UNIVERSITÄT
LEIPZIG

Institut für Informatik
Fakultät für Mathematik und Informatik
Professur für Data Privacy and Security

Multimodale Datenfusion zur Klassifikation von Angriffsphasen in Advanced Persistent Threat Attacken

Masterarbeit

vorgelegt von:
Simon Mellert

Matrikelnummer:
3785150

Betreuer:
Prof. Dr.-Ing. habil. Erik Buchmann
Badr-Eddine Bouhlal

in Kooperation mit

DLR
Data Science

Leipzig, 17.12.2025

Abstract

Die vorliegende Arbeit untersucht den Einsatz multimodaler Fusion von Informationen aus Netzwerkaufzeichnungen und Host-Log-Nachrichten zur verbesserten Klassifikation von Netzwerk-Flows vor dem Hintergrund zunehmender Bedrohungen durch Advanced Persistent Threats (APTs) und aktueller moderner Natural-Language-Processing-Modelle. Im Rahmen der Untersuchung werden verschiedene Fusionsansätze evaluiert, die sowohl statistische Merkmale von Netzwerk-Flows als auch unterschiedliche Repräsentationen von Log-Nachrichten einbeziehen. Für die Textdaten werden dabei moderne kontextualisierte Embeddings sowie TF-IDF-skalierte Bag-of-Words-Embeddings systematisch miteinander verglichen. Auf Basis der verfügbaren Datensätze wird eine Trainingspipeline entwickelt, die eine effiziente Optimierung der Modellparameter sowie eine reproduzierbare Evaluierung sicherstellt. Die Ergebnisse werden zudem nach APT-Angriffsphasen aufgeschlüsselt und zeigen, dass multimodale Fusion abhängig von den Modellen und der Fusionsmethode bei der Erkennung von Aktivitäten in bestimmten Phasen der Angriffe einen Vorteil gegenüber unimodalen Ansätzen bietet. Die Arbeit demonstriert damit, wie durch den kombinierten Einsatz moderner Embedding-Verfahren und multimodaler Modellierungsstrategien heterogene sicherheitsrelevante Daten effektiv für die Erkennung von APT-Angriffen genutzt werden können.

Simon Mellert
3785150

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Rechnernetze bilden als kritische Infrastruktur das Rückgrat vieler Unternehmen und öffentlicher Einrichtungen. Sie realisieren globale Kommunikation und den Austausch sensibler Daten, womit sie den Ablauf moderner Wirtschafts- und Verwaltungsprozesse ermöglichen. Ein Ausfall oder eine Kompromittierung dieser Systeme kann weitreichende Folgen haben, von Produktionsausfällen über wirtschaftliche Schäden bis hin zu Sicherheitsrisiken für Menschen und Umwelt [1].

Die ökonomische Relevanz von Cyberangriffen ist hoch. Laut der Wirtschaftsschutz-Studie des Branchenverbands der deutschen Informations- und Telekommunikationsbranche Bitkom belaufen sich die Schäden deutscher Unternehmen im Jahr 2025 auf rund 289,2 Milliarden Euro, ein Anstieg um 8,5 Prozentpunkte gegenüber dem Vorjahr. Dabei haben besonders die gezielten Angriffe durch ausländische Nachrichtendienste, hauptsächlich aus Russland und China, verstärkt zugenommen [2].

Eine besondere Bedrohung stellen Advanced Persistent Threats (APTs) dar. Dabei handelt es sich um gut organisierte, oft staatlich finanzierte Hackergruppen, die gezielt Unternehmen und kritische Infrastruktur angreifen. APTs zeichnen sich dabei durch ihre langfristige und koordinierte Vorgehensweise aus, wodurch sie schwer zu erkennen sind. Typischerweise verlaufen APT-Angriffe in mehreren Phasen, von der Infiltration über Lateral Movement innerhalb des Netzwerks bis hin zur Datenextraktion oder Sabotage. In jeder Phase setzen Angreifer unterschiedliche Techniken ein. Je nach Technik hinterlassen sie dabei charakteristische Indizien in unterschiedlichen Datenquellen wie Netzwerkkommunikation oder Systemlogs kompromittierter Systemen [3].

Vor diesem Hintergrund widmet sich die vorliegende Arbeit der Frage, ob moderne Methoden der multimodalen Datenanalyse genutzt werden können, um APT-Angriffe besser zu erkennen. Die Motivation für diese Untersuchung wird im Folgenden Abschnitt näher erläutert.

1.1. Hintergrund und Motivation

Die Erkennung und Abwehr von APT-Angriffen stellt Sicherheitsforscher vor besondere Herausforderungen. Während breit gestreute Angriffe oft durch klassische Schutzmechanismen erkannt werden können, zeichnen sich APTs durch ihre Zielgerichtetheit, Langfristigkeit und Anpassungsfähigkeit aus. In jeder Angriffsphase erzeugen die Angreifer unterschiedliche Indizien in verschiedenen Datenquellen [3]. Beispielsweise taucht eine untypische Verbindung von außerhalb in den Netzwerkdaten auf oder eine sonst seltene Fehlermeldung wird vom System erzeugt. Solche einzelnen Hinweise besitzen jedoch nur eine begrenzte Aussagekraft, wenn sie isoliert betrachtet werden.

Bisherige Arbeiten versuchen APTs meist durch die Analyse einzelner Datenquellen oder durch die Kombination mehrstufiger Erkennungsmechanismen zu identifizieren [4][5][6]. Damit konnten bereits vielversprechende Ergebnisse erzielt werden. Allerdings wurde selten untersucht, ob durch die kombinierte Nutzung von Daten unterschiedlicher Modalitäten und die Berücksichtigung möglicher Interaktionen in den Daten die Erkennung weiter verbessern könnte.

Neue Ansätze im Bereich des Natural Language Processing ermöglichen es, mit Transformer-basierten Modellen den semantischen Inhalt von Texten in Vektordarstellungen abzubilden. Untersuchungen zu Anomalieerkennungsverfahren auf Basis von Log-Nachrichten, nutzen diese Vektordarstellungen bereits, um den Log-Inhalt als Eingabe für verschiedene Machine- und Deep-Learning-Modelle aufzubereiten [7][8].

Multimodal Fusion, bei der Informationen aus unterschiedlichen Datenquellen in einem Modell zusammengeführt werden, wird beispielsweise in der medizinischen Bildverarbeitung bereits erfolgreich eingesetzt, um CT-Bilder mit klinischen Patienteninformationen zu kombinieren und dadurch präzisere Diagnosen zu ermöglichen [9]. Übertragen auf die Anomalieerkennung in Logs eröffnet dies das Potenzial, Textinformationen mit weiteren Systemdaten wie Zeitreihen- oder Netzwerkmetriken zu verknüpfen und so komplexe Zusammenhänge zwischen Systemzustand und Ereignissen besser zu erfassen.

1.2. Zielsetzung

Motiviert durch diese Entwicklungen ist es Ziel dieser Arbeit zu untersuchen, durch die Kombination von multimodaler Datenfusion, maschinellem Lernen und semantischen Embeddings eine verbesserte Erkennung von APT-Angriffen erreicht werden kann. Aufbauend auf aktuellen Entwicklungen in der Anomalieerkennung auf Basis von Log-Nachrichten sowie auf Fortschritten der multimodalen Datenanalyse soll geprüft werden, inwiefern die Integration verschiedener Datenquellen zur Optimierung der Früherkennung von Angriffen beiträgt.

Somit ergeben sich für diese Arbeit die folgenden Forschungsfragen:

- Können moderne Text-Encoder-Modelle die Semantik von Log-Nachrichten erfassen, und wie wirkt sich diese semantische Repräsentation auf die Erkennung von Anomalien in Form von APT-Angriffen aus?
- Welchen Einfluss hat die multimodale Fusion von Netzwerk- und Log-Nachrichten auf die Erkennungsleistung von APT-Angriffen und der unterschiedlichen Angriffsphasen?

Um diese Forschungsfragen zu beantworten, wird zunächst ein geeigneter Datensatz ausgewählt, welcher sowohl Netzwerkdaten als auch Log-Nachrichten enthält und sich für die multimodale Analyse eignet. Anschließend werden diese Daten aufbereitet und in klassische Bag-Of-Word Vektoren, semantische Text-Embeddings und Statistik-Vektoren transformiert. Darauf aufbauend werden verschiedene Fusionsstrategien untersucht, um die Interaktionen zwischen den Elementen der Modalitäten zu erfassen. Der genaue Aufbau der Arbeit wird im Folgenden erläutert.

1.3. Aufbau der Arbeit

Die vorliegende Abschlussarbeit ist in sieben Kapitel gegliedert. Im nachfolgenden zweiten Kapitel werden die theoretischen Grundlagen der Arbeit erläutert. Zunächst wird das Konzept der APT noch einmal genau vorgestellt. Dabei werden die typischen Merkmale und die gängigen Modelle

zur Einordnung der APT-Phasen beschrieben. Anschließend werden klassische und moderne Methoden zur Darstellung von Text in numerischen Vektoren vorgestellt, dazu zählen Tokenisierung, Bag-of-Words, TF-IDF und semantische Embeddings mit Transformer-Modellen. Im weiteren Verlauf wird die Synthetic Minority Over-sampling Technique (SMOTE) als Methode zum Umgang mit unausgeglichenen Datensätzen beschrieben. Zudem werden die Grundlagen und Ansätze der Multimodalfusion erläutert, welche für die Kombination unterschiedlicher Datenquellen im Rahmen dieser Arbeit relevant sind.

Anschließend wird der aktuelle Forschungsstand im Bereich der Intrusion Detection Systeme behandelt. Dabei liegt der Fokus auf bestehenden Ansätzen zur Anomalieerkennung sowohl auf Netzwerk- als auch auf Host-Log-Ebene. Darauf aufbauend werden die ersten multimodalen Ansätze betrachtet und die gefundene Forschungslücke erläutert.

Auf Basis der Forschungslücke wird im vierten Kapitel das zu untersuchende Klassifikationsproblem formal definiert und die unimodalen sowie multimodalen Modelle beschrieben, welche zur Lösung des Problems untersucht werden. Ebenfalls wird erläutert, wie mittels Cross-Validation und Random-Search innerhalb der Trainingspipeline geeignete Parameter für die Modelle gewählt werden und anhand welcher Evaluationsmetriken die Modelle bewertet werden.

Das fünfte Kapitel beschreibt die methodische Vorgehensweise der Arbeit zur Untersuchung der multimodalen Fusion für die APT-Angriffserkennung. Es wird erläutert, nach welchen Kriterien ein Datensatz ausgewählt wurde und wie dieser für die Versuche vorbereitet wurde. Zusätzlich wird in diesem Kapitel die Implementierung der Trainingspipeline beschrieben und wie SMOTE als Oversampling Methode eingesetzt wird, um die Trainingsdaten zu erweitern.

Im sechsten Kapitel werden die erzielten Ergebnisse präsentiert und ausgewertet. Dabei werden zunächst die Resultate der unimodalen und multimodalen Klassifikationsansätze dargestellt und anschließend miteinander verglichen, wobei unter anderem auch betrachtet wird, welchen Effekt der Einsatz von Embeddings auf die uni- und multimodale Klassifikation hat. Ergänzend werden die Ergebnisse nach Anwendung der SMOTE-Methode analysiert, um den Effekt des Resamplings auf die Modellleistung zu bewerten.

Das siebte Kapitel schließt die Arbeit mit einer zusammenfassenden Diskussion ab. Die wichtigsten Ergebnisse werden im Kontext der Forschungsfragen interpretiert und kritisch reflektiert. Abschließend werden die Limitationen der Arbeit aufgezeigt und ein Fazit gezogen, das einen Ausblick auf mögliche zukünftige Forschungsarbeiten bietet.

2. Grundlagen

Im folgenden Kapitel werden die für das Verständnis der Arbeit relevanten, theoretischen Grundlagen und Methoden vorgestellt. Zunächst werden Advanced Persistent Threats genauer betrachtet, und die Besonderheiten sowie Abläufe moderner Cyberangriffe beschrieben. Anschließend werden gängige Techniken der natürlichen Sprachverarbeitung behandelt, insbesondere Methoden zur Textrepräsentation wie Tokenisierung, Bag-of-Words sowie moderne kontextualisierte Embeddings. Danach wird auf die SMOTE als Resampling-Verfahren zur Erzeugung künstlicher Datenpunkte eingegangen. Abschließend wird das Konzept des Multimodal Learning und die Methoden zur Multimodal Fusion vorgestellt, welches es ermöglichen Informationen aus unterschiedlichen Datenmodalitäten gemeinsam zu verarbeiten und zu interpretieren.

2.1. Advanced Persistent Threats

Als Advanced Persistent Threats bezeichnet man Organisationen oder staatlich finanzierte Gruppen, die komplexe und zielgerichtete Hackerattacken durchführen. Die von APTs durchgeführten Angriffe unterscheiden sich von anderen Angriffsmustern mitunter durch ihre lang andauernde Persistenz und dem Einsatz neuer oder speziell für das Ziel angepasster Methoden. Zudem nutzen APTs häufig Verschleierungstechniken, um ihre Aktivitäten zu verbergen und einer Entdeckung zu entgehen [10].

Beispiele sind die Lazarus Group, eine staatliche Hackergruppe aus Nordkorea, sowie Cozy Bear (APT29) aus Russland [11][12]. Beide Gruppen haben in den letzten Jahren schwerwiegende Angriffe auf Supply-Chains und kritische Infrastrukturen durchgeführt. Dabei haben sie Schäden in Milliardenhöhe angerichtet. Beispielsweise werden alleine die Verluste der durch den Angriff auf SolarWinds im März 2020 betroffenen Firmen auf insgesamt 90 Millionen US-Dollar geschätzt [13].

Es gibt verschiedene Modelle, anhand welcher Forscher systematische Abläufe von APT-Angriffen als Taxonomie beschreiben. Eines der bekannteren Modelle ist das Intrusion Kill Chain Modell, welches die Angriffe als sieben aufeinanderfolgende Phasen beschreibt: Reconnaissance, Weaponization, Delivery, Exploitation, Installation, Command and Control und Actions on Objectives [14].

Das MITRE ATT&CK Framework geht darüber hinaus und fungiert als umfassende Wissensdatenbank über Angriffstechniken und Taktiken. Als Teil des Frameworks fasst die ATT&CK Matrix for Enterprise systematisch gesammelte und somit bekannte Techniken als die folgenden Taktiken zusammen [15]:

- Reconnaissance: Sammeln von Informationen über Zielpersonen oder Systeme, um Schwachstellen zu identifizieren.
- Resource Development: Aufbau von Infrastruktur und Werkzeugen, welche für Angriffe benötigt werden.
- Initial Access: Erster erfolgreicher Zugang zum Zielnetzwerk durch Exploits oder kompromittierte Anmeldedaten.

- Execution: Ausführen von schädlichem Code oder Befehlen auf einem kompromittiertem System.
- Persistence: Maßnahmen, welche es erlauben, auch nach Gegenmaßnahmen weiterhin Zugang zu behalten.
- Privilege Escalation: Erlangen höherer Berechtigungen, um eingeschränkte Zugänge zu erweitern.
- Defense Evasion: Methoden, um Erkennung oder Analyse durch Verteidigungsmechanismen zu umgehen.
- Credential Access: Zugriff auf Anmeldedaten, um Zugang auszuweiten.
- Discovery: Erkundung der internen Netzwerkumgebung um weitere Ziele zu finden.
- Lateral Movement: Internes Vordringen innerhalb eines Netzwerks, über zusätzliche Systeme.
- Collection: Zusammenstellen und Aufbereiten relevanter Daten für Exfiltration.
- Command and Control: Aufbau und Kommunikation über Kanal zur Fernsteuerung kompromittierter Systeme.
- Exfiltration: Heraustransportieren gestohlener Daten aus dem Zielnetzwerk.
- Impact: Aktionen, die Verfügbarkeit, Integrität oder Vertraulichkeit von Systemen beeinträchtigen.

Im weiteren Verlauf dieser Arbeit wird auf die im ATT&CK Framework definierten Taktiken Bezug genommen, da diese von vielen anderen wissenschaftlichen Arbeiten und Datensätzen als Grundlage für die Analyse von APTs genutzt wird [16][17][18].

2.2. Logrepräsentation für maschinelles Lernen

Um Logs mithilfe von Machine-Learning-Methoden analysiert zu analysieren, muss dieser zunächst in eine numerische Repräsentation umgewandelt werden, da die meisten Modelle für Klassifikation, Clustering oder Regression nur numerische Daten verarbeiten können. Eine geeignete Textrepräsentation ist daher wichtig, um semantische und syntaktische Informationen in eine mathematisch interpretierbare Struktur zu überführen. Die Qualität dieser Repräsentation beeinflusst maßgeblich die Leistungsfähigkeit nachgelagerter Modelle [19].

Häufig liegen Log-Nachrichten jedoch zunächst in un- oder semistrukturierter Form als Zeichenketten vor. In andren Arbeiten werden Logs ähnlich wie Texte betrachtet, da sie aus einer Kombination von Wörtern und Symbolen bestehen. Folglich können Methoden, die sich für das Natural Language Processing (NLP) als effektiv erwiesen haben, auch auf die Analyse von Log-Dateien angewendet werden [20][21][22].

Im Folgenden werden die in dieser Arbeit verwendeten NLP-Methoden zur Erzeugung von Textrepräsentationen erläutert. Dazu zählt die Tokenisierung, das klassische Bag-of-Words-Modell mit TF-IDF-Skalierung zu Erstellung von spärlich besetzter Vektorrepräsentationen sowie Deep-Learning-Methoden um moderne, dichte und kontextualisierte Embeddings zu erzeugen.

2.2.1. Tokenisierung

Tokenisierung ist meist einer der ersten Schritte der Textverarbeitung. Dabei wird ein Text in einzelne Elemente, sogenannte Tokens zerlegt. Diese können Wörter, Wortteile, Satz- oder Sonderzeichen sein. Die Tokenisierung legt die Grundlage für alle weiteren Repräsentationsformen, da sie bestimmt, welche Textelemente überhaupt als Features berücksichtigt werden [23]. Für diese Arbeit ist die Wort-Tokenisierung und die modernere WordPiece-Tokenisierung besonders relevant.

Wort-Tokenisierung

Bei der einfachen Wort-Tokenisierung wird Text anhand von Worten aufgeteilt. Dabei trennen Leer- oder Satzzeichen Worte voneinander und jedes Wort wird als eigenständiges Token betrachtet [23]. Die Wort-Tokenisierung ist einfach und effizient, führt jedoch zu Problemen bei komplexen zusammengesetzten Wörtern oder Texten aus technischen Domänen. Beispielsweise treten in Log-Nachrichten häufig zusammengesetzte oder kombinierte Wörter auf, wie beispielsweise Variablen oder Funktionsnamen in Fehlermeldungen. Eine rein wortbasierte Tokenisierung behandelt solche Begriffe als separate Token was das Token-Vokabular stark wachsen lässt.

Darüber hinaus erschwert die Wort-Tokenisierung den Umgang mit unbekannten Wörtern, wodurch das Out-of-Vocabulary-Problem entsteht. Wörter die nicht im Trainingsvokabular enthalten sind, also dem Modell unbekannte Tokens, können nicht interpretiert werden [24]. Dies kann insbesondere bei der Log-Analyse problematisch sein, da dort häufig neue, dynamisch generierte Wörter wie IDs, Hashes oder Pfadangaben vorkommen.

Subwort-Tokenisierung und BBPE Encoding

Um beide Probleme zu umgehen, wird für moderne Natural-Language-Processing-Modelle Subwort-Tokenisierung eingesetzt. Dabei werden lange oder komplexe Wörter in kleinere Einheiten zerlegt, die anschließend als einzelne Tokens verarbeitet werden. Häufig wird dafür WordPiece [25] oder Byte Pair Encoding (BPE) [26] eingesetzt. Das in dieser Arbeit verwendete Modell nutzt byte-level Byte-Pair-Encoding (BBPE) als eine Variante des BPE [27].

Bei BBPE wird der Text als UTF-8-Bytefolge kodiert, und das Vokabular besteht zu Beginn aus allen 256 möglichen Byte-Werten sowie den notwendigen Spezialtokens. Anfangs gelten alle einzelnen Bytes als elementare Tokens. Anschließend wird, wie bei BPE, wiederholt das häufigste Byte-Paar zu einem neuen Token zusammengeführt, bis die maximale Vokabulargröße erreicht ist. Damit lassen sich von den Modellen alle Texte verarbeiten, die in UTF-8 darstellbar sind [28].

Aufbauend auf den beschriebenen Tokenisierungsmethoden werden im Folgenden Bag-of-Words (BoW) sowie transformerbasierte Verfahren vorgestellt, mit denen numerische Repräsentation der extrahierten Tokensequenzen erzeugt werden. Dies ermöglicht deren Weiterverarbeitung mit nachgelagerten Machine-Learning-Modellen.

2.2.2. Vector Space Model

Das Vector Space Model (VSM) [29] bildet die Grundlage für die nachfolgenden Methoden zur Repräsentation von Texten. Die Idee besteht in der Darstellung von Dokumenten und Queries als Vektoren in einem mehrdimensionalen kontinuierlichen Raum.

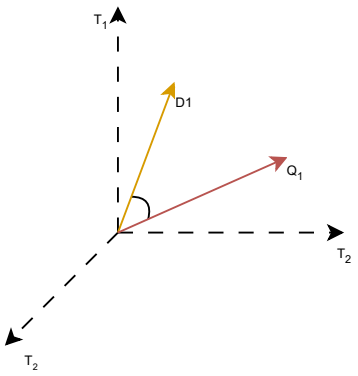

Abbildung 2.1.: Visualisierung von Dokumenten und Queries im VSM [29]

Zentral für das VSM ist die Möglichkeit durch die Kosinus-Ähnlichkeit, die Ähnlichkeiten zwischen Dokumenten und Queries zu messen(siehe Abbildung 2.1).

Im klassischen VSM werden Dokumente abgebildet, sodass diejenigen, die viele gemeinsame oder ähnliche Wörter enthalten, in der Darstellung nah beieinander erscheinen. Moderne Erweiterungen, wie Word Embeddings oder semantische Vektorraum Modelle, nutzen zusätzlich semantische Informationen, um Ähnlichkeiten besser abzubilden.

2.2.3. Bag-of-Words und TF-IDF

Das BoW-Modell ist ein einfaches und klassisches Verfahren zur numerischen Repräsentation von Texten in einem VSM. Dabei wird ein Dokument als Menge von Token gezählt, ohne die Reihenfolge oder syntaktische Struktur zu berücksichtigen. Jedes Dokument wird durch einen Vektor der Länge des gesamten Vokabulars dargestellt, wobei jeder Eintrag die Häufigkeit eines bestimmten Tokens angibt [30].

Durch die Abbildung von Dokumenten als BoW-Vektoren gehen jedoch sämtliche Informationen über die Wortreihenfolge verloren. Ein weiterer Nachteil der BoW-Repräsentation besteht darin, dass alle Wörter als gleich wichtig betrachtet werden, wodurch häufig vorkommende Wörter das Ergebnis dominieren, obwohl sie für die semantische Trennung von Dokumenten wenig informativ sind [31]. Um dieses Problem zu vermeiden, wird oft die TF-IDF Gewichtung eingesetzt. Dabei wird jedes Token nicht nur nach seiner Häufigkeit im Dokument gewichtet, sondern auch nach seiner Seltenheit im Korpus durch [30]:

$$\text{tfidf}_{d,t} = \text{tf}_{d,t} \cdot \log \frac{N}{\text{df}_t} \tag{2.1}$$

Hierbei bezeichnet $\text{tf}_{d,t}$ die Häufigkeit des Terms t im Dokument d, df_t die Anzahl der Dokumente, in denen t vorkommt, und N die Gesamtzahl der Dokumente im Korpus. Diese Skalierung sorgt

dafür, dass seltene, aber informative Wörter stärker hervorgehoben werden, während häufige, wenig informative Wörter abgeschwächt werden.

Auch mit Skalierung behandelt Bag-of-Words Tokens als orthogonale Einheiten, was zu spärlich besetzten Vektoren führt, die weder die semantische Nähe von Wörtern noch deren Kontext im Satz berücksichtigen und abbilden. Im Gegensatz dazu übertragen Embedding-Verfahren Textdaten in dichte, semantisch strukturierte Vektorräume [30].

2.2.4. Embeddings

Embeddings sind dichte Vektorrepräsentationen von Wörtern, Sätzen oder Dokumenten in einem hochdimensionalen Raum. Im Gegensatz zu Bag-of-Words oder TF-IDF erfassen sie nicht nur die Häufigkeit von Tokens, sondern auch deren Bedeutung sowie die Beziehungen zwischen ihnen. Wörter oder Texte, die inhaltlich ähnlich sind, liegen im Embedding-Raum nah beieinander [32].

Die Grundlage von Embeddings bildet die Verteilungshypothese, die besagt, dass Wörter, die in ähnlichen Kontexten auftreten, auch ähnliche Bedeutungen haben. Mittels Deep-Learning-Modellen werden diese Beziehungen durch Optimierung einer Zielfunktion erlernt. Die Zielfunktion sorgt dafür, dass häufig zusammen auftretende, also semantisch verwandte, Tokens im Raum zusammengeführt werden, während unähnliche Tokens weiter voneinander entfernt liegen [31].

Dabei unterscheidet man zwischen statischen Wort-Embeddings und kontextualisierten Embeddings. Statische Wort-Embedding-Methoden wie Word2Vec [33] oder GloVe [34] ordnen jedem Wort einen festen Vektor zu, der während des Trainings aus großen Textkorpora gelernt wird.

Modernere Transformer-Modelle wie ELMo [35] oder BERT [36] berücksichtigen zusätzlich den Wortkontext, indem sie durch Multi-Head-Attention die umliegenden Tokens betrachten und so kontextualisierte Embeddings erzeugen. Dadurch wird dasselbe Wort in unterschiedlichen Kontexten als unterschiedlicher Vektor abgebildet, was Mehrdeutigkeiten berücksichtigt und die semantische Genauigkeit erhöht.

Um mittels moderner, vortrainierter Sprachmodelle Satz- oder Dokumenten-Embeddings zu erstellen, werden die kontextualisierten Token-Vektoren zu einem kombinierten Vektor aggregiert. Dies geschieht häufig durch Mittelung, gewichtete Summierung oder durch ein spezielles [CLS]-Token. Das [CLS]-Token Dieses Token wird am Beginn der Tokensequenz eingefügt, und seine Vektorrepräsentation enthält die zusammengefassten Kontextinformationen der Eingabesequenz [31].

Embeddings bieten den Vorteil, dass sie semantische Zusammenhänge direkt in den Vektoren repräsentieren. Dadurch können Machine-Learning-Modelle Texte besser vergleichen und analysieren, was insbesondere bei Aufgaben wie Klassifikation, Clustering oder semantischer Suche zu verbesserten Ergebnissen führt. Sie bilden damit die Grundlage vieler moderner Verfahren der Sprachverarbeitung.

2.3. Synthetic Minority Over-sampling Technique

Machine-Learning-Modelle, welche auf unausgeglichen Datensätzen trainiert werden, also auf Datensätzen mit stark ungleich verteilten Klassen, stoßen häufig auf Probleme beim Lernen von guten Entscheidungsgrenzen [37]. Bei Intrusion-Detection-Datensätzen, die aus einer Mischung von normalen sowie anomalen Daten bestehen, ist mit einer solchen Imbalance zu rechnen, da normales Verhalten in realistischen Umgebungen überproportional häufiger vorkommt und sich somit leichter erfassen lässt als Angreiferverhalten.

Eine Möglichkeit, diesem Problem beim Training von Deep-Learning-Modellen entgegenzuwirken, besteht darin, die Kostenfunktion durch Klassengewichtung anzupassen. Dabei werden Fehlklassifikationen der Minderheitenklasse stärker gewichtet. Dies führt dazu, dass die Gradienten für falsch klassifizierte Beispiele der Minderheitenklasse größer werden, wodurch das Modell während des Trainings verstärkt auf diese Klasse angepasst wird.

Eine weitere Herangehensweise sind Resampling-Verfahren wie Over- und Undersampling. Beim Undersampling werden Datenpunkte der Mehrheitsklasse verworfen und somit zufällig aus den Trainingsdaten entfernt, was jedoch zu einem Informationsverlust führen kann. Beim Oversampling wird die Minderheitsklasse künstlich vergrößert, entweder durch das Duplizieren vorhandener Datenpunkte oder durch die Generierung neuer, synthetischer Daten [38].

Einfaches Duplizieren von Datenpunkten der Minderheitsklassen kann zu Overfitting führen. Mit SMOTE [38] werden hingegen neue synthetische Samples für die Minderheitsklasse erzeugt, indem zwischen existierenden Datenpunkten der Minderheitsklasse Interpoliert wird. Dazu wählt man für jeden Datenpunkt i einen zufälligen Referenzpunkt i_n aus k Nachbarn derselben Klasse aus.

Entlang der Verbindung zwischen i und i_n wird anschließend zufällig ein neuer Punkt generiert. Dieser Vorgang wird so oft wiederholt, bis genügend synthetische Punkte erzeugt wurden. Durch die neuen Punkte werden die Modelle gezwungen, größere und somit generelle Entscheidungsregionen für die Minderheitsklassen zu lernen [38]. Algorithmus 1 zeigt den Ausschnitt der SMOTE-Populationsroutine.

Algorithm 1 SMOTE-Populate($N, i,$ nnarray) [38]

```
1: i = minority class sample
2: nnarray = k nearest neighbors of i
3: N = Number of SMOTE samples
4: while N ≠ 0 do
5:     Choose a random number between 1 and k, call it nn
6:     for attr ← 1 to numattrs do
7:         dif = Sample[nnarray[nn]][attr] − Sample[i][attr]
8:         gap = random number between 0 and 1
9:         Synthetic[newindex][attr] = Sample[i][attr] + gap ∗ dif
10:    end for
11:    newindex++
12:    N = N − 1
13: end while
14: return
```

2.4. Multimodal Learning

Wie eingangs beschrieben, versteht man unter Multimodal Learning die Teildisziplin des maschinellen Lernens, welche sich mit der Integration von Daten aus unterschiedlichen Modalitäten befasst. In der vorliegenden Untersuchung stehen insbesondere Textdaten und Netzwerk-Flow-Features im Zentrum. Ziel ist es, die komplementären Informationen dieser Modalitäten zu nutzen, um Angriffe innerhalb eines Netzwerks zu erkennen.

Baltrusaitis et al. [39] identifizieren in ihrer Arbeit Representation, Translation, Alignment, Fusion und Co-Learning als die fünf zentralen Herausforderungen des Multimodal Learning. Für diese Arbeit sind insbesondere Representation und Fusion von Relevanz, da sie bestimmen, wie heterogene Modalitäten in einen Repräsentationsraum überführt und dort kombiniert werden können.

Repräsentation bezeichnet das Problem der Abbildung von Elementen in Form von Feature-Vektoren. Die Herausforderung in Bezug auf Multimodal Learning besteht darin, Feature-Darstellungen zu finden, welche trotz Heterogenität modalitätsübergreifend die vorhandenen Relationen zwischen den Elementen, wie komplementäre oder redundante Informationen, erhalten [39].

In Abschnitt 2.2.3 und 2.2.4 wurden bereits Verfahren zur Extraktion und Konvertierung von Text in Bag-Of-Word-Vektoren und Embeddings vorgestellt. Die Netzwerk-Flows hingegen werden im Laufe der Arbeit durch Merkmale wie Paketanzahl, Flussdauer oder durchschnittliche Paketgröße charakterisiert. Somit liegen lediglich getrennte Repräsentationen der einzelnen Modalitäten vor, heißt Text- und Netzwerkdaten werden unabhängig voneinander abgebildet und zunächst nicht in einem gemeinsamen Merkmalsraum integriert.

Multimodal Fusion

Da die beiden Modalitäten isoliert repräsentiert werden, müssen sie in der Learning-Pipeline miteinander kombiniert werden. Dieser Schritt wird als Multimodal Fusion bezeichnet. Sie beschreibt den Prozess, wann und wie aus verschiedenen Modalitäten gewonnene Repräsentationen miteinander kombiniert werden, um komplementäre Informationen zu integrieren.

In der Literatur werden, wie in Abbilding 2.2 dargestellt, multimodale Modelle und die darin angewandten Fusionstechniken häufig anhand der Ebene, auf welcher die Fusion durchgeführt wird, eingeordnet in [40]:

- Input Level (Early Fusion): Die Daten werden vor oder direkt nach der Feature-Extraktion, jedoch vor dem Lernalgorithmus zusammengeführt.
- Feature Level (Intermediate Fusion): Die Modalitäten werden auf den Zwischenebenen des Modells als latente Repräsentation kombiniert.
- Decision Level (Late Fusion): Jede Modalität wird zunächst von separaten Modellen verarbeitet, die Modellentscheidungen werden anschließend kombiniert.

Die eingesetzte Fusionsebene hat Einfluss auf die Komplexität der Feature-Beziehungen, welche das jeweilige Modell abbilden kann. Mit Early Fusion können dabei sehr komplexe Interaktionen zwischen den Modalitäten erfasst werden, da den Modellen nahezu unverarbeitete Daten zur Verfügung stehen. Late Fusion kann hingegen nur Entscheidungsbeziehungen ohne direkte Feature-Interaktion abbilden, während Intermediate Fusion Beziehungen zwischen bereits extrahierten latenten Feature-Repräsentationen erfassen kann [40].

Die Heterogenität der Modalitäten bestimmt, auf welcher Ebene eine Fusion der Daten möglich ist. Modalitäten mit homogenen Elementen lassen sich bereits vor dem Modell, beispielsweise als CNN-Channels zusammenführen. Elemente sehr unterschiedlicher Modalitäten müssen zunächst in eine gemeinsame, repräsentative Form überführt werden. Ein Beispiel hierfür ist CLIP [41], welches Bilder- und Texte in einen gemeinsamen semantischen Raum projiziert, um ihre Relationen abzubilden.

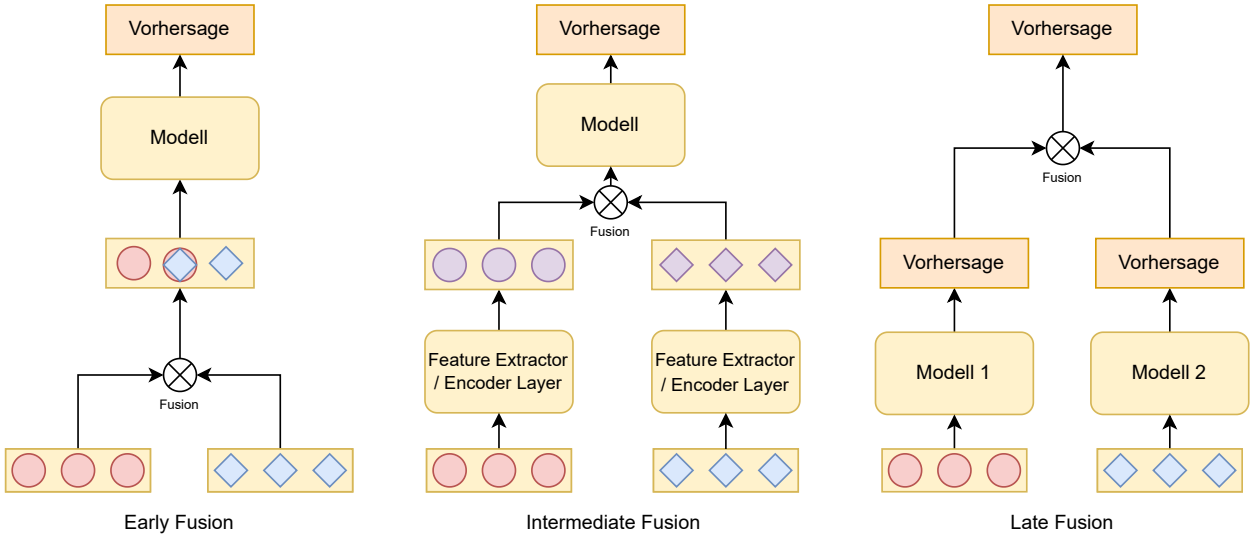

Abbildung 2.2.: Übersicht der Fusionsebenen in multimodalen Modellen [40]

Intermidate-Fusionsmethoden

Verschiedene Verfahren zur Merkmalsfusion auf Feature-Ebene wurden bereits untersucht. Beispielsweise identifizierten Cui et al. [42] in ihrer Arbeit zur Deep Multimodal Fusion von Bild- und Nicht-Bild-Daten Operator-, Attention-, Tensor, Subspace- sowie Graphbasierte Fusionsmethoden.

Einfache Operations-Fusion ist die Fusion durch elementweise Addition, Mittelwertbildung, Multiplikation oder Vektorverkettung. Addition, Mittelwertbildung und Multiplikation setzen voraus, dass die zu fusionierenden Repräsentationen dieselbe Dimensionsgröße und semantische Bedeutung haben [42].

Die Verkettung kombiniert Repräsentationen zu einem erweiterten Vektor. Dabei bleiben sämtliche Informationen erhalten. Interaktionen zwischen den Dimensionen werden jedoch nicht explizit

erfasst. Häufig wird die Verkettung daher mit einem nichtlinearen Modell kombiniert, das die Beziehungen zwischen den Merkmalen erlernen kann. Gleichzeitig steigt jedoch die Dimensionalität der resultierenden gemeinsamen Repräsentation [42].

Bei der Tensor-Fusion wird das äußere Produkt der Feature-Vektoren gebildet. Auf diese Weise entsteht ein Tensor, der einer Co-Occurrence-Matrix entspricht und somit Interaktionen höherer Ordnung erfasst [42]. Interessanterweise konnten Zadeh et al. in ihren Untersuchungen zeigen, dass trotz der hohen Dimensionalität des resultierenden Tensors kein Overfitting auftritt [43].

Bei der Subspace-Fusion werden die Modalitäten durch Encoder oder durch Projektionen in einen gemeinsamen latenten Raum abgebildet, in dem semantisch ähnliche Repräsentationen nah beieinander und unähnliche weiter voneinander entfernt liegen [42].

Attention Fusion nutzt den Attention-Mechanismus, um dynamisch und kontextsensitiv zu bestimmen, welche Features oder Datenquellen für die aktuelle Aufgabe besonders relevant sind. Durch die gewichtete Hervorhebung wichtiger Informationsträger können irrelevante oder redundante Merkmale abgeschwächt werden [44].

Graph Fusion modelliert multimodale Daten als Knoten in einem Graphen und ihre Wechselwirkungen als Kanten. Graph Neural Networks iterieren über die Knoten, um Informationen von Nachbarn mittels Aggregationsfunktionen zu propagieren [42].

Late-Fusionsmethoden

Bei der Late Fusion, auch Modell Ensamble genannt, wird für jede Modalität ein eigenes Modelle trainiert. Bei der Inferenz werden die Vorhersagen der Modelle bestimmt und kombiniert. Die Fusion erfolgt somit erst nach der unimodalen Verarbeitung [40].

Gängige Strategien für die Kombination der Entscheidungen sind Hard und Soft Majority Voting. Beim Hard Majority Voting wird die Klasse gewählt, die von der Mehrheit der unimodalen Modelle vorhergesagt wird. Demgegenüber summiert das Soft Majority Voting die vorhergesagten Wahrscheinlichkeiten der Modelle und wählt die Klasse mit der höchsten aggregierten Wahrscheinlichkeit [45].

Darüber hinaus lassen sich Gewichtungen einführen, sodass zuverlässigeren oder leistungsstärkeren Modalitäten ein höherer Einfluss auf die finale Entscheidung zukommt. Die Gewichtung folgt entweder anhand vorab definierter Werte oder lernbasiert durch Optimierung im Rahmen eines Metamodells [45].

3. Verwandte Arbeiten

In diesem Kapitel wird ein Überblick über den aktuellen Forschungsstand im Bereich der Intrusion Detection gegeben. Zunächst werden die auf Netzwerk- und Hostlog-Ebene angewandten Methoden zur getrennten Intrusion Detection betrachtet, um einen Überblick zu bekommen, welche Ansätze und Techniken bereits untersucht wurden. Anschließend werden bereits existierende multimodale Intrusion Detection Ansätze vorgestellt. Abschließend wird die bestehende Forschung kritisch analysiert, um die erkannten Lücken aufzuzeigen, die den Ausgangspunkt für die vorliegende Arbeit bilden.

3.1. Methoden der Intrusion Detection

Im Grunde lässt sich die Erkennung von APT Angriffen dem Forschungsgebiet der Intrusion Detection Systeme (IDS) oder Intrusion Prevention Systeme (IPS) zuordnen. Diese leisten mehr als nur die reine Angriffserkennung, da sie beispielsweise verteilt Informationen sammeln, Wissensdatenbanken aufbauen und an Echtzeitanforderungen gebunden sind [46]. Diese zusätzlichen Bestandteile und Rahmenbedingungen sind für die vorliegende Arbeit zwar nicht relevant, dennoch sollen die zugrunde liegenden Methoden sowie Techniken betrachtet werden, um zu verstehen, welche Algorithmen für die Angriffserkennung bereits untersucht wurden.

IDS arbeiten meist entweder mit signaturbasierter oder anomaliebasierter Erkennung. Signatur- oder auch patternbasierte Verfahren nutzen zur Identifizierung Signaturen und Muster von bereits bekannten Angriffen, wie etwa Dateihashes, bekannte bösartige IP-Adressen oder Nutzerverhalten [47]. Diese Methoden setzen voraus, dass die bei Angriffen verwendeten Techniken bereits zuvor erkannt und als Signaturen erfasst wurden. Da APTs häufig neuartige Angriffstechniken und Zero Day Exploits einsetzen, stoßen signaturbasierte Erkennungsverfahren insbesondere bei der Identifikation von APT-Angriffen an ihre Grenzen.

Anomaliebasierte Erkennungsverfahren hingegen zielen darauf ab, ungewöhnliches, von normalem abweichenden Verhalten im von ihnen überwachten System zu identifizieren, häufig indem eine Baseline erstellt und Abweichungen von dieser erkannt werden.

Hostlogbasierte Anomalieerkennung

Moderne hostlogbasierte IDS basieren überwiegend auf Anomalieerkennung. Dabei werden abnormale und auffällige Muster in der Abfolge von Log-Sequenzen identifiziert, um potenzielle Angriffe frühzeitig zu erkennen. Dazu wurden bereits verschiedene Ansätze untersucht, sowohl klassische Machine-Learning-Methoden als auch moderne Supervised-, Semi-Supervised- und Unsupervised-Learning-Methoden [48].

Ein Beispiel für einen Unsupervised-Learning-Ansatz ist das von Du et al. vorgestellte DeepLog-Modell [49]. Dabei werden Log-Nachrichten als Sequenzen von Ereignissen betrachtet. Da einzelne

Log-Nachrichten häufig viele variable Informationen enthalten, werden sie zunächst in eine abstrakte Template-Repräsentation überführt (siehe auch Abschnitt 5.2.3). Auf diese Weise entsteht aus einer Vielzahl unterschiedlicher Log-Nachrichten eine strukturierte Folge von Template-Indizes.

Anschließend lernt ein Long Short-Term Memory-Modell (LSTM), welche Template-Reihenfolgen unter normalen Bedingungen typischerweise auftreten. Während des Systembetriebs kann das Modell auf Basis der gelernten Sequenzen vorhersagen, welches Template als Nächstes zu erwarten ist. Die Vorhersage ist eine Wahrscheinlichkeitsverteilung über mögliche nächste Templates. Erscheint ein Template, das nicht zu den Top-K vorhergesagten Möglichkeiten gehört, wird es als Anomalie klassifiziert.

Meng et al. erweitern dieses Konzept durch den Einsatz von semantischen Embeddings in Log-Sequenzen mit ihrem template2vec-Ansatz und LogAnomaly [50]. Ähnlich wie bei DeepLog lernt ein LSTM die Vorhersage der nächsten Top-K Log-Templates in einer normalen Template-Sequenz. Allerdings werden die Log-Sequenzen hier nicht als Template-Index-Folgen, sondern als Sequenzen von Template-Vektoren dargestellt, die aus der Kombination von Word-Vektoren erstellt werden. Durch die Verwendung eines angepassten statischen Embeddingverfahrens werden Vektoren erzeugt, die die semantische Bedeutung von Log-Templates in die Anomalieerkennung einfließen lassen. Zusätzlich kommt ein zweites LSTM-Modell zum Einsatz, das das quantitative Vorkommen von Log-Templates innerhalb einer Log-Sequenz lernt. Beide Modelle werden anschließend durch einen Attention-Layer kombiniert. LogAnomaly zählt damit zu den ersten Ansätzen, die Embeddings zur Anomalieerkennung in Log-Sequenzen einsetzten.

In den letzten Jahren haben transformerbasierte Sprachmodelle wie BERT [36] und GPT [51] die Erfassung semantischer Bedeutungen durch kontextualisierte Embeddings erheblich verbessert. Wie bereits in Abschnitt 2.2 beschrieben, können diese Modelle nicht nur statische beschreibende Embeddings für Wörter erzeugen, sondern auch deren umliegenden Kontext erfassen. Der Supervised-Learning-Ansatzt LogLLM von Guan et al. [7] nutzt diesen Fortschritt, um mithilfe von BERT als Encoder-Modell aus Log-Templates semantische Embedding-Sequenzen zu erzeugen und diese, nach einer Projektion in den Llama Embedding Raum, mithilfe eines Llama-Decoder-Modell zu klassifizieren.

Ein weiterer transformerbasierter Ansatz ist LogBERT von Guo et al. [52]. LogBERT modelliert Log-Sequenzen ebenfalls auf Template-Ebene, jedoch mit einem transformerbasierten Modell. Anstatt Wörter zu tokenisieren, wird für jedes Log-Template ein Token im BERT-Vokabular erstellt. Das Modell wird mittels Masked Log Template Key Prediction trainiert, wobei zufällig ausgewählte Template-Tokens maskiert und vom Modell vorhergesagt werden. Durch diese Vorgehensweise lernt LogBERT die sequentiellen Abhängigkeiten in den Log-Sequenzen. Während der Erkennungsphase wird das Modell ähnlich wie bei DeepLog oder LogAnomaly genutzt um anhand der restlichen Tokens die Top-K nächsten Log-Template-Kandidaten vorherzusagen. Abweichungen von diesen Vorhersagen werden wieder als Anomalien klassifiziert.

Netzwerk basierte Anomalieerkennung

Netzwerk basierte Intrusion Detection ist ebenfalls ein breitgefächertes Forschungsgebiet, wobei bösartige Netzwerkaktivität auf Basis von Paketen, Flow- oder Statistiken bis hin zu graphbasierten Ansätzen untersucht und erkannt wird. Rohe Pakete werden häufig als Bilder interpretiert und mit modernen Bildverarbeitungsmethoden analysiert. Netzwerkpakete sind lediglich Bytefolgen, die im Kontext ihrer Protokolle Bedeutung haben, daher ist es sinnvoll, Expertenwissen zu nutzen, um gezielt Merkmale wie Paketgrößen, Zeitabstände oder Header-Felder zu extrahieren. Da es sich bei Netzwerk-Statistiken um gut strukturierte Feature-Vektoren handelt, eignen sich zur Klassifikation alle gängigen Machine-Learning-Verfahren als auch neuronale Netze.

Watson untersucht in seiner Arbeit den Einsatz von Multi Layer Perceptrons (MLPs) und Convolutional Neural Networks (CNNs) zur Anomalieerkennung im Kontext von Netzwerk-Intrusionen anhand von Flow-Statistiken und übertragenen Payloads. Seinen Ergebnissen zufolge ermöglicht die payloadbasierte Klassifikation zwar die Erkennung von Angriffsanomalien, bietet jedoch trotz höherer Komplexität und Trainingsaufwand keinen signifikanten Mehrwert gegenüber der rein statistikbasierten Klassifikation [53].

Zur statistikbasierten Erkennung vergleichen Vinayakumar et al. [54] ein MLP mit unterschiedlichen CNN-Architekturen. Während 1-D CNNs lokale Muster erfassen, modellieren Sequenzverfahren globale Zusammenhänge. Ihre Untersuchungen zeigen jedoch, dass 1-D CNNs ausreichen und hybride Ansätze mit RNN, LSTM oder GRU für reine Statistikmerkmale keinen Vorteil bringen.

Trotzdem prüfen neuere Arbeiten weiterhin den kombinierten Einsatz von CNNs und Sequenzmodellen auf Netzwerk-Statistiken. Sinha et al. [55] untersuchen ein CNN-BiLSTM, Dai et al. [56] setzen auf ein CNN-BiLSTM-Attention-Netz, und Wang et al. [57] untersuchen den Einsatz von TabTransformer zu Klassifikation. Alle drei Arbeiten konnten gute Ergebnisse erziehlen.

3.2. Multimodale Ansätze

Neben rein log- oder netzwerkbasierten Verfahren gewinnen Ansätze an Bedeutung, die mehrere Datenquellen kombinieren, um die Erkennungsleistung von Intrusion Detection Systemen zu erhöhen. Im Kontext dieser Arbeit stehen insbesondere Untersuchungen im Vordergrund, die moderne Deep-Learning-Modelle durch Fusionsmethoden miteinander verbinden.

Kiflay et al. [58] verfolgen einen klassischen Machine-Learning-Ansatz, bei dem für jeden Netzwerkfluss sowohl Payload-Bytes als auch statistische Merkmale als Feature-Vektoren extrahiert und jeweils durch separate Random-Forest-Modelle klassifiziert werden. Die beiden Modelle werden anschließend mittels Soft-Decision-Fusion zu einer Late-Fusion-Struktur kombiniert, wodurch die Vorhersagen beider Modelle integriert werden, ohne die ursprünglichen Features zu verschmelzen.

Ihre Arbeit vergleichen sie mit der von Min et al. [59], welche Feature-Level-Fusion einsetzen, indem sie Payload-Features mittels Deep-Learning extrahieren. Hierzu werden die Payload-Bytes zunächst mithilfe der Skip-Gram-Variante des Word2Vec-Ansatzes in Byte-Embeddings überführt, welche

anschließend durch ein CNN kombiniert werden. Diese werden schließlich mit den statistischen Netzwerkmerkmalen konkateniert und gemeinsam durch einen Random Forest klassifiziert. Kiflay et al. evaluieren ihr Modell auf dem UNSW-NB15-Datensatz und zeigen, dass die Late-Fusion-Architektur bei geringerer Modellkomplexität eine verbesserte Angriffserkennung erreicht.

Im Vergleich dazu untersuchen Agrafiotis et al. [60] einen multimodalen Ansatz, bei dem mehrere Deep-Learning-Modelle durch Intermediate-Fusion kombiniert werden, um drei unterschiedliche Repräsentationen von Flows für die Klassifikation zu nutzen. Dazu extrahieren sie zunächst statistische Features mithilfe von CICFlowMeter, sie erzeugen Byte-Embeddings, indem die ersten 1024 Bytes eines Flows in Zahlenwerte von 0 bis 255 überführt werden, sowie Bilddarstellungen, in denen dieselben Bytes als 32×32-Graustufenbilder abgebildet werden. Für jede Merkmalsart wird ein separates Modell trainiert, ein MLP für die statistische Features, ein LSTM für die Byte-Embeddings und ein CNN für die Bildrepräsentationen.

Die Modelle werden zunächst einzeln mit einem Softmax-Layer für die Klassifikation trainiert. Anschließend wird dieser Layer entfernt, die Zwischenschichten der Modelle werden konkateniert und durch einen gemeinsamen Fully-Connected- sowie abschließenden Softmax-Layer weiter trainiert, bis das Gesamtsystem konvergiert. Das fusionierte Modell wird anhand von Flooding- und Brute-Force-Angriffen evaluiert und zeigt sowohl hinsichtlich Genauigkeit als auch F1-Score eine bessere Leistung als die einzelnen unimodalen Modelle. Ein Vergleich mit anderen Ansätzen wurde jedoch nicht durchgeführt.

Hwang et al. [61] verfolgen ebenfalls einen multimodalen Ansatz, der jedoch Flow-Statistiken, Systemlogs und Hoststatistiken integriert. Die Autoren modellieren Netzwerkflüsse als eindimensionale Graustufenbilder mit einer Auflösung von 180 Pixeln, die durch ein Convolutional Neural Network mit zwei Faltungsschichten, Max-Pooling und drei vollvernetzten Schichten verarbeitet werden. Systemlogs werden mithilfe des Drain-Algorithmus in Templates überführt, mit Word2Vec in Vektoren transformiert, über TF-IDF gewichtet und zu Sequenzen fester Länge zusammengefasst, die einem LSTM mit sequentieller Self-Attention zugeführt werden. Hoststatistiken, welche Ressourcenmetriken wie CPU- und RAM-Auslastung repräsentieren, werden über ein Multi-Layer-Perceptron (MLP) klassifiziert.

Die drei Modelle werden zunächst unabhängig voneinander trainiert und anschließend auf Entscheidungsebene kombiniert. Dazu nutzen sie eine Hard-Decision-Fusion, bei der ein Zeitslot als anomal gilt, sobald eines der Modelle einen Angriff erkennt. Zur Bewertung wird ein eigens generierter Datensatz verwendet, der mit einer auf früheren Arbeiten basierenden Toolchain erstellt wurde und verschiedene, nach dem MITRE ATT&CK-Framework simulierte Angriffsszenarien abbildet. Die Ergebnisse zeigen deutliche Verbesserungen der kombinierten Architektur im Vergleich zu unimodalen und klassischen Machine-Learning-Verfahren.

Liu et al. kritisieren in ihrer Arbeit, dass viele bestehende Untersuchungen Host-Events zu stark in statistische Feature-Vektoren komprimieren, um sie in eine vergleichbare Repräsentation wie Netzwerk-Feature-Vektoren zu überführen. Dieser Prozess führt jedoch häufig zu Informationsverlusten. Mit dem Framework CDFF sowie dem darauf aufbauenden CIDS-Net stellen Liu et al. [62] einen Ansatz zur gemeinsamen Erfassung und Analyse von Netzwerk-Flow-Daten und System-Logs

vor. Dabei handelt es sich um ein transformerbasiertes, multimodales Intrusion-Detection-Modell, das beide Datenquellen integriert, um Angriffe präziser zu erkennen.

CIDS-Net basiert ebenfalls auf drei Encoder-Modellen, die jeweils unterschiedliche Eingabemodalitäten verarbeiten und mittels Feature-Fusion kombiniert werden. Für die Flow-Statistik-Vektoren verwenden die Autoren ein MLP Modell und evaluieren ergänzend die Eignung von TabNet. Ein Event-Feature-Encoder modelliert Systemereignisse als multivariate Zeitreihen und nutzt transformerbasierte Sequenzschichten zur Abbildung zeitlicher Abhängigkeiten. Der Event-Message-Encoder konvertiert Log-Nachrichten über BERT in semantische Embeddings und verarbeitet diese anschließend mit weiteren Transformerschichten, um kontextuelle Beziehungen innerhalb der Log-Texte zu erfassen. Auf diese Weise kombiniert das Modell die Stärken moderner hostlogbasierter IDS-Methoden nach Guan et al. mit etablierten netzwerkbasierten Verfahren.

Die Ausgaben der drei Encoder werden anschließend konkateniert und durch ein Fully Connected Network mit abschließendem Soft-Max Layer fusioniert. In der Evaluation erzielt CIDS-Net gegenüber unimodalen Baselines eine deutlich präzisere Klassifikation, insbesondere bei selten auftretenden Angriffstypen. Die Ergebnisse verdeutlichen, dass die kombinierte Analyse von Netzwerkdaten, Systemereignissen und Log-Nachrichten die Erkennungsleistung bei komplexen Angriffen wesentlich verbessert.

3.3. Forschungslücke

Obwohl in der bisherigen Forschung viele Verfahren zur Erkennung von Angriffen auf Netzwerk- und Host-Ebene untersucht wurden, bestehen weiterhin Defizite in der kombinierten Analyse dieser Datenquellen. Unimodale Ansätze sind jeweils auf die verfügbare Informationen aus der untersuchten Modalität beschränkt. Beispielsweise können rein netzwerkbasierte Verfahren lediglich Informationen aus übertragenen Paketen oder Flows nutzen, während hostbasierte Ansätze ausschließlich auf System- und Logdaten zurückgreifen können [55][8]. Dadurch bleibt ein erheblicher Teil von Informationen unberücksichtigt.

Frühe hostlogbasierte Verfahren modellieren Logabfolgen meist auf Basis von Template-Indizes, wodurch semantische Zusammenhänge und inhaltliche Bedeutung einzelner Nachrichten weitgehend vernachlässigt werden. Auch moderne transformerbasierte Modelle wie LogBERT [52] oder LogLLM [7] fokussieren überwiegend auf syntaktische Sequenzmuster statt auf den vollständigen semantischen Inhalt der Lognachrichten. Somit bleibt die potenzielle Stärke kontextualisierter Embeddings zur Abbildung von Ereignisbeziehungen bislang unbeachtet.

Auf Netzwerkebene liefern statistische und payloadbasierte Merkmale zwar wertvolle Hinweise auf verdächtiges Verhalten, ihre Aussagekraft wird jedoch durch Verschlüsselung und Spoofing-Techniken eingeschränkt. Dadurch kann es schwieriger werden, komplexe und mehrstufige Angriffsszenarien wie APTs zu erkennen, wenn nur Netzwerkdaten berücksichtigt werden.

Neuere multimodale Arbeiten versuchen diese Schwächen zu kompensieren, indem sie Daten unterschiedlicher Quellen fusionieren. Die untersuchten Fusionsstrategien beschränken sich aufgrund der Heterogenität der Daten auf Late- oder Intermediate-Fusion-Architekturen, bei denen entweder nur

Entscheidungsergebnisse kombiniert oder Zwischenschichten verknüpft werden. Zwar integrieren einige Ansätze Netzwerk- und Hostdaten, doch bleibt die semantische Dimension der Lognachrichten in diesen Verfahren unterrepräsentiert. Zudem wird der konkrete Einfluss verschiedener Fusionsarten auf die Erkennung von APT-Angriffen in keiner der betrachteten Arbeiten ausreichend analysiert.

Damit besteht eine Forschungslücke in der Kombination semantischer Lognachrichten mit Netzwerk-Flow-Daten innerhalb eines multimodalen Modells. Die vorliegende Arbeit adressiert diese Lücke, indem sie den Einfluss der Fusion kontextualisierter Logrepräsentationen und von Netzwerkinformationen auf die Erkennung von APT-Aktivitäten untersucht, wobei sowohl Late- als auch Intermediate-Fusion-Ansätze in Kombination mit Deep-Learning- und Machine-Learning-Klassifikatoren verglichen werden.

4. Forschungsmethode

Im folgenden Kapitel wird das Vorgehen der vorliegenden Arbeit zur Untersuchung der Forschungsfragen erläutert. Zunächst erfolgt die formale Problemdefinition, gefolgt von einer detaillierten Beschreibung der untersuchten unimodalen und multimodalen Modelle sowie ihrer Parameter. Daraufhin wird der Aufbau der Trainingspipeline, mit welcher die Modellparameter bestimmt, sowie die Modelle bewertet werden, beschrieben. Abschließend werden die Metriken erläutert, anhand welcher die Modelle bewertet und miteinander verglichen werden.

4.1. Problemdefinition

Ziel dieser Arbeit ist die Analyse, wie sich die multimodale Verarbeitung statistischer Netzwerk-Flow-Merkmale in Kombination mit semantisch repräsentierten Lognachrichten auf die Erkennung von Netzwerk-Flows auswirkt, die spezifischen Phasen von APT-Angriffen zugeordnet sind.

APT-Angriffe sind durch ein mehrstufiges Vorgehen gekennzeichnet, bei dem die einzelnen Phasen unterschiedliche Netzwerk- und Systemaktivitäten aufweisen (siehe Abschnitt 2.1). Vor diesem Hintergrund wird untersucht, ob multimodale Klassifikationsansätze die Detektion phasenspezifischer Netzwerk-Flows verbessern.

Hierzu werden Netzwerk- und Hostlog-Informationen kombiniert, um Angriffsverhalten kontextsensitiv zu erkennen. Die formale Definition eines binären Klassifikationsproblems dient der klaren Beschreibung des Modellziels sowie des Zusammenhangs zwischen Datenquellen und Entscheidungsfunktion.

Sei die Menge aller Hosts durch $\mathcal{H}$ gegeben. Ein Flow zwischen zwei Hosts $h, h' \in \mathcal{H}$ wird durch einen Featurevektor $f_{h,h'} \in \mathbb{R}^d$ beschrieben, welcher statistische Eigenschaften der Verbindung von h zu h' abbildet.

Zusätzlich sei eine Sequenz zugehöriger Lognachrichten

$$l_{h,h'} = (l_{h,h',1}, \ldots, l_{h,h',n}), \quad l_{h,h',j} \in \mathcal{L}, \tag{4.1}$$

definiert, die auf den beteiligten Hosts während der Entstehung des Flows generiert wurden. Diese Sequenz wird mittels einer Abbildung

$$\Phi : \mathcal{L}^* \longrightarrow (\mathbb{R}^k)^* \tag{4.2}$$

in eine Sequenz aus Repräsentationsvektoren im Merkmalsraum $\mathbb{R}^k$ überführt, siehe Abschnitt 2.2. Gesucht ist dann eine Klassifikationsfunktion

$$h : \mathbb{R}^d \times (\mathbb{R}^k)^* \longrightarrow \{0, 1\}, \tag{4.3}$$

die entscheidet, ob ein Flow mit den zugehörigen Lognachrichten als normal (0) oder anomal (1) einzustufen ist. Wie die Abbildungen sowohl der Flow-Featurevektoren als auch der Lognachricht-Sequenzen konstruiert werden und welche Verfahren hierfür zum Einsatz kommen, wird im späteren Kapitel 5.2 genauer beschrieben.

4.2. Modelle und Modellparameter

Zur Untersuchung des Einflusses multimodaler Klassifikation werden zunächst unimodale Modelle trainiert. Diese Modelle dienen als Baseline, um zu evaluieren, in welchem Maße Angriffe anhand der einzelnen Datenquellen erkannt werden können. Außerdem bilden die leistungsstärksten unimodalen Modelle anschließend die Grundlage für die multimodalen Late-Fusion-Ansätze.

4.2.1. Unimodale Modelle

Als Klassifikationsalgorithmen werden in dieser Arbeit Random-Forest-Modelle [63] und MLP-Modelle [64] untersucht. Random Forest dient dabei als Referenzmodell, um die Performance klassischer Machine-Learning-Methoden gegenüber Deep Learning Ansätzen zu bewerten. Aufgrund seiner Skaleninvarianz benötigt Random Forest nur wenig Datenvorbereitung und hat in anderen Untersuchungen bereits bessere Ergebnisse insbesondere auf tabellarischen Daten erzielt, sodass er sich gut als Baseline und für den Vergleich mit den neuronalen Modellen eignet [65].

Wie im späteren Abschnitt 4.3 detaillierter beschrieben, werden die Modellparameter innerhalb der Trainingspipline mittels Cross-Validation für jede Modalität optimiert. Tabelle 4.1 fasst die untersuchten Hyperparameter und Variablen zusammen. Für den Random Forest werden insbesondere folgende Parameter untersucht:

- Anzahl der Bäume: Eine größere Anzahl an Entscheidungsbäumen verringert durch die Mittelung der Vorhersagen den Generalisierungsfehler des Modells, führt jedoch auch zu höherem Rechenaufwand beim Training und der Inferenz. Da der Fehler mit steigender Baumzahl konvergiert [63] und mit den entstehenden Modellen viele Samples untersucht werden, soll ein geeigneter Kompromiss zwischen Robustheit und Effizienz gefunden werden.
- Maximale Baumtiefe: Die Tiefe der Bäume bestimmt den Grad der Modellkomplexität. Eine größere Tiefe ermöglicht das Erlernen komplexerer Entscheidungsgrenzen, birgt jedoch das Risiko von Overfitting.
- Split-Kriterium: Dieses definiert die Metrik zur Bewertung der Aufteilung in jedem Knoten. Das gewählte Kriterium beeinflusst, wie stark der Informationsgewinn bei Feature-Splits gewichtet wird.
- Klassengewichtung: Da in sicherheitsrelevanten Datensätzen häufig ein Ungleichgewicht zwischen normalen und anomalen Klassen besteht (siehe Abschnitt 2.3), kann eine per Klassengewichtung beim Training eingeführt werden. Gewichtet werden die Klassen invers proportional zur Klassenhäufigkeit mit der Funktion *compute_class_weight* [66] mit:

$$w_i = \frac{N}{K \cdot N_i} \tag{4.4}$$

wobei w_i die Gewichtung der Klasse i, N die Gesamtanzahl an Samples, K die Anzahl der Klassen und N_i die Anzahl der Samples der Klasse i sind [66].

Das MLP dient als Deep-Learning-Ansatz, um die Leistungsfähigkeit moderner Architekturen gegenüber klassischen Machine-Learning zu bewerten. Der Input-Layer enthält eine der jeweiligen Merkmalsdimension entsprechende Anzahl an Neuronen, gefolgt von mehreren vollvernetzten Hidden-Layer, welche jeweils mit Dropout-Layern kombiniert werden. Für das MLP umfasst die Hyperparametersuche:

- Anzahl und Neuronen der Hidden Layer: Eine größere Neuronenzahl und Tiefe der Schichten erhöt die Modellkomplexität, kann jedoch auch das Risiko für Overfitting erhöhen.
- Aktivierungsfunktionen: Bestimmen die Art der Nichtlinearität innerhalb des Netzwerks. Um den Parametersuchraum einzuschränken, werden ausschließlich Rectified Linear Unit (ReLU) als de-facto-Standard sowie Tanh als alternative Aktivierungsfunktion berücksichtigt [67].
- Lernrate: Die Lernrate steuert die Schrittweite der Gradientenaktualisierung, während des Trainings. Eine zu hohe Lernrate kann zu Instabilität führen, weil die Parameter des Modells bei jedem Update zu große Sprünge machen und somit den optimalen Bereich überspringen. Eine zu niedrige Lernrate hingegen kann den Trainingsprozess stark verlangsamen.
- Dropout: Dropout reduziert Overfitting, indem während des Trainings zufällig Neuronen deaktiviert werden. Die Variation des Dropout-Faktors zeigt den Einfluss der Regularisierung auf die Modellverallgemeinerung.
- Gewichtung der Klassen: Wie bereits für die Random Forest Parameter beschrieben wird eine invers proportionale Klassengewichtung eingeführt nach 4.4.

Modell	**Parameter**	**Werte**
Random Forest	Baum Anzahl	[100, 300, 500]
	Maximale Baumtiefe	[10, 30, 50]
	Kriterium	[Gini, Entropy]
	Klassen-Gewichtung	[Ohne, Balanciert]
MLP	Hidden Layer	[(1024, 512, 256), (512,), (2048, 1024), (256, 256, 256, 128), (256,), (256, 128)]
	Aktivierungsfunktion	[relu, tanh]
	Lernrate	[0.01, 0.001, 0.0001]
	Dropout	[0, 0.2, 0.5]
	Klassen-Gewichtung	[Ohne, Balanciert]

Tabelle 4.1.: Übersicht der evaluierten Hyperparameter für die unimodalen Random Forest und MLP Klassifikatoren. Rot markierte Parameter wurden nach Untersuchen in Kapitel 5.5 entfernt, grüne hinzugefügt

Im Folgenden werden modalitätsspezifische Anpassungen der Modelle und Vorverarbeitungsschritte erläutert, die sich aus den jeweiligen Datenrepräsentationen und Modellarchitekturen ergeben. Eine eindeutige Abgrenzung zur Trainingspipeline ist dabei nicht einfach möglich, da beide Aspekte eng miteinander verknüpft sind.

Embedding-Pooling

Wie in Abschnitt 4.1 beschrieben, liegen die Log-Nachrichten für die Klassifikation als Sequenzen von Repräsentationsvektoren vor. Da sowohl Random Forests als auch MLPs keine Sequenzverarbeitung unterstützen, wird ein Pooling-Ansatz verwendet, um diese Sequenzen in feste Vektorrepräsentationen zu überführen.

Pooling ist eine einfache und zugleich effektive Methode, bei der die Vektoren einer Sequenz zu einem einzelnen Repräsentationsvektor aggregiert werden, beispielsweise durch Max-, Min- oder Average-Pooling. Mean Averaging hat sich insbesondere bei der Kombinierung von Dokumentenembeddings[68] bewährt. Dabei werden die Vektoren über alle Sequenzelemente gemittelt:

$$l_{\text{avg}} = \frac{1}{n} \sum_{i=1}^{n} l_i \quad \text{mit} \quad l_{\text{avg}} \in \mathbb{R}^{1024}. \tag{4.5}$$

Betrachtet man die semantischen Embeddings, so liegen diese in einem Raum, in dem semantische und strukturelle Ähnlichkeiten durch Abstände beschrieben werden. Zwei Log-Nachrichten $\mathbf{l}_i$ und $\mathbf{l}_j$ gelten als ähnlich, wenn ihr euklidischer Abstand klein oder der Kosinusabstand hoch ist [23].

Durch das Pooling entsteht ein Repräsentationsvektor, der den Zentroid der gesamten Embedding-Sequenz im Vektorraum bildet und damit deren semantischen Inhalt erfasst. Auf diese Weise werden die Informationen der vollständigen Log-Sequenz kompakt zusammengefasst.

Flow-Statistiken

Wie in Abschnitt 4.1 erläutert, liegt für jeden Netzwerkflow ein Feature-Vektor vor. Die genaue Extraktion dieser Merkmale wird in Kapitel 5.2 beschrieben, während die Tabelle 5.1 den Aufbau des resultierenden Feature-Vektors darstellt. Neben Zeitstempeln, IP-Adressen, Ports und Protokollnummern umfasst dieser ausschließlich numerische, nicht-ordinal, kategorische Merkmale.

Bei der Vorverarbeitung der Flows werden IP-Adressen, Ports und Zeitstempel aus den Vektoren entfernt, um Overfitting zu vermeiden. Dadurch soll verhindert werden, dass Modelle spezifische Adressen oder Zeitpunkte mit Angriffen assoziieren, anstatt allgemeine Muster zu lernen.

Da Port- und Protokollnummern nicht-ordinale, kategorische Merkmale sind, werden dem MLP, wie in Abbildung 4.1 dargestellt, zwei Embedding-Layer hinzugefügt [70]. Die Embedding-Layer ordnen jeder Kategorie einen dichten Vektor in einem 10-Dimensionalen Raum zu. Dazu erhält jede Kategorie einen eindeutigen Index. Das Embedding-Layer enthält eine Matrix, deren Zeilen den Vektoren der einzelnen Kategorien entsprechen. Bei der Inferenz wählt das Modell die Zeile aus, die dem Index der jeweiligen Kategorie entspricht. Während des Trainings werden die Werte

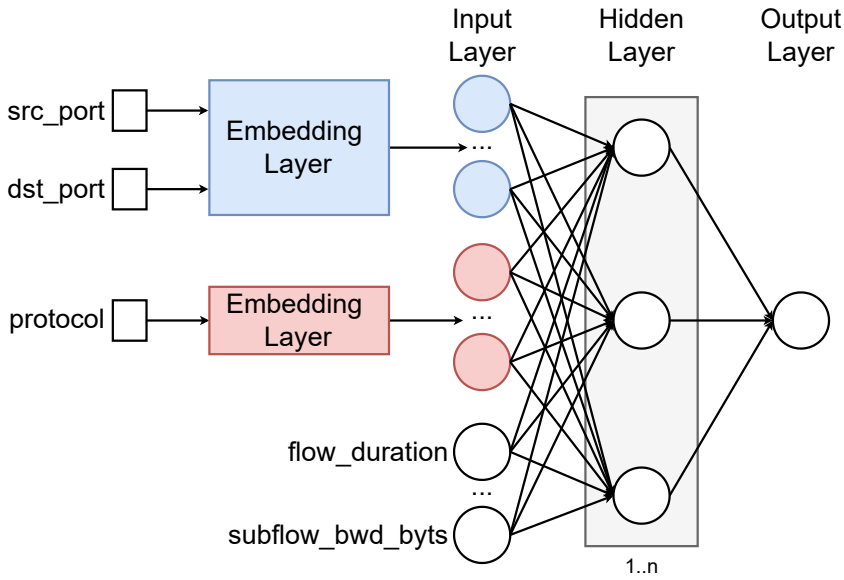

Abbildung 4.1.: MLP mit Embedding-Layer für Port- und Protokoll-Features [69]

der Embedding-Matrix zusammen mit den anderen Gewichten des Netzes über Backpropagation gelernt. Damit wird die Eingabedimension im Vergleich zu klassischen One-Hot-Encodings kleiner gehalten, was bei Portnummern mit dem Wertebereich von 0 bis 65.535 von Vorteil ist. Für Random Forest ist diese Vorverarbeitung mit Embedding-Layer nicht notwendig, da Baum-Modelle direkt mit kategorischen Features arbeiten können.

4.2.2. Multimodale Klassifikation

In Abschnitt 2.4 wurden die Konzepte der Early-, Intermediate- und Late-Fusion eingeführt. Aufbauend darauf wird im Folgenden beschrieben, wie durch Soft-Decision-Fusion und Feature-Konkatenation unter Verwendung nichtlinearer Modelle die Erkennung von APT-Angriffen untersucht wird.

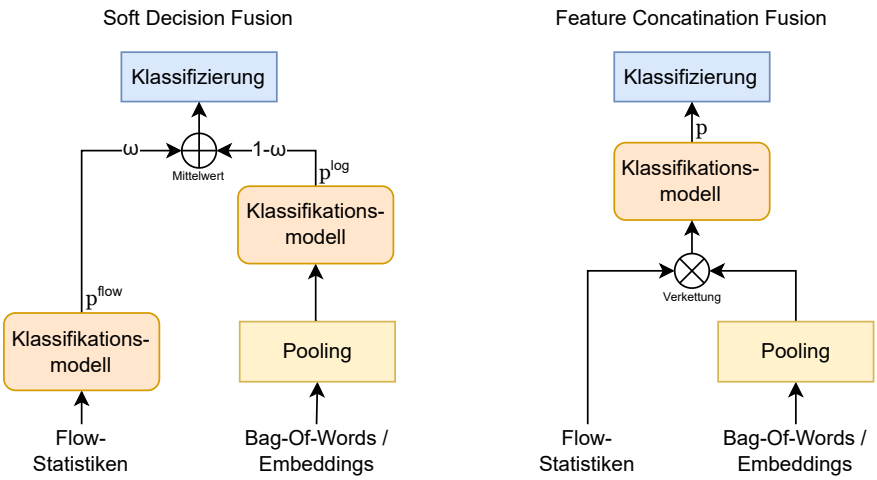

Abbildung 4.2.: Schematische Darstellung der multimodalen Klassifikatoren [9]

Late Fusion durch Soft Decision Fusion

Für die Soft Decision Fusion werden die Wahrscheinlichkeitsverteilungen der besten MLP-Modelle für die Flow- und Log-Klassifikation kombiniert. Sei $p^{(\text{flow})} \in [0, 1]$ die vom Flow-Modell bestimme Anomaliewahrscheinlichkeit und $p^{(\text{log})} \in [0, 1]$ die entsprechende Wahrscheinlichkeit des Log-Modells. In der ungewichteten Variante ergibt sich die Fusion als arithmetisches Mittel:

$$p^{(\text{SDF})} = \tfrac{1}{2}(p^{(\text{flow})} + p^{(\text{log})}) \tag{4.6}$$

In der gewichteten Variante wird ein Parameter $\omega \in (0, 1)$ eingeführt, sodass:

$$p^{(\text{SDF})} = \omega\, p^{(\text{flow})} + (1 - \omega)\, p^{(\text{log})} \tag{4.7}$$

Der Parameter ω wird im Rahmen der Hyperparameter-Optimierung mittels Random Search (siehe Abschnitt 4.3) bestimmt, um ein optimales Gleichgewicht zwischen beiden Modellen zu finden.

Intermediate Fusion durch Feature Concatination

Wie in Abbildung 4.2 dargestellt, werden bei der Intermediate Fusion mittels Feature-Konkatenation die Repräsentationen beider Modalitäten zunächst zusammengeführt. Sei $f \in \mathbb{R}^{d_f}$ die Darstellung der Flow-Daten und $l \in \mathbb{R}^{d_l}$ jene der Log-Daten, wobei jeweils entweder gepoolte Embeddings oder TF-IDF-Vektoren verwendet werden. Durch die Konkatenation entsteht ein gemeinsamer Merkmalsvektor:

$$z = [f; l] \in \mathbb{R}^{d_f + d_l} \tag{4.8}$$

Die ursprünglichen Richtungen und Lagen der beiden Vektoren bleiben erhalten, sie existieren jedoch nun in orthogonal getrennten Unterräumen innerhalb eines größeren Merkmalsraums. Dadurch liegen die Merkmale beider Modalitäten nebeneinander, ohne unmittelbar zu interagieren. Erst das nachfolgende Modell kann nichtlineare Zusammenhänge und Wechselwirkungen im kombinierten Raum erfassen.

Als Klassifikatoren werden, analog zu Abschnitt 4.2.1, Random Forest und MLPs eingesetzt. Die für beide Modelle untersuchten Hyperparameter sind in Tabelle 4.2 zusammengefasst und entsprechen der Beschreibung in Abschnitt 4.2.1.

Modell	Parameter	Werte
Late Fusion + Random Forest	Baum Anzahl	[100, 300, 500]
	Maximale Baumtiefe	[10, 30, 50]
	Kriterium	[Gini, Entropy]
	Klassen-Gewichtung	[Ohne, Balanciert]
Late Fusion + MLP	Hidden Layer	[(512,), (1024, 512, 256), (256, 256, 256, 128), (2048, 1024), (1024, 512), (512, 256, 128),]
	Aktivierungsfunktion	[relu, tanh]
	Lernrate	[0.01, 0.001, 0.0001]
	Dropout	[0, 0.2, 0.5]
	Klassen-Gewichtung	[Ohne, Balanciert]
Weighted SDF	w	(0 - 1)

Tabelle 4.2.: Übersicht der evaluierten Hyperparameter für Random Forest, MLP und Fusionsmethoden. Rot markierte Parameter wurden nach Untersuchen in Kapitel 5.5 entfernt, grüne hinzugefügt

4.3. Trainingspipeline

Im Folgenden wird das Vorgehen zum Trainieren und Evaluieren der in den Abschnitten 4.2.1 und 4.2.2 beschriebenen Klassifikatoren erläutert. Eine Übersicht über die gesamte Trainingspipeline ist in Abbildung 4.3 dargestellt.

Ziel der Trainingspipeline ist es, für jedes Modell eine möglichst zuverlässige Einschätzung der generalisierten Klassifikationsleistung zu erhalten. Damit wird untersucht, inwieweit die erlernten Entscheidungsgrenzen in der Lage sind, bislang unbekannte Anomalien bzw. Angriffe korrekt zu erkennen, nachdem das Modell auf andere Angriffstypen trainiert wurde.

Wie in der Übersicht 4.3 dargestellt, wird der Datensatz zunächst in ein Trainings- und ein Testset unterteilt. Diese Aufteilung wird vorgenommen, um die Leistungsfähigkeit der Modelle abschließend bewerten zu können. Das Trainingsset dient dazu, die am besten generalisierenden Hyperparameter der Modelle zu bestimmen und die Modelle zu trainieren. Das Testset hingegen wird erst nach dem Tuning und dem Training verwendet, um zu überprüfen, wie gut die Modelle auf bisher unbekannte Daten generalisierten.

In anderen Arbeiten erfolgt solch eine Aufteilung häufig zufällig anhand eines prozentualen Anteils der Gesamtdaten [7, 62]. Da der untersuchte Datensatz, wie in Abschnitt 5.1 beschrieben, aus acht separaten Simulationen besteht, werden die Splits simulationsbasiert vorgenommen. Dadurch wird sichergestellt dass es kein Information Leakage gibt und, dass das Modell keine simulationsspezifischen Strukturen erlernt, die die Klassifikationsleistung bei der Modellbewertung künstlich erhöhen könnten.

Zur Überwachung des Lernprozesses und zur Vermeidung von Overfitting wird bei den MLP-Modellen ein Szenario aus dem finalen Trainingsdatensatz als Validierungsdaten ausgewählt und

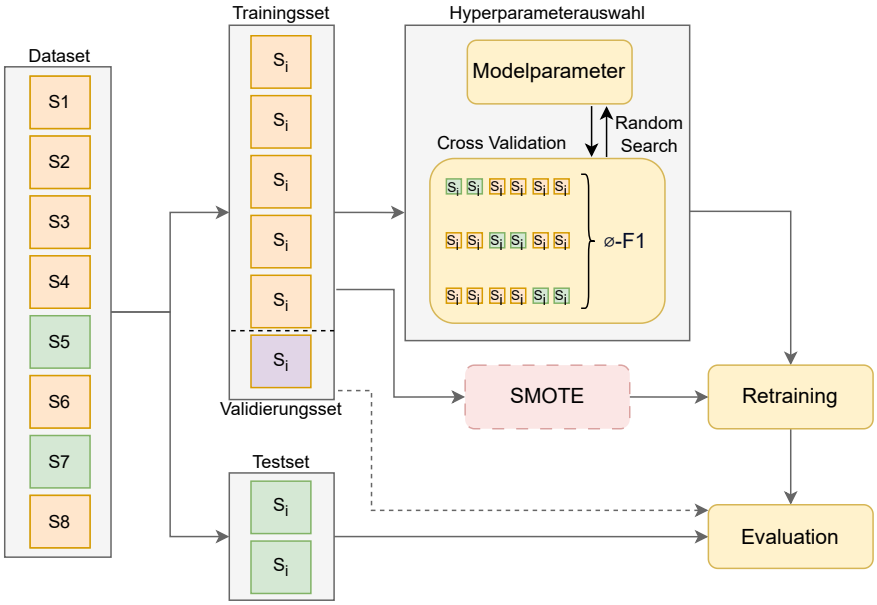

Abbildung 4.3.: Schematische Darstellung der Trainingspipeline mit Hyperparametersuche und optionalem Einsatz von SMOTE

während des Trainings zurückgehalten. Dieser Validierungssplit ermöglicht eine fortlaufende Kontrolle der Modellleistung während des Trainings und dient als Grundlage für mögliches Early Stopping [71].

Um die Modelle fair vergleichen zu können, müssen geeignete Hyperparameter bestimmt werden, die eine gute Generalisierung auf den Trainingsdaten ermöglichen. Die Auswahl dieser Parameter erfolgt im Hinblick auf den Bias-Variance-Tradeoff, da Parameter wie die Baumtiefe oder Hidden Layer einen Einfluss auf die Modellvarianz haben und ein ausgewogenes Verhältnis zwischen Bias und Varianz entscheidend für die Modellleistung ist [71]. Gut gewählte Parameter verhindern somit Overfitting oder Underfitting und stellen so sicher, dass die untersuchten Leistungsunterschiede die tatsächliche Modellfähigkeit widerspiegeln.

Die dazu jeweils untersuchten Hyperparameter sind in den Abschnitten 4.2.1 und 4.2.2 zu den entsprechenden Modellen beschrieben. Zur Begrenzung des Optimierungsaufwands wird ein Random-Search-Verfahren eingesetzt, bei dem 20 zufällige Konfigurationen aus den Hyperparameter-Suchräumen der Modelle ausgewählt und getestet werden. Bei den in Tabelle 4.1 beschriebenen Parametern für die Random-Forest-Klassifikatoren deckt dieses Vorgehen beispielsweise 55,5 %, also mehr als die Hälfte aller möglichen Parameterkombinationen, ab.

Für eine robuste Einschätzung der Modellleistung bei unterschiedlichen Hyperparameterkonfigurationen wird eine 3-Fold-Cross-Validation durchgeführt, bei der jeweils zwei Szenarien als Evaluationsset aus den Trainingsdaten verwendet werden. Die Aufteilung in mehrere Validierungssets reduziert das Risiko von Overfitting, indem sie verhindert, dass Parameter gewählt werden, die nur für einen bestimmten Train-/Validierungssplit vorteilhaft sind. Anstatt die Bewertung auf einem einzelnen Split basieren zu lassen, werden die Ergebnisse über alle drei Folds gemittelt. Dadurch werden Hyperparameter bevorzugt, die im Durchschnitt eine stabile und hohe Performance erzielen.

Als Auswahlkriterium für die Parameter dient der durschnittliche macro F1-Score der entstehenden Modelle. Er wird verwendet, weil er Precision und Recall für jede Klasse gleichermaßen berücksichtigt und die so entstehenden F1-Scores anschließend gemittelt werden (siehe Abschnitt 4.4). Im Kontext der Intrusion Detection, bei der als Angriffe erkannte legitime Aktivitäten falsche Alarme auslösen und verpasste Angriffe kritisch sind, bietet der macro F1-Score eine faire Bewertung der Modellleistung, selbst wenn die Klassen unausgeglichen sind. Auch hierfür erfolgt die Aufteilung in Folds simulationsbasiert, um Information Leakage zu vermeiden.

Abschließend werden die finalen Modelle auf dem Hold-Out-Testset evaluiert und miteinander verglichen. Die dafür verwendeten Bewertungsmetriken werden im späteren Abschnitt 4.4 beschrieben.

4.4. Evaluation

Im folgenden Kapitel werden die zur Bewertung der Modelle genutzten Evaluationsmetriken definiert. Die trainierten Modelle werden dazu, wie in Abschnitt 4.3 beschrieben, auf das zurückgehaltene Testset angewendet. Dieses umfasst zwei Szenarien des Datensatzes und repräsentiert somit zwei Angriffe in den Modellen bislang unbekannten Netzwerken. Zur Beurteilung der Klassifikationsleistung werden Accuracy, Precision, Recall, F1-Score sowie der AUC-Wert bestimmt. Da die Klassen unterschiedlich häufig vertreten sind, werden Precision, Recall und F1-Score als Makro-Metriken berechnet, heißt es werden die Ergebnisse beider Klassen bestimmt und gemittelt, sodass alle Klassen gleich gewichtet in die Bewertung einfließen.

Accuracy, Precision und Recall

Die Genauigkeit (engl. Accuracy) bestimmt den Anteil korrekt klassifizierter Samples an der Gesamtzahl aller bewerteten Datenpunkte des Testsets. Bei unausgeglichenen Klassenverteilungen, wie im vorliegenden Datensatz mit mehr regulären als anomalen Flows, kann die Accuracy jedoch ein verzerrtes Bild der Modellleistung vermitteln, da ein Modell auch mit schlechterer Erkennung der Minderheitsklasse einen hohen Genauigkeitswert erreichen kann, indem es überwiegend die Mehrheitsklasse korrekt klassifiziert.

Accuracy ist definiert als [72]:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.9}$$

dabei sind True Positives (TP) die korrekt als Angriff erkannten Flows, True Negatives (TN) die korrekt als normal klassifizierten Flows, False Positives (FP) die fälschlicherweise als Angriff klassifizierten normalen Flows sowie False Negatives (FN) die übersehenen Angriffsflows.

Die Precision gibt den Anteil korrekt identifizierter Angriffe an allen als Angriff vorhergesagten Flows an [72]:

$$Precision = \frac{TP}{TP + FP} \tag{4.10}$$

Sie gibt an, wie zuverlässig das Modell positive Vorhersagen trifft. Ein hoher Precision-Wert zeigt, dass die vom Modell erzeugten Alarme überwiegend korrekt sind.

Recall misst den Anteil korrekt erkannter Angriffe an allen tatsächlich vorhandenen Angriffsflows:

$$Recall = \frac{TP}{TP + FN} \tag{4.11}$$

Ein hoher Recall ist besonders in sicherheitskritischen Anwendungen von Bedeutung, da das Übersehen von Angriffen ein potenziell hohes Risiko darstellt.

F1-Score

Da Precision und Recall in einem Zielkonflikt stehen, eine Erhöhung der Precision durch strengere Klassifikationskriterien führt oft zu einer niedrigeren Erkennungsrate und umgekehrt wird der F1-Score als harmonischen Mittel der beiden Werte bestimmt [72]:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{4.12}$$

Der F1-Score stellt insbesondere bei unausgeglichenen Klassenverteilungen ein robusteres Maß für die Modellleistung dar. Wie in Abschnitt 4.3 beschrieben, dient der macro F1-Score in dieser Arbeit als Kriterium zur Auswahl der optimalen Hyperparameter, da er das beste Gleichgewicht zwischen Precision und Recall widerspiegelt.

AUC

Zusätzlich wird zum Vergleich die Receiver Operating Characteristic-Kurve (ROC) und die Fläche unter der ROC-Kruve, bezeichnet als Area Under the Curve (AUC), betrachtet. Die ROC-Kurve stellt die True-Positive-Rate (TPR) in Abhängigkeit von der False-Positive-Rate (FPR) für verschiedene Entscheidungsschwellen dar [72]:

$$TPR = \frac{TP}{TP + FN}, \qquad FPR = \frac{FP}{FP + TN} \tag{4.13}$$

Die AUC beschreibt die Fläche unter dieser Kurve und liefert eine schwellwertunabhängige Maßzahl für die Genauigkeit des Klassifikators. Ein Wert von $AUC = 1$ steht für eine perfekte Unterscheidung zwischen normalen und anomalen Flows, während $AUC = 0.5$ einer zufälligen Klassifikation entspricht.

5. Implementierung und Durchführung

In diesem Kapitel wird die praktische Umsetzung der unimodalen und multimodalen Klassifikation beschrieben. Es umfasst die Auswahl eines geeigneten Intrusion Detection Datensatzes, welcher sowohl Netzwerk- als auch Hostdaten von APT-Angriffszenarien abbildet, als auch die Datenaufbereitung behandelt. Dabei werden Biflows extrahiert, Logdaten geparsed, in eine strukturierte Form überführt und anschließend mit den Flows korreliert sowie gelabelt. Darüber hinaus wird die Implementierung der Trainingspipeline der Modelle erläutert, einschließlich dem Einsatz von SMOTE zum erzeugen von zusätzlichen Trainingsdaten sowie notwendiger Anpassungen und besonderer Aspekte während des Trainingsprozesses.

5.1. Datengrundlage

Im folgenden wird ein geeigneter Datensatz für die nachfolgenden Untersuchungen ausgewählt. Dafür sollen folgende Anforderungen vom Datensatz erfüllt sein:

- Um eine binäre Klassifikation durchzuführen, umfasst der Datensatz sowohl reguläre Nutzeraktivitäten als auch mehrstufige, APT-ähnliche Angriffsabläufe (siehe Kapitel 2.1).
- Netzwerkverkehr und System-Logdaten liegen vollständig vor, sodass eine multimodale Klassifikation möglich ist.
- Netzwerk- und Host-Daten sind zeitlich und logisch verknüpfbar, wodurch eine eindeutige Zuordnung zwischen Netzwerk-Flow und Log-Nachrichten möglich ist.
- Für eine saubere Labelzuweisung sind Angriffsaktionen klar abgrenzbar und den zugehörigen Netzwerkaktivitäten präzise zuordenbar.
- Das Datenvolumen ist ausreichend, um Modelle zu trainieren und deren Leistungsfähigkeit zu evaluieren.

Die Verfügbarkeit geeigneter realer Datensätze ist stark begrenzt, da Netzwerk-Payloads und Host-Logs häufig sensible Informationen enthalten, deren Veröffentlichung aus Datenschutzgründen nicht unbedenklich sind. Daher basieren die meisten verfügbaren Datensätze auf simulierten Netzwerkumgebungen [73].

Goldschmidt et al. [73] haben in ihrer Arbeit einen aktuellen Überblick über verfügbare Network-Intrusion-Datasets vorgestellt. Aus den gesammelten Datensätzen wurden Unraveled [74] sowie das AIT Log Data Set v2.0 (AIT-LDS) [75] als geeignete Kandidaten für den Einsatz in dieser Arbeit genauer betrachtet. Während die Host-Logaufzeichnungen von Unraveled unvollständig sind, bietet AIT-LDS vollständige Informationen über die gesamte Simulation und erfüllt alle definierten Anforderungen.

AIT-LDS Datensatz

Der AIT Log Data Set [76] ist eine vom Austrian Institute of Technology erstellte Sammlung aufgezeichneter Host-Logs, Netzwerk-Aufzeichnungen und Konfigurationsdateien zur Evaluation von Intrusion Detection Systemen.

Erfasst wurden die Daten in acht unterschiedlichen simulierten Unternehmensnetzwerken, welche jeweils einen realistischen Angriffsfall darstellen sollen. Die Simulationen umfassen externe sowie interne Nutzer, verschiedene Anwendungsserver wie Samba-Dateifreigaben, WordPress-, VPN-, Proxy-, Mail- und Cloudshare-Server sowie DNS-Server und externe Mailserver. Die Anzahl der Hosts variiert je nach Simulationsszenario.

Die Szenarien fox, harrison, wardbeck und wheeler bilden jeweils fünf Tage an simulierten Aktivitäten ab. Fox besteht aus 30 Netzwerkteilnehmern. Harrison umfasst 27 Teilnehmer, verfügt über weniger Mailserver und weist weniger interne, dafür jedoch mehr remote Mitarbeiter auf.

Wardbeck beinhaltet ebenfalls 30 Teilnehmer, allerdings mit einem stärkeren Fokus auf interne und remote Mitarbeitende bei vergleichsweise wenigen externen Nutzern. Wheeler besitzt mit 36 Teilnehmern sowohl mehr interne als auch mehr externe Nutzer. Zudem zeigen fox, harrison und wheeler im Vergleich zu wardbeck deutlich höhere Scan-Aktivitäten, da Discovery-Scans mit nmap über größere IP-Adressbereiche durchgeführt werden. In wheeler entfällt außerdem das Passwort-Cracking. Es wird angenommen, dass bereits gültige Zugangsdaten vorliegen, wodurch entsprechende Log-Einträge fehlen und der Angriff direkt in den nächsten Schritt übergeht.

Die Szenarien russellmitchell und santos erfassen jeweils nur vier Tage und enthalten dadurch insgesamt weniger Daten. Santos umfasst 30 Teilnehmer, während russellmitchell mit 22 Teilnehmern das kleinste Szenario ist. Beide weisen zudem geringe Scan-Aktivitäten auf.

Die Szenarien shaw und wilson decken dagegen sechs Tage Aktivität ab. Wilson umfasst 36 Teilnehmer und verfügt über besonders viele externe Nutzer. Shaw enthält dagegen nur 26 Teilnehmer, ein geringes Scan-Aufkommen und zusätzlich keine DNS Log-Nachrichten zur Exfiltration via DNS-teal.

In jedem Szenario wird nach einer gewissen Zeit ein mehrstufiger, APT-ähnlicher Angriff simuliert. Die Angriffe basieren auf der Annahme, dass der Angreifer bereits kompromittierte VPN-Zugangsdaten besitzt und sich mit gängigen Penetrationstools durch das Netzwerk bewegt.

Der grobe Ablauf ist in allen Szenarien ähnlich. Zunächst baut der Angreifer eine VPN-Verbindung ins Netzwerk auf. Anschließend erfolgen Discovery-Scans mit Traceroute und nmap (dns-brute), gefolgt von einem Scan nach aktiven Host-IPs im Subnetz. Auf den gefundenen Hosts werden anschließend mittels nmap nach aktiven Diensten auf Ports gesucht.

Unter der Annahme, dass einer der Server eine WordPress-Seite hostet, wird anschließend mit WPScan und DIRB nach Schwachstellen gesucht. Eine vorhandene WordPress-Schwachstelle wird ausgenutzt um eine Webshell zu erzeugen, über die sich der Angreifer einloggt. Auf dem Server werden anschließend diverse Reconnaissance-Befehele wie whoami, uname, netstat usw. ausgeführt.

Danach wird das Passwort eines höher privilegierten Accounts durch das Cracken der WordPress-Passworthashes mit John the Ripper erlangt. Nach dem Login in diesen Account führt der Angreifer erneut verschiedene Reconnaissance-Befehle aus, um weitere Informationen zu sammeln, und exfiltriert Daten mittels DNSteal.

Die konkrete Reihenfolge und Parameter der ausgeführten Befehle variiert zwischen den Szenarien, um unterschiedliche Angriffstaktiken abzubilden. Die exakte Abfolge ist für jedes Szenario im Datensatz als Ground Truth dokumentiert.

Für die Untersuchungen, werden die relevanten Daten aus allen acht Szenarien des AIT-LDS extrahiert um so eine möglichst große Datenbasis für das Training der Modelle und die Untersuchung zur Verfügung zu haben. Da die Angreifer in den Szenarien ähnliche Tools einsetzen, aber den Angriffsablauf und wie sie die Tools einsetzten variieren, werden die Szenarien im Folgenden als unabhängige Angriffe, durchgeführt von der selben Gruppe betrachtet.

5.2. Datenvorbereitung

Bevor die Daten des AIT-LDS für die in Kapitel 4 beschriebenen Untersuchungen eingesetzt werden können, müssen sie in eine strukturierte Form überführt werden, um der in Abschnitt 4.1 definierten Problemstellung zu entsprechen. In den folgenden Abschnitten wird erläutert, wie Log-Nachrichten und Flow-Statistiken aus den semistrukturierten Log-Sammlungen und Paketaufzeichnungen extrahiert und gelabelt werden. Anschließend werden die Log-Nachrichten in Bag-of-Words- sowie Embedding-Repräsentationen transformiert.

5.2.1. Extraktion der Biflows aus PCAPs

Zusätzlich zum AIT-LDS steht der AIT Netflow Data Set (AIT-NDS) [77] zur Verfügung. Dieser bildet die Verbindungsdaten der im AIT-LDS enthaltenen PCAPs ebenfalls als Biflows ab. Allerdings liegen keine Informationen zur Extraktionsmethode oder zur Definition eines Flows vor. Daher wurden im Rahmen dieser Arbeit eigene Flow-Statistiken erzeugt, um sicherzustellen, dass diese der nachfolgenden Definition entsprechen und mit den vorbereiteten Logs korreliert werden können.

Für die Analyse wird der bidirektionale Datenverkehr (Biflows) zwischen zwei Hosts extrahiert [78]. Die Extraktion der Flows erfolgt mit der, auf Scapy [79] basierenden, Python-Implementierung des CICFlowMeter [80], da dieses Tool bereits häufig für Arbeiten im Bereich der netzwerkbasierten Intrusion Detection eingesetzt wurde [81][82] und dem Autor aufgrund seiner Python-Kenntnisse die Funktionsweise des Programms kontrollierbar und anpassbar ist.

In dieser Arbeit werden Biflows mit dem ersten Paket einer Kommunikation initiiert und lassen sich eindeutig durch Quell- und Ziel-IP-Adresse, Portnummern sowie den Protokolltyp identifizieren. Sie enden entweder bei verbindungsorientierten Protokollen wie TCP mit der Beendigung der Verbindung oder nach einem Timeout von zwei Minuten. Der Timeout orientiert sich am, im RFC 4787 definierten, minimalen NAT-Timeout für UDP [83]. Eine TCP-Verbindung gilt als beendet, wenn

einer der beiden Kommunikationspartner ein Paket mit gesetztem RST-Flag sendet oder beide ein Paket mit FIN-Flag übermitteln. Wie in Tabelle 5.3 dargestellt, wurden auf diese Weise 5,15 Millionen Flows aus dem AIT-LDS extrahiert.

Feature	**Beschreibung / Bedeutung**
src_ip	Quell-IP-Adresse des ersten Pakets im Flow
dst_ip	Ziel-IP-Adresse des ersten Pakets im Flow
src_port	Quell-Port des ersten Pakets im Flow
dst_port	Ziel-Port des ersten Pakets im Flow
protocol	Transportprotokoll (TCP, UDP)
timestamp	Zeitstempel des ersten Pakets
flow_duration	Dauer des Flows
flow_byts_s	Übertragene Bytes pro Sekunde im Flow
flow_pkts_s	Übertragene Pakete pro Sekunde im Flow
fwd_pkts_s	Durchschnittliche Pakete pro Sekunde von Quelle zu Ziel (Vorwärtsrichtung)
bwd_pkts_s	Durchschnittliche Pakete pro Sekunde Ziel zu Quelle (Rückwärtsrichtung)
tot_fwd_pkts	Gesamtanzahl Vorwärts-Pakete
tot_bwd_pkts	Gesamtanzahl Rückwärts-Pakete
totlen_fwd_pkts	Gesamtlänge der Vorwärts-Pakete (Bytes)
totlen_bwd_pkts	Gesamtlänge der Rückwärts-Pakete (Bytes)
fwd_pkt_len_max	Maximale Länge eines Vorwärts-Pakets
fwd_pkt_len_min	Minimale Länge eines Vorwärts-Pakets
fwd_pkt_len_mean	Durchschnittliche Länge Vorwärts-Paket
fwd_pkt_len_std	Standardabweichung der Vorwärts-Paketlängen
bwd_pkt_len_max	Maximale Länge eines Rückwärts-Pakets
bwd_pkt_len_min	Minimale Länge eines Rückwärts-Pakets
bwd_pkt_len_mean	Durchschnittliche Länge Rückwärts-Paket
bwd_pkt_len_std	Standardabweichung der Rückwärts-Paketlängen
pkt_len_max	Maximale Paketlänge (gesamt)
pkt_len_min	Minimale Paketlänge (gesamt)
pkt_len_mean	Durchschnittliche Paketlänge (gesamt)
pkt_len_std	Standardabweichung der Paketlängen
pkt_len_var	Varianz der Paketlängen
fwd_header_len	Summierte Header-Länge aller Vorwärts-Pakete
bwd_header_len	Summierte Header-Länge aller Rückwärts-Pakete
fwd_seg_size_min	Kleinste TCP-Segmentgröße in Vorwärtsrichtung
fwd_act_data_pkts	Anzahl der Forward-Pakete, die Nutzdaten transportieren
flow_iat_mean	Mittlere Zeit zwischen Paketen (Interarrival Time (IAT))
flow_iat_max	Maximale Interarrival Time
flow_iat_min	Minimale Interarrival Time
flow_iat_std	Standardabweichung der Interarrival Time
fwd_iat_tot	Gesamt-IAT Vorwärtsrichtung

fwd_iat_max	Maximale IAT Vorwärtsrichtung
fwd_iat_min	Minimale IAT Vorwärtsrichtung
fwd_iat_mean	Mittlere IAT Vorwärtsrichtung
fwd_iat_std	Standardabweichung IAT Vorwärtsrichtung
bwd_iat_tot	Gesamt-IAT Rückwärtsrichtung
bwd_iat_max	Maximale IAT Rückwärtsrichtung
bwd_iat_min	Minimale IAT Rückwärtsrichtung
bwd_iat_mean	Mittlere IAT Rückwärtsrichtung
bwd_iat_std	Standardabweichung IAT Rückwärtsrichtung
fwd_psh_flags	Anzahl PSH-Flags Vorwärtsrichtung
	(Daten ungepuffert an Anwendungsschicht übertragen)
bwd_psh_flags	Anzahl PSH-Flags Rückwärtsrichtung
fwd_urg_flags	Anzahl URG-Flags Vorwärtsrichtung
	(Signalisiert priorisierte Daten)
bwd_urg_flags	Anzahl URG-Flags Rückwärtsrichtung
fin_flag_cnt	Anzahl FIN-Flags
	(Kennzeichnet korrekte TCP-Verbindungsbeendigung)
syn_flag_cnt	Anzahl SYN-Flags (Anzahl Verbindungsaufbauversuche)
rst_flag_cnt	Anzahl RST-Flags
	(Abgebrochene oder fehlerhafte Verbindungen)
psh_flag_cnt	Anzahl aller PSH-Flags
ack_flag_cnt	Anzahl ACK-Flags (Bestätigungen übertragener Daten)
urg_flag_cnt	Anzahl aller URG-Flags
ece_flag_cnt	Anzahl ECE-Flags (ECN congestion Indikator)
down_up_ratio	Verhältnis Down-/Upload
pkt_size_avg	Durchschnittliche Paketgröße
init_fwd_win_byts	Initiale TCP-Window-Size (Vorwärts)
init_bwd_win_byts	Initiale TCP-Window-Size (Rückwärts)
active_max	Längste kontinuierliche Zeit mit aktivem Datentransfer
active_min	Kürzeste identifizierte aktive Phase
active_mean	Durchschnittliche Aktivitätszeit
active_std	Standardabweichung Aktivitätszeit
idle_max	Längste Leerlaufphase (keine Pakete)
idle_min	Kürzeste Leerlaufphase
idle_mean	Durchschnittliche Leerlaufzeit
idle_std	Standardabweichung der Leerlaufzeit
fwd_byts_b_avg	Durchschnittliche Bytes pro Forward-Bulk
	(zusammenhängender Datenblock)
fwd_pkts_b_avg	Durchschnittliche Anzahl Pakete pro Forward-Bulk
bwd_byts_b_avg	Durchschnittliche Bytes pro Rückwärts-Bulk
bwd_pkts_b_avg	Durchschnittliche Anzahl Pakete pro Rückwärts-Bulk.
fwd_blk_rate_avg	Durchschnittliche Datenrate der Forward-Bulk-Transfers
bwd_blk_rate_avg	Durchschnittliche Datenrate der Rückwärts-Bulk-Transfers

Duplizierte Features	
fwd_seg_size_avg	= fwd_pkt_len_mean
bwd_seg_size_avg	= bwd_pkt_len_mean
cwr_flag_count	= fwd_urg_flags
subflow_fwd_pkts	= tot_fwd_pkts
subflow_bwd_pkts	= tot_bwd_pkts
subflow_fwd_byts	= totlen_fwd_pkts
subflow_bwd_byts	= totlen_bwd_pkts

Tabelle 5.1.: Übersicht der mit CICFlowMeter extrahierten Flow-Features

Die extrahierten Flow-Features sind in der Tabelle 5.1 dargestellt und bestehen hauptsächlich aus detaillierten statistischen Merkmalen wie Paketgrößen, Interarrival Times, Header- und Payload-Informationen sowie protokollspezifische Informationen (TCP-Flags), welche die Kommunikation des abgebildeten Bi-Flows charakterisieren.

In dieser Arbeit werden bestimmte Merkmale bewusst aus der Feature-Menge ausgeschlossen. Dazu gehören insbesondere IP-Adressen sowie die Zeitstempel des Flows. Der Grund dafür ist, Overfitting zu vermeiden. Modelle könnten anhand dieser Features einzelne IP-Adressen oder konkrete Zeitpunkte mit Angriffen zu verknüpfen, anstatt allgemeine Muster zu lernen. Die Portnummern bleiben jedoch erhalten, da sie ein wichtiges Indiz zur Erkennung von Anwendungsprotokollen, wie zum Beispiel HTTP, sind. Die entfernten Features sind in Tabelle 5.1 unterstrichen markiert.

Die verbleibenden Merkmale sind überwiegend nicht kategorische numerische Features. Ausnahmen sind lediglich das Protokollfeld und Source- so wie Destination-Port. Weitere Vorbereitungsschritte sind modellabhängig und werden in späteren Abschnitten der Arbeit besprochen.

5.2.2. Flow-Labeling

Das zuvor beschriebene AIT-NDS beinhaltet einen Labeling-Algorithmus, der auf Basis von Hostname sowie Start- und Endzeit einzelnen Flows jeweils einen Angriffsschritt zuweist [77]. Die Angriffsschritte sind in einer Ground-Truth für jedes Szenario mit zugehörigen Start- und Endzeiten dokumentiert. In dieser Arbeit wird dieser Algorithmus übernommen, jedoch erweitert. Jedem Angriffsschritt wird zusätzlich eine MITRE ATT&CK-Taktik zugeordnet. Dafür wurde für jeden Schritt eine passende Technik aus der ATT&CK-Matrix ausgewählt. Die resultierenden Zuordnungen sind in Tabelle 5.2 dargestellt. Dieses Vorgehen ermöglicht eine detaillierte Analyse, für welche Angriffsphasen sich die Erkennung durch die Fusion verbessert oder verschlechtert.

Tabelle 5.3 zeigt die Labelverteilung pro Szenario. Dabei lassen sich starke Unterschiede in den Verteilungen erkennen, insbesondere beim Credential Access. Diese resultieren aus der Variation zwischen den Angriffen. Wie in Kapitel 5.1 beschrieben, werden zwar dieselben Schritte durchgeführt, jedoch variieren die eingesetzten Techniken. So wird beispielsweise beim Credential Access je nach Szenario zwischen Offline- oder Online-Cracking unterschieden, wobei Letzteres deutlich mehr Flows erzeugt.

Durchgeführter Angriffsschritt	MITRE ATT&CK Taktik
Mit VPN verbinden • VPN Verbindung trennen	Initial Access
Traceroute • Nmap • Nmap (dns-brute) • WPScan • DIRB	Reconnaissance
Upload RCE-Shell	Persistence
John the Ripper • WordPress Benutzer lesen • Shadow-File lesen	Credential Access
Benutzer-ID prüfen • Netstat prüfen • Resolv lesen • Netzwerkkonfiguration prüfen • Laufende Prozesse prüfen • Release prüfen • Gruppeninformationen lesen • Passwd-Datei lesen • Datum prüfen • WordPress Konfiguration prüfen • Web-Verzeichnisse listen • Profil lesen • WWW-Verzeichnisse listen • Who ausführen • Letzte Logins prüfen • ID prüfen • Whoami ausführen • Kernel-Version prüfen • Kernel-Info prüfen • Festplattenbelegung prüfen • Netstat NAT prüfen • Home-Verzeichnisse listen • Netstat Listening Sockets prüfen • CPU-Informationen prüfen • Systemlaufzeit prüfen • PWD ausführen • Verzeichnisinhalt listen • Netstat UDP prüfen • sudo Berechtigungen prüfen • Root-Verzeichnis anzeigen • Getent prüfen	Discovery
Reverse Shell starten	Command and Control
PTY öffnen	Execution
Benutzer wechseln	Privilege Escalation
Befehlsverlauf löschen	Defense Evasion
Daten exfiltrieren (DNSteal)	Exfiltration

Tabelle 5.2.: Übersicht der Angriffsschritte und deren Klassifikation anhand der MITRE ATT&CK-Matrix

Die Tabelle verdeutlicht außerdem ein Ungleichgewicht in der Labelverteilung, das beim Modelltraining berücksichtigt werden muss, da dies je nach Verfahren die Modellqualität beeinflussen kann.

5.2.3. Vorbereitung der Log-Nachrichten

In verteilten, heterogenen Netzwerkinfrastrukturen stammen Systemlogs häufig von unterschiedlichen Quellsystemen und Softwarekomponenten. Das führt zu stark variierenden Strukturen zwischen den Quellen und Nachrichten. Auch im AIT-LDS liegen Host-Logs unstrukturiert vor und unterscheiden sich in Format und Inhalt.

Viele Arbeiten adressieren diese Heterogenität durch ML-basierte Log-Template-Parser wie Drain [84] oder Spell [85]. Dabei betrachten sie den variablen Anteil in Log-Nachrichten oft als Rauschen und konzentrieren sich auf Abfolgen von Templates (siehe Abschnitt 3.1). Allerdings führt dieses Vorgehen zu einem Informationsverlust, da semantisch relevante Details einzelner Nachrichten verworfen werden. Listing 5.1 zeigt beispielhaft den Unterschied zwischen einer Log-Nachricht und ihrer Template-Repräsentation.

Es lässt sich argumentieren, dass die Variablen eine Rolle für die Semantik der Nachrichten spielt. Kontext-Embeddings bieten hier einen möglichen Ansatz um diese zu erfassen.

	Shaw	Harrison	Russel Mitchel	Wheeler	Wardbeck	Santos	Wilson	Fox	Gesamt
Logs (Gesamt)	1.091.098	1.506.837	766.005	1.864.572	1.116.292	1.005.113	2.308.070	1.357.655	11.015.642
Flows (Gesamt)	730.529	620.189	509.734	809.688	575.683	439.897	966.172	501.916	5.153.808
Benign	648.550	557.877	443.611	733.983	540.534	407.244	893.325	446.786	4.671.910
Exfiltration	81.194	47.044	33.441	38.355	30.153	25.696	42.439	29.337	327.659
Reconnaissance	171	12.801	32.527	37.266	3.059	5.580	28.123	25.350	144.877
Credential Access	579	2.363	80	4	1.818	1.301	2.217	400	8.762
Discovery	14	21	54	30	78	55	8	26	286
Cmd & Ctrl	16	44	14	38	30	14	33	13	202
Initial Access	3	31	3	8	7	2	13	2	69
Persistence	2	2	2	4	2	3	14	2	31
Defense Evasion	-	2	2	-	2	2	-	-	8
Privilege Escalation	-	4	-	-	-	-	-	-	4

Tabelle 5.3.: Übersicht der extrahierten Flow und Log Datenpunkte pro Simulationsszenario

```
1    N: Jul 23 10:16:01 server bash[1234]: Executed command: whoami
2    T: <*> <*> <*>[<*>]: Executed command: <*>
```

Listing 5.1: Vergleich Informationswert Log-Nachricht und Log-Tempalte

Um den vollständigen Inhalt der Log-Nachrichten zu erhalten und gleichzeitig eine einheitliche Form für die Weiterverarbeitung zu schaffen, werden die Nachrichten mithilfe regexbasierter Regeln analysiert. Für jede im AIT-NDS-Datensatz enthaltene Logquelle wird dazu ein spezifischer Regex-Parser entwickelt, der Zeitstempeln und Nachrichteninhalten extrahiert. Die so aufbereiteten Log-Nachrichten werden für jedes Szenario in einem Pandas DataFrame mit den Spalten Host, Zeitstempel und Nachricht gespeichert. Auf diese Weise wird unter anderem die zusätzliche Komplexität ML-basierter Parser vermieden sowie Fehler, die wiederum Rauschen erzeugen könnten, reduziert.

Wie in Tabelle 5.3 zu sehen ist, wurden insgesamt 11,02 Mio. Log-Nachrichten aus dem AIT-LDS extrahiert. Obwohl sie im AIT-LDS enthalten sind, wurden Nachrichten von externen Anwendern und des Angreifers nicht berücksichtigt werden, da sie einem realistischen IDS ebenfalls nicht zur Anomalieerkennung zur Verfügung stehen würden.

5.2.4. Vorfilterung der Daten

Die ursprüngliche Datenmenge ist mit 5,15 Mio. Flows und 11,02 Mio. Logs sehr groß. Dies führt nicht nur zu einem hohen Zeitaufwand beim Training der Modelle, sondern auch bei der Vektorisierung mit modernen Embeddingmodellen. Aus zeitlichen Gründen wurde daher beschlossen, die Datenmenge vor der Log-Vektorisierung weiter einzuschränken.

Da in dieser Arbeit Flow-Log-Kombinationen untersucht werden, werden zunächst alle Flows aus der Datenmenge entfernt, welche nicht in Relation zu einer Log-Nachricht stehen. Diese sind für die folgenden Untersuchungen nicht relevant, da sich die Klassifikation dieser Flows nicht verändert. In Relation bedeutet, wie in Abschnitt 4.1 beschrieben, dass einer der beiden Hosts während der Flow-Dauer einen Log-Eintrag erzeugt. Dies Einschränkung reduziert die Datenmenge leicht auf 3,51 Mio. Flows, während die Loganzahl unverändert bleibt. Daher werden weitere Einschränkungen vorgenommen.

Wie in Abschnitt 5.2.2 bereits angemerkt, sind die extrahierten Daten stark unausgeglichen, was sich negativ auf das Modelltraining auswirkt. Eine Reduktion der anomalen Flows und Logs ist ohnehin nicht erwünscht, da diese nur selten auftreten. Um dem entgegenzuwirken, werden pro Szenario zufällig normale Flows entfernt, bis innerhalb des Szenarios die normalen Flows 60% ausmachen.

Somit weisen die Szenarien anschließend eine ähnliche Klassenverteilung auf, was im späteren Cross-Validation-Verfahren von Vorteil ist. Da, wie im Kapitel 4.3 beschrieben, die Splits anhand der Szenarien erstellt werden, ermöglicht die ähnliche Verteilung eine konsistente Bewertung der Modelle, denn in jeder Konstellation sind die Trainings-, Test- und Validierungssets gleich verteilt.

Zusätzlich stehen mit 975,3 Tausend Flows weiterhin ausreichend Daten für das Training und die Modellbewertung zur Verfügung.

Durch das vorläufige Undersampling bildet der Datensatz keine realistische Klassenverteilung mehr ab. Ziel dieser Arbeit ist jedoch nicht die Bewertung der Klassifikation im realen Einsatz, sondern die Prüfung, ob die Fusion generell die Erkennung von Anomalien unterstützt. Daher stellt diese Einschränkung kein Problem dar.

Abschließend werden alle Log-Nachrichten, welche nicht mehr in Relation zu einem Flow stehen, aus dem Datensatz entfernt. Dadurch besteht der Datensatz abschließend aus 975 Tsd. Flows und 7,71 Mio. Logs. Die genaue Klassenverteilung ist in Tabelle 5.4 dargestellt. Das Verfahren zur Flow-Log-Korrelation wird in Abschnitt 5.2.6 genauer erläutert.

	Shaw	**Harrison**	**Russel Mitchel**	**Wheeler**	**Wardbeck**	**Santos**	**Wilson**	**Fox**	**Gesamt**
Logs (Gesamt)	9.425	1.220.558	633.979	1.472.457	797.758	744.115	1.775.260	1.056.459	7.710.011
Flows (Gesamt)	574	152.875	157.772	188.540	86.800	80.753	171.315	136.680	975.309
Benign	343	91.725	94.663	113.124	52.080	48.453	102.789	82.008	585.185
Exfiltration	-	46.987	33.385	38.291	30.104	25.646	42.402	29.297	246.112
Reconnaissance	138	12.253	29.619	37.048	3.039	5.505	24.083	25.121	136.806
Credential Access	66	1.822	30	4	1.462	1.076	1.974	212	6.646
Discovery	14	21	54	30	78	55	8	26	286
Cmd & Ctrl	8	34	14	32	26	14	32	12	172
Initial Access	1	25	3	7	7	2	13	2	60
Persistence	2	2	2	4	2	2	14	2	30
Defense Evasion	2	2	2	-	2	-	-	-	8
Privilege Escalation	-	4	-	-	-	-	-	-	4

Tabelle 5.4.: Übersicht der gefilterten Flow und Log Datenpunkte pro Simulationsszenario

5.2.5. Vektorisierung von Log-Nachrichten

Wie in Abschnitt 2.2 erläutert, müssen Textdaten zunächst in eine Vektor-Repräsentation überführt werden. In dieser Arbeit werden dafür sowohl ein Bag-of-Words-Ansatz mit TF-IDF-Gewichtung als auch ein transformerbasiertes Modell zur Generierung kontextualisierter Embeddings untersucht. Dies ermöglicht eine vergleichende Analyse, um zu prüfen, ob moderne Embeddings Vorteile bei der Verarbeitung variabler Log-Nachrichten bieten.

Es wird erwartet, dass die subwortbasierte Tokenisierung, wie sie in Kombination mit Transformermodellen eingesetzt wird, eine verbesserte Handhabung unbekannter oder seltener Tokens ermöglichen (siehe Abschnitt 2.2.1). Darüber hinaus sollten kontextualisierte Embeddings die semantischen Informationen der Logs präziser erfassen als reine Bag-Of-Words-Vektoren. Dieser zusätzliche Informationsgehalt könnte zu einer verbesserten Erkennung von Anomalien beitragen.

Bag-Of-Words

Für die Umsetzung des BoW-Ansatzes wird die TfidfVectorizer-Implementierung aus Scikit-Learn (Version 1.7.2) [86] eingesetzt, wobei jede einzelne Log-Nachricht als eigenes Dokument behandelt wurde. Log-Nachrichten unterscheiden sich deutlich von natürlicher Sprache, da sie fragmentarisch sind und viele technische Begriffe sowie dynamische Werte wie IP-Adressen oder Pfade enthalten. Bei Bag-of-Words-Verfahren führt dies zu einem sehr großen, hochdimensionalen und spärlich besetzten Vektorraum. Um dem entgegenzuwirken, wird das Vokabular auf die Top 1024 häufigsten Tokens reduziert, sodass die Dimensionalität der erzeugten Feature-Vektoren mit der, der semantischen Embeddings übereinstimmt. Dadurch wird eine Vergleichbarkeit ermöglicht, die zeigt, wie viele Informationen aus den Lognachrichten durch die verschiedenen Repräsentationen für die Klassifikation transportiert werden.

Zusätzlich wurden englische Stoppwörter entfernt und nur Token mit mindestens zwei Buchstaben berücksichtigt, wobei Leer- und Sonderzeichen als Trennzeichen dienten. Dadurch sollte verhindert werden, dass sehr kurze Zeichenfolgen oder häufig vorkommende Wörter, die keinen inhaltlichen Mehrwert für die Analyse liefern, das Vokabular vergrößern. Zum selben Zweck wurden alle Wörter einheitlich klein geschrieben um sie nicht in doppelter Form im Vokabular zu haben.

Um das Out-of-Vocabulary-Problem korrekt abzubilden, wurden für alle Log-Nachrichten Repräsentationen erzeugt, die sich an den im Trainingsdatensatz enthaltenen Begriffen orientieren. Wie in der Trainingspipeline beschrieben, können die betrachteten Szenarien sowohl im Trainings- als auch im Testdatensatz vorkommen und sind damit Bestandteil der Cross-Validierung. Für die Vorbereitung der TF-IDF-basierten Bag-of-Words-Repräsentationen erfolgte eine feste Zuordnung der Szenarien. Szenario Wardbeck und Fox wurden zufällig dem Validierungsset zugewiesen, während die übrigen Szenarien dem Trainingsset zugeordnet wurden. Darauf basierend wurden drei Cross-Validation-Folds definiert ([Shaw, Harrison], [Russell-Mitchell, Wheeler], [Santos, Wilson] als Validierungssets), wobei jeweils die verbleibenden Szenarien als Trainingsdaten dienten. Eine vollständig zufällige Aufteilung hätte zwar eine robustere Schätzung der Generalisierungsleistung ermöglicht, jedoch den Aufwand für die Vorbereitung der Bag-of-Words-Darstellungen deutlich erhöht. Entsprechend wurden für jede Log-Nachricht im Trainingsset vier und für Nachrichten im Testset eine Repräsentation erzeugt.

```
1    N: 2025-09-21 12:01:45 INFO: User logged in from 192.168.1.5 session ID abc123xyz.
2    T: ['info', 'user', 'logged', 'session', 'id', 'abc123xyz']
```

Listing 5.2: Beispiel Log-Nachricht vor und nach Word-Tokenisierung

Listing 5.2 zeigt beispielhaft eine Log-Nachricht vor und nach dem Wort-Tokenisierungsschritt. Listing 5.3 zeigt hingegen die Byte-Pair-Tokenisierung des im Qwen3-Modell eingesetzten Tokenizers. Im Vergleich zur Word-Tokenisierung werden seltene Worte wie die Session-ID 'abc123xyz' in Subwort-Einheiten zerlegt. Bei einem klassischen Bag-of-Words-Verfahren würde ein solches seltenes Wort wahrscheinlich nicht berücksichtigt, wenn es nicht in den Trainingsdaten enthalten ist und somit im Feature-Vektor nicht auftaucht.

```
1   N: 2025-09-21 12:01:45 INFO: User logged in from 192.168.1.5 session ID abc123xyz.
2   T: ['2', '0', '2', '5', '-', '0', '9', '-', '2', '1', 'Ġ', '1', '2', ':', '0', '1',
        ':', '4', '5', 'ĠINFO', ':', 'ĠUser', 'Ġlogged', 'Ġin', 'Ġfrom', 'Ġ', '1',
       '9', '2', '.', '1', '6', '8', '.', '1', '.', '5', 'Ġsession', 'ĠID', 'Ġabc',
       '1', '2', '3', 'xyz', '.']
```

Listing 5.3: Beispiel Log-Nachricht vor und nach Qwen3 Byte-Pair-Tokenisierung

Embeddings

Transformerbasierte Modelle sind in der Lage, kontextsensitive und dichte Embeddings zu erzeugen. Da es sich um neuronale Netze handelt, können sie entweder Ende-zu-Ende gemeinsam mit dem Klassifikationsmodell durch Backpropagation trainiert oder als eigenständige Text-Encoder genutzt werden.

Das Ziel dieser Arbeit besteht nicht in der Entwicklung eines neuen Encoder-Modells für Log-Nachrichten. Dies wäre mit erheblichem Aufwand in Datenaufbereitung, Architekturdesign und Training verbunden. Um Log-Nachrichten als Embeddings abzubilden wird stattdessen das vortrainierte Modell Qwen3-Embedding-0.6B [87] ohne Finetuning als Encoder eingesetzt. Da das Modell ursprünglich auf natürlicher Sprache trainiert wurde, soll der Vergleich mit Bag-of-Words-Vektoren zunächst Aufschluss über die relative Qualität der erzeugten Repräsentationen geben.

Qwen3-Embedding-0.6B wurde ausgewählt, da es trotz seiner Modellgröße und somit vergleichsweise geringen Inferenzgeschwindigkeit robuste Embeddings für die unterschiedlichen Klassifikationsbenchmarks im Massive Multilingual Text Embedding Benchmark [88] erzielt. Darüber hinaus ist das Modell mit der Python-Bibliothek SentenceTransformers (Version 5.1.1) [89] kompatibel, welche für die Inferenz eingesetzt wird.

5.2.6. Flow-Log Korrelation

Wie in der Problemdefinition in Abschnitt 4.1 erläutert, ist für die gemeinsame Klassifikation eine Korrelation der extrahierten Log-Nachrichten und Biflows erforderlich. Das Verfahren orientiert sich am in Liu et al. [62] beschriebenen Alignment-Ansatz zwischen Logs und Flows.

Zur Implementierung wurde der gesamte Datensatz in eine SQLite-Datenbank überführt, um eine effiziente Verknüpfung der Flows und Logs zu ermöglichen. Die Korrelation erfolgte über SQL-Joins, wobei Zeitstempel und Hostzuordnungen als Verknüpfungsbedingungen dienten. Für jeden Flow wurden alle Log-Nachrichten ermittelt, deren Zeitstempel innerhalb der Start- und Endzeit des

Flows

src_ip	dst_ip	src_port	dst_port	protocol	timestamp	duration	...
192.168.96.4	192.168.127.254	60198	53	17	1642580167.65389	0.041935	...

↔

Logs

system	timestamp	message	embedding	source	...
inet-firewall	1642580167	Starting Daily apt upgrade and clean activities...	[0.00930242 0.0084364 -0.01182509 ...]	syslogs	...
inet-dns	1642580167	query[A] updates.owncloud.com from 192.168.96.4	[-0.01710655 -0.02168673 -0.01417108 ...]	dnsmasq	...
inet-firewall	1642580168	Started Daily apt upgrade and clean activities.	[0.02205768 0.0164262 -0.01040217 ...]	syslogs	...
inet-dns	1642580168	reply updates.owncloud.com is <CNAME>	[-0.02993712 0.01211833 -0.01081694 ...]	dnsmasq	...

Abbildung 5.1.: Beispielhafte Zuordnung eines Flows zu den entsprechenden Log-Nachrichten.

Flows liegen und die entweder vom Client- oder Server-Host stammen. Um die Speicherbelastung zu begrenzen, wurde die Korrelation batchweise durchgeführt. Dazu wurden die Parameter Limit und Offset in Kombination mit dem Szenario eingesetzt, um nur Teilmengen der Flows in den Hauptspeicher zu laden und Out-of-Memory-Fehler zu vermeiden. Listing 5.4 zeigt exemplarisch den zur Erzeugung der Embeddingbasierten Flow-Log-Korrelation verwendeten SQL-Query.

```
1   SELECT
2   f.id AS flow_id,
3   l.embedding AS embedding
4   FROM (
5     SELECT id, scenario, start, end, client_host, server_host
6     FROM flows
7     WHERE scenario = ?
8     ORDER BY id ASC
9     LIMIT ? OFFSET ?
10  ) f
11  JOIN logs l
12  ON l.scenario = f.scenario
13  AND l.timestamp BETWEEN f.start AND f.end
14  AND (l.system = f.client_host OR l.system = f.server_host)
15  ORDER BY f.id, l.timestamp;
```

Listing 5.4: SQL-Korrelation zwischen Flows und Logs

Wie beispielhaft in Listing 5.5 zu sehen, wurden zur Verbesserung der Abfragegeschwindigkeit Indizes angelegt, um Suchoperationen innerhalb der großen Log-Tabelle zu optimieren.

```
1   CREATE INDEX IF NOT EXISTS idx_logs_scenario_system_timestamp
2   ON logs(scenario, system, timestamp);
```

Listing 5.5: Beispiel eines zur Beschleunigung der Joins verwendeten Indexes

Da Log-Nachrichten punktuell und Flows über eine Zeitspanne erfasst werden, wurde der Beginn eines Flows aufgerundet und das Ende abgerundet, um den vollständigen Kontext zu erfassen, ohne relevante Ereignisse auszuschließen. Nach der Korrelation werden die Log-Nachrichten eines Flows in zeitlich aufsteigender Reihenfolge sortiert, um die in Abschnitt 4.1 definierte Sequenz $l_{h,h'}$ zu erzeugen. Diese Sequenz bildet die Grundlage für die Repräsentation der Flows durch Embedding- oder TF-IDF-basierte Vektoren:

$$l = (l_1, l_2, \ldots, l_n), \quad n \in \mathbb{N}, \quad l_j \in \mathbb{R}^{1024}. \tag{5.1}$$

5.3. Einsatz von SMOTE Oversampling zur Reduktion der Klassenimbalance

Zur Untersuchung des Einflusses der Klassenimbalance im vorhandenen Datensatz auf das Training der Klassifikationsmodelle wird neben dem im Abschnitt 5.2 beschriebenen Undersampling auch der Einsatz von Oversampling betrachtet. Hierfür kommt das SMOTE Verfahren zum Einsatz (siehe Abschnitt 2.3). Durch SMOTE werden neue, synthetische Datenpunkte für die Minderheitsklassen generiert und den Trainingsdaten hinzugefügt, wodurch ein ausgeglicheneres Klassenverhältnis entsteht.

Da SMOTE neue Stichproben durch Interpolation bestehender Datenpunkte erzeugt, wird die Methode nicht auf die binären Klassen Anomalie und Normal angewendet, sondern auf die jeweiligen Angriffsphasen-Subklassen. Dadurch wird vermieden, dass Datenpunkte entstehen, welche die Merkmale verschiedener Angriffstypen kombinieren und somit potenziell Rauschen im Datensatz verursachen.

Darüber hinaus wird SMOTE für jedes Szenario des Trainingsdatensatzes separat angewendet. Obwohl in diesem Kontext kein Risiko für Information Leakage besteht, sollen synthetische Stichproben nicht aus einer Kombination verschiedener Szenarien entstehen, um zusätzliches Rauschen zu vermeiden.

Die Zielgröße der jeweiligen Klasse t_{target} wird bestimmt durch:

$$t_{\text{target}} = \min\left(\max\left(n_{\text{attack}} \cdot f_{\max},\, t_{\min}\right),\, n_{\text{normal}} \cdot 0.5\right) \tag{5.2}$$

mit $f_{\max} = 10$ und $t_{\min} = 5000$, wobei n_{class} die Anzahl der Stichproben der jeweiligen Minderheitsklasse und n_{normal} die Anzahl der normalen Samples des jeweiligen Szenarios beschreibt.

Damit wird sichergestellt, dass keine der Minderheitsklassen mehr als die Hälfte der Anzahl der Benign-Samples umfasst und so das Trainingsset wieder dominiert. Gleichzeitig erreichen absolute Minderheitsklassen, wie beispielsweise Persistence oder Command & Control, so mindestens 5000 Stichproben und werden leicht überproportional im Vergleich zu den anderen Klassen erweitert, was zu einem ausbalancierteren Trainingsset führen soll.

Es wird erwartet, dass das Resampling insbesondere die Erkennungsleistung für Minderheitsklassen verbessert, da diese durch die erhöhte Anzahl an Trainingsbeispielen stärker in den Optimierungsprozess einfließen. Das Modell wird dadurch gezwungen, die charakteristischen Muster dieser Klassen besser zu erfassen. Gleichzeitig besteht jedoch durch die Generierung synthetischer Datenpunkte ein erhöhtes Risiko für Overfitting, da einige erzeugte Stichproben bestehenden Punkten sehr ähnlich sein oder nicht der realen Datenverteilung entsprechen können. Dies kann zu zusätzlichem Rauschen führen und die Trainingsstabilität beeinträchtigen [38].

Wie in der Übersicht zur Trainingspipeline (Abbildung 4.3) dargestellt, wird SMOTE ausschließlich beim Retraining angewendet. Die Hyperparameter der Modelle werden hierbei nicht erneut optimiert, stattdessen werden die zuvor ermittelten Parameter beibehalten, um eine gute Vergleichbarkeit der Modelle zu ermöglichen.

5.4. Implementierungsdetails und Trainingsumgebung

Die Implementierung der Datenextraktion und des Modelltrainings ist zur einfacheren Handhabung in zwei klar voneinander getrennte Projekte aufgeteilt. Im ersten Schritt erfolgt die Datenextraktion nach Abschnitt 5.2 in Python (Version 3.10.14). Für die Flow-Extraktion kommt das Python CICFlowMeter (Version 0.3.0) [80] zum Einsatz. Für die Verarbeitung der Logdaten in TF-IDF Vektoren wird scikit-learn (Version 1.7.2) [86] und sentence-transformers (Version 5.1.1) [89] für die Generierung von Embeddings eingesetzt. Zur Vorverarbeitung der Texte und insbesondere zur Entfernung irrelevanter Terme wird der Stopwortkorpus des Natural Language Toolkit (NLTK) (Version 3.9.1) [90] verwendet.

Im Rahmen der Flow-Extraktion wird CICFlowMeter gezielt erweitert und angepasst, um die in Abschnitt 5.2.1 definierte Flow-Definition exakt abzubilden. Diese Modifikationen sind notwendig, da die ursprüngliche Implementierung aufgrund von mehreren Bugs als frühzeitig oder nicht beendet erkennt, was zu fehlenden beziehungsweise falschen Flow-Instanzen führt.

Der zweite Implementierungsschritt umfasst das Modelltraining sowie die Hyperparameteroptimierung. Klassische ML-Modelle werden ebenfalls mit scikit-learn (Version 1.7.2) [86] realisiert, während für neuronale Modelle PyTorch (Version 2.8.0) [91] als Standard Deep Learning Bibliothek eingesetzt wird.

Zur Untersuchung des Einflusses von SMOTE auf die Modelle wird die Implementierung der Bibliothek imbalanced-learn (Version 0.14.0) [92] verwendet.

Beide Prozesse werden auf dem High-Performance-Data-Analysis-Cluster (HPDA) des Deutschen Zentrums für Luft- und Raumfahrt durchgeführt. Die Datenextraktion erfolgt auf einem System mit einem Intel Xeon Platinum 8260 und 1536 GiB Arbeitsspeicher, während für das Modelltraining ein System mit einem AMD EPYC 7402, 512 GiB RAM sowie einer NVIDIA Tesla A100 GPU verwendet wird.

5.5. Probleme beim Training der Expermiente

Wie in Kapitel 4.3 beschrieben, wird beim Training der MLPs ein Szenario für die Validierung zurückgehalten, um den Trainingsprozess zu überwachen. Nach dem Training des ersten MLP-Modells zeigte die Analyse der Lernkurven für die Flow-Statistiken und Embeddings (siehe Abbildung 5.2), dass die Modelle trotz optimierter Hyperparameter vermutlich zur Überanpassung neigen und das Training insbesondere beim Embedding-Modell sprunghaft verläuft und nicht konvergiert.

Das sprunghafte Verhalten des Embedding-Modells lässt sich auf die gewählten Hyperparameter zurückführen. Die Trainingspipeline hat eine Parameterkombination mit einer hohen Learning-Rate von 0,01 gewählt, was wahrscheinlich dazu geführt hat, dass das Modell beim Training zu stark in Richtung der Gradienten optimiert wurde. Dies führte zu einer Instabilität im Trainingsverlauf.

Für die Überanpassungen kommen mehrere Ursachen infrage. Zum einen könnten die ermittelten Hyperparameter zu einem zu komplexen Modell geführt haben, das nicht nur die zugrunde liegenden

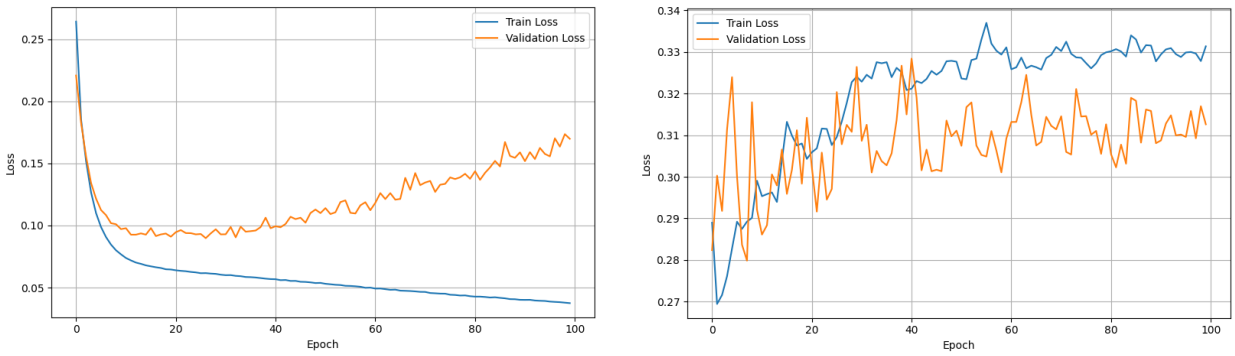

Abbildung 5.2.: Trainingskurven der unimodalen MLP-Modelle: links für Flow-Statistiken, rechts für Embeddings

Zusammenhänge, sondern auch das Rauschen in den Trainingsdaten lernt. Zum anderen könnte das Modell zu lange trainiert worden sein oder es stehen insgesamt zu wenige Trainingsdaten zur Verfügung, um eine gute Generalisierung zu gewährleisten.

Um der Überkomplexität entgegenzuwirken und dabei nicht die Ausdrucksstärke des Modells zu beschränken, können verschiedene Regularisierungsmethoden eingesetzt werden. Eine dieser Methoden, Dropout, wurde bereits im Rahmen der Hyperparameteroptimierung berücksichtigt, wobei alle gefundenen Modellkonfigurationen einen Dropout-Wert größer null verwendeten.

Um zunächst den Einfluss einer hohen Lernrate und der L2-Regularisierung zu untersuchen, wurden beide Modelle erneut mit denselben Hyperparametern trainiert. Das Flow-Statistik-MLP mit aktivierter L2-Regularisierung und das Embedding-MLP mit reduzierter Lernrate und L2-Regularisierung. Die dabei entstandenen Trainingskurven sind in Abbildung 5.3 zusehen.

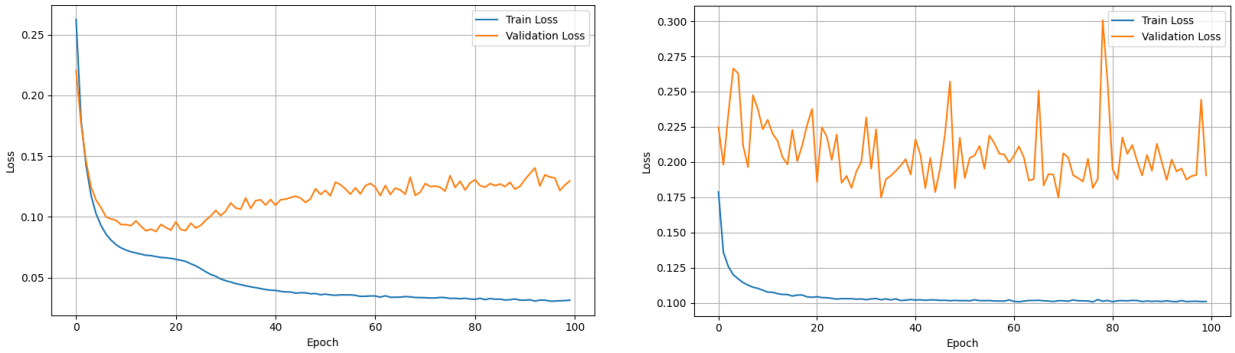

Abbildung 5.3.: Trainingskurven der unimodalen MLP-Modelle nach Anpassungen des Trainings: links für Flow-Statistiken, rechts für Embeddings

Die neuen Lernkurven zeigen, dass das Training für die Flow-Statistiken vermutlich weiterhin überangepasst ist, was darauf hindeutet, dass das Modell möglicherweise zu lange trainiert wurde. Um diesem Effekt entgegenzuwirken, wurde eine Early-Stopping-Strategie implementiert, bei der das Training abgebrochen wird, sobald sich der Validierungsfehler über 5 Epochen nicht weiter verringert.

Beim Embedding-MLP zeigt der Validierungsfehler nach wie vor ein sprunghaftes Verhalten, während der Fehler des Trainingssets konvergiert. Dies deutet darauf hin, dass das Modell potenziell zu komplex für die gegebenen Daten ist.

Um die Modelle weiter zu stabilisieren, wurde für alle MLP-Modelle die Hyperparametersuche erneut durchgeführt. Dabei wurden Early Stopping und L2-Regularisierung berücksichtigt, hohe Lernraten ausgeschlossen und zusätzlich Optionen für eine geringere Anzahl an Hidden Layers sowie schmalere Layer eingeführt. Die entfernten Hyperparameter sind in den Tabellen 4.1 und 4.2 rot markiert, die neu hinzugefügten hingegen Grün.

Die nachfolgenden Ergebnisse beziehen sich auf die nach der beschriebenen Anpassungen entstandenen Modelle.

6. Ergebnisse

In diesem Kapitel werden die Ergebnisse der in Kapitel 4 beschriebenen Experimente zur unimodalen und multimodalen Klassifikation von Netzwerk-Flows in Kombination mit Log-Nachrichten vorgestellt und analysiert. Ziel ist es, die Leistungsfähigkeit der entwickelten Modelle zu bewerten und den Einfluss unterschiedlicher Datenquellen auf die Klassifikationsgenauigkeit zu untersuchen.

Zunächst werden die Resultate der unimodalen Ansätze betrachtet, bei denen die einzelnen Datenmodalitäten isoliert zur Klassifikation genutzt werden. Ein besonderes Augenmerk liegt dabei auf dem Vergleich zwischen kontextualisierten Embeddings und Bag-of-Words-Vektoren bei der Angriffserkennung durch Analyse von Log-Nachrichtensequenzen. Anschließend werden die multimodalen Modelle ausgewertet, welche beide Informationsquellen kombinieren, um potenzielle Synergieeffekte zwischen den Modalitäten zu identifizieren. Abschließend werden Modelle analysiert, die mit durch SMOTE erweiterten Daten trainiert werden, um den Einfluss der veränderten Klassenbalance auf die Erkennungsleistung zu untersuchen.

6.1. Unimodal Klassifikation

Die Unimodalklassifikation wurde zunächst getrennt auf Basis der drei betrachteten Datenarten, Flow-Statistiken, TF-IDF und Embedding, durchgeführt, mit Random Forests und MLPs als Klassifikatoren.

Die Modelle, Trainingspipeline und die Evaluationsmetriken entsprechen dem in Abschnitt 4.2.1 und 4.3 beschriebenen Vorgehen. Als Evaluations-Simulationsszenario dienen die in Tabelle 5.4 beschrieben Szenarien Wardbeck und Fox. Die Ergebnisse aller Klassifikatoren sind in Tabelle 6.1 zusammengefasst.

Es ist zu erkennen, dass die Klassifikatoren auf Basis der Flow-Statistiken die besten Werte in allen untersuchten Metriken erreichen und insgesamt erstaunlich gut zwischen normalen und Angriffs-Flows unterscheiden können. Random Forest erzielt mit 98,62% die höchste Gesamtgenauigkeit sowie einen nahezu perfekten AUC-Wert, während das MLP mit 98,26% Genauigkeit leicht darunterliegt.

Die bessere Leistung des Random Forest bei den Flow-Statistiken ist erwartbar, da Random Forests auch in anderen Untersuchungen besonders gut mit tabellarischen Daten funktionieren [65].

Ebenfalls zeigt sich, dass die korrelierten Log-Sequenzdarstellungen erstaunlich gut darin sind Flows zu beschreiben. Mit ihnen erreicht das Beste Modell, das MLP auf Basis der Embedding-Repräsentation, eine Genauigkeit von 96,53%, was nur etwas geringer als die Flow-Statistik-Klassifikation ist. Auch die F1-Scores spiegeln ein ähnliches Verhältnis wider.

Diese Überlegenheit der Statistik-basierten Klassifikation ist zu erwarten, da die Flow-Daten die Eigenschaften der Netzwerkflüsse direkt und quantitativ erfassen und somit eine präzisere Beschreibung der zu klassifizierenden Einheiten liefern. Die reine Klassifikation anhand der Log-Daten schneidet, auch mit den gewichteten Bag-of-Words-Vektoren, jedoch erstaunlich gut ab, was zeigt,

dass selbst einfache textbasierte Features nützliche Informationen zur Anomalieerkennung enthalten.

Encoding	Klassifikator	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	AUC (%)
CICFlowMeter	Random Forest	**98,62**	**98,80**	**98,33**	**98,55**	**99,0**
CICFlowMeter	MLP	98,26	98,45	97,95	98,18	**99,0**
TF-IDF	Random Forest	93,75	93,18	94,20	93,57	96,0
TF-IDF	MLP	93,23	92,65	93,70	93,04	96,0
Embedding	Random Forest	92,38	91,85	92,48	92,12	96,0
Embedding	MLP	96,53	96,58	96,18	96,37	**99,0**

Tabelle 6.1.: Klassifikationsergebnisse der unimodalen Modelle mit verschiedenen Encodings und Klassifikatoren

Es ist jedoch zu erkennen, dass die gemittelten Embeddings mehr Informationen transportieren und in Kombination mit dem MLP basierten Klassifikator zu einer besseren Klassifikationsleistung führen.

Mit TF-IDF erreicht Random Forest eine Genauigkeit von 93,75%, während das MLP mit denselben TF-IDF-Vektoren geringfügig schlechter abschneidet. Im Gegensatz dazu kann das MLP in Kombination mit den Embeddings deutlich bessere Ergebnisse erzielen und erreicht eine Genauigkeit von 96,53%. Wie erwartet ist der Random Forest in Kombination mit den dichten Vektoren weniger effektiv und erzielt lediglich eine Genauigkeit von 92,38%.

Diese Ergebnisse deuten darauf hin, dass die Vorteile der kontextualisierten Embeddings bei der Erkennung von Angriffen effektiv genutzt werden können. Wie angenommen, ermöglichen sie eine feinere Trennung zwischen anomalen und normalen Sequenzen. Gleichzeitig scheinen die Log-Daten inhaltlich ausreichend strukturiert zu sein, sodass TF-IDF als einfachere Repräsentation bereits eine vergleichbare Erkennung ermöglichen.

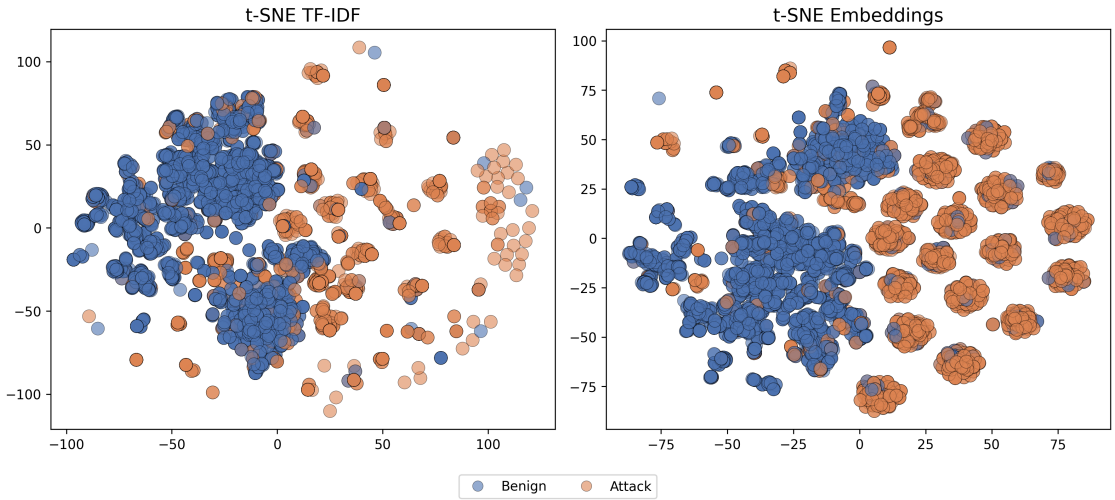

Abbildung 6.1.: t-SNE Plots der Log-Sequenz Vektordarstellungen. Links: semantische Embeddings. Rechts: häufigkeitsbasiert TF-IDF Vektoren

Die t-SNE-Projektion der Feature-Räume in Abbildung 6.1 verdeutlicht dies zusätzlich. Die TF-IDF-Vektoren bilden leicht überlappende, aber dennoch erkennbare Cluster von Anomalien und Normalfällen, während die Embedding-basierten Features ähnliche, jedoch klarer abgegrenzte Cluster mit geringerer Streuung aufweisen. In beiden Abbildungen ist jedoch keine klare Trennung zwischen Clustern normaler und anomaler Log-Sequenzen zu erkennen.

Abbildung 6.2 zeigt die Confusion Matrix des Flow-Statistik basierten Random Forest Klassifikators. Wie zu sehen ist, werden die meisten Flows korrekt klassifiziert. 133.763 normale Flows werden korrekt als True Negatives erkannt und 86.623 Angriffs-Flows korrekt als Angriffe (True Positives) identifiziert.

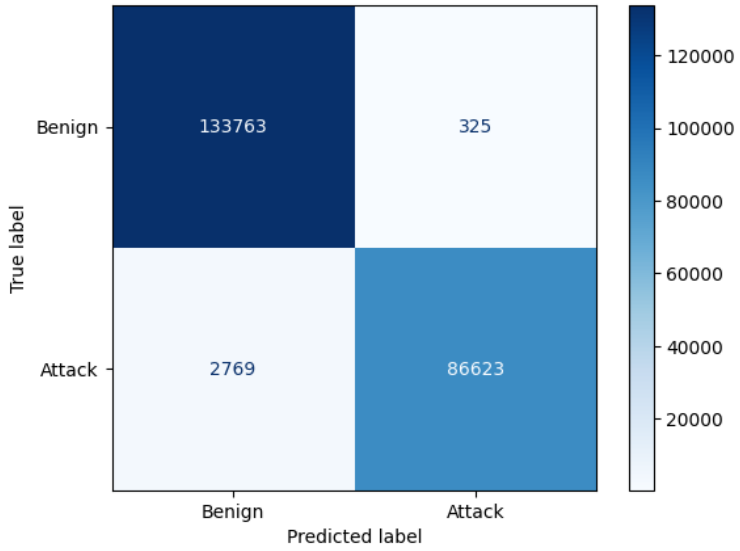

Abbildung 6.2.: Confusion Matrix des besten unimodalen Random Forest Klassifikationsmodells trainiert mit Flow-Features

Demgegenüber stehen 2.769 False Negatives, bei denen anormale Flows irrtümlich als normal erkannt werden und somit unentdeckt bleiben.

Darüber hinaus wurden 325 False Positives erzeugt, also Fälle, in denen legitime Flows fälschlicherweise als anormal erkannt werden. Solche Fehlalarme sind kritisch, da sie unnötige Sicherheitswarnungen erzeugen, die die Aufmerksamkeit des Sicherheitspersonals binden und bei zu häufigem Auftreten zu Alarmmüdigkeit führt, was die zeitnahe Identifikation echter Bedrohungen erschwert.

Insgesamt zeigen die Ergebnisse der unimodalen Klassifikation, dass die Flow-Statistik Repräsentation die Klassifikation von Angriffs-Flows am zuverlässigsten ermöglicht, während textbasierte Features, unabhängig von der Repräsentation als Bag-Of-Words oder Embeddings, ergänzende, aber schwächere Signale liefern. Für eine detailliertere Bewertung wird im Folgenden die Genauigkeit der Modelle pro Angriffsklasse betrachtet. Dadurch lässt sich erkennen, welche Angriffsarten von den verschienden Modellen besonders zuverlässig erkannt werden und bei welchen Klassen die Fehlklassifikationen stärker auftreten.

Genauigkeit pro Angriffsphase

Betrachtet man die in Tabelle 6.2 dargestellte Klassifikationsgenauigkeit der Modelle pro APT-Angriffsphase, zeigen sich ebenfalls deutliche Unterschiede zwischen den Datenquellen, sowie den Klassifikatoren.

Wie nach dem Gesamtergebnis zu erwarten, erzielt das Flow-Statistik MLP Modell in nahezu allen Angriffsklassen die besten Ergebnisse. Insbesondere bei den häufig vorkommenden normalen Flows erreichen sowohl Random Forest als auch das MLP extrem hohe Genauigkeiten von über 99%. Auch bei der Erkennung von Exfiltration oder Reconnaissance Flows erzielen die Flow-Statistik Modelle mit bis zu 99,99% bzw. 96,25% ebenfalls bessere Ergebnisse.

Auffällig ist jedoch, dass bei selten vertretenen Angriffsklassen, insbesondere Command & Control sowie Initial Access, die Genauigkeit deutlich abfällt. Hier erreicht das Random Forest mit den Flow-Statistiken nur rund 52,6% bzw. 22,2% Genauigkeit, während das MLP im Fall von Initial Access überhaupt keine Flows als Anomalie erkennt. Diese Diskrepanz ist vermutlich auf das unausgeglichene Klassenverhältnis in den Trainingsdaten zurückzuführen. In Abschnitt 6.3 wird dies weiter untersucht.

	CICFlowMeter		**TF-IDF**		**Embedding**		
Angriffsphase	**RF**	**MLP**	**RF**	**MLP**	**RF**	**MLP**	**Support**
Benign	**99,76**	99,53	91,97	91,34	91,98	97,94	134,088
Exfiltration	**99,99**	99,42	98,75	98,12	97,26	99,39	59,401
Reconnaissance	96,25	95,77	97,38	**97,53**	89,58	89,66	28,160
Credential Access	1,25	1,19	**1,37**	**1,37**	1,19	1,31	1,674
Discovery	**75,0**	**75,0**	75,0	73,08	73,08	73,08	104
Command & Control	**52,63**	39,47	34,21	31,58	31,58	31,58	38
Initial Access	**22,22**	0,0	0,0	0,0	0,0	0,0	9
Persistence	75,0	0,0	**100,0**	**100,0**	50,0	**100,0**	4
Defense Evasion	**100,0**	**100,0**	**100,0**	**100,0**	**100,0**	**100,0**	2

Tabelle 6.2.: Pro Angriffsphasen Accuracy der unimodalen Modelle mit verschiedenen Encodings und Klassifikatoren

Ebenso lässt sich erkennen, weshalb das MLP in Kombination mit der Embedding Repräsentation der Log-Sequenzen eine höhere Gesamtgenauigkeit erzielt als die TF-IDF-basierten Modelle. Offenbar ermöglichen die Embeddings eine präzisere Erkennung normaler Flows, die in den Daten stark überrepräsentiert sind. Dadurch verbessert sich die Gesamtgenauigkeit des Modells, auch wenn die Bag-Of-Word-Vektoren, bei seltener auftretenden Angriffstypen wie Reconnaissance teilweise bessere Ergebnisse erzielen.

6.2. Multimodale Klassifikation

In diesem Kapitel werden die Ergebnisse der multimodalen Klassifikation vorgestellt, bei der die Informationen aus Flow-Statistiken und Log-Sequenz-Encodings miteinander kombiniert wurden.

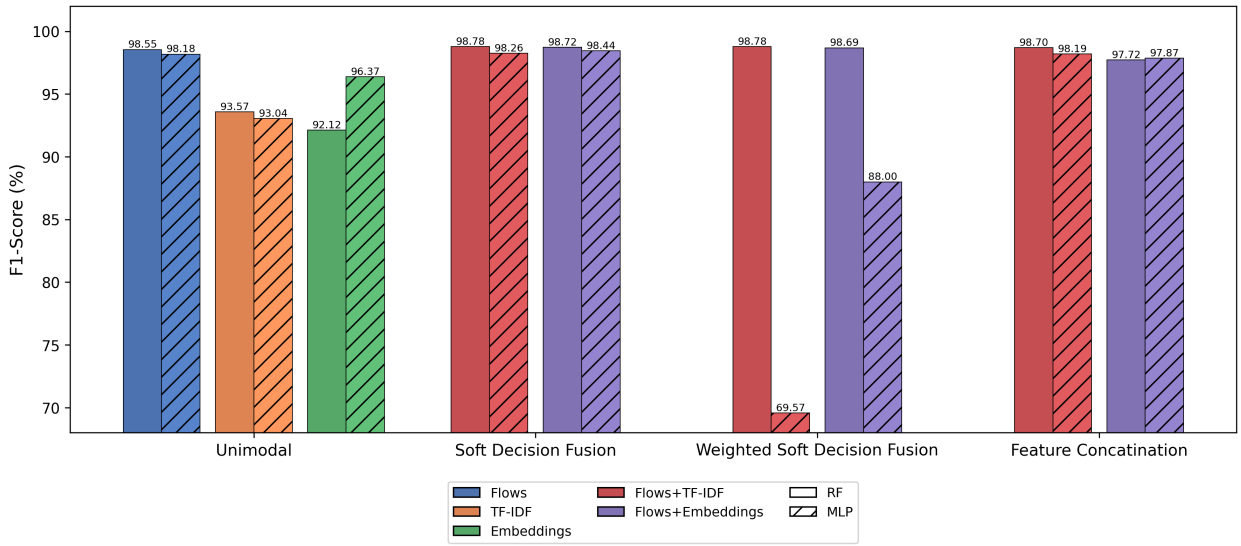

Abbildung 6.3.: Vergleich Makro F1-Scores der unimodalen und multimodalen Klassifikationsmodelle

Zunächst werden die Resultate der in Abschnitt 4.2.2 beschriebenen Late Fusion durch Soft-Decision-Fusion präsentiert, gefolgt von den Ergebnissen der Intermediate Fusion durch Feature-Konkatenation.

6.2.1. Soft Decision Fusion

Wie in 4.2.2 beschrieben, wurden für die gewichtete und ungewichtete Soft Decision Fusion jeweils die geschätzten Wahrscheinlichkeiten der besten Random-Forest- und MLP-Modelle kombiniert. Die entsprechenden Ergebnisse sind in Tabelle 6.3 dargestellt.

Abbildung 6.3 veranschaulicht zum Vergleich den Makro F1-Score der untersuchten Klassifikationsmodelle. Im Vergleich zum besten unimodalen Modell, dem Random Forest auf Basis der Flow-Statistiken, liefert die Kombination beider Random-Forest-Modelle (Flow-Statistiken und TF-IDF-Sequenzvektoren) leicht verbesserte Ergebnisse. Der Makro F1-Score steigt von 0,9855 auf 0,9878, während die Genauigkeit von 98,62% auf 98,83% zunimmt. Dieses Resultat ist nachvollziehbar, da bereits das Modell mit Flow-Statistiken die besten Einzelergebnisse erzielte und auch der Random Forest mit TF-IDF-Vektoren solide Leistungen zeigte.

Auch die MLP-Modelle profitierten grundsätzlich von der Decision Fusion und erzielten leicht bessere Ergebnisse. Allerdings führten die Vorteile der MLPs in Verbindung mit den Embedding-Log-Sequenz-Vektoren nicht zu einer weiteren Verbesserung der Gesamtleistung. Wahrscheinlich liegt dies daran, dass das MLP-Modell für die Flow-Statistiken vergleichsweise schwach abschnitt und diese, wie bereits vermutet, den größten Informationsgehalt transportieren.

Diese Vermutung wird zusätzlich durch die ermittelten Gewichtungsfaktoren ω verstärkt. Interessanterweise ergab die Hyperparametersuche aus Abschnitt 4.2.2 einen identischen Gewichtungsfaktor von 0.524 für alle Modelle. Das entspricht einer leicht stärkeren Gewichtung der Flow-Modelle.

Betrachtet man ausschließlich die Anomalieerkennungsmodelle, welche Flows und Embeddings kombinieren, zeigt sich ebenfalls, dass die MLP-Modelle schlechter abschneiden als die Random-Forest-Modelle. Auch dies lässt sich auf die bessere Leistungsfähigkeit der Random Forests auf Basis der Flow-Statistiken zurückführen.

Insgesamt konnten die MLP-Modelle mit der gewichteten Soft Decision Fusion keine überzeugenden Ergebnisse erzielen. In einigen Fällen verschlechterten sich ihre Leistungen sogar deutlich, wodurch sie für die weitere Betrachtung weniger relevant sind.

Encoding	**Klassifikator**	ω	**Accuracy** (%)	**Precision** (%)	**Recall** (%)	**F1-Score** (%)	**AUC** (%)
Flow + TF-IDF	Random Forest	0,5	**98,83**	**99,01**	**98,57**	**98,78**	**99,0**
Flow + TF-IDF	Random Forest	0,524	98,83	**99,01**	98,56	**98,78**	**99,0**
Flow + TF-IDF	MLP	0,5	98,33	98,43	98,09	98,26	**99,0**
Flow + TF-IDF	MLP	0,524	69,57	72,35	72,41	69,57	82,0
Flow + Embedding	Random Forest	0,5	**98,78**	**98,98**	**98,49**	**98,72**	**99,0**
Flow + Embedding	Random Forest	0,524	98,75	98,95	98,46	98,69	**99,0**
Flow + Embedding	MLP	0,5	98,51	98,63	98,27	98,44	**99,0**
Flow + Embedding	MLP	0,524	88,10	88,21	89,79	88,00	**99,0**

Tabelle 6.3.: Übersicht der multimodalen Klassifikationsleistungen basierend auf Soft Decision Fusion und Weighted Soft Decision Fusion der jeweils besten unimodalen Modelle

Betrachtet man die Confusion Matrix des besten Modells (Random Forest mit Flow-Statistiken und TF-IDF-Vektoren) in Abbildung 6.4 so sieht man die Verbesserung. Es wurden nur noch 164 normale Flows fälschlicherweise als Angriff klassifiziert. Dies ist eine deutliche Reduktion im Vergleich zu den 325 Falsch Positiv erkannten Flows des besten unimodalen Modells. Dennoch bleiben weiterhin 2442 Angriffsflows als False Negatives unerkannt.

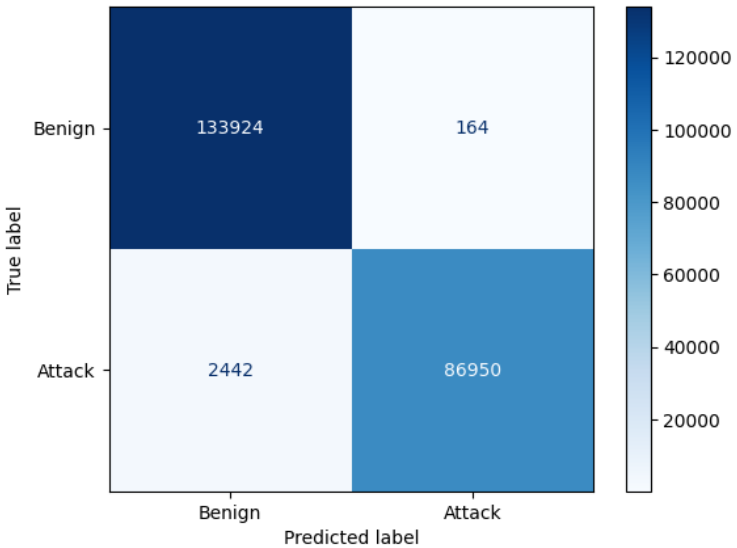

Abbildung 6.4.: Confusion Matrix des besten Soft Decision Fusion Klassifikationsmodells (Random Forest mit Flow-Statistiken und TF-IDF).

Insgesamt führte die Soft Decision Fusion zu einer stabileren Erkennung von Anomalien. Die Random-Forest-basierten Klassifikatoren erzielten dabei die besten Ergebnisse, während auch die MLP-Modelle in gewissem Maße von der Fusion profitierten. Die gewichtete Soft Decision Fusion brachte jedoch nur geringe Verbesserungen und führte bei den MLPs teilweise sogar zu einer Verschlechterung der Leistung.

Genauigkeit pro Angriffsphase

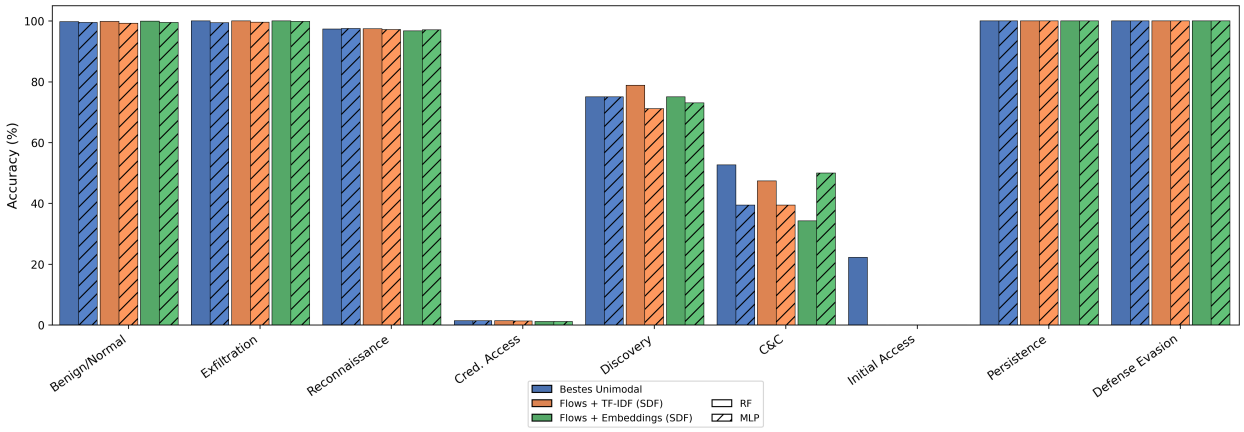

Abbildung 6.5.: Vergleich Erkennungsgenauigkeit zwischen besten unimodalen Ergebnissen und ungewichteter Soft-Decision-Fusion

Da, wie bereits erwähnt, die gewichtete Soft-Decision-Fusion nur schlechtere oder gleichwertige Ergebnisse erzielte, wird im Folgenden ausschließlich die Erkennungsgenauigkeit der ungewichteten Variante näher betrachtet.

Tabelle 6.4 gibt einen Überblick über die Erkennungsgenauigkeit der Soft-Decision-Fusion-Modelle pro APT-Angriffsklasse. Abbildung 6.5 stellt ergänzend die Genauigkeit über die APT-Phasen im Vergleich zu den besten unimodalen Ergebnissen dar.

	TF-IDF		**Embedding**		
Angriffsphase	**RF**	**MLP**	**RF**	**MLP**	**Support**
Benign	99,88	99,28	**99,94**	99,48	134.088
Exfiltration	**99,99**	99,59	99,98	99,82	59.401
Reconnaissance	**97,39**	97,15	96,71	97,12	28.160
Credential Access	**1,43**	1,31	1,19	1,19	1.674
Discovery	**78,85**	71,15	75,00	73,08	104
Command & Control	47,37	39,47	34,21	**50,00**	38
Initial Access	0,00	0,00	0,00	0,00	9
Persistence	**100,00**	**100,00**	**100,00**	**100,00**	4
Defense Evasion	**100,00**	**100,00**	**100,00**	**100,00**	2

Tabelle 6.4.: Pro Angriffsphasen Accuracy (%) der multimodalen Modelle basierend auf Soft Decision Fusion (ohne Gewichtung)

Zu sehen ist, dass die Genauigkeit beim Erkennen der Kategorien Persistence und Defense Evasion konstant bleibt und alle Modelle diese Flows zuverlässig als Angriffe identifizieren.

Auch bei den Mehrheitsklassen Benign, Exfiltration, Reconnaissance sowie Credential Access verhalten sich die kombinierten Modelle ähnlich wie die besten unimodalen Varianten. Eine leichte Verbesserung zeigt sich jedoch bei der Erkennung normaler Flows. Die Genauigkeit des Random-Forest-Modells mit Embeddings verbesserte sich von sich von 99,76% auf 99,94%.

Ebenso sind bei Credential Access und Discovery kleinere Verbesserungen zu beobachten, was darauf hinweist, dass beide Datenquellen komplementäre Informationen beisteuern und gemeinsam Flows erkannt werden, welche vorher als Normal galten.

Für die Klassen Command and Control und Initial Access ist hingegen eine leichte Verschlechterung festzustellen. Dies könnte darauf hindeuten, dass die beiden kombinierten Modelle in diesen Fällen zu unterschiedlichen Einschätzungen gelangen und ihre Fusion somit kein konsistentes Ergebnis hervorbringt.

Die Kategorien Persistence und Defense Evasion zeigen sich dagegen stabiler als zuvor. Trotz der geringen Anzahl an Testbeispielen konnten hier alle Soft Decision Fusion Modelle diese mit 100% Genauigkeit korrekt klassifizieren.

6.2.2. Intermediate Feature Concatination Fusion

Die Ergebnisse der in Abschnitt 4.2.2 beschriebenen Intermediate-Fusion sind in Tabelle 6.5 dargestellt. Zum besseren Vergleich der Makro F1-Scores sind diese zusätzlich in Abbildung 6.3 visualisiert.

Deutlich erkennbar ist, dass sowohl die Random-Forest- als auch die MLP-basierten Klassifikatoren mit der Kombination aus Flow-Statistiken und TF-IDF-basierten Log-Sequenzrepräsentationen bessere Ergebnisse erzielen als mit den Embedding-basierten Varianten.

Allerdings erreicht selbst das beste Intermediate-Fusion-Modell auf Basis von Random Forest nicht die hohe Präzision des besten Soft-Decision-Fusion-Modells. Damit ergibt sich kein unmittelbarer Hinweis auf eine verbesserte Klassifikationsleistung durch Abbilden möglicher cross-modaler Interaktionen.

Encoding	**Klassifikator**	**Accuracy** (%)	**Precision** (%)	**Recall** (%)	**F1-Score** (%)	**AUC** (%)
Flow + TF-IDF	Random Forest	**98,76**	**98,91**	**98,52**	**98,70**	**99,0**
Flow + TF-IDF	MLP	98,27	98,35	98,04	98,19	**99,0**
Flow + Embedding	Random Forest	97,83	98,24	97,29	97,72	**99,0**
Flow + Embedding	MLP	97,95	97,92	97,82	97,87	**99,0**

Tabelle 6.5.: Übersicht der multimodalen Klassifikationsleistungen basierend auf Flow-Statistiken konkateniert mit verschiedenen Log-Sequenzrepräsentationen

Ein anderes Bild ergibt sich jedoch, wenn die nach APT-Angriffsklassen aufgeschlüsselte Erkennungsgenauigkeit betrachtet wird. Dies ist in Tabelle 6.6 und zum besseren Vergleich in Abbildung 6.6 dargestellt.

	Flow + TF-IDF		**Flow + Embedding**		
Angriffsphase	**RF**	**MLP**	**RF**	**MLP**	**Support**
Benign	99,74	99,19	**99,97**	98,5	134.088
Exfiltration	**99,99**	99,33	99,92	99,74	59.401
Reconnaissance	97,47	**97,57**	89,16	97,51	28.160
Credential Access	1,25	1,43	1,43	**1,55**	1.674
Discovery	75,96	75,0	73,08	**76,92**	104
Command & Control	**57,89**	50,0	31,58	47,37	38
Initial Access	**22,22**	0,0	11,11	0,0	9
Persistence	**100,0**	**100,0**	**100,0**	**100,0**	4
Defense Evasion	**100,0**	**100,0**	**100,0**	**100,0**	2

Tabelle 6.6.: Pro Angriffsphasen Accuracy (%) der multimodalen Modelle basierend auf Intermediate Feature Concatination

Die Intermediate-Fusion-Modelle zeigen eine geringere Genauigkeit bei der Erkennung der häufig vorkommenden normalen Flows, können jedoch Flows anderer Angriffskategorien teils besser zuordnen. So werden beispielsweise 57,89% der Command and Control Flows vom Random-Forest-Modell, welches Flow-Statistiken mit TF-IDF-Repräsentationen kombiniert, korrekt erkannt. Ein deutlicher Anstieg gegenüber 52,63% bzw. 50% Genauigkeit bei den Soft-Decision-Fusions-Modellen. Auch die beiden MLP-Modelle erreichen für die Klassen Credential Access und Reconnaissance leicht höhere Erkennungsraten von 1,55% bzw. 97,57%.

Diese Ergebnisse deuten doch auf eine potenziell verbesserte Klassifikationsleistung durch die Erfassung von in den Daten vorhandenen Interaktionen beider Modalitäten hin. Gleichzeitig führt die einfache Konkatenation der Feature-Repräsentationen auch zu einer erschwerten Trennung der Klassen, was sich insbesondere in der reduzierten Erkennung der normalen Flows zeigt.

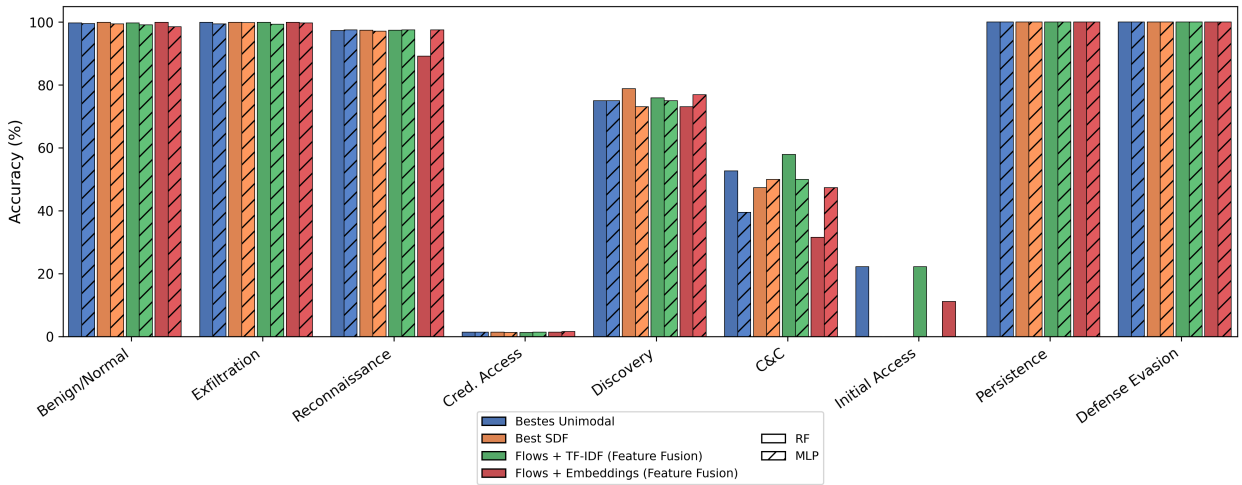

Abbildung 6.6.: Vergleich Erkennungsgenauigkeit zwischen besten unimodalen Ergebnissen, Soft-Decision-Fusion und Late-Fusion-Modellen

6.3. Einfluss von SMOTE auf die Modellergebnisse

Um zu untersuchen, inwieweit die Klassenimbalance einen Einfluss auf die Anomalieerkennung der Klassifikationsmodelle hat, wird im Folgenden das SMOTE Verfahren eingesetzt um die Imbalance auszugleichen. Dabei werden aus den vorhandenen Minderheitenklassen-Samples durch Interpolation neue synthetische Instanzen generiert (siehe Abschnitt 2.3). Anschließend wurden die zuvor bestimmten Hyperparameter erneut genutzt, um sowohl die unimodalen als auch die multimodalen Modelle mit den erweiterten Datensätzen zu trainieren. Die Evaluation erfolgte dabei auf denselben Testdaten wie zuvor, um die Ergebnisse direkt vergleichen zu können.

Abbildung 6.7 zeigt die binäre Klassenverteilung vor und nach dem Einsatz von SMOTE unter Verwendung der Formel 5.2 für das Szenario Wheeler. Es ist zu erkennen, dass nach dem Resampling eine leichte Verschiebung zugunsten der Angriffsklasse besteht, die jedoch im Verhältnis zur ursprünglichen Verteilung ausgewogener ist.

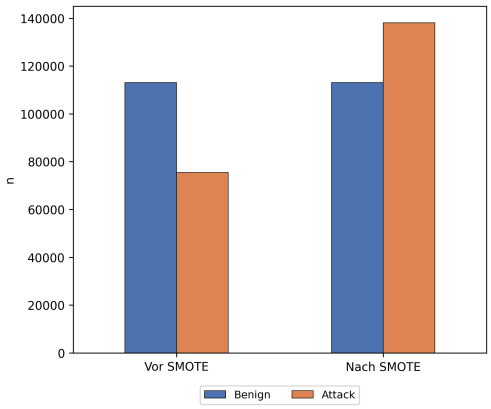

Abbildung 6.7.: Binäre Stichprobenverteilung nach SMOTE-Oversampling auf Szenario Wheeler

In Abbildung 6.8 ist zudem auf einer logarithmischen Skala dargestellt, wie die jeweiligen Subklassen abhängig von ihrer ursprünglichen Größe durch das Oversampling unterschiedlich stark erweitert wurden.

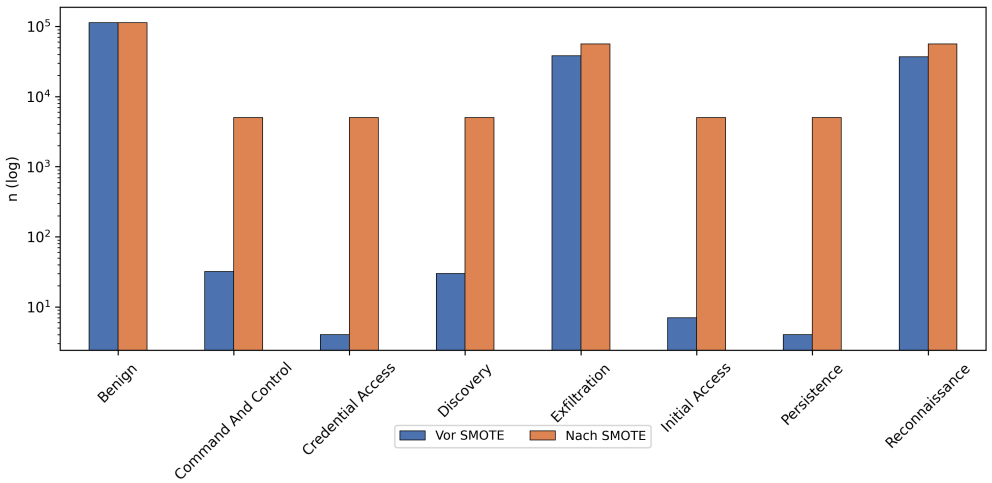

Abbildung 6.8.: Stichprobenverteilung der Subklassen nach SMOTE-Oversampling auf Szenario Wheeler

Um den Effekt von SMOTE auf die Daten zu veranschaulichen, zeigt Abbildung 6.9 eine t-SNE-Projektion der Angriffssamples in Form von Flow-Statistiken vor und nach dem Einsatz von SMOTE.

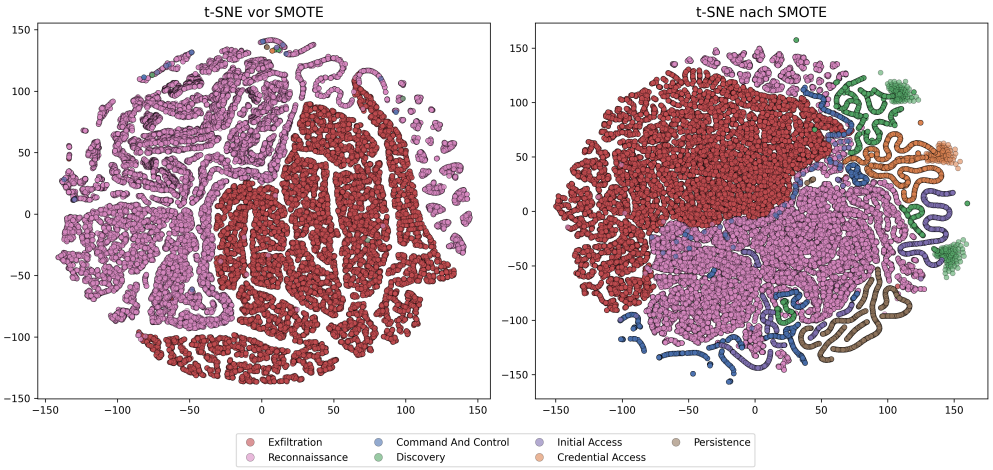

Abbildung 6.9.: t-SNE vergleich der Flow-Statistiken vor und nach Oversampling durch SMOTE auf Szenario Wheeler

Es ist deutlich zu erkennen, dass nach dem Oversampling deutlich mehr Exfiltration- und Reconnaissance-Samples vorliegen und dass ihre jeweiligen Ähnlichkeitscluster klarer sichtbar sind.

Die Projektion des erweiterten Datensatzes zeigt deutliche Cluster auch für die ansonsten selten vorkommenden Klassen. Allerdings lässt sich nicht beurteilen, wie realistisch diese künstlichen Ähnlichkeitscluster sind und ob die Testdaten möglicherweise andere Muster aufweisen. Besonders die Datenpunkte der Klasse Command and Control bilden kein klares abgegrenztes Cluster, sondern wirkt eher wie Rauschen innerhalb der Projektion.

Im Folgenden wird der Einfluss des Resamplings auf die Klassifikationsgenauigkeit sowohl der unimodalen als auch der multimodalen Modelle analysiert.

Unimodale Klassifikation mit SMOTE

Wie in Tabelle 6.7 und in der Vergleichsabbildung 6.10 zu sehen, führt das Resampling mit SMOTE bei der unimodalen Klassifikation zunächst zu keiner Verbesserung der Klassifikatoren. Alle F1-Scores fallen im Vergleich zur Variante ohne SMOTE niedriger aus, was darauf hindeutet, dass die synthetischen Stichproben zusätzliches Rauschen in die Trainingsdaten einbringen und die Modelle somit schlechtere Entscheidungsgrenzen lernen.

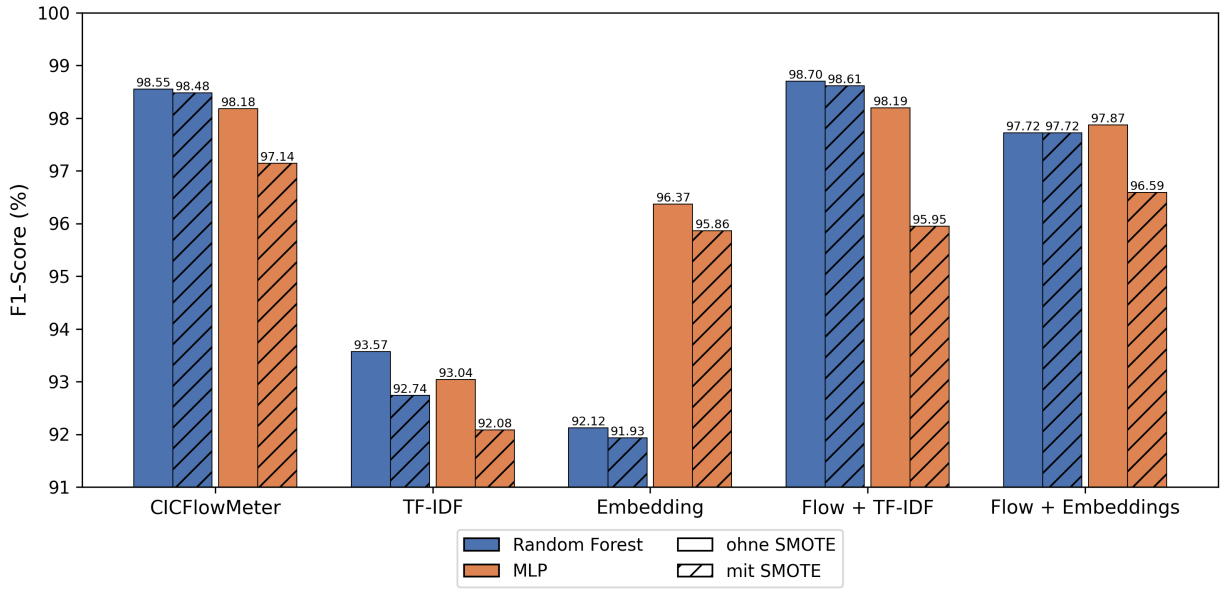

Abbildung 6.10.: Vergleich F1-Score zwischen unimodalen und multimodalen Modellen mit und ohne SMOTE

Encoding	**Klassifikator**	**Accuracy** (%)	**Precision** (%)	**Recall** (%)	**F1-Score** (%)	**AUC** (%)
CICFlowMeter	Random Forest	**98,55**	**98,73**	**98,26**	**98,48**	**100,0**
CICFlowMeter	MLP	97,26	97,20	97,08	97,14	99,0
TF-IDF	Random Forest	92,92	92,33	93,52	92,74	96,0
TF-IDF	MLP	92,26	91,67	92,93	92,08	96,0
Embedding	Random Forest	92,20	91,68	92,26	91,93	97,0
Embedding	MLP	96,00	95,60	96,16	95,86	99,0

Tabelle 6.7.: Übersicht der unimodalen Klassifikationsleistungen unter Verwendung von SMOTE-Oversampling der Trainingsdaten

Betrachtet man die per-Klassen-Genauigkeiten in Tabelle 6.8 sowie den entsprechenden Vergleich in Abbildung 6.11, zeigt sich, dass das unimodale Flow-Statistik-MLP-Modell zwar weiterhin weniger effektiv bei der Erkennung von Normalen, Exfiltration und Reconnaissance Flows ist, das Resampling mit SMOTE jedoch insbesondere die Erkennungsleistung für Credential Access, Discovery und Command and Control verbessert. Bemerkenswert ist, dass für Credential Access die Genauigkeit von zuvor lediglich 1,55% auf 13,28% gesteigert werden konnte, was einer Verbesserung um 11,73 Prozentpunkte entspricht. Insgesamt zeigt sich, dass die MLP-Modelle in der unimodalen Klassifikation besonders gut auf das Resampling ansprechen und unterrepräsentierte Klassen besser erkennen als vorher.

	CICFlowMeter		TF-IDF		Embedding		
Angriffsphase	**RF**	**MLP**	**RF**	**MLP**	**RF**	**MLP**	**Support**
Benign	**99,69**	97,96	90,49	89,59	91,98	95,32	134.088
Exfiltration	**99,98**	98,59	98,70	98,19	97,26	99,37	59.401
Reconnaissance	95,99	96,18	97,60	97,76	89,58	**97,68**	28.160
Credential Access	1,79	**14,28**	5,50	5,97	1,19	4,66	1.674
Discovery	75,00	**80,77**	75,00	78,85	73,08	76,92	104
Command & Control	57,89	**63,16**	52,63	55,26	31,58	42,11	38
Initial Access	22,22	**55,56**	22,22	11,11	0	11,11	9
Persistence	**100,0**	**100,0**	**100,0**	**100,0**	50,0	**100,00**	4
Defense Evasion	**100,0**	**100,0**	**100,0**	**100,0**	**100,0**	**100,0**	2

Tabelle 6.8.: Pro Angriffsphasen Accuracy (%) der unimodalen Modelle unter Verwendung von SMOTE-Oversampling der Trainingsdaten

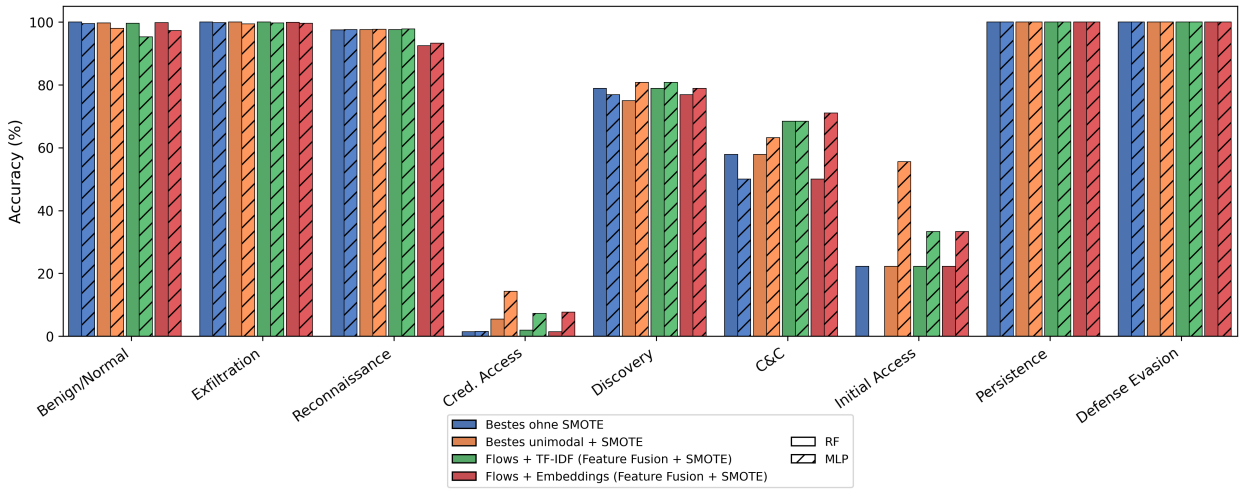

Abbildung 6.11.: Vergleich der Erkennungsgenauigkeiten mit und ohne SMOTE pro Angriffsklasse

Multimodal Klassifikation mit SMOTE

Die F1-Scores der multimodalen Klassifikation mit Feature-Konkatenation und SMOTE, dargestellt in Tabelle 6.9 sowie in Abbildung 6.10, zeigen, dass auch diese multimodalen Modelle durch das Resampling keine besseren Gesamtergebnisse erzielen. Im besten Fall erreichen sie eine vergleichbare Leistung zu den Varianten ohne SMOTE, insgesamt fällt die Erkennungsleistung jedoch eher geringer aus.

Encoding	Klassifikator	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	AUC (%)
Flow + TF-IDF	Random Forest	**98,67**	**98,78**	**98,45**	**98,61**	**99,0**
Flow + TF-IDF	MLP	96,08	95,67	96,29	95,95	**99,0**
Flow + Embedding	Random Forest	97,83	98,24	97,29	97,72	**99,0**
Flow + Embedding	MLP	96,73	96,61	96,58	96,59	**99,0**

Tabelle 6.9.: Übersicht der multimodalen Klassifikationsleistungen unter Verwendung von SMOTE-Oversampling

Die Ergebnisse pro Angriffsklassen in Tabelle 6.10 und Abbildung 6.11 zeigen jedoch auch hier ein differenzierteres Bild. Auch bei der Feature-Konkatenation profitieren insbesondere die MLP-Modelle welche Flows mit Embeddings und SMOTE-Resampling nutzen und erzielen eine deutlich bessere Erkennungsleistung für die Klassen Command & Control, Credential Access und Discovery.

	Flow + TF-IDF		Flow + Embedding		
Angriffsphase	**RF**	**MLP**	**RF**	**MLP**	**Support**
Benign	99,56	95,24	**99,83**	97,33	134.088
Exfiltration	**99,99**	99,73	99,93	99,58	59.401
Reconnaissance	97,55	**97,77**	92,42	93,26	28.160
Credential Access	1,91	7,29	1,49	**7,65**	1.674
Discovery	78,85	**80,77**	76,92	78,85	104
Command & Control	68,42	68,42	50,00	**71,05**	38
Initial Access	22,22	**33,33**	22,22	**33,33**	9
Persistence	**100,00**	**100,00**	**100,00**	**100,00**	4
Defense Evasion	**100,00**	**100,00**	**100,00**	**100,00**	2

Tabelle 6.10.: Pro Angriffsphasen Accuracy (%) der multimodalen Modelle unter Verwendung von SMOTE-Oversampling

Abschließend lässt sich feststellen, dass weder die unimodalen noch die multimodalen Modelle durch den Einsatz von SMOTE auf den Trainingsdaten insgesamt bessere Ergebnisse erzielen konnten. In allen betrachteten Metriken schneiden die Modelle gleichwertig oder schlechter ab als ohne SMOTE.

Lediglich bei der Analyse einzelner Angriffsphasen zeigen die Modelle eine verbesserte Fähigkeit, Flows von Nischenklassen zu erkennen. Gleichzeitig geht diese Verbesserung jedoch zu Lasten der Genauigkeit bei den übrigen Klassen, wodurch die Gesamtleistung nicht gesteigert wird.

7. Fazit und Ausblick

In dieser Arbeit wurde untersucht, welchen Einfluss multimodale Fusionsverfahren auf die Erkennung von Advanced Persistent Threats haben. Zu diesem Zweck wurde ein Datensatz erstellt, in dem Bi-Flows über statistische Vektoren beschrieben und die zugehörigen sequenziellen Log-Nachrichten extrahiert sowie gelabelt wurden. Darüber hinaus wurde eine Trainingspipeline entwickelt, die eine systematische Hyperparameteroptimierung für verschiedene Klassifikatoren ermöglicht und zugleich eine reproduzierbare Auswertung ohne Information Leakage sicherstellt.

Auf Basis der vorbereiteten Daten erfolgte eine Analyse unimodaler und multimodaler Klassifikationsmodelle unter Anwendung von Late- und Intermediate-Fusion-Strategien. Zusätzlich wurde untersucht, ob kontextualisierte Embeddings eine geeignetere Repräsentation unstrukturierter Log-Nachrichten darstellen als klassische Bag-of-Words-Ansätze.

Im Folgenden werden die zentralen Ergebnisse der Untersuchungen zusammengefasst, die wesentlichen Limitationen diskutiert und mögliche Perspektiven für zukünftige Forschungsarbeiten aufgezeigt.

7.1. Zusammenfassung der zentralen Ergebnisse

Die Ergebnisse der unimodalen Klassifikation zeigen bereits, dass die auf Bi-Flows basierende Erkennung von APT-Angriffen die besten Resultate mit den Flow-Statistiken erzielte. Besonders das Random-Forest-Modell konnte ohne zusätzliche Informationen eine Genauigkeit von 98,62%, einen Makro F1-Score von 0,986 sowie einen nahezu perfekten AUC-Wert erreichen.

Auch die ausschließlich auf Lognachricht-Sequenzen basierende Erkennung mittels durch TF-IDF gewichteter Bag-of-Words-Repräsentationen führte zu überzeugenden Resultaten. Allein aus den im Kontext einzelner Flows entstandenen Nachrichten konnten diese mit einer Genauigkeit von über 90% korrekt klassifiziert werden. Das deutet darauf hin, dass die Lognachrichten bereits wesentliche Informationen zur Unterscheidung unterschiedlicher Flow-Kategorien enthalten.

Die besten Ergebnisse der logbasierten Klassifikation wurden jedoch mit kontextuellen Embeddings erzielt. MLPs erreichten dabei eine Genauigkeit von 96.53%, was einer Steigerung um 2.78 Prozentpunkten gegenüber dem besten Bag-of-Words-Modell entspricht. Obwohl die untersuchten Encoder ursprünglich zur Repräsentation natürlicher Sprache trainiert wurden, erfassen sie offenbar auch den semantischen Inhalt von Lognachrichten ausreichend präzise. Dies unterstützt die Hypothese, dass semantische Embeddings zur verbesserten Erkennung von APT-Angriffen beitragen können.

Eine Betrachtung der per Angriffsklasse erzielten Genauigkeiten verdeutlicht, dass die Modelle unterschiedliche Angriffstypen gut erkennen. Dies legte nahe, dass eine Fusion der Informationsquellen potenziell zu einer insgesamt besseren Klassifikation führen könnte.

Für die multimodale Klassifikation wurden die leistungsstärksten unimodalen Modelle mittels Late Fusion über Soft-Decision-Fusion kombiniert. Durch die Zusammenführung der Klassifikationsscores beider Modelle konnten, wie erwartet, leicht verbesserte Gesamtergebnisse erzielt werden.

Besonders die Random-Forest-Modelle erreichten mit einer Genauigkeit von 98,83% sehr gute Ergebnisse.

Die ungleich gewichtete Fusion zeigte hingegen deutlich schlechtere Resultate als die gleich gewichteten Modelle. Dies deutet darauf hin, dass beide Informationsquellen gleichwertig zur Entscheidungsfindung beitragen. Eine Betrachtung der pro Angriffklassengenauigkeit bestätigte dies.

Mittels Soft Decision Fusion konnte die Erkennung von Flows der Credential Access und Command & Control, die zuvor überwiegend von den rein logbasierten Modellen erkannt wurden, verbessert werden. Damit zeigt sich, dass bereits einfache multimodale Fusion zur Erhöhung der Erkennungsleistung von APT-Angriffen beitragen kann.

Im Gegensatz dazu konnten die multimodalen Modelle mit Intermediate Fusion durch Merkmalskonkatenation keine signifikanten Leistungssteigerungen erzielen. Besonders die Kombination von Flow-Statistiken und Log-Embeddings führte nicht zu den erwarteten Verbesserungen, sondern lag insgesamt auf einem ähnlichen oder leicht niedrigeren Niveau als die Soft-Decision-Fusion. Es ergaben sich somit keine Hinweise darauf, dass durch Feature-Fusion zusätzliche Interaktionen zwischen den Modalitäten im Trainingsprozess effektiv erfasst wurden und, dass die gemeinsame Repräsentation der Modalitäten im Feature-Raum nicht automatisch zu einer besseren Integration führt.

Eine erneute Analyse der klassenspezifischen Genauigkeiten zeigte jedoch, dass durch Fusion auch in diesem Fall seltene Angriffsklassen tendenziell besser erkannt wurden. Der ergänzende Einsatz von SMOTE erlaubte zudem eine Beurteilung des Einflusses der Datenimbalance. Zwar verschlechterte sich die Gesamtleistung der Modelle nach dem Oversampling leicht, jedoch verbesserte sich die Erkennung von Minderheitsklassen, insbesondere der APT-Phasen Discovery und Command and Control, gegenüber den unimodalen Vergleichsmodellen.

7.2. Limitationen und Ausblick

Abschließend bleibt zu erwähnen, dass die Untersuchung dieser Arbeit lediglich auf einem einzelnen Datensatz, der ausschließlich synthetisch generierte APT ähnliche Angriffe umfasst. Dadurch ist die Übertragbarkeit der Ergebnisse auf reale Angriffsszenarien nur bedingt gegeben.

Darüber hinaus erfolgte die Hyperparameteroptimierung lediglich über Random Search, wodurch nicht sichergestellt ist, dass stets die global optimalen Modellparameter gefunden wurden. Eine umfassendere grid- oder bayes-basierte Suche könnte hier eine robustere Modellkonfiguration ermöglichen.

Zudem wäre ein weiterer detaillierterer Vergleich zwischen unterschiedlichen Log-Repräsentationen im besonderen im Bezug auf die Sequenzdarstellung vorteilhaft. Die in dieser Arbeit erstellen Embeddings durch das nicht unangepasste Encoder-Modell sollten mit einem extra für Log-Nachrichten trainierten Encoder verglichen werden. Ebenso beschränkt sich die vorliegende Arbeit auf die statische Repräsentationen von Log-Sequenzen mittels Pooling-Verfahren. Künftige Arbeiten sollten hier dynamische Ansätze, etwa sequenzbasierte Modelle wie RNNs oder Transformer, einbeziehen, um die zeitlichen Abhängigkeiten zwischen Logeinträgen besser zu erfassen.

Schließlich zeigte sich, dass die Intermediate-Fusion-Modelle zwar bestimmte Angriffsklassen besser erkennen, jedoch gleichzeitig eine geringere Präzision bei normalen Flows aufweisen. Eine vertiefende Analyse dieses Effekts könnte aufzeigen, wie durch gezielte Anpassungen der Fusionsstrategien der Nutzen multimodaler Integration weiter gesteigert werden kann.

7.3. Fazit

Zusammenfassend konnte in dieser Arbeit gezeigt werden, dass die multimodale Fusion von Flow-Statistiken und Log-Nachrichten ein vielversprechender Ansatz zur Verbesserung der Erkennung von APT-Angriffen darstellt. Insbesondere konnte der Einsatz moderner Embeddings demonstrieren, dass der semantische Inhalt unstrukturierter Log-Nachrichten besser erfasst werden kann, was zu einer aussagekräftigeren Datenbasis für die Angriffserkennung führt.

Dennoch sind weitere Untersuchungen erforderlich, um die praktische Umsetzbarkeit in realen Szenarien zu evaluieren und zu analysieren, wie die verbesserte Erkennung von Minderheitsklassen in robustere Modelle integriert werden kann. Die Ergebnisse dieser Arbeit legen jedoch nahe, dass die Kombination aus Flow-Daten und semantisch angereicherten Log-Nachrichten ein vielversprechendes Potenzial für die Weiterentwicklung von Sicherheitsanalysen bietet.

Literatur

[1] Nandita Pattnaik et al. *It's more than just money: The real-world harms from ransomware attacks*. 2023. DOI: 10.48550/ARXIV.2307.02855.

[2] Bitkom e. V. *Wirtschaftsschutz 2025*. 1. Sep. 2025. URL: https://www.bitkom.org/sites/main/files/2025-09/bitkom-pressekonferenz-wirtschaftsschutz-cybercrime.pdf (besucht am 14. 11. 2025).

[3] Branka Stojanović, Katharina Hofer-Schmitz und Ulrike Kleb. „APT datasets and attack modeling for automated detection methods: A review". In: *Computers & Security* 92 (Mai 2020), S. 101734. ISSN: 0167-4048. DOI: 10.1016/j.cose.2020.101734.

[4] Cho Do Xuan und Mai Hoang Dao. „A novel approach for APT attack detection based on combined deep learning model". In: *Neural Computing and Applications* 33.20 (Apr. 2021), S. 13251–13264. ISSN: 1433-3058. DOI: 10.1007/s00521-021-05952-5.

[5] Jaafer Al-Saraireh und Ala' Masarweh. „A novel approach for detecting advanced persistent threats". In: *Egyptian Informatics Journal* 23.4 (Dez. 2022), S. 45–55. ISSN: 1110-8665. DOI: 10.1016/j.eij.2022.06.005.

[6] Nadim Ibrahim et al. „An optimized hybrid ensemble machine learning model combining multiple classifiers for detecting advanced persistent threats in networks". In: *Journal of Big Data* 12.1 (Aug. 2025). ISSN: 2196-1115. DOI: 10.1186/s40537-025-01272-w.

[7] Wei Guan et al. *LogLLM: Log-based Anomaly Detection Using Large Language Models*. 2024. DOI: 10.48550/ARXIV.2411.08561.

[8] Junwei Zhou et al. „DeepSyslog: Deep Anomaly Detection on Syslog Using Sentence Embedding and Metadata". In: *IEEE Transactions on Information Forensics and Security* 17 (2022), S. 3051–3061. ISSN: 1556-6021. DOI: 10.1109/tifs.2022.3201379.

[9] Shih-Cheng Huang et al. „Multimodal fusion with deep neural networks for leveraging CT imaging and electronic health record: a case-study in pulmonary embolism detection". In: *Scientific Reports* 10.1 (Dez. 2020). ISSN: 2045-2322. DOI: 10.1038/s41598-020-78888-w.

[10] Adel Alshamrani et al. „A Survey on Advanced Persistent Threats: Techniques, Solutions, Challenges, and Research Opportunities". In: *IEEE Communications Surveys & Tutorials* 21.2 (2019), S. 1851–1877. ISSN: 2373-745X. DOI: 10.1109/comst.2019.2891891.

[11] *Lazarus Group MITRE ATT&CK*. URL: https://attack.mitre.org/versions/v17/groups/G0032/ (besucht am 20. 10. 2025).

[12] *APT29 MITRE ATT&CK*. URL: https://attack.mitre.org/versions/v17/groups/G0016/ (besucht am 20. 10. 2025).

[13] Rahaf Alkhadra et al. „Solar Winds Hack: In-Depth Analysis and Countermeasures". In: *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. IEEE, Juli 2021. DOI: 10.1109/icccnt51525.2021.9579611.

[14] Eric Hutchins, Michael Cloppert und Rohan Amin. „Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains". In: *Leading Issues in Information Warfare & Security Research* 1 (Jan. 2011).

[15] *MITRE ATT&CK*. URL: https://attack.mitre.org/ (besucht am 20. 10. 2025).

[16] Jingci Zhang et al. „ATT&CK-based Advanced Persistent Threat attacks risk propagation assessment model for zero trust networks“. In: *Computer Networks* 245 (Mai 2024), S. 110376. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2024.110376.

[17] Abdussamad Syed et al. „Comprehensive Advanced Persistent Threats Dataset“. In: *IEEE Networking Letters* 7.2 (Juni 2025), S. 150–154. ISSN: 2576-3156. DOI: 10.1109/lnet.2025.3551989.

[18] Syed Sohaib Karim et al. „Advanced Persistent Threat (APT) and intrusion detection evaluation dataset for linux systems 2024“. In: *Data in Brief* 54 (Juni 2024), S. 110290. ISSN: 2352-3409. DOI: 10.1016/j.dib.2024.110290.

[19] Olesia Barkovska et al. „Analysis of the impact of the contextual embeddings usage on the text classification accuracy“. In: *Radioelectronic and Computer Systems* 2024.3 (Aug. 2024), S. 67–79. ISSN: 1814-4225. DOI: 10.32620/reks.2024.3.05.

[20] Lukas Layer et al. „Automatic log analysis with NLP for the CMS workflow handling“. In: *EPJ Web of Conferences* 245 (2020). Hrsg. von C. Doglioni et al., S. 03006. ISSN: 2100-014X. DOI: 10.1051/epjconf/202024503006.

[21] Piotr Ryciak, Katarzyna Wasielewska und Artur Janicki. „Anomaly Detection in Log Files Using Selected Natural Language Processing Methods“. In: *Applied Sciences* 12.10 (Mai 2022), S. 5089. ISSN: 2076-3417. DOI: 10.3390/app12105089.

[22] Egil Karlsen et al. „Large language models and unsupervised feature learning: implications for log analysis“. In: *Annals of Telecommunications* 79.11–12 (Apr. 2024), S. 711–729. ISSN: 1958-9395. DOI: 10.1007/s12243-024-01028-2.

[23] Daniel Jurafsky und James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models*. 3rd. Online Manuskript veröffentlicht August 24, 2025. 2025. URL: https://web.stanford.edu/~jurafsky/slp3/.

[24] Sangwhan Moon und Naoaki Okazaki. „Effects and Mitigation of Out-of-vocabulary in Universal Language Models“. In: *Journal of Information Processing* 29.0 (2021), S. 490–503. ISSN: 1882-6652. DOI: 10.2197/ipsjjip.29.490.

[25] Yonghui Wu et al. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. 2016. DOI: 10.48550/ARXIV.1609.08144.

[26] Rico Sennrich, Barry Haddow und Alexandra Birch. *Neural Machine Translation of Rare Words with Subword Units*. 2015. DOI: 10.48550/ARXIV.1508.07909.

[27] An Yang et al. *Qwen3 Technical Report*. 2025. DOI: 10.48550/ARXIV.2505.09388.

[28] Changhan Wang, Kyunghyun Cho und Jiatao Gu. *Neural Machine Translation with Byte-Level Subwords*. 2019. DOI: 10.48550/ARXIV.1909.03341.

[29] G. Salton, A. Wong und C. S. Yang. „A vector space model for automatic indexing“. In: *Communications of the ACM* 18.11 (Nov. 1975), S. 613–620. ISSN: 1557-7317. DOI: 10.1145/361219.361220.

[30] Christopher D. Manning, Prabhakar Raghavan und Hinrich Schütze. *Introduction to information retrieval*. Reprinted. Cambridge [u.a.]: Cambridge Univ. Press, 2009. 482 S. ISBN: 9780521865715.

[31] Mohammad Taher Pilehvar und Jose Camacho-Collados. *Embeddings in Natural Language Processing: Theory and Advances in Vector Representations of Meaning*. Springer International Publishing, 2021. ISBN: 9783031021770. DOI: 10.1007/978-3-031-02177-0.

[32] Carl Allen. *Towards a Theoretical Understanding of Word and Relation Representation*. 2022. DOI: 10.48550/ARXIV.2202.00486.

[33] Tomas Mikolov et al. „Efficient Estimation of Word Representations in Vector Space“. In: (16. Jan. 2013). DOI: 10.48550/ARXIV.1301.3781.

[34] Jeffrey Pennington, Richard Socher und Christopher Manning. „Glove: Global Vectors for Word Representation“. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014. DOI: 10.3115/v1/d14-1162.

[35] Matthew E. Peters et al. „Deep contextualized word representations“. In: (15. Feb. 2018). DOI: 10.48550/ARXIV.1802.05365.

[36] Jacob Devlin et al. „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding“. In: (11. Okt. 2018). DOI: 10.48550/ARXIV.1810.04805.

[37] Haibo He, Hrsg. *Imbalanced learning. Foundations, algorithms, and applications*. Hoboken, NJ: Wiley, 2013. 216 S. ISBN: 9781118646106.

[38] N. V. Chawla et al. „SMOTE: Synthetic Minority Over-sampling Technique“. In: *Journal of Artificial Intelligence Research* 16 (Juni 2002), S. 321–357. ISSN: 1076-9757. DOI: 10.1613/jair.953.

[39] Tadas Baltrusaitis, Chaitanya Ahuja und Louis-Philippe Morency. „Multimodal Machine Learning: A Survey and Taxonomy“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.2 (Feb. 2019), S. 423–443. ISSN: 1939-3539. DOI: 10.1109/tpami.2018.2798607.

[40] Dhanesh Ramachandram und Graham W. Taylor. „Deep Multimodal Learning: A Survey on Recent Advances and Trends“. In: *IEEE Signal Processing Magazine* 34.6 (Nov. 2017), S. 96–108. ISSN: 1053-5888. DOI: 10.1109/msp.2017.2738401.

[41] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. DOI: 10.48550/ARXIV.2103.00020.

[42] Can Cui et al. „Deep multimodal fusion of image and non-image data in disease diagnosis and prognosis: a review“. In: *Progress in Biomedical Engineering* 5.2 (Apr. 2023), S. 022001. ISSN: 2516-1091. DOI: 10.1088/2516-1091/acc2fe.

[43] Amir Zadeh et al. *Tensor Fusion Network for Multimodal Sentiment Analysis*. 2017. DOI: 10.48550/ARXIV.1707.07250.

[44] Valerio Guarrasi et al. „A systematic review of intermediate fusion in multimodal deep learning for biomedical applications“. In: *Image and Vision Computing* 158 (Mai 2025), S. 105509. ISSN: 0262-8856. DOI: 10.1016/j.imavis.2025.105509.

[45] Zhi-Hua Zhou. *Ensemble methods. Foundations and algorithms*. Online-Ausg. Chapman & Hall. Boca Raton, FL: Taylor & Francis, 2013. 234 S. ISBN: 9781439830031.

[46] Rebecca Bace und Peter Mell. *Intrusion Detection Systems*. en. 1. Nov. 2001.

[47] Adam Khalid et al. „Advanced Persistent Threat Detection: A Survey“. In: *2021 3rd International Cyber Resilience Conference (CRC)*. IEEE, Jan. 2021, S. 1–6. DOI: 10.1109/crc50527.2021.9392626.

[48] Van-Hoang Le und Hongyu Zhang. „Log-based anomaly detection with deep learning: how far are we?“ In: *Proceedings of the 44th International Conference on Software Engineering*. ICSE '22. ACM, Mai 2022, S. 1356–1367. DOI: 10.1145/3510003.3510155.

[49] Min Du et al. „DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning“. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS '17. ACM, Okt. 2017, S. 1285–1298. DOI: 10.1145/3133956.3134015.

[50] Weibin Meng et al. „LogAnomaly: Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs“. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. IJCAI-2019. International Joint Conferences on Artificial Intelligence Organization, Aug. 2019, S. 4739–4745. DOI: 10.24963/ijcai.2019/658.

[51] Alec Radford et al. *Improving Language Understanding by Generative Pre-Training*. Techn. Ber. OpenAI, 2018.

[52] Haixuan Guo, Shuhan Yuan und Xintao Wu. „LogBERT: Log Anomaly Detection via BERT“. In: (7. März 2021). DOI: 10.48550/ARXIV.2103.04475.

[53] Gavin Watson. „A Comparison of Header and Deep Packet Features when Detecting Network Intrusions“. en. In: (2018). DOI: 10.13016/M2G737680.

[54] R Vinayakumar, K P Soman und Prabaharan Poornachandran. „Applying convolutional neural network for network intrusion detection“. In: *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, Sep. 2017, S. 1222–1228. DOI: 10.1109/icacci.2017.8126009.

[55] Jay Sinha und M. Manollas. „Efficient Deep CNN-BiLSTM Model for Network Intrusion Detection“. In: *Proceedings of the 2020 3rd International Conference on Artificial Intelligence and Pattern Recognition*. AIPR 2020. ACM, Juni 2020, S. 223–231. DOI: 10.1145/3430199.3430224.

[56] Wei Dai et al. „Network Intrusion Detection Method Based on CNN-BiLSTM-Attention Model“. In: *IEEE Access* 12 (2024), S. 53099–53111. ISSN: 2169-3536. DOI: 10.1109/access.2024.3384528.

[57] Xiaosong Wang et al. „Advanced Network Intrusion Detection with TabTransformer“. In: *Journal of Theory and Practice of Engineering Science* 4.03 (März 2024), S. 191–198. ISSN: 2790-1505. DOI: 10.53469/jtpes.2024.04(03).18.

[58] Aklil Kiflay et al. „Network intrusion detection leveraging multimodal features“. In: *Array* 22 (Juli 2024), S. 100349. ISSN: 2590-0056. DOI: 10.1016/j.array.2024.100349.

[59] Erxue Min et al. „TR-IDS: Anomaly-Based Intrusion Detection through Text-Convolutional Neural Network and Random Forest“. In: *Security and Communication Networks* 2018 (Juli 2018), S. 1–9. ISSN: 1939-0122. DOI: 10.1155/2018/4943509.

[60] George Agrafiotis et al. „Advancing Cybersecurity with AI: A Multimodal Fusion Approach for Intrusion Detection Systems“. In: *2024 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*. IEEE, Juli 2024, S. 51–56. DOI: 10.1109/meditcom61057.2024.10621237.

[61] Ren-Hung Hwang et al. „Host-based intrusion detection with multi-datasource and deep learning“. In: *Journal of Information Security and Applications* 78 (Nov. 2023), S. 103625. ISSN: 2214-2126. DOI: 10.1016/j.jisa.2023.103625.

[62] Jinxin Liu et al. „Collaborative Feature Maps of Networks and Hosts for AI-driven Intrusion Detection“. In: *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*. IEEE, Dez. 2022, S. 2662–2667. DOI: 10.1109/globecom48099.2022.10000985.

[63] Leo Breiman. „Random Forests“. In: *Machine Learning* 45.1 (Okt. 2001), S. 5–32. ISSN: 1573-0565. DOI: 10.1023/a:1010933404324.

[64] Marius-Constantin Popescu et al. „Multilayer perceptron and neural networks“. In: *WSEAS Trans. Cir. and Sys.* 8.7 (Juli 2009), S. 579–588. ISSN: 1109-2734.

[65] Léo Grinsztajn, Edouard Oyallon und Gaël Varoquaux. „Why do tree-based models still outperform deep learning on tabular data?“ In: (18. Juli 2022). DOI: 10.48550/ARXIV.2207.08815.

[66] Scikit-learn developers. *compute_class_weight - scikit-learn documentation*. Scikit-learn. 2025. URL: https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html (besucht am 17. 11. 2025).

[67] Shiv Ram Dubey, Satish Kumar Singh und Bidyut Baran Chaudhuri. *Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark*. 2021. DOI: 10.48550/ARXIV.2109.14545.

[68] Sonal Sannigrahi, Josef van Genabith und Cristina Espana-Bonet. *Are the Best Multilingual Document Embeddings simply Based on Sentence Embeddings?* 2023. DOI: 10.48550/ARXIV.2304.14796.

[69] Karim Sattar et al. „Transparent deep machine learning framework for predicting traffic crash severity“. In: *Neural Computing and Applications* 35.2 (Sep. 2022), S. 1535–1547. ISSN: 1433-3058. DOI: 10.1007/s00521-022-07769-2.

[70] Cheng Guo und Felix Berkhahn. *Entity Embeddings of Categorical Variables*. 2016. DOI: 10.48550/ARXIV.1604.06737.

[71] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. Hrsg. von Aaron Courville und Yoshua Bengio. Adaptive computation and machine learning. Cambridge, Massachusetts: MIT Press, 2016. 1775 S. ISBN: 9780262337373.

[72] Juan Terven et al. „A comprehensive survey of loss functions and metrics in deep learning“. In: *Artificial Intelligence Review* 58.7 (Apr. 2025). ISSN: 1573-7462. DOI: 10.1007/s10462-025-11198-7.

[73] Patrik Goldschmidt und Daniela Chudá. „Network intrusion datasets: A survey, limitations, and recommendations“. In: *Computers & Security* 156 (Sep. 2025), S. 104510. ISSN: 0167-4048. DOI: 10.1016/j.cose.2025.104510.

[74] Sowmya Myneni et al. „Unraveled — A semi-synthetic dataset for Advanced Persistent Threats“. In: *Computer Networks* 227 (Mai 2023), S. 109688. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2023.109688.

[75] Max Landauer et al. „Maintainable Log Datasets for Evaluation of Intrusion Detection Systems“. In: *IEEE Transactions on Dependable and Secure Computing* 20.4 (Juli 2023), S. 3466–3482. ISSN: 2160-9209. DOI: 10.1109/tdsc.2022.3201582.

[76] Max Landauer et al. *AIT Log Data Set V2.0*. 2022. DOI: 10.5281/ZENODO.5789064.

[77] Francesca Soro et al. *AIT Netflow Data Set*. 2022. DOI: 10.5281/ZENODO.6610489.

[78] B. Trammell und E. Boschi. *Bidirectional Flow Export Using IP Flow Information Export (IPFIX)*. Jan. 2008. DOI: 10.17487/rfc5103.

[79] Gabriel Potter Philippe Biondi und Contributors. *Scapy*. URL: https://github.com/secdev/scapy (besucht am 24. 11. 2025).

[80] Le Hieu und Contributors. *Python CICFlowMeter*. URL: https://github.com/hieulw/cicflowmeter (besucht am 19. 11. 2025).

[81] Maria Rodriguez et al. „Evaluation of Machine Learning Techniques for Traffic Flow-Based Intrusion Detection“. In: *Sensors* 22.23 (Nov. 2022), S. 9326. ISSN: 1424-8220. DOI: 10.3390/s22239326.

[82] Qianru Zhou und Dimitrios Pezaros. „Evaluation of Machine Learning Classifiers for Zero-Day Intrusion Detection“. In: (9. Mai 2019). DOI: 10.48550/ARXIV.1905.03685. arXiv: 1905.03685 [cs.CR].

[83] C. Jennings. *Network Address Translation (NAT) Behavioral Requirements for Unicast UDP*. Hrsg. von F. Audet. Jan. 2007. DOI: 10.17487/rfc4787.

[84] Pinjia He et al. „Drain: An Online Log Parsing Approach with Fixed Depth Tree“. In: *2017 IEEE International Conference on Web Services (ICWS)*. IEEE, Juni 2017, S. 33–40. DOI: 10.1109/icws.2017.13.

[85] Min Du und Feifei Li. „Spell: Streaming Parsing of System Event Logs“. In: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, Dez. 2016. DOI: 10.1109/icdm.2016.0103.

[86] F. Pedregosa et al. „Scikit-learn: Machine Learning in Python“. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830.

[87] Yanzhao Zhang et al. „Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models“. In: (5. Juni 2025). DOI: 10.48550/ARXIV.2506.05176. arXiv: 2506.05176 [cs.CL].

[88] Niklas Muennighoff et al. „MTEB: Massive Text Embedding Benchmark“. In: (13. Okt. 2022). DOI: 10.48550/ARXIV.2210.07316. arXiv: 2210.07316 [cs.CL].

[89] *SentenceTransformers Documentation*. URL: https://www.sbert.net/index.html (besucht am 19. 11. 2025).

[90] Steven Bird und Edward Loper. „NLTK: The Natural Language Toolkit“. In: *Proceedings of the ACL Interactive Poster and Demonstration Sessions*. Barcelona, Spain: Association for Computational Linguistics, Juli 2004, S. 214–217. URL: https://aclanthology.org/P04-3031/.

[91] Adam Paszke et al. „PyTorch: An Imperative Style, High-Performance Deep Learning Library“. In: (3. Dez. 2019). DOI: 10.48550/ARXIV.1912.01703. arXiv: 1912.01703 [cs.LG].

[92] Guillaume Lemaitre, Fernando Nogueira und Christos K. Aridas. „Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning“. In: (21. Sep. 2016). DOI: 10.48550/ARXIV.1609.06570. arXiv: 1609.06570 [cs.LG].

Erklärung

Ich versichere, dass ich die vorliegende Arbeit mit dem Thema:

„Multimodale Datenfusion zur Klassifikation von Angriffsphasen in Advanced Persistent Threat Attacken“

selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinngemäße Zitate als solche gekennzeichnet. Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann. Ich versichere, dass das elektronische Exemplar mit den gedruckten Exemplaren übereinstimmt.

Leipzig, den 17.12.2025

Simon Mellert