur in der Lage, das deutsche Datumsformat weiterzuverarbeiten, da keine Systeme mit anderen Sprachen zur Verfügung standen, um Beispieldaten zu generieren.

1. Die Priorität wird ausgelesen, und die Felder `facility` und `severity` befüllt.
2. Der Solarwinds-Zeitstempel wird übersprungen, da er nur den Zeitpunkt des Absendens dokumentiert.
3. Der Solarwinds-Hostname wird ebenfalls übersprungen.
4. Es wird überprüft, ob `MSWinEventLog` folgt. Wenn das Schlüsselwort nicht vorkommt, wird die Verarbeitung abgebrochen.
5. Danach werden die einzelnen Felder eingelesen – immer bis zum nächsten Tab.
6. Das `DATA`-Feld wird Zeile für Zeile eingelesen:
   (a) Zu Beginn der Zeile sind keine Leerzeichen: Es wird der Gruppenname gespeichert, um die folgenden Schlüssel-Wert-Paare der richtigen Gruppe zuordnen zu können.
   (b) Zu Beginn der Zeile sind Leerzeichen: Es wird nach dem ersten Doppelpunkt gesucht, und der Schlüssel und dazugehörige Wert bestimmt.

8.3 Verarbeiten von Cisco-WLC-Nachrichten in der Diplomarbeit

Neben Syslog-Nachrichten können auch SNMP-Trap-Nachrichten (siehe Kapitel 5.2.2 auf Seite 18) zum Übermitteln von Event-Informationen verwendet werden. Der genauere Aufbau der SNMP-Trap-Nachrichten wurde bereits in anderen Kapiteln abgehandelt (siehe Kapitel 5.2.2 auf Seite 18) und wird deshalb hier nicht weiter ausgeführt. Im Gegensatz zu Syslog-Nachrichten sind SNMP-Trap-Nachrichten binäre Nachrichten und nicht in Textform. Aus diesem Grund sind die Nachrichten auch tendenziell einfacher verarbeitbar. In der Diplomarbeit wurden nur SNMP-Trap-Nachrichten eines Cisco-WLC benötigt, da dieser essentielle Informationen ausschließlich über SNMP bereitstellen kann (siehe Kapitel 5.3.2 auf Seite 24).

Es folgt eine Auflistung der in der Diplomarbeit verwendeten Object Identifier (OIDs). Diese OIDs werden vom *Indexer* erkannt und verarbeitet, andere müssten erst implementiert werden. Der OID `1.3.6.1.4.1.9` steht für die Firma Cisco [7], `1.3.6.1.4.1.9.9` steht für `ciscoMgmt`.

Zeit (`sysUpTime`: OID 1.3.6.1.2.1.1.3[.0]))
Die Zeit in Hunderdstelsekunden seit dem der SNMP-Dienst des Systems gestartet bzw. neu initialisiert wurde. Dieser OID wird vom *Indexer* nicht verarbeitet.

Aktion (`snmpTrapOID`: OID 1.3.6.1.6.3.1.1.4.1[.0])
Die Aktion bzw. das Event, welche/s das Senden des SNMP-Traps verursacht hat.

AP Hostname (`cLApName`: OID 1.3.6.1.4.1.9.9.513.1.1.1.1.5[.0])
Der Hostname des Access Point, an welchem sich der Benutzer angemeldet hat.

AP MAC-Adresse (`cldcApMacAddress`: OID 1.3.6.1.4.1.9.9.599.1.3.1.1.8[.0])
Die MAC-Adresse des Access Point, an welchem sich der Benutzer angemeldet hat.

Interface Nummer (`cLApDot11IfSlotId`: OID 1.3.6.1.4.1.9.9.513.1.2.1.1.1[.0])
Die Nummer des Interfaces, an dem der Access Point die Authentifizierungsanfrage erhalten hat. Dieser OID wird vom *Indexer* nicht verarbeitet.

Client MAC-Adresse (`cldcClientMacAddress`: OID 1.3.6.1.4.1.9.9.599.1.3.1.1.1[.0])
Die MAC-Adresse des Client, welcher versucht hat, sich anzumelden.

Client IP-Adresse (`cldcClientIPAddress`: OID 1.3.6.1.4.1.9.9.599.1.3.1.1.10[.0])
Die IP-Adresse des Client, welcher versucht hat, sich anzumelden.

Benutzername (`cldcClientUsername`: OID 1.3.6.1.4.1.9.9.599.1.2.1[.0])
Der Benutzername des Benutzers, welcher versucht hat, sich anzumelden.

SSID (`cldcClientSsid`: OID 1.3.6.1.4.1.9.9.599.1.2.2[.0])
Die SSID des WLANs, in welchem die Authentifizierungsanfrage gestellt wurde.

8.4 Sniffer

8.4.1 Einleitung zum Sniffer

Der Sniffer ist einer der sechs modularen Komponenten und für die Informationserbringung durch DPI zuständig. Dafür akzeptiert das Programm alle PDUs, die den Server erreichen, inspiziert aber nur jene, die DNS-Informationen enthalten, bis in die Tiefe von OSI-Schicht sieben. Das Programm extrahiert dann alle Informationen aus der PDU und speichert diese übersichtlich und gut-maschinenlesbar in der zentralen Datenbank ab. Somit können durch den Sniffer statistische Aussagen über den DNS Verkehr getroffen und Angriffe erkannt werden. Für Informationen zur genaueren Defi-

nition von DPI und welche Informationen dadurch extrahierbar sind, siehe Kapitel 6 auf Seite 33. Die gewonnenen Informationen werden in drei verschiedenen Sammlungen der Datenbank abgelegt. Jede dieser Sammlungen beinhaltet jeweils individuell gewonnenen Informationen, die somit schnell und ohne umständlichen Datenbankabfragen visualisiert werden können.

Wie schon in vorherigen Kapiteln angesprochen (siehe Kapitel 6.6 auf Seite 37), wird durch einen Switch beinahe der gesamte Datenverkehr an den Sniffer weitergeleitet. Dieser befasst sich dann tiefer mit dem Inhalt der empfangenen Daten. Dazu entpackt das Programm jeden Kopfbereich des PDUs einzeln und speichert die Informationen intern ab. Bei diesem Prozess werden erstmals alle Informationen zwischengespeichert, damit sie für die spätere Endspeicherung zur Verfügung stehen. Allerdings werden bei der Endspeicherung nur relevante Informationen gesichert, um einerseits Speicherplatz zu minimieren und andererseits, um die menschliche Lesbarkeit zu wahren. Durch diese Funktionalität können zum Beispiel die IP-Adresse oder allfällige Portnummern zu der Analyse der Nutzdaten hinzugezogen werden. Außerdem sind die Informationen relevant, um überhaupt entscheiden zu können, ob die nächste OSI-Schicht noch inspiziert oder ob das komplette PDU nicht verworfen werden soll. Bei der obersten Schicht des OSI-Modells angekommen, wird nun der DNS Kopf-, als auch der Datenbereich inspiziert.

8.4.2 Der Entpackungsprozess des Sniffers

Die Daten kommen als verpackter Bytestrom beim Sniffer an. Dieser Bytestrom stellt zwar die verschiedenen OSI-Schichten dar, allerdings sind diese seriell als Bits aneinander gereiht. Mittels der Python-Bibliothek `struct`, die eigens für die Verarbeitung von verpackten Byteobjekten erstellt wurde, entpackt der Sniffer den Bytestrom und unterteilt diesen nun in die einzelnen OSI-Schichten.

Der Syntax für das Entpacken von Byteobjekten mithilfe dieser Bibliothek ist nach näherer Betrachtung einfach zu verstehen:

- Es wird eine Variable deklariert – genauer gesagt ein Array–, welche die extrahierten Werte speichert. Diese Variable wird mit der Funktion `struct.unpack(FORMAT, Bytes)` erstellt.
- Das Format setzt sich aus einem Zeichen für die Bytereihenfolge (LSB oder MSB) und Abgrenzungsindikatoren zusammen.
- Die Abgrenzungsindikatoren bestehen wiederum aus einem Buchstaben, welcher für einen Python Datentyp steht. Bei manchen Indikatoren (s, p und P) kann auch eine Zahl vorangestellt sein, die für die Anzahl der Bytes für dieses Element steht. Allerdings gibt es bei den meisten Indikatoren eine individuelle Standardlänge.
- Wichtig ist noch, dass die Länge des Bytestroms gleich groß wie die vom Format spezifizierte sein muss, da sonst die Methode eine Fehlermeldung ausgibt.

Im nachfolgenden Beispiel wird der Entpackungsprozess veranschaulicht. Dabei wird der Bytestrom in der ersten Zeile unter Verwendung der `struct.unpack()` Methode und durch Anwendung des Formates in die einzelnen Felder entpackt. Das Format beschreibt drei – unterschiedlich große – Felder. Die ersten beiden Felder besitzen den Python Datentyp String mit einer Größe von je sechs Byte und das letzte Feld den Datentyp `int` mit einer Größe von zwei Byte. Für die spätere Verarbeitung werden diese Felder dann in dem Array `fields` und in den einzelnen Variablen gespeichert.

```
import struct
fields = struct.unpack("!6s6sH", bytestream[0:14])
_dest_mac = fields[0] # Von Byte 0 bis 5
_src_mac = fields[1] # Von Byte 6 bis 11
_type = fields[2] # Von Byte 12 bis 13
```

Diese Technik wird nicht nur auf der zweiten OSI-Schicht (Sicherungsschicht oder *Datalink* genannt), sondern bis hinauf zur höchsten OSI-Schicht angewandt. Ermöglicht wird dies durch die Vereinheitlichung der einzelnen Protokolle durch das OSI-Modell und durch die Standardisierung der einzelnen PDU-Kopfbereiche durch die jeweiligen RFCs. Erst dadurch können Information aus Schicht 3 und 4, sowie der Schicht 7 extrahiert werden.

8.4.3 Die Analyse eines DNS-Pakets

DNS überträgt Daten immer komprimiert – diese Funktion wird *Message Compression* genannt und dient der Datenminimierung. Durch *Message Compression* wird zwar Bandbreite während der Übertragung gespart, allerdings verkomplizierte dies den Ausleseprozess der Daten. Das liegt daran, da durch die Komprimierung ein FQDN-Segment oder aber auch ein kompletter FQDN nicht zweimal in einer PDU vorkommen kann, sondern es wird bei dem zweiten Vorkommen lediglich auf die Stelle des Ersten referenziert. In beispielsweise jeder DNS Antwort wird nochmals der Anfrage FQDN abgebildet, um für diesen eine individuelle Antwort bereitzustellen. Außerdem kann eine DNS Antwort PDU mehrere Antworten für denselben FQDN beinhalten. Die einzelnen Antworten beinhalten nicht nur den Anfrage FQDN und die Antwort, sondern auch zusätzliche Informationen wie:

- den Antworttypen (ist ein Indikator für das Format der Antwort),
- die Klasse (gibt an, wo die Antwort gültig ist),
- die *Time To Live* (gibt an, wie lange die Antwort zwischengespeichert werden soll) und
- die Antwortlänge (stellt die Länge der Antwort dar)

In Abbildung 8.2 auf der nächsten Seite wird *Message Compression* vereinfacht veranschaulicht. Dabei stellt der Buchstabe B ein beliebiges Byte einer DNS PDU dar. Ein

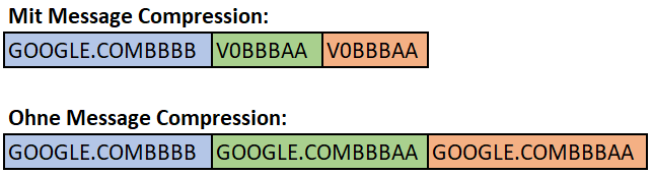

Abbildung 8.2: Visualisierung der Datenreduktion durch Message Compression

Verweis wird durch eine spezielle Bitfolge (V) initiiert, und die Zahl dahinter stellt den Index der Verweisquelle dar – der Anfang von `GOOGLE.COM`. Die tatsächliche Antwort (A) folgt nach einer Bytefolge an zusätzlichen Informationen auf den Anfrage FQDN und die einzelnen Antworten und die Anfrage sind voneinander abgetrennt und farblich hervorgehoben dargestellt. Man erkennt schnell, das *Message Compression* die Größe der Nachricht drastisch – im Fall des Beispiels auf etwa zwei Drittel – verkleinert.

8.4.4 Verarbeitung von Echtdaten einer Schulklasse

Um das Benutzerverhalten von DNS in echt und nicht in einer abgeschotteten Testumgebung zu analysieren, bedarf es Daten, die von echten Nutzern generiert werden. Dazu wurde das Komplettsystem mit vereinfachten und weniger Funktionen implementiert, welche die Daten einer Schülerklasse der HTL Rennweg analysieren. Diese Version beinhaltet nur die Funktionalität des Sniffers. Dabei werden nun keine personenbezogenen Daten gespeichert, und somit ist diese Anwendung datenschutzkonform. Der Netzaufbau dieser Infrastruktur ähnelt dem der Testumgebung sehr stark, da auch hier mittels eines Port-Mirrors der Paketfluss auf den Sniffer umgeleitet wird. Für genauere Information, wie DPI in der Diplomarbeit angewandt wird, siehe Kapitel 6.6 auf Seite 37.

Der Sniffer speichert nun – genau wie in der Testumgebung – alle DNS Anfragen und Antworten in einer lokalen Datenbank mit derselben Struktur wie im Komplettsystem ab. Dabei wird das Identifikatorformat allerdings verändert, um beim Vereinen der beiden Datenstrukturen Überlappungsfehler vorzubeugen. Dadurch kann diese lokale Datenbank einfach exportiert und im Komplettsystem importiert werden. Nach diesem Vorgang sind die Daten wie die aus der Testumgebung verfügbar und können verarbeitet oder visualisiert werden.

8.5 Enricher

8.5.1 Einleitung zum Enricher

Der Enricher ist einer der sechs modularen Komponenten und für das Hinzufügen von extern gewinnbaren Informationen zuständig. Extern sind die Informationen deswegen, weil sie nicht direkt aus den Lognachrichten ausgelesen werden können, sondern durch die Konsultierung anderer Informationsquellen gewonnen werden. Dafür wurden verschiedene Möglichkeiten implementiert, um die internen Daten jeweils mit den individuell externen Informationen zu vervollständigen. Dieser Vorgang der Informationsgewinnung wird *enrichen* oder zu Deutsch *anreichern* der Daten genannt.

Weiters war es ein Teil der Anforderungen an den Enricher, mögliche Angriffe auf das Netzwerk automatisiert zu erkennen. Dafür wurden die Angriffe manuell durchgeführt, die anfallenden Daten analysiert und eine automatische Erkennung für vier davon implementiert. Insgesamt wurden acht Angriffe analysiert. Im Rahmen der Diplomarbeit war es allerdings nur möglich, vier dieser zu erkennen. Für die Gründe dafür siehe Kapitel 8.5.11 auf Seite 84. Als Standardreaktion für das Erkennen eines Angriffes wird ein Vorfall im Dashboard angezeigt (siehe Kapitel 9.6 auf Seite 94), und der Administrator kann darauf reagieren.

Im Rahmen des Enrichers werden vorerst vier verschiedene externe Informationsquellen konsultiert. Sollten die Daten noch zusätzlich durch andere Informationsquellen angereichert werden müssen, so greift der modular entwickelte Aufbau des Programms, und die gewünschte Funktion kann unkompliziert implementiert und eingebunden werden.

8.5.2 WebUntis Schnittstelle

In der HTL Rennweg wird WebUntis für die Verwaltung der Stundenpläne verwendet. Dabei besitzt jeder Schüler und jede Schülerin einen individuellen Stundenplan, respektive jede Klasse einen Klassenaccount und jede Lehrkraft einen Lehrkraftaccount. Als Teil der Schule besteht nun die Möglichkeit mit einem Application Programming Interface (API) und einem speziellen Universalaccount die aktuelle Unterrichtseinheit eines jeden Schülers und jeder Schülerin herauszufinden. Bei der Anmeldung im WLAN, welche im eigens erstellten Netz mit dem echten Namen durchgeführt wird, kann nun eine Abfrage im Hintergrund durchgeführt werden, die ermittelt, welche Unterrichtsstunde besagter Schüler oder besagte Schülerin aktuell besuchen sollte, bzw. in welchem Raum diese stattfindet. Weiterführend könnte evaluiert werden, an welcher Stelle im Schulhaus sich der Schüler/die Schülerin versucht einzuwählen. Durch die Überlagerung dieser beider Informationen könnte nun überprüft werden, ob sich der Schüler/die Schülerin tatsächlich an der richtigen Stelle aufhält und vermutlich den

Unterricht besucht oder nicht. Allerdings bedarf es dafür einen digitalen Grundriss der Schule und dies ist nicht Teil dieser Diplomarbeit.

8.5.3 Verarbeitung von MAC-Adressen

Bei einigen Logs wird die MAC-Adresse des Gerätes mitprotokolliert. Diese Adresse wird von den verschiedenen Herstellern in unterschiedlichen Formaten angegeben. Für eine einheitliche Weiterverarbeitung muss die MAC-Adresse zunächst normalisiert werden. Nach diesem Vorgang kann die Adresse durch eine Abfrage gegenüber einer Liste eindeutig einem Hersteller zugordnet werden. Das wird durch den zweiteiligen Aufbau der MAC-Adresse ermöglicht. Besagte Adresse besteht aus zwei 24 Bit langen Hälften. Dabei identifiziert die erste Hälfte – der Organizationally Unique Identifier (OUI) – den Hersteller und die zweite Hälfte das individuelle Gerät des Herstellers. Durch im Internet verfügbare Listen kann nun ein Abgleich des Herstellers erstellt und somit eine genauere physische Identifizierung des Gerätes durchgeführt werden. Dadurch schränkt sich bei der Suche nach einem bestimmten Gerät die mögliche Anzahl an Geräten ein.

8.5.4 Auflösen von IP-Adressen zu Domain-Namen

Für Menschen sind Zahlen und somit auch IP-Adressen nur schwer merkbar. Viel schneller prägt sich der Mensch allerdings einen FQDN oder anders gesagt Domainnamen ein. Deswegen kann der Enricher eine Abfrage durchführen, um eine IP-Adresse zu einem FQDN aufzulösen – er führt einen sogenannten *Reverse Lookup* durch. Als Beispiel kann dem Enricher eine IP-Adresse wie *8.8.8.8* als Parameter übergeben werden und dieser retourniert den zugehörigen FQDN zur aktuellen Zeit dazu (dns.google).

8.5.5 Geolokalisieren von IP-Adressen

Das Geolokalisieren von IP-Adressen funktioniert dank öffentlich zugänglichen Datenbanken, welche die grobe geografische Verteilung von ebensolchen Adressen abbilden. Dabei ist zu beachten, dass diese Informationen nicht die exakte Position bis auf die Hausnummer genau der IP-Adresse verraten, diese allerdings auf einen verhältnismäßig kleinen Bereich einschränken. Die mögliche Abweichung hängt von dem jeweiligen Land und der dortig dokumentierten Infrastruktur ab. Allerdings kann in den meisten Fällen davon ausgegangen werden, dass zumindest das Land richtig herausgefunden wird. In vielen Fällen kann die IP-Adresse einem konkreten Bundesland oder sogar einer bestimmten Stadt zugeordnet werden. Das Geolokalisieren ist vor allem dann sinnvoll, wenn schnell erkannt werden soll, aus welchem Land ein Angriff auf das Netzwerk stattfindet. Weiters kann hierdurch darauf geschlossen werden, in welches

Land die meisten Verbindungen aufgebaut werden oder wieviel Bandbreite das jeweilige Land reserviert. Die Geolokalisierung findet in der Diplomarbeit primär als Datenbankabfrage einer lokalgespeicherten Datenbank und sekundär als Webanfrage über ein Online-API statt. Die zweite Möglichkeit wurde für den Ausfall der Ersteren oder deren erfolglosen Lokalisierens der IP-Adresse implementiert. Dadurch erhält man mit geringen Aufwand eine ausfallsichere Lösung, die das Lokalisieren von IPs ermöglicht.

8.5.6 Erzeugung von Angriffen

Die Angriffe werden wegen der Skalierbarkeit nicht im Skript des Enriches, sondern in einem separaten Programm identifiziert. Dieses Programm wird Incidents genannt, da es Informationen aus allen in Kapitel 5 auf Seite 15 beschriebenen Informationsquellen zusammenfassend analysieren und gegebenenfalls einen Vorfall oder eine Anomalie identifizieren kann. In der Folge werden die individuellen Angriffe kurz vorgestellt und die Erkennung in Form von Pseudocode beschrieben. Eine grafische Erklärung der einzelnen Module ist in Kapitel 2.1 auf Seite 5 zu finden. Dort wird das Incidents-Programm im Rahmen des Enrichers dargestellt.

8.5.7 Portscan

Ein häufig auftretender und somit wichtig zu erkennender Angriff stellt ein Portscan dar. Dabei versucht der Angreifer, offene Ports eines Gerätes oder des gesamten Netzes herauszufinden. Dadurch erhält er Informationen darüber, auf welche Weise er ein bestimmtes Gerät angreifen kann. Ein Portscan ist somit eine Art Vorreiter auf einen folgenden Angriff.

Als Informationsquelle werden ASA Logdaten mit dem Typen *710005* herangezogen. Diese beschreiben verweigerte Verbindungsanfragen. Da ein Großteil der Verbindungsanfragen bei einem Portscan verweigert werden, eignet sich dieser Typ besonders gut für diesen Zweck.

Ein Portscan wird in der Diplomarbeit als solcher identifiziert, wenn:

- eine IP-Adresse
- innerhalb von einer Minute
- mehr als 100 fehlgeschlagene Verbindungsanfragen
- auf mehr als 100 verschiedene Ports

erstellt. Ab dann wird dies als Anomalie in Form eines Datenbankeintrages mit folgenden Attributen gespeichert:

- Die IP-Adresse des angreifenden Gerätes
- Die IP-Adresse des angegriffenen Gerätes
- Die Anzahl der Verbindungsversuche
- Die Anzahl der unterschiedlichen Ports
- Der Typ *portscan* identifiziert diesen
- Ein Zeitstempel, der den Start des Angriffes darstellt

8.5.8 DNS-Tunnel

Eine weitere häufig auftretende Anomalie stellt ein DNS-Tunnel dar. Dabei werden Informationen als DNS Daten getarnt übertragen. Dies ist insbesondere schlimm, da dadurch unternehmenskritische oder private Information aus dem eigenen Netzwerk gelangen können.

Als Informationsquelle werden ASA Logdaten mit dem Typen *302016* herangezogen. Diese protokollieren den Abbau einer UDP oder TCP Verbindung.

Ein DNS-Tunnel wird in der Diplomarbeit als solcher identifiziert, wenn:

- eine UDP Verbindung
- mit Quell- oder Ziel-Port von 53
- Daten mit einer Größe von mehr als 512 Bytes

übertragen hat. Ab dann wird dies als Anomalie in Form eines Datenbankeintrages mit folgenden Attributen gespeichert:

- Die IP-Adresse des erzeugenden Gerätes
- Die IP-Adresse des empfangenden Gerätes
- Die Größe der übertragenen Daten
- Der Typ *dns_tunnel* identifiziert diesen
- Ein Zeitstempel, der den Start des Angriffes darstellt

8.5.9 DoS-Angriff

Einige Angreifer haben nicht das Ziel, sich selbst zu bereichern oder Informationen zu stehlen, sondern lediglich den größtmöglichen Schaden anzurichten. Dafür eignet sich ein Denial of Service (DoS) Angriff besonders gut. Hierbei werden von einem Gerät zahlreiche Verbindungen zu einem anderen aufgebaut, die verhindern sollen, dass tatsächliche Nutzdaten von dem angegriffenen Gerät übertragen werden können. Dadurch ist das Gerät gar nicht mehr oder nur mit einer großen Latenz für echte Nutzer erreichbar.

Als Informationsquelle werden hier Cisco ASA Logdaten mit den Typen *302016* und *302015* herangezogen. Diese repräsentieren jeweils erfolgreich und nicht erfolgreich aufgebaute Verbindungen.

Eine DoS Angriff wird in der Diplomarbeit als solcher identifiziert, wenn:

- ein Gerät
- an ein anderes Gerät
- innerhalb von einer Minute
- in Summe mehr als 10000
- erfolgreiche oder nicht erfolgreiche Verbindungen

aufzubauen versucht. Ab dann wird dies als Anomalie in Form eines Datenbankeintrages mit folgenden Attributen gespeichert:

- Die IP-Adresse des angreifenden Gerätes
- Die IP-Adresse des angegriffenen Gerätes
- Die Anzahl der erfolgreichen Verbindungsversuche
- Die Anzahl der nicht erfolgreichen Verbindungsversuche
- Der Typ *dos* identifiziert diesen
- Ein Zeitstempel, der den Start des Angriffes darstellt

8.5.10 Abnormales Loginverhalten

Ein von der Norm abweichendes Loginverhalten kann ein Indikator dafür sein, dass entweder jemand versucht, die Anmeldedaten eines Benutzers zu erraten oder aber dafür, dass die Anmeldedaten publik wurden. Eine Anomalie im Loginverhalten lässt sich also aufgrund übermäßig vieler erfolgreicher oder fehlgeschlagener Loginversuche erkennen.

Als Informationsquelle werden Windows Logdaten mit den Typen *4624* und *4624* für die native Windowsanmeldung, Windows Logdaten mit den Typen *6272*, *6273* und *6274* für die WLAN-Anmeldung und Fortigate Logdaten mit den Typen *0100032001* und *0100032002* für die Fortigateanmeldung herangezogen. Dabei beschreibt jeweils ein Logtyp eine erfolgreiche und der andere eine nicht erfolgreiche Anmeldung. Bei der WLAN-Anmeldung wird zusätzlich noch zwischen einer unerfolgreichen Anmeldung aufgrund falschen Passworts und aufgrund falschen Benutzernamens unterschieden. Das Loginverhalten der ASA wird nicht untersucht, da auf dieser nur die Konsolenanmeldung aktiviert ist und man von einer physischen Sicherheit im Normalfall ausgehen kann.

Eine Anomalie im Loginverhalten wird in der Diplomarbeit als solche identifiziert, wenn:

- sich ein Benutzer
- auf einer Plattform
- innerhalb von 3 Minuten
- entweder mehr als 5 Mal nicht erfolgreich oder mehr als 10 Mal erfolgreich

anmeldet. Ab dann wird dies als Anomalie in Form eines Datenbankeintrages gespeichert. In der Datenbank werden je nach Anmeldeweg, auf welchem die Anomalie stattfindet, unterschiedliche Informationen abgelegt. Dabei ist aber immer die Anzahl und der Status der Anmeldeversuche sowie das Gerät selbst und ein Zeitstempel für den Start des Angriffes ersichtlich. In der Datenbank wird eine Anomalie im Anmeldeverhalten durch den Typ *login* (siehe Kapitel 7 auf Seite 43) identifiziert.

8.5.11 Nicht erkennbare Anomalien

8.5.11.1 Login-Bruteforce

Anfangs sollte ein etwaiger Login Bruteforce erkannt werden. Allerdings stellte sich schnell heraus, dass aktuell kaum Brutforces auf die tatsächliche Infrastruktur durchgeführt werden. Das liegt daran, dass die meisten Geräte einen solchen Angriff sehr schnell erkennen und somit blockieren können. In der Realität finden Bruteforces trotzdem noch statt. Im Laufe der Diplomarbeit wurden zwei Bruteforces analysiert:

- Bei *WLAN Bruteforces* wird ein Ausschnitt von dem Datenverkehr im Netz vom Angreifer mitprotokolliert und aufgrund der Verwendung von möglicherweise alten Standards kann dadurch auf die Anmeldedaten rückgeschlossen werden.
- Bei *Hash Bruteforces* wird eine Liste von verhashten Passwörtern herangezogen und aufgrund des stupiden Testens das Passwort nach genügend Zeit herausgefunden. Dabei testet ein Computer jede mögliche Passwortkombination gegen die Hashfunktion und überprüft, ob der Ergebnis Hash gleich dem gesuchten Hash ist. Hier ist vor allem die Passwortkomplexität für die erfolgreiche Durchführung hinderlich.

Bei diesen Bruteforce Angriffen kann dem Systemadministrator nicht mitgeteilt werden, ob ein solcher Angriff aktuell durchgeführt wird, da der Angriff hier nicht gegen die systemeigene Infrastruktur durchgeführt wird. Erst der nachfolgende Angriffsversuch wird mitprotokolliert und kann erkannt werden (siehe Kapitel 8.5.10 auf der vorherigen Seite).

8.5.11.2 Call-Home

Ein Call-Home ist eine immer wiederkehrende Verbindung. Das heißt, eine Verbindung, die in regelmäßigen Abständen zu ein und derselben IP-Adresse aufgebaut wird. Hierbei

sind vorrangig das massive Datenaufkommen und die Komplexität einer effizienten Abfrage die hinderlichen Probleme für eine zeiteffiziente und automatisierte Erkennung des Angriffes. Wegen dieser Gründe kann ein solcher Angriff nicht von Argos erkannt werden.

8.5.11.3 Torrent-Download

Ein Torrent Download weist starke Ähnlichkeiten mit einem herkömmlichen Download auf. Allerdings wird hier ein separates Protokoll verwendet, das ähnlich wie TCP funktioniert. Bei Torrents werden die einzelnen Segmente jedoch nicht von einem, sondern von vielen verschiedenen Servern heruntergeladen. Da dies das Überwachen der Downloads beinahe unmöglich macht, wird dieser Ablauf, nicht nur wie ursprünglich beim Herunterladen von Linux-Distributionen, sondern vor allem auch für illegale Zwecke verwendet. Eine solche Anomalie lässt sich zwar für sich ziemlich leicht erkennen, allerdings werden die meisten solcher Downloads durch ein Virtual Private Network (VPN) durchgeführt. Durch die Verschlüsselung eines solchen privaten Netzes kann nicht mehr nachvollzogen werden, ob aktuell ein Torrentdownload oder ein herkömmlicher Download durchgeführt wird.

8.5.11.4 VPN-Detection

Ein vom Betreiber ungewolltes VPN kann dafür genutzt werden, um unbemerkt Daten zu übertragen. Das kann einerseits von einem Mitarbeiter genutzt werden, um unproduktives Verhalten vor dem Vorgesetzten zu verstecken oder aber von einem Angreifer, um gestohlene Daten unidentifizierbar zu machen. Dabei transversiert der Netzverkehr immer die ASA Firewall. Somit könnte hier eine *Inspection Policy* implementiert werden, die beim Erkennen eines VPNs eine Warnung von sich gibt. Die große Schwierigkeit hierbei ist, dass es unzählige verschiedene VPN-Implementationen gibt. Deswegen gibt es förmlich kein standardisiertes Verfahren, um alle VPNs erkennen zu können.

Vor allem *OpenVPN* ist für dessen Dynamisierung der Ports bekannt. Dadurch kann *OpenVPN*-Administrator die Ports für sein VPN frei wählen und somit als zum Beispiel Webverkehr tarnen. Weiters gibt es noch *IPsec*, *Wireguard* und viele andere VPN-Lösungen, die auf unterschiedliche Protokolle und Ports beruhen.

8.6 Anonymizer

Der Anonymizer ist ein Python Programm, das dafür zuständig ist, die personenbezogenen Daten, die in den Log-Nachrichten enthalten sind, zu anonymisieren bzw.

zu pseudonymisieren. Bei der Anonymisierung werden die originalen Daten durch zufällig generierte ersetzt, ohne dass ein Rückschluss auf die ursprünglichen Daten möglich ist. Bei der Pseudonymisierung werden die originalen Daten, so wie bei der Anonymisierung, durch zufällig generierte ersetzt, es wird hingegen gespeichert, welche Daten durch welche ersetzt wurden. Diese Zuordnung, welche originalen Daten, welchen zufällig generierten entsprechen, wird in Comma-Separated Values (CSV)-Dateien gespeichert.

Mit Hilfe des *Anonymizers* ist es möglich, vier verschiedene Typen von Daten zu verarbeiten:

- IP-Adressen: Hierbei handelt es sich um IP-Adressen in der Dezimalpunktschreibweise.
- MAC-Adressen: Hierbei handelt es sich um ganz normale MAC-Adressen in unterschiedlichsten Formaten, die alle vom *Anonymizer* verarbeitet werden können.
- Benutzernamen: Hierbei handelt es sich um Benutzernamen, die z. B. zur Anmeldung verwendet werden.
- Benutzer-IDs: Hierbei handelt es sich um IDs, wie z. B. der SID, der in Windows NT automatisch vergeben wird.

Um diese Daten zu anonymisieren bzw. zu pseudonymisieren, wird zufällig eine alphanumerische Zeichenkette, bestehend aus American Standard Code for Information Interchange (ASCII) Groß- und Kleinbuchstaben und Ziffern von eins bis neun, generiert. Dies wird von der folgenden Funktion durchgeführt:

```
def generate_random_string(length: int) -> str:
    return ''.join([random.choice(string.ascii_letters + string.digits)
                    for n in range(length)])
```

Diese zufällig konstruierte Zeichenkette wird jedoch nicht einfach so verwendet. Vor die zufälligen Zeichen werden die Zeichen `Arg_` gesetzt, um zu symbolisieren, dass es sich um eine vom *Anonymizer* generierte Zeichenkette handelt. Weiters wird zwischen `Arg_` und den zufälligen Zeichen eine für den Typ der Daten spezifische Zeichenkette gehängt, um diese später unterscheiden zu können. So wird bei einer IP-Adresse `ip_`, bei einer MAC-Adresse `mac_`, bei einem Benutzernamen `user_` und bei einer Benutzer-ID `userid_` hinzugefügt.

So wird z. B. aus der IP-Adresse `192.168.1.1` nach der Verarbeitung durch den *Anonymizer* die Zeichenkette `Ar_ip_aYEmVGZg9Ay1t03`, die dann so für die weitere Verarbeitung in Argos verwendet wird.

9 Auswerten von Logdaten

9.1 Searcher

Der *Searcher* bietet ein einfaches Interface für die Abfragen des Dashboards (siehe Kapitel 9.3 auf Seite 89). Er besteht grundsätzlich aus zwei Komponenten:

- dem *Searcher* selbst, welcher für komplexere Abfragen und
- der API, welche eher einfachere Abfragen verarbeitet.

Der *Searcher* selbst ist als Daemon konzipiert, um die Antwortzeiten an den Client zu verkürzen. Im Endeffekt wird der *Searcher* nur für Abfragen verwendet, in denen ein gewisses Nutz- oder Logdatenvolumen von Benutzern und Geräten über einen bestimmten Zeitraum gefragt ist. Die Anfrage beginnt mit einem `select` gefolgt von den Typen von Logdaten, die ausgewählt werden sollen. Danach kann ein Zeitraum angegeben werden:

- `since`: seit einem fixen Zeitpunkt
- `last`: beginnend beim angegebenen Zeitinterval vor dem aktuellen Zeitpunkt
- `for`: vom beginnenden Zeitpunkt ein gewisses Zeitintervall
- `to`: bis zu einem fixen Zeitpunkt

Optional kann man die Anfrage noch mit `|` (Pipe) aggregieren. Nach dem `|` folgen die Parameter, nach denen gruppiert werden soll. Mit dem Schlüsselwort `interval=` kann angegeben werden, in welchem Zeitinterval gruppiert werden soll. Andere Parameter werden als Schlüssel verwendet und zusätzlich mit `->` umbenannt. Der spezielle Parameter `sum()` gibt an, dass die Werte dieses Schlüssels summiert werden. Zur Veranschaulichung ein Beispiel:

```
select cisco:asa/302016, cisco:asa/302014, since 2020-03-10, for 24h
| group interval=1m, data.ipp_out.addr -> addr, sum(data.bytes) ->
count
```

Die API des *Searchers* ist nur auf statisch festgelegte Abfragen programmiert. Diese Abfragen dienen vor allem dazu, von der Web-Application des *Dashboards* Informationen aus der DB abzufragen.

9.2 Query-Language

Die *Argos-Query-Language* (AQL) ist ein Teil des *Searches*, kann aber unabhängig von ihm verwendet werden. Die AQL ist ein einzelnes Programm, das von anderen Programmen (z. B. dem Dashboard) verwendet wird, um *Query Strings* zu vervollständigen und in der DB auszuführen. Dieses Feature soll es Benutzern des Dashboards ermöglichen, einfache benutzerdefinierte Abfragen der DB durchzuführen und so einen schnelleren Überblick über alle Daten zu erhalten. Die Syntax ist sehr an die von bereits bestehenden Lösungen (siehe Kapitel 3 auf Seite 7) angelehnt. *Query-Strings* sind einfach betrachtet leerzeichengetrennte Schlüssel-Wert-Paar (*Key-Value-Pairs*). Die Schlüssel werden so übernommen, dass im `data`-Feld der DB die Schlüssel vorkommen müssen bzw. auch dem Wert entsprechen müssen. Es gibt einige vordefinierte Schlüssel:

- `_TYPE` – entspricht dem `type`-Feld in der DB.
- `_HOST` – wählt alle Nachrichten aus, in denen die angegebene IP-Adresse bzw. der angegebene Hostname vorkommt.
- `_PORT` – wählt alle Nachrichten aus, in denen die angegebene Portnummer vorkommt.
- `_MAC` – wählt alle Nachrichten aus, in denen die angegebene MAC-Adresse vorkommt.
- `_USER` – wählt alle Nachrichten aus, in denen der angegebene Benutzername vorkommt.

Es folgt ein kurzes Beispiel für einen *Query-String*:

```
_TYPE=cisco:asa/106015 _HOST=192.168.1.1 _PORT=80 tcp_flags=RST
```

Es werden alle Nachrichten ausgewählt,

- die vom Typ `cisco:asa/106015` sind,
- in denen die IP-Adresse 192.168.1.1 vorkommt,
- in denen der Port 80 vorkommt,
- die das TCP-Flag `RST` gesetzt haben (`data.tcp_flags` = `RST`).

Das Python-Programm wird wie folgt ausgeführt:

```
./aql.py QUERY_STRING [-s] [-e] [-f DATE] [-t DATE] [-p POS]
```

Query-String
: Der vollständige oder noch unvollständige *Query-String*, wie oben beschrieben.

`-s, --suggest`
: Ein Vorschlag für die Vervollständigung des Wortes beim Cursor (`-p` oder beim letzten Zeichen) wird zurückgegeben.

-e, --execute
Die Query wird ausgeführt und zurückgegeben.

-f, --from-date
Die Query beinhaltet nur Nachrichten ab dem angegebenen Zeitpunkt.

-t, --to-date
Die Query beinhaltet nur Nachrichten bis zu dem angegebenen Zeitpunkt.

-p, --position
Die Position des Cursors im *Query-String*. Vor allem verwendet vom Dashboard. Der Standardwert ist –1, also das letzte Zeichen im String.

9.3 Dashboard

Um die gespeicherten Daten visuell darzustellen, wurde ein Webinterface erstellt. Dieses bietet einen Überblick über die wichtigsten Informationen, die den Log-Nachrichten entnommen werden können.

Um das User Interface (UI) möglichst ansprechend zu gestalten, wurde ein *Bootstrap Template* verwendet, welches einige Möglichkeiten bietet, die Daten darzustellen. Diese Funktionen reichen von Tabellen über Komponenten wie Buttons oder Karten, die kurze Infos geben, bis hin zur einfachen Implementierung von Graphen, mit denen die Daten sehr einfach aufgeschlüsselt werden können.

Genau handelt es sich um das *Sufee HTML5 Admin Dashboard Template*, welches von Colorlib entwickelt und unter der MIT-Lizenz veröffentlicht wurde.

Als Web-Server wurde der Apache HTTP Server verwendet, welcher auf dem Log-Server installiert wurde, um die Anzahl an benötigten physischen/virtualisierten Geräten in der Topologie nicht unnötig zu erhöhen.

Die Kommunikation zwischen *Javascript*, mit dem die Tabellen und Graphen dargestellt werden und der MongoDB Datenbank, in der die Daten gespeichert werden, wird über eine API bereitgestellt. Diese API wurde in den Programmiersprachen PHP: Hypertext Preprocessor (PHP) und Python programmiert, wobei hierbei Python direkt mit der Datenbank interagiert und PHP als Mittelsmann für diese dient und die Antwort an JavaScript weiterreicht.

9.3.1 Landing-Page

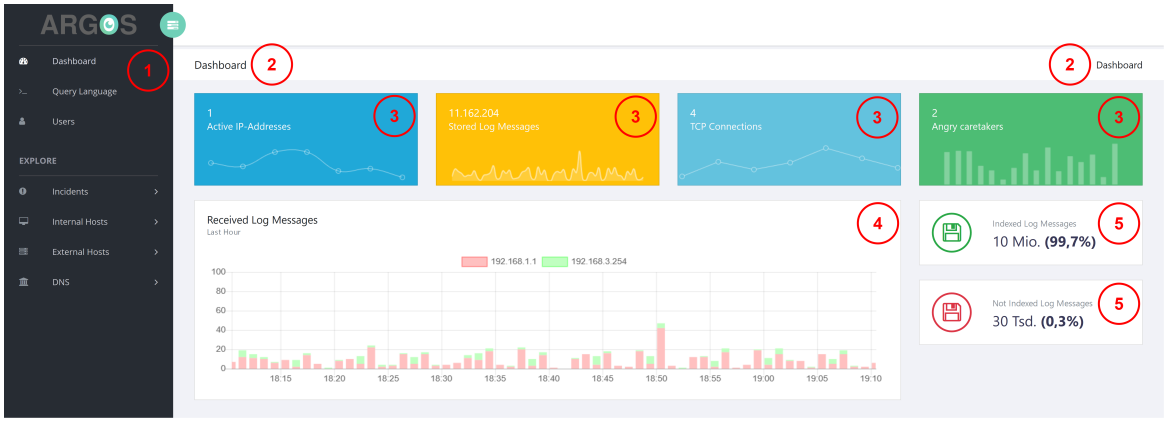

Abbildung 9.1: Die Landing Page des Dashboards

In der Abbildung 9.1 kann man die *Landing Page*, also die Seite, die gezeigt wird, wenn man das Argos-Dashboard öffnet, sehen. Die wichtigsten Komponenten sind zu Erklärungszwecken jeweils mit einer Nummer versehen, die auf der echten Weboberfläche nicht zu sehen sind. Die einzelnen Komponenten sind:

1. Auf der linken Seite befindet sich eine Navigationsleiste. Diese zieht sich durch alle Seiten des Dashboards hindurch und erlaubt es, sofort jeden Teil des Dashboards zu öffnen, ohne den Überblick zu verlieren.

2. Ein weiteres Element, das sich durch alle Seiten hindurchzieht, ist die *Breadcrumb*-Leiste. Sie zeigt, auf welcher Seite man sich momentan befindet und ob es sich bei dieser um eine Unterseite einer anderen Seite handelt.

3. Darunter befinden sich vier *Widgets*, in denen in kurzer Form momentane Informationen über das Netzwerk und das Sammeln von Log-Nachrichten angezeigt werden.

4. Der Graph in der Mitte der Seite zeigt die momentanen Log-Nachrichten Quellen in unterschiedlichen Farben an, wobei die Anzahl an gesendeten Log-Nachrichten zeitabhängig dargestellt wird.

5. Die *Widgets* rechts zeigen, wie viele Log-Nachrichten erfolgreich vom *Indexer* indiziert und wie viele Log-Nachrichten nicht indiziert werden konnten. Denn es kann sein, dass eine Log-Nachricht nicht indiziert wurde, wenn diese Art von Log-Nachrichten von einem Netzwerkgerät zwar an den Log-Server gesendet, dieser diese Art aber nicht verarbeiten kann und daher unindiziert lässt.

9.4 Query-Language im Dashboard

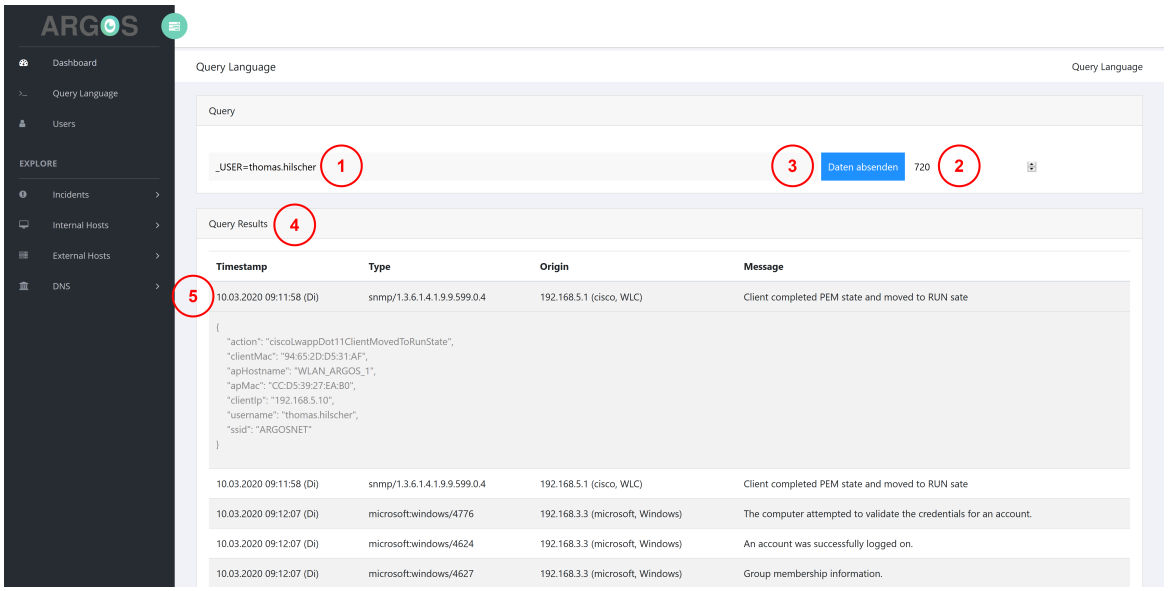

Abbildung 9.2: Die Query Language im Dashboard

Die AQL, die im Kapitel 9.2 auf Seite 88 beschrieben wird, ist ebenfalls am Dashboard zu finden. Hier ist es möglich, die Suchen mittels eines Webinterfaces anstatt des CLIs durchzuführen. Dieses wird in der Abbildung 9.2 dargestellt und enthält folgende Teilbereiche:

1. In dieser Suchleiste wird der *Query-String* für die AQL-Abfrage eingegeben. Weitere Informationen, wie dieser aufgebaut ist, sind im Kapitel 9.2 auf Seite 88 zu finden.

2. Hier wird spezifiziert, welcher Zeitraum durchsucht werden soll. Konkret werden die Stunden angegeben, vor denen die Log-Nachricht spätestens generiert wurde.

3. Mit einem Klick auf dieses blaue Feld wird die Abfrage abgeschickt.

4. In dieser Tabelle sind die Ergebnisse der Abfrage zu finden. Diese sind nach deren *Timestamp* sortiert. Unter *Type* wird der Log-Typ angegeben, der zeigt, um welche Form von Log-Nachricht es sich handelt. Unter *Origin* wird die Quelle der Log-Nachricht angezeigt, und das *Message*-Feld wird genutzt, um eine beschreibende Nachricht für diesen Log-Eintrag zu zeigen.

5. Mit einem Klick auf die Zeile einer Log-Nachricht werden genauere Informationen zu dieser angezeigt. Hierfür werden alle gespeicherten Informationen zu dieser Log-Nachricht als JSON-Objekt dargestellt.

9.5 User-Pages

9.5.1 Users

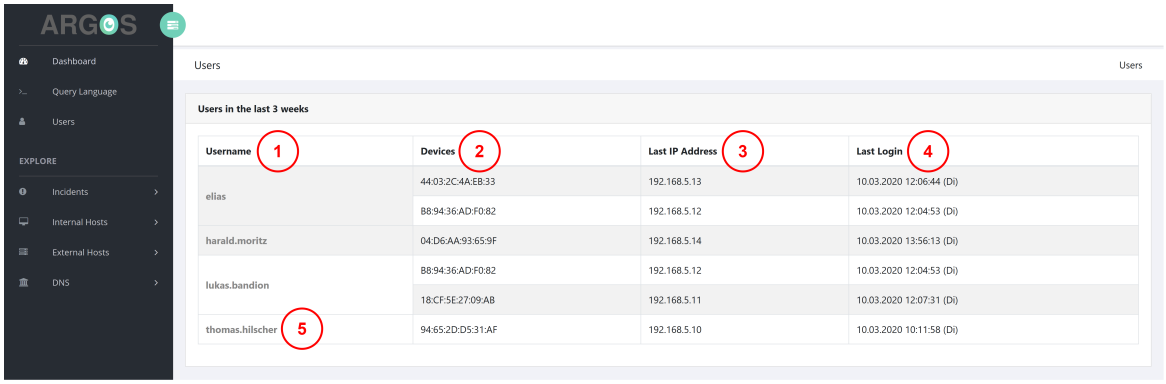

Abbildung 9.3: Ein Überblick über die letzten angemeldeten Benutzer

Ein wichtiger Bestandteil ist es, am Dashboard anzeigen zu können, welche Benutzer gerade im Netzwerk angemeldet sind bzw. sich in den letzten paar Stunden angemeldet haben. Diese Informationen können auf der *Users* Seite des Dashboards eingesehen werden (siehe Kapitel 9.3). Es werden die folgenden Daten zu den einzelnen Benutzer angezeigt:

1. Der Benutzername des Nutzers, der mittels des Anonymizers (siehe Kapitel 8.6 auf Seite 85) pseudonymisiert werden könnte, was in diesem Fall aus Anschaulichkeitsgründen aber nicht gemacht wurde.

2. Für jeden Benutzer werden die Geräte angezeigt, mit denen sich dieser angemeldet hat. Ein Gerät wird hier mittels seiner MAC-Addresse repräsentiert. Einem Benutzer können mehrere Geräte zugeordnet werden.

3. Zu jedem Gerät wird die IP-Adresse angezeigt, die dieses verwendet. Dies ist immer die letzte IP-Adresse, die dieses verwendet hat, da ein Gerät über die Zeit verschiedenste IP-Adressen verwenden kann.

4. Weiters wird das Datum der letzten Anmeldung des Geräts gezeigt.

5. Sollte man auf den Benutzernamen eines Nutzers klicken, kommt man auf die Unterseite dieses Nutzers. Dies wird im Kapitel 9.5.2 auf der nächsten Seite beschrieben.

9.5.2 User

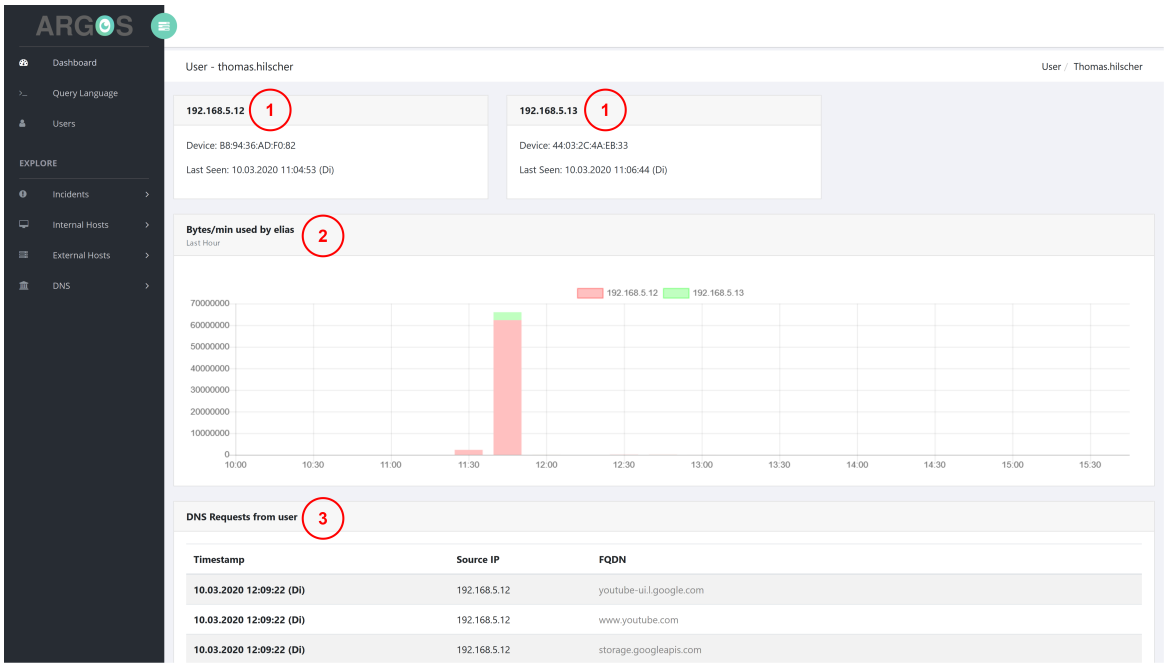

Abbildung 9.4: Genauere Informationen zu einem Benutzer

Für jeden Benutzer gibt es eine Unterseite, die spezifischere Informationen zu diesem anzeigt. Zu diesen gehören:

1. Eine Auflistung aller Geräte, mit denen sich dieser Benutzer im Netzwerk angemeldet hat. Zu jedem Gerät wird oben fett die IP-Adresse angezeigt. Darunter sind die MAC-Adresse des Geräts und das Datum der letzten Anmeldung zu finden.

2. In der Mitte der Seite ist ein Graph zu sehen, der zeigt, wie viel Bandbreite der Benutzer in letzter Zeit benötigt hat. Hier werden alle Geräte des Benutzers in verschiedenfarbigen Balken dargestellt. Einzelne Geräte können auch mit einem Klick auf die Legende ausgeblendet werden. Der Graph wird kontinuierlich neu geladen, um immer die aktuellsten Informationen darzustellen.

3. Ganz unten auf der Seite ist eine Tabelle zu finden, die die DNS-Abfragen des Benutzers darstellt. Die Ergebnisse sind nach dem *Timestamp*, also der Uhrzeit, wann die Abfrage durchgeführt wurde, sortiert. Zu jeder Abfrage wird die IP-Adresse des Geräts, das die Abfrage durchgeführt hat angezeigt. Ganz rechts ist der FQDN zu finden, nach dem in der Abfrage gesucht wurde. Diese Tabelle aktualisiert sich ebenfalls in einem periodischen Zeitintervall.

9.6 Incidents

9.6.1 Incidents-Übersicht

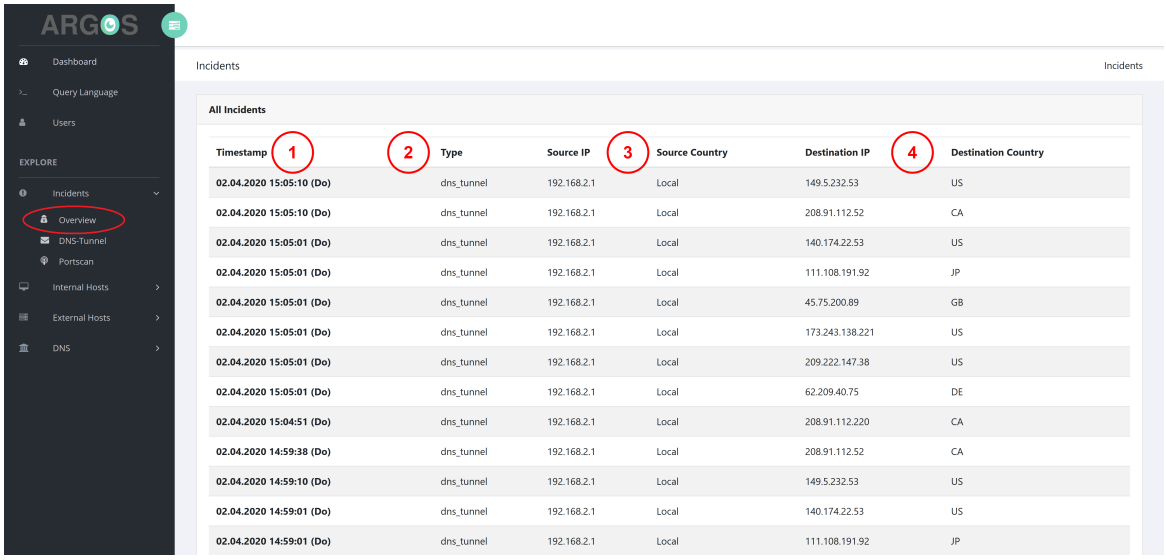

Abbildung 9.5: Übersicht aller implementierten Incidents

Wie im Kapitel 8.5.6 auf Seite 81 beschrieben, wurden im Laufe der Diplomarbeit einige *Incidents* ausgearbeitet, die gewisse Angriffe auf das Netzwerk darstellen. Um diese im Dashboard darzustellen, wurde eine *Incidents*-Übersicht Seite und für jeden *Incident* eine Unterseite erstellt. Die Übersichtsseite ist in der Abbildung 9.5 zu sehen und enthält folgende Informationen:

1. Der *Timestamp* des *Incidents*, also die Uhrzeit zu der dieser eingetreten ist. Die Ergebnisse ist nach dem *Timestamp* sortiert.

2. Um was für einen *Incident* es sich konkret gehandelt hat. Momentan sind zwei verschiedene Typen von *Incidents* implementiert: Der *DNS-Tunnel* und der *Portscan*. In diesem Fall ist anzumerken, dass der *DNS-Tunnel* hier überwiegt, da dieser in der Topologie der Diplomarbeit wesentlich häufiger auftritt.

3. Die *Source IP-Adresse*, von der der *Incident* ausgegangen ist. Zu dieser IP-Adresse kommt noch dazu, aus welchem Land diese IP-Adresse stammt. Dies wird mittels einer lokalen Datenbank abgefragt und bestimmt nur welchem Land diese IP-Adresse zugeilt wurde. Ein Ländercode von *Local* steht für IP-Adressen die aus dem internen Argos-Netzwerk stammen.

4. Rechts befindet sind die *Destination IP-Adresse*, also das empfangende Ende der Verbindung, zu sehen. Zu dieser wird ebenfalls der Ländercode angezeigt.

9.6.2 DNS-Tunnel-Incident

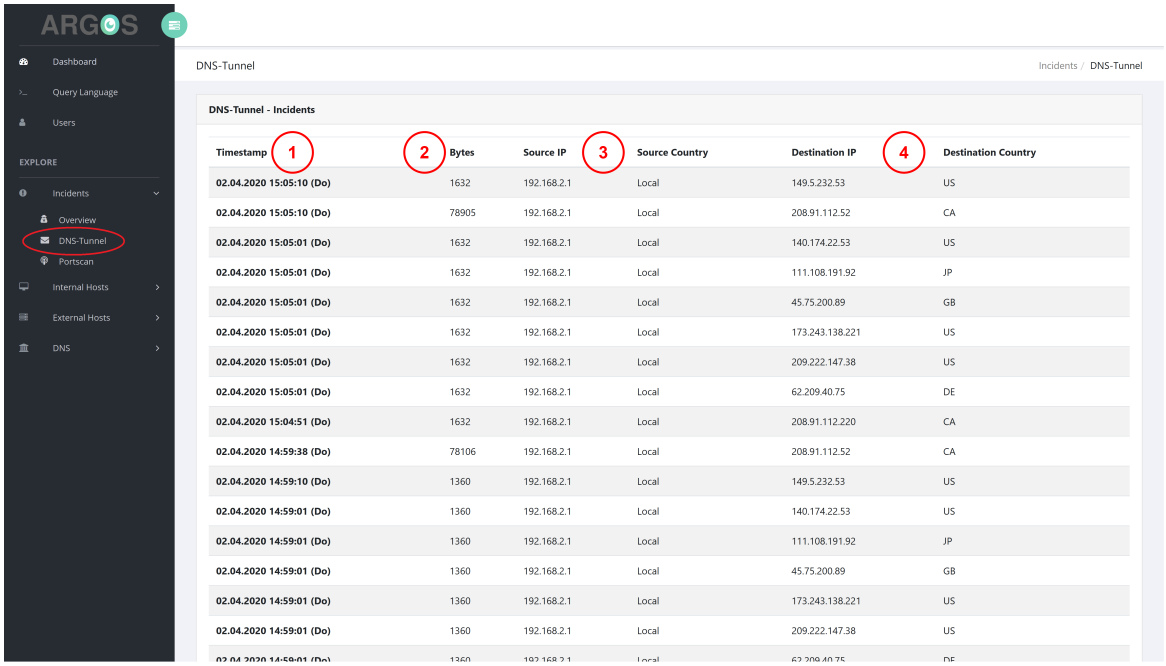

Abbildung 9.6: Der DNS-Tunnel Incident

In der Abbildung 9.6 wird eine der Unterseiten der *Incidents* dargestellt. In diesem Fall handelt es sich um den *DNS-Tunnel Incident*, auf den hier speziell eingegangen wird. Zu diesem werden folgende Informationen angezeigt:

1. Zu Beginn ist wieder der *Timestamp* des *Incidents* zusehen. Die Ergebnisse ist nach dem *Timestamp* sortiert.

2. Hier ist ein für den *Incident* spezifisches Feld zu sehen. Dieses zeigt wie viele Byte Nutzdaten innerhalb dieses DNS-Tunnel übertragen wurden.

3. Mittig befindet sich die *Source IP-Adresse*. Diese ist in diesem Fall die des *Clients*, der versteckte Daten in Form von DNS-Pakten verschickt. Zu diesem wird wieder der Ländercode angezeigt.

4. Auf der rechten Seite befindet sich die *Destination IP-Adresse*. Bei dieser handelt es sich im Fall des DNS-Tunnel um den Server, an den die DNS-Pakete mit versteckten Daten geschickt werden. Zu diesem wird wieder der Ländercode angezeigt.

9.6.3 Portscan-Incident

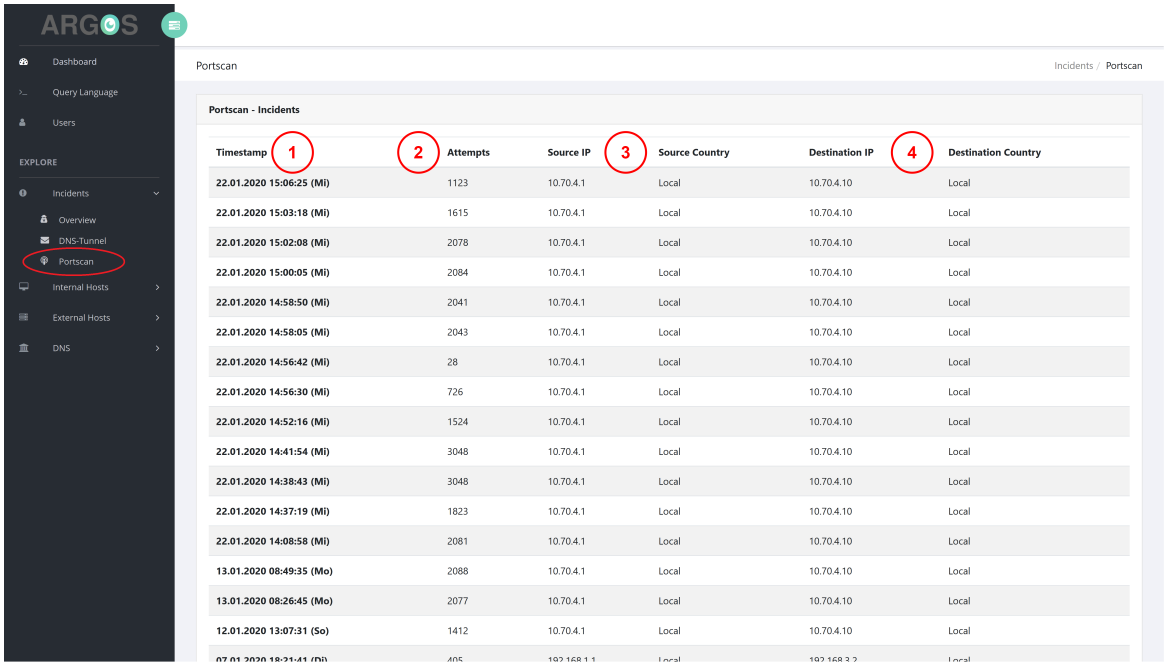

Abbildung 9.7: Der Portscan Incident

In der Abbildung 9.7 wird die zweite der Unterseiten der *Incidents* dargestellt. In diesem Fall handelt es sich um den *Portscan Incident*, auf den hier speziell eingegangen wird. Zu diesem werden folgende Informationen angezeigt:

1. Zu Beginn ist wieder der *Timestamp* des *Incidents* dargestellt. Die Ergebnisse ist wie immer nach diesem sortiert.

2. Hier ist wie beim DNS-Tunnel ein für den *Incident* spezifisches Feld zu finden. Dieses zeigt, wie viele Versuche im Laufe des *Portscan Incidents* stattgefunden haben.

3. Mittig befinden sich wieder die *Source IP-Adresse* und der dazugehörige Ländercode. Bei diesem *Incident* handelt es sich hier um das Gerät, das den Portscan durchgeführt hat.

4. Auf der rechten Seite der Tabelle sind wieder die *Destination IP-Adresse* und deren Ländercode. In diesem Fall handelt es sich hier um das Gerät, dessen Ports durchprobiert wurden.

10 Künstliche Intelligenz

10.1 Einleitung zu Künstlicher Intelligenz

Künstliche Intelligenz (KI) ist ein Teilgebiet der Informatik und beschäftigt sich damit, einen Computer intelligentes Problemlösungsverhalten beizubringen, mit dem es Aufgaben auf neuem und effizienterem Weg lösen kann, für die ein Mensch Intelligenz benötigt. [10, 9]

Heutzutage findet künstliche Intelligenz nahezu überall Anwendung, denn sie ist aus dem alltäglichen Leben nicht mehr wegzudenken. Gerade in Bereichen wie der Spracherkennung, der Bilderkennung und Suchmaschinen kommen wir täglich in Kontakt mit künstlicher Intelligenz. Weniger bewusst ist den meisten Menschen, dass Künstliche Intelligenz (KI) auch schon Einzug in Bereiche wie der Medizin, dem Militär oder der Forschung bzw. Entwicklung gefunden hat.

10.2 Unterschied zwischen Mensch und KI erkennen

Oftmals steht man in der virtuellen Welt vor der Herausforderung überhaupt zu erkennen, ob es sich beim Gegenüber um einen Menschen oder eine Maschine, die mittels KI intelligent wirken soll, handelt. Am häufigsten tritt dies bei sogenannten Chatbots auf, denn Chatbots sind spezialisiert darauf, in einem Chat Antworten zu geben, die möglichst menschenähnlich sind. Oder anders ausgedrückt – Chatbots sind Dialogsysteme mit natürlichsprachlichen Fähigkeiten textueller oder auditiver Art. [2]

Der nach Alan Turing (* 1912; † 1954) benannte Turing Test prüft, ob ein Algorithmus bzw. eine KI für einen Menschen durch verschiedene Fragen zweifelsfrei bestimmt werden kann oder bei bestehender der KI nicht mehr bestimmbar ist.

Ob bis heute eine Maschine den Turing Test bestanden hat, bleibt debattierbar, denn unter gewissen Voraussetzungen konnten Roboter Menschen in die Irre führen, dennoch hat noch keine künstliche Intelligenz wahre Intelligenz gezeigen. Maschinen haben Menschen schon in vielen Kategorien geschlagen, sie können schneller rechnen, beherrschen Schach und schlagen sogar den Weltmeister im Spiel *Go*, dennoch, wenn

man die Schachintelligenz auf *Go* ansetzt oder umgekehrt, funktioniert es nicht mehr. Das Problem, das die meisten KIs heutzutage haben, ist, dass sie nur ein spezielles Thema beherrschen, aber nicht allgemein intelligent sind. Ihr Gehirn ist nur auf eine, wenn auch sehr große Aufgabenstellung programmiert und kann nicht darüber hinaus lernen. Hierzu gibt es auch noch keine nennenswerten Ansätze, und genau da fällt die KI beim Turing Test durch.

10.3 Machine Learning

Machine Learning ist der Überbegriff, dass Maschinen Lösungen anhand vieler Beispiele erlernen können. Es besteht aus vielen Teilgebieten, wie dem Deep Learning (siehe Kapitel 10.5 auf Seite 102), dem Natural language processing und neuronalen Netzwerken.

Über Machine Learning wird, anhand unzähliger Inputdaten, versucht, die Fehler in der Vorhersage zu minimieren, indem zum Beispiel Bilder einem Label zugeordnet werden, indem die Maschine lernt. Diese Art des Lernens findet in der Bilderkennung Anwendung. Dieses Lernen passiert im wichtigsten Teil des Machine Learning, nämlich im neuronalen Netzwerk.

10.4 Neuronales Netzwerk

10.4.1 Das Netzwerk

Ein neuronales Netzwerk ist der Teil der KI, in dem die Berechnungen stattfinden. Die gesamte Berechnung des Outputs erfolgt unabhängig von der Problemart und der Inputdaten im Netzwerk. Ein neuronales Netzwerk kann als vieldimensionale Funktion angesehen werden, die nur versucht, immer weniger Fehler zu machen, um ein Problem genauer zu lösen.

Das Netzwerk besteht dazu aus zwei essenziellen Bestandteilen – die Neuronen und die Gewichte. Das Neuron fasst die Inputdaten der Verbindungen zwischen den Neuronen zusammen und berechnet daraus einen Output, während die Gewichte die Werte der Neuronen auslesen, sie mit ihrem Gewicht verändern und über die Verbindungen an weitere Neuronen weitergeben.

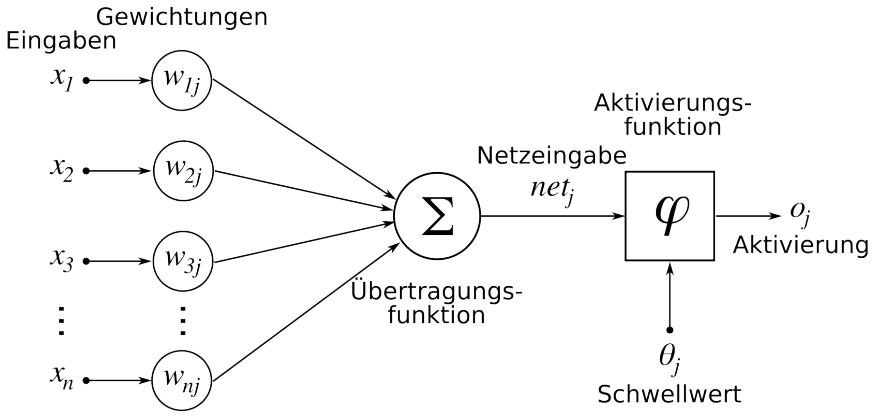

Abbildung 10.1: Bild eines Neurons [1]

10.4.2 Das Neuron

Ein Neuron ist für die Berechnung zuständig, indem es die Werte der vorherigen Neuronen mit den Gewichten multipliziert und summiert. Danach wird meist um einen Bias (einem statischen Wert) addiert. Durch verschiedene nicht-lineare Aktivierungsfunktionen, siehe Kapitel 10.4.3, wird die Summe normalisiert. Das Ergebnis dient dann wieder als Eingabe des nächsten Neurons.

Neuronen sind dem im Gehirn befindlichen Neuronen nachempfunden, doch für neuronale Netzwerke angepasst. Während im Gehirn die häufigste Neuronen Art nur die Inputs zusammenfasst und bei Überschreitung eines gewissen Wertes selbst ein Signal ausstößt, werden in neuronalen Netzwerken meist relative Ausgangswerte zum Eingangswert weitergegeben. In den folgenden Absätzen werden dazu zwei wichtige Vertreter der sogenannten Aktivierungsfunktionen vorgestellt.

10.4.3 Aktivierungsfunktionen

10.4.3.1 Aktivierungsfunktionen Allgemein

Die Aktivierungsfunktion beschreibt, wie die Summe des Inputs eines Neurons verarbeitet wird, um diese dann wieder an das nächste Neuron weiterzugeben. Die Wahl der Aktiverungsfunktion hat einen großen Einfluss auf die Funktion des neuronalen Netzwerkes, aber sie hat auch einen großen Einfluss auf die Performance.

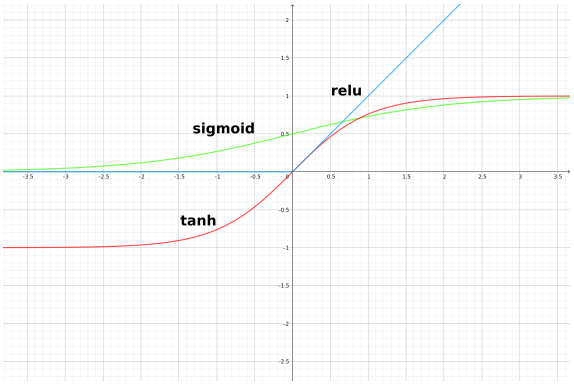

Abbildung 10.2: Verschiedene Aktivierungsfunktionen zum Vergleich.

10.4.3.2 Sigmoidfunktion

Oftmals kommt die Sigmoidfunktion als Aktivierungsfunktion bei neuronalen Netzwerken zum Einsatz. Der S-förmige Graph der Funktion hat den Vorteil, dass die Ausgangswerte zwischen 0 und 1 sind.

$$sig(x) = \frac{1}{1 + e^{-x}}$$

10.4.3.3 Tanh Funktion

Die Tanh Funktion sieht der Sigmoidfunktion sehr ähnlich und bewirkt auch als Aktivierungsfunktion dasselbe. Im Unterschied zur Sigmoidfunktion befindet sich die Tanh Funktion im Intervall zwischen -1 und +1 und ihre Steigung und die daraus resultierende Lernrate ist höher.

$$f(x) = tanh(x)$$

10.4.3.4 Rectifier Funktion

Die Rectifier Funktion kurz relu-Funktion für Rectifier Linear Unit schneidet nur negative Werte weg. Diese Funktion kommt in den letzten Jahren immer mehr zur Anwendung, denn sie hat, obwohl sie sehr simpel erscheint, sehr gute Ergebnisse und genau, weil sie so einfach ist, auch eine sehr gute Performance.

$$relu(x) = max(0, x)$$

10.4.4 Gewichte

Gewichte bzw. Gewichtungen bilden den veränderbaren Teil des Netzwerkes, denn während ein neuronales Netzwerk lernt, werden die Gewichte angepasst, indem sie mit der sogenannten Lernrate verändert werden. Das Gewicht nimmt den Wert von einem Neuron, multipliziert es mit sich und gibt den neuen Wert dann wieder in das nächste Neuron weiter. Die Veränderung der Gewichte kann auf verschiedenste Art passieren – zufällig, genaustens berechnet mit Backpropagation (siehe Kapitel 10.4.5.1) oder semi-zufällig mit Verstärkung bei positiver Veränderung oder Zurücknahme bei negativer Veränderung, wie es in Reenforcement Learning verwendet wird.

10.4.5 Optimizer

10.4.5.1 Backpropagation

Um von dem Ergebnis eines Rechenzyklusses, mathematisch auf die spezifische Veränderung eines Gewichtes zu kommen, werden Optimizer verwendet. Diese ermöglichen es aufgrund des Fehlers des Netzwerkes anhand von verschiedenen Verfahren das neuronale Netzwerk zu verbessern. Backpropagation beschreibt den Vorgang, um vom Fehler einer Vorhersage auf die Veränderung eines einzelnen Gewichtes schließen zu können. Backpropagation beschreibt damit den generellen Prozess, der in den meisten Optimizern wie zum Beispiel Root Mean Square Propagation (RMSProp) Optimizer und Adam Optimizer zur Anwendung kommt. Es stellt damit ein sehr weit verbreitetes Verfahren zum Erlernen künstlicher neuronaler Netzwerke dar.

10.4.5.2 RMSProp Optimizer

Der RMSProp Optimizer basiert wie die meisten Optimizer auf dem Backpropagation Verfahren. Der Vorteil von diesem ist, dass er für jedes Gewicht eine dynamische Lernrate benutzt und somit besonders schnell lernt, da unterschiedliche Gewichte andere Lernraten brauchen, um schnellstmöglich das beste Ergebnis zu erzielen.

10.4.5.3 Adam Optimizer

Der Adaptive Moment Estimation (Adam) Optimizer ist eine verbesserte Version des RMSProp Optimizers, da er noch weiter optimiert und daher sogar noch performanter rechnet. Genauso wie der RMSProp Optimizer benutzt dieser eine dynamische Lernrate für jedes einzelne Gewicht.

10.5 Deep Learning

Deep Learning ist ein Unterbereich des Machine Learnings und ist besonders bei komplexen Problemen, wie der Image Recognition oder der Spracherkennung wichtig, da die Komplexität bei dieser Form von Problemen signifikant ansteigt. Deep Learning bedient sich der Backpropagation, einem Prozess, um den Fehler schnell zu ermitteln und die Gewichte demensprechend zu verändern. Damit kann das neuronale Netzwerk gezielt angepasst werden, um einen schnellen Lernerfolg zu erzielen. Deep Learning beschreibt meist größere und komplexere Netzwerke, die nicht mehr auf einfachen PCs oder Laptops berechnet werden können, sondern auf ganzen Serverfarmen mit GPUs berechnet werden müssen. Weiters nutzen Deep Learning Modelle, anders als Machine Learning Modelle, nicht nur Forwardpropagation, also die Berechnung hin zum Output, sondern auch schon im Berechnungszyklus, Backpropagation, um somit besonderes Verhalten, wie die Nachbildung eines Gedächtnisses, herauszufordern. Zusätzlich wird in Deep Learning das neuronale Netzwerk zwar größer und komplexer, jedoch kann mit dynamischen Verbindungen zwischen den Neuronen auch Gewichte weg- oder hinzugefügt werden, die über mehrere Layer hinweg gehen können.

10.6 Inputdaten

10.6.1 Inputdatentypen

Bei der KI spielen die Inputdaten eine große Rolle, denn bei Machine Learning kommt es darauf an, nicht nur die Daten blind in eine KI zu füttern, sondern zuerst die Daten für eine KI aufzubereiten, damit diese effizient lernt bzw. überhaupt das lernt, was von ihr verlangt wird. Zur Datenaufbereitung zählt unter anderem zu erkennen, um welche Form der Daten es sich handelt. Verschiedene Datenarten bedeuten, dass nicht jede Art der KI geeignet ist oder überhaupt Ergebnisse liefern kann. Meist handelt es sich um eine Vielzahl verschiedener Datenarten, doch kann man diese in unterschiedliche Arten gliedern.

10.6.2 Die Intervallskala

Die Intervallskala ist die bekannteste Datenart, denn jede messbare Größe kann man in der Intervallskala darstellen. Daten in der Intervallskala haben zueinander einen Bezug und man kann sagen, dass zwei Meter doppelt so lang sind wie ein Meter. Damit kann man diese Daten normalisieren, indem man die Varianz berechnet und dann die normalisierten Daten als Input des ersten Neurons eingibt.

10.6.3 Die Ordinalskala

Die Ordinalskala beschreibt Zahlen, die zwar durch natürliche Zahlen abgebildet werden können, jedoch keine relative Bedeutung zueinander haben. Man kann zwar sagen, dass die Note 1 besser ist als die Note 2, dennoch war die Note 1 nicht doppelt so gut wie die Note 2. Ein weiteres Beispiel dafür sind Rennplatzierungen, bei denen es oft nur um Hundertstel geht, dennoch die Plätze 1, 2 etc. verteilt werden. Oftmals stehen die Rohdaten zur Verfügung, mit denen diese Platzierung zustande gekommen sind, und diese wären für ein neuronales Netz in den meisten Fällen die präferierten Inputdaten, dennoch kann man die Ordinalskala nicht immer vermeiden. Die Ordinalskala kann man in diesem Fall wie die Intervallskala durch die Standardnormalverteilung abbilden, doch diese würde für die KI den Anschein erwecken, dass die Note 1 doppelt so gut war wie die Note 2. Hierfür wurde die One-Hot-Codierung entwickelt – die Codierung dient dazu verschiedene nicht miteinander verknüpfte Inputdaten jeweils getrennt voneinander zu betrachten. Die One-Hot-Codierung wird dazu, in einem Beispiel der Nominalskala erklärt.

10.6.4 Die Nominalskala

Die Nominalskala steht für Werte, die zwar logisch miteinander verknüpft sind, dennoch nichts miteinander zu tun haben. Ein Beispiel wäre dafür die Augenfarbe, denn diese kann blau, grün, braun etc. sein, trotzdem ist keine besser als eine andere oder doppelt bzw. ein Vielfaches. Die Nominalskala stellt damit einen Zustand aus 1 aus n dar, der voneinander unabhängig ist. Damit dieser in eine KI eingespielt werden kann, muss die One-Hot-Codierung verwendet werden, damit dieser richtig abgebildet wird.

10.6.5 One-Hot-Codierung

Die One-Hot-Codierung gibt mehreren, miteinander verknüpften Neuronen die Funktion einen Zustand darzustellen. Ist die Augenfarbe eines Menschen zum Beispiel grün, leuchtet nur das zweite Neuron, sein Input ist auf 1 gesetzt, während die anderen Neuronen auf 0 gesetzt werden. Wenn die Aufgabenfarbe des zweiten Menschen braun ist, wird nur das dritte Neuron auf 1 gesetzt. Das gleiche kann für die Platzierung bei einem Rennen oder bei den Noten angewendet werden, doch es müssen immer so viele Neuronen sein, wie es Zustände gibt. Die Anzahl der Neuronen steigt durch die Verwendung der One-Hot-Codierung sehr schnell an, doch damit werden die Inputdaten rein faktisch richtig dargestellt.

10.7 Arten von Problemstellungen

10.7.1 Klassifikation

Bei der Klassifikation geht es darum, dass die KI aus einer Menge von Label erkennt, mit welchem dieser die Inputdaten am besten zusammenpassen. Die Klassifikation findet meist in der Imagerekognition statt, denn hier wird versucht, von für die KI unbekannten Bildern darauf zu schließen, ob es sich bei dem Bild um zum Beispiel eine Katze, einen Hund oder einen Vogel handelt. Der Programmierer muss sich hier bewusst werden, dass die KI keine Ahnung hat, um was es sich bei dem Label handelt, sondern sie nur gelernt hat, dass manche Eigenschaften für ein Label charakteristisch sind und dass es damit den geringsten Fehler produziert. Beim Output eines Klassifikationsnetzwerkes handelt es sich meist um die One-Hot-Codierung siehe Kapitel 10.6.5 auf der vorherigen Seite, der dann anhand des Neurons mit dem höchsten Output erkennt, dass es sich um dieses zu einer gewissen Wahrscheinlichkeit handelt. Das große Problem der Klassifikation ist, dass im Vorhinein Unmengen von Daten händisch oder durch eine andere Weise klassifiziert werden müssen, damit die KI dann aus den richtigen Labeln auf Daten schließen kann, die sie noch nicht gesehen hat. Bei der Klassifikation handelt es sich somit um ein sogenanntes Supervised Learning siehe Kapitel 10.8.1 auf der nächsten Seite.

10.7.2 Clustering

Beim Clustering geht es in erster Linie darum, dass die KI selbst Muster in den Daten erkennt, die dem Nutzer der KI womöglich noch nicht einmal aufgefallen sind. Dazu werden die Inputdaten von der KI in verschiedene Cluster eingeteilt, bei denen sich die Daten clusterintern wenig unterscheiden, doch zwischen den Clustern gibt es große Unterschiede in den Datensätzen. Ein klassisches Beispiel des Clustering wäre, dass nur anhand des Einkaufsverhalten die KI auf das Geschlecht eines Menschen schließen kann. Hierzu hat die KI stark vereinfacht erkannt, dass Männer eher Elektronik Artikel einkaufen und Frauen eher Gewand. Die KI erkennt nur eine kleine Abweichung zwischen den Elektronik Artikel und clustert die Kunden zusammen, während auf der anderen Seite ein Cluster aus Kunden entsteht , der vermehrt Gewand kauft. Bei Clustering kommt Unsupervised Learning zur Anwendung, bei dem es anfangs noch nicht klar ist, was die KI letztendlich erkennen wird. Die gleiche KI Konfiguration hätte mit anderen Zufallsgewichten am Anfang einen Zusammenhang zwischen teureren Artikeln und reicheren Kunden erkennen können, vollkommen unabhängig vom Geschlecht, siehe Kapitel 10.8.2 auf der nächsten Seite.

10.8 Verschiedene KI-Systeme

10.8.1 Supervised Learning

Beim Supervised Learning wird ein im Vorfeld erstelltes Datenset mit den erwarteten Ergebnissen versehen (gelabelt). Diesen bereits ausgewerteten Datensatz nutzt das neuronale Netzwerk, um mit Backpropagation die Gewichte auf die Trainingsdaten anzupassen. Davor wird der Datensatz in Trainings- und Testdaten geteilt, um nach dem Lernen der Trainingsdaten mit den Testdaten die richtige Funktion der KI zu validieren. Als Beispiel für Supervised Learning kann die Klassifikation gesehen werden, siehe Kapitel 10.7.1 auf der vorherigen Seite.

10.8.2 Unsupervised Learning

Unsupervised Learning bezieht sich, im Gegensatz zum Supervised Learning, nicht auf zuvor bestimmte Label, sondern es versucht, diese selbstständig zu finden. Dadurch, dass eine Unsupervised Learning KI keine Vorgaben erhält, muss sie Zusammenhänge in den Daten selbstständig erkennen. Diese Zusammenhänge können für den Nutzer zum einen sehr offensichtlich sein oder zum anderen völlig neue Informationen bringen. Das bekannteste Beispiel für Unsupervised Learning ist das Clustering, siehe Kapitel 10.10.6 auf Seite 118. Das Problem, dass hierbei entsteht ist, dass die KI Cluster erkennen kann, welche für den Menschen keinen Sinn ergeben können bzw. die der Mensch nicht erkennt. Um dieses Problem zu lösen muss man, mit Verständnis der Inputdaten, siehe Kapitel 10.6 auf Seite 102, eine Anpassung vornehmen, um das gewünschte Verhalten der KI zu unterstützen.

10.8.3 Reinforcement Learning

Als Hybrid dieser beiden Ansätze kann das Reinforcement Learning gesehen werden, da es die Vorteile der anderen beiden Technologien vereint. Es verbindet den unüberwachten Lernzyklus des Unsupervised Learnings mit den überwachten Verbesserungen des Netzwerks des Supervised Learnings, indem erwünschtes Verhalten bestärkt wird. Dies führt in weiterer Folge dazu, dass die KI gewünschtes Verhalten lernt, ohne dass jemals Datensätze präpariert werden mussten. Oftmals werden Reinforcement Learning Modelle mit der Evolution verglichen, da dies eine Art des Trainings der KI sein kann. Dazu werden viele Modelle erstellt, um das gleiche Problem zu lösen, doch nur die besten werden weiterverfolgt. Damit werden über Generationen von Modellen immer bessere Modelle entwickelt. Dies entspricht einer Analogie zur Evolution, da auch dort nur das überlebensfähigste Lebewesen fortbesteht und neue Generationen hervorbringt.

Über Generationen entsteht daraus das am besten angepasste Lebewesen oder - wie beim Reinforcement Learning - das am besten funktionierende Modell.

10.9 Deep Learning mit einem IDS/IPS

10.9.1 TensorFlow

Für die Umsetzung der Künstliche Intelligenz wurde im Zuge der Diplomarbeit TensorFlow gewählt, ein vom Google-Brain Team entwickeltes KI Framework. TensorFlow baut die Keras Bibliothek auf, diese bietet eine Schnittstelle, um effizient mit der Entwicklung von komplexen neuronalen Netzwerken zu beginnen zu können.

Um ein grundsätzliches Verständnis von TensorFlow zu bekommen, empfehlen sich die Tutorials der offiziellen Website, da diese die ersten Schritte verständlich und gut strukturiert erklären. [15]

10.9.2 Multiseries Forecasting

10.9.2.1 Zeitabhängige Prognosen

Bei der Prognose des Datenaufkommens ist vor allem die Abhängigkeit der Zeit von großer Bedeutung, da das Datenaufkommen zwar sprunghaft ist, dennoch einem gewissen Trend verfolgt. Je nach Modell berücksichtigen oder vernachlässigen neuronale Netzwerke diese Veränderungen über die Zeit. Da Time Series Forecasting Modelle auf diese Veränderungen eingehen, kann man unter anderem Anwendungsgebiete wie Wettervorhersagen, Messwertanalysen und Aktienkursvorhersagen gut mit ihnen abdecken. Für die Diplomarbeit wurde dieses Modell angepasst, um auf den Trend in Datenaufkommen einer Netzwerkanalyse einzugehen.

10.9.2.2 Wetterdaten Tutorial

Die offizielle TensorFlow Website stellt auch ein Tutorial zu Time Series Forecasting bereit. [16] Dieses basiert auf den Daten des Max-Planck-Institutes für Biogeochemie in Jena, die Wetterdaten, im Zeitraum zwischen 2003 und 2016, im 10-Minutentakt gemessen haben. Das Tutorial geht im ersten Schritt nur auf ein Feature ein, doch der Datensatz enthält 14 verschiedene Features, wovon in einem weiteren Schritt das neuronale Netzwerk lernt, auf bis zu drei einzugehen. Dies wird in folgenden Abschnitten als Multiseries Forecasting bezeichnet. In den Inputdaten kann der Verlauf

des Luftdruckes in blau, den Verlauf der Temperatur in orange und den Verlauf der Dichte in grün betrachtet werden. Hierbei wurde nur auf sieben der 13 Jahre eingegangen, um die Datenmenge und damit die Berechnungszeit einzugrenzen.

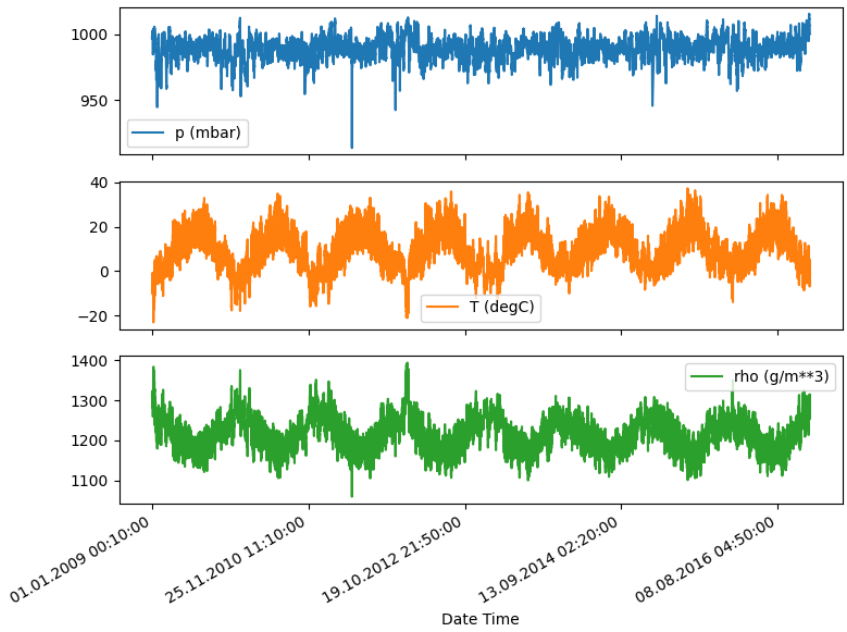

Abbildung 10.3: Die Inputdaten des Multiseries Forecasting Tutorials

10.9.2.3 Multiseries Forecasting in Aktion

Im folgenden Abschnitt wird das Programm zur Prognose der Wetterdaten genauer betrachtet, wobei aufgrund der geringen Relevanz für die vorliegende Diplomarbeit auf eine genaue Analyse der Daten an sich verzichtet wird. Weiterführende Informationen dazu finden sich im Anhang.

Zuerst wird mittels `pandas` [14], einem Datenanalysetool für Python, die Rohdaten eingelesen, die wichtigen Felder herausgefiltert und dargestellt, wie in Abbildung 10.3 ersichtlich.

```
df = pd.read_csv('jena_climate_2009_2016.csv')
                        # Druck     , Temperatur , Dichte
features_considered = ['p (mbar)', 'T (degC)' , 'rho (g/m**3)']
features = df[features_considered]
features.index = df['Date Time']
features.plot(subplots=True)
```

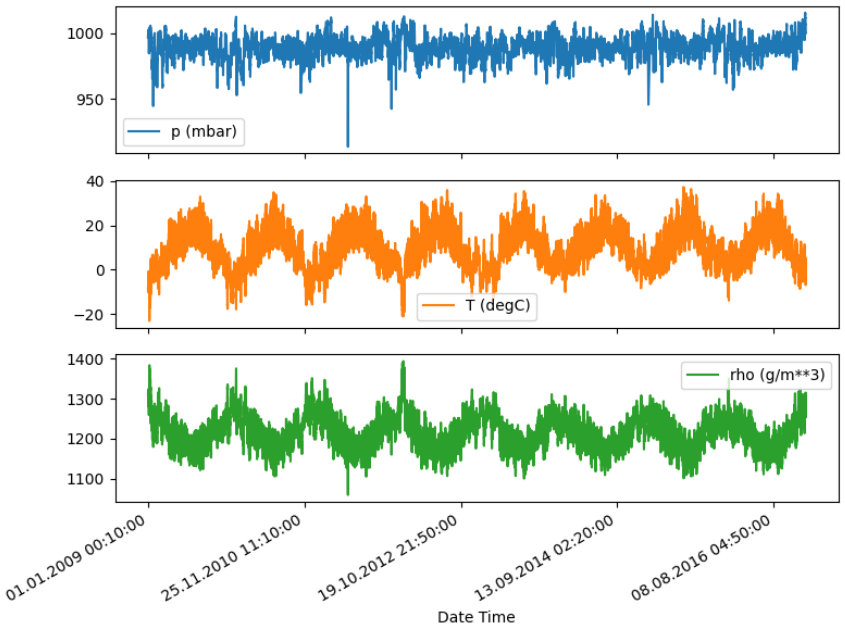

Abbildung 10.4: Die Features, die von der KI berücksichtigt werden.

Ein essentieller Schritt stellt die Normalisierung der Daten dar, indem von jedem Datenpunkt der Mittelwert subtrahiert wird und das Ergebnis durch die Standardabweichung dividiert wird (vgl. Z-Transformation). Dies kann ebenfalls mittels `pandas` übersichtlich durchgeführt werden:

```
dataset = features.values
data_mean = dataset[:TRAIN_SPLIT].mean(axis=0)
data_std = dataset[:TRAIN_SPLIT].std(axis=0)
dataset = (dataset - data_mean) / data_std
```

Danach werden einige Variablen definiert, um den weiteren Verlauf zu gliedern.

```
""" Definiert die Anzahl der Datenpunkte, aus denen
    eine Vorhersage gebildet wird (720 entsprechen 5 Tagen) """
past_history = 720
""" Definiert die Anzahl der Datenpunkte, die die KI vorhersagen
    muss (72 entspricht 12 Stunden). """
future_target = 72
""" Damit wird nur jeder 6. Datenpunkt genommen, um die Menge der
    Daten wieder zu verringern. """
STEP = 6
""" Definiert die Größe eines Datensatzes. """
BATCH_SIZE = 256
""" Definiert wie viele Datensätze in den Speicher
    geladen werden. """
```

```
BUFFER_SIZE = 10000
""" Definiert, wie viele Datensätze in einem Lernzyklus
    berechnet werden. """
EVALUATION_INTERVAL = 200
""" Definiert, die Anzahl der Lernzyklen. """
EPOCHS = 10
""" Definiert, wie viele Datenpunkte Training Daten sein
    sollten (sollte bei 3/4 liegen). """
TRAIN_SPLIT = 300000
```

Zu guter Letzt ist das gewählte Modell ausschlaggebend, da darüber entschieden wird, wie effizient die KI trainiert.

```
multi_step_model = tf.keras.models.Sequential()
multi_step_model.add(tf.keras.layers.LSTM(
    32,
    return_sequences=True,
    input_shape=x_train_multi.shape[-2:]
))
multi_step_model.add(tf.keras.layers.LSTM(16, activation='relu'))
multi_step_model.add(tf.keras.layers.Dense(72))
multi_step_model.compile(
    optimizer=tf.keras.optimizers.RMSprop(clipvalue=1.0),
    loss='mae'
)
```

In dem Tutorial wurde ein sequentielles Modell aus drei verarbeitenden Layern mit der Keras Bibliothek erstellt. Unter einem sequentiellen Modell versteht man, dass jedes Neuron in jedem Layer mit allen Neuronen der vorherigen und nachfolgenden Layern verbunden ist:

1. Der erste Layer beinhaltet die rohen Inputdaten und ist gleichzeitig ein Long short-term memory (LSTM) Layer, der aus dem neuronalen Netzwerk mittels einer Art eingebautem Gedächtnis, ein Time Series Forecasting Modell macht. Dieser Layer basiert auf der tanh Aktivierungsfunktion, siehe Kapitel 10.4.3.3 auf Seite 100.

2. Der zweite Layer ist ein LSTM Layer mit einer relu Aktivierungsfunktion, siehe Kapitel 10.4.3.4 auf Seite 100.

3. Der dritte Layer ist ein Dense Layer mit 72 Neuronen, die die Vorhersage des Netzwerkes darstellen.

Ein weiterer wichtiger Faktor ist der Optimizer, siehe Kapitel 10.4.5 auf Seite 101. In diesem Fall wurde der RMSprop Optimizer verwendet.

Mit diesem Modell kann eine Vorhersage über das Wetter getätigt werden:

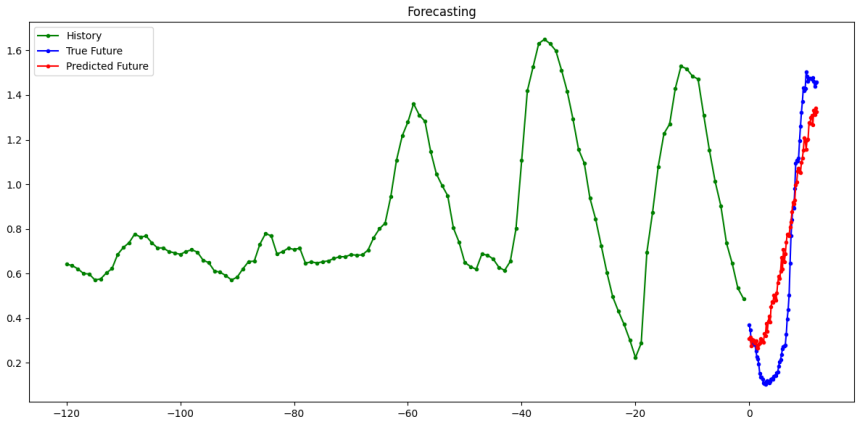

Abbildung 10.5: Das Ergebnis des Multiseries Forecasting Tutorials

10.9.2.4 Multiseries Forecasting mit eigenen Daten

Das Modell wurde mit Echtdaten aus der Loganalyse bespielt, dazu wurden die Summe aller Bytes, die auf der Cisco ASA im Zeitraum von zwei Monaten im Beispiel Netzwerk, siehe Kapitel 4 auf Seite 13 über TCP und UDP Verbindungen verbraucht worden sind, eingespeist. Diese Daten wurden in Blöcken von jeweils fünf Minuten zusammengefasst, was zu über 20.000 Datenpunkten führte. Aufgrund dieser Datenmenge sind die einzelnen Datenpunkte in der Abbildung 10.6 nicht mehr voneinander unterscheidbar.

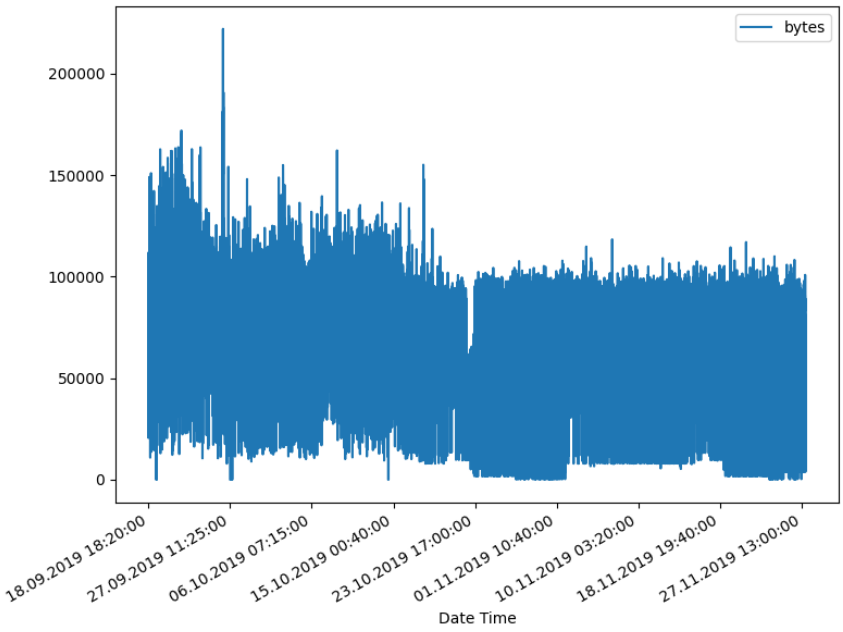

Abbildung 10.6: Die Anzahl der Bytes, die in zwei Monaten angefallen sind.

Da am Tag, speziell zu Unterrichtszeiten, mehr Daten verbraucht werden als in der Nacht, ist die Anzahl der Bytes stark von der Uhrzeit abhängig. Auch an den Wochenenden sinkt der Datenverbrauch. Ausgehend von diesen Erkenntnissen wurde das neuronale Netzwerk mit den Inputs Datum und Uhrzeit versorgt, um die Summe der Bytes vorherzusagen. Im Zuge dieser Vorgehensweise ergeben sich mehrere Inputfeatures, die man zur Vorhersage benutzen kann.

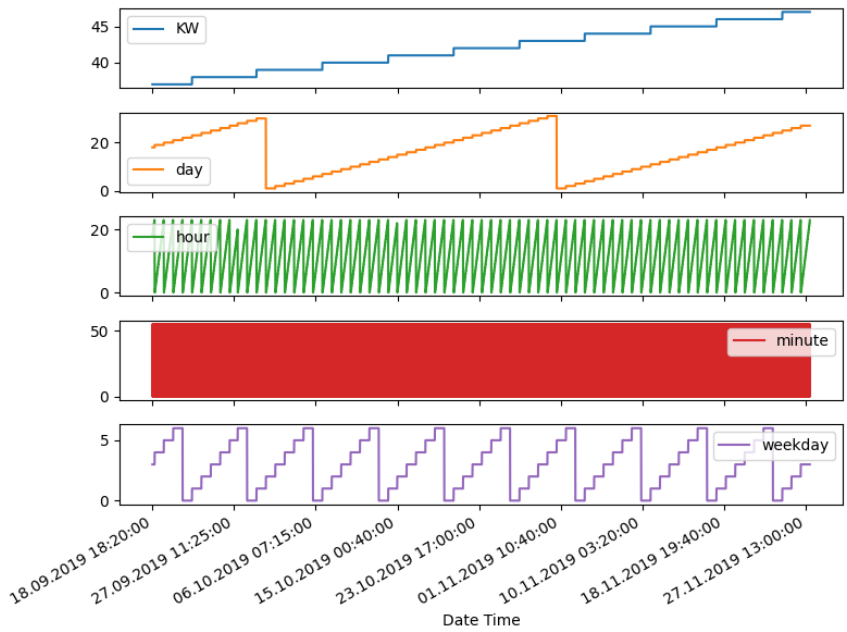

Abbildung 10.7: Features, die von der KI beachtet werden, um die Summe der Bytes vorauszusagen.

Die KI konnte anhand dieses Modelles den Trend richtig erkennen, scheiterte jedoch an der Erkennung etwaiger Ausreißer, um genaue Vorhersagen treffen zu können. In Abbildung 10.8 auf der nächsten Seite kann man in grün den Verlauf, von fünf zufällig aus dem Datenset ausgewählten Tagen, sehen. Die KI sagt dann drei weitere Tage, anhand des Verlaufs der Vergangenheit, voraus. Dies lässt sich im Bild in rot erkennen. Man erkennt, dass die KI die Wirklichkeit richtig abbilden konnte, auch wenn die Ausreißer nicht erkannt wurden. Eine mögliche Erklärung für dieses Verhalten ist, dass die KI zwar eine Vorhersage erstellt, doch aufgrund der relativ konstanten Werte gelernt hat, dass sie dort am wenigsten Fehler kreiert, auch wenn sie damit in Kauf nimmt, die Ausreißer nicht zu erkennen.

Betrachtet man die Echtdaten ohne Ausreißer, so unterliegen diese Schwankungen, welche einem Rauschen ähneln. Aufgrund der Schwankungen in den Echtdaten, weist die Prognose der KI ebenfalls Schwankungen auf, da sie diese nachahmt.

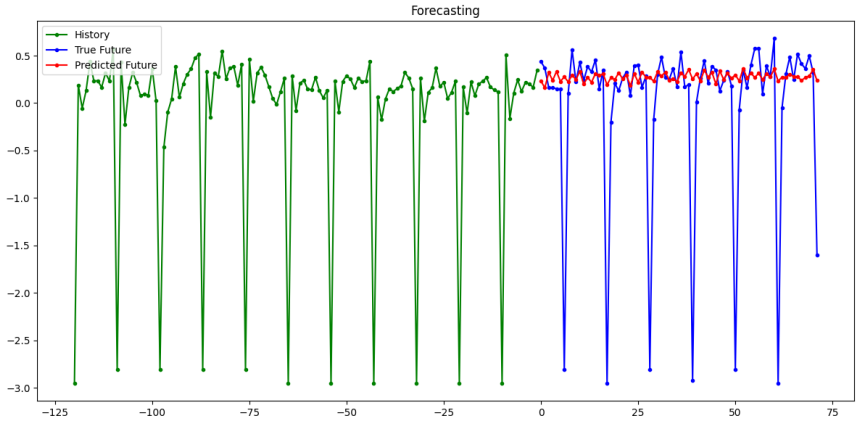

Abbildung 10.8: Die Voraussage der KI, im Vergleich zu den Echtdaten.

10.9.3 Time Series Forecasting

10.9.3.1 Time Series Forecasting in Aktion

Auch wenn das Multi Series Forecasting bereits eine gute Prognose liefern konnte, wurde das Modell überarbeitet und zusätzlich ein anderes Tutorial eingepflegt, welches speziell auf die Prognose eines einzelnen Features zugeschnitten wurde: [17]

```
model = keras.Sequential()
model.add(keras.layers.LSTM(
    units=128,
    input_shape=(X_train.shape[1], X_train.shape[2])
))
model.add(keras.layers.Dense(units=1))
model.compile(
    loss='mean_squared_error',
    optimizer=keras.optimizers.Adam(0.001)
)
```

Die Veränderungen zum vorherigen Beispiel sind, dass der Output-Layer nur ein Neuron hat und nicht alle Outputs auf einmal berechnet werden, sondern Schritt für Schritt. Dies macht sowohl ergebnistechnisch als auch performancemäßig wenig Unterschied und ist eher eine Designfrage. Zusätzlich wurde ein LSTM-Layer entfernt, und als Optimizer wurde der Adam Optimizer verwendet, siehe Kapitel 10.4.5 auf Seite 101. Als Input wurde eine Sinusfunktion gewählt, der zufällige Werte hinzugefügt wurden.

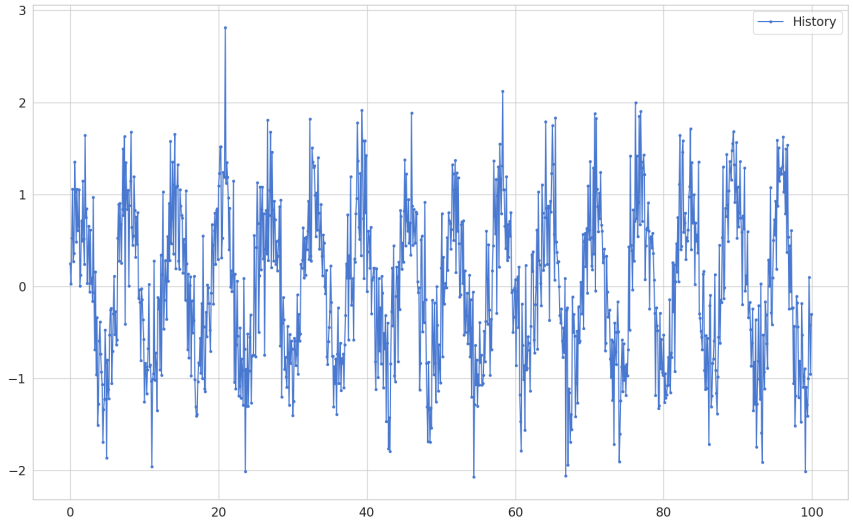

Abbildung 10.9: Inputdaten des Time Series Forecasting Tutorials als Form einer Sinusfunktion mit zufälligen Zusätzen.

Mit diesem Modell erzielte die KI ein deutlich besseres Ergebnis in der Vorhersage der Daten. Dabei hat sie die zugrundeliegende Sinusfunktion gut erkannt, ohne auf die Zufallsveränderung einzugehen. Damit handelt es sich hier um ein gutes Beispiel für ein Time Series Forecasting Modell, um Byteströme vorherzusagen.

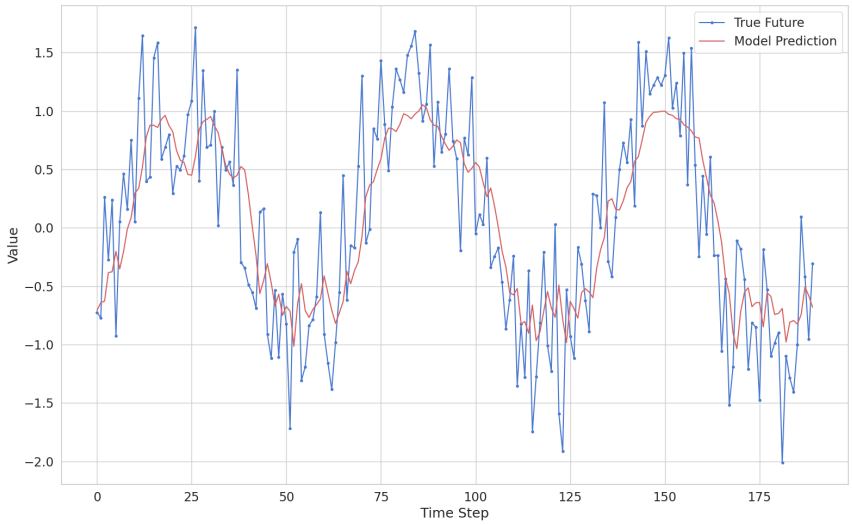

Abbildung 10.10: Ergebnis des Tutorials der Vorhersage der Sinusfunktion.

10.9.3.2 Time Series Forecasting mit periodischen Echtdaten

Für den Versuch, Byteströme mit diesem Modell vorherzusagen, wurde zuerst ein kleinerer Datensatz verwendet, um möglichst genau dem Tutorial mit einem Datensatz aus 1.000 Punkten zu entsprechen.

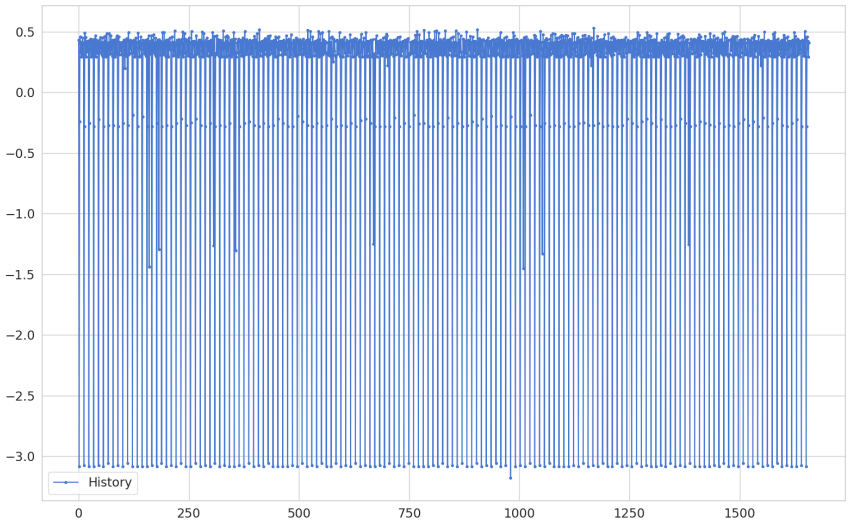

Abbildung 10.11: Ein Bytestrom, der stark periodisch ist, um die KI darauf zu trainieren.

Dazu wurde ein extra dafür, in einem Zeitraum von fünf Tagen, aufgenommener Datensatz verwendet. Dieser ist periodisch und somit für die KI leicht erlernbar.

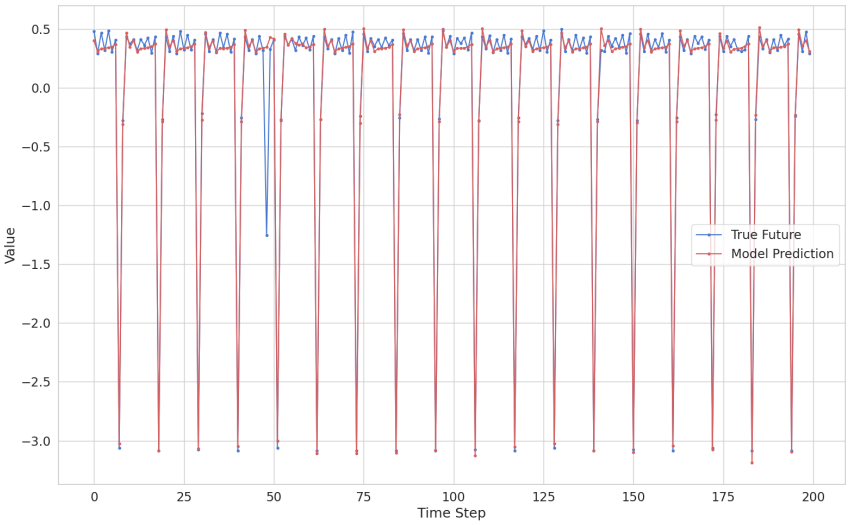

Abbildung 10.12: Das Ergebnis der KI, um den periodischen Bytestrom vorherzusagen.

In der Abbildung sieht man, dass die KI sehr gut auf das periodische Verhalten der Testdaten eingehen kann. Beim ungefähr 50. Datenpunkt kann man erkennen, dass ein Einbruch der Übertragung zu einer „Fehlvorhersage“ führt, was für ein positives Verhalten der KI spricht. Einmalige Ausreißer in den Daten führen nicht direkt zu einer Anpassung der Vorhersage und somit nicht zu einer Überkorrektur.

10.9.3.3 Time Series Forecasting mit Echtdaten

Nach erfolgreicher Testphase mit den extra angefertigten Testdaten, kann das neuronale Netzwerk mit Echtdaten trainiert werden. Dazu wurden die Testdaten aus dem Multiseries Forecasting Beispiel verwendet, siehe Kapitel 10.9.2.4 auf Seite 110. Diese Daten weisen in weiten Teilen ebenfalls periodische Verläufe auf, sind aber stark tageszeit-, wochentags- und wochenabhängig.

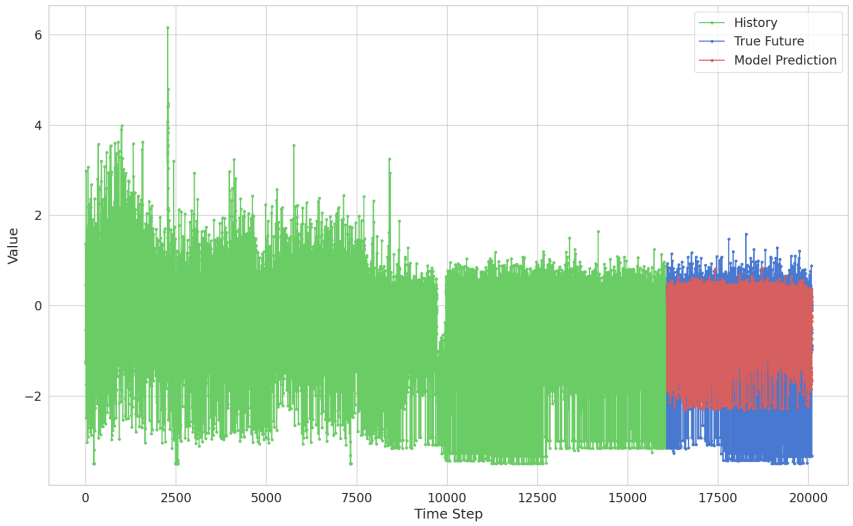

Abbildung 10.13: Die Prognose der KI, wenn sie Echtdaten vorhersagen muss.

Die Abbildung 10.13 zeigt einen groben Überblick über die Vorhersage der KI. Die historischen Daten sind wie auch schon in Abbildung 10.8 auf Seite 112 in grün, die wahre Zukunft in blau und die Prognose der KI in rot. Man erkennt hier, dass die KI nicht die genauen Werte vorhersagt. Die Prognose der KI sieht bereits sehr vielversprechend aus, auch wenn das neuronale Netzwerk Probleme hat, Ausreißer zu erkennen, ersichtlich an den roten Werten verglichen mit den blauen. Erst bei einer genaueren Betrachtung lassen sich exaktere Analysen erstellen.

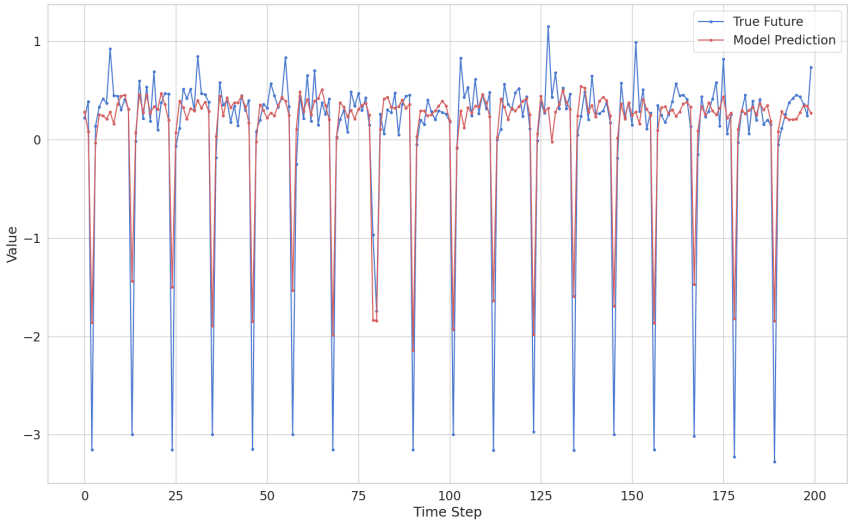

Abbildung 10.14: Vergrößerung des Ergebnisses, um genauere Analysen durchführen zu können.

In der genaueren Analyse kann man das vorausgesagte Verhalten betrachten. Die KI kann aufgrund der Historie zwar nicht gut mit Ausreißern umgehen, dennoch kann sie diese vorhersagen. Beim ungefähr 80. Datenpunkt sagt die KI zufälligerweise einen, wenn auch schwächeren, Ausreißer relativ genau voraus. Die daraus resultierende präzisere Vorhersage ist jedoch in diesem Fall unerwünscht, da die KI anhand von allgemeinen Daten und nicht anhand von Ausreißern lernen soll.

10.9.4 Ergebnis

10.9.4.1 Zusammenfassung Künstliche Intelligenz

Ausgehend von der anfänglichen Zielsetzung, ein Deep Learning Modell für die Voraussage des Netzwerkverkehrs einzusetzen, konnte in der Diplomarbeit gezeigt werden, dass, ein Modell entwickelt werden kann, das Byteströme vorhersagt. Anhand dieses Modelles konnte ebenfalls gezeigt werden, dass Ausreißer in den Daten zwar zu kurzfristigen Anpassungen der KI führen können, diese jedoch für die allgemeine Genauigkeit der Prognosen kaum negative Langzeiteffekte haben. Auch wenn das Modell, um in einem Analysetool in der Praxis eingesetzt werden zu können, noch weiterer Verbesserungen bedürfen würde, konnte grundsätzlich gezeigt werden, dass künstliche Intelligenz in der Lage ist, solche Prognosen anzustellen.

10.9.4.2 Multiseries Forecasting ungeeignet

Inputparameter, wie beispielsweise die Zeit, stellten keine guten Prädiktoren für die Byteströme dar. Dies begründet sich damit, dass die Byteströme weitgehend unabhängig von diesen Inputparametern waren. Der Ansatz wurde verworfen, da Byteströme hauptsächlich von vergangenen Byteströmen abhängen.

10.9.4.3 Timeseries Forecasting als optimale Lösung

Aussagekräftige Prognosen konnten bei Datensätzen, die die Summe der Bytes im 10-minuten Takt wiedergeben, erzielt werden. Das funktionierende Timeseries Forecasting Modell basiert auf vier Layern und benutzt Long short-term memory (LSTM) Neuronen, die ein Gedächtnis nachahmen.

10.10 Fazit

10.10.1 Künstliche Intelligenz in der Netzwerktechnik

KI ist ein wichtiges Werkzeug zur Verarbeitung von großen Mengen an Daten. In der Netzwerktechnik gewinnt die KI mehr an Bedeutung und so ist es wichtig, die Grundlagen dieser Technologie zu verstehen. Firmen wie Cisco oder Fortigate entwickeln SIEM Lösungen, die aufgrund der für den Menschen nicht mehr fassbaren Komplexität nur mehr mit künstlicher Intelligenz unter Kontrolle gebracht werden können.

10.10.2 Unterschied zwischen Theorie und Praxis

Das Problem dieser Technologie ist, dass sie nicht mehr greifbar ist und ein großen Unterschied zwischen Theorie und Praxis besteht. In der Theorie kann zwar geklärt werden, wie einfache KIs funktionieren, jedoch haben diese mit Deep Learning wenig zu tun, da sie zwar auf das selbe Grundgerüst aufbauen, dennoch auf viel komplexeren mathematischen Funktionen und Algorithmen beruhen. Trotz der umfangreichen Ausarbeitung dieses Themas bleiben viele Aspekte künstlicher Intelligenzen ungeklärt, wobei diese für die Anwendung nur bedingt wichtig sind. Das Verständnis dieser kann die KI, wenn auch nur bedingt, verbessern.

10.10.3 Modell der Diplomarbeit

In dieser Diplomarbeit kann man einen funktionierenden Trend der KI erkennen, jedoch bleibt fraglich, ob diese KI in Kombination mit den Inputdaten bessere Ergebnisse erzielen könnte. Zum einen Bedarf es komplexerer neuronaler Netze, um bessere Ergebnisse zu erzielen und zum anderen bleibt es debatierbar, ob Byteströme ideal zum Vorhersagen sind, da sie, wie in vorherigen Kapiteln erklärt nicht wirklich von Inputparametern abhängen, wovon Prognosen anderer Werte, wie das Wetter profitieren. Ein weiterer Aspekt der gegen die Verwendung von Bytesströmen spricht ist, dass sie sprunghaften Veränderungen unterliegen. Fraglich bleibt, ob die Verarbeitung der Zeit ideal war, denn man hätte die Stunde und die Minute nicht voneinander trennen müssen oder ob die KI etwas aus der Kalenderwoche hätte lernen können bei einem Zeitraum von ungefähr zwölf Wochen, die sich damit nicht wiederholen.

10.10.4 Rechenleistung und Dauer

Um eine KI auf Industriestandard zu trainieren, die exakte Prognosen treffen kann, bedarf es Rechenleistungen, die die Kapazitäten eines handelsüblichen PCs überschreiten. Da für diese Diplomarbeit die dafür benötigte Hardware nicht zur Verfügung stand, mussten Abstriche gemacht werden. Die vorhandenen Mittel führten dazu, dass die KI bereits mit vier Layern mindestens zehn Minuten pro Trainingsvorgang brauchte. Um komplexere KIs zu trainieren, bräuchte man entweder mehr Rechenleistung oder mehr Zeit pro Trainingsdurchlauf.

10.10.5 Datenmenge und Inputdaten

Für die in dieser Diplomarbeit genutzte KI waren die Anzahl an Datenpunkten ausreichend, um ein aussagekräftiges Ergebnis zu erzielen. Das Hinzufügen weiterer Daten ergab, aufgrund der Komplexität des neuronalen Netzwerks, längere Trainingszeiten ohne Verbesserung der Prognose.

10.10.6 Clustering

Eine auf diese Diplomarbeit aufbauende Arbeit könnte sich mit der Anwendung von Unsupervised Learning beschäftigen. Es wäre in diesem Zusammenhang von Interesse, zu erforschen, welche Zusammenhänge eine KI finden würde, wenn Verbindungen zwischen Usern und dem Internet als Input genutzt werden würden. In weiterführenden Projekten könnte dieser Zusammenhang ermittelt werden, wobei fraglich bleibt, ob die genutzten Inputdaten die richtigen für diesen Anwendungsfall wären.

11 Datenschutz

11.1 Datenschutz in Österreich

Seit dem 25. Mai 2018 ist die Datenschutz-Grundverordnung (im kompletten Titel: Verordnung (EU) 2016/679 des Europäischen Parlaments und des Rates vom 27. April 2016 zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten, zum freien Datenverkehr und zur Aufhebung der Richtlinie 95/46/EG) die Grundlage des Datenschutzrechts in der EU und damit auch in Österreich. Daher ist es auch wichtig, dass im Laufe dieser Diplomarbeit darauf geachtet wurde, das in Österreich allgemein geltende Datenschutzrecht zu beachten. Dies ist vor allem von Bedeutung, da es Teil der Diplomarbeit ist, personenbezogene Daten in vielen verschiedenen Arten zu verarbeiten. Aus diesem Grund ist der Schutz dieser Daten ein wichtiger Bestandteil der Diplomarbeit.

In Österreich gelten allgemein zwei verschiedene Gesetze, die in Bezug auf den Datenschutz für eine Diplomarbeit wie diese relevant sind:

- Datenschutzgesetz (DSG)
- Datenschutz-Grundverordnung (DSGVO)

Das Datenschutzgesetz (DSG), BGBl. I Nr. 165/1999 idgF., ist grundsätzlich das geltende österreichische Datenschutzgesetz. Dieses dient als Ergänzung der Datenschutz-Grundverordnung (DSGVO) in den Punkten, in denen diese es den einzelnen EU-Staaten freistellt, geringfügige Ergänzungen vorzunehmen. Beim DSG ist es wichtig zu beachten, dass sich dieses im Zuge des Datenschutz-Anpassungsgesetz 2018, BGBl. I Nr. 120/2017, sehr stark verändert hat und offiziell nun auch nicht mehr *Datenschutzgesetz 2000 (DSG 2000)*, sondern *Datenschutzgesetz (DSG)* heißt. Durch diese umfangreichen Änderungen sind so gut wie alle Verweise auf das alte DSG 2000 nicht mehr gültig.

Mit Hilfe der DSGVO, einer Verordnung der Europäischen Union, wird einheitlich geregelt, wie personenbezogene Daten innerhalb der Europäischen Union verarbeitet werden dürfen. Es soll vor allem der Schutz der personenbezogenen Daten in den Vordergrund gerückt und sichergestellt werden, so dass dieser ordnungsgemäß durchgeführt wird. Weiters soll aber auch garantiert werden, dass der freie Datenverkehr in der Europäische Union (EU) weiterhin stattfinden kann. Durch den Erlass dieser Verordnung wurde die *Richtlinie 95/46/EG (Datenschutzrichtlinie)* ersetzt.

Eine weitere wichtige Richtlinie, die in Bezug auf Datenschutz genannt werden sollte, ist die *Datenschutz-Richtlinie für den Bereich Justiz und Inneres (EU) 2016/680*, die ebenfalls im DSG geregelt wurde. Sie betrifft diese Diplomarbeit aber nicht direkt und wird daher nicht weiter erwähnt.

11.2 Grundsätze der DSGVO

Grundsätzlich ist jeder von der DSGVO betroffen, der in irgendeiner Art personenbezogene Daten verarbeitet. Hierfür gibt es keine Ausnahmen, es gilt also immer darauf zu achten, dass personenbezogene Daten richtig gehandhabt werden. Dazu hat die EU sechs Grundsätze definiert, die es immer zu beachten gilt:

Rechtmäßigkeit
Die personenbezogenen Daten müssen auf eine rechtmäßige Weise, in einer für die betroffene Person nachvollziehbaren Weise verarbeitet werden.

Zweckbindung
Die personenbezogenen Daten dürfen nur für den Zweck verwendet werden, für den sie auch erhoben wurden. Sie dürfen nicht in einer Weise weiterverarbeitet werden, die nicht mit dem Grund der originalen Erhebung übereinstimmt, außer es handelt sich um im Interesse der Öffentlichkeit liegende Archivzwecke, wissenschaftliche oder historische Forschungszwecke oder für statistische Zwecke.

Datenminimierung
Die personenbezogenen Daten müssen immer auf das notwendigste Maß beschränkt werden, so dass das Ausmaß an Daten dem Zweck immer noch genügt.

Richtigkeit
Die personenbezogenen Daten müssen immer sachlich korrekt und auf dem neusten Stand gehalten werden. Es müssen weiters alle möglichen Maßnahmen getroffen werden, um unrichtige Daten zu berichtigen.

Speicherbegrenzung
Die personenbezogenen Daten müssen in einer Form gespeichert werden, die es nur so lange erlaubt die betroffene Person zu identifizieren, wie es unbedingt für den Zweck der Verarbeitung notwendig ist.

Integrität und Vertraulichkeit
Die personenbezogenen Daten müssen so verarbeitet werden, dass die Sicherheit dieser Daten immer gewährleistet ist. Dies inkludiert den Schutz vor jeglichen unbefugten, unrechtmäßigen oder unbeabsichtigten Veränderungen der personenbezogenen Daten.

Der Begriff “Verarbeitung” in Bezug auf personenbezogene Daten in der DSGVO findet sehr breite Anwendung. Er bezieht sich auf jeden Vorgang, egal ob mit oder ohne Hilfe von automatisierten Verfahren, der sich mit personenbezogenen Daten befasst. Dies kann das Erheben, das Erfassen, die Organisation, das Speichern, das Übermitteln oder auch das Löschen der Daten betreffen.

11.3 Vorgehensweise in der Diplomarbeit

In der Diplomarbeit Argos wurden die in Kapitel 11.2 auf der vorherigen Seite erwähnten Grundsätze nach den besten Möglichkeiten des Teams eingehalten:

- Personenbezogene Daten werden nur erhoben, nachdem die betroffene Person informiert wurde, dass Daten gesammelt werden und wofür diese verwendet werden.
- Die gesammelten Daten werden nur für die Auswertung im Rahmen der Diplomarbeit Argos verwendet und sonst nicht weitergegeben.
- Es werden nur die Log-Nachrichten gesammelt, die für die im Rahmen der Diplomarbeit Argos durchgeführten Auswertungen notwendig sind.
- Es werden nur die Log-Nachrichten auf den Netzwerkgeräten konfiguriert, die für die Auswertung interessant sind.
- Daten, die nicht benötigt werden, werden erst gar nicht an den Log-Server gesendet.
- Die gespeicherten Daten werden nur so lange aufbehalten, wie sie für die Auswertung wirklich benötigt werden und sonst gelöscht.
- Die Vertraulichkeit der personenbezogenen Daten wird durch den sogenannten *Anonymizer* (siehe Kapitel 8.6 auf Seite 85) gewährleistet. Dieser führt eine Anonymisierung bzw. Pseudonymisierung der wichtigen personenbezogenen Daten durch. Dadurch ist kein Rückschluss auf die betroffene Person möglich.

Abbildungsverzeichnis

2.1 Modul-Architektur bei Argos 5

4.1 Der schematisch logische Netzwerkaufbau eines modernen Netzwerkes . . 13
4.2 Der Aufbau des Netzwerkes bei Argos 14

5.1 Auszug des SNMP MIB Baums 19
5.2 Konfiguration von SNMP-Traps auf einem Cisco WLC 25
5.3 Konfiguration von Syslog auf einem Cisco WLC 25
5.4 Der Ablauf von beiden verschiedenen Weiterleitungen für Syslog 29
5.5 Die Oberfläche für die Log-Weiterleitung 31

6.1 Der Aufbau für DNS Doctoring auf der Cisco ASA 36
6.2 Die Funktionsweise von SSL-Inspection 37
6.3 Der schematisch logische Netzaufbau des Netzes bei Argos 38
6.4 Der Aufbau eines DNS Headers 40

7.1 Veranschaulichung eines Indizes in MongoDB [13] 45

8.1 Die Variablenvererbung in Lark 70
8.2 Visualisierung der Datenreduktion durch Message Compression 78

9.1 Die Landing Page des Dashboards 90
9.2 Die Query Language im Dashboard 91
9.3 Ein Überblick über die letzten angemeldeten Benutzer 92
9.4 Genauere Informationen zu einem Benutzer 93
9.5 Übersicht aller implementierten Incidents 94
9.6 Der DNS-Tunnel Incident 95
9.7 Der Portscan Incident 96

10.1 Bild eines Neurons [1] 99
10.2 Verschiedene Aktivierungsfunktionen zum Vergleich. 100
10.3 Die Inputdaten des Multiseries Forecasting Tutorials 107
10.4 Die Features, die von der KI berücksichtigt werden. 108
10.5 Das Ergebnis des Multiseries Forecasting Tutorials 110
10.6 Die Anzahl der Bytes, die in zwei Monaten angefallen sind. 110
10.7 Features, die von der KI beachtet werden, um die Summe der Bytes vorauszusagen. 111
10.8 Die Voraussage der KI, im Vergleich zu den Echtdaten. 112

10.9 Inputdaten des Time Series Forecasting Tutorials als Form einer Sinusfunktion mit zufälligen Zusätzen. 113
10.10 Ergebnis des Tutorials der Vorhersage der Sinusfunktion. 113
10.11 Ein Bytestrom, der stark periodisch ist, um die KI darauf zu trainieren. . 114
10.12 Das Ergebnis der KI, um den periodischen Bytestrom vorherzusagen. . . . 114
10.13 Die Prognose der KI, wenn sie Echtdaten vorhersagen muss. 115
10.14 Vergrößerung des Ergebnisses, um genauere Analysen durchführen zu können. 116

Abkürzungsverzeichnis

ACL Access Control List. 33

AD Active Directory. 27, 28, 47

Adam Adaptive Moment Estimation. *Glossar:* Adaptive Moment Estimation

AMP Advanced Malware Protection. 33

AP Access Point. 22, 24, 48, 73

API Application Programming Interface. 78, 79, 87, 89

AQL *Argos-Query-Language*. 88

ASA Adaptive Security Appliance. 2, 15, 23, 24, 32–34, 59, 60, 63–66, 69, 80–82, 84

ASCII American Standard Code for Information Interchange. 19, 20, 84

ASN.1 Abstract Syntax Notation One. 19

AWS Amazon Web Services. 9

CIM Common Information Model. 9

CLI Command Line Interface. 23, 25

CMDB Configuration Management Database. 11

COBIT Control Objectives for Information and Related Technologies. 7

CSV Comma-Separated Values. 84

DB Datenbank. 3, 4, 6, 41–44, 55, 60, 70, 87–89

DBMS Datenbankmanagementsystem. 41, 42

DNS Domain Name System. 6, 33, 34, 37–39, 44, 49, 51, 52, 54, 74, 76, 77, 80

DoS Denial of Service. 81

DPI Deep Packet Inspection. 2, 3, 31, 32, 35–38, 74, 77

DSGVO Datenschutz-Grundverordnung. 1, 2

EDV Elektronische Datenverarbeitung. 2

FQDN Fully-Qualified Domain Name. 33, 34, 38, 52, 54, 76, 78

GPO Group Policy Object. 29, 30

HTL Höhere Technische Lehranstalt. 1

HTML Hypertext Markup Language. 89

HTTP Hypertext Transfer Protocol. 11, 29, 89

HTTPS Hypertext Transfer Protocol Secure. 11, 29, 35

ID Identifikator. 38, 84, 85

IDS Intrusion Detection System. 7

IP Internet Protocol. 15, 24–26, 28, 30, 33, 34, 38, 41, 45, 46, 48, 49, 51, 54, 55, 58, 73, 74, 78–81, 83–85, 88

IPFIX IP Flow Information Export. 15

IPS Intrusion Prevention System. 7, 25, 60

ISO Internationale Organisation für Normung. 7

ISP Internet Service Provider. 13

IT Informationstechnologie. 1, 7, 9

ITIL IT Infrastructure Library. 7

JSON JavaScript Object Notation. 2, 9, 42, 58

KMU kleines und mittleres Unternehmen. iii

LALR Look-ahead LR Parser. 68, *Glossar:* Look-ahead LR Parser

LAN Local Area Network. 14

LR Links nach rechts, rechtsreduzierender Parser. 68

LSTM Long short-term memory. *Glossar:* Long short-term memory

LWAP Light Weight AP. 24

MAC Media Access Control. 2, 48, 49, 58, 73, 78, 84, 85, 88

MacOS Macintosh Operating System. 9

MIB Management Information Base. 19

MIT Massachusetts Institute of Technology. 89

MO Managed Object. 19

NGFW Next Generation Firewall. 32–34

NoSQL Not only SQL. 42

NPS Network Policy Server. 27

NT New Technology. 84

NTP Network Time Protocol. 57

OID Object Identifier. 19–21, 73

OSI Open Systems Interconnection. 31, 33, 34, 38, 74, 75

OUI Organizationally Unique Identifier. 78

PDU Protocol Data Unit. 31–34, 38, 74–76

PHP PHP: Hypertext Preprocessor. 3, 89

RAM Random Access Memory. 23

RFC Request for Comments. 16, 17, 19, 75

RMSProp Root Mean Square Propagation. *Glossar:* Root Mean Square Propagation

SEM Security Event Management. 7

SID Security Identifier. 30, 84

SIEM Security Information and Event Management. iii, v, ix, 2, 6–8, 10–12, 25, 41, *Glossar:* Security Information and Event Management

SIM Security Information Management. 7

SME small and medium-sized enterprise. v

SMTP Simple Mail Transfer Protocol. 23

SNMP Simple Network Monitoring Protocol. 6, 15, 16, 18–26, 59, 73

SOX Sarbanes-Oxley Act. 7

SPL Search Processing Language. 10

SSID Service Set Identifier. 48, 74, *Glossar:* Service Set Identifier

SSL Secure Socket Layer. 3, 17, 35

TCP Transmission Control Protocol. 16, 17, 44, 47, 67, 80, 83, 88

TLS Transport Layer Security. 3

TTL Time to Live. 52, 54

UDP User Datagram Protocol. 17, 20, 80

UI User Interface. 89

VLAN Virtual LAN. 14

VM Virtual Machine. 10

VPN Virtual Private Network. 32, 83, 84

VTY Virtual Terminal Line. 22

WLAN Wireless LAN. 1, 14, 15, 24, 25, 27, 47, 48, 74, 78, 82

WLC WLAN-Controller. 1, 15, 22, 24, 25, 59, 73

XML Extensible Markup Language. 9, 30

Glossar

Adaptive Moment Estimation Eine neuere Version des RMSProp Optimizers, der performanter ist und auch jedes Gewicht dynamisch adaptiert. *Abkürzungsverzeichnis:* Adam

Daemon bezeichnet unter Unix oder unixartigen Systemen ein Programm, das im Hintergrund abläuft und bestimmte Dienste zur Verfügung stellt. 44, 87

Index eine von der Datenstruktur getrennte Indexstruktur in einer Datenbank, die die Suche und das Sortieren nach bestimmten Feldern beschleunigt. 42, 43

Long short-term memory Long short-term memory (langes Kurzzeitgedächtnis) ist eine Technik, die dafür sorgt, dass Neurale Netzwerke ein Gedächtnis haben. *Abkürzungsverzeichnis:* LSTM

Look-ahead LR Parser Look-ahead LR Parser, wobei die Zahl in der Klammer die Anzahl der vorausschauenden Felder beschreibt. 68, *Abkürzungsverzeichnis:* LALR

MongoDB eine dokumentenorientierte NoSQL-Datenbank. https://www.mongodb.com/ 42–44

OSI-Modell ein Referenzmodell für Netzwerkprotokolle als Schichtenarchitektur. 31, 74, 75

OSI-Schicht Das OSI-Modell hat sieben Schichten: 1 – Bitübertragung (Physical), 2 – Sicherung (Data Link), 3 – Vermittlung-/Paket (Network), 4 – Transport (Transport), 5 – Sitzung (Session), 6 – Darstellung (Presentation), 7 – Anwendung (Application) 31, 33, 34, 38, 74, 75

Python eine universelle, üblicherweise interpretierte höhere Programmiersprache. https://www.python.org/ 5, 6, 42, 43, 57, 61, 75, 84, 88, 89

Root Mean Square Propagation Ein Optimizer, der die Lernrate für jedes Gewicht dynamisch adaptiert. *Abkürzungsverzeichnis:* RMSProp

Security Information and Event Management kombiniert die zwei Konzepte Security Information Management (SIM) und Security Event Management (SEM) für die Echtzeitanalyse von Sicherheitsalarmen aus den Quellen Anwendungen und Netzwerkkomponenten. iii, v, 2, 41, *Abkürzungsverzeichnis:* SIEM

Service Set Identifier ist ein frei wählbarer Name eines Service Sets, durch den es ansprechbar wird. Da diese Kennung oftmals manuell von einem Benutzer in Geräte eingegeben werden muss, ist sie oft eine Zeichenkette, die für Menschen leicht lesbar ist, und sie wird daher allgemein als (Funk-)Netzwerkname des WLANs bezeichnet. 48, 74, *Abkürzungsverzeichnis:* SSID

Socket ist ein vom Betriebssystem bereitgestelltes Objekt, das als Kommunikationsendpunkt dient. Ein Programm verwendet Sockets, um Daten mit anderen Programmen auszutauschen. 55

Thread bezeichnet einen Ausführungsstrang oder eine Ausführungsreihenfolge in der Abarbeitung eines Programms. Ein Thread ist Teil eines Prozesses. 55, 58

Literaturverzeichnis

[1] *Künstliches Neuron – Wikipedia.* https://de.wikipedia.org/wiki/K%C3%BCnstliches_Neuron, Abruf: 2020-04-02

[2] Bendel, Oliver: Chatbot. In: *Gabler Wirtschaftslexikon* (2018). https://wirtschaftslexikon.gabler.de/definition/chatbot-54248/version-277297

[3] Brook, Chris: Grundinformationen zu Deep Packet Inspection. Version: 2018. https://digitalguardian.com/blog/what-deep-packet-inspection-how-it-works-use-cases-dpi-and-more, Abruf: 2020-01-06. 2018. – Forschungsbericht

[4] Cisco: Cisco ASA Series Syslog Messages / Cisco Systems, Inc. Version: 2020. https://www.cisco.com/c/en/us/td/docs/security/asa/syslog/b_syslog.pdf, Abruf: 2020-04-02. 2020. – Forschungsbericht

[5] Erez, Shinan: *Lark - a modern parsing library for Python.* https://github.com/lark-parser/lark

[6] Gerhards, R.: RFC 5424 – The Syslog Protocol. Version: 2009. https://tools.ietf.org/html/rfc5424, Abruf: 2020-03-14. 2009. – Forschungsbericht

[7] Global OID reference database (Hrsg.): *Reference record for OID 1.3.6.1.4.1.9.* https://oidref.com/1.3.6.1.4.1.9: Global OID reference database

[8] Güttich, Götz: Firewallimplementationen von DPI. Version: 2017. https://www.security-insider.de/grundlagen-der-next-generation-firewall-ngfw-a-648098/, Abruf: 2020-01-06. 2017. – Forschungsbericht

[9] Lackes, R. ; Siepermann, M.: Künstliche Intelligenz (KI). In: *Gabler Wirtschaftslexikon* (2018). https://wirtschaftslexikon.gabler.de/definition/kuenstliche-intelligenz-ki-40285/version-263673

[10] Lämmel, U. ; Cleve, J.: *Künstliche Intelligenz.* Carl Hanser Verlag GmbH & Company KG, 2012 https://books.google.at/books?id=JKlPAgAAQBAJ. – ISBN 9783446428737

[11] Maier, Matthias: Log Management vs. Security Event Management. In: *Computerwoche* (2013). https://www.computerwoche.de/a/was-sim-und-sem-von-siem-unterscheidet

[12] Microsoft (Hrsg.): *Windows Event Forwarding*. https://docs.microsoft.com/en-us/windows/win32/wec/windows-event-collector: Microsoft, 2018

[13] MongoDB, Inc. (Hrsg.): *Indexes – MongoDB Manual*. https://docs.mongodb.com/manual/indexes/: MongoDB, Inc.

[14] pandas (Hrsg.): *pandas Homepage*. https://pandas.pydata.org/: pandas

[15] TensorFlow (Hrsg.): *Tensorflow Tutorials*. https://www.tensorflow.org/tutorials: TensorFlow

[16] TensorFlow (Hrsg.): *Time series forecasting*. https://www.tensorflow.org/tutorials/structured_data/time_series: TensorFlow

[17] Venelin, Valkov: Time Series Forecasting with LSTMs using TensorFlow 2 and Keras in Python. In: *towards data science* (2019). https://towardsdatascience.com/time-series-forecasting-with-lstms-using-tensorflow-2-and-keras-in-python-6ceee9c6c651