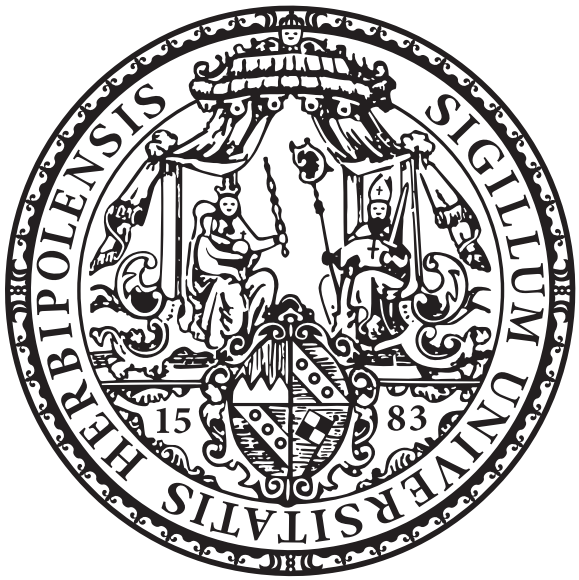
Detektion sicherheitskritischer Ereignisse in Unternehmensnetzwerken mittels Data Mining

vorgelegt von

Markus Ring

Würzburg, 2020



Dissertation zur Erlangung des naturwissenschaftlichen Doktorgrades
der Bayerischen Julius-Maximilians-Universität Würzburg

Abstract

E-mails, online banking and video conferences have become an integral part of our daily lives. All these processes transmit confidential data and personal information over insecure lines. There are many concepts, methods and procedures to protect digital data against unauthorised access and manipulation which can be summarised under the term IT security. Typical security mechanisms are firewalls and virus scanners. Such approaches are usually rule-based and check files or incoming network traffic against a list of known attack signatures. Consequently, these approaches can only detect known signatures and do not offer protection against zero-day exploits. Generally, there is a race between IT security experts and hackers in which hackers try to find new ways and methods to trick existing security solutions while IT security experts try to improve their security mechanisms.

This work aims at the detection of attack scenarios in company networks using data mining methods. Data mining methods are able to learn and generalise from representative training data. Consequently, these methods can be used to detect new attack scenarios if the new attack scenarios overlap with known attack scenarios or differ significantly from normal behaviour. This work focuses on the analysis of network-based data in NetFlow format, since this provides an aggregate view of what is going on in the network. Mostly, network-based data can not be shared due to privacy concerns which calls for the generation of synthetic, but realistic network data. Further, network-based data consists of continuous and categorical attributes which complicates their analysis, in particular comparing these data with respect to their (dis)similarity.

This work provides methodological contributions to each of the three mentioned challenges above. The developed methods ConDist and IP2Vec are two different approaches for distance calculation between categorical values. ConDist is a generally usable distance measure for calculating distances between objects with continuous and categorical attributes. IP2Vec is specialised on network-based data and transforms categorical values into semantic-preserving continuous vectors.

Further, this work provides an exhaustive overview about network-based data sets and proposes two new approaches for generating realistic network-based data. The first approach rebuilds company networks in a test environment and simulates typical user activities by automated Python scripts. The second approach is based on Generative Adversarial Networks and generates synthetic data. Generative Adversarial Networks learn the characteristics of network-based data and generate new data with the same underlying characteristics. While the first approach is able to create new data sets, the second approach can be used to enrich existing data sets with additional data.

Finally, this work provides two contributions to the detection of attack scenarios. The first contribution provides a general concept for attack detection, which is oriented towards the typical phases of attack scenarios. The second contribution proposes an unsupervised and a supervised method for detecting slow port scans with high accuracy.

Abstrakt

E-Mails, Online Banking und Videokonferenzen sind aus unserem heutigen Alltag nicht mehr wegzudenken. Bei all diesen Aktivitäten werden zahlreiche personenbezogene Informationen und vertrauenswürdige Daten digital übertragen und gespeichert. Zur Sicherstellung der digitalen Daten vor unbefugten Zugriffen und Manipulationen existieren verschiedenste Konzepte, Methoden und Verfahren, die sich unter dem Begriff IT-Sicherheit zusammenfassen lassen. Klassische Sicherheitslösungen aus dem Bereich IT-Sicherheit sind Firewalls und Virenscanner. Derartige Ansätze sind meist regelbasiert und prüfen Dateien beziehungsweise eingehenden Netzwerkverkehr anhand einer Liste bekannter Angriffssignaturen. Folglich können diese Systeme nur bereits bekannte Angriffsszenarien detektieren und bieten keinen Schutz vor neuartigen Angriffen. Somit entsteht im Bereich IT-Sicherheit ein Wettlauf zwischen Hackern und IT-Sicherheitsexperten, bei dem die Hacker stets nach neuen Mitteln und Wegen suchen, die existierenden Sicherheitslösungen zu überwinden, während IT-Sicherheitsexperten stetig ihre Schutzmechanismen verbessern.

Die vorliegende Arbeit widmet sich der Detektion von Angriffsszenarien in Unternehmensnetzwerken mithilfe von Data Mining-Methoden. Diese Methoden sind in der Lage anhand von repräsentativen Daten die darin enthaltenen Strukturen zu erlernen und zu generalisieren. Folglich können sich Data Mining-Methoden grundsätzlich zur Detektion neuer Angriffsszenarien eignen, wenn diese Angriffsszenarien Überschneidungen mit bekannten Angriffsszenarien aufweisen oder sich wesentlich vom bekannten Normalverhalten unterscheiden. In dieser Arbeit werden netzwerkbasierte Daten im NetFlow Format analysiert, da diese einen aggregierten Überblick über das Geschehen im Netzwerk bieten. Häufig können Netzwerkdaten aufgrund datenschutzrechtlicher Bedenken nicht veröffentlicht werden, was für die Erzeugung synthetischer, aber realistischer Netzwerkdaten spricht. Des Weiteren führt die Beschaffenheit der Netzwerkdaten dazu, dass eine Kombination von kontinuierlichen und kategorischen Attributen analysiert werden muss, was vor allem das Vergleichen der Daten bezüglich ihrer Ähnlichkeit erschwert.

Diese Arbeit liefert methodische Beiträge zu jeder der drei genannten Herausforderungen. Im Bereich der Abstandsberechnung kategorischer Werte werden mit ConDist und IP2Vec zwei unterschiedliche Ansätze entwickelt. ConDist ist ein universell einsetzbares Abstandsmaß zur Berechnung von Abständen zwischen Datenpunkten, die aus kontinuierlichen und kategorischen Attributen bestehen. IP2Vec ist auf Netzwerkdaten spezialisiert und transformiert kategorische Werte in kontinuierliche Vektoren.

Im Bereich der Generierung realistischer Netzwerkdaten werden neben einer ausführlichen Literaturrecherche zwei unterschiedliche Ansätze vorgestellt. Zunächst wird ein auf Simulation basierter Ansatz zur Generierung flowbasierter Datensätze entwickelt. Dieser Ansatz basiert auf einer Testumgebung und simuliert typische Benutzeraktivitäten durch automatisierte Python Skripte. Parallel hierzu wird ein zweiter Ansatz zur synthetischen Generierung flowbasierter Netzwerkdaten durch Modellierung mithilfe von Generative Adversarial Networks entwickelt. Dieser Ansatz erlernt die zugrundeliegenden Eigenschaften der Netzwerkdaten und ist anschließend in der Lage, neue Netzwerkdaten mit gleichen Eigenschaften zu generieren. Während sich der erste Ansatz zur Erstellung neuer Datensätze eignet, kann der zweite Ansatz zur Anreicherung existierender Datensätze genutzt werden.

Schließlich liefert diese Arbeit noch zwei Beiträge zur Detektion von Angriffsszenarien. Im ersten Beitrag wird ein Konzept zur Detektion von Angriffsszenarien entwickelt, welches sich an die typischen Phasen eines Angriffsszenarios orientiert. Im zweiten Beitrag werden eine überwachte und eine unüberwachte Methode zur Detektion von langsamen Port Scans vorgestellt.

Danksagung

Die Promotion war ein langer Weg, auf dem mich viele Personen unterstützt haben. Zuallererst möchte ich mich bei Prof. Dr. Andreas Hotho und Prof. Dr. Dieter Landes bedanken, die es mir ermöglicht haben, eine kooperative Promotion unter ihrer Betreuung durchzuführen. Vielen Dank für die hilfreichen und wegweisenden Diskussionen, für das mir entgegengebrachte Vertrauen und für die vielen spannenden Themen, die ich in den letzten Jahren bearbeiten durfte. Durch eure flexible und offene Art war es mir möglich, meine Tätigkeit als wissenschaftlicher Mitarbeiter und meine Promotion trotz verschiedener Standorte erfolgreich zu meistern und mich gleichzeitig persönlich und beruflich über die Promotion hinaus weiter zu entwickeln.

Ein großer Dank geht auch an meine Arbeitskollegen an der Hochschule Coburg und der Universität Würzburg. Ohne eure Unterstützung bei fachlichen und organisatorischen Problemen, Motivationshilfen und entspannten Kicker- und Kaffeepausen wäre der Arbeitsalltag nur schwer vorstellbar. Auch wenn die Arbeitsgruppe über die letzten Jahre zahlreich angewachsen ist, möchte ich dennoch jeden Einzelnen nennen: Martin Becker, Florian Buckermann, Alexander Dallmann, Padraig Davidson, Michael Ebert, Elisabeth Fischer, Pascal Förtsch, Dominik Grüdl, Lena Hettinger, Konstantin Kobs, Tobias Koopmann, Anna Krause, Florian Lautenschlager, Thomas Nieber, Janna Omeliyanenko, Florian Otto, Deniz Scheuring, Daniel Schlör, Michael Steininger, Julian Tritscher, Sebastian Wankerl, Maximilian Wolf, Sarah Wunderlich, Albin Zehe und Daniel Zoller.

Des Weiteren möchte ich mich bei Matthias Ring und Tobias Windisch bedanken, die während meiner Promotion nicht nur stets ein offenes Ohr für mich hatten, sondern auch für fachliche Diskussionen offen waren.

Eine Promotion geht auch mit einigen persönlichen Herausforderungen einher. Deshalb möchte ich mich bei meinen Eltern und bei meiner Frau Theresa für die Unterstützung, ihren Glauben an mich und den Rückhalt in all den Jahren vor und während meiner Promotion bedanken.

Vielen Dank an euch alle!

Markus Ring

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Liste der Algorithmen

Abkürzungsverzeichnis

B-WGAN-GP	Binar-based - Wasserstein Generative Adversarial Networks - Gradient Penalty
BG	Background
BWV	Bedingte Wahrscheinlichkeitsverteilung
CBDL	Context Based Distance Learning
CBOW	Continuous Bag of Words
CDbw	Compose Density between and within clusters
CIDDS	Coburg Intrusion Detection Data Sets
CMAR	Classification based on Multiple Association Rules
ConDist	Context-based Categorical Distance Measure
CPCQ	Contrast Pattern based Clustering Quality index
CRISP-DM	Cross-Industry Standard Process for Data Mining
CSV	Character/Comma Separated Values
CUF	Coburg-Utility-Framework
DBScan	Density-Based Spatial Clustering of Applications with Noise
dev	Developer-Subnetz
DILCA	DIstance Learning of Categorical Attributs
DDoS	Distributed Denial of Service
DNS	Domain Name System
DoS	Denial of Service
DPI	Deep Packet Inspection
E-WGAN-GP	Embedding-based - Wasserstein Generative Adversarial Networks - Gradient Penalty
FAIR	Findability, Accessibility, Interoperability, and Reusability
FN	False Negative
FP	False Positive
FTP	File Transfer Protocol
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IG	Informationsgewinn
IGMP	Internet Group Management Protocol
IPFIX	Internet Protocol Flow Information Export
IPS	Intrusion Prevention System
IRC	Internet Relay Chat

ISP	Internet Service Provider
IP	Internet Protocol
IT	Informationstechnologie
GAN	Generative Adversarial Network
GB	Gigabyte
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
KDD	Knowledge Discovery from Data
KMU	Kleine und mittlere Unternehmen
kNN	k-Nächster Nachbar
mgt	Management-Subnetz
N-WGAN-GP	Numeric-based - Wasserstein Generative Adversarial Networks - Gradient Penalty
NMI	Normalized Mutual Information
OF	Occurrence Frequency
off	Office-Subnetz
OPTICS	Ordering Points To Identify the Clustering Structure
OS	Operating System
PHP	PHP: Hypertext Preprocessor
RELU	Restricted Linear Unit
RUDY	R-U-Dead-Yet
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
SPICE	Stealthy Probing and Intrusion Correlation Engine
SPSD	Supervised Port Scan Detection
srv	Server-Subnetz
SSH	Secure Shell
SVM	Support Vector Machine
t-SNE	t-Distributed Stochastic Neighbor Embedding
TCP	Transmission Control Protocol
TN	True Negative
TP	True Positive
TRW	Threshold Random Walk
TTUR	Two time-scale update rule
UDP	User Datagram Protocol
UPSD	Unsuperivsed Port Scan Detection
WGAN-GP	Wasserstein Generative Adversarial Networks - Gradient Penalty
WGAN	Wasserstein Generative Adversarial Networks
WLAN	Wireless Local Area Network

1. Einleitung

1.1. Motivation

IT-Sicherheit ist eine essentielle Angelegenheit für Unternehmen. Sensible Unternehmensdaten stellen ein wertvolles Gut dar, welches vor unautorisierten Zugriffen und Manipulationen zu schützen ist [LOS+13]. Diebstahl von sensiblen Daten (beispielsweise Konstruktionsentwürfe oder Kundendaten) kann zum Verlust des Wissensvorsprungs gegenüber Konkurrenten und zu großen Imageschäden für Unternehmen führen. Beispielsweise resultierte der Datendiebstahl von mehr als 75 Millionen Kundendaten bei Sony in einem erheblichen Vertrauensverlust der Kunden [Myr11]. Die Kosten für diesen Hackerangriff beliefen sich auf schätzungsweise 171 Millionen Dollar [Hac11]. Ein anderes Beispiel ist der Hackerangriff auf Facebook im September 2018. Bei diesem Angriff stahlen die Hacker private Daten von ungefähr 30 Millionen Profilen [Bri18]. Mittlerweile stehen auch kleine und mittlere Unternehmen (KMU) im Fokus von Hackerangriffen.

Die zunehmende Digitalisierung von Prozessen, welche sich zum Beispiel in vernetzten Lieferketten, cloudbasierten Lösungen oder Industrie 4.0 offenbart, führt dazu, dass die zu schützenden Daten parallel zur potentiellen Angriffsfläche zunehmen. Diverse Berichte bestätigen das Wachstum von kriminellen Aktivitäten gegen IT-Infrastrukturen. Ciscos jährlicher Sicherheitsreport des Jahres 2018 [Cis18] berichtet beispielsweise von einem elffachen Anstieg von Malware, der an Sicherheitsprodukten der Firma Cisco im Zeitraum von Januar 2016 bis Oktober 2017 zu beobachten war. Das Jahr 2017 zeigte zudem, dass Sicherheitsbedrohungen auch aus neuen unerwarteten Quellen wie Kryptowährungen entstehen können [Sym18]. Laut Symantecs Internet Security Threat Report [Sym18] stieg die Anzahl erkannter Coin Miners im Jahr 2017 um 8500 Prozent sowie die Anzahl neuartiger Ransomware um 46 Prozent. Coin Miners stehlen Rechenleistung zur Berechnung von Kryptowährungen und Ransomware verschlüsselt Dateien auf Computern und fordert zur Zahlung von Lösegeld auf. Abbildung 1.1 enthält die Entwicklung von Angriffsvarianten aus dem McAfee Labs Threats Report [BCC+18].

Abbildung 1.1 zeigt (von oben nach unten) die Anzahl neuartiger Malware, Ransomware und Coin Miner Malware auf der linken Seite und deren Gesamtanzahl auf der rechten Seite zwischen dem dritten Quartal 2016 und dem zweiten Quartal 2018. Abbildung 1.1 und die Berichte [Cis18] und [Sym18] bestätigen somit die These, dass die Anzahl neuartiger Angriffsszenarien gegen IT-Systeme stetig ansteigt.

Unternehmen verwenden diverse Sicherheitssysteme zum Schutz ihrer Daten [RWG+17a]. Klassische Sicherheitssysteme wie Firewalls oder Virenscanner können jedoch in der Regel nur bereits bekannte Angriffsszenarien erkennen und verhindern [GPR+08]. Deshalb muss beispielsweise eine stetige Aktualisierung der Liste bekannter Signaturen bei Virenscannern erfolgen. Anomaliebasierte Sicherheitssysteme hingegen modellieren normales Benutzerverhalten mithilfe geeigneter Trainingsdaten. Abweichungen vom erlernten Verhalten stellen Auffälligkeiten dar, die einer genaueren Analyse bedürfen. Somit können anomaliebasierte Sicherheitssysteme auch neuartige, bisher unbekannte Angriffsszenarien entdecken und treten der Herausforderung der kontinuierlichen Entwicklung neuer Angriffsszenarien entgegen.

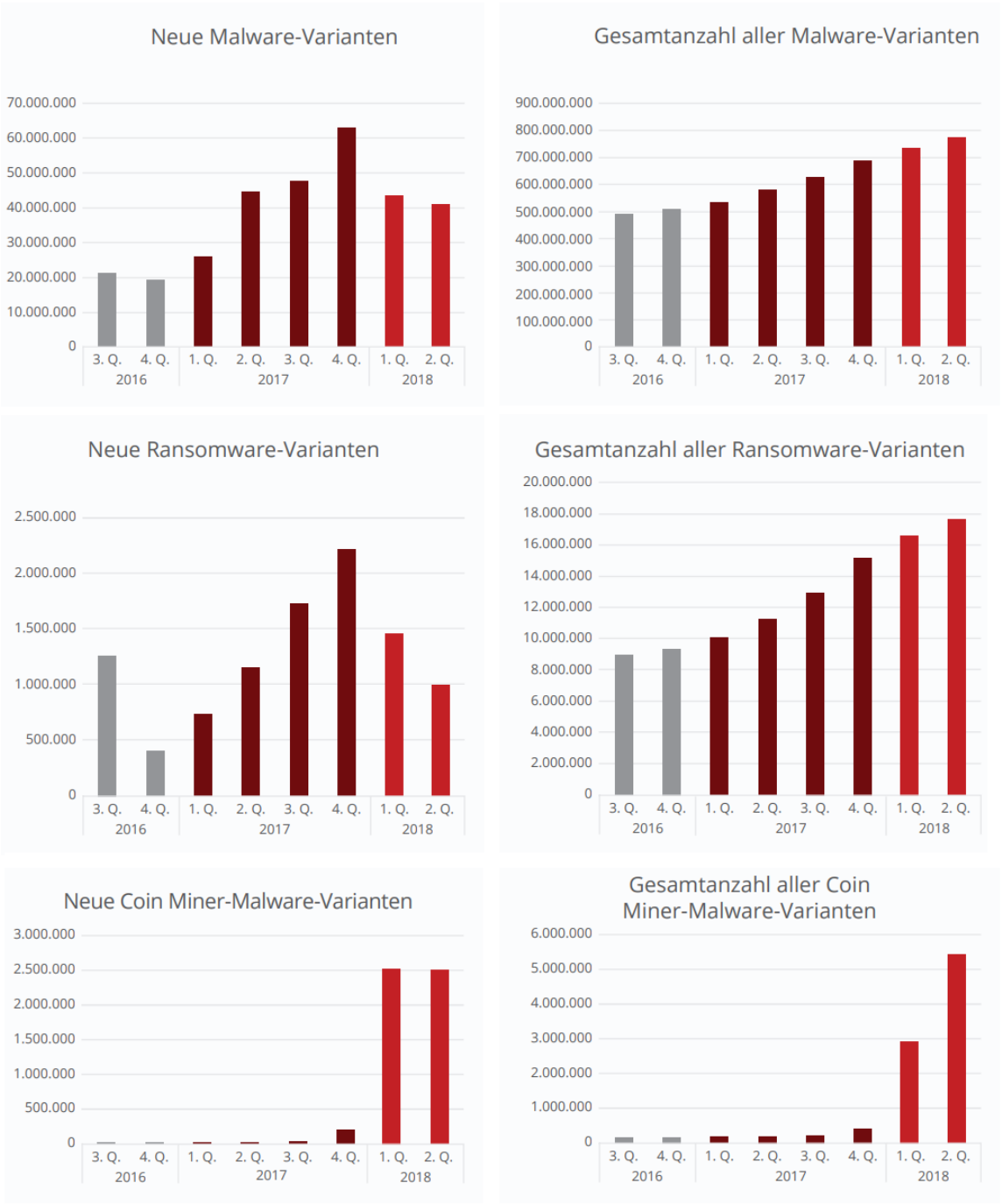


Abbildung 1.1.: Zunahme von Malware-, Ransomewar- und Coin Miner-Varianten im Zeitraum vom 3. Quartal 2016 bis zum 2. Quartal 2018 nach [BCC+18].

IT-Sicherheitssysteme basieren in der Regel auf netzwerk- oder hostbasierten Daten. Der Fokus dieser Arbeit liegt auf der Verarbeitung netzwerkbasierter Daten, da diese leicht aufgenommen werden können und einen Gesamtüberblick des Unternehmensnetzwerkes bieten. Laut Hofstede et al. [HPS+18] bieten netzwerkbasierte Sicherheitslösungen eine bessere Skalierbarkeit als

hostbasierte Sicherheitslösungen. Des Weiteren können laut Ficke et al. [FSB+18] netzwerkbasierte Sicherheitslösungen Angriffe bereits detektieren, bevor diese den Zielhost erreichen. Netzwerkdaten liegen in der Regel in paket- oder flowbasiertem Format vor (siehe Abschnitt 4.2). Traditionelle Intrusion Detection Systeme (IDS) verwenden meist paketbasierte Netzwerkdaten und detektieren Angriffe mithilfe von Deep Packet Inspection (DPI) [XCS+16, Pax99, BK13]. DPI analysiert nicht nur die Verbindungsinformationen, sondern auch die übertragenen Nutzdaten (Payload). Der zunehmende Anteil verschlüsselter Netzwerkverbindungen erschwert den erfolgreichen Einsatz von DPI [Koc11]. Basierend auf dieser Entwicklung gibt es einige Arbeiten, die sich speziell mit der Analyse von verschlüsselten Nutzdaten beschäftigen. An dieser Stelle sei auf [FTV+10, SLP+15, HKH+17] verwiesen. Weitere negative Einflussfaktoren für DPI sind zunehmende datenschutzrechtliche Bedenken [ZTS+13] und wachsende Datenmengen [GZL06]. Aufgrund dieser Entwicklungen hat sich die flowbasierte Datenanalyse als alternativer Ansatz herauskristallisiert [SSS+10]. Flowbasierte Netzwerkdaten reduzieren die Datenmenge, sind resistent gegenüber verschlüsselten Verbindungen und mindern datenschutzrechtliche Bedenken. Aus diesen Gründen fokussiert diese Arbeit auf die Verarbeitung flowbasierter Netzwerkdaten.

Im Vergleich zu paketbasierten Netzwerkdaten beinhalten flowbasierte Netzwerkdaten jedoch weniger Informationen. Deshalb forschen unterschiedliche Arbeitsgruppen an der Auswertung und Integration zusätzlicher Informationsquellen. Sabottke et al. [SSD15] untersuchen beispielsweise auf Twitter verbreitete Informationen über Schwachstellen. Sapienza et al. [SBD+17] verwenden Forenbeiträge aus dem Dark Web und Twitter Posts von IT-Sicherheitsexperten, um ein Frühwarnsystem zu entwickeln. Viele Ansätze nutzen aufgrund der Beschaffenheit dieser Informationsquellen (Log-Dateien, Forenbeiträge, ...) häufig Methoden aus dem Bereich Text Mining. Exemplarisch sei an dieser Stelle auf [RL06, RL07, MT18a] verwiesen.

Die vorliegende Zielsetzung *Detektion von sicherheitskritischen Ereignissen in Unternehmensnetzwerken mittels Data Mining* wirft verschiedene Fragestellungen auf, die über das Thema IT-Sicherheit hinausgehen. Dies betrifft vor allem zwei Bereiche, die Generierung realistischer Daten und die Verarbeitung heterogener Daten. Unter heterogenen Daten werden in diesem Zusammenhang Daten verstanden, die aus kontinuierlichen und kategorischen Werten bestehen. Einerseits erfordert die Beschaffenheit flowbasierter Netzwerkdaten die Entwicklung neuer grundlegender Beiträge zum Umgang mit heterogenen Daten. Andererseits benötigen überwachte Data Mining-Algorithmen gelabelte Trainingsdaten, sodass die Generierung realistischer Netzwerkdaten ein weiteres zentrales Forschungsgebiet darstellt. Die Herausforderungen dieser beiden Problemstellungen und der eigentlichen Detektion von Angriffsszenarien sind im nachfolgenden Abschnitt genauer spezifiziert.

1.2. Herausforderungen

Forscher verfolgen bereits seit Jahrzehnten die Idee, neue Angriffe mittels Data Mining zu erkennen. Trotz intensiver Forschung ist die Anzahl erreichter Erfolge jedoch geringer als in anderen Anwendungsgebieten. Beispielsweise werden neuronale Netze erfolgreich für selbstfahrende Autos [BTD+16], zur Gesichtserkennung [LGT+97, LLS+15], oder in Spielen wie GO [SHM+16] eingesetzt. Im Speziellen haben vor allem tiefe faltende neuronale Netze (Deep Convolutional Neural Networks) in den letzten Jahren einige Durchbrüche in den Bereichen Bild-, Audio- und Videoverarbeitung erreicht [LBH15]. Im Gegensatz dazu stehen die wesentlichen Durchbrüche im Bereich Intrusion Detection noch aus.

Dieser Abschnitt behandelt die speziellen Herausforderungen für das Anwenden von Data Mining-Algorithmen im Anwendungsgebiet flowbasierter Intrusion Detection. Data Mining-

Algorithmen sind systematische Methoden zum Extrahieren von Wissen aus Daten und optimieren Bedingungen nach rein formalen Kriterien, ohne die Bedeutung der zugrunde liegenden Daten zu berücksichtigen. Deshalb ist die Anwendung von Data Mining-Algorithmen mit unterschiedlichen Herausforderungen in unterschiedlichen Anwendungsbereichen verbunden. Beispielsweise liegen im Bereich Bildverarbeitung hochdimensionale Datenpunkte oder im Bereich Textverarbeitung nicht kontinuierliche Attribute vor. Eine Behandlung der besonderen Herausforderungen im Bereich der Detektion sicherheitskritischer Ereignisse mithilfe flowbasierter Netzwerkdaten findet im Folgenden statt.

1.2.1. Herausforderung 1 - Beschaffenheit der Daten

Flowbasierte Netzwerkdaten bestehen aus kontinuierlichen und kategorischen Attributen (siehe Abschnitt 4.2). Ein Flow enthält kontinuierliche Attribute wie die Anzahl übertragener Bytes und kategorische Attribute wie IP-Adressen. Die Kombination aus kontinuierlichen und kategorischen Attributen erschwert die Anwendung vieler Data Mining-Algorithmen, da diese häufig auf die Verarbeitung von Daten mit ausschließlich kontinuierlichen oder ausschließlich kategorischen Attributen fokussiert sind. Beispielsweise können neuronale Netze oder Support Vektor Maschinen (SVMs) ausschließlich kontinuierliche Eingabewerte verarbeiten. Damit diese Verfahren kategorische Werte verarbeiten können, müssen diese durch geeignete Vorverarbeitungsschritte wie Binarisierung vorverarbeitet werden. Der Apriori Algorithmus [AS94] zum Auffinden von Assoziationsregeln basiert hingegen auf kategorischen Werten.

Folglich stellt die Beschaffenheit der Daten eine zentrale Herausforderung dar, die durch den Entwurf geeigneter Abstandsmaße für flowbasierte Netzwerkdaten oder deren semantikerhaltende Transformation in kontinuierliche Darstellungen angegangen werden kann.

1.2.2. Herausforderung 2 - Fehlende Datensätze

Im Anwendungsgebiet flowbasierter Intrusion Detection ist ein Mangel an öffentlich verfügbaren und gelabelten Datensätzen zu verzeichnen [SP10]. Die Datensätze DARPA98/99 [LHF+00, MIT] und KDD CUP 99 [Uni99] sind die verbreitetsten Datensätze im Bereich Intrusion Detection. Diese Datensätze wurden vor mehr als 20 Jahren aufgenommen und spiegeln somit kein aktuelles Nutzerverhalten und keine aktuellen Angriffsszenarien wider. Nichtsdestotrotz verwenden viele aktuelle Arbeiten diese Datensätze, beispielsweise [LKT15, JJC+16, AG15, TK17, BCM+16, KKT+16, TMM+16]. Özgür und Erdem [ÖE16] untersuchten 149 Publikationen im Zeitraum 2010 bis 2015 und kamen zu dem Ergebnis, dass KDD CUP 99 der meistgenutzte Datensatz ist.

Die Verwendung von realen Netzwerkdaten ist ebenfalls problematisch, da für diese keine Grundwahrheit (Labels) zur Verfügung steht, welche die Daten in mindestens zwei Klassen, *normal* und *verdächtig*, unterteilen. Zudem gestaltet sich manuelles Labeln als schwierig und zeitaufwendig, da in größeren Unternehmen stündlich mehrere Millionen flowbasierte Datenpunkte anfallen. Des Weiteren stellen sich die Veröffentlichung realer Netzwerkdaten und die explizite Ausführung von Angriffsszenarien in Produktivumgebungen als schwierig heraus.

Überwachte Data Mining-Algorithmen benötigen jedoch gelabelte Datensätze in der Trainingsphase. Zusätzlich bieten gelabelte Datensätze eine sehr gute Möglichkeit zur Evaluierung der Leistungsfähigkeit neu entwickelter Methoden. Deshalb stellt der Mangel an öffentlich verfügbaren gelabelten und aktuellen Datensätzen eine weitere zentrale Herausforderung dar.

1.2.3. Herausforderung 3 - Detektion von Angriffsszenarien

Die Detektion von Angriffsszenarien ist eine sehr komplexe Herausforderung, die sich aus mehreren Aspekten zusammensetzt.

Benutzeraktivitäten in Netzwerkdaten. Klassifikatoren (siehe Abschnitt 3.1) ordnen jeden Datenpunkt einer vordefinierten Klasse zu. In der Bildverarbeitung gehört beispielsweise jedes Bild bezüglich dessen Inhalts einer Klasse an. Der MNIST Datensatz besteht aus 70.000 Bildern mit handgeschriebenen Ziffern. Jedes Bild repräsentiert eine Ziffer, was dessen Klasse widerspiegelt. Diese Situation ist für flowbasierte Netzwerkdaten anders, da Benutzeraktivitäten auf Anwendungsebene verschiedene Mengen an Netzwerkdaten verursachen. Wenn ein Mitarbeiter eine E-Mail versendet, ist auf Netzwerkebene ein Flow zum E-Mail Server zu beobachten. Wenn ein Mitarbeiter hingegen eine Webseite im Browser aufruft, können mehrere Flows beobachtet werden, da der Rechner des Mitarbeiters zunächst eine DNS-Anfrage an den DNS-Server sendet. Danach baut der Rechner eine Verbindung zu Port 80 des Web-Servers auf. Nach diesem initialen Verbindungsaufbau lädt der Rechner gegebenenfalls weitere Bilder und Inhalte anderer Webseiten nach. Für dieses Nachladen baut der Rechner neue Verbindungen zum Web-Server und zu den anderen Web-Servern auf. Während eine E-Mail genau einen Flow auf Netzwerkebene verursacht, ist für den Aufruf einer Webseite eine Sequenz von Flows zu beobachten. Des Weiteren gibt es viele Benutzeraktivitäten auf Anwendungsebene, die keinen Netzwerkverkehr verursachen wie das Erstellen eines Word-Dokuments. Folglich stellen sich Benutzeraktivitäten auf Anwendungsebene durch eine beliebige Anzahl von Flows auf Netzwerkebene dar und es gibt keine absolute Gewissheit, welche Flows welchen Benutzeraktivitäten zuzuschreiben sind.

Unterschiedliche Rollen. Unterschiedliche Mitarbeiter eines Unternehmens nehmen unterschiedliche Rollen ein. Als Beispiel dient das Arbeitsverhalten eines Sekretärs und eines Systemadministrators. Die Hauptaufgabe des Systemadministrators ist die Administration der internen Server. Folglich baut der Systemadministrator viele SSH-Verbindungen zu den Servern auf. Im Gegensatz dazu nutzt der Sekretär seine Netzwerkverbindung primär zum Telefonieren, E-Mail schreiben und Organisieren von Terminen. In diesem Szenario wäre es folglich normal, wenn der Systemadministrator eine SSH-Verbindung zu einem internen Server aufbaut, aber sehr verdächtig, wenn der Sekretär dies tun würde. Als Folge dessen spielt der Benutzer, der auf Netzwerkebene durch die Quell-IP-Adresse identifiziert werden kann, eine wichtige Rolle.

Veränderung des Verhaltens. Eine weitere Herausforderung ist, dass sich der Netzwerkverkehr eines Unternehmens stetig ändert. Beispielsweise ändert sich das Netzwerkverhalten durch die Einführung neuer Dienste, die Abschaltung alter Dienste oder außerhalb typischer Arbeitszeiten.

Hohe Kosten für Fehler. Fehlentscheidungen verursachen hohe Kosten im Bereich IT-Sicherheit. Im einfachsten Fall kann ein IDS zwei Arten von Fehlern, Fehlalarmierungen (False Positives) und nicht erkannte Angriffe (False Negatives), machen (siehe Abschnitt 2.5.1). Eine Fehlalarmierung würde bedeuten, dass das Sicherheitssystem einen Alarm generiert, obwohl der zugrundeliegende Netzwerkverkehr legitim ist. Schon sehr geringe Fehlalarmierungsraten machen ein Sicherheitssystem unbrauchbar [Axe99]. Beim flowbasierten UGR'16 Datensatz würde eine Fehlalarmierungsrate von 1 Promille dazu führen, dass das System durchschnittlich alle 0,66 Sekunden eine Fehlalarmierung generiert. Koch et al. [KGR14] kommen zu der Erkenntnis, dass anomaliebasierte IDS häufig unter hohen Fehlalarmierungsraten aufgrund von fehlendem Wissen über Normalverhalten leiden. Im Falle eines False Negatives würde das System einen Angriff als normalen Netzwerkverkehr klassifizieren und der Angriff unentdeckt bleiben. Die Folgen für False Negatives können sehr unterschiedlich ausfallen. In einigen Fällen erleidet das Unternehmen keinen Schaden. In anderen Fällen erleidet das Image eines Unternehmens Schaden

oder es ist ein Verlust beziehungsweise eine Manipulation vertraulicher Daten zu verzeichnen (siehe Beispiele in Abschnitt 1.1).

Echtzeitanforderung. IT-Sicherheitssysteme müssen Daten in Echtzeit verarbeiten können, um das Stehlen und Manipulieren von Daten zu verhindern. Diese Anforderung geht mit einer großen Menge von Daten einher, da Unternehmensnetzwerke täglich Millionen bis Milliarden Flows verursachen. Beispielsweise beinhaltet der UGR'16 Datensatz [MCM+18] den Netzwerkverkehr von einem Internet Service Provider (ISP) über 4 Monate und besteht aus ungefähr 17 Milliarden Flows. Um diesen Datensatz in Echtzeit zu verarbeiten, ist die Analyse von ungefähr 1.500 Flows pro Sekunde erforderlich.

Intelligente Gegner. In vielen Anwendungsgebieten gibt es vordefinierte Probleme. An dieser Stelle sei beispielsweise die Gesichtserkennung von Personen auf Bildern oder das autonome Fahren eines Autos genannt. Im Anwendungsgebiet IT-Sicherheit werden die Angriffsszenarien jedoch von intelligenten Hackern ausgeführt. Es ist davon auszugehen, dass die Hacker auch Wissen über die Detektionsmethoden besitzen und stets auf neuen Wegen versuchen, ihr irreguläres Verhalten zu verschleiern.

1.3. Lösungsansätze der Arbeit

Im Mittelpunkt dieser Arbeit steht die Analyse flowbasierter Netzwerkdaten zur Detektion sicherheitskritischer Ereignisse in Unternehmensnetzwerken. In Abbildung 1.2 sind die Hauptbeiträge dieser Arbeit dargestellt. Diese lassen sich in drei Kategorien Datengenerierung, Behandlung heterogener Daten und Datenanalyse unterteilen. In den drei Bereichen wurden sowohl neuartige methodische (blau) als auch konzeptionelle (rot) Ansätze beigetragen. Dies spiegelt sich im Aufbau der Arbeit wider.

- Die Berechnung von Abständen zwischen Datenpunkten ist ein Kernthema vieler Data Mining-Algorithmen. Diese Arbeit entwickelt ein Abstandsmaß für heterogene Daten und einen Ansatz zur Transformation von IP-Adressen in kontinuierliche Vektoren (siehe Abschnitt 1.3.1).
- Netzwerkbasierte Daten enthalten personenbezogene Informationen und können häufig aus datenschutzrechtlichen Gründen nicht veröffentlicht werden. Diese Arbeit entwickelt zwei Ansätze zur Generierung gelabelter flowbasierter Netzwerkdaten (siehe Abschnitt 1.3.2).
- Diese Arbeit leistet zwei Beiträge zur Detektion von Angriffsszenarien in flowbasierten Netzwerkdaten. Während der erste Beitrag ein allgemeines Konzept entwirft, umfasst der zweite Beitrag eine neuartige Methode zur Detektion langsamer Port Scans (siehe Abschnitt 1.3.3).

1.3.1. Heterogene Daten

Wie bereits in Abschnitt 1.2.1 erwähnt, bestehen flowbasierte Netzwerkdaten aus kontinuierlichen und kategorischen Attributen. Gleichzeitig können viele Algorithmen wie neuronale Netze ausschließlich kontinuierliche Eingabedaten verarbeiten. Die nächsten beiden Absätze fassen die Beiträge zur Handhabung heterogener Daten (Herausforderung 1) zusammen.

Kapitel 5 stellt ein Abstandsmaß zur Berechnung von Abständen zwischen Datenpunkten mit heterogenen Attributen vor. Das Abstandsmaß benötigt keine gelabelten Trainingsdaten und erlernt unüberwacht Abstände für kategorische Attribute. Viele Ansätze greifen bei der Berechnung von Abständen zwischen kategorischen Attributen auf den Hamming-Abstand zurück.

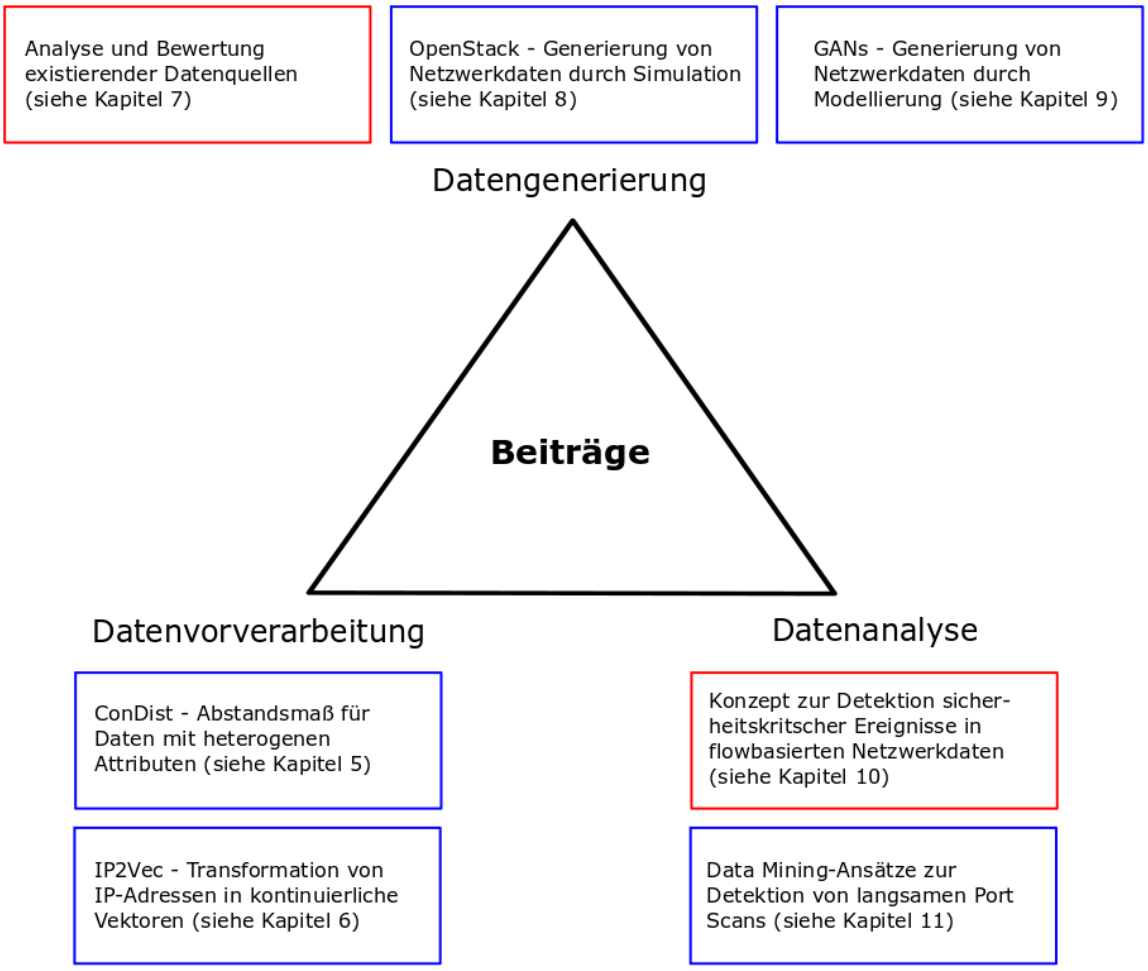


Abbildung 1.2.: Hauptbeiträge dieser Dissertation.

Jedoch unterscheidet der Hamming-Abstand lediglich zwischen Gleichheit und Ungleichheit bei kategorischen Ausprägungen, wodurch Informationen verloren gehen können. Existierende kontextbasierte kategorische Abstandsmaße extrahieren Informationen aus korrelierten Kontextattributen. Diese Abstandsmaße erfüllen jedoch nicht immer die Kriterien einer Metrik oder haben Schwierigkeiten bei Daten, bei denen die Attribute nicht in Korrelation zueinander stehen. Das entwickelte Abstandsmaß ConDist greift zur Abstandsberechnung ebenfalls auf Kontextattribute zurück und setzt an den beiden oben genannten Schwachstellen an. Kapitel 5 beweist, dass es sich bei ConDist um eine Metrik handelt. Des Weiteren berücksichtigt ConDist auch den Grad der Korrelationen zwischen Attributen bei der Abstandsberechnung. Dies wirkt sich dahingehend aus, dass keine unkorrelierten Kontextattribute berücksichtigt werden, die fehlerhafte Informationen in die Abstandsberechnung einbringen könnten. Folglich kann ConDist auch für unkorrelierte Datensätze verwendet werden, würde sich aber in diesem Fall auf den Hamming-Abstand reduzieren, da keine Zusatzinformationen extrahiert werden können.

IP-Adressen sind kategorische Werte und allgegenwärtig in netzwerkbasierten Daten. Aufgrund der großen Dimensionalität (2^{32} für IPv4, 2^{128} für IPv6) scheidet der Ansatz einer Binarisierung von IP-Adressen aus. Viele existierende Intrusion Detection Methoden verwerfen IP-Adressen in der Analyse. Da IP-Adressen jedoch die Identifikation von Hosts erlauben und

Hosts unterschiedliche Nutzungsprofile aufweisen (siehe Herausforderung 3), führt das Verwerfen von IP-Adressen zu einem erheblichen Informationsverlust. Ausgefeiltere Ansätze transformieren IP-Adressen in geographische Koordinaten. Geographische Koordinaten stellen kontinuierliche Attribute dar und es können existierende Abstandsmaße verwendet werden. Die berechneten Abstände beinhalten jedoch wenig Information über die tatsächlichen Ähnlichkeiten zwischen den IP-Adressen bezüglich ihres Verhaltens. An diesem Punkt setzt die Methode IP2Vec (siehe Kapitel 6) an und transformiert IP-Adressen mit ähnlichen Netzwerkverhalten auf ähnliche kontinuierliche Vektoren. IP2Vec greift auf verfügbare Informationen aus flowbasierten Netzwerkdaten zurück und definiert IP-Adressen als ähnlich, wenn die IP-Adressen ähnliche Netzwerkverbindungen aufbauen. Gleichzeitig transformiert IP2Vec nicht nur IP-Adressen in numerische Vektoren, sondern auch alle anderen kategorischen Attribute von flowbasierten Netzwerkdaten.

1.3.2. Generierung von Netzwerkdaten

Herausforderung 2 (siehe Abschnitt 1.2.2) betrifft die fehlenden öffentlichen Datensätze im Bereich netzwerkbasierter Intrusion Detection. Im Rahmen dieser Arbeit wird zunächst eine ausführliche Recherche existierender Quellen für netzwerkbasierte Daten durchgeführt, bevor zwei Ansätze zur Generierung gelabelter Datensätze abgeleitet werden.

Gelabelte Trainingsdatensätze sind für überwachte Data Mining-Methoden unerlässlich und eignen sich sehr gut zur Evaluation von Intrusion Detection Systemen (IDS). In Kapitel 7 wird eine ausführliche Recherche existierender Quellen für netzwerkbasierte Daten durchgeführt. Neben klassischen Datensätzen werden auch Netzwerkdaten-Generatoren berücksichtigt. Hierbei werden speziell zur Analyse von Datensätzen mehrere Bewertungskriterien definiert und die gefundenen Datensätze diesbezüglich analysiert. Basierend auf dieser Analyse erarbeitet Kapitel 7 einige Schlussfolgerungen für das Erstellen neuer Datensätze.

Für Daten aus realen Unternehmensnetzwerken steht in der Regel keine Grundwahrheit (Labels) zur Verfügung. Des Weiteren können in realen Umgebungen keine Angriffsszenarien ausgeführt werden, da diese den normalen Tagesbetrieb gefährden würden. In Kapitel 8 wird eine Simulationsumgebung zur Generierung von flowbasierten Netzwerkdaten in OpenStack aufgebaut. Schwerpunkt dieses Beitrags ist die Generierung von realistischem Benutzerverhalten durch automatisierte Python Skripte, welche den grundlegenden Unterschied zu normalen Testumgebungen ausmachen. Gleichzeitig können in dieser Simulationsumgebung verschiedene Angriffsszenarien ausgeführt werden. Da sowohl normales Benutzerverhalten als auch Angriffsszenarien automatisiert ausgeführt werden, ist ein exaktes Labelling der generierten Daten möglich. Im Vergleich zu existierenden Datensätzen wurde in Kapitel 8 ein spezieller Fokus auf realistisches Normalverhalten gelegt, welches sich auch durch die Einbeziehung eines externen Servers, der direkt mit dem Internet verbunden ist, kennzeichnet. Auch bringt die Nutzung der virtuellen Testumgebung verschiedene Vorteile mit sich. Die Virtualisierung erlaubt die flexible Erstellung verschiedener Netzwerkstrukturen, welche auf verschiedene Unternehmensstrukturen angepasst werden können und ermöglicht die dauerhafte Aktualisierung und das Aufzeichnen neuer Datensätze.

In Kapitel 9 wird ein alternativer Ansatz zur Generierung von synthetischen Netzwerkdaten durch Modellierung entwickelt. Modellierungsansätze verwenden flowbasierte Netzwerkdaten als Eingabe, erlernen die zugrundeliegende Eigenschaften und sind anschließend in der Lage, neue synthetische Daten mit den gleichen Eigenschaften zu erzeugen. Bisherige Ansätze sind häufig statisch und benötigen eine Vielzahl benutzerdefinierter Parameter oder die Integration von Domänenwissen zur schrittweisen Generierung von Flows. In diesem Kapitel werden erstmals Generative Adversarial Networks (GANs) zum Erzeugen von flowbasierten Netzwerkdaten verwendet.

Zwar wurden GANs bereits im Bereich flowbasierter Intrusion Detection verwendet, jedoch wurden bisher immer nur ausgewählte kontinuierliche Attribute mit GANs erzeugt. Schwerpunkt dieses Beitrags ist die Berücksichtigung aller kategorischer Attribute bei der Erstellung flowbasierter Netzwerkdaten. Hierfür werden unterschiedliche Verfahren zur Handhabung kategorischer Attribute getestet und miteinander verglichen. Diese Vorgehensweise kann speziell dafür genutzt werden, existierende Datensätze mit synthetisch erzeugten Datenpunkten anzureichern und somit eine größere Varianz zu erzeugen. Somit ist dieser Ansatz als Erweiterung und nicht als Alternative zum Simulationsansatz in Kapitel 8 zu werten.

1.3.3. Erkennung von Angriffsszenarien

Die Detektion sicherheitskritischer Ereignisse in flowbasierten Netzwerkdaten bringt einige Herausforderungen mit sich. In den nächsten beiden Absätzen werden die Beiträge zur Detektion von Angriffsszenarien, welche ein Konzept zur Detektion von Angriffen und eine neuartige Methode zur Detektion von langsamen Port Scans umfassen, kurz vorgestellt.

In Kapitel 10 wird ein Konzept zur Detektion von Angriffen in flowbasierten Netzwerkdaten entworfen. Dieses Konzept berücksichtigt die Tatsache, dass sich Benutzeraktivitäten auf Netzwerkebene durch unterschiedlich lange Sequenzen von Flows kennzeichnen und dass verschiedene Nutzer verschiedene Rollen einnehmen (siehe Herausforderung 3). Hierfür wird die grundlegende Vorgehensweise von Angriffsszenarien analysiert und darauf basierend Netzwerkdaten aus drei unterschiedlichen Perspektiven bewertet. Zusätzlich sieht das Konzept die Integration von verfügbaren Domänenwissen vor. Das Konzept unterscheidet sich im Vergleich zu existierenden Lösungen durch die zusätzliche Integration von Domänenwissen und die Analyse der flowbasierten Netzwerkdaten aus drei Perspektiven, welche auf die typischen Phasen von Angriffsszenarien angepasst sind. Zur Realisierung des Konzepts wurde das Coburg-Utility-Framework (CUF) (siehe Abschnitt 3.6) erweitert, welches auf einer Pipe-and-Filter Architektur basiert und somit die flexible Gestaltung diverser Arbeitsabläufe ermöglicht.

Der letzte Beitrag dieser Dissertation sind zwei Methoden zur Detektion von langsamen Port Scans. Beide Methoden setzen auf eine neuartige Vorverarbeitungskette flowbasierter Netzwerkdaten, welche speziell auf die Detektion von Port Scans angepasst ist. Eine Methode basiert auf sequentiellen Hypothesentests und benötigt keine Trainingsdaten. Die zweite Methode basiert auf Klassifikatoren und benötigt gelabelte Trainingsdaten. Beide Methoden, sowie die Vorverarbeitungskette, wurden in das von Kapitel 10 vorgestellte Konzept integriert. Durch die spezielle Vorverarbeitung der Daten detektieren die beiden Methoden Port Scans mit höherer Genauigkeit als in existierenden Lösungen.

1.4. Aufbau der Arbeit

Diese Arbeit ist in fünf Teile (I) Grundlagen, (II) Behandlung heterogener Daten, (III) Generierung von Netzwerkdaten, (IV) Detektion von Angriffsszenarien und (V) Zusammenfassung und Ausblick unterteilt.

Teil (I) beschäftigt sich mit den Grundlagen dieser Arbeit und ist in drei Kapiteln untergliedert. Zunächst behandelt Kapitel 2 Grundlagen aus dem Bereich Data Mining, bevor sich Kapitel 3 mit der wesentlichen Funktionsweise verwendeter Methoden und Visualisierungsverfahren befasst. Kapitel 4 beleuchtet die Anwendungsdomäne IT-Sicherheit. Dies umfasst die Diskussion existierende Sicherheitslösungen, typische Angriffsarten und verfügbare Datenquellen.

Teil (II) beschäftigt sich mit der Handhabung heterogener Daten und im Speziellen mit den Umgang kategorischer Attribute. Zunächst stellt Kapitel 5 ein Abstandsmaß zur Berechnung von Abständen zwischen Datenpunkten mit heterogenen Attributen vor. Das entworfene Abstandsmaß ConDist ist ein allgemeines Abstandsmaß und nicht nur auf die Domäne IT-Sicherheit beschränkt. Kapitel 6 präsentiert IP2Vec, eine Methode zur Transformation kategorischer Attribute in kontinuierliche Vektoren. Dieser Ansatz ist auf flowbasierte Netzwerkdaten fokussiert.

Teil (III) beschäftigt sich mit der Generierung von Netzwerkdaten und besteht aus drei Kapiteln. Kapitel 7 entwickelt geeignete Bewertungskriterien für netzwerkbasierte Daten und bewertet existierende Datensätze anhand dieser Bewertungskriterien. Kapitel 8 stellt einen Ansatz zur Generierung netzwerkbasierter Daten in einer Simulationsumgebung vor. Ergänzend dazu, wird in Kapitel 9 ein generatives Modell entwickelt, welches flowbasierte Netzwerkdaten als Eingabe verwendet und darauf basierend neue synthetische Daten generieren kann.

Teil (IV) beschäftigt sich mit der Detektion von Angriffsszenarien und ist in zwei Kapiteln unterteilt. Kapitel 10 stellt einen Arbeitsablauf zur Detektion von Angriffsszenarien vor. Der Arbeitsablauf berücksichtigt die typischen Phasen eines Angriffsszenarios, sieht die Integration von zusätzlichen Domänenwissen, sowie die Analyse der Daten aus drei verschiedenen Perspektiven vor. Eine konkrete Methode zur Detektion von langsamen Port Scans findet sich in Kapitel 11.

Am Ende der Arbeit (Teil V) findet sich eine Zusammenfassung und ein Ausblick für zukünftige Arbeiten.

Teil I.

Grundlagen

2. Data Mining

Diese Arbeit zielt auf die Detektion sicherheitskritischer Ereignisse in Unternehmensnetzwerken mithilfe von Data Mining-Algorithmen. Deshalb fasst dieses Kapitel die erforderlichen Grundlagen aus dem Bereich Data Mining zusammen. Zunächst findet eine generelle Definition und Einordnung des Begriffs Data Mining statt. Danach wird ein weit verbreitetes Prozessmodell vorgestellt, welches im Rahmen dieser Arbeit Einsatz findet. Im Anschluss behandelt dieses Kapitel relevante grundlegende Themen, nämlich die unterschiedlichen Datentypen, Abstands- und Ähnlichkeitsmaße sowie Evaluierungsmaßzahlen.

2.1. Allgemeines

2.1.1. Definition und Einordnung

Für den Begriff Data Mining gibt es in der Literatur keine einheitliche Definition. Beispielsweise beschreiben Fayyad et al. [FPS96] Data Mining als die Anwendung spezieller Algorithmen zum Auffinden von Mustern in Daten. Eine ähnliche Definition geben Witten und Frank [WFH05], indem sie Data Mining als einen automatisierten oder teilautomatisierten Prozess zum Auffinden nützlicher Informationen aus Daten definieren. Han et al. [HKP11] verwenden eine breitere Definition und bezeichnen den gesamten Prozess der Entdeckung interessanter Muster und Kenntnisse aus großen Datenmengen als Data Mining. Einige Definitionen setzen Data Mining häufig mit den Begriff Knowledge Discovery from Data (KDD) gleich, während andere Definitionen Data Mining als einen wesentlichen Schritt des KDD-Prozesses definieren [HKP11]. All diese Definitionen haben gemeinsam, dass der Begriff Data Mining die Anwendung von Algorithmen zum Auffinden von Wissen aus Daten beschreibt. In dieser Arbeit wird die Definition von Han et al. [HKP11] übernommen.

Data Mining-Algorithmen lassen sich grob in zwei Kategorien überwacht (supervised) und unüberwacht (unsupervised) unterteilen. Überwachte Algorithmen benötigen gelabelte Daten, das heißt, die Datenpunkte des Datensatzes sind in vordefinierte Klassen unterteilt. Überwachte Algorithmen greifen auf diese Klassenzugehörigkeiten in der Trainingsphase zurück. Typische Vertreter dieser Kategorie sind Klassifikatoren (siehe Abschnitt 3.1). Unüberwachte Algorithmen arbeiten hingegen auf nicht gelabelten Datenpunkten. Derartige Verfahren suchen beispielsweise nach Mustern und ähnlichen Datenpunkten in einem Datensatz. Clusterverfahren (siehe Abschnitt 3.2) sind typische Vertreter dieser Kategorie. Neben überwachten und unüberwachten Verfahren gibt es noch teil-überwachte und schwach-überwachte Verfahren. Teil-überwachte Verfahren stellen eine Kombination der ersten beiden Verfahren dar und greifen auf Daten zurück, bei denen nur eine Teilmenge der zur Verfügung stehenden Datenpunkte über Klassenlabels verfügt [HKP11]. Schwach-überwachte Verfahren verarbeiten Daten deren Labels möglicherweise fehlerhaft, ungenau oder unvollständig sind. Im Folgenden werden die zwei relevanten Teilgebiete Klassifikation und Clusteranalyse kurz erläutert.

2.1.2. Klassifikation

Klassifikation ist ein Teilgebiet des Data Mining und gehört zur Kategorie überwachter Lernverfahren [HKP11]. Klassifikatoren erlernen ein Modell, welches in der Lage ist, Objekte (sogenannte Datenpunkte) in vordefinierte Klassen zu unterteilen. Merkmale (sogenannte Attribute) beschreiben die Eigenschaften der Objekte. Das Modell wird in einer Trainingsphase mithilfe eines Datensatzes erlernt. Die Datenpunkte müssen mit entsprechenden Klassenzugehörigkeiten (sogenannten Labels) gekennzeichnet sein und sollten die Grundgesamtheit der Datenverteilung möglichst repräsentativ widerspiegeln. Nach der Trainingsphase kann das Modell in der Klassifikationsphase neue Datenpunkte, die keine Labels besitzen, den vordefinierten Klassen zuordnen [HKP11].

2.1.3. Clusteranalyse

Clusteranalyse ist ein weiteres Teilgebiet im Bereich Data Mining und gehört zur Kategorie unüberwachter Lernverfahren [HKP11]. Die Clusteranalyse ordnet Datenpunkte in Gruppen (sogenannte Cluster). Ziel der Clusteranalyse ist es, dass die Datenpunkte innerhalb eines Clusters so ähnlich wie möglich und Datenpunkte aus verschiedenen Clustern möglichst unähnlich sind [HKP11]. Je besser dieses Ziel erreicht wird, desto besser eignet sich das entsprechende Verfahren zur Clusteranalyse. Zur Berechnung von Ähnlichkeiten werden klassische Ähnlichkeitsmaße (siehe Abschnitt 2.4) verwendet.

2.2. CRISP-DM Modell

CRISP-DM (Cross-Industry Standard Process for Data Mining) [She00] ist ein Standardmodell für Data Mining-Abläufe. Abbildung 2.1 zeigt das CRISP-DM Modell.

Das CRISP-DM Modell ist ein Prozessmodell zur vollständigen Planung und Durchführung von Data Mining-Projekten. Das Modell umfasst die folgenden sechs Phasen: Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation und Deployment. Grundsätzlich ist CRISP-DM ein iteratives Modell. Dies verdeutlicht auch der äußere graue Kreis in Abbildung 2.1, da auch nach erfolgreicher Deployment Phase der Prozess noch nicht beendet ist und wieder von vorne beginnt.

Im Folgenden findet eine genauere Betrachtung der sechs Phasen nach [She00] statt. Die erste Phase Business Understanding beschäftigt sich mit dem Verstehen von Projektzielen aus wirtschaftlicher Sicht und der Überführung dieser Ziele in entsprechende Data Mining-Fragestellungen. Data Understanding ist die zweite Phase und zielt auf das Verständnis der zur Verfügung stehenden Daten. Hierzu gehört das Sammeln der Daten aus verfügbaren Datenquellen, das Untersuchen der Daten nach relevanten Informationen, sowie die Qualitätsprüfung der Daten. Die dritte Phase Data Preparation umfasst alle Schritte der Datenvorverarbeitung wie beispielsweise die Selektion geeigneter Attribute, das Bereinigen von offensichtlich falschen Werten, das Formatieren von Werten (zum Beispiel Entfernen von Sonderzeichen aus Zeichenketten), die Vereinheitlichung von Einheiten (zum Beispiel Umwandlung von Kilogramm in Gramm) oder die Transformation kategorischer beziehungsweise kontinuierlicher Attribute. In der vierten Phase Modelling werden geeignete Algorithmen zur Problemlösung ausgewählt und mit verschiedenen Parameterkonfigurationen optimiert. Des Weiteren werden in dieser Phase geeignete Testszenarien entworfen. Bevor die Lösungsansätze produktiv zum Einsatz kommen, findet eine Evaluierung in der fünften Phase statt. In der Phase Evaluation wird überprüft, ob der Ansatz wirklich das grundlegende wirtschaftliche Ziel löst und wie gut die entworfenen Lösungen generalisieren. Die

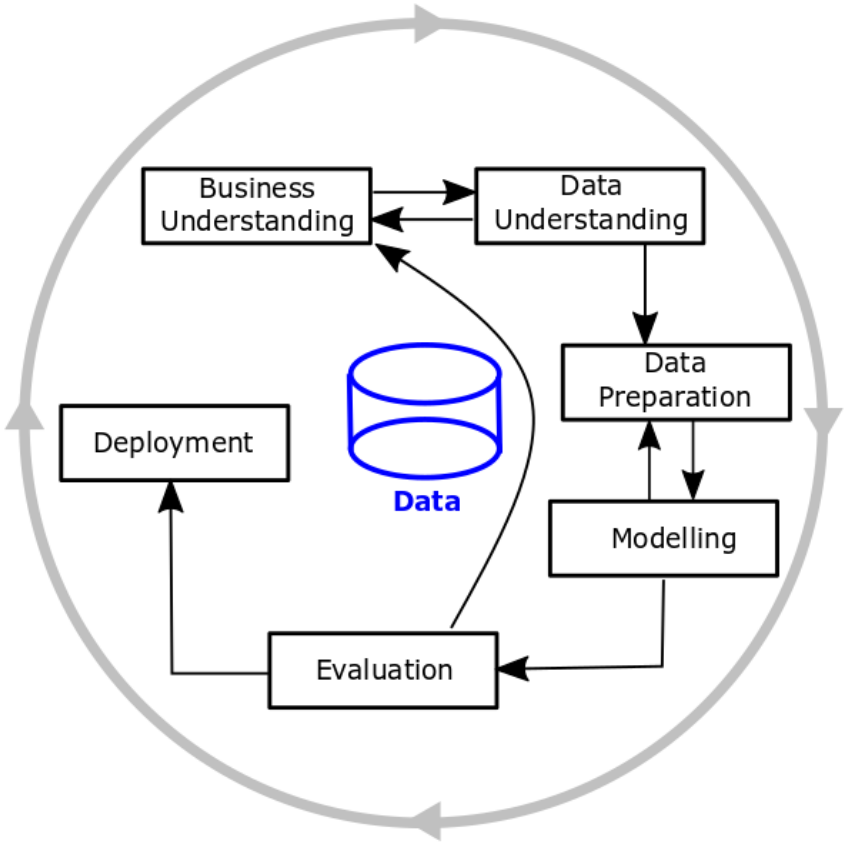


Abbildung 2.1.: CRISP-DM Modell nach Shearer [She00].

sechste Phase Deployment beschäftigt sich mit der Integration des entworfenen Lösungsansatzes in realen Umgebungen. Typische Aufgaben in dieser Phase sind das Entwerfen eines Bereitstellungsplans, Planüberwachung und Planwartung, die Erstellung des Abschlussberichts sowie die Überprüfung des Gesamtprojekts.

Übergeordnetes Ziel dieser Arbeit ist die Detektion von Angriffsszenarien in flowbasierten Netzwerkdaten. Somit ist das Business Understanding dieser Dissertation die Erkennung von Angriffsszenarien bei einer niedrigen Anzahl von Fehlmeldungen. Bei der Analyse zur Verfügung stehender Daten (Data Understanding Phase) handelt es sich um flowbasierte Netzwerkdaten (siehe Abschnitt 4.2.2). Die Datenvorverarbeitung (Data Preparation Phase) nimmt einen wesentlichen Teil dieser Dissertation ein. Dies wird unter anderem daraus ersichtlich, dass sich die zwei Kapitel 5 und 6 ausschließlich mit der Verarbeitung kategorischer Attribute beschäftigen. In der Modellierungsphase werden Verfahren zur Detektion von Angriffen verwendet. Während Kapitel 10 einen generellen Ansatz basierend auf dem CRISP-DM Modell entwirft, erfolgt in Kapitel 11 der konkrete Einsatz von Klassifikatoren und sequentiellen Hypothesentests zur Detektion von Port Scans. Auch die Evaluationsphase nimmt einen wesentlichen Teil der Dissertation ein. Zur Evaluierung werden gelabelte Testdaten beziehungsweise Testszenarien in Kapitel 8 und 9 entworfen. Die Deployment Phase würde die konkrete Integration der entwickelten Verfahren in Unternehmen umfassen. Da die Entwicklung eines produktiven Systems nicht der Fokus dieser Arbeit ist, findet die Integration und Anwendung der entwickelten Verfahren in experimentellen Testumgebungen statt.

2.3. Datentypen

Ein Datenstrom beziehungsweise ein Datensatz D besteht aus einer Menge von Datenpunkten. Jeder Datenpunkt $\boldsymbol{O_i} \in D$ stellt ein Objekt dar und besitzt bestimmte Eigenschaften. Diese Eigenschaften werden durch Merkmale, sogenannte Attribute, abgebildet. Ein Datenpunkt $\boldsymbol{O_i}$ sei durch einen m-dimensionalen Vektor $\boldsymbol{O_i} = (o_{i_1}, o_{i_2}, ..., o_{i_m})$ repräsentiert. Der Datentyp eines Attributs hängt von dessen Wertebereich ab. Eine Definition der vier wichtigsten Datentypen findet im Folgenden statt.

2.3.1. Kontinuierliche Attribute

Kontinuierliche Attribute sind quantitativ und können durch Zahlen (Ganzzahlen oder reelle Zahlen) gemessen und dargestellt werden [HKP11]. Beispiele für kontinuierliche Attribute stellen die Entfernung zweier Städte, das Gewicht einer Person, oder die Dauer einer Netzwerkverbindung dar. Vorteil kontinuierlicher Attribute ist, dass für deren Abstandsberechnung gängige Metriken wie der Euklidischer-Abstand verwendet werden können.

2.3.2. Kategorische Attribute

Kategorische Attribute beinhalten symbolische Werte (Buchstaben, Wörter, Zahlen und Sonderzeichen) und jeder Wert repräsentiert eine Art Zustand beziehungsweise eine bestimmte Ausprägung [HKP11]. Als Folge dessen besitzen kategorische Attribute keine natürliche Ordnung, was die Anwendung metrischer Abstandsmaße erschwert. Beispielsweise stellt die Haarfarbe einer Person ein kategorisches Attribut dar, welches die Ausprägungen *blond*, *braun*, *rot*, *grau*, *weiß* und *schwarz* annehmen kann. Im Kontext IT-Sicherheit stellt beispielsweise die IP-Adresse ein kategorisches Attribut dar. Oftmals werden kategorische Attribute auch als nominale Attribute bezeichnet.

2.3.3. Binäre Attribute

Binäre Attribute sind kategorische Attribute mit exakt zwei unterschiedlichen Ausprägungen [HKP11]. Beispielsweise stellt der einfache Test, ob eine Person verschuldet ist (*ja* oder *nein*), ein binäres Attribut dar. Ein weiteres Beispiel für ein binäres Attribut ist die Präsenz des TCP-Flags *SYN* in einem Netzwerkpaket. Aufgrund des eingeschränkten Wertebereiches können binäre Attribute leicht auf numerische Werte transformiert werden, um sinnvolle Berechnungen zwischen den Werten durchzuführen. Oftmals findet eine Transformation auf die Werte 0 und 1 statt.

2.3.4. Ordinale Attribute

Ordinalen Attributen liegt eine Ordnungsrelation zugrunde, auch wenn die Ausprägungen des Attributs kategorischer Natur sind [HKP11]. Ein Beispiel wäre die Größe eines Getränks (*klein*, *mittel*, *groß*). Ordinale Attribute lassen sich mithilfe von Domänenwissen auf einen kontinuierlichen Wertebereich transformieren.

2.4. Abstands- und Ähnlichkeitsmaße

Data Mining-Algorithmen benötigen häufig Abstandsmaße (beziehungsweise Ähnlichkeitsmaße), um den Abstand (beziehungsweise Ähnlichkeit) zwischen Datenpunkten zu berechnen. Dieser Abschnitt führt vier Maße ein, die im weiteren Verlauf der Arbeit Einsatz finden.

2.4.1. Minkowski-Abstand

Beim Minkowski-Abstand handelt es sich genau genommen nicht um ein konkretes Abstandsmaß, sondern um eine Generalisierung vieler Abstandsmaße. Der Minkowski-Abstand stellt eine Metrik dar und eignet sich zur Berechnung von Abständen zwischen Datenpunkten $\boldsymbol{O_i}$ und $\boldsymbol{O_j}$, welche durch kontinuierliche Attribute charakterisiert sind. Der Minkowski-Abstand ist wie in Gleichung 2.1 definiert (siehe auch [HKP11], [TSK06]).

$$d_{minkowski}(\boldsymbol{O_i}, \boldsymbol{O_j}) = \sqrt[q]{\sum_{l=1}^{m} |o_{i_l} - o_{j_l}|^q}, \tag{2.1}$$

wobei $\boldsymbol{O_i}$ und $\boldsymbol{O_j}$ zwei Datenpunkte sind und m die Anzahl der Attribute. Somit repräsentiert o_{i_l} den Wert des Datenpunkts $\boldsymbol{O_i}$ im Attribut l. Der Parameter q ist in der Regel eine natürliche Zahl. Wird der Parameter $q = 1$ gesetzt, so ergibt sich der Manhattan-Abstand oder für den Wert $q = 2$, ergibt sich der in Gleichung 2.2 dargestellte Euklidische-Abstand.

$$d_{euklid}(\boldsymbol{O_i}, \boldsymbol{O_j}) = \sqrt[2]{\sum_{l=1}^{m} |o_{i_l} - o_{j_l}|^2} \tag{2.2}$$

Wird der Parameter $q = \infty$ gesetzt, so ergibt sich die in Formel 2.3 dargestellte Maximumsnorm.

$$d_{maximum}(\boldsymbol{O_i}, \boldsymbol{O_j}) = \sqrt[\infty]{\sum_{l=1}^{m} |o_{i_l} - o_{j_l}|^\infty} \tag{2.3}$$

2.4.2. Hamming-Abstand

Der Hamming-Abstand ist ebenfalls eine Metrik und eignet sich zum Berechnen von Abständen zwischen Datenpunkten mit kategorischen Attributen, da lediglich die Gleichheit beziehungsweise Ungleichheit einzelner Werte berücksichtigt werden [HKP11]. Formell ist der Hamming-Abstand wie folgt definiert:

$$d_{hamming}(\boldsymbol{O_i}, \boldsymbol{O_j}) = \sum_{l=1}^{m} \begin{cases} 0 & \text{falls} \quad o_{i_l} = o_{j_l} \\ 1 & \text{falls} \quad o_{i_l} \neq o_{j_l} \end{cases}, \tag{2.4}$$

wobei o_{i_l} und o_{j_l} die Ausprägungen der Datenpunkte $\boldsymbol{O_i}$ und $\boldsymbol{O_j}$ im Attribut l sind und m die Anzahl der Attribute darstellt.

2.4.3. GRAPH

Coull et al. [CMB11] präsentieren ein Abstandsmaß zum Berechnen von Abständen zwischen IP-Adressen. Im Folgenden wird dieses Abstandsmaß als GRAPH bezeichnet.

GRAPH ist ein hierarchisches Abstandsmaß und unterscheidet auf erster Hierarchieebene IP-Adressen in die Kategorien *Unicast* und *Other*. *Unicast* IP-Adressen unterteilen sich in einer zweiten Hierarchiestufe in *Private* und *Public* IP-Adressen. Nicht *Unicast* IP-Adressen unterteilen sich in der zweiten Hierarchiestufe in vier Kategorien (*Multicast*, *Broadcast*, *Link Local* und *Default Network*). So ist beispielsweise der Abstand zwischen einer *Public* IP-Adresse und einer *Private* IP-Adresse $d(Private, Public) = 64$ oder der Abstand einer *Private* IP-Adresse zu einer *Multicast* IP-Adresse $d(Private, Multicast) = 128$. Der Abstand zwischen zwei IP-Adressen der gleichen Kategorie wird mithilfe des Hamming-Abstands (siehe Abschnitt 2.4.2) auf der 32-Bit Repräsentation von IPv4-Adressen berechnet. So besitzen etwa die IP-Adressen 192.168.100.7 und 192.168.100.8 den Abstand 4. Der Abstand zweier IP-Adressen liegt immer im Intervall $[0, 128]$. Abbildung 2.2 zeigt einen Überblick von GRAPH.

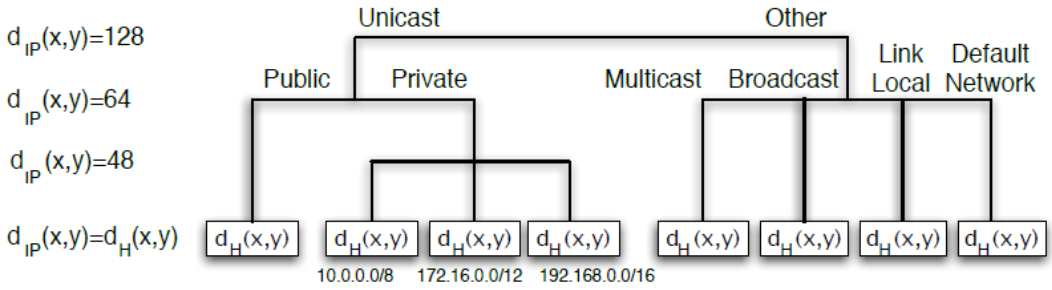


Abbildung 2.2.: Hierarchie des Abstandsmaßes GRAPH nach Coull et al. [CMB11].

2.4.4. Cosinus-Ähnlichkeit

Die Cosinus-Ähnlichkeit wird häufig im Bereich Text Mining und zur Berechnung von Ähnlichkeiten zwischen Vektoren verwendet [MSC+13]. Dieses Maß interpretiert die Datenpunkte als m-dimensionale Vektoren. Die Ähnlichkeit zweier Datenpunkte berechnet sich aus dem Cosinuswinkel der durch diese Datenpunkte aufgespannten Vektoren (siehe [HKP11] und [TSK06]). Eine mathematische Definition ist in Gleichung 2.5 zu sehen.

$$cos(\boldsymbol{O_i}, \boldsymbol{O_j}) = \frac{\boldsymbol{O_i} * \boldsymbol{O_j}}{||\boldsymbol{O_i}|| * ||\boldsymbol{O_j}||} = \frac{\sum_{l=1}^{m} o_{i_l} * o_{j_l}}{\sqrt{\sum_{l=1}^{m} (o_{i_l})^2} * \sqrt{\sum_{l=1}^{m} (o_{j_l})^2}}, \tag{2.5}$$

wobei o_{i_l} und o_{j_l} die Ausprägungen der Datenpunkte $\boldsymbol{O_i}$ und $\boldsymbol{O_j}$ im Attribut l sind und m die Anzahl der Attribute darstellt.

Die Cosinus-Ähnlichkeit $cos(\boldsymbol{O_i}, \boldsymbol{O_j})$ ist auf das Intervall $[-1, 1]$ normiert, wobei der Wert 1 die maximale und der Wert -1 die minimale Ähnlichkeit repräsentiert. Ein Wert von $cos(\boldsymbol{O_i}, \boldsymbol{O_j}) = 0$ würde bedeuten, dass die durch die Datenpunkte $\boldsymbol{O_i}$ und $\boldsymbol{O_j}$ aufgespannten Vektoren zueinander senkrecht stehen.

2.5. Evaluierungsmaßzahlen

In diesem Abschnitt werden verwendete Evaluierungsmaßzahlen für Klassifikations- und Clusterverfahren vorgestellt.

2.5.1. Zuweisungsmatrix

Die Zuweisungsmatrix (Assignment Matrix) wird häufig zur Ergebnisrepräsentation verwendet. Oftmals existieren sogenannte 2-Klassen-Probleme, das heißt, die Datenpunkte müssen zwei vordefinierten Klassen (*ja*, *nein*) zugeordnet werden. In dieser Arbeit findet diese Art der Klassifikation beispielsweise durch die Separierung von Netzwerkdaten in die zwei Klassen *Angriff* und *normal* statt. Eine derartige Zuweisungsmatrix ist in Anlehnung an [HKP11] und [WFH05] in Tabelle 2.1 dargestellt.

Tabelle 2.1.: Zuweisungsmatrix eines Zweiklassenproblems.

		vorhergesagte Klasse	
		Angriff	normal
tatsächliche Klasse	Angriff	True Positives (TP)	False Negatives (FN)
	normal	False Positives (FP)	True Negatives (TN)

In Tabelle 2.1 werden verschiedene Begriffe eingeführt, die nun definiert werden. TP (True Positives) ist die Anzahl der Datenpunkte, die der Klasse *Angriff* angehören und vom Klassifikator der Klasse *Angriff* zugeordnet werden. TN (True Negatives) ist die Anzahl der Datenpunkte, die der Klasse *normal* angehören und vom Klassifikator der Klasse *normal* zugeordnet werden. FP (False Positives) ist die Anzahl der Datenpunkte, die der Klasse *normal* angehören und vom Klassifikator der Klasse *Angriff* zugeordnet werden. FN (False Negatives) ist die Anzahl der Datenpunkte, die der Klasse *Angriff* angehören und vom Klassifikator der Klasse *normal* zugeordnet werden. Somit stellen TP und TN korrekt klassifizierte Datenpunkte dar, während FP und FN die Anzahl der Fehlentscheidungen repräsentieren.

Falls mehr als zwei Klassen existieren, kann die Zuweisungsmatrix, wie in Tabelle 2.2 dargestellt, erweitert werden [WFH05]. Korrekt klassifizierte Datenpunkte befinden sich in Tabelle 2.2 auf der Diagonalen (siehe Zellen mit grünen Hintergrund).

Tabelle 2.2.: Zuweisungsmatrix eines Mehrklassenproblems.

		vorhergesagte Klasse				
		$Klasse_1$	$Klasse_2$	...	$Klasse_d$	Summe
tatsächliche Klasse	$Klasse_1$	20	1	...	2	213
	$Klasse_2$	1	23	...	2	226
	...	1	0	...	3	171
	$Klasse_d$	1	1	...	14	106
	Summe	213	226	171	106	—

2.5.2. Klassifikationsgenauigkeit

Die Klassifikationsgenauigkeit (Accuracy) ist ein weit verbreitetes Evaluationsmaß im Bereich Klassifikation und gibt den prozentualen Anteil der korrekt klassifizierten Datenpunkte an. Gleichung 2.6 zeigt die Berechnung der Klassifikationsgenauigkeit für ein 2-Klassen-Problem nach [HKP11].

$$Klassifikationsgenauigkeit = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.6}$$

TP steht für True Positive, TN für True Negative, FP für False Positive und FN für False Negative (siehe Tabelle 2.1). Die Klassifikationsgenauigkeit ist auf das Intervall $[0, 1]$ normiert, wobei höhere Werte bessere Klassifikationsergebnisse kennzeichnen.

Falls es sich um ein Mehrklassenproblem handelt, ergibt sich die Klassifikationsgenauigkeit aus der Summe der korrekt klassifizierten Datenpunkte (siehe grün hinterlegte Zellen in Tabelle 2.2), dividiert durch die Anzahl aller Datenpunkte.

2.5.3. Kreuzvalidierung

Für die Evaluierung von Klassifikatoren wird ein Datensatz in einen Trainingsdatensatz und einen Evaluierungsdatensatz unterteilt. Der Trainingsdatensatz dient zum Trainieren des Klassifikators. Der Evaluierungsdatensatz dient zum Überprüfen des trainierten Klassifikators und sollte Datenpunkte beinhalten, die nicht im Trainingsdatensatz vorhanden sind, um zu evaluieren, ob der Klassifikator korrekt generalisieren kann und nicht nur die Trainingsdatenpunkte auswendig lernt.

Da die Unterteilung eines Datensatzes in zwei Teilmengen zu unerwünschten Nebeneffekten wie ungünstige Klassenverteilungen in den Teilmengen führen kann, wird in der Regel eine Kreuzvalidierung durchgeführt. Die k-fachen Kreuzvalidierung (k-fold cross validation) unterteilt den zur Verfügung stehenden Datensatz D in k disjunkte Teilmengen $D_1, D_2, ..., D_k$. Anschließend werden $k-1$ Teilmengen verwendet, um den Klassifikator zu trainieren und eine Teilmenge D_i als Evaluierungsdatensatz zurückgehalten. Dieser Vorgang wird insgesamt k mal wiederholt, sodass jede Teilmenge D_j einmal als Evaluierungsdatensatz und $k-1$ mal als Trainingsdatenmenge verwendet wird. Ein typischer Wert für k ist der Wert 10. In diesem Fall handelt sich um eine 10-fache Kreuzvalidierung (siehe [HKP11] und [Bis06]).

2.5.4. Homogenität

Homogenität [RH07] ist ein Gütemaß zur Bewertung von Clusterverfahren mithilfe von gelabelten Datenpunkten. Das Maß evaluiert, ob die Datenpunkte eines Clusters der gleichen Klasse angehören und basiert auf einem entropiebasierten Ansatz. $H(K)$ sei die Entropie der Klassenverteilung ($K = \{k_i|1, ..., u\}$) und ist wie in Gleichung 2.7 definiert.

$$H(K) = -\sum_{k=1}^{|K|} \frac{n_k}{n} \log \frac{n_k}{n} \tag{2.7}$$

In Gleichung 2.7 beschreibt der Parameter n die Anzahl der Datenpunkte und n_k die Anzahl der Datenpunkte der Klasse k. Des Weiteren sei $C = \{c_i|1, ..., w\}$ die Menge der erstellten Cluster. Die bedingte Entropie $H(K|C)$ berechnet sich wie in Gleichung 2.8 dargestellt.

$$H(K|C) = -\sum_{c=1}^{|C|} \sum_{k=1}^{|K|} \frac{n_{k,c}}{n} \log \frac{n_{k,c}}{n_c}, \tag{2.8}$$

wobei n die Anzahl der Datenpunkte und n_c die Anzahl der Datenpunkte in Cluster c darstellen. Der Parameter $n_{k,c}$ gibt die Anzahl der Datenpunkte an, die der Klasse k angehören und Cluster c zugeordnet wurden. Letztendlich berechnet sich die Homogenität wie in Gleichung 2.9 dargestellt.

$$\text{Homogenität} = \begin{cases} 1 & \text{falls} \quad H(K) = 0 \\ 1 - \frac{H(K|C)}{H(K)} & \text{sonst} \end{cases} \tag{2.9}$$

Die bedingte Entropie $H(K|C)$ ist stets kleiner gleich der Klassenentropie $H(K)$. Somit ist die Homogenität auf das Intervall $[0, 1]$ normiert. Wenn das Ergebnis der Clusteranalyse perfekt ist, nimmt die bedingte Entropie $H(K|C)$ den Wert 0 an und die Homogenität den Wert 1. Folglich weisen bessere Ergebnisse im Clustering höhere Werte im Maß Homogenität auf. Für den Fall, dass im Datensatz nur eine Klasse existiert, wird die Homogenität per Definition auf 1 gesetzt [RH07].

2.5.5. Vollständigkeit

Vollständigkeit [RH07] ist ein weiteres Gütemaß zur Bewertung von Clusterverfahren, wenn Klassenlabels verfügbar sind. Vollständigkeit ist als Ergänzung des Maßes Homogenität zu betrachten. Die Homogenität bewertet, ob alle Datenpunkte eines Clusters der gleichen Klasse angehören. Die Vollständigkeit hingegen bewertet, ob alle Datenpunktes einer Klasse dem gleichen Cluster angehören. Hierfür definiert sich die Vollständigkeit symmetrisch zur Homogenität. $H(C)$ sei die Entropie der Verteilung der Cluster ($C = \{c_i|1, ..., z\}$) und ist wie in Gleichung 2.10 definiert.

$$H(C) = -\sum_{c=1}^{|C|} \frac{n_c}{n} \log \frac{n_c}{n}, \tag{2.10}$$

wobei n die Anzahl der Datenpunkte und n_c die Anzahl der Datenpunkte des Clusters c ist. Die bedingte Entropie $H(C|K)$ der Verteilung der Cluster unter Berücksichtigung der Verteilung der Klassen K berechnet sich wie in Gleichung 2.11 dargestellt.

$$H(C|K) = -\sum_{k=1}^{|K|} \sum_{c=1}^{|C|} \frac{n_{k,c}}{n} \log \frac{n_{k,c}}{n_k}, \tag{2.11}$$

wobei n die Anzahl der Datenpunkte und n_k die Anzahl der Datenpunkte der Klasse k sind. Der Parameter $n_{c,k}$ beschreibt die Anzahl der Datenpunkte, die Klasse k angehören und Cluster c zugeordnet wurden.

Letztendlich berechnet sich die Vollständigkeit wie in Gleichung 2.12 dargestellt.

$$\text{Vollständigkeit} = \begin{cases} 1 & \text{falls} \quad H(C) = 0 \\ 1 - \frac{H(C|K)}{H(C)} & \text{sonst} \end{cases} \tag{2.12}$$

Genau wie die Homogenität ist die Vollständigkeit auf das Intervall $[0, 1]$ normiert, wobei höhere Werte bessere Ergebnisse im Clustering indizieren. Per Definition ist die Vollständigkeit 1, wenn nur ein Cluster existiert [RH07].

2.5.6. Normalized Mutual Information (NMI)

Normalized Mutual Information (NMI) [SG03] ist ein weiteres Maß zur Bewertung von Clusterverfahren und benötigt ebenfalls die Präsenz von Klassenlabels. In dieser Arbeit wird der in Strehl und Ghosh [SG03] definierte NMI verwendet:

$$NMI(K, C) = \frac{I(K, C)}{\sqrt{H(K) * H(C)}}, \tag{2.13}$$

wobei $H(K)$ die Entropie der Klassenverteilung (siehe Gleichung 2.7) und $H(C)$ die Entropie der Clusterverteilung (siehe Gleichung 2.10) sind. $I(K, C)$ ist wie folgt definiert:

$$I(K,C) = \sum_{k=1}^{|K|} \sum_{c=1}^{|C|} p(k,c) \log \frac{p(k,c)}{p(k)p(c)}, \tag{2.14}$$

wobei $p(k)$ die Wahrscheinlichkeit der Klasse k, $p(c)$ die Wahrscheinlichkeit des Clusters c und $p(k, c)$ die Wahrscheinlichkeit der Klasse k und des Clusters c repräsentiert. Der NMI ist auf das Intervall $[0, 1]$ normiert, wobei höhere Werte bessere Ergebnisse indizieren.

Im weiteren Verlauf der Arbeit werden die hier vorgestellten Evaluierungsmaßzahlen zur Bewertung der entwickelten Methoden genutzt. So nutzt beispielsweise Kapitel 5 die Klassifikationsgenauigkeit und den NMI zur Evaluierung des Abstandsmaßes ConDist. Die Evaluierung von IP2Vec (Kapitel 6) findet mithilfe der Evaluierungsmaße Zuweisungsmatrix, Homogenität und Vollständigkeit statt.

3. Methodische Grundlagen

Dieses Kapitel befasst sich mit den methodischen Grundlagen dieser Arbeit. Hierfür werden die verwendeten Klassifikatoren und Clusterverfahren kurz erklärt. Danach werden zwei spezielle Methoden, Word2Vec und Generative Adversarial Networks (GANs), erläutert, welche die Basis für die Weiterentwicklung spezifischer Verfahren in den Kapiteln 6 und 9 darstellen. Das Kapitel endet mit der Vorstellung der verwendeten Visualisierungsverfahren sowie das im Rahmen dieser Dissertation weiterentwickelte Coburg-Utililty-Framework (CUF).

3.1. Klassifikatoren

Im Folgenden soll die grundsätzliche Funktionsweise der in dieser Arbeit verwendeten Klassifikatoren kurz erläutert werden.

3.1.1. k-Nächster-Nachbar

Der k-Nächster-Nachbar Klassifikator (kNN) verfügt über keine explizite Trainingsphase und nutzt in der Klassifikationsphase die Datenpunkte der Trainingsdatenmenge [HKP11]. Der kNN Klassifikator wird in Kapitel 5 zur Evaluierung des Abstandsmaßes ConDist verwendet.

Angenommen, die Menge der Trainingsdatenpunkte sei D und es soll ein beliebiger als Vektor repräsentierter Datenpunkt $\boldsymbol{O_i}$ klassifiziert werden. In diesem Fall berechnet der kNN Klassifikator den Abstand $d(\boldsymbol{O_i}, \boldsymbol{O_j})$ zu allen Datenpunkten aus der Trainingsdatenmenge $\boldsymbol{O_j} \in D$. Anschließend werden die Datenpunkte der Trainingsdatenmenge D hinsichtlich ihrer Abstände zum Datenpunkt $\boldsymbol{O_i}$ aufsteigend sortiert. Danach werden die k Datenpunkte mit den geringsten Abständen aus der Trainingsdatenmenge D ausgewählt und hinsichtlich ihrer Klassenlabels analysiert. Der Datenpunkt $\boldsymbol{O_i}$ wird dann der Klasse zugewiesen, die in den k nächstgelegenen Nachbarn am häufigsten auftritt. Somit kommt dem verwendeten Abstandsmaß eine zentrale Bedeutung in der Klassifikationsphase zu. Der Parameter k stellt einen benutzerspezifischen Eingabeparameter dar [HKP11]. Die soeben beschriebene Vorgehensweise, als auch der Einfluss des Parameters k, soll mithilfe von Abbildung 3.1 graphisch illustriert werden.

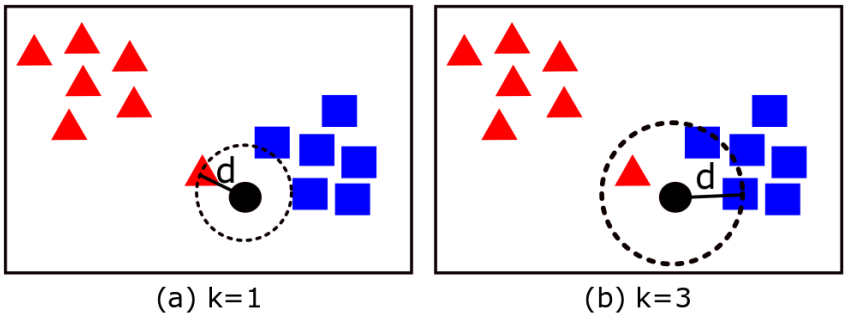


Abbildung 3.1.: Beispiel kNN Klassifikation mit Parameter $k = 1$ auf der linken Seite (a) und $k = 3$ auf der rechten Seite (b).

In Abbildung 3.1 ist die Klassifikation eines Datenpunktes mit zwei unterschiedlichen Werten für den Parameter k zu sehen. Die Datenpunkte aus der D sind in zwei Klassen (*rote Dreiecke* und *blaue Quadrate*) unterteilt. Der zu klassifizierte Datenpunkt ist als *schwarzer Kreis* dargestellt. In Abbildung 3.1 (a) wird der Parameter $k = 1$ gesetzt und es wird nur der nächstgelegene Nachbar berücksichtigt. In diesem Fall wird der Datenpunkt der Klasse der *roten Dreiecke* zugeordnet. In Abbildung 3.1 (b) wird der Parameter $k = 3$ gesetzt und die 3 nächsten Nachbarn ergeben sich aus zwei *blauen Quadraten* und einem *roten Dreieck*. Folglich wird in diesem Fall der Datenpunkt der Klasse der *blauen Quadrate* zugeordnet, da die Mehrheit der k nächstgelegenen Nachbarn dieser Klasse angehören. Für genauere Informationen über den kNN Klassifikator sei auf Han et al. [HKP11] verwiesen.

3.1.2. Entscheidungsbäume

3.1.2.1. Allgemeines

Entscheidungsbäume werden in Kapitel 11 zur Detektion von Port Scans verwendet. Entscheidungsbäume erstellen einen Baum zur Klassifikation neuer Datenpunkte und bestehen im Wesentlichen aus drei Elementen: innere Knoten, Blattknoten und Kanten. Der Wurzelknoten ist der Startpunkt des Klassifikators. Jeder innere Knoten hat einen Vaterknoten und kann beliebig viele Kindknoten besitzen. Die Blattknoten besitzen keine weiteren Kindknoten und weisen den Datenpunkten die Klassen zu. Innere Knoten überprüfen den Wert eines Attributs. Kanten stellen die unterschiedlichen Ausprägungen eines Attributs dar und verbinden die Knoten miteinander. Somit kann jeder Pfad vom Wurzelknoten zu einen beliebigen Blattknoten als Klassifikationsregel interpretiert werden [HKP11]. In der Trainingsphase wird der Entscheidungsbaum erlernt und in der Klassifikationsphase wird der Entscheidungsbaum zur Klassifikation neuer Datenpunkte genutzt.

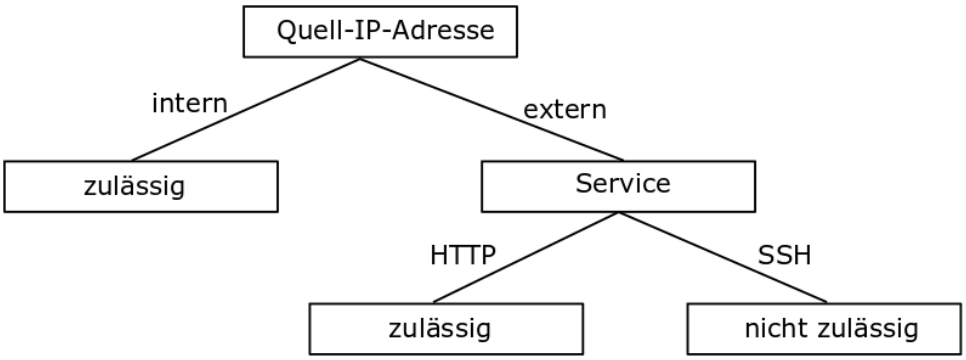


Abbildung 3.2.: Visualisierung eines Entscheidungsbaums.

Abbildung 3.2 zeigt einen Entscheidungsbaum, welcher Netzwerkverbindungen in zwei Klassen *zulässig* und *nicht zulässig* klassifiziert. Zunächst wird die Quell-IP-Adresse überprüft und in zwei Ausprägungen *intern* und *extern* unterschieden. Interne IP-Adressen werden stets der Klasse *zulässig* zugeordnet. Für externe IP-Adressen prüft der Entscheidungsbaum in einem zweiten Schritt das Attribut Service, um die Klassifikation durchzuführen. Handelt es sich beim angefragten Service um *HTTP*, so wird die Verbindung der Klasse *zulässig* zugeordnet. Handelt es sich beim angefragten Service um *SSH*, so wird die Verbindung der Klasse *nicht zulässig* zugeordnet.

3.1.2.2. Trainingsphase

In der Trainingsphase wird der Entscheidungsbaum mithilfe von gelabelten Datenpunkten erstellt. Beginnend beim Wurzelknoten wird das bestmögliche Split-Attribut ausgewählt und dem aktuellen Knoten zugewiesen, welches die Trainingsdatenmenge möglichst gut separiert. Die Auswahl des Split-Attributs erfolgt durch ein Entscheidungskriterium, welches im nächsten Abschnitt genauer definiert wird. Für jede mögliche Ausprägung des Attributs wird ein Kindknoten erstellt. Anschließend werden die Datenpunkte der Trainingsdatenmenge bezüglich ihrer Ausprägungen im ausgewählten Attribut unterteilt und den entsprechenden Kindknoten zugewiesen. Dieser Vorgang wird solange wiederholt, bis alle Trainingsdatenpunkte, die einem inneren Knoten zugeordnet wurden, der gleichen Klasse angehören oder bis alle zur Verfügung stehenden Attribute auf den aktuellen Pfad verwendet wurden. Trifft einer dieser beiden Fälle zu, ist der entsprechende Knoten ein sogenannter Blattknoten. Blattknoten führen keine weiteren Überprüfungen durch und weisen stattdessen den Datenpunkten Klassenlabels zu (siehe Abbildung 3.2). Wenn sich in einem Blattknoten Trainingsdatenpunkte mit unterschiedlichen Klassenlabels befinden, wird in der Regel eine Mehrheitsentscheidung vorgenommen.

Nach Erstellung des Entscheidungsbaums findet meist noch ein sogenannter Pruning Schritt statt. In diesem Schritt werden unnötige Zweige des Entscheidungsbaums entfernt, um die Komplexität zu reduzieren und eine höhere Generalisierung des Modells zu erreichen. Diese Generalisierung soll gegen Overfitting schützen und zu besseren Ergebnissen führen. Unter Overfitting wird im Allgemeinen die Überanpassung eines Modells auf die zugrundeliegenden Trainingsdaten verstanden [BL17]. Modelle, die unter Overfitting leiden, erreichen auf den Trainingsdaten sehr gute Ergebnisse, aber auf neuen unbekannten Daten wesentlich schlechtere Ergebnisse. Für weitere Informationen über Pruning sei auf Han et al. [HKP11] verwiesen.

3.1.2.3. Entscheidungskriterium

Das Entscheidungskriterium bestimmt die Auswahl des Split-Attributs. Hierbei wird die Auswahl möglichst diskriminativer Attribute in Bezug zu den Klassenlabels angestrebt, um möglichst kleine und kompakte Bäume zu erstellen. Bekannte Entscheidungsbaumverfahren wie *ID3* [Qui86], *C4.5* [Qui93] oder *CART* [BFO$^+$84] unterschieden sich primär durch das verwendete Entscheidungskriterium. Im Rahmen dieser Arbeit wurden Entscheidungsbaum-Klassifikatoren aus der Bibliothek *sklearn* genutzt, welche die Entscheidungskriterien *ID3* und *GINI-Index* bereitstellt. Da diese beiden Entscheidungskriterien zu keinen signifikanten Unterschieden in den Experimenten führten, wurde der *GINI-Index* ausgewählt, da dieser oftmals übersichtlichere Entscheidungsbäume erstellt. Deshalb wird im Folgenden der verwendete *Gini-Index* [BFO$^+$84] näher beschrieben. Der *Gini-Index* basiert auf der Wahrscheinlichkeit von Fehlklassifikationen. Zunächst wird die Fehlklassifikationswahrscheinlichkeit im Datensatz D mithilfe von Gleichung 3.1 berechnet.

$$Gini(D) = 1 - \sum_{i=1}^{|K|} p_i^2, \tag{3.1}$$

wobei $p_i = \frac{|K_{i,D}|}{|D|}$ die Wahrscheinlichkeit der Klasse i in D ist und durch Abzählen ermittelt wird. Die Anzahl der Klassen wird durch den Parameter $|K|$ festgelegt. Anschließend werden die Datenpunkte bezüglich ihrer Werte in Attribut X in zwei Teilmengen D_1 und D_2 aufgespalten und erneut die Wahrscheinlichkeit für Fehlklassifikation berechnet (siehe Gleichung 3.2). Eine Besonderheit des *Gini-Index* ist, dass dieser die Datenmenge stets in zwei Teilmengen D_1 und

D_2 aufteilt, auch wenn das Attribut X mehrere Ausprägungen besitzt. In diesen Fällen werden mehrere Werte pro Teilmenge zusammengefasst.

$$Gini_X(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2) \tag{3.2}$$

Letztlich ergibt sich der Gini-Index $\Delta Gini(D)$ wie in Gleichung 3.3 dargestellt.

$$\Delta Gini(D) = Gini(D) - Gini_X(D) \tag{3.3}$$

Ziel ist es hierbei, $\Delta Gini(D)$ zu maximieren, sodass die verbleibenden Teilmengen D_1 und D_2 die Datenpunkte bestmöglich bezüglich ihrer Klassenzuordnung aufteilen. Folglich wird das Attribut X ausgewählt, welches $\Delta Gini(D)$ maximiert.

3.1.3. Support Vektor Maschinen

3.1.3.1. Allgemeines

Erstmals wurden Support Vector Machinen (SVMs) 1992 von Vladimir Vapnik [BGV92] beschrieben. SVMs sind binäre Klassifikatoren, die linear separierbare sowie nicht linear separierbare Daten klassifizieren können. SVMs werden in Kapitel 11 zur Detektion von Port Scans verwendet. Dieser Abschnitt erläutert die wesentliche Idee von SVMs.

SVMs können nur Datenpunkte mit ausschließlich kontinuierlichen Attributen verarbeiten. Ziel einer SVM ist es, die Datenpunkte verschiedener Klassen anhand einer Hyperebene zu separieren. Im zwei-dimensionalen Raum ist die Hyperebene eine Gerade, im drei-dimensionalen Raum wäre diese eine Ebene und so weiter. Dabei wird die Hyperebene so gewählt, dass der Abstand zwischen den beiden Klassen maximal ist [HKP11]. Diese Vorgehensweise ist in Abbildung 3.3 mithilfe eines Beispiels dargestellt, indem sich die Datenpunkte in zwei Klassen (*gelbe Dreiecke* und *rote Kreise*) unterteilen lassen.

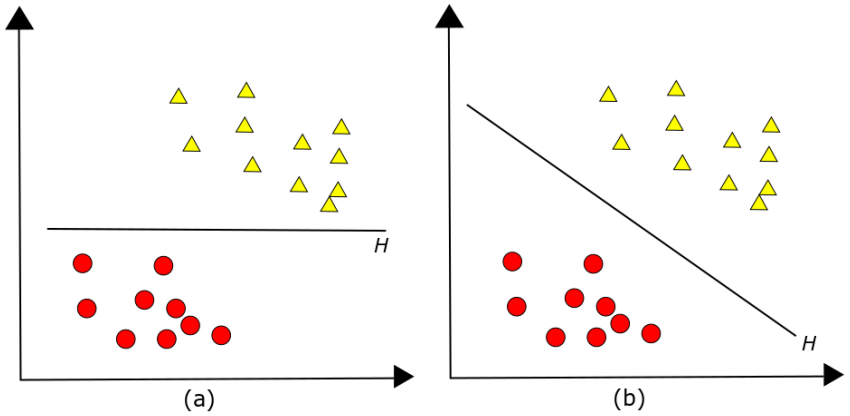


Abbildung 3.3.: Finden der optimalen Hyperebene H.

In Abbildung 3.3 sind zwei unterschiedliche Trenngeraden H eingezeichnet, welche die Datenpunkte in zwei Klassen unterteilen. SVMs suchen in der Trainingsphase die optimale Hyperebene, welche die gegebenen Trainingsdaten möglichst gut separiert und den Abstand zu den Datenpunkten der beiden Klassen maximiert. Die Hyperebene H in Abbildung 3.3(b) erfüllt diese Aufgabe besser als die Hyperebene H in Abbildung 3.3(a). SVMs nutzen in der Klassifikationsphase die Hyperebene um neue Datenpunkte den vordefinierten Klassen zuzuordnen. Im

obigen Beispiel werden Datenpunkte oberhalb der Hyperebene H der Klasse *gelbe Dreiecke* und unterhalb der Hyperebene H der Klasse *rote Kreise* zugeordnet.

3.1.3.2. Funktionsweise

Gegeben sei ein Trainingsdatensatz mit $(\boldsymbol{X_1}, y_1), (\boldsymbol{X_2}, y_2), ..., (\boldsymbol{X_p}, y_p), (\boldsymbol{X_q}, y_q)..., (\boldsymbol{X_m}, y_m)$, wobei $\boldsymbol{X_i}$ die Datenpunkte und y_i die Klassenlabels darstellen. Die Klassenlabels können die Werte -1 und $+1$ annehmen. SVMs beschreiben die Hyperebene H mithilfe einer Funktion der Form $f(\boldsymbol{X_i}) = \boldsymbol{W}\boldsymbol{X_i} + b$, wobei $\boldsymbol{W}$ ein Vektor und b ein Skalar ist [HKP11].

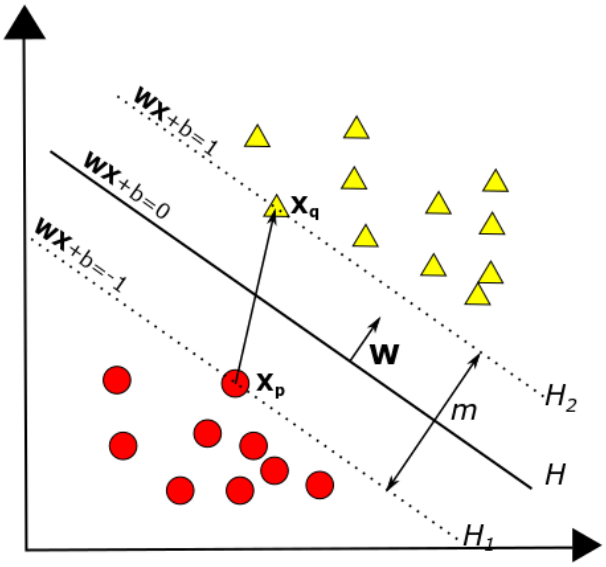


Abbildung 3.4.: Hyperebene, Stützhyperebenen und maximaler Abstand.

Stützhyperebenen (siehe H_1 und H_2 in Abbildung 3.4) sind Hyperebenen die parallel und mit gleichem Abstand zur Hyperebene H verlaufen [Ham09]. Für alle Datenpunkte $\boldsymbol{X_i}$, die oberhalb der Hyperebene H liegen, gilt: $\boldsymbol{W}\boldsymbol{X_i} + b \geq 1$. Für alle Datenpunkte $\boldsymbol{X_i}$, die unterhalb der Hyperebene H liegen, gilt: $\boldsymbol{W}\boldsymbol{X_i} + b \leq -1$. Durch die geschickte Wahl der Klassenlabels y_i können diese beiden Bedingungen wie in Gleichung 3.4 dargestellt zusammengefasst werden [HKP11].

$$y_i(\boldsymbol{W}\boldsymbol{X_i} + b) \geq 1 \quad \forall i \tag{3.4}$$

Datenpunkte, die auf den Stützhyperebenen liegen, werden als Support Vektoren bzw. Stützvektoren bezeichnet [HKP11]. Des Weiteren sei $\boldsymbol{W}$ ein Vektor der orthogonal zur Hyperebene H verläuft. Ziel ist es den Abstand m, auch Margin genannt, zwischen den Stützhyperebenen zu maximieren. Gleichung 3.5 zeigt die Berechnung des Abstands.

$$m = (\boldsymbol{X_q} - \boldsymbol{X_p})\frac{\boldsymbol{W}}{||\boldsymbol{W}||} = (\frac{1-b}{\boldsymbol{W}} - \frac{-1-b)}{\boldsymbol{W}})\frac{\boldsymbol{W}}{||\boldsymbol{W}||} = \frac{2}{||\boldsymbol{W}||} \tag{3.5}$$

Dies kann zu folgender Aufgabenstellung umgestellt werden:

$$\textit{minimiere } \frac{1}{2}||\boldsymbol{W}||^2 \text{ unter der Nebenbedingung } y_i(\boldsymbol{W}\boldsymbol{X_i} + b) \geq 1 \quad \forall i \tag{3.6}$$

SVMs lösen das Optimierungsproblem aus Gleichung 3.6 durch Anwendung der Lagrange'schen Optimierungstheorie. Eine wichtige Erkenntnis beim Lösen dieses Optimierungsproblems ist es, dass die optimale Hyperebene H ausschließlich vom Skalarprodukt der Support Vektoren abhängig ist [Ham09].

Lineare Funktionen sind häufig nicht ausreichend, um die Datenpunkte in ihrer ursprünglichen Dimension zu separieren. In diesen Fällen sehen SVMs die Einführung von sogenannten Slack-Variablen ξ_i vor, welche Fehler bei der Klassifikation erlauben. Diese Erweiterung wird im Allgemeinen als Soft-Margin bezeichnet [Ham09]. Die für Soft-Margins erweiterte Zielfunktion ist in Gleichung 3.7 abgebildet.

$$minimiere\ \frac{1}{2}||\boldsymbol{W}||^2 + C\sum_{i=1}^{m}\xi_i \ \text{ unter der Nebenbedingung } \ y_i(\boldsymbol{W}\boldsymbol{X_i} + b) \geq 1 - \xi_i \ \ \forall i, \quad (3.7)$$

wobei C eine Konstante größer gleich 0 ist und ξ_i die Slack-Variablen mit Wertebereich $\xi_i \geq 0$ darstellen. Für weitergehende Informationen sei an dieser Stelle auf [Ham09], [HKP11] und [SS01] verwiesen.

3.1.3.3. Kernel-Funktionen

Oftmals können die Datenpunkte im ursprünglichen Raum nicht linear voneinander getrennt werden. SVMs können dieses Problem durch die Verwendung von sogenannten Kernel-Funktionen lösen. Eine Kernel-Funktion transformiert Datenpunkte in hochdimensionale Datenräume, in denen die Datenpunkte wiederum linear separierbar sind. Da die Transformation aller Datenpunkte in hochdimensionale Datenräume sehr rechenaufwendig ist, wird ein sogenannter Kernel-Trick verwendet. Sei $\phi(\boldsymbol{X_i})$ die Transformation des Datenpunktes $\boldsymbol{X_i}$ in den neuen hochdimensionalen Datenraum. In diesem Fall besagt der Kernel-Trick, dass das Skalarprodukt zweier transformierter Datenpunkte $\phi(\boldsymbol{X_i}) \cdot \phi(\boldsymbol{X_j})$ durch eine Kernel-Funktion $K(\boldsymbol{X_i}, \boldsymbol{X_j})$ im ursprünglichen Datenraum berechnet werden kann. Der Kernel-Trick kann in SVMs angewendet werden, da das Ergebnis der Optimierung von Gleichung 3.6 bzw. 3.7 lediglich vom Skalarprodukt der Support Vektoren abhängig ist. [HKP11]

Im Rahmen dieser Arbeit wird der in Gleichung 3.8 dargestellte *RBF Kernel* (auch *Gaussian Kernel* genannt) verwendet.

$$K(\boldsymbol{X_i}, \boldsymbol{X_j}) = exp(-\frac{||\boldsymbol{X_i}, \boldsymbol{X_j}||^2}{2\sigma^2}), \quad (3.8)$$

wobei $\boldsymbol{X_i}$ und $\boldsymbol{X_j}$ zwei Datenpunkte im ursprünglichen Raum sind und σ ein benutzerspezifischer Parameter ist.

3.1.4. Neuronale Netze

Künstliche neuronale Netze versuchen, die Struktur des menschlichen Gehirns nachzubauen und bestehen aus einer Vielzahl von Neuronen, die miteinander verbunden sind [HKP11]. Die verwendeten Methoden in den Kapiteln 6 und 9 basieren auf neuronalen Netzen. Deshalb soll in diesem Abschnitt ein kurzer Einblick in den Aufbau und die Funktionsweise von neuronalen Netzen gegeben werden.

3.1.4.1. Aufbau

Abbildung 3.5 stellt den klassischen Aufbau eines neuronalen Netzes schematisch dar. Ein neuronales Netz besteht aus einer Eingabeschicht, einer Ausgabeschicht und beliebig vielen versteckten Schichten. Jede Schicht besteht aus einem oder mehreren Neuronen. Über die Eingabeschicht werden die Datenpunkte eingelesen und an der Ausgabeschicht kann das Ergebnis abgelesen werden. Die Eingabedaten dürfen ausschließlich kontinuierliche Attribute beinhalten. Die Neuronen der verschiedenen Schichten sind durch gewichteten Verbindungen verknüpft.

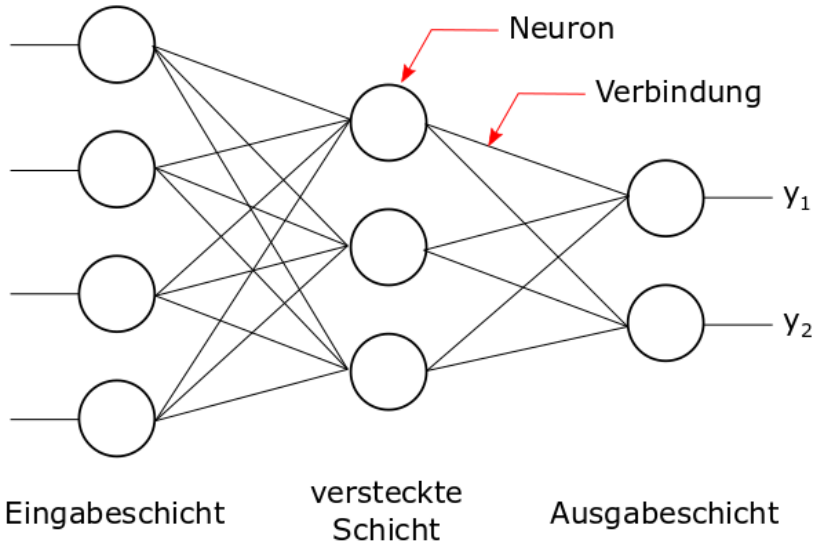


Abbildung 3.5.: Schematische Darstellung eines neuronalen Netzes.

Das neuronale Netz in Abbildung 3.5 besitzt vier Eingabeneuronen und zwei Ausgabewerte y_1 und y_2. In diesem Fall bestehen die zu verarbeitenden Datenpunkte aus vier Attributen $\boldsymbol{X_i} = (x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4})$. Somit könnte dieses neuronale Netz beispielsweise verwendet werden, um Datenpunkte in zwei vordefinierte Klassen $KlasseA$ und $KlasseB$ zu klassifizieren. In diesem Fall würde ein Datenpunkt der Klasse $KlasseA$ zugeordnet werden, wenn $y_1 > y_2$ oder Klasse $KlasseB$, wenn $y_1 < y_2$. In Abbildung 3.5 sind die wesentlichen Bestandteile (Neuronen und gewichtete Verbindungen) eines neuronalen Netzes gekennzeichnet.

3.1.4.2. Das Neuron

Ein neuronales Netz besteht aus einer Vielzahl von Neuronen. Abbildung 3.6 stellt den Aufbau eines einzelnen Neurons dar.

Die Eingabe eines Neurons ergibt sich aus der gewichteten Summe der Eingabewerte, die mit einem Bias addiert werden. Die Gewichte w_k repräsentieren die gewichteten Verbindungen aus Abbildung 3.5. Das Gewicht σ kontrolliert den Einfluss des Bias, welcher stets den Wert *1* besitzt. Der Eingabewert eines Neurons i berechnet sich wie in Gleichung 3.9 dargestellt.

$$neuron_{i_{eingabe}} = \sigma + \sum_{k=1}^{m} w_k \cdot x_k, \tag{3.9}$$

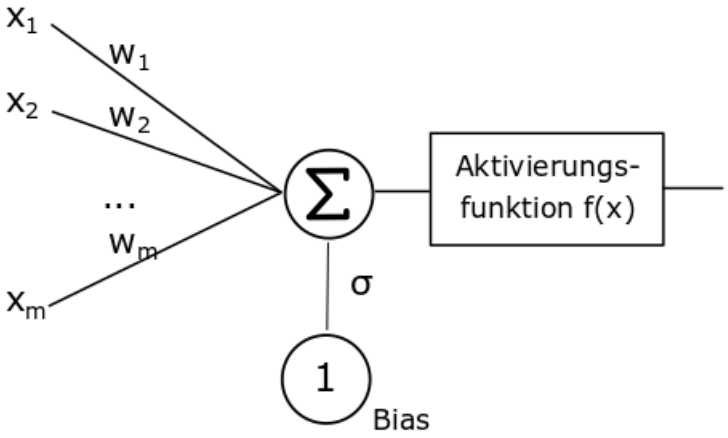


Abbildung 3.6.: Schematische Darstellung eines Neurons.

wobei m die Anzahl der Eingänge darstellt. Nachdem die Eingabe des Neurons i berechnet wurde, wird diese in eine Aktivierungsfunktion gegeben, welche die Ausgabe des Neurons i berechnet (siehe Gleichung 3.10). Danach wird der berechnete Wert an die nächste Schicht weitergeleitet.

$$neuron_{i_{ausgabe}} = f_i(x) = f_i(\sigma + \sum_{k=1}^{m} x_k \cdot w_k), \tag{3.10}$$

Im Folgenden werden die im Rahmen dieser Dissertation verwendeten Aktivierungsfunktionen $f(x)$ kurz definiert.

Linear. Die lineare Aktivierungsfunktion ist die einfachste Aktivierungsfunktion $f(x) = x$ und gibt die Eingabe unverändert weiter.

ReLU. ReLU (Restricted Linear Unit) ist eine einfache nicht lineare Aktivierungsfunktion und setzt negative Werte auf 0. ReLU ist wie folgt definiert: $f(x) = max(0, x)$.

Leaky ReLU. Leaky ReLU ist eine Variante von ReLU und setzt nicht alle negativen Werte auf 0. Stattdessen werden negative Werte mit einem benutzerdefinierten Wert α multipliziert. Leaky ReLU ist wie folgt definiert: $f(x) = max(\alpha x, x)$.

Sigmoid. Die Sigmoid Aktivierungsfunktion ist eine nicht-lineare Aktivierungsfunktion, welche das Ergebnis auf das Intervall $[0, 1]$ normiert. Sigmoid ist wie folgt definiert: $f(x) = \frac{1}{1+e^{-x}}$.

Softmax. Softmax Aktivierungsfunktionen werden in der Regel nur in der Ausgabeschicht verwendet und transformieren die Ausgabewerte in Wahrscheinlichkeiten, sodass die Summe aller Ausgaben 1 ergibt. Gleichung 3.11 zeigt die Berechnung für das Neuron i.

$$f_i(z) = \frac{exp(z_i)}{\sum_{j=1}^{q} exp(z_j)}, \tag{3.11}$$

wobei z_i dem Eingabewert des Neurons i (siehe Gleichung 3.9) entspricht und q die Anzahl der Ausgabeneuronen ist.

3.1.4.3. Training

Das Training eines neuronalen Netzes soll mithilfe eines Beispiels erläutert werden. In Abbildung 3.7 ist ein neuronales Netz mit sechs Neuronen zu sehen. Die Neuronen N_1, N_2 und N_3 sind in der Eingabeschicht, die Neuronen N_4 und N_5 in der versteckten Schicht und das Neuron N_6 in der Ausgabeschicht. Im ersten Schritt werden alle Gewichte w_{ij} des neuronalen Netzes

und die Bias mit Zufallswerten initialisiert. Danach wird das Netz mit den Trainingsdatenpunkten trainiert. Dabei beschreibt die Anzahl der Epochen e, wie oft jeder Datenpunkt aus dem Trainingsdatensatz zum Training verwendet werden soll.

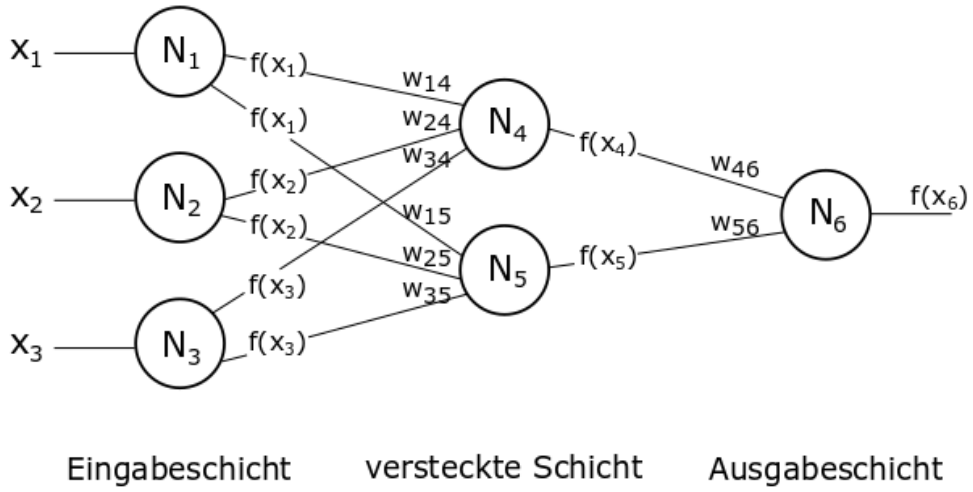


Abbildung 3.7.: Neuronales Netz mit eingezeichneten Gewichten und Aktivierungsfunktionen.

Vorwärtspfad. Im betrachteten Beispiel besitzt jeder Datenpunkt $\boldsymbol{X_i}$ drei Attribute x_1, x_2 und x_3, die an der Eingabeschicht angelegt werden. Anschließend werden die Ausgabewerte $(f(x_1), f(x_2), f(x_3))$ der Neuronen der Eingabeschicht (N_1, N_2, N_3) mithilfe von Gleichung 3.10 berechnet und an die Neuronen der ersten versteckten Schicht (N_4, N_5) weitergeleitet. Danach werden die Ausgabewerte der versteckten Schicht $(f(x_4), f(x_5))$ berechnet und an das Neuron der Ausgabeschicht (N_6) weitergeleitet. Die Eingangswerte der Neuronen der versteckten Schicht und der Ausgabeschicht ergeben sich aus der gewichteten Summe der Ausgabewerte der Neuronen aus der vorherigen Schicht. Am Ende dieses Vorgangs kann das Ergebnis des neuronalen Netzes $(f(x_6))$ an der Ausgabeschicht abgelesen werden und der Vorwärtspfad ist beendet.

Rückwärtspfad. Im Rückwärtspfad werden die berechneten Ergebnisse des Vorwärtspfads verwendet und mit den erwarteten Ergebnissen verglichen. Basierend auf diesen Vergleich werden die Gewichte des neuronalen Netzes mithilfe des Backpropagation Algorithmus angepasst [HKP11]. Backpropagation verwendet das Gradientenabstiegsverfahren, welches ein iteratives Verfahren zur Minimierung einer Zielfunktion ist [GBC18]. Die Zielfunktion wird in diesem Zusammenhang als sogenannte Fehlerfunktion (engl. Loss-Function) bezeichnet und berechnet den Fehler des neuronalen Netzes. Es existieren verschiedene Fehlerfunktion wie beispielsweise die mittlere quadratische Abweichung oder die absolute Differenz zwischen den berechneten und den erwarteten Werten.

Es wird angenommen, dass in diesem Beispiel die mittlere quadratische Abweichung L (siehe Gleichung 3.12) als Fehlerfunktion verwendet wird.

$$L = (y - f(x))^2, \tag{3.12}$$

wobei $f(x)$ die Ausgabe des neuronalen Netzes und y das erwartete Ergebnis darstellen. In Abbildung 3.7 entspricht $f(x) = f(x_6)$. Das Gradientenabstiegsverfahren verwendet die Ableitung der Fehlerfunktion zur Optimierung der Gewichte. Backpropagation beginnt mit der Optimierung der Gewichte w_{46} und w_{56} in der Ausgabeschicht und berechnet zunächst die Ableitungen

$\frac{dL}{dw_{46}}$ und $\frac{dL}{dw_{56}}$ der Fehlerfunktion L nach den Gewichten w_{46} und w_{56}. Anschließend passt das Gradientenabstiegsverfahren die Gewichte w_{46} und w_{56} wie in Gleichung 3.13 dargestellt an.

$$w_{ij} = w_{ij} - l\frac{dL}{dw_{ij}}, \tag{3.13}$$

wobei l die Lernrate des neuronalen Netzes ist und typischerweise im Intervall $[0, 1]$ liegt. Der berechnete Fehler in der Ausgabeschicht wird im Anschluss dazu verwendet, um den Fehler der vorletzten Schicht zu berechnen. Hierfür werden die Ableitungen ($\frac{dL}{dw_{14}}$, $\frac{dL}{dw_{24}}$, $\frac{dL}{dw_{34}}$, $\frac{dL}{dw_{15}}$, $\frac{dL}{dw_{25}}$, $\frac{dL}{dw_{35}}$) berechnet und die entsprechenden Gewichte w_{ij} nach Gleichung 3.13 angepasst. Dieser Vorgang wird solange iterativ wiederholt, bis die Ausgaben der Eingabeschicht erreicht sind. Für genauere Informationen zu Backpropagation sei auf [HKP11], [WFH05] und [RHW86] verwiesen.

3.2. Clusterverfahren

Im Folgenden werden die zwei Algorithmen WARD [WJ63] und DBScan [EKS+96] genauer vorgestellt, da diese in späteren Kapiteln eingesetzt werden.

3.2.1. WARD

WARD [WJ63] gehört zur Kategorie hierarchisch agglomerativer Clusterverfahren und wird in Kapitel 5 zur Evaluierung des Abstandsmaßes ConDist verwendet. Hierarchisch agglomerative Clusterverfahren arbeiten iterativ und interpretieren zu Beginn jeden Datenpunkt als eigenen Cluster. Anschließend werden die paarweisen Abstände aller Cluster berechnet und die zwei ähnlichsten Cluster zu einem größeren Cluster verschmolzen. Dieser Vorgang wird solange wiederholt, bis eine vom Benutzer definierte Anzahl von Clustern erreicht oder ein zuvor maximal definierter Abstand zwischen zwei Clustern überschritten wird. In Abbildung 3.8 ist der Vorgang einer Clusteranalyse von 7 Datenpunkten in einem sogenannten Dendrogramm dargestellt.

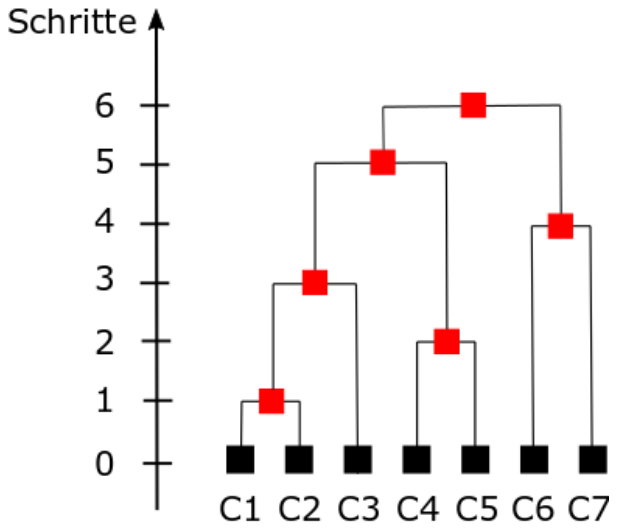


Abbildung 3.8.: Dendrogramm einer hierarchisch agglomerativen Clusteranalyse.

Zunächst stellt jeder Datenpunkt einen eigenen Cluster dar (siehe schwarze Quadrate in Abbildung 3.8). Danach werden iterativ die zwei ähnlichsten Cluster miteinander verbunden (siehe

rote Quadrate). Auf der y-Achse sind die Arbeitsschritte dargestellt. In Schritt 0 existieren 7 Cluster, in Schritt 1 werden die Cluster $C1$ und $C2$ miteinander verschmolzen und es existieren nur noch 6 Cluster. In Schritt 2 werden die Cluster $C4$ und $C5$ miteinander verschmolzen und es existieren noch 5 Cluster. Dieser Prozess wird solange wiederholt, bis in Schritt 6 nur noch ein Cluster existiert.

Der WARD Clustering Algorithmus basiert auf dem Kriterium der minimalen Varianz innerhalb der Cluster. Im ersten Schritt wird der paarweise Abstand aller Datenpunkte berechnet. Hierfür wird in der Regel der Euklidische-Abstand (siehe Abschnitt 2.4.1) verwendet. Zu Beginn wird jeder Datenpunkt als Cluster angesehen und somit existieren n Cluster. Angenommen, im ersten Schritt werden die Cluster $Cluster_i$ und $Cluster_j$ miteinander verbunden. Mithilfe der Aktualisierungsformel von Lance und Williams [LW67] kann der Abstand des neuen Clusters $Cluster_{ij}$ zu allen anderen Clustern $Cluster_k$ laut [ML14] wie folgt berechnet werden:

$$d_{(ij),k} = \frac{n_i + n_k}{n_i + n_j + n_k} d_{ik} + \frac{n_j + n_k}{n_i + n_j + n_k} d_{jk} - \frac{n_k}{n_i + n_j + n_k} d_{ij}, \tag{3.14}$$

wobei n_i die Anzahl der Datenpunkte in Cluster $Cluster_i$, n_j die Anzahl der Datenpunkte in Cluster $Cluster_j$, n_k die Anzahl der Datenpunkte in Cluster $Cluster_k$, d_{ij} der Abstand der Cluster $Cluster_i$ und $Cluster_j$, d_{ik} der Abstand der Cluster $Cluster_i$ und $Cluster_k$ und d_{jk} der Abstand der Cluster $Cluster_j$ und $Cluster_k$ ist. Gleichung 3.14 berechnet die Abstände zwischen den neuen Cluster $Cluster_{ij}$ zu allen anderen Cluster $Cluster_k$ und ermöglicht das anschließende Verbinden der zwei nächstgelegenen Cluster. Dieser Vorgang wird solange wiederholt, bis die vom Benutzer definierte Anzahl Cluster erreicht ist.

In Kapitel 5 wird das in Ring et al. [ROB+15] entwickelte Abstandsmaß ConDist zur initialen Berechnung der Abstandsmatrix verwendet.

3.2.2. DBScan

DBScan (Density-Based Spatial Clustering of Applications with Noise) [EKS+96] gehört zur Kategorie dichtebasierter Clusterverfahren und basiert auf der Idee, dass Cluster dichte Regionen im Datenraum darstellen und durch nicht dichte Regionen voneinander getrennt werden [HKP11]. Eine Region wird als dicht bezeichnet, wenn diese mehr als eine vordefinierte Anzahl von Datenpunkten beinhaltet. Alle Datenpunkte werden als Vektoren $\boldsymbol{O_i}$ repräsentiert. DBScan benötigt drei Eingabeparameter, den Datensatz D, die Größe der ϵ-Umgebung und die Mindestanzahl von Datenpunkten $minPts$ in einer ϵ-Umgebung, sodass diese als dicht bezeichnet werden kann. DBScan wird im Folgenden mithilfe des Pseudocodes in Algorithmus 1 erklärt.

Im ersten Schritt markiert DBScan alle Datenpunkte als unbesucht (Zeile 6). Anschließend wird iterativ für alle unbesuchten Datenpunkte $\boldsymbol{O_j}$ überprüft, ob in deren ϵ-Umgebung mindestens $minPts$ Datenpunkte liegen. Falls dies nicht der Fall ist, wird der Datenpunkt $\boldsymbol{O_j}$ zunächst als Rauschen markiert (Zeile 26). Sonst wird ein neuer Cluster $Cluster_i$ erstellt und der Datenpunkt $\boldsymbol{O_j}$ dem Cluster $Cluster_i$ zugeordnet (Zeile 11 bis 12). Der Cluster $Cluster_i$ wächst nun solange weiter, bis die Datenpunktdichte in der Nachbarschaft den vorgegeben Schwellwert $minPts$ unterschreitet. Hierfür werden zunächst alle Datenpunkte aus der ϵ-Umgebung von $\boldsymbol{O_j}$ einer Menge Ψ zugeordnet (Zeile 13). Für alle Datenpunkte $\boldsymbol{O_i} \in \Psi$ wird nun überprüft, ob in deren ϵ-Umgebung ebenfalls mindestens $minPts$ Datenpunkte liegen (Zeile 17). Falls dies der Fall ist, werden die Datenpunkte aus der entsprechenden ϵ-Umgebung von $\boldsymbol{O_i}$ der Menge Ψ hinzugefügt (Zeile 18). Anschließend werden alle Datenpunkte aus Ψ, die noch keinem Cluster zugeordnet sind, dem Cluster $Cluster_i$ zugewiesen (Zeile 21-23).

```
1  DBScan(D,ϵ,minPts)
2  // Eingabeparameter:
3  //    D - Datensatz
4  //    ϵ - Größe der ϵ-Umgebung
5  //    minPts - Mindestanzahl von Datenpunkten, dass eine ϵ-Umgebung als dicht
     bezeichnet wird
6  markiere alle Datenpunkte O_j ∈ D als unbesucht
7  while nicht alle Datenpunkte als besucht markiert do
8  |  wähle einen unbesuchten Datenpunkt O_j aus
9  |  markiere O_j als besucht
10 |  if ϵ-Umgebung von O_j beinhaltet mindestens minPts Datenpunkte then
11 |  |  erstelle neuen Cluster Cluster_i
12 |  |  füge O_j Cluster Cluster_i zu
13 |  |  Ψ sei die Menge aller Datenpunkte aus der ϵ-Umgebung von O_j
14 |  |  forall Datenpunkte O_i in Ψ do
15 |  |  |  if O_i ist unbesucht then
16 |  |  |  |  markiere O_i als besucht
17 |  |  |  |  if ϵ-Umgebung von O_i beinhaltet mindestens minPts Datenpunkte then
18 |  |  |  |  |  Füge diese Datenpunkte der Menge Ψ hinzu
19 |  |  |  |  end
20 |  |  |  end
21 |  |  |  if Datenpunkt O_i noch keinem Cluster zugeordnet then
22 |  |  |  |  Füge O_i Cluster Cluster_i zu
23 |  |  |  end
24 |  |  end
25 |  else
26 |  |  markiere O_j als Rauschen
27 |  end
28 end
```

Algorithmus 1: Pseudocode von DBScan in Anlehnung an Han et al. [HKP11].

Zur Berechnung, ob ein Datenpunkt $\boldsymbol{O_i}$ in der ϵ-Umgebung von $\boldsymbol{O_j}$ liegt, verwendet DBScan ein beliebiges Abstandsmaß. Ist der ermittelte Abstand kleiner als ϵ, so liegt der Datenpunkt $\boldsymbol{O_i}$ in der ϵ-Umgebung von $\boldsymbol{O_j}$. Folglich spielt der direkte Abstand zwischen zwei Datenpunkten eine sehr große Rolle in DBScan. Im Gegensatz zu WARD (siehe Abschnitt 3.2.1) werden nicht alle Datenpunkte zwingend einem Cluster zugewiesen. Aus diesen Gründen wird in Kapitel 6 DBScan zum Clustern erlernter Vektorrepräsentationen für IP-Adressen genutzt. Eine Weiterentwicklung von DBScan ist OPTICS (Ordering Points To Identify the Clustering Structure) [ABK+99], welcher auch in der Lage ist, Cluster mit unterschiedlicher Dichte zu generieren. Auf derartige Erweiterungen wird in Abschnitt 6.3 explizit verzichtet, da die absoluten Abstände der erlernten Vektorrepräsentationen untersucht werden sollen.

3.3. Word2Vec

Dieser Abschnitt erläutert die Methode Word2Vec [MCC+13, MSC+13]. Word2Vec ist eine Methode zur Transformation von Wörtern in kontinuierliche Vektorrepräsentationen, sogenannte Embeddings. Word2Vec gehört zur Kategorie unüberwachter Verfahren und nutzt einen großen Textkorpus zum Training. Die nachfolgende Beschreibung der Funktionsweise lehnt sich an [RLD+17] und [McC16] an.

3.3.1. Definition von Word2Vec

Word2Vec ist ein neuronales Netz (siehe Kapitel 3.1.4) mit einer versteckten Schicht. Die Architektur ist in Abbildung 3.9 graphisch dargestellt.

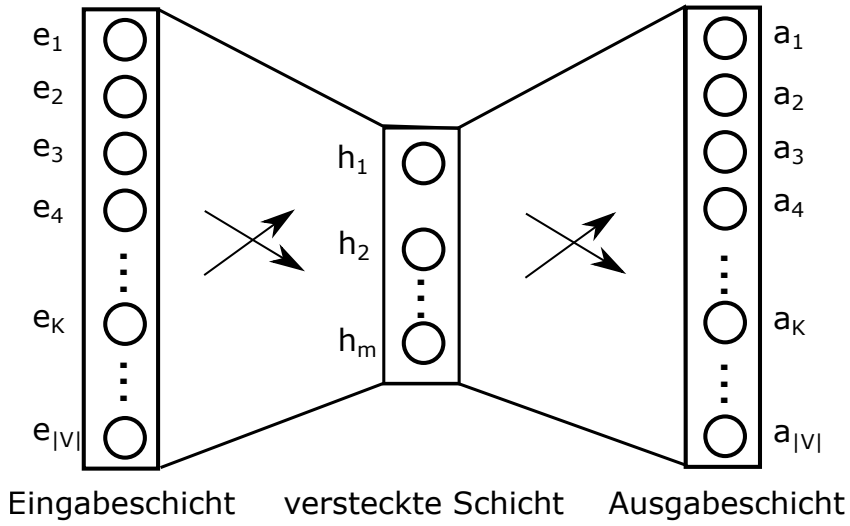


Abbildung 3.9.: Architektur von Word2Vec.

Die Anzahl der Neuronen in der versteckten Schicht ist viel kleiner als die Anzahl der Neuronen in der Eingabe- und Ausgabeschicht. Da neuronale Netze ausschließlich kontinuierliche Werte und keine Wörter als Eingabewerte verarbeiten können, findet zunächst eine Transformation in One-Hot-Vektoren statt. Die Länge des One-Hot-Vektors entspricht der Größe des Vokabulars $|V|$. Angenommen, das Vokabular des zugrundeliegenden Textkorpus umfasst 100.000 verschiedene Wörter, dann hat jeder One-Hot-Vektor 100.000 Dimensionen und für jedes Wort

ist eine Dimension 1, während alle anderen Dimensionen den Wert 0 aufweisen. Die Anzahl der Neuronen in der Eingabe- und Ausgabeschicht des neuronalen Netzes ist gleich der Größe des Vokabulars $|V|$. Somit lässt sich jedem Wort i des Vokabulars genau ein Neuron der Eingabeschicht e_i und ein Neuron der Ausgabeschicht a_i zuordnen. Die Ausgabeschicht des neuronalen Netzes verfügt über eine Softmax Aktivierungsfunktion, welche die Werte der Ausgabeneuronen auf Wahrscheinlichkeiten transformiert. Mikolov et al. [MSC+13] verwendeten 300 Neuronen in der versteckten Schicht.

3.3.2. Training

Word2Vec erhält in der Trainingsphase einen großen Textkorpus als Eingabe. Das folgende Beispiel zielt auf die Veranschaulichung der Trainingsphase. Angenommen, es liegt der Satz *"PCs sind immer häufiger Angriffen ausgesetzt"* als Trainingssatz vor. Zunächst selektiert Word2Vec ein sogenanntes Eingabewort des Satzes. Danach extrahiert Word2Vec mithilfe einer vordefinierten Fenstergröße umgebende Wörter, sogenannte Kontextwörter. Basierend auf den Eingabe- und Kontextwörtern erstellt Word2Vec die Trainingsbeispiele. Dieser Prozess ist in Tabelle 3.1 mit einer Fenstergröße von 2 graphisch dargestellt.

Tabelle 3.1.: Generierung von Trainingsbeispielen in Word2Vec. Eingabewörter sind durch türkisen Hintergrund hervorgehoben. Kontextwörter sind durch grauen Hintergrund hervorgehoben.

							Eingabewort	Kontextwort
PCs	sind	immer	häufiger	Angriffen	ausgesetzt	→	PCs	sind
							PCs	immer
PCs	sind	immer	häufiger	Angriffen	ausgesetzt	→	sind	PCs
							sind	immer
							sind	häufiger
PCs	sind	immer	häufiger	Angriffen	ausgesetzt	→	immer	PCs
							immer	sind
							immer	häufiger
							immer	Angriffen
PCs	sind	immer	häufiger	Angriffen	ausgesetzt	→	häufiger	sind
							häufiger	immer
							häufiger	Angriffen
							häufiger	ausgesetzt
PCs	sind	immer	häufiger	Angriffen	ausgesetzt	→	Angriffen	immer
							Angriffen	häufiger
							Angriffen	ausgesetzt
PCs	sind	immer	häufiger	Angriffen	ausgesetzt	→	ausgesetzt	häufiger
							ausgesetzt	Angriffen

Tabelle 3.1 zeigt die Generierung von Trainingsbeispielen aus einem Satz mit einer Fenstergröße von 2. Die Menge der Kontextwörter ist abhängig von der Fenstergröße. Eine Fenstergröße von 2 führt zur Aufnahme von vier Kontextwörtern, zwei Kontextwörter ergeben sich aus den linken Nachbarn und zwei aus den rechten Nachbarn des Eingabeworts. Das Eingabewort ist durch türkisen Hintergrund hervorgehoben und die zugehörigen Kontextwörter durch grauen Hintergrund. Word2Vec erstellt für das erste Wort *PCs* zwei Trainingspaare, in denen das Wort *PCs* die Rolle des Eingabeworts übernimmt. Die dazugehörigen Kontextwörter sind *sind* und *immer*. Für das zweite Wort *sind* entstehen drei Trainingspaare, in denen das Wort *sind* die Rolle Eingabeworts übernimmt. Dieser Vorgang wird für alle Wörter des Satzes wiederholt.

Für Word2Vec gibt es zwei unterschiedliche Ansätze, das CBOW Modell und das Skip-gram Modell. Beim CBOW Modell versucht Word2Vec das Eingabewort mithilfe des Kontextes zu erraten. Beim Skip-gram Modell versucht Word2Vec den Kontext mithilfe des Eingabeworts zu erraten. Diese Arbeit setzt auf das Skip-gram Modell. Das Skip-gram Modell legt das Eingabewort als One-Hot-Vektor an das neuronale Netz an und versucht, das entsprechende Kontextwort des jeweiligen Trainingsbeispiels vorherzusagen. Für die Trainingsbeispiele in Tabelle 3.1 ist die Wahrscheinlichkeit für das konkrete Kontextwort 1 und für alle anderen Wörter des Vokabulars 0. Folglich indiziert die Ausgabeschicht des neuronalen Netzes die Wahrscheinlichkeit für jedes Wort aus dem Vokabular, dass dieses in der Umgebung des Eingabeworts auftritt.

3.3.3. Negative Sampling

Negative Sampling ist eine Methode zur Reduzierung der Trainingszeit. Angenommen, das Vokabular besteht aus 100.000 Wörtern und das neuronale Netz verfügt über 300 Neuronen in der versteckten Schicht. Folglich hätte das neuronale Netz 60.000.000 Gewichte. Gleichzeitig ist das Training mit einen großen Textkorpus erforderlich, welcher Millionen bis Milliarden von Trainingsbeispielen beinhaltet. Eine Anpassung von 60.000.000 Gewichten pro Trainingsbeispiel würde eine sehr lange Zeit benötigen. Aus diesem Grund hat Mikolov et al. die Methode Negative Sampling eingeführt [MSC+13]. Negative Sampling aktualisiert für jedes Trainingsbeispiel nur einen kleinen Teil der Gewichte. Anstatt alle Gewichte des neuronalen Netzes zu aktualisieren, passt Negative Sampling nur die Gewichte des korrekten Ausgabeneurons und die Gewichte von ein paar zufällig ausgewählten Ausgabeneuronen an. Für genauere Informationen zu Negative Sampling sei auf [MSC+13] verwiesen.

3.3.4. Ermittlung der Vektorrepräsentationen

Nach erfolgreicher Trainingsphase steht die ursprüngliche Trainingsaufgabe nicht mehr im Vordergrund. Stattdessen erstellt Word2Vec die kontinuierlichen Vektorrepräsentationen (Embeddings) der Wörter aus den Gewichten der versteckten Schicht des neuronalen Netzes. Die Vektorrepräsentation eines Worts i entspricht somit den Gewichten zwischen den entsprechenden Eingangsneuron e_i und der versteckten Schicht. Die Dimensionalität der Vektorrepräsentationen ergibt sich somit aus der Anzahl Neuronen in der versteckten Schicht.

Derartige Vorgehensweisen finden häufig im Bereich des unüberwachten maschinellen Lernens statt. Beispielsweise verwenden Autoencoder ebenfalls neuronale Netze, in denen die versteckte Schicht viel weniger Neuronen als die Eingabe- und Ausgabeschicht besitzen, um so eine Generalisierung der Eingabedaten zu erhalten.

3.4. GANs

GANs (Generative Adversarial Networks) [GPM+14] sind generative Modelle zum Erstellen synthetischer Daten und finden in Kapitel 9 ihren Einsatz zur Generierung flowbasierter Netzwerkdaten.

GANs bestehen aus zwei neuronalen Netzen und basieren auf einem spieltheoretischen Ansatz [GBC18], in dem ein Generator-Netzwerk (Generator) gegen ein Diskriminator-Netzwerk (Diskriminator) antreten muss. In diesem Szenario versucht der Diskriminator, echte Datenpunkte von synthetisch generierten Datenpunkten zu unterscheiden. Gleichzeitig versucht der Generator möglichst realistische Datenpunkte zu generieren, sodass der Diskriminator diese nicht mehr von realen Datenpunkten unterscheiden kann. Dieses Spiel zwischen Generator und Diskriminator hat zur Folge, dass beide Spieler im Lauf der Zeit ihre eigenen Leistungen steigern. Dieser iterative Vorgang endet, sobald der Diskriminator nicht mehr in der Lage ist, reale Datenpunkte von synthetisch erzeugten Datenpunkten zu unterschieden. Die Architektur und der spieltheoretische Ansatz ist in Abbildung 3.10 graphisch dargestellt.

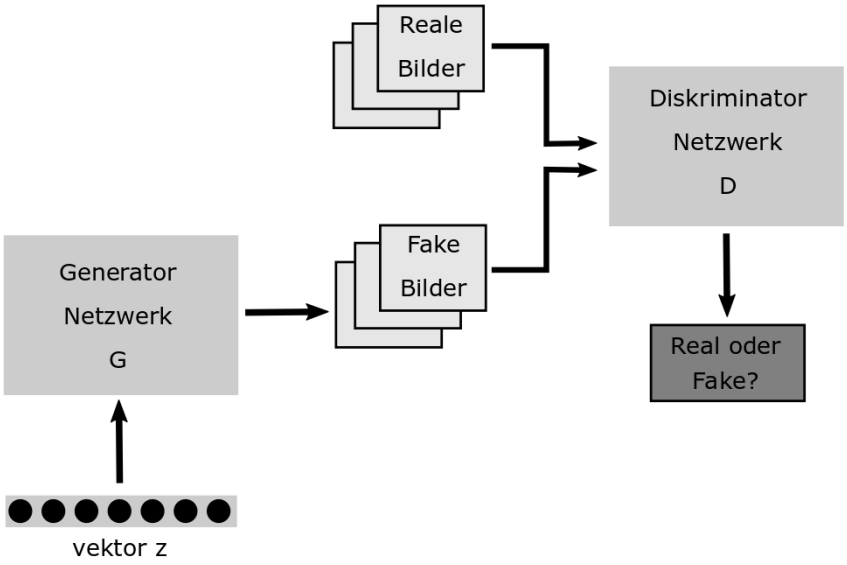


Abbildung 3.10.: Architektur eines GANs.

Im einfachsten Fall stellt jeweils ein neuronales Netz (siehe Abschnitt 3.1.4) den Generator und den Diskriminator dar. Ziel des Generators ist das Erlernen der Verteilung p_g über die Daten $\boldsymbol{x}$. Der Generator bekommt einen Vektor von einer beliebigen Verteilung $p_z(\boldsymbol{z})$ als Eingabe. Der Diskriminator berechnet die Wahrscheinlichkeit, dass die Eingabe aus den realen Daten stammt und kein synthetisches Beispiel des Generators darstellt. Der Diskriminator wird stets so trainiert, dass reale Datenpunkte das Label 1 und synthetisch erzeugte Datenpunkte das Label 0 erhalten. Gleichzeitig wird der Generator so trainiert, dass von $G(\boldsymbol{Z_i})$ erzeugte Datenpunkte vom Diskriminator den Wert 1 erhalten. Spieltheoretisch betrachtet stellen GANs ein Minimax-Spiel zwischen den Generator G und Diskriminator D mit der Optimierungsfunktion $V(G, D)$ (siehe Gleichung 3.15) dar [GPM+14].

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]. \tag{3.15}$$

Goodfellow et al. [GPM+14] zeigen in ihrer Arbeit, dass das globale Optimum von Gleichung 3.15 bei $p_g = p_{data}$ liegt. In diesem Fall würde der Generator die ursprüngliche Verteilung perfekt widerspiegeln und realistische Daten generieren.

1 GAN($m, q, schritte$)
2 // Eingabeparameter:
3 // m - Größe eines Batches
4 // q - Anzahl wie oft der Diskriminator pro Schritt trainiert wird
5 // *schritte* - Anzahl der Trainingsschritte
6 $i = 0$
7 **while** $i <$ *schritte* **do**
8 $j = 0$
9 **while** $j < q$ **do**
10 Erstelle Batch aus m Eingabedaten $\boldsymbol{Z_1}, \boldsymbol{Z_2}, ..., \boldsymbol{Z_m}$ aus $p_g(\boldsymbol{z})$.
11 Erstelle Batch aus m realen Datenpunkten $\boldsymbol{X_1}, \boldsymbol{X_2}, ..., \boldsymbol{X_m}$ aus den realen Trainingsdaten $p_{data}(\boldsymbol{x})$.
12 Aktualisiere Diskriminator D durch Erhöhung des stochastischen Gradienten wie folgt:
13 $$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{X_i}\right) + \log\left(1 - D\left(G\left(\boldsymbol{Z_i}\right)\right)\right)\right].$$
14 $j = j + 1$
15 **end**
16 Erstelle Batch aus m Eingabedaten $\boldsymbol{Z_1}, \boldsymbol{Z_2}, ..., \boldsymbol{Z_m}$ aus $p_g(\boldsymbol{z})$.
17 Aktualisiere Generator G durch Verringern des stochastischen Gradienten wie folgt:
18 $$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}_i\right)\right)\right).$$
19 $i = i + 1$
20 **end**

Algorithmus 2: Training eines GANs nach Goodfellow et al. [GPM+14].

In Algorithmus 2 ist das Training eines GANs in Pseudocode dargestellt. Zur Optimierung der neuronalen Netze schlagen Goodfellow et al. [GPM+14] das stochastische Gradientenverfahren vor (siehe Zeile 12 und 17). Das Gradientenverfahren ist ein iteratives Verfahren und strebt die Minimierung einer Zielfunktion an [GBC18]. Dafür berechnet das Gradientenverfahren die Ableitung der Zielfunktion und führt eine entsprechende Parameteranpassung durch (siehe Abschnitt 3.1.4). Das stochastische Gradientenverfahren ist eine Erweiterung des Gradientenverfahrens, indem es die tatsächlichen Gradienten schätzt und somit zu einer Laufzeitoptimierung führt [GBC18].

Das GAN erhält drei Übergabeparameter, die Größe eines Batches m, die Anzahl der Trainingsschritte für den Diskriminator q und die Anzahl der allgemeinen Trainingsschritte *schritte*. Die Größe des Batches m bestimmt, wie viele Datenpunkte im neuronalen Netz verarbeitet werden, bevor die Optimierungsfunktion die Parameter anpasst.

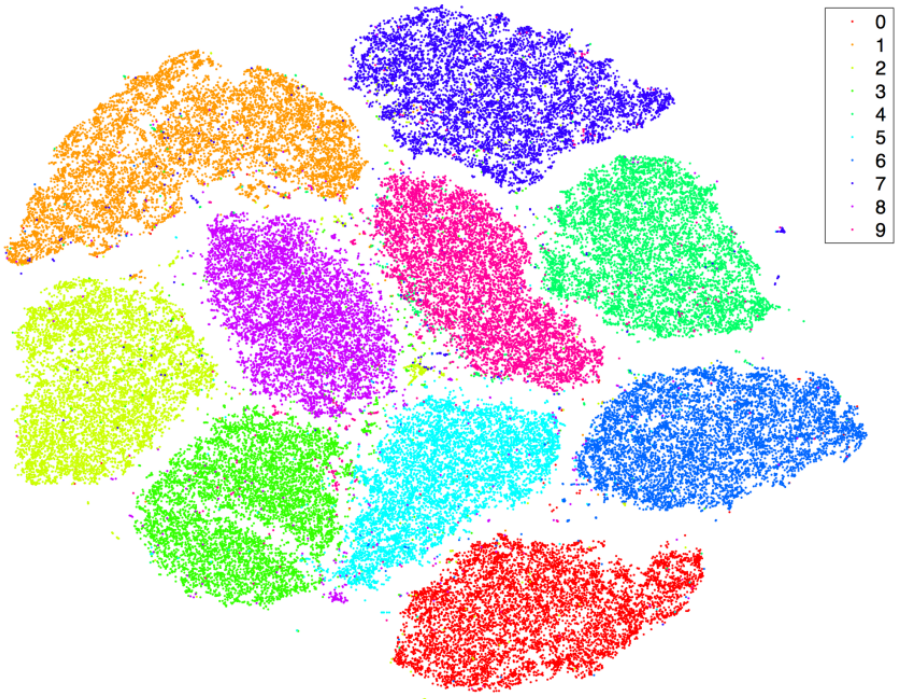


Abbildung 3.11.: t-SNE Visualisierung des MNIST Datensatzes [Maa14]. Die unterschiedlichen Farben indizieren die unterschiedlichen Klassen des Datensatzes.

GANs trainieren den Diskriminator und Generator iterativ. Jeder Trainingsschritt beginnt mit dem Training des Diskriminators D (Zeile 9 bis 15). Der Diskriminator wird pro Trainingsschritt q mal trainiert. Der Trainingsprozess erstellt zunächst einen Batch aus synthetisch erzeugten Daten (Zeile 10) und einen Batch aus realen Daten (Zeile 11). Anschließend lernt der Diskriminator mithilfe dieser beiden Batches und aktualisiert, wie in Zeile 13 dargestellt, die Gewichte des neuronalen Netzes. Im Anschluss findet das Training des Generators G statt. Hierfür erstellt der Trainingsprozess wiederum einen Batch aus synthetisch erzeugten Daten (Zeile 16) und bewertet diesen mithilfe des Diskriminators D. Basierend auf der Bewertung des Diskriminators findet eine Anpassung der Gewichte des Generators G statt, wie in Zeile 18 dargestellt.

3.5. Visualisierung

3.5.1. t-SNE

t-SNE (t-Distributed Stochastic Neighbor Embedding) [MH08] ist eine nicht-lineare Methode zur Dimensionsreduzierung. Die Methode transformiert hochdimensionale Objekte in zwei- oder drei-dimensionale Datenpunkte, sodass ähnliche Objekte hohe Ähnlichkeiten und verschiedene Objekte geringe Ähnlichkeiten in den transformierten Datenpunkten aufweisen.

t-SNE ist ein zweistufiges Verfahren. Im ersten Schritt berechnet t-SNE die Abstände zwischen den hochdimensionalen Objekten und transformiert diese in Wahrscheinlichkeiten p_{ij}, sodass niedrige Abstände zu hohen Wahrscheinlichkeiten und große Abstände zu geringen Wahrscheinlichkeiten führen. Der zweite Schritt ist iterativ. Hier berechnet t-SNE zunächst die Abstände zwischen den niedrigdimensionalen Datenpunkten und transformiert diese ebenfalls in Wahr-

scheinlichkeiten q_{ij}. Anschließend minimiert t-SNE die Kullback-Leibler Divergenz zwischen den Wahrscheinlichkeitsverteilungen im ursprünglichen und transformierten Raum.

Nach der Transformation können die berechneten Datenpunkte im zwei- oder dreidimensionalen Koordinatensystem bezüglich ihrer Ähnlichkeiten visualisiert werden. Im Gegensatz zur Hauptkomponentenanalyse stellen die neu berechneten Dimensionen der Objekte keine Linearkombinationen der ursprünglichen Attribute dar. Folglich ermöglichen die Koordinaten der Objekte keine direkten Rückschlüsse auf konkrete Werte in den ursprünglichen Dimensionen.

In Abbildung 3.11 ist die Visualisierung der nach t-SNE transformierten Datenpunkte des MNIST Datensatzes zu sehen. Der MNIST Datensatz beinhaltet Bilder von handgeschriebenen Zahlen (0 bis 9). MNIST kann in 10 Klassen unterteilt werden, wobei jede handgeschriebene Zahl eine Klasse darstellt. Jedes Bild besitzt eine Größe von 28x28 Pixeln und kann somit als 784-dimensionaler Datenpunkt interpretiert werden. Abbildung 3.11 visualisiert jedes Bild als zweidimensionalen Datenpunkt. Die Farben der Datenpunkte geben Auskunft über die entsprechenden Klassenzugehörigkeiten. In Abbildung 3.11 lassen sich Datenpunkte der gleichen Klasse visuell gruppieren. Folglich kann t-SNE die Ähnlichkeiten der 784-dimensionalen Datenpunkte (bezüglich ihrer Klassenzuordnung) im zwei-dimensionalen Raum gut widerspiegeln.

Viele Arbeiten [BM17, KHB14, LCH+16] nutzen t-SNE zur Visualisierung von Wort-Embeddings. Somit hat sich t-SNE in den letzten Jahren als beliebte Visualisierungstechnik für Embeddings entwickelt und wird auch in Kapitel 6 zur Visualisierung von IP-Adressen verwendet.

3.5.2. Violin-Plots

Violin-Plots eignen sich sehr gut zum Visualisieren von Verteilungen und ähneln Box-Plots. Beide Visualisierungsverfahren ermöglichen das Ablesen von Minimum, Maximum oder Median. Im Vergleich zu Box-Plots repräsentiert die Breite der Linien bei Violin-Plots die Dichte der Wahrscheinlichkeitsverteilung. Je breiter die Linie an einer Position, desto höher ist die Verteilungsdichte. Die Breite der Linien stellt somit ein geglättetes Histogramm dar. Der Wertebereich der Verteilungen wird auf der y-Achse aufgetragen.

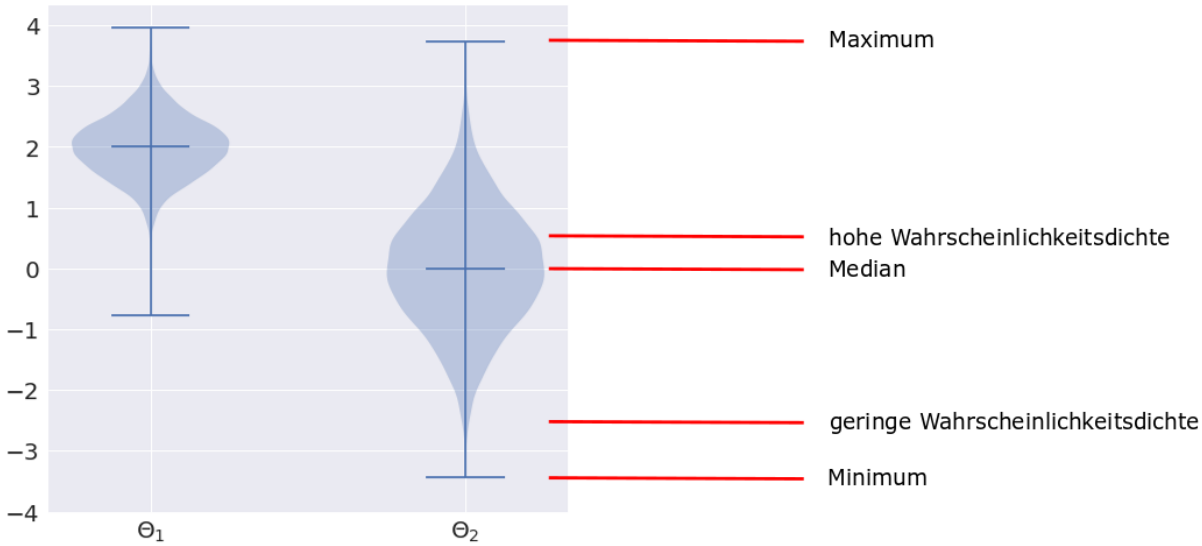


Abbildung 3.12.: Visualisierung von zwei Normalverteilungen durch Violin-Plots.

In Abbildung 3.12 sind zwei Normalverteilungen Θ_1 mit $N(\mu = 2, \sigma = 0{,}5)$ und Θ_2 mit $N(\mu = 0, \sigma = 1)$ durch Violin-Plots dargestellt. Für Θ_2 ist das Minimum, Median, Maximum sowie ein beliebiger Punkt mit niedriger und hoher Wahrscheinlichkeitsdichte in Abbildung 3.12 gekennzeichnet.

Kapitel 9 verwendet Violin-Plots, um die Verteilungen von realen und generierten flowbasierten Netzwerkdaten zu vergleichen.

3.6. Coburg-Utility-Framework

Diese Arbeit strebt die Detektion sicherheitskritischer Ereignisse in flowbasierten Netzwerkdaten durch Anwendung und Kombination verschiedener Data Mining-Methoden an. Häufig lassen sich geeignete Arbeitsabläufe und Konfigurationen nur durch experimentelles Testen ermitteln. Daher ist eine flexible Entwicklungsumgebung erforderlich, welche die leichte Evaluation unterschiedlicher Methoden und Konfigurationen ermöglicht. Diese Entwicklungsumgebung muss eine breite Auswahl von Tools zum Modellieren, Visualisieren und Evaluieren aufweisen. Das Coburg-Utility-Framework (CUF) stellt ein derartiges Unterstützungswerkzeug dar. Im Rahmen dieser Dissertation wurde CUF erheblich weiterentwickelt und mit neuen Funktionalitäten versehen. Beispielsweise wurde das heterogene Abstandsmaß ConDist (siehe Kapitel 5) oder auch die neuartige Vorverarbeitungskette von Netzwerk-Datenpunkten zur Detektion von Port Scans (siehe Kapitel 11) im CUF implementiert und evaluiert. Der nächste Abschnitt erläutert die grundlegende Architektur von CUF. Abschnitt 3.6.2 gibt einen Überblick der verfügbaren Algorithmen zur Online- und Offlineanalyse.

Die nachfolgenden Inhalte wurden bereits in der Publikation „*A Toolset for Intrusion and Insider Threat Detection*“ [RWG+17a] veröffentlicht.

3.6.1. Architektur

CUF basiert auf einer Pipes-and-Filter-Architektur [RWG+17a]. Pipes-and-Filter-Architekturen eignen sich sehr gut zur Verarbeitung von Datenströmen [BMR+96]. Filter stellen dabei unabhängige Verarbeitungseinheiten dar und manipulieren die Eingabedaten. Pipes sind für die Übertragung der Daten zwischen den Filtern zuständig. Somit können Filter und Pipes als Dienste mit gemeinsamen Schnittstellen realisiert und voneinander gekapselt werden. Die Pipes-and-Filter-Architektur erlaubt das einfache Einbinden neuer Filter, da beispielsweise die Integration eines neuen Klassifikators einen neuen Filter darstellt und alle existierenden Filter und Pipes unverändert bleiben.

CUF erweitert die ursprüngliche Pipes-and-Filter-Architektur dahingehend, dass Datenströme aufgeteilt und wieder zusammengeführt werden können. Dies erlaubt die gleichzeitige Evaluation verschiedener Instanzen des gleichen Filters mit unterschiedlichen Parameterkonfigurationen in einem einzigen Durchlauf. Da jeder dieser Verarbeitungsschritte als separater Filter in CUF implementiert ist, können verschiedene Arbeitsabläufe einfach konfiguriert und ausgeführt werden. Abbildung 3.13 zeigt einen Arbeitsablauf zur Verarbeitung flowbasierter Netzwerkdaten im CUF.

Der in Abbildung 3.13 dargestellte Arbeitsablauf verwendet zunächst einen Eingabefilter zum Einlesen der Daten aus einer CSV-Datei. Danach wird ein Vorverarbeitungsfilter eingesetzt, welcher den Zeitstempel eines Flows in Jahr, Monat, Tag, Stunde, Minute und Sekunde zerlegt. Der Algorithmus ROCK zur Clusteranalyse stellt den dritten Filter in der Verarbeitungskette dar. ROCK gruppiert die Datenpunkte des eintreffenden Datenstroms in eine vordefinierte Anzahl

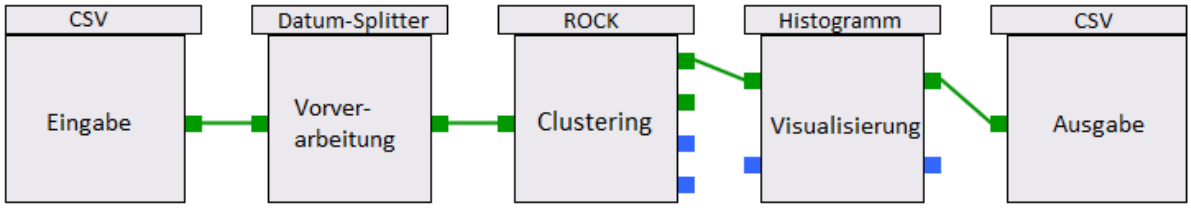


Abbildung 3.13.: Darstellung eines Arbeitsablaufs im CUF.

von Klassen. Der vierte Filter visualisiert die geclusterten Datenpunkte mithilfe von Histogrammen. Der letzte Filter der Verarbeitungskette speichert die Datenpunkte in einer CSV-Datei auf Festplatte ab. Ein- und Ausgabeschnittstellen der Filter sind durch farbige Quadrate in Abbildung 3.13 dargestellt. Die Eingabeschnittstellen befinden sich auf der linken Seite und die Ausgabeschnittstellen auf der rechten Seite. In diesem Beispiel identifizieren *grüne Quadrate* Datenpunkte und *blaue Quadrate* Cluster. In Abbildung 3.13 ist beispielsweise ersichtlich, dass der Histogrammfilter zwei unterschiedliche Eingabeformate, einzelne Datenpunkte oder Cluster, unterstützt. Die Variabilität der Filter, Daten in verschiedenen Ein- und Ausgabeformaten einzulesen beziehungsweise zu erzeugen, unterstützt das Aufteilen beziehungsweise Verbinden von Datenströmen.

Eine grundlegende Herausforderung bei der Verarbeitung von Netzwerkdaten ist die große Datenmenge. Unternehmensnetzwerke generieren pro Stunde mehrere Millionen Flows. Deswegen ist eine effiziente Verarbeitungskette erforderlich. Die Pipes-and-Filter Architektur von CUF führt dazu, dass die Filter unabhängig voneinander arbeiten und somit leicht parallelisiert werden können.

3.6.2. Filter in CUF

Dieser Abschnitt bietet einen Überblick verfügbarer Filter in CUF. Zur Erhöhung der Flexibilität wurden viele Algorithmen selbst implementiert. Dies betrifft beispielsweise die Clusterverfahren k-Prototypes [Hua98] und WARD [WJ63], um angepasste Abstandsmaße für die Abstandsberechnung zu integrieren. Für Algorithmen, bei denen keine Anpassungen erforderlich waren, wurde auf frei verfügbare Implementierungen zurückgegriffen. Beispielsweise wurde der J48 Klassifikator aus WEKA[1] übernommen.

Im Folgenden findet eine Unterteilung der Filter in sechs Kategorien Eingabe und Ausgabe, Vorverarbeitung, Clusteranalyse, Klassifikation, Evaluierung und Visualisierung statt.

3.6.2.1. Filter zur Eingabe und Ausgabe

Ein- und Ausgabefilter dienen zum Einlesen beziehungsweise Ausgeben der Daten. In CUF stehen unterschiedliche Eingabefilter zum Einlesen von Daten aus Binärdateien, Textdateien oder Datenbanken zur Verfügung. Für jeden Eingabefilter existiert ein dazugehöriger Ausgabefilter, welcher die Daten im entsprechenden Format schreiben kann. Für Textdateien stehen die zwei Formate CSV und HTML zur Verfügung. CSV ist bei vielen Entwicklungsumgebungen Standard und ermöglicht das einfache Abspeichern und Einlesen von Dateien. Eine manuelle Analyse von großen CSV-Dateien kann schnell unübersichtlich werden. Deshalb gibt es noch das HTML-Format, welches die Daten in formatierten Tabellen abspeichert.

[1] http://www.cs.waikato.ac.nz/ml/index.html

Die vorliegende Arbeit zielt primär auf die Analyse flowbasierter Datenströme. Jedoch stehen derartige Datenströme in experimentellen Testszenarien häufig nicht zur Verfügung. Stattdessen liegen die Daten oftmals in CSV-Dateien vor. Dabei enthält jeder Datenpunkt (Flow) einen Zeitstempel, welcher den Zeitpunkt angibt, zu dem der Flow am Netzwerkgerät aufgenommen wurde. Zur Simulation von Datenströmen in experimentellen Testszenarien enthält CUF einen entsprechenden Filter. Dieser Filter simuliert einen Datenstrom, indem er die Datenpunkte einliest und das Attribut Zeitstempel auswertet. Basierend auf den Zeitstempeln extrahiert der Filter die Zeitabstände zwischen den ursprünglichen Flows und leitet die eingelesenen Flows mit den ursprünglichen Zeitabständen weiter.

3.6.2.2. Filter zur Vorverarbeitung

CUF verfügt über eine große Auswahl verschiedener Vorverarbeitungsalgorithmen. Vorverarbeitungsalgorithmen entfernen Inkonsistenzen, vereinheitlichen Wertebereiche, überarbeiten Attributwerte oder selektieren Attribute. Einige Filter nutzen explizites Domänenwissen, um weitere Informationen an die Datenpunkte zu heften. Hier sei als Beispiel der GeoIPFilter genannt, der auf eine externe Datenbank zugreift, um die geographischen Koordinaten einer IP-Adresse zu ermitteln und an jeden Flow anzuhängen.

Ein wichtiger Aspekt bei der Analyse von Netzwerkdaten ist die Berücksichtigung heterogener Daten. Damit CUF mit heterogenen Daten umgehen kann, wurden einige Vorverarbeitungsalgorithmen zur Diskretisierung kontinuierlicher Attribute (zum Beispiel Hellinger-Diskretisierung [Lee07] oder die Multi-Intervall-Diskretisierung [FI93]) oder zur Transformation kategorischer Attribute in kontinuierliche Attribute (zum Beispiel durch Binarisierung) integriert.

3.6.2.3. Filter zur Clusteranalyse

Ein generelles Ziel der Datenanalyse ist, ein besseres Verständnis für die Daten zu erhalten [LOS+13]. Die Clusteranalyse (siehe Abschnitt 3.2) ist ein unüberwachtes Verfahren und gruppiert Datenpunkte bezüglich ihrer Ähnlichkeiten in Gruppen, sogenannte Cluster. CUF verfügt über diverse Algorithmen zur Clusteranalyse. So befinden sich partitionierende Algorithmen (k-Prototypes [Hua98]), hierarchische Algorithmen (WARD [MC12], ROCK [GRS99], QROCK [DMP05]), gitter- und dichtebasierte Algorithmen (CLIQUE [AGG+98], pMafia [GNC99], fpMafia [LL05]) in CUF. CUF erlaubt die Ausführung dieser Clusterverfahren mit unterschiedlichen Abstandsmaßen. Des Weiteren verfügt CUF über die Stream-Clusterverfahren CluStream [AHW+03] und DenStream [CEQ+06].

3.6.2.4. Filter zur Klassifikation

Klassifikatoren erlernen ein Modell, welches in der Lage ist, Objekte in vordefinierte Klassen zu unterteilen. Das Modell wird in einer Trainingsphase mithilfe eines Datensatzes erlernt und kann anschließend zur Klassifikation neuer Objekte, die keine Labels besitzen, verwendet werden. Der zweistufige Klassifikationsprozess ist in Abschnitt 2.1.2 beschrieben. CUF verfügt über die Klassifikatoren CMAR [LHP01], J48 (Open Source Implementierung von C4.5 [Qui93]) und k-Nächster-Nachbar (nach Beschreibung von [HKP11]).

3.6.2.5. Filter zur Evaluation

CUF bietet unterschiedliche Methoden zur Evaluierung an. Eine Möglichkeit zur Evaluierung bieten Standardmaße wie Klassifikationsgenauigkeit, F1-Score, Recall und Precision [HKP11]. Diese Maße sind jedoch nur für gelabelte Datensätze nutzbar.

Falls keine gelabelten Datensätze zur Verfügung stehen, bietet CUF sogenannte intrinsische Validierungsmaße an. Intrinsische Validierungsmaße bewerten die Ergebnisse der Clusteranalyse hinsichtlich ihrer Kompaktheit und Trennbarkeit. Kompaktheit bedeutet, dass die Datenpunkte eines Clusters sehr ähnlich sein sollten und Datenpunkte aus verschiedenen Clustern sollten möglichst unähnlich sein (Trennbarkeit). Diese Maße geben jedoch nur eine Gesamtbewertung der Ergebnisse und liefern keine detaillierten Informationen. Hassani und Seidl [HS15] bewerteten elf intrinsische Validierungsmaße bezüglich ihrer Eignung auf Datenströme und identifizierten Calinski-Harabasz [CH74] als bestes intrinsische Validierungsmaß. CUF verfügt über die vielversprechenden Validierungsmaße Calinski-Harabasz [CH74], CPCQ-Index [LD12], CDbw-Index [CSL04], Davies-Bouldin-Index [DB79] und Silhouette Index [Rou87].

3.6.2.6. Filter zur Visualisierung

Visualisierungsverfahren stellen eine weitere Möglichkeit zur Evaluierung in CUF dar. Beispielsweise dient der selbst entwickelte EventsPerTime Filter (siehe Abbildung 3.14) zur Visualisierung von Flows über Zeiträume. Histogrammdarstellungen ermöglichen die Visualisierung von Datenverteilungen. Weiterhin sind im CUF komplexere Visualisierungsverfahren wie RadViz [HGM+97] oder die nach [Ins85] beschriebenen Methode Parallelkoordinaten vorhanden.

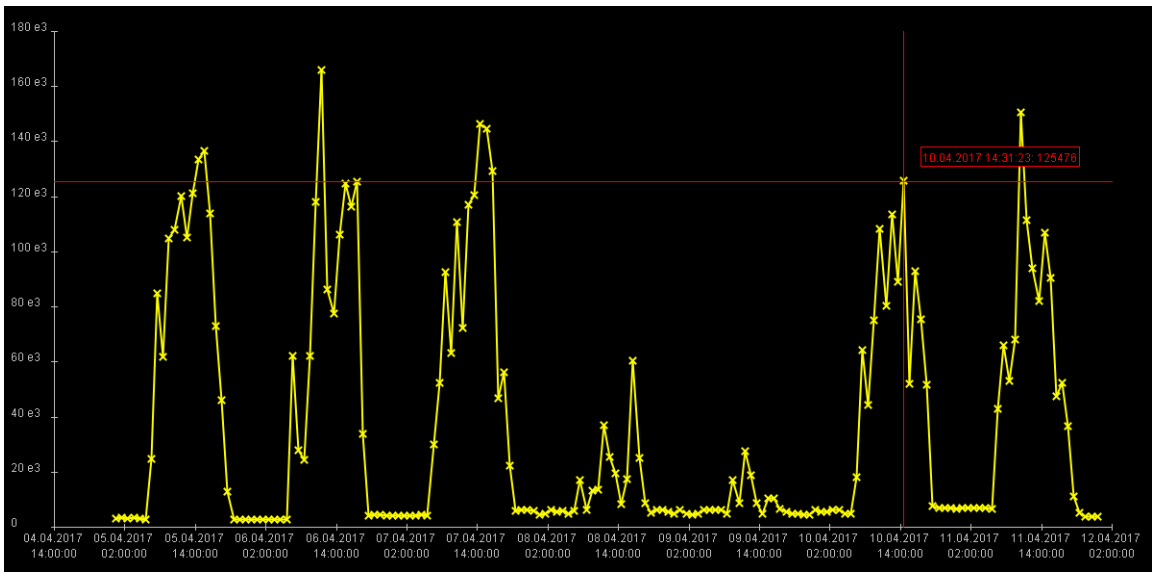


Abbildung 3.14.: Graphische Darstellung der Anzahl Flows pro Stunde durch den EventsPerTime Filter.

Abbildung 3.15 visualisiert einen Ausschnitt flowbasierter Netzwerkdaten, die einen Port Scan beinhalten, in Parallelkoordinatendarstellung. In dieser Abbildung stellt jedes Attribut (Zeitstempel, Quell-IP-Adresse, Ziel-IP-Adresse und Ziel-Port) eine vertikale Achse dar. Alle Achsen stehen parallel zueinander und jede Linie (von links nach rechts) repräsentiert einen Datenpunkt. Abbildung 3.15 zeigt, dass innerhalb eines kurzen Zeitraums von einer Quell-IP-Adresse

viele verschiedene Ziel-Ports auf unterschiedlichen Ziel-IP-Adressen angesprochen wurden (siehe schwarze Linien in Abbildung 3.15). Einzelne Datenpunkte können ausgewählt und hervorgehoben werden (siehe rote Linie in Abbildung 3.15).

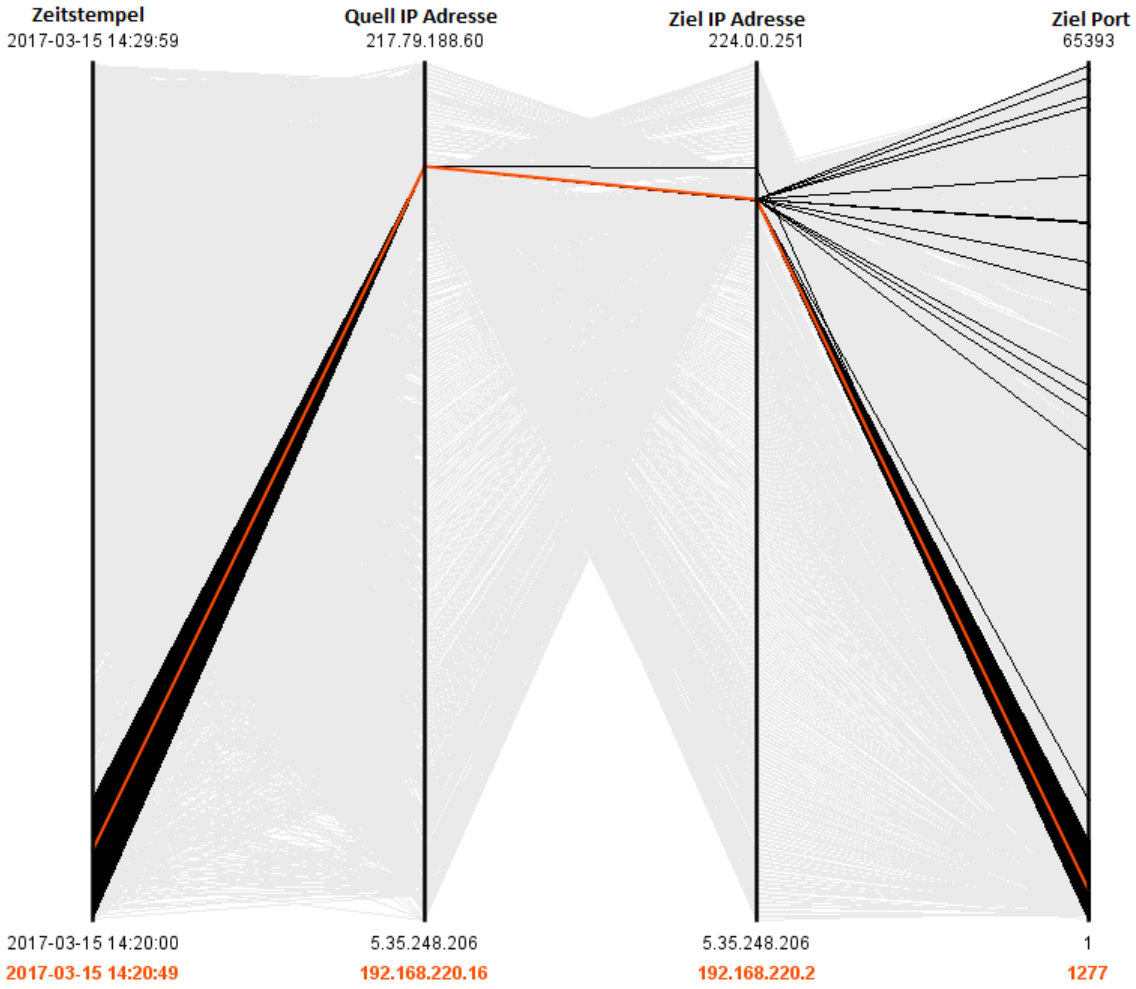


Abbildung 3.15.: Parallelkoordinatendarstellung eines Port Scans.

4. IT-Sicherheit

Ziel dieser Arbeit ist die Detektion sicherheitskritischer Ereignisse in Unternehmensnetzwerken mittels Data Mining-Algorithmen. Im vorherigen Kapitel wurden die methodischen Grundlagen aus dem Bereich Data Mining behandelt. Dieses Kapitel beschäftigt sich mit relevanten Grundlagen auf dem Gebiet IT-Sicherheit. Zunächst werden die verschiedenen Arten von Sicherheitssystemen diskutiert. Danach werden die in dieser Arbeit zugrundeliegenden Datenquellen genauer analysiert. Zuletzt beschreibt dieses Kapitel den Ablauf typischer Angriffsszenarien und verschiedene Angriffsarten, die in den analysierten Datensätzen dieser Arbeit auftreten.

4.1. Angriffsdetektion

Methoden zur Detektion sicherheitskritischer Ereignisse lassen sich in signaturbasierte und anomaliebasierte Ansätze unterscheiden [GDM+09]. Koch et al. [KGR14] nehmen eine ähnliche Unterteilung in wissensbasierte und verhaltensbasierte Ansätze vor. Hierbei entsprechen wissensbasierte Systeme den signaturbasierten Systemen und verhaltensbasierte Systeme den anomaliebasierten Systemen. Neben diesen beiden Kategorien definieren einige Autoren wie Buczak und Guven [BG16] oder Bhuyan et al. [BBK14] noch einen weiteren, hybriden Ansatz, der eine Kombination der beiden Ansätze darstellt.

4.1.1. Signaturbasierte Verfahren

Signaturbasierte Verfahren verwenden Wissen über bekannte Angriffe zur Detektion sicherheitskritischer Ereignisse. Hierfür vergleichen diese Ansätze den eintreffenden Datenstrom mit bekannten Signaturen von Angriffsszenarien. Falls eine Übereinstimmung vorliegt, markieren diese Systeme die aktuellen Daten als Angriff. Falls keine Übereinstimmung vorliegt, markieren diese Systeme die aktuellen Daten als normalen Netzwerkverkehr. Signaturbasierte Ansätze zeichnen sich durch zuverlässige Erkennungsraten aus, generieren nahezu keine Fehlalarmierungen (False Positives) und gehen mit der Einschränkung einher, dass sie nur bereits bekannte Angriffssignaturen erkennen können und somit eine stetige Aktualisierung ihrer Signaturen-Listen benötigen. Ein Beispiel für einen signaturbasierten Ansatz sind klassische Antiviren-Scanner, welche Dateien auf bekannte Signaturen von Schadsoftware überprüfen.

4.1.2. Anomaliebasierte Verfahren

Anomaliebasierte Verfahren modellieren Normalverhalten anhand von Trainingsdaten und kennzeichnen Abweichungen vom erlernten Verhalten als mögliche Anomalien [BG16]. Diese Vorgehensweise ermöglicht die Detektion bisher unbekannter Angriffsmuster. Jedoch generieren anomaliebasierte Verfahren häufig eine höhere Anzahl von Fehlalarmierungen (False Positives). Aufgrund der hohen Datenmengen kann bereits eine geringe Anzahl von Fehlalarmierungen ein Sicherheitssystem unbrauchbar machen.

Der in dieser Arbeit entwickelte Lösungsansatz (siehe Kapitel 10 und 11) gehört der Kategorie anomaliebasierter Sicherheitssysteme an. Der Ansatz nutzt zwar gelabelte Daten von Normal-

verhalten und von Angriffsszenarien in der Trainingsphase, generalisiert jedoch die erlernten Muster, sodass auch Abweichungen und neue Variationen erkannt werden können.

4.2. Datenquellen im Bereich IT-Sicherheit

Unterschiedlichste Datenquellen generieren relevante Daten für die Domäne IT-Sicherheit. Beispiele sind Logdateien aus Sicherheitssystemen wie Firewalls oder IDS. Eine weitere Quelle sind hostbasierte Daten wie Logdateien unterschiedlicher Anwendungen. Beispielsweise protokolliert der SSH-Daemon auf dem Linux-System Ubuntu jede erfolgreiche und fehlgeschlagene Anmeldung in der Logdatei */var/log/auth.log*. Somit liefern hostbasierte Datenquellen sehr detaillierte Informationen.

Während hostbasierte Daten detaillierte Informationen über einzelne Systeme liefern, umfassen netzwerkbasierte Daten Informationen über alle am Netzwerk angeschlossenen Systeme. Die Aufzeichnung und Analyse netzwerkbasierter Daten gestaltet sich im Vergleich zu hostbasierten Daten einfacher, führt zu weniger datenschutzrechtlichen Problemen und gibt einen besseren Überblick über den Gesamtzustand eines Unternehmensnetzwerks. Diese Vorteile gehen jedoch mit dem Verlust der Informationstiefe einher. Die folgende Analyse fokussiert auf netzwerkbasierte Daten, welche im Mittelpunkt dieser Arbeit stehen. Netzwerkbasierte Daten können an Netzwerkgeräten aufgenommen und in zwei Kategorien unterteilt werden. Paketbasierte Netzwerkdaten enthalten den kompletten Netzwerkverkehr durch Spiegelung der Ports an Netzwerkgeräten. Flowbasierte Netzwerkdaten enthalten lediglich Metainformationen über die erstellten Netzwerkverbindungen.

4.2.1. Paketbasierte Netzwerkdaten

Zwei wichtige Modelle im Bereich Computernetze sind das ISO/OSI-Modell (International Standardization Organization/Open System Interconnection) und das TCP/IP-Modell (Transmission Control Protocol/Internet Protocol). ISO/OSI-Modell ist ein Referenzmodell, welches die Kommunikation zwischen Systemen beschreibt und besteht aus sieben hierarchisch angeordneten Schichten [TW11]. Das TCP/IP-Modell entstand durch die Abstraktion vorhandener Protokolle und der größte Teil der Internetkommunikation basiert auf diesem Modell. Im Folgenden wird deshalb das TCP/IP-Modell verwendet, um den Aufbau von Netzwerkdaten zu beschreiben.

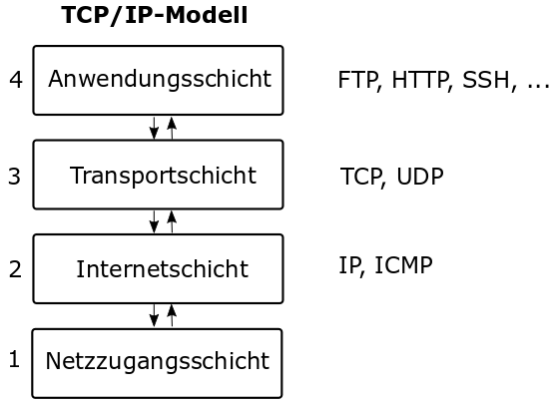


Abbildung 4.1.: TCP/IP-Modell nach [Eck18].

Das TCP/IP-Modell ist in Abbildung 4.1 zu sehen und besteht aus vier Schichten. Die Anwendungsschicht umfasst Protokolle auf Betriebssystemebene wie beispielsweise FTP, HTTP, SSH oder SMTP. Die Transportschicht und die Internetschicht stellen die zentralen Schichten des TCP/IP-Modells dar. Transmission Control Protocol (TCP) und User Datagram Protocol (UDP) sind Protokolle der Transportschicht und stellen Ende-zu-Ende Verbindungen für die Protokolle der Anwendungsschicht bereit [TW11]. Das IP Protokoll ist auf der Internetschicht angesiedelt und stellt einen paketvermittelnden verbindungslosen Dienst zwischen Endgeräten dar [Eck18]. Die Netzzugangsschicht ist für die physikalische Übertragung der Daten zuständig.

Paketbasierte Netzwerkdaten werden auf Schicht 1 (Netzzugangsschicht) des TCP/IP-Modells abgefangen und beinhalten somit die Header der Internet- und Transportschicht sowie die kompletten Paketinhalte. Oftmals liegen paketbasierte Netzwerkdaten im pcap Format vor und die zur Verfügung stehenden Metadaten hängen von den gewählten Protokollen der Internet- und Transportschicht ab. Im Folgenden findet eine Analyse der wichtigsten Protokolle Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Internet Control Message Protocol (ICMP) und Internet Protocol (IP) statt.

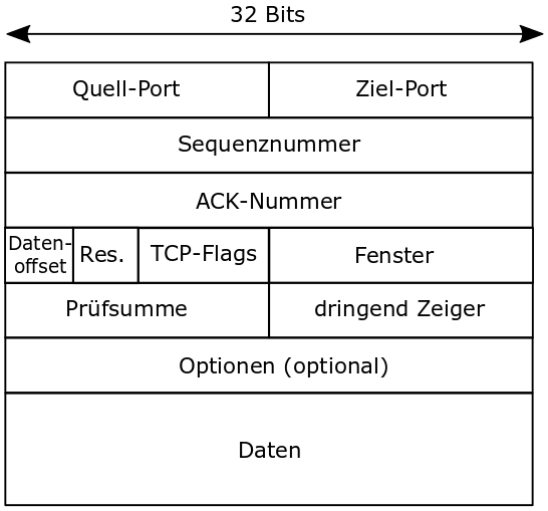


Abbildung 4.2.: TCP-Header nach [TW11].

TCP und UDP stellen Transportprotokolle für die Übertragung von Daten über ein Netzwerk dar (Schicht 3 des TCP/IP-Modells). Während es sich bei TCP um ein verbindungsorientiertes Transportprotokoll für unzuverlässige Netzwerke handelt, ist UDP ein verbindungsloses Transportprotokoll [TW11]. TCP stellt durch Nummerierung und Quittierung der Netzwerkpakete sicher, dass gesendete Daten den Empfänger erreichen. Hierfür nutzt TCP die Attribute Sequenznummer, ACK-Nummer und Prüfsumme des TCP-Headers in Abbildung 4.2. Des Weiteren beinhaltet der TCP-Header noch weitere Attribute wie Quell-Port, Ziel-Port und TCP-Flags. Beim verbindungslosen Transportprotokoll UDP findet hingegen keine Überprüfung statt, ob Netzwerkpakete tatsächlich ihr Ziel erreicht haben. Dies wird durch den kleineren Header in Abbildung 4.3 ersichtlich, der lediglich die vier Attribute Quell-Port, Ziel-Port, Länge und Prüfsumme umfasst. Durch diesen Aufbau eignet sich UDP beispielsweise für Sprachübertragungen, wenn der Verlust einiger Netzwerkpakete keine großen Auswirkungen verursacht und ein erneutes Senden aufgrund der Echtzeitanforderung keinen Sinn ergibt.

ICMP dient zum Austausch von Informations- und Fehlernachrichten und nutzt IP als Kommunikationsbasis. Der ICMP-Header ist in Abbildung 4.4 dargestellt. Der ICMP-Header enthält

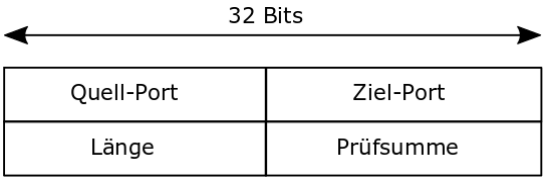


Abbildung 4.3.: UDP-Header nach [TW11].

lediglich die Felder Typ, Code, Prüfsumme und das optionale Feld Daten. Beispielsweise verursachen Ping-Anfragen ICMP-Pakete des Typs 8 (Echo-Anfrage). Eine Anfrage an nicht existierende Ziele führt zur Erstellung von ICMP-Paketen des Typs 3 (Ziel nicht erreichbar). Detailliertere Informationen werden durch das Attribut Code übertragen. So würde beispielsweise eine nicht erreichbare Ziel-IP-Adresse einen Code von 1 (Host nicht erreichbar) und ein nicht erreichbarer Port einen Code von 3 (Port nicht erreichbar) verursachen.

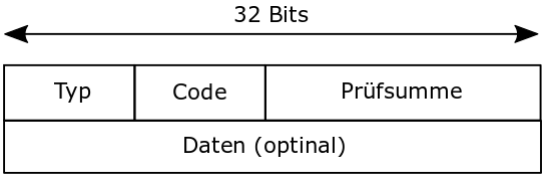


Abbildung 4.4.: ICMP-Header nach [Pos81].

IP ist ein Protokoll der Internetschicht und für die Übertragung der Datenpakete von Sender zu Empfänger zuständig (Schicht 2 des TCP/IP-Modells). Der IP-Header, welcher von TCP, UDP und ICMP genutzt wird, ist in Abbildung 4.5 graphisch dargestellt und beinhaltet unter anderem Informationen über die Quell- und Ziel-IP-Adresse, sowie Transportinformationen (zum Beispiel Time to Live) und Prüfsummen.

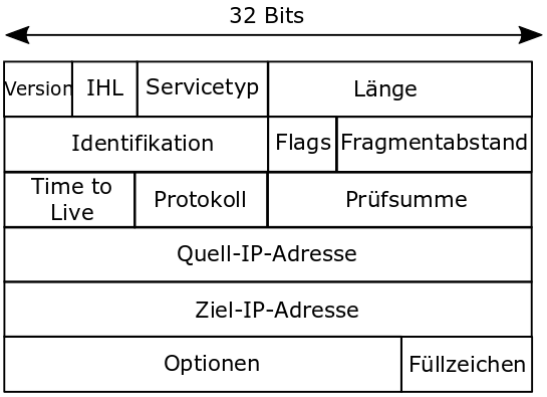


Abbildung 4.5.: IP-Header nach [TW11].

4.2.2. Flowbasiertes Netzwerkdaten

Flowbasierte Netzwerkdaten stellen eine aggregierte Form paketbasierter Netzwerkdaten dar und beinhalten lediglich Metainformationen über Netzwerkverbindungen. Diese Metainformationen setzen sich aus den Headern der Internet- und Transportschicht zusammen. Die Ursprünge flowbasierter Netzwerkdaten gehen bis ins Jahr 1991 zurück [HČT+14].

Ein Flow fasst alle übertragenen Netzwerkpakete innerhalb eines Zeitfensters zusammen, welche in vordefinierten Attributen die gleichen Ausprägungen annehmen. Häufig wird hierfür das sogenannte Fünf-Tupel verwendet. Die Fünf-Tupel-Definition fasst alle Netzwerkpakete mit der gleichen Quell-IP-Adresse, Quell-Port, Ziel-IP-Adresse, Ziel-Port und Transport-Protokoll zu einem Flow zusammen [Cla08]. Diese Zusammenfassung stoppt entweder durch eine aktive oder inaktive Zeitüberschreitung. Die inaktive Zeitüberschreitung dient zur Konsolidierung des Datenstroms und greift, wenn das letzte zum Flow zugehörige Netzwerkpaket vor mehr als α Sekunden aufgezeichnet wurde. Die aktive Zeitüberschreitung dient zur Vermeidung von Zusammenfassungen über zu lange Zeiträume und greift, wenn die Sammlung von Netzwerkpaketen eines Flows β Sekunden überschreitet. NetFlow setzt die Werte standardmäßig auf $\alpha = 15$ und $\beta = 1800$.

Flowbasierte Netzwerkdaten liegen im unidirektionalen oder bidirektionalen Format vor. Das unidirektionale Format fasst alle Netzwerkpakete von Host A zu Host B in einen Flow zusammen, welche in den oben genannten Attributen übereinstimmen. Analog dazu werden die Netzwerkpakete von Host B zu Host A in einem zweiten unidirektionalen Flow zusammengefasst. Im Gegensatz dazu beinhaltet ein bidirektionaler Flow beides, die Netzwerkpakete von Host A zu Host B sowie die Netzwerkpakete von Host B zu Host A. John et al. [JDC10] fanden heraus, dass asymmetrisches Routing in großen Unternehmensnetzwerken bidirektionale Flows verfälschen kann. Aus diesem Grund stehen unidirektionale Flows im Fokus dieser Arbeit.

Tabelle 4.1.: Typische Attribute in unidirektionalen flowbasierten Netzwerkdaten.

#	**Name**	**Beschreibung**	**Attributtyp**
1	Quell-IP	Quell-IP-Adresse	kategorisch
2	Quell-Port	Quell-Port	kategorisch
3	Ziel-IP	Ziel-IP-Adresse	kategorisch
4	Ziel-Port	Ziel-Port	kategorisch
5	Protokoll	Transport-Protokoll	kategorisch
6	Zeitstempel	Zeitstempel des ersten Netzwerkpakets	kontinuierlich
7	Dauer	Dauer des Flows (Zeitraum zwischen ersten und letzten Netzwerkpaket des Flows)	kontinuierlich
8	Bytes	Anzahl übertragener Bytes	kontinuierlich
9	Pakete	Anzahl übertragener Pakete	kontinuierlich
10	TCP-Flags	*ODER* Konkatenation aller TCP-Flags	kategorisch/binär

Standardisierte und weit verbreitete flowbasierte Formate sind NetFlow [Cla04], IPFIX [Cla08, CTA13], sFlow [PPM01] und OpenFlow [MAB+08]. Der von der Internet Engineering Task Force (IETF) entwickelte Standard IPFIX basiert auf NetFlow v9. Die Relevanz flowbasierter Datenanalyse hat alle großen Hersteller dazu veranlasst, ihre Netzwerkkomponenten so zu erweitern, dass diese Flows sammeln und exportieren können [USB17]. Die Verbreitung und Integration von Flow-Exportern in aktueller Hardware führt dazu, dass das Aufnehmen von Flows fast keine Kosten verursacht [GHK14]. Tabelle 4.1 liefert einen Überblick typischer Attribute in unidirektionalen flowbasierten Netzwerkdaten, die stets vorhanden sind. Aus Tabelle 4.1 ist ersichtlich, dass Flows im Wesentlichen die Quelle und das Ziel von Netzwerkverbindungen beschreiben und keine Nutzdaten beinhalten. Je nach Flow-Format und Flow-Exporter kann ein Flow noch weitere Attribute wie beispielsweise die Anzahl übertragener Bytes pro Sekunde oder die Anzahl übertragener Bytes im ersten Paket beinhalten. Für IPFIX sind mehr als 100 Attribute definiert. Grundsätzlich ist eine Transformation paketbasierter Netzwerkdaten in flowbasierter

Netzwerkdaten möglich. Diese Umwandlung kann mit Tools wie nfdump[1], Tranalyzer[2] oder YAF[3] erfolgen. Eine Transformation in umgekehrter Richtung ist jedoch nicht möglich. Haddadi und Zincir-Heywood [HZ16] untersuchen den Einfluss unterschiedlicher Flow-Exporter auf nachfolgende Analysemethode. Hierfür verwendeten die Autoren verschiedene Klassifikatoren und klassifizierten Netzwerkverkehr in zwei Klassen, normal und abnormal. Ergebnis von [HZ16] ist, dass die fünf untersuchten Flow-Exporter teilweise signifikanten Einfluss auf die Ergebnisse nachgelagerter Analysemethoden besitzen. In der experimentellen Studie von [HZ16] erzielte der Flow-Exporter Tranalyzer die besten Ergebnisse.

4.2.3. Schlussfolgerung netzwerkbasierter Datenquellen

Netzwerkbasierte Sicherheitssysteme arbeiten in der Regel auf paket- oder flowbasierten Netzwerkdaten. Diese Arbeit fokussiert sich auf unidirektionale flowbasierte Netzwerkdaten. Im Vergleich zu paketbasierten Netzwerkdaten führt dies zu einer Reduzierung der zu analysierenden Datenmenge. Sperotto und Pras [SP11] verglichen die angefallene Datenmenge zwischen paketbasierten und flowbasierten Netzwerkdaten an der Universität Twente und beobachteten eine Reduzierung der Datenmenge auf 0,1 Prozent der ursprünglichen Datenmenge. Des Weiteren verfügen flowbasierte Netzwerkdaten über keine Nutzdaten, was zu einer Reduzierung datenschutzrechtlicher Aspekte führt. Verschlüsselte Verbindungen haben ebenfalls keinen Einfluss auf flowbasierte Netzwerkdaten. Flows bestehen aus einer vordefinierten Menge von Attributen und können leicht an zentralen Komponenten wie Switches oder Firewalls abgefangen werden. Gleichzeitig ermöglicht die Verwendung von Flows die Überwachung des kompletten Unternehmensnetzwerks. Beispielsweise würden Firewall-Logdateien die analysierten Daten auf den Datenverkehr beschränken, der die Firewall passiert und somit die Detektion von Insider-Angriffen nicht erlauben. Im Gegensatz dazu passiert auch interner Netzwerkverkehr die Switches und Router eines Unternehmens. Folglich erlaubt die Verwendung von flowbasiertem Netzwerkverkehr die Überwachung des kompletten Netzwerkverkehrs eines Unternehmens unter Wahrung datenschutzrechtlicher Aspekte. Flows eignen sich laut [SSS+10] sehr gut für die Detektion von DDoS-Angriffen, Port Scans, Würmern (Ausbreitungsphase) und Botnets. Die Gemeinsamkeit dieser Angriffe liegt darin, dass sie direkt messbare Auswirkungen in flowbasierten Netzwerkvekehr aufweisen (zum Beispiel Anzahl Flows in einem Zeitintervall oder verdächtige Portnummern) [HČT+14]. Diese Vorteile gehen jedoch auch mit Nachteilen einher. Die Reduzierung auf flowbasierte Netzwerkdaten geht primär mit einem Informationsverlust einher, welcher laut Sperotto und Pras [SP11] dazu führt, dass flowbasierte Ansätze nicht die gleiche Genauigkeit wie paketbasierte Ansätze erreichen können.

4.3. Typische Phasen eines Angriffsszenarios

Angriffsszenarien folgen häufig vordefinierten Phasen. Die Beschreibung der nachfolgenden Inhalte basiert auf [RWG+17a]. Skoudis und Liston [SL06] geben eine weit verbreitete Definition, bei der ein Angriffszenario in die folgenden 5 Phasen unterteilt wird:

1. Reconnaissance
2. Scanning

[1] https://github.com/phaag/nfdump
[2] https://tranalyzer.com/
[3] https://tools.netsa.cert.org/yaf/

3. Gaining Access
4. Maintaining Access
5. Covering Tracks

Jede Phase zeichnet sich durch unterschiedliche Ziele und daran angepasste Methoden aus. In der ersten Phase (Reconnaissance) versucht der Angreifer möglichst viele Informationen über das Zielsystem zu erlangen. Laut Skoudis und Liston [SL06] gehören diverse Aktivitäten zu dieser Phase wie Recherche im Internet, Social Engineering oder auch das Durchsuchen von Abfällen nach Dokumenten mit Anmeldedaten wie Benutzername und Passwort (Dumpster Diving). Ein beliebter Startpunkt ist das Durchsuchen von Firmenwebseiten, um an Informationen wie Namen von Mitarbeitern zu gelangen. Beim Social Enginieering stellen häufig Geschäftsführer oder Administratoren beliebte Ziele dar. Eine weitere Methode zur Sammlung von Informationen in der Reconnaissance-Phase sind Phishing-Mails, in denen der Angreifer den Adressaten zur Passworteingabe oder Angabe andere Informationen auffordert. Durch Abfragen von DNS-Einträgen können IP-Adressen der Zielorganisationen ermittelt werden.

Die zweite Phase (Scanning) untersucht das Zielsystem genauer und greift auf Informationen der Reconnaissance-Phase zurück. Im Prinzip ist die Scanning-Phase eine Weiterführung der ersten Phase. In der Scanning-Phase wird nach Hosts und offenen Ports gesucht, um potentielle Angriffspunkte auszumachen. Hierfür verwenden Angreifer oftmals Port Scans. Nmap[4] ist ein sehr bekanntes Scanning Tool, welches auch genauere Analysen wie die Erkennung bestimmter Betriebssysteme erlaubt.

In der dritten Phase (Gaining Access) findet der eigentliche Angriff statt. Typische Angriffsszenarien sind SSH-Brute-Force Angriffe, SQL-Injections, PHP-Exploits oder DoS-Angriffe (siehe Abschnitt 4.4). In dieser Phase versucht der Angreifer Zugriff auf einen Rechner zu erlangen. DoS-Angriffe versuchen die Verfügbarkeit eines Diensts durch das Überfluten mit Anfragen zu beeinträchtigen [Che06].

In der vierten Phase (Maintaining Access) versucht der Angreifer seinen Zugang zum System zu erhalten. Typische Techniken sind laut Skoudis and Liston [SL06] Trojanische Pferde, Bots, Backdoors und Rootkits. Bereits komprimierte Systeme stellen dabei häufig den Startpunkt für weitergehende Angriffe im Zielnetzwerk dar.

Die letzte Phase (Covering Tracks) vervollständigt den Zyklus eines Angriffsszenarios. Angreifer bevorzugen in der Regel unentdeckt zu bleiben, um die Kontrolle über ein System für einen längeren Zeitraum zu behalten. Dadurch können Hacker zu späteren Zeitpunkten Daten und CPU Leistung stehlen oder weitere Angriffe durchführen [SL06].

Ein typisches Angriffsszenario muss nicht zwingend alle Phase beinhalten. Beispielsweise könnte ein Insider bereits die Ziele kennen und somit die Informationsakquisitionsphasen 1 und 2 überspringen. Des Weiteren sind nicht alle Angreifer daran interessiert, den Zugriff zu erhalten (Phase 4) oder ihre Spuren zu verwischen (Phase 5).

4.4. Angriffsarten

Dieser Abschnitt beschreibt einige Angriffsarten, die im weiteren Verlauf der Arbeit eine Rolle spielen. Hierbei handelt es sich um Port Scans, Denial-of-Service (DoS), SSH-Brute-Force und Botnets. Für eine weitergehende Taxonomy und Einteilung von Angriffsarten sei auf [HH05] und [IW08] verwiesen.

[4] https://nmap.org/

4.4.1. Port Scan

Bei einem Port Scan sendet der Angreifer Netzwerkpakete an Ziel-IP-Adressen und Ziel-Ports und versucht dadurch, potentielle Angriffspunkte in einer Zielorganisation zu ermitteln. Somit stellen Port Scans an sich keinen direkten Angriff auf ein System dar und sammeln lediglich Informationen über offene Dienste und verfügbare IP-Adressen. Diese Informationen werden später für gezielte Angriffe auf Systeme genutzt.

In Kapitel 11 werden zwei Methoden zur Detektion von Port Scans vorgestellt. Deshalb werden im Folgenden die wesentlichen Eigenschaften verschiedener Port Scanning-Techniken genauer analysiert. Es existieren viele verschiedene Techniken für aktives Port Scanning und das Verhalten von Angreifern und Opfern variiert je nach Technik. Port Scans lassen sich einerseits in vertikale und horizontale Port Scans unterteilen. Vertikale Port Scans werden genutzt, um die offenen Ports eines Hosts zu identifizieren. Horizontale Port Scans werden genutzt, um bestimmte offene Ports in einer Menge von Hosts zu identifizieren [SHM02]. Andererseits kann Port Scanning bezüglich der Transport- und Netzwerk-Protokolle unterteilt werden. Die nachfolgende Beschreibung unterschiedlicher Port Scan-Techniken basiert auf [RLH18] und ist anhand der zugrundeliegenden Protokolle TCP, UDP und ICMP strukturiert.

TCP-Scanning. Typischer Vertreter sind SYN-Scans. Bei SYN-Scans versendet der Angreifer ein Netzwerkpaket mit gesetzten TCP-Flag SYN und initiiert somit den 3-Way-Handshake. Falls das Netzwerkpaket an einen geschlossen TCP-Port gesendet wird, antwortet das Opfer mit dem TCP-Flag RST. Falls das Netzwerkpaket an einen offenen TCP Port gesendet wird, bestätigt das Opfer den Verbindungsaufbau und antwortet mit den TCP-Flags SYN und ACK. Der Angreifer schließt den 3-Way-Handshake für offene TCP-Ports in der Regel jedoch nicht ab, um Logeinträge beim Opfer zu vermeiden. Deshalb wird diese Vorgehensweise häufig als Stealthy SYN-Scan bezeichnet [SHM02].

FIN-Scans sind eine weitere Ausprägung von TCP-Scanning. Bei FIN-Scans versendet der Angreifer Netzwerkpakete mit gesetzten TCP-Flag FIN, um Firewallregeln zu umgehen. Falls das Netzwerkpaket an einen geschlossen TCP-Port gesendet wird, antwortet das Opfer mit den TCP-Flag RST. Falls das Netzwerkpaket auf einen offenen TCP-Port trifft, wird vom Opfer keine Antwort generiert [UPK+09].

Des Weiteren gibt es noch eine große Anzahl weiterer TCP-Scanning-Techniken (zum Beispiel XMAS-Scan oder ACK-Scan), die sich lediglich durch die gesetzten TCP-Flags in den Netzwerkpaketen unterscheiden [UPK+09].

UDP-Scanning. UDP-Scanning unterscheidet sich signifikant von TCP-Scanning. Falls ein Angreifer ein Netzwerkpaket zu einem offenen UDP-Port sendet, wird nicht zwingend eine Antwort generiert. Ein ausbleibendes Antwortpaket kann jedoch auch ein Anzeichen dafür sein, dass das Netzwerkpaket durch eine Firewall blockiert wurde. Falls der Angreifer ein Netzwerkpaket an einen geschlossenen UDP-Port sendet, antwortet das Opfer mit einer Nachricht des Typs ICMP-Unreachable [BBK11]. Moderne Betriebssysteme limitieren häufig die Anzahl von Nachrichten des Typs ICMP-Unreachable auf 1 pro Sekunde [Lyo08].

ICMP-Scanning. Ping Scans werden zur Identifikation von Hosts und nicht zur Identifizierung offener Ports verwendet und basieren auf dem Protokoll ICMP. Bei dieser Technik sendet der Angreifer ICMP Netzwerkpakete vom Typ Echo an IP-Adressen [VCI+99]. Falls ein Host diese IP-Adresse besitzt, antwortet dieser mit einem ICMP Netzwerkpaket vom Typ Echo Replay.

Falls diese IP-Adresse im Netzwerk nicht vergeben ist, antwortet der Router mit einem ICMP Netzwerkpaket vom Typ Unreachable. Je nach Konfiguration der Hosts und Router können diese automatisierten Antworten auch deaktiviert werden.

4.4.2. Denial-of-Service

Ziel eines Denial-of-Service (DoS) Angriffs ist es, einen bestimmten Service so zu beeinträchtigen, dass kein regulärer Betrieb mehr möglich ist. Hierfür sendet der Angreifer eine große Anzahl von Anfragen an den Service, sodass dieser keine regulären Anfragen mehr entgegennehmen kann. Häufig nutzen DoS Angriffe Softwarefehler von Anwendungen aus [KAG+18].

Eine Weiterentwicklung von DoS-Angriffe sind sogenannte Distributed-Denial-of-Service (DDoS) Angriffe. DDoS Angriffe senden die Anfragen nicht nur von einem Rechner, sondern von vielen verschiedenen Rechnern, um Schutzmechanismen zu umgehen und größere Auslastungen zu erreichen. Die Besonderheit hierbei ist die Koordination der verschiedenen Rechner, sodass alle Rechner ihre Anfragen zeitgleich stellen. Zur Ausführung und Koordination von DDoS-Angriffen werden häufig Botnetze verwendet [KAG+18].

4.4.3. SSH-Brute-Force

Bei SSH-Brute-Force Angriffen versucht der Angreifer Zugang auf ein System über das SSH-Protokoll zu erhalten. Die Kenntnis über eine gültige Kombination aus Benutzername und Passwort stellt dabei die einfachste Zugangsmöglichkeit dar. Bei Brute-Force Angriffen versucht der Angreifer, das Passwort für einen Benutzeraccount durch mehrmalige Anmeldeversuche zu erraten [NKC+15]. Angreifer setzen hierfür oftmals sogenannte Wörterbücher ein, die Listen von oft verwendeten Passwörtern beinhalten.

4.4.4. Botnetze

Ein Botnetz ist eine Gruppe von infizierten Hosts, die mit einem sogenannten Botnetz-Kontroller kommunizieren [Eck18]. Der Botnetz-Kontroller ist in der Lage, infizierte Hosts zu steuern und kommuniziert häufig über das IRC (Internet Relay Chat) Protokoll [GH07]. Botnetze können aus mehreren tausenden infizierten Hosts bestehen und werden zum Ausführen anderer Angriffe verwendet. Häufig dienen Botnetze zum Ausführen von DDoS-Angriffen oder zum Versenden von Spam E-Mails [Eck18]. In letzter Zeit stehlen Botnetz-Kontroller immer häufiger auch die Rechenleistung infizierter Hosts zur Berechnung von Kryptowährungen.

4.4.5. Weitere Angriffsarten

Viren, Würmer und Trojanische Pferde werden im Allgemeinen als Schadsoftware bezeichnet [Eck18]. Derartige Programme führen für den Benutzer verborgene und bösartige Aktionen aus. Ein Virus hängt sich in der Regel an ein Wirtprogramm an und versucht, sich dann zu verbreiten [Eck18]. Würmer sind im Gegensatz zu Viren eigenständige Programme mit der Fähigkeit sich zu vermehren [Eck18]. Ein Trojanisches Pferd ist ein Programm, dessen tatsächliche Funktionalität von der angegebenen Funktionalität abweicht [Eck18]. Diese Programme erfüllen zwar in der Regel ihre Soll-Funktionalitäten, führen jedoch weitere verborgene Aktionen aus (zum Beispiel Spionage von Passwörtern). Ein bekanntes Beispiel aus jüngerer Vergangenheit ist das Trojanische Pferd WannaCry, welches mehr als 200.000 Windows-Systeme befiehl und

mehrere Millionen Euro Schaden verursachte [Eck18]. Einmal mit dem Trojaner infiziert, verschlüsselte WannaCry alle Daten des Systems und forderte anschließend zu einer Zahlung auf, um den Schlüssel zur Entschlüsselung zu erhalten.

Eine weitere Art von Angriffen sind sogenannte Buffer-Overflow-Angriffe. Buffer-Overflow-Angriffe nutzen Schwachstellen im Programmcode aus. Ansatzpunkt sind Programme, die Daten einlesen und in Variablen abspeichern, ohne zu Überprüfen, ob die Größe der Variablen eingehalten wird. Bei Überschreitung der vorgesehenen Variablengröße kann der Angreifer andere Bereiche des Arbeitsspeichers überschreiben und somit weiterführende Befehle ausführen [Eck18].

Backdoors sind eine weitere Art, sich Zugang zu einem System zu verschaffen. Hier programmiert der Entwickler einer Software einen weiteren (verborgenen) Zugang zu dieser Software. Dies kann im einfachsten Fall durch eine vordefinierte Kombination aus Benutzername und Passwort geschehen [KAG+18].

Eine weitere Angriffsart ist das Versenden von Spam E-Mails. Diese enthalten oftmals Schadsoftware wie Würmer oder Trojanische Pferde, werden jedoch nur durch explizites Öffnen eines Anwenders ausgeführt [Eck18].

Teil II.

Behandlung heterogener Daten

5. Das heterogene Abstandsmaß ConDist

Dieses Kapitel beschäftigt sich mit der Berechnung von Abständen zwischen Datenpunkten mit heterogenen Attributen und stellt einen Lösungsansatz für den Umgang mit heterogenen Daten (siehe Herausforderung 1 in Abschnitt 1.2.1) vor. Data-Mining-Algorithmen sind häufig darauf ausgelegt ausschließlich kontinuierliche oder ausschließlich kategorische Attribute zu verarbeiten. Im Bereich IT-Sicherheit liegen jedoch meist heterogene Daten vor. Heterogene Daten bestehen aus kontinuierlichen und kategorischen Attributen und erschweren somit die Anwendung diverser Data Mining-Algorithmen. Dieses Kapitel stellt das Abstandsmaß ConDist [ROB+15] und dessen Erweiterung zur automatischen Schwellwertberechnung [RLH15] vor. ConDist ist ein unüberwachtes Abstandsmaß für Daten mit heterogenen Attributen und kann paarweise Abstände zwischen Datenpunkten berechnen. Die nachfolgenden Inhalte basieren auf den Publikationen [ROB+15] und [RLH15].

5.1. Einleitung

Eine wesentliche Aufgabe im Bereich Data Mining ist die Abstandsberechnung zwischen zwei Datenpunkten [TSK06, SAW15]. In diesem Kapitel stellt jeder Datenpunkt ein Objekt mit vordefinierten Attributen (Merkmalen) dar. Folglich können Datenpunkte $\boldsymbol{O_i} \in D$ als m-dimensionale Vektoren $\boldsymbol{O_i} = (o_{i_1}, o_{i_2}, ..., o_{i_m})$ repräsentiert werden. Die Menge aller Datenpunkte sei D.

Problem. Kontinuierliche Attribute besitzen eine Ordnungsrelation und häufig wird der in Abschnitt 2.4.1 vorgestellte Minkowski-Abstand zur Abstandsberechnung verwendet [ASN14]. Die Abstandsberechnung für kategorische Attribute hingegen ist eine nicht triviale Herausforderung, da kategorische Attribute keine natürliche Ordnungsrelation besitzen [BCK08, PSL+15]. Dennoch existieren einige Methoden zur Berechnung von Abständen zwischen kategorischen Werten. Eine mögliche Methode ist der Hamming-Abstand (siehe Abschnitt 2.4.2), welcher kategorische Attributwerte lediglich auf Gleichheit oder Ungleichheit prüft. Der Abstand identischer Attributwerte ist 0 und der Abstand nicht identischer Attributwerte 1. Eine weitere Möglichkeit besteht darin, kategorische Attribute in binäre Attribute zu transformieren. Hierbei wird aus jeder möglichen Ausprägung eines kategorischen Attributs ein neues binäres Attribut erstellt [ASN14]. Ein offensichtlicher Nachteil dieser beiden Ansätze ist, dass diese den Grad der Ähnlichkeit zwischen zwei verschiedenen kategorischen Ausprägungen nicht widerspiegeln [HW06].

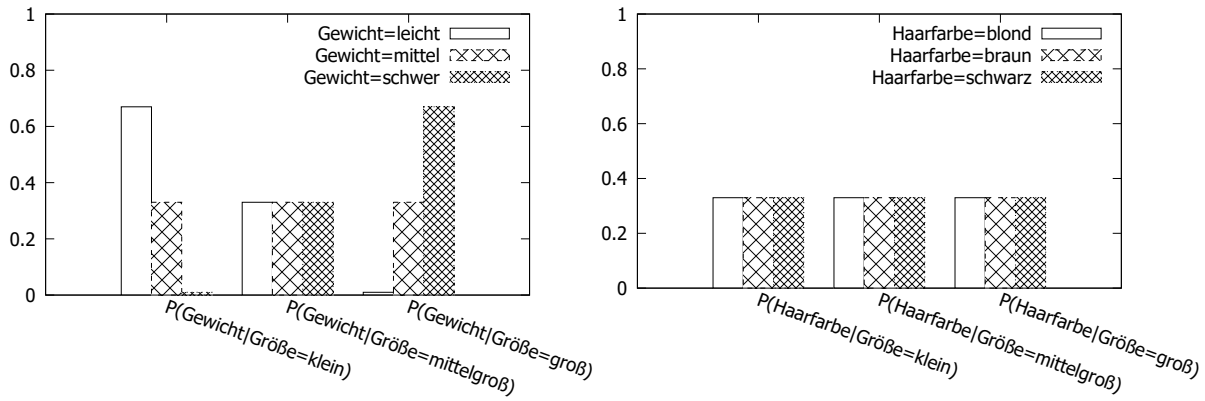
Ziel. Primäres Ziel dieses Kapitels ist die Entwicklung eines heterogenen Abstandsmaßes, welches den Grad der Ähnlichkeit zwischen zwei verschiedenen kategorischen Ausprägungen widerspiegeln kann. Das Abstandsmaß soll auf beliebige Datensätze angewendet und auch in unüberwachten Szenarien eingesetzt werden können. Folglich darf auf keine Klassenlabels der Datenpunkte zurückgegriffen und kein anwendungsspezifisches Wissen über die zugrundeliegenden Daten in das Abstandsmaß modelliert werden.

Ansatz und Beiträge. Das in diesem Kapitel vorgestellte Abstandsmaß ConDist (Context-based Categorical Distance Measure) verwendet Informationen aus der zur Verfügung stehenden Datenmenge zur Berechnung von Abständen zwischen kategorischen Werten. ConDist berech-

net den Abstand zwischen zwei Datenpunkten als die gewichtete Summe der Abstände pro Attribut. Hierfür nutzt ConDist existierende Korrelationen zwischen Attributen aus. Um die Abstände zwischen zwei kategorischen Werten in einem Attribut (im Folgenden als Zielattribut bezeichnet) zu berechnen, identifiziert ConDist zunächst korrelierte Kontextattribute. Die Abstandsberechnung für das Zielattribut basiert auf der Idee, dass ähnliche Attributwerte im Zielattribut eine ähnliche Verteilung in korrelierten Kontextattributen aufweisen. Unähnliche Attributwerte im Zielattribut sollten hingegen unterschiedliche Verteilungen in korrelierten Kontextattributen aufweisen. Kernidee von ConDist ist somit die Berücksichtigung von korrelierten Kontextattributen, um relevante Abstandsinformationen aus der vorhandenen Datenmenge zu extrahieren. Gleichzeitig reduziert ConDist den Einfluss von unkorrelierten Kontextattributen, da ConDist auch die Qualität der Informationen berücksichtigt, die aus korrelierten Attributen extrahierbar sind. Le und Ho [LH05] zeigten in einer empirischen Studie, dass die Attribute einer Datenmenge häufig in Korrelation stehen. Diese Tatsache wird weiterhin dadurch unterlegt, dass es ganze Forschungsfelder gibt, die sich mit dem Auffinden und Beseitigen von Korrelationen beschäftigen (zum Beispiel Merkmalsauswahl [GE03, TAL14, RE15].

Tabelle 5.1.: Beispieldatensatz bestehend aus 9 Personen nach [ROB+15].

#	Größe	Gewicht	Haarfarbe
1	klein	leicht	blond
2	klein	leicht	braun
3	klein	mittel	schwarz
4	mittelgroß	leicht	schwarz
5	mittelgroß	mittel	braun
6	mittelgroß	schwer	blond
7	groß	mittel	blond
8	groß	schwer	braun
9	groß	schwer	schwarz



(a) BWV des Attributs Gewicht (G)

(b) BWV des Attributs Haarfarbe (F)

Abbildung 5.1.: Verteilung der Daten im Beispieldatensatz nach [ROB+15]. Diese Abbildung zeigt die bedingten Wahrscheinlichkeitsverteilungen (BWV) für die Kontextattribute Gewicht und Haarfarbe bei unterschiedlichen Ausprägungen im Zielattribut Größe.

Die grundlegende Idee von ConDist soll mit folgenden Beispiel erläutert werden. Jede Person aus Tabelle 5.1 sei durch drei kategorische Attribute beschrieben. Die Tatsache, dass Größe und Gewicht ordinale Attribute sind, wird ignoriert und nur für Evaluationszwecke genutzt. Im Folgenden sollen die Abstände zwischen den Ausprägungen des Attributs Größe berechnet werden, das heißt, Größe ist in diesem Fall das Zielattribut. Wie bereits erwähnt, verwendet ConDist die bedingten Wahrscheinlichkeitsverteilungen von korrelierten Kontextattributen zur Abstandsberechnung. Für die Attribute Gewicht und Haarfarbe wären dies die bedingten Verteilungen $P(Y|\text{Größe=klein})$, $P(Y|\text{Größe=mittelgroß})$ und $P(Y|\text{Größe=groß})$, welche auch in Abbildung 5.1 zu sehen sind. Abbildung 5.1a zeigt, dass das Attribut Gewicht in Korrelation zum Attribut Größe steht, da in der Regel größere Personen ein höheres Gewicht aufweisen. Im Gegensatz dazu gibt es keine signifikante Korrelation zwischen den Attributen Haarfarbe und Größe. Die bedingten Wahrscheinlichkeitsverteilungen sind somit für das Attribut Haarfarbe identisch für alle Werte des gegebenen Zielattributs Größe (siehe Abbildung 5.1b). Folglich liefert das Attribut Haarfarbe keine Zusatzinformationen für die Abstandsberechnung. Aus diesem Grund empfiehlt es sich nicht, das Kontextattribut Haarfarbe für die Abstandsberechnung des Zielattributs Größe zu berücksichtigen. Im Vergleich dazu sind die bedingten Wahrscheinlichkeitsverteilungen für das Attribut Gewicht unterschiedlich und es trägt Informationen zur Abstandsberechnung bei (siehe Abbildung 5.1a).

Dieses Kapitel stellt das unüberwachte Abstandsmaß ConDist vor, welches Abstände zwischen Datenpunkten mit heterogenen Attributen berechnen kann. Zur Berechnung von Abständen zwischen kategorischen Werten greift ConDist auf statistische Zusammenhänge aus dem zugrundeliegenden Datensatz in Form von Korrelationen zwischen den Attributen zurück und versucht somit die fehlende inhärente Ordnungsrelation kategorischer Attribute zu kompensieren.

5.2. Verwandte Arbeiten

Dieser Abschnitt untersucht existierende kategorische Abstandsmaße. Grundsätzlich lassen sich Abstandsmaße in überwacht und unüberwacht einteilen. Überwachte Abstandsmaße nutzen Klassenzugehörigkeiten aus, um Abstände zwischen kategorischen Ausprägungen zu erlernen. Unüberwachte Abstandsmaße verwenden keine Informationen über Klassenzugehörigkeiten bei der Abstandsberechnung. Deshalb können unüberwachte Abstandsmaße generell und überwachte Abstandsmaße nur in Kombination mit gelabelten Trainingsdaten verwendet werden. Da ConDist ein unüberwachtes Abstandsmaß ist, fokussiert die Analyse ausschließlich auf unüberwachte und keine überwachten Abstandsmaße wie VDM [SW86], HVDM [WM97] oder KDML [HCC+13]. Im Folgenden werden unüberwachte Abstandsmaße in drei Kategorien unterteilt: (I) Abstandsberechnung ohne Berücksichtigung von Kontextattributen, (II) Abstandsberechnung unter Berücksichtigung aller Kontextattribute und (III) Abstandsberechnung unter Berücksichtigung einer Teilmenge von Kontextattributen.

Eine Übersicht von Abstandsmaßen der Kategorie (I) ist in Boriah et al. [BCK08] zu finden. Im Gegensatz zu ConDist ignorieren diese Abstandsmaße verfügbare Kontextattribute. Ein Beispiel ist das Abstandsmaß Eskin [EAP+02], welches auf die Kardinalität des Zielattributs zurückgreift, um Abstände zwischen kategorischen Werten zu definieren. Je größer die Kardinalität des Zielattributs, desto geringer ist der Abstand verschiedener Werte. Die Abstandsmaße Gambaryan [Gam64] oder Goodall [Goo66] berücksichtigen die Wahrscheinlichkeiten der konkreten Attributwerte zur Abstandsberechnung. Diese und noch weitere Abstandsmaße wurden in Boriah et al. [BCK08] hinsichtlich Ausreißererkennung evaluiert, mit dem Ergebnis, dass kein Abstandsmaß signifikant besser ist.

Kategorie (II) beinhaltet kategorische Abstandsmaße, die alle Kontextattribute berücksichtigen, unabhängig davon, ob diese korreliert oder unkorreliert sind. Li und Ho [LH05] präsentieren ein Abstandsmaß dieser Kategorie. Deren Abstandsmaß berechnet zunächst die bedingten Wahrscheinlichkeitsverteilungen $P(Y|X)$ für alle Kontextattribute Y. Anschließend berechnen sich die Abstände im Attribut X basierend auf den Ähnlichkeiten der bedingten Wahrscheinlichkeitsverteilungen. Ein ähnlicher Ansatz wird in [AD07] verwendet. Das Abstandsmaß aus [AD07] definiert den Abstand zweier kategorischer Werte im Zielattribut basierend auf den gemeinsamen Wahrscheinlichkeitsverteilungen der Kontextattribute.

Abstandsmaße der Kategorie (III) verwenden nur eine Teilmenge der zur Verfügung stehenden Kontextattribute. DILCA [IPM09] ist ein Vertreter dieser Kategorie und verwendet Symmetric Uncertainty (SU) [YL03] für die Selektion geeigneter Kontextattribute. SU berechnet die Korrelation zwischen zwei Attributen. Eine ähnliche Vorgehensweise findet bei CBDL [KHH11] statt. CBDL verwendet unterschiedliche Methoden wie SU oder Chi-Square Test zur Identifikation von korrelierten Kontextattributen. Jia und Cheung [JCL14] stellen ein weiteres Abstandsmaß dieser Kategorie vor. Zur Berechnung und Gewichtung von korrelierten Kontextattributen verwendet das Abstandsmaß den Normalized Mutual Information (NMI) [ACW+05] (siehe Kapitel 2.5.6). Eine Besonderheit von [JCL14] im Vergleich zu anderen Abstandsmaßen ist, dass die Abstände zwischen zwei kategorischen Ausprägungen in einem Attribut je nach Abhängigkeit der konkreten Ausprägungen in den anderen Attributen der jeweiligen Datenpunkte variieren können.

ConDist ignoriert keine Kontextattribute wie Kategorie (I) und verwendet auch nicht alle verfügbaren Kontextattribute wie Kategorie (II). Stattdessen lässt sich ConDist in Kategorie (III) einordnen, fügt jedoch den ausgewählten Kontextattributen zusätzliche Gewichtungsfaktoren hinzu. Darüber hinaus wird das Zielattribut selbst in die Abstandsberechnung mit einbezogen. Im Gegensatz zum Abstandsmaß von Jia und Cheung [JCL14], welches ebenfalls Gewichtungsfaktoren verwendet, haben zwei bestimmte Werte innerhalb eines Attributs stets den gleichen Abstand, unabhängig von den dazugehörigen Datenpunkten. Dies ermöglicht ConDist die Berechnung einer paarweisen Abstandsmatrix pro Attribut, was zu geringeren Rechenaufwänden und somit zu Geschwindigkeitsoptimierungen bei der Abstandsberechnung führt.

5.3. Definition von ConDist

Dieser Abschnitt definiert das heterogene Abstandsmaß ConDist. Zunächst wird die grundlegende Idee und die Hauptformel vorgestellt (Abschnitt 5.3.1). Da ConDist zunächst den Abstand pro Attribut berechnet, ist eine Abstandsfunktion pro Attribut erforderlich, welche in Abschnitt 5.3.2 definiert wird. Der Abstand zweier Datenpunkte ergibt sich aus der gewichteten Summe der Abstände pro Attribut. Die verwendete Gewichtungsfunktion ist in Abschnitt 5.3.3 beschrieben. Die Abstandsfunktion einzelner Attribute und die Gewichtungsfunktion berücksichtigen korrelierte Kontextattribute. Abschnitt 5.3.4 definiert, wie die Menge korrelierter Kontextattribute abgeleitet und wie eine entsprechende Einflussfunktion berechnet wird, welche die unterschiedliche Menge an nutzbarer Information in Abhängigkeit vom Grad der Korrelation berechnet. Abschnitt 5.3.5 beschreibt den Umgang mit kontinuierlichen Attributen. Zuletzt zeigt Abschnitt 5.3.6, dass ConDist eine Metrik ist.

5.3.1. Hauptformel

Dieser Abschnitt beschreibt die grundlegende Idee und die Hauptformel von ConDist zur Abstandsberechnung zweier Datenpunkte. Angenommen, $\boldsymbol{O_i}$ und $\boldsymbol{O_j}$ sind zwei Datenpunkte aus

einem Datensatz D und jeder Datenpunkt sei durch m Attribute beschrieben. Des Weiteren sei der Wert von Attribut X des Datenpunktes $\boldsymbol{O_i}$ mit o_{i_X} dargestellt. ConDist basiert auf einem zweistufigen Ansatz: Im ersten Schritt wird die Distanz zwischen den Datenpunkten $\boldsymbol{O_i}$ und $\boldsymbol{O_j}$ in allen Attributen separat berechnet und danach werden diese Abstände mit angepassten Gewichten multipliziert und aufsummiert. Formal ist der Abstand zwischen zwei Datenpunkten $\boldsymbol{O_i}$ und $\boldsymbol{O_j}$ als die gewichtete Summe der Abstände über alle Attribute definiert:

$$ConDist(\boldsymbol{O_i}, \boldsymbol{O_j}) = \sum_X \omega(X) \cdot d_X(\boldsymbol{O_i}, \boldsymbol{O_j}), \tag{5.1}$$

wobei $\omega(X)$ den Gewichtungsfaktor für das Attribut X (siehe Abschnitt 5.3.3) und $d_X(\boldsymbol{O_i}, \boldsymbol{O_j})$ den Abstand der Datenpunkte $\boldsymbol{O_i}$ und $\boldsymbol{O_j}$ im Attribut X (siehe Kapitel 5.3.2) darstellt. Die Abstandsfunktion d_X basiert auf der Idee, dass ähnliche Attributwerte ähnliche Verteilungen in korrelierten Kontextattributen besitzen. Der Gewichtungsfaktor ω_X berücksichtigt die Anzahl der korrelierten Kontextattribute und deren Korrelationsgrad zum Zielattribut X. Sowohl die Abstandsfunktion für Attribute d_X als auch die Gewichtungsfunktion ω_X nutzen Korrelationen zwischen Attributen, um verfügbare Informationen eines Datensatzes zu extrahieren.

5.3.2. Abstandsfunktion d_X

Der Abstand $\hat{d}_X$ zweier kategorischer Werte in einem Attribut X basiert auf der Annahme, dass Attributwerte $x \in dom(X)$ ähnlich sind, falls diese mit ähnlichen Wahrscheinlichkeitsverteilungen in korrelierten Kontextattributen Y einhergehen. Um den Abstand zwischen zwei Datenpunkten $\boldsymbol{O_i}$ und $\boldsymbol{O_j}$ im Attribut X zu bestimmen, berechnet d_X zunächst den Euklidischen-Abstand zwischen den bedingten Wahrscheinlichkeitsverteilungen $P(Y|X = o_{i_X})$ und $P(Y|X = o_{j_X})$ für jedes Attribut Y aus der Menge der korrelierten Kontextattribute $context_X$ des Zielattributs X. Anschließend wird dieser mit der Einflussfunktion $impact_X(Y)$ (Abschnitt 5.3.4) multipliziert. Dieser Vorgang wird für alle Kontextattribute aus $Y \in context_X$ wiederholt und aufsummiert. Die Einflussfunktion berücksichtigt die Tatsache, dass die extrahierbare Informationsmenge für das Zielattribut X in einem Kontextattribut Y mit einer sehr hohen oder sehr niedrigen Korrelation abnimmt (siehe Abschnitt 5.3.4). Die resultierende Formel[1] für die Abstandsfunktion $\hat{d}_X$ ist wie folgt definiert:

$$\hat{d}_X(\boldsymbol{O_i}, \boldsymbol{O_j}) = \sum_{Y \in context_X} impact_X(Y) \sqrt{\sum_{y \in dom(Y)} \Big(p(y|o_{i_X}) - p(y|o_{j_X})\Big)^2}, \tag{5.2}$$

wobei $dom(Y)$ der Wertebereich des Attributs Y ist, $p(y|o_{i_X}) = p(y|X = o_{i_X})$ die Wahrscheinlichkeit, dass das Kontextattribut Y die Ausprägung $y \in dom(Y)$ annimmt, unter der Bedingung, dass der Wert $o_{i_X} \in dom(X)$ im Datensatz D beobachtet werden kann. Wie bereits erwähnt, verwendet $\hat{d}_X$ eine Menge von korrelierten Kontextattributen $context_X$.

Die Definition von $context_X$ ist in Abschnitt 5.3.4 beschrieben. Da jedes Attribut mit sich selbst in Korrelation steht, ist das Zielattribut X stets in der Menge der Kontextattribute vorhanden. Die Motivation für die Berücksichtigung des Zielattributs ist auf zwei Ursachen zurückzuführen. Zum einen stellt dies sicher, dass die Menge der Kontextattribute $context_X$

[1] In den Experimenten von Ring et al. [ROB+15] wurde aufgrund eines Programmierfehlers die Formel $\hat{d}_X(\boldsymbol{O_i}, \boldsymbol{O_j})$ wie folgt berechnet:

$$\sum_{Y \in context_X} impact_X(Y) \sqrt{\sum_{y \in dom(Y), p(y|o_{i_X}) > 0} \Big(p(y|o_{i_X}) - p(y|o_{j_X})\Big)^2 + \sum_{y \in dom(Y), p(y|o_{j_X}) > 0} \Big(p(y|o_{i_X}) - p(y|o_{j_X})\Big)^2}$$

nicht leer ist, auch wenn alle Attribute voneinander unabhängig sind. Zum anderen garantiert die Aufnahme des Zielattributs X, dass zwei unterschiedliche Werte eines kategorischen Attributs stets einen Abstand größer als 0 besitzen.

Es ist zu beachten, dass ConDist den berechneten Abstand $\hat{d}_X(\boldsymbol{O_i}, \boldsymbol{O_j})$ aus Gleichung 5.2 wie folgt normiert:

$$d_X(\boldsymbol{O_i}, \boldsymbol{O_j}) = \frac{\hat{d}_X(\boldsymbol{O_i}, \boldsymbol{O_j})}{d_{X,max}} \tag{5.3}$$

Gleichung 5.3 zeigt die von ConDist genutzte Normierung auf das Intervall $[0, 1]$, wobei $d_{X,max}$ den maximal auftretenden Abstand innerhalb des Zielattributs X darstellt. Für den Fall, dass keine korrelierten Kontextattribute identifiziert werden können, ergibt sich für alle Kombination von unterschiedlichen Ausprägungen im Zielattribut X der gleiche (maximale) Abstand. In diesem Fall reduziert sich ConDist auf den Hamming Abstand und unterscheidet lediglich zwischen Gleichheit und Ungleichheit kategorischer Werte.

5.3.3. Gewichtungsfunktion $\omega(X)$

Die Gewichtungsfunktion $\omega(X)$ ist insbesondere für Datensätze erforderlich, in denen einige, jedoch nicht alle Attribute in Korrelation stehen. Diese Notwendigkeit wird im Folgenden mithilfe des Beispiels aus Tabelle 5.1 erläutert. In diesem Beispiel existiert für das Attribut Haarfarbe kein korreliertes Kontextattribut. Folglich führen die normierten Ergebnisse von Gleichung 5.2 immer zu der maximalen Distanz 1 für alle nicht identischen Attributwerte. Für das Attribut Gewicht identifiziert ConDist das korrelierte Kontextattribut Größe. Als Folge dessen kann ConDist sinnvollere Abstände für das Attribut Gewicht als für das Attribut Haarfarbe berechnen. Allerdings sind die durchschnittlichen Abstände im Attribut Haarfarbe größer als im Attribut Gewicht. Folglich haben unterschiedliche Werte im Attribut Haarfarbe implizit einen größeren Einfluss auf die Abstandsberechnung als unterschiedliche Werte im Attribut Gewicht. Um dieses Problem zu lösen, weist die Gewichtungsfunktion $\omega(X)$ jedem Attribut X eine Gewichtung zu, welche auf (I) der Anzahl der identifizierten korrelierten Kontextattribute und (II) deren Auswirkungen auf das Zielattribut X (siehe Gleichung 5.4) basiert.

$$\omega(X) = 1 + \frac{\sum_{Y \in context_X} impact_X(Y)}{m \cdot c}, \tag{5.4}$$

wobei $context_X$ und $impact_X(Y)$ wie in Kapitel 5.3.4 definiert sind und m die Anzahl der Attribute ist. Der Parameter c ist eine Konstante und stellt einen Normierungsfaktor dar. Der Wert des Parameters c entspricht dem Maximum der Einflussfunktion $impact_X(Y)$. Die Funktion $impact_X(Y)$ wird in Kapitel 5.3.4 hergeleitet und das Maximum dieser Funktion ist unabhängig von den zugrundeliegenden Daten und entspricht $\frac{8}{27}$.

5.3.4. Korrelation, Kontext und Einfluss

Die Abstandsfunktion für Attribute d_X (Kapitel 5.3.2) und die Gewichtungsfunktion ω_X (Kapitel 5.3.3) verwenden den Grad der Korrelation zwischen kategorischen Attributen sowie eine zusätzliche Einflussfunktion. Diese werden im Folgenden definiert.

Korrelation $cor(X|Y)$**.** Für die Identifikation geeigneter Kontextattribute ist ein Korrelationsmaß notwendig. ConDist verwendet ein Korrelationsmaß basierend auf der Grundlage des in der Informationstheorie weit verbreiteten Informationsgewinns (IG) [LR05]. Der IG berechnet sich wie folgt:

$$IG(X|Y) = H(X) - H(X|Y), \tag{5.5}$$

wobei $H(X)$ die Entropie des Attributs X und $H(X|Y)$ die bedingte Entropie des Attributs X gegeben Attribut Y ist. Der Informationsgewinn $IG(X|Y)$ ist immer kleiner oder gleich der Entropie $H(X)$. Basierend auf dieser Beobachtung ist die Funktion $cor(X|Y)$ wie folgt definiert:

$$cor(X|Y) = \frac{IG(X|Y)}{H(X)}, \tag{5.6}$$

und beschreibt somit ein auf das Intervall $[0, 1]$ normiertes Korrelationsmaß. Die Qualität möglicher Schlussfolgerungen aus einem Kontextattribut Y zum Zielattribut X kann sich von der Qualität der Schlussfolgerungen aus dem Kontextattribut X zum Zielattribut Y unterscheiden. Diesen Aspekt berücksichtigt die asymmetrische Korrelationsfunktion $cor(X|Y)$ und ermöglicht es somit, stets die maximale Menge an nützlicher Information zu extrahieren.

Kontext $context_X$**.** Sowohl die Abstandsfunktion für Attribute d_X (Kapitel 5.3.2) in ConDist als auch die Gewichtungsfunktion ω_X (Kapitel 5.3.3) basieren auf einer Menge von korrelierten Kontextattributen $context_X$. Diese Menge wird mithilfe der zuvor definierten Korrelationsfunktion $cor(X|Y)$ und mit einem benutzerdefinierten Schwellenwert θ ermittelt. Das heißt, ein Kontextattribut Y ist nur dann in $context_X$ enthalten, wenn dessen Korrelation mit dem Zielattribut X größer oder gleich den benutzerdefinierten Schwellenwert θ ist, siehe Gleichung 5.7.

$$context_X = \{Y \mid cor(X|Y) \geq \theta\} \tag{5.7}$$

Einflussfunktion $impact_X(Y)$**.** Sowohl die Abstandsfunktion für Attribute d_X (Kapitel 5.3.2) als auch die Gewichtungsfunktion ω_X (Kapitel 5.3.3) verwenden eine sogenannte Einflussfunktion $impact_X(Y)$. Diese Funktion steuert auf Basis der Korrelationsfunktion $cor(X|Y)$ den Einfluss eines Kontextattributs Y auf das Zielattribut X.

Ein hoher Korrelationswert bedeutet, dass eine Ausprägung des Kontextattributs $Y \in context_X$ den Wert des Zielattributs X mit hoher Wahrscheinlichkeit impliziert. Hier sei auf das Beispiel aus Tabelle 5.1 verwiesen. Wenn bekannt ist, dass eine Person ein leichtes Körpergewicht aufweist, dann ist es wahrscheinlicher, dass diese Person klein ist. Somit gibt es im Extremfall von perfekt korrelierten Attributen keine Überschneidungen der bedingten Wahrscheinlichkeitsverteilungen $P(Y|X = o_{i_X})$ und $P(Y|X = o_{j_X})$ für $o_{i_X} \neq o_{j_X}$.

Ein niedriger Korrelationswert bedeutet hingegen, dass der Wert eines Kontextattributs $Y \in context_X$ wenig bis keine Präferenz für einen bestimmten Wert des Zielattributs X impliziert. Dies bedeutet, dass der Abstand zwischen den bedingten Wahrscheinlichkeitsverteilungen $P(Y|X = o_{i_X})$ und $P(Y|X = o_{j_X})$ zufällig sein kann, das heißt, eventuell sogar zufällige oder falsche Informationen beisteuert. Deshalb verwirft ConDist nicht korrelierte Kontextattribute, um den Einfluss falscher Informationen zu vermeiden. Im Gegensatz dazu tragen perfekt korrelierte Attribute keine falschen Informationen zur Abstandsberechnung bei. Da diese jedoch ausschließlich hohe Distanzen für unterschiedliche Ausprägungen beisteuern, sollte deren Einfluss reduziert werden. Andernfalls reduziert sich der durch andere Kontextattribute ermittelte Abstand.

Aus diesem Grund verwendet ConDist eine Einflussfunktion $impact_X(Y)$, die (I) schnell mit zunehmender Korrelation zwischen Attributen steigt, (II) langsam mit bestehender, teilweiser Korrelation steigt und (III) bei nahezu perfekter Korrelation abnimmt. In einer experimentellen Studie hat sich folgende Einflussfunktion bewährt, welche in Abbildung 5.2 dargestellt und wie in Formel 5.8 definiert ist.

$$impact_X(Y) = cor(X|Y)\Big(1 - 0{,}5 \cdot cor(X|Y)\Big)^2. \tag{5.8}$$

Grundsätzlich lässt sich die Einflussfunktion durch andere Funktionen ersetzen, die die drei oben definierten Eigenschaften erfüllen.

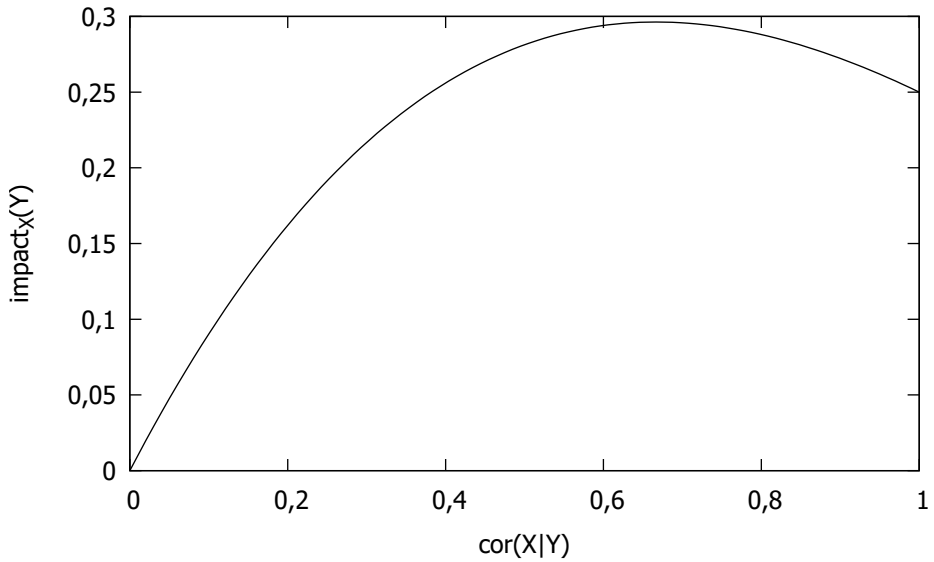


Abbildung 5.2.: Verlauf der Einflussfunktion $impact_X(Y)$ [ROB+15].

5.3.5. Kontinuierliche Attribute

Viele reale Datensätze enthalten sowohl kontinuierliche als auch kategorische Attribute. Um *ConDist* auf solche Datensätze anwenden zu können, sind zwei Situationen zu unterscheiden. Entweder ist das Zielattribut oder das Kontextattribut kontinuierlich.

Wenn das Zielattribut kontinuierlich ist, sind keine Kontextattribute erforderlich. In diesem Fall nutzt ConDist den Minkowski-Abstand (siehe Abschnitt 2.4.1) zur Abstandsberechnung zweier Werte. Da ConDist den Abstand zwischen kategorischen Werten auf das Intervall $[0, 1]$ normiert, sollten die Wertebereiche kontinuierlicher Zielattribute ebenfalls normiert werden. Die Normierung dient dazu, dass kontinuierliche Attribute keinen wesentlich größeren (bzw. kleineren) Einfluss auf die Abstandsberechnung zweier Objekte $\boldsymbol{O_i}$ und $\boldsymbol{O_j}$ besitzen, wenn deren Wertebereich sehr groß (bzw. klein) ist. Da kontinuierliche Attribute die Berechnung aussagekräftiger Abstände erlauben, sollte die Gewichtungsfunktion $\omega(X)$ (siehe Abschnitt 5.3.3) auf den Maximalwert gesetzt werden.

Der unendliche Wertebereich eines kontinuierlichen Attributs verhindert dessen direkte Nutzung als Kontextattribut. Deshalb sieht ConDist die Diskretisierung kontinuierlicher Kontextattribute vor. Die im Zuge der Diskretisierung entstandenen Intervalle können anschließend als kategorische Werte verarbeitet werden. Die Diskretisierung kann durch verschiedene Algorithmen

wie TUBE [SF05], Multi-Intervall Diskretisierung [FI93] oder Hellinger-Diskretisierung [Lee07] erfolgen.

5.3.6. Beweis Metrik

Im Folgenden wird der Beweis erbracht, dass es sich bei ConDist um eine Metrik handelt. Zunächst wird die allgemeine Definition einer Metrik eingeführt und anschließend für die Funktion $\hat{d}_X$ (Gleichung 5.2) bewiesen. Da ConDist (siehe Gleichung 5.1) eine gewichtete Summe verschiedener $\hat{d}_X$ ist und jedes $\hat{d}_X$ eine Metrik darstellt, ist ConDist ebenfalls eine Metrik.

5.3.6.1. Allgemeine Definition einer Metrik

Ω sei eine beliebige Menge mit $a, b, c \in \Omega$. Die Funktion $d_{metrik}(a, b)$ ist eine Metrik auf Ω, wenn nach Cleve und Lämmel [CL14] die folgenden vier Bedingungen erfüllt sind:
(1) Identität: $d_{metrik}(a, b) = 0 \Leftrightarrow a = b$
(2) Positive Definitheit: $d_{metrik}(a, b) \geq 0$
(3) Symmetrie: $d_{metrik}(a, b) = d_{metrik}(b, a)$
(4) Dreiecksungleichung: $d_{metrik}(a, b) \leq d_{metrik}(a, c) + d_{metrik}(c, b)$

Lemma.

$$d_m(a, b) = \sum_{Y \in C} g_Y \cdot d_{metrik}\Big(f_Y(a), f_Y(b)\Big) \tag{5.9}$$

Die Funktion $d_m(a, b)$ aus Gleichung 5.9 ist eine Metrik, wenn folgende Bedingungen gelten:

- f_Y ist eine beliebige mathematische Funktion aus $\mathbb{R}$,
- $g_Y >= 0 \; \forall \; Y \in C$,
- $\exists \; Y' \in C$ mit $g_{Y'} > 0$ und mindestens ein $f_{Y'}$ ist injektiv und
- d_{metrik} ist eine Metrik.

Beweis. Für das obige Lemma werden nun die obigen vier Bedingungen nach Cleve und Lämmel [CL14] gezeigt.

- zu (1)
  - „=>“ wenn $a = b$, dann ist $f_Y(a) = f_Y(b) \; \forall \; Y \in C$, da f_Y eine Funktion ist. Daraus folgt $d_m(a, b) = 0$.
  - „<=“ falls $d_m(a, b) = 0$, dann gilt $\sum_{Y \in C} g_Y \cdot d_{metrik}(f_Y(a), f_Y(b)) = 0$. Daraus folgt $g_Y \cdot d_{metrik}(f_Y(a), f_Y(b)) = 0 \; \forall \; Y$. Da es mindestens ein Y' mit $g_{Y'} > 0$ gibt, folgt dass $d_{metrik}(f_{Y'}(a), f_{Y'}(b)) = 0$ sein muss und somit $f_{Y'}(a) = f_{Y'}(b)$, da $f_{Y'}$ injektiv ist. Daraus folgt $a = b$.
- zu (2) $g_Y \geq 0 \; \forall \; Y \in C$ und d_{metrik} ist eine Metrik
- zu (3) Symmetrie ist gegeben, da d_{metrik} eine Metrik ist

- zu (4) es gilt zu zeigen, dass $d_m(a,b) \leq d_m(a,c) + d_m(c,b)$ gilt.
  $d_m(a,b) \leq d_m(a,c) + d_m(c,b)$
  $\Leftrightarrow 0 \leq d_m(a,c) + d_m(c,b) - d_m(a,b)$
  $\Leftrightarrow 0 \leq \sum_{Y \in C} g_Y \cdot d_{metrik}\Big(f_Y(a), f_Y(c)\Big) + \sum_{Y \in C} g_Y \cdot d_{metrik}\Big(f_Y(c), f_Y(b)\Big) - \sum_{Y \in C} g_Y \cdot d_{metrik}\Big(f_Y(a), f_Y(b)\Big)$
  $\Leftrightarrow 0 \leq \sum_{Y \in C} g_Y \cdot \Big[d_{metrik}\Big(f_Y(a), f_Y(c)\Big) + d_{metrik}\Big(f_Y(c), f_Y(b)\Big) - d_{metrik}\Big(f_Y(a), f_Y(b)\Big)\Big]$
  Da $d_{metrik}(a,b)$ eine Metrik und f_Y eine Funktion sind, gilt $\forall\ Y_* \in C$:
  $d_{metrik}\Big(a,b\Big) \leq d_{metrik}\Big(a,c\Big) + d_{metrik}\Big(c,b\Big)$
  $\Leftrightarrow 0 \leq d_{metrik}\Big(a,c\Big) + d_{metrik}\Big(c,b\Big) - d_{metrik}\Big(a,b\Big)$
  $\Leftrightarrow 0 \leq d_{metrik}\Big(f_{Y*}(a), f_{Y*}(c)\Big) + d_{metrik}\Big(f_{Y*}(c), f_{Y*}(b)\Big) - d_{metrik}\Big(f_{Y*}(a), f_{Y*}(b)\Big)$
  // q.e.d.

5.3.6.2. Beweis ConDist

Satz. $\hat{d}_X$ ist eine Metrik auf $dom(X)$ mit $o_{i_X}, o_{j_X} \in dom(X)$.
Beweis. Setze

- $g_Y := impact_X(Y)$,
- $f_Y := \begin{pmatrix} p(y_1|o_{i_X}) \\ p(y_2|o_{i_X}) \\ ... \\ p(y_l|o_{i_X}) \end{pmatrix}\ \forall\ Y \in context_X$ mit $l = |dom(Y)|$,
- $C := context_X$,
- $d_{metrik} := \|\cdot\|_2$ und
- $Y' := X$ mit $g_{Y'} > 0$ und $f_{Y'}$ ist injektiv

Dann gilt: $\hat{d}_X(a,b) = \sum_{Y \in C} g_Y \cdot d_{metrik}\Big(f_Y(a), f_Y(b)\Big)\ \forall\ a,b \in dom(X)$. Für $X \in context_X$ gilt: $impact_X(X) = 0{,}25 > 0$. Weiterhin ist f_Y injektiv auf $dom(X)$, da $p(y_k|o_{i_X}) = 1$ für $y_k = o_{i_X}$ und $p(y_k|o_{i_X}) = 0\ \forall\ y_k \neq o_{i_X}$. Da $X \in context_X$ setze $Y' := X$. Für Y' gilt somit $g_{Y'} = impact_X(Y') > 0$ und $f_{Y'}$ ist injektiv auf $dom(X)$. Für alle $a, b \in dom(X)$ gilt: $f_{Y'}(a) = f_{Y'}(b)$ „=>“ $a = b$, da
für $a = b$ gilt: $p(a|a) = p(a|b) = p(b|a) = p(b|b) = 1$ und
für $a \neq b$ gilt: $p(a|b) = p(b|a) = 0$.

5.4. Automatische Schwellwertberechnung

Im vorherigen Abschnitt 5.3 wurde das Abstandsmaß ConDist zur Berechnung von Abständen zwischen zwei Datenpunkten mit heterogenen Attributen vorgestellt. Dieser Abschnitt erweitert ConDist dahingehend, dass der benutzerdefinierte Parameter θ entfernt und automatisiert aus

den Daten erlernt wird. Dieser Abschnitt wurde bereits in der Publikation *„Automatic Threshold Calculation for the Categorical Distance Measure ConDist“* [RLH15] veröffentlicht.

Abschnitt 5.4.1 erläutert zunächst, warum ConDist einen zusätzlichen Schwellwertparameter θ benötigt. Anschließend definiert Abschnitt 5.4.2 die Vorgehensweise zur automatischen Schwellwertberechnung.

5.4.1. Problemstellung

ConDist extrahiert Informationen von korrelierten Kontextattributen zur Berechnung von Abständen im Zielattribut. Die Berechnung der Korrelation basiert auf dem Informationsgewinn und ConDist berechnet diesen mit der Funktion $cor(X|Y)$. Theoretisch sollte ConDist's Einflussfunktion $impact_X(Y)$ (siehe Abschnitt 5.3.4) den Einfluss von Kontextattributen ohne zusätzlichen Schwellwert regulieren. Die Experimente aus Abschnitt 5.5 zeigen jedoch, dass ein zusätzlicher Schwellwert θ erforderlich ist.

Die Einflussfunktion $impact_X(Y)$ (Abschnitt 5.3.4) kontrolliert den Einfluss von Kontextattributen und ist abhängig von der Korrelationsfunktion $cor(X|Y)$ (siehe Abschnitt 5.3.4). Das folgende Beispiel zeigt eine Situation, in der die beiden Funktionen den Einfluss von Kontextattributen ohne zusätzlichen Schwellwert θ nicht korrekt steuern (siehe Datensatz in Tabelle 5.2).

Tabelle 5.2.: Beispieldatensatz bestehend aus Personen [RLH15].

#	**Geschlecht**	**Haarfarbe**	**Größe**
1	männlich	braun	groß
2	männlich	blond	groß
3	männlich	schwarz	mittelgroß
4	männlich	braun	mittelgroß
5	weiblich	blond	mittelgroß
6	weiblich	schwarz	klein
7	weiblich	braun	klein
8	weiblich	blond	klein

Angenommen, es sollen Abstände für das Zielattribut Größe berechnet werden. In diesem Fall sind Geschlecht und Haarfarbe die verfügbaren Kontextattribute. Für die betrachtete Bevölkerung wird angenommen, dass in Realität die Attribute Haarfarbe und Größe voneinander unabhängig sind und die Attribute Größe und Geschlecht in Korrelation stehen. ConDist's Korrelationsfunktion $cor(X|Y)$ und Einflussfunktion $impact_X(Y)$ berechnen die folgenden Ergebnisse auf den in Tabelle 5.2 dargestellten Datensatz.

$$cor(\text{Größe}|\text{Geschlecht}) = \frac{IG(\text{Größe}|\text{Geschlecht})}{H(\text{Größe})} \approx \frac{1{,}561 - 0{,}906}{1{,}561} \approx 0{,}420 \tag{5.10}$$

$$cor(\text{Größe}|\text{Haarfarbe}) = \frac{IG(\text{Größe}|\text{Haarfarbe})}{H(\text{Haarfarbe})} \approx \frac{1{,}561 - 1{,}439}{1{,}561} \approx 0{,}122 \tag{5.11}$$

$$impact_{\text{Größe}}(\text{Geschlecht}) \approx 0{,}262 \tag{5.12}$$

$$impact_{\text{Größe}}(\text{Haarfarbe}) \approx 0{,}108 \tag{5.13}$$

Wie erwartet, besitzt das Kontextattribut Geschlecht einen größeren Einfluss auf das Zielattribut Größe als das Kontextattribut Haarfarbe. Allerdings hat das Kontextattribut Haarfarbe ebenfalls einen geringen Einfluss auf das Zielattribut Größe. In diesem Beispiel würden die Auswirkungen des stark korrelierten Kontextattributs Geschlecht die Auswirkungen des Kontextattributs Haarfarbe reduzieren. Falls jedoch ausschließlich das Kontextattribut Haarfarbe zur Verfügung stehen würde, hätte dessen geringer Einfluss entscheidende Auswirkungen auf die Abstände innerhalb des Attributs Körpergröße. Dieser Einfluss ist im obigen Beispiel auf die kleine nicht repräsentative Datenmenge zurückzuführen, was zur Folge hat, dass die geschätzten Wahrscheinlichkeitsverteilungen der Korrelationsfunktion $cor(X|Y)$ nicht korrekt sind. Die identifizierten Korrelationen sind konzeptionell nicht beabsichtigt, da das Kontextattribut Haarfarbe vom Zielattribut Größe unabhängig ist. In diesem Fall wäre es besser, das Attribut Haarfarbe nicht als Kontextattribut für das Zielattribut Größe zu berücksichtigen. Deshalb verfügt ConDist über einen benutzerdefinierten Schwellwert θ, um Kontextattribute mit geringen Korrelationen in nicht repräsentativen Datenmengen zu eliminieren.

5.4.2. Datengetriebene Schwellwertberechnung

In diesem Abschnitt wird ein datengetriebener Ansatz zur automatisierten Schwellwertberechnung für ConDist vorgestellt. Während in Ring et al. [ROB+15] ein globaler benutzerdefinierter Schwellwert θ verwendet wurde, berechnet die nachfolgende Methode für jedes Paar von Zielattribut X und Kontextattribut Y einen individuellen Schwellwert $\theta_{X|Y}$. Diese individuellen Schwellwerte können besser auf die spezifischen Korrelationsanforderungen von zwei konkreten Attributen angepasst werden. Die automatische Berechnung berücksichtigt primär die Anzahl der Datenpunkte und die Kardinalität der Attribute. Des Weiteren werden die berechneten Schwellwerte auch bei der Anwendung von ConDists Einflussfunktion berücksichtigt. Die Einflussfunktion kontrolliert den Einfluss von korrelierten Kontextattributen bei der Abstandsberechnung zwischen kategorischen Werten und steuert dadurch die Menge der genutzten Informationen aus einem korrelierten Kontextattribut.

Das Beispiel aus dem vorherigen Abschnitt zeigt, dass nicht repräsentative oder zu kleine Datensätze problematisch für die Berechnung der Korrelation sein können. Grund hierfür ist, dass die Korrelationsfunktion $cor(X|Y)$ auf dem Informationsgewinn $IG(X|Y)$ beruht, der wiederum auf die Entropie des Attributs X und der bedingten Entropie des Attributs X bei gegebenem Attribut Y beruht. Die Entropie $H(X)$ und die bedingte Entropie $H(X|Y)$ sind wie folgt definiert:

$$H(X) = -\sum_{x \in dom(X)} p(x) \log_2 \big(p(x)\big) \text{ und} \tag{5.14}$$

$$H(X|Y) = -\sum_{y \in dom(Y)} p(y) \sum_{x \in dom(X)} p(x|y) \log_2 \big(p(x|y)\big), \tag{5.15}$$

wobei $p(x)$ die Wahrscheinlichkeit des Wertes x und $p(x|y)$ die bedingte Wahrscheinlichkeit des Wertes x bei gegebenem Wert y in Attribut Y im Datensatz D darstellen. Folglich sind die Wahrscheinlichkeitsverteilungen $P(X)$ und $P(Y)$ der Attribute X und Y zu ermitteln. Hier ist jedoch zu beachten, dass diese Wahrscheinlichkeitsverteilungen nicht berechnet, sondern nur mithilfe des vorliegenden Datensatzes durch Abzählen geschätzt werden können. Je größer der zugrundeliegende Datensatz ist, desto realistischer sind die Schätzungen der Wahrscheinlichkeitsverteilungen. Folglich erhöht sich die Wahrscheinlichkeit von Fehlern durch Abzählen für kleine Datensätze.

Zwei Attribute X und Y sind in der Korrelationsfunktion $cor(X|Y)$ genau dann unabhängig (beziehungsweise unkorreliert), wenn gilt:

$$H(X) = H(X|Y) \tag{5.16}$$

Gleichung 5.16 fordert, dass die Wahrscheinlichkeitsverteilung von Attribut X identisch zu den bedingten Wahrscheinlichkeitsverteilungen von Attribut X bei beliebigen Werten $y \in dom(Y)$ sind. Je größer die Kardinalität der Attribute X und Y sind, also $|dom(X)|$ und $|dom(Y)|$, desto mehr Datenpunkte werden benötigt, um die Wahrscheinlichkeitsverteilungen und bedingten Wahrscheinlichkeitsverteilungen korrekt zu schätzen. Folglich sind auch die Kardinalitäten und Wahrscheinlichkeitsverteilungen der kategorischen Attribute X und Y bei der Berechnung des Schwellwerts zu berücksichtigen. Beide Faktoren reflektieren sich in der Entropie eines Attributs. Deshalb berechnet sich der datengetriebene Schwellwert $\theta_{X|Y}$ wie folgt:

$$\theta_{X|Y} = \frac{H(X) \cdot H(Y)}{n}, \tag{5.17}$$

wobei n die Anzahl der Datenpunkte im Datensatz D ist. Der Schwellwert $\theta_{X|Y}$ sinkt mit steigender Anzahl von Datenpunkten und steigt mit zunehmender Kardinalität kategorischer Attribute. Die Berechnung von $\theta_{X|Y}$ ist einfach und ersetzt den benutzerdefinierten Parameter θ.

Wird die soeben vorgestellte Methode zur automatischen Schwellwertberechnung $\theta_{X|Y}$ auf das Beispiel des vorherigen Abschnitts 5.4.1 angewendet, ergeben sich die folgenden Ergebnisse:

$$\theta_{\text{Größe|Geschlecht}} = \frac{H(\text{Größe}) \cdot H(\text{Geschlecht})}{n} \approx \frac{1{,}561 \cdot 1}{8} \approx 0{,}195 \text{ und} \tag{5.18}$$

$$\theta_{\text{Größe|Haarfarbe}} = \frac{H(\text{Haarfarbe}) \cdot H(\text{Größe})}{n} \approx \frac{1{,}561 \cdot 1{,}561}{8} \approx 0{,}305. \tag{5.19}$$

In diesem Fall würde der Wert der Korrelationsfunktion für das Attribut Geschlecht (vergleiche Gleichung 5.10) den Schwellwert $\theta_{\text{Größe|Geschlecht}}$ überschreiten, während der Wert für das Attribut Haarfarbe (vergleiche Gleichung 5.11) den entsprechenden Schwellwert $\theta_{\text{Größe|Haarfarbe}}$ nicht überschreitet. Der datengetriebene Schwellwert $\theta_{X|Y}$ fügt lediglich die Attribute Geschlecht und Größe in die Menge der korrelierten Kontextattribute $context_{\text{Größe}}$ für das Zielattribut Größe hinzu.

Anpassung der Einflussfunktion. Der Schwellwert $\theta_{X|Y}$ kann als Schätzung des Fehlers der Korrelationsfunktion $cor(X|Y)$ angesehen werden, der möglicherweise durch Abzählen der Wahrscheinlichkeitsverteilungen $P(X)$ und $P(Y)$ aus dem Datensatz entstanden ist. Deshalb sollte dieser Wert auch bei ConDist's Einflussfunktion $impact_X(Y)$ Berücksichtigung finden. Gleichung 5.20 zeigt die Anpassung der Einflussfunktion.

$$impact_X(Y) = \begin{cases} 0 & \text{falls } cor(X|Y) < \theta_{X|Y} \\ cor_\theta(X|Y)\Big(1 - \frac{1}{2} cor_\theta(X|Y)\Big)^2 & \text{falls } cor(X|Y) \geq \theta_{X|Y} \end{cases}, \tag{5.20}$$

wobei $cor_\theta(X|Y)$ den aktualisierten auf das Intervall $[0, 1]$ skalierten Korrelationswert entspricht und wie folgt definiert ist:

$$cor_\theta(X|Y) = \frac{cor(X|Y) - \theta_{X|Y}}{1 - \theta_{X|Y}}. \tag{5.21}$$

Wenn die angepasste Einflussfunktion auf das Beispiel aus Tabelle 5.2 angewendet wird, ergeben sich folgende Ergebnisse:

$$cor_\theta(\text{Größe}|\text{Geschlecht}) \approx \frac{0{,}420 - 0{,}195}{1 - 0{,}195} \approx 0{,}280 \tag{5.22}$$

$$impact_{\text{Größe}}(\text{Geschlecht}) \approx 0{,}280\big(1 - 0{,}5 \cdot 0{,}280\big)^2 \approx 0{,}207 \tag{5.23}$$

$$impact_{\text{Größe}}(\text{Haarfarbe}) = 0 \tag{5.24}$$

5.5. Experimente

Dieser Abschnitt führt eine experimentelle Evaluierung von ConDist und dessen Variante mit automatischer Schwellwertberechnung ConDist $\theta_{X|Y}$ durch. Diese Evaluierung findet in den zentralen Anwendungsgebieten der Clusteranalyse und Klassifikation statt. Baselines stellen die kontextbasierten Abstandsmaße DILCA [IPM09], JiaCheung [JCL14] und Quang [LH05] sowie die nicht kontextbasierten Abstandsmaße Eskin [EAP+02], Gambaryan [Gam64], Hamming-Abstand (siehe Abschnitt 2.4.2) und Occurrence Frequency (OF) dar. DILCA wird auf den parameterfreien Ansatz $DILCA_{RR}$ (wie in [IPM09] beschrieben) konfiguriert und für das Abstandsmaß JiaCheung wird der Schwellwertparameter $\beta = 0{,}2$ gesetzt.

5.5.1. Evaluierungsmethode

Klassifikation. In diesem Evaluationsszenario werden die Abstandsmaße mit dem kNN Klassifikator (siehe Abschnitt 3.1.1) evaluiert. Der benutzerdefinierte Parameter Anzahl nächster Nachbarn ist für alle Experimente auf $k = 7$ fixiert. Auf den Datensätzen wird eine 10-fache Kreuzvalidierung (siehe Abschnitt 2.5.3) durchgeführt und die Ergebnisse mit dem Maß Klassifikationsgenauigkeit (siehe Abschnitt 2.5.2) bewertet. Um störende Einflussfaktoren der zufällig erzeugten Teilmengen in der Kreuzvalidierung zu reduzieren, wird diese 100-mal mit unterschiedlichen Teilmengen pro Datensatz wiederholt und der Mittelwert über alle Durchgänge herangezogen. Zusätzlich beinhalten die Ergebnistabellen die Standardabweichung der durchschnittlichen Klassifikationsgenauigkeit über die 100 Durchläufe.

Clusteranalyse. In diesem Evaluationsszenario werden die Abstandsmaße mit dem WARD Algorithmus [WJ63] (siehe Abschnitt 3.2.1) evaluiert. WARD verwendet die jeweiligen Abstandsmaße zur Berechnung der initialen Abstandsmatrix aller Datenpunkte. Die Anzahl zu generierender Cluster ist ein benutzerdefinierter Eingabeparameter. Dieser Parameter wird stets mit der Anzahl der im Datensatz befindlichen Klassen gleichgesetzt. Ergebnisse werden mithilfe des NMI Maßes [SG03] (siehe Abschnitt 2.5.6) bewertet, wobei 0 für ein schlechtes Ergebnis und 1 für ein perfektes Ergebnis in Bezug auf die vordefinierten Klassen steht.

Datensätze. Die Evaluierung findet auf Basis geeigneter Datensätze aus dem UCI Machine Learning Repository [DKT17] statt. Das Repository verfügt über 440 Datensätze (Stand 07.09.2018). Zur Identifikation geeigneter Datensätze findet zunächst die nachfolgende Selektion statt: Default Task=Classification ∧ Attribute Type=Categorical ∧ Data Type=Multivariate. Von diesen Datensätzen werden im nächsten Schritt alle Datensätze verworfen, die weniger

Tabelle 5.3.: Übersicht Evaluierungsdatensätze. Die letzte Spalte beinhaltet die durchschnittliche Korrelation zwischen jedem Paar von Attributen in einem Datensatz und wird mithilfe der Funktion $cor(X|Y)$ (siehe Gleichung 5.6) berechnet. Die Datensätze sind in drei Subgruppen von korreliert bis unkorreliert unterteilt.

Datensatz	**Datenpunkte**	**Attribute**	**Klassen**	**Durchschn. Korrelation** $cor(X\|Y)$
Teaching Assistant Evaluation	151	5	3	0,336
Soybean Large	307	35	19	0,263
Breast Cancer Winconsin Original	699	10	2	0,216
Mushroom-Extended	8416	22	2	0,162
Mushroom	8124	22	2	0,161
Dermatology	366	34	6	0,098
Lymphography	148	18	4	0,070
Soybean Small	47	35	19	0,070
Breast Cancer	286	9	2	0,054
Audiology-Standard	226	69	24	0,044
Hayes-Roth	160	4	3	0,045
Post-Operative Patient	90	8	3	0,031
Tic-Tac-Toe Endgame	958	9	2	0,012
Monks Problem 1	432	6	2	0,000
Balance-Scale	625	4	3	0,000
Car Evaluation	1728	6	4	0,000
Nursey	12960	8	5	0,000

als 25 Datenpunkte besitzen (zum Beispiel Balloons) oder primär aus binären Attributen bestehen (zum Beispiel Chess). Zur Ermittlung weiterer Datensätze erfolgt folgende Selektion im UCI Machine Learning Repository: Default Task=Classification $\wedge$ Attribute Type=Mixed $\wedge$ Data Type=Multivariate. Von diesen Datensätzen werden jene selektiert, die hauptsächlich aus kategorischen Attributen bestehen und einige Ganzahl-Attribute (Integer-Attribute) mit wenigen Ausprägungen beinhalten. Konkret handelt es sich um die Datensätze Teaching Assistant Evaluation, Breast Cancer Winconsin Original, Dermatology und Post-Operative Patient. Idee hinter dieser Selektion ist, die Integer Attribute als kategorische Attribute zu interpretieren, da viele Baseline Abstandsmaße die Präsenz numerischer Attribute nicht explizit berücksichtigen.

Die selektierten Datensätze sind in Tabelle 5.3 aufgeführt. Zur besseren Analyse und Diskussion der Ergebnisse findet eine Unterteilung der Datensätze in drei Untergruppen korreliert, geringfügig-korreliert und unkorreliert statt. Da es für diese Untergruppen keine fix definierten Grenzwerte gibt, werden im Rahmen dieser Arbeit die folgenden Werte verwendet: korreliert (durchschnittliche Korrelation $cor(X|Y) \geq 0{,}05$), geringfügig-korreliert (durchschnittliche Korrelation $0{,}05 > cor(X|Y) > 0$) und unkorreliert (durchschnittliche Korrelation $cor(X|Y) \approx 0$).

5.5.2. Experiment 1 – Auswirkung unterschiedlicher Schwellwerte

Experiment 1 analysiert die Auswirkungen unterschiedlicher benutzerdefinierter Schwellwerte θ (siehe Abschnitt 5.3.4) und der automatischen Schwellwertberechnung $\theta_{X|Y}$ (siehe Abschnitt 5.4) im Klassifikationskontext.

Tabelle 5.4.: Ergebnisse von Experiment 1. Diese Tabelle zeigt die Klassifikationsgenauigkeiten vom kNN-Klassifikator in Prozent. Als Abstandsmaß wurde ConDist mit verschiedenen benutzerdefinierten Schwellwerten verwendet.

Datensatz	$\theta_{X\|Y}$	$\theta_{0,00}$	$\theta_{0,01}$	$\theta_{0,02}$	$\theta_{0,03}$	$\theta_{0,05}$	$\theta_{0,10}$	$\theta_{0,20}$	$\theta_{0,50}$	$\theta_{1,00}$
Teaching Assistant Evaluation	48,02 ±1,63	47,94 ±1,68	47,95 ±1,68	47,94 ±1,68	47,95 ±1,68	**48,08** ±1,72	47,84 ±1,70	47,83 ±1,69	47,19 ±1,86	45,84 ±3,08
Soybean Large	91,54 ±0,35	91,60 ±0,36	91,61 ±0,36	91,61 ±0,35	91,60 ±0,35	**92,00** ±0,41	89,30 ±0,38	89,00 ±0,39	89,24 ±0,48	91,30 ±0,39
Breast Cancer Winconsin Original	96,04 ±0,19	96,04 ±0,19	96,04 ±0,19	96,04 ±0,19	96,04 ±0,19	96,04 ±0,19	96,05 ±0,18	**96,22** ±0,19	95,25 ±0,29	95,25 ±0,29
Dermatology	96,58 ±0,47	**96,63** ±0,45	**96,63** ±0,44	96,59 ±0,47	96,59 ±0,45	96,59 ±0,48	96,19 ±0,57	96,09 ±0,50	96,45 ±0,67	95,90 ±0,68
Lymphography	82,99 ±1,49	**83,30** ±1,54	**83,30** ±1,56	83,26 ±1,53	83,13 ±1,45	82,89 ±1,46	82,02 ±1,55	82,05 ±1,25	81,25 ±1,43	81,26 ±1,44
Breast Cancer	73,58 ±1	73,62 ±0,90	73,57 ±0,92	73,53 ±0,99	73,51 ±0,97	73,78 ±0,98	74,11 ±1,02	73,99 ±1,21	**74,35** ±1,21	74,34 ±1,23
Audiology Standard	66,01 ±1,48	66,14 ±1,46	66,10 ±1,49	66,09 ±1,43	66,39 ±1,42	**66,46** ±1,49	65,03 ±1,59	61,55 ±1,68	61,51 ±2,00	61,54 ±2,08
Hayes-Roth	68,79 ±2,02	68,34 ±2,20	68,49 ±2,11	68,63 ±2,19	68,76 ±2,11	**69,66** ±2,36	64,47 ±3,22	64,47 ±3,22	64,47 ±3,22	64,47 ±3,22
Post-Operative Patient	69,19 ±1,39	**69,64** ±1,20	69,61 ±1,25	69,52 ±1,3	69,50 ±1,27	69,61 ±1,27	68,32 ±2,32	68,58 ±2,33	68,59 ±2,31	68,59 ±2,31
TicTacToe	**99,99** ±0,03	**99,99** ±0,03	99,87 ±0,11	99,93 ±0,08	99,92 ±0,10	94,74 ±0,96	94,74 ±0,96	94,74 ±0,96	94,74 ±0,96	94,74 ±0,96
Car	**90,56** ±0,53	88,89 ±1,39	**90,56** ±0,53	**90,56** ±0,53	**90,56** ±0,53	**90,56** ±0,53	**90,56** ±0,53	**90,56** ±0,53	**90,56** ±0,53	**90,56** ±0,53
Nursey	**94,94** ±0,18	94,44 ±0,77	**94,94** ±0,18	**94,94** ±0,18	**94,94** ±0,18	**94,94** ±0,18	**94,94** ±0,18	**94,94** ±0,18	**94,94** ±0,18	**94,94** ±0,18
Monks Problem 1	**97,32** ±1,03	94,70 ±2,39	**97,32** ±1,03	**97,32** ±1,03	**97,32** ±1,03	**97,32** ±1,03	**97,32** ±1,03	**97,32** ±1,03	**97,32** ±1,03	**97,32** ±1,03
Balance-Scale	**78,66** ±1,24	77,18 ±1,02	**78,66** ±1,24	**78,66** ±1,24	**78,66** ±1,24	**78,66** ±1,24	**78,66** ±1,24	**78,66** ±1,24	**78,66** ±1,24	**78,66** ±1,24
Durchschnitt	82,44	82,03	82,44	82,47	**82,49**	82,24	81,40	81,14	81,04	81,05

ConDist berücksichtigt ausschließlich korrelierte Kontextattribute in der Abstandsberechnung. Der erforderliche Korrelationsgrad wird mithilfe des Parameters θ gesteuert. Folglich nimmt ConDist ausschließlich Kontextattribute Y in der Menge der korrelierten Kontextattribute $context_X(Y)$ auf, deren Korrelationswert größer oder gleich dem benutzerdefinierten Schwellwert $cor(X|Y) \geq \theta$ beziehungsweise $cor(X|Y) \geq \theta_{X|Y}$ ist. Der Schwellwert θ gibt also den minimalen Wert an, den die Funktion $cor(X|Y)$ zurückgeben muss, damit Y als Kontextattribut in der Menge der korrelierten Kontextattribute $context_X$ vom Zielattribut X aufgenommen wird. Höhere Schwellwerte führen generell zu weniger Kontextattributen. Im Extremfall $\theta = 0$ werden alle Attribute in die Menge $context_X$ aufgenommen. Die automatische Schwellwertberechnung $\theta_{X|Y}$ folgt dem im Absatz 5.4.2 definierten Ansatz.

Die Ergebnisse des Experiments sind in Tabelle 5.4 zu sehen, wobei jede Spalte die durchschnittliche Klassifikationsgenauigkeit einschließlich der Standardabweichung über 100 Durchläufe in Prozent angibt.

Tabelle 5.4 zeigt, dass der benutzerdefinierte Schwellwert $\theta = 0{,}03$ im Durchschnitt die besten Ergebnisse erzielt. Die Methode zur automatischen Schwellwertberechnung $\theta_{X|Y}$ sowie die benutzerdefinierten Schwellwerte $\theta = 0{,}01$ und $\theta = 0{,}02$ erreichen ähnlich gute Ergebnisse.

Für fehlende Schwellwerte ($\theta = 0$) ist eine sinkende Klassifikationsgenauigkeit zu beobachten. Für zu groß gewählte Schwellwerte θ nimmt die Klassifikationsgenauigkeit ebenfalls ab. Die durchschnittliche Klassifikationsgenauigkeit steigt zunächst bei geringen Schwellwerten θ an, erreicht einen Höchstwert bei $\theta = 0{,}03$, nimmt bei mittelhohen Schwellwerten wieder langsam ab, erreicht ein Minimum bei $\theta = 0{,}5$ und steigt mit sehr hohen Schwellwerten wieder langsam an.

Für viele Datensätze bleibt die Klassifikationsgenauigkeit mit steigenden Schwellwerten stabil. Die Ursache hierfür ist, dass ab einem bestimmten Schwellwert keine Kontextattribute außer das Zielattribut selbst identifiziert werden. In diesen Fällen reduziert sich das Abstandsmaß ConDist auf den Hamming-Abstand. Je geringer die Korrelationen der Attribute sind, desto schneller ist dieser Effekt zu sehen. Für unkorrelierte Datensätze ist dieser Effekt bereits ab einem Schwellwert von $\theta = 0{,}01$ zu erkennen. In den nachfolgenden Experimenten wird wie in Ring et al. [ROB+15] der benutzerdefinierte Schwellwert $\theta = 0{,}02$ und die automatische Schwellwertberechnung $\theta_{X|Y}$ aus Ring et al. [RLH15] verwendet.

5.5.3. Experiment 2 – Eignung von ConDist in Klassifikationsszenarien

Experiment 2 untersucht ob sich unterschiedliche Abstandsmaße unterschiedlich gut in Klassifikationsszenarien eignen. Hierfür vergleicht Experiment 2 ConDist mit anderen kategorischen Abstandsmaßen im Kontext Klassifikation und verwendet alle Datensätze aus Tabelle 5.3. Die Ergebnisse dieses Experiments sind in Tabelle 5.5 abgebildet, mit Ausnahme der Datensätze Mushroom-Extended, Mushroom und Soybean Small. Für diese Datensätze erreichen alle Abstandsmaße 100 Prozent Klassifikationsgenauigkeit. Folglich haben die verschiedenen Abstandsmaße keinen Einfluss auf das Ergebnis. Eine Berücksichtigung dieser Datensätze würde zu keinem Informationsgewinn führen und die Unterschiede zwischen den Abstandsmaßen im Mittel reduzieren.

ConDist mit dem benutzerdefinierten Schwellwert $\theta = 0{,}02$ erreicht im Durchschnitt die höchste Klassifikationsgenauigkeit von allen Abstandsmaßen in Kombination mit den kNN Klassifikator. Bei korrelierten und geringfügig korrelierten Datensätzen erzielen kontextbasierte kategorische Abstandsmaße (ConDist, DILCA, JiaCheung und Quang) meist bessere Ergebnisse als andere Abstandsmaße. Im Falle unkorrelierter Daten sind kontextbasierte Abstandsmaße schlechter als ConDist und nicht kontextbasierte Abstandsmaße.

Tabelle 5.5.: Ergebnisse von Experiment 2. Die Zellen enthalten die Klassifikationsgenauigkeiten eines kNN-Klassifikators mit unterschiedlichen kategorischen Abstandsmaßen in Prozent. Die Datensätze sind von oben nach unten in drei Teilmengen korreliert, geringfügig korreliert und unkorreliert unterteilt.

Datensatz	**ConDist**	**ConDist** $\theta_{X\|Y}$	**DILCA**	**Eskin**	**Gambar-yan**	**Ham-ming**	**Jia-Cheung**	**OF**	**Quang**
Teaching Assistant. E.	47,94 ±1,68	48,02 ±1,63	**50,68** ±2,01	48,79 ±2,03	49,44 ±1,94	45,84 ±3,08	49,54 ±2,45	39,16 ±2,13	44,48 ±1,97
Soybean Large	91,61 ±0,35	91,54 ±0,35	91,48 ±0,37	89,83 ±0,41	87,18 ±0,44	91,30 ±0,39	89,54 ±0,41	89,61 ±0,51	**91,89** ±0,41
Breast Cancer Winconsin Original	96,04 ±0,19	96,04 ±0,19	95,55 ±0,29	95,67 ±0,26	92,84 ±0,27	95,25 ±0,29	95,36 ±0,28	72,47 ±0,33	**96,28** ±0,20
Dermatology	96,59 ±0,47	96,58 ±0,47	**97,97** ±0,40	94,92 ±0,76	91,69 ±0,58	95,90 ±0,68	97,55 ±0,35	61,12 ±0,90	96,70 ±0,51
Lymphography	83,26 ±1,53	82,99 ±1,49	82,09 ±1,40	79,17 ±1,42	80,72 ±1,31	81,26 ±1,44	**83,95** ±1,26	72,77 ±1,54	81,33 ±1,48
Breast Cancer	73,53 ±0,99	73,58 ±1,00	73,33 ±0,97	73,03 ±1,23	74,13 ±0,94	**74,33** ±1,23	73,67 ±0,85	68,66 ±1,06	70,41 ±1,14
Audiology Standard	66,09 ±1,43	66,01 ±1,48	62,31 ±1,46	63,45 ±0,99	**66,16** ±1,13	61,27 ±2,14	60,09 ±1,57	51,87 ±1,08	52,02 ±1,68
Hayes-Roth	68,63 ±2,19	68,79 ±2,02	67,59 ±2,77	46,71 ±4,20	60,84 ±2,72	64,47 ±3,22	68,27 ±1,98	46,70 ±3,09	**71,19** ±2,25
Post-Operative Patient	69,52 ±1,30	69,16 ±1,39	68,22 ±1,57	68,41 ±1,93	**69,69** ±1,49	68,59 ±2,31	67,28 ±1,57	69,44 ±1,45	68,69 ±1,48
Tic-Tac-Toe Endgame	99,93 ±0,08	**99,99** ±0,03	90,65 ±0,61	94,74 ±0,96	98,25 ±0,30	94,74 ±0,96	99,93 ±0,09	76,80 ±0,77	99,65 ±0,17
Car Evaluation	**90,56** ±0,53	**90,56** ±0,53	90,25 ±1,23	90,03 ±0,52	90,25 ±0,56	**90,56** ±0,53	90,01 ±0,60	87,83 ±0,65	88,25 ±1,27
Nursey	**94,94** ±0,18	**94,94** ±0,18	92,61 ±0,81	93,29 ±0,14	93,24 ±0,14	**94,94** ±0,18	93,32 ±0,15	94,65 ±0,19	94,72 ±0,60
Monks Problem 1	**97,32** ±1,03	**97,32** ±1,03	92,06 ±2,96	95,60 ±1,13	93,78 ±1,31	**97,32** ±1,02	96,77 ±1,00	96,85 ±1,06	90,50 ±3,19
Balance-Scale	**78,66** ±1,24	**78,66** ±1,24	78,43 ±1,57	**78,66** ±1,24	77,16 ±1,32	**78,66** ±1,24	78,65 ±1,25	78,54 ±1,25	77,51 ±1,58
Durchschnitt	**82,47**	82,44	80,94	79,45	80,38	81,03	81,71	71,89	80,26

Statistische Signifikanzprüfung. Demšar [Dem06] beschäftigt sich mit dem statistischen Vergleich von Klassifikatoren über mehrere Datensätze hinweg und empfiehlt den Wilcoxon Signed-Ranks-Test [Wil45] für den Vergleich von zwei Klassifikatoren und den Friedman-Test [Fri37, Fri40] für den Vergleich von mehreren Klassifikatoren.

Im Folgenden werden die Ergebnisse aus Tabelle 5.4 auf statistisch signifikante Unterschiede mithilfe des Friedman-Tests und des Wilcoxon-Signed-Ranks-Tests geprüft. Zunächst wird der Friedman-Test zum Vergleich aller Abstandsmaße und anschließend der Wilcoxon-Signed-Ranks-Test für eine genauere Analyse durchgeführt. Der Friedman-Test stellt in Experiment 2 die Nullhypothese auf, dass die Mediane der Klassifikationsgenauigkeiten des kNN-Klassifikators mit unterschiedlichen Abstandsmaßen gleich sind. Der Friedman-Test ergibt einen ρ-Wert von 0,00006 und ist somit für $\rho < 0{,}05$ signifikant, sodass die Nullhypothese verworfen werden kann. Der Wilcoxon-Signed-Ranks-Test [Wil45] stellt die Nullhypothese auf, dass die Klassifikationsgenauigkeiten des kNN-Klassifikators mit ConDist (mit benutzerdefiniertem Schwellwert $\theta = 0{,}02$) und des kNN-Klassifikators mit einem anderen Abstandsmaß gleich sind.

Der Wilcoxon-Signed-Ranks-Test mit einen Signifikanzniveau von $\alpha = 0{,}05$ wird für ConDist und den anderen kategorischen Abstandsmaßen durchgeführt. Die Ergebnisse dieses Tests sind in Tabelle 5.6 zu sehen und zeigen einen signifikanten Unterschied von ConDist zu den Abstandsmaßen DILCA, Eskin, Gambaryan, Hamming, OF und Quang auf. ConDist weist jedoch keinen signifikanten Unterschied zu ConDist $\theta_{X|Y}$ und JiaCheung auf (siehe Tabelle 5.6).

Tabelle 5.6.: Ergebnisse des Wilcoxon-Signed-Ranks-Tests für Experiment 2. Der Test wurde zwischen ConDist und den anderen Abstandsmaßen durchgeführt. Der Wilcoxon-Signed-Rank-Test ist bei 14 Datensätzen und einem Signifikanzniveau von $\alpha = 0{,}05$ für $W < 21$ signifikant.

	DILCA	**Eskin**	**Gambaryan**	**Hamming**	**JiaCheung**	**OF**	**Quang**
W-Wert	20	5,5	12,5	13	30,5	0	18,5
Signifikant	ja	ja	ja	ja	nein	ja	ja

5.5.4. Experiment 3 – Eignung von ConDist zur Clusteranalyse

Experiment 3 vergleicht ConDist mit anderen kategorischen Abstandsmaßen im Kontext Clusteranalyse. Hierfür verwendet Experiment 3 alle Datensätze aus Tabelle 5.3. Die Ergebnisse des Experiments sind in Tabelle 5.7 zu sehen.

Für einige Datensätze (Teaching Assistang Evaluation, Lymphography, Breast Cancer, Hayes-Roth, Post-Operative Patient, Tic-Tac-Toe Endgame, Monks Problem 1, Balance-Scale, Nursey und Car Evaluation) kann das Clustering die vordefinierten Klassen nicht rekonstruieren. Bei den anderen Datensätzen ist kein Abstandsmaß konstant besser als alle anderen Abstandsmaße. In Tabelle 5.7 ist jedoch zu sehen, dass ConDist (mit benutzerdefiniertem Schwellwert und automatisch berechnetem Schwellwert) im Mittel konstant gute Ergebnisse erzielt, während andere Abstandsmaße bei einigen Datensätzen wesentlich niedrigere NMI-Werte erzielen. Als Beispiele für diese Beobachtung seien das Abstandsmaß Quang auf dem Mushroom Datensatz, das Abstandsmaß OF auf dem Dermatology Datensatz oder das Abstandsmaß Eskin auf dem Lymphography Datensatz genannt.

Statistische Signifikanzprüfung. Wie in den vorherigen Experimenten wird zunächst der Friedman-Test auf den Ergebnissen von Tabelle 5.7 durchgeführt. In Experiment 3 stellt

Tabelle 5.7.: Ergebnisse von Experiment 3. Vergleich der kategorischen Abstandsmaße im Bereich der Clusteranalyse. Jede Spalte beinhaltet den NMI für das Ergebnis der Clusteranalyse durch WARD mit den jeweiligen genutzten Abstandsmaßen. Die Datensätze sind von oben nach unten in drei Teilmengen bezüglich ihrer durchschnittlichen Korrelation unterteilt.

Datensatz	**ConDist**	**ConDist** $\theta_{X\|Y}$	**DILCA**	**Eskin**	**Gambar-yan**	**Ham-ming**	**Jia-Cheung**	**OF**	**Quang**
Teaching Assistant Evaluation	0,078	0,078	**0,085**	**0,085**	**0,085**	0,044	**0,085**	0,060	0,042
Soybean Large	0,776	0,782	0,785	0,758	0,772	0,793	0,735	**0,805**	0,778
Breast Cancer Winconsin Original	0,777	0,681	0,557	0,749	0,601	0,621	0,656	0,217	**0,798**
Mushroom Extended	**0,597**	**0,597**	**0,597**	0,317	**0,597**	**0,597**	0,223	**0,597**	0,245
Mushroom	**0,594**	**0,594**	**0,594**	0,312	**0,594**	**0,594**	**0,594**	0,312	0,241
Dermatology	0,854	0,860	**0,946**	0,817	0,863	0,847	0,879	0,292	0,859
Lymphography	0,189	0,194	0,303	0,165	0,163	0,226	0,207	0,243	**0,320**
Soybean Small	0,688	0,688	0,690	0,687	0,692	0,689	**0,701**	0,690	0,692
Breast Cancer	0,061	0,002	**0,080**	0,031	0,001	0,027	0,076	0,002	0,002
Audiology-Standard	0,636	0,646	0,584	0,623	0,620	0,568	**0,678**	0,439	0,508
Hayes-Roth	0,067	0,045	0,027	0,004	0,029	0,007	0,006	0,012	**0,166**
Post-Operative Patient	0,011	0,022	0,017	0,018	0,017	0,019	**0,025**	0,032	0,033
Tic-Tac-Toe Endgame	0,0003	0,053	0,004	0,003	0,047	0,032	**0,082**	0,001	0,001
Monks Problem 1	0,000	0,000	0,015	0,000	0,000	0,000	0,000	**0,030**	0,015
Balance-Scale	**0,083**	**0,083**	0,036	0,064	0,064	**0,083**	0,067	0,064	0,036
Car Evaluation	0,062	0,062	0,036	**0,150**	0,031	0,062	0,150	0,062	0,036
Nursey	0,048	0,048	0,006	0,037	0,037	0,048	0,037	**0,098**	0,006
Durchschnitt	**0,325**	0,320	0,315	0,284	0,307	0,309	0,306	0,233	0,298

Tabelle 5.8.: Ergebnisse des Wilcoxon-Signed-Ranks-Tests für Experiment 3. Der Test wurde zwischen ConDist und den anderen Abstandsmaßen durchgeführt. Der Wilcoxon-Signed-Ranks-Test ist bei 17 Datensätzen und einem Signifikanzniveau von $\alpha = 0{,}05$ für $W < 34$ signifikant.

	DILCA	**Eskin**	**Gambaryan**	**Hamming**	**JiaChung**	**OF**	**Quang**
W-Wert	74,5	27,5	43,5	61,5	72,5	53,5	56
Signifikant	nein	ja	nein	nein	nein	nein	nein

der Friedman-Test die Nullhypothese auf, dass die Mediane der NMI-Werte des WARD-Clusterverfahrens mit unterschiedlichen Abstandsmaßen gleich sind. Der Friedman-Test ergibt einen ρ-Wert von 0,503 und ist somit für $\rho < 0{,}05$ nicht signifikant. In diesem Fall kann die Nullhypothese, dass alle Abstandsmaße gleichwertig sind, nicht abgelehnt werden.

Der Wilcoxon-Signed-Ranks-Test [Wil45] stellt die Nullhypothese auf, dass die NMI-Werte des WARD-Clusterverfahrens mit ConDist und des WARD-Clusterverfahrens mit einem anderen Abstandsmaß gleich sind. Anschließend wurde der Wilcoxon-Signed-Ranks-Test (mit Signifikanzniveau $\alpha = 0{,}05$) zwischen ConDist mit benutzerdefinierten Schwellwert $\theta = 0{,}02$ und den anderen Abstandsmaßen durchgeführt. Mit Ausnahme von Eskin zeigen die Ergebnisse des Wilcoxon-Signed-Ranks-Tests keine statistisch signifikanten Unterschiede (siehe Tabelle 5.8).

5.5.5. Experiment 4 - Analyse der berechneten Abstandsmatrix

Experiment 4 führt eine genaue Analyse der berechneten Abstände von ConDist durch. Hierfür werden ein korrelierter Datensatz (Breast Cancer) und ein unkorrelierter Datensatz (Car Evaluation) mit ordinalen Attributen verwendet. Diese ordinalen Attribute werden in ConDist wie kategorische Attribute behandelt, ermöglichen es jedoch, die Ergebnisse besser zu interpretieren.

In Experiment 4 wird ConDist zunächst mit dem benutzerdefinierten Schwellwert $\theta = 0{,}02$ auf dem kompletten Datensatz Breast Cancer berechnet und anschließend das Attribut age genauer betrachtet. Tabelle 5.9 gibt die berechneten Abstände zwischen den Werten des Attributs age (Alter) wieder. Die berechneten Abstände in Tabelle 5.9 deuten darauf hin, dass ConDist die allgemeinen Tendenzen der Attributwerte gut widerspiegelt. So weist ConDist beispielsweise den Werten $d(20-29, 30-39) = 0{,}624$ einen geringen Abstand als den Werten $d(20-29, 50-59) = 0{,}857$ zu. Des Weiteren ist der maximale Abstand 1 zwischen den erwarteten Werten $20-29$ und $70-79$ zu finden.

Tabelle 5.9.: Abstandsmatrix für das Attribut age des Datensatzes Breast Cancer.

age	**20-29**	**30-39**	**40-49**	**50-59**	**60-69**	**70-79**
20-29	0,0	0,624	0,646	0,857	0,996	1,0
30-39	0,624	0,0	0,546	0,747	0,890	0,923
40-49	0,646	0,546	0,0	0,712	0,854	0,891
50-59	0,857	0,747	0,712	0,0	0,640	0,683
60-69	0,996	0,890	0,854	0,640	0,0	0,557
70-79	1,0	0,923	0,891	0,683	0,557	0,0

Anschließend wird ConDist mit dem benutzerdefinierten Schwellwert $\theta = 0{,}02$ auf dem kompletten Datensatz Car Evaluation berechnet und das Attribut safety genauer betrachtet. Die ermittelten Abstände sind in Tabelle 5.10 zu sehen.

Tabelle 5.10.: Abstandsmatrix für das Attribut safety des Datensatzes Car Evaluation.

safety	low	medium	high
low	0,0	1,0	1,0
medium	1,0	0,0	1,0
high	1,0	1,0	0,0

Tabelle 5.10 zeigt, dass ConDist für das Attribut safety des Datensatzes Car Evaluation die Beziehungen nicht widerspiegeln kann. Stattdessen weist ConDist, wie der Hamming-Abstand, identischen Werten den Abstand 0 und nicht identischen Werten den Abstand 1 zu.

5.6. Diskussion

5.6.1. Experiment 1 – Auswirkung unterschiedlicher Schwellwerte

Hohe benutzerdefinierte Schwellwerte θ führen zu niedrigeren Klassifikationsgenauigkeiten für korrelierte Datensätze, als Beispiel sei auf die Schwellwerte $\theta = 0{,}2$, $\theta = 0{,}5$ und $\theta = 1{,}0$ in den Datensätzen Teaching Assistant Evaluation und Lymphography (siehe Tabelle 5.4) verwiesen. Folglich verwerfen zu hoch gesetzte Schwellwerte nützliche Kontextattribute. Eine ähnliche Beobachtung kann für zu niedrig gewählte Schwellwerte bei geringfügig korrelierten Datensätzen gemacht werden. Fast alle Schwellwerte erreichen dieselben Ergebnisse für unkorrelierte Datensätze. Lediglich das Entfernen des Schwellwerts ($\theta = 0$) führt zu schlechteren Ergebnissen. In diesem Fall nimmt ConDist unkorrelierte Kontextattribute in die Menge $context_X$ mit auf, die fehlerhafte Informationen in die Abstandsberechnung einbringen können.

Die automatische Schwellwertberechnung $\theta_{X|Y}$ erreicht für korrelierte und unkorrelierte Datensätze gute Ergebnisse. Insgesamt betrachtet erreicht $\theta_{X|Y}$ ähnlich gute Klassifikationsgenauigkeiten wie gut gewählte Schwellwerte $\theta = 0{,}02$ oder $\theta = 0{,}03$.

Die Ergebnisse aus Experiment 1 deuten darauf hin, dass die automatische Schwellwertberechnung $\theta_{X|Y}$ im Vergleich zu benutzerdefinierten Schwellwerten überlegen ist, falls diese schlecht gewählt sind und vergleichbar ist, falls die benutzerdefinierten Schwellwerte gut gewählt sind. Folglich ist die automatische Schwellwertberechnung $\theta_{X|Y}$ dem ursprünglichen Ansatz von [ROB+15] vorzuziehen, da der benutzerdefinierte Parameter θ bei gleichbleibender Qualität eliminiert wird.

5.6.2. Experiment 2 – Eignung von ConDist in Klassifikationsszenarien

In Experiment 2 erreichen kategorische Abstandsmaße, welche Kontextattribute in der Abstandsberechnung berücksichtigen, eindeutig bessere Ergebnisse für korrelierte Datensätze. Unter diesen Abstandsmaßen gibt es jedoch keinen eindeutigen Sieger für korrelierte Datensätze.

Für unkorrelierte Datensätze erreichen ältere kontextbasierte Abstandsmaße (DILCA, Quang und JiaCheung) schlechtere Ergebnisse als Abstandsmaße, die keine Kontextattribute berücksichtigen. DILCA und Quang verwenden auch in Datensätzen ohne Korrelationen ausschließlich Kontextattribute zur Berechnung von Abständen im Zielattribut, was zu zufälligen Abständen führen kann. Im Vergleich dazu erreicht ConDist auch gute Ergebnisse für unkorrelierte Datensätze. Dies lässt sich darauf zurückführen, dass ConDist auch das Zielattribut selbst in der Abstandsberechnung berücksichtigt. Dieser Effekt wird im Vergleich der Abstandsmaße ConDist und Hamming-Abstand deutlich. Falls keine Korrelationen vorhanden sind, verwendet

ConDist ausschließlich das Zielattribut selbst als Kontextattribut und reduziert sich in diesem Fall auf den Hamming-Abstand. Deshalb führen diese beiden Abstandsmaße zu gleichen Ergebnissen in unkorrelierten Datensätzen (Monks Problem 1, Balance-Scale, Nursey und Car Evaluation) (siehe Tabelle 5.5). Bei korrelierten und geringfügig korrelierten Datensätzen wird die zusätzliche Berücksichtigung von Kontextattributen seitens ConDist zum Vorteil gegenüber dem Hamming-Abstand und führt zu besseren Ergebnissen. Der Wilcoxon-Signed-Ranks-Test bestätigt die Überlegenheit von ConDist gegenüber dem Hamming-Abstand (siehe Tabelle 5.6).

5.6.3. Experiment 3 – Eignung von ConDist zur Clusteranalyse

Die meisten kategorischen Abstandsmaße erreichen in der experimentellen Evaluierung im Kontext Clusteranalyse ähnliche Ergebnisse (siehe Tabelle 5.7). Dies liegt vermutlich daran, dass der Algorithmus zur Clusteranalyse selbst und seine Fähigkeit, die vordefinierten Klassen zu rekonstruieren, einen höheren Einfluss auf die Ergebnisse hat als die verwendeten Abstandsmaße. Die Ergebnisse zeigen jedoch, dass einzelne Abstandsmaße für individuelle Datensätze regelmäßig große Leistungseinbußen im Vergleich zu anderen Abstandsmaßen aufzeigen. Beispielsweise erreicht das Abstandsmaß von JiaCheung oftmals sehr gute Ergebnisse, jedoch auf den Mushroom Datensatz nur ein sehr schlechtes Ergebnis. Ähnliche Beobachtungen können für die Abstandsmaße DILCA, Eskin, OF und Quang auf den Datensätzen Breast Cancer Winconsin Original, Mushroom, Mushroom Extended, Dermatology und Audiology-Standard gemacht werden. Einzig das vorgestellte Abstandsmaß ConDist erreicht für jeden Datensatz stets gute Ergebnisse und stellt somit das stabilste Abstandsmaß auf den verschiedenen Datensätzen dar. Dies ist auch durch den NMI Wert ersichtlich, der für ConDist in Experiment 3 im Durchschnitt am höchsten ist.

Die Nullhypothese, dass alle Abstandsmaße gleichwertig sind, kann durch den Friedman-Test in Experiment 3 nicht widerlegt werden. Der Wilcoxon-Signed-Ranks-Test zeigt auch keine statistisch signifikanten Unterschiede in der Leistung von ConDist zu den anderen Abstandsmaßen (mit Ausnahme von Eskin). Jedoch führen die Ergebnisse von Experiment 3 zu der Annahme, dass ConDist als robusteres Abstandsmaß im Kontext Clusteranalyse angesehen werden kann.

5.6.4. Experiment 4 - Analyse der errechneten Abstandsmatrix

In Experiment 4 wurden die berechneten Abstandsmatrizen von ConDist genauer analysiert. Hierfür wurde ein korrelierter und ein nicht korrelierter Datensatz ausgewählt. Für beide Datensätze wurde exemplarisch ein ordinales Attribut ausgewählt, welches in ConDist als kategorisches Attribut behandelt wurde und somit Rückschlüsse auf die berechneten Abstände erlaubt.

Für das Attribut age des korrelierten Datensatzes spiegeln sich die wesentlichen Tendenzen der Attributwerte wider. So nehmen beispielsweise mit zunehmenden Altersintervallen die Abstände zum Wert $20-29$ zu. Jedoch ist auch ersichtlich, dass die berechneten Abstände zwar die Tendenzen korrekt abbilden, aber nicht die Abstände zwischen mehreren Werten. Die Altersintervalle $20-29$ und $30-39$ weisen in Realität einen viel kleineren Abstand auf als die Intervalle $20-29$ und $50-59$. Dies spiegelt sich jedoch im Vergleich der Abstände $d(20-29, 30-39)$ und $d(20-29, 50-59)$ nicht so wider.

Für das Attribut safety des unkorrelierten Datensatzes Car Evaluation konnte ConDist die wesentlichen Strukturen nicht widerspiegeln. Dies liegt daran, dass ConDist keine weiteren korrelierten Kontextattribute (außer das Zielattribut selbst) identifizieren konnte. Folglich reduziert ConDist sich in diesem Fall auf den Hamming-Abstand und vergleicht die Werte lediglich auf Gleichheit beziehungsweise Ungleichheit. Für dieses Beispiel ist positiv anzumerken, dass

ConDist unkorrelierte Attribute ignoriert und somit keine fehlerhaften Informationen in die Abstandsberechnung einfließen lässt.

5.7. Zusammenfassung

Kategorische Abstandsmaße sind eine notwendige Anforderung für die erfolgreiche Anwendung von Data Mining auf Daten mit kategorischen Attributen. Mit ConDist wurde ein allgemeines und robustes Abstandsmaß für Datensätze vorgestellt, die aus kontinuierlichen und kategorischen Attributen bestehen.

ConDist ist ein unüberwachtes Abstandsmaß, welches automatisiert verfügbare Informationen aus Korrelationen zwischen Attributen extrahiert und zur Abstandsberechnung zwischen kategorischen Werten verwendet. ConDist basiert auf der Annahme, dass kategorische Werte ähnlich sind, wenn diese mit ähnlichen Wahrscheinlichkeitsverteilungen in korrelierten Kontextattributen einhergehen. Durch diesen Ansatz versucht ConDist die fehlende inhärente Ordnungsrelation kategorischer Attribute zu kompensieren, indem interne Zusammenhänge aus dem Datensatz ausgewertet werden. Die experimentelle Evaluierung deutet darauf hin, dass sich ConDist als Abstandsmaß für kategorische Attribute eignet. Liegen Datensätze mit Korrelationen zwischen Attributen vor, ist die Leistung von ConDist vergleichbar mit bekannten kontextbasierten kategorischen Abstandsmaßen und nicht kontextbasierten Abstandsmaßen überlegen. Im Falle von schwach und nicht korrelierten Datensätzen ist ConDist vergleichbar mit nicht kontextbasierten kategorischen Abstandsmaßen und bekannten kontextbasierten kategorischen Abstandsmaßen überlegen. Der Wilcoxon-Signed-Ranks-Test bestätigt die Gesamtverbesserung von ConDist im Evaluationsszenerio Klassifikation. Im Evaluationsszenerio Clusteranalyse konnte diese Verbesserung nicht statistisch bestätigt werden.

Eine Erweiterung von ConDist erlaubt die automatische Berechnung des Schwellwerts θ, so dass sich dessen manuelle Festlegung erübrigt. Der Schwellwert θ steuert die Auswahl geeigneter Kontextattribute und dient dazu, unkorrelierte Kontextattribute in der Abstandsberechnung zu verwerfen. Kernidee dieser Erweiterung ist, für jedes Paar von Zielattribut X und Kontextattribut Y einen individuellen Schwellwert aus der Datenmenge zu berechnen und den global benutzerdefinierten Schwellwert θ zu eliminieren. Die automatisiert berechneten Schwellwerte $\theta_{X|Y}$ hängen von der Anzahl Datenpunkte sowie der Entropie der Attribute X und Y ab. Daher können diese individuellen Schwellwerte besser auf die spezifischen Korrelationsanforderungen von Ziel- und Kontextattribut angepasst werden. Folglich eliminiert diese Erweiterung den einzigen benutzerdefinierten Parameter von ConDist und erleichtert somit dessen Anwendung. Die experimentelle Evaluierung dieser Erweiterung zeigt, dass die automatisch berechneten Schwellwerte $\theta_{X|Y}$ vergleichbare Ergebnisse wie gut gewählte benutzerdefinierte Schwellwerte θ und bessere Ergebnisse als schlecht gewählte benutzerdefinierte Schwellwerte θ liefern.

6. Lernen von Ähnlichkeiten zwischen IP Adressen

Im vorherigen Kapitel wurde mit ConDist ein allgemeines Abstandsmaß für Daten mit heterogenen Attributen vorgestellt. ConDist berechnet für jedes kategorische Attribut eine Abstandsmatrix. Dies hat zur Folge, dass ConDist lediglich paarweise Abstände zwischen kategorischen Werten, jedoch keine Mittelwerte über mehrere Werte berechnen kann. Kategorische Attribute von flowbasierten Netzwerkdaten umfassen große Wertebereiche. Deshalb sind die Berechnung und der Speicherbedarf einer Abstandsmatrix nicht zu unterschätzen. Im Gegensatz dazu können Transformationen kategorischer Attribute in kontinuierliche Attribute die oben genannten Nachteile eliminieren. Dieses Kapitel ist speziell auf die in flowbasierten Netzwerkdaten vorhandenen Attribute fokussiert und stellt die Methode IP2Vec zur Transformation von IP-Adressen in kontinuierliche Vektorrepräsentationen vor. Diese erlauben die Anwendung von Standardmetriken, die Berechnung von Mittelwerten und die Visualisierung von IP-Adressen. Dieses Kapitel adressiert somit die Herausforderung der Beschaffenheit flowbasierter Netzwerkdaten, indem es kategorischen Attribute in kontinuierliche Vektoren überführt und zur Datenvorverarbeitung im Bereich IT-Sicherheit genutzt werden kann. Die nachfolgenden Inhalte basieren auf der Publikation *„IP2Vec: Learning Similarities between IP Addresses“* [RLD+17].

6.1. Einleitung

Die Forschung an anomaliebasierten Methoden zur Detektion neuer und unbekannter Angriffe im Bereich IT-Sicherheit ist seit vielen Jahren ein aktives Forschungsfeld [GDM+09]. Kernidee anomaliebasierter Methoden (siehe Abschnitt 4.1.2) ist die Modellierung von Normalverhalten. Abweichungen vom erlernten Normalverhalten stellen Anomalien dar und können auf verdächtiges Verhalten hinweisen. Für die Erkennung von Anomalien ist ein geeignetes Ähnlichkeitsmaß unerlässlich [BCK08]. Dieses Kapitel konzentriert sich auf die Berechnung von Ähnlichkeiten zwischen IP-Adressen.

Problem. Die Berechnung von Ähnlichkeiten zwischen IP-Adressen ist nicht trivial, da IP-Adressen kategorische Attribute sind und somit keine Standardmaße wie der Minkowski-Abstand (siehe Abschnitt 2.4.1) oder die Cosinus-Ähnlichkeit (siehe Abschnitt 2.4.4) verwendet werden können. Das im vorherigen Kapitel vorgestellte Abstandsmaß ConDist könnte zur Berechnung von Abständen zwischen IP-Adressen genutzt werden. ConDist berechnet eine paarweise Abstandsmatrix pro Attribut, was im Falle von IP-Adressen, die bis zu 2^{32} Ausprägungen besitzen können, einen sehr großen Speicherbedarf und enorme Rechenzeiten mit sich bringen würde. Deshalb werden in diesem Kapitel weder ConDist noch andere korrelationsbasierte kategorische Abstandsmaße berücksichtigt. IP-Adressen stellen omnipräsente Attribute im Bereich IT-Sicherheit dar und treten in paket- und flowbasierten Netzwerkdaten [TW11] sowie in diversen hostbasierten Logdateien [LZO10] auf. Viele Data Mining-Methoden können nicht direkt auf flowbasierten Netzwerkdaten angewendet werden, da diese kategorische Attribute wie IP-Adressen und Ports, aber auch numerische Attribute wie Bytes und Pakete beinhalten (siehe

Abschnitt 4.2.2). Gleichzeitig beinhalten IP-Adressen wichtige Informationen, da sich Hosts über ihre IP-Adressen identifizieren lassen und somit einen eindeutigen Fußabdruck eines Hosts beziehungsweise eines Benutzers darstellen. Im weiteren Verlauf dieses Kapitels wird der Begriff Host implizit mit dem kompletten Netzwerkverkehr einer Quell-IP-Adresse gleichgesetzt.

Ziel. Primäres Ziel dieses Kapitels ist die Transformation von IP-Adressen in kontinuierliche Vektorrepräsentationen, welche Informationen über das Netzwerkverhalten der IP-Adressen beinhalten. Für die Transformation soll die Methode verfügbare Kontextinformationen aus zugrundeliegenden flowbasierten Netzwerkdaten nutzen. Die erstellten Vektorrepräsentationen berücksichtigen somit Ähnlichkeiten der IP-Adressen bezüglich der verursachten Netzwerkverbindungen. Folglich können die erlernten Vektorrepräsentationen als Eingabedaten für nachfolgende Analyseverfahren oder Visualisierungsverfahren dienen.

Ansatz und Beiträge Die Repräsentation nicht kontinuierlicher Attribute ist ein bekanntes Problem im Bereich Text Mining und es existieren verschiedene Lösungsansätze für diese Problemstellung [WIZ15]. Einer dieser Lösungsansätze ist das von Mikolov et al. [MSC+13] entwickelte Word2Vec. Word2Vec (siehe Abschnitt 3.3) ist ein Algorithmus, der einen Textkorpus als Eingabe verwendet und darauf aufbauend kontinuierliche Vektorrepräsentationen für die darin enthaltenen Wörter erlernt. Im Text Mining wird Word2Vec bereits seit Jahren erfolgreich eingesetzt [BDK14, CM14].

Da die zugrundeliegende Ausgangslage von Word2Vec einige Parallelen zu flowbasierten Netzwerkdaten aufweist, wird in diesem Kapitel der von Word2Vec verwendete Ansatz auf flowbasierte Netzwerkdaten übertragen. Die resultierende Methode wird als IP2Vec bezeichnet. Der aufgezeichnete Netzwerkverkehr kann als Textkorpus angesehen werden, wobei kategorische Attribute (IP-Adressen, Ports und Transport-Protokoll) das Vokabular darstellen. Kernidee ist das Training eines speziellen neuronalen Netzes. Dieses Netz verfügt über eine versteckte Schicht, die viel weniger Neuronen als die Ein- und Ausgabeschicht besitzt. Die Anzahl der Neuronen in der Ein- und Ausgabeschicht sind identisch und entsprechen der Größe des Vokabulars. Nachdem das neuronale Netz trainiert wurde, wird es nicht für die Aufgabe verwendet, für die es trainiert wurde. Stattdessen nutzt IP2Vec die Gewichte der versteckten Schicht als Vektorrepräsentationen für die Einträge des Vokabulars. Folglich stehen für alle Einträge des Vokabulars (auch Ports und Transport-Protokolle) kontinuierliche Vektorrepräsentationen zur Verfügung. Die vorgestellte Methode IP2Vec wird experimentell zur Clusteranalyse und Visualisierung von IP-Adressen evaluiert. In der experimentellen Evaluierung erlernt IP2Vec Vektorrepräsentationen für zwei flowbasierte Datensätze. Diese werden anschließend mit DBScan (siehe Abschnitt 3.2.2) bezüglich ihrer Ähnlichkeit gruppiert und mit t-SNE (siehe Abschnitt 3.5.1) visualisiert.

Hauptbeitrag dieses Kapitels ist IP2Vec, eine unüberwachte Methode zur Transformation von IP-Adressen in kontinuierliche Vektorrepräsentationen, welche Informationen über das Netzwerkverhalten der IP-Adressen widerspiegeln und zur Berechnung von Ähnlichkeiten genutzt werden können. Hierfür baut IP2Vec auf der Kernidee von Word2Vec auf und überträgt diese auf flowbasierte Netzwerkdaten.

6.2. Verwandte Arbeiten

Dieser Abschnitt analysiert den Umgang mit IP-Adressen in netzwerkbasierten Methoden zur Detektion von Angriffen. Einen generellen Überblick von Ähnlichkeitsmaßen im Bereich netzwerkbasierter Angriffserkennung liefern Weller-Fahy et al. [WBS15]. Weller-Fahy et al. diskutieren verschiedene Ähnlichkeitsmaße, deren Verbreitung und theoretischen Hintergrund. Der folgende Überblick konzentriert sich lediglich auf die Ähnlichkeitsberechnung von IP-Adressen und

berücksichtigt keine weiteren Attribute flowbasierter Netzwerkdaten. Die Ansätze werden in drei Gruppen unterteilt, wobei (I) IP-Adressen ignorieren, (II) abgeleitete Attribute aus IP-Adressen extrahieren und (III) IP-Adressen auf kontinuierliche Werte transformieren oder Metriken auf IP-Adressen definieren.

Tran et al. [TJH12] stellen ein Echtzeit IDS vor, welches in Kategorie (I) fällt. Die Autoren verwenden ein neuronales Netz und integrieren dieses auf einem FPGA. Das neuronale Netz berücksichtigt vier Attribute (Pakete, Bytes, Dauer und TCP-Flags) von flowbasierten Netzwerkdaten. IP-Adressen finden hierbei keine Berücksichtigung. Najafabadi et al. [NKN+16] nutzen Klassifikatoren zur Detektion von RUDY-Angriffe in flowbasierten Netzwerkdaten. RUDY-Angriffe sind DoS-Angriffe auf Anwendungsebene und verursachen somit viel weniger Netzwerkverkehr als klassische DoS-Angriffe. Zur Detektion dieses Angriffstyps extrahieren Najafabadi et al. [NKN+16] diverse Attribute von flowbasierten Netzwerkdaten, jedoch keine IP-Adressen. Einen ähnlichen Ansatz nutzen Najafabadi et al. [NKC+15] zur Analyse von SSH-Brute-Force Angriffen. Die Autoren evaluieren verschiedene Klassifikatoren auf unterschiedlichen Subsets von Attributen. IP-Adressen sind jedoch in keinem der generierten Subsets zu finden. Stattdessen versuchen die Autoren eine Anpassung an bestimmte Netzwerkstrukturen durch den Ausschluss von IP-Adressen gezielt zu vermeiden. DISCLOSURE [BBR+12] ist eine Methode zur Detektion von Botnet Command and Control Servern auf flowbasierten Netzwerkdaten. Die Autoren verwenden IP-Adressen und Ports, um zwischen Client und Server zu unterscheiden, aber bei der anschließenden Berechnung von Ähnlichkeiten finden IP-Adressen keine Berücksichtigung. Weitere Arbeiten, die auf IP-Adressen verzichten, sind beispielsweise [WHZ11], [IAB17] oder [SP14].

Kategorie (II) beinhaltet netzwerkbasierte Ansätze, welche neuartige Attribute aus IP-Adressen extrahieren. Ein typischer Ansatz ist, aus IPv4-Adressen 32 binäre Attribute zu extrahieren [VM14]. Eine anderer oft verwendeter Ansatz ist, Netzwerkverkehr über Zeitfenster zu aggregieren und basierend auf diesen Zusammenfassungen neue Attribute zu berechnen. Einen derartigen Ansatz präsentiert Garcia et al. [GGS+14] zur Detektion von Botnets. BClus [GGS+14] unterteilt den flowbasierten Netzwerkverkehr in Zeitfenster und fasst alle Flows pro Quell-IP-Adresse pro Zeitfenster zusammen. Basierend auf diesen Zusammenfassungen werden neue Attribute, wie beispielsweise die Anzahl von eindeutig adressierten Ziel-IP-Adressen, erstellt und zur weiteren Verarbeitung genutzt. Ein ähnlicher Ansatz wurde von Mathur et al. [MCB13] vorgestellt. Mathur et al. fassen alle Flows von der gleichen Quell-IP-Adresse innerhalb eines Zeitfensters zusammen und berechnen dann weiterführende Attribute. Die Autoren interpretieren IP-Adressen als 32 Bit Integer und berechnen deren arithmetisches Mittel oder die Entropie von der Verteilung von Ziel-IP-Adressen. Li et al. [LGB+13] verwenden Methoden des maschinellen Lernens zur Klassifikation von Hosts auf flowbasierten Netzwerkdaten. Hierbei aggregieren die Autoren ebenfalls Flows über Zeitfenster und extrahieren Attribute wie die Summe der ersten Bytes von Ziel-IP-Adressen.

Ansätze der Kategorie (III) definieren Metriken für IP-Adressen. Wang et al. [WP17] definieren eine einfache Metrik, indem die Autoren IPv4-Adressen als 32 binäre Attribute interpretieren und darauf den Minkowski-Abstand berechnen. Ein weiterer beliebter Ansatz ist die Transformation von IP-Adressen in geographische Koordinaten. Dieser Ansatz wurde beispielsweise von Hu et al. [HHP12] oder Jiang et al. [JLM16] verwendet. Vorteil dieser Transformation ist, dass geographische Koordinaten reelle Zahlen sind und somit Standardmaße zur Berechnung von Ähnlichkeiten verwendet werden können. Nachteil dieser Vorgehensweise ist jedoch, dass die transformierten Werte keine Informationen über das Verhalten der IP-Adressen beinhalten. Ein ausgeklügelterer Ansatz wurde von Coull et al. [CMB11] entwickelt. Die Autoren nutzen

Domänenwissen, um eine Hierarchie für IP-Adressen zu erstellen und leiten aus dieser Hierarchie Abstände für IP-Adressen ab. Der Abstand zwischen zwei IP-Adressen ergibt sich durch die Ebene der Hierarchie, auf der sich die IP-Adressen unterscheiden. Coull et al. verwenden hierfür die Einteilung in *Unicast*, *Multicast*, *Broadcast*, *Public* oder *Privat* IP-Adresse, um ihre Hierarchie aufzubauen. Eine solche Metrik berücksichtigt jedoch nur die Netzwerkstruktur und nicht das tatsächliche Netzwerkverhalten einer IP-Adresse. Jakalan et al. [JGS+16] entwickeln eine Methode, welche IP-Adressen in intern und extern bezüglich des beobachteten Netzwerkverkehrs unterteilt. Anschließend berechnen die Autoren lediglich Ähnlichkeiten für die internen IP-Adressen. Grundlegende Idee von [JGS+16] ist, dass interne IP-Adressen ähnlich zueinander sind, wenn diese mit den gleichen externen IP-Adressen kommunizieren. Kommunizieren die internen IP-Adressen mit unterschiedlichen externen IP-Adressen, so sind diese als unähnlich zu werten.

Zwei weitere Ansätze, die in dieser Übersicht nicht unerwähnt bleiben sollen, sind die Arbeiten von Henry [Hen16] und Mimura und Tanaka [MT18b]. Henry [Hen16] adaptiert Word2Vec, um Ähnlichkeiten zwischen Flows zu berechnen. IP2Vec ist im Grundsatz ähnlich, adaptiert Word2Vec jedoch, um Ähnlichkeiten zwischen IP-Adressen basierend auf flowbasierten Netzwerkdaten zu erlernen. Dies ermöglicht die Berechnung von Ähnlichkeiten zwischen IP-Adressen, Ports und Transport-Protokollen. Flow2Vec [Hen16] hingegen arbeitet auf kompletten Flows und führt somit zu einem sehr großen Vokabular, da eine Abbildung aller möglichen Kombinationen aus IP-Adressen, Ports, Bytes, Pakete und so weiter zu eindeutigen Werten stattfindet. Mimura und Tanaka [MT18b] adaptieren Doc2Vec [LM14] auf paketbasierte Netzwerkdaten. Die Autoren ignorieren zwar IP-Adressen, verwenden jedoch die Attribute Transport-Protokoll, Ports und Größe der Pakete. Dabei fassen die Autoren immer 100 Pakete als einen Paragraphen zusammen und lernen für jeden Paragraphen eine kontinuierliche Vektorrepräsentation, welche für spätere Klassifikationsaufgaben verwendet wird.

IP2Vec gliedert sich in Kategorie (III) ein, da es IP-Adressen in kontinuierliche Vektorrepräsentationen transformiert, was die anschließende Verwendung von Standardmaßen wie der Cosinus-Ähnlichkeit (siehe Abschnitt 2.4.4) ermöglicht. Während Coull et al. [CMB11] zusätzliches Domänenwissen zur Definition von Abständen verwenden, erlernt IP2Vec Ähnlichkeiten basierend auf den beobachteten Netzwerkverbindungen automatisiert und unüberwacht. Im Vergleich zu Jakalan et al. [JGS+16], definiert IP2Vec Ähnlichkeiten zwischen allen IP-Adressen unter Berücksichtigung von adressierten IP-Adressen, Ports und Transport-Protokollen.

sectionIP2Vec

In diesem Abschnitt wird die Methode IP2Vec detailliert vorgestellt. Zunächst wird die grundlegende Idee und das Modell von IP2Vec beschrieben (siehe Abschnitt 6.2.1). Da IP2Vec auf einem neuronalen Netz basiert, werden eine Trainingsaufgabe und Trainingsbeispiele benötigt, deren Definitionen in Abschnitt 6.2.2 gegeben sind. Abschnitt 6.2.3 beschreibt, wie die Vektorrepräsentationen für IP-Adressen aus dem erlernten neuronalen Netz abgeleitet werden. Abschnitt 6.2.4 diskutiert Unterschiede zwischen IP2Vec und Word2Vec.

6.2.1. Definition von IP2Vec

Ziel von IP2Vec ist es, die kategorischen Attribute eines Flows (IP-Adressen, Ports und Transport-Protokoll) in kontinuierliche Vektorrepräsentationen zu transformieren, sodass sich die ursprünglichen Ähnlichkeiten zwischen den Werten in den Vektoren reflektieren. Hierfür folgt IP2Vec der Kernidee von Word2Vec (siehe Abschnitt 3.3) und adaptiert diese auf flowbasierte Netzwerkdaten.

IP2Vec basiert auf einem neuronalen Netz mit einer versteckten Schicht, welche viel weniger Neuronen als die Ein- und Ausgabeschicht besitzt. Die Ausgabeschicht besitzt eine zusätzliche Softmax-Funktion (siehe Abschnitt 3.1.4), welche die Werte aller Ausgabeneuronen in Wahrscheinlichkeiten transformiert. Die Architektur ist in Abbildung 6.1 graphisch dargestellt.

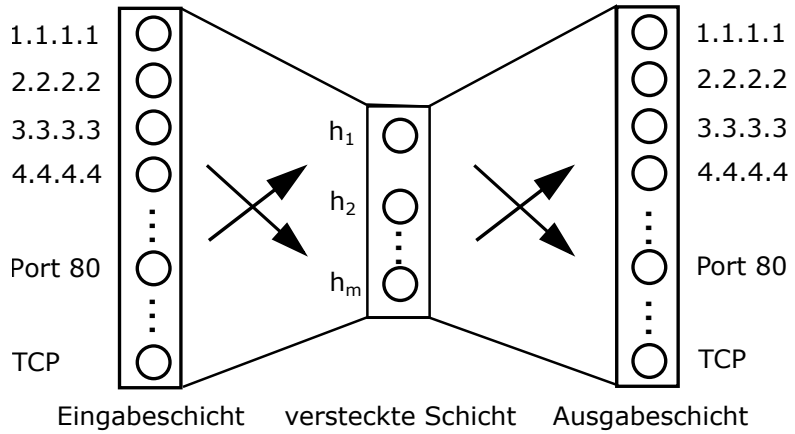


Abbildung 6.1.: Architektur von IP2Vec.

Wie in Abbildung 6.1 zu sehen, stellen die Attribute flowbasierter Netzwerkdaten die Eingangsdaten von IP2Vec dar. Somit ist ein Flow in Analogie zu einem Satz in Word2Vec zu interpretieren. In dieser Analogie stellen die Attribute eines Flows die Wörter des Satzes dar. Da kategorische Attribute wie IP-Adressen, Ports und Transport-Protokoll nicht direkt in neuronale Netze gegeben werden können, muss zunächst eine Konvertierung in One-Hot-Vektoren erfolgen. Die Länge des One-Hot-Vektors entspricht dabei der Größe des Eingabevokabulars $|V|$. Angenommen, das Vokabular V beinhaltet 100.000 IP-Adressen, 20.000 Ports und 4 Transport-Protokolle, dann hat jeder One-Hot-Vektor 120.004 Dimensionen und für jeden Attributwert beziehungsweise Wort ist eine Dimension 1 und alle anderen Dimensionen des Vektors 0. Jeder Wert des Vokabulars wird einem Neuron in der Eingabe- und Ausgabeschicht zugewiesen (siehe Abbildung 6.1). Folglich hängt das Vokabular von den zugrundeliegenden flowbasierten Netzwerkdaten und den selektierenden Attributen ab. IP2Vec verwendet die Quell-IP-Adresse, den Ziel-Port und das Transport-Protokoll als Eingabewörter (siehe nächster Abschnitt). Ähnlich wie bei Word2Vec werden die Gewichte der versteckten Schicht als kontinuierliche Vektorrepräsentationen der Eingabewörter nach dem Training verwendet (siehe Abschnitt 6.2.3).

6.2.2. Training

IP2Vec zielt darauf ab, IP-Adressen (und auch Ports und Transport-Protokolle) in Vektoren zu transformieren, sodass IP-Adressen mit ähnlichen Netzwerkverhalten ähnliche Vektorrepräsentationen erhalten. Dafür verwendet IP2Vec aufgezeichneten Netzwerkverkehr in flowbasierter Form. Jeder Flow beschreibt eine Netzwerkverbindung und die Attribute eines Flows stehen miteinander in Verbindung.

Die Trainingsaufgabe für das neuronale Netz wird so definiert, dass IP2Vec die Kontextattribute eines Flows vorhersagt, wenn der Wert eines speziellen Attributs desselben Flows an der Eingabeschicht angelegt wird. Somit kommt der Auswahl der Kontextattribute eine zentrale Bedeutung zu. Die strukturierte Form flowbasierter Netzwerkdaten erlaubt eine gezielte Selektion. IP2Vec fokussiert hierbei nur Attribute, welche den Typ und das Ziel von Netzwerkverbin-

dungen beschreiben. Als Folge dessen werden die Attribute Quell-IP-Adresse, Ziel-IP-Adresse, Ziel-Port und Transport-Protokoll aus flowbasierten Netzwerkdaten verwendet. Ein Beispiel mit fünf Flows mit dieser ausgewählten Attributmenge ist in Tabelle 6.1 zu sehen.

Tabelle 6.1.: Darstellung von 5 Flows mit ausgewählten Attributen.

#	Quell-IP-Adresse	Ziel-IP-Adresse	Ziel-Port	Transport-Protokoll
1	192.168.100.5	192.168.220.9	51479	TCP
2	192.168.220.9	192.168.100.5	445	TCP
3	216.58.210.19	192.168.200.8	44444	TCP
4	4.4.4.4	2.2.2.2	80	TCP
5	192.168.220.14	53.53.53.53	53	UDP

Basierend auf den in Tabelle 6.1 angegebenen Attributen verwendet IP2Vec die Quell-IP-Adresse, den Ziel-Port und das Transport-Protokoll als Eingabewörter. Für jedes Eingabewort selektiert IP2Vec eine individuell angepasste Menge von Kontextwörtern. Diese Selektion berücksichtigt Domänenwissen und die entsprechende Zielstellung. Die Generierung von Trainingsbeispielen ist in Tabelle 6.2 generell und in Tabelle 6.3 mithilfe des Beispiels dargestellt. Eingabewörter sind durch türkisen Hintergrund und Kontextwörter durch grauen Hintergrund hervorgehoben. Auf der rechten Seite sind die generierten Trainingsbeispiele für die entsprechenden Kombinationen aus Eingabewort und Kontextwort zu sehen. In Tabelle 6.3 ist ein Beispiel für diese Generierung anhand von Flow #4 aus Tabelle 6.1 abgebildet. Pro Flow erstellt IP2Vec fünf Trainingsbeispiele. In der Trainingsphase wird das Eingabewort am neuronalen Netz angelegt und dieses versucht die Wahrscheinlichkeit der Kontextwörter vorherzusagen. Für die Trainingsbeispiele (siehe Tabelle 6.2 und 6.3) ist die Wahrscheinlichkeit für das konkrete Kontextwort 1 und für alle anderen Wörter des Vokabulars 0.

Tabelle 6.2.: Vorgehensweise zur Generierung von Trainingsbeispielen in IP2Vec.

Quell-IP-Adr.	Ziel-IP-Adr.	Ziel-Port	Trans. Proto.		Eingabewort	Kontextwort
Quell-IP-Adr.	Ziel-IP-Adr.	Ziel Port	Trans. Proto.	→	Quell-IP-Adr.	Ziel-IP-Adr.
					Quell-IP-Adr.	Ziel-Port
					Quell-IP-Adr.	Trans. Proto.
Quell-IP-Adr.	Ziel-IP-Adr.	Ziel-Port	Trans. Proto.	→	Ziel-Port	Ziel-IP-Adr.
Quell-IP-Adr.	Ziel-IP-Adr.	Ziel-Port	Trans. Proto.	→	Trans. Proto.	Ziel-IP-Adr.

IP2Vec wählt für das Eingabewort Quell-IP-Adresse die Ziel-IP-Adresse, den Ziel-Port und das Transport Protokoll als Kontextwörter aus. Der Grund für diese Auswahl ist, dass die Ähnlichkeit von IP-Adressen basierend auf den von ihnen erstellten Netzwerkverbindungen berechnet werden soll. Für das Eingabewort Ziel-Port verwendet IP2Vec lediglich die Ziel-IP-Adresse als Kontextwort. Hier berücksichtigt IP2Vec die Tatsache, dass ein Ziel-Port in Korrelation zu einer Ziel-IP-Adresse steht und ein Quell-Port in Korrelation zu einer Quell-IP-Adresse. Somit erzeugt die Trainingsphase Beispiele, die das neuronale Netz anregen, Ports, die von gleichen IP-Adressen verwendet werden, höhere Ähnlichkeiten zuzuweisen. Dieser Umstand führt dazu, dass typische Userports eine höhere Ähnlichkeit untereinander aufweisen als zu typischen Serverports wie 80 oder 443.

Tabelle 6.3.: Beispiel zur Generierung von Trainingsbeispielen in IP2Vec. Diese Abbildung zeigt die beispielhafte Generierung von Trainingsbeispielen in IP2Vec für Flow #4 aus Tabelle 6.1.

Quell-IP-Adr.	Ziel-IP-Adr.	Ziel-Port	Trans. Proto.		Eingabewort	Kontextwort
4.4.4.4	2.2.2.2	80	TCP	→	4.4.4.4	2.2.2.2
					4.4.4.4	80
					4.4.4.4	TCP
4.4.4.4	2.2.2.2	80	TCP	→	80	2.2.2.2
4.4.4.4	2.2.2.2	80	TCP	→	TCP	2.2.2.2

Anmerkung 1. IP2Vec verwendet wie Word2Vec Negative Sampling zur Reduzierung der Trainingszeit. Negative Sampling führt dazu, dass nur eine kleine Menge anstelle aller Gewichte des neuronalen Netzes pro Trainingsbeispiel aktualisiert werden. Für genauere Informationen zu Negative Sampling sei auf Abschnitt 3.3.3 und auf Mikolov et al. [MSC+13] verwiesen.

Anmerkung 2. IP2Vec verwendet nicht die Ziel-IP-Adresse als Eingabewort, da die zugrundeliegenden Daten unidirektionale Verbindungen darstellen. Antworten von Ziel-IP-Adressen auf Anfragen stehen in weiteren unidirektionalen Flows zur Verfügung, in denen die Rollen von Ziel-IP-Adresse und Quell-IP-Adresse vertauscht sind (siehe Flow #1 und #2 in Tabelle 6.1). Aus diesem Grund würde eine Berücksichtigung der Ziel-IP-Adresse als Eingabewort zu Duplikaten bei der Generierung von Trainingsbeispielen führen. Einzige Einschränkung bei dieser Vorgehensweise ist, dass für Broadcast IP-Adressen, welche ausschließlich als Ziel-IP-Adresse auftreten, keine sinnvollen Vektorrepräsentationen erlernt werden. Würden die flowbasierten Netzwerkdaten jedoch im bidirektionalen Format vorliegen, sind weitere Trainingsbeispiele mit der Ziel-IP-Adresse und den Quell-Port als Eingabewörter notwendig.

Anmerkung 3. Bei der Generierung von Trainingsbeispielen fällt auf, dass alle selektierten Attribute für Eingabe- und Kontextwörter auch in paketbasierten Netzwerkdaten auftreten. IP-Adressen und Ports befinden sich im Paket-Header (siehe Abschnitt 4.2.1) und das Transport-Protokoll würde sich aus dem jeweiligen Packet-Header ableiten lassen. Folglich könnte der vorgestellte Ansatz IP2Vec direkt auf paketbasierte Netzwerkdaten übertragen werden.

6.2.3. Ermittlung der Vektorrepräsentationen

IP2Vec verwendet das neuronale Netz nicht für die Aufgabe, für die es trainiert wurde. Stattdessen nutzt IP2Vec die Gewichte der versteckten Schicht als Vektorrepräsentationen für IP-Adressen, Ports und Transport-Protokolle. Dies hat zur Folge, dass nicht nur Ähnlichkeiten zwischen IP-Adressen berechnet werden können, sondern auch Ähnlichkeiten zwischen Ports oder Transport-Protokollen. Theoretisch ermöglicht dies auch die Berechnung von Ähnlichkeiten zwischen einer IP-Adresse und einem Port, aber die Sinnhaftigkeit dieser Berechnung wäre zweifelhaft.

Abbildung 6.2 veranschaulicht die Extraktion von Vektoren aus dem trainierten neuronalen Netz. Zur besseren Übersicht sind in Abbildung 6.2 nur die Gewichte eines Eingabeneurons und eines Ausgabeneurons eingezeichnet. Die Gewichte $\boldsymbol{w_4} = (w_{4,1}, w_{4,2}, ..., w_{4,m})^T$ stellen die Vektorrepräsentation für die IP-Adresse 4.4.4.4 dar.

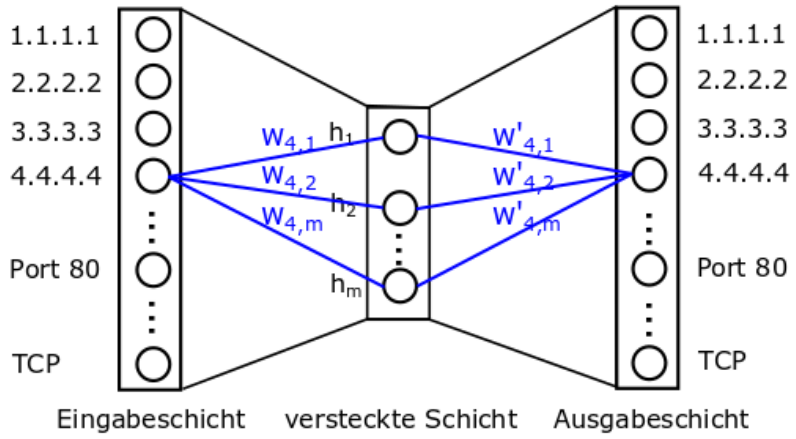


Abbildung 6.2.: Extraktion von Vektorrepräsentationen. Die resultierende Vektordarstellung ergibt sich aus den blau eingezeichneten Gewichten $w_{4,1}$, $w_{4,2}$ bis $w_{4,m}$.

Anschauliche Überprüfung. Abbildung 6.3 veranschaulicht mithilfe eines Beispiels, warum das Training zu sinnvollen Ergebnissen führt. Angenommen, die IP-Adressen 1.1.1.1 und 4.4.4.4 sind ähnlich und bauen ähnliche Netzwerkverbindungen auf. Der One-Hot-Vektor für die IP-Adresse 1.1.1.1 sei $\boldsymbol{e_1}$ und der Ergebnisvektor am neuronalen Netz sei $\boldsymbol{a_1}$. Der One-Hot-Vektor für die IP-Adresse 4.4.4.4 sei $\boldsymbol{e_4}$ und der Ergebnisvektor am neuronalen Netz sei $\boldsymbol{a_4}$. Folglich muss das neuronale Netz für beide IP-Adressen ähnliche Ergebnisse vorhersagen. Dies soll durch die farbig eingezeichneten Ergebnisvektoren $\boldsymbol{a_1}$ und $\boldsymbol{a_4}$ in Abbildung 6.3 verdeutlicht werden. Auf der linken Seite sind die One-Hot-Vektoren $\boldsymbol{e_1}$ und $\boldsymbol{e_4}$ zu sehen, die als Eingabe in das neuronale Netz fließen und auf der rechten Seite sind die generierten Ergebnisse dargestellt. Die Gewichte der versteckten Schicht für die IP-Adressen 1.1.1.1 seien $\boldsymbol{w_1} = (w_{1,1}, w_{1,2}, ..., w_{1,m})^T$ und $\boldsymbol{w_4} = (w_{4,1}, w_{4,2}, ..., w_{4,m})^T$ für 4.4.4.4. Eine Möglichkeit für das neuronale Netz, ähnliche Ergebnisse für unterschiedliche Eingabedaten zu generieren, ist die Zuweisung ähnlicher Gewichte (siehe blaue und rote Linien in Abbildung 6.3) in der versteckten Schicht, d. h. $\boldsymbol{w_1} - \boldsymbol{w_4} \to 0$. Und genau diese Gewichte entsprechen den Vektorrepräsentationen der IP-Adressen 1.1.1.1 und 4.4.4.4.

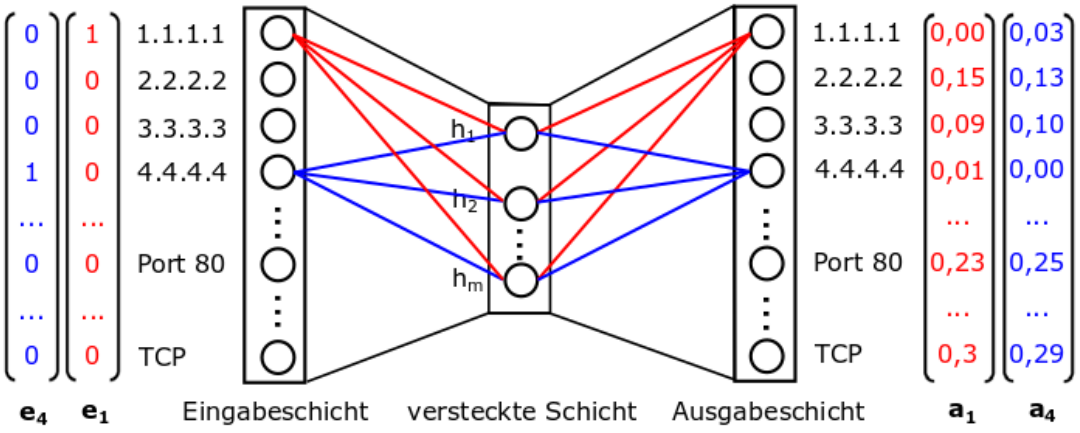


Abbildung 6.3.: Vergleich von zwei IP-Adressen mit ähnlichen Netzwerkverhalten. Der Eingabevektor, die entsprechenden Gewichte und die generierten Ergebnisse für die IP-Adresse 1.1.1.1 sind *rot* und für die IP-Adresse 4.4.4.4 *blau* eingezeichnet.

6.2.4. Unterschiede zu Word2Vec

IP2Vec und Word2Vec (siehe Abschnitt 3.3) unterscheiden sich in einigen Punkten. Einen zentralen Unterschied bilden die zugrundeliegenden Daten. Word2Vec verwendet einem Textkorpus als Eingabe, während IP2Vec auf unidirektionale flowbasierte Netzwerkdaten zurückgreift. In diesem Zusammenhang können Flows zwar als Analogie zu Sätzen verstanden werden, besitzen jedoch aufgrund der vorgegebenen Attribute eine feste Struktur. Dies ermöglicht IP2Vec unter Einbeziehung von Domänenwissen eine gezieltere Erstellung von Trainingsbeispielen durch die Selektion von ausgewählten Eingabewörtern mit individuell angepassten Kontextwörtern.

Ein weiterer zentraler Unterschied ist die Veränderung der inhaltlichen Bedeutung von Wörtern über Zeiträume. Kim et al. [KCH+14] und Hamilton et al. [HLJ16] zeigen zwar, dass sich die Bedeutung einzelner Wörter über die Zeit verändert, doch dieser Vorgang ist viel langsamer als bei flowbasierten Netzwerkdaten. Bei flowbasierten Netzwerkdaten können sich die Netzwerkverbindungen eines Hosts sehr schnell durch bestimmte Ereignisse verändern. Beispielsweise würde die Infektion eines Hosts mit einem Virus oder das Anbieten eines neuen Services auf einem Host, die vom Host aufgebauten Netzwerkverbindungen stark beeinflussen. Folglich können sich Ähnlichkeiten zwischen IP-Adressen bei IP2Vec viel schneller ändern als Wortähnlichkeiten bei Word2Vec.

6.3. Experimente

Dieses Kapitel führt eine experimentelle Evaluierung von IP2Vec durch. Diese Evaluierung findet in den Anwendungsgebieten der Clusteranalyse und Visualisierung statt.

6.3.1. Evaluierungsmethodik

In diesem Abschnitt findet eine experimentelle Evaluierung von IP2Vec statt. Ziel von IP2Vec ist es, IP-Adressen mit ähnlichen Verhalten auf ähnliche Vektorrepräsentationen abzubilden. Um dies zu evaluieren, werden die Vektorrepräsentationen mittels Clusteranalyse und Visualisierung analysiert. Die Evaluierung ist erfolgreich, wenn IP-Adressen mit ähnlichen Verhalten den gleichen Clustern zugeordnet werden beziehungsweise in der Visualisierung nahe beieinander liegen. Folgerichtig sollten IP-Adressen mit unterschiedlichen Verhalten in unterschiedlichen Clustern liegen und größere Abstände in der Visualisierung aufweisen. Im Folgenden wird der Begriff Host als Synonym zu IP-Adresse verwendet, da ein Host durch seine IP-Adresse bestimmt werden kann.

6.3.1.1. Baseline GRAPH

Abschnitt 6.2 ordnet IP2Vec der Kategorie (III) zu, welche Metriken auf IP-Adressen definieren. Da es sich bei den relevanten IP-Adressen der Datensätze um interne IP-Adressen handelt, ist eine Transformation in geographische Koordinaten nicht sinnvoll. Deshalb wird das von Coull et al. [CMB11] entwickelte Abstandsmaß GRAPH (siehe Abschnitt 2.4.3) als Baseline verwendet. GRAPH normiert den Abstand zweier IP-Adressen auf das Intervall $[0, 128]$. Die von GRAPH berechneten Abstände werden durch folgende Transformation in Ähnlichkeiten umgewandelt:

$$sim_{GRAPH} = 1.0 - \frac{d_{GRAPH}(x, y)}{128}, \tag{6.1}$$

wobei d_{GRAPH} der berechnete Abstand von GRAPH ist.

6.3.1.2. Evaluierungsdatensätze

Die zwei flowbasierten Datensätze CTU-13 [GGS+14] und CIDDS-001 [RWG+17c] (siehe Abschnitt 8.3) stellen die Evaluierungsdatensätze dar. Der CTU-13 Datensatz enthält sowohl normalen, als auch verdächtigen Netzwerkverkehr in Form von Botnetzen. Insgesamt beinhaltet dieser Datensatz 13 verschiedene Szenarien. Experiment 1 verwendet Szenario 9, da dieses, wie Szenario 10, mit 10 infizierten Hosts die größte Anzahl von Botnetz infizierten Hosts beinhaltet. Jede IP-Adresse wird als ein Host identifiziert und in diesem Szenario gibt es mehr als 300.000 verschiedene IP-Adressen. In Experiment 1 wird IP2Vec und GRAPH auf allen IP-Adressen trainiert. Anschließend werden jedoch nur die IP-Adressen vom aufgenommen Subnetz (147.34.X.X) für den Visualisierungsprozess und für die Clusteranalyse verwendet. Diese IP-Adressen sind in zwei Klassen, *infiziert* und *normal*, unterteilt.

Der zweite Evaluierungsdatensatz ist CIDDS-001 [RWG+17c]. Dieser Datensatz ist in Abschnitt 8.3 ausführlich beschrieben. Experiment 2 berücksichtigt lediglich den innerhalb der OpenStack-Umgebung aufgezeichneten Netzwerkverkehr von Woche 3. Dieser Netzwerkverkehr beschreibt die Verbindungen von 7 internen Servern, 19 internen Clients und enthält keine Angriffe. Zielsetzung des Experiments ist die Unterscheidung der IP-Adressen in die zwei Klassen *Server* und *Client*. Wie für den CTU-13 Datensatz verwenden IP2Vec und GRAPH alle IP-Adressen in der Trainingsphase. Im anschließenden Visualisierungsprozess und Clusterverfahren werden jedoch zur besseren Übersicht erneut nur die IP-Adressen (192.168.X.X) aus dem aufgezeichneten Subnetz berücksichtigt.

6.3.1.3. Visualisierung

Zur Visualisierung wird die Methode t-SNE [MH08] verwendet. t-SNE führt eine nichtlineare Dimensionsreduzierung durch und achtet darauf, dass Datenpunkte, die im hochdimensionalen Raum geringe Abstände besitzen, mit geringen Abstände im niedrig-dimensionalen Raum einhergehen. Diese Visualisierungsmethode wird häufig zur Visualisierung von Embeddings verwendet (zum Beispiel in [LBH15, MLD15, LCH+16]). Genauere Informationen zu diesem Verfahren sind in Abschnitt 3.5.1 gegeben.

6.3.1.4. Clusteranalyse

Im Evaluationsszenario Clusteranalyse werden die erlernten Vektorrepräsentationen für IP-Adressen mithilfe eines Clusterverfahrens in Gruppen unterteilt. Bei der Wahl eines geeigneten Clusterverfahrens sind einige Aspekte zu berücksichtigen. Das Verhalten normaler Hosts kann untereinander sehr unterschiedlich sein, wenn beispielsweise die Arbeitsweise eines Systemadministrators mit einem Angestellten im Personalbereich verglichen wird. Gleiche Beobachtungen können gemacht werden, wenn Server unterschiedliche Dienste anbieten. Von IP2Vec wird erwartet, dass es auch derartige Unterschiede erlernen kann. Deshalb sollte ein Clusterverfahren ausgewählt werden, welches nicht zwingend alle IP-Adressen in eine vordefinierte Anzahl von Cluster einteilt. Folglich fällt der klassische k-Means Algorithmus [Mac67] nicht in die engere Auswahl. Stattdessen soll ein Algorithmus ausgewählt werden, welcher lediglich IP-Adressen mit hoher Ähnlichkeit demselben Cluster zuweist und Ausreißer zulässt. Folglich eignen sich dichtebasierte Clusterverfahren wie DBScan [EKS+96] oder hierarchische Clusterverfahren wie QROCK [DMP05].

Aufgrund der Eignung und weiten Verbreitung von DBScan [EKS+96], wird dieser Algorithmus verwendet. Eine Beschreibung von DBScan ist in Abschnitt 3.2.2 gegeben. Der Algorithmus

verfügt über zwei benutzerdefinierte Parameter $minPts$ und ϵ. Der Parameter $minPts$ wird für alle Experimente auf 3 gesetzt und der Parameter ϵ wird für jede Clusteranalyse experimentell optimiert. Da es im CTU-13 Datensatz lediglich 10 mit Botnetz infizierte Hosts und im CIDDS-001 Datensatz lediglich 7 Server gibt, würden größere Werte für den Parameter $minPts$ wenig Sinn ergeben. Die Evaluierung der Clusteranalyse findet mithilfe von Zuweisungsmatrizen und der Maße Homogenität und Vollständigkeit (siehe Abschnitt 2.5) statt. Die beiden Evaluierungsmaße sind auf das Intervall $[0, 1]$ normiert, wobei höhere Werte bessere Ergebnisse indizieren.

6.3.1.5. Konfiguration von IP2Vec

IPv4-Adressen besitzen eine Länge von 32 Bit. Das Abstandsmaß GRAPH berechnet den Abstand zwischen zwei IP-Adressen aus dem gleichen Subnetz mithilfe des Hamming-Abstands (siehe Abschnitt 2.4.2) auf der 32-Bit Repräsentation von IPv4-Adressen. In diesem Fall verarbeitet GRAPH IP-Adressen als Objekte mit 32 binären Attributen. Um die Vergleichbarkeit von GRAPH und IP2Vec zu gewährleisten, wird IP2Vec in allen Experimenten mit 32 Neuronen in der versteckten Schicht konfiguriert. Eine Vorstudie zeigte, dass die Anzahl von Neuronen in der versteckten Schicht keinen wesentlichen Einfluss auf die Ergebnisse ausübt, solange die Anzahl der Neuronen nicht viel kleiner oder viel größer gesetzt wird.

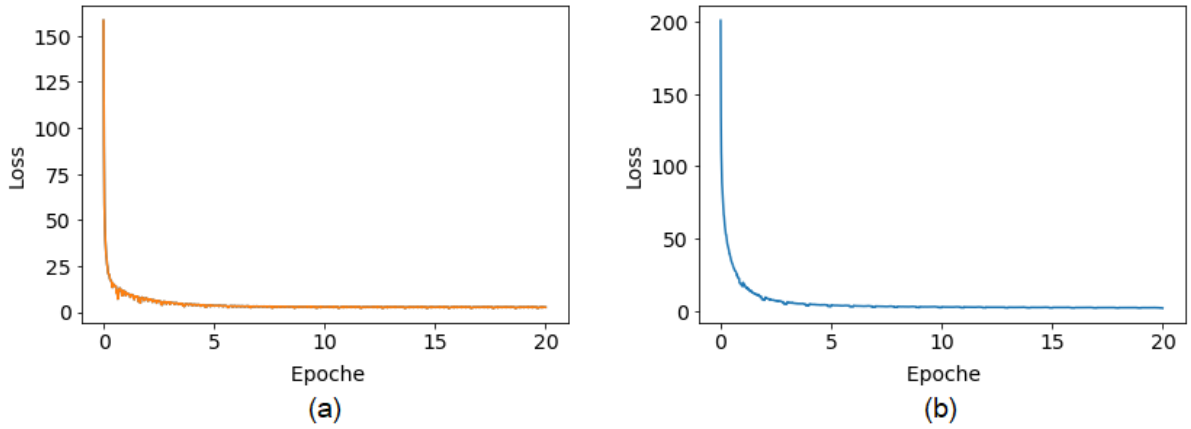


Abbildung 6.4.: Verlauf der Loss-Funktionen. Die linke Abbildung (a) zeigt den Verlauf der Loss-Funktion von IP2Vec für Woche 3 des CIDDS-001 Datensatzes. Die rechte Abbildung (b) zeigt den Verlauf der Loss-Funktion von IP2Vec für Szenario 9 des CTU-13 Datensatzes.

In Abbildung 6.4 ist der Verlauf der Loss-Funktion von IP2Vec für die Datensätze CIDDS-001 und CTU-13 zu sehen. Der Abbildung kann entnommen werden, dass die Werte der Loss-Funktion ab Epoche 7 relativ stabil bleiben. Deshalb wird IP2Vec in den folgenden Experimenten stets für 10 Epochen trainiert.

6.3.2. Experiment 1 - Identifizierung von Botnets

Experiment 1 lernt die Abstandsmaße IP2Vec und GRAPH auf Szenario 9 des CTU-13 Datensatzes an. Nach der Trainingsphase werden die IP-Adressen aus dem Subnetz 147.34.X.X mit t-SNE visualisiert und mit DBScan einer Clusteranalyse unterzogen.

Visualisierung. Abbildung 6.5 zeigt die t-SNE Visualisierung für IP2Vec. In der Visualisierung können die infizierten Hosts (blaue Kreuze) von normalen Hosts (rote Kreise) visuell

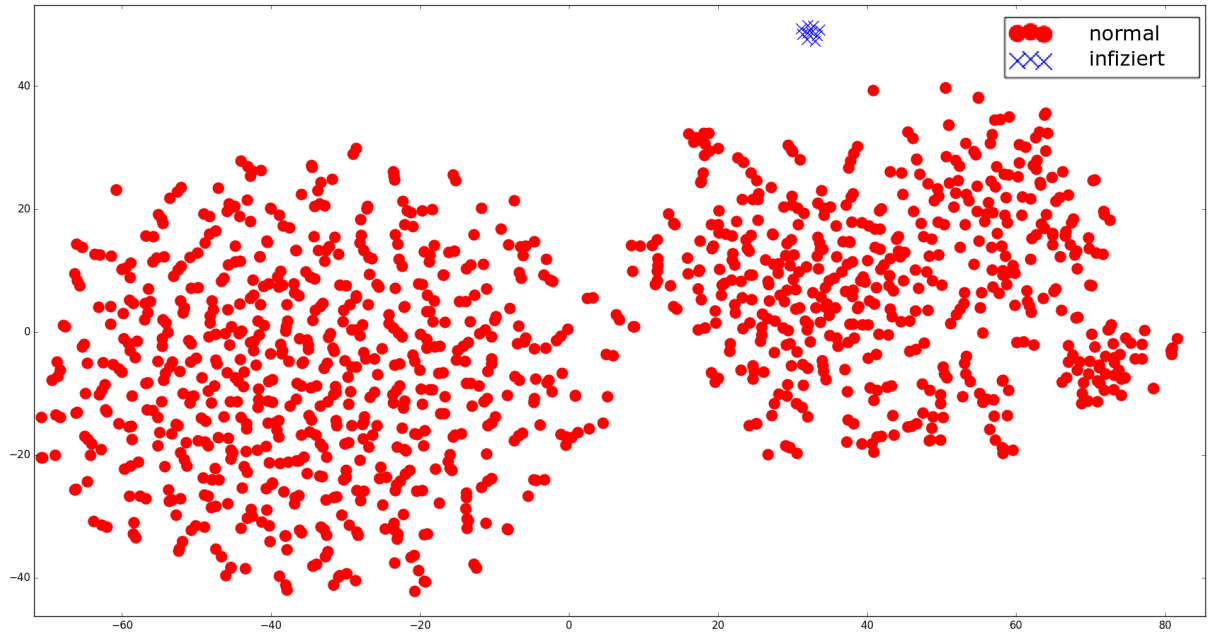


Abbildung 6.5.: t-SNE Visualisierung der mit IP2Vec erlernten Vektorrepräsentationen für IP-Adressen aus Szenario 9 des CTU-13 Datensatzes nach [RLD+17].

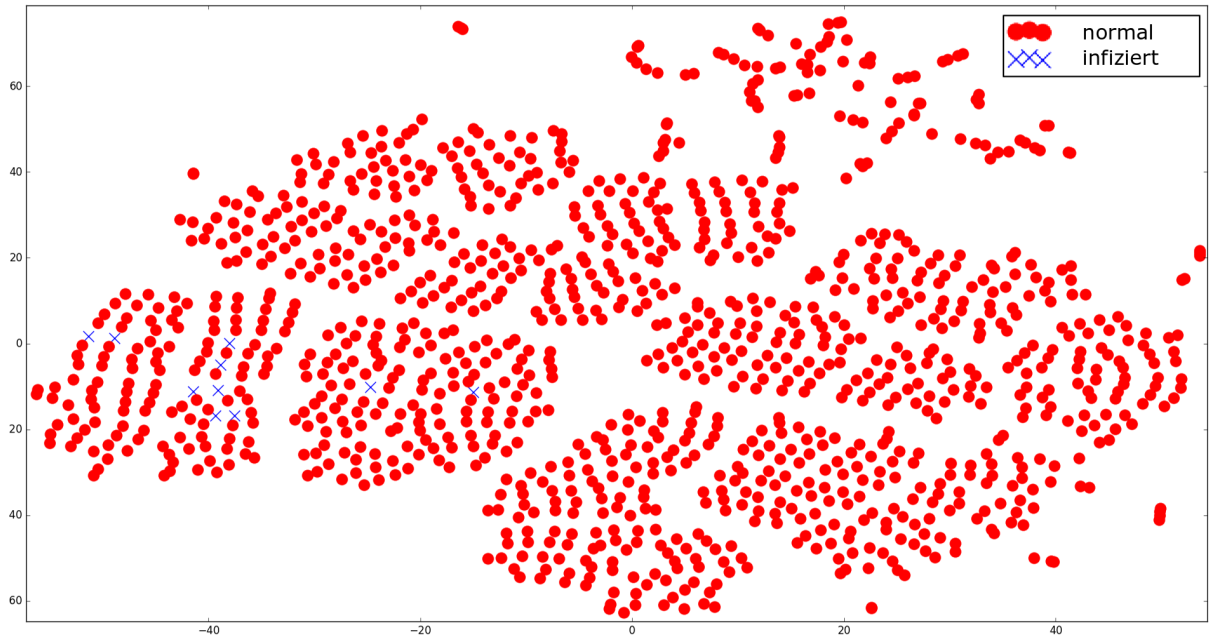


Abbildung 6.6.: t-SNE Visualisierung der mit GRAPH erlernten Abstände zwischen IP-Adressen aus Szenario 9 des CTU-13 Datensatzes nach [RLD+17].

getrennt werden. Dabei weisen die infizierten Hosts sehr hohe Ähnlichkeiten zueinander auf und bilden eine homogene Gruppe in der Visualisierung. Des Weiteren ist zu erkennen, dass das Verhalten normaler Hosts sehr unterschiedlich ist und nicht infizierte Hosts größere Abstände zueinander aufweisen.

In Abbildung 6.6 sind die gleichen Daten basierend auf dem Abstandsmaß GRAPH visualisiert. In dieser Visualisierung ist es nicht möglich, die infizierten Hosts von den normalen Hosts visuell zu trennen. Die nicht infizierten Hosts verteilen sich relativ gleichmäßig in der Visualisierung. Infizierte Hosts weisen größere Abstände zueinander auf und der nächstgelegene Nachbar eines infizierten Hosts ist in der Regel ein normaler Host.

Clusteranalyse. Anschließend werden beide Ähnlichkeitsmaße genutzt, um IP-Adressen bezüglich ihres Verhaltens in zwei Gruppen (*infiziert* und *normal*) zu clustern. Die Zuweisungsmatrix für IP2Vec ist in Tabelle 6.4 und für GRAPH in Tabelle 6.5 abgebildet. Tabelle 6.4 zeigt, dass DBScan in der Lage ist, normale von infizierten IP-Adressen zu separieren, wenn IP2Vec als Ähnlichkeitsmaß verwendet wird. Das Clusterverfahren hat in Kombination mit IP2Vec 3 Cluster und 156 Ausreißer generiert. Dabei konnten alle mit Botnetz infizierten Hosts dem Cluster 2 zugeordnet werden. Die nicht infizierten Hosts teilen sich auf die Cluster 1 und 3 sowie den Ausreißern auf. Die Verwendung von GRAPH als Ähnlichkeitsmaß führt zu keinen eindeutigen (homogenen) Clustern. In diesem Fall hat DBScan ebenfalls drei Cluster und ähnlich viele Ausreißer erzeugt. Bei diesem Clustering sind jedoch alle 10 infizierten Hosts mit den Großteil der nicht infizierten Hosts in Cluster 1 vermischt.

Tabelle 6.4.: Zuweisungsmatrix für den CTU-13 Datensatz (Szenario 9) unter Verwendung von IP2Vec als Ähnlichkeitsmaß.

Klasse	Cluster 1	Cluster 2	Cluster 3	Anzahl Ausreißer
normal	1015	0	3	156
infiziert	0	10	0	0

Tabelle 6.5.: Zuweisungsmatrix für den CTU-13 Datensatz (Szenario 9) unter Verwendung von GRAPH als Ähnlichkeitsmaß.

Klasse	Cluster 1	Cluster 2	Cluster 3	Anzahl Ausreißer
normal	1017	3	3	151
infiziert	10	0	0	0

Evaluierungsmaße für die Clustering-Ergebnisse sind für beide Ansätze in Tabelle 6.6 gegenübergestellt. Beide Ähnlichkeitsmaße führen zu einer ähnlichen Anzahl von Ausreißern. Insgesamt betrachtet erzeugt DBScan in Verbindung mit IP2Vec Cluster, die eine höhere Homogenität und Vollständigkeit aufweisen. Dies spiegelt sich vor allem im Evaluierungsmaß Homogenität wider, wo IP2Vec homogene Cluster erzeugt und den Bestwert erreicht, wohingegen GRAPH nicht in der Lage war, die Cluster korrekt zu trennen und nur schlechte Werte erzielt.

Tabelle 6.6.: Evaluierungsmaßzahlen der Clusteranalyse für Szenario 9 des CTU-13 Datensatzes.

Ähnlichkeitsmaß	Homogenität	Vollständigkeit
IP2Vec	**1.0**	**0.1072**
GRAPH	0.0248	0.0029

6.3.3. Experiment 2 - Identifizierung von Server und Clients

Experiment 2 trainiert IP2Vec und GRAPH auf den oben beschriebenen Teil des CIDDS-001 Datensatzes. Nach der Trainingsphase werden alle internen IP-Adressen (192.168.X.X) mit t-SNE visualisiert und mit DBScan einer Clusteranalyse unterzogen.

Visualisierung. Die Visualisierung ist für IP2Vec in Abbildung 6.7(a) und für GRAPH in Abbildung 6.7(b) zu sehen. Unter Verwendung von IP2Vec ist eine visuelle Separierung von Clients und Servern möglich. Hierbei ist zu erkennen, dass die Ähnlichkeiten zwischen verschiedenen Servern geringer ausfallen. Abbildung 6.7(b) illustriert die Visualisierung derselben IP-Adressen für das Abstandsmaß GRAPH. In dieser Darstellung können Server und Clients visuell nicht sauber voneinander getrennt werden. Stattdessen weisen nahezu alle IP-Adressen ähnliche Abstände zu ihren nächsten Nachbarn auf.

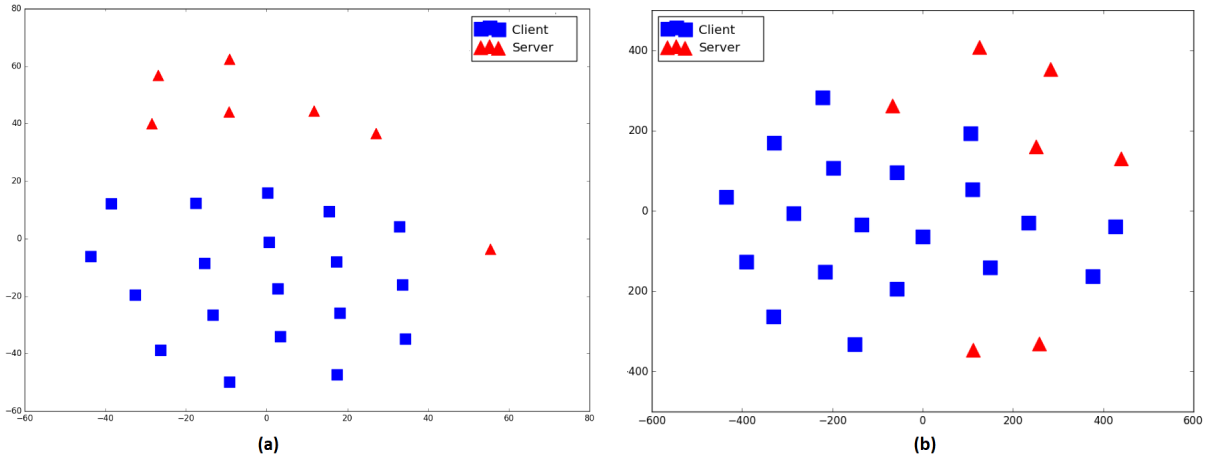


Abbildung 6.7.: t-SNE Visualisierung für IP-Adressen aus Woche 3 des CIDDS-001 Datensatzes [RLD+17]. In der linken Abbildung (a) werden Ähnlichkeiten zwischen IP-Adressen mit IP2Vec und in der rechten Abbildung (b) mit GRAPH ermittelt.

Clusteranalyse. Anschließend werden beide Ähnlichkeitsmaße genutzt, um IP-Adressen bezüglich ihres Verhaltens in zwei Gruppen (*Server* und *Client*) zu clustern. Die Zuweisungsmatrix für IP2Vec ist in Tabelle 6.7 und für GRAPH in Tabelle 6.8 abgebildet. Ergebnisse der Evaluierungsmaße sind für beide Ansätze in Tabelle 6.9 gegenübergestellt.

Tabelle 6.7.: Zuweisungsmatrix für den CIDDS-001 Datensatz (Woche 3) unter Verwendung von IP2Vec als Ähnlichkeitsmaß.

Klasse	Cluster 1	Cluster 2	Anzahl Ausreißer
server	0	6	1
client	19	0	0

Für IP2Vec konnte DBScan Server und Clients in verschiedene Cluster gruppieren. Das Clusterverfahren hat mit IP2Vec 2 Cluster und 1 Ausreißer generiert. Dabei konnten alle Clients dem Cluster 1 zugeordnet werden. Die Server teilen sich auf Cluster 2 und den einen Ausreißer auf. Die Verwendung von GRAPH als Ähnlichkeitsmaß führt zu keinen eindeutigen (homogenen) Clustern. In diesem Fall hat DBScan drei Cluster und keinen Ausreißer erzeugt. Bei diesem

Tabelle 6.8.: Zuweisungsmatrix für den CIDDS-001 Datensatz (Woche 3) unter Verwendung von GRAPH als Ähnlichkeitsmaß.

Klasse	Cluster 1	Cluster 2	Cluster 3	Anzahl Ausreißer
server	2	1	4	0
client	17	2	0	0

Clustering verteilen sich die 7 Server auf alle drei Cluster. Die meisten Clients wurden dem Cluster 1 zugeordnet.

Tabelle 6.9.: Evaluierungsmaßzahlen der Clusteranalyse für Woche 3 des CIDDS-001 Datensatzes.

Ähnlichkeitsmaß	Homogenität	Vollständigkeit
IP2Vec	**1.0**	**0.8406**
GRAPH	0.4518	0.3434

Die Ergebnisse der Evaluierungsmaße sind für beide Ähnlichkeitsmaße in Tabelle 6.9 dargestellt. Diese zeigen, dass IP2Vec in beiden Maßen Homogenität und Vollständigkeit bessere Ergebnisse erzielt als GRAPH. Dies spiegelt sich vor allem im Evaluierungsmaß Homogenität wider, wo IP2Vec homogene Cluster erzeugt und den Bestwert erreicht. Auch konnte IP2Vec wesentlich bessere Werte für das Maß Vollständigkeit erzielen, da alle Clients Cluster 1 und fast alle Server Cluster 2 zugeordnet wurden.

6.4. Diskussion

6.4.1. Experiment 1 - Identifikation von Botnets

In Experiment 1 wurden die beiden Ähnlichkeitsmaße hinsichtlich ihrer Eignung zur Detektion von mit Botnetzen infizierten IP-Adressen evaluiert. In diesem Experiment erzielte IP2Vec bessere Ergebnisse als GRAPH sowohl in der Visualisierung als auch in der Clusteranalyse. Der Grund hierfür ist, dass GRAPH lediglich statische hierarchische Informationen zur Abstandsberechnung verwendet und das tatsächliche Verhalten von IP-Adressen nicht widerspiegeln kann. Im Gegensatz dazu berücksichtigt IP2Vec das beobachtbare Netzwerkverhalten der IP-Adressen. Mit Botnetzen infizierte IP-Adressen weisen in der Regel ähnliche Verhaltensmuster auf, da sich diese versuchen zu verbreiten, den Bot-Master zu kontaktieren oder gemeinsam das gleiche Ziel zu attackieren. Aus diesem Grund führen in IP2Vec infizierte IP-Adressen zu ähnlicheren Vektorrepräsentationen zueinander, als zu nicht infizierten IP-Adressen.

Im Evaluierungskontext Clusteranalyse ist eine Trennung der generierten Vektordarstellungen von IP2Vec für normale und infizierte IP-Adressen möglich. Eine detaillierte Analyse zeigt, dass die infizierten Hosts sehr hohe Ähnlichkeiten zueinander aufweisen, sodass alle infizierten Hosts bereits bei kleinen ϵ-Werten einem gemeinsamen Cluster zugeordnet werden konnten.

Eine weitere Erkenntnis von Experiment 1 ist, dass die durch IP2Vec erlernten Vektorrepräsentationen für normale Hosts geringere Ähnlichkeiten zueinander aufweisen, als für infizierte Hosts. Dies kann auf die Ursache zurückgeführt werden, dass sich das Verhalten normaler Hosts naturgemäß stark unterscheidet, da unterschiedliche Webseiten aufgerufen und zur Verfügung stehenden Dienste unterschiedlich intensiv genutzt werden. Deshalb konnten für IP2Vec nicht

alle normalen Hosts einem gemeinsamen Cluster zugeordnet werden. Um die Ähnlichkeit zwischen normalen Hosts zu erhöhen, müsste explizites Domänenwissen integriert werden, indem beispielsweise typisch genutzte externe IP-Adressen, die für IP2Vec unterschiedliche Ziele darstellen, generalisiert und als ein Ziel zusammengefasst werden. Derartige Zusammenfassungen würden die Ähnlichkeiten zwischen normalen Hosts in IP2Vec erhöhen.

6.4.2. Experiment 2 - Identifizierung von Server und Clients

In Experiment 2 wurden die beiden Ähnlichkeitsmaße hinsichtlich ihrer Eignung zur Trennung von Clients und Servern evaluiert.

Auch in diesem Experiment konnte IP2Vec bessere Ergebnisse als GRAPH erzielen. Die Verwendung des Abstandsmaßes GRAPH führte zu keiner erfolgreichen Trennung von Servern und Clients. Diese Tatsache zeigt sich in Abbildung 6.6, da Server und Clients nicht eindeutig voneinander getrennt werden können und bestätigt sich auch in der Clusteranalyse, da die entstandenen Cluster 1 und 2 (siehe Tabelle 6.8) sowohl Clients als auch Server beinhalten. Dies liegt vor allem daran, dass GRAPH das Verhalten der IP-Adressen nicht berücksichtigt und stattdessen nur auf das Wissen der vordefinierten Hierarchie zurückgreift. Damit GRAPH diese Aufgabenstellung korrekt bewältigen könnte, müsste in der Hierarchie eine weitere Ebene eingeführt werden, welche die interne Struktur des Netzwerks berücksichtigt.

Die Verwendung des Ähnlichkeitsmaßes IP2Vec führt zur erfolgreichen Trennung von Servern und Clients. In Experiment 2 wurde ein Datensatz ohne Angriffsszenarien (Woche 3 des CIDDS-001 Datensatzes) ausgewählt. In der Regel bieten Server Dienste an, welche von Clients genutzt werden. Dies hat zur Folge, dass Clients fast ausschließlich Netzwerkverbindungen zu Servern und nicht zu anderen Clients aufbauen. Des Weiteren verwenden Server häufig sogenannte standardisierte Ports (0-1023), während Clients höhere Ports verwenden. IP2Vec ist in der Lage diese Zusammenhänge zu erfassen. Die durch IP2Vec erlernten Vektorrepräsentationen von Servern und Clients können sowohl in Abbildung 6.7 als auch in den entstandenen Clustern (siehe Tabelle 6.7) voneinander getrennt werden.

Eine weitere Beobachtung für IP2Vec ist, dass in der Clusteranalyse alle Clients dem gleichen Cluster zugeordnet werden konnten, währenddessen sich ein Server keinem Cluster zuordnen ließ. Insgesamt beinhaltet der CIDDS-001 Datensatz sieben interne Server. Darunter befinden sich drei Drucker, ein Web-Server, ein E-Mail-Server, ein Backup-Server und ein Netzlaufwerk. Da die Server unterschiedliche Dienste anbieten, unterscheidet sich das Verhalten der verschiedenen Server stärker als das der Clients. Dies ist die Ursache dafür, dass ein Server (das Netzwerklaufwerk) stark unterschiedliches Verhalten aufweist und keinem Cluster zugeordnet werden konnte.

6.5. Zusammenfassung

IP-Adressen sind allgegenwärtig in netzwerkbasierten Daten. Die Tatsache, dass IP-Adressen kategorische Attribute sind und über keine natürliche Ordnung verfügen, erschwert die Visualisierung und Ähnlichkeitsberechnung von IP-Adressen.

In diesem Kapitel wurde eine unüberwachte Methode namens IP2Vec vorgestellt, welche IP-Adressen in kontinuierliche Vektoren transformiert, die Informationen über das Netzwerkverhalten der IP-Adressen beinhalten. Für die Transformation verwendet IP2Vec verfügbare Kontextinformationen aus flowbasierten Netzwerkdaten und kompensiert somit die fehlende inhärente Ordnungsrelation. IP2Vec adaptiert den Ansatz von Word2Vec und verwendet ein neuronales

Netz mit einer versteckten Schicht. Die versteckte Schicht verfügt über viel weniger Neuronen als die Ein- und Ausgabeschicht und führt dadurch zu einer Generalisierung. Nach dem Training des neuronalen Netzes stellen die Gewichte der versteckten Schicht die kontinuierlichen Vektorrepräsentationen der IP-Adressen dar.

Ein Vorteil von IP2Vec ist die Transformation von IP-Adressen in kontinuierliche Vektoren. Diese Vektoren können als Eingabe für nachgelagerte Analysemethoden oder Visualisierungsverfahren verwendet werden. Im vorgestellten Ansatz wurden die Quell-IP-Adresse, Ziel-IP-Adresse, Ziel-Port und Transport-Protokoll verwendet. Dies hat den zusätzlichen Vorteil, dass für alle kategorischen Attribute eines Flows kontinuierliche Vektorrepräsentationen erlernt werden. Folglich können auch Ähnlichkeiten zwischen Ports oder Transport-Protokollen berechnet werden. Ein weiterer Vorteil von IP2Vec ist, dass die selektierten Kontextattribute für jede Zielstellung adaptiv anpassbar sind. Falls ein Vergleich von IP-Adressen bezüglich der benötigten Bandbreiten erforderlich sein sollte, können die Attribute Bytes und Pakete in die Menge der Kontextattribute aufgenommen werden.

Die experimentelle Evaluierung von IP2Vec unterstreicht dessen grundsätzliche Eignung. Für beide Evaluationsszenarien, Trennung von normalen und infizierten Hosts sowie Trennung von Servern und Clients, konnte IP2Vec gute Ergebnisse in der Clusteranalyse und Visualisierung erzielen.

Um das Anwendungsgebiet von IP2Vec auf Datenströme zu erweitern, sind zukünftig zwei weitere Herausforderungen zu bewältigen. Wenn auf Datenströmen gearbeitet wird, treten im Lauf der Zeit neue IP-Adressen auf, für die keine Vektorrepräsentationen vorhanden sind. Des Weiteren muss die Tatsache berücksichtigt werden, dass sich das Verhalten von IP-Adressen im Lauf der Zeit verändern kann und somit Aktualisierungen der Vektorrepräsentationen erforderlich sind. Beide Herausforderungen stellen somit relevante Fragestellungen für den erfolgreichen Einsatz von IP2Vec in Klassifikationsszenarien dar.

Teil III.

Generierung von Netzwerkdaten

7. Analyse existierender netzwerkbasierter Daten

Teil III beschäftigt sich mit der Generierung flowbasierter Netzwerkdaten (siehe Herausforderung 2 in Abschnitt 1.2.2). In diesem Kapitel findet zunächst ein Literaturüberblick existierender paket- und flowbasierter Daten statt. Dieser berücksichtigt auch paketbasierte Daten, da diese verlustfrei in flowbasierte Daten umgewandelt werden können. Um die Eignung von konkreten Datensätzen für bestimmte Anwendungsszenarien bewerten zu können, identifiziert dieses Kapitel zunächst diverse Bewertungskriterien. Diese decken ein breites Spektrum an typischen Eigenschaften ab und lassen sich in fünf Kategorien unterteilen. Basierend auf diesen Bewertungskriterien wird ein umfassender Überblick existierender Datensätze gegeben und Besonderheiten hervorgehoben. Darüber hinaus thematisiert dieses Kapitel Netzwerkdaten-Generatoren als weitere Quelle für netzwerkbasierte Daten. Das Kapitel endet mit einer Diskussion und Schlussfolgerungen für die Nutzung und Generierung netzwerkbasierter Datensätze. Die Inhalte von diesem Kapitel wurden bereits in [RWS+19] publiziert.

7.1. Bewertungskriterien

Um die Eignung von netzwerkbasierten Datensätzen im Bereich Intrusion Detection für bestimmte Anwendungsszenarien beurteilen zu können, ist die Definition einer allgemeingültigen Bewertungsgrundlage erforderlich. Das Konzept FAIR [WDA+16] definiert vier Prinzipien, welche Daten aus der Forschung erfüllen sollen, nämlich Auffindbarkeit (Findability), Zugänglichkeit (Accessibility), Interoperabilität (Interoperability) und Wiederverwendbarkeit (Reusability). FAIR beschreibt dabei domänenunabhängige Prinzipien, die auf ein breites Spektrum von wissenschaftlichen Daten anwendbar sind. Im Folgenden werden spezifische Bewertungskriterien für netzwerkbasierte Daten erarbeitet, die im Einklang mit FAIR [WDA+16] stehen. Zunächst wurden existierenden Bewertungskriterien für Datensätze analysiert. Existierende Datensätze legen unterschiedlichen Wert auf unterschiedliche Aspekte. Beispielsweise fokussiert der UGR'16 Datensatz [MCM+18] eine lange Aufnahmezeit zur Abbildung periodischer Effekte, währenddessen der ISCX 2012 Datensatz [SST+12] das exakte Labeln der Daten anstrebt. Des Weiteren bewerten einige Arbeiten die Präsenz bestimmter Angriffsszenarien (zum Beispiel DoS, Port Scans oder SSH-Brute-Force) oder bestimmter Arten von Normalverhalten (zum Beispiel HTTP, HTTPS oder SSH) als separate Bewertungskriterien. Die nachfolgende Auflistung vereinheitlicht und generalisiert existierende Bewertungskriterien. Beispielsweise verwendet dieses Kapitel lediglich die Präsenz von Angriffsszenarien und Normalverhalten als jeweils ein Bewertungskriterium. Besonderheiten werden in textueller Form für jeden Datensatz in Abschnitt 7.2 diskutiert.

Im Folgenden werden typische Bewertungskriterien eingeführt und in fünf Kategorien Allgemeine Informationen, Beschaffenheit der Daten, Datenmenge, Aufnahmeumgebung und Evaluierung unterteilt. Dieses Kapitel verzichtet auf eine Gewichtung der Bewertungskriterien, da unterschiedliche Anwendungsszenarien unterschiedliche Anforderungen mit sich bringen.

7.1.1. Allgemeine Informationen

Die folgenden vier Bewertungskriterien reflektieren generelle Informationen über das Alter, die Verfügbarkeit und die Präsenz von Normalverhalten und Angriffsszenarien der Datensätze.

Jahr der Erstellung. Netzwerkverhalten verändert sich im Laufe der Zeit. Unternehmen führen neue Netzwerkdienste ein, schalten alte Dienste ab und es erscheinen täglich neue Angriffsszenarien. Somit kommt dem Alter eines Datensatzes eine zentrale Bedeutung zu. Dieses Kriterium beschreibt das Jahr, in welchem der zugrundeliegende Netzwerkverkehr aufgenommen wurde, und nicht das Jahr, in welchem er veröffentlicht wurde. Beispielsweise ist das Jahr der Erstellung für den NSL-KDD Datensatz 1998 (siehe Tabelle 7.2), da dieser die Netzwerkdaten des DARPA 1998 Datensatzes aufbereitet hat.

Verfügbarkeit. Nur frei verfügbare Datensätze können als Grundlage für den Vergleich verschiedener Methoden und IDS dienen. Gleichzeitig kann die Qualität eines Datensatzes nur dann von Dritten geprüft beziehungsweise bestätigt werden, wenn dieser öffentlich zugänglich ist. Das Bewertungskriterium Verfügbarkeit besitzt in Tabelle 7.2 die drei Ausprägungen *ja*, *a.A. (auf Anfrage)* und *nein*. *Auf Anfrage* bedeutet, dass die Datensätze nicht frei zugänglich sind und Zugriff nur gegebenenfalls auf Anfrage gewährt wird.

Das Bewertungskriterium Verfügbarkeit steht im Zusammenhang mit den Prinzipien der Auffindbarkeit (Findability), Zugänglichkeit (Accessibility) und Wiederverwendbarkeit (Reusability) von FAIR. Während FAIR die Zugänglichkeit der Daten über eine eindeutige und persistente Identifizierung und Wiederverwendbarkeit durch entsprechende Lizenzen anstrebt, prüft diese Arbeit lediglich, ob Daten auffindbar und zugänglich sind.

Normalverhalten. Dieses Bewertungskriterium informiert über die Präsenz von normalen Benutzerverhalten in einem Datensatz und kann die Ausprägungen *ja* oder *nein* annehmen. Der Wert *ja* indiziert die Präsenz von normalen Benutzerverhalten, sagt jedoch nichts über die Präsenz von Angriffsszenarien aus. Der Wert *nein* indiziert das Fehlen von normalen Benutzerverhalten. Grundsätzlich wird die Qualität eines IDS in erster Linie durch die Erkennungsrate von Angriffen und der Anzahl von Fehlalarmierungen bestimmt [Owe10]. Deswegen ist die Präsenz von normalen Benutzerverhalten in einem Evaluierungsdatensatz unverzichtbar. Fehlendes Normalverhalten macht einen Datensatz jedoch nicht unbrauchbar und deutet stattdessen darauf hin, dass dieser Datensatz mit weiteren Daten kombiniert werden muss. Ein valider Ansatz wäre die Kombination mit realen Netzwerkverkehr eines Unternehmens. Derartige Kombination werden oftmals als Overlaying oder Salting [AH11, CRK+11] bezeichnet.

Angriffsszenarien. Datensätze sollten diverse Angriffsszenarien beinhalten. Dieses Bewertungskriterium prüft, ob im Datensatz Angriffsszenarien vorhanden sind oder nicht und kann die Werte *ja* oder *nein* annehmen. Sobald mindestens ein Angriffsszenario im Datensatz vorhanden ist, wird der Wert auf *ja* gesetzt. Detaillierte Informationen über Angriffsszenarien der Datensätze sind in Tabelle 7.3 beschrieben.

7.1.2. Beschaffenheit der Daten

Bewertungskriterien dieser Kategorie behandeln das Format und die zur Verfügung stehenden Attribute und Zusatzinformationen.

Metadaten. Inhaltliche Interpretation von paket- und flowbasierten Netzwerkdaten ist schwierig für Dritte. Daher sollten Datensätze mit zusätzlichen Metadaten veröffentlicht werden, welche Informationen über die Netzwerkstruktur, IP-Adressen und Angriffsszenarien beinhalten. Dieses Bewertungskriterium bezieht sich auf die Verfügbarkeit zusätzlicher Metadaten und kann die Werte *ja*, *einige* und *nein* annehmen.

Das Bewertungskriterium Metadaten findet sich im Einklang mit allen vier Prinzipien von FAIR. FAIR strebt verfügbare Metadaten in standardisierten Formaten an, die frei zugänglich sind und genaue Beschreibungen der Attribute und Daten liefern. Dieses Bewertungskriterium prüft die Verfügbarkeit von Metadaten, jedoch nicht deren Format und Einhaltung anderer Formalitäten.

Format. Datensätze erscheinen in unterschiedlichen Formaten. Dieses Bewertungskriterium liefert Informationen über das Format der Datensätze. Netzwerkbasierte Daten können die Formate *Paket*, *unidirektionale Flows*, *bidirektionale Flows* oder *Sonstige* annehmen. *Sonstige* beschreibt alle Datensätze, die in keinem standardisierten Format aufgenommen wurden. Dies können beispielsweise flowbasierte Netzwerkdaten sein, die mit zusätzlichen Informationen aus hostbasierten Datenquellen angereichert wurden. Des Weiteren informiert dieses Bewertungskriterium über die Präsenz zusätzlicher Logdateien.

Im FAIR Prinzip der Interoperabilität (Interoperability) wird eine formale, zugängliche und allgemein anwendbare Wissensrepräsentation der Daten und Metadaten zugrunde gelegt. Diese Anforderung kann durch einheitliche und standardisierte Datenformate realisiert werden. Das Bewertungskriterium Format untersucht die entsprechenden Formate der Netzwerkdaten.

Anonymität. Datensätze können häufig aus datenschutzrechtlichen Gründen nicht oder nur in anonymisierter Form veröffentlicht werden. Dieses Bewertungskriterium analysiert, ob Daten anonymisiert wurden und welche Attribute von dieser Anonymisierung betroffen sind. Der Wert *keine* deutet darauf hin, dass keine Attribute anonymisiert wurden. Der Wert *ja (IPs)* würde indizieren, dass IP-Adressen entweder anonymisiert oder entfernt wurden. Ebenso zeigt *ja (Payload)* an, dass Paketinhalte aus paketbasierten Netzwerkdaten anonymisiert oder entfernt wurden.

7.1.3. Datenmenge

Bewertungskriterien dieser Kategorie beschreiben das Volumen und die Aufnahmedauer.

Volumen. Dieses Bewertungskriterium beschreibt die Größe des Datensatzes und gibt die Anzahl der darin befindlichen Datenpunkte/Pakete/Flows oder die physikalische Größe in Gigabyte (GB) an.

Aufnahmedauer. Um periodische Effekte (Tageszeit/Nacht oder Wochentag/Wochenende) zu berücksichtigen, sollten Datensätze Netzwerkdaten über längere Zeiträume beinhalten. Dieses Bewertungskriterium liefert Informationen über die Zeitspanne des aufgenommenen Netzwerkverkehrs.

7.1.4. Aufnahmeumgebung

Die drei nachfolgenden Bewertungskriterien geben genauere Informationen über die Netzwerkumgebungen, in welchen die Datensätze aufgenommen wurden.

Typ. Dieses Bewertungskriterium beschreibt die Art des Netzwerkverkehrs und besitzt die Ausprägungen *real*, *simuliert* und *modelliert*. *Real* bedeutet, dass der Netzwerkverkehr in einer realen Netzwerkumgebung aufgenommen wurde. *Simuliert* bedeutet, dass der Netzwerkverkehr in einer Testumgebung oder mithilfe einer Simulation aufgenommen wurde. *Modelliert* bedeutet, dass der Netzwerkverkehr durch ein Modell (zum Beispiel mit einem Netzwerkdaten-Generator) synthetisch erzeugt und nicht an einem Netzwerkgerät aufgenommen wurde.

Netzwerkumgebung. Netzwerkumgebungen von kleinen und mittleren Unternehmen (KMUs) unterscheiden sich grundlegend von Internet Service Providern (ISP). Infolgedessen erfordern verschiedene Netzwerkumgebungen unterschiedliche Sicherheitssysteme und speziell dafür angepasste Evaluierungsdatensätze. In diesem Bewertungskriterium wird die zugrundeliegende Netzwerkumgebung des Datensatzes beschrieben.

Kompletter Netzwerkverkehr. Dieses Bewertungskriterium prüft, ob der Datensatz den kompletten Netzwerkverkehr eines Netzwerks mit mehreren Hosts, Servern und Netzwerkgeräten beinhaltet. Beispielsweise würden Datensätze mit ausschließlich SSH-Netzwerkverkehr oder Honeypot-Datensätze den Wert *nein* für dieses Bewertungskriterium annehmen. Laut [JS11] ist ein Honeypot eine Technologie, die dazu dienen soll, Hacker anzulocken und somit frühzeitig Informationen über deren Angriffsverhalten zu sammeln. In der Regel findet an Honeypots nur verdächtiges Angriffsverhalten und kein reguläres Benutzerverhalten statt.

7.1.5. Evaluation

Bewertungskriterien dieser Kategorie behandeln relevante Eigenschaften für die Evaluation von IDS, nämlich ob die Datensätze vordefinierte Teilmengen besitzen, über Klassenlabels verfügen, und bezüglich ihrer Klassen balanciert sind.

Teilmengen. Dieses Bewertungskriterium prüft die Existenz vordefinierter Teilmengen für Training und Evaluierung und kann die Werte *ja*, *nein* und *nicht definiert (n.d.)* annehmen.

Balanciert. Data Mining-Methoden benötigen in der Regel Trainingsdatensätze, welche bezüglich ihrer Klassenlabels balanciert sind. Balanciert bedeutet in diesem Zusammenhang, dass der Datensatz von jeder Klasse die gleiche Anzahl Datenpunkte besitzt. Diese Tatsache würde jedoch die Realität nicht reflektieren, da realer Netzwerkverkehr nicht balanciert ist und normalerweise mehr Datenpunkte für normales Benutzerverhalten als für Angriffsszenarien enthält. Dieses Bewertungskriterium evaluiert, ob ein Datensatz hinsichtlich seiner Klassenlabels balanciert ist und nimmt die Ausprägungen *ja*, *nicht definiert (n.d.)* oder *nein* an.

Labels. Gelabelte Datensätze sind notwendig, um überwachte Algorithmen zu trainieren und eignen sich sehr gut zum Evaluieren von IDS. Dieses Bewertungskriterium analysiert die Präsenz von Klassenlabels und nimmt den Wert *ja* an, wenn mindestens zwei Klassenlabels (*normal* und *Angriff*) vorhanden sind. Mögliche Ausprägungen umfassen die Werte *ja*, *ja (BG)*, *ja (IDS)*, *indirekt* und *nein*. *Ja (BG)* bedeutet, dass es eine dritte Klasse *Background* gibt, deren Datenpunkte sowohl normalen Netzwerkverkehr als auch Angriffsszenarien darstellen können. *Ja (IDS)* bedeutet, dass eine Art IDS zur Erstellung der Labels verwendet wurde. Folglich können einige Labels falsch sein, wenn das IDS falsch lag. *Indirekt* bedeutet, dass keine Labels zur

Verfügung stehen, stattdessen gibt es zusätzliche Logdateien, aus denen eigene Labels erstellt werden können.

7.2. Datensätze

Realistische Datensätze sind eine wesentliche Anforderung für die erfolgreiche Entwicklung anomaliebasierter IDS [DM17]. Dieser Abschnitt diskutiert die Ergebnisse einer Literaturrecherche existierender netzwerkbasierter Datensätze. Die Literaturrecherche berücksichtigt auch paketbasierte Datensätze, da paketbasierte Daten verlustfrei in flowbasierte Daten umgewandelt werden können. Hostbasierte Datensätze wie ADFA-LD [CH13] finden jedoch keine Berücksichtigung. Für einen Überblick hostbasierter Datensätze sei an dieser Stelle auf Bridges et al. [BGVI+19] verwiesen.

Das Format eines Datensatzes und die Präsenz von Labels sind die ersten Bewertungskriterien bei der Suche nach geeigneten netzwerkbasierten Datensätzen. Die Methode (überwacht oder unüberwacht) entscheidet, ob Labels erforderlich sind und welche Arten von Daten (*Pakete*, *Flows*, *Sonstige*) verarbeitet werden. Deshalb kategorisiert Tabelle 7.1 die existierenden Datensätze bezüglich dieser Bewertungskriterien. Einige Datensätze wie CTU-13 kommen in mehreren Spalten vor, da diese in unterschiedlichen Datenformaten angeboten werden.

Tabelle 7.1.: Übersicht existierender netzwerkbasierter Datensätze [RWS+19]. (+) Datensatz ist frei verfügbar. (?) keine Informationen über die Verfügbarkeit gefunden. (–) Datensatz ist nicht frei verfügbar.

	Format		
Labels	Paket	Flow	Sonstige
ja	(+) Botnet	(+) CICIDS 2017	(+) AWID
	(+) CIC DoS	(+) CIDDS-001	(+) KDD CUP 99
	(+) CICIDS 2017	(+) CIDDS-002	(+) Kyoto 2006+
	(+) DARPA	(+) ISCX 2012	(+) NSL-KDD
	(+) DDoS 2016	(+) TUIDS	(+) UNSW-NB 15
	(+) ISCX 2012	(+) Twente	(?) PU-IDS
	(+) ISOT	(–) IRSC	(?) SSENET-2011
	(+) NDSec-1		(?) SSENET-2014
	(+) NGIDS-DS		(–) SANTA
	(+) TRAbID		
	(+) TUIDS		
	(+) UNSW-NB15		
	(–) IRSC		
ja mit BG	(+) CTU-13	(+) CTU-13	
		(+) UGR'16	
ja (IDS)		(?) PUF	
indirekt	(+) CDX	(+) SSHCure	
nein	(+) Booters	(+) Kent 2016	
		(+) UNIBS	
	(+) LBNL	(+) Unified Host and Network	

Ein detaillierter Vergleich aller Datensätze findet sich in Tabelle 7.2. Die in den Datensätzen enthaltenen Angriffsszenarien stellen ein wichtiges Bewertungskriterium dar und werden deshalb in Tabelle 7.3 ausführlich analysiert. Für eine genaue Definition der Angriffsarten sei auf Abschnitt 4.4 und auf Anwar et al. [AZZ+17] verwiesen. Einige Datensätze sind modifizierte beziehungsweise kombinierte Versionen anderer Datensätze. Die Beziehungen zwischen den Datensätzen aus Tabelle 7.2 stellt Abbildung 7.1 graphisch dar.

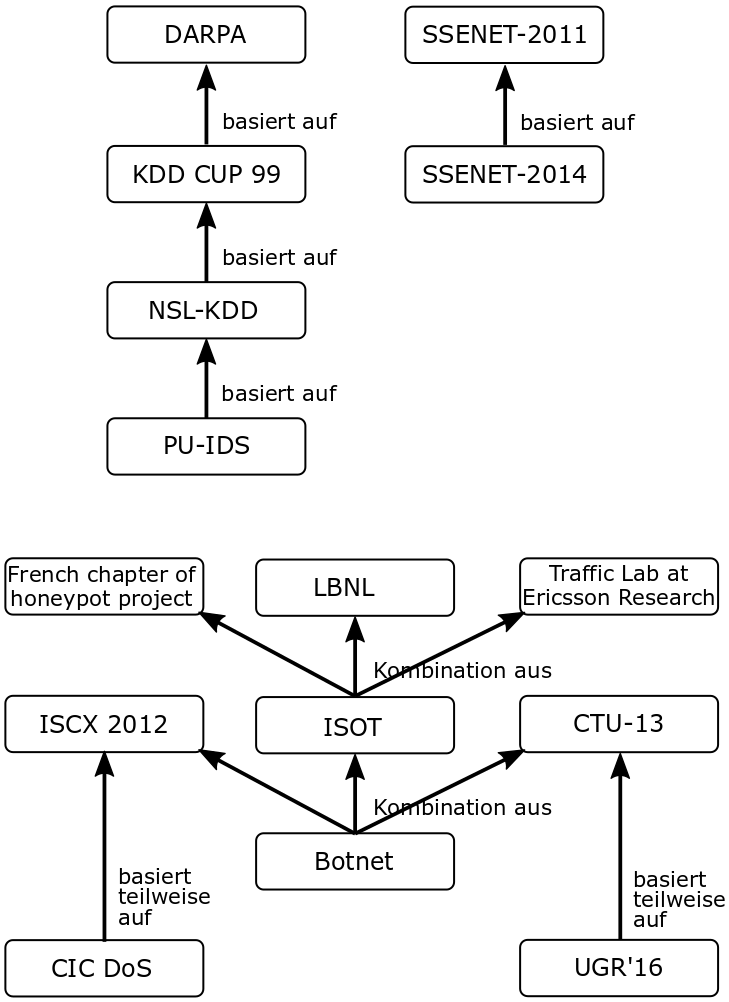


Abbildung 7.1.: Beziehungen und Zusammenhänge existierender netzwerkbasierter Datensätze [RWS+19].

Im Folgenden werden existierende netzwerkbasierte Datensätze vorgestellt und Besonderheiten hervorgehoben.

AWID [KKS+16]. AWID ist ein Intrusion Detection Datensatz mit speziellen Fokus auf 802.11 Netzwerke. Zur Erstellung des Datensatzes wurde der Netzwerkverkehr eines WLANs aufgezeichnet. Das Netzwerk umfasste 11 Clients, der Netzwerkverkehr wurde für eine Stunde im paketbasierten Format aufgezeichnet und neue Datenpunkte mit jeweils 156 Attribute pro Paket erstellt. Der AWID Datensatz ist frei verfügbar[1] und ist dem Format *Sonstige* zuzuordnen. Insgesamt umfasst der Datensatz 37 Millionen Datenpunkte und enthält 16 spezifische Angriffss-

[1] http://icsdweb.aegean.gr/awid/index.html

Tabelle 7.2.: Überblick netzwerkbasierter Datensätze nach Ring et al. [RWS+19].

	Allgemeine Informationen				Beschaffenheit der Daten			Datenmenge		Aufnahmeumgebung			Evaluation		
Datensatz	Jahr der Erstellung	Verfügbarkeit	Normalverhalten	Angriffsszenarien	Metadaten	Format	Anonymität	Volumen	Aufnahmedauer	Typ	Netzwerkumgebung	Kompl. Netz.	Teilmengen	Balanciert	Labels
AWID [KKS+16]	2015	a.A.	ja	ja	ja	Sonstige	keine	37M DP	1 Stunde	simuliert	kleine Netzwerkumgebung	ja	ja	nein	ja
Booters [SRH+15]	2013	ja	nein	ja	nein	Paket	ja (Booters)	250GB Pakete	2 Tage	echt	kleine Netzwerkumgebung	nein	nein	nein	nein
Botnet [BJS+14]	2014	ja	ja	ja	ja	Paket	keine	14GB Pakete	n.d.	simuliert	diverse Netzwerkumgebungen	ja	ja	nein	ja
CIC DoS [JGS+17]	2017	ja	ja	ja	nein	Paket	keine	4,6GB Pakete	24 Stunden	simuliert	kleine Netzwerkumgebung	ja	nein	nein	ja
CICIDS 2017 [SLG18]	2017	ja	ja	ja	ja	Paket, bi. Flow	keine	3,1M Flows	5 Tage	simuliert	kleine Netzwerkumgebung	ja	nein	nein	ja
CIDDS-001 [RWG+17c]	2017	ja	ja	ja	ja	uni. Flow	ja (IPs)	32M Flows	28 Tage	simuliert und echt	kleine Netzwerkumgebung	ja	nein	nein	ja
CIDDS-002 [RWG+17b]	2017	ja	ja	ja	ja	uni. Flow	ja (IPs)	15M Flows	14 Tage	simuliert	kleine Netzwerkumgebung	ja	nein	nein	ja
CDX [SOC+09]	2009	ja	ja	ja	ja	Paket	keine	14GB Pakete	4 Tage	echt	kleine Netzwerkumgebung	ja	nein	nein	indirekt
CTU-13 [GGS+14]	2013	ja	ja	ja	ja	Paket, uni. Flow, bi. Flow	ja (payload)	81M Flows	125 Stunden	echt	Universitätsnetzwerk	ja	nein	nein	ja (BG)
DARPA [LFG+00, LHF+00, MIT]	1998/99	ja	ja	ja	ja	Paket, host-logs	keine	n.d.	7/5 weeks	simuliert	kleine Netzwerkumgebung	ja	ja	nein	ja
DDoS 2016 [AAH+16]	2016	ja	ja	ja	nein	Paket	ja (IPs)	2,1M Pakete	n.d.	modelliert	n.d.	n.d.	nein	nein	ja
IRSC [ZKS+15]	2015	nein	ja	ja	nein	Paket, Flow	n.d.	n.d.	n.d.	echt	Reale Netzwerkumgebung	ja	n.d.	n.d.	ja
ISCX 2012 [SST+12]	2012	ja	ja	ja	ja	Paket, bi. Flow	keine	2M Flows	7 Tage	simuliert	kleine Netzwerkumgebung	ja	nein	nein	ja
ISOT [STG+11]	2010	ja	ja	ja	ja	Paket	keine	11GB Pakete	n.d.	simuliert	kleine Netzwerkumgebung	ja	nein	nein	ja
KDD CUP 99 [Uni99]	1998	ja	ja	ja	nein	Sonstige	keine	5M DP	n.d.	simuliert	kleine Netzwerkumgebung	ja	ja	nein	ja
Kent 2016 [Ken15, Ken16]	2016	ja	ja	n.d.	nein	uni. Flow, logs	ja	130M Flows	58 Tage	echt	Enterprise Netzwerkumgebung	ja	nein	nein	nein
Koyto 2006+ [STO+11]	2006 bis 2009	ja	ja	ja	nein	Sonstige	ja (IPs)	93M DP	3 years	echt	Honeypot	nein	nein	nein	ja
LBNL [PAB+05]	2004 / 2005	ja	ja	ja	nein	Paket	ja	160M Pakete	100/5 Stunden	echt	Enterprise Netzwerkumgebung	ja	nein	nein	ja
NDSec-1 [BHK+17]	2016	a.A.	nein	ja	nein	Paket, logs	keine	3,5M Pakete	n.d.	simuliert	kleine Netzwerkumgebung	ja	nein	nein	ja
NGIDS-DS [HHS+17]	2016	ja	ja	ja	nein	Paket, host-logs	keine	1M Pakete	5 Tage	simuliert	kleine Netzwerkumgebung	ja	nein	nein	ja
NSL-KDD [TBL+09]	1998	ja	ja	ja	nein	Sonstige	keine	150k DP	n.d.	simuliert	kleine Netzwerkumgebung	ja	ja	nein	ja
PU-IDS [SKS15]	1998	n.d.	ja	ja	nein	Sonstige	keine	200k DP	n.d.	modelliert	kleine Netzwerkumgebung	ja	nein	nein	ja
PUF [SSG18]	2018	nein	ja	ja	nein	uni. Flow	ja (IPs)	300k Flows	3 Tage	echt	Universitätsnetzwerk	nein	nein	nein	ja (IDS)
SANTA [WKZ+14]	2014	nein	ja	ja	nein	Sonstige	ja (payload)	n.d.	n.d.	echt	ISP	ja	n.d.	nein	ja
SSENET-2011 [VHS11]	2011	n.d.	ja	ja	nein	Sonstige	keine	n.d.	4 Stunden	simuliert	kleine Netzwerkumgebung	ja	nein	nein	ja
SSENET-2014 [BS14]	2011	n.d.	ja	ja	nein	Sonstige	keine	200k DP	4 Stunden	simuliert	kleine Netzwerkumgebung	ja	ja	ja	ja
SSHCure [HHS+14]	2013 / 2014	ja	ja	ja	nein	uni. Flow, bi. Flow, logs	ja (IPs)	2,4GB Flows (komprimiert)	2 Monate	echt	Universitätsnetzwerk	ja	nein	nein	indirekt
TRAbID [VSO17]	2017	ja	ja	ja	nein	Paket	ja (IPs)	460M Pakete	8 Stunden	simuliert	kleine Netzwerkumgebung	ja	ja	nein	ja
TUIDS [GBB+12], [BBK15]	2011 / 2012	a.A.	ja	ja	nein	Paket, bi. Flow	keine	250k Flows	21 Tage	simuliert	mittlere Netzwerkumgebung	ja	ja	nein	ja
Twente [SSV+09]	2008	ja	nein	ja	ja	uni. Flow	ja (IPs)	14M Flows	6 Tage	echt	Honeypot	nein	nein	nein	ja
UGR 16 [MCM+18]	2016	ja	ja	ja	einige	uni. Flows	ja (IPs)	16900M Flows	4 Monate	echt	ISP	ja	ja	nein	ja (BG)
UNIBS [GSD+09]	2009	a.A.	ja	nein	nein	Flow	ja (IPs)	79k Flows	3 Tage	echt	Universitätsnetzwerk	ja	nein	nein	nein
Unified Host und Network [TKH18]	2017	ja	ja	n.d.	nein	bi. Flows, host-logs	ja (IPs und Datum)	150GB Flows (komprimiert)	90 Tage	echt	Enterprise Netzwerkumgebung	ja	nein	nein	nein
UNSW-NB15 [MS15]	2015	ja	ja	ja	ja	Paket, Sonstige	keine	2M DP	31 Stunden	simuliert	kleine Netzwerkumgebung	ja	ja	nein	ja

Kompl. Netz. = Kompletter Netzwerkverkehr, n.d. = nicht definiert, uni. = unidirektionale, bi. = bidirektionale, DP = Datenpunkte

Tabelle 7.3.: Angriffsszenarien der Datensätze. Detaillierte Informationen über konkret ausgeführte Angriffsszenarien und Tools stehen in Klammern nach Ring et al. [RWS+19].

Datensatz	Angriffsszenarien
AWID [KKS+16]	Typische Angriffe gegen 802.11 Netzwerke (Authentication Requests, ARP-Flooding, Injection, Probe Request)
Booters [SRH+15]	9 unterschiedliche DDoS Angriffe
Botnet [BJS+14]	Botnetze (Menti, Murlo, Neris, NSIS, Rbot, Sogou, Strom, Virut, Zeus)
CIC DoS [JGS+17]	DoS Angriffe auf Anwendungsebene (ausgeführt durch ddossim, Goldeneye, hulk, RUDY, Slowhttptest, Slowloris)
CICIDS 2017 [SLG18]	Botnetz (Ares), Cross-Site-Scripting, DoS (ausgeführt durch Hulk, GoldenEye, Slowloris, Slowhttptest), DDoS (ausgeführt durch LOIC), Heartbleed, Infiltration, SSH-Brute-Force, SQL-Injection
CIDDS-001 [RWG+17c]	DoS, Port Scans (Ping-Scan, SYN-Scan), SSH-Brute-Force
CIDDS-002 [RWG+17b]	Port Scans (ACK-Scan, FIN-Scan, Ping-Scan, UDP-Scan, SYN-Scan)
CDX [SOC+09]	nicht definiert
CTU-13 [GGS+14]	Botnetze (Menti, Murlo, Neris, NSIS, Rbot, Sogou, Virut)
DARPA [LFG+00],[LHF+00],[MI	DoS, Privilege Escalation (Remote-to-Local, User-to-Root), Probing
DDoS 2016 [AAH+16]	DDoS (HTTP-Flooding, SIDDOS, ICMP-Flooding, UDP-Flooding)
IRSC [ZKS+15]	nicht definiert
ISCX 2012 [SST+12]	Vier Angriffsszenarien (1: Infiltrieren des Netzwerks von innen; 2: DoS gegen HTTP; 3: DDoS anhand eines IRC Botnetzes; 4: SSH-Brute-Force)
ISOT [STG+11]	Botnetze (Storm, Waledac)
KDD CUP 99 [Uni99]	DoS, Privilege Escalation (Remote-to-Local, User-to-Root), Probing
Kent 2016 [Ken15], [Ken16]	nicht definiert
Koyto 2006+ [STO+11]	Verschiedene Angriffe gegen Honeypots (zum Beispiel DoS, Exploits, Malware, Port Scans, Shellcode)
LBNL [PAB+05]	Port Scans
MAWI [FBA+10]	DoS, DDoS, Port Scans, Würmer und weitere Angriffe aus realen Netzwerkverkehr
NDSec-1 [BHK+17]	Botnetz (Citadel), Brute-Force (gegen FTP, HTTP und SSH), DDoS (HTTP-Flooding, SYN-Flooding und UDP-Flodding), Exploits, Injections, Probing, Spoofing
NGIDS-DS [HHS+17]	Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, Würmer
NSL-KDD [TBL+09]	DoS, Privilege Escalation (Remote-to-Local, User-to-Root), Probing
PU-IDS [SKS15]	DoS, Privilege Escalation (Remote-to-Local, User-to-Root), Probing
PUF [SSG18]	Angriffe gegen DNS
SANTA [WKZ+14]	(D)DoS Angriffe (DNS-Amplification, ICMP-Flooding, RUDY, SYN-Flooding), Heartbleed, Port Scans
SSENET-2011 [VHS11]	DoS (ausgeführt durch LOIC), Port Scans (ausgeführt durch Angry IP Scanner, Nessus, Nmap), verschiedene Angrifftools (zum Beispiel Metasploit)
SSENET-2014 [BS14]	Botnetze, Flooding, Privilege Escalation, Probing
SSHCure [HHS+14]	SSH-Brute-Force
TRAbID [VSO17]	DoS (HTTP-Flooding, ICMP-Flooding, SMTP-Flooding, SYN-Flooding), Port Scans (ACK-Scan, FIN-Scan, NULL-Scan, OS-Fingerprinting, Service-Fingerprinting, UDP-Scan, XMAS-Scan)
TUIDS [GBB+12], [BBK15]	Botnetz (IRC), DDoS (Fraggle-Flooding, Ping-Flooding, RST-Flooding, ICMP-Flooding, SYN-Flooding, UDP-Flooding), Port Scans (FIN-Scan, NULL-Scan, UDP-Scan, XMAS-Scan), verteilte Port Scans, SSH-Brute-Force
Twente [SSV+09]	Angriffe gegen einen Honeypot mit drei Diensten (FTP, HTTP, SSH)
UGR'16 [MCM+18]	Blacklist, Botnetz (Neris), DoS, Port Scans, SSH-Brute-Force, Spam
UNIBS [GSD+09]	keine
Unified Host und Network [TKH18]	nicht definiert
UNSW-NB15 [MS15]	Backdoors, DoS, Exploits, Fuzzers, Reconnaissance, Shellcode, Würmer

zenarien gegen 802.11 Netzwerke. AWID ist gelabelt und besitzt vordefinierte Trainings- und Evaluierungsteilmengen.

Booters [SRH+15]. Booters sind als Service angebotene Distributed Denial of Service (DDoS) Angriffe. Santanna et al. [SRH+15] veröffentlichten im Jahr 2013 den Booters Datensatz, welcher 9 verschiedene DDoS-Angriffe beinhaltet. Der resultierende Datensatz ist im paketbasierten Format aufgenommen und beinhaltet mehr als 250 Gigabyte Netzwerkverkehr. Einzelne Pakete sind nicht gelabelt, aber die unterschiedlichen Booter sind in unterschiedliche Dateien aufgeteilt. Der Datensatz ist frei verfügbar[2] und die Namen der Booter sind anonymisiert.

Botnet [BJS+14]. Dieser Datensatz ist auf Botnetz-Angriffsszenarien fokussiert und stellt eine Kombination aus existierenden Datensätzen dar. Die Autoren des Datensatzes verwendeten die Overlay-Methode aus [AH11], um Teile der Datensätze ISOT [STG+11], ISCX 2012 [SST+12] und CTU-13 [GGS+14] miteinander zu kombinieren. Der Datensatz enthält normales Benutzerverhalten sowie verschiedene Botnetze im paketbasierten Format und ist öffentlich verfügbar[3]. Botnet ist in einem 5,3 Gigabyte großen Trainingsdatensatz und einem 8,5 Gigabyte großen Evaluierungsdatensatz aufgeteilt.

CIC DoS [JGS+17]. Der CIC DoS Datensatz stammt vom Canadian Institute for Cybersecurity und ist frei verfügbar[4]. Die Intention der Autoren war das Erstellen eines Datensatzes mit auf Anwendungsebene durchgeführten DoS-Angriffen. CIC DoS enthält vier unterschiedliche Arten von DoS-Angriffen und normales Benutzerverhalten wurde durch Integration von Daten aus dem ISCX 2012 Datensatz [SST+12] hinzugefügt. Der resultierende Datensatz ist im paketbasierten Format aufgezeichnet und enthält Netzwerkverkehr über einem Zeitraum von 24 Stunden.

CICIDS 2017 [SLG18]. CICIDS 2017 legt großen Wert auf die Präsenz von realistischen Benutzerverhalten und wurde als neuer Datensatz zur Entwicklung von IDS und IPS entwickelt. Dieser Datensatz wurde in einer simulierten Netzwerkumgebung aufgezeichnet und beinhaltet 3,1 Millionen bidirektionale Flows über einem Zeitraum von 5 Tagen. Der Datensatz ist ebenfalls im paketbasierten Format[5] erhältlich. Die Autoren extrahierten 80 flowbasierte Attribute und bieten weitere Metainformationen über IP-Adressen und Angriffsszenarien an. Sogenannte B-Profile wurden zum Erstellen von normalen Benutzerverhalten verwendet. Der CICIDS 2017 Datensatz beinhaltet einen großen Umfang verschiedener Angriffsszenarien wie SSH-Brute-Force, Heartbleed oder DDoS (siehe Tabelle 7.3).

CIDDS-001 [RWG+17c] und CIDDS-002 [RWG+17b]. Die beiden Datensätze CIDDS-001 und CIDDS-002 wurden im Rahmen dieser Dissertation erstellt und werden in den Abschnitten 8.3 und 8.4 ausführlich diskutiert. Die beiden Datensätze wurden zum Vergleich mit existierenden Datensätzen in den Tabellen 7.1, 7.2 und 7.3 mit aufgenommen.

CDX [SOC+09]. Sangster el al. [SOC+09] stellen ein Konzept zum Erstellen netzwerkbasierter Datensätze aus Warfare-Wettbewerben vor. Die Autoren diskutieren Vor- und Nachteile eines derartigen Ansatzes und erstellen einen Datensatz, welcher paketbasierten Netzwerkverkehr von 4 Tagen beinhaltet. CDX ist frei verfügbar[6] und beinhaltet diverse Angriffsszenarien sowie normales Benutzerverhalten. Zusätzlich stehen weitere Informationen über IP-Adressen zur Verfügung. Die Pakete sind nicht gelabelt, aber es stehen hostbasierte Logdateien und Warnungen eines IDS zur Verfügung.

[2] https://www.simpleweb.org/wiki/index.php
[3] http://www.unb.ca/cic/datasets/botnet.html
[4] http://www.unb.ca/cic/datasets/dos-dataset.html
[5] http://www.unb.ca/cic/datasets/ids-2017.html
[6] https://www.usma.edu/crc/sitepages/datasets.aspx

CTU-13 [GGS+14]. Dieser Datensatz wurde in einem Universitätsnetzwerk aufgenommen und beinhaltet 13 verschiedene Szenarien und pro Szenario wurde ein Botnetz ausgeführt. Zusätzliche Informationen über IP-Adressen und Netzwerkstrukturen stehen auf der Webseite zur Verfügung. CTU-13 ist in paketbasierten, unidirektionalen flowbasierten und bidirektionalen flowbasierten Format verfügbar[7]. Der Datensatz wurde mithilfe eines dreistufigen Prozesses gelabelt. Zunächst wurde der Netzwerkverkehr, der von Hosts verursacht wurde, die mit dem Botnetz infiziert sind, als *botnet* gelabelt. Im zweiten Schritt wurde Netzwerkverkehr, welcher auf spezifische Filter anschlägt, als *normal* gelabelt. Die restlichen Daten wurden mit dem Label *background* versehen.

DARPA [LHF+00],[MIT],[LFG+00]. Die DARPA 1998/99 Datensätze sind die verbreitetsten Datensätze im Bereich Intursion Detection und wurden am MIT Lincoln Laboratory entwickelt. Für die Generierung der beiden Datensätze wurde eine Netzwerkumgebung aufgesetzt und für sieben beziehungsweise fünf Wochen der Netzwerkverkehr im paketbasierten Format aufgezeichnet. Die Datensätze enthalten normales Benutzerverhalten und diverse Angriffe wie DoS, Port Scans oder Buffer Overflows. Zusätzliche Metadaten können auf der Webseite[8] gefunden werden. Obwohl (oder wegen) der weiten Verbreitung kritisieren einige Arbeiten die große Anzahl redundanter Datenpunkte und künstlicher Effekte dieser Datensätze [TBL+09, McH00, MC03].

DDoS 2016 [AAH+16]. Alkasassbeh et al. [AAH+16] veröffentlichten im Jahr *2016* einen paketbasierten Datensatz. Der Datensatz wurde mit dem Netzwerk Simulator NS2 erstellt und beinhaltet normalen Netzwerkverkehr und vier unterschiedliche Arten von DDoS-Angriffen: UDP-Flooding, ICMP-Flooding, HTTP-Flooding und SIDDOS. Die 2,1 Millionen Pakete des Datensatzes stehen in Researchgate[9] zur Verfügung.

IRSC [ZKS+15]. IRSC war als neuer Intrusion Detection Datensatz mit realen Netzwerkverkehr und unterschiedlichen Angriffsszenarien angedacht. Zuech et al. [ZKS+15] zeichneten diesen Datensatz im Jahr 2015 auf. IRSC enthält normales Benutzerverhalten, von den Autoren durchgeführte Angriffe sowie aus dem Internet aufgenommenen Netzwerkverkehr. Zum Labeln des Datensatzes wurde das IDS Snort in Kombination mit manueller Analyse von IT-Sicherheitsexperten verwendet. Aus datenschutzrechtlichen Gründen kann der Datensatz jedoch nicht veröffentlicht werden.

ISCX 2012 [SST+12]. ISCX 2012 ist ein weit verbreiteter Datensatz im Bereich Intrusion Detection und wurde im Jahr 2012 aufgenommen. Der Datensatz ist im paket- und bidirektionalen flowbasierten Format verfügbar[10]. ISCX 2012 wurde in einer simulierten Testumgebung aufgenommen und umfasst Netzwerkverkehr von 7 Tagen. Normales Benutzerverhalten und vier Angriffsszenarien wurden durch Skripte simuliert.

ISOT [STG+11]. ISOT wurde zur Evaluierung von P2P Botnetzen [STG+11] entwickelt. Der Datensatz ist frei verfügbar und beinhaltet 11 GB paketbasierten Netzwerkverkehr. Der ISOT Datensatz entstand im Jahr 2010 durch Kombination von normalen Netzwerkverkehr aus den Ericsson Research Lab [SOM+08], den Lawrence Berkeley National Lab (LBNL) [PAB+05] und der Integration von verdächtigen Netzwerkverkehr aus den Honeypot Projekt des French Chapter[11]. Der angegebene Link zum Honeypot Projekt des French Chapter ist aktuell nicht erreichtbar (Stand 5 August 2020).

[7] https://www.stratosphereips.org/datasets-ctu13
[8] https://www.ll.mit.edu/ideval/docs/index.html
[9] https://www.researchgate.net/profile/Mouhammd_Al-Kasassbeh/publication/292967044_Dataset-_Detecting_Distributed_Denial_of_Service_Attacks_Using_Data_Mining_Techniques
[10] http://www.unb.ca/cic/datasets/ids.html
[11] http://honeynet.org/chapters/france

KDD CUP 99 [Uni99]. Dieser Datensatz[12] basiert auf dem DARPA Datensatz und gehört zu den verbreitetsten Datensätzen im Bereich Intrusion Detection. KDD CUP 99 umfasst fünf Millionen Datenpunkte und liegt in keinem standardisierten Format vor und gehört somit der Kategorie Sonstige an. KDD CUP 99 beinhaltet normales Benutzerverhalten sowie verschiedene Angriffsszenarien durch die Ausführung von beispielsweise DoS- oder Buffer Overflow-Angriffen. Der Datensatz beinhaltet jedoch keine IP-Adressen, was typische Attribute netzwerkbasierter Daten sind.

Kent 2016 [Ken15], [Ken16]. Kent 2016 ist ein Intrusion Detection Datensatz, welcher mehrere Datenquellen beinhaltet. Neben unidirektionalen flowbasierten Netzwerkdaten stehen auch einige hostbasierte Datenquellen wie Windows Ereignisse auf der Webseite[13] zur Verfügung. Dieser Datensatz wurde über einen Zeitraum von 58 Tage am Los Alamos National Laboratory Netzwerk aufgenommen. Aus datenschutzrechtlichen Gründen wurden jedoch viele Attribute anonymisiert und die Flows besitzen keine Labels.

Kyoto 2006+ [STO+11]. Kyoto 2006+ ist ein frei verfügbarer Datensatz[14], welcher realen Netzwerkverkehr beinhaltet. Da es sich um einen Honeypot Datensatz handelt, befindet sich kaum normales Benutzerverhalten im Datensatz. Die Labels wurden durch das IDS Bro [15] erstellt. Kyoto 2006+ liegt in keinem standardisierten Format vor. Stattdessen wurden sogenannte Sessions aus den paketbasierten Netzwerkdaten extrahiert, die teilweise ähnliche Attribute wie der KDD CUP 99 Datensatz beinhalten. Der Datensatz umfasst Netzwerkverkehr von 3 Jahren und wird durch 93 Millionen Sessions abgebildet.

LBNL [PAB+05]. Im Bereich Intrusion Detection wird häufig auf den LBNL Datensatz verwiesen. Hauptmotivation für die Erstellung dieses Datensatzes war jedoch die Analyse von Merkmalen des Netzwerkverkehrs großer Unternehmen. Folglich zielt LBNL auf eine andere Zielstellung ab und verfügt über keine Labels, die den Netzwerkverkehr in *normal* und *Angriff* unterteilen. Dennoch könnte der Datensatz nach Ansicht der Autoren als Hintergrundverkehr für IDS verwendet werden, da er fast ausschließlich normales Nutzerverhalten enthält. LBNL kann auf der Webseite[16] im paketbasierten Format heruntergeladen werden, ist jedoch aus datenschutzrechtlichen Gründen anonymisiert.

NDSec-1 [BHK+17]. NDSec-1 wurde als Angriffsrepositorium entwickelt. Aufgrund des fehlenden Normalverhaltens ist dieser Datensatz zur Integration in andere Datensätze oder zur Integration in realen Netzwerkverkehr gedacht. Der Datensatz ist auf Anfrage erhältlich[17] und liegt im paketbasierten Format vor. NDSec-1 beinhaltet eine Vielzahl von Angriffsszenarien wie Botnetze, Brute-Force, Exploits und Injections.

NGIDS-DS [HHS+17]. NGIDS-DS ist ein simulierter Datensatz zur Evaluierung von IDS. Der Datensatz wurde im Jahr 2016 aufgenommen und beinhaltet 1 Million Netzwerkpakete. Zur Erstellung des Normalverhaltens und der Angriffsszenarien wurde im NGIDS-DS Datensatz das Tool IXIA Perfect Storm verwendet. Folglich ist die Qualität des Netzwerkverkehrs hinsichtlich Realität ausschließlich vom IXIA Perfect Storm Tool abhängig. Der Datensatz beinhaltet sieben unterschiedliche Arten von Angriffsszenarien (DoS, Würmer, etc.) und ist im paketbasierten Format frei verfügbar[18]. Des Weiteren stehen diverse hostbasierte Logdateien als zusätzliche Informationsquelle zur Verfügung.

[12] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[13] https://csr.lanl.gov/data/cyber1/
[14] http://www.takakura.com/Kyoto_data/
[15] https://www.bro.org/
[16] http://icir.org/enterprise-tracing/
[17] http://www2.hs-fulda.de/NDSec/NDSec-1/
[18] https://research.unsw.edu.au/people/professor-jiankun-hu

NSL-KDD [TBL+09]. NSL-KDD ist eine Weiterentwicklung des häufig in der Literatur kritisierten KDD CUP 99 Datensatzes. Ein wesentliches Problem des KDD CUP 99 Datensatzes ist die große Menge von Duplikaten [TBL+09]. Deswegen wurden doppelte Datenpunkte entfernt und die verbleibenden 150.000 Datenpunkte in neue Teilmengen für Training und Evaluation unterteilt. NSL-KDD beinhaltet somit die gleichen Attribute wie KDD CUP 99. Es sei jedoch angemerkt, dass die zugrundeliegenden Daten aus dem Jahr 1998 stammen. Der Datensatz ist frei verfügbar[19].

PU-IDS [SKS15]. Singh et al. [SKS15] entwickelten einen Generator für Netzwerkdaten, welcher basierend auf einer Menge von Eingabedaten neue Daten generieren kann. Die Autoren verwendeten den NSL-KDD Datensatz als Eingabe für ihren Generator, generierten 200.000 neue Datenpunkte und veröffentlichten diese als PU-IDS Datensatz. Folglich stammen die grundlegenden Eigenschaften des Netzwerkverkehrs vom PU-IDS Datensatz aus dem Jahr 1998, da NSL-KDD auf KDD CUP 99 beruht und dieser wiederum auf DARPA 98 (siehe auch Abbildung 7.1). Der Datensatz hat die gleichen Attribute wie KDD CUP 99 und NSL-KDD.

PUF [SSG18]. Der flowbasierte PUF Datensatz wurde in einem Campus-Netzwerk über einen Zeitraum von 3 Tagen aufgezeichnet und beinhaltet ausschließlich DNS-Verbindungen. Von den 298.463 unidirektionalen Flows stellen 38.120 Angriffsdaten dar und die verbleibenden Flows reflektieren normales Benutzerverhalten. Sharma et al. [SSG18] verwendeten ein IPS zum Labeln der Flows. Aus datenschutzrechtlichen Gründen wurden alle IP-Adressen aus dem Datensatz entfernt. Der Datensatz ist jedoch nicht frei verfügbar.

SANTA [WKZ+14]. SANTA ist ein weiterer Datensatz zur Evaluierung von IDS. Der Datensatz wurde in einer ISP-Umgebung aufgenommen und beinhaltet realen Netzwerkverkehr. SANTA liegt in einem nicht standardisierten Format vor und wurde durch eine aufwendige manuelle Analyse gelabelt. Die Datenpunkte ähneln NetFlow, verfügen jedoch über weitere Attribute, die aus paketbasierten Daten berechnet wurden. Der Datensatz ist nicht öffentlich verfügbar.

SSENet-2011 [VHS11]. SSENet-2011 ist ein simulierter Intrusion Detection Datensatz in nicht standardisierten Format. Der Datensatz umfasst Netzwerkverkehr über einen Zeitraum von vier Stunden und wird durch Datenpunkte mit jeweils 24 Attributen beschrieben. Mehrere reale Personen erzeugten normales Benutzerverhalten durch einen Webbrowser. Des Weiteren befinden sich einige Angriffe wie Port Scans im Datensatz. Jedoch lassen sich nur wenig Informationen über die durchgeführten Angriffe finden und der Datensatz ist nicht frei verfügbar.

SSENet-2014 [BS14]. SSENet-2014 basiert auf SSENet-2011. Dieser Datensatz liegt in keinem standardisierten Format vor und jeder Datenpunkt wird durch 28 Attribute beschrieben. Die Attribute wurde in Anlehnung an KDD CUP 99 erstellt. SSENet-2014 ist der einzige balancierte Trainingsdatensatz dieser Literaturrecherche. Der Datensatz ist nicht öffentlich verfügbar, wodurch keine genaueren Informationen über Normalverhalten und Angriffsszenarien gegeben werden können.

SSHCure [HHS+14]. Hofstede et al. [HHS+14] verwendeten zum Evaluieren ihres Angriffserkennungstools für SSH-Brute-Force Angriffe einen neuen Datensatz mit ausschließlich SSH-Verbindungen. Dieser Datensatz wird im Folgenden als SSHCure bezeichnet. SSHCure ist in einen Trainings- und Evaluierungsdatensatz unterteilt. Jeder dieser Teile enthält flowbasierten Netzwerkverkehr über einen Zeitraum von einem Monat. Die Flows sind nicht gelabelt, jedoch stehen hostbasierte Logdateien zur Verfügung, die Aussagen über erfolgreiche und fehlgeschlagene Anmeldungen geben. Der Datensatz ist frei verfügbar[20].

[19] http://www.unb.ca/cic/datasets/nsl.html
[20] https://www.simpleweb.org/wiki/index.php

TRAbID [VSO17]. Viegas et al. [VSO17] veröffentlichten im Jahr 2017 die TRAbID Datenbank. Diese Datenbank enthält 16 unterschiedliche Szenarien zur Evaluierung von IDS. Alle Szenarien wurden in einer simulierten Umgebung mit jeweils 1 Server und 100 Clients aufgenommen. Jedes Szenario wurde über einen Zeitraum von 30 Minuten aufgenommen und beinhaltet normalen Netzwerkverkehr und genau ein Angriffsszenario. Einige Clients führten ausschließlich Angriffe durch, während die meisten Clients ausschließlich normale Benutzeranfragen an den Server sendeten. Basierend auf dieser Tatsache wurden die IP-Adressen der Clients verwendet, um den aufgezeichneten Netzwerkverkehr zu labeln. Alle Clients sind Linux-Maschinen und das normale Nutzerverhalten umfasst HTTP-, SMTP-, SSH- und SNMP-Verkehr. Die Angriffsszenarien umfassen Port Scans und DoS-Angriffe. Der Datensatz ist öffentlich zugänglich[21].

TUIDS [GBB+12], [BBK15]. Der gelabelte TUIDS Datensatz lässt sich in drei Teile TUIDS Intrusion, TUIDS Coordinated Scan und TUIDS DDoS unterteilen. Der Datensatz wurde in einer simulierten Testumgebung mit 250 Clients aufgezeichnet und umfasst Netzwerkverkehr von 7 Tagen. Der Datensatz beinhaltet normales Benutzerverhalten und verschiedene Angriffsszenarien wie DDoS und Port Scans. TUIDS steht in paket- und bidirektionalen flowbasierten Format zur Verfügung[22].

Twente [SSV+09]. Sperotto et al. [SSV+09] veröffentlichten einen der ersten flowbasierten Datensätze im Jahr 2008. Bei diesem Datensatz handelt es sich um den Netzwerkverkehr eines Honeypots, auf dem die drei Dienste FTP, SSH und HTTP angeboten wurden. Folglich enthält der Datensatz fast ausschließlich Angriffsdaten und besitzt kein Normalverhalten. Der frei verfügbare Datensatz[23] beinhaltet 14 Millionen unidirektionale Flows. Aus datenschutzrechtlichen Gründen wurden alle IP-Adressen anonymisiert.

UGR'16 [MCM+18]. UGR'16 ist ein Intrusion Detection Datensatz mit Fokus auf einem großen Zeitraum, um periodische Effekte zu erfassen. UGR'16 wurde in einer ISP Umgebung für einen Zeitraum von 4 Monaten aufgenommen und beinhaltet knapp 17 Milliarden unidirektionale Flows. Die Autoren haben zusätzliche Angriffe in den Datensatz eingestreut, die mit *attack* gelabelt wurden. IP-Adressen sind anonymisiert und die Flows sind in die drei Klassen *normal*, *attack* und *background* klassifiziert. Der Datensatz ist frei verfügbar[24].

UNIBS 2009 [GSD+09]. Der UNIBS 2009 [GSD+09] wurde nicht für als Intrusion Detection Datensatz veröffentlicht. Die ursprüngliche Zielsetzung war die Identifikation von Anwendungen (zum Beispiel Webbrowser, Skype oder E-Mail) auf flowbasierten Netzwerkdaten. Deshalb sind die 79.000 Flows des Datensatzes mit Labels versehen, welche die Anwendung der Flows beschreiben. UNIBS 2009 beinhaltet ausschließlich normales Benutzerverhalten und ist frei verfügbar[25].

Unified Host and Network Data Set [TKH18]. Der Datensatz wurde in einer realen Netzwerkumgebung (Los Alamos National Laboratory) aufgenommen und steht in stark anonymisierter Form zur Verfügung[26]. Die Daten liegen im bidirektionalen flowbasierten Format vor und sind nicht gelabelt. Der Datensatz umfasst eine Aufnahmedauer von 90 Tagen.

UNSW-NB15 [MS15]. UNSW-NB15 wurde speziell als Datensatz für netzwerkbasierte IDS erstellt. Der Datensatz beinhaltet neun unterschiedliche Arten von Angriffen wie Backdoors, DoS, Exploits oder Würmer. Durchgeführte Angriffsszenarien und normales Benutzerverhalten wurden durch das IXIA Perfect Storm Tool erzeugt. Der Datensatz liegt im paket- und

[21] https://secplab.ppgia.pucpr.br/trabid
[22] http://agnigarh.tezu.ernet.in/~dkb/resources.html
[23] https://www.simpleweb.org/wiki/index.php
[24] https://nesg.ugr.es/nesg-ugr16/index.php
[25] http://netweb.ing.unibs.it/ntw/tools/traces/
[26] https://csr.lanl.gov/data/2017/

flowbasierten Format vor, umfasst einen Zeitraum von 31 Stunden und ist per Webseite[27] frei verfügbar.

Dieser Abschnitt zeigt, dass einige Datensätze im Bereich Netzwerkdaten existieren und sich bezüglich ihrer Formate, Verfügbarkeit und Präsenz von Labels gruppieren lassen (siehe Tabelle 7.1). Vor allem in den letzten Jahren wurden vermehrt neue Datensätze erstellt, die sich auf spezielle Angriffsszenarien und Netzwerkumgebungen fokussieren. Diese Entwicklung unterstreicht die Tatsache, dass für die Entwicklung neuer Intrusion Detection Methoden speziell angepasste Datensätze benötigt werden. Es ist jedoch festzustellen, dass einige Datensätze nicht verfügbar sind und oftmals keine genaueren Metadaten zur Verfügung stehen, welche von Dritten für genauere Analysen benötigt werden. Eine weitere Beobachtung ist, dass die in die Jahre gekommenen Datensätze DARPA und KDD CUP 99 immer noch häufig genutzt und als Grundlage zum Erstellen neuer Datensätze verwendet werden.

7.3. Netzwerkdaten-Generatoren

Netzwerkdaten-Generatoren stellen neben Datensätzen eine weitere Quelle für netzwerkbasierte Daten dar. Während Datensätze in der Regel aufgenommenen Netzwerkverkehr aus realen Netzwerk- oder Testumgebungen beinhalten, generieren Netzwerkdaten-Generatoren synthetische Daten anhand von Modellen. Diese Modelle verwenden entweder benutzerspezifische Eingabeparameter oder extrahieren Eigenschaften aus gegebenen Netzwerkverkehr. Im Gegensatz zu existierenden Datensätzen können Netzwerkdaten-Generatoren Netzwerkverkehr erzeugen, der bestimmte Strukturen und Anforderungen berücksichtigt.

Molnár et al. [MMS13] geben einen Überblick von Netzwerkdaten-Generatoren und kategorisieren diese in Bezug zu ihren Einsatzzweck. Des Weiteren analysieren die Autoren die verwendeten Evaluierungsmethoden und kommen zu dem Schluss, dass es keine allgemein gültigen Evaluierungsmethoden gibt. Diese Tatsache erschwert den Vergleich verschiedener Netzwerkdaten-Generatoren und definiert die Entwicklung geeigneter Evaluierungsmethoden als aktives Forschungsgebiet.

Dieses Kapitel fokussiert die Analyse auf flowbasierte Netzwerkdaten-Generatoren. Im Folgenden werden flowbasierte Netzwerkdaten-Generatoren in vier Kategorien (I) Replay Engines, (II) Maximum Throughput Generatoren, (III) Angriffsgeneratoren und (IV) High-Level Generatoren eingeteilt. Des Weiteren wird eine Kategorie (V) eingeführt, welche mithilfe von GANs Netzwerkdaten erzeugt.

Kategorie I Ansätze dieser Kategorie verfolgen eine andere Zielstellung, nämlich die tatsächliche Generierung von Netzwerkpaketen auf Netzwerkebene und nicht die Generierung von neuartigen Netzwerkverkehr. Derartige Methoden werden in der Literatur ebenfalls als Netzwerkdaten-Generatoren bezeichnet und der Vollständigkeit in diesem Überblick berücksichtigt.

Replay Engines verwenden in der Regel aufgenommenen paketbasierten Netzwerkverkehr und geben lediglich die Pakete zu einem späteren Zeitpunkt wider. Ziel dieser Netzwerkdaten-Generatoren ist die exakte Wiedergabe des Datenstroms unter Berücksichtigung der ursprünglichen Zeit zwischen den Paketen. TCPReplay[28] und TCPivo [FGB+03] sind bekannte Generatoren dieser Kategorie. Netzwerkdaten-Generatoren dieser Kategorie können somit lediglich bereits aufgenommenen Netzwerkverkehr wiedergeben, jedoch keine neuen Netzwerkdaten erzeugen.

[27] https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/
[28] https://linux.die.net/man/1/tcpreplay

Kategorie II Wie der Name schon sagt, dienen Maximum Throughput Generatoren zum Überprüfen der Ende-zu-Ende Netzwerkgeschwindigkeit. Ein derartiger Generator ist Iperf [29], welcher das Messen von Bandbreite, Verzögerungen und Verlustraten von Netzwerkverbindungen erlaubt. Ähnliche Zielstellungen verfolgen Rygielski et al. [RKZ13] und Rygielski und Kounev [RK14]. Beide Arbeiten stellen Modelle zur Vorhersage des netzwerkbasierten Datendurchsatzes in Rechenzentren vor. Somit besitzen Generatoren dieser Kategorie eine andere Zielsetzung und streben nicht das Generieren von inhaltlich sinnvollen Netzwerkverbindungen an.

Kategorie III Angriffsgeneratoren verwenden echten Netzwerkverkehr als Eingabe und kombinieren diesen mit synthetisch erzeugten Angriffen. FLAME [BWM08] und ID2T [VCM+16] sind Generatoren dieser Kategorie. FLAME verwendet einen regelbasierten Ansatz, um Angriffe wie Port Scans oder DoS in die Eingabedaten zu injizieren. ID2T verwendet ebenfalls regelbasierte Skripte zum Erzeugen von Angriffsdaten oder manipuliert die Parameter der Eingabedaten, damit diese die typischen Eigenschaften von Angriffsvektoren aufweisen. Ein weiterer Generator dieser Kategorie wurde von Sperotto et al. [SSB+09] vorgestellt. Sperotto et al. [SSB+09] modellieren SSH-Brute-Force Angriffe auf Flow-Ebene mithilfe eines Hidden Markov Modells. Das entworfene Modell ist in der Lage die typische Anzahl von Bytes, Paketen und Flows pro Sekunde für einen SSH-Brute-Force Angriff zu simulieren. Dabei werden jedoch keine vollständigen flowbasierten Netzwerkdaten erzeugt. GENESIDS [ED18] ist ein Generator für HTTP-Angriffsdaten, welcher auf Basis von benutzerdefinierten Beschreibungen arbeitet.

Kategorie IV High-Level Generatoren generieren synthetischen Netzwerkverkehr für Angriffsszenarien und normales Benutzerverhalten. Stiborek et al. [SRP15] präsentieren drei Methoden, um reales hostbasiertes Benutzerverhalten auf Flow-Ebene zu erzeugen. Die drei Methoden verwenden echten Netzwerkverkehr als Eingabe und extrahieren typische Beziehungen zwischen den Flows und innerhalb eines Flows. Anschließend werden synthetische flowbasierte Daten mithilfe von Wahrscheinlichkeitsmodellen, die mit den extrahierten Daten parametrisiert wurden, generiert. Siska et al. [SSK+10] entwickelten eine graphbasierte Methode zur Generierung flowbasierter Netzwerkdaten, welche realen Netzwerkverkehr als Eingabe verwendet und daraus Muster extrahiert. Hierbei wird pro Port (zum Beispiel Port 80 (HTTP) oder Port 53 (DNS)) ein Muster erzeugt. Diese Muster beinhalten Informationen über Mittelwert und Standardabweichung der Attribute Dauer und Pakete sowie Listen mit typischen Quell- und Ziel-IP-Adressen. Die automatisch extrahierten Muster können mit benutzerdefinierten Mustern erweitert werden. Letztendlich generiert der Netzwerkdaten-Generator neuen Netzwerkverkehr durch die Selektion von Werten aus den zur Verfügung stehenden Mustern. Iannucci et al. [IKG+17] stellen zwei weitere graphbasierte Netzwerkdaten-Generatoren Barabasi-Albert (PGPBA) und Kronecker (PGSK) vor. Beide Netzwerkdaten-Generatoren werden mit Netzwerkverkehr im paketbasierten Format initialisiert. Beide Modelle generieren im ersten Schritt den Wert für das Attribut Bytes. Anschließend werden alle anderen flowbasierten Attribute in Abhängigkeit vom Attribut Bytes berechnet.

Kategorie V GANs sind generative Modelle zur Erzeugung synthetischer Daten (siehe Abschnitt 3.4). Yin et al. [YZL+18] veröffentlichten die Methode Bot-Gan zur Erkennung von Botnet Angriffen. Bot-Gan generiert synthetische Netzwerkdaten, um die Anzahl der Trainingsdatenpunkte zu erhöhen und somit indirekt auch die Erkennungsrate von anomaliebasierten

[29] https://iperf.fr/

IDS durch größere Trainingsdatensätze zu verbessern. Bot-Gan erstellt jedoch keine vollständigen flowbasierten Netzwerkdaten, da keine kategorischen Attribute wie IP-Adressen und Ports erzeugt werden. Darüber hinaus gibt es weitere Ansätze, die durch das Erzeugen synthetischer Daten die Verbesserung der Erkennungsraten anomaliebasierter Sicherheitssysteme anstreben. Hier sei beispielsweise die Arbeit von Zheng et al. [ZZS+18] genannt. Zheng et al. [ZZS+18] erhöhen die Betrugserkennung bei Banküberweisungen mithilfe eines GAN-basierten Ansatzes.

Rigaki und Garcia [RG18] veröffentlichten einen GAN-basierten Ansatz, der die Kommunikation von Malware anpasst, so dass diese nicht mehr von Sicherheitssystemen detektiert werden kann. Deren Methode erlernt typische Eigenschaften von regulären Facebook-Chats auf Ebene von flowbasierten Netzwerkdaten. Diese Eigenschaften werden anschließend auf das Kommunikationsverhalten der Malware übertragen. Zur Evaluierung ihres Ansatzes verwenden die Autoren ein IPS, welches auf Markov Modellen beruht und drei flowbasierte Attribute auswertet. Mit diesen Ansatz konnten Rigaki und Garcia [RG18] erfolgreich das IPS umgehen. Daneben existieren noch weitere GAN-basierte Ansätze, die bösartigen Netzwerkverkehr mithilfe von synthetisch erzeugten Daten so anpassen, dass dieser Sicherheitssysteme austricksen kann. Beispielhaft seien IDSGAN [LSX18], MalGAN [HT17] oder Grosse et al. [GPM+17] genannt.

7.4. Diskussion und Schlussfolgerungen

Netzwerkbasierte Datensätze sind wichtiger Bestandteil für die Forschung im Bereich Intrusion Detection, da diese zum Trainieren und Evaluieren von IDS verwendet werden können. Anomaliebasierte IDS modellieren Normalverhalten basierend auf repräsentativen Trainingsdaten und heben Abweichungen vom erlernten Verhalten als Anomalien hervor (siehe Abschnitt 4.1.2). Signaturbasierte IDS verwenden hingegen eine Liste von bekannten Angriffsmustern und gleichen den eintreffenden Datenstrom mit diesen Signaturen ab (siehe Abschnitt 4.1.1). Somit eignen sich qualitativ hochwertige Datensätze zur Konfiguration und Evaluierung anomalie- und signaturbasierter IDS. Im Folgenden werden einige Erkenntnisse und Schlussfolgerungen für die Nutzung und Erstellung neuer Datensätze diskutiert.

Datensatzqualität. Die stetig steigende Anzahl verschiedener Angriffsszenarien vereint mit der zunehmenden Anzahl von Anwendungen und komplexeren Softwareprodukten führt dazu, dass Datensätze aktuellen und realen Netzwerkverkehr beinhalten sollten. Da es kein IDS gibt, welches alle Datenpunkte korrekt klassifiziert, kann das ausschließliche Labeln von Netzwerkdaten durch IDS zu Fehlern führen und erfordert eine manuelle Überprüfung. Aus diesen Gründen sollte ein Evaluierungsdatensatz aktuell und korrekt gelabelt sein, einen großen Zeitraum von möglichst nicht anonymisierten Daten umfassen und realistisches Normalverhalten sowie alle Arten von Angriffen beinhalten. Ein derartiger Datensatz existiert jedoch nicht und wird voraussichtlich niemals veröffentlicht. Falls datenschutzrechtliche Aspekte ausgeblendet werden könnten und echter Netzwerkverkehr über einen großen Zeitraum (in paketbasiertem Format) mit allen Arten von Angriffsszenarien aufgenommen werden könnte, wäre exaktes Labelling sehr zeitaufwendig. Folglich würde der Labelling Prozess dazu führen, dass der Datensatz schon wieder etwas veraltet ist, da täglich neue Angriffe aufkommen (siehe Abschnitt 1.1). Aus diesen Gründen sind bei einigen Bewertungskriterien stets Abstriche zu machen. Diese Tatsache stellt jedoch keine gravierende Einschränkung dar, da nur in den wenigsten Fällen alle Bewertungskriterien erfüllt werden müssen. Beispielsweise werden nicht alle Arten von Angriffsszenarien benötigt, wenn eine neue Methode zur Erkennung von Port Scans evaluiert werden soll, oder es wird keine komplette Netzwerkkonfiguration benötigt, wenn nur die Sicherheit eines Web-Servers

evaluiert werden soll. Deshalb ist es wichtig, Datensätze zu verwenden, die alle Anforderungen der zugrundeliegenden Aufgabenstellung erfüllen.

Mehrerer Datensätze. Wie erwähnt, gehen Datensätze mit unterschiedlichen Einschränkungen einher. Dieses Kapitel zeigt jedoch, dass eine ganze Reihe von netzwerkbasierten Datensätzen und Netzwerkdaten-Generatoren existieren. Um Schwachstellen und Artefakte einiger Datensätze zu reduzieren und die Abdeckung neuer Intrusion Detection Methoden zu erhöhen, empfiehlt sich die Nutzung mehrerer Datensätze zur Evaluierung. Dadurch kann Overfitting auf bestimmte Datensätze und der Einfluss künstlicher Artefakte eines Datensatzes vermieden werden und gleichzeitig die Methoden in einem allgemeineren Kontext evaluiert werden. Eine andere Möglichkeit wäre die Evaluierung mit einem frei verfügbaren Datensatz und die anschließende Übertragung und Evaluierung in Produktivsystemen.

Um die Reproduzierbarkeit der Ergebnisse zu steigern, sollten neu entwickelte Verfahren stets mit mindestens einem frei verfügbaren Datensatz evaluiert werden.

Vordefinierte Teilmengen. Datensätze sollten vordefinierte Teilmengen für Training und Evaluation beinhalten, um die Vergleichbarkeit der erzielten Ergebnisse von Intrusion Detection Methoden zu erhöhen. Dadurch kann sichergestellt werden, dass alle Methoden auf den gleichen Daten trainiert und evaluiert werden.

Die direkte Anwendung von gängigen Evaluierungstechniken aus dem Data Mining (zum Beispiel x-fache Kreuzvalidierung, siehe Abschnitt 2.5.3) wäre kontraproduktiv. Eine Kreuzvalidierung würde zeitliche Zusammenhänge aus flowbasierten Netzwerkdaten verwerfen und dafür sorgen, dass in den erstellten Teilmengen für das Training häufig Flows aus allen Angriffsszenarien enthalten wären. Dies würde dazu führen, dass die Methoden bessere Ergebnisse auf den Evaluierungsteilmengen erzielen und die Generalisierung der Methoden nicht korrekt evaluiert wird. Im Bereich Intrusion Detection wäre die Trennung eines Datensatzes hinsichtlich der Zeit sinnvoll. Beispielsweise könnte ein Datensatz, der zwei Wochen Netzwerkdaten umfasst, so aufgeteilt werden, dass der Netzwerkverkehr aus der ersten Woche zum Training und aus der zweiten Woche zum Evaluieren herangezogen wird.

Engere Zusammenarbeit. In den letzten Jahren wurden viele Datensätze veröffentlicht und weitere Datensätze sind zu erwarten. Deshalb könnte die Gemeinschaft von einer engeren Zusammenarbeit und einer allgemein anerkannten Plattform für den Austausch von IDS Datensätzen ohne Zugangsbeschränkungen profitieren. Beispielsweise arbeiten Cermak et al. [CJV+18] an einer derartigen Plattform und Ring et al. [RWG+17c] veröffentlichten ihre Simulationsskripte für normales Benutzerverhalten.

Veröffentlichung. Weiterhin wird die Veröffentlichung neuer netzwerkbasierter Datensätze empfohlen. Ausschließlich frei verfügbare Datensätze können von Dritten genutzt und bezüglich ihrer Qualität geprüft und bestätigt werden. Des Weiteren wird die Veröffentlichung zusätzlicher Informationen über IP-Adressen und Angriffsszenarien zur besseren Interpretation für Dritte empfohlen.

Standardformate. Für die meisten Intrusion Detection Methoden müssen die Eingabedaten in einem standardisierten Format vorliegen. Deshalb empfiehlt sich die Veröffentlichung von Datensätzen in standardisierten Formaten. Für einige Attribute aus vorverarbeiteten Datensätzen ist es fragwürdig, ob diese in Echtzeit berechnet werden können, was die Anwendbarkeit eines

IDS beeinflussen würde. Gleichzeitig erzielen viele Ansätze [JJC+16, MJ15, TRM09, LXZ+12, NKN+16] auf diesen vorverarbeiteten Datensätzen jedoch sehr gute Ergebnisse, was darauf hindeutet, dass sich die berechneten Attribute der Vorverarbeitung eignen. Deshalb empfiehlt sich die Veröffentlichung der Daten in standardisierten Formaten sowie die Veröffentlichung der entsprechenden Vorverarbeitungsskripte. Dies würde den zusätzlichen Vorteil mit sich bringen, dass die Vorverarbeitungsskripte auch auf andere Datensätzen angewendet werden könnten.

Anonymisierung. Ein weiterer wichtiger Punkt ist die Anonymisierung, da jedes anonymisierte beziehungsweise entfernte Attribut die Analyse der Netzwerkdaten erschwert. Deswegen sollten nur die Attribute anonymisiert und entfernt werden, die aus datenschutzrechtlichen Gründen zwingend zu entfernen sind, da bereits teilweise vorhandene Attribute bei der Detektion sicherheitskritischer Ereignisse positiven Einfluss besitzen können. Beispielsweise berücksichtigte Mahoney [Mah03] die ersten 48 Bytes von jedem Netzwerkpaket (beginnend beim IP-Header) und erreichte sehr gute Ergebnisse. Derartige Attribute könnten beispielsweise mit dem Flow Exporter YAF [IT10] erstellt werden. YAF erlaubt beispielsweise das Extrahieren der ersten n Bytes oder das Berechnen der Entropie über den Paketinhalt (Payload). Es existieren verschiedenste Methoden zum Anonymisieren von Netzwerkdaten, sodass Attribute nicht entfernt werden müssen. Beispielsweise stellen Xu et al. [XFA+02] eine Prefix-erhaltende Methode zur Anonymisierung von *IP-Adressen* vor. Eine weitere Methode stellen Fang und Paxon [PP03] zur Anonymisierung von Payload im paketbasierten Format vor. Das Tool von Pang et al. [PAP+06] erlaubt die Anonymisierung von IP-Adressen und berechnet die Prüfsumme für die anonymisierten IP-Adressen neu. Für weitergehende Informationen über unterschiedliche Techniken zur Anonymisierung sei auf Kelly et al. [KRG+08] verwiesen.

7.5. Fazit

Diese Dissertation strebt die Detektion sicherheitskritischer Ereignisse in Unternehmensnetzwerken mithilfe von Data Mining-Methoden an. Im Mittelpunkt steht dabei die Analyse unidirektionaler flowbasierter Netzwerkdaten. Die Entwicklung neuer Intrusion Detection Methoden erfordert aktuelle, gelabelte und angepasste Datensätze.

Obwohl in diesem Kapitel etliche netzwerkbasierte Datensätze vorgestellt wurden, eignen sich nur wenige Datensätze für die Zielsetzung dieser Arbeit. Vorgestellte Datensätze mit einer dritten Klasse *background* oder durch IDS gelabelte Datensätze eignen sich für die Entwicklung und Evaluierung neuer Intrusion Detection Methoden nur bedingt. Durch IDS gelabelte Datensätze können fehlerhaft gelabete Datenpunkte beinhalten und somit falsche Informationen in der Trainingsphase beisteuern. Mit *background* gelabelte Datenpunkte können Normalverhalten oder Angriffsszenarien darstellen und somit keiner Klasse eindeutig zugeordnet werden. Folglich schränkt sich die Auswahl zur Verfügung stehender Datensätze auf die Zeile *Labels=ja* und die Spalten *Format=Paket* und *Format=Flow* in Tabelle 7.1 ein. Diese Auswahl umfasst 18 Datensätze und muss noch weiter reduziert werden, da einige Datensätze wie DARPA, ISOT oder Twente veraltet sind. Andere Datensätze wie AWID, Botnet oder CIC DoS entfallen aufgrund der speziellen Fokussierung auf bestimmte Angriffsarten. Deshalb schränkt sich die Auswahl zur Verfügung stehender Datensätze erheblich ein. Folglich ist eine flexible Umgebung zur Generierung gelabelter und speziell angepasster Datensätze für die Entwicklung neuer Intrusion Detection Methoden notwendig. Eine derartige Umgebung wird im nachfolgenden Kapitel vorgestellt. Neuere Datensätze wie CICIDS 2017, TRAbID oder NDSec-1 wurden erst nach Veröffentlichung des Simulationsansatzes in Kapitel 8 erstellt.

Netzwerkdaten-Generatoren stellen eine sinnvolle Ergänzung zu klassischen Datensätzen dar, da speziell die in Kategorie V vorgestellten Ansätze zur Anreicherung existierender Datensätze genutzt werden können. Aktuelle Ansätze dieser Kategorie generieren jedoch nur Teile von flowbasierten Netzwerkdaten. Aus diesem Grund wird in Kapitel 9 eine GAN-basierte Methode zur Generierung flowbasierter Netzwerkdaten unter Berücksichtigung aller typischen Attribute eines Flows vorgestellt.

8. Generierung flowbasierter Netzwerkdaten durch Simulation

Dieses Kapitel beschäftigt sich mit der Generierung realistischer Netzwerkdaten. Wie bereits in der Einleitung erörtert, beinhalten Netzwerkdaten personenbezogene Informationen und können häufig aus datenschutzrechtlichen Gründen nicht veröffentlicht werden. In diesem Kapitel wird ein Konzept zur Generierung realistischer flowbasierter Netzwerkdaten durch Simulation vorgestellt, um den Mangel an öffentlich verfügbaren Datensätzen zu reduzieren (siehe Herausforderung 2 in Abschnitt 1.2.2). Das Konzept sieht die Erzeugung von normalen Benutzerverhalten und Angriffsszenarien durch parametrisierbare Skripte vor. Der generierte Netzwerkverkehr wird im NetFlow Format gespeichert und mit exakten Labels versehen. Im Vergleich zu standardmäßig verwendeten Testumgebungen liegt der Schwerpunkt in der Simulation von realistischem Benutzerverhalten. Basierend auf diesem Konzept werden zwei flowbasierte Datensätze CIDDS-001 und CIDDS-002 erstellt.

Die Inhalte des Kapitels wurden bereits in den Publikationen [RWG+17c] und [RWG+17b] veröffentlicht.

8.1. Einleitung

Durch die zunehmende Digitalisierung ist die Erkennung sicherheitskritischer Ereignisse in Unternehmensnetzwerken zu einem unerlässlichen Faktor herangewachsen [SN13, SSF16, Kim14]. Für diese Aufgabe werden häufig anomalie- oder signaturbasierte IDS verwendet [LLL+13, PQW10, AG11]. Dabei müssen IDS stetig weiterentwickelt werden, da täglich neue Bedrohungsszenarien erscheinen [TSS14]. Dieses Kapitel konzentriert sich auf die Generierung von gelabelten flowbasierten Netzwerkdaten, welche für die Entwicklung und Evaluierung netzwerkbasierter IDS erforderlich sind.

Problem. Zur Entwicklung, Konfiguration und Evaluierung netzwerkbasierter IDS werden aktuelle und gelabelte Datensätze benötigt. Jedoch stellt der Mangel an öffentlich verfügbaren und gelabelten Datensätzen eine der größten Herausforderungen in diesem Bereich dar [SP10, HHS+17, MBM15]. Die Verwendung von realen Netzwerkverkehr ist ebenfalls problematisch, da einerseits für diesen keine Grundwahrheit (Labels) zur Verfügung steht und andererseits keine Angriffsszenarien in Produktivumgebungen ausgeführt werden können. Kapitel 7 zeigt, dass viele existierende Datensätze wie KDD CUP 99 [Uni99] veraltet sind und somit keine aktuellen Bedrohungssituationen reflektieren. Des Weiteren sind neuere Datensätze wie SSHCure [HHS+14] oder Botnet [BJS+14] häufig auf spezielle Szenarien ausgelegt. Beispielsweise beinhaltet der SSHCure Datensatz [HHS+14] ausschließlich SSH-Verbindungen oder der Botnet Datensatz [BJS+14] umfasst primär durch Botnetze verursachten Netzwerkverkehr. Andere Datensätze wie IRSC [ZKS+15] oder SANTA [WKZ+14] sind nicht öffentlich verfügbar oder nur in stark anonymisierter Form wie der Unified Host and Network Datensatz [TKH18].

Ziel. Ziel dieses Kapitels ist der Entwurf eines flexiblen Konzepts zur Erstellung realistischer und gelabelter Datensätze. Die Kernidee ist der Nachbau von Unternehmensnetzwerken in ei-

ner leicht anpassbaren Testumgebung. Im klassischen Sinne sind Testumgebungen isoliert und dienen zum Evaluieren neuer Software, Netzwerkkonfigurationen oder Updates [SL05, BM15]. Das vorgestellte Konzept sieht eine Erweiterung dahingehend vor, dass auch typische Benutzeraktivitäten wie Senden und Empfangen von E-Mails, Surfen im Internet oder Drucken von Dokumenten stattfinden und Angriffsszenarien parallel durchgeführt werden können. Das Konzept soll sich zur kontinuierlichen Generierung von Datensätzen eignen, die auf notwendige Anwendungsszenarien angepasst werden können.

Ansatz und Beiträge. Primäre Herausforderung ist die Diskrepanz der Notwendigkeit von normalen Benutzerverhalten in Datensätzen und der Tatsache, dass Testumgebungen kein normales Benutzerverhalten beinhalten. An diesem Punkt setzt das vorgestellte Konzept an und strebt die realistische Simulation von normalen Benutzerverhalten durch die Ausführung von konfigurierbaren Skripten an. Um möglichst realistischen Netzwerkverkehr zu generieren, werden mehrere Rahmenbedingungen definiert, die von den Skripten eingehalten werden müssen. Im Gegensatz zu Produktivumgebungen ermöglichen Testumgebungen die Ausführung von Malware und anderer Angriffsszenarien. Angriffsszenarien werden wie normale Benutzeraktionen von Skripten ausgeführt und über Konfigurationsdateien gesteuert. Da alle Aktionen automatisiert durch Skripte mit Log-Mechanismen ausgeführt werden, ist ein exaktes Labelling des aufgenommenen Netzwerkverkehrs möglich. Des Weiteren sieht das Konzept den Einsatz externer Server vor, welche realen und aktuellen Bedrohungen aus dem Internet ausgesetzt sind. Schließlich wird der vorgestellte Ansatz zur Erstellung der flowbasierten Datensätzen CIDDS-001 [RWG+17c] und CIDDS-002 [RWG+17b] verwendet. In beiden Datensätzen wird ein kleines Unternehmen mit mehreren Servern und Clients simuliert und diverse Angriffsszenarien ausgeführt. Der entstehende Netzwerkverkehr wird im unidirektionalen NetFlow Format aufgenommen, gelabelt und öffentlich zur Verfügung gestellt.

Das Kapitel beinhaltet zwei Hauptbeiträge. Zunächst wird in diesem Kapitel ein Konzept zur Generierung realistischer Netzwerkdaten in einer Testumgebung vorgestellt. Danach wird dieses Konzept zur Generierung von gelabelten flowbasierten Datensätzen verwendet.

8.2. Konzept zur Datengenerierung

In diesem Abschnitt wird das Konzept zur Datengenerierung detailliert beschrieben. Zunächst werden die Anforderungen erläutert und ein genereller Überblick gegeben. Danach wird die Generierung von normalen Benutzerverhalten und Angriffsszenarien behandelt. Zuletzt werden die Themen Monitoring, Labeln der Daten und Anonymisierung diskutiert.

8.2.1. Anforderungen

Abschnitt 7.1 definiert Bewertungskriterien für netzwerkbasierte Datensätze. Im Folgenden sollen die Anforderungen des Konzepts anhand dieser Bewertungskriterien definiert werden.

Eine zentrale Anforderung netzwerkbasierter Datensätze ist, dass diese aktuell und auf spezifischen Gegebenheiten angepasst sein sollen. Folglich muss das Konzept die kontinuierliche Generierung von Daten ermöglichen, verschiedene Netzwerkstrukturen (zum Beispiel KMU oder Universitätsnetzwerk) abbilden können und die Durchführung unterschiedlicher Angriffsszenarien gestatten. Somit behandelt die erste Anforderung die Bewertungskriterien Jahr der Erstellung, Angriffsszenarien, Aufnahmedauer, Volumen, komplette Netzwerkkonfiguration sowie die Art des Netzwerkverkehrs aus Abschnitt 7.1.

Die zweite Anforderung zielt auf die Integration von realistischen Benutzeraktivitäten. Viele anomaliebasierte Systeme erlernen Normalverhalten und heben Abweichungen von diesem als Anomalie hervor. Folglich muss ein Datensatz möglichst realistisches Benutzerverhalten integrieren, was dem Bewertungskriterium Normalverhalten aus Abschnitt 7.1 entspricht.

Die dritte Anforderung ist das exakte Labeln des Netzwerkverkehrs und das Bereitstellen zusätzlicher Informationen (Bewertungskriterien Labels und Metadaten). Flowbasierte Netzwerkdaten beinhalten wenig Informationen und sind auch für IT-Sicherheitsexperten schwer zu interpretieren. Deshalb sollen Zusatzinformationen über Angriffsszenarien und IP-Adressen bereitgestellt werden, um auch Dritten inhaltlich sinnvolle Schlussfolgerungen zu ermöglichen.

Die vierte Anforderung ist die Möglichkeit zur Integration von realem Netzwerkverkehr, dessen Ursprung nicht in der Simulation oder in der Ausführung von Penetrationtests liegt.

Die fünfte Anforderung zielt auf das Veröffentlichen der Datensätze und der zugrundeliegenden Skripte, um eine möglichst große Verbreitung und Nachvollziehbarkeit zu erlauben. Zur Wahrung datenschutzrechtlicher Aspekte werden öffentliche IP-Adressen unkenntlich gemacht. Diese Anforderung schränkt die Datensätze in den Bewertungskriterien Format und Anonymität ein.

8.2.2. Überblick

Das Konzept baut auf einer Testumgebung auf, die in der Softwareplattform OpenStack realisiert wird. OpenStack ermöglicht das Erstellen kompletter Testumgebungen durch die Erstellung virtueller Testnetzwerke, virtueller Netzwerkgeräte und virtueller Maschinen. In Otto et al. [ORL+16] wurde die Aufnahme von Netzwerkverkehr in einer OpenStack-Umgebung bereits erfolgreich getestet. Die Verwendung einer virtuellen Umgebung als Testumgebung bringt einige Vorteile mit sich. Einer ist die leichte und einfache Kontrolle über das komplette Netzwerk. Beispielsweise können Firewall-Regeln speziell für spezifische Testszenarien definiert werden. Des Weiteren erlaubt die virtuelle Testumgebung den einfachen Aufbau unterschiedlicher Netzwerken zur Simulation unterschiedlicher Organisationen. Ein weiterer Vorteil ist, dass in einer virtuellen Testumgebung dauerhaft neue Datensätze generiert und die Skripte zur Simulation von Benutzeraktivitäten einfach aktualisiert werden können. Auf diese Weise können kontinuierlich neue Angriffsszenarien und neue Benutzeraktivitäten integriert werden (Anforderung 1).

Zur Simulation von normalem Benutzerverhalten führen parametrisierte Python Skripte typische Benutzeraktivitäten auf den Clients aus (Anforderung 2). Die Skripte berücksichtigen einige Rahmenbedingungen, um die Benutzeraktivitäten möglichst realistisch nachzuahmen (siehe nächster Abschnitt). Zur Generierung von Angriffsdaten ist die Ausführung diverser Angriffsszenarien vorgesehen (siehe Abschnitt 8.2.4). Da normales Benutzerverhalten und Angriffsszenarien durch automatisierte Skripte ausgeführt werden, ist ein exaktes Labeln des Netzwerkverkehrs möglich (Anforderung 3). Ein Vorteil des Konzepts im Vergleich zu Netzwerkdaten-Generatoren ist, dass die Simulation bereits alle Parameter berücksichtigen, die das zeitliche Verhalten beeinflussen könnten. Hier sei beispielsweise die Antwortzeit eines Servers, die aktuelle CPU-Auslastung eines Clients oder Überlastungen der Netzwerkverbindungen genannt.

Das Konzept sieht die Möglichkeit zur Integration von realen Netzwerkverkehr aus dem Internet vor (Anforderung 4). Hierfür werden externe Server eingesetzt, die aktuellen Angriffen aus dem Internet ausgesetzt sind. Im Vergleich zu Honeypots werden diese Server jedoch regulär von Clients aus der Testumgebung genutzt. Folglich kann an diesen Servern sowohl normales Benutzerverhalten als auch unterschiedliche Angriffsszenarien beobachtet werden.

Des Weiteren sieht das Konzept die Aufnahme des Netzwerkverkehrs an den virtuellen Netzwerkgeräten im standardisierten NetFlow Format vor. Da alle durchgeführten Benutzeraktivi-

täten auf den Clients durch Skripte ausgeführt und mitprotokolliert werden, können die aufgenommenen Daten veröffentlicht und mit Zusatzinformationen angereicht werden (Anforderung 5). Grundsätzlich ermöglicht diese Vorgehensweise auch das Mitschneiden des Netzwerkverkehrs in anderen flowbasierten Formaten oder im paketbasierten Format.

8.2.3. Generierung von normalen Benutzerverhalten

In diesem Konzept übernehmen Skripte die Ausführung von typischen Benutzeraktivitäten auf den Clients. Hierfür muss unter anderem die Heterogenität der Betriebssysteme sichergestellt werden. Durch die Verwendung der plattformunabhängigen Sprache Python wird diese Voraussetzung erfüllt, da Python die Ausführung der Simulationsskripte auf unterschiedlichen Betriebssystemen wie Windows oder Linux ermöglicht.

Zur Simulation von realistischem Benutzerverhalten müssen die Skripte mehrere Rahmenbedingungen beachten, die im Folgenden diskutiert werden. Mitarbeiter führen im Arbeitsalltag vielfältigste Aktionen wie Schreiben von E-Mails, Surfen im Internet, Drucken von Dokumenten, Kopieren von Dateien oder Kaffeepausen aus. Diese Heterogenität von Benutzeraktivitäten muss sich in den Skripten widerspiegeln. Des Weiteren ist die Tatsache zu berücksichtigen, dass verschiedene Mitarbeiter mit unterschiedlicher Geschwindigkeit arbeiten und mannigfaltige Aufgaben besitzen, sodass sich deren Aktivitäten unterscheiden. Hierfür sieht das vorgestellte Konzept die Verwendung von Konfigurationsdateien vor, in denen die Frequenz und die Benutzeraktivitäten der simulierten Clients gesteuert werden können.

In Abbildung 8.1 ist der Ausschnitt einer Konfigurationsdatei eines Clients zu sehen. Die Konfigurationsdatei ist in mehrere Abschnitte gegliedert. Der Abschnitt *actions* definiert die auszuführenden Benutzeraktivitäten. Je höher der Wert der zugewiesenen Zahl, desto größer ist die Wahrscheinlichkeit, dass diese Aktion ausgeführt wird. Ein Wert von 0 (siehe *attacking* in Abbildung 8.1) würde indizieren, dass dieser Client keine Angriffe ausführt. Die individuelle Konfigurationsdatei pro Client ermöglicht die Zuweisung unterschiedlicher Nutzungsprofile. Des Weiteren können die Arbeitszeiten eines Clients in den Abschnitten *workdays* und *workinghours* definiert werden. Arbeitstage werden durch 1 markiert und arbeitsfreie Tage durch 0. Die in *workinghours* definierten Arbeitszeiten gelten für alle Arbeitstage und werden mit einer täglichen Abweichung von maximal 30 Minuten eingehalten.

```
1  [actions]
2  browsing = 30
3  mailing = 10
4  printing = 10
5  copyfiles = 10
6  copysea = 6
7  ssh = 4
8  meeting = 3
9  offline = 7
10 private = 10
11 breaks = 10
12 attacking = 0
13
14 [workdays]
15 monday = 1
16 tuesday = 0
17 wednesday = 1
18 thursday = 1
19 friday = 1
20 saturday = 0
21 sunday = 0
22
23 [workinghours]
24 clock_in = 10
25 clock_out = 17
```

Abbildung 8.1.: Ausschnitt aus einer Konfigurationsdatei.

Die Benutzeraktivitäten selbst müssen ebenfalls variieren. Beim Kopieren von Dateien ist zu berücksichtigen, dass die Anzahl und Größe der zu kopierenden Dateien variiert. Für das Versenden von E-Mails muss ebenfalls die Anzahl der Anhänge variieren. Um Klickverhalten beim

Browsen möglichst realistisch nachzuahmen, sollte die Zeit, bis die nächste Seite aufgerufen wird, durch zufällige Zeitabstände definiert werden. Zur Erfüllung dieser Anforderungen werden die Skripte zufallsgesteuert und nicht periodisch aufgerufen. Allerdings finden diese Aufrufe nicht komplett zufallsgesteuert statt und orientieren sich an den typischen Arbeitszeiten. Deswegen definiert die Konfigurationsdatei auch Arbeitstage und Arbeitszeiten für jeden Client. Typischerweise führen Angestellte nicht dauerhaft Aktivitäten aus, die Netzwerkverkehr erzeugen. Um dies zu berücksichtigen, sieht das Konzept die Durchführung von Besprechungen, Offline-Arbeit oder Kaffeepausen vor. Grundsätzlich führen die Skripte nur Aktivitäten während der Arbeitszeit aus und beenden diese in den Pausen, am Abend und am Wochenende.

Da dieses Kapitel lediglich die Generierung von realistischen Netzwerkverkehr beabsichtigt, wird auf die Simulation von Aktivitäten verzichtet, die keinen Netzwerkverkehr erzeugen. Hier sei beispielsweise das Erstellen von PowerPoint Präsentationen genannt. Des Weiteren erhebt diese Arbeit keinen Anspruch auf Vollständigkeit bei der Erzeugung von Netzwerkverkehr. Vielmehr wurden die Skripte zur Simulation von Benutzeraktivitäten so konzipiert, dass diese leicht für weitere Benutzeraktivitäten erweitert werden können, jedoch bereits typische Aktivitäten wie Browsen realisieren.

8.2.4. Generierung von Angriffsszenarien

Die Präsenz von Angriffsszenarien ist für Datensätze zur Evaluierung von IDS unerlässlich. Im vorgestellten Konzept sind drei unterschiedliche Methoden zur Generierung von Angriffsdaten vorgesehen. Bei der ersten Methode übernimmt ein Benutzer die Kontrolle einer virtuellen Maschine aus der Testumgebung und führt die Angriffe manuell aus.

Die zweite Methode verwendet die Angriffsskripte aus dem CIDDS Repositorium [Rin17]. Konfigurationsdateien steuern dabei Frequenz und Typ der Angriffe. Vorteil dieser Methode ist, dass die Angriffsskripte alle Aktivitäten automatisch protokollieren und zum Labeln verwendet werden können. Aktuell befinden sich Skripte zum Ausführen von Port Scans, Ping Scans, SSH-Brute-Force und DoS-Angriffen im Repositorium. Die Angriffsskripte aus dem CIDDS Repositorium erheben keinen Anspruch auf Vollständigkeit. Hier wurde ebenfalls das Augenmerk auf die leichte Integration weiterer Angriffsarten gelegt und typische Angriffsszenarien realisiert.

Die dritte Methode zur Generierung von Angriffsdaten ist die Verwendung externer Server. Diese sind direkt am Internet angebunden und somit aktuellen Angriffen ausgesetzt. Jedoch müssen diese Server auch Dienste anbieten, die mindestens von Clients der Testumgebung normal genutzt werden. Andernfalls würden diese Server ausschließlich verdächtigen Netzwerkverkehr aufnehmen und wären Honeypots gleichzusetzen. Externe Server wurden beispielsweise bei der Erstellung des CIDDS-001 Datensatzes [RWG+17c] verwendet.

8.2.5. Überwachung

Bei der Simulation von Benutzeraktivitäten durch Skripte können zur Laufzeit diverse Fehler auftreten. Beispielsweise führte die Ansteuerung des Webbrowsers Firefox durch Selenium unter Windows zu einigen Softwareabstürzen. Diese Softwareabstürze führten dazu, dass keine weiteren Benutzeraktivitäten ausgeführt wurden. In anderen Fällen schlug das automatische Starten des Hauptskripts zur Simulation nach einem Neustart des Betriebssystems fehl. Deshalb wurde im Rahmen des Konzepts ein Überwachungssystem (siehe Abbildung 8.2) entwickelt, welches die Überwachung der Clients ermöglicht.

Der linke Bereich in Abbildung 8.2 zeigt generelle Informationen über die Konfiguration der Clients an und beinhaltet zum Beispiel deren Arbeitstage und typische Arbeitszeiten. Der rechte

Client	Arbeitsstunden	Arbeitstage							Browsen	E-Mail	...	Pausen	Summe
		M	D	M	D	F	S	S					
dev-deb-1 (220.2)	09:00-20:00	x	x	x	x	x	x	-	19% 2017-08-08 10:49:43	1%	...	5%	80.56h
dev-deb-2 (220.3)	08:30-16:30	x	x	x	x	x	-	-	23%	2%	...	15%	62.44h
dev-deb-3 (220.4)	09:00-15:00	-	x	x	x	x	-	x	23%	5%	...	18%	53.21h
dev-deb-4 (220.5)	07:30-16:30	x	x	x	-	x	-	-	23%	3%	...	9%	67.39h
dev-deb-5 (220.6)	07:00-16:00	x	x	x	x	x	-	-	23%	0,5%	...	15%	71.21h
off-deb-1 (210.4)	09:30-17:00	-	x	x	x	x	-	-	7%	11% 2017-08-08 18:39:11	...	33%	61.83h
off-deb-2 (210.5)	08:00-16:00	x	-	x	x	x	-	-	12% 2017-08-08 17:11:26	5%	...	14%	66.25h
off-deb-3 (210.6)	06:00-14:00	x	x	-	x	x	-	-	18%	9%	...	11%	56.99h

Abbildung 8.2.: Überwachungssystem der Simulationsumgebung.

Bereich in Abbildung 8.2 zeigt Informationen über den aktuellen Status der Benutzeraktivitäten, beispielsweise wie viel Prozent der Arbeitszeit ein Client mit spezifischen Aktivitäten wie Browsen oder E-Mail beschäftigt war.

Wenn ein Client während seiner Arbeitszeit über einen vordefinierten Zeitraum keine Aktivität ausführt, hebt das Überwachungssystem die gesamte Zeile mit roter Hintergrundfarbe hervor (siehe Client *dev-deb-2* in Abbildung 8.2). Schlägt eine einzelne Aktivität fehl, wird die betroffene Zelle mit gelber Hintergrundfarbe hervorgehoben (siehe Aktivität Browsen für Client *dev-deb-1* in Abbildung 8.2). Eine erfolgreich ausgeführte Aktivität, nachdem sie bei einem vorherigen Versuch fehlgeschlagen ist, bewirkt, dass der Hintergrund der Zelle grau eingefärbt wird (siehe Aktivität E-Mail für Client *off-deb-1* in Abbildung 8.2). Das Überwachungssystem extrahiert die hierfür erforderlichen Informationen aus den mitprotokollierten Logdateien der Clients. Durch diese Übersicht kann der Administrator während der Aufnahme eines Datensatzes schnell auf Fehlzustände der Clients aufmerksam gemacht werden und entsprechende Maßnahmen einleiten.

8.2.6. Labelling

Das Labeln der aufgenommenen Flows ist ein unverzichtbarer Schritt bei der Erstellung von Datensätzen. Alle Clients protokollieren ihre Aktivitäten in einem standardisierten Format. Basierend auf diesen protokollierten Daten werden vier unterschiedliche Labels pro Flow erzeugt. Eine Übersicht der Labelattribute ist in Tabelle 8.1 zu sehen.

Das erste Labelattribut class kann die Werte *normal*, *attacker*, *victim*, *suspicious* und *unknown* annehmen. Flows, die von Skripten für normales Benutzerverhalten erzeugt wurden, bekommen den Wert *normal* zugewiesen. Flows, die von Skripten zur Generierung von Angriffsszenarien erzeugt wurden, bekommen den Wert *attacker* oder *victim* zugewiesen. Diese Unterteilung soll dazu dienen, den Angreifer und die Opfer eines Angriffsszenarios eindeutig

Tabelle 8.1.: Übersicht der Labelattribute.

#	**Name**	**Beschreibung**
1	class	Klassenlabel des Flows (*normal*, *attacker*, *victim*, *suspicious* oder *unknown*)
2	attackType	Angriffstyp (*PortScan*, *PingScan*, *DoS*, *SSH-BruteForce* oder - - - für normalen Netzwerkverkehr)
3	attackID	Eindeutige ID eines Angriffsszenarios
4	attackDescription	Detaillierte Beschreibung des Angriffsszenarios

identifizieren zu können. Falls die IP-Adresse des Angreifers der Quell-IP-Adresse des Flows entspricht, wird der Flow der Klasse *attacker* zugeordnet. Falls die IP-Adresse des Angreifers der Ziel-IP-Adresse des Flows entspricht, handelt es sich um eine Antwort des Opfers und der Flow wird der Klasse *victim* zugeordnet. Flows, denen kein eindeutig gutartiges bzw. bösartiges Verhalten zugeordnet werden kann, bekommen die Klassen *suspicious* oder *unknown* zugewiesen. Diese Einteilung wird im Folgenden genauer erläutert.

Im Allgemeinen unterscheidet sich das Labeln innerhalb und außerhalb der OpenStack-Umgebung. Der in der OpenStack-Umgebung aufgenommene Netzwerkverkehr ist einfach zu labeln, da Zeitstempel, Quell-IP-Adressen und Ziel-IP-Adressen der ausgeführten Angriffe bekannt sind. Deshalb kann dieser Netzwerkverkehr mit den entsprechenden Werten *attacker* und *victim* gelabelt werden. Der verbleibende Netzwerkverkehr wird als *normal* gelabelt.

Das Labeln außerhalb der OpenStack-Umgebung, also an externen Servern, ist aufwendiger. Alle Clients aus der Simulationsumgebung greifen mit der gleichen IP-Adresse auf externe Server zu. Diese Daten können mit *normal* gelabelt werden, da keine Angriffe aus der OpenStack-Umgebung auf externe Server ausgeführt werden. Des Weiteren kann der Netzwerkverkehr zu den IP-Adressen der Attacker Rechner mit *victim* beziehungsweise *attacker* gelabelt werden, da von diesen IP-Adressen ausschließlich Angriffe durchgeführt werden. Für den verbleibenden Netzwerkverkehr werden die folgenden zwei Regeln angewendet. (1) Netzwerkverkehr zu angebotenen Diensten des externen Servers wird mit *unknown* gelabelt, da nicht eindeutig gesagt werden kann, ob diese Flows Normalverhalten oder Angriffsverhalten repräsentieren. (2) Alle anderen Anfragen zu nicht angebotenen Diensten des externen Servers werden als *suspicious* gekennzeichnet, da diese nicht für öffentliche Benutzer verfügbar sind und eine normale Nutzung unwahrscheinlich erscheint.

Das zweite Labelattribut heißt attackType und beinhaltet den konkreten Angriffstyp. Mögliche Ausprägungen sind beispielsweise *DoS*, *portScan* oder - - - für normalen Netzwerkverkehr. In der Regel verursacht ein Angriffsszenario mehr als einen Flow auf Netzwerkebene. Damit alle Flows eines Angriffs zusammengefasst werden können und eine Unterscheidung verschiedener Angriffe möglich ist, weist das dritte Label attackID allen Flows, die durch denselben Angriff entstanden sind, die gleiche ID zu. Das vierte Labelattribut attackDescription beinhaltet genauere Informationen über das jeweilige Angriffsszenario. Beispielsweise sind hier die Parameterkonfigurationen von Nmap bei Ausführung eines Port Scans zu finden. Dieser detaillierte Labelling Prozess soll genauere Ergebnisanalysen ermöglichen. Zusätzlich sieht das Konzept die Veröffentlichung der mitprotokollierten Logdateien der Clients vor.

8.2.7. Anonymisierung

Die DSVGO Artikel 4 Absatz 1 definiert IP-Adressen als personenbezogene Daten. Deshalb werden alle öffentlichen IP-Adressen unkenntlich gemacht. Eine Anonymisierung muss gewährleisten, dass die transformierten IP-Adressen nicht mehr auf ihre ursprünglichen Werte zurückgerechnet werden können. Eine häufig genutzte Möglichkeit zur Anonymisierung ist die Anwendung von Hashfunktionen auf IP-Adressen. Da die Menge der IP-Adressen endlich ist, müssen diese mit einer zufälligen Zeichenfolge (sogenannten Salt) erweitert werden, damit diese nicht zurückgerechnet werden können. Wenn der Salt gespeichert wird und bestimmten Personen zugänglich ist, spricht die DSVGO von einer Pseudoanonymisierung. Ein Nachteil dieser Vorgehensweise ist, dass Zusammenhänge zwischen IP-Adressen (beispielsweise gleiche Subnetze) verloren gehen.

Deshalb wird im Rahmen dieser Arbeit eine leicht abgeänderte Variante dieser Vorgehensweise angewendet. Zunächst wird allen öffentlichen IPv4-Adressen eine 24 Bit Subnetzmaske zugrunde gelegt. Folglich bilden die ersten drei Bytes der IP-Adresse den Netzanteil und das letzte Byte bildet den Hostanteil. Das angewendete Verfahren transformiert den Netzanteil in eine zufällige Ganzzahl. IP-Adressen mit gleichen Netzanteil werden stets auf die gleiche Ganzzahl transformiert. Der Hostanteil wird unverändert übernommen. Diese Methode erhält Informationen über Subnetze und ist ausreichend für die Simulationsumgebung, da die Netzwerkdaten von automatisierten Skripten und nicht von realen Personen erzeugt werden. Eine Anwendung dieser Methode auf personenbezogene Daten müsste noch mit dem Datenschutz abgeklärt werden. Folgende IP-Adressen werden gesondert behandelt: Die IP-Adresse des externen DNS-Servers wird mit *DNS*, die IP-Adresse des externen Servers mit *EXT_SERVER*, die öffentliche IP-Addresse der Testumgebung mit *OPENSTACK_NET* und die IP-Adressen der öffentlichen Angreifer mit *ATTACKER1*, *ATTACKER2* und *ATTACKER3* ersetzt. Tabelle 8.2 zeigt einige Beispiele dieser Transformation.

Tabelle 8.2.: Exemplarische Anonymisierung von IP-Adressen.

ursprüngliche IP-Adresse		anonymisierte IP-Adresse
28.102.3.251	→	4711_251
28.102.3.23	→	4711_23
156.204.34.23	→	2342_23
201.13.175.87	→	9721_87
201.13.175.88	→	9721_88
201.13.176.87	→	3721_87
192.168.100.4	→	192.168.100.4
192.168.200.8	→	192.168.200.8
192.168.220.15	→	192.168.220.15

Für interne IP-Adressen findet eine unveränderte Übernahme statt. Alle anderen Attribute aus flowbasierten Netzwerkdaten werden keiner weiteren Transformation unterzogen und fließen unverändert in den Datensatz ein.

8.3. CIDDS-001 Datensatz

8.3.1. Aufbau der Testumgebumg

Dieser Abschnitt beschreibt die Testumgebung und den aufgenommenen Netzwerkverkehr des CIDDS-001 Datensatzes. Für den CIDDS-001 Datensatz wurde ein kleines Unternehmen mit

mehreren Servern und Clients nachgebaut. Abbildung 8.3 zeigt einen Überblick der simulierten Umgebung.

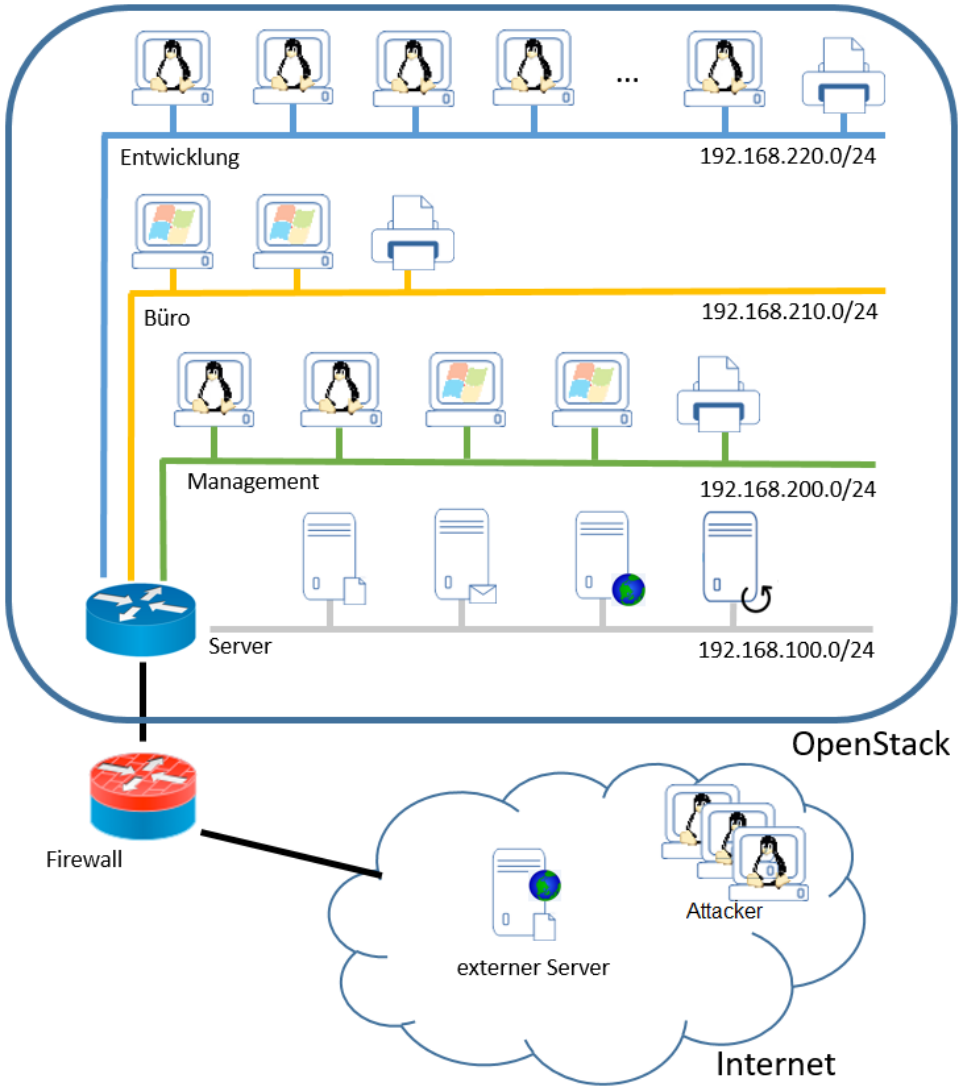


Abbildung 8.3.: Testumgebung des CIDDS-001 Datensatzes [RWG+17c].

Der Router der OpenStack-Umgebung verbindet die vier internen Subnetze miteinander und leitet die ein- und ausgehenden Netzwerkpakete an die Firewall weiter. Die vier Subnetze spiegeln die interne Struktur des simulierten Unternehmensnetzwerks wider. Es existieren drei Subnetze für Clients und ein Server-Subnetz. Das Server-Subnetz (192.168.100.0/24) beinhaltet ein Netzlaufwerk, einen E-Mail-Server, einen Web-Server und einen Backup-Server. Die drei Client-Subnetze reflektieren die Unternehmensstruktur der Abteilungen, da die Entwicklungs-, Büro- und Managementabteilung jeweils ein eigenes Subnetz besitzen. Des Weiteren befindet sich ein externer Server im Internet. Dieser Server bietet eine Webseite und einen Dateisynchronisationsservice für die Clients der OpenStack-Umgebung an. Beim Dateisynchronisationsservice handelt es sich um die Software Seafile. An den drei Attacker Rechnern im Internet wurde kein Netzwerkverkehr aufgenommen.

8.3.2. Analyse des generierten Netzwerkverkehrs

Im oben beschriebenen Szenario wurde der Netzwerkverkehr für einen Zeitraum von vier Wochen (im März/April 2017) aufgenommen. Tabelle 8.3 zeigt die Anzahl der Flows und durchgeführten Angriffe im CIDDS-001 Datensatz. Dieser besteht aus zwei Teilen, einem internen OpenStack-Teil und einem am externen Server aufgezeichneten Teil. Wie Tabelle 8.3 entnommen werden kann, wurde der größte Teil des Netzwerkverkehrs innerhalb der OpenStack-Umgebung aufgenommen, in der ungefähr 31 Millionen Flows mitgeschnitten wurden. Insgesamt wurden explizit 92 Angriffen ausgeführt, davon entfallen 70 Angriffe auf die OpenStack-Umgebung und 22 Angriffe richteten sich gegen den externen Server. Der Datensatz wurde im unidirektionalen flowbasierten Format NetFlow aufgenommen (siehe Kapitel 4.2.2) und jeder Flow enthält neben den in Tabelle 4.1 dargestellten Attributen die vier Labelattribute class, attackType, attackID und attackDescription.

Tabelle 8.3.: Überblick des CIDDS-001 Datensatzes.

	OpenStack	Externer Server	Summe
Anzahl Flows	31.287.934	671.241	31.959.175
Anzahl durchgeführter Angriffe	70	22	92

8.3.2.1. OpenStack-Teil

Tabelle 8.4 zeigt die Verteilung der Klassenlabels. Die erste Spalte gibt das Klassenlabel und die Spalten zwei und drei geben die Anzahl der Flows sowie deren prozentualen Anteil an. 90 Prozent der aufgezeichneten Flows stellen Normalverhalten dar. Da für alle durchgeführten Angriffsszenarien Zeitstempel und IP-Adressen bekannt sind, konnten alle Flows eindeutig einer Klasse zugeordnet werden. Folglich ist die Anzahl der Flows mit den Labels *suspicious* und *unknown* jeweils 0. Flows, die den Klassen *attacker* und *victim* zugeordnet sind, stellen jeweils 5 Prozent des gesamten Netzwerkverkehrs dar.

Tabelle 8.4.: Klassenverteilung der Flows des CIDDS-001 Datensatzes innerhalb der OpenStack-Umgebung.

Klasse	Anzahl Flows	in Prozent
normal	28.051.907	89,66
attacker	1.656.605	05,29
victim	1.579.422	05,05
suspicious	0	00,00
unknown	0	00,00
Summe	31.287.934	100,00

Der zeitliche Verlauf des aufgezeichneten Netzwerkverkehrs in der OpenStack-Umgebung ist in Abbildung 8.4 dargestellt. Jede Linie stellt den Verlauf des Netzwerkverkehrs einer Woche dar. Abbildung 8.4 reflektiert die typischen Arbeitszeiten der Clients. Als Beispiel wird Montag betrachtet. Zunächst liegt die Anzahl Flows pro Stunde von 00:00 Uhr bis ca. 06:00 Uhr auf konstant niedrigen Niveau. Gegen 06:00 Uhr, wenn die ersten Clients das Arbeiten beginnen, steigt die Anzahl Flows pro Stunde bis ca. 11:30 Uhr. Zwischen 11:30 Uhr und 14:00 Uhr machen die meisten Clients Mittagspause, wodurch sich die Anzahl Flows pro Stunde reduziert. Ab 14:00

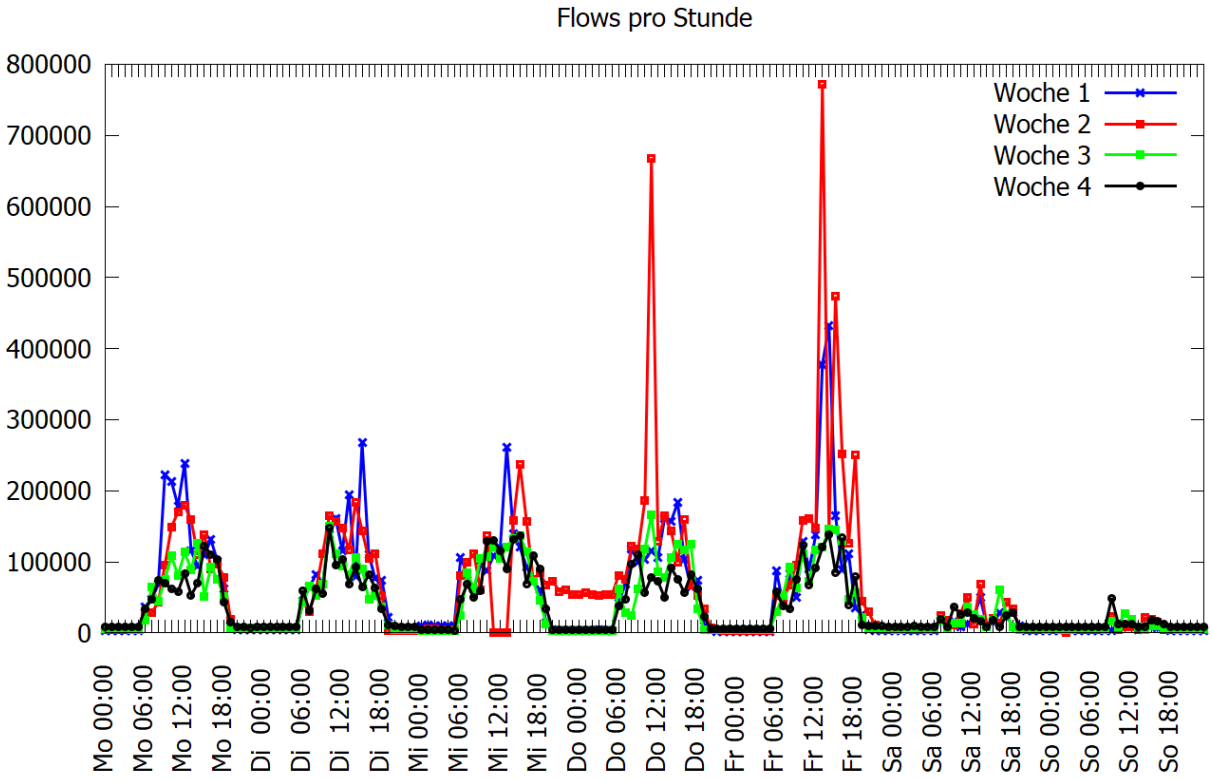


Abbildung 8.4.: Zeitlicher Verlauf der am Router der OpenStack-Umgebung aufgenommenen Flows. Die y-Achse stellt die Anzahl der Flows pro Stunde dar. Jede Linie repräsentiert den Netzwerkverkehr einer Woche.

Uhr ist wiederum ein kleiner Anstieg zu erkennen. Die meisten Clients beenden ihre Arbeitstätigkeit zwischen 16:00 und 18:00 Uhr, wodurch sich die sinkende Anzahl von Flows pro Stunde ab 16:00 Uhr erklären lässt. Ab ca. 19:00 Uhr ist die Anzahl Flows pro Stunde konstant niedrig. Diese werden hauptsächlich durch standardmäßige Aktualisierungsanfragen der Betriebssysteme und Synchronisationsdienste (Netzlaufwerk und Seafile) verursacht. Des Weiteren ist an typischen Arbeitstagen (Montag bis Freitag) eine größere Anzahl von Flows zu beobachten. Am Wochenende arbeiten weniger bis keine Clients, was zu den geringen Ausschlägen führt. Die nächtlich durchgeführten Backups der Server verursachen nur eine sehr geringe Anzahl Flows, die in Abbildung 8.4 nicht zu erkennen ist. Die größeren Ausschläge in Woche 2 am Donnerstag und Freitag wurden von DoS-Angriffen verursacht. Des Weiteren ist ein Abfall von Flows pro Stunde am Mittwoch um 12:00 Uhr in Woche 2 zu erkennen. Dieser ist auf einen Systemfehler in OpenStack zurückzuführen.

8.3.2.2. Externer Server Teil

Der zweite Teil des CIDDS-001 Datensatzes wurde am externen Server aufgezeichnet (siehe Abbildung 8.4). Der externe Server ist direkt am Internet angebunden und somit aktuellen und realen Bedrohungen aus dem Internet ausgesetzt. Als Folge dessen unterliegt die Klassenverteilung des externen Servers einer größeren Varianz, welche in Tabelle 8.5 dargestellt ist.

Die erste Spalte in Tabelle 8.5 beinhaltet das Klassenlabel und die Spalten zwei und drei geben die Anzahl der Flows sowie deren prozentualen Anteil wieder. In den vier Wochen wur-

Tabelle 8.5.: Klassenverteilung der Flows des CIDDS-001 Datensatzes am externen Server.

Klasse	Anzahl	in Prozent
normal	134.240	20,00
attacker	12.260	01,83
victim	8.907	01,32
suspicious	437.911	65,24
unknown	77.923	11,61
Summe	671.241	100,00

den knapp 700.000 Flows aufgezeichnet. Etwa ein Fünftel dieser Flows wurde von Clients aus der OpenStack-Umgebung verursacht. Explizit ausgeführte Angriffsszenarien von den Attacker-Rechnern verursachten weitere 3 Prozent des Netzwerkverkehrs (siehe Klassenlabels *attacker* und *victim*). Da der externe Server einen Web-Server für Kunden bereitstellt, können alle Anfragen und Antworten auf den Ports 80 und 443 entweder normale Kundenanfragen oder Angriffe repräsentieren. Deshalb werden diese Flows mit den Label *unknown* versehen. Der verbleibende Netzwerkverkehr, welcher den größten Anteil mit 65 Prozent darstellt, wird als *suspicious* gelabelt. In der nachfolgenden Tabelle 8.6 findet eine genauere Analyse der als *suspicious* gekennzeichneten Flows statt.

Tabelle 8.6.: Analyse der als *suspicious* gelabelten Flows.

	Anzahl	in Prozent
Flows zu Port 22 (SSH)	337.427	77,05
Flows zu Port 23 (telnet)	38.050	08,89
Flows von Port 80 (HTTP)	634	00,14
Flows zu Port 8000 (Seafile)	1.462	00,33
verbleibende Flows	60.608	13,84
Summe suspicious Flows	437.911	100,00

Diese Analyse führt zu der Erkenntnis, dass ungefähr 77 Prozent der Flows Anfragen auf Port 22 darstellen und somit Loginversuche am Server repräsentieren, die nicht von den Clients der OpenStack ausgeführt wurden. Des Weiteren stellen telnet Anfragen (Port 23) weitere 9 Prozent des Netzwerkverkehrs dar. Zugriffe auf den ausschließlich für OpenStack-Clients angelegten Seafile-Server stellen weitere 1500 Flows dar. Zuletzt konnte noch ein externer Port Scan entdeckt werden, der von Port 80 ausgeführt wurde. Dieser Port Scan umfasst 0,14 Prozent der als *suspicious* gekennzeichneten Flows.

Abbildung 8.5 zeigt den zeitlichen Verlauf der aufgenommenen Flows am externen Server. Dieser unterscheidet sich signifikant vom Netzwerkverkehr der OpenStack-Umgebung. Eine steigende Anzahl Flows pro Stunde ist während typischer Arbeitszeiten nicht zu beobachten, da der größte Teil des Netzwerkverkehrs von verdächtigen Verbindungen von außerhalb verursacht wird. Somit ist der externe Server dauerhaft Angriffen ausgesetzt. Die meisten Ausschläge in Abbildung 8.5 sind durch explizit ausgeführte Angriffe der Attacker-Rechner entstanden.

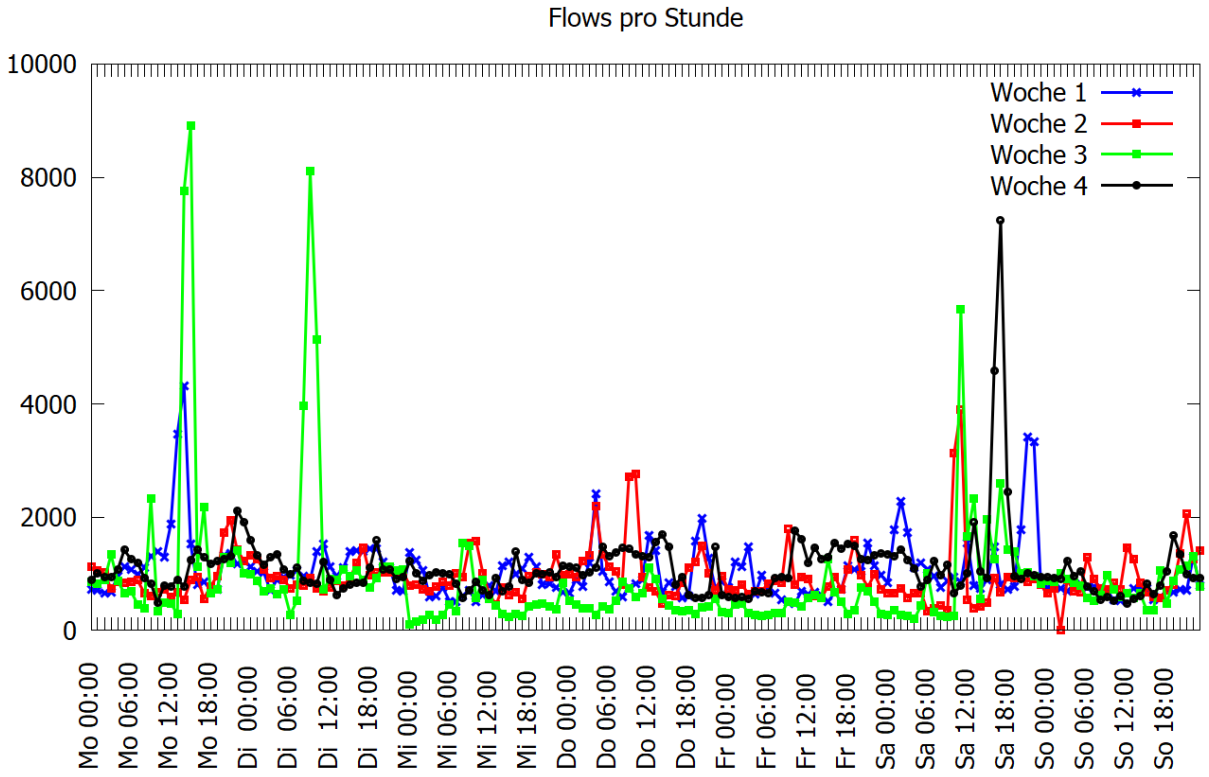


Abbildung 8.5.: Zeitlicher Verlauf der am externen Server aufgenommenen Flows. Die y-Achse stellt die Anzahl der Flows pro Stunde dar. Jede Linie repräsentiert den Netzwerkverkehr einer Woche.

8.4. CIDDS-002 Datensatz

8.4.1. Aufbau der Testumgebung

Für die Erstellung des CIDDS-002 Datensatzes wurde ebenfalls ein kleines Unternehmensnetzwerk simuliert, dessen Architektur in Abbildung 8.6 graphisch dargestellt ist.

Das simulierte Unternehmensnetzwerk enthält vier Server, zwei Drucker und 23 Clients. Wie in Abbildung 8.6 zu sehen, besteht die Netzwerkumgebung aus drei Subnetzen. Das Server-Subnetz (192.168.100.0/24) beinhaltet die internen Server (Netzlaufwerk, E-Mail-Server, Web-Server und Backup-Server). Daneben gibt es noch zwei weitere Subnetze für Clients, das Entwickler-Subnetz (192.168.220.0/24) und das Büro-Subnetz (192.168.210.0/24). Im Entwickler-Subnetz befinden sich ausschließlich Linux-Clients und ein Drucker, während im Büro-Subnetz Windows und Linux-Clients sowie ein Drucker zu finden sind. Im CIDDS-002 Datensatz wurden ausschließlich Port Scan Angriffe durchgeführt und kein zusätzlicher externen Server verwendet.

8.4.2. Analyse des generierten Netzwerkverkehrs

Die in Abbildung 8.6 dargestellte Architektur wurde zur Generierung von Netzwerkverkehr über einen Zeitraum von 2 Wochen (im August 2017) verwendet. Der CIDDS-002 Datensatz besteht aus zwei Wochen, Woche 1 und Woche 2. Die Verteilung der Klassenlabels ist in Tabelle 8.7 zu sehen. Da in der Testumgebung kein externer Server verwendet wurde, konnten alle Flows

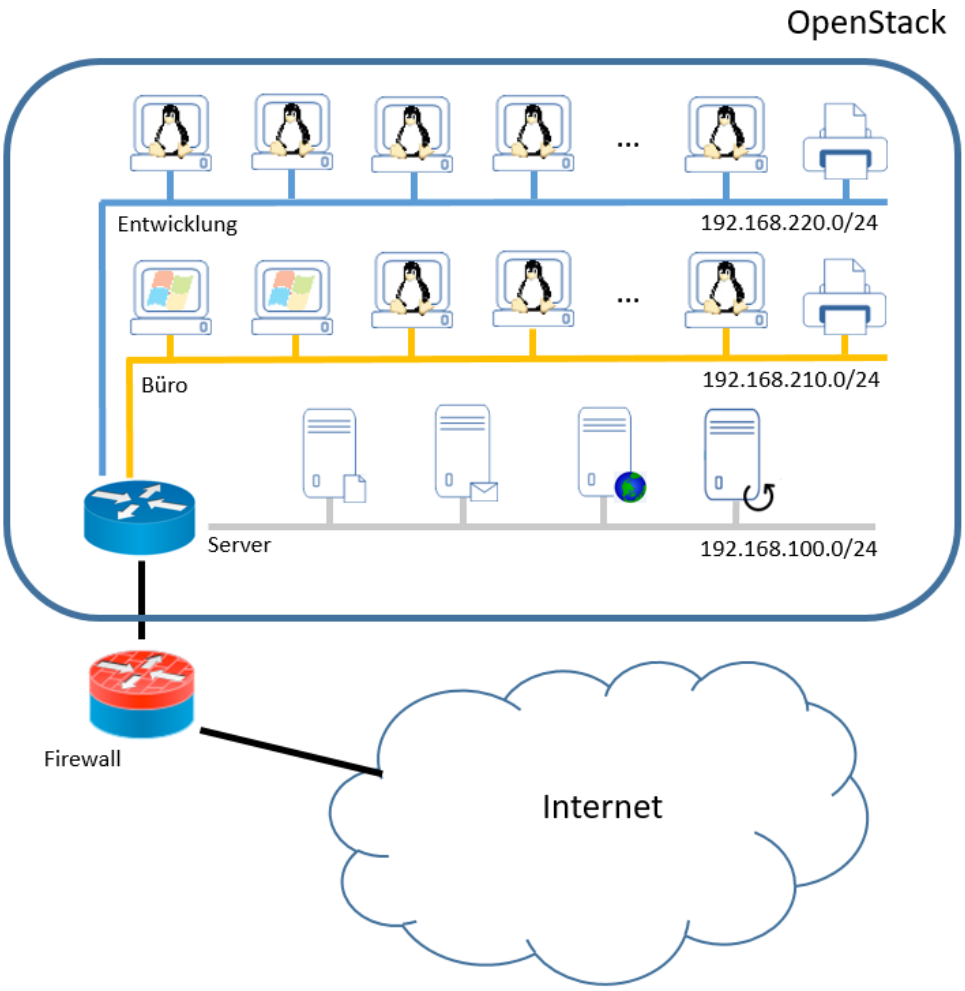


Abbildung 8.6.: Testumgebung des CIDDS-002 Datensatzes [RWG+17b].

den drei Klassen *normal*, *attacker* und *victim* zugeordnet werden. Den größten Anteil des Netzwerkverkehrs stellen normale Benutzeraktivitäten (Klassenlabel *normal*) dar. In Tabelle 8.7 ist ersichtlich, dass beide Wochen eine ähnliche Anzahl von Flows besitzen.

Tabelle 8.7.: Klassenverteilung der Flows im CIDDS-002 Datensatz.

Klasse	Woche 1	Woche 2	Summe
normal	7.919.622	7.678.921	15.598.543
attacker	162.688	189.065	351.753
victim	103.682	107.205	210.887
Summe	8.185.992	7.975.191	16.161.183

Der zeitliche Verlauf der aufgenommenen Flows ist in Abbildung 8.7 dargestellt. Auf der y-Achse ist die Anzahl der Flows pro Stunde zu sehen und jede Linie in der Abbildung repräsentiert den Netzwerkverkehr einer Woche. In Abbildung 8.7 sind die typischen Arbeitszeiten leicht zu erkennen. So steigt bei typischen Arbeitstagen die Anzahl der Flows pro Stunde ab 06:00 Uhr bis zur Mittagszeit (gegen 11:30 Uhr) und sinkt ab 16:00 Uhr, wenn die ersten Clients Feierabend machen. Nachts nimmt die Anzahl der Flows ab und bleibt auf konstant niedrigen Niveau.

Die beobachteten Flows in der Nacht werden hauptsächlich durch standardmäßige Aktualisierungsanfragen der Betriebssysteme und Synchronisationsdienste (Netzlaufwerk) verursacht. Am Samstag arbeitet nur ein Client und am Sonntag arbeitet kein Client, was zu geringeren bis keinen Ausschlägen an diesen Tagen führt. Wie schon im CIDDS-001 Datensatz generieren die nächtlichen Backups so wenig Flows, dass diese zu keinen ersichtlichen Ausschlägen führen. Das höhere Volumen von Flows pro Stunde in der Nacht von Montag auf Dienstag in Woche 1 ist auf einen Port Scan zurückzuführen. Des Weiteren wurde ein Systemausfall am Freitag von 03:52 bis 09:08 in Woche 1 simuliert, der zu einen Datenverlust führte.

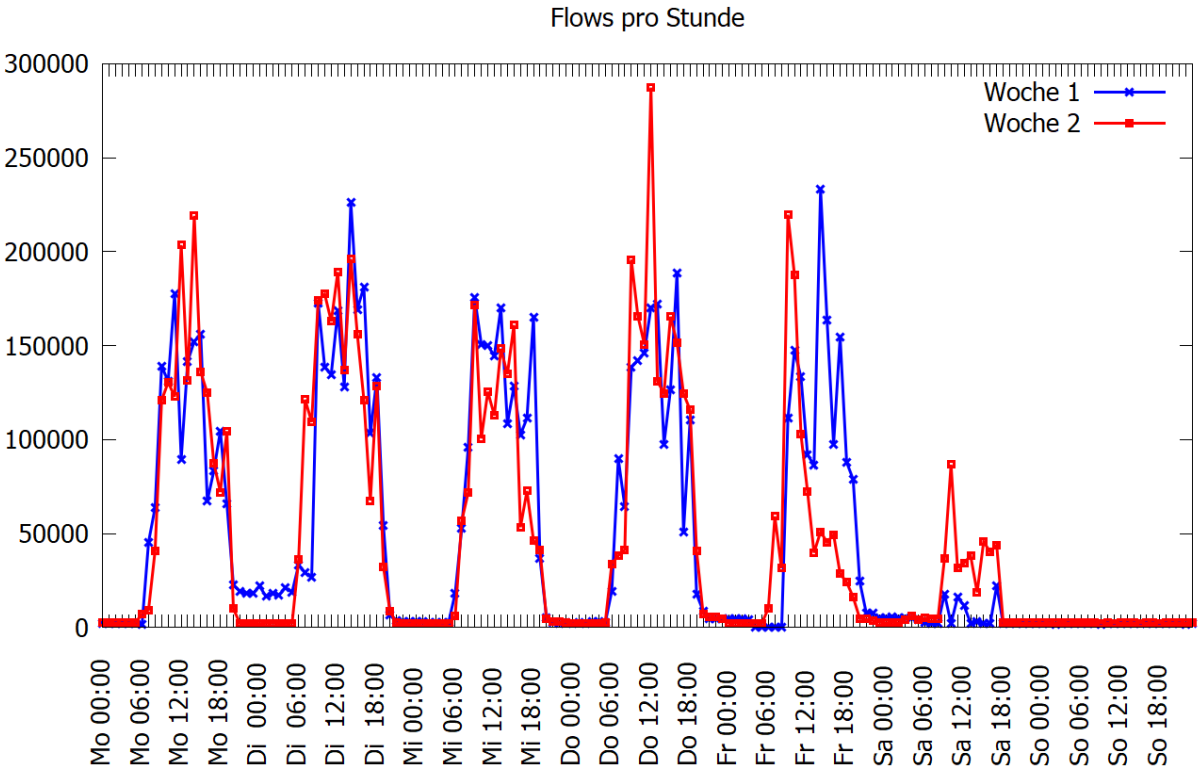


Abbildung 8.7.: Zeitlicher Verlauf der aufgenommenen Flows. Die y-Achse stellt die Anzahl der Flows pro Stunde dar. Jede Linie repräsentiert den Netzwerkverkehr einer Woche.

Im CIDDS-002 Datensatz wurde das Tool Nmap[1] zum Ausführen der Port Scans verwendet. Tabelle 8.8 liefert genauere Informationen über Anzahl und Art der ausgeführten Port Scans in Woche 1 und Woche 2. Insgesamt wurden die fünf verschiedene Arten ACK-Scans, SYN-Scans, FIN-Scans, UDP-Scans und Ping-Scans durchgeführt. Die Port Scans unterscheiden sich primär durch die unterschiedlichen TCP-Flags, die bei den Anfragen gesetzt werden. So wird beispielsweise beim FIN-Scan lediglich das FIN-Flag gesetzt, um Firewallregeln zu umgehen. Im Vergleich dazu sendet der SYN-Scan das SYN-Flag zum Initiieren einer Netzwerkverbindung, schließt den Verbindungsaufbau jedoch nicht ab. Der Parameter T steuert die Geschwindigkeit, in welcher das Tool Nmap die Port Scans durchführt. Ein Wert von $T = 0$ führt dazu, dass alle fünf Minuten ein Testpaket versendet wird, während beim Wert $T = 1$ alle 15 Sekunden und

[1] https://nmap.org/

beim Wert $T = 2$ alle 0,4 Sekunden ein Testpaket versendet wird. Grundsätzlich führen höhere Werte von T zu einer höheren Frequenz beim Versenden der Testpakete [Lyo08].

Tabelle 8.8.: Anzahl ausgeführter Port Scans im CIDDS-002 Datensatz. Der Parameter T kontrolliert die Ausführungsgeschwindigkeit eines Port Scans.

	SYN Scan -T 1	SYN Scan -T 2	SYN Scan -T 3	ACK Scan -T 1	ACK Scan -T 2	ACK Scan -T 3	UDP Scan -T 0	UDP Scan -T 1	UDP Scan -T 2	UDP Scan -T 3	FIN Scan -T 1	FIN Scan -T 2	FIN Scan -T 3	Ping Scan -T1	Ping Scan -T2	Summe
Woche 1	2	1	1	2	1	2	1	2	2	0	0	2	3	0	1	20
Woche 2	3	2	0	1	0	1	0	1	3	1	3	3	2	2	1	23
Summe	5	3	1	3	1	3	1	3	5	1	3	5	5	2	2	43

8.5. Zusammenfassung

Die Entwicklung, Konfiguration und Evaluierung netzwerkbasierter IDS erfordert aktuelle und gelabelte Datensätze. Jedoch fehlen aufgrund mehrerer Ursachen, wie beispielsweise der Komplexität des Labelling Prozesses oder aus datenschutzrechtliche Gründen, öffentlich verfügbare, aktuelle und gelabelte Datensätzen.

In diesem Kapitel wurde ein Konzept zur Generierung netzwerkbasierter Datensätze vorgestellt. Das Konzept sieht die Verwendung einer Testumgebung vor und simuliert normales Benutzerverhalten durch Python Skripte. Um die Differenz zwischen realen und simulierten Benutzerverhalten möglichst gering zu halten, berücksichtigen die Skripte gewisse Rahmenbedingungen. Diese Rahmenbedingungen beinhalten die Einhaltung typischer Arbeitszeiten und das zufallsbedingte Ausführen unterschiedlicher Aktionen wie Browsen, Drucken oder Kopieren von Dateien. Die Testumgebung wurde in OpenStack realisiert. Im Vergleich zu Produktivumgebungen können in dieser Testumgebung Angriffsszenarien ausgeführt werden. Ein Vorteil des vorgestellten Konzepts besteht darin, dass kontinuierlich neue Datensätze erzeugt und somit stetig neues Benutzerverhalten und neuartige Angriffsszenarien integriert werden können. Zur Integration von echtem Netzwerkverkehr sieht das Konzept die Integration externer Server vor. Diese Server bieten Dienste für die simulierten Clients an, sind jedoch realen und aktuellen Bedrohungen aus dem Internet ausgesetzt. Durch diese Vorgehensweise und der Veröffentlichung der Datensätze und Skripte konnten die vordefinierten Anforderungen, also die Aspekte (1) aktueller Netzwerkverkehr sowie anpassbare Netzwerkstrukturen, (2) Präsenz realistischer Benutzeraktivitäten, (3) exaktes Labeln und Bereitstellung zusätzlicher Informationen, (4) Integration von echten Netzwerkverkehr und (5) Veröffentlichung der Daten, erreicht werden.

Neben den soeben genannten Vorteilen geht das Konzept jedoch auch mit Einschränkungen einher. Die wesentliche Einschränkung ist, dass nur durch Python Skripte implementierte Benutzerverhalten simuliert werden können. Dies schränkt beispielsweise die Anzahl existierender Angriffsszenarien auf derzeit vier Angriffstypen ein. Sollen neuartige Benutzerverhalten oder Angriffsszenarien simuliert werden, müssen diese zunächst implementiert werden. Eine andere offene Fragestellung ist, inwiefern sich der simulierte Netzwerkverkehr von tatsächlichen Netzwerkverkehr unterscheidet. Zwar werden die simulierten Benutzeraktionen tatsächlich in der

Testumgebung ausgeführt, was zu realistischen Verzögerungen und real übertragenen Daten führt, jedoch ist die Frage, ob die Häufigkeit und Nutzungsdauer der simulierten Dienste der Realität entspricht, noch ungeklärt und muss in zukünftigen Arbeiten stärker fokussiert werden.

Basierend auf dem vorgestellten Konzept wurden zwei Datensätze, CIDDS-001 und CIDDS-002, generiert. In beiden Datensätzen wurde die Netzwerkumgebung eines kleinen Unternehmens simuliert. Für den CIDDS-001 Datensatz wurde ein zusätzlicher externer Server verwendet. Anschließend wurde Netzwerkverkehr für 4 Wochen (CIDDS-001) und 2 Wochen (CIDDS-002) aufgezeichnet. Beide Datensätze beinhalten größtenteils normales Benutzerverhalten und verschiedene Angriffsszenarien. Der Netzwerkverkehr wurde im unidirektionalen NetFlow Format aufgezeichnet und in einer aufwendigen Nachbearbeitung mit Labels versehen und analysiert. Hierbei wurde vor allem der zeitliche Verlauf der Flows, in dem sich die typischen Arbeitszeiten korrekt widerspiegeln und die Verteilung der Klassenlabels analysiert.

9. Generierung flowbasierter Netzwerkdaten durch Modellierung

Dieses Kapitel adressiert ebenfalls die Herausforderung fehlender Datensätze (siehe Abschnitt 1.2.2) und beschäftigt sich mit der synthetischen Generierung flowbasierter Netzwerkdaten durch Modellierung. Der vorgestellte Ansatz beruht auf sogenannten Generative Adversarial Networks (GANs) und verwendet flowbasierte Netzwerkdaten als Eingabe. Der Ansatz erlernt die zugrundeliegenden Eigenschaften der flowbasierten Netzwerkdaten und ist anschließend in der Lage, neuen Netzwerkverkehr mit gleichen Eigenschaften zu generieren. Somit ist der nachfolgende Ansatz als Ergänzung für den Generierungsansatz mittels Simulation aus Kapitel 8 zu werten, da existierende Datensätze durch den vorgestellten Ansatz mit synthetisch modellierten Flows angereichert werden können.

Der Inhalt dieses Kapitels wurde bereits veröffentlicht und basiert auf der Publikation *„Flow-based Network Traffic Generation using Generative Adversarial Networks“* [RSL+19].

9.1. Einleitung

Für die Entwicklung, Konfiguration und Evaluierung signatur- und anomaliebasierter Sicherheitssysteme werden gelabelte Datensätze benötigt. Die beiden vorherigen Kapitel diskutierten bereits die Diskrepanz zwischen der Notwendigkeit gelabelter Datensätze und dem Mangel an öffentlich verfügbaren Datensätzen. Dieses Kapitel konzentriert sich auf die Generierung flowbasierter Netzwerkdaten, welche für die Entwicklung und Evaluierung netzwerkbasierter Sicherheitssysteme erforderlich sind.

Problem. Das Normalverhalten der Nutzer ändert sich im Laufe der Zeit und es entstehen täglich neue Angriffsszenarien. Signaturbasierte Sicherheitssysteme verwenden eine Liste bekannter Signaturen von Angriffen und können Angriffe, deren Signaturen sich von den bekannten Signaturen unterscheiden, häufig nicht erkennen [MXM+16]. Anomaliebasierte Sicherheitssysteme hingegen erlernen ein Normalverhalten und heben Abweichungen vom erlernten Verhalten als Anomalien hervor. Sobald unbekanntes Verhalten auftritt, muss das anomaliebasierte Sicherheitssystem entscheiden, ob es sich um legitimes oder verdächtiges Verhalten handelt. Dieser Umstand birgt eine große Herausforderung, da Netzwerkverkehr eine große Diversität aufweist, sich im Laufe der Zeit ändert und die Legitimität einer Aktion auch vom Benutzer und Zeitpunkt abhängen. Beispielsweise könnte eine bestimmte Aktion für Benutzer A legitim sein, aber wenn ein Benutzer B die gleiche Aktion ausführt, muss das Sicherheitssystem einen Alarm auslösen. Folglich müssen die Daten, auf welchen das Sicherheitssystem trainiert wird, repräsentativ und möglichst allumfassend sein.

Ziel. Datensätze mit großer Varianz können die Robustheit anomaliebasierter Sicherheitssysteme erhöhen. Dieses Kapitel zielt auf die Entwicklung eines Modells zur Generierung realistischer flowbasierter Netzwerkdaten ab. Das Modell verwendet flowbasierte Netzwerkdaten als Eingabe und wird darauf optimiert, die Eigenschaften der Eingabedaten zu erlernen und neue synthetische Daten zu generieren, welche die gleichen Eigenschaften aufweisen. Anschlie-

ßend können die ursprünglichen Datensätze mit den generierten Daten angereichert und zum Trainieren und Evaluieren anomaliebasierter Sicherheitssysteme verwendet werden. Ein grundsätzlicher Vorteil synthetisch erzeugter Daten ist, dass diese mit weniger datenschutzrechtlichen Einschränkungen einhergehen, da es sich um keine realen Nutzerdaten handelt.

Ansatz und Beiträge. Generative Adversarial Networks (GANs) [GPM+14] sind eine verbreitete Methode zur Generierung synthetischer Daten. GANs bestehen aus zwei neuronalen Netzen, einen Generator-Netzwerk und einen Diskriminator-Netzwerk. Das Generator-Netzwerk generiert synthetische Daten und das Diskriminator-Netzwerk versucht, die synthetisch erzeugten Daten von realen Daten zu unterscheiden. GANs erzielen bemerkenswerte Ergebnisse bei der Generierung von Bildern [RMC16, LTH+17, IZZ+17, KAL+18] und wurden auch schon in anderen Anwendungsgebieten wie der Textgenerierung [YZW+17, ZGC16, GLC+18] oder der Generierung von Molekülketten [PRU+18] eingesetzt.

In diesem Kapitel werden GANs auf die Domäne IT-Sicherheit angepasst und zur Generierung flowbasierter Netzwerkdaten verwendet. GANs können in ihrer ursprünglichen Form nur kontinuierliche Attribute verarbeiten, flowbasierte Netzwerkdaten beinhalten jedoch auch kategorische Attribute wie IP-Adressen (siehe Kapitel 4.2.2). Aus diesem Grund werden verschiedene Ansätze zur Handhabung kategorischer Attribute untersucht. Der erste Ansatz behandelt kategorische Attribute wie Ports als numerische Werte. Der zweite Ansatz erstellt binäre Attribute aus kategorischen Attributen. Der dritte Ansatz verwendet IP2Vec (siehe Kapitel 6) um kontinuierliche Vektorrepräsentationen für IP-Adressen und andere Attribute zu erhalten. Nach dieser Datenvorverarbeitung werden mithilfe von Improved Wasserstein GANs (WGAN-GP) [GAA+17] unter Berücksichtigung der zweifachen zeitlichen Aktualisierungsregel (TTUR) von Heusel et al. [HRU+17] neue flowbasierte Netzwerkdaten basierend auf dem CIDDS-001 Datensatz (siehe Abschnitt 8.3) erzeugt. Anschließend erfolgt eine Bewertung der generierten Daten mit diversen Evaluierungsmaßen.

Das Kapitel hat drei Beiträge. Der Hauptbeitrag ist die Verwendung von GANs zur Generierung flowbasierter Netzwerkdaten. Hierfür werden drei unterschiedliche Methoden zur Vorverarbeitung evaluiert und miteinander verglichen. Des Weiteren wird ein neues Evaluierungsmaß zur Bewertung der generierten Daten vorgestellt und die Methode IP2Vec so erweitert, dass auch Vektordarstellungen für die Attribute Dauer, Bytes und Pakete erlernt werden.

9.2. Konzept zur Modellierung

In diesem Abschnitt wird zunächst die Wahl eines geeigneten Modells zur Generierung flowbasierter Netzwerkdaten beschrieben (siehe Abschnitt 9.2.1). Danach werden drei unterschiedliche Vorverarbeitungsmethoden in Abschnitt 9.2.2 erläutert, die erforderlich sind, damit das ausgewählte Modell auf heterogene flowbasierte Netzwerkdaten angewendet werden kann.

9.2.1. Wahl des generativen Modells

Diskriminative Modelle lernen die Grenzen zwischen Klassen und klassifizieren Datenpunkte in vordefinierte Klassen [HKP11]. Diese werden häufig zur Detektion sicherheitskritischer Ereignisse verwendet (beispielsweise in [WFS+11, BJS+14, SP15, PAG+07, KAT07, HSC+11]). Generative Modelle gehen davon aus, dass die beobachteten Daten von einem Modell erzeugt wurden und versuchen dieses Modell zu erlernen. Anschließend können die generativen Modelle auch zum Erzeugen neuer Daten genutzt werden. Viele generative Modelle verwenden die Maximum-Likelihood-Schätzung, um die Parameter einer Wahrscheinlichkeitsverteilung zu

schätzen. Da die Wahrscheinlichkeitsfunktion oft unbekannt oder die Berechnung sehr rechenaufwendig ist, umgehen beispielsweise Restricted Boltzmann Machines [HS06] das Schätzen der Maximum Likelihood durch Rekonstruktion der Eingabedaten.

GANs verfolgen einen Ansatz aus der Spieltheorie. Ein Generator-Netzwerk versucht Beispiele einer echten Datenverteilung zu emittieren. Gleichzeitig versucht ein Diskriminator-Netzwerk die generierten Beispiele von realen Beispielen zu unterscheiden. Beide Netzwerke werden iterativ trainiert, bis der Diskriminator nicht mehr in der Lage ist, generierte von realen Beispielen zu unterscheiden. GANs besitzen den Vorteil, dass der Generator nur wenigen Restriktionen unterliegt [Goo16]. Daneben wird der Generator niemals mit realen Beispielen konfrontiert, stattdessen bekommt dieser nur einen Vektor mit zufällig generierten Werten als Eingabe. Der Generator wird mit den berechneten Loss des Diskriminators durch Backpropagation trainiert. Deshalb ist auch die Wahrscheinlichkeit von Overfitting durch Speichern und Reproduzieren von realen Beispielen für GANs nahezu auszuschließen [Goo16].

Goodfellow et al. [GPM+14] nennen als weiteren Vorteil von GANs deren Möglichkeit zur Generierung komplexer Verteilungen, was auf einige Attribute von flowbasierten Netzwerkdaten zutrifft. Allerdings benötigen die ursprünglichen GANs von Goodfellow et al. [GPM+14] differenzierbare Eingabedaten, was für kategorische Attribute wie IP-Adressen nicht zutrifft. Des Weiteren sind die ursprünglichen GANs sensitiv gegenüber Parameterkonfigurationen und die Loss-Funktion korreliert häufig nicht mit der Qualität generierter Daten. Wasserstein GANs (WGANs) [ACB17] nutzen den Wasserstein-Abstand als Loss-Funktion, ersetzen das Diskriminator-Netzwerk durch ein sogenanntes Critic-Netzwerk und stellen eine Weiterentwicklung der ursprünglichen GANs dar. Arjovsky et al. [ACB17] zeigen, dass die Loss-Funktion von WGANs mit der Qualität der erzeugten Daten korreliert. Des Weiteren zeigen Gulrajani et al. [GAA+17], dass WGANs in der Lage sind, diskrete Verteilungen über einen kontinuierlichen Raum zu modellieren. WGANs beschränken die Gewichte des Critic-Netzwerks, um die Differenzierbarkeit zu gewährleisten. Diese Beschränkung der Gewichte wird in Arjovsky et al. [ACB17] als Weight Clipping bezeichnet. Gulrajani et al. [GAA+17] veröffentlichten eine Methode zum besseren Training von WGANs, indem die Autoren Weight Clipping entfernen und durch eine sogenannte Gradient Penalty ersetzen. Die Idee von Gradient Penalty ist die Erhaltung der Differenzierbarkeit, indem die Norm des Gradienten bezüglich der Eingabewerte angepasst wird [GAA+17]. Die Autoren bezeichnen diese verbesserte Trainingsmethode als WGAN-GP.

Ein weiteres Forschungsfeld im Bereich GANs ist deren fehlende Konvergenz. Heusel et al. [HRU+17] schlagen eine zweifache zeitliche Aktualisierungsregel (TTUR) für das Training von GANs mit beliebigen Loss-Funktionen vor. Die Grundidee von TTUR ist die Verwendung unterschiedlicher Lernraten für den Generator und den Critic beziehungsweise Diskriminator. Heusel et al. [HRU+17] zeigen, dass GANs durch TTUR unter milden Annahmen konvergieren.

Aus den genannten Gründen werden in diesem Kapitel Improved Wasserstein Generative Adversarial Networks (WGAN-GP) [GAA+17] mit der zweifachen zeitlichen Aktualisierungsregel von Heusel et al. [HRU+17] zur Generierung flowbasierter Netzwerkdaten verwendet. Für eine genauere Definition der Funktionsweise der verwendeten WGANs sei auf [ACB17], [GAA+17] und [HRU+17] verwiesen. Die Grundidee und Funktionsweise ursprünglicher GANs [GPM+14] wird in Abschnitt 3.4 erläutert.

9.2.2. Datenvorverarbeitung

In diesem Abschnitt werden drei unterschiedliche Methoden zur Transformation flowbasierter Netzwerkdaten vorgestellt, sodass diese von Improved Wasserstein Generative Adversarial Networks (WGAN-GP) verarbeitet werden können.

9.2.2.1. Generelle Vorverarbeitung

Die drei nachfolgenden Methoden verwenden den gleichen Vorverarbeitungsansatz für die Attribute Zeitstempel, Transport-Protokoll und TCP-Flags (siehe Tabelle 4.1).

In der Regel ist der konkrete Zeitstempel für Sicherheitssysteme nur bedingt relevant. Stattdessen extrahieren Sicherheitssysteme häufig zusätzliche Informationen aus dem Zeitstempel, wie beispielsweise *„Fällt dieser Zeitstempel auf einen Arbeitstag?“* oder *„Tritt das Event außerhalb der typischen Arbeitszeit auf?“*. Deshalb werden bei den vorgestellten Methoden lediglich der Wochentag und die Uhrzeit anstelle konkreter Zeitstempel generiert. Zur internen Repräsentation des Wochentags werden sieben binäre Attribute istMontag, istDienstag, istMittwoch, istDonnerstag, istFreitag, istSamstag und istSonntag erstellt. Zur Repräsentation der Tageszeit wird die Uhrzeit als Sekunden pro Tag $[0, 86400)$ interpretiert und auf das Intervall $[0, 1]$ normiert.

Das Attribut Transport-Protokoll (siehe #5 in Tabelle 4.1) wird durch drei binäre Attribute istUDP, istTCP und istICMP dargestellt. Die gleiche Vorverarbeitung findet für TCP-Flags (siehe #10 in Tabelle 4.1) statt, welche folglich durch sechs binäre Attribute istURG, istACK, ist PSH, istSYN, istRES und istFIN dargestellt werden.

9.2.2.2. Methode 1 - Numerische Transformation

IP-Adressen und Ports sind kategorische Attribute, deren Ausprägungen Zahlen sind. Methode 1 interpretiert diese Attribute dennoch als numerische Werte. Diese Methode wird als N-WGAN-GP (Numeric-based Improved Wasserstein Generative Adversarial Networks) bezeichnet.

Bei dieser Transformation wird jedes Oktett einer IP-Adresse auf das Intervall $[0, 1]$ normiert. Die IP-Adresse 192.168.220.14 wird beispielsweise in vier kontinuierliche Attribute $ip\_1 = 192/255 = 0{,}7529$, $ip\_2 = 168/255 = 0{,}6588$, $ip\_3 = 220/255 = 0{,}8627$ und $ip\_4 = 14/255 = 0{,}0549$ transformiert. Ein ähnlicher Ansatz wird für Ports durchgeführt, indem diese auf das Intervall $[0, 1]$ normiert werden. Beispielsweise wird der Port 80 auf ein kontinuierliches Attribut mit dem Wert $80/65535 = 0{,}00122$ transformiert.

Die Attribute Dauer, Bytes und Pakete (siehe Attribute #7, #8 and #9 in Tabelle 4.1) werden jeweils auf das Intervall $[0, 1]$ normiert. In Tabelle 9.1 ist ein Beispiel für die Transformation und ein Vergleich der drei Methoden abgebildet.

9.2.2.3. Methode 2 - Binäre Transformation

Die zweite Methode erstellt mehrere binäre Attribute für IP-Adressen, Ports, Bytes und Pakete. Diese Methode wird als B-WGAN-GP (Binary-based Improved Wasserstein Generative Adversarial Networks) bezeichnet.

B-WGAN-GP transformiert jedes Oktett einer IP-Adresse in dessen 8-Bit Binärdarstellung. Folglich extrahiert dieser Ansatz aus jeder IPv4-Adresse 32 binäre Attribute. Beispielsweise wandelt dieser Ansatz die IP-Adresse 192.168.220.14 in 11000000 10101000 11011100 00001110 um. Ports werden ebenfalls in ihre 16-Bit Binärdarstellung konvertiert und als 16 binäre Attribute interpretiert. Der Port 80 wird beispielsweise durch die 16 binären Attribute 00000000 01010000

Tabelle 9.1.: Überblick der drei Vorverarbeitungsmethoden. In der ersten Spalte sind die Attribute eines Flows mit Beispielwerten aufgelistet. Die weiteren Spalten zeigen die transformierten Attribute und zugehörigen Werte für jede Methode an [RSL+19].

	N-WGAN-GP		**B-WGAN-GP**		**E-WGAN-GP**	
Attribut / Wert	Attribut	Wert	Attribut	Wert	Attribut	Wert
Zeitstempel 2018-05-28 11:39:23	istMontag istDienstag istMittwoch istDonnerstag istFreitag istSamstag istSonntag Tageszeit	1 0 0 0 0 0 0 $\frac{41963}{86400} = 0{,}485$	istMontag istDienstag istMittwoch istDonnerstag istFreitag istSamstag istSonntag Tageszeit	1 0 0 0 0 0 0 $\frac{41963}{86400} = 0{,}485$	istMontag istDienstag istMittwoch istDonnerstag istFreitag istSamstag istSonntag Tageszeit	1 0 0 0 0 0 0 $\frac{41963}{86400} = 0{,}485$
Dauer 1,503	norm_dur	$\frac{1{,}503 - dur_{min}}{dur_{max} - dur_{min}}$	norm_dur	$\frac{1{,}503 - dur_{min}}{dur_{max} - dur_{min}}$	dur_1 ... dur_m	$\begin{pmatrix} e_1 \\ \ldots \\ e_m \end{pmatrix}$
Transport-Protokoll TCP	istTCP istUDP istICMP	1 0 0	istTCP istUDP istICMP	1 0 0	istTCP istUDP istICMP	1 0 0
IP-Adresse 192.168.210.5	ip_1 ip_2 ip_3 ip_4	$\frac{192}{255} = 0{,}7529$ $\frac{168}{255} = 0{,}6588$ $\frac{210}{255} = 0{,}8627$ $\frac{5}{255} = 0{,}0196$	ip_1 bis ip_8 ip_9 bis ip_16 ip_17 bis ip_24 ip_25 bis ip_32	1,1,0,0,0,0,0,0 1,0,1,0,1,0,0,0 1,1,0,1,0,0,1,0 0,0,0,0,0,1,0,1	ip_1 ... ip_m	$\begin{pmatrix} e_1 \\ \ldots \\ e_m \end{pmatrix}$
Port 53872	pt	$\frac{53872}{65535} = 0{,}8220$	pt_1 bis pt_8 pt_9 bis pt_16	1,1,0,1,0,0,1,0 0,1,1,1,0,0,0,0	pt_1 ... pt_m	$\begin{pmatrix} e_1 \\ \ldots \\ e_m \end{pmatrix}$
Bytes 144	norm_byt	$\frac{144 - byt_{min}}{byt_{max} - byt_{min}}$	byt_1 bis byt_8 byt_9 bis byt_16 byt_17 bis byt_24 byt_25 bis byt_32	0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0 1,0,0,1,0,0,0,0	byt_1 ... byt_m	$\begin{pmatrix} e_1 \\ \ldots \\ e_m \end{pmatrix}$
Pakete 1	norm_pk	$\frac{1 - pk_{min}}{pk_{max} - pk_{min}}$	pk_1 bis pk_8 pk_9 bis pk_16 pk_17 bis pk_24 pk_25 bis pk_32	0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,1	pk_1 ... pk_m	$\begin{pmatrix} e_1 \\ \ldots \\ e_m \end{pmatrix}$
TCP-Flags .A..S.	istURG istACK istPSH istRES istSYN istFIN	0 1 0 0 1 0	istURG istACK istPSH istRES istSYN istFIN	0 1 0 0 1 0	istURG istACK istPSH istRES istSYN istFIN	0 1 0 0 1 0

repräsentiert. Die Attribute Bytes und Pakete werden ebenfalls binär repräsentiert und ihre Länge auf 32 binäre Attribute beschränkt.

Das Attribut Dauer wird anhand der Eingabedaten auf das Intervall $[0, 1]$ normiert. Eine beispielhafte Transformation für einen Flow ist in Tabelle 9.1 zu sehen.

9.2.2.4. Methode 3 - Embedding Transformation

Die dritte Methode nennt sich E-WGAN-GP (Embedding-based Improved Wasserstein Generative Adversarial Networks) und transformiert IP-Adressen, Ports, Dauer, Bytes und Pakete in sogenannte Embeddings.

Embeddings stellen m-dimensionale Vektorrepräsentationen im kontinuierlichen Raum $\mathbb{R}^m$ dar. Dieser Ansatz basiert auf der Grundidee von IP2Vec (siehe Kapitel 6) und erweitert diese, sodass auch Vektorrepräsentationen für die Attribute Dauer, Bytes und Pakete erlernt werden. Zu diesem Zweck wird das Vokabular von IP2Vec um die Werte dieser Attribute erweitert und zusätzliche Trainingspaare extrahiert. Die angepasste Generierung von Trainingspaaren ist in Tabelle 9.2 dargestellt. Eine beispielhafte Transformation ist in Tabelle 9.1 zu sehen.

Tabelle 9.2.: Erweiterte Generierung von Trainingsbeispielen in IP2Vec [RSL+19]. Es werden die folgenden Abkürzungen verwendet: Quell-IP (Quelle-IP-Adresse), Ziel-IP (Ziel-IP-Adresse) und Proto (Transport-Protokoll).

									Eingabewert	**Ausgabewert**
Quell-IP	Quell-Port	Ziel-IP	Ziel-Port	Proto	Bytes	Pakete	Dauer	→	Quell-IP	Ziel-IP
								→	Quell-IP	Quell-Port
								→	Quell-IP	Proto
Quell-IP	Quell-Port	Ziel-IP	Ziel-Port	Proto	Bytes	Pakete	Dauer	→	Ziel-IP	Quell-IP
								→	Ziel-IP	Ziel-Port
								→	Ziel-IP	Proto
Quell-IP	Quell-Port	Ziel-IP	Ziel-Port	Proto	Bytes	Pakete	Dauer	→	Quell-Port	Quell-IP
Quell-IP	Quell-Port	Ziel-IP	Ziel-Port	Proto	Bytes	Pakete	Dauer	→	Ziel-Port	Ziel-IP
Quell-IP	Quell-Port	Ziel-IP	Ziel-Port	Proto	Bytes	Pakete	Dauer	→	Bytes	Pakete
								→	Bytes	Dauer
Quell-IP	Quell-Port	Ziel-IP	Ziel-Port	Proto	Bytes	Pakete	Dauer	→	Pakete	Bytes
								→	Pakete	Dauer
Quell-IP	Quell-Port	Ziel-IP	Ziel-Port	Proto	Bytes	Pakete	Dauer	→	Dauer	Pakete

Aus jedem Flow werden 13 Trainingspaare bestehend aus einem Eingabewert und einem erwarteten Ausgabewert extrahiert. Der Eingabewert ist durch türkisen Hintergrund in Tabelle 9.2 hervorgehoben. Die erwarteten Ausgabewerte sind durch grauen Hintergrund hervorgehoben. Multi- und Broadcast IP-Adressen erscheinen in unidirektionalen Flows nur als Ziel-IP-Adressen, was zur Folge hat, dass für diese IP-Adressen keine sinnvollen Vektorrepräsentationen erlernt werden, wenn die Ziel-IP-Adresse nicht als Eingabewort berücksichtigt wird. Deshalb extrahiert E-WGAN-GP auch Trainingspaare mit der Ziel-IP-Adresse als Eingabewert.

E-WGAN-GP transformiert einige Attribute in Vektorrepräsentationen, welche nach der Generierung durch das Generator-Netzwerk wieder in den ursprünglichen Wertebereich zurück transformiert werden müssen. Hierfür berechnet E-WGAN-GP zunächst die Ähnlichkeit des generierten Vektors zu allen bekannten Vektorrepräsentationen des jeweiligen Attributs. Zur Berechnung von Ähnlichkeiten wird das Cosinusmaß (siehe Abschnitt 2.4.4) verwendet. Danach selektiert E-WGAN-GP die ähnlichste bekannte Vektorrepräsentation und transformiert den generierten Vektor auf den ursprünglichen Wert der ähnlichsten bekannten Vektorrepräsentation zurück. Diese Vorgehensweise kann mit einen kNN Klassifikator (siehe 3.1.1) mit Parameter

$k = 1$ verglichen werden. In diesem Fall würde jede bekannte Vektorrepräsentation einer Klasse entsprechen und das Label der Klasse ist der Wert im ursprünglichen Raum.

9.3. Experimente

In diesem Abschnitt findet eine experimentelle Evaluierung der vorgestellten Methoden zur Generierung flowbasierter Netzwerkdaten mithilfe des CIDDS-001 Datensatzes statt.

9.3.1. Datensatz

In den Experimenten wird der CIDDS-001 Datensatz (siehe Abschnitt 8.3) als Eingabe verwendet. Die Informationen über die bei der Generierung verwendeten Subnetze werden zur späteren Evaluierung der generierten Daten verwendet. Dabei werden die folgenden Abkürzungen verwendet: Entwickler-Subnetz (*dev*), Büro-Subnetz (*off*), Management-Subnetz (*mgt*), Server-Subnetz (*srv*) und externer Netzwerkverkehr (*ext*).

Der CIDDS-001 Datensatz beinhaltet Netzwerkverkehr über einen Zeitraum von vier Wochen. In den nachfolgenden Experimenten wird nur der Netzwerkverkehr verwendet, der innerhalb der OpenStack-Umgebung (siehe Abbildung 8.3) aufgenommen wurde. Die ersten beiden Wochen des Datensatzes beinhalten normalen Netzwerkverkehr und unterschiedliche Angriffsszenarien. Die Wochen 3 und 4 beinhalten ausschließlich normalen Netzwerkverkehr. Für die experimentelle Evaluierung wird der Datensatz in zwei Teile, Woche 1 und Woche 2 bis Woche 4, aufgeteilt. Diese Aufteilung wird verwendet, um einen möglichst großen Trainingsdatensatz (Woche2-4) zu erhalten und einen Referenzdatensatz (Woche1), der ebenfalls normalen und verdächtigen Netzwerkverkehr beinhaltet (siehe Abschnitt 8). Der Teil Woche2-4 beinhaltet ungefähr 22 Millionen Flows und der Teil Woche1 umfasst ungefähr 8,5 Millionen Flows.

9.3.2. Definition einer Baseline

Als Baseline wird ein generatives Modell verwendet, welches synthetische Flows basierend auf den beobachteten Wahrscheinlichkeitsverteilungen der Eingangsdaten erstellt. Die Baseline schätzt dabei die Wahrscheinlichkeitsverteilungen für jedes Attribut durch Abzählen. Synthetische Flows werden durch zufälliges Ziehen aus den beobachteten Wahrscheinlichkeitsverteilungen generiert. Jedes Attribut wird hierbei unabhängig von den anderen Attributen gezogen. Die Baseline zieht auch das Attribut Zeitstempel aus den beobachteten Daten und generiert somit nicht zwei neue Attribute Wochentag und Tageszeit wie die vorgestellten Methoden.

9.3.3. Evaluierungsmethodik

Die Evaluierung von generativen Modellen ist keine triviale Aufgabe und stellt ein aktuelles Forschungsgebiet dar [SBL+18, LKM+18]. Das ursprüngliche Anwendungsgebiet von GANs findet sich in der Generierung von Bildern. Deshalb sind die meisten existierenden Evaluierungsmaße für GANs auf die Bewertung synthetisch erzeugter Bilder fokussiert. Im Folgenden werden existierende Evaluierungsmaße für GANs analysiert und auf ihre Übertragbarkeit auf flowbasierte Netzwerkdaten geprüft. Borji [Bor19] analysiert verschiedene Evaluierungsmaße für GANs. Häufig werden generierte Bilder durch Darstellung und menschliches Feedback beurteilt. Ein weit verbreitetes Maß ist der Inception Score [SGZ+16]. Der Inception Score klassifiziert generierte Bilder in 1000 vordefinierte Klassen mithilfe des Inception Net v3 [SVI+16]. Dabei wird geprüft, wie eindeutig die generierten Bilder einer Klasse zugeordnet und wie viele verschiedene Klasse

erzeugt werden. Dieses Maß ist jedoch nicht auf das vorliegende Szenario anwendbar, da das Inception Net v3 lediglich Bilder und keine flowbasierten Netzwerkdaten klassifizieren kann.

In der Domäne IT-Sicherheit gibt es weder einen Konsens, wie Netzwerkdaten-Generatoren zu bewerten sind, noch standardisierte Evaluierungsmaße [MMS13]. Glasser und Lindauer [GL13] behandeln ebenfalls das Problem der Evaluierung von synthetisch erzeugten Netzwerkdaten. Glasser und Lindauer [GL13] verwenden Feedback von Domänenexperten, um die Qualität der synthetisch generierten Netzwerkdaten zu bewerten. Stiborek et al. [SRP15] verwenden eine Anomaliefunktion zur Evaluierung der generierten Daten. Siska et al. [SSK+10] und Iannucci et al. [IKG+17] bauen Graphen basierend auf den generierten Daten und bewerten die Vielfalt der erzeugten Daten durch Vergleich der Knoten und Kanten zwischen erzeugtem und realem Netzwerkverkehr. Andere Netzwerkdaten-Generatoren fokussieren oft einzelne Aspekte in der Evaluierung, beispielsweise vergleichen Sommers und Barford [SB04] und Botta et al. [BDP12] die Verteilung der Attribute Bytes und Pakete mit echten Netzwerkverkehr.

Da es keine einheitliche, allgemein anerkannte Evaluierungsmethode gibt, werden im Folgenden unterschiedliche Ansätze zur Evaluierung des erzeugten Netzwerkverkehrs verwendet. Um die Vielfalt und Verteilungen der generierten Daten zu bewerten, werden die Attribute in Abschnitt 9.3.4.2 visualisiert und der Euklidische Abstand zwischen den generierten und realen Netzwerkdaten (Abschnitt 9.3.4.3) berechnet. Des Weiteren werden neu entwickelte Domain Knowledge Checks (siehe Abschnitt 9.3.4.4) verwendet, um die Abhängigkeiten zwischen den Attributen eines Flows zu beurteilen. Diese Tests basieren auf einer ähnlichen Idee wie der Ansatz von Glasser und Lindauer [GL13]. Während Glasser und Lindauer Domänenexperten aktiv in den Evaluierungsprozess mit einbeziehen, wird beim nachfolgenden Ansatz Domänenwissen in automatisierbare Testfunktionen ausgelagert und genutzt.

9.3.4. Generierung flowbasierter Netzwerkdaten

Alle vier Ansätze trainieren auf Woche2-4 des CIDDS-001 Datensatzes und generieren jeweils 8,5 Millionen Flows.

9.3.4.1. Parameterkonfigurationen

N-WGAN-GP, B-WGAN-GP und E-WGAN-GP nutzen vorwärtsgerichtete neuronale Netze (siehe Abschnitt 3.1.4) für Generator und Critic. Des Weiteren werden die Parameterkonfigurationen von Heusel et al. [HRU+17] übernommen und die neuronalen Netze fünf Epochen lang trainiert. Eine Beobachtung war, dass eine größere Anzahl von Trainingsepochen weder zu geringeren Werten in der Loss-Funktion, noch zu einer besseren Ergebnisqualität führte. Wie in Heusel et al. [HRU+17], werden für den Generator und Critic jeweils 3 versteckte Schichten konfiguriert. Im Generator werden ReLU Aktivierungsfunktionen und im Critic LeakyReLU Aktivierungsfunktionen mit $\alpha = 0.2$ verwendet. Des Weiteren wird die Lernrate für den Critic auf 0.0003 und für den Generator auf 0.0001 gesetzt. Um die Anzahl der Neuronen pro versteckter Schicht zu identifizieren, wurde eine Parameterstudie im Bereich 8 bis 192 durchgeführt. Diese Studie ergab, dass B-WGAN-GP und E-WGAN-GP mit 80 Neuronen pro versteckter Schicht die besten Ergebnisse erzielen. Deshalb wird die Anzahl Neuronen pro versteckter Schicht auf 80 fixiert. Grundsätzlich existierten im Bereich 64 bis 144 Neuronen pro Schicht keine großen Unterschiede. Für N-WGAN-GP wird die Anzahl der Neuronen pro versteckter Schicht auf 24 fixiert, da die numerische Repräsentation viel weniger Dimensionen als B-WGAN-GP und E-WGAN-GP besitzt.

Die Methode E-WGAN-GP berechnet in einem vorgelagerten Schritt Vektorrepräsentationen mithilfe von IP2Vec. IP2Vec wird so konfiguriert, dass es 10 Epochen trainiert und 20 Neuronen in der versteckten Schicht verwendet.

9.3.4.2. Visualisierung

Abbildung 9.1 visualisiert die zeitliche Verteilung der Flows. Die drei Linien repräsentieren die Anzahl Flows pro Stunde für die Baseline, E-WGAN-GP und der Referenzwoche Woche1. Da alle drei Ansätze N-WGAN-GP, B-WGAN-GP und E-WGAN-GP den Zeitstempel auf die gleiche Art und Weise generieren, ist zur besseren Übersicht in Abbildung 9.1 nur ein Ansatz abgebildet. E-WGAN-GP reflektiert die wesentliche Verteilung der Flows pro Stunde. Im CIDDS-001 Datensatz absolvieren die simulierten Benutzer regelmäßig Kaffee- und Mittagspausen was zu einer gezackten Linie (zum Beispiel gegen 12:00 an Arbeitstagen für Woche1) führt. Im Vergleich hierzu weist E-WGAN-GP einen glatteren Verlauf als die Referenzwoche auf.

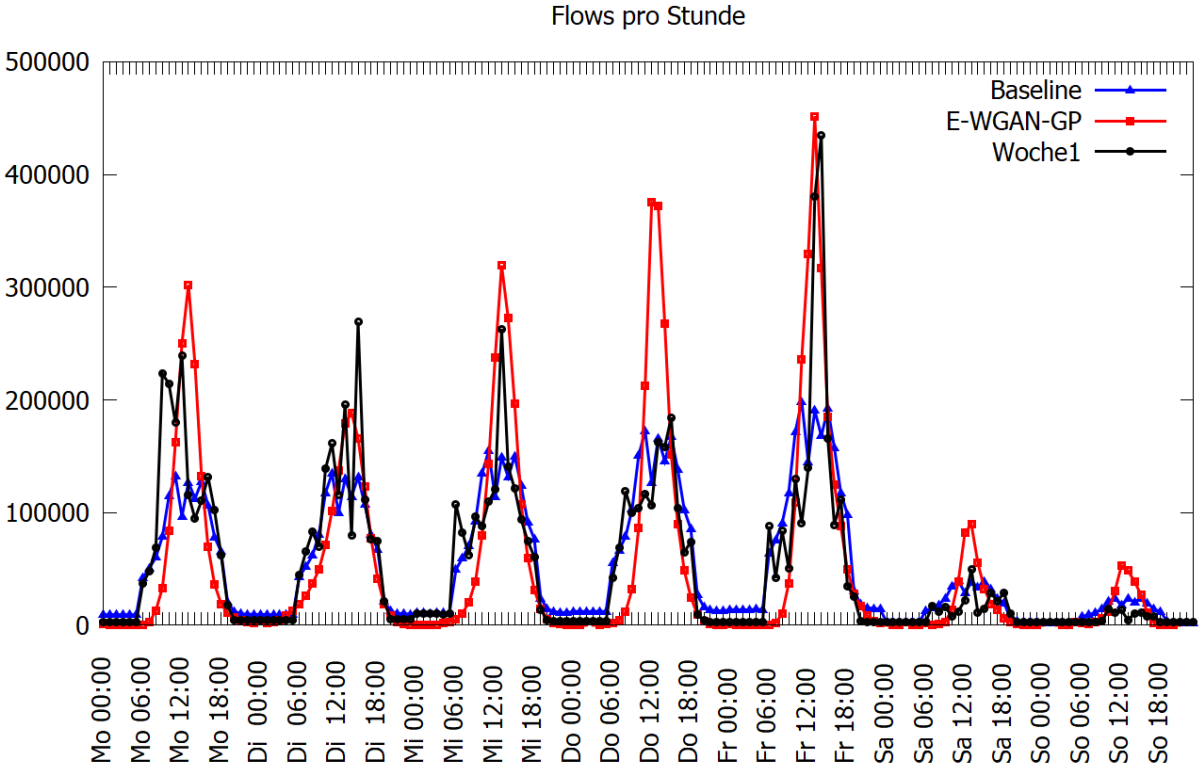


Abbildung 9.1.: Zeitliche Verteilung der Flows pro Stunde nach Ring et al. [RSL+19].

In den folgenden zwei Abbildungen werden die generierten Daten mithilfe von Violindarstellungen (siehe Abschnitt 3.5.2) visualisiert, um einen tieferen Einblick zu erhalten. Die Abbildungen 9.2 und 9.3 zeigen die Referenzwoche Woche1 (erste Zeile), die generierten Daten der Baseline (zweite Zeile) und die generierten Daten der drei vorgestellten Methoden N-WGAN-GP, B-WGAN-GP und E-WGAN-GP (dritte bis fünfte Zeile).

Jede Violindarstellung zeigt die bedingte Verteilung des Attributs Quell-Port (in Abbildung 9.2) beziehungsweise des Attributs Ziel-IP-Adresse (in Abbildung 9.3) bei unterschiedlichen Quell-IP-Adressen. Dabei werden die Quell-IP-Adressen nach Subnetz gruppiert. In Abbildung 9.2 sind die Quell-Ports auf der y-Achse eingetragen. In Abbildung 9.3 wird die 32-Bit Darstellung

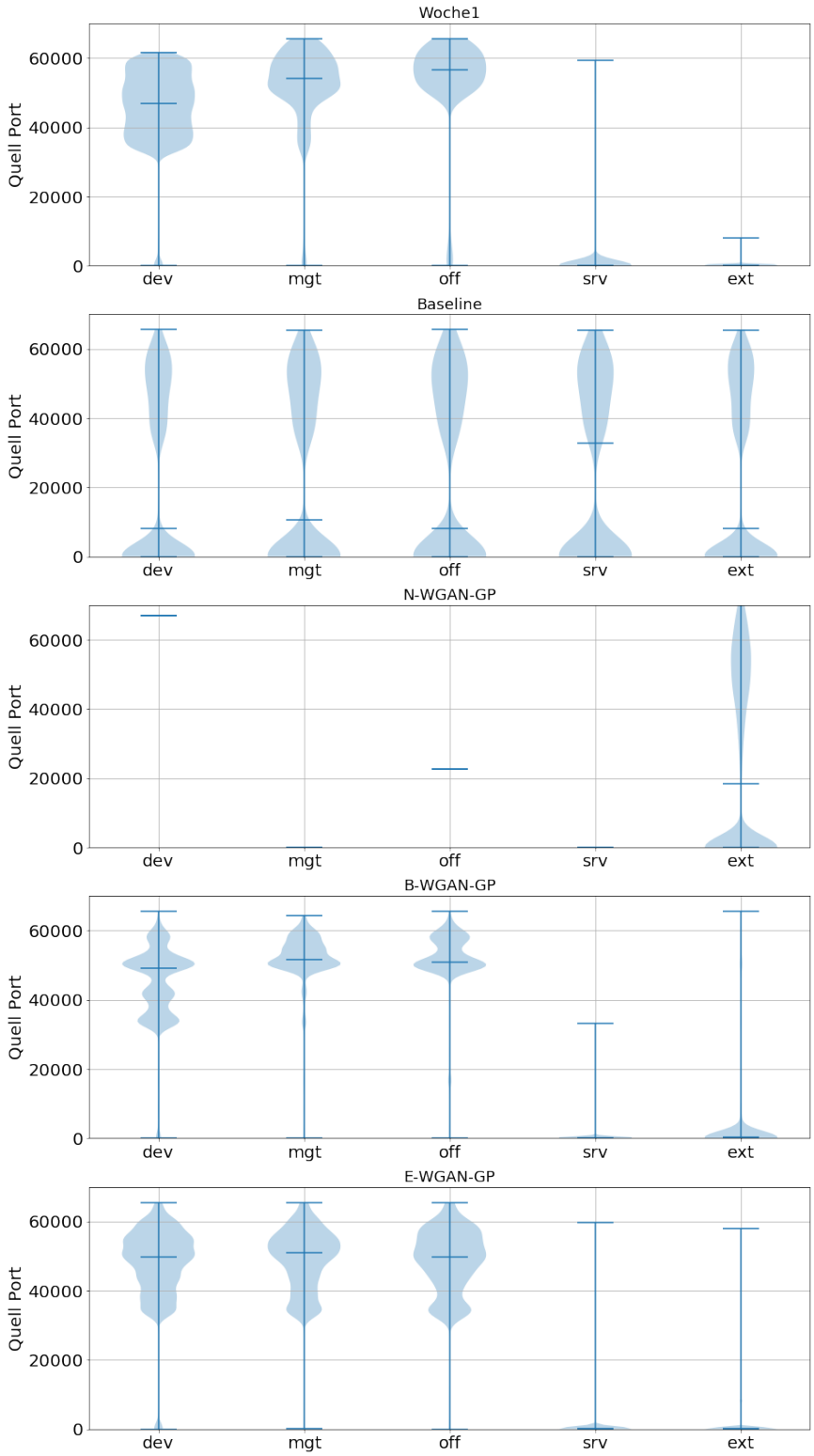


Abbildung 9.2.: Bedingte Verteilungen der Quell-Ports bei unterschiedlichen Quell-IP-Adressen gruppiert nach Subnetz nach Ring et al. [RSL+19].

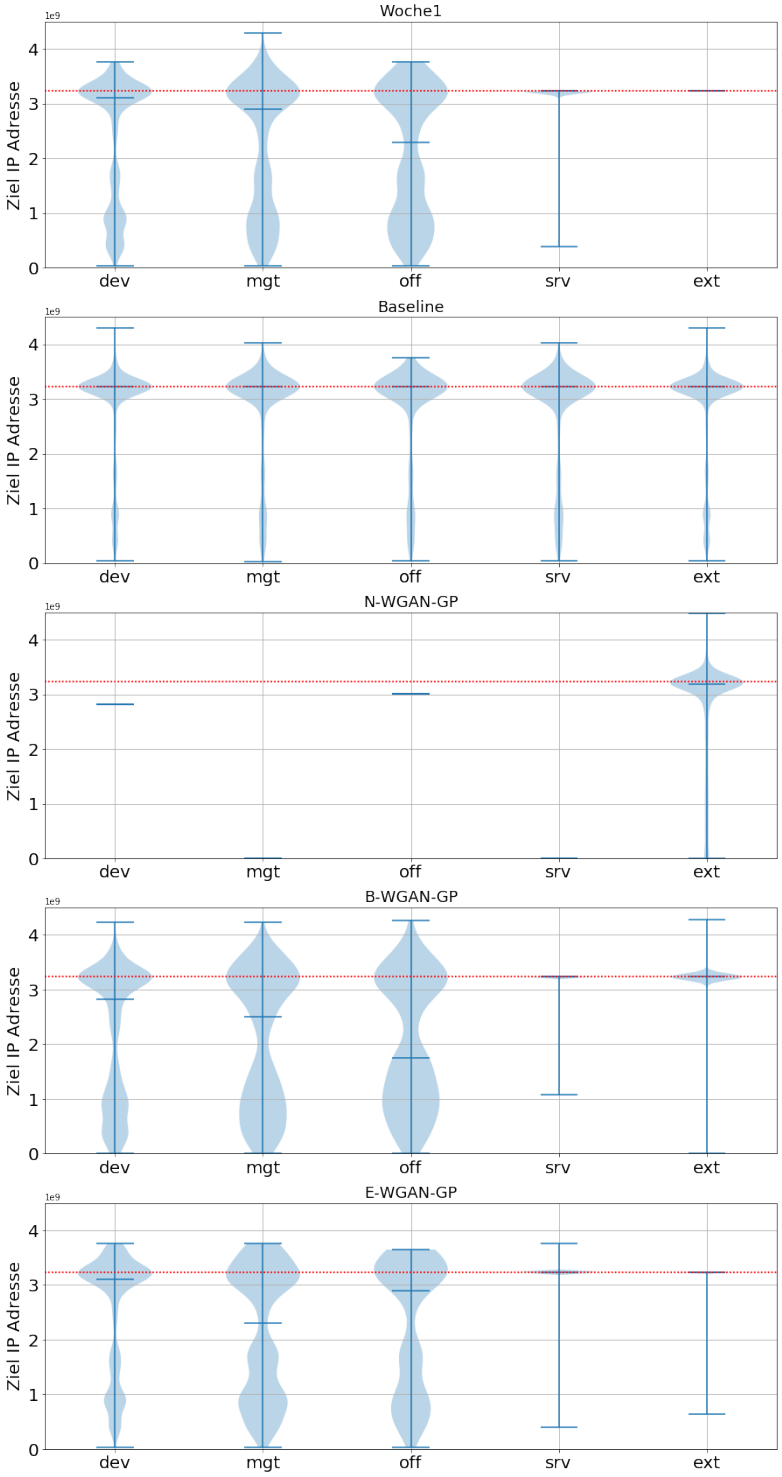


Abbildung 9.3.: Bedingte Verteilungen der Ziel-IP-Adressen bei unterschiedlichen Quell-IP-Adressen gruppiert nach Subnetz nach Ring et al. [RSL+19]. Rote Linien indizieren den Bereich der internen IP-Adressen (192.168.X.X).

der Ziel-IP-Adressen als positive Ganzzahl interpretiert und entsprechend auf der y-Achse eingetragen. Der Bereich der internen IP-Adressen (192.168.X.X) ist durch eine rot gepunktete Linie hervorgehoben. Die IP-Adressen aus den unterschiedlichen Subnetzen zeichnen sich durch unterschiedliche Verhalten aus. Beispielsweise stellen IP-Adressen aus den *mgt* Subnetz typische Clients dar, welche Services wie HTTP oder SMTP nutzen. Im Gegensatz dazu befinden sich im *srv* Subnetz Server, welche derartige Dienste anbieten. Dieses Wissen wurde nicht explizit in den Datengenerierungsprozess modelliert.

Im Folgenden werden die bedingten Verteilungen des Attributs Quell-Port bei unterschiedlichen Subnetzen der Quell-IP-Adressen (siehe Abbildung 9.2) diskutiert. In der ersten Zeile (Woche1) können typische Verteilungen für Client-Ports (Subnetze *dev*, *mgt* und *off*) und typische Verteilungen für Server-Ports (Subnetze *ext* und *srv*) unterschieden werden. Die Baseline (zweite Zeile) kann die Unterschiede der bedingten Verteilungen in Abhängigkeit vom Subnetz der Quell-IP-Adresse nicht erfassen und modelliert stets Verteilungen, welche Kombinationen der fünf Subnetze darstellen. B-WGAN-GP und E-WGAN-GP erfassen die bedingten Wahrscheinlichkeitsverteilungen für das Attribut Quell-Port bei unterschiedlichen Quell-IP-Adressen (nach Subnetzen gruppiert) sehr gut. N-WGAN-GP kann die bedingten Verteilungen nicht korrekt darstellen. Für N-WGAN-GP kann beobachtet werden, dass nahezu ausschließlich Flows mit externen IP-Adressen selektiert wurden. Eine genauere Analyse der generierten Daten führt zu der Erkenntnis, dass N-WGAN-GP häufig bei der Generierung exakter Werte fehlschlägt. Fast alle generierten Daten sind dem Subnetz *ext* zugewiesen, daher ist es kaum verwunderlich, dass die Verteilung in diesem Subnetz eine Kombination aller fünf Subnetze der Eingangsdaten darstellt. Diese Beobachtung kann sowohl für die bedingte Verteilung der Quell-Ports (Abbildung 9.2) als auch für die bedingte Verteilung der Ziel-IP-Adressen (Abbildung 9.3) gemacht werden.

Für das Attribut Ziel-IP-Adresse stellt die Verteilung der Referenzwoche Woche1 eine Mischung aus internen und externen IP-Adressen für die Subnetze *dev*, *mgt* und *off* dar (siehe Abbildung 9.3). Dies entspricht den Rollen typischer Clients, die im Internet surfen (externe Ziel-IP-Adressen) und interne Serverdienste wie Drucker oder E-Mail nutzen. Für das Subnetz *ext* müssen alle Ziel-IP-Adressen im internen Bereich liegen, da Netzwerkverkehr, welcher von externen Quellen verursacht wird und externe Ziele hat, nicht durch die simulierte Umgebung des CIDDS-001 Datensatzes läuft. Als Folge dessen gibt es keinen einzigen Flow im CIDDS-001 Datensatz, der eine externe Quell-IP-Adresse und eine externe Ziel-IP-Adresse besitzt. Diese Tatsache ist in Abbildung 9.3 dadurch zu erkennen, dass für Woche1 die Ziel-IP-Adressen aus dem *ext* Subnetz nur einen sehr kleinen Bereich umfassen. E-WGAN-GP und B-WGAN-GP erfassen diese Eigenschaft sehr gut, während die Baseline und N-WGAN-GP diese Tatsache nicht widerspiegeln.

9.3.4.3. Euklidischer Abstand

In dieser Evaluierung wird Euklidische Abstand pro Attribut zwischen den Verteilungen der Eingabedaten Woche2-4 und den generierten Flows berechnet. An dieser Stelle wird der Euklidische Abstand der Kullback-Leibler Divergenz vorgezogen, um Berechnungsprobleme für Werte zu umgehen, die eine empirische Häufigkeit von 0 besitzen. Tabelle 9.3 zeigt die Ergebnisse dieser Evaluierung. Das Attribut Zeitstempel wird hier nicht berücksichtigt, da sich im vorherigen Abschnitt eine detaillierte Analyse der generierten Zeitstempel befindet, siehe Abbildung 9.1.

Netzwerkverkehr ändert sich im Laufe der Zeit und deshalb ist eine genaue Reproduktion der Wahrscheinlichkeitsverteilungen nicht erstrebenswert. Diese Tatsache wird durch die Euklidischen Abstände der Wahrscheinlichkeitsverteilungen zwischen Woche1 und Woche2-4 ersicht-

Tabelle 9.3.: Euklidischer Abstand zwischen den Eingabedaten Woche2-4 und den generierten flowbasierten Netzwerkdaten pro Attribut.

Attribut	Baseline	N-WGAN-GP	B-WGAN-GP	E-WGAN-GP	Woche1
Dauer	0,0002	0,4764	0,4764	0,0525	0,0347
Transport-Protokoll	0,0001	0,0014	0,0042	0,0015	0,0223
Quell-IP-Adresse	0,0003	0,1679	0,0773	0,0988	0,1409
Quell-Port	0,0003	0,5658	0,0453	0,0352	0,0436
Ziel-IP-Adresse	0,0003	0,1655	0,0632	0,1272	0,1357
Ziel-Port	0,0003	0,5682	0,0421	0,0327	0,0437
Bytes	0,0002	0,5858	0,0391	0,0278	0,0452
Pakete	0,0004	1,0416	0,0578	0,0251	0,0437
TCP-Flags	0,0003	0,0217	0,0618	0,0082	0,0687

lich, wo die Abstände im CIDDS-001 Datensatz zwischen 0,02 und 0,14 liegen. Folglich sollte der generierte Netzwerkverkehr ähnliche Euklidische Abstände wie Woche1 zu den Trainingsdaten aufweisen. Gleichzeitig ist jedoch zu berücksichtigen, dass es keinen korrekten Wert für den Abstand zwischen Trainingsdaten und synthetisch erzeugen Netzwerkdaten gibt. Tabelle 9.3 zeigt, dass die Baseline die geringsten Abstände aufweist und die ursprünglichen Verteilungen nahezu perfekt nachbildet. Die generierten Daten von E-WGAN-GP weisen ähnliche Abstände wie Woche1 zu den Trainingsdaten auf.

Die generierten Daten von N-WGAN-GP unterscheiden sich in einigen Attributen erheblich von Woche2-4. Der Grund hierfür ist, dass N-WGAN-GP viele Werte nicht exakt reproduzieren kann und somit häufig neue Werte generiert. Im Vergleich dazu fallen die Abstände des binären Ansatzes B-WGAN-GP wesentlich kleiner aus. Einzige Ausnahme ist das Attribut Dauer. Dies kann auf die ursprüngliche Verteilung des Attributs Dauer in den Trainingsdaten zurückgeführt werden, da die meisten Flows sehr kleine Werte im Attribut Dauer aufweisen. Weiterhin führt die Normierung auf das Intervall $[0, 1]$ dazu, dass fast alle Flows sehr niedrige Werte in diesem Attribut besitzen. Deshalb neigen N-WGAN-GP und B-WGAN-GP zur Generierung der kleinstmöglichen Dauer für alle Flows.

9.3.4.4. Domain Knowledge Checks

Die Domain Knowledge Checks dienen zur Evaluierung der intrinsischen Qualität der generierten Flows. Zu diesem Zweck werden mehrere Plausibilitätsregeln abgeleitet, welche die generierte Daten erfüllen müssen, um als realistisch zu gelten. Hierfür werden die folgenden sieben Plausibilitätsregeln angewendet.

- Plausibilitätsregel 1: Falls das Transport-Protokoll UDP ist, dürfen keine TCP-Flags gesetzt sein.
- Plausibilitätsregel 2: Der CIDDS-001 Datensatz wurde in einer simulierten Testumgebung aufgezeichnet. Deswegen muss mindestens eine IP-Adresse (Quell-IP-Adresse oder Ziel-IP-Adresse) pro Flow intern sein (192.168.X.X).
- Plausibilitätsregel 3: Falls der Flow normales Benutzerverhalten beschreibt (Klasse *normal*) und der Quell- oder Ziel-Port den Wert 80 (HTTP) oder 443 (HTTPS) besitzt, muss das Transport-Protokoll TCP sein.

- Plausibilitätsregel 4: Falls der Flow normales Benutzerverhalten beschreibt (Klasse *normal*) und der Quell- oder Ziel-Port den Wert 53 (DNS) besitzt, muss das Transport-Protokoll UDP sein.

- Plausibilitätsregel 5: Falls eine *Multi-* oder *Broadcast* IP-Adresse im Flow auftritt, muss diese in der Ziel-IP-Adresse stehen.

- Plausibilitätsregel 6: Falls der Flow eine Netbios Nachricht (Ziel-Port 137 oder 138) darstellt, muss die Quell-IP-Adresse intern sein (192.168.X.X) und die Ziel-IP-Adresse muss ein interner Broadcast sein (192.168.X.255).

- Plausibilitätsregel 7: TCP, UDP und ICMP Pakete besitzen eine minimale und eine maximale Größe. Deswegen prüft Test 7 das Verhältnis zwischen Bytes und Pakete pro Flow mit folgender Regel: $42 \cdot Pakete \leq Bytes \leq 65.535 \cdot Pakete$

Tabelle 9.4.: Ergebnisse der Domain Knowledge Checks in Prozent.

	Baseline	**N-WGAN-GP**	**B-WGAN-GP**	**E-WGAN-GP**	**Woche1**
Plausibilitätsregel 1	14,08	96,46	97,88	**99,77**	100,0
Plausibilitätsregel 2	81,26	0,61	98,90	**99,98**	100,0
Plausibilitätsregel 3	86,90	95,45	**99,97**	**99,97**	100,0
Plausibilitätsregel 4	15,08	7,14	**99,90**	99,84	100,0
Plausibilitätsregel 5	**100,0**	25,79	47,13	99,80	100,0
Plausibilitätsregel 6	0,07	0,00	40,19	**92,57**	100,0
Plausibilitätsregel 7	71,26	**100,0**	85,32	99,49	100,0

Tabelle 9.4 zeigt die Ergebnisse der Domain Knowledge Checks für die generierten Daten. Die Referenzwoche Woche1 erreicht 100 Prozent für jede Plausibilitätsregel. Dieses Ergebnis ist nicht überraschend, da es sich hierbei um realen Netzwerkverkehr aus der gleichen Netzwerkumgebung handelt. Die Baseline ist nicht in der Lage, die Abhängigkeiten zwischen den Attributen zu erfassen und erreicht nur schlechte Ergebnisse. Dies ist besonders in den Plausibilitätsregeln 1, 4 und 6 ersichtlich, in denen der Zusammenhang mehrerer Attribute evaluiert wird. Da *Multi-* und *Broadcast* IP-Adressen in den Trainingsdaten nur im Attribut Ziel-IP-Adresse auftreten, kann die Baseline für Plausibilitätsregel 5 keine Fehler machen und erreicht 100 Prozent.

Von den generativen Modellen erreicht E-WGAN-GP im Durchschnitt die besten Ergebnisse. Die Verwendung von Vektorrepräsentationen führt zu sinnvollen Ähnlichkeiten innerhalb kategorischer Attribute und unterstützt das Erlernen von Zusammenhängen. B-WGAN-GP generiert Flows mit guten Ergebnissen in den Plausibilitätsregeln 1 bis 4, zeigt jedoch Schwächen in den Plausibilitätsregeln 5 und 6, wo mehrere interne Zusammenhänge erfasst werden müssen. Der numerische Ansatz N-WGAN-GP besitzt die geringste Ergebnisqualität in den durchgeführten Plausibilitätsregeln. Insbesondere zeigt Plausibilitätsregel 4, dass die Normierung aller Quell-Ports beziehungsweise aller Ziel-Ports auf ein einziges kontinuierliches Attribut nicht zielführend ist. Das Abbilden von 2^{16} unterschiedlichen Ports auf ein kontinuierliches Attribut führt dazu, dass das transformierte Attribut zu viele Werte repräsentieren muss und verhindert somit eine gute Rekonstruktion. Im Vergleich hierzu erreicht B-WGAN-GP eine bessere Rekonstruktion, da dieser Ansatz alle Quell- beziehungsweise Ziel-Ports mit jeweils 16 binären Attributen repräsentiert. Im nächsten Abschnitt findet eine detaillierte Diskussion der Ergebnisse statt.

9.4. Diskussion

N-WGAN-GP ist ein einfacher Ansatz und interpretiert kategorische Attribute als numerische Zahlen. Diese Handhabung führt dazu, dass nicht existierende Ähnlichkeiten zwischen kategorischen Werten erzeugt werden. Beispielsweise wird durch diese Transformation den IP-Adressen 192.168.220.10 und 191.168.220.10 eine sehr hohe Ähnlichkeit zugewiesen, obwohl es sich bei der ersten IP-Adresse 192.168.220.10 um eine private IP-Adresse handelt und die zweite IP-Adresse 191.168.220.10 eine öffentliche IP-Adresse darstellt. Diese Tatsache führt dazu, dass bereits kleine Fehler im Generierungsprozess signifikante Auswirkungen haben können. Dieser Effekt ist deutlich in Test 2 von Tabelle 9.4 zu erkennen, da N-WGAN-GP Probleme mit der Generierung privater IP-Adressen hat. Statt privater IP-Adressen generiert N-WGAN-GP häufig IP-Adressen die in einem Oktett einen fehlerhaften Wert wie 191.168.X.X oder 192.167.X.X besitzen. Im ursprünglichen Anwendungsgebiet (Bildergenerierung) von GANs führen kleine Fehler nur zu kleinen Auswirkungen. Beispielsweise würde die Helligkeit eines Pixelwertes von 191 anstelle von 192 zu fast keinen Auswirkungen führen und wäre für das menschliche Auge (fast) nicht zu erkennen. Eine weitere Beobachtung ist, dass die direkte Normierung der Attribute Bytes und Pakete auf das Intervall $[0, 1]$ zu keinen guten Ergebnissen führt. Die generierten Daten von N-WGAN-GP werden anschließend mit den originalen Trainingsdaten wieder auf den ursprünglichen Wertebereich transformiert. Grundsätzlich besitzt realer flowbasierter Netzwerkverkehr häufig Flows mit bestimmten Werten in den Attributen Bytes und Pakete. Diese Werte trifft N-WGAN-GP meist nicht exakt und dies führt zu größeren Abständen in Tabelle 9.3. Insgesamt eignet sich die Methode N-WGAN-GP nicht zur Generierung realistischer flowbasierter Netzwerkdaten.

B-WGAN-GP extrahiert binäre Attribute aus kategorischen Attributen. Diese Transformation sorgt dafür, dass vorhandene strukturelle Informationen erhalten bleiben. Dadurch ordnet diese Transformation den kategorischen Werten größere Wertebereiche im transformierten Raum zu. Während N-WGAN-GP ein kontinuierliches Attribut für die Darstellung von 2^{16} unterschiedlichen Ports verwendet, stellt B-WGAN-GP das Attribut Port mithilfe von 16 binären Attributen dar. Diese beiden Aspekte führen zu einer verbesserten Generierung der kategorischen Werte eines Flows, was in den Ergebnissen der Domain Knowledge Checks (siehe Test 2 und 4 in Tabelle 9.4) zu erkennen ist. Weiterhin wird in den Visualisierungen (siehe Abbildungen 9.2 und 9.3) ersichtlich, dass B-WGAN-GP die wesentlichen inhaltlichen Zusammenhänge der Flows reflektiert. Im Vergleich zu E-WGAN-GP ist B-WGAN-GP weniger restriktiv, da es die Erstellung neuer Werte ermöglicht.

E-WGAN-GP lernt Vektorrepräsentationen für IP-Adressen, Ports, Bytes, Pakete und Dauer. Die Vektorrepräsentationen berücksichtigen Kontextinformationen aus dem zugrundeliegenden Datensatz. Als Folge dessen ist die Generierung weniger fehleranfällig, da kleine Fehler im transformierten Raum das Ergebnis im ursprünglichen Raum nicht wesentlich verändern. Wenn das GAN einen kleinen Fehler bei der Generierung von IP-Adressen verursacht, würde es mit hoher Wahrscheinlichkeit die Vektorrepräsentation einer anderen, jedoch ähnlichen IP-Adresse erzeugen. Beispielsweise würde die IP-Adresse 192.168.220.5 als nächster Nachbar anstelle der erwarteten IP-Adresse 192.168.220.13 erzeugt werden. Das es sich jedoch bei beiden Werten um private IP-Adressen von internen Clients handelt, wäre der Fehler, ähnlich wie beim ursprünglichen Anwendungsgebiet der Bildergenerierung, nur marginal. Als Folge dessen erreicht E-WGAN-GP die besten Ergebnisse in der experimentellen Evaluierung. Im Gegensatz zu N-WGAN-GP und B-WGAN-GP kann diese Methode jedoch keine neuen Werte erzeugen, da generierte Werte stets auf die ähnlichste Vektorrepräsentation zurückgeführt werden. Diese

Vorgehensweise stellt kein Problem für die Attribute Bytes, Pakete und Dauer dar. Wenn der Datensatz groß genug ist, stehen Vektorrepräsentationen für alle wichtigen Werte von Bytes, Pakete und Dauer zur Verfügung. Als Beispiel soll das Attribut Bytes dienen. Angenommen, es existieren Vektorrepräsentationen für die Werte $b_1, b_2, b_3, ..., b_{k-1}, b_k$ des Attributs Bytes und diese decken den möglichen Wertebereich ausreichend ab. Da lediglich die Größenordnung des Attributs von Bedeutung ist und den konkreten Werten nur geringe Bedeutung zukommt, können nicht vorhandene Werte b_x durch verfügbare Werte ersetzt werden, ohne die Semantik zu beeinträchtigen.

Diese Situation kann sich für IP-Adressen und Ports ändern. IP-Adressen identifizieren Hosts und repräsentieren somit das komplette Netzwerkverhalten eines Web-Servers, eines Linux-Clients und so weiter. Das Erstellen neuer IP-Adressen geht einher mit der Erstellung neuer Hosts mit eigenem Netzwerkverhalten. Um die Frage zu beantworten, ob die Generierung neuer IP-Adressen notwendig ist, muss der Zweck analysiert werden, in dem die generierten Daten später verwendet werden. Falls der Trainingsdatensatz beispielsweise 100.000 IP-Adressen beinhaltet und zur Evaluierung eines IDS verwendet werden soll, wird vermutlich kein Bedarf für die Generierung neuer IP-Adressen bestehen. Diese Aussage ist jedoch nicht universell gültig und muss von Fall zu Fall geprüft werden. Hierfür sind die folgenden zwei Fragen zu beantworten. (1) Gibt es bereits ausreichend IP-Adressen im Datensatz? (2) Gibt es einen Bedarf an bisher unbekannten IP-Adressen? Falls die Erstellung unbekannter IP-Adressen erforderlich ist, ist E-WGAN-GP für diese Aufgabe nicht geeignet. Falls jedoch keine neuen IP-Adressen benötigt werden, erreicht E-WGAN-GP die beste Qualität und sollte zur der Generierung synthetischer Flows genutzt werden.

Die Situation für Ports ist ähnlich wie für IP-Adressen. Es existieren 65536 verschiedene Ports und die meisten dieser Ports sollten durch den Trainingsdatensatz abgedeckt werden. Die Erstellung neuer Ports ist ebenfalls mit der Erstellung von neuem Verhalten gekoppelt. Falls der Trainingsdatensatz nur SSH-Verbindungen (Port 22) und HTTP-Verbindungen (Port 80), jedoch keine FTP-Verbindungen (Port 20 und 21) beinhaltet, kann das Modell keine realistischen FTP-Verbindungen generieren, da dieses niemals derartige Verbindungen gesehen hat. Bei FTP-Verbindungen werden in der Regel mehrere Dateien übertragen, was zur Folge hat, dass die dazugehörigen Flows größere Werte in den Attributen Bytes und Pakete aufweisen, als typische SSH- und HTTP-Verbindungen. Aus diesem Grund würde es keinen Sinn ergeben, Flows mit den Ports 20 und 21 zu generieren, da sich FTP-Verbindungen deutlich von den bekannten SSH- und HTTP-Verbindungen unterscheiden. Diese Situation wäre jedoch anders für Ports im dynamischen Bereich (Client Ports). In diesem Bereich verwenden Clients beliebige Ports zum Aufbau unterschiedlicher Verbindungen. Folglich könnten im dynamischen Bereich neue Ports ohne semantische Fehler erzeugt werden. Die Methode E-WGAN-GP verfügt teilweise über dieses Domänenwissen (z. B. Ähnlichkeiten zwischen Ports im dynamischen Bereich), da die von IP2Vec erzeugten Vektorrepräsentationen derartige Zusammenhänge implizit lernen. In zukünftigen Erweiterungen könnte derartiges Domänenwissen explizit formuliert und in die generativen Modelle integriert werden. Eine weitere zukünftige Erweiterung könnte die Evaluierung anderer Embeddingverfahren sein. Das verwendete IP2Vec lernt Embeddings für ganze Wörter. Es existieren jedoch auch Embeddingverfahren, welche Repräsentationen für einzelne Zeichen erlernen und somit größere Freiheitsgrade bei der Erzeugung neuer IP-Adressen und Ports erzielen könnten.

Die experimentellen Evaluierungen zeigen, dass GANs im Allgemeinen in der Lage sind, die bedingten Wahrscheinlichkeitsverteilungen von Netzwerkverkehr zu erfassen, falls geeignete Datenrepräsentationen gewählt werden. Dies ist der Fall für B-WGAN-GP und E-WGAN-GP (siehe

Abbildungen 9.2 and 9.3). Während die Unterschiede zwischen B-WGAN-GP und E-WGAN-GP in der Visualisierung nur geringfügig erscheinen, zeigen die Domain Knowledge Checks größere Leistungsunterschiede auf. Grundsätzlich erreicht E-WGAN-GP die besten Ergebnisse, ist jedoch im Vergleich zu B-WGAN-GP bei der Generierung neuer Flows auf einen vordefinierten Wertebereich beschränkt.

Ein offener Punkt, der in diesem Kapitel nicht behandelt wurde, ist die Generierung von Sequenzen von Flows. Der ausgewählte Ansatz ist zwar in der Lage, einzelne Flows zu generieren, welche beispielsweise realistische DNS- oder HTTP-Anfragen repräsentieren. Die zeitliche Reihenfolge zwischen den Flows wird jedoch nicht berücksichtigt. Beispielsweise würde ein DNS-Flow einem HTTP-Flow vorausgehen. Die Berücksichtigung derartiger Sequenzen stellt eine offene Fragestellung für zukünftige Arbeiten dar. Somit kann dieser Ansatz im derzeitigen Stand genutzt werden, um zusätzliche Daten für Sicherheitssysteme zu generieren, welche Flows individuell verarbeiten (beispielsweise [NKK+14, NKN+16, TJH12]) und für alle Ansätze, die auf Datensätzen ohne Zeitstempeln wie KDD CUP 99 (zum Beispiel [WFS+11, CNM16]) arbeiten. Er sollte jedoch nicht verwendet werden, um zusätzliche Trainingsdaten für Sicherheitssysteme zu generieren, die auf Sequenzen von Flows wie [SYB06] oder auf Zeitfenstern wie [GGS+14] basieren.

9.5. Zusammenfassung

Netzwerkbasierte Datensätze sind für die Entwicklung, Konfiguration und Evaluierung anomaliebasierter Sicherheitssysteme erforderlich. Da Netzwerkverkehr einer zeitlichen Veränderung unterliegt, sollten anomaliebasierte Sicherheitssysteme stets mit repräsentativen Datensätzen trainiert werden, die Varianzen aufweisen, damit die Sicherheitssysteme besser generalisieren und neu auftretenden Netzwerkverkehr korrekt klassifizieren können.

In diesem Kapitel wurde ein Ansatz zur Generierung flowbasierter Netzwerkdaten mittels Improved Wasserstein GANs (WGAN-GP) vorgestellt. Die GANs werden mit flowbasierten Netzwerkdaten initialisiert, lernen deren Eigenschaften und können anschließend synthetische Daten erzeugen, welche die gleichen Eigenschaften besitzen. Eine grundlegende Herausforderung ist, dass GANs nur kontinuierliche Eingabedaten verarbeiten können und flowbasierte Netzwerkdaten auch kategorische Attribute wie IP-Adressen und Ports beinhalten. Deswegen wurden drei unterschiedliche Ansätze zur Vorverarbeitung kategorischer Attribute entwickelt. Im ersten Ansatz N-WGAN-GP werden IP-Adressen und Ports als kontinuierliche Eingabewerte interpretiert und numerische Attribute wie Bytes und Pakete auf das Intervall $[0, 1]$ normiert. Der zweite Ansatz B-WGAN-GP erstellt binäre Attribute aus kategorischen und numerischen Attributen. Der dritte Ansatz E-WGAN-GP transformiert kategorische Werte in Vektorrepräsentationen mithilfe von IP2Vec. Die experimentelle Evaluierung zeigt, dass insbesondere E-WGAN-GP Flows mit hoher Qualität generieren kann. Die von B-WGAN-GP generierten Flows erreichen ähnlich gute Ergebnisse und sind zudem in der Lage, neue Werte zu erzeugen. Die Qualität der von N-WGAN-GP erzeugten Flows ist wesentlich schlechter. Daher kann diese Variante nicht zur Generierung synthetischer Flows empfohlen werden. Da B-WGAN-GP und E-WGAN-GP in der Lage sind, Flows mit hoher Qualität zu generieren, können diese beiden Ansätze verwendet werden, um existierende Datensätze mit synthetisch erzeugten Flows zu erweitern und somit die Varianz dieser Datensätze zu erhöhen. Folglich eignen sich GANs grundsätzlich zur Generierung flowbasierter Netzwerkdaten.

In Zukunft soll geprüft werden, ob sich GANs in Kombination mit LSTMs auch zur Generierung von Flow-Sequenzen eignen und ob verfügbares Domänenwissen (beispielsweise die

definierten Plausibilitätsregeln) explizit in die generativen Modelle integriert werden kann. Um die Einschränkung bei der Generierung neuer IP-Adressen für E-WGAN-GP aufzuheben, bietet sich eine Kombination aus E-WGAN-GP und B-WGAN-GP an, bei der die ersten drei Oktette einer IP-Adresse durch Vektorrepräsentationen und das letzte Oktett durch 8 binäre Attribute dargestellt werden.

Teil IV.

Detektion sicherheitskritischer Ereignisse

10. Konzept zur Detektion sicherheitskritischer Ereignisse

In diesem Kapitel wird ein Konzept zur Detektion sicherheitskritischer Ereignisse in Unternehmensnetzwerken vorgestellt. Das Konzept sieht die Verarbeitung flowbasierter Netzwerkdaten vor und berücksichtigt dabei die in Abschnitt 1.2 identifizierten Herausforderungen. Kernideen sind die Integration von verfügbaren Domänenwissen, die Aggregation von Netzwerkdaten über Zeitfenster sowie die Analyse des Netzwerkverkehrs aus verschiedenen Perspektiven. Zusätzlich greift das Konzept auf Erkenntnisse und Ergebnisse der vorherigen Teile *Behandlung heterogener Daten* (Teil II) und *Generierung von Netzwerkdaten* (Teil III) zurück.

Die nachfolgenden Inhalte wurden bereits in [RWG+17a] publiziert.

10.1. Einleitung

Die immer stärker ansteigende Komplexität netzwerkbasierter Computersysteme durch Digitalisierung, Industrie 4.0 und andere Entwicklungen führt dazu, dass auch die potentielle Angriffsfläche für Hacking-Angriffe wächst. Dies wird beispielsweise in der Anzahl neu auftretender Varianten von Malware oder Ransomware ersichtlich. Während sich die Teile II und III mit grundlegenden Herausforderungen in diesem Themengebiet beschäftigen, geht dieses Kapitel die Herausforderung der Detektion sicherheitskritischer Ereignisse an.

Problem. Realer Netzwerkverkehr unterliegt einer dauerhaften Veränderung. Einerseits erscheinen täglich neue Angriffsszenarien, andererseits ändern sich die täglichen Aktivitäten von Mitarbeitern durch Zuweisung neuer Aufgaben, Übernahme anderer Tätigkeiten, Einführung neuer Systeme und so weiter. Diese Tatsache führt dazu, dass regelbasierte Sicherheitssysteme neu auftretende Muster von Netzwerkverkehr nicht immer korrekt als Normalverhalten oder Angriffsszenario klassifizieren können.

Ziel. Dieses Kapitel strebt den Entwurf eines anomaliebasierten Konzepts zur Detektion sicherheitskritischer Ereignisse in flowbasierten Netzwerkdaten an. Um dieses Ziel zu erreichen, ist eine flexible Methode erforderlich, die existierende Angriffsmuster erkennen und generalisieren kann. Hierfür soll die Tatsache ausgenutzt werden, dass Angriffsszenarien häufig vordefinierten Phasen folgen. Um der Komplexität von Netzwerkverkehr entgegenzutreten, soll dieser aus verschiedenen Perspektiven, welche den typischen Phasen von Angriffsszenarien entsprechen, analysiert werden.

Ansatz und Beiträge. In diesem Kapitel wird ein neuartiges Konzept zur Detektion sicherheitskritischer Ereignisse vorgestellt. Das Konzept sieht die Verarbeitung von flowbasierten Netzwerkdaten (siehe Kapitel 4.2.2) vor, da diese leicht an zentralen Netzwerkgeräten wie Switches und Firewalls abgefangen und der zunehmenden Bedrohung von Innerhalb (Insider-Angriffe) gerecht werden können. Speziell für die Detektion von Insider-Angriffen ist es schwierig, auf ungewöhnliches Verhalten hinzuweisen, da diese Verhaltensweisen unter leicht unterschiedlichen Umständen völlig normal sein können.

Die Tatsache, dass unterschiedliche Benutzeraktivitäten auf Betriebssystemebene unterschiedlich viele Flows auf Netzwerkebene generieren, die Zulässigkeit einer Aktion von Benutzer und Zeitpunkt abhängen und flowbasierte Netzwerkdaten nur Verbindungsinformationen und keine Nutzdaten beinhalten (siehe Herausforderung 3 in Abschnitt 1.2.3) erschwert die Analyse von flowbasierten Netzwerkdaten. Deshalb wird eine mehrstufige Verarbeitungskette angestrebt. Zunächst erfolgt eine Anreicherung flowbasierter Netzwerkdaten mit Domänenwissen. Hierbei werden beispielsweise zusätzliche Informationen über den Ursprung (intern/extern) von IP-Adressen bereitgestellt. Anschließend werden die flowbasierten Netzwerkdaten in einem weiteren Verarbeitungsschritt in Datenpunkte transformiert, welche die zugrundeliegenden Benutzeraktivitäten auf Betriebssystemebene besser beschreiben. Zur Detektion sicherheitskritischer Ereignisse werden die neu berechneten Datenpunkte aus drei Perspektiven analysiert. Diese Perspektiven sind auf die typischen Phasen eines Angriffs angepasst und erlauben die Berechnung von Datenpunkten mit speziell angepassten Attributen. Die Aufspaltung in drei Perspektiven reduziert die Komplexität der einzelnen Perspektiven.

Der Hauptbeitrag dieses Kapitels ist die Präsentation eines Konzepts zur anomaliebasierten Detektion sicherheitskritischer Ereignisse in Unternehmensnetzwerken.

10.2. Verwandte Arbeiten

In diesem Abschnitt werden existierende netzwerkbasierte Methoden zur Detektion sicherheitskritischer Ereignisse genauer untersucht. Bhuyan et al. [BBK14] liefern einen umfassenden Überblick über paket- und flowbasierte Ansätze. Ein aktueller Überblick von Data Mining-Methoden und Methoden des maschinellen Lernens im Bereich IT-Sicherheit ist in [BG16] zu finden. Weitere Übersichtsarbeiten, mit einem Fokus auf spezielle Teilgebiete im Bereich Intrusion Detection sind [GDM+09], [WBS15], [THL+09], [NMY+18], [JYP18], [PP07b], [CG12] oder [KKK+17]. Die nachfolgende Literaturrecherche fokussiert auf flowbasierte Ansätze und kategorisiert diese in drei Gruppen. (I) Behandlung einzelner Flows, (II) Aggregation aller Flows pro Zeitfenster und (III) Aggregation aller Flows eines Hosts pro Zeitfenster.

Kategorie I Winter et al. [WHZ11] veröffentlichten einen Ansatz der Kategorie (I). Grundlegende Idee der Autoren ist, dass Data Mining-Methoden besser im Finden von Ähnlichkeiten als im Finden von Anomalien sind. Deshalb verwenden Winter et al. eine One-Class SVM und trainieren diese ausschließlich auf verdächtigem Netzwerkverhalten. Die Autoren verwendeten den Honeypot Datensatz von [SSV+09]. In der Evaluierung stellen Flows, die der Klasse zugeordnet werden können, verdächtiges Verhalten dar und werden als Anomalie klassifiziert. Ein weiterer Ansatz dieser Kategorie wurde von Tran et al. [TJH12] veröffentlicht. Die Autoren verwenden ein neuronales Netz und integrieren dieses auf einem FPGA. Zum Training und zur Detektion von Angriffen extrahieren die Autoren vier Attribute aus flowbasiertem Netzwerkverkehr (Pakete, Bytes, Dauer und TCP-Flags). Najafabadi et al. [NKK+14] evaluieren die Eignung von vier Klassifikatoren zur Detektion von SSH-Brute-Force Angriffen. Hierfür extrahieren die Autoren acht Attribute aus flowbasierten Netzwerkdaten und erkennen Angriffe mit hoher Genauigkeit. Eine andere Arbeit beschäftigt sich mit der Erkennung von RUDY-Angriffen durch Klassifikatoren [NKN+16]. RUDY-Angriffe sind DoS-Angriffe, die auf Anwendungsebene stattfinden und somit viel weniger Netzwerkverkehr als traditionelle DoS-Angriffe verursachen. Najafabadi et al. [NKN+16] evaluieren ihren Ansatz anhand des SANTA Datensatzes [WKZ+14], der eine Kombination aus flow- und paketbasierten Daten darstellt.

Ansätze, die auf dem KDD CUP 1999 Datensatz angewendet wurden, können der Kategorie (I) zugeordnet werden, da die Datenpunkte keine Zeitstempel beinhalten und in der Regel separat verarbeitet werden. Auf KDD CUP 1999 wurden bereits verschiedene Analyseverfahren wie Assoziationsregeln [TRM09], Bayessches Netze [JZA07], Clusterverfahren [LL05, ZXW05], Evolutionäre Algorithmen [Kha11], Hidden Markov Modelle [JP05], Naive Bayes Klassifikatoren [ABE04, PP07a], Random Forrest Klassifikatoren [ZZH08, GG07] oder SVMs [AYL+11, HSC+11] untersucht.

Kategorie II Ansätze der zweiten Kategorie fassen alle Flows über Zeitfenster zusammen. Wagner et al. [WFS+11] entwickelten einen speziellen Kernel für anomaliebasierte Angriffserkennung. Grundlegende Idee von [WFS+11] ist die Unterteilung des flowbasierten Datenstroms in gleich große Zeitfenster. Anschließend wird jedes Zeitfenster als ein Datenpunkt für den entwickelten Kernel betrachtet. Der Kernel berücksichtigt die IP-Adressen und die übertragenen Bytes aller Flows innerhalb eines Zeitfensters. Wagner et al. integrieren diese Kernelfunktion in eine One-Class SVM, evaluieren ihren Ansatz bei einem ISP und konnten Auffälligkeiten wie DoS Angriffe erkennen. Nychis et al. [NSA+08] veröffentlichten einen Entropie-basierten Ansatz. Die Autoren unterteilen den flowbasierten Datenstrom in Zeitfenstern von 5 Minuten und berechnen für jedes Zeitfenster die Verteilung von sieben flowbasierten Attributen. Basierend auf diesen Verteilungen berechnen die Autoren Entropiewerte, die zur Erkennung von Anomalien verwendet werden.

Kategorie III Ansätze der dritten Kategorie aggregieren pro Host den flowbasierten Netzwerkverkehr über Zeitfenster. In der Regel werden Hosts anhand der Quell-IP-Adressen der Flows identifiziert. Basierend auf diesen Zusammenfassungen berechnen Ansätze der Kategorie III pro Host und pro Zeitfenster neue Datenpunkte mit speziell angepassten Attributen. BClus [GGS+14] verwendet diesen Ansatz zur Detektion von Botnets. Zunächst aggregiert BClus den kompletten Netzwerkverkehr für jede Quell-IP-Adresse über Zeitfenster. Danach werden für jede Aggregation neue Attribute wie beispielsweise die Anzahl der adressierten IP-Adressen berechnet. Einen weiteren Ansatz dieser Kategorie stellen Najafabadi et al. [NKC+15] vor. Die Autoren fassen flowbasierten Netzwerkverkehr über 5 Minuten Zeitfenster zusammen, der in den Attributen Quell-IP-Adresse, Ziel-IP-Adresse und Ziel-Port übereinstimmt. Basierend auf diesen Zusammenfassungen werden neue Datenpunkte mit Attributen wie durchschnittlich übertragene Bytes oder Standardabweichung der übertragenen Bytes berechnet. Danach trainieren Najafabadi et al. [NKC+15] verschiedene Klassifikatoren auf den so erstellten Datenpunkten um SSH-Brute-Force Angriffe zu detektieren. Sourek et al. [ŠKŽ15] fassen ebenfalls unidirektionale Flows zu sogenannten Events mit neu berechneten Attributen zusammen. Die Zusammenfassungen basieren auf Regeln, sodass beispielsweise alle Flows mit der gleichen Quell-IP-Adresse und dem gleichen Quell-Port zusammengefasst werden. Anschließend wenden die Autoren Random Forrest Klassifikatoren zur Angriffsdetektion an. Ein ähnlicher Ansatz wird in MINDS (Minnesota Intrusion Detection System) [CEE+06] verfolgt. MINDS verarbeitet zwar jeden Flow separat, verwendet jedoch vorhergehende Flows für die Berechnung zusätzlicher Attribute. Das System verfügt über unterschiedliche Module, die auf verschiedene Data Mining-Methoden wie Anomalieerkennung (Local outlier factor [BKN+00]) oder Clusterverfahren (Shared Nearest Neighbor [ESK03]) zurückgreifen.

Das hier vorgestellte Konzept verarbeitet nicht alle Flows separat (Kategorie I) und fasst auch nicht alle Flows über Zeitfenster zusammen (Kategorie II). Stattdessen folgt das Konzept

Kategorie III und aggregiert pro Host die flowbasierten Netzwerkdaten über Zeitfenster. Alleinstellungsmerkmal im Vergleich zu anderen Ansätzen von Kategorie III ist die Erstellung mehrerer Datenpunkte pro Host und pro Zeitfenster. Die Erzeugung mehrerer Datenpunkte erlaubt die angepasste Erstellung von Datenpunkten, welche das Netzwerk-, Service-, und Benutzerverhalten eines Hosts beschreiben. Dadurch können die erstellten Datenpunkte aus verschiedenen Perspektiven bewertet werden und es wird kein Klassifikator wie in Wagner et al. [WFS+11] oder in Winter et al. [WHZ11] benötigt, der in der Lage sein muss, alle Angriffsarten zu erkennen.

10.3. Problemstellung und Schlussfolgerungen für flowbasierter Netzwerkdatenanalyse

Ziel des Konzepts ist die Detektion von sicherheitskritischen Ereignissen in flowbasierten Netzwerkdaten mithilfe von Data Mining-Methoden. Deshalb werden in diesem Abschnitt die Auswirkungen typischer Angriffsszenarien auf flowbasierte Netzwerkdaten und Data Mining-Methoden diskutiert.

Das Konzept sieht die Verarbeitung unidirektionaler Flows (siehe Abschnitt 4.2.2) vor. Diese können leicht an zentralen Komponenten wie Switches oder Firewalls abgefangen werden und ermöglichen die Überwachung des kompletten Unternehmensnetzwerks. Reguläre Benutzeraktivitäten wie Browsen im Internet, aber auch die Durchführung von Angriffsszenarien wie Port Scans, erzeugen nicht nur einzelne Flows, sondern Sequenzen von Flows in flowbasierten Netzwerkdaten.

Die individuellen Phasen eines Angriffs (siehe Abschnitt 4.3) haben unterschiedliche Auswirkungen auf flowbasierte Netzwerkdaten. In der Reconnaissance-Phase werden Informationen über das Zielnetzwerk durch Methoden wie Durchsuchen von Facebook Profilen gesammelt. Derartige Aktionen verursachen keinen Netzwerkverkehr, der die Netzwerkgeräte des Unternehmens passiert. Somit sind Aktivitäten dieser Phase bei der Analyse nicht zu detektieren. Die anderen vier Phasen verursachen jedoch beobachtbaren Netzwerkverkehr. In der fünften Phase eines Angriffsszenarios (siehe Abschnitt 4.3) versuchen Angreifer die Spuren ihrer Aktivitäten zu löschen, indem sie Logdateien auf Hosts manipulieren. Der hier vorgestellte Ansatz verwendet ausschließlich flowbasierte Netzwerkdaten und ist somit resistent gegen das Verschleiern von Spuren auf Hosts. Dies wäre nur möglich, wenn der Angreifer die entsprechenden Netzwerkkomponenten direkt manipuliert.

Aus der Analyse typischer Angriffsszenarien und der zugrundeliegenden Daten können folgende Schlussfolgerungen gezogen werden.

(1) Der geringe Informationsinhalt einzelner Flows und die große Anzahl verschiedener Anwendungen macht es nahezu unmöglich zu entscheiden, ob ein einzelner Flow Normalverhalten oder einen Angriff darstellt. Des Weiteren zeichnen sich typische Benutzeraktivitäten und Angriffsszenarien häufig durch Sequenzen von Flows aus. Dies kann ganz einfach anhand eines vertikalen Port Scans erläutert werden. Ein vertikaler Port Scan scant viele bis alle Ports eines Zielsystems [SHM02]. Da der Quell-Port und der Ziel-Port bei der Generierung von Flows berücksichtigt werden, verursacht jeder gescannte Port einen neuen Flow. Ein anderes Beispiel könnte der Aufruf einer Webseite sein. Oftmals binden Webseiten Bilder und Teile anderer Webseiten ein. In solchen Fällen öffnet der Client verschiedene Quell-Ports, um die Bilder nachzuladen oder sendet Anfragen an weitere Web-Server (unterschiedliche Ziel-IP-Adressen), um Teile anderer Webseiten anzufordern. Deshalb ist es sinnvoller Sequenzen von Flows anstelle einzelner Flows zu bewerten. Es gibt jedoch keine allgemeingültige Regel, wie eine sinnvolle Zusammenfassung von Flows auszusehen hat [ŠKŽ15].

(2) Unterschiedliche Angriffsphasen wirken sich unterschiedlich auf flowbasierte Netzwerkdaten aus. In der Scanning-Phase werden verfügbare Dienste und Hosts angefragt, während in der Gaining Access-Phase konkrete Dienste angegriffen werden. In der Maintaining Access-Phase sehen vermutlich Attribute wie Bytes und Pakete (siehe Tabelle 4.1) normal aus, aber der Ursprung (Quell-IP-Adresse) dieser Verbindungen wirkt verdächtig. Aus diesen Gründen ergibt es mehr Sinn, die flowbasierten Netzwerkdaten aus verschiedenen Sichten zu analysieren, um die verschiedene Angriffsphasen zu erkennen.

(3) Clusterverfahren (Abschnitt 3.2), Klassifikatoren (Abschnitt 3.1) und Algorithmen zur Ausreißererkennung basieren in der Regel auf einer Definition von Abstand oder Ähnlichkeit. Eine direkte Anwendung derartiger Algorithmen auf flowbasierte Netzwerkdaten (siehe Tabelle 4.1) kann zu fatalen Problemen führen. Wenn beispielsweise derselbe Angriff zweimal ausgeführt wird, und die zweite Ausführung zu einem späteren Zeitpunkt von einer anderen Quell-IP-Adresse auf einer anderen Ziel-IP-Adresse stattfindet, unterscheiden sich mindestens drei der zehn Attribute eines Flows. Infolgedessen weisen die Flows der beiden Angriffe einen höheren Abstand auf und werden als unähnlich gewertet. Daher ist die Abstrahierung einiger Attribute durch gezielte Vorverarbeitung erforderlich.

(4) Informationen aus flowbasierten Netzwerkdaten sind begrenzt. Deshalb sollten flowbasierte Netzwerkdaten mit verfügbaren Domänenwissen angereichert werden.

10.4. Konzept zur Analyse von Datenströmen

10.4.1. Überblick

Dieser Abschnitt gibt einen Überblick des Konzepts und diskutiert die grundlegenden Ideen. Die Besonderheit des Konzepts im Vergleich zu existierenden Ansätzen ist die spezielle Anpassung auf unterschiedliche Angriffsphasen durch die Einführung mehrerer Perspektiven sowie die Integration von zusätzlichem Domänenwissen. Die vorgesehene Verarbeitungskette ist in Abbildung 10.1 graphisch dargestellt.

Der IP Address Info Filter ist der erste Filter der Verarbeitungskette, dieser erhält den flowbasierten Datenstrom der Netzwerkkomponenten und integriert die verfügbaren Zusatzinformationen über das Netzwerk (siehe Abschnitt 10.4.2). Anschließend wird der Datenstrom zum Service Detection Filter (Abschnitt 10.4.3) weitergeleitet. Dieser Filter versucht den Dienst eines Flows (zum Beispiel SSH, DNS oder HTTP) zu identifizieren. Danach wird der Datenstrom an den Collecting Filter weitergeleitet. Dieser Filter berücksichtigt ebenfalls verfügbares Domänenwissen über das Netzwerk und sammelt pro Host alle Flows im vorgegebenen Zeitfenster. Die Host-Identifizierung findet dabei auf Basis der Quell-IP-Adressen der Flows statt. Der Parameter δ bestimmt die Größe der Zeitfenster in Sekunden. Je größer der Parameter δ, desto mehr Speicherplatz ist notwendig. Gleichzeitig steigt vermutlich die Qualität der Zusammenfassungen mit zunehmender Fenstergröße. Anschließend erstellt der Collecting Filter pro Host und pro Zeitfenster jeweils einen Netzwerk-Datenpunkt, einen Benutzer-Datenpunkt und für jeden identifizierten Dienst einen Service-Datenpunkt. Jeder dieser Datenpunkte wird anschließend von einer speziellen Perspektive, welche an die typischen Phasen eines Angriffsszenarios angelehnt sind, bewertet (siehe Abbildung 10.1). Die Netzwerk-Datenpunkte enthalten spezifische Informationen über das Netzwerkverhalten der Hosts, die Service-Datenpunkte enthalten spezifische Informationen über die Nutzung konkreter Dienste und die Benutzer-Datenpunkte enthalten spezifische Informationen über das Verhalten eines Hosts und sind in Abschnitt 10.4.4 beschrieben. Durch die Kombination der Integration von Domänenwissen und der Zusammenfas-

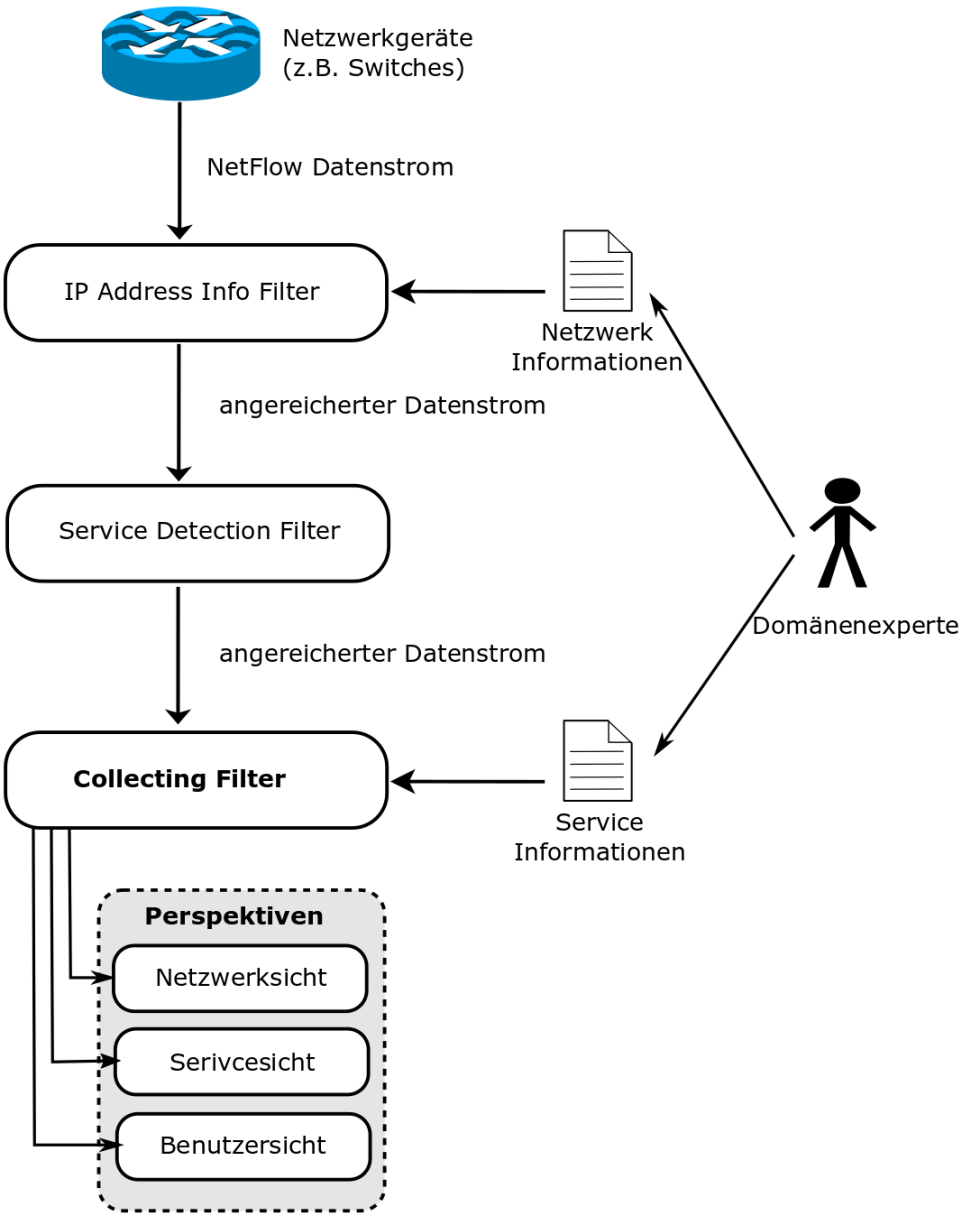


Abbildung 10.1.: Überblick des Konzepts zur Angriffsdetektion nach Ring et al. [RWG+17a].

sung von Flows über Zeitfenster können wertvolle Attribute für nachgelagerte Analysemethoden berechnet werden.

Die einzelnen Filter aus Abbildung 10.1 können leicht parallelisiert und somit zu einer schnelleren Verarbeitung des Datenstroms verhelfen, was der hohen Datenmenge, die im Bereich IT-Sicherheit in Echtzeit verarbeitet werden muss, entgegen kommt.

10.4.2. Integration von Domänenwissen

Die Leistung von Intrusion Detection Methoden lässt sich durch die Integration von zusätzlichem Domänenwissen verbessern. Aus diesem Grund strebt das Konzept die Integration von Zusatzinformationen über IP-Adressen und Ports an. Diese Informationen werden mithilfe des IP Address Info Filters in den Datenstrom eingebracht. Hierfür muss durch den Netzwerkadministrator eine Konfigurationsdatei mit allen IP-Adressbereichen des Unternehmens angelegt werden. Ein Beispiel ist in Abbildung 10.2 zu sehen.

```
#Subnet IP Address, Subnetwork, Organization, isIntern, isServer
192.168.1.0 , 16, General   , 1, 0
192.168.1.0 , 24, Management, 1, 0
192.168.2.0 , 24, Developer , 1, 0
192.168.3.0 , 24, Server    , 1, 0
192.168.3.2 , 32, MailServer, 1, 1
192.168.3.3 , 32, FileServer, 1, 1
192.168.3.4 , 32, WebServer , 1, 1
192.168.2.42, 32, PrinterMan, 1, 1
192.168.3.42, 32, PrinterDev, 1, 1
```

Abbildung 10.2.: Beispielhafte Konfigurationsdatei zur Integration von Domänenwissen [RWG+17a].

Die erste Spalte in Abbildung 10.2 beinhaltet entweder konkrete IP-Adressen oder Subnetze. Die zweite Spalte definiert die Größe des Subnetzes aus Spalte 1, wobei der Wert 32 darauf hindeutet, dass es sich um eine einzelne IP-Adresse handelt. Die dritte Spalte weist jedem Subnetz eine Organisation zu. Die letzten beiden Spalten identifizieren, ob es sich um interne IP-Adressen beziehungsweise Server handelt oder nicht. Normalerweise sind alle Server als intern gekennzeichnet. Externen Servern können auch spezifische Rollen (zum Beispiel Clouddienste oder Server anderer Unternehmen für gemeinsame Projekte) zugewiesen werden. Jede IP-Adresse, die keinem Eintrag der Konfigurationsdatei zugeordnet werden kann, wird als externe IP-Adresse behandelt. Basierend auf diesen Informationen werden den Quell- und Ziel-IP-Adressen der Flows die entsprechenden Organisationen und Flags zugewiesen.

Die zweite Konfigurationsdatei beinhaltet Informationen über die genutzten Ports im Unternehmen. Diese ermöglicht die Einbringung von spezifischen Informationen zu internen Servern. Beispielsweise können hier bestimmte offene Ports wie 22 (SSH) oder 80 (HTTP) konkreten IP-Adressen zugewiesen werden. Der Collecting Filter nutzt diese Konfigurationsdatei zur Identifikation von Clients und Servern.

10.4.3. Service Detection Filter

Die IANA[1] (Internet Assigned Numbers Authority) weist Diensten (zum Beispiel HTTP, SSH, DNS oder FTP) standardisierte Portnummern zu. Der Service Detection Filter identifiziert die zugrundeliegende Dienste der Flows und hängt diese Information an die flowbasierten Netzwerkdaten an. Hierfür wertet der Service Detection Filter die Portnummern der Flows aus und vergleicht diese mit der durch IANA standardisierten Liste.

10.4.4. Analysesichten

In diesem Abschnitt werden die drei Analysesichten (siehe Abbildung 10.1) genauer vorgestellt.

Netzwerksicht. Diese Sicht analysiert die generierten Netzwerk-Datenpunkte des Collecting Filters. Die Netzwerksicht prüft das grundsätzliche Verhalten von Hosts im Netzwerk. Primäres Ziel ist somit die Detektion der Scanning-Phase von Angriffsszenarien (siehe Abschnitt 4.3). Folglich stellen IP-Range Scans und Ports Scans die wesentlichen Angriffsszenarien dar, welche in dieser Sicht erkannt werden sollen.

Die generierten Netzwerk-Datenpunkte beinhalten Attribute, welche für die Detektion von Port Scans angepasst sind. Diese beschreiben etwa die Anzahl und die Art der erstellten Netzwerkverbindungen eines Hosts. In Kapitel 11 wird eine konkrete Realisierung beschrieben, das heißt, die Generierung von Netzwerk-Datenpunkten und deren Analyse.

Servicesicht. Diese Sicht analysiert die generierten Service-Datenpunkte des Collecting Filters. Die Servicesicht prüft, ob die Nutzung der Dienste regelkonform abläuft. Primäres Ziel ist somit die Detektion der Gaining Access-Phase von Angriffsszenarien (siehe Abschnitt 4.3). DoS und SSH-Brute-Force Angriffe stellen Vertreter dieser Phase dar.

Um missbräuchlich genutzte Dienste zu erkennen, müssen Flows über größere Zeiträume für einen Dienst aggregiert werden. Deswegen werden alle Flows pro Host, pro Zeitfenster und pro Dienst (siehe Service Detection Filter in Abschnitt 10.4.3) aggregiert. Basierend auf diesen Aggregationen erstellt der Collecting Filter mehrere Service-Datenpunkte mit speziell angepassten Attributen. Die berechneten Attribute beinhalten Werte wie die Summe der übertragenen Bytes und Pakete, die durchschnittliche Dauer der Flows oder die Anzahl der Flows. Weitere nützliche Attribute wie die Anzahl der offenen Verbindungen oder die Anzahl erfolgreich geschlossener Verbindungen können aus TCP-Flags gewonnen werden. Für die Berechnung der Attribute werden auch zusätzliche Informationen über die Quelle und das Ziel basierend auf dem Domänenwissen aus Abschnitt 10.4.2 genutzt.

In der Literatur existieren etliche Methoden, die sich als Analysemethoden der Servicesicht eignen. Beispielsweise stellen Najafabadi et al. [NKC+15] einen Ansatz zur Detektion von SSH-Brute-Force Angriffen vor. Zunächst aggregieren die Autoren Flows über Zeitfenster und generieren dann spezielle Datenpunkte zur Detektion von SSH-Brute-Force Angriffen. Die generierten Datenpunkte werden mit verschiedenen Klassifikatoren analysiert. SSHCure [HHH+12] wäre eine weitere Methode, die als Analysemethode der Servicesicht verwendet werden könnte.

Benutzersicht. Diese Sicht analysiert die generierten Benutzer-Datenpunkte des Collecting Filters. Die Benutzersicht prüft, ob die Nutzung der Dienste für den jeweiligen Host normal

[1] `http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml`

oder verdächtig ist. Primäres Ziel dieser Sicht ist die Detektion bereits infizierter Hosts, also die Maintaining Access-Phase (siehe Abschnitt 4.3).

Angreifer verfügen über gültige Zugänge (Benutzername und Passwort) für bereits übernommene Hosts. Werden Verbindungen von Angreifern zu bereits übernommenen Hosts gestartet, so scheinen viele Merkmale des beobachtbaren Netzwerkverkehrs normal. Beispielsweise unterscheiden sich die Attribute Bytes, Pakete oder Dauer nicht von legitimen SSH-Verbindungen. Deshalb berücksichtigt die Benutzersicht die Quellen, Ziele und genutzten Dienste der Flows. Hierfür greift das Konzept auf das Domänenwissen aus den Abschnitten 10.4.2 und 10.4.3 zurück.

Kapitel 6 stellt eine umgesetzte Analysemethode der Benutzersicht dar. IP2Vec greift auf Verbindungsinformationen (Ziel-IP-Adresse, Ziel-Port und Transport-Protokoll) zurück und lernt Vektorrepräsentationen für IP-Adressen. Da die Quell-IP-Adressen der Flows zur Host- bzw. Benutzer-Identifizierung genutzt werden können, beinhalten die Vektorrepräsentationen Informationen über das Verhalten der Benutzer. Angenommen, eine Abteilung besteht aus fünf Mitarbeitern und der Rechner eines Mitarbeiters ist mit einem Botnetz infiziert. In diesem Fall wäre davon auszugehen, dass sich die Netzwerkverbindungen der vier nicht infizierten Rechner vom infizierten Rechner unterscheiden. IP2Vec würde diesen Umstand insofern widerspiegeln, dass sich die Vektorrepräsentation des infizierten Rechners stärker von den nicht infizierten Rechnern unterscheidet. In Abschnitt 6.3 (siehe Experiment 1) wurde ein derart gelagertes Experiment durchgeführt. In diesem Experiment wurden Vektorrepräsentationen für IP-Adressen mit IP2Vec erstellt. Einige IP-Adressen waren mit einem Botnetz infiziert. Anschließend wurden die Vektorrepräsentationen mit DBScan bezüglich ihrer Ähnlichkeit geclustert. Beim Clustering konnten alle infizierten IP-Adressen erfolgreich demselben Cluster zugeordnet werden, während die nicht infizierten IP-Adressen anderen bzw. keinem Cluster zugeordnet wurden.

10.5. Zusammenfassung

In diesem Kapitel wurde ein Konzept zur Detektion sicherheitskritischer Ereignisse in Unternehmensnetzwerken vorgestellt. Das Konzept basiert auf flowbasierte Netzwerkdaten, welche einige Herausforderungen mit sich bringen. Unterschiedliche Benutzeraktivitäten auf Betriebssystemebene verursachen unterschiedlich viele Flows auf Netzwerkebene. Deshalb müssen zusammengehörige Flows identifiziert und zusammengefasst werden, um den beobachteten Netzwerkverkehr korrekt bewerten zu können. Des Weiteren hängt die Befugnis bestimmter Aktionen vom Benutzer und vom Zeitpunkt ab. Beispielsweise sind einige Benutzeraktivitäten für gewisse Personen zulässig, währenddessen diese für andere Personen unzulässig sind. Um diese komplexen Herausforderungen anzugehen, verfolgt das Konzept einen mehrstufigen Lösungsansatz. Zunächst reichert das Konzept die Daten mit verfügbaren Domänenwissen an. Danach werden pro Host alle Flows über Zeitfenster aggregiert und darauf aufbauend neue Datenpunkte berechnet. Diese Datenpunkte sollen die Benutzeraktivitäten beschreiben, welche die Flows verursacht haben. Anschließend werden diese Datenpunkte aus drei verschiedenen Perspektiven analysiert, welche an den typischen Hackingphasen angelehnt sind. Die erste Perspektive konzertiert sich auf das allgemeine Netzwerkverhalten von Hosts, die zweite Perspektive auf die konforme Nutzung von Diensten und die dritte Perspektive auf das typische Nutzerverhalten. Durch diese Vorgehensweise wird die Gesamtkomplexität auf verschiedene Perspektiven aufgeteilt. Das Konzept wurde prototypisch im Coburg-Utility-Framework (siehe Abschnitt 3.6) integriert. Im besonderen Interesse dieser Arbeit ist die erste Perspektive, welche die Netzwerksicht fokussiert. Diese wird im nächsten Kapitel beschreiben und erzielte Ergebnisse diskutiert.

11. Detektion der Scanning-Phase

Im vorherigen Kapitel wurde ein Konzept zur Detektion sicherheitskritischer Ereignisse in Unternehmensnetzwerken vorgestellt. Das Konzept sieht die Analyse von flowbasierten Netzwerkdaten aus verschiedenen Perspektiven vor, welche an typische Hackingphasen gekoppelt sind. Dieses Kapitel adressiert ebenfalls die Detektion von Angriffsszenarien (siehe Herausforderung 3 in Abschnitt 1.2.3), indem es einen Teil des Konzepts aus Kapitel 10 realisiert und einen Ansatz zur Detektion von (langsamen) Port Scans präsentiert. Einerseits stellen (langsame) Port Scans häufig Indikatoren für schwerwiegendere Angriffe dar, andererseits ist deren Detektion aufgrund der großen Datenmengen in Unternehmensnetzwerken schwierig. Deshalb wird in diesem Kapitel ein Ansatz für die Vorverarbeitung flowbasierter Netzwerkdaten entwickelt, der speziell auf die Erkennung von (langsamen) Port Scans zugeschnitten ist. Basierend auf dieser Vorverarbeitungskette, welche mit der Generierung von Netzwerk-Datenpunkten (siehe Kapitel 10) endet, werden ein überwachter und ein unüberwachter Algorithmus zur Detektion von Port Scans evaluiert.

Die nachfolgenden Inhalte basieren auf der Publikation *„Detection of Slow Port Scans in Flow-based Network Traffic“* [RLH18].

11.1. Einleitung

Dieses Kapitel zielt auf die frühzeitige Detektion neuartiger Angriffsszenarien. Hierfür wird die Tatsache ausgenutzt, dass Angriffsszenarien oft einer allgemeinen Abfolge von fünf Phasen (siehe Hackingphasen in Abschnitt 4.3) folgen. Skoudis and Liston [SL06] bezeichnen diese Phasen als Reconnaissance, Scanning, Gaining Access, Maintaining Access und Covering Tracks. Dieses Kapitel beschäftigt sich mit der Detektion der Scanning Phase, in der Angreifer die Ziele (Hosts und Netzwerke) ihrer Angriffe identifizieren [DD11].

Problem. Normalerweise erzeugen Port Scans viele Flows in kurzer Zeit auf unterschiedlichen Ports und/oder IP-Adressen. Derartige Port Scans können leicht erkannt werden, indem beispielsweise die Anzahl angefragter Ports pro IP-Adresse oder die Anzahl angefragter IP-Adressen pro Port ausgewertet werden. Gewiefte Angreifer scannen das Zielnetzwerk beziehungsweise den Zielrechner jedoch langsam, um Auffälligkeiten zu vermeiden. Langsam bedeutet in diesem Zusammenhang, dass nicht dauerhaft Netzwerkpakete versendet werden, sondern beispielsweise nur einzelne Netzwerkpakete alle 15 Sekunden oder alle 5 Minuten. Derartige Port Scans werden häufig nicht erkannt, da diese im Netzwerkverkehr eines Unternehmens in der großen Datenmenge untergehen.

Ziel. Port Scans sind typische Methoden der Scanning-Phase und stellen somit einen wesentlichen Bestandteil von Angriffsszenarien dar. Zwar verursachen Port Scans als solche keinen direkten Schaden, sind aber oft ein Indikator für schwerwiegendere Angriffe. Daher zielt dieses Kapitel auf die Detektion von (langsamen) Port Scans zur frühzeitigen Identifikation nachfolgender Angriffsszenarien.

Ansatz und Beiträge. Dieses Kapitel stellt eine neue Methode zur Erkennung von (langsamen) Port Scans in flowbasierten Netzwerkdaten vor und kann somit als Analysemethode der Netzwerksicht des vorherigen Kapitels eingesetzt werden. Grundlegende Idee ist die Anrei-

cherung des flowbasierten Netzwerkverkehrs mit Wissen über das Unternehmensnetzwerk und die Berücksichtigung von Domänenwissen über Port Scans. Hierfür wird eine innovative Vorverarbeitungskette zur Erkennung von (langsamen) Port Scans entwickelt. Die Tatsache, dass ein Flow nur eine Verbindung beschreibt und ein Port Scan mehrere Verbindungen verursacht, wird dadurch gelöst, dass die Flows über Zeitfenster aggregiert werden. Basierend auf diesen Aggregationen werden neue Datenpunkte - welche im Folgenden als Netzwerk-Datenpunkte bezeichnet werden - erstellt. Die Netzwerk-Datenpunkte beinhalten Attribute wie die Anzahl an Flows zu nicht existierenden internen IP-Adressen und stellen gleichzeitig die Basis für zwei Analysealgorithmen zur Detektion der Port Scans dar. Ein Analysealgorithmus ist unüberwacht und basiert auf sequentiellen Hypothesentests, während der andere Algorithmus überwacht ist und auf Klassifikatoren beruht. In der Regel erreichen überwachte Algorithmen bessere Ergebnisse, benötigen aber im Gegensatz zu unüberwachten Algorithmen gelabelte Trainingsdaten. Die beiden Analysealgorithmen werden mit anderen Algorithmen auf dem CIDDS-001 Datensatz (siehe Abschnitt 8.3) verglichen und evaluiert.

Der Hauptbeitrag dieses Kapitels ist die innovative Vorverarbeitungskette für die Erstellung von Netzwerk-Datenpunkten, welche speziell angepasste Attribute zur Detektion von Port Scans besitzen. Basierend auf den erstellten Netzwerk-Datenpunkten werden zwei Ansätze zur Detektion von (langsamen) Port Scans in flowbasierten Netzwerkdaten vorgestellt.

11.2. Verwandte Arbeiten

Dieser Abschnitt fasst eine Literaturrecherche existierender Algorithmen zur Detektion von Port Scans zusammen. Bhuyan et al. [BBK11] geben einen Überblick über das Thema Port Scanning und über Methoden zur Detektion von Port Scans. Die Autoren kommen zu der Erkenntnis, dass die meisten Ansätze paketbasierte Netzwerkdaten verwenden und dass fehlende öffentliche Datensätze die Evaluierung und den Vergleich unterschiedlicher Methoden erschweren. Bou-Harb et al. [BDA14] liefern einen aktuelleren Überblick. Die Autoren kategorisieren Port Scans bezüglich ihrer Natur, Strategie und Ansätze. Des Weiteren werden verschiedene Scanning Methoden erklärt und ein Überblick über existierende Ansätze zur Detektion verteilter Port Scans gegeben. Im Folgenden werden existierende Algorithmen zur Detektion von Port Scans basierend auf den zugrundeliegenden Daten kategorisiert und vorgestellt.

Paketbasierte Ansätze. Ein weit verbreiteter Ansatz zur Detektion von Port Scans auf Paketebene wurde von Staniford et al. [SHM02] veröffentlicht. Die Autoren bestimmen für jedes Netzwerkpaket einen sogenannten Anomaly-Score, der sich aus der Auftrittswahrscheinlichkeit der IP-Adressen und Ports zusammensetzt. Falls der Anomaly-Score einen Schwellwert überschreitet, werden verdächtige Netzwerkpakete zu SPICE (Stealthy Probing and Intrusion Correlation Engine) weitergeleitet. SPICE speichert die verdächtigen Netzwerkpakete in einem Korrelationsgraph ab und verwendet anschließend Schwellwerte, um Port Scanner zu identifizieren.

Ein weiterer paketbasierter Lösungsansatz ist TRW (Threshold Random Walk) [JPB+04]. TRW basiert auf der Annahme, dass Port Scanner mehr fehlgeschlagene TCP-Verbindungen verursachen als normale Clients. Dabei entscheiden die gesetzten TCP-Flags der Netzwerkpakete, ob diese erfolgreich oder fehlgeschlagen sind. Zur Detektion von Port Scans verwendet TRW sequentielle Hypothesentests und definiert zwei Hypothesen H_0 (die Quell-IP-Addresse ist ein normaler Client) und H_1 (die Quell-IP-Adresse ist ein Port Scanner). Nach jedem TCP-Paket wird das Verhältnis zwischen erfolgreichen und fehlgeschlagenen TCP-Verbindungen pro Host

überprüft. Falls ein vorgegebener Schwellwert über- oder unterschritten wird, kann eine der Hypothesen akzeptiert und der Host als normaler Client oder Port Scanner markiert werden.

Die beiden Ansätze von Staniford et al. [SHM02] und Jung et al. [JPB+04] arbeiten auf paketbasierten Daten, wohingegen der vorgestellte Lösungsansatz flowbasierte Netzwerkdaten verwendet.

Flowbasierte Ansätze. Sridharan et al. [SYB06] übertragen den Ansatz sequentieller Hypothesentests von Jung et al. [JPB+04] auf flowbasierte Netzwerkdaten. Die Autoren verzichten jedoch auf eine Definition von erfolgreich und fehlgeschlagen. Stattdessen aggregiert deren Methode TAPS die flowbasierten Netzwerkdaten für jede Quell-IP-Adresse über Zeitfenster und berechnet das Verhältnis aus adressierten Ziel-IP-Adressen und Ziel-Ports. TAPS basiert auf der Annahme, dass das berechnete Verhältnis für Port Scanner entweder sehr niedrige (vertikale Port Scans) oder sehr hohe Werte (horizontale Port Scans) annimmt. Überschreitet das Verhältnis einen vordefinierten Schwellwert, kann die Quell-IP-Adresse als normaler Client oder Port Scanner klassifiziert werden. Des Weiteren haben die Autoren in ihrer Arbeit [SYB06] den paketbasierten Ansatz TRW auf flowbasierte Netzwerkdaten übertragen und bezeichnen diesen als TRW-SYN. TRW-SYN verarbeitet jeden Flow separat und entscheidet mithilfe der gesetzten TCP-Flags und Anzahl übertragener Pakete, ob die Flows erfolgreich oder fehlgeschlagen sind.

TFDS [ZF09] ist ein weiterer Ansatz dieser Kategorie und verarbeitet unidirektionale Flows. TFDS basiert auf der Annahme, dass normaler Netzwerkverkehr eine größere Varianz als ein Port Scan aufweist. Deswegen aggregiert TFDS Flows über Zeitfenster und analysiert die Größe und Varianz der Flows pro Quell-IP-Adresse. Zur Identifikation von Port Scannern verwendet TFDS ebenfalls sequentielle Hypothesentests wie TAPS [SYB06] und TRW [JPB+04]. Zhang und Fang [ZF09] vergleichen ihren Ansatz mit TAPS und erreichen bessere Ergebnisse. Die hohe Anzahl von Fehlalarmierungen ist jedoch eine Schwäche beider Ansätze, TAPS und TFDS.

Gates et al. [GMK+06] nutzen ein Bayessches Regressionsmodell zur Detektion von Port Scans in einer ISP-Umgebung. Zunächst werden die Flows pro Quell-IP-Adresse über Zeitfenster aggregiert und darauf aufbauend neue Datenpunkte mit spezifischen Attributen erstellt. Die Aggregation stoppt, wenn eine Quell-IP-Adresse fünf Minuten lang keinen weiteren Flow verursacht. Diese Annahme mag in der Vergangenheit für ISP eine sinnvolle Vorgehensweise gewesen sein, führt jedoch heutzutage in Unternehmen, wo Clients permanent Synchronisationen verschiedener Dienste (E-Mail-Clients, Netzlaufwerke, ...) durchführen, dazu, dass Flows über sehr große Zeitfenster aggregiert werden. Webster et al. [WGE+15] erweitern den Ansatz von Gates et al. [GMK+06] und evaluieren die Leistung diverser Klassifikatoren zur Detektion von Port Scans in einem Universitätsnetzwerk.

Der in diesem Kapitel entwickelte Ansatz aggregiert Flows über Zeitfenster und führt somit keine Verarbeitung separater Flows wie TRW-SYN [SYB06] durch. Im Vergleich zu Gates et al. [GMK+06] und Webster et al. [WGE+15] wird zusätzliches Domänenwissen über das zugrundeliegende Netzwerk in den Analyseprozess mit eingebunden. Folglich können auch spezifischere Attribute wie die Anzahl adressierter Flows an nicht existierenden internen Ziel-IP-Adressen berechnet werden, welche spezielles Wissen über das Netzwerk und die Hosts voraussetzen.

11.3. Detektionsansätze

In diesem Abschnitt werden zwei Ansätze zur Detektion langsamer Port Scans vorgestellt. Zunächst wird die grundlegende Idee (Abschnitt 11.3.1) erläutert. Danach wird die neuartige Vor-

verarbeitungskette für flowbasierte Netzwerkdaten (Abschnitt 11.3.2) vorgestellt, welche mit der Generierung der Netzwerk-Datenpunkten endet. Im Anschluss werden die beiden Detektionsmethoden UPSD (Unsupervised Port Scan Detection) und SPSD (Supervised Port Scan Detection) detailliert beschrieben.

11.3.1. Grundlegende Idee

Die grundlegende Vorgehensweise ist in Abbildung 11.1 graphisch dargestellt. Zunächst wird der flowbasierte Netzwerkverkehr an zentralen Netzwerkkomponenten wie Switches und Firewalls abgefangen. Im ersten Vorverarbeitungsschritt werden die Flows mit zusätzlichem Wissen angereichert. Konkret wird die Information hinzugefügt, ob es sich bei den IP-Adressen um interne oder externe IP-Adressen handelt. Hierfür muss ein Domänenexperte eine Datei bereitstellen, die Informationen über alle internen IP-Adressbereiche beinhaltet. Diese Aufgabe wird in der Implementierung durch den IP Address Info Filter aus Abschnitt 10.4.2 übernommen.

Im zweiten Vorverarbeitungsschritt werden die Flows über Zeitfenster aggregiert und basierend auf diesen Aggregationen neue Datenpunkte mit speziell angepassten Attributen erstellt. Dadurch kann die Herausforderung angegangen werden, dass sich Ports Scans durch Sequenzen von Flows kennzeichnen. Ergebnis dieses Schritts sind die neu erstellten Netzwerk-Datenpunkte. Diese Aufgabe wird in der Implementierung durch den Collecting Filter aus Abschnitt 10.4.1 übernommen.

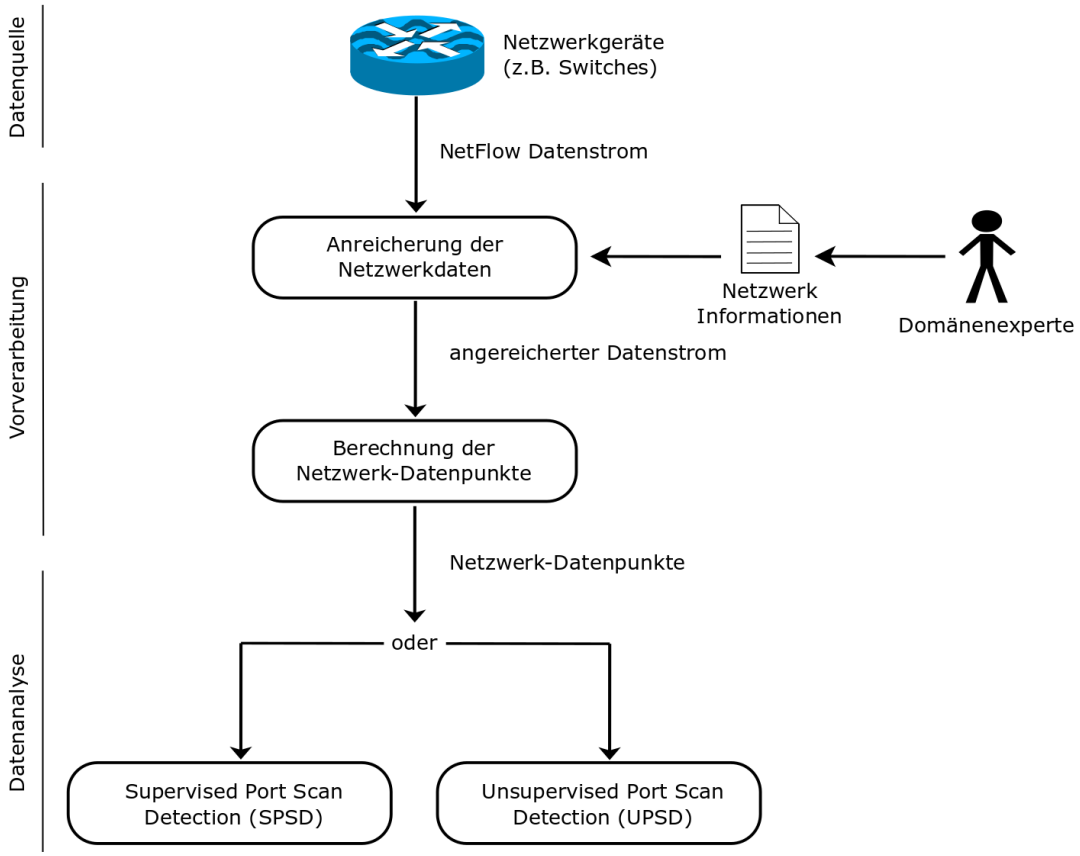


Abbildung 11.1.: Ansätze zur Detektion von (langsamen) Port Scans nach Ring et al. [RLH18].

Nach der Vorverarbeitungskette stehen die zwei Ansätze UPSD und SPSD zur Detektion von (langsamen) Port Scans. UPSD ist ein unüberwachter Ansatz und basiert auf sequentiellen Hypothesentests. SPSD ist ein überwachter Ansatz und basiert auf Klassifikatoren. Beide Ansätze verwenden die erstellten Netzwerk-Datenpunkte als Eingabe und alarmieren gegebenenfalls über Port Scanner.

11.3.2. Vorverarbeitung flowbasierter Netzwerkdaten

Die Vorverarbeitungskette in Abbildung 11.1 endet mit der Generierung von Netzwerk-Datenpunkten, welche anschließend die Eingabe der Analysealgorithmen SPSD und UPSD darstellen. Zunächst werden alle Flows innerhalb eines Zeitfensters von δ Sekunden gesammelt. Basierend auf diesen Sammlungen wird pro Quell-IP-Adresse und pro Zeitfenster ein Netzwerk-Datenpunkt erstellt.

Bei der Erstellung der Netzwerk-Datenpunkte wird die Tatsache berücksichtigt, dass unterschiedliche Port Scanning-Techniken unterschiedlichen Einfluss auf flowbasierten Netzwerkverkehr haben (siehe Abschnitt 4.4.1). Eine genauere Analyse von Port Scans führt zu zwei wesentliche Erkenntnisse. (1) Anfragen an geschlossene Ports sind aufgrund des vordefinierten Verhaltens der Opfer leichter zu erkennen als Anfragen an geöffnete Ports. (2) Angreifer weichen häufig von vordefinierten Protokollrichtlinien ab, um Sicherheitsmechanismen zu umgehen. Deswegen erscheint die Identifikation von Opfern einfacher zu sein, da sich diese an vordefinierte Protokollrichtlinien halten.

Grundsätzlich versuchen Angreifer mithilfe von Port Scans Informationen über offene Ports zu erhalten, die für spätere Angriffe genutzt werden können. Basierend auf dieser Annahme ist es für Angreifer beim Scannen eines Netzwerks wahrscheinlicher, nicht existierende IP-Adressen anzufragen, als für normale Clients. Diese Tatsache wird bei der Erstellung von Netzwerk-Datenpunkten ausgenutzt. Des Weiteren wird das Wissen über die interne Netzstruktur verwendet und alle internen IP-Adressen und offenen TCP-Ports von internen IP-Adressen gespeichert. Diese beiden Listen werden zu Beginn erstellt und mit den eintreffenden Datenstrom kontinuierlich erweitert. Die Liste der internen IP-Adressen wird erweitert, sobald ein neuer Flow mit einer unbekannten internen IP-Adresse im Attribut Quell-IP-Adresse erscheint. Die Liste, welche die Kombination von internen IP-Adressen und offenen TCP-Ports beinhaltet, wird durch die Auswertung beliebiger Kombinationen aus Quell-IP-Adresse und Quell-Port (Berücksichtigung der standardisierten Ports zwischen 0 und 1023) erreicht, bei denen eine gültige TCP-Verbindung beobachtet werden kann. In Tabelle 11.1 ist ein Überblick der berechneten Attribute der Netzwerk-Datenpunkte zu sehen. Im Folgenden wird die eindeutige Kombination von Ziel-IP-Adresse und Ziel-Port als Ziel bezeichnet. Die eindeutige Kombination von Quell-IP-Adresse und Quell-Port wird im Folgenden als Quelle bezeichnet.

Beispiel. Im Folgenden soll die Berechnung der Attribute mithilfe eines Beispiels (siehe Tabelle 11.2 und 11.3) erläutert werden. Tabelle 11.2 gibt einen Überblick über die internen IP-Adressen und offenen TCP-Ports. Tabelle 11.3 zeigt die gesammelten Flows innerhalb eines Zeitfensters. Das erste Attribut eines Netzwerk-Datenpunkts ist die IP-Adresse des Hosts und dient ausschließlich zur Identifikation. Im Beispiel soll ein Netzwerk-Datenpunkt für die IP-Adresse 192.168.220.16 berechnet werden.

In den Attributen #2 and #3 eines Netzwerk-Datenpunkts (siehe Tabelle 11.1) wird das typische Verhalten der Opfer (siehe Abschnitt 4.4.1) berücksichtigt. Das Attribut UR-Zähler zählt die Anzahl der empfangenen ICMP Unreachable Flows und ist ein Indikator für UDP-

Tabelle 11.1.: Aufbau eines Netzwerk-Datenpunktes nach Ring et al. [RLH18].

#	Name	Beschreibung
1	IP	Quell-IP-Adresse
2	UR-Zähler	Anzahl empfangener ICMP Unreachable Nachrichten
3	RST-Zähler	Anzahl empfangener RST Flags von unterschiedlichen Quellen
4	RwA-Zähler	Anzahl adressierter Ziele ohne Antwort
5	NeIP-Zähler	Anzahl Flows an nicht existierenden internen Ziel-IP-Adressen
6	NeTCP-Zähler	Anzahl Flows an nicht existierenden internen TCP-Services
7	Auffälligkeitszähler	Anzahl, wie oft diese Quell-IP-Adresse in Folge einen Wert größer 0 in mindestens einem der Attribute #2 bis #6 hatte

Tabelle 11.2.: Bekannte interne IP-Adressen und bekannte offene TCP-Ports.

Bekannte interne IP-Adressen	Bekannte offene TCP-Ports
192.168.100.5	192.168.100.5 Port 80
192.168.100.6	192.168.100.6 Port 22
192.168.100.15	192.168.100.15 Port 443
192.168.220.16	

Scans. Im Beispiel besitzt dieses Attribut den Wert 2 für die IP-Adresse 192.168.220.16, da diese zwei ICMP Unreachable Nachrichten (siehe Flows #18 und #20 in Tabelle 11.3) empfängt.

Das Attribut RST-Zähler stellt einen Indikator für TCP-Scans dar. Falls ein Port Scanner ein Netzwerkpaket zu einem geschlossenen TCP-Port sendet, antwortet der Empfänger mit einem RST-Flag. Das Attribut RST-Zähler versucht diesen Effekt einzufangen. Hierbei ist jedoch zu beachten, dass RST-Flags auch in anderen Zusammenhängen versendet werden, um beispielsweise offene TCP-Verbindungen zu schließen. Deshalb werden nur empfangene RST-Flags von unterschiedlichen Quellen berücksichtigt, denen eine Anfrage ohne gültige TCP-Verbindung voraus geht. Der Wert des Attributs RST-Zähler ist im Beispiel 2. Dieser Wert ergibt sich aus den Anfragen in den Flows #7, #8 und #9 und den entsprechenden Antworten der Flows #10, #11 und #12. Die Flows #10 und #11 zählen lediglich einfach, da diese von der gleichen Quelle versendet wurden. Weiterhin ist keine Anfrage von der betrachteten IP-Adresse 192.168.220.16 zu der IP-Adresse 8.8.8.8 zu sehen. Deswegen wird das RST-Flag aus Flow #4 nicht berücksichtigt. Die RST-Flags der Flows #5 and #6 gehen ebenfalls nicht in der Zählung mit ein, da für diese Flows eine gültige TCP-Verbindung beobachtet werden kann (siehe Flows #1 und #3).

Oftmals blockieren Firewalls oder andere Sicherheitsmechanismen die Netzwerkpakete der Port Scanner. In diesen Fällen können unidirektionale Flows der Port Scanner zu potentiellen Opfern beobachtet werden, obwohl diese Anfragen niemals ihr Ziel erreichen. Als Folge dessen generieren die potentiellen Opfer keine Antworten und es werden keine entsprechenden unidirektionalen Flows vom Opfer zum Port Scanner aufgezeichnet. Diesen Umstand versucht das Attribut RwA-Zähler darzustellen und zählt die Anzahl der Flows, für die keine entsprechenden Flows in die andere Richtung beobachtet werden können. Im Beispiel tragen die Flows #13, #14, #15, #16, #17 und #19 zur Berechnung bei, wobei die Flows #13, #14, #15 das gleiche Ziel haben und deshalb in Summe nur als 1 gezählt werden. Somit ist der Wert von RwA-Zähler 4.

Die Wahrscheinlichkeit, nicht existierende interne IP-Adressen oder nicht geöffnete TCP-Ports anzufragen, ist für Port Scanner viel höher als für legitime Hosts. Die Attribute #5 und #6 versuchen diese Heuristik abzubilden. NeIP-Zähler zählt die Anzahl von Anfragen an nicht exis-

Tabelle 11.3.: Gesammelte Flows im betrachteten Zeitfenster.

#	Proto	Quell-IP	Quell-Port	Ziel-IP	Ziel-Port	TCP-Flags
1	TCP	192.168.100.5	80	192.168.220.16	53321	.A....
2	TCP	192.168.220.16	53333	192.168.100.5	80	.A..S.
3	TCP	192.168.100.5	80	192.168.220.16	53333	.A..S.
4	TCP	8.8.8.8	80	192.168.220.16	47898	...R..
5	TCP	192.168.100.5	80	192.168.220.16	53333	...R..
6	TCP	192.168.100.5	80	192.168.220.16	53321	...R..
7	TCP	192.168.220.16	53337	192.168.100.5	22	....S.
8	TCP	192.168.220.16	53338	192.168.100.5	22	....S.
9	TCP	192.168.220.16	53339	192.168.100.5	23	....S.
10	TCP	192.168.100.5	22	192.168.220.16	53337	...R..
11	TCP	192.168.100.5	22	192.168.220.16	53338	...R..
12	TCP	192.168.100.5	23	192.168.220.16	53339	...R..
13	TCP	192.168.220.16	53340	192.168.100.15	443	....S.
14	TCP	192.168.220.16	53341	192.168.100.15	443	....S.
15	TCP	192.168.220.16	53342	192.168.100.15	443	....S.
16	TCP	192.168.220.16	53343	192.168.100.15	22	....S.
17	ICMP	192.168.220.16	0	192.168.220.44	8.0	.....
18	ICMP	192.168.220.1	0	192.168.220.16	3.1	.....
19	UDP	192.168.220.16	34345	192.168.100.5	53	.....
20	ICMP	192.168.100.5	0	192.168.220.16	3.3	.....

tierende interne Ziel-IP-Adressen. In diesem Beispiel sind alle IP-Adressen aus dem Subnetz 192.168.X.X intern und somit ist der Wert des Attributs NeIP-Zähler 1 (siehe Flow #17), da die IP-Adresse 192.168.220.44 nicht existiert (siehe Tabelle 11.2).

Des Weiteren werden alle offenen bekannten TCP-Ports für interne IP-Adressen abgespeichert (siehe Tabelle 11.2). Das Attribut NeTCP-Zähler zählt die Anzahl an Flows, die an unterschiedliche Ziele mit nicht als bekannt geöffnete TCP-Ports interner IP-Adressen gerichtet sind. Der Wert beträgt im Beispiel 3 (siehe Flows #7, #8, #9 und #16). Die Flows #7 und #8 adressieren das gleiche Ziel und werden nur einfach gewertet.

Das Attribut Auffälligkeitszähler dient zur Detektion von Port Scans, die sich über längere Zeiträume erstrecken. Die Attribute UR-Zähler, RST-Zähler, RwA-Zähler, NeIP-Zähler und NeTCP-Zähler sind Indikatoren für Port Scans. Falls all diese Attribute den Wert 0 besitzen, wird der Wert des Attributs Auffälligkeitszähler auf 0 gesetzt. Andernfalls wird der Wert des Attributs Auffälligkeitszähler auf $z + 1$ gesetzt, wobei z der Wert des Attributs Auffälligkeitszähler im vorherigen Zeitfenster ist. Folglich enthält das Attribut Auffälligkeitszähler die Information, in wie vielen Zeitfenstern in Folge die Quell-IP-Adresse einen Wert größer 0 in mindestens einer Indikatorvariablen aufweist. Diese Idee beruht darauf, dass fehlerhaft konfigurierte Clients nur kurzzeitig Indikatorvariablen mit Werten größer 0 aufweisen, währenddessen Port Scanner dies über längere Zeiträume tun.

11.3.3. UPSD - Unsupervised Port Scan Detection

Der unüberwachte Ansatz UPSD transferiert die Idee sequentieller Hypothesentests von [JPB+04] auf die im vorherigen Abschnitt erstellten Netzwerk-Datenpunkte.

Sequentielle Hypothesentests aktualisieren eine Likelihood Variable nach jedem eintreffenden Datenpunkt eines Datenstroms, um diesen einer vordefinierten Hypothese H_0 oder H_1 zuzuordnen. Im vorliegenden Szenario besteht der Datenstrom aus Netzwerk-Datenpunkten mit gleicher Quell-IP-Adresse. Dabei beschreibt die Hypothese H_0 normale Hosts und die Hypothese H_1 Port Scanner. Jedem Netzwerk-Datenpunkt i kann eine Indikatorvariable Y_i zugewiesen werden. Zunächst werden in Gleichung 11.1 die zwei möglichen Ausprägungen der Indikatorvariable Y_i, erfolgreich und nicht erfolgreich, definiert.

$$Y_i = \begin{cases} 0 & \text{falls der Netzwerk-Datenpunkt } i \text{ erfolgreich ist} \\ 1 & \text{falls der Netzwerk-Datenpunkt } i \text{ nicht erfolgreich ist} \end{cases} \tag{11.1}$$

Im Folgenden wird definiert, wann Netzwerk-Datenpunkte erfolgreich beziehungsweise nicht erfolgreich sind. Primäres Ziel dieser Arbeit ist die Detektion von Port Scans bei einer möglichst geringen Anzahl von Fehlalarmierungen. Deshalb werden nur starke Anzeichen für Port Scans aus Tabelle 11.1 gewählt. Eine experimentelle Analyse der berechneten Netzwerk-Datenpunkte ergab, dass die Attribute RST-Zähler und RwA-Zähler häufig Werte größer 0 für normalen Netzwerkverkehr annehmen. Ein Grund hierfür ist die Unterteilung des Datenstroms in Zeitfenster und deren Verarbeitung. Angenommen, Host A sendet zu Host B im Zeitfenster t einen unidirektionalen Flow und die Antwort von Host B zu Host A ist erst im nachfolgenden Zeitfenster $t+1$ zu sehen. In diesem Fall würde das Attribut RwA-Zähler 1 für Host A im Zeitfenster t und 1 für Host B im Zeitfenster $t+1$ sein. Aus diesen Gründen berücksichtigt UPSD ausschließlich die Attribute UR-Zähler, NeIP-Zähler und NeTCP-Zähler. UPSD berechnet für diese drei Attribute eines Netzwerk-Datenpunkts i die Summe λ_i. Falls $\lambda_i = 0$, dann ist der Netzwerk-Datenpunkt i erfolgreich. Falls $\lambda_i > 0$, dann ist der Netzwerk-Datenpunkt i nicht erfolgreich. Das Attribut Auffälligkeitszähler wird nicht verwendet, da sequentielle Hypothesentests von Natur aus Sequenzinformationen berücksichtigen.

UPSD verwendet die nachfolgenden Wahrscheinlichkeiten für das Auftreten von erfolgreichen beziehungsweise nicht erfolgreichen Netzwerk-Datenpunkten i, unter der Annahme, dass die Hypothese H_0 beziehungsweise H_1 zutrifft.

$$Pr[Y_i = 0|H_0] = \theta_0 \tag{11.2}$$

$$Pr[Y_i = 1|H_0] = 1 - \theta_0 \tag{11.3}$$

$$Pr[Y_i = 0|H_1] = \theta_1 \tag{11.4}$$

$$Pr[Y_i = 1|H_1] = 1 - \theta_1 \tag{11.5}$$

Die Gleichungen 11.2, 11.3, 11.4 und 11.5 beschreiben die bedingten Wahrscheinlichkeiten, dass ein Netzwerk-Datenpunkt i erfolgreich (beziehungsweise nicht erfolgreich) ist, wenn die Hypothese H_0 (beziehungsweise H_1) zutrifft. Die Parameter θ_0 und θ_1 stellen benutzerdefinierte Eingabeparameter dar. Die Wahrscheinlichkeit erfolgreicher Netzwerk-Datenpunkte sollte für normale Hosts H_0 größer sein als für Port Scanner H_1. Diese Unterschiede müssen laut [JPB+04] für die Hypothese H_0 und H_1 statistisch signifikant sein.

Für jeden Netzwerk-Datenpunkt i wird die Likelihood Variable der entsprechenden Quell-IP-Adresse wie folgt aktualisiert:

$$ratio_i^{quellIP} = ratio_{i-1}^{quellIP} * argmax(\lambda_i, 1) * \frac{Pr[Y_i|H_1]}{Pr[Y_i|H_0]}, \tag{11.6}$$

wobei $ratio_{i-1}^{quellIP}$ der Wert der Likelihood Variable im vorherigen Zeitfenster ist und Y_i die Indikatorvariable des Netzwerk-Datenpunkts i darstellt. Die Variable λ_i ist die Summe der Attribute UR-Zähler, NeIP-Zähler und NeTCP-Zähler des aktuellen Netzwerk-Datenpunkts i. Nach jeder Aktualisierung wird das neue Verhältnis $ratio_i^{quellIP}$ mit zwei benutzerdefinierten Schwellwerten η_0 und η_1 verglichen. Falls $ratio_i^{quellIP}$ einen der beiden Schwellwerte überschreitet (beziehungsweise unterschreitet), wird die Quell-IP-Adresse mit dem entsprechenden Klassenlabel versehen und der Wert des Verhältnisses auf 1 zurückgesetzt. Falls $ratio_i^{quellIP} < \eta_0$, dann wird die Hypothese H_0 (normaler Host) angenommen und der Quell-IP-Adresse das Label *normal* zugewiesen. Falls $ratio_i^{quellIP} > \eta_1$, dann wird die Hypothese H_1 (Port Scanner) angenommen, der Quell-IP-Adresse das Label *attacker* zugewiesen und ein Alarm ausgelöst. Falls $\eta_0 < ratio_i^{quellIP} < \eta_1$, dann kann keine Hypothese angenommen werden und der entsprechende Netzwerk-Datenpunkt wird der Klasse *normal* zugewiesen, da noch kein hinreichender Verdacht besteht, dass ein Port Scan vorliegt. Der Wert für $ratio_i^{quellIP}$ liegt stets im Intervall $(0, \infty)$. Das Verhältnis $\frac{Pr[Y_i|H_1]}{Pr[Y_i|H_0]}$ sollte stets für erfolgreiche Netzwerk-Datenpunkte kleiner 1 und für nicht erfolgreiche Netzwerk-Datenpunkte größer 1 sein. Deshalb sollten die Werte für η_0 im Intervall $(0, 1)$ und für η_1 im Intervall $(1, \infty)$ liegen. Der resultierende Algorithmus von UPSD ist in Algorithmus 3 dargestellt.

```
1  UPSD(S, θ0, θ1, η0, η1)
2  // Eingabeparameter:
3  //   S - Datenstrom bestehend aus Netzwerk-Datenpunkten
4  //   θ0, θ1, η0, η1 - Benutzerdefinierte Parameter
5  quellIPRatios // HashMap zum Speichern der aktuellen Verhältnisse
6  for event ∈ S do
7  |   ip = event.quellIP
8  |   if ip ∉ quellIPRatios then
9  |   |   quellIPRatios.add(ip, 1.0)
10 |   end
11 |   ratio = quellIPRatios.get(ip)
12 |   λi = event.UR_Zähler + event.NeIP_Zähler + event.NeTCP_Zähler
13 |   if λi > 0 then
14 |   |   ratio = ratio * λi * (1−θ1)/(1−θ0)
15 |   else
16 |   |   ratio = ratio * 1 * θ1/θ0
17 |   end
18 |   quellIPRatios.put(ip, ratio)
19 |   if ratio > η1 then
20 |   |   markiere ip als Port Scanner
21 |   |   quellIPRatios.put(ip, 1.0)
22 |   end
23 |   if ratio < η0 then
24 |   |   markiere ip als normal
25 |   |   quellIPRatios.put(ip, 1.0)
26 |   end
27 end
```

Algorithmus 3: Unüberwachter Algorithmus UPSD nach Ring et al. [RLH18].

11.3.4. SPSD - Supervised Port Scan Detection

SPSD ist eine überwachte Methode und verwendet Klassifikatoren zur Detektion von (langsamen) Port Scans. Klassifikatoren erlernen ein Modell und sind anschließend in der Lage, neue Datenpunkte in vordefinierte Klassen einzuordnen (siehe Abschnitt 3.1). Im Rahmen dieser Arbeit werden für SPSD zwei unterschiedliche Klassifikatoren, Entscheidungsbäume (siehe Abschnitt 3.1.2) und SVMs (siehe Abschnitt 3.1.3), evaluiert. Beide Klassifikatoren bringen unterschiedliche Vorteile mit sich. Die von Entscheidungsbäumen erzeugten Modelle können einfach interpretiert werden, da jeder Pfad vom Wurzelknoten zu einem Blattknoten eine Klassifikationsregel darstellt. Im Vergleich dazu können SVMs nicht-lineare Zusammenhänge besser erfassen und erzielen oftmals höhere Klassifikationsgenauigkeiten.

Für SPSD sind Netzwerk-Datenpunkte die zu klassifizierten Datenpunkte, welche in die Klassen *normal* und *attacker* eingeordnet werden sollen. Zur Klassifikation werden die Attribute #2, #3, #4, #5, #6 und #7 aus Tabelle 11.1 verwendet. Durch das Attribut Auffälligkeitszähler (siehe Attribut #7) versucht SPSD die zeitliche Komponente zu erhalten und die Werte vorheriger Zeitfenster zu berücksichtigen. Falls SPSD einen Netzwerk-Datenpunkt der Klasse *attacker* zuordnet, wird ein Alarm ausgelöst, dass die zugehörige Quell-IP-Adresse ein Port Scanner ist.

11.4. Experimente und Diskussion

In diesem Abschnitt findet eine experimentelle Evaluierung der vorgestellten Ansätze UPSD und SPSD auf dem CIDDS-001 Datensatz [RWG+17c] (siehe Abschnitt 8.3) statt. Als Baseline werden TRW-SYN [SYB06], TFDS [ZF09] und Webster et al. [WGE+15] verwendet.

11.4.1. Datensatz

Zur Evaluierung wird der CIDDS-001 Datensatz (siehe Abschnitt 8.3) verwendet. Der Datensatz wurde über einen Zeitraum von vier Wochen aufgenommen und besteht aus zwei Teilen. Der eine Teil beinhaltet den aufgezeichneten Netzwerkverkehr aus der OpenStack-Umgebung und der andere Teil beinhaltet den aufgezeichneten Netzwerkverkehr des externen Servers. In dieser Evaluierung wird nur der Netzwerkverkehr aus der OpenStack-Umgebung verwendet, da dieser keine als *unknown* gelabelte Flows beinhaltet. Des Weiteren beinhalten die Wochen 1 und 2 diverse Angriffsszenarien, während die Wochen 3 und 4 frei von Angriffsdaten sind. Deshalb werden in den nachfolgenden Experimenten nur die ersten beiden Wochen berücksichtigt und als $Woche_1$ und $Woche_2$ bezeichnet.

Es werden folgenden Anpassungen vorgenommen. Zunächst werden alle Angriffe außer Port Scans aus dem Datensatz entfernt. Der verbleibende Datensatz ist in drei Klassen (*attacker*, *victim* und *normal*) unterteilt. Da der Fokus dieser Arbeit auf der Detektion von Angreifern liegt, werden alle als *victim* gelabelte Flows der Klasse *normal* zugeordnet. Tabelle 11.4 gibt einen Überblick der durchgeführten Port Scans in den selektierten Daten.

Im CIDDS-001 Datensatz wurden die Port Scans mithilfe des Tools Nmap ausgeführt. Insgesamt wurden 26 Port Scans in $Woche_1$ und 14 Port Scans in $Woche_2$ durchgeführt (siehe Tabelle 11.4). Der Parameter T steuert im Tool Nmap die Geschwindigkeit der Port Scans. Je größer der Wert von T, desto schneller führt das Tool den Scan durch. Beispielsweise führen die Werte $T = 1$ und $T = 2$ dazu, dass alle 15 beziehungsweise alle 0,4 Sekunden eine Anfrage zu einem Port versendet wird. Für den Wert $T = 3$ versendet Nmap dauerhaft Anfragen zu Ports und es existiert kein fester Zeitabstand zwischen den gesendeten Paketen [Lyo08]. Im nachfol-

Tabelle 11.4.: Port Scans im CIDDS-001 Datensatz. Der Parameter T gibt die Geschwindigkeit der durchgeführten Scans an. Je größer der Wert von T, desto größer die Ausführungsgeschwindigkeit.

Parameter	$Woche_1$	$Woche_2$
T=1	7	6
T=2	10	7
T=3	9	1
Summe	26	14

genden Experiment werden durchgeführte Port Scans mit dem Parameter $T = 1$ als langsam bezeichnet.

11.4.2. Evaluationsstrategie

TRW-SYN verarbeitet jeden Flow separat und weist somit jedem Flow ein Label zu. Die Methoden TFDS, Webster, UPSD und SPSD aggregieren Flows über Zeitfenster und weisen der kompletten Kollektion ein Label zu. Diese Vorgehensweise kann dazu führen, dass einige Kollektionen Flows von beiden Klassen (*attacker* und *normal*) beinhalten. Im Folgenden werden diese Kollektionen als gemischte Kollektionen bezeichnet. Gemischte Kollektionen können nicht eindeutig einer Klasse zugewiesen werden. In dieser Arbeit werden alle gemischten Kollektionen mit der Klasse *attacker* gelabelt, da es das Ziel von Angreifern sein kann, den Scanning Prozess unter normalen Benutzeranfragen zu tarnen.

Deshalb werden als Evaluationskriterien die Anzahl erkannter Port Scans und die Anzahl von Fehlalarmierungen (False Positives) herangezogen. Es wird auf die Klassifikationsgenauigkeit als Evaluationsmaß verzichtet, da die gemischten Kollektionen die Ergebnisse verzerren könnten, da die verschiedenen Ansätze keine oder unterschiedlich große Kollektionen erzeugen.

11.4.3. Parameterkonfigurationen

Dieser Abschnitt beschreibt die Parameterkonfigurationen der verschiedenen Methoden. Des Weiteren sollte berücksichtigt werden, dass SPSD und Webster überwachte Methoden sind, welche gelabelte Daten in der Trainingsphase benötigen. Im Vergleich dazu sind TRW-SYN, TFDS und UPSD unüberwachte Methoden und benötigen keine gelabelten Trainingsdaten.

11.4.3.1. TRW-SYN

TRW-SYN benötigt vier benutzerdefinierte Eingabeparameter und wird mit zwei unterschiedlichen Parameterkonfigurationen getestet. Zunächst werden die empfohlenen Werte aus Sridharan et al. [SYB06] übernommen. Diese Konfiguration wird als TRW-SYN (default) bezeichnet: $\eta_0 = 0{,}01$, $\eta_1 = 99$, $\theta_0 = 0{,}8$ und $\theta_1 = 0{,}2$.

Danach werden die Parameter auf $Woche_1$ des CIDDS-001 Datensatzes optimiert und für $Woche_2$ übernommen. Diese Konfiguration wird als TRW-SYN (optimiert) bezeichnet: $\eta_0 = 0{,}0001$, $\eta_1 = 99999$, $\theta_0 = 0{,}6$ und $\theta_1 = 0{,}3$.

11.4.3.2. TFDS

Die Methode TFDS benötigt sieben benutzerdefinierte Eingabeparameter und wird ebenfalls mit zwei unterschiedlichen Parameterkonfigurationen evaluiert. Zunächst werden die empfohlenen Werte aus Zhang und Fang [ZF09] übernommen. Diese Konfiguration wird als TFDS (default) bezeichnet: $\eta_0 = \frac{1}{256}$, $\eta_1 = 256$, $\theta_0 = 0{,}8$, $\theta_1 = 0{,}2$, $c = 10$, $\alpha = 0{,}1$ und $t = 20$.

Danach werden die Parameter auf $Woche_1$ des CIDDS-001 Datensatzes optimiert und für $Woche_2$ übernommen. Diese Konfiguration wird als TFDS (optimiert) bezeichnet: $\eta_0 = \frac{1}{10000}$, $\eta_1 = 10000$, $\theta_0 = 0{,}75$, $\theta_1 = 0{,}25$, $c = 10$, $\alpha = 0{,}15$ und $t = 40$.

11.4.3.3. UPSD

UPSD erstellt pro Quell-IP-Adresse und pro Zeitfenster einen Netzwerk-Datenpunkt. Die Länge des Zeitfensters δ wird auf *60* Sekunden gesetzt. Des Weiteren benötigt UPSD wie TRW-SYN vier benutzerdefinierte Eingabeparameter. Hierfür werden die Parameter auf $Woche_1$ des CIDDS-001 Datensatzes optimiert und für $Woche_2$ übernommen. Die Parameter werden wie folgt gesetzt: $\eta_0 = 0{,}01$, $\eta_1 = 99$, $\theta_0 = 0{,}8$ und $\theta_1 = 0{,}2$.

11.4.3.4. SPSD

Für SPSD wird wie für UPSD die Fenstergröße auf $\delta = 60$ Sekunden gesetzt. SPSD basiert auf Entscheidungsbäumen oder SVMs. Da Klassifikatoren in der Trainingsphase gelabelte Daten benötigen, werden die Klassifikatoren auf $Woche_1$ trainiert, um $Woche_2$ zu klassifizieren beziehungsweise auf $Woche_2$ trainiert, um $Woche_1$ zu klassifizieren. Alle kontinuierlichen Attribute werden auf das Intervall $[0, 1]$ normiert.

Im Rahmen dieser Arbeit wurden die Implementierungen der Klassifikatoren aus der scikit-learn[1] Bibliothek Version 0.19.2 verwendet. Beide Klassifikatoren besitzen konfigurierbare Parameter. Für SVM wird der *RBF* Kernel gewählt und die Parameter $C = 10000$ und $\gamma = 1{,}0$ (SPSD mit SVM) gesetzt. Für Entscheidungsbäume wird der *GINI Index* als Splitkriterium gewählt (SPSD mit Entscheidungsbaum).

11.4.3.5. Webster

Webster et al. [WGE+15] aggregieren alle Flows pro Quell-IP-Adresse und erstellen basierend auf diesen Zusammenfassungen neue Datenpunkte mit angepassten Attributen. Die Autoren extrahieren Attribute mithilfe eines Feature Selektion Algorithmus und diskretisieren alle kontinuierlichen Attribute. Basierend auf dieser Vorverarbeitung evaluieren die Autoren verschiedene Klassifikatoren.

Im Folgenden wird dieser Ansatz übernommen und es werden die gleichen Attribute wie in [WGE+15] berechnet. Basierend auf diesen Datenpunkten wird der Random Forest Klassifikator verwendet, da dieser die besten Ergebnisse in Webster et al. [WGE+15] erzielt. Konkret wurde die Random Forest Implementierung aus der Bibliothek scikit-learn[2] Version 0.19.2 verwendet. Dieser Ansatz wird als Webster bezeichnet und der Parameter lower bound auf 1 gesetzt. Der Parameter lower bound beschreibt in Webster et al. [WGE+15] die Mindestanzahl von Datenpunkten, die eine Quell-IP-Adresse verursachen muss, damit diese vom Klassifikator klassifiziert und nicht verworfen werden. Es wird jedoch eine Anpassung bei der Evaluierung vorgenommen. Webster et al. [WGE+15] aggregieren die Flows pro Quell-IP-Adresse über den

[1] https://scikit-learn.org/stable/
[2] https://scikit-learn.org/stable/

kompletten Datensatz. Da die in diesem Kapitel angeführten Ansätze die Detektion von Port Scans in Datenströmen anstreben, ist die Sammlung der Flows über den kompletten Datensatz ungeeignet. Deshalb wird der Datenstrom wie bei SPSD und UPSD in Zeitfenster von 60 Sekunden unterteilt und pro Zeitfenster und pro Quell-IP-Adresse ein Datenpunkt berechnet. Wie bei SPSD wird der Random Forest Klassifikator auf $Woche_1$ trainiert, um $Woche_2$ zu klassifizieren und umgekehrt.

11.4.4. Ergebnisse

Ein Überblick der Ergebnisse ist in Tabelle 11.5 zu sehen. Die Tabelle enthält die Anzahl detektierter Angriffe sowie die Anzahl der Fehlalarmierungen. TFDS erkannte die wenigsten Port Scans. Durch Parameteroptimierung konnte die Anzahl erkannter Port Scans für $Woche_1$ auf 9 und für $Woche_2$ auf 4 gesteigert und gleichzeitig die Anzahl von Fehlalarmierungen reduziert werden. TRW-SYN detektiert mehr Port Scans, generiert jedoch eine höhere Anzahl von Fehlalarmierungen als TFDS. Durch Parameteroptimierung konnte vor allem die große Anzahl von Fehlalarmierungen für TRW-SYN reduziert werden. Es ist jedoch festzuhalten, dass TRW-SYN die meisten Fehlalarmierungen generiert.

UPSD detektiert alle Port Scans für $Woche_1$ und $Woche_2$ und generiert dabei nur wenige Fehlalarmierungen. SPSD mit SVM konnte ebenfalls alle Port Scans für beide Wochen detektieren, ohne eine einzige Fehlalarmierung auszulösen. Die überwachte Methode Webster erkannte ebenfalls alle Port Scans für $Woche_1$ und $Woche_2$, generierte jedoch 22 beziehungsweise 6 Fehlalarmierungen.

Tabelle 11.5.: Anzahl erkannter Port Scans und Fehlalarmierungen. Woche1 beinhaltet 26 und Woche2 beinhaltet 14 Port Scans.

Methode	$Woche_1$		$Woche_2$	
	Erkannte Port Scans	Fehlalarmierungen	Erkannte Port Scans	Fehlalarmierungen
TFDS (default)	6	35	2	16
TFDS (optimiert)	9	29	4	8
TRW-SYN (default)	15	1685	8	7495
TRW-SYN (optimiert)	14	78	7	118
UPSD	**26**	1	**14**	2
SPSD mit Entscheidungsbaum	**26**	**0**	**14**	8
SPSD mit SVM	**26**	**0**	**14**	**0**
Webster	**26**	22	**14**	6

Einen detaillierteren Überblick erkannter Port Scans liefert Tabelle 11.6, die zeigt, dass TFDS Schwierigkeiten bei der Detektion langsamer Port Scans (siehe Spalte $T = 1$) hat. TRW-SYN kann hingegen Port Scans mit unterschiedlich durchgeführten Geschwindigkeiten erkennen.

11.4.5. Diskussion

Die überwachte Methode SPSD erreicht die besten Ergebnisse, siehe Tabelle 11.5. SPSD mit SVM detektiert alle Port Scans und weist gleichzeitig die niedrigste Rate von Fehlalarmierungen auf. Für SPSD wurden Entscheidungsbäume und SVMs verwendet. Beide Klassifikatoren haben

Tabelle 11.6.: Detaillierte Übersicht detektierter Port Scans. Der Parameter T steuert die Geschwindigkeit eines Port Scans.

Methode	$Woche_1$			$Woche_2$		
	T1	T2	T3	T1	T2	T3
TFDS (default)	0	0	6	1	0	1
TFDS (optimiert)	0	4	5	0	3	1
TRW-SYN (default)	4	5	6	3	4	1
TRW-SYN (optimiert)	3	5	6	2	4	1
UPSD	**7**	**10**	**9**	**6**	**7**	**1**
SPSD mit Entscheidungsbaum	**7**	**10**	**9**	**6**	**7**	**1**
SPSD mit SVM	**7**	**10**	**9**	**6**	**7**	**1**
Webster	**7**	**10**	**9**	**6**	**7**	**1**
vorhandene Port Scans	7	10	9	6	7	1

ihre Vorteile. Während SVMs die besseren Ergebnisse erzielten (0 Fehlalarmierungen), erstellen Entscheidungsbäume für Menschen interpretierbare Regeln.

Die Methode Webster detektiert ebenfalls alle Port Scans für $Woche_1$ und $Woche_2$, generiert jedoch eine größere Anzahl von Fehlalarmierungen als SPSD und UPSD (siehe Tabelle 11.5).

Die Methode TRW-SYN ist auf die Detektion von TCP-SYN Scans beschränkt, was eine Erklärung für die geringe Anzahl erkannter Angriffe ist, da im CIDDS-001 Datensatz auch andere Scan Techniken ausgeführt wurden. Gleichzeitig konnte TRW-SYN auch langsam durchgeführte Port Scans detektieren, siehe Spalte $T = 1$ in Tabelle 11.6. Der CIDDS-001 Datensatz wurde in einer virtuellen Umgebung mit virtuellen Switch (OpenVSwitch) aufgezeichnet. OpenVSwitch verwendet kürzere Timeouts als physikalische Netzwerkkomponenten bei der Erstellung von flowbasierten Netzwerkdaten. Die kürzeren Timeouts könnten für TRW-SYN dazu führen, dass mehr Verbindungen als fehlgeschlagen interpretiert werden. Deswegen wurde bei der Parameteroptimierung versucht, die Auswirkungen von fehlgeschlagenen Verbindungen durch Erhöhung des Parameters η_1 zu reduzieren (siehe TRW-SYN (optimiert)). Diese Optimierung führte zu einer erheblichen Reduzierung der Fehlalarmierungen, jedoch sank auch die Anzahl erkannter Port Scans in $Woche_1$ und $Woche_2$ um jeweils einen Port Scan.

Des Weiteren kann Tabelle 11.5 entnommen werden, dass TFDS die geringste Erkennungsrate aufweist. Eine genauere Analyse zeigt, dass TFDS weder in der Lage ist, langsame Port Scans zu detektieren, noch Port Scans, bei denen die Angreifer versuchen ihre Spuren zu verwischen, wenn diese beispielsweise gleichzeitig reguläre Aktivitäten wie Browsen durchführen. Dies wird in Tabelle 11.6 bestätigt, da TFDS Schwierigkeiten bei der Detektion langsamer (siehe Spalte $T = 1$) und mittlerer Port Scans (siehe Spalte $T = 2$) hat. Generell können die Angreifer im CIDDS-001 Datensatz reguläre Aktivitäten parallel zu Angriffen ausführen. Diese Tatsache führt dazu, dass sich langsam ausgeführte Port Scans häufiger mit regulären Aktivitäten vermischen. Für schnell ausgeführte Port Scans ($T = 3$) erreicht TFDS höhere Erkennungsraten.

Die unüberwachte Methode UPSD erreicht das zweitbeste Ergebnis. UPSD konnte alle Port Scans in $Woche_1$ und $Woche_2$ detektieren und generierte nur 1 beziehungsweise 2 Fehlalarmierungen. Somit erreicht UPSD ähnlich gute Ergebnisse wie SPSD mit SVM, benötigt aber keine gelabelten Trainingsdaten. Im Grunde ähnelt UPSD den Methoden TRW-SYN und TFDS, da alle drei Ansätze sequentielle Hypothesentests verwenden. Der Hauptunterschied dieser Methoden liegt in der Definition erfolgreicher und nicht erfolgreicher Netzwerk-Datenpunkte. Während

TRW-SYN und TFDS hierfür nur auf allgemeine Attribute aus flowbasierten Netzwerkverkehr zurückgreifen, berücksichtigt UPSD auch Domänenwissen über die Netzwerkstruktur.

Insgesamt betrachtet erreicht SPSD mit SVM die beste Erkennungsrate gepaart mit der niedrigsten Rate von Fehlalarmierungen. Des Weiteren können SPSD und UPSD verschiedene Arten von Port Scans erkennen und sind nicht auf einzelne Techniken wie TRW-SYN beschränkt. SPSD benötigt jedoch wie Webster korrekt gelabelte Trainingsdaten, um ein Klassifikationsmodell zu erlernen. Derartige Daten stehen jedoch nicht immer zur Verfügung. Im Gegensatz dazu erreicht UPSD ähnlich gute Ergebnisse und benötigt keine gelabelten Trainingsdaten. Stattdessen wird UPSD mithilfe von benutzerdefinierten Parametern konfiguriert. Das gleiche gilt für TRW-SYN und TFDS. Jedoch erzielen diese beiden Ansätze schlechtere Ergebnisse in den untersuchten Evaluierungskriterien als UPSD.

Speziell die Ergebnisse in Tabelle 11.6 zeigen, dass die vier Methoden TRW-SYN, UPSD, SPSD und Webster in der Lage sind, auch langsam durchgeführte Port Scans zu detektieren. Jedoch generiert Webster eine höhere Anzahl von Fehlalarmierungen und TRW-SYN ist auf die Detektion von TCP-SYN Scans beschränkt. TFDS war nicht in der Lage, langsam durchgeführte Port Scans zu erkennen. Somit konnten in der experimentellen Evaluierung lediglich die zwei Methoden UPSD und SPSD langsam durchgeführte Ports Scans mit hoher Genauigkeit detektieren.

11.5. Zusammenfassung

Angriffsszenarien folgen häufig vordefinierten Abläufen. In der Scanning-Phase versuchen Angreifer mithilfe von Port Scans Informationen über das Zielsystem zu erhalten. Deshalb können Port Scans als Indikatoren für zukünftige Angriffsszenarien interpretiert werden. Aufgrund der großen Datenmengen in Unternehmensnetzwerken gestaltet sich jedoch die Detektion von langsamen Port Scans als Herausforderung.

In diesem Kapitel wurden zwei Ansätze zur Detektion von (langsamen) Port Scans auf flowbasierten Netzwerkdaten vorgestellt. Diese Ansätze können als Analysemethoden der Netzwerksicht des vorherigen Kapitels eingesetzt werden. Beide Ansätze verwenden die gleiche neuartige Vorverarbeitungskette für flowbasierte Netzwerkdaten, was gleichzeitig die Hauptinnovation dieser Ansätze darstellt. Die Vorverbreitungskette reichert zunächst die flowbasierten Netzwerkdaten mit zusätzlichen Wissen über die Netzwerkstruktur an und aggregiert anschließend die Flows über Zeitfenster. Die über Zeitfenster gesammelten Flows werden dann genutzt, um pro Zeitfenster und pro Quell-IP-Adresse einen sogenannten Netzwerk-Datenpunkt zu erstellen. Die Attribute der Netzwerk-Datenpunkte wurden unter Berücksichtigung von Domänenwissen entwickelt und beschreiben beobachtbare Eigenschaften von Port Scans in flowbasierten Netzwerkdaten. UPSD und SPSD verwenden diese Netzwerk-Datenpunkte als Eingabe. UPSD basiert auf sequentiellen Hypothesentests und ist eine unüberwachte Methode. SPSD basiert auf Klassifikatoren und ist eine überwachte Methode. Während SPSD gelabelte Trainingsdaten benötigt, müssen für UPSD mehrere benutzerdefinierte Parameter konfiguriert werden. Durch die Erstellung und Verarbeitung von Netzwerk-Datenpunkten reduzieren beide Ansätze die Datenmenge, was zu geringeren Analyseaufwänden führt. UPSD und SPSD wurden experimentell auf dem CIDDS-001 Datensatz evaluiert. Die Ergebnisse deuten auf deren Eignung hin, da beide Ansätze sehr hohe Erkennungsraten bei geringen Anzahlen von Fehlalarmierungen erreichen und die Ergebnisse anderer etablierter Methoden übertreffen.

Die Erweiterung der Ansätze UPSD und SPSD zur Detektion verteilter Port Scans, also wenn die Anfragen nicht nur von einem, sondern mehreren Angreifern erzeugt werden, stellen

Anknüpfungspunkte für zukünftige Arbeiten dar. Teil II und Teil III dieser Arbeit beschäftigten sich mit methodischen Verfahren zur Verarbeitung heterogener Daten und zur Generierung von Netzwerkdaten. In diesem Teil wurden Data Mining-Methoden in der Anwendungsdomäne IT-Sicherheit eingesetzt, indem ein Konzept zur Detektion sicherheitskritischer Ereignisse in Netzwerkdaten entworfen und erfolgreich zur Detektion von Port Scans umgesetzt wurde.

Teil V.

Zusammenfassung und Ausblick

12. Zusammenfassung

IT-Sicherheit ist ein essentielles Thema in der Informatik. Auf der einen Seite werden immer mehr persönliche Daten in Clouds gespeichert und Arbeitsabläufe durch Industrie 4.0 digitalisiert. Auf der anderen Seite wächst die Anzahl neuartiger Angriffe auf IT-Infrastrukturen täglich an. Da digital gespeicherte Informationen sensible Daten beinhalten und das Wissen eines Unternehmens repräsentieren, müssen diese vor unbefugten Zugriffen und Manipulationen geschützt werden.

Diese Arbeit zielt auf die Detektion sicherheitskritischer Ereignisse in Unternehmensnetzwerken auf Basis flowbasierter Netzwerkdaten mittels Data Mining. Basierend auf dieser Zielstellung wurden die Beschaffenheit der Daten, der Mangel an öffentlich verfügbaren Datensätzen und die eigentliche Detektion sicherheitskritischer Ereignisse als wesentliche Herausforderungen identifiziert. Im Folgenden werden die erarbeiteten Beiträge anhand dieser drei Herausforderungen zusammengefasst.

Beschaffenheit der Daten. Flowbasierte Netzwerkdaten enthalten kontinuierliche Attribute wie Anzahl übertragener Pakete und kategorische Attribute wie IP-Adressen. Zur Handhabung heterogener Daten wurde ein Abstandsmaß ConDist sowie eine Methode zur Transformation kategorischer Werte in kontinuierliche Vektoren namens IP2Vec vorgestellt.

Das Abstandsmaß ConDist dient zur Berechnung von Abständen zwischen Objekten mit heterogenen Attributen (Kapitel 5). ConDist stellt ein allgemein einsetzbares Abstandsmaß dar und ist nicht auf Daten aus dem Bereich IT-Sicherheit beschränkt. Zur Abstandsberechnung zwischen kategorischen Werten greift ConDist auf Informationen aus korrelierten Kontextattributen zurück. ConDist berücksichtigt nicht nur die Präsenz von Korrelationen zwischen Attributen, sondern auch deren Korrelationsgrad und lässt diesen über einen Gewichtungsfaktor in die Abstandsberechnung mit einfließen. Es wurde der Beweis erbracht, dass ConDist eine Metrik ist und eine Erweiterung vorgestellt, die ConDist frei von benutzerdefinierten Parametern macht. In einer experimentellen Evaluierung wurde ConDist erfolgreich in den Bereichen Klassifikation und Clustering eingesetzt. Dabei erzielte ConDist bessere Ergebnisse als frühere korrelationsbasierte Abstandsmaße. Somit konnten die Ergebnisse von Klassifikatoren und Clusterverfahren bei der Verarbeitung von Datensätzen mit kategorischen Attributen durch die Verwendung des Abstandsmaßes ConDist verbessert werden. Diese Verbesserung konnte in den Bereichen Klassifikation und Clusteranalyse gezeigt, die statistische Signifikanz konnte jedoch nur im Klassifikationsszenario erreicht werden.

Weiterhin wurde IP2Vec vorgestellt, eine Methode zur Transformation von IP-Adressen in Vektorrepräsentationen, welche Informationen über das Netzwerkverhalten der IP-Adressen widerspiegeln (Kapitel 6). Eine Besonderheit von IP2Vec im Vergleich zu existierenden Ähnlichkeitsmaßen für IP-Adressen ist, dass IP2Vec die Ähnlichkeiten basierend auf dem tatsächlichen Verhalten der Hosts und nicht anhand von geographischen Koordinaten oder anderen Strukturen berechnet. Die grundsätzliche Eignung von IP2Vec wurde in zwei verschiedenen Szenarien experimentell bestätigt, nämlich der Trennung von normalen und infizierten Hosts sowie der Trennung von Servern und Clients. Mithilfe von IP2Vec konnten bessere Ergebnisse bei der

Clusteranalyse und Visualisierung von IP-Adressen im Vergleich zu herkömmlichen Ansätzen erzielt werden. IP2Vec konnte beispielsweise mit Botnetzen infizierte IP-Adressen eine höhere gegenseitige Ähnlichkeit zuweisen, sodass diese IP-Adressen in einer Clusteranalyse dem gleichen Cluster zugeordnet wurden und in der Visualisierung hohe Ähnlichkeiten zueinander aufwiesen.

Generierung realistischer Netzwerkdaten. Aus Sicht von Unternehmen beinhalten flowbasierte Netzwerkdaten schützenswerte Informationen. Beispielsweise könnten Hacker anhand flowbasierter Netzwerkdaten Rückschlüsse auf Unternehmensnetzwerke und verfügbare Dienste ziehen. Deshalb existieren nur wenig öffentlich verfügbare und gelabelte flowbasierte Netzwerkdatensätze. Im Rahmen dieser Arbeit wurde zunächst eine Literaturrecherche existierender Datensätze und Netzwerkdaten-Generatoren für netzwerkbasierte Daten durchgeführt (siehe Kapitel 7). Basierend auf dieser Analyse wurden Schlussfolgerungen für die Erstellung neuer Datensätze diskutiert und zwei Ansätze zur Generierung flowbasierter Netzwerkdaten entwickelt.

Der erste Ansatz basiert auf einer virtuellen Simulationsumgebung in der Unternehmensnetzwerke nachgebaut und typisches Benutzerverhalten simuliert werden kann (Kapitel 8). Der Ansatz sieht die Simulation typischer Benutzeraktivitäten und verschiedener Angriffsszenarien durch Python Skripte vor. Dadurch ermöglicht die Simulationsumgebung im Vergleich zu realen Produktivumgebungen auch die gezielte Ausführung von Angriffsszenarien. Des Weiteren können kontinuierlich neue Benutzeraktivitäten, Angriffsszenarien und Verbesserungen integriert und auf spezifische Szenarien angepasste Datensätze generiert werden. Die Simulationsumgebung wurde zur Generierung der Intrusion Detection Datensätze CIDDS-001 und CIDDS-002 verwendet. Für beide Datensätze wurde das Netzwerk eines kleinen Unternehmens simuliert und der Netzwerkverkehr für 4 beziehungsweise 2 Wochen aufgezeichnet, mit Labels versehen und analysiert. Beide Datensätze wurden in anonymisierter Form mit zusätzlichen Informationen veröffentlicht. Durch die entwickelte Simulationsumgebung können gelabelte Datensätze aufgezeichnet und somit die Herausforderung mangelnder verfügbarer Datensätze gelöst werden.

Oftmals können die Ergebnisse von tiefen neuronalen Netzen oder anderer Algorithmen mithilfe größerer Trainingsdatensätze verbessert werden. An dieser Stelle setzt der zweite Ansatz an und stellt eine Methode zur Anreicherung existierender Datensätze mit synthetisch erzeugten Flows vor (Kapitel 9). Diese Methode verwendet Wasserstein Generative Adversarial Networks (WGANs). Da WGANs ausschließlich kontinuierliche Attribute verarbeiten können und flowbasierte Netzwerkdaten auch kategorische Attribute beinhalten, wurden drei verschiedene Transformationsansätze eingeführt. Alle drei Ansätze werden mit aufgezeichneten Netzwerkdaten trainiert, erlernen die zugrundeliegende Verteilung der Netzwerkdaten und sind anschließend in der Lage, neue Netzwerkdaten mit gleichen Eigenschaften zu generieren. Somit ist dieser Ansatz nicht als Alternative zur Simulationsumgebung, sondern als Ergänzung zu bewerten, da die Ansätze zur Anreicherung existierender Datensätze verwendet werden können. Die drei Ansätze wurden mithilfe des CIDDS-001 Datensatzes evaluiert. Da die Evaluation generativer Modelle selbst keine trivial Aufgabe ist, wurden verschiedene Evaluationen durchgeführt und eine neue Methode eingeführt, welche auf die Verwendung von Domänenwissen zurückgreift und explizite Tests formuliert. Die Ergebnisse der Evaluierungen zeigen, dass zwei der drei entworfenen Ansätze (B-WGAN-GP und E-WGAN-GP) Flows von sehr hoher Qualität erzeugen und sich zur Anreicherung existierender Datensätze eignen können.

Detektion von Angriffsszenarien. Teil IV dieser Arbeit beschäftigte sich mit der Detektion von Angriffsszenarien. Um die Komplexität dieser Herausforderung anzugehen, wurde zunächst ein Konzept zur Detektion von Angriffsszenarien mittels Data Mining-Algorithmen vorgestellt

(Kapitel 10). Das Konzept analysiert Netzwerkdaten im weit verbreiteten flowbasierten Format und zielt auf die Integration von zusätzlichem Domänenwissen. Da die Befugnis bestimmter Aktionen vom Benutzer und Zeitpunkt abhängen, werden die Netzwerkdaten über Zeitfenster aggregiert und pro Zeitfenster mehrere Datenpunkte berechnet. Diese Datenpunkte bewerten den Netzwerkverkehr aus drei verschiedenen Perspektiven und sind auf typische Hackingphasen angepasst. Die erste Perspektive konzertiert sich auf das allgemeine Netzwerkverhalten von Hosts und dient zur Detektion der Scanning-Phase. Die zweite Perspektive analysiert sich auf die konforme Nutzung von Diensten und dient zur Detektion der Gaining Access-Phase. Die dritte Perspektive untersucht das typische Nutzerverhalten und dient zur Detektion bereits infiltrierter Hosts (Maintaining Access-Phase). Die Methode IP2Vec steht im Einklang mit der dritten Perspektive und konnte erfolgreich zur Identifikation infizierter Rechner (Botnetze) getestet werden. Durch diese Vorgehensweise wird die Gesamtkomplexität reduziert und auf verschiedene Perspektiven verteilt.

Zuletzt wurde ein Teil des Konzepts realisiert, der sich auf die Netzwerksicht fokussiert und zwei Ansätze zur Detektion von (langsamen) Port Scans vorstellt (Kapitel 11). Die beiden Ansätze UPSD und SPSD verwenden die gleiche neuartige Vorverarbeitungskette, welche den flowbasierten Datenstrom mit Zusatzinformationen anreichert und sogenante Netzwerk-Datenpunkte erstellt, die speziell auf die Detektion von Port Scans ausgelegt sind. UPSD ist eine unüberwachte Methode und basiert auf sequentiellen Hypothesentests. SPSD ist eine überwachte Methode und basiert auf Klassifikatoren. UPSD und SPSD wurden experimentell auf dem CIDDS-001 Datensatz evaluiert und mit existierenden Methoden verglichen. Die beiden Methoden erreichen sehr hohe Erkennungsraten bei geringen Anzahlen von Fehlalarmierungen und übertreffen die Ergebnisse existierender Methoden, was für deren Eignung spricht. Da Port Scans häufig Indikatoren für schwerwiegendere Angriffe darstellen, können die beiden vorgestellten Methoden somit die frühzeitige Detektion von Angriffsszenarien unterstützen.

13. Ausblick

In dieser Dissertation wurden mehrere Beiträge entwickelt, die zu einem Fortschritt im aktuellen Stand der Technik geführt haben. Neben Beiträgen zur Detektion von Angriffsszenarien wurden auch grundlegende Themen wie die Berechnung von Abständen zwischen heterogenen Daten oder die Generierung realistischer Testdaten behandelt. In diesem Abschnitt wird auf mögliche Anknüpfungspunkte eingegangen.

Generierung realistischer Testdaten ist ein wichtiger Aspekt für die Entwicklung von IDS. Die entwickelte Simulationsumgebung in OpenStack stellt eine Basis zur Erstellung gelabelter Netzwerkdaten dar. Da sich Normalverhalten und Angriffsszenarien im Lauf der Zeit verändern, müssen auch die Skripte zur Simulation von Benutzerverhalten stetig aktualisiert und erweitert werden. Das automatisierte Lernen sinnvoller Konfigurationsparameter für die Skripte stellt einen interessanten Ansatzpunkt für zukünftige Arbeiten dar. Des Weiteren wurde ein auf Modellierung basierter Ansatz zur Generierung flowbasierter Netzwerkdaten vorgestellt. Dieser Ansatz basiert auf WGANs und wurde bisher lediglich zur Generierung einzelner Flows verwendet. Die Generierung von Flowsequenzen stellt eine offene Fragestellung für zukünftige Arbeiten dar. Ein naheliegender Ansatzpunkt wäre die Verwendung von LSTMs in den WGANs.

Das erarbeitete Konzept zur Detektion sicherheitskritischer Ereignisse wurde für die Detektion von Port Scans umgesetzt, was der Detektion der Scanning-Phase entspricht. Naheliegende Erweiterungen sind die Entwicklung von Angriffsdetektoren für die anderen beiden Perspektiven, die sich auf die konforme Nutzung der Dienste und das typische Nutzerverhalten konzentrieren. Zur Auswertung der Nutzersicht bietet sich die entwickelte Methode IP2Vec an, da IP2Vec für die Berechnung der Vektorrepräsentationen das tatsächliche Benutzerverhalten der Hosts berücksichtigt. Des Weiteren bieten sich Methoden aus dem Bereich Adversarial Learning zur Detektion von Angriffsszenarien an. Derartige Methoden erzeugen durch die Generierung synthetischer Beispiele größere Varianzen in den Trainingsdaten und können somit zu besseren Erkennungsraten von IT-Sicherheitssystemen führen. Beispielsweise könnten GANs in Kombination mit realen IDS trainiert werden, um neuartige Angriffsszenarien zu erzeugen, die von aktuellen Sicherheitssystemen nicht detektiert werden.

Neben den in dieser Arbeit fokussierten Herausforderungen stellen die Themen Visualisierung und Erklärbarkeit weitere relevante Forschungsgebiete im Bereich IT-Sicherheit dar. Die Interpretation von tiefen neuronalen Netzen ist eine offene Forschungsfrage [AKM18]. Beiträge, die zur Erklärung von Fehlentscheidungen durch neuronale Netze in für Menschen verständliche Hinweise führen, würden die Akzeptanz dieser Verfahren wesentlich erhöhen. Beispielsweise verwenden Marino et al. [MWM18] einen auf Adversarial Learning basierten Ansatz, um die Ursachen von Fehlentscheidungen zu identifizieren.

In dieser Arbeit wurden verschiedene Methoden und Konzepte zur Berechnung von Abständen zwischen heterogenen Daten, zur Generierung realistischer Testdaten und zur Detektion von Angriffsszenarien entwickelt. Während einige Beiträge auf Domänenwissen der Anwendungsdomäne IT-Sicherheit zurückgreifen und somit ein spezielles Anwendungsgebiet besitzen, stellen andere Beiträge generell einsetzbare Methoden dar, die auch in anderen Anwendungsgebieten nutzbar sind. Beispielsweise kann das heterogene Abstandsmaß ConDist ohne weitere Anpassungen auf beliebige Datensätze mit kategorischen Attributen angewendet werden.

Anhang

A Notationen

ConDist Notationen (Kapitel 5)

Abkürzung	Bedeutung
α	Höhe des Signifikanzniveaus vom Wilcoxon-Signed-Ranks-Test
Θ	Benutzerdefinierter Schwellwert zur Selektion korrelierter Kontextattribute
$\Theta_{X\|Y}$	Automatischer Schwellwert für das Kontextattribut Y für Zielattribut X
$context_X$	Menge der Kontextattribute für das Zielattribut X
$cor(X\|Y)$	Korrelationsfunktion zwischen den Attributen X und Y
c	Konstante mit dem Wert $\frac{8}{27}$
D	Datensatz / Datenmenge
$d_{X,max}$	Maximaler Abstand zwischen den Datenpunkten im Attribut X
$\hat{d}_X(\boldsymbol{O_i}, \boldsymbol{O_j})$	Abstand zwischen den Datenpunkten $\boldsymbol{O_i}$ und $\boldsymbol{O_j}$ im Attribut X
$d_X(\boldsymbol{O_i}, \boldsymbol{O_j})$	Normierter Abstand zwischen den Datenpunkten $\boldsymbol{O_i}$ und $\boldsymbol{O_j}$ im Attribut X
$H(X)$	Entropie des Attributs X
$H(X\|Y)$	Bedingte Entropie des Attributs X gegeben Attribut Y
$IG(X\|Y)$	Informationsgewinn von X gegeben Y
$impact_X(Y)$	Wert der Einflussfunktion des Kontextattributs Y auf das Zielattribut X
k	Anzahl nächster Nachbarn für kNN-Klassifikator
m	Dimensionalität eines Datenpunktes
n	Anzahl Datenpunkte eines Datensatzes D
$\boldsymbol{O_i}$	Beliebiger Datenpunkt aus der Datenmenge D
$\boldsymbol{O_j}$	Beliebiger Datenpunkt aus der Datenmenge D
o_{i_X}	Wert des Datenpunktes $\boldsymbol{O_i}$ im Attribut X
ρ	p-Wert zur Ermittlung der Signifikanz
$p(x)$	Wahrscheinlichkeit des Wertes x
$p(Y = y\|o_{i_X})$	Wahrscheinlichkeit des Wertes y in Attribut Y gegeben Wert o_{i_X} in Attribut X
$P(Y\|X = o_{j_X})$	Wahrscheinlichkeitsverteilung des Attributs Y gegeben Wert o_{j_X} in Attribut X
X	Beliebiges Attribut aus dem Datensatz D
Y	Beliebiges Attribut aus dem Datensatz D
x	Beliebiger Wert im Attribut X, $x \in dom(X)$
$w(X)$	Gewichtungsfunktion für das Attribut X
W	W-Wert zur Ermittlung der Signifikanz für den Wilcoxon-Signed-Ranks-Test

IP2Vec Notationen (Kapitel 6)

Abkürzung	Bedeutung
ϵ	Größe der ϵ-Umgebung in DBScan
$minPts$	Definiert die Mindestanzahl von Datenpunkten in einer ϵ-Umgebung, sodass diese in DBScan als dicht bezeichnet wird
m	Dimensionalität der Vektorrepräsentation beziehungsweise des Embeddings
V	Vokabular
w_1	1-Komponente der Vektordarstellung einer IP-Adresse
w_2	2-Komponente der Vektordarstellung einer IP-Adresse
w_m	m-Komponente der Vektordarstellung einer IP-Adresse

Notationen zur Simulation netzwerkbasierter Daten (Kapitel 8)

Abkürzung	Bedeutung
T	Parameter zur Konfiguration der Geschwindigkeit eines Port Scans

Notationen zur Modellierung netzwerkbasierter Daten (Kapitel 9)

Abkürzung	Bedeutung
A	Beliebiger Benutzer
B	Beliebiger Benutzer
b_1, b_2, b_k	Ausprägungen im Attribut Bytes
D	Diskriminator-Netz
G	Generator-Netz
m	Dimensionalität einer Vektorrepräsentation
z	Eingabevektor von Rauschen

Notationen zum Konzept zur Detektion sicherheitkritischer Ereignisse (Kapitel 10)

Abkürzung	Bedeutung
δ	Zeitfenster zur Aggregation von Flows in Sekunden

UPSD und SPDP Notationen (Kapitel 11)

Abkürzung	Bedeutung
λ	Summe der Attribute UR-Zähler, NeIP-Zähler und NeTCP-Zähler in einem Zeitfenster t
γ	Parameter des RBF-Kernels einer SVM
δ	Zeitfenster zur Aggregation von Flows in Sekunden
η_0	Benutzerdefinierter Schwellwert zur Annahme der Hypothese H_0
η_1	Benutzerdefinierter Schwellwert zur Annahme der Hypothese H_1
Θ_0	Benutzerdefinierter Parameter für die Wahrscheinlichkeit erfolgreicher Netzwerk-Datenpunkte unter der Bedingung, dass der Netzwerk-Datenpunkt von einem normalen Host verursacht wurde
Θ_1	Benutzerdefinierter Parameter für die Wahrscheinlichkeit erfolgreicher Netzwerk-Datenpunkte unter der Bedingung, dass der Netzwerk-Datenpunkt von einem Port-Scanner verursacht wurde
A	Beliebiger Host
B	Beliebiger Host
C	Parameter des RBF-Kernels einer SVM
H_0	Hypothese H_0 (es handelt sich bei der IP-Adresse um einen normalen Client)
H_1	Hypothese H_1 (es handelt sich bei der IP-Adresse um einen Scanner)
S	Datenstrom bestehend aus Netzwerk-Datenpunkten
t	Ein beliebiges Zeitfenster im Datenstrom S
T	Parameter zur Konfiguration der Geschwindigkeit eines Port Scans
Y_t	Indikatorvariable für Netzwerk-Datenpunkte
z	Auffälligkeitszähler

Literaturverzeichnis

[AAH+16] Alkasassbeh, Mouhammd; Al-Naymat, Ghazi; Hassanat, Ahmad; Almseidin, Mohammad: Detecting Distributed Denial of Service Attacks Using Data Mining Techniques. In: *International Journal of Advanced Computer Science and Applications (IJACSA)* 7 (2016), Nr. 1. http://dx.doi.org/10.14569/IJACSA.2016.070159

[ABE04] Amor, Nahla B.; Benferhat, Salem; Elouedi, Zied: Naive Bayes vs Decision Trees in Intrusion Detection Systems. In: *ACM Symposium on Applied Computing* ACM, 2004, S. 420–424. http://dx.doi.org/10.1145/967900.967989

[ABK+99] Ankerst, Mihael; Breunig, Markus M.; Kriegel, Hans-Peter; Sander, Jörg: OPTICS: Ordering Points To Identify the Clustering Structure. In: *ACM Sigmod Record* 28 (1999), Nr. 2, S. 49–60. http://dx.doi.org/10.1145/304181.304187

[ACB17] Arjovsky, Martin; Chintala, Soumith; Bottou, Léon: Wasserstein GAN. In: *arXiv preprint arXiv:1701.07875* (2017)

[ACW+05] Au, Wai-Ho; Chan, Keith C.; Wong, Andrew K.; Wang, Yang: Attribute Clustering for Grouping, Selection, and Classification of Gene Expression Data. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* 2 (2005), Nr. 2, S. 83–101. http://dx.doi.org/10.1109/TCBB.2005.17

[AD07] Ahmad, Amir; Dey, Lipika: A method to compute distance between two categorical values of same attribute in unsupervised learning for categorical data set. In: *Pattern Recognition Letters* 28 (2007), Nr. 1, S. 110–118. http://dx.doi.org/10.1016/j.patrec.2006.06.006

[AG11] Ashoor, Asmaa S.; Gore, Sharad: Importance of Intrusion Detection System (IDS). In: *International Journal of Scientific and Engineering Research* 2 (2011), Nr. 1, S. 1–4

[AG15] Aissa, Naila B.; Guerroumi, Mohamed: A Genetic Clustering Technique for Anomaly-based Intrusion Detection Systems. In: *IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/-Distributed Computing (SNPD)* IEEE, 2015, S. 81–86. http://dx.doi.org/10.1109/SNPD.2015.7176182

[AGG+98] Agrawal, Rakesh; Gehrke, Johannes; Gunopulos, Dimitrios; Raghavan, Prabhakar: Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In: *International Conference on Management of Data*, ACM Press, 1998, S. 94–105. http://dx.doi.org/10.1145/276304.276314

[AH11] Aviv, Adam J.; Haeberlen, Andreas: Challenges in Experimenting with Botnet Detection Systems. In: *Conference on Cyber Security Experimentation and Test (CEST)*. Berkeley, CA, USA : USENIX Association, 2011, S. 6–6

[AHW+03] Aggarwal, Charu C.; Han, Jiawei; Wang, Jianyong; Yu, Philip S.: A Framework for Clustering Evolving Data Streams. In: *International Conference on Very Large Data Bases (VLDB)*, Morgan Kaufmann, 2003, S. 81–92

[AKM18] Amarasinghe, Kasun; Kenney, Kevin; Manic, Milos: Toward Explainable Deep Neural Network based Anomaly Detection. In: *IEEE International Conference on Human System Interaction (HSI)* IEEE, 2018, S. 311–317. http://dx.doi.org/10.1109/HSI.2018.8430788

[AS94] Agrawal, Rakesh; Srikant, Ramakrishnan: Fast Algorithms for Mining Association Rules. In: *International Conference on Verly Large Data Bases (VLDB)* Bd. 1215, 1994, S. 487–499

[ASN14] Alamuri, Madhavi; Surampudi, Bapi R.; Negi, Atul: A Survey of Distance/-Similarity Measures for Categorical Data. In: *International Joint Conference on Neural Networks (IJCNN)* IEEE, 2014, S. 1907–1914. http://dx.doi.org/10.1109/IJCNN.2014.6889941

[Axe99] Axelsson, Stefan: The Base-Rate Fallacy and its Implications for the Difficulty of Intrusion Detection. In: *ACM Conference on Computer and Communications Security* ACM, 1999, S. 1–7. http://dx.doi.org/10.1145/319709.319710

[AYL+11] Amiri, Fatemeh; Yousefi, MohammadMahdi R.; Lucas, Caro; Shakery, Azadeh; Yazdani, Nasser: Mutual information-based feature selection for intrusion detection systems. In: *Journal of Network and Computer Applications* 34 (2011), Nr. 4, S. 1184–1199. http://dx.doi.org/10.1016/j.jnca.2011.01.002

[AZZ+17] Anwar, Shahid; Zain, Jasni M.; Zolkipli, Mohamad F.; Inayat, Zakira; Khan, Suleman; Anthony, Bokolo; Chang, Victor: From Intrusion Detection to an Intrusion Response System: Fundamentals, Requirements, and Future Directions. In: *Algorithms* 10 (2017), Nr. 2, S. 39. http://dx.doi.org/10.3390/a10020039

[BBK11] Bhuyan, Monowar H.; Bhattacharyya, DK; Kalita, Jugal K.: Surveying Port Scans and Their Detection Methodologies. In: *The Computer Journal* 54 (2011), Nr. 10, S. 1565–1581. http://dx.doi.org/10.1093/comjnl/bxr035

[BBK14] Bhuyan, Monowar H.; Bhattacharyya, Dhruba K.; Kalita, Jugal K.: Network Anomaly Detection: Methods, Systems and Tools. In: *IEEE Communications Surveys and Tutorials* 16 (2014), Nr. 1, S. 303–336. http://dx.doi.org/10.1109/SURV.2013.052213.00046

[BBK15] Bhuyan, Monowar H.; Bhattacharyya, Dhruba K.; Kalita, Jugal K.: Towards Generating Real-life Datasets for Network Intrusion Detection. In: *IJ Network Security* 17 (2015), Nr. 6, S. 683–701

[BBR+12] Bilge, Leyla; Balzarotti, Davide; Robertson, William; Kirda, Engin; Kruegel, Christopher: Disclosure: Detecting Botnet Command and Control Servers Through Large-scale NetFlow Analysis. In: *Annual Computer Security Applications Conference* ACM, 2012, S. 129–138. http://dx.doi.org/10.1145/2420950.2420969

[BCC+18] Beek, Christiaan; Castillo, Carlos; Cochin, Cedric; Dolezal, Ashley; Grobman, Steve; McFarland, Charles; Minihane, Niamh; Palm, Chris; Peterson, Eric; Povolny, Steve; Samani, Raj; Schmugar, Craig; Sims, ReseAnne; Sommer, Dan; Sun, Bing: McAfee Labs Threats-Report / McAfee Labs. Version: September 2018. https://www.mcafee.com/enterprise/de-de/assets/reports/rp-quarterly-threats-sep-2018.pdf. McAfee, September 2018. – Forschungsbericht. – Letzter Zugriff am 5 August 2020

[BCK08] Boriah, Shyam; Chandola, Varun; Kumar, Vipin: Similarity Measures for Categorical Data: A Comparative Evaluation. In: *SIAM International Conference on Data Mining*, 2008, S. 243–254. http://dx.doi.org/10.1137/1.9781611972788.22

[BCM+16] Bontemps, Loïc; Cao, Van L.; McDermott, James; Le-Khac, Nhien-An: Collective Anomaly Detection Based on Long Short-Term Memory Recurrent Neural Networks. In: *International Conference on Future Data and Security Engineering* Springer, 2016, S. 141–152. http://dx.doi.org/10.1007/978-3-319-48057-2_9

[BDA14] Bou-Harb, Elias; Debbabi, Mourad; Assi, Chadi: Cyber Scanning: A Comprehensive Survey. In: *IEEE Communications Surveys and Tutorials* 16 (2014), Nr. 3, S. 1496–1519. http://dx.doi.org/10.1109/SURV.2013.102913.00020

[BDK14] Baroni, Marco; Dinu, Georgiana; Kruszewski, Germán: Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In: *Annual Meeting of the Association for Computational Linguistics*, 2014, S. 238–247. http://dx.doi.org/10.3115/v1/P14-1023

[BDP12] Botta, Alessio; Dainotti, Alberto; Pescapé, Antonio: A tool for the generation of realistic network workload for emerging networking scenarios. In: *Computer Networks* 56 (2012), Nr. 15, S. 3531–3547. http://dx.doi.org/10.1016/j.comnet.2012.02.019

[BFO+84] Breiman, Leo; Friedman, Jerome; Olshen, Richard; Stone, Charles: *Classification and Regression Trees*. 1. Auflage. Taylor and Francis, 1984. – ISBN 0412048418

[BG16] Buczak, Anna L.; Guven, Erhan: A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. In: *IEEE Communications Surveys and Tutorials* 18 (2016), Nr. 2, S. 1153–1176. http://dx.doi.org/10.1109/COMST.2015.2494502

[BGV92] Boser, Bernhard E.; Guyon, Isabelle M.; Vapnik, Vladimir N.: A Training Algorithm for Optimal Margin Classifiers. In: *Workshop on Computational Learning Theory* ACM, 1992, S. 144–152. http://dx.doi.org/10.1145/130385.130401

[BGVI+19] Bridges, Robert A.; Glass-Vanderlan, Tarrah R.; Iannacone, Michael D.; Vincent, Maria S.; Chen, Qian: A Survey of Intrusion Detection Systems Leveraging Host Data. In: *ACM Computing Surveys* 52 (2019), Nr. 6. http://dx.doi.org/10.1145/3344382

[BHK+17] Beer, Frank; Hofer, Tim; Karimi, David; Bühler, Ulrich: A new Attack Composition for Network Security. In: *10. DFN-Forum Kommunikationstechnologien* Gesellschaft für Informatik eV, 2017, S. 11–20

[Bis06] Bishop, Christopher M.: *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 1. Auflage. Berlin, Heidelberg : Springer-Verlag, 2006. – ISBN 0387310738

[BJS+14] Beigi, Elaheh B.; Jazi, Hossein H.; Stakhanova, Natalia; Ghorbani, Ali A.: Towards Effective Feature Selection in Machine Learning-Based Botnet Detection Approaches. In: *IEEE Conference on Communications and Network Security (CNS)* IEEE, 2014, S. 247–255. http://dx.doi.org/10.1109/CNS.2014.6997492

[BK13] Bhattacharyya, Dhruba K.; Kalita, Jugal K.: *Network Anomaly Detection: A Machine Learning Perspective*. 1. Auflage. CRC Press, 2013. – ISBN 978–1466582088

[BKN+00] Breunig, Markus M.; Kriegel, Hans-Peter; Ng, Raymond T.; Sander, Jörg: LOF: Identifying Density-based Local Outliers. In: *ACM SIGMOD International Conference on Management of Data*, ACM, 2000, S. 93–104. http://dx.doi.org/10.1145/342009.335388

[BL17] Buduma, Nikhil; Locascio, Nicholas: *Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms*. 1. Auflage. O'Reilly Media, 2017. – ISBN 9781491925614

[BM15] Bath, Graham; McKay, Judy: *Praxiswissen Softwaretest-Test Analyst und Technical Test Analyst: Aus-und Weiterbildung zum Certified Tester-Advanced Level nach ISTQB-Standard*. 3. Auflage. dpunkt, 2015. – ISBN 3864901375

[BM17] Bamler, Robert; Mandt, Stephan: Dynamic Word Embeddings. In: *International Conference on Machine Learning* JMLR, 2017, S. 380–389

[BMR+96] Buschmann, Frank; Meunier, Regine; Rohnert, Hans; Sommerlad, Peter; Stal, Michael: *Pattern-Oriented Software Architecture - Volume 1: A System of Patterns*. 1. Auflage. Wiley Publishing, 1996. – ISBN 9780471958697

[Bor19] Borji, Ali: Pros and cons of GAN evaluation measures. In: *Computer Vision and Image Understanding* 179 (2019), S. 41–65. http://dx.doi.org/10.1016/j.cviu.2018.10.009

[Bri18] Briegleb, Volker: *Facebook-Hack: Sensible Daten von Millionen Nutzern entwendet*. https://www.heise.de/newsticker/meldung/Facebook-Hack-Sensible-Daten-von-Millionen-Nutzern-entwendet-4190206.html. Version: 12.10.2018. – (Letzter Zugriff am 5 August 2020)

[BS14] Bhattacharya, Sangeeta; Selvakumar, S: SSENet-2014 Dataset: A Dataset for Detection of Multiconnection Attacks. In: *International Conference on Eco-friendly Computing and Communication Systems (ICECCS)* IEEE, 2014, S. 121–126. http://dx.doi.org/10.1109/Eco-friendly.2014.100

[BTD+16] Bojarski, Mariusz; Testa, Davide d.; Dworakowski, Daniel; Firner, Bernhard; Flepp, Beat; Goyal, Prasoon; Jackel, Lawrence D.; Monfort, Mathew; Muller, Urs; Zhang, Jiakai; Zhang, Xin; Zhao, Jake: End to End Learning for Self-Driving Cars. In: *arXiv preprint arXiv:1604.07316* (2016)

[BWM08] Brauckhoff, Daniela; Wagner, Arno; May, Martin: FLAME: A Flow-Level Anomaly Modeling Engine. In: *Workshop on Cyber Security Experimentation and Test (CSET)*, USENIX Association, 2008, S. 1:1–1:6

[CEE+06] Chandola, Varun; Eilertson, Eric; Ertoz, Levent; Simon, Gyorgy; Kumar, Vipin: Data Mining for Cyber Security. In: Singhal, A. (Hrsg.): *Data Warehousing and Data Mining Techniques for Computer Security*. 1. Auflage. Springer, 2006, S. 83–107

[CEQ+06] Cao, Feng; Ester, Martin; Qian, Weining; Zhou, Aoying: Density-Based Clustering over an Evolving Data Stream with Noise. In: *SIAM International Conference on Data Minning (SDM)* Bd. 6 Society for Industrial and Applied Mathematics, 2006, S. 328–339. http://dx.doi.org/10.1137/1.9781611972764.29

[CG12] Catania, Carlos A.; Garino, Carlos G.: Automatic network intrusion detection: Current techniques and open issues. In: *Computers and Electrical Engineering* 38 (2012), Nr. 5, S. 1062–1072. http://dx.doi.org/10.1016/j.compeleceng.2012.05.013

[CH74] Caliński, Tadeusz; Harabasz, Jerzy: A Dendrite Method for Cluster Analysis. In: *Communications in Statistics-theory and Methods* 3 (1974), Nr. 1, S. 1–27. http://dx.doi.org/10.1080/03610927408827101

[CH13] Creech, Gideon; Hu, Jiankun: Generation of a new IDS test dataset: Time to retire the KDD collection. In: *IEEE Wireless Communications and Networking Conference (WCNC)* IEEE, 2013, S. 4487–4492. http://dx.doi.org/10.1109/WCNC.2013.6555301

[Che06] Chen, Eric Y.: Detecting DoS attacks on SIP systems. In: *IEEE Workshop on VoIP Management and Security* IEEE, 2006, S. 53–58. http://dx.doi.org/10.1109/VOIPMS.2006.1638123

[Cis18] Cisco: Cisco 2018 - Annual Cybersecurity Report / Cisco. Version: 2018. https://www.cisco.com/c/dam/m/hu_hu/campaigns/security-hub/pdf/acr-2018.pdf. Cisco, 2018. – Forschungsbericht. – Letzter Zugriff am 5 August 2020

[CJV+18] Cermak, Milan; Jirsik, Tomas; Velan, Petr; Komarkova, Jana; Spacek, Stanislav; Drasar, Martin; Plesnik, Tomas: Towards Provable Network Traffic Measurement and Analysis via Semi-Labeled Trace Datasets. In: *IEEE Network Traffic Measurement and Analysis Conference (TMA)* IEEE, 2018, S. 1–8. http://dx.doi.org/10.23919/TMA.2018.8506498

[CL14] Cleve, Jürgen; Lämmel, Uwe: *Data Mining*. 1. Auflage. Oldenbourg Wissenschaftsverlag, 2014 (De Gruyter Studium). – ISBN 9783486713916

[Cla04] Claise, Benoit: Cisco Systems NetFlow Services Export Version 9 / Internet Engineering Task Force. Version: Oktober 2004. https://tools.ietf.org/html/rfc3954. IETF, Oktober 2004 (3954). – RFC. – 1–3 S. – (Letzter Zugriff am 5 August 2020)

[Cla08] Claise, Benoit: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information / Internet Engineering Task Force. Version: Januar 2008. https://tools.ietf.org/html/rfc5101. IETF, Januar 2008 (5101). – RFC. – (Letzter Zugriff am 5 August 2020)

[CM14] Chen, Danqi; Manning, Christopher: A Fast and Accurate Dependency Parser using Neural Networks. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, S. 740–750. http://dx.doi.org/10.3115/v1/D14-1082

[CMB11] Coull, Scott E.; Monrose, Fabian; Bailey, Michael: On Measuring the Similarity of Network Hosts: Pitfalls, New Metrics, and Empirical Analyses. In: *Network and Distributed System Security Symposium*, 2011, S. 1–16

[CNM16] Cao, Van L.; Nicolau, Miguel; McDermott, James: A Hybrid Autoencoder and Density Estimation Model for Anomaly Detection. In: *International Conference on Parallel Problem Solving from Nature* Springer, 2016, S. 717–726. http://dx.doi.org/10.1007/978-3-319-45823-6_67

[CRK+11] Celik, Berkay; Raghuram, Jayaram; Kesidis, George; Miller, David J.: Salting Public Traces with Attack Traffic to Test Flow Classifiers. In: *Workshop on Cyber Security Experimentation and Test (CSET)*, 2011

[CSL04] Chou, Chien-Hsing; Su, Mu-Chun; Lai, Eugene: A new cluster validity measure and its application to image compression. In: *Pattern Analysis and Applications* 7 (2004), Nr. 2, S. 205–220. http://dx.doi.org/10.1007/s10044-004-0218-1

[CTA13] Claise, Benoit; Trammell, Brian; Aitken, Paul: Specification of the IP flow information export (IPFIX) protocol for the exchange of flow information / Internet Engineering Task Force. Version: 2013. https://tools.ietf.org/html/rfc7011. IETF, 2013 (7011). – RFC. – (Letzter Zugriff am 5 August 2020)

[DB79] Davies, David L.; Bouldin, Donald W.: A Cluster Separation Measure. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1 (1979), Nr. 2, S. 224–227. http://dx.doi.org/10.1109/TPAMI.1979.4766909

[DD11] Dua, Sumeet; Du, Xian: *Data Mining and Machine Learning in Cybersecurity*. 1. Auflage. CRC press, 2011. – ISBN 978–1–4398–3942–3

[Dem06] Demšar, Janez: Statistical Comparisons of Classifiers over Multiple Data Sets. In: *The Journal of Machine Learning Research (JMLR)* 7 (2006), S. 1–30

[DKT17] Dheeru, Dua; Karra Taniskidou, Efi: *UCI Machine Learning Repository*. http://archive.ics.uci.edu/ml. Version: 2017. – (Letzter Zugriff am 5 August 2020)

[DM17] Das, Rishabh; Morris, Thomas H.: Machine Learning and Cyber Security. In: *IEEE International Conference on Computer, Electrical and Communication Engineering (ICCECE)* IEEE, 2017, S. 1–7. http://dx.doi.org/10.1109/ICCECE.2017.8526232

[DMP05] Dutta, Mala; Mahanta, A K.; Pujari, Arun K.: QROCK: A quick version of the ROCK algorithm for clustering of categorical data. In: *Pattern Recognition Letters* 26 (2005), Nr. 15, S. 2364–2373. http://dx.doi.org/10.1016/j.patrec.2005.04.008

[EAP+02] Eskin, Eleazar; Arnold, Andrew; Prerau, Michael; Portnoy, Leonid; Stolfo, Sal: A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data. In: *Applications of Data Mining in Computer Security* 6 (2002), S. 77–102

[Eck18] Eckert, Claudia: *IT-Sicherheit: Konzepte-Verfahren-Protokolle*. 10. Auflage. De Gruyter, 2018. – ISBN 978–3–11–055158–7

[ED18] Erlacher, Felix; Dressler, Falko: How to Test an IDS?: GENESIDS: An Automated System for Generating Attack Traffic. In: *Workshop on Traffic Measurements for Cybersecurity (WTMC)*. New York, NY, USA : ACM, 2018, S. 46–51. http://dx.doi.org/10.1145/3229598.3229601

[EKS+96] Ester, Martin; Kriegel, Hans-Peter; Sander, Jörg; Xu, Xiaowei: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: *International Conference on Knowledge Discovery and Data Mining*, AAAI, 1996, S. 226–231

[ESK03] Ertöz, Levent; Steinbach, Michael; Kumar, Vipin: Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data. In: *SIAM International Conference on Data Mining* SIAM, 2003, S. 47–58. http://dx.doi.org/10.1137/1.9781611972733.5

[FBA+10] Fontugne, Romain; Borgnat, Pierre; Abry, Patrice; Fukuda, Kensuke: MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking. In: *International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*. New York, NY, USA : ACM, 2010, S. 8:1–8:12. http://dx.doi.org/10.1145/1921168.1921179

[FGB+03] Feng, Wu-chang; Goel, Ashvin; Bezzaz, Abdelmajid; Feng, Wu-chi; Walpole, Jonathan: TCPivo: A High-Performance Packet Replay Engine. In: *ACM Workshop on Models, Methods and Tools for Reproducible Network Research* ACM, 2003, S. 57–64. http://dx.doi.org/10.1145/944773.944783

[FI93] Fayyad, Usama M.; Irani, Keki B.: Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In: *International Joint Conference on Artificial Intelligence (IJCAI)*, Morgan Kaufmann, 1993, S. 1022–1029

[FPS96] Fayyad, Usama; Piatetsky-Shapiro, Gregory; Smyth, Padhraic: From Data Mining to Knowledge Discovery in Databases. In: *AI Magazine* 17 (1996), Nr. 3, S. 37–37. http://dx.doi.org/10.1609/aimag.v17i3.1230

[Fri37] Friedman, Milton: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. In: *Journal of the American Statistical Association* 32 (1937), Nr. 200, S. 675–701. http://dx.doi.org/10.2307/2279372

[Fri40] Friedman, Milton: A Comparison of Alternative Tests of Significance for the Problem of m Rankings. In: *The Annals of Mathematical Statistics* 11 (1940), Nr. 1, S. 86–92. http://dx.doi.org/10.1214/aoms/1177731944

[FSB+18] Ficke, Eric; Schweitzer, Kristin M.; Bateman, Raymond M.; Xu, Shouhuai: Characterizing the Effectiveness of Network-based Intrusion Detection Systems. In: *IEEE Military Communications Conference (MILCOM)*, 2018, S. 76–81

[FTV+10] Fadlullah, Zubair M.; Taleb, Tarik; Vasilakos, Athanasios V.; Guizani, Mohsen; Kato, Nei: DTRAB: Combating Against Attacks on Encrypted Protocols Through Traffic-Feature Analysis. In: *IEEE/ACM Transactions on Networking (TON)* 18 (2010), Nr. 4, S. 1234–1247. http://dx.doi.org/10.1109/TNET.2009.2039492

[GAA+17] Gulrajani, Ishaan; Ahmed, Faruk; Arjovsky, Martin; Dumoulin, Vincent; Courville, Aaron C.: Improved Training of Wasserstein GAN. In: *Advances in Neural Information Processing Systems (NIPS)*, 2017, S. 5769–5779

[Gam64] Gambaryan, P.: A Mathematical Model of Taxonomy. In: *Izvest. Akad. Nauk Armen. SSR* 17 (1964), Nr. 12, S. 47–53

[GBB+12] Gogoi, Prasanta; Bhuyan, Monowar H.; Bhattacharyya, D; Kalita, Jugal K.: Packet and Flow Based Network Intrusion Dataset. In: *International Conference on Contemporary Computing* Springer, 2012, S. 322–334. http://dx.doi.org/10.1007/978-3-642-32129-0_34

[GBC18] Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron: *Deep Learning*. 1. Auflage. MIT press, 2018. – ISBN 978–3–95845–700–3

[GDM+09] Garcia-Teodoro, Pedro; Diaz-Verdejo, Jesus; Macia-Fernandez, Gabriel; Vázquez, Enrique: Anomaly-based network intrusion detection: Techniques, systems and challenges. In: *Computers and Security* 28 (2009), Nr. 1, S. 18–28. http://dx.doi.org/10.1016/j.cose.2008.08.003

[GE03] Guyon, Isabelle; Elisseeff, André: An Introduction to Variable and Feature Selection. In: *Journal of Machine Learning Research* 3 (2003), Nr. Mar, S. 1157–1182

[GG07] Gharibian, Farnaz; Ghorbani, Ali A.: Comparative Study of Supervised Machine Learning Techniques for Intrusion Detection. In: *Conference on Communication Networks and Services Research (CNSR)* IEEE, 2007, S. 350–358. http://dx.doi.org/10.1109/CNSR.2007.22

[GGS+14] Garcia, Sebastian; Grill, Martin; Stiborek, Jan; Zunino, Alejandro: An empirical comparison of botnet detection methods. In: *Computers and Security* 45 (2014), S. 100–123. http://dx.doi.org/10.1016/j.cose.2014.05.011

[GH07] Goebel, Jan; Holz, Thorsten: Rishi: Identify Bot Contaminated Hosts by IRC Nickname Evaluation. In: *HotBots* 7 (2007), S. 8–8

[GHK14] Golling, Mario; Hofstede, Rick; Koch, Robert: Towards Multi-layered Intrusion Detection in High-Speed Networks. In: *IEEE International Conference On Cyber Conflict (CyCon 2014)* IEEE, 2014, S. 191–206. http://dx.doi.org/10.1109/CYCON.2014.6916403

[GL13] Glasser, Joshua; Lindauer, Brian: Bridging the Gap: A Pragmatic Approach to Generating Insider Threat Data. In: *Workshop on Research for Insider Threat (WRIT), IEEE Security and Privacy Workshops (SPW)* IEEE, 2013, S. 98–104. http://dx.doi.org/10.1109/SPW.2013.37

[GLC+18] Guo, Jiaxian; Lu, Sidi; Cai, Han; Zhang, Weinan; Yu, Yong; Wang, Jun: Long Text Generation via Adversarial Training with Leaked Information. In: *AAAI Conference on Artificial Intelligence*, 2018, S. 5141–5148

[GMK+06] Gates, Carrie; McNutt, Joshua J.; Kadane, Joseph B.; Kellner, Marc I.: Scan Detection on Very Large Networks Using Logistic Regression Modeling. In: *IEEE Symposium on Computers and Communications (ISCC)* IEEE, 2006, S. 402–408. http://dx.doi.org/10.1109/ISCC.2006.142

[GNC99] Goil, Sanjay; Nagesh, Harsha; Choudhary, Alok: MAFIA: Efficient and Scalable Subspace Clustering for Very Large Data Sets. In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* ACM, 1999, S. 443–452

[Goo66] Goodall, David W.: A New Similarity Index Based on Probability. In: *Biometrics* (1966), S. 882–907. http://dx.doi.org/10.2307/2528080

[Goo16] Goodfellow, Ian: NIPS 2016 tutorial: Generative Adversarial Networks. In: *arXiv preprint arXiv:1701.00160* (2016)

[GPM+14] Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Yoshua: Generative Adversarial Nets. In: *Advances in Neural Information Processing Systems (NIPS)*, 2014, S. 2672–2680

[GPM+17] Grosse, Kathrin; Papernot, Nicolas; Manoharan, Praveen; Backes, Michael; McDaniel, Patrick: Adversarial Examples for Malware Detection. In: *European Symposium on Research in Computer Security* Springer, 2017, S. 62–79. http://dx.doi.org/10.1007/978-3-319-66399-9_4

[GPR+08] Giacinto, Giorgio; Perdisci, Roberto; Rio, Mauro d.; Roli, Fabio: Intrusion Detection in Computer Networks by a Modular Ensemble of One-Class Classifiers. In: *Information Fusion* 9 (2008), Nr. 1, S. 69–82. http://dx.doi.org/10.1016/j.inffus.2006.10.002

[GRS99] Guha, Sudipto; Rastogi, Rajeev; Shim, Kyuseok: ROCK: A Robust Clustering Algorithm for Categorical Attributes. In: *International Conference on Data Engineering* IEEE, 1999, S. 512–521. http://dx.doi.org/10.1109/ICDE.1999.754967

[GSD+09] Gringoli, Francesco; Salgarelli, Luca; Dusi, Maurizio; Cascarano, Niccolo; Risso, Fulvio; Kimberly, Claffy: GT: Picking up the Truth from the Ground for Internet Traffic. In: *ACM SIGCOMM Computer Communication Review* 39 (2009), Nr. 5, S. 12–18. http://dx.doi.org/10.1145/1629607.1629610

[GZL06] Gao, Ming; Zhang, Kenong; Lu, Jiahua: Efficient Packet Matching for Gigabit Network Intrusion Detection using TCAMs. In: *IEEE International Conference on Advanced Information Networking and Applications (AINA)* IEEE, 2006, S. 249–254. http://dx.doi.org/10.1109/AINA.2006.164

[Hac11] Hachman, Mark: PlayStation Hack to Cost Sony $171M; Quake Costs Far Higher. In: *PC Magazine UK* (23.5.2011). https://uk.pcmag.com/news/106573/playstation-hack-to-cost-sony-171m-quake-costs-far-higher. – (Letzter Zugriff am 5 August 2020)

[Ham09] Hamel, Lutz H.: *Knowledge discovery with support vector machines*. 1. Auflage. John Wiley and Sons, 2009. – ISBN 978–0470371923

[HCC+13] He, Yujie; Chen, Wenlin; Chen, Yixin; Mao, Yi: Kernel Density Metric Learning. In: *International Conference on Data Mining (ICDM)* IEEE, 2013, S. 271–280. http://dx.doi.org/10.1109/ICDM.2013.153

[HČT+14] Hofstede, Rick; Čeleda, Pavel; Trammell, Brian; Drago, Idilio; Sadre, Ramin; Sperotto, Anna; Pras, Aiko: Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX. In: *IEEE Communications Surveys and Tutorials* 16 (2014), Nr. 4, S. 2037–2064. http://dx.doi.org/10.1109/COMST.2014.2321898

[Hen16] Henry, Ed: *Netflow and word2vec - flow2vec [Blogpost]*. https://edhenry.github.io/2016/12/21/Netflow-flow2vec/. Version: 21.12.2016. – (Letzter Zugriff am 5 August 2020)

[HGM+97] Hoffman, Patrick; Grinstein, Georges; Marx, Kenneth; Grosse, Ivo; Stanley, Eugene: DNA visual and analytic data mining. In: *IEEE Conference on Visualization (Cat. No. 97CB36155)*, 1997, S. 437–441. http://dx.doi.org/10.1109/VISUAL.1997.663916

[HH05] Hansman, Simon; Hunt, Ray: A taxonomy of network and computer attacks. In: *Computers and Security* 24 (2005), Nr. 1, S. 31–43. http://dx.doi.org/10.1016/j.cose.2004.06.011

[HHH+12] Hellemons, Laurens; Hendriks, Luuk; Hofstede, Rick; Sperotto, Anna; Sadre, Ramin; Pras, Aiko: SSHCure: A Flow-Based SSH Intrusion Detection System. In: *International Conference on Autonomous Infrastructure, Management and Security (IFIP)* Springer, 2012, S. 86–97. http://dx.doi.org/10.1007/978-3-642-30633-4_11

[HHP12] Hu, Zi; Heidemann, John; Pradkin, Yuri: Towards Geolocation of Millions of IP Addresses. In: *ACM Conference on Internet Measurement*, ACM, 2012, S. 123–130. http://dx.doi.org/10.1145/2398776.2398790

[HHS+14] Hofstede, Rick; Hendriks, Luuk; Sperotto, Anna; Pras, Aiko: SSH compromise detection using NetFlow/IPFIX. In: *ACM SIGCOMM Computer Communication Review* 44 (2014), Nr. 5, S. 20–26. http://dx.doi.org/10.1145/2677046.2677050

[HHS+17] Haider, Waqas; Hu, Jiankun; Slay, Jill; Turnbull, Benjamin; Xie, Yi: Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. In: *Journal of Network and Computer Applications* 87 (2017), S. 185–192. http://dx.doi.org/10.1016/j.jnca.2017.03.018

[HKH+17] Han, Juhyeng; Kim, Seongmin; Ha, Jaehyeong; Han, Dongsu: SGX-Box: Enabling Visibility on Encrypted Traffic using a Secure Middlebox Module. In: *Asia-Pacific Workshop on Networking* ACM, 2017, S. 99–105. http://dx.doi.org/10.1145/3106989.3106994

[HKP11] Han, Jiawei; Kamber, Micheline; Pei, Jian: *Data Mining: Concepts and Techniques*. 3. Auflage. Elsevier, 2011. – ISBN 9780123814791

[HLJ16] Hamilton, William L.; Leskovec, Jure; Jurafsky, Dan: Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. In: *Annual Meeting of the Association for Computational Linguistics* Bd. 1, 2016, S. 1489–1501. http://dx.doi.org/10.18653/v1/P16-1141

[HPS+18] Hofstede, Rick; Pras, Aiko; Sperotto, Anna; Rodosek, Gabi D.: Flow-Based Compromise Detection: Lessons Learned. In: *IEEE Security and Privacy* 16 (2018), Nr. 1, S. 82–89. http://dx.doi.org/10.1109/MSP.2018.1331021

[HRU+17] Heusel, Martin; Ramsauer, Hubert; Unterthiner, Thomas; Nessler, Bernhard; Hochreiter, Sepp: GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In: *Advances in Neural Information Processing Systems (NIPS)*, 2017, S. 6629–6640

[HS06] Hinton, Geoffrey E.; Salakhutdinov, Ruslan R.: Reducing the Dimensionality of Data with Neural Networks. In: *Science* 313 (2006), Nr. 5786, S. 504–507. http://dx.doi.org/10.1126/science.1127647

[HS15] Hassani, Marwan; Seidl, Thomas: Internal Clustering Evaluation of Data Streams. In: *Trends and Applications in Knowledge Discovery and Data Mining*, Springer, 2015, S. 198–209. http://dx.doi.org/10.1007/978-3-319-25660-3_17

[HSC+11] Horng, Shi-Jinn; Su, Ming-Yang; Chen, Yuan-Hsin; Kao, Tzong-Wann; Chen, Rong-Jian; Lai, Jui-Lin; Perkasa, Citra D.: A novel intrusion detection system based on hierarchical clustering and support vector machines. In: *Expert Systems with Applications* 38 (2011), Nr. 1, S. 306–313. http://dx.doi.org/10.1016/j.eswa.2010.06.066

[HT17] Hu, Weiwei; Tan, Ying: Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN. In: *arXiv preprint arXiv:1702.05983* (2017)

[Hua98] Huang, Zhexue: Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values. In: *Data Mining and Knowledge Discovery* 2 (1998), Nr. 3, S. 283–304. http://dx.doi.org/10.1023/A:1009769707641

[HW06] Hsu, Chung-Chian; Wang, Sheng-Hsuan: An Integrated Framework for Visualized and Exploratory Pattern Discovery in Mixed Data. In: *IEEE Transactions on Knowledge and Data Engineering* 18 (2006), Nr. 2, S. 161–173. http://dx.doi.org/10.1109/TKDE.2006.23

[HZ16] Haddadi, Fariba; Zincir-Heywood, Nur: Benchmarking the Effect of Flow Exporters and Protocol Filters on Botnet Traffic Classification. In: *IEEE Systems Journal* 10 (2016), Nr. 4, S. 1390–1401. http://dx.doi.org/10.1109/JSYST.2014.2364743

[IAB17] Idhammad, Mohamed; Afdel, Karim; Belouch, Mustapha: DoS Detection Method based on Artificial Neural Networks. In: *International Journal of Advanced Computer Science and Applications (IJACSA)* 8 (2017), Nr. 4, S. 465–471

[IKG+17] Iannucci, Stefano; Kholidy, Hisham A.; Ghimire, Amrita D.; Jia, Rui; Abdelwahed, Sherif; Banicescu, Ioana: A Comparison of Graph-Based Synthetic Data Generators for Benchmarking Next-Generation Intrusion Detection Systems. In: *IEEE International Conference on Cluster Computing (CLUSTER)* IEEE, 2017, S. 278–289. http://dx.doi.org/10.1109/CLUSTER.2017.54

[Ins85] Inselberg, Alfred: The plane with parallel coordinates. In: *The Visual Computer* 1 (1985), Nr. 2, S. 69–91. http://dx.doi.org/10.1007/BF01898350

[IPM09] Ienco, Dino; Pensa, Ruggero G.; Meo, Rosa: Context-Based Distance Learning for Categorical Data Clustering. In: *Advances in Intelligent Data Analysis VIII*, Springer, 2009, S. 83–94. http://dx.doi.org/10.1007/978-3-642-03915-7_8

[IT10] Inacio, Christopher M.; Trammell, Brian: YAF: Yet Another Flowmeter. In: *Large Installation System Administration Conference*, 2010, S. 107–118

[IW08] Igure, Vinay M.; Williams, Ronald D.: Taxonomies of Attacks and Vulnerabilities in Computer Systems. In: *IEEE Communications Surveys and Tutorials* 10 (2008), Nr. 1. http://dx.doi.org/10.1109/COMST.2008.4483667

[IZZ+17] Isola, Phillip; Zhu, Jun-Yan; Zhou, Tinghui; Efros, Alexei A.: Image-to-Image Translation with Conditional Adversarial Networks. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* IEEE, 2017, S. 5967–5976. http://dx.doi.org/10.1109/CVPR.2017.632

[JCL14] Jia, Hong; Cheung, Yiu-ming; Liu, Jiming: A New Distance Metric for Unsupervised Learning of Categorical Data. In: *International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2014, S. 1893–1899. http://dx.doi.org/10.1109/IJCNN.2014.6889890

[JDC10] John, Wolfgang; Dusi, Maurizio; Claffy, Kimberly C.: Estimating Routing Symmetry on Single Links by Passive Flow Measurements. In: *International Wireless Communications and Mobile Computing Conference* ACM, 2010, S. 473–478. http://dx.doi.org/10.1145/1815396.1815506

[JGS+16] Jakalan, Ahmad; Gong, Jian; Su, Qi; Hu, Xiaoyan; Abdelgder, Abdeldime M.: Social relationship discovery of IP addresses in the managed IP networks by observing traffic at network boundary. In: *Computer Networks* 100 (2016), S. 12–27. http://dx.doi.org/10.1016/j.comnet.2016.02.012

[JGS+17] Jazi, Hossein H.; Gonzalez, Hugo; Stakhanova, Natalia; Ghorbani, Ali A.: Detecting HTTP-based application layer DoS attacks on web servers in the presence of sampling. In: *Computer Networks* 121 (2017), S. 25–36. http://dx.doi.org/10.1016/j.comnet.2017.03.018

[JJC+16] Ji, Soo-Yeon; Jeong, Bong-Keun; Choi, Seonho; Jeong, Dong H.: A multi-level intrusion detection method for abnormal network behaviors. In: *Journal of Network and Computer Applications* 62 (2016), S. 9–17. http://dx.doi.org/10.1016/j.jnca.2015.12.004

[JLM16] Jiang, Hao; Liu, Yaoqing; Matthews, Jeanna N.: IP Geolocation Estimation using Neural Networks with Stable Landmarks. In: *IEEE Conference on Computer Communications Workshops* IEEE, 2016, S. 170–175. http://dx.doi.org/10.1109/INFCOMW.2016.7562066

[JP05] Joshi, Shrijit S.; Phoha, Vir V.: Investigating Hidden Markov Models Capabilities in Anomaly Detection. In: *Annual Southeast Regional Conference Volume 1*. New York, NY, USA : ACM, 2005, S. 98–103. http://dx.doi.org/10.1145/1167350.1167387

[JPB+04] Jung, Jaeyeon; Paxson, Vern; Berger, Arthur W.; Balakrishnan, Hari: Fast Portscan Detection Using Sequential Hypothesis Testing. In: *IEEE Symposium on Security and Privacy* IEEE, 2004, S. 211–225. http://dx.doi.org/10.1109/SECPRI.2004.1301325

[JS11] Joshi, RC; Sardana, Anjali: *Honeypots: A New Paradigm to Information Security*. 1. Auflage. CRC Press, 2011. – ISBN 978–1578087082

[JYP18] Jing, Xuyang; Yan, Zheng; Pedrycz, Witold: Security Data Collection and Data Analytics in the Internet: A Survey. In: *IEEE Communications Surveys and Tutorials* (2018), S. 1–34. http://dx.doi.org/10.1109/COMST.2018.2863942

[JZA07] Jemili, Farah; Zaghdoud, Montaceur; Ahmed, Mohamed B.: A Framework for an Adaptive Intrusion Detection System using Bayesian Network. In: *IEEE Intelligence and Security Informatics* IEEE, 2007, S. 66–70. http://dx.doi.org/10.1109/ISI.2007.379535

[KAG+18] Kofler, Michael; Aigner, Roland; Gebeshuber, Klaus; Hackner, Thomas; Kania, Stefan; Klöp, Peter; Neugebauer, Frank; Widl, Markus; Zingsheim, Andre: *Hacking and Security: Das umfassende Handbuch*. 1. Auflage. Rheinwerk Computing, 2018. – ISBN 978–3836245487

[KAL+18] Karras, Tero; Aila, Timo; Laine, Samuli; Lehtinen, Jaakko: Progressive Growing of GANs for Improved Quality, Stability, and Variation. In: *International Conference on Learning Representations*, 2018

[KAT07] Khan, Latifur; Awad, Mamoun; Thuraisingham, Bhavani: A new intrusion detection system using support vector machines and hierarchical clustering. In: *The VLDB Journal* 16 (2007), Oct, Nr. 4, S. 507–521. http://dx.doi.org/10.1007/s00778-006-0002-5

[KCH+14] Kim, Yoon; Chiu, Yi-I; Hanaki, Kentaro; Hegde, Darshan; Petrov, Slav: Temporal Analysis of Language through Neural Language Models. In: *ACL Workshop on Language Technologies and Computational Social Science*, 2014, S. 61–65. http://dx.doi.org/10.3115/v1/W14-2517

[Ken15] Kent, Alexander D.: *Comprehensive, Multi-Source Cyber-Security Events*. Los Alamos National Laboratory, 2015. – http://dx.doi.org/10.17021/1179829

[Ken16] Kent, Alexander D.: Cyber-Security Data Sources for Dynamic Network Research. In: *Dynamic Networks and Cyber-Security*, World Scientific, 2016, S. 37–65. http://dx.doi.org/10.1142/9781786340757_0002

[KGR14] Koch, Robert; Golling, Mario; Rodosek, Gabi D.: Behavior-based Intrusion Detection in Encrypted Environments. In: *IEEE Communications Magazine* 52 (2014), July, Nr. 7, S. 124–131. http://dx.doi.org/10.1109/MCOM.2014.6852093

[Kha11] Khan, Sadiq Ali M.: Rule based Network Intrusion Detection using Genetic Algorithm. In: *International Journal of Computer Applications* 18 (2011), Nr. 8, S. 26–29. http://dx.doi.org/10.5120/2303-2914

[KHB14] Kočiský, Tomáš; Hermann, Karl M.; Blunsom, Phil: Learning Bilingual Word Representations by Marginalizing Alignments. In: *Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2014, S. 224–229. http://dx.doi.org/10.3115/v1/P14-2037

[KHH11] Khorshidpour, Zeinab; Hashemi, Sattar; Hamzeh, Ali: CBDL: Context-Based Distance Learning for Categorical Attributes. In: *International Journal of Intelligent Systems* 26 (2011), Nr. 11, S. 1076–1100. http://dx.doi.org/10.1002/int.20499

[Kim14] Kim, Do-Yeon: Cyber security issues imposed on nuclear power plants. In: *Annals of Nuclear Energy* 65 (2014), S. 141 – 143. http://dx.doi.org/10.1016/j.anucene.2013.10.039

[KKK+17] Kwon, Donghwoon; Kim, Hyunjoo; Kim, Jinoh; Suh, Sang C.; Kim, Ikkyun; Kim, Kuinam J.: A survey of deep learning-based network anomaly detection. In: *Cluster Computing* (2017), September, S. 949—-961. http://dx.doi.org/10.1007/s10586-017-1117-8

[KKS+16] Kolias, Constantinos; Kambourakis, Georgios; Stavrou, Angelos; Gritzalis, Stefanos: Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats and a Public Dataset. In: *IEEE Communications Surveys Tutorials* 18 (2016), Nr. 1, S. 184–208. http://dx.doi.org/10.1109/COMST.2015.2402161. – ISSN 1553–877X

[KKT+16] Kim, Jihyun; Kim, Jaehyun; Thu, Huong Le T.; Kim, Howon: Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection. In: *International Conference on Platform Technology and Service (PlatCon)* IEEE, 2016, S. 1–5. http://dx.doi.org/10.1109/PlatCon.2016.7456805

[Koc11] Koch, Robert: Towards Next-Generation Intrusion Detection. In: *IEEE International Conference on Cyber Conflict (ICCC)* IEEE, 2011, S. 1–18

[KRG+08] Kelly, Douglas J.; Raines, Richard A.; Grimaila, Michael R.; Baldwin, Rusty O.; Mullins, Barry E.: A Survey of State-of-the-Art in Anonymity Metrics. In: *ACM Workshop on Network Data Anonymization* ACM, 2008, S. 31–40. http://dx.doi.org/10.1145/1456441.1456453

[LBH15] LeCun, Yann; Bengio, Yoshua; Hinton, Geoffrey: Deep learning. In: *Nature* 521 (2015), Nr. 7553, S. 436–444. http://dx.doi.org/10.1038/nature14539

[LCH+16] Li, Jiwei; Chen, Xinlei; Hovy, Eduard; Jurafsky, Dan: Visualizing and Understanding Neural Models in NLP. In: *North American Chapter of the Association for Computational Linguistics – Human Language Technologies (NAACL HLT)*, 2016, S. 681–691. http://dx.doi.org/10.18653/v1/N16-1082

[LD12] Liu, Qingbao; Dong, Guozhu: CPCQ: Contrast pattern based clustering quality index for categorical data. In: *Pattern Recognition* 45 (2012), Nr. 4, S. 1739–1748. http://dx.doi.org/10.1016/j.patcog.2011.10.007

[Lee07] Lee, Chang-Hwan: A Hellinger-based discretization method for numeric attributes in classification learning. In: *Knowledge-Based Systems* 20 (2007), Nr. 4, S. 419–425. http://dx.doi.org/10.1016/j.knosys.2006.06.005

[LFG+00] Lippmann, Richard P.; Fried, David J.; Graf, Isaac; Haines, Joshua W.; Kendall, Kristopher R.; McClung, David; Weber, Dan; Webster, Seth E.; Wyschogrod, Dan; Cunningham, Robert K.; Zissman, Marc A.: Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation. In: *DARPA Information Survivability Conference and Exposition (DISCEX)* Bd. 2 IEEE, 2000, S. 12–26. http://dx.doi.org/10.1109/DISCEX.2000.821506

[LGB+13] Li, Bingdong; Gunes, Mehmet H.; Bebis, George; Springer, Jeff: A Supervised Machine Learning Approach to Classify Host Roles On Line Using sFlow. In: *Workshop on High Performance and Programmable Networking*, ACM, 2013, S. 53–60. http://dx.doi.org/10.1145/2465839.2465847

[LGT+97] Lawrence, Steve; Giles, Lee; Tsoi, Chung; Back, Andrew: Face Recognition: A Convolutional Neural Network Approach. In: *IEEE Transactions on Neural Networks* 8 (1997), Nr. 1, S. 98–113. http://dx.doi.org/10.1109/72.554195

[LH05] Le, Si Q.; Ho, Tu B.: An association-based dissimilarity measure for categorical data. In: *Pattern Recognition Letters* 26 (2005), Nr. 16, S. 2549–2557. http://dx.doi.org/10.1016/j.patrec.2005.06.002

[LHF+00] Lippmann, Richard; Haines, Joshua W.; Fried, David J.; Korba, Jonathan; Das, Kumar: The 1999 DARPA Off-Line Intrusion Detection Evaluation. In: *Computer networks* 34 (2000), Nr. 4, S. 579–595. http://dx.doi.org/10.1016/S1389-1286(00)00139-0

[LHP01] Li, Wenmin; Han, Jiawei; Pei, Jian: CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules. In: *IEEE International Conference on Data Mining* IEEE, 2001, S. 369–376. http://dx.doi.org/10.1109/ICDM.2001.989541

[LKM+18] Lucic, Mario; Kurach, Karol; Michalski, Marcin; Gelly, Sylvain; Bousquet, Olivier: Are GANs Created Equal? A Large-Scale Study. In: *Advances in Neural Information Processing Systems*, 2018, S. 698–707

[LKT15] Lin, Wei-Chao; Ke, Shih-Wen; Tsai, Chih-Fong: CANN: An intrusion detection system based on combining cluster centers and nearest neighbors. In: *Knowledge-based systems* 78 (2015), S. 13–21. http://dx.doi.org/10.1016/j.knosys.2015.01.009

[LL05] Leung, Kingsly; Leckie, Christopher: Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters. In: *Australasian Conference on Computer Science (ACSC)*, Australian Computer Society, Inc., 2005, S. 333–342

[LLL+13] Liao, Hung-Jen; Lin, Chun-Hung R.; Lin, Ying-Chih; Tung, Kuang-Yuan: Intrusion detection system: A comprehensive review. In: *Journal of Network and Computer Applications* 36 (2013), Nr. 1, S. 16–24. http://dx.doi.org/10.1016/j.jnca.2012.09.004

[LLS+15] Li, Haoxiang; Lin, Zhe; Shen, Xiaohui; Brandt, Jonathan; Hua, Gang: A Convolutional Neural Network Cascade for Face Detection. In: *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, S. 5325–5334. http://dx.doi.org/10.1109/CVPR.2015.7299170

[LM14] Le, Quoc; Mikolov, Tomas: Distributed Representations of Sentences and Documents. In: *International Conference on Machine Learning*, 2014, S. 1188–1196

[LOS+13] Landes, Dieter; Otto, Florian; Schumann, Sven; Schlottke, Frank: Identifying Suspicious Activities in Company Networks Through Data Mining and Visualization. In: Rausch, Peter (Hrsg.); Sheta, Alaa F. (Hrsg.); Ayesh, Aladdin (Hrsg.): *Business Intelligence and Performance Management*, Springer, 2013, S. 75–90. http://dx.doi.org/10.1007/978-1-4471-4866-1_6

[LR05] Lehmann, Erich L.; Romano, Joseph P.: *Testing Statistical Hypotheses*. 3. Auflage. Springer, 2005 (Springer Texts in Statistics). – ISBN 978–0–387–98864–1

[LSX18] Lin, Zilong; Shi, Yong; Xue, Zhi: IDSGAN: Generative Adversarial Networks for Attack Generation against Intrusion Detection. In: *arXiv preprint arXiv:1809.02077* (2018)

[LTH+17] Ledig, Christian; Theis, Lucas; Huszár, Ferenc; Caballero, Jose; Cunningham, Andrew; Acosta, Alejandro; Aitken, Andrew; Tejani, Alykhan; Totz, Johannes; Wang, Zehan; Shi, Wenzhe: Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* IEEE, 2017, S. 105–114

[LW67] Lance, Godfrey N.; Williams, William T.: A general theory of classificatory sorting strategies: 1. Hierarchical systems. In: *The computer journal* 9 (1967), Nr. 4, S. 373–380. http://dx.doi.org/10.1093/comjnl/9.4.373

[LXZ+12] Li, Yinhui; Xia, Jingbo; Zhang, Silan; Yan, Jiakai; Ai, Xiaochuan; Dai, Kuobin: An efficient intrusion detection system based on support vector machines and gradually feature removal method. In: *Expert Systems with Applications* 39 (2012), Nr. 1, S. 424–430. http://dx.doi.org/10.1016/j.eswa.2011.07.032

[Lyo08] Lyon, Gordon F.: *Nmap network scanning: The official Nmap project guide to network discovery and security scanning.* 1. Auflage. Insecure, 2008. – ISBN 978–0–9799587–1–7

[LZO10] Lin, Ying; Zhang, Yan; Ou, Yang-jia: The Design and Implementation of Host-based Intrusion Detection System. In: *International Symposium on Intelligent Information Technology and Security Informatics (IITSI)* IEEE, 2010, S. 595–598. http://dx.doi.org/10.1109/IITSI.2010.127

[Maa14] Maaten, Laurens van d.: Accelerating t-SNE using Tree-Based Algorithms. In: *The Journal of Machine Learning Research* 15 (2014), Nr. 1, S. 3221–3245

[MAB+08] McKeown, Nick; Anderson, Tom; Balakrishnan, Hari; Parulkar, Guru; Peterson, Larry; Rexford, Jennifer; Shenker, Scott; Turner, Jonathan: OpenFlow: Enabling Innovation in Campus Networks. In: *ACM SIGCOMM Computer Communication Review* 38 (2008), Nr. 2, S. 69–74. http://dx.doi.org/10.1145/1355734.1355746

[Mac67] MacQueen, James: Some Methods for Classification and Analysis of Multivariate Observations. In: *Berkeley Symposium on Mathematical Statistics and Probability* Bd. 1 Oakland, CA, USA, 1967, S. 281–297

[Mah03] Mahoney, Matthew V.: Network Traffic Anomaly Detection based on Packet Bytes. In: *ACM Symposium on Applied Computing* ACM, 2003, S. 346–350. http://dx.doi.org/10.1145/952532.952601

[MBM15] Małowidzki, Marek; Berezinski, P; Mazur, Michał: Network Intrusion Detection: Half a Kingdom for a Good Dataset. In: *NATO STO SAS-139 Workshop, Portugal*, 2015

[MC03] Mahoney, Matthew V.; Chan, Philip K.: An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection. In: *International Workshop on Recent Advances in Intrusion Detection* Springer, 2003, S. 220–237. http://dx.doi.org/10.1007/978-3-540-45248-5_13

[MC12] Murtagh, Fionn; Contreras, Pedro: Algorithms for hierarchical clustering: an overview. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2 (2012), Nr. 1, S. 86–97. http://dx.doi.org/10.1002/widm.53

[MCB13] Mathur, Suhas; Coskun, Baris; Balakrishnan, Suhrid: Detecting Hidden Enemy Lines in IP Address Space. In: *Workshop on New Security Paradigms* ACM, 2013, S. 19–30. http://dx.doi.org/10.1145/2535813.2535816

[MCC+13] Mikolov, Tomas; Chen, Kai; Corrado, Greg; Dean, Jeffrey: Efficient Estimation of Word Representations in Vector Space. In: *arXiv preprint arXiv:1301.3781* (2013), S. 1–12

[McC16] McCormick, Chris: *Word2Vec Tutorial - The Skip-Gram Model [Blogpost].* http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/. Version: 19.4.2016. – (Letzter Zugriff am 5 August 2020)

[McH00] McHugh, John: Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory. In: *ACM Transactions on Information and System Security (TISSEC)* 3 (2000), Nr. 4, S. 262–294. http://dx.doi.org/10.1145/382912.382923

[MCM+18] Macia-Fernandez, Gabriel; Camacho, José; Magan-Carrion, Roberto; Garcia-Teodoro, Pedro; Therón, Roberto: UGR'16: A New Dataset for the Evaluation of Cyclostationarity-Based Network IDSs. In: *Computers and Security* 73 (2018), S. 411–424. http://dx.doi.org/10.1016/j.cose.2017.11.004

[MH08] Maaten, Laurens van d.; Hinton, Geoffrey: Visualizing Data using t-SNE. In: *Journal of Machine Learning Research (JMLR)* 9 (2008), S. 2579–2605

[MIT] MIT Lincoln Laboratory: *Cyber Systems and Technology*. https://www.ll.mit.edu/ideval/docs/index.html. – (Letzter Zugriff am 5 August 2020)

[MJ15] Modi, Urvashi; Jain, Anurag: A survey of IDS classification using KDD CUP 99 dataset in WEKA. In: *International Journal of Scientific and Engineering Research* 6 (2015), Nr. 11, S. 947–954

[ML14] Murtagh, Fionn; Legendre, Pierre: Ward's Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward's Criterion? In: *Journal of classification* 31 (2014), Nr. 3, S. 274–295. http://dx.doi.org/10.1007/s00357-014-9161-z

[MLD15] Melamud, Oren; Levy, Omer; Dagan, Ido: A Simple Word Embedding Model for Lexical Substitution. In: *Workshop on Vector Space Modeling for Natural Language Processing, North American Chapter of the Association for Computational Linguistics – Human Language Technologies (NAACL HLT)*, 2015, S. 1–7. http://dx.doi.org/10.3115/v1/W15-1501

[MMS13] Molnár, Sándor; Megyesi, Peter; Szabo, Geza: How to Validate Traffic Generators? In: *IEEE International Conference on Communications Workshops (ICC)* IEEE, 2013, S. 1340–1344. http://dx.doi.org/10.1109/ICCW.2013.6649445

[MS15] Moustafa, Nour; Slay, Jill: UNSW-NB15: A Comprehensive Data set for Network Intrusion Detection systems (UNSW-NB15 Network Data Set). In: *Military Communications and Information Systems Conference (MilCIS)* IEEE, 2015, S. 1–6. http://dx.doi.org/10.1109/MilCIS.2015.7348942

[MSC+13] Mikolov, Tomas; Sutskever, Ilya; Chen, Kai; Corrado, Greg S.; Dean, Jeff: Distributed Representations of Words and Phrases and their Compositionality. In: *Advances in Neural Information Processing Systems (NIPS)*, 2013, S. 3111–3119

[MT18a] Mimura, Mamoru; Tanaka, Hidema: A Linguistic Approach Towards Intrusion Detection in Actual Proxy Logs. In: *International Conference on Information and Communications Security*, Springer International Publishing, 2018, S. 708–718. http://dx.doi.org/10.1007/978-3-030-01950-1_42

[MT18b] Mimura, Mamoru; Tanaka, Hidema: Reading Network Packets as a Natural Language for Intrusion Detection. In: *International Conference on Information*

Security and Cryptology (ICISC). Cham : Springer International Publishing, 2018, S. 339–350. http://dx.doi.org/10.1007/978-3-319-78556-1_19

[MWM18] Marino, Daniel L.; Wickramasinghe, Chathurika S.; Manic, Milos: An Adversarial Approach for Explainable AI in Intrusion Detection Systems. In: *Annual Conference of the IEEE Industrial Electronics Society (IECON)*, 2018, S. 3237–3243

[MXM+16] Meng, Guozhu; Xue, Yinxing; Mahinthan, Chandramohan; Narayanan, Annamalai; Liu, Yang; Zhang, Jie; Chen, Tieming: Mystique: Evolving Android Malware for Auditing Anti-Malware Tools. In: *Asia Conference on Computer and Communications Security* ACM, 2016, S. 365–376. http://dx.doi.org/10.1145/2897845.2897856

[Myr11] Myrrhe, Anke: Hacker stehlen Sony Millionen Kundendaten. In: *Der Tagesspiegel* (27.4.2011). https://www.tagesspiegel.de/medien/digitale-welt/datenklau-hacker-stehlen-sony-millionen-kundendaten/4102306.html. – (Letzter Zugriff am 5 August 2020)

[NKC+15] Najafabadi, Maryam M.; Khoshgoftaar, Taghi M.; Calvert, Chad; Kemp, Clifford: Detection of SSH Brute Force Attacks Using Aggregated Netflow Data. In: *IEEE International Conference on Machine Learning and Applications* IEEE, 2015, S. 283–288. http://dx.doi.org/10.1109/ICMLA.2015.20

[NKK+14] Najafabadi, Maryam M.; Khoshgoftaar, Taghi M.; Kemp, Clifford; Seliya, Naeem; Zuech, Richard: Machine Learning for Detecting Brute Force Attacks at the Network Level. In: *International Conference on Bioinformatics and Bioengineering (BIBE)* IEEE, 2014, S. 379–385. http://dx.doi.org/10.1109/BIBE.2014.73

[NKN+16] Najafabadi, Maryam M.; Khoshgoftaar, Taghi M.; Napolitano, Amri; Wheelus, Charles: RUDY Attack: Detection at the Network Level and Its Important Features. In: *International Florida Artificial Intelligence Research Society Conference*, 2016, S. 288–293

[NMY+18] Nisioti, Antonia; Mylonas, Alexios; Yoo, Paul D.; Katos, Vasilios: From Intrusion Detection to Attacker Attribution: A Comprehensive Survey of Unsupervised Methods. In: *IEEE Communications Surveys and Tutorials* 20 (2018), Nr. 4, S. 3369–3388. http://dx.doi.org/10.1109/COMST.2018.2854724

[NSA+08] Nychis, George; Sekar, Vyas; Andersen, David G.; Kim, Hyong; Zhang, Hui: An Empirical Evaluation of Entropy-based Traffic Anomaly Detection. In: *ACM SIGCOMM Conference on Internet measurement* ACM, 2008, S. 151–156. http://dx.doi.org/10.1145/1452520.1452539

[ÖE16] Özgür, Atilla; Erdem, Hamit: A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015. In: *PeerJ PrePrints 4:e1954v1* (2016)

[ORL+16] Otto, Florian; Ring, Markus; Landes, Dieter; Hotho, Andreas: Creation of Specific Flow-Based Training Data Sets for Usage Behaviour Classification. In: *European Conference on Cyber Warfare and Security (ECCWS)*, 2016, S. 437–440

[Owe10] Owezarski, Philippe: A database of anomalous traffic for assessing profile based IDS. In: *International Workshop on Traffic Monitoring and Analysis* Springer, 2010, S. 59–72. http://dx.doi.org/10.1007/978-3-642-12365-8_5

[PAB+05] Pang, Ruoming; Allman, Mark; Bennett, Mike; Lee, Jason; Paxson, Vern; Tierney, Brian: A First Look at Modern Enterprise Traffic. In: *ACM SIGCOMM Conference on Internet Measurement (IMC)*. Berkeley, CA, USA : USENIX Association, 2005, S. 15–28

[PAG+07] Peddabachigari, Sandhya; Abraham, Ajith; Grosan, Crina; Thomas, Johnson: Modeling intrusion detection system using hybrid intelligent systems. In: *Journal of Network and Computer Applications* 30 (2007), Nr. 1, S. 114–132. http://dx.doi.org/10.1016/j.jnca.2005.06.003

[PAP+06] Pang, Ruoming; Allman, Mark; Paxson, Vern; Lee, Jason: The Devil and Packet Trace Anonymization. In: *ACM SIGCOMM Computer Communication Review* 36 (2006), Nr. 1, S. 29–38. http://dx.doi.org/10.1145/1111322.1111330

[Pax99] Paxson, Vern: Bro: A System for Detecting Network Intruders in Real-Time. In: *Computer Networks* 31 (1999), Nr. 23-24, S. 2435–2463. http://dx.doi.org/10.1016/S1389-1286(99)00112-7

[Pos81] Postel, J.: Internet Control Message Protocol / Internet Engineering Task Force. Version: September 1981. https://tools.ietf.org/html/rfc792. IETF, September 1981 (792). – RFC. – (Letzter Zugriff am 5 August 2020)

[PP03] Pang, Ruoming; Paxson, Vern: A High-level Programming Environment for Packet Trace Anonymization and Transformation. In: *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ACM, 2003, S. 339–351. http://dx.doi.org/10.1145/863955.863994

[PP07a] Panda, Mrutyunjaya; Patra, Manas R.: Network Intrusion Detection using Naive Bayes. In: *International Journal of Computer Science and Network Security* 7 (2007), Nr. 12, S. 258–263

[PP07b] Patcha, Animesh; Park, Jung-Min: An overview of anomaly detection techniques: Existing solutions and latest technological trends. In: *Computer Networks* 51 (2007), Nr. 12, S. 3448–3470. http://dx.doi.org/10.1016/j.comnet.2007.02.001

[PPM01] Phaal, Peter; Panchen, Sonia; McKee, Neil: InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks / Internet Engineering Task Force. Version: September 2001. https://tools.ietf.org/html/rfc3176. IETF, September 2001 (3176). – RFC. – (Letzter Zugriff am 5 August 2020)

[PQW10] Patel, Ahmed; Qassim, Qais; Wills, Christopher: A survey of intrusion detection and prevention systems. In: *Information Management and Computer Security* 18 (2010), Nr. 4, S. 277–290. http://dx.doi.org/10.1108/09685221011079199

[PRU+18] Preuer, Kristina; Renz, Philipp; Unterthiner, Thomas; Hochreiter, Sepp; Klambauer, Günter: Fréchet ChemNet Distance: A Metric for Generative Models for Molecules in Drug Discovery. In: *Journal of Chemical Information and Modeling* 58 (2018), Nr. 9, S. 1736–1741. http://dx.doi.org/10.1021/acs.jcim.8b00234

[PSL+15] Park, Simon Soon-Hyoung; Song, Justin J.; Lee, James Jung-Hoon; Lee, Wookey; Ree, Sangbok: How to measure similarity for multiple categorical data sets? In: *Multimedia Tools and Applications* 74 (2015), Nr. 10, S. 3489–3505. http://dx.doi.org/10.1007/s11042-014-1914-5

[Qui86] Quinlan, Ross: Induction of Decision Trees. In: *Machine learning* 1 (1986), Nr. 1, S. 81–106. http://dx.doi.org/10.1007/BF00116251

[Qui93] Quinlan, John R.: *C4. 5: Programs for Empirical Learning*. 1. Auflage. 1993

[RE15] Ring, Matthias; Eskofier, Björn: Optimal Feature Selection for Nonlinear Data using Branch-and-Bound in Kernel Space. In: *Pattern Recognition Letters* 68 (2015), S. 56–62. http://dx.doi.org/10.1016/j.patrec.2015.08.007

[RG18] Rigaki, Maria; Garcia, Sebastian: Bringing a GAN to a Knife-fight: Adapting Malware Communication to Avoid Detection. In: *Deep Learning and Security Workshop, IEEE Security and Privacy Workshops (SPW)*, 2018, S. 70–75. http://dx.doi.org/10.1109/SPW.2018.00019

[RH07] Rosenberg, Andrew; Hirschberg, Julia: V-measure: A conditional entropy-based external cluster evaluation measure. In: *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007, S. 410–420

[RHW86] Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J.: Learning representations by back-propagating errors. In: *Nature* 323 (1986), Nr. 6088, S. 533–536. http://dx.doi.org/10.1038/323533a0

[Rin17] Ring, Markus: *CIDDS - Coburg Intrusion Detection Data Sets [Github Repository]*. https://github.com/markusring/CIDDS. Version: 2017. – (Letzter Zugriff am 5 August 2020)

[RK14] Rygielski, Piotr; Kounev, Samuel: Data Center Network Throughput Analysis Using Queueing Petri Nets. In: *IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW)*, 2014, S. 100–105. http://dx.doi.org/10.1109/ICDCSW.2014.11

[RKZ13] Rygielski, Piotr; Kounev, Samuel; Zschaler, Steffen: Model-Based Throughput Prediction in Data Center Networks. In: *IEEE International Workshop on Measurements Networking*, 2013, S. 167–172. http://dx.doi.org/10.1109/IWMN.2013.6663797

[RL06] Rieck, Konrad; Laskov, Pavel: Detecting Unknown Network Attacks using Language Models. In: *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)* Springer, 2006, S. 74–90. http://dx.doi.org/10.1007/11790754_5

[RL07] Rieck, Konrad; Laskov, Pavel: Language models for detection of unknown attacks in network traffic. In: *Journal in Computer Virology* 2 (2007), Nr. 4, S. 243–256. http://dx.doi.org/10.1007/s11416-006-0030-0

[RLD+17] Ring, Markus; Landes, Dieter; Dallmann, Alexander; Hotho, Andreas: IP2Vec: Learning Similarities between IP Adresses. In: *Workshop on Data Mining for Cyber Security (DMCS), International Conference on Data Mining Workshops (ICDMW)*, IEEE, 2017, S. 657–666. http://dx.doi.org/10.1109/ICDMW.2017.93

[RLH15] Ring, Markus; Landes, Dieter; Hotho, Andreas: Automatic Threshold Calculation for the Categorical Distance Measure ConDist. In: *LWA 2015 Workshops: KDML, FGWM, IR, and FGDB*, 2015, S. 52–63

[RLH18] Ring, Markus; Landes, Dieter; Hotho, Andreas: Detection of slow port scans in flow-based network traffic. In: *PLOS ONE* 13 (2018), 09, Nr. 9, S. 1–18. http://dx.doi.org/10.1371/journal.pone.0204507

[RMC16] Radford, Alec; Metz, Luke; Chintala, Soumith: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In: *International Conference on Learning Representations (ICLR)*, 2016

[ROB+15] Ring, Markus; Otto, Florian; Becker, Martin; Niebler, Thomas; Landes, Dieter; Hotho, Andreas: ConDist: A Context-Driven Categorical Distance Measure. In: *European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)* Springer, 2015, S. 251–266. http://dx.doi.org/10.1007/978-3-319-23528-8_16

[Rou87] Rousseeuw, Peter J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. In: *Journal of Computational and Applied Mathematics* 20 (1987), S. 53–65. http://dx.doi.org/10.1016/0377-0427(87)90125-7

[RSL+19] Ring, Markus; Schlör, Daniel; Landes, Dieter; Hotho, Andreas: Flow-based Network Traffic Generation using Generative Adversarial Networks. In: *Computer and Security* 82 (2019), S. 156–172. http://dx.doi.org/10.1016/j.cose.2018.12.012. – ISSN 0167–4048

[RWG+17a] Ring, Markus; Wunderlich, Sarah; Grüdl, Dominik; Landes, Dieter; Hotho, Andreas: A Toolset for Intrusion and Insider Threat Detection. In: Palomares, Iván (Hrsg.); Kalutarage, Harsha (Hrsg.); Huang, Yan (Hrsg.): *Data Analytics and Decision Support for Cybersecurity: Trends, Methodologies and Applications*, Springer, 2017, S. 3–31. http://dx.doi.org/10.1007/978-3-319-59439-2_1

[RWG+17b] Ring, Markus; Wunderlich, Sarah; Grüdl, Dominik; Landes, Dieter; Hotho, Andreas: Creation of Flow-Based Data Sets for Intrusion Detection. In: *Journal of Information Warfare (JIW)* 16 (2017), S. 40–53

[RWG+17c] Ring, Markus; Wunderlich, Sarah; Grüdl, Dominik; Landes, Dieter; Hotho, Andreas: Flow-based benchmark data sets for intrusion detection. In: *European Conference on Cyber Warfare and Security (ECCWS)*. ACPI, 2017, S. 361–369

[RWS+19] Ring, Markus; Wunderlich, Sarah; Scheuring, Deniz; Landes, Dieter; Hotho, Andreas: A Survey of Network-based Intrusion Detection Data Sets. In: *Computers and Security* 96 (2019), S. 147–167. http://dx.doi.org/10.1016/j.cose.2019.06.005

[SAW15] Shirkhorshidi, Ali S.; Aghabozorgi, Saeed; Wah, Teh Y.: A Comparison Study on Similarity and Dissimilarity Measures in Clustering Continuous Data. In: *PLOS ONE* 10 (2015), 12, Nr. 12, S. 1–20. http://dx.doi.org/10.1371/journal.pone.0144059

[SB04] Sommers, Joel; Barford, Paul: Self-Configuring Network Traffic Generation. In: *ACM Internet Measurement Conference (ACM IMC)* ACM, 2004, S. 68–81. http://dx.doi.org/10.1145/1028788.1028798

[SBD+17] Sapienza, Anna; Bessi, Alessandro; Damodaran, Saranya; Shakarian, Paulo; Lerman, Kristina; Ferrara, Emilio: Early Warnings of Cyber Threats in Online Discussions. In: *Workshop on Data Mining for Cyber Security (DMCS), International Conference on Data Mining Workshops (ICDMW)* IEEE, 2017, S. 667–674. http://dx.doi.org/10.1109/ICDMW.2017.94

[SBL+18] Sajjadi, Mehdi S. M.; Bachem, Olivier; Lucic, Mario; Bousquet, Olivier; Gelly, Sylvain: Assessing Generative Models via Precision and Recall. In: *Advances in Neural Information Processing Systems*. 2018, S. 5234–5243

[SF05] Schmidberger, Gabi; Frank, Eibe: Unsupervised Discretization using Tree-based Density Estimation. In: *European Conference on Principles of Data Mining and Knowledge Discovery* Springer, 2005, S. 240–251. http://dx.doi.org/10.1007/11564126_26

[SG03] Strehl, Alexander; Ghosh, Joydeep: Cluster Ensembles — a Knowledge Reuse Framework for Combining Multiple Partitions. In: *The Journal of Machine Learning Research* 3 (2003), S. 583–617. http://dx.doi.org/10.1162/153244303321897735

[SGZ+16] Salimans, Tim; Goodfellow, Ian; Zaremba, Wojciech; Cheung, Vicki; Radford, Alec; Chen, Xi: Improved Techniques for Training GANs. In: *Advances in Neural Information Processing Systems (NIPS)*, 2016, S. 2234–2242

[She00] Shearer, Colin: The CRISP-DM Model: The New Blueprint for Data Mining. In: *Journal of Data Warehousing* 5 (2000), Nr. 4, S. 13–22

[SHM02] Staniford, Stuart; Hoagland, James A.; McAlerney, Joseph M.: Practical automated detection of stealthy portscans. In: *Journal of Computer Security* 10 (2002), Nr. 1-2, S. 105–136

[SHM+16] Silver, David; Huang, Aja; Maddison, Chris J.; Guez, Arthur; Sifre, Laurent; Driessche, George van d.; Schrittwieser, Julian; Antonoglou, Ioannis; Panneershelvam, Veda; Lanctot, Marc; Dieleman, Sander; Grewe, Dominik; Nham, John; Kalchbrenner, Nal; Sutskever, Ilya; Lillicrap, Timothy; Leach, Madeleine; Kavukcuoglu, Koray; Graepel, Thore; Hassabis, Demis: Mastering the game of Go with deep neural networks and tree search. In: *Nature* 529 (2016), Nr. 7587, S. 484–489. http://dx.doi.org/10.1038/nature16961

[SKS15] Singh, Raman; Kumar, Harish; Singla, Ravinder K.: A Reference Dataset for Network Traffic Activity Based Intrusion Detection System. In: *International Journal of Computers Communications and Control* 10 (2015), Nr. 3, S. 390–402. http://dx.doi.org/10.15837/ijccc.2015.3.1924

[ŠKŽ15] Šourek, Gustav; Kuželka, Ondřej; Železnỳ, Filip: Learning to detect network intrusion from a few labeled events and background traffic. In: *International Conference on Autonomous Infrastructure, Management and Security (IFIP)* Springer, 2015, S. 73–86. http://dx.doi.org/10.1007/978-3-319-20034-7_9

[SL05] Spillner, Andreas; Linz, Tilo: *Basiswissen Softwaretest.* 3. Auflage. dpunkt, 2005. – ISBN 3–89864–358–1

[SL06] Skoudis, Edward; Liston, Tom: *Counter Hack Reloaded: A Step-by-step Guide to Computer Attacks and Effective Defenses.* 2. Auflage. Prentice Hall Press, 2006. – ISBN 9780131481046

[SLG18] Sharafaldin, Iman; Lashkari, A H.; Ghorbani, Ali A.: Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In: *International Conference on Information Systems Security and Privacy (ICISSP)*, 2018, S. 108–116. http://dx.doi.org/10.5220/0006639801080116

[SLP+15] Sherry, Justine; Lan, Chang; Popa, Raluca A.; Ratnasamy, Sylvia: Blindbox: Deep Packet Inspection over Encrypted Traffic. In: *ACM SIGCOMM Computer Communication Review* 45 (2015), Nr. 4, S. 213–226. http://dx.doi.org/10.1145/2785956.2787502

[SN13] Solms, Rossouw v.; Niekerk, Johan v.: From information security to cyber security. In: *Computers and Security* 38 (2013), S. 97–102. http://dx.doi.org/10.1016/j.cose.2013.04.004

[SOC+09] Sangster, Benjamin; O'Connor, TJ; Cook, Thomas; Fanelli, Robert; Dean, Erik; Morrell, Christopher; Conti, Gregory J.: Toward Instrumenting Network Warfare Competitions to Generate Labeled Datasets. In: *Workshop on Cyber Security Experimentation and Test (CSET)*, 2009

[SOM+08] Szabó, Géza; Orincsay, Dániel; Malomsoky, Szabolcs; Szabó, István: On the Validation of Traffic Classification Algorithms. In: *International Conference on Passive and Active Network Measurement* Springer, 2008, S. 72–81. http://dx.doi.org/10.1007/978-3-540-79232-1_8

[SP10] Sommer, Robin; Paxson, Vern: Outside the Closed World: On Using Machine Learning For Network Intrusion Detection. In: *IEEE Symposium on Security and Privacy* IEEE, 2010, S. 305–316. http://dx.doi.org/10.1109/SP.2010.25

[SP11] Sperotto, Anna; Pras, Aiko: Flow-based Intrusion Detection. In: *IFIP/IEEE International Symposium on Integrated Network Management (IM)* IEEE, 2011, S. 958–963. http://dx.doi.org/10.1109/INM.2011.5990529

[SP14] Stevanovic, Matija; Pedersen, Jens M.: An efficient flow-based botnet detection using supervised machine learning. In: *IEEE International Conference*

on Computing, Networking and Communications IEEE, 2014, S. 797–801. http://dx.doi.org/10.1109/ICCNC.2014.6785439

[SP15] Stevanovic, Matija; Pedersen, Jens M.: An analysis of network traffic classification for botnet detection. In: *IEEE International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)* IEEE, 2015, S. 1–8. http://dx.doi.org/10.1109/CyberSA.2015.7361120

[SRH+15] Santanna, Joes J.; Rijswijk-Deij, Roland van; Hofstede, Rick; Sperotto, Anna; Wierbosch, Mark; Granville, Lisandro Z.; Pras, Aiko: Booters - An analysis of DDoS-as-a-service attacks. In: *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, S. 243–251. http://dx.doi.org/10.1109/INM.2015.7140298

[SRP15] Stiborek, Jan; Rehák, Martin; Pevnỳ, Tomáš: Towards scalable network host simulation. In: *International Workshop on Agents and Cybersecurity*, 2015, S. 27–35

[SS01] Schölkopf, Bernhard; Smola, Alexander: *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. 1. Auflage. MIT press, 2001. – ISBN 9780262256933

[SSB+09] Sperotto, Anna; Sadre, Ramin; Boer, Pieter-Tjerk d.; Pras, Aiko: Hidden Markov Model modeling of SSH brute-force attacks. In: *International Workshop on Distributed Systems: Operations and Management* Springer, 2009, S. 164–176. http://dx.doi.org/10.1007/978-3-642-04989-7_13

[SSD15] Sabottke, Carl; Suciu, Octavian; Dumitras, Tudor: Vulnerability Disclosure in the Age of Social Media: Exploiting Twitter for Predicting Real-World Exploits. In: *USENIX Security Symposium*, 2015, S. 1041–1056

[SSF16] Safa, Nader S.; Solms, Rossouw v.; Furnell, Steven: Information security policy compliance model in organizations. In: *Computers and Security* 56 (2016), S. 70–82. http://dx.doi.org/10.1016/j.cose.2015.10.006

[SSG18] Sharma, Rohini; Singla, Ravinder K.; Guleria, Ajay: A New Labeled Flow-based DNS Dataset for Anomaly Detection: PUF Dataset. In: *International Conference on Computational Intelligence and Data Science (ICCIDS)* Bd. 132, 2018, S. 1458–1466. http://dx.doi.org/10.1016/j.procs.2018.05.079

[SSK+10] Siska, Peter; Stoecklin, Marc P.; Kind, Andreas; Braun, Torsten: A Flow Trace Generator using Graph-based Traffic Classification Techniques. In: *International Wireless Communications and Mobile Computing Conference (IWCMC)* ACM, 2010, S. 457–462. http://dx.doi.org/10.1145/1815396.1815503

[SSS+10] Sperotto, Anna; Schaffrath, Gregor; Sadre, Ramin; Morariu, Cristian; Pras, Aiko; Stiller, Burkhard: An Overview of IP Flow-based Intrusion Detection. In: *IEEE Communications Surveys and Tutorials* 12 (2010), Nr. 3, S. 343–356. http://dx.doi.org/10.1109/SURV.2010.032210.00054

[SST+12] Shiravi, Ali; Shiravi, Hadi; Tavallaee, Mahbod; Ghorbani, Ali A.: Toward developing a systematic approach to generate benchmark datasets for intrusion

detection. In: *Computers and Security* 31 (2012), Nr. 3, S. 357–374. http://dx.doi.org/10.1016/j.cose.2011.12.012

[SSV+09] Sperotto, Anna; Sadre, Ramin; Vliet, Frank v.; Pras, Aiko: A Labeled Data Set For Flow-based Intrusion Detection. In: *International Workshop on IP Operations and Management* Springer, 2009, S. 39–50. http://dx.doi.org/10.1007/978-3-642-04968-2_4

[STG+11] Saad, Sherif; Traore, Issa; Ghorbani, Ali; Sayed, Bassam; Zhao, David; Lu, Wei; Felix, John; Hakimian, Payman: Detecting P2P Botnets through Network Behavior Analysis and Machine Learning. In: *International Conference on Privacy, Security and Trust (PST)* IEEE, 2011, S. 174–180. http://dx.doi.org/10.1109/PST.2011.5971980

[STO+11] Song, Jungsuk; Takakura, Hiroki; Okabe, Yasuo; Eto, Masashi; Inoue, Daisuke; Nakao, Koji: Statistical Analysis of Honeypot Data and Building of Kyoto 2006+ Dataset for NIDS Evaluation. In: *Workshop on Building Analysis Datasets and Gathering Experience Returns for Security* ACM, 2011, S. 29–36. http://dx.doi.org/10.1145/1978672.1978676

[SVI+16] Szegedy, Christian; Vanhoucke, Vincent; Ioffe, Sergey; Shlens, Jon; Wojna, Zbigniew: Rethinking the Inception Architecture for Computer Vision. In: *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, S. 2818–2826. http://dx.doi.org/10.1109/CVPR.2016.308

[SW86] Stanfill, Craig; Waltz, David: Toward Memory-Based Reasoning. In: *Communications of the ACM* 29 (1986), Nr. 12, S. 1213–1228. http://dx.doi.org/10.1145/7902.7906

[SYB06] Sridharan, Avinash; Ye, Tao; Bhattacharyya, Supratik: Connectionless Port Scan Detection on the Backbone. In: *IEEE International Performance Computing and Communications Conference* IEEE, 2006, S. 10–19. http://dx.doi.org/10.1109/.2006.1629454

[Sym18] Symantec: Internet Security Threat Report (ISTR. 23 (2018), March. https://docs.broadcom.com/doc/istr-23-2018-en. – Letzter Zugriff am 5 August 2020

[TAL14] Tang, Jiliang; Alelyani, Salem; Liu, Huan: Feature Selection for Classification: A Review. In: Aggarwal, Charu C. (Hrsg.): *Data Classification: Algorithms and Applications*. 1. Auflage. CRC Press, 2014. – ISBN 9781466586758, S. 37–64

[TBL+09] Tavallaee, Mahbod; Bagheri, Ebrahim; Lu, Wei; Ghorbani, A. Ali: A Detailed Analysis of the KDD CUP 99 Data Set. In: *IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, 2009, S. 1–6. http://dx.doi.org/10.1109/CISDA.2009.5356528

[THL+09] Tsai, Chih-Fong; Hsu, Yu-Feng; Lin, Chia-Ying; Lin, Wei-Yang: Intrusion detection by machine learning: A review. In: *Expert Systems with Applications* 36 (2009), Nr. 10, S. 11994–12000. http://dx.doi.org/10.1016/j.eswa.2009.05.029

[TJH12] Tran, Quang A.; Jiang, Frank; Hu, Jiankun: A Real-Time NetFlow-based Intrusion Detection System with Improved BBNN and High-Frequency Field Programmable Gate Arrays. In: *IEEE International Conference on Trust, Security and Privacy in Computing and Communications* IEEE, 2012, S. 201–208. http://dx.doi.org/10.1109/TrustCom.2012.51

[TK17] Thaseen, Ikram S.; Kumar, Cherukuri A.: Intrusion detection model using fusion of chi-square feature selection and multi class SVM. In: *Journal of King Saud University - Computer and Information Sciences* 29 (2017), Nr. 4, S. 462–472. http://dx.doi.org/10.1016/j.jksuci.2015.12.004

[TKH18] Turcotte, Melissa J. M.; Kent, Alexander D.; Hash, Curtis: Unified Host and Network Data Set. In: *Data Science for Cyber-Security*, World Scientific, Nov 2018, S. 1–22. http://dx.doi.org/10.1142/9781786345646_001

[TMM+16] Tang, Tuan A.; Mhamdi, Lotfi; McLernon, Des; Zaidi, Syed Ali R.; Ghogho, Mounir: Deep Learning Approach for Network Intrusion Detection in Software Defined Networking. In: *IEEE International Conference on Wireless Networks and Mobile Communications (WINCOM)* IEEE, 2016, S. 258–263. http://dx.doi.org/10.1109/WINCOM.2016.7777224

[TRM09] Tajbakhsh, Arman; Rahmati, Mohammad; Mirzaei, Abdolreza: Intrusion Detection Using Fuzzy Association Rules. In: *Applied Soft Computing* 9 (2009), Nr. 2, S. 462–469. http://dx.doi.org/10.1016/j.asoc.2008.06.001

[TSK06] Tan, Pang-Ning; Steinbach, Michael; Kumar, Vipin: *Introduction to Data Mining*. 1. Auflage. Pearson Addison Wesley, 2006. – ISBN 0321321367

[TSS14] Tang, Adrian; Sethumadhavan, Simha; Stolfo, Salvatore J.: Unsupervised Anomaly-based Malware Detection using Hardware Features. In: *International Workshop on Recent Advances in Intrusion Detection* Springer, 2014, S. 109–129. http://dx.doi.org/10.1007/978-3-319-11379-1_6

[TW11] Tanenbaum, Andrew S.; Wetherall, David: *Computer Networks*. 5. Auflage. Prentice Hall, 2011. – ISBN 978–0132126953

[Uni99] University of California Irvine: *KDD Cup 1999 Data*. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. Version: 1999. – (Letzter Zugriff am 5 August 2020)

[UPK+09] Udhayan, J; Prabu, M. M.; KrishnanAravinda, V.; Anitha, R: Reconnaissance Scan Detection Heuristics to disrupt the pre-attack information gathering. In: *International Conference on Network and Service Security* IEEE, 2009, S. 1–5

[USB17] Umer, Muhammad F.; Sher, Muhammad; Bi, Yaxin: Flow-based intrusion detection: Techniques and challenges. In: *Computers and Security* 70 (2017), S. 238–254. http://dx.doi.org/10.1016/j.cose.2017.05.009

[VCI+99] Vivo, Marco d.; Carrasco, Eddy; Isern, Germinal; Vivo, Gabriela O d.: A Review of Port Scanning Techniques. In: *ACM SIGCOMM Computer Communication Review* 29 (1999), Nr. 2, S. 41–48. http://dx.doi.org/10.1145/505733.505737

[VCM+16] Vasilomanolakis, Emmanouil; Cordero, Carlos G.; Milanov, Nikolay; Mühlhäuser, Max: Towards the creation of synthetic, yet realistic, intrusion detection datasets. In: *IEEE Network Operations and Management Symposium (NOMS)* IEEE, 2016, S. 1209–1214. http://dx.doi.org/10.1109/NOMS.2016.7502989

[VHS11] Vasudevan, Aravind R.; Harshini, E.; Selvakumar, S.: SSENet-2011: A Network Intrusion Detection System dataset and its comparison with KDD CUP 99 dataset. In: *Second Asian Himalayas International Conference on Internet (AHICI)*, 2011, S. 1–5. http://dx.doi.org/10.1109/AHICI.2011.6113948

[VM14] Valentín, Kristián; Malỳ, Michal: Network Firewall using Artificial Neural Networks. In: *Computing and Informatics* 32 (2014), Nr. 6, S. 1312–1327

[VSO17] Viegas, Eduardo K.; Santin, Altair O.; Oliveira, Luiz S.: Toward a Reliable Anomaly-Based Intrusion Detection in Real-World Environments. In: *Computer Networks* 127 (2017), S. 200–216. http://dx.doi.org/10.1016/j.comnet.2017.08.013

[WBS15] Weller-Fahy, David J.; Borghetti, Brett J.; Sodemann, Angela A.: A Survey of Distance and Similarity Measures Used Within Network Intrusion Anomaly Detection. In: *IEEE Communications Surveys and Tutorials* 17 (2015), Nr. 1, S. 70–91. http://dx.doi.org/10.1109/COMST.2014.2336610

[WDA+16] Wilkinson, Mark D.; Dumontier, Michel; Aalbersberg, IJsbrand J.; Appleton, Gabrielle; Axton, Myles; Baak, Arie; Blomberg, Niklas; Boiten, Jan-Willem; Silva Santos, Luiz B.; Bourne, Philip E.; Bouwman, Jildau; Brookes, Anthony J.; Clark, Tim; Crosas, Merce; Dillo, Ingrid; Dumon, Olivier; Edmunds, Scott; Evelo, Chris T.; Finkers, Richard; Gonzalez-Beltran, Alejandra; Gray, Alasdair J.; Groth, Paul; Goble, Carole; Grethe, Jeffrey S.; Heringa, Jaap; Hoen, Peter A.; Hooft, Rob; Kuhn, Tobias; Kok, Ruben; Kok, Joost; Lusher, Scott J.; Martone, Maryann E.; Mons, Albert; Packer, Abel L.; Persson, Bengt; Rocca-Serra, Philippe; Roos, Marco; Schaik, Rene van; Sansone, Susanna-Assunta; Schultes, Erik; Sengstag, Thierry; Slater, Ted; Strawn, George; Swertz, Morris A.; Thompson, Mark; Lei, Johan van d.; Mulligen, Erik van; Velterop, Jan; Waagmeester, Andra; Wittenburg, Peter; Wolstencroft, Katherine; Zhao, Jun; Mons, Barend: The FAIR Guiding Principles for scientific data management and stewardship. In: *Scientific Data* 3 (2016). http://dx.doi.org/10.1038/sdata.2016.18

[WFH05] Witten, Ian H.; Frank, Eibe; Hall, Mark A.: *Data Mining: Practical Machine Learning Tools and Techniques.* 2. Auflage. Morgan Kaufmann, 2005. – ISBN 0120884070

[WFS+11] Wagner, Cynthia; François, Jérôme; State, Radu; Engel, Thomas: Machine Learning Approach for IP-Flow Record Anomaly Detection. In: *International Conference on Research in Networking* Springer, 2011, S. 28–39. http://dx.doi.org/10.1007/978-3-642-20757-0_3

[WGE+15] Webster, Ashton; Gratian, Margaret; Eckenrod, Ryan; Patel, Daven; Cukier, Michel: An Improved Method for Anomaly-Based Network Scan Detection.

In: *International Conference on Security and Privacy in Communication Systems* Springer, 2015, S. 385–400. http://dx.doi.org/10.1007/978-3-319-28865-9_21

[WHZ11] Winter, Philipp; Hermann, Eckehard; Zeilinger, Markus: Inductive Intrusion Detection in Flow-Based Network Data Using One-Class Support Vector Machines. In: *International Conference on New Technologies, Mobility and Security* IEEE, 2011, S. 1–5. http://dx.doi.org/10.1109/NTMS.2011.5720582

[Wil45] Wilcoxon, Frank: Individual Comparisons by Ranking Methods. In: *Biometrics Bulletin* 1 (1945), Nr. 6, S. 80–83. http://dx.doi.org/10.2307/3001968

[WIZ15] Weiss, Sholom M.; Indurkhya, Nitin; Zhang, Tong: *Fundamentals of Predictive Text Mining*. 2. Auflage. Springer, 2015. – ISBN 978–1447167495

[WJ63] Ward Jr, Joe H.: Hierarchical Grouping to Optimize an Objective Function. In: *Journal of the American Statistical Association* 58 (1963), Nr. 301, S. 236–244. http://dx.doi.org/10.1080/01621459.1963.10500845

[WKZ+14] Wheelus, Charles; Khoshgoftaar, Taghi M.; Zuech, Richard; Najafabadi, Maryam M.: A Session Based Approach for Aggregating Network Traffic Data - The SANTA Dataset. In: *IEEE International Conference on Bioinformatics and Bioengineering (BIBE)* IEEE, 2014, S. 369–378. http://dx.doi.org/10.1109/BIBE.2014.72

[WM97] Wilson, Randall; Martinez, Tony: Improved Heterogeneous Distance Functions. In: *Journal of Artificial Intelligence Research (JAIR)* 6 (1997), S. 1–34. http://dx.doi.org/DigitalObjectIdentifierSystem.html

[WP17] Wang, Jing; Paschalidis, Ioannis C.: Botnet Detection Based on Anomaly and Community Detection. In: *IEEE Transactions on Control of Network Systems* 4 (2017), Nr. 2, S. 392–404. http://dx.doi.org/10.1109/TCNS.2016.2532804

[XCS+16] Xu, Chengcheng; Chen, Shuhui; Su, Jinshu; Yiu, Siu M.; Hui, Lucas C. K.: A Survey on Regular Expression Matching for Deep Packet Inspection: Applications, Algorithms, and Hardware Platforms. In: *IEEE Communications Surveys Tutorials* 18 (2016), Nr. 4, S. 2991–3029. http://dx.doi.org/10.1109/COMST.2016.2566669

[XFA+02] Xu, Jun; Fan, Jinliang; Ammar, Mostafa H.; Moon, Sue B.: Prefix-Preserving IP Address Anonymization: Measurement-based security evaluation and a new cryptography-based Scheme. In: *IEEE International Conference on Network Protocols* IEEE, 2002, S. 280–289. http://dx.doi.org/10.1109/ICNP.2002.1181415

[YL03] Yu, Lei; Liu, Huan: Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution. In: *International Conference on Machine Learning (ICML)*, 2003, S. 856–863

[YZL+18] Yin, Chuanlong; Zhu, Yuefei; Liu, Shengli; Fei, Jinlong; Zhang, Hetong: An Enhancing Framework for Botnet Detection Using Generative Adversarial Networks. In: *International Conference on Artificial Intelligence and Big Data (ICAIBD)*, 2018, S. 228–234. http://dx.doi.org/10.1109/ICAIBD.2018.8396200

[YZW+17] Yu, Lantao; Zhang, Weinan; Wang, Jun; Yu, Yong: SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In: *Conference on Artificial Intelligence (AAAI)* AAAI Press, 2017, S. 2852–2858

[ZF09] Zhang, Yu; Fang, Binxing: A Novel Approach to Scan Detection on the Backbone. In: *International Conference on Information Technology: New Generations (ITNG)* IEEE, 2009, S. 16–21. http://dx.doi.org/10.1109/ITNG.2009.16

[ZGC16] Zhang, Yizhe; Gan, Zhe; Carin, Lawrence: Generating Text via Adversarial Training. In: *NIPS Workshop on Adversarial Training* Bd. 21, 2016

[ZKS+15] Zuech, Richard; Khoshgoftaar, Taghi; Seliya, Naeem; Najafabadi, Maryam; Kemp, Clifford: A New Intrusion Detection Benchmarking System. In: *International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, AAAI Press, 2015, S. 252–256

[ZTS+13] Zhao, David; Traore, Issa; Sayed, Bassam; Lu, Wei; Saad, Sherif; Ghorbani, Ali; Garant, Dan: Botnet detection based on traffic behavior analysis and flow intervals. In: *Computers and Security* 39 (2013), S. 2–16. http://dx.doi.org/10.1016/j.cose.2013.04.007

[ZXW05] Zhang, Yu-Fang; Xiong, Zhong-Yang; Wang, Xiu-Qiong: Distributed Intrusion Detection based on Clustering. In: *IEEE International Conference on Machine Learning and Cybernetics* Bd. 4 IEEE, 2005, S. 2379–2383. http://dx.doi.org/10.1109/ICMLC.2005.1527342

[ZZH08] Zhang, Jiong; Zulkernine, Mohammad; Haque, Anwar: Random-Forests-Based Network Intrusion Detection Systems. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38 (2008), Nr. 5, S. 649–659. http://dx.doi.org/10.1109/TSMCC.2008.923876

[ZZS+18] Zheng, Yu-Jun; Zhou, Xiao-Han; Sheng, Wei-Guo; Xue, Yu; Chen, Sheng-Yong: Generative adversarial network based telecom fraud detection at the receiving bank. In: *Neural Networks* 102 (2018), S. 78–86. http://dx.doi.org/10.1016/j.neunet.2018.02.015