

Anomalieerkennung in Keycloak-Logs mittels maschinellen Lernens: Vergleich von Hybridmodellen

Vorgelegt von:

Ümmühan Ay

Matrikelnummer: 7060837

Eingereicht am: 24.03.2025

Abgabedatum: 08.09.2025

Bearbeitungszeitraum: 12 Wochen

Duale Hochschule Baden-Württemberg Stuttgart
Fakultät für Informatik

Erstprüfer: Eric Hämmerle
Zweitprüfer: Dr. Janko Dietzsch

Danksagung

Ich will mich hiermit bei meinem betrieblichen Betreuer Eric Hämmerle bedanken, sowie meinen Vorgesetzten Frithard Meyer von Uptrup, dass diese Arbeit ermöglicht wurde. Ebenfalls bedanke ich mich bei meiner Familie und meine Freunden für die Unterstützung und Motivation, die dazu geführt hat, diese Arbeit fertig zu stellen. Ebenfalls bedanke ich mich bei meinem Betreuer der DHBW, Herr Dr. Janko Dietzsch für das Feedback.

Abstract

This thesis examines the implementation and comparison of machine learning models and hybrid architectures for anomaly detection and the detection of intrusion and insider threat attacks in the Keycloak product. The goal was to provide the company with cost-effective tools, as conventional machine learning solutions have been associated with high costs to date. This thesis compares three hybrid models: LSTM autoencoders combined with one of the following algorithms: Isolation Forest, One-Class SVM, and DBSCAN. Previous studies show in particular that LSTM autoencoders in combination with Isolation Forest achieve high results. However, the analyses in this thesis show that the hypothesis cannot be clearly verified or falsified, as LSTM-AE with OCSVM or Isolation Forest achieved comparable overall performance. Log data was used as the data set, which was originally self-generated but ultimately produced by Selenium tests in Keycloak.

Zusammenfassung

Diese Arbeit untersucht die Implementierung und den Vergleich von maschinellen Lernmodellen und hybriden Architekturen zur Anomalieerkennung und zur Erkennung von Intrusions- sowie Insider-Threat-Attacken im Produkt Keycloak. Ziel war es, dem Unternehmen kosteneffiziente Werkzeuge bereitzustellen, da konventionelle maschinelle Lernlösungen bisher mit hohen Kosten verbunden sind. In dieser Arbeit wurden drei Hybridmodelle verglichen: LSTM-Autoencoder kombiniert mit jeweils einem der Algorithmen Isolation Forest, One-Class SVM und DBSCAN. Bisherige Studien zeigen insbesondere, dass LSTM-Autoencoder in Kombination mit Isolation Forest hohe Ergebnisse erzielt. Die Analysen dieser Arbeit zeigen jedoch, dass die Hypothese weder eindeutig verifiziert noch falsifiziert werden kann, da LSTM-AE mit OCSVM oder Isolation Forest insgesamt vergleichbare Leistungen erreichten. Als Datensatz wurden Log-Daten verwendet, die ursprünglich selbst generiert, schließlich aber durch Selenium-Tests in Keycloak erzeugt wurden.

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne unzulässige fremde Hilfe angefertigt habe. Alle verwendeten Quellen und Hilfsmittel sind in der Arbeit angegeben. Diese Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Ort, Datum: 25.08.2025

Unterschrift:

A handwritten signature in blue ink, appearing to read 'Thimothée H.'.

Inhaltsverzeichnis

1	Einleitung	1
1.1	Benutzerverhalten und Anomalien	1
1.2	Kategorisierung von Anomalien	1
1.3	Regelbasierte Methoden und weitere Maßnahmen in Keycloak	3
1.4	Grenzen der Sicherheitsmaßnahmen in Keycloak	4
1.5	Maschinelle Lernmodelle als zukünftige Sicherheitsmaßnahme	5
1.6	User (and Entity) Behavior Analytics (U(E)BA)	6
2	Forschungsstand, Herausforderungen und Hypothesen	6
2.1	Bisheriger Forschungsstand	6
2.2	Hypothese	7
3	Methoden und Störfaktoren	9
3.1	Metriken	9
3.2	Störfaktoren	12
4	Angewandte Technologien	13
4.1	Scikit-learn Learn	13
4.2	Keras	13
4.3	TensorFlow	14
4.4	Verworfenen Technologie: Deep GPU Xceleration (DGX)	14
5	Keycloak	14
5.1	Terminologien in Keycloak	14
5.2	Komponenten der Log-Erzeugung	16
5.3	Log-Dateien in Keycloak	17
5.4	Event-Logs und Audit-Logs	18
5.5	Aufbau der Event-Logs	18
6	Theoretischer Hintergrund der Modelle	20
6.1	LSTM	20
6.2	Autoencoder	21
6.3	Isolation Forest	23
6.4	One-Class SVM	24
6.5	DBSCAN	25
6.6	Weitere Algorithmen für zukünftige Studien	26

7	Implementierung	27
7.1	Hybridmodelle mit LSTM-AE	27
7.1.1	Implementierung: Isolation Forest und OCSVM	31
7.1.2	Implementierung: DBSCAN	32
7.2	Alternative Architekturen	32
8	Generierung der Logs	33
8.1	Erster Entwurf der Angriffsfälle	34
8.2	Automatische Erstellung der Keycloak-Logs	34
8.3	Grenzen dieser Lösung	36
9	Zweiter Ansatz: Manuelle Log-Generierung und durch Selenium	36
9.1	Anbindung der Keycloak-API	37
9.2	Datenbeschaffung durch Selenium	38
9.3	Mögliche Angriffsszenarien basierend auf CVE und CWE	42
9.4	Angriffsszenarien	43
9.4.1	Privilege Escalation	43
9.4.2	Angriffsszenario: Account-Sabotage durch privilegierten Benutzer	45
9.4.3	Angriffsszenario: Dictionary-Angriff über die Web-Oberfläche	46
9.5	Änderung der Timestamps durch Unix Timestamp	48
10	Durchführung des Trainings	49
10.1	Die Anzahl der Log-Einträge	50
10.2	Testfälle	51
11	Auswertung	53
11.1	Berechnung der Mittelwerte	53
11.1.1	Berechnung der Standardabweichung und Varianz	54
11.1.2	Friedman-Test	55
11.1.3	Nemenyi-Test	57
12	Diskussion	59
13	Fazit	61
14	Ausblick	61
14.1	Wirtschaftliche Anwendung	61
14.2	Wissenschaftlicher Ausblick	61
14.3	Angriffe	62
14.3.1	Testlauf 1	74
14.3.2	Testlauf 2	75

14.3.3	Testlauf 3	76
14.3.4	Testlauf 4	77
14.3.5	Testlauf 5	78
14.3.6	Testlauf 1	80
14.3.7	Testlauf 2	81
14.3.8	Testlauf 3	82
14.3.9	Testlauf 4	83
14.3.10	Testlauf 5	84

Abbildungsverzeichnis

1	Beispielhafte Konfiguration von Authentication Flows in Keycloak: Hier ist bspw. die Client Authentication, Standard Flow, Service Accounts Roles und Direct Access Grant aktiviert.	16
2	Log-Ausgaben aus einer Quarkus-basierten Keycloak-Instanz.	17
3	Hier kann man in Keycloak die User und Admin Events einsehen. Man erkennt hier in den User-Events schon ein paar Einträge mit dem entsprechenden Event-Typ	20
4	Visuelle Darstellung der Funktionsweise eines Autoencoders. Das Kästchen in der Mitte stellt den Bottleneck dar. [Online]. Quelle: https://www.researchgate.net/figure/Structure-of-the-autoencoder_fig1_36507443 , letzter Zugriff am: 15.07.2025	22
5	Visuelle Darstellung der Funktionsweise des Isolation Forest [Online]. Quelle: https://www.researchgate.net/figure/Anomaly-Detection-using-Isolation-Forest_fig3_350551253 , letzter Zugriff am: 15.07.2025	23
6	Visuelle Darstellung der OCSVM [Online]. Quelle: https://www.mdpi.com/2076-3417/13/3/1734 , letzter Zugriff am: 15.07.2025	25
7	Beispielhafte Darstellung von DBSCAN-Clustern. Quelle: https://www.baeldung.com/wp-content/uploads/sites/4/2023/04/clusters.png , letzter Zugriff am: 15.07.2025	26
8	Gesamtarchitektur des LSTM-Autoencoders	29
9	Ausschnitt der generierten Logs, Man erkennt hier schon ein Log mit dem Label 0, also ein normaler Log, unterhalb ist eines mit dem Label 1- eine Anomalie. Man erkennt hier schon den Typ LOGIN_ERROR.	35
10	Daten, die an den Keycloak-Server gesendet werden müssen	37
11	Beispielhafte Darstellung der generierten Benutzer für den Testfall	40
12	Benötigte Rollen für den Client Nintendo	41
13	Aktivitätsdiagramm Privilege Escalation	44
14	Aktivitätsdiagramm von Account-Sabotage	45
15	Aktivitätsdiagramm von Dictionary	47
16	Aktuelle Uhrzeit nach dem Unix-System [Online]. Verfügbar unter: https://www.unixtimestamp.com/ , abgerufen am 18. August 2025.	48
17	Überblick der Konstanten Werte und der unabhängigen Variable	52

Tabellenverzeichnis

1	Ausgewählte Parameter der Modellkonfiguration	28
2	Vergleich der Modelle bei Sequenzlänge 25	53
3	Übersicht der Ergebnisse für Sequenzlänge 45	53
4	Übersicht der Ergebnisse	54
5	Varianz der Modelle bei Sequenzlänge 25	54
6	Varianz der Modelle bei Sequenzlänge 45	54
7	Standardabweichung der Modelle bei Sequenzlänge 25	54
8	Standardabweichung der Modelle bei Sequenzlänge 45	55
9	Friedman-Test Ergebnisse (seq. 25)	56
10	Friedman-Test Ergebnisse (seq. 45)	56
11	Nemenyi Post-hoc Test: F1-Score, Sequenzlänge 25	57
12	Nemenyi Post-hoc Test: MCC, Sequenzlänge 25	58
13	Nemenyi Post-hoc Test: Balanced Accuracy, Sequenzlänge 25	58
14	Nemenyi Post-hoc Test: F1-Score, Sequenzlänge 45	58
15	Nemenyi Post-hoc Test: MCC, Sequenzlänge 45	58
16	Nemenyi Post-hoc Test: Balanced Accuracy, Sequenzlänge 45	58
17	Ergebnisse von LSTM-AE-IF (Sequenzlänge 25, Testlauf 1)	74
18	Ergebnisse von LSTM-AE-OCSVM (Sequenzlänge 25, Testlauf 1)	74
19	Ergebnisse von LSTM-AE-DBSCAN(Sequenzlänge 25, Testlauf 1)	75
20	Ergebnisse von LSTM-AE-IF (Sequenzlänge 25, Testlauf 2)	75
21	Ergebnisse von LSTM-AE-OCSVM (Sequenzlänge 25, Testlauf 2)	75
22	Ergebnisse von LSTM-AE-DBSCAN (Sequenzlänge 25, Testlauf 2)	76
23	Ergebnisse von LSTM-AE-IF (Sequenzlänge 25, Testlauf 3)	76
24	Ergebnisse von LSTM-AE-OCSVM (Sequenzlänge 25, Testlauf 3)	76
25	Ergebnisse von LSTM-AE-DBSCAN (Sequenzlänge 25, Testlauf 3)	77
26	Ergebnisse von LSTM-AE-IF (Sequenzlänge 25, Testlauf 4)	77
27	Ergebnisse von LSTM-AE-OCSVM (Sequenzlänge 25, Testlauf 4)	77
28	Ergebnisse von LSTM-AE-DBSCAN (Sequenzlänge 25, Testlauf 4)	78
29	Ergebnisse von LSTM-AE-IF (Sequenzlänge 25, Testlauf 5)	78
30	Ergebnisse von LSTM-AE-OCSVM (Sequenzlänge 25, Testlauf 5)	78
31	Ergebnisse von LSTM-AE-DBSCAN (Sequenzlänge 25, Testlauf 5)	79
32	Ergebnisse von LSTM-AE-IF (Sequenzlänge 45, Testlauf 1)	80
33	Ergebnisse von LSTM-AE-OCSVM (Sequenzlänge 45, Testlauf 1)	80
34	Ergebnisse von LSTM-AE-DBSCAN (Sequenzlänge 45, Testlauf 1)	81
35	Ergebnisse von LSTM-AE-IF (Sequenzlänge 45, Testlauf 2)	81
36	Ergebnisse von LSTM-AE-OCSVM (Sequenzlänge 45, Testlauf 2)	81
37	Ergebnisse von LSTM-AE-DBSCAN (Sequenzlänge 45, Testlauf 2)	82

38	Ergebnisse von LSTM-AE-IF (Sequenzlänge 45, Testlauf 3)	82
39	Ergebnisse von LSTM-AE-OCSVM (Sequenzlänge 45, Testlauf 3)	82
40	Ergebnisse von LSTM-AE-DBSCAN (Sequenzlänge 45, Testlauf 3)	83
41	Ergebnisse von LSTM-AE-IF (Sequenzlänge 45, Testlauf 4)	83
42	Ergebnisse von LSTM-AE-OCSVM (Sequenzlänge 45, Testlauf 4)	83
43	Ergebnisse von LSTM-AE-DBSCAN (Sequenzlänge 45, Testlauf 4)	84
44	Ergebnisse von LSTM-AE-IF (Sequenzlänge 45, Testlauf 5)	84
45	Ergebnisse von LSTM-AE-OCSVM (Sequenzlänge 45, Testlauf 5)	84
46	Ergebnisse von LSTM-AE-DBSCAN (Sequenzlänge 45, Testlauf 5)	85

Abkürzungsverzeichnis

ABAC: Attribute-Based Access Control

AE: Autoencoder

AUC-PR: Area Under the Precision and Recall Curve

AUC-ROC: Area Under the ROC Curve

CVE: Common Vulnerability and Exposures

CWE: Common Weakness Enumeration

DBSCAN: Density-Based Spatial Clustering of Applications with Noise

DGX: Deep GPU Xceleration

IAM: Identity und Access Management

IdP: Identity Provider (Identitätsanbieter)

IF: Isolation Forest

LSTM: Long Short Term Memory

MCC: Matthews Correlation Coefficient

ML: Machine Learning

NaN: Not a Number

OCSVM: One Class Support Vector Machine

OIDC: OpenID Connect

RBAC: Role-Based Access Control

RNN: Recurrent Neural Network

SAML: Security Assertion Markup Language

SPI: Serial Peripheral Interface

SSO: Single-Sign-On

UBA: User Behavior Analytics

UEBA: User and Entity Behavior Analytics

Glossar

ν Parameter der One-Class SVM, der den Anteil an erwarteten Ausreißern im Trainingsdatensatz steuert. 51

ABAC Zugriffskontrolle basierend auf Benutzerattributen wie Rolle, Standort, Zeit etc. 24

Access Logs Protokolle von HTTP-Zugriffen und Statuscodes. Werden häufig auf Webserver- oder Proxy-Ebene erstellt, nicht direkt in Keycloak. 38

Account Sabotage Ein Szenario, in dem ein Benutzer absichtlich Ressourcen löscht, verändert oder auf sie zugreift, um einem System oder einem anderen Benutzer zu schaden. 62

Adam-Optimierer Ein adaptiver Optimierungsalgorithmus für neuronale Netze, der den Gradientenabstieg beschleunigt. Er kombiniert die Vorteile von AdaGrad und RMSProp, indem er für jeden Parameter individuelle Lernraten unter Berücksichtigung der ersten und zweiten Momentabschätzung berechnet. Besonders geeignet für große Modelle und Datenmengen. 50

Admin-Logs Logs, die administrative Operationen wie CREATE, UPDATE oder DELETE dokumentieren. Enthalten z. B. das Feld *operationType*. 38

AE Autoencoder – Neuronales Netzwerk zur komprimierten Darstellung und Rekonstruktion von Daten. 26

AUC-PR Area Under the Precision and Recall Curve – misst die Leistung eines Modells im Umgang mit unausgeglichene Klassen. 30

AUC-ROC Area Under the Receiver Operating Characteristic Curve – misst die Trennschärfe eines Klassifikationsmodells. 30

Audit-Logs Logs zur Nachvollziehbarkeit administrativer Änderungen. In Keycloak Teil der Admin-Logs. 38

Authentication Flow Ablauf, der festlegt, wie sich Benutzer oder Clients bei Keycloak authentifizieren. 35

Authorization Code Grant OAuth2-Grant-Typ, bei dem der Benutzer über einen Code authentifiziert wird, der vom Client gegen ein Token eingetauscht wird. 35

authType Ein Attribut in Keycloak-Logs, das angibt, über welchen Mechanismus sich ein Benutzer authentifiziert hat (z. B. password, token). 54

Balanced Accuracy Durchschnitt aus Sensitivität (Recall) und Spezifität; besonders geeignet bei unausgegleichen Klassenverhältnissen. 30

Batch-Size Die Anzahl an Trainingsbeispielen, die in einem Schritt (Iteration) gleichzeitig durch das neuronale Netz propagiert und zur Gradientenberechnung genutzt werden. Eine kleinere Batch Size führt zu häufigeren Gewichtsanpassungen, während eine größere Batch Size stabilere Gradienten liefert, aber mehr Speicher benötigt. 47

. 21

Bottleneck Engste Stelle eines Autoencoders, in der die komprimierte Datenrepräsentation gespeichert wird. 41

Brute-Force-Angriff Angriffsmethode, bei der Passwörter oder Zugangsdaten durch systematisches Ausprobieren aller möglichen Kombinationen erraten werden. 23

category Bezeichnung die Log-Art bzw. den Ereignistypen, z.B. LOGIN. 39

Client Anwendung oder Dienst, der Keycloak zur Authentifizierung und Autorisierung verwendet. 34

Client Credentials Grant OAuth2-Grant-Typ, bei dem sich ein Client ohne Benutzerinteraktion authentifiziert. 35

CLIENT LOGIN Event-Typ in Keycloak, der bei Anmeldung über einen Drittanbieter-Client (z. B. via OAuth2) generiert wird. 40

Client-Role Client-spezifische Rollen in Keycloak, die Zugriffsrechte innerhalb einer bestimmten Anwendung steuern. 36

CODE TO TOKEN Keycloak-Event, das auftritt, wenn ein Authentifizierungscode erfolgreich gegen ein Token eingetauscht wird. 40

CVE Ein öffentliches Verzeichnis bekannter Sicherheitslücken in Soft- und Hardware, das durch eindeutige Identifikationsnummern, Kurzbeschreibungen und betroffene Versionen strukturiert ist. 8, 61

CPunkte in DBSCAN, die keinem Cluster zugeordnet werden könnenWE Ein Katalog von typischen Schwachstellenmustern (Weaknesses), die potenziell zu Sicherheitslücken führen können, z. B. unzureichende Authentifizierung oder unsichere Verarbeitung von Daten. 8, 61

CRUD Abkürzung für Create, Read, Update, Delete; beschreibt die grundlegenden Operationen auf Datenbanken oder Entitäten in IT-Systemen. 35

Custom-Logs Benutzerspezifische oder erweiterte Logs, die über Keycloak-Extensions (z. B. SPIs) erzeugt werden. 38

DBSCAN Density-Based Spatial Clustering of Applications with Noise – Clustering-Verfahren für dichte Datenregionen. 26

Decoder Teil eines Autoencoders, der versucht, aus der komprimierten Repräsentation die Originaldaten zu rekonstruieren. 41

Denoising Autoencoder Autoencoder, der lernt, verrauschte Eingaben zu bereinigen und zu rekonstruieren. 42

DGX Deep GPU Xceleration – NVIDIA-Hardwareplattform zur KI-Beschleunigung. 7, 34

Dropout-Rate Der Anteil der Neuronen, die während des Trainings eines neuronalen Netzes zufällig deaktiviert (auf 0 gesetzt) werden, um Überanpassung (Overfitting) zu reduzieren. Eine Dropout-Rate von 0.1 bedeutet, dass 10% der Neuronen pro Trainingsschritt deaktiviert werden. 47

Encoder Teil eines Autoencoders, der Daten in eine niedrigdimensionale Repräsentation komprimiert. 41

Epochen Ein Begriff aus dem maschinellen Lernen, der die Anzahl der vollständigen Durchläufe über den gesamten Trainingsdatensatz während des Trainings eines neuronalen Netzes bezeichnet. Eine Epoche entspricht einem Zyklus, in dem jedes Trainingsbeispiel einmal verwendet wurde. 32

Eps Radius in DBSCAN, der zur Definition der Nachbarschaft eines Punktes dient. 27

Event-Logs Protokolle von Benutzer- und Administratoraktionen innerhalb von Keycloak. Enthalten Informationen zu Zeitpunkt, Nutzer-ID, Realm usw. 38

Feature Space (Merkmalsraum) Der Raum, in dem Datenpunkte durch Merkmale abgebildet werden, häufig hochdimensional. 44

γ Parameter der RBF-Kernel-Funktion bei der One-Class SVM; bestimmt, wie stark ein einzelner Trainingspunkt die Entscheidungsgrenze beeinflusst. Mathematisch tritt γ in der RBF-Kernel-Formel auf:

$$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right)$$

wobei x_i, x_j Trainingspunkte sind.. 51

Grant-Type Verfahren, mit dem ein Client oder Benutzer bei einem Authentifizierungsserver ein Token anfordert. 35

- HDBSCAN** Hierarchical Density-Based Spatial Clustering of Applications with Noise, eine hierarchische Erweiterung von DBSCAN. 45
- Hidden State** Interner Zustand eines RNNs, der Informationen über vorherige Zeitschritte speichert. 41
- Hybridmodell** Ein Modell, das zwei oder mehr unterschiedliche Verfahren kombiniert, z. B. LSTM-Autoencoder mit einem klassischen ML-Algorithmus wie Isolation Forest. 26
- IAM** Identity and Access Management – Verwaltung von Identitäten und Zugriffsrechten. 24
- Impersonate** Ein Event in Keycloak, bei dem ein Administrator die Rolle eines anderen Benutzers übernimmt; wird zur Fehlersuche oder zu Testzwecken verwendet. 61
- Insider Threat** Sicherheitsbedrohung, die von Personen innerhalb einer Organisation ausgeht, bspw. durch missbräuchliches Verhalten von Mitarbeitenden mit legitimen Zugriffsrechten. 23
- Isolation Tree** Binärer Baum aus dem Isolation Forest-Verfahren zur Isolierung von Datenpunkten. 43
- JBoss Logging** Logging-API aus dem JBoss-Ökosystem. Bietet eine Abstraktionsschicht über verschiedene Logging-Backends wie log4j, JUL oder slf4j. 36
- Jupyter Notebook** Eine interaktive Web-Anwendung, die es ermöglicht, Python-Code auszuführen, Ergebnisse direkt anzuzeigen, Text, Formeln und Visualisierungen zu kombinieren und somit exploratives Programmieren und dokumentiertes Data Science-Arbeiten zu erleichtern. 49
- K-Means** Clustering-Verfahren, das Daten in vordefinierte Cluster basierend auf Mittelwerten aufteilt. 46
- Keras** Eine hochgradig abstrahierende Deep-Learning-API in Python, die ursprünglich eigenständig entwickelt wurde und heute als offizieller Bestandteil von TensorFlow dient. Keras ermöglicht eine einfache und modulare Modellierung neuronaler Netzwerke. 33
- Keycloak** Open-Source Identity-Management-Lösung mit Funktionen wie SSO, RBAC und Brute-Force-Schutz. 7, 24
- Learning Rate** Ein Hyperparameter, der die Schrittgröße bei der Anpassung der Gewichte während des Trainings eines neuronalen Netzes bestimmt. Eine kleinere Lernrate führt zu langsameren, aber stabileren Anpassungen, während eine größere Lernrate schneller, aber instabiler sein kann. 47

LOF Local Outlier Factor, ein Verfahren zur Anomalieerkennung über Vergleich lokaler Dichten. 46

log_level Gibt die Kritikalität oder den Zweck eines Log-Eintrags an (z. B. INFO, ERROR, FATAL, DEBUG). 39

Log4j Beliebtes Java-Logging-Framework, bekannt durch seine Flexibilität. Wegen Sicherheitslücken (bspw. Log4Shell) kritisch betrachtet. 37

LOGIN ERROR Event-Typ, der bei einem fehlgeschlagenen Benutzer-Login generiert wird. 40

LSTM Long Short-Term Memory – Neuronales Netz zur Verarbeitung von Sequenzdaten mit Langzeitspeicher. 26

LSTM-Autoencoder Ein neuronales Netzwerk, das auf Long Short-Term Memory-Zellen basiert, um Sequenzen zu kodieren und zu rekonstruieren, oft zur Anomalieerkennung in Zeitreihen. 27

manage-users Keycloak-Rolle, die es erlaubt, Benutzer in einem Realm zu verwalten (z. B. anlegen, bearbeiten, löschen). 54

Matthews Correlation Coefficient (MCC) Eine robuste Metrik, die alle Werte der Konfusionsmatrix berücksichtigt; besonders geeignet bei unausgeglichene Datensätzen. 30

Mean Squared Error (MSE) Ein gängiges Fehlermaß zur Bewertung der Modellgüte bei kontinuierlichen Werten. Es berechnet die mittlere quadratische Differenz zwischen den vorhergesagten und den tatsächlichen Werten. Größere Abweichungen werden durch das Quadrieren stärker gewichtet, was das Modell zu präziser Vorhersage zwingt. 49

MinPts Minimale Anzahl an Punkten innerhalb des Eps-Radius, damit ein Punkt als Kernpunkt eines Clusters gilt. 27

ML Machine Learning – Teilgebiet der KI, bei dem Systeme aus Daten lernen. 25

Noise Punkte in DBSCAN, die keinem Cluster zugeordnet werden können. 45

OCSVM One-Class Support Vector Machine – Algorithmus zur Anomalieerkennung mit einseitiger Klassifikation. 26, 44

operationType Feld in Admin-Logs, das die Art der Admin-Aktion angibt, z. B. CREATE, UPDATE, DELETE oder ACTION. 40

Overfitting Überanpassung eines Modells an die Trainingsdaten, sodass die Generalisierungsfähigkeit auf unbekannte Daten stark eingeschränkt ist. Typische Indikatoren sind stark abweichende Trainings- und Validierungsfehler. Overfitting tritt häufig bei komplexen Modellen mit zu vielen Parametern oder bei kleinen, nicht repräsentativen Datensätzen auf. 26

Privilege Escalation Ein Angriffsszenario, bei dem sich ein Benutzer unbefugt höhere Rechte verschafft, z. B. um auf administrative Funktionen zuzugreifen. 62

Quarkus Modernes Java-Framework, optimiert für containerisierte und cloud-native Anwendungen. Ab Keycloak Version 17 als Basisplattform genutzt. 36

RBAC Role-Based Access Control – Zugriffskontrolle auf Basis definierter Rollen. 24

Realm Isolierte Umgebung in Keycloak, in der Benutzer, Rollen und Clients separat verwaltet werden. 34

realm-admin Eine Administratorrolle in Keycloak, die vollständige administrative Rechte innerhalb eines bestimmten Realms gewährt. 36, 54

Realm-Role Globale Rollen in Keycloak, die auf alle Clients eines Realms angewendet werden können. 36

. 59

REFRESH_TOKEN Keycloak-Event-Typ, der beim Aktualisieren eines Zugriffstokens generiert wird. 39

. 48

Rekonstruktionsfehler Differenz zwischen Originaldaten und den durch einen Autoencoder rekonstruierten Daten. 41

Residual Block Ein architektonisches Element in tiefen neuronalen Netzwerken, das eine direkte Identitätsverbindung (Shortcut Connection) zwischen Eingabe und Ausgabe eines Layers herstellt. Dadurch lernt das Netzwerk nicht die gesamte Transformation, sondern nur die Differenz (*Residuum*) zur Eingabe. Dies erleichtert die backpropagation und reduziert das Vanishing-Gradient-Problem, wodurch tiefere Netze stabiler trainiert werden können. 53

RNN Recurrent Neural Network – Neuronales Netz zur Verarbeitung zeitabhängiger Daten. 33, 41

Scikit-learn Eine weit verbreitete Python-Bibliothek für maschinelles Lernen mit Werkzeugen für Klassifikation, Regression, Clustering und Modellbewertung. 7, 33

Security Logs Protokollieren sicherheitsrelevante Ereignisse wie z. B. verdächtige Logins, Policy-Verletzungen oder Blockierungen. 38

Selenium Ein Framework zur Automatisierung von Webbrowser-Interaktionen, häufig für Testzwecke genutzt. 56

Sequenzlänge Die Anzahl aufeinanderfolgender Datenpunkte oder Log-Einträge, die gleichzeitig als Eingabe für ein Modell wie ein LSTM-Autoencoder verwendet werden. Die Wahl der Sequenzlänge beeinflusst, wie viel zeitlicher Kontext das Modell berücksichtigt und kann sich auf die Erkennungsrate von Anomalien auswirken. 27

Server-Logs Logs, die Betriebsstatus, Fehler und Debugging-Informationen des Keycloak-Servers dokumentieren. 38

Service Account Konto, das einem Client gehört und ohne Benutzerinteraktion für automatisierte Vorgänge eingesetzt wird. 36

SLF4J Simple Logging Facade for Java – eine Logging-Fassade, die unterschiedliche Logging-Backends wie log4j unterstützt. 37

SPI SPI (Service Provider Interface) ist ein Programmier-Schnittstellenkonzept. 38

Spoofing Täuschungstechniken, bei denen sich ein Angreifer als vertrauenswürdige Instanz ausgibt, um Zugang zu Informationen oder Systemen zu erlangen. 23

SSO Single Sign-On – Einmalige Authentifizierung für den Zugriff auf mehrere Anwendungen. 24

Syslog Standardprotokoll zur Übertragung von Logmeldungen in Netzwerkkumgebungen, z. B. an zentrale Logserver. 37

TensorFlow Ein Open-Source-Framework für maschinelles Lernen, entwickelt von Google. TensorFlow bietet eine flexible Infrastruktur zur Definition, Optimierung und Ausführung von Rechenmodellen, insbesondere neuronalen Netzwerken. 7, 34

timestamp Zeitmarke, zu der ein Event im Log registriert wurde. Wichtige Information für zeitbasierte Analysen. 39

UBA User Behavior Analytics – Analyse des Benutzerverhaltens zur Erkennung von Anomalien und Sicherheitsbedrohungen. 25

UEBA User and Entity Behavior Analytics – Verfahren zur Analyse des Verhaltens von Benutzern und technischen Entitäten mit dem Ziel, sicherheitsrelevante Anomalien zu identifizieren. 25

Vanishing Gradient Ein Problem beim Backpropagation-Algorithmus, bei dem die Gradienten beim Durchlaufen vieler Schichten oder Zeitschritte exponentiell kleiner werden. Dies führt dazu, dass frühe Schichten kaum lernen, da ihre Gewichtsanpassungen gegen null gehen. Besonders relevant bei rekurrenten Netzen (RNNs). Lösungsansätze umfassen LSTM-Zellen, Residual-Verbindungen oder geeignete Aktivierungsfunktionen.

41

Variational Autoencoder (VAE) Autoencoder-Variante, die Wahrscheinlichkeitsverteilungen in der Bottleneck-Schicht modelliert, häufig für generative Modelle. 42

view-events Keycloak-Rolle, die es erlaubt, Event-Logs eines Realms einzusehen. 59

WebDriver Die Komponente von Selenium, die es ermöglicht, Browser programmgesteuert zu steuern. 57

WildFly Java-Anwendungsserver, früher Basis von Keycloak. Verwendet JBoss Logging und bietet umfangreiche Java EE-Funktionalität. 36

Zero-Day-Angriff Angriff auf eine bislang unbekannte Sicherheitslücke, für die noch kein Patch existiert. 24

1 Einleitung

Laut dem Lagebericht des Bundesamtes für Sicherheit in der Informationstechnik (BSI) 2023 hat die Zahl von Cyberangriffen auf Unternehmen ein bislang nie dagewesenes Niveau erreicht [1]. Angreifende entwickeln fortlaufend neue Techniken, um bestehende Sicherheitsmechanismen zu umgehen und wirtschaftliche oder strategische Vorteile zu erzielen. Vor diesem Hintergrund stellt sich die zentrale Frage: Wie können Unternehmen potenziell gefährliche Angreifer frühzeitig erkennen, um Schäden effektiv zu verhindern? Die Beantwortung dieser Frage ist essenziell, um die Widerstandsfähigkeit von Informationssystemen gegenüber stetig wachsenden Bedrohungen zu erhöhen.

1.1 Benutzerverhalten und Anomalien

Das Benutzerverhalten umfasst sämtliche Aktionen, Interaktionsmuster und Gewohnheiten einer Benutzers in IT-Systemen. Die Analyse dieser Verhaltensweisen ermöglicht die Identifikation typischer Nutzungsmuster sowie die Erkennung von Abweichungen, die potenziell auf Sicherheitsvorfälle, unautorisierte Zugriffe oder Missbrauch hinweisen können.

Anomalien bezeichnen Datenmuster, die signifikant von erwarteten oder üblichen Nutzungsmuster abweichen. Mathematisch gesehen, ist eine Anomalie, ein abweichender Punkt der "normalen" Bereichen N_1 und N_2 [2, S.2].

Neben systemweiten Anomalien – etwa in Netzwerkverkehr oder Gerätezuständen – rücken zunehmend verhaltensbezogene Auffälligkeiten in den Fokus der Forschung. Im Rahmen dieser Arbeit werden Anomalien, die das Benutzerverhalten betreffen, gezielt als *verhaltensbezogene Anomalien* bezeichnet. Sie charakterisieren das individuelle Nutzungsmuster einzelner Anwender innerhalb einer Anwendung oder eines IT-Systems und ermöglichen die Identifikation von Abweichungen, die auf Sicherheitsvorfälle oder Missbrauch hindeuten können.

1.2 Kategorisierung von Anomalien

Nach Chandola et Al. [2, S.7] werden Anomalien in drei Hauptkategorien eingeteilt:

- **Punkt-Anomalien:** Einzelne Dateninstanzen, die im Vergleich zum Großteil der Daten als anomal gelten. Beispiel: Eine Kreditkartentransaktion, deren Betrag deutlich höher ist als die sonst üblichen Ausgaben einer Person.
- **Kontextuelle Anomalien:** Dateninstanzen, die nur in einem spezifischen Kontext als anomal betrachtet werden, ansonsten jedoch normal sind.
- **Kollektive Anomalien:** Gruppen von Datenpunkten, die zusammen eine Anomalie darstellen, obwohl einzelne Punkte für sich genommen normal erscheinen.

Dabei umfassen kontextuelle Anomalien zwei Arten von Attributen:

1. **Kontextuelle Attribute:** Bestimmen den Kontext, z. B. geografische Lage oder Zeitstempel.
2. **Verhaltensbezogene Attribute:** Beschreiben das tatsächliche Verhalten oder die Charakteristik der Dateninstanz, z. B. die gemessene Temperatur oder der Betrag einer Transaktion.

Nach Foorthuis et al. werden Anomalien entsprechend ihrer Merkmale unterschiedlich kategorisiert; sie unterscheiden dabei neun Übertypen [3], wobei die Klassifikation auf mathematischen Kriterien basiert. Sanni weist hingegen darauf hin, dass Anomalien im Kontext des Benutzerverhaltens als Abweichungen von üblichen Verhaltensmustern interpretiert werden können [4, S.3]. Damit wird deutlich, dass es sowohl mathematische Kategorisierungen als auch begriffsbasierte, auf reale Anwendungen übertragbare Klassifikationen gibt. Aufgrund der unscharfen Definition des Anomaliebegriffs, der auch außerhalb der Informatik für unterschiedliche Konzepte verwendet wird, ist es erforderlich, für die vorliegende Arbeit eine klar operationalisierte Definition festzulegen.

Im Kontext dieser Arbeit wird definiert, dass Anomalien welche sich im Benutzerverhalten äußern, bspw. unter anderem folgende sind:

- ungewöhnliche Login-Zeiten,
- verdächtige Standorte,
- auffällige Rollenprofile,
- unübliche Zugriffsrechte,
- eine Häufung von Fehlern in der Authentifizierung oder Autorisierung,
- Ungewöhnliche IP-Adresse.

Ein Benutzer, der sich mitten in der Nacht außerhalb der üblichen Geschäftszeiten einloggt, zeigt ein ungewöhnliches Verhalten und weicht somit vom normalen Nutzungsmuster ab. Ebenfalls wird davon ausgegangen, dass die meisten Personen eher in der Region arbeiten, wo sie wohnen¹. Moderne Szenarien, wie Homeoffice oder flexible Arbeitszeiten, müssen jedoch berücksichtigt werden: So kann es vorkommen, dass Personen weiter entfernt wohnen oder Entwickler bis spät in die Nacht an einem Projekt arbeiten. Wenn solche Fälle im Unternehmen üblich sind, stellen sie keine Anomalien dar, sondern den Standardbetrieb. Unübliche Zugriffsrechte auf sensible Daten können dagegen Anomalien darstellen, insbesondere wenn ein Benutzer eigentlich keinen Zugriff auf bestimmte Ressourcen haben sollte oder plötzlich mehrere Rollen erhält, die ihm zu viele Rechte einräumen. Häufige Anmelde-

oder Autorisierungsfehler können ebenfalls auf Brute-Force-Angriffe oder andere Intrusions-Angriffe-Versuche hinweisen.

Komplexe Abweichungen – etwa Veränderungen in gewohnten Arbeitsabläufen oder zeitlichen Mustern von Datei- und Netzwerkzugriffen – gelten ebenfalls als Anomalien. Ungewöhnliche IP-Adressen können auf unautorisierten Zugriff, Spoofing-Attacken oder die Nutzung anonymisierender Dienste (z.B. VPN, TOR) hinweisen. Im regulären Betrieb sind IP-Adressen meist geografisch, organisatorisch oder netztechnisch eingeschränkt. Weichen Zugriffe deutlich von bekannten Mustern oder erlaubten Adressräumen ab, kann dies auf externe Angreifer hindeuten. Im Nutzerkontext sind IP-Auffälligkeiten oft ein erstes Signal für Insider-Threat-Attacks oder gestohlene Zugangsdaten.

Einige aufgezählten Aspekte sind in *Intrusions-Angriffen* und *Insider Threat-Attacks* vorhanden. Intrusionen sind von außen kommende Angriffe, welche versuchen in das Innere eines Systems zu gelangen, z.B. Brute-Force- Angriffe. Insider-Threat-Attacken können im System selbst schon auftreten, bspw. könnte der Angreifer ein Mitarbeiter im Unternehmen selbst sein.

Insider-Threat-Attacks äußern sich nach Legg et Al. [5, S.2] in den genannten Aspekten:

- Benutzer zeigt **abweichendes Verhalten**, z. B.: loggt sich deutlich früher ein als üblich
- Benutzer zeigt **bestimmtes Verhalten übermäßig häufig**, z. B. viele Dateien auf einmal herunterladen
- Benutzer weist **neue Attribute oder Aktionen** auf, die zuvor nicht beobachtet wurden, was auf eine **Insider-Attacke** hindeuten kann

In dieser Arbeit wird der Begriff verhaltensbezogene Anomalien mit den Erscheinungsformen von Intrusionen und Insider-Threat-Attacks gleichgesetzt, um den Fokus auf sicherheitsrelevante Abweichungen im Nutzerverhalten zu legen. Diese definitorische Vereinfachung ermöglicht eine klare Fokussierung auf sicherheitsrelevante Abweichungen im Nutzerverhalten. Es sei jedoch darauf hingewiesen, dass Anomalien technisch betrachtet als Indikatoren für unterschiedliche Angriffstypen interpretiert werden können.

1.3 Regelbasierte Methoden und weitere Maßnahmen in Keycloak

Das Unternehmen intension nutzt das Produkt Keycloak ². Dies ist ein IAM-Tool³, das unter anderem SSO (Single Sign-On) bietet⁴.

Keycloak stellt vorgefertigte Sicherheitsmechanismen zur Verfügung, die größtenteils regelbasiert arbeiten. Solche Maßnahmen definieren explizit, welche Aktivitäten im System erlaubt oder verboten sind. Grundlage dieser Regeln sind meist bekannte Angriffsmuster aus

früheren Sicherheitsvorfällen. Zusätzlich bietet Keycloak klassische Zugriffskontrollmechanismen wie RBAC und ABAC, Brute-Force-Erkennung sowie die Möglichkeit zur Integration externer Netzwerkschutzsysteme wie Firewalls.

1.4 Grenzen der Sicherheitsmaßnahmen in Keycloak

Zwar bietet Keycloak regelbasierte Sicherheitsmaßnahmen an- diese beruhen jedoch größtenteils auf vordefinierten Bedingungen.

Waltl et al. (2017) betonen bspw., dass "regelbasierte Informationsextraktionssysteme (IE) zwar Vorteile wie Transparenz und Nachvollziehbarkeit bieten, jedoch auch mehrere Einschränkungen aufweisen. Ein wesentlicher Nachteil ist der hohe manuelle Aufwand bei der Erstellung und Pflege der Regeln. Dies führt zu einer begrenzten Flexibilität, da neue oder komplexe Angriffstechniken nur schwer abgebildet werden können. Ein weiteres Problem ist die mangelnde Interoperabilität der Regelsprachen, was die Integration in heterogene Systeme erschwert. Zudem erfordern regelbasierte Systeme oft eine hohe Anzahl an Regeln, um auch geringfügige Variationen in den Eingabedaten zu erfassen, was die Wartung und Erweiterung der Systeme erschwert. Diese Einschränkungen machen regelbasierte Ansätze insbesondere in dynamischen und komplexen Anwendungsbereichen wie der Sicherheitsanalyse weniger geeignet.

Neue, bislang unbekannte Angriffsarten (sogenannte Zero-Day-Angriffe) entziehen sich dieser Logik, da keine entsprechenden Regeln vorliegen. Angreifende nutzen gezielt sogenannte Adversarial Attacks, um diese vorhersehbaren Schutzmechanismen zu umgehen. Da die Regeln dieser Systeme häufig offen dokumentiert sind, lassen sich entsprechende Gegenstrategien eher entwickeln.

Ein weiteres Beispiel ist der integrierte Brute-Force-Schutz von Keycloak, der jedoch nachweislich umgangen werden kann – etwa durch sogenannte *Timing-Angriffe*⁵. Dabei analysiert ein Angreifer die Antwortzeit des Systems, um Rückschlüsse auf den Authentifizierungsprozess zu ziehen. Wird beispielsweise ein korrekter Benutzername, aber ein (fast) korrektes Passwort eingegeben, kann die Antwortzeit minimal verlängert sein – was auf die sequentielle Überprüfung der Passwortzeichen hindeutet. Solche zeitlichen Unterschiede im Millisekundenbereich lassen sich mit ausreichend vielen Versuchen systematisch auswerten. Auch die Zugriffskontrollmodelle zeigen Schwächen: RBAC wird bei wachsender Komplexität und Vielzahl an Rollen schnell unübersichtlich, während ABAC zwar flexibler ist, dafür jedoch hohen Pflegeaufwand verursacht und anfällig für Fehler ist – insbesondere, wenn viele Attribute aktuell gehalten werden müssen. Zudem lassen sich regelbasierte Systeme durch Taktiken umgehen, die nicht explizit durch die Regeln abgedeckt sind, beispielsweise Social-Engineering-Angriffe. So könnte ein Angreifer versuchen, administrative Rechte zu erlangen, ohne dass das Verhalten gegen eine festgelegte Regel verstößt. Maschinelle Lernmethoden hingegen sind in der Lage, statistische Abweichungen im Verhalten zu erkennen und somit

potenziell anomale Aktionen aufzudecken, da sie Wahrscheinlichkeiten und Muster bewerten, anstatt ausschließlich deterministischen Regeln zu folgen.

Obwohl für viele Schwachstellen bereits Gegenmaßnahmen existieren, erfordert deren Erkennung und Behebung stets Zeit. Neue Verwundbarkeiten und Sicherheitslücken entstehen kontinuierlich – etwa die Möglichkeit eine Zertifikatsprüfung zu umgehen ⁶, was jedoch primär netzwerkbasierte Sicherheitslücken betrifft und damit außerhalb des Fokus dieser Arbeit liegt. Diese Beispiele verdeutlichen die aktuellen Grenzen der in Keycloak implementierten Schutzmechanismen.

Diese Einschränkungen zeigen, dass regelbasierte Sicherheitsarchitekturen bei der Erkennung komplexer oder neuartiger Bedrohungen an ihre methodischen Grenzen stoßen. Zur Erhöhung der Erkennungsfähigkeit und Flexibilität können daher dynamische, lernfähige Systeme, wie sie durch maschinelles Lernen realisiert werden, eingesetzt werden. Dies bedeutet nicht, dass regelbasierte Methoden an sich unwirksam sind, sondern dass bestehende Sicherheitsansätze ergänzt werden können, um Systeme wie Keycloak weiter abzusichern.

1.5 Maschinelle Lernmodelle als zukünftige Sicherheitsmaßnahme

Die Motivation dieser Arbeit besteht darin, geeignete, hybride, maschinelle Lernmodelle zu identifizieren, welche verhaltensbezogenen Anomalien in den Log-Einträgen (Logs) aus Keycloak erkannt werden können. Maschinelle Lernmodelle stellen heute eine vielversprechende Ergänzung zu klassischen, regelbasierten Sicherheitsmaßnahmen dar, da sie dynamisch auf neue Bedrohungen reagieren können.

Im Gegensatz zu statischen Regeln, die lediglich bekannte Muster erfassen, sind ML-Modelle in der Lage, aus großen Datenmengen zu lernen und auch unbekannte Anomalien zu identifizieren, die auf potenzielle Angriffe hindeuten. Dadurch sind sie besonders effektiv im Umgang mit Zero-Day-Angriffen oder komplexen, sich ständig wandelnden Bedrohungsszenarien [6]. Ein weiterer Vorteil ist die Fähigkeit zur kontinuierlichen Anpassung, sodass sich ML-Modelle automatisch auf veränderte Rahmenbedingungen einstellen, ohne dass ein manueller Eingriff notwendig ist.

Im Ausblick besteht Interesse daran, das leistungsfähigste Modell in einer Softwarelösung zu verwenden. Zudem spart man Kosten ein, da herkömmliche maschinelle Analysetools häufig mit hohen Lizenzgebühren verbunden sind. Aufgrund datenschutzrechtlicher Vorgaben wurde entschieden, die Keycloak-Logs selbst zu generieren. Dies wird in den Kapiteln 8 und 9 näher erläutert.

1.6 User (and Entity) Behavior Analytics (U(E)BA)

UEBA ist ein ganzheitlicher Ansatz zur Gewährleistung einer erstklassigen Sicherheit in einem Unternehmen und zur Erkennung von Benutzern, die eine Sicherheitsverletzung im System verursachen könnten. UEBA kann normales und abnormales Verhalten sowohl von Menschen als auch von Computern identifizieren. Die Analyse erfolgt dabei häufig unter Einsatz maschineller Lernverfahren.

User and Entity Behavior Analytics (UEBA) stellt eine Weiterentwicklung von User Behavior Analytics (UBA) dar. Während UBA sich ausschließlich auf das Verhalten von Benutzern konzentriert, berücksichtigt UEBA zusätzlich auch Entitäten wie Server, Anwendungen oder Geräte sowie deren Interaktionen mit den Nutzern.

Da sich diese Arbeit auf Keycloak und das Benutzerverhalten im Rahmen von Authentifizierungsprozessen konzentriert, erfolgt keine vollständige Berücksichtigung aller Entitäten oder ihrer Netzwerkbeziehungen.

Streng genommen handelt es sich in dieser Arbeit nicht um eine Echtzeit-UBA-Anwendung, da die Auswertung auf bereits gesammelten Logdaten erfolgt. Für die Umsetzung eines Hybridmodells in einer produktiven Softwarelösung würde diese Methodik jedoch den Kern einer UBA-Lösung darstellen.

2 Forschungsstand, Herausforderungen und Hypothesen

2.1 Bisheriger Forschungsstand

Die Literatur legt nahe, dass klassische Verfahren wie Isolation Forest (IF) besonders effektiv in der Anomalieerkennung sind [7]. Verfahren wie One-Class SVM (OCSVM) erkennen Anomalien teilweise sogar präziser als andere Modelle [8]. Auch DBSCAN wird erfolgreich für die Cluster-basierte Anomalieerkennung eingesetzt, wobei "multiple Parameter für Optimierung verwendet werden [9].

Für sequenzielle Daten wie Zeitreihen oder Nutzerinteraktionen zeigen neuronale Netze, insbesondere Long Short-Term Memory (LSTM) und Autoencoder (AE), signifikante Vorteile, da sie zeitliche Abhängigkeiten und komplexe Muster besser erfassen können und teilweise andere Modelle wie TimeGPT übertreffen [10]. Zur Erkennung von Intrusionen und "benutzerbasierten Anomalien eignet sich ein LSTM-AE gut [11].

Hybridmodelle, also Ansätze, bei denen mehrere Algorithmen und/oder neuronale Netze kombiniert werden, zeigen ebenfalls vielversprechende Ergebnisse:

Aus der Literatur ergibt sich, dass Kombinationen aus LSTM und Autoencoder als besonders robust und effektiv gelten – etwa in Szenarien zur Luftqualitätsüberwachung, mit einer Genauigkeit von 99,5 % [12].

Andere Studien zeigen, dass sich LSTM-AE in Kombination mit IF zu einem leistungsfähigen Modell zur Anomalieerkennung entwickeln lässt, das ebenfalls hohe Genauigkeiten erzielt [13]. Weitere Arbeiten, wie die von Ghrib et al. [14], untersuchen hybride Ansätze unter Verwendung von IF: Es hat eine geringere Rechenkomplexität. Zudem weisen Hybridmodelle mit IF eine hohe Effektivität bei der Erkennung von Intrusionen auf [15].

Eine Studie von Ergen et al. [16] zeigt, dass die Kombination von LSTM-basierten Architekturen mit OCSVM zu signifikant verbesserten Erkennungsraten bei der Anomalieerkennung führt – insbesondere bei sequenziellen Daten. In einer weiteren Studie wird LSTM mit AE und OCSVM kombiniert, und Explainable AI wird genutzt, um Intrusionen zu erkennen, wobei ebenfalls sehr gute Ergebnisse erzielt werden [17].

Zhou et al. zeigen, dass im IoT-Bereich eine Kombination von LSTM-AE und DBSCAN unnormale Temperaturen deutlich besser erkennen kann [18]. Eine weitere Studie demonstriert, dass auch DBSCAN in Kombination mit AE Intrusionen sehr effektiv erkennen kann [19].

Die Literatur zeigt jedoch auch potenzielle Schwächen der genannten Netzwerke und Modelle. LSTM- und Autoencoder-Modelle sind anfällig für Overfitting und benötigen große Datenmengen [20].

Isolation Forest und OCSVM sind oft sensitiv gegenüber der Wahl der Parameter und muss je nach Architektur angepasst werden, wie auch in einer Studie nach Chua et Al. diskutiert wurde [21]. DBSCAN ist ebenfalls sehr empfindlich gegenüber der Wahl von Parametern wie Eps und MinPts, erkennt Cluster mit unterschiedlicher Dichte nur schlecht und kann bei hohem Rauschen versagen [22].

Hybridmodelle haben ebenfalls ihre Schwächen: Zwar verbessern sie häufig die Leistung, gleichzeitig erschwert ihre Komplexität die Interpretierbarkeit und den Einsatz in Echtzeitanwendungen [23].

Insgesamt zeigt sich, dass hybride Ansätze dieser Verfahren dennoch vorteilhaft sind, um die Herausforderungen bei der Erkennung verhaltensbezogener Anomalien in realen Anwendungen effizient zu bewältigen. Die betrachteten Arbeiten betonen zudem unterschiedliche Anomalie-Typen, für die jeweiligen Modelle eingesetzt werden.

2.2 Hypothese

Um die Forschung zu vertiefen, soll gezielt untersucht werden, wie verschiedene Hybridmodelle zur Erkennung verhaltensbezogener Anomalien, Intrusionen und Insider-Threat-Angriffe eingesetzt werden können.

Basierend auf der Literatur besteht daher Interesse an einem umfassenden Vergleich zwischen Hybridmodellen aus LSTM-AE und den weiteren genannten klassischen maschinellen Lernalgorithmen sowie an einem Vergleich dieser Modelle einzeln.

Daraus wurde folgende Hypothese abgeleitet:

Es wird erwartet, dass der LSTM-Autoencoder in Kombination mit Isolation Forest unter unterschiedlichen Sequenzlängen höhere Ergebnisse in den Metriken F1-Score, Balanced Accuracy und MCC erzielt als die Kombinationen des LSTM-Autoencoders mit DBSCAN bzw. OCSVM.

Die zugehörige Nullhypothese lautet:

Unter verschiedenen Sequenzlängen zeigt der LSTM-Autoencoder in Kombination mit Isolation Forest keine höheren Ergebnisse in den Metriken F1-Score, Balanced Accuracy und MCC als die Kombinationen mit DBSCAN oder OCSVM.

Folgende Hybridmodelle werden also implementiert:

- LSTM-AE und Isolation Forest
- LSTM-AE und One-Class SVM
- LSTM-AE und DBSCAN

Dabei wird ein Keycloak-Datensatz verwendet, welches von den Modellen analysiert wird.

Die Sequenzlänge wird als unabhängige Variable untersucht, da sie die Stabilität und Genauigkeit der Modellvorhersagen beeinflussen kann - der Fokus auf dem besten Modell in dieser Arbeit zeigt sich durch seine Stabilität und Genauigkeit. Mit verschiedenen Metriken wird geprüft, welches Modell hinsichtlich Klassifikationsgenauigkeit und Konsistenz die besten Ergebnisse erzielt.

Unter Konsistenz wird verstanden, welches Modell nicht zufällig, sondern anhand eines stabilen Algorithmus klassifiziert- die Bewertung dieser Eigenschaft erfolgt durch die Messung des Matthews Correlation Coefficient (MCC).

Die Klassifikationsgenauigkeit gibt an, wie oft korrekt Anomalien erkannt wurden und weist auf eine geringe False-Positive und False-Negative Rate hin, die sich mit dem F1-Score und auch der Balanced Accuracy messen lässt. Auf die genannten Metriken wird in Kapitel 3 nochmals genauer eingegangen.

Die Hypothese, dass LSTM-Autoencoder (LSTM-AE) in Kombination mit Isolation Forest (IF) besonders effektiv ist, stützt sich auf die komplementären Stärken beider Methoden. LSTM-AE kann zeitabhängige Muster im Benutzerverhalten erfassen und Unterschiede zwischen normalem und anomalem Verhalten über Rekonstruktionsfehler erkennen. Zudem wird angenommen, dass Isolation Forest als einzelnes Modell effizienter Anomalien erkennen konnte als OCSVM und DBSCAN [24]. Ziel ist es, zu prüfen, ob diese Robustheit auch bestehen bleibt, wenn Isolation Forest mit einem LSTM-AE kombiniert wird- und unter welchen Sequenzlängen.

Zusätzlich wurden die Einzelmodelle Isolation Forest, OCSVM und DBSCAN auf den Keycloak-Daten explorativ angewendet. Diese Evaluation ist nicht Bestandteil der zentralen Vergleichsstudie, dient jedoch als Ergänzung zur Einordnung der Ergebnisse der Hybridmodelle.

Sie ermöglicht einen informellen Eindruck davon, wie sich gängige Einzelverfahren im gegebenen Kontext verhalten, ohne dass daraus direkte Schlussfolgerungen im Sinne der zentralen Forschungsfrage abgeleitet werden. Zudem kann so geprüft werden, ob komplexere Architekturen eventuell auch Nachteile haben und einfache Modelle unter bestimmten Bedingungen überlegen sind.

Die Kombination von Hybridmodellen mit zwei verbundenen Netzwerken und drei klassischen Lernalgorithmen bietet einen interessanten Vergleich, auch wenn der Vergleich zwischen den Modellen als unausgeglichen wahrgenommen werden kann. Es wird jeweils das Hybridmodell LSTM-AE mit einem der drei Lernalgorithmen kombiniert und miteinander verglichen.

Tran et al. zeigen beispielsweise bereits, dass ein Vergleich von LSTM-AE mit OCSVM bspw. durchgeführt wurde und dass das Hybridmodell LSTM-AE Anomalien präziser einteilen konnte [25]. Daher erscheint ein erweiterter Vergleich zwischen komplexeren Hybridmodellen, die noch klassische Lernalgorithmen verwenden, und den einfachen Algorithmen sinnvoll.

Zudem werden komplexere Daten verwendet, nämlich Logdaten, die Aufschluss über das Benutzerverhalten geben und keine vorgefertigten Datensätze, wie sie in den erwähnten Studien beschrieben sind.

Der Vergleich trägt somit dazu bei, bisherige Studien zu ergänzen und einen neuen wissenschaftlichen Mehrwert zu schaffen, indem komplexe Hybridmodelle miteinander verglichen werden.

3 Methoden und Störfaktoren

3.1 Metriken

Typischen Verfahren wie die Berechnung der Precision, des Recall und F1-Score werden angewandt. Die Metriken folgen alle unterschiedlichen:

Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$ gibt den Anteil korrekt klassifizierter Beispiele an allen Beispielen an.

Precision = $\frac{TP}{TP+FP}$ misst den Anteil korrekt als positiv klassifizierter Beispiele an allen als positiv vorhergesagten.

Recall = $\frac{TP}{TP+FN}$ zeigt, wie viele der tatsächlichen Positiven korrekt erkannt wurden.

F1-Score = $2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$ ist das harmonische Mittel von Precision und Recall.

Dabei stehen TP, TN, FP, FN für True-Positives, True-Negatives, False-Positives und False-Negatives.

Bei der Auswahl der Metriken muss man einige bekannte Probleme betrachten:

Eine Studie weist aber daraufhin, dass das Öfteren diese Metrik angewandt wird und oft nicht valide Begründungen für die Klassifikationsgenauigkeit des Modells bietet. U.a. wies diese Studie daraufhin, dass ein Großteil, der Studien nur anhand der Accuracy der maschinellen Lernmodelle ihre Effizienz misst und andere Eigenschaften ausblenden [26].

Zudem wird hingewiesen, dass durch stark unausgewogenen Klassen die Accuracy irreführend ist. Dies liegt daran, weil die Accuracy nur angibt, wie viele Datenpunkte richtig klassifiziert wurden. Wenn aber die Klassen unausgeglichen sind (z.B. 99 % der Datenpunkte gehören Klasse A der Rest zu Klasse B), ist es wahrscheinlicher, dass das Modell sich immer nur entweder für Klasse A oder Klasse B entscheidet- da es mehr Objekte von Klasse A gibt- diese dahin zu klassifizieren, obwohl es noch eine kleinere Klasse B gab, die aber nicht berücksichtigt wurde. Es wird dann zwar eine hohe gute Accuracy von 99 % erreicht, was aber bedeutet, dass 1 %, die von Klasse B ebenfalls zu Klasse A zugeordnet wurden. Dies nennt man auch Accuracy Paradoxon. Um dieses Paradoxon zu vermeiden, will man in dieser Arbeit Metriken einsetzen, welche ausschlaggebender sind.

Dies bedeutet nicht, dass die Accuracy keine Aussagekraft bei der Bewertung der Modelle besitzt. Vielmehr ist sie eine häufig verwendete Metrik, während andere, effizientere Bewertungsmethoden, die sich insbesondere für unausgeglichene Datensätze eignen, oftmals unberücksichtigt bleiben. Die gleichzeitige Verwendung einer Vielzahl weiterer Metriken wäre in diesem Kontext jedoch übermäßig und würde den Fokus der Analyse verwässern.

Es muss ebenfalls erwartet werden, dass es in dieser Arbeit zu einem Klassenungleichgewicht kommen kann, da Anomalien in den Testdaten sehr wenig auftreten, wie es in der Realität erwartet wird. Die Literatur zeigt, dass die Auswirkungen dieses Ungleichgewichts relativ sind und je nach eingesetztem Lernalgorithmus und Datensituation variieren [27].

Fourure et al. weisen darauf hin, dass der Fokus in vielen Studien häufig auf dem F1-Score liegt. Dieser ist jedoch stark abhängig von der Wahl des Evaluationsprotokolls, etwa der Kontaminationsrate oder des Schwellenwerts, wodurch er künstlich erhöht werden kann und keine tiefgehenden oder verlässlichen Aussagen über die Modellleistung liefert [28]. Es wird kritisiert, dass dieser zwar eine gute Gewichtung zwischen Precision und Recall zeigt, jedoch an sich nicht genau angeben kann, ob nun wirkliche das Modell Probleme in False-Positive-Klassifizierungen oder False-Negative-Klassifizierungen hat.

Eine weitere Untersuchung von Fourure et al. zeigt, dass in vielen Vergleichsstudien auch wie

zuvor erwähnt die Accuracy oder dem F1-Score betrachtet wird, während andere wichtige Metriken oft vernachlässigt bleiben. Zudem zeigen Sie, dass durch die Auswahl bestimmter Trainings- und Testdatensätze kann der F1-Score künstlich erhöht werden, was zu einer verzerrten Einschätzung der Modellleistung führt [28].

Insgesamt lässt sich festhalten, dass jede Metrik individuelle Stärken und Schwächen besitzt. Gleichzeitig wird kritisiert, dass viele Studien auf Metriken zurückgreifen, die nicht ausreichend aussagekräftig sind.

Generell ist die Kritik, dass nur ein einfacher Prozentsatz nicht ausschlaggebend für die Fähigkeiten des Modells ist - dennoch werden diese als übliche Metriken angewandt. Um Störfaktoren der üblichen Metriken noch zusätzlich zu verringern, werden modernere Metriken angewandt:

- **Area Under the ROC Curve (AUC-ROC):** Diese Kurve zeigt, wie sich die True-Positive-Rate und False-Positive-Rate bei verschiedenen Schwellenwerten verändern. Für jeden Schwellenwert kann man ablesen, wie viele False Positives bzw. False Negatives entstehen. Die AUC-ROC fasst die Performance über alle Schwellenwerte zusammen.
- **Area Under the Precision-Recall Curve (AUC-PR):** Diese Kurve zeigt die Beziehung zwischen Precision und Recall bei verschiedenen Schwellenwerten. Für jeden Schwellenwert misst man neu, wie genau (Precision) und wie vollständig (Recall) die positiven Fälle erkannt werden. So erkennt man, bei welchem Schwellenwert die Balance zwischen Precision und Recall am besten ist.
- **Matthews Correlation Coefficient (Matthews Correlation Coefficient (MCC)):** Eine umfassende Metrik, die alle vier Werte der Verwirrungsmatrix (True Positives, True Negatives, False Positives, False Negatives) berücksichtigt. MCC liefert einen Wert zwischen -1 und 1, wobei 1 perfekte Vorhersage, 0 zufällige Vorhersage und -1 vollständig falsche Vorhersage bedeutet. Besonders geeignet für unausgeglichene Datensätze.
- **Balanced Accuracy:** Der Anteil der negativen Beispiele, die fälschlicherweise als positiv klassifiziert wurden. Sie wird berechnet als $FPR = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}}$ und ist wichtig, um die Fehlerquote bei der Erkennung negativer Fälle zu bewerten.

Auch wenn diese Metriken mehr Informationen über die Verarbeitungsweise der Modelle schließen lassen, haben diese ebenfalls Einschränkungen. Davis und Goadrich zeigen, dass ROC und PR Kurven durch eine eindeutige mathematische Beziehung miteinander verbunden sind. Für ein gegebenes Datenset enthalten beide Kurven die gleichen Informationen, werden aber unterschiedlich dargestellt. [29].

Studien zeigen, dass alternative Metriken wie MCC und Balanced Accuracy insbesondere bei unausgeglichene Datensätzen vernünftige Eigenschaften aufweisen. Chicco und Jurman zeigen, dass MCC im Vergleich zu Accuracy und F1 Score robustere und wahrheitsgetreuere Ergebnisse liefert. Brodersen et al. zeigen, dass Balanced Accuracy durch die gleichgewichtete Berücksichtigung beider Klassen Verzerrungen bei unbalancierten Klassen vermeidet [30, 31].

Außer diesen Metriken wurde noch überlegt, einen sogenannten Cross-Validation-Test durchzuführen. Cross-Validation ⁷ ist keine Metrik wie z.B. der F1-Score, sondern eine Methode zur Modellbewertung. Sie unterteilt dabei den Datensatz in k gleich große Teile. Das Modell wird dann k -mal trainiert, jedes mal mit $k-1$ Teilen als Trainingsdaten. Der übrig gebliebene Teil wird als Testdatensatz und Validierungsdatensatz verwendet. Der Sinn dahinter ist die Trainingsdaten so zu verteilen, dass das Modell jedes mal neu trainiert wird. Dies soll Overfitting auf den Daten vermeiden. Zudem soll sie die Ergebnisse nach jedem Training des k -Datensatzes zeigen, damit sich erkennen lässt, ob es an bestimmten Stellen Abweichungen gab. Dieser wurde jedoch verworfen, um die Auswertung und wissenschaftlichen Test an einer anderen Art der Bewertung zu verwenden, welche eher üblich ist in wissenschaftlichen Arbeiten.

3.2 Störfaktoren

Für einen objektiven Vergleich der Modelle ist insbesondere die Art der Datenvorverarbeitung entscheidend. Alle Features in den Logs werden numerisch verarbeitet; jedoch ist nicht bekannt, welche Features das Lernverhalten der Modelle maßgeblich beeinflussen. Eine sorgfältige Vorverarbeitung ist daher notwendig, um sicherzustellen, dass alle relevanten Features, korrekt berücksichtigt werden.

Ein weiterer wesentlicher Faktor ist die Wahl der Modellparameter. Parameter wie beispielsweise die Anzahl der Epochen oder die Gesamtarchitektur können das Verhalten und die Ergebnisse der Modelle erheblich beeinflussen. Die einzelnen Algorithmen wie IF reagieren sensibel auf Parameter-Änderungen und dass die Ergebnisse dadurch erheblich beeinflusst werden. Dementsprechend muss man vorsichtig sein und gewährleisten, keine unangemessenen Parameter zu setzen und dafür zu sorgen, dass trotz der Setzung einer fairer Vergleich ermöglicht wird.

Die Zufälligkeit der Daten beeinflusst ebenfalls die Modellleistung. Die Datensätze müssen einerseits zufällig generiert sein, gleichzeitig aber einen kontinuierlichen Zusammenhang aufweisen, wie er bei realen Cyberangriffen vorkommt. Dies ist insbesondere für LSTM-Netzwerke relevant, die auf sequenzielle Abhängigkeiten angewiesen sind. Kontextlose oder stark fragmentierte Datenpunkte könnten das Lernen solcher Modelle erschweren. Von daher werden Ergebnisse mit einer hohen False-Positive-Rate erwartet.

Auch die verfügbare Rechenleistung kann die Verarbeitungsgenauigkeit beeinflussen. Zu kurze oder zu lange Trainingszeiten sowie eine hohe Rechenintensität wirken sich negativ auf die Modellleistung aus. Daher werden in allen Modellen Regulierungstechniken⁸ implementiert, um eine ausgewogene Trainingsdauer und Leistungsfähigkeit zu gewährleisten.

Der Aufbau der zu verarbeitenden Daten ist ein weiterer entscheidender Faktor. Bei teilweise synthetisch generierten Logs bestimmt der Anteil der als anomal gekennzeichneten Zeilen die Bewertung der Modelle.

Die Generierung synthetischer Logs bringt methodische Herausforderungen mit sich. Alternativ wurden auch reale Daten in Betracht gezogen, etwa durch automatisierte Tests oder firmeninterne Logs. Wie bereits in Kapitel 2 erläutert ist die Benutzung von Unternehmensdaten aufgrund den datenschutzrechtlichen Herausforderungen nicht machbar. Diese entsprechen nicht exakt dem realen Benutzerverhalten, was die Aussage der Hypothese etwas beschränkt auf die für den Vergleich genutzten generierten Daten. Ein Vorteil ist jedoch, dass sich der Aufbau und die Struktur hierfür kontrollieren lassen.

Acikmese und Alptekin beschreiben, dass LSTM Modelle in ihrem Experiment als datenhungrig erscheinen, da die geringe Datengröße nicht ausreichte, um eine gute Generalisierung zu erreichen [32]. Ein kleiner Datensatz kann zu einer Unterforderung des Modells führen und die Ergebnisse signifikant beeinflussen. Auch in dieser Arbeit ist dies zu berücksichtigen.

Beim Vergleich der Modellleistungen mit synthetischen und realen Daten ist zu erwarten, dass die Ergebnisse variieren, da sich die Feature-Struktur und die logische Beschaffenheit der Logs zwischen den Datensätzen unterscheiden.

4 Angewandte Technologien

4.1 Scikit-learn Learn

Scikit-learn⁹ ist eine weitverbreitete Open-Source-Bibliothek für maschinelles Lernen in Python. Sie stellt eine Vielzahl von Werkzeugen für klassische Machine-Learning-Aufgaben bereit, darunter Klassifikation, Regression und Clustering sowie Modellbewertung und -selektion. Scikit-learn bietet eine Reihe vorimplementierter Modelle wie IF, OSCVM und DBSCAN.

4.2 Keras

Keras¹⁰ ist eine benutzerfreundliche, modulare und erweiterbare Open-Source-Bibliothek für die Entwicklung von Deep-Learning-Modellen in Python. Sie bietet eine einfache und intuitive API, die die schnelle Implementierung und das Training neuronaler Netze ermöglicht. Keras unterstützt verschiedene neuronale Netzwerkarchitekturen, darunter Convolutional Neural

Networks (CNNs), Recurrent Neural Networks (RNNs) und AE's.

4.3 TensorFlow

TensorFlow¹¹ ist ein umfangreiches Open-Source-Framework von Google für maschinelles Lernen und Deep Learning. Es ermöglicht die effiziente Definition, Optimierung und Ausführung von numerischen Berechnungen. TensorFlow bietet Skalierbarkeit auf verschiedenen Plattformen von mobilen Geräten bis zu großen verteilten Systemen und unterstützt sowohl CPU- als auch GPU-Beschleunigung. Keras ist in TensorFlow integriert und fungiert als API, die den Zugang zu den leistungsfähigen Funktionen von TensorFlow erleichtert.

4.4 Verworfenen Technologie: Deep GPU Xceleration (DGX)

Eine DGX ist ein Hochleistungsrechnersystem von NVIDIA¹². Es eignet sich insbesondere für das effiziente Training rechenintensiver Modelle des maschinellen Lernens, wie beispielsweise LSTM-Netze. Die DHBW stellt hierfür das Modell DGX H100 zur Verfügung, auf dem die Trainingsprozesse ausgeführt wurden. NVIDIA DGX H100-Systeme sind mit zwei Intel Xeon 8480C-Prozessoren ausgestattet, die gemeinsam über insgesamt 112 CPU-Kerne verfügen¹³.

Im weiteren Entwicklungsverlauf erfolgte die Modellierung zunehmend CPU-basiert, sodass ein alternatives System genutzt wurde. Da lediglich ein kleiner Datensatz für Training und Test vorliegt, war der Einsatz einer Hochleistungs-GPU nicht mehr erforderlich.

5 Keycloak

Keycloak ist ein IAM-Tool mit SSO-Funktion und stellt ein rollenbasiertes Identitäts- und Zugriffsmanagementsystem (RBAC) dar. Es ermöglicht die sichere und strukturierte Verwaltung von Benutzer- und Unternehmensdaten. Bei der Anmeldung über Keycloak bei einem Dienstleister werden durch SSO die Authentifizierungs- und Autorisierungsdaten an die jeweilige Anwendung übertragen.

5.1 Terminologien in Keycloak

Für ein besseres Verständnis der zentralen Begriffe und Abläufe wird in diesem Abschnitt ein Überblick über die wichtigsten Terminologien und Konzepte gegeben.

Realms und Clients:

Keycloak organisiert Anwendungen in sogenannten *Realms*. Ein Realm stellt eine isolierte Umgebung dar, in der Benutzer, Rollen und *Clients* verwaltet werden. Die Clients repräsentieren dabei die jeweiligen Anwendungen, die unter anderem Benutzer- und Grup-

pendaten verwalten. Innerhalb eines Realms können mehrere Clients existieren, welche die jeweiligen Anwendungen oder Dienste repräsentieren, die Keycloak für Authentifizierung und Autorisierung nutzt.

Standardmäßig existiert ein *Master-Realm*, der übergeordnete Verwaltungsrechte besitzt. In diesem Realm befindet sich der *Admin*, der Zugriff auf alle weiteren Realms und deren Komponenten hat. Der Admin kann CRUD-Operationen auf Benutzer, Gruppen und Clients ausführen. Normale Benutzer sehen ausschließlich den eigenen Realm und besitzen nur eingeschränkte Rechte, sofern ihnen keine zusätzlichen Rollen zugewiesen wurden.

Grant-Typen und Authentication Flows:

Keycloak unterstützt unterschiedliche *Grant-Types*, welche die Authentifizierung von Benutzern oder Clients definieren:

- **Authorization Code Grant:** Der Benutzer wird zur Keycloak-Loginseite weitergeleitet. Nach erfolgreicher Authentifizierung erhält der Client einen *Authorization Code*, der gegen ein *Access Token* ¹⁴ eingetauscht werden kann. Das Passwort wird dabei nicht an die Client-Anwendung weitergegeben.
- **Client Credentials Grant:** Dieser Grant-Type wird für Clients ohne Benutzerinteraktion verwendet. Der Client authentifiziert sich mit seiner Client-ID und einem geheimen Schlüssel (*Client-Secret*), um ein Access Token zu erhalten.

Die **Client-Authentifizierungsmethode** definiert eine technische Form des Identitätsnachweises vom Clients gegenüber dem Server, z. B. mittels Client-Secret oder Zertifikat. Während der Grant-Type die Art der Anmeldung bestimmt, legt die Client-Authentifizierungsmethode die konkrete Authentifizierung fest.

Zusätzlich existieren in Keycloak sogenannte *Authentication Flows*, die den Ablauf der Authentifizierung steuern:

- **Standard Flow:** Verwendet in der Regel den Authorization Code Grant¹⁵ mit Weiterleitung zur Login-Seite.
- **Direct Access Grant:** Direkte Anmeldung über Benutzername und Passwort, beispielsweise für Skripte oder mobile Anwendungen.

Hier sind nur einige erwähnt, welche in dieser Arbeit relevant sind. Natürlich gibt es mehrere Grant-Typen.

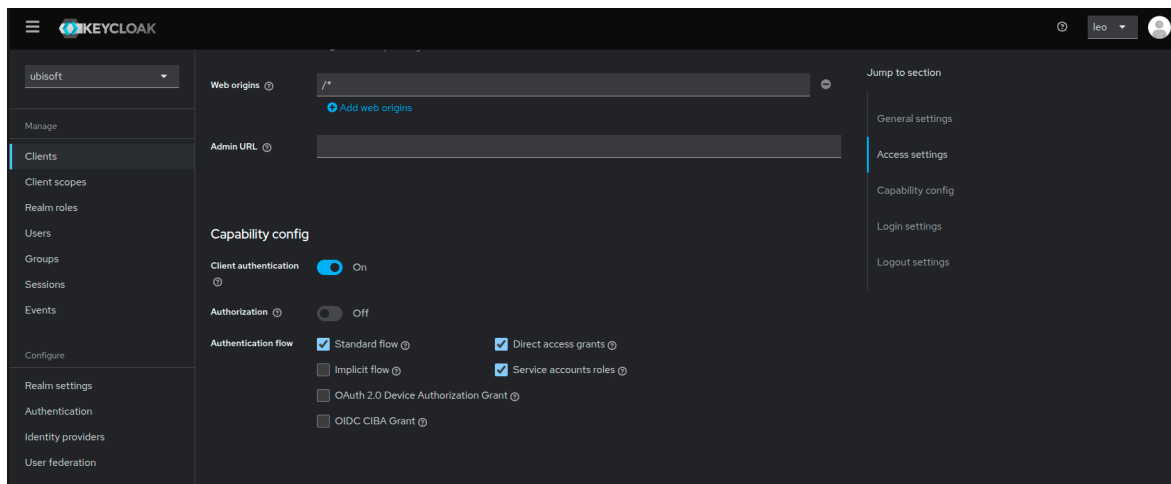


Abbildung 1: Beispielhafte Konfiguration von Authentication Flows in Keycloak: Hier ist bspw. die Client Authentication, Standard Flow, Service Accounts Roles und Direct Access Grant aktiviert.

Zudem unterscheidet Keycloak zwei zentrale Rollentypen:

- **Realm-Roles:** Gültig innerhalb eines gesamten Realms und geeignet für übergreifende Rechte, z. B. *realm-admin* für alle Clients.
- **Client-Rolea:** Spezifisch für einen einzelnen Client und ermöglichen eine feinere Zugriffskontrolle innerhalb der jeweiligen Anwendung, z.B. *view-events*.

Service Account Roles:

Clients können sich in Keycloak auch selbst authentifizieren, ohne Benutzerinteraktion. Wird die *Service Account*-Funktion aktiviert, erhält der Client ein eigenes Konto, das mit Rollen ausgestattet werden kann, um Berechtigungen auf bestimmte Ressourcen oder Operationen zu erhalten, beispielsweise für automatisierte Backend-Operationen.

5.2 Komponenten der Log-Erzeugung

Keycloak generiert Logs über verschiedene Frameworks und Komponenten. Eines dieser Frameworks ist das Logging-Framework von JBoss ¹⁶, wobei moderne Versionen von Keycloak zusätzlich das *Quarkus-Framework* nutzen.

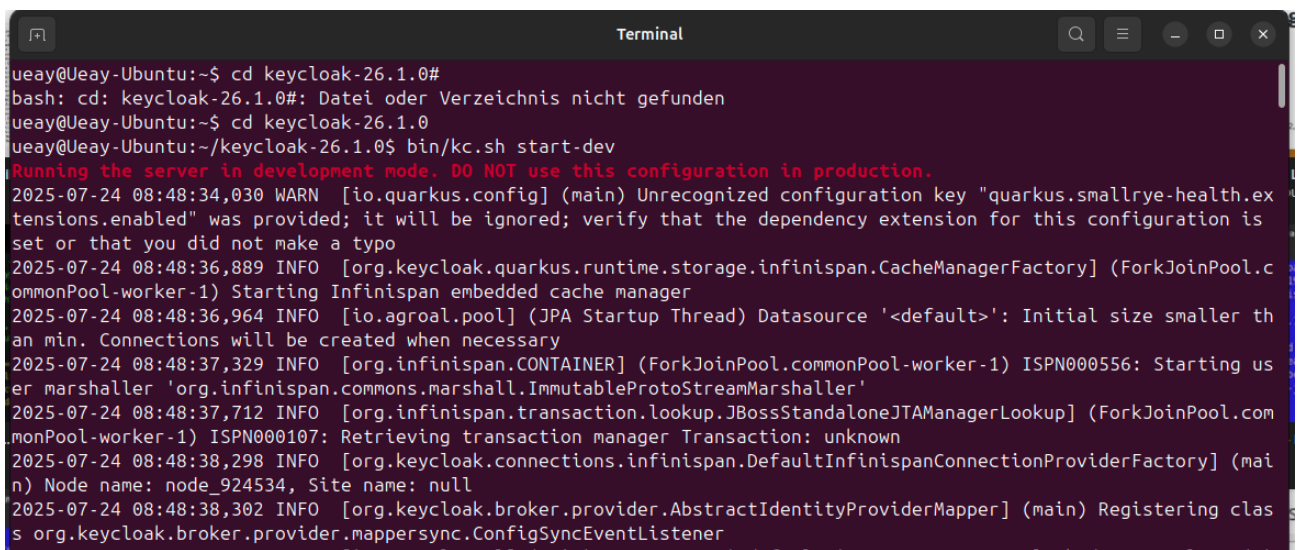
Frühere Versionen von Keycloak basierten auf dem **WildFly**-Anwendungsserver¹⁷, der die **JBoss Logging**-Infrastruktur einsetzte. Ab Keycloak Version 17 wurde WildFly durch das **Quarkus**-Framework ersetzt, das weiterhin JBoss Logging unterstützt.

JBoss Logging fungiert in Keycloak als zentrale Logging-API. Es abstrahiert verschiedene Logging-Backends (z. B. Log4j und SLF4J) und ermöglicht die strukturierte Ausgabe sicherheitsrelevanter Informationen. Entwickler müssen sich dabei nicht auf ein spezifisches

Logging-Backend festlegen, sondern können über die einheitliche JBoss-API unterschiedliche Ausgabekanäle konfigurieren, wie Konsole, Datei oder Syslog.

Ursprünglich wurden Logs für diese Arbeit über JBoss Logging direkt an ein Terminal ausgegeben. In späteren Versionen wurde entschieden, die Keycloak-API für die Log-Ausgabe zu verwenden. Detaillierte Informationen zur Nutzung der API werden in den folgenden Kapiteln behandelt.

Generell hat ein Log in Quarkus folgenden Aufbau: [Datum, Zeit] [Log-Typ] [Java-Klasse] [Fehlermeldung].

A screenshot of a terminal window titled "Terminal" showing the startup logs of a Keycloak instance. The user is in a directory named "keycloak-26.1.0" and runs "bin/kc.sh start-dev". The logs show a warning about an unrecognized configuration key, followed by several information messages from Infinispan and Keycloak components, including cache manager startup, datasource initialization, and transaction manager retrieval. The logs are formatted with timestamps, log levels, and class names.

```
ueay@Ueay-Ubuntu:~$ cd keycloak-26.1.0#
bash: cd: keycloak-26.1.0#: Datei oder Verzeichnis nicht gefunden
ueay@Ueay-Ubuntu:~$ cd keycloak-26.1.0
ueay@Ueay-Ubuntu:~/keycloak-26.1.0$ bin/kc.sh start-dev
Running the server in development mode. DO NOT use this configuration in production.
2025-07-24 08:48:34,030 WARN [io.quarkus.config] (main) Unrecognized configuration key "quarkus.smallrye-health.extensions.enabled" was provided; it will be ignored; verify that the dependency extension for this configuration is set or that you did not make a typo
2025-07-24 08:48:36,889 INFO [org.keycloak.quarkus.runtime.storage.infinispan.CacheManagerFactory] (ForkJoinPool.commonPool-worker-1) Starting Infinispan embedded cache manager
2025-07-24 08:48:36,964 INFO [io.agroal.pool] (JPA Startup Thread) Datasource '<default>': Initial size smaller than min. Connections will be created when necessary
2025-07-24 08:48:37,329 INFO [org.infinispan.CONTAINER] (ForkJoinPool.commonPool-worker-1) ISPN000556: Starting user marshaller 'org.infinispan.commons.marshall.ImmutableProtoStreamMarshaller'
2025-07-24 08:48:37,712 INFO [org.infinispan.transaction.lookup.JBossStandaloneJTAManagerLookup] (ForkJoinPool.commonPool-worker-1) ISPN000107: Retrieving transaction manager Transaction: unknown
2025-07-24 08:48:38,298 INFO [org.keycloak.connections.infinispan.DefaultInfinispanConnectionFactory] (main) Node name: node_924534, Site name: null
2025-07-24 08:48:38,302 INFO [org.keycloak.broker.provider.AbstractIdentityProviderMapper] (main) Registering class org.keycloak.broker.provider.mappersync.ConfigSyncEventListener
```

Abbildung 2: Log-Ausgaben aus einer Quarkus-basierten Keycloak-Instanz.

Insgesamt stellt die Kombination von JBoss Logging und Quarkus eine flexible und erweiterbare Logging-Lösung dar. In dieser Arbeit diente sie lediglich zu Beginn der Arbeitsphase dazu Orientierung zu gewinnen, wie die Keycloak-Logs aufgebaut sind und den ersten Ansatz (s. Kapitel 8) zu realisieren.

5.3 Log-Dateien in Keycloak

Logging bzw. Protokollierung ist ein essenzieller Bestandteil der Überwachung von IT-Systemen. Logs dienen im IT-Monitoring der Prävention und Detektion von Anomalien, sei es im Netzwerk, auf IoT-Geräten oder innerhalb von IAM-Softwarelösungen.

Keycloak unterteilt Logs in folgende Kategorien:

- Server-Logs: Informationen zum Betriebsstatus, Fehler- und Debugging-Informationen
- Event-Logs: Benutzer- und Admin-Events

- Audit-Logs: Nachvollziehbarkeit von Änderungen durch Administratoren. Audit-Logs gelten als Unterkategorie der Event-Logs.
- Access Logs: Zugriffsversuche und HTTP-Statuscodes (HTTP Layer / Reverse Proxy)
- Security Logs: Sicherheitsrelevante Ereignisse und Warnungen
- Custom-Logs: Erweiterungen und Plugins können eigene Logs erzeugen (SPI)

Die Server-Logs werden vom Keycloak-Server erzeugt und über das Quarkus-Framework verarbeitet. Sie sind unter der Kategorie *org.keycloak.services* zu finden.

Event-Logs und Audit-Logs werden durch den *EventListenerProvider*¹⁸ in Keycloak erzeugt und unter der Kategorie *org.keycloak.events* geführt. Access-Logs entstehen hingegen auf der Anwendungs- und Netzebene, meist außerhalb von Keycloak, und sind mit *io.quarkus.http* gekennzeichnet.

5.4 Event-Logs und Audit-Logs

Für die Analyse in dieser Arbeit werden die Event-Logs und Audit-Logs betrachtet. Diese enthalten zentrale Informationen zum Benutzerverhalten.

Nicht untersucht werden in dieser Arbeit Server-Logs und Access-Logs, da diese durch andere Komponenten in Keycloak erzeugt werden und nicht direkt das Benutzerverhalten widerspiegeln, da sie netzbasierte Logs sind- wobei zur Analyse von ungewöhnlicher, genutzter Geräte auch dazu dienen, Anomalien zu erkennen. Da diese Arbeit hauptsächlich Intrusionen und Insider-Attacken direkt durch ungewöhnliches Verhalten erkennen will und nicht durch Geräte, werden nur die beiden Event-Typen Event-Logs und Audit-Logs genutzt. Da keine SPI verwendet wird und nur eine einzelne Keycloak-Instanz untersucht wird, fallen Custom-Logs als Trainingsdaten weg.

Keycloak unterscheidet zwei Arten von Event-Logs:

- Admin-Logs (Audit-Logs)
- User-Logs

Admin-Logs dokumentieren alle Admin-Operationen, während User-Logs ausschließlich Ereignisse regulärer Benutzer enthalten.

5.5 Aufbau der Event-Logs

Typische Features in den User-Event-Logs sind:

- timestamp

- log_level
- category
- type
- ipAddress
- realmId
- clientId
- userId

Wichtige Informationen, wie z. B. der Standort des Benutzers, fehlen aus datenschutzrechtlichen Gründen. Dies kann die Erkennung von Anomalien erschweren, da ungewöhnliche Standorte (z. B. Pentagon) nicht identifiziert werden können. Timestamp hingegen ist ein zentrales Feature, das den Zeitpunkt der Log-Erstellung angibt und sich für Analysen über lange Zeit, wie etwa mit LSTM-Autoencoder-Hybridmodellen, eignet.

Das Feature "category" beschreibt, aus welchem Modul das Log erzeugt wurde. Darüber hinaus sind die Ereignis-Typen ("type" hier) relevant.

Beispielsweise bezeichnet LOGIN einen Event-Typ, der ausgelöst wird, wenn ein Benutzer sich anmeldet. Weitere Event-Typen sind bspw. REFRESH.TOKEN, bei dem ein Nutzer einen neuen Token erhält.

Keycloak unterscheidet zudem verschiedene Log-Level, welche den Erfolg einer Operation anzeigen oder zusätzliche Informationen bereitstellen. Bekannte Log-Level sind z. B. INFO, FATAL oder ERROR ¹⁹. Das Log-Level bestimmt, ob ein Eintrag der Fehlerbehebung, Warnung oder allgemeinen Information dient. Zudem kann man steuern, welche Logs ausgegeben werden, wenn man die bestimmte Berechtigung (*Maintainer*) dazu hat.

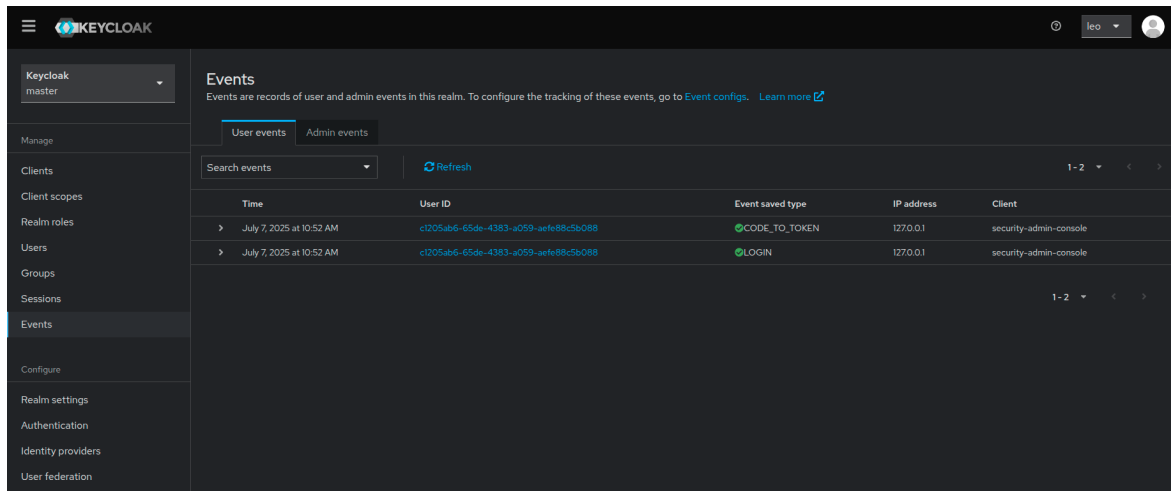


Abbildung 3: Hier kann man in Keycloak die User und Admin Events einsehen. Man erkennt hier in den User-Events schon ein paar Einträge mit dem entsprechenden Event-Typ

Je nach Event-Typ können zusätzliche Features aufgezeichnet werden. Beim LOGIN werden beispielsweise die Token-URL, Verbindungsart und das Authentifizierungsprotokoll protokolliert. Anhand Beispiele aus dem internen Unternehmen sind die häufigsten Event-Typen:

- LOGIN: Benutzer meldet sich erfolgreich über Keycloak an.
- CODE TO TOKEN: Wird erzeugt, sobald der Benutzer nach Authentifizierung einen Access Token erhält.
- CLIENT LOGIN: Log-Eintrag bei Anmeldung über einen Drittanbieter.
- LOGIN ERROR: Fehler beim Login über Keycloak.
- CODE TO TOKEN ERROR: Falsche Anmeldedaten, Server lehnt die Authentifizierung ab.

In den Admin-Logs sind zusätzliche Features enthalten, die in User-Logs nicht vorkommen, insbesondere das Feature `operationType`. Diese beschreiben typische CRUD-Operationen wie UPDATE, CREATE, DELETE und ACTION²⁰.

6 Theoretischer Hintergrund der Modelle

6.1 LSTM

Long Short-Term Memory (LSTM) ist ein spezieller Typ rekurrenter neuronaler Netzwerke (RNN), welches entwickelt wurde, um Informationen über längere Zeiträume hinweg zu

speichern und verarbeiten [33]. RNNs verarbeiten sequentielle Daten, bei denen die Reihenfolge der Eingaben eine zentrale Rolle spielt, und speichern frühere Informationen in einem sogenannten "Hidden State".

Klassische RNNs stoßen bei langen Sequenzen auf das Problem des Vanishing Gradient, da der Gradient beim Zurückpropagieren über viele Zeitschritte sehr klein wird und frühere Informationen verloren gehen können. LSTMs umgehen dieses Problem durch eine spezielle Zellstruktur, die relevante Informationen über längere Zeiträume hinweg speichern kann [34, S.19].

LSTMs finden Anwendung in zahlreichen Bereichen, z.B. bei Vorhersagemodellen oder im Internet der Dinge (IoT) [35].

Die Sequenzlänge und die Netzwerkgröße beeinflussen maßgeblich die Trainingsdynamik von rekurrenten neuronalen Netzen. Größere oder tiefere Netzwerke können komplexere Muster erfassen, erhöhen jedoch den Rechenaufwand und den Speicherbedarf. Insbesondere lange Sequenzen erschweren das Lernen, da die Gradienten während der Rückpropagation durch die Zeit entweder verschwinden oder explodieren können, was die Konvergenz des Trainings erschwert [36].

Die Wahl der Sequenzlänge beeinflusst also nach Bengio den Lernerfolg des Modells. Sehr lange Sequenzen erhöhen die Komplexität des Trainings und erschweren die Konvergenz, während kürzere Sequenzen das Modell auf lokale Muster fokussieren und die Erkennung von Anomalien verbessern.

Es gibt auch LSTM-Architekturen, welche flexibler und robuster mit der Sequenzlänge sind: "Hierarchisch-multiskalige" LSTM-Modelle lernen dynamisch, auf welchen Zeitskalen Informationen aktualisiert oder aggregiert werden sollten. Hierdurch können zeitliche Abhängigkeiten über lange Sequenzen effizienter modelliert werden, da das Modell gleichzeitig Abhängigkeiten auf mehreren Skalen berücksichtigt [37].

LSTMs generalisieren Zählverhalten über deutlich längere Sequenzen oft nicht zuverlässig, selbst wenn sie es auf kürzeren Sequenzen scheinbar korrekt gelernt haben [38].

Aufgrund der begrenzten Datenmenge und den genannten Nachteilen von langen Sequenzlängen liegt der Fokus auf kurzen Sequenzen.

6.2 Autoencoder

Ein Autoencoder ist ein neuronales Netzwerk, das aus einer Encoder- und einer Decoder-Schicht besteht [39, S.3]. Der Encoder komprimiert die Eingabedaten in eine kompakte Bottleneck-Repräsentation, wodurch wesentliche Merkmale extrahiert und redundante Informationen reduziert werden. Der Decoder rekonstruiert anschließend die ursprünglichen Daten, wobei der Rekonstruktionsfehler – die Differenz zwischen Eingabe und Ausgabe –

erzeugt wird.

Signifikante Abweichungen von gelernten Mustern führen zu höheren Rekonstruktionsfehlern und markieren potenzielle Anomalien. Die Dimension des Bottlenecks spielt dabei eine entscheidende Rolle: Ein zu kleines Bottleneck kann relevante Informationen verlieren, während ein zu großes Bottleneck dem Netzwerk ermöglicht, die Daten ohne echte Verdichtung durchzuschleusen. Um die Extraktion vielfältiger Merkmale zu fördern, kann die Verlustfunktion modifiziert werden, sodass überlappende oder redundante Merkmale minimiert werden [40].

Erweiterte Autoencoder-Varianten umfassen beispielsweise den Variational Autoencoder (VAE), der probabilistische und generative Eigenschaften besitzt, sowie den Denoising Autoencoder-Autoencoder, der verrauschte Eingaben bereinigt. Mehrschichtige Autoencoder ermöglichen darüber hinaus die Extraktion komplexerer Merkmalsstrukturen. Studien zeigen, dass diese erweiterten Architekturen bei der Anomalieerkennung effektiver sind: So weisen Pantelidis et al. darauf hin, dass der VAE besser Anomalien erkennen kann als der normale Autoencoder [41], Skaf et al. berichten eine bessere Leistung für den Denoising-AE [42], und Wei et al. zeigen, dass mehrschichtige Autoencoder Anomalien besonders zuverlässig identifizieren können [43].

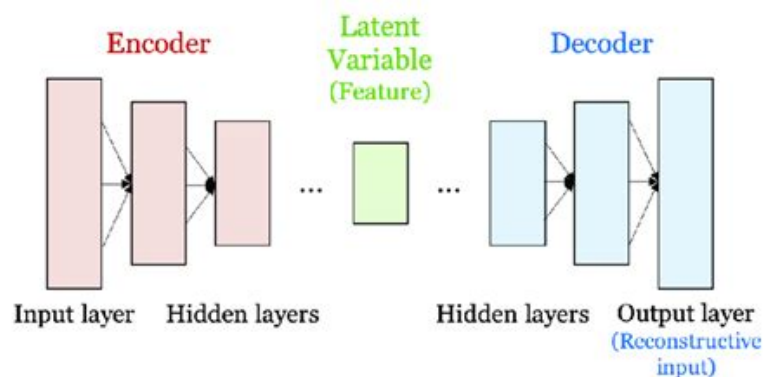


Abbildung 4: Visuelle Darstellung der Funktionsweise eines Autoencoders. Das Kästchen in der Mitte stellt den Bottleneck dar. [Online]. Quelle: https://www.researchgate.net/figure/Structure-of-the-autoencoder_fig1_36507443, letzter Zugriff am: 15.07.2025

Wie im Kapitel zum Forschungsstand diskutiert, lassen sich besonders Autoencoder effektiv mit LSTM-Netzwerken kombinieren, wenn es um sequenzielle oder zeitabhängige Daten geht. Ein LSTM-Autoencoder verwendet LSTM-Zellen im Encoder und Decoder, um komplexe zeitliche Muster zu erfassen und zu rekonstruieren. Diese Architektur ist insbesondere für die Anomalieerkennung in Zeitreihen geeignet, beispielsweise in der Überwachung von Maschinenzuständen, Finanzdaten oder der Netzwerksicherheit. Anomalien manifestieren sich häufig durch signifikant erhöhte Rekonstruktionsfehler, da zeitliche Muster nicht korrekt rekonstruiert werden können.

6.3 Isolation Forest

Isolation Forest ist ein unüberwachter Lernalgorithmus, der Daten durch Partitionierung trennt, indem er binäre Bäume erzeugt. Dabei werden die Daten so lange aufgeteilt, bis keine weitere Isolierung möglich ist. Dieses Vorgehen erlaubt es, Anomalien zu erkennen: Datenpunkte, die in wenigen Partitionierungsschritten isoliert werden, gelten als potenzielle Anomalien. Datenpunkte, die ähnliche Merkmalswerte aufweisen, werden dagegen nicht stark isoliert. Je stärker sich ein Datenpunkt anhand seiner Merkmale isolieren lässt, desto wahrscheinlicher ist es, dass er keiner üblichen Menge zugeordnet werden kann [44, S. 2]. Isolation Forest eignet sich zudem für hochdimensionale Daten [44, S. 10].

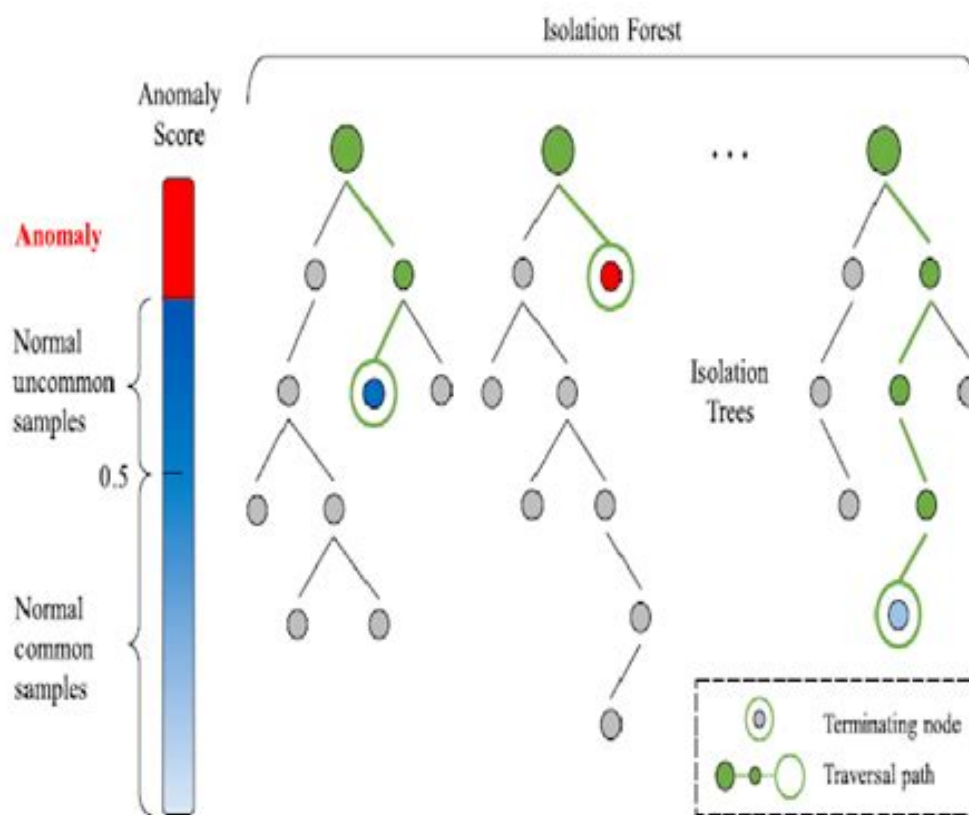


Abbildung 5: Visuelle Darstellung der Funktionsweise des Isolation Forest [Online]. Quelle: https://www.researchgate.net/figure/Anomaly-Detection-using-Isolation-Forest-18_fig3_350551253, letzter Zugriff am: 15.07.2025

Ein Isolation Tree besteht aus inneren Knoten und Blattknoten. Jeder innere Knoten enthält eine Testregel, die die Datenpunkte in zwei Gruppen aufteilt. Die Testregel basiert auf der Auswahl eines Merkmals q und eines Schwellenwerts p : Datenpunkte mit einem Wert für q kleiner als p gelangen in den linken Kindknoten, alle anderen in den rechten.

Zum Aufbau eines solchen Baums wird eine Stichprobe der Datenpunkte rekursiv aufgeteilt, wobei Merkmal und Split-Wert zufällig gewählt werden. Dieser Prozess wird so lange wiederholt, bis

- die maximale Baumhöhe erreicht ist,
- nur noch ein Datenpunkt im Teilbaum übrig ist, oder
- alle Datenpunkte im aktuellen Teilbaum identisch sind [44, S.3].

Das Ergebnis ist ein vollständiger binärer Baum, bei dem jeder Datenpunkt in einem Blattknoten isoliert wird.

Wenn alle Datenpunkte unterschiedlich sind, gilt:

- Die Anzahl der Blattknoten entspricht der Anzahl der Datenpunkte n .
- Die Anzahl der inneren Knoten beträgt $n - 1$.
- Insgesamt besitzt der Baum $2n - 1$ Knoten.
- Der Speicherbedarf wächst somit linear mit der Anzahl der Datenpunkte.

Isolation Forest unterteilt die Datenpunkte zufällig, was die Nachvollziehbarkeit der Partitionen einschränkt, jedoch die Effizienz des Algorithmus steigert.

6.4 One-Class SVM

OCSVM ist ein unüberwachter Algorithmus, der Anomalien erkennt, indem er eine Trennlinie (oder Grenzfläche) zwischen normalen Datenpunkten und potenziellen Ausreißern in einem hochdimensionalen Raum lernt, wobei die Flexibilität der Kernelmethoden es erlaubt, auch komplexe Grenzen zu modellieren [45].

Zur Modellierung dieser Support-Region wird eine Funktion $f(x)$ gelernt, die auf den meisten Trainingsdaten positive Werte annimmt und auf Punkten außerhalb der Verteilung negative Werte. Durch den Einsatz von Kernelfunktionen kann die OCSVM auch komplexe, nichtlineare Grenzen um die Datenverteilung modellieren. Auf diese Weise identifiziert das Modell effektiv Ausreißer, die sich deutlich von den normalen Mustern unterscheiden [46, S.6].

Viele Angriffe lassen sich durch die Analyse von System-Logdaten erkennen. Ein Ansatz auf Basis der OCSVM wurde beispielsweise mit abstrahierten Benutzeraudit-Logs aus dem DARPA-Datensatz von 1999 trainiert [47]. Durch die Abbildung der Daten in einen höherdimensionalen Raum können auch nichtlinear trennbare Daten effektiv separiert werden, wodurch die OCSVM für die Anomalieerkennung geeignet ist.

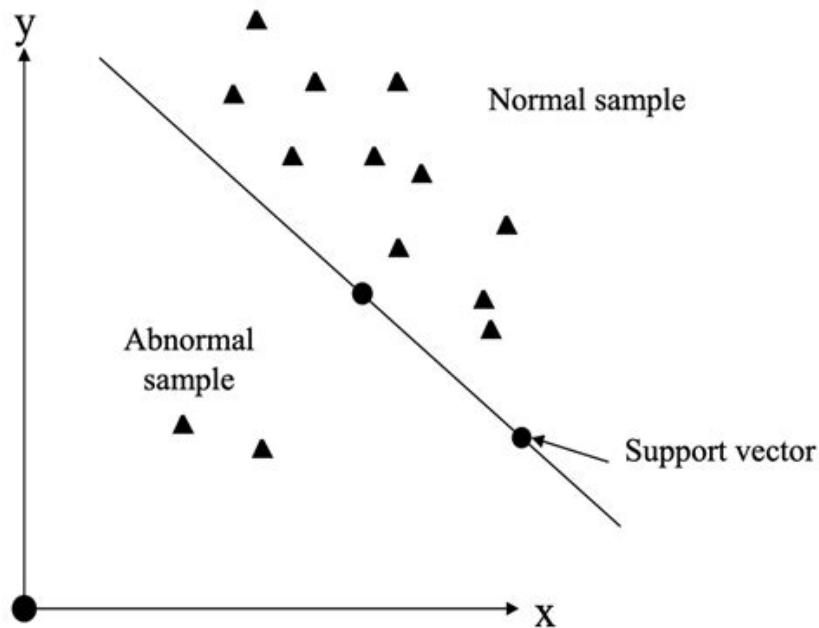


Abbildung 6: Visuelle Darstellung der OCSVM [Online]. Quelle: <https://www.mdpi.com/2076-3417/13/3/1734>, letzter Zugriff am: 15.07.2025

Ein zentrales Problem der OCSVM besteht darin, dass der Ursprung im Feature Space (Merkmalsraum) als Referenzpunkt für Anomalien dient, was in der Praxis nicht immer zutrifft [48]. Die Autoren erklären, dass die Trennung der Zielklasse vom Ursprung – also von den potenziellen Ausreißern – häufig missverstanden wird. Sie schlagen eine geometrische Interpretation vor, bei der die Zielklasse vom übrigen Raum getrennt wird, insbesondere wenn ein Gauß-Kernel verwendet wird. Da diese Arbeit jedoch Klassen von Scikit verwendet, wird stattdessen der RBF-Kernel verwendet. Die Nachteile von OCSVM wird wie bei IF und DBSCAN durch Parametrisierung ausgeglichen.

6.5 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) ist ein dichte-basiertes Clusteringverfahren, das Datenpunkte anhand ihrer lokalen Dichte gruppiert [49].

Die Dichte wird durch die Anzahl der Nachbarpunkte innerhalb eines definierten Radius (Eps) bestimmt. Ein Punkt gilt als Kernpunkt, wenn sich mindestens eine vorgegebene Mindestanzahl von Punkten (MinPts) innerhalb dieses Radius befindet. Die Wahl dieser Parameter ist entscheidend für die Qualität der Clusterbildung.



Abbildung 7: Beispielhafte Darstellung von DBSCAN-Clustern. Quelle: <https://www.baeldung.com/wp-content/uploads/sites/4/2023/04/clusters.png>, letzter Zugriff am: 15.07.2025

Der Algorithmus arbeitet wie folgt:

1. Ein unbesuchter Punkt p wird ausgewählt.
2. Alle Punkte, die sich innerhalb des Radius Eps um p befinden, werden bestimmt.
3. Bildet p zusammen mit seinen Nachbarn einen Kernpunkt (d.h. die Anzahl der Nachbarn $\geq \text{MinPts}$), wird ein neues Cluster gebildet. Alle direkt dichten Nachbarn werden in dasselbe Cluster aufgenommen.
4. Jeder Punkt, der keinem Cluster zugeordnet werden kann, wird als Rauschpunkt (*Noise*) markiert.
5. Der Vorgang wird wiederholt, bis alle Punkte besucht wurden.

Eine hohe Anzahl von Rauschpunkten kann darauf hinweisen, dass die Parameterwahl oder die Klassifikation nicht optimal ist [49].

6.6 Weitere Algorithmen für zukünftige Studien

Als Alternative zu DBSCAN wurde auch das hierarchische Verfahren HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) in Betracht gezogen. HDBSCAN ist in vielerlei Hinsicht robuster als DBSCAN, insbesondere bei der Identifikation von Clustern in hochdimensionalen oder verrauschten Daten [50]. Ein Vorteil von HDBSCAN besteht darin, dass keine globale Dichte-Schwelle erforderlich ist und sich das Verfahren besser an lokale Datenstrukturen anpasst. Für zukünftige Arbeiten wäre eine Untersuchung von HDBSCAN in Kombination mit LSTM-Autoencodern denkbar.

Darüber hinaus wurden weitere Anomalieerkennungsverfahren evaluiert, darunter K-Means und LOF (Local Outlier Factor). K-Means wird in der Studie von Vences et Al. als besonders effektiv eingeschätzt [51]. LOF erkennt Ausreißer, indem die lokale Dichte eines Datenpunkts mit der seiner Nachbarn verglichen wird; Punkte mit stark abweichender Dichte werden als potenzielle Anomalien markiert. Die Wirksamkeit von LOF zur Anomalieerkennung wurde von Breunig nachgewiesen [52].

Beide Verfahren eignen sich grundsätzlich für die Anomalieerkennung. Budiarto et Al. zeigen jedoch, dass OCSVM mehr Anomalien erkennen kann als die beiden Verfahren [53]. Zudem leidet LOF unter dem „Fluch der Dimensionalität“, da Dichte-Schätzungen im euklidischen Raum (z.B. k-Nächste-Nachbarn-Distanzen) in hochdimensionalen Räumen an Aussagekraft verlieren [54, S.366].

DBSCAN, OCSVM und Isolation Forest wurden trotz ähnlicher Einschränkungen ausgewählt, um eine möglichst hohe Varianz in der Methodenauswahl zu gewährleisten. Algorithmen mit stark ähnlicher Funktionsweise, wie K-Means (Clustering-Ansatz ähnlich DBSCAN) oder LOF (Parallelen zum Isolation Forest), könnten in weiteren Studien noch genauer untersucht werden. Ein weiterer Entscheidungsgrund war die Skalierbarkeit: K-Means ohne Optimierungsmechanismen bspw. stößt bei großen Datensätzen an Grenzen [55].

Auch das LOF-Verfahren kann bei großen Datenmengen ohne Performance-Optimierung hohe Rechenzeiten verursachen; entsprechende Ansätze zur Verbesserung der Laufzeit, etwa durch Block-Größe-Optimierung, werden derzeit erforscht [56].

Wie im Abschnitt zu Autoencodern bereits erläutert, existieren diverse Varianten, die potenziell eine präzisere Anomalieerkennung ermöglichen. In der vorliegenden Arbeit wird jedoch auf eine Evaluierung dieser Alternativen verzichtet, da der Fokus auf dem LSTM-Autoencoder in Kombination mit anderen Algorithmen liegt. Ausschlaggebend hierfür ist insbesondere die vergleichsweise einfache Implementierung des LSTM-Autoencoders sowie die Verfügbarkeit detaillierter Anleitungen zu dessen Aufbau und Training.

7 Implementierung

7.1 Hybridmodelle mit LSTM-AE

Die Modelle wurden nach einer einheitlichen Architektur implementiert:.

Parameter	Wert
Batch-Size	32
encoder_layers	[128, 64]
decoder_layers	[64, 128]
Learning Rate	0.001
epochs	50

Tabelle 1: Ausgewählte Parameter der Modellkonfiguration

Die Encoder- und Decoder-Architektur umfasst jeweils zwei Schichten mit den Dimensionen 128 und 64. Diese Konfiguration wurde gewählt, um eine ausreichende Komprimierung der zeitlichen Muster zu ermöglichen, ohne dass wesentliche Informationen verloren gehen. Eine geringere Breite würde die Informationsrepräsentation einschränken, während eine größere Tiefe zu Überanpassung führen könnte. Diese Parameterauswahl orientiert sich an Toor et al. [57], welche in LSTM-basierten Architekturen erfolgreich eingesetzt wurde.

Die Lernrate von 0.001 hat sich laut Santos et Al. als stabil erwiesen und bietet Vorteile gegenüber kleineren Lernraten wie 0.0001 [58]. Die Batch-Size von 32 wurde gewählt, da kleine Batches laut Masters und Luschi [59] die stabilste Trainingsdynamik erzielen.

Zudem wird *EarlyStopping*²¹ eingesetzt. Diese Technik verhindert Overfitting, indem das Training automatisch abgebrochen wird, sobald die Validierungsleistung über eine definierte Anzahl von Epochen (Patience) keine Verbesserung zeigt. Obwohl die maximale Epochenzahl auf 50 gesetzt ist, wird diese nicht zwingend erreicht; die tatsächlich trainierten Epochen variieren je nach Modell. Dabei wurde Patience auf 2 festgelegt. Nach Hussein et Al. wurde erkannt, dass eine Patience von 2 bei 20 Epochen ausreichend ist für einen geringen Validierungsverlust [60]. Auch wenn die Patience von 5 die höchste Accuracy erzielte- waren diese Unterschiede sehr minimal. Man geht davon aus, dass dementsprechend eine Patience von 2 ausreichen. Da die Ressourcen dieser Arbeit (darunter auch Zeit) zudem begrenzt sind, entschied man sich darauf die Patience auf diese Anzahl zu setzen.

Die Daten werden zunächst in numerische Form überführt, falls sie noch nicht numerisch sind. NaN- oder unklare Werte werden bewusst nicht bereinigt, sondern als eigene Kategorie behandelt, um potenzielle Anomalien nicht zu entfernen. Diese Transformation erfolgt mittels LabelEncoder²². Anschließend werden die Daten mit dem StandardScaler normalisiert²³.

Obwohl die Mehrheit der Merkmale kategorial war, wurde auf ein One-Hot-Encoding²⁴ bewusst verzichtet. Stattdessen kam ein Label-Encoding zum Einsatz, um die Eingabedimension kompakt zu halten und eine effiziente Sequenzverarbeitung im LSTM zu ermöglichen. Ein One-Hot-Encoding hätte bei den zahlreichen Kategorien zu einer stark erhöhten Vektordimension geführt und damit sowohl Trainingszeit als auch Speicherbedarf deutlich gesteigert. Für die vorliegende Aufgabe der Anomalieerkennung ist eine kontinuierliche numerische Re-

präsentation ausreichend, da das Ziel nicht die Klassifikation einzelner Kategorien, sondern die Erkennung von Abweichungen in den Sequenzmustern ist.

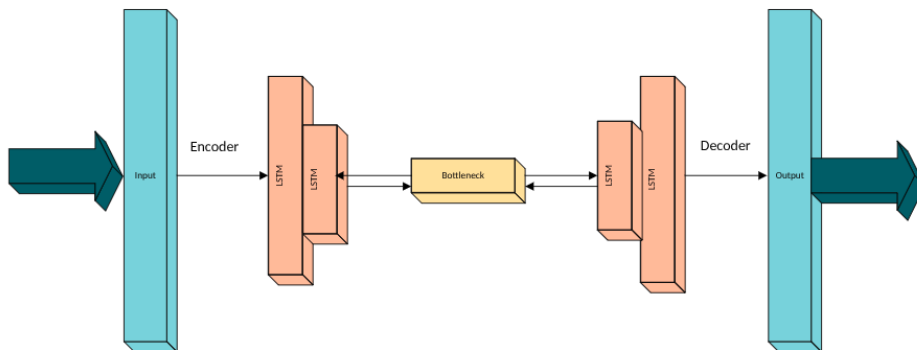


Abbildung 8: Gesamtarchitektur des LSTM-Autoencoders

Die Rekonstruktionsfehler des LSTM-AE werden anschließend von den Anomalieerkennungs-Algorithmen verarbeitet, wie auch in Malhotra et al. [61] gezeigt wird. Diese modulare Struktur erlaubt ein separates Training, Testen und Evaluieren der einzelnen Modelle. Die Implementierung erfolgt in einem Jupyter Notebook.

Encoder-Decoder Struktur und Bidirektionales LSTM Die Architektur folgt dem Prinzip eines Encoder-Decoder-Systems, bei dem der Encoder die wesentlichen Merkmale der Eingabesequenzen in einem kompakten latenten Raum repräsentiert. Dieser latente Raum fasst die zeitlichen Muster zusammen und ermöglicht eine effiziente Datenrepräsentation. Das bidirektionale LSTM im Encoder erfasst dabei sowohl Informationen aus der Vergangenheit als auch aus der Zukunft der Sequenz. Dies ist insbesondere bei der Erkennung von Angriffs- oder Anomalieszenarien von Vorteil, da potenziell verdächtige Ereignisse oft nur im Kontext benachbarter Zeitpunkte eindeutig identifiziert werden können. Der Decoder nutzt die latente Repräsentation, um die ursprüngliche Sequenz zu rekonstruieren, wodurch eine präzise Nachbildung der Eingabedaten ermöglicht wird. Nach Acharya et Al. eignet sich ein bidirektionales LSTM optimal zur Anomalieerkennung [62].

Verlustfunktion und Optimierung Zur Bewertung der Modellgüte wird die mittlere quadratische Abweichung (Mean Squared Error (MSE) ²⁵) herangezogen, welche die Differenz zwischen Eingabe- und Ausgabesequenzen quantifiziert. MSE eignet sich zur Messung der Rekonstruktionsqualität bei kontinuierlichen Werten, da größere Fehler durch die Quadratisierung stärker gewichtet werden, was eine präzise Anpassung des Modells erzwingt.

Die Optimierung erfolgt mittels des Adam-Optimierer-Optimierers ²⁶, welcher durch Lernraten für jeden Parameter besticht und somit eine schnelle und stabile Konvergenz auch bei komplexen Modellen gewährleistet. Die Lernrate stellt dabei einen wesentlichen Hyperparameter dar, der das Tempo der Gewichtsaktualisierungen steuert und einen Balanceakt zwischen schnellem Lernen und stabiler Anpassung darstellt.

Gudla et Al. zeigen bspw., dass der Adam-optimierer sich mit einer Lernrate von 0.001 optimal für LSTM-Netzwerke eignet, um bspw. DDoS-Angriffe zu erkennen [63].

Berechnung des Rekonstruktionsfehlers Die Idee, dass sich Mean Square Error den Rekonstruktionsfehler ist inspiriert von einer Studie von Torabi et Al. [64, S. 5]. Die allgemeine Formel zur Berechnung des Fehlers lautet wie folgt:

$$\text{Reconstruction Error} = \frac{1}{T \cdot F} \sum_{t=1}^T \sum_{f=1}^F (x_{t,f} - \hat{x}_{t,f})^2$$

Im Grunde genommen wird hier jeder Fehler als Zeitpunkt dargestellt. Jeder Zeitpunkt (I) und jedes Feature F stellt hier eine Abhängigkeit dar. Alle rekonstruierbaren Werte werden von den nicht-rekonstruierbaren über jedes Logs hinweg berechnet – und anschließend wird der Durchschnitt berechnet. Dies ist dann der gesamte Rekonstruktionsfehler.

Ermittlung des Schwellenwertes Die Frage, ab welchem Punkt ein Modell eine Anomalie erkennen sollte, ist entscheidend für die Güte der Detektion: Soll bereits eine geringe Abweichung eines Features als Anomalie gelten, oder erst größere Diskrepanzen? Jablonski et al. [65] schlagen hierfür einen algorithmischen Ansatz vor: Zunächst werden alle Rekonstruktionsfehler aus den Daten extrahiert. Anschließend wird der optimale Schwellenwert algorithmisch anhand der größten Lücke im sortierten Fehlervektor bestimmt – also zwischen den beiden Werten, die im Fehlervektor am weitesten auseinanderliegen.

Die Arbeit von Jablonski et Al. inspirierte dazu, einen eigenen Ansatz zu entwickeln: Zunächst werden alle Rekonstruktionsfehler aus den Daten extrahiert und sortiert. Der optimale Schwellenwert wird dann als die Mitte zwischen den beiden Werten mit dem größten Abstand im sortierten Fehlervektor definiert.

In dieser Arbeit wurde dieser erfundene Ansatz jedoch verworfen, da er zu geringeren Precision- und Recall-Werten in allen Hybridmodellen führt. Vermutlich liegt dies daran, dass Anomalien im Verhältnis zur Gesamtanzahl normaler Logs sehr selten auftreten. Für eine zuverlässige Anomalieerkennung in Zeitreihen ist es daher notwendig, die Rohwerte der Anomalie-Scores (also die vom Modell berechneten Rekonstruktionsfehler) in binäre Entscheidungen zu überführen. Hierzu wird ein Schwellenwert festgelegt: Werte oberhalb des Schwellenwerts werden als Anomalien klassifiziert, Werte darunter als normal.

Die Wahl des Schwellenwerts beeinflusst direkt das Verhältnis zwischen Precision (Anteil kor-

rekt erkannter Anomalien unter allen als anomal klassifizierten Punkten) und Recall (Anteil erkannter Anomalien unter allen tatsächlichen Anomalien). Um dieses Verhältnis optimal zu berücksichtigen, wird der F1-Score als kombinierte Kennzahl verwendet. Der Schwellenwert wird so gewählt, dass der F1-Score maximiert wird. Es ist auch in der Literatur als verlässliche Methode zur Bestimmung von Schwellenwerten etabliert [66]. Dabei gehen Wilkinghof et Al. noch ein wenig weiter und schlagen einen neueren Ansatz vor, nämlich, den "F1-EV Score" zu verwenden. Man entschied sich jedoch nur dazu den F1-Score zu optimieren, weil dies praktischer umzusetzen war in dieser Arbeit.

7.1.1 Implementierung: Isolation Forest und OCSVM

Anstatt die standardmäßige Vorhersage (-1 = anomal, 1 = normal) direkt zu verwenden, werden die Rohscores²⁷ des Modells genutzt, um mittels einer F1-basierten Schwellenwert-Optimierung [66] eine binäre Klassifikation zu erzeugen. Für jeden Testpunkt wird dabei ein optimaler Schwellenwert bestimmt, der die F1-Metrik maximiert. Alle Punkte mit Scores oberhalb dieses Schwellenwerts werden als Anomalien (1) klassifiziert, alle anderen als normal (0). Diese Vorgehensweise ist besonders sinnvoll, da Anomalien in Zeitreihendaten häufig selten auftreten und eine Balance zwischen Precision und Recall entscheidend ist.

Da die Daten sequenzbasiert aufgebaut sind, wird jedes Fenster der Sequenz bewertet: Wenn innerhalb des Fensters mindestens ein Punkt anomal ist, erhält das Fenster das Label 1, sonst 0. Dadurch werden die binären Vorhersagen konsistent mit den Testlabels verglichen.

Dieser Ansatz erfolgt für IF und OCSVM.

Die Parameter des IF werden folgendermaßen gesetzt:

- contamination= 0.0008
- random_state= 42
- max_samples= auto

Die Parameter des OCSVM werden folgendermaßen gesetzt:

- contamination= nu
- random_state= 42
- kernel= rbf
- gamma= scale

Die Setzung von "contamination" und "nu" beruht auf den erwarteten Anomalien (wird in den Kapiteln bzgl. der Ansätze genauer erläutert). Der Parameter max_samples wird

auf Auto gesetzt, damit IF flexibel auf größere Datensätze reagiert. Der Parameter "random_state" bleibt dem Standardwert 42. Bei OCSVM bleibt der Standardkernel auf "rbf" gesetzt für nicht-lineare Verarbeitung, sowie der Standardwert β scale bei "gamma".

7.1.2 Implementierung: DBSCAN

Das DBSCAN-Modell wird direkt auf die skalierten Rekonstruktionsfehler der Testdaten angewendet. Dabei wird der Parameter `eps` automatisch über ein Raster von 0.01 bis 1.0 angepasst, sodass die Anzahl der als Anomalien klassifizierten Punkte ungefähr der erwarteten Anzahl an Anomalien entspricht. `min_samples` wird auf 3 gesetzt, und die Distanzmetrik ist `euclidean`.

Die Methode `fit_predict()` ²⁸ liefert für jeden Punkt entweder eine Clusterzuweisung (0, 1, ...) oder -1. Punkte mit der Kennzeichnung -1 werden als Anomalien interpretiert.

Auf Basis der Vorhersagen werden verschiedene Evaluationsmetriken berechnet, darunter Precision, Recall, F1-Score, ROC-AUC, PR-AUC, Accuracy, Matthews-Korrelationskoeffizient (MCC) und Balanced Accuracy, um die Leistungsfähigkeit des Modells zu beurteilen.

7.2 Alternative Architekturen

Es ist ebenfalls möglich, zunächst eines der Algorithmen zu trainieren und anschließend dieselben Daten in den LSTM-Autoencoder einzuspeisen. Damit kann auch ein Hybridmodell in umgekehrter Reihenfolge, beispielsweise zwischen Isolation Forest und LSTM-AE, erstellt werden [67]. In der Studie von Priyanto wurde zunächst IF verwendet, um die Rohdaten automatisch zu klassifizieren und sogenannte Pseudo-Labels für Anomalien und normale Zustände zu generieren [67]. Diese vorläufige "Labelung" diente dazu, das Training des nachfolgenden LSTM-Autoencoders zu unterstützen. Das LSTM-Autoencoder-Modell wurde anschließend mit den so gelabelten Daten trainiert, um zeitliche Muster besser zu erfassen und Anomalien präziser zu erkennen. Durch diese Kombination aus einem schnellen, unüberwachten Verfahren zur Vorfilterung und einem sequenzbasierten Modell zur detaillierten Analyse konnte die Anomalieerkennung verbessert werden.

Darüber hinaus wurden auch größere Hybridmodelle diskutiert. Beispielsweise wurde eine Kombination aus LSTM-AE, DBSCAN und Isolation Forest vorgeschlagen: Die Daten werden zunächst vom LSTM-Autoencoder verarbeitet, die daraus resultierenden Fehler anschließend von DBSCAN geclustert, um unterschiedliche Fehlerarten zu identifizieren. Abschließend entscheidet der Isolation Forest, welche Punkte als Anomalien klassifiziert werden. Diese Variante wurde jedoch nicht umgesetzt, da sie zu komplex ist, insbesondere, weil in dieser Arbeit nur verhaltensbezogene Anomalien aus Event-Logs untersucht werden.

Darüber hinaus ermöglichen einfachere hybride Modelle eine leichtere Interpretierbarkeit und Wartbarkeit, was in produktiven Systemen zu höherer Stabilität führt.

Die ergänzenden klassischen Algorithmen dienen hier vor allem zur Unterstützung bei der Erkennung von Ausreißern oder Clustern, wodurch eine effiziente und dennoch aussagekräftige Erkennung gewährleistet wird.

Durch die klare Trennung der Methoden in der gewählten Architektur lassen sich zudem die jeweiligen Stärken und Schwächen besser analysieren und bewerten, was den wissenschaftlichen Vergleich und die spätere Optimierung erleichtert.

Für das LSTM-AE wurde zudem noch überlegt, sogenannte *Residual Blocks* einzubauen. Diese Blöcke fügen der normalen Verarbeitung eine direkte Verbindungsleitung zwischen Ein- und Ausgangsschicht hinzu, sodass die Eingabe unverändert zum Ausgang addiert wird. Dadurch kann das Netzwerk lernen, nur die *Differenz* (Residual genannt) zwischen Ein- und Ausgang zu modellieren. Diese Architektur erleichtert das Training tiefer Netzwerke, da sie beim Backpropagation-Prozess den Gradientenfluss verbessert und das Problem des Vanishing Gradient reduziert. Ohne Residual Blöcke kann der Gradient in sehr tiefen Netzen zu klein werden, wodurch das Lernen erschwert wird. Mit Residual-Verbindungen wird dieser Effekt abgefedert, was zu stabilerem und effizienterem Training führt. Auch nach der Studie nach He et Al. ist dies bspw. gut zur Erkennung von Bildern [68].

Dennoch wurde diese Architektur verworfen, da Residual Blöcke insbesondere bei kleineren Trainingsdatensätzen, wie in dieser Arbeit mit etwa wenigen Datenpunkten, wenig Mehrwert bieten und unter Umständen sogar kontraproduktiv sein können. Die Blöcke kommen vor allem bei sehr großen Datensätzen und tiefen Netzwerken mit mehreren Hunderttausend bis Millionen Trainingsbeispielen sinnvoll zum Einsatz. In zukünftigen Arbeiten kann es einen hohen Mehrwert jedoch bieten.

Für den Autoencoder bestand noch die Möglichkeit, Deep Autoencoder-Modell zu verwenden, welches sogenannte *Attention Layer* [69] hat. Dies wurde jedoch auch verworfen, da für den Testfall dieser Arbeit wie bereits erwähnt keine zu komplexe Architektur notwendig ist.

8 Generierung der Logs

Es standen verschiedene Ideen zur Verfügung, mit welchen Daten man die Modelle trainieren kann. Es wurden zwei mögliche Ansätze bearbeitet. Beim ersten wurden die Logs automatisch durch Python-Skripts erzeugt. Der zweite erzeugte die Tests zum Großteil durch automatisierte Tests in Keycloak selbst und manche Angriffe wurden manuell ausgeführt. Schlussendlich entschied man sich für den letzten Ansatz, da man in diesem Fall Logs aus Keycloak direkt hat und keine selbst interpretierte Struktur der Logs erfunden werden musste wie beim ersten Ansatz.

8.1 Erster Entwurf der Angriffsfälle

Zu den häufigsten Insider-Threat-Angriffen in IT-Systemen zählen laut aktuellem Forschungsstand unter anderem der Missbrauch privilegierter Rollen und Rechte, Datendiebstahl sowie die Manipulation kritischer Daten, wie etwa das Löschen wichtiger Dateien [70, S.6]. Auch wenn der Dictionary-Angriff als Angriff nicht genannt wird, gehört er zu den bekannteren Intrusionsangriffen.

Die folgenden Angriffsszenarien wurden daher modelliert und in die Log-Daten integriert:

- *Anwendungsfall 1:* Dictionary-Angriffe bei der Anmeldung
- *Anwendungsfall 2:* Löschen und Verändern sensibler Konfigurationen wie Clients, Benutzer, Passwörter und Realms
- *Anwendungsfall 3:* Erlangen privilegierter Rollen, z.B. manage-users und realm-admin

Darüber hinaus wurden bekannte Anomalien wie ungewöhnliche IP-Adressen und Login-Zeitpunkte bei der Generierung der Daten berücksichtigt. Besonderes Augenmerk liegt dabei auf einer hohen Variabilität der Angriffsformen, um die Aussagekraft der Tests zu erhöhen.

8.2 Automatische Erstellung der Keycloak-Logs

Die Log-Dateien werden mithilfe eines Python-Skripts automatisiert generiert.

Der erste Anwendungsfall simuliert Dictionary-Angriffe, indem eine Abfolge mehrerer Log-Einträge mit dem Event-Typ `LOGIN_ERROR` erzeugt wird. Diese Einträge können ungewöhnliche Zeitstempel oder IP-Adressen enthalten, was jedoch nicht zwingend notwendig ist – bereits die Häufung von `LOGIN_ERROR`-Ereignissen deutet auf ein anomales Verhalten hin. Da Keycloak die Sperrung einer IP-Adresse nach einer konfigurierbaren Anzahl fehlgeschlagener Anmeldeversuche ermöglicht, wird dieser Parameter bei der Log-Erzeugung berücksichtigt. Die Anzahl der erzeugten fehlerhaften Logins variiert zufällig zwischen drei und fünfzehn Einträgen. Diese Auswahl basiert auf praktischer Erfahrung: Bereits ab drei gescheiterten Versuchen wird das Verhalten oft als verdächtig eingestuft – auch wenn es sich nicht immer um einen Angriff handelt.

Im zweiten Anwendungsfall werden Log-Einträge erzeugt, die auf das Löschen oder Modifizieren von Entitäten hinweisen. Hierzu gehören beispielsweise Event-Typen wie `DELETE`, `UPDATE_PASSWORD` oder `UPDATE_PROFILE`. Es werden mindestens fünf (bis 15) solcher Einträge pro Sequenz erzeugt, wobei auf ausreichend Variabilität geachtet wird. Die Ereignisse werden in zufälligen Kombinationen generiert, um zu vermeiden, dass das Modell nur eine festgelegte Angriffssignatur erlernt. Würde beispielsweise das Feature `authType` ständig zusammen mit dem Event-Typ `UPDATE` auftreten, bestünde die Gefahr des Overfittings: Das Modell würde dann nur dieses konkrete Muster als Anomalie erkennen und andere

Formen übersehen.

Im dritten Anwendungsfall werden Angriffe inszeniert, in denen ein Nutzer, der ursprünglich keine privilegierten Rollen hatte, plötzlich Administratorrechte erhält. Bei der Generierung wird sichergestellt, dass die betreffende Session einen Benutzer enthält, der zu Beginn keine Admin-Rollen besitzt und später Rollen wie `realm-admin` oder `manage-users` zugewiesen bekommt.

Normale Benutzer-Sessions enthalten keine ungewöhnlichen IP-Adressen oder Zeitstempel. Als ungewöhnliche Uhrzeiten gelten in dieser Arbeit Zugriffe zwischen 0:00 Uhr und 6:00 Uhr morgens.

Die Definition ungewöhnlicher IP-Adressen orientiert sich überwiegend an den von der IANA bereitgestellten Daten²⁹.

```
Tests > {} test_logs.json > ...
23 {
24   "authDetails": {
25     "ipAddress": "43.243.100.34",
26     "realmId": "aws",
27     "clientId": "security-admin-console",
28     "sessionId": "b9c7bbdf-4b98-478d-af91-93c31f9a457a"
29   },
30   "operationType": "CREATE",
31   "resourceType": "groups",
32   "resourcePath": "groupss/d75a63cb-1754-49fa-9b29-ac419e9f1bca",
33   "label": 0
34 },
35 {
36   "timestamp": "2025-07-20T19:00:27.042229+02:00",
37   "log_level": "error",
38   "category": "org.keycloak.events",
39   "type": "LOGIN_ERROR",
40   "realmId": "telekom",
41   "realmName": "telekom",
42   "clientId": "admin-cli",
43   "ipAddress": "198.205.39.117",
44   "error": "invalid_credentials",
45   "username": "Luca",
46   "details": {
47     "auth_type": "oauth2-client-credentials",
48     "connection": "openid-connect",
49     "scope": "phone",
50     "grant_type": "authorization code",
51     "refresh_token_id": "9bce28bc-e800-4650-8461-f04c5bbf5099",
52     "code_id": "9d363895-f4b3-45b3-8e96-834ecb61a42b"
53   },
54   "label": 1
55 },
56 {
57   "timestamp": "2025-07-20T19:06:36.534525+02:00",
```

Abbildung 9: Ausschnitt der generierten Logs, Man erkennt hier schon ein Log mit dem Label 0, also ein normaler Log, unterhalb ist eines mit dem Label 1- eine Anomalie. Man erkennt hier schon den Typ `LOGIN_ERROR`.

Bei der Generierung der Log-Daten wurde auch berücksichtigt, dass neben eindeutig normalen und anomalen Sessions auch uneindeutige Sequenzen erzeugt werden, die als *Noise* klassifiziert werden können. Solche Fälle treten in realen Systemen häufig auf – etwa wenn ein normaler Nutzer temporär priorisierte Rollen übernimmt oder häufig Passwörter ändert, ohne dass ein tatsächlicher Angriff vorliegt. Um eine zu starke Sensitivität der Modelle gegenüber solchen Mustern zu vermeiden, wird bewusst ein gewisser Anteil an Rauschen erzeugt.

8.3 Grenzen dieser Lösung

Die Analyse basierte ausschließlich auf den wichtigsten, verfügbaren Log-Daten, die bekannte Systemaktivitäten und deren Merkmale widerspiegeln. Durch diesen Ansatz hat man die Kontrolle über die Logs und wie viele ca. erzeugt werden. Eine tiefere Untersuchung der internen Struktur und des vollständigen Aufbaus der Event-Typen in Keycloak war aufgrund fehlender umfassender Dokumentation und eingeschränktem Zugriff auf administrative Systemressourcen nicht möglich. Diese Einschränkungen begrenzen die Tiefe und Genauigkeit der Auswertung und können die Aussagekraft der Ergebnisse beeinflussen.

Eine zentrale Herausforderung bestand in der realistischen Modellierung anomaler Sessions. Dabei musste unterschieden werden, ob eine Session normal, anormal oder als sogenannter Noise einzustufen ist, also als legitimes, aber ungewöhnliches Verhalten, das fälschlich als Anomalie interpretiert werden könnte. Da echte Anomalien in der Realität selten auftreten, wurde bei der Datengenerierung heuristisch eine Auftrittswahrscheinlichkeit von 2, % für anomale Sessions angenommen. Dadurch ist nicht gewährleistet, dass die erzeugten Daten eine realistischere Verteilung widerspiegeln.

Ein weiterer Nachteil der synthetischen Generierung liegt darin, dass zufällig zusammengesetzte Ereignismuster entstehen können, die keinem realistischen Angriff entsprechen. Um dem entgegenzuwirken, wurde eine Logik implementiert, die grundlegende Systemregeln von Keycloak berücksichtigt: So darf ein Benutzer mit eindeutiger ID nur in einem Realm existieren, es gibt jeweils nur einen Administrator mit explizit definierten Rollen und höherer Priorität, und normale Benutzer können keine Operationen ausführen, die mit dem Attribut *operationType* (CRUD-Operationen) verbunden sind — diese bleiben ausschließlich Administratoren vorbehalten. Zudem wird sichergestellt, dass jeder Benutzer genau einer Session und einem Realm zugeordnet ist, sofern es sich nicht explizit um einen Angreifer handelt.

Aufgrund dieser Einschränkungen sowie der im späteren Verlauf beobachteten Resultate wurde der vorgestellte Ansatz verworfen. Die Ergebnisse waren dabei sehr gut, was auch daran lag, dass mehr Anomalien erzeugt wurden, als in der Realität auftreten würden. Zudem können durchaus Logs bis 100.000 generiert werden. Für diesen Fall wurden auch längere Sequenzlängen verwendet und der LSTM-AE wurde sozusagen mit mehr Daten trainiert, wie es in der Praxis üblich ist.

9 Zweiter Ansatz: Manuelle Log-Generierung und durch Selenium

Wegen den zuvor genannten Nachteilen am ersten Ansatz, wurde stattdessen ein sinnvollerer Ansatz gewählt: Die Daten werden direkt aus dem Keycloak-System erhoben, um reale Anwendungsfälle abzubilden. Es wurde die Idee verfolgt, eine lokale Keycloak-Instanz auf-

zusetzen und von dieser Instanz aus die Logs zu erheben und zu nutzen. Dazu müssen die Logs jedoch manuelle erstellt werden. Da auch dieser Weg zeitintensiv ist, wurde zusätzlich erwogen, mittels sogenannter Selenium-Tests sowohl Angriffsszenarien als auch gewöhnliche Keycloak-Daten zu simulieren. So durch konnten viel schneller Logs erzeugt werden und es wurde viel Arbeitsaufwand erspart. Eine manuelle Durchführung wäre praktisch nicht möglich gewesen, da man dafür mehrere Benutzer gebraucht hätte, die diese Instanz nutzen und auch dabei täglich. Durch Selenium wurde gewährleistet, dass man immer noch Kontrolle darüber hat, welche Logs generiert werden und wie viele. Als Grundlage zur Definition normaler Keycloak-Logs dient dabei eine Orientierung an firmeninternen Log-Daten. Somit wird auch vermieden sensible Kundendaten (also Keycloak-Logs aus Keycloak-Instanzen der Kunden) zu erheben.

9.1 Anbindung der Keycloak-API

Um Zugriff auf die Keycloak-Logs zu erhalten, ist es erforderlich, eine eigene Keycloak-Instanz zu installieren und sich mit Administratorrechten anzumelden. Die Standard-Werte sind dafür als Benutzername sowie Passwort „admin“.

Der Zugriff auf die Event-Logs erfolgt über die Keycloak-API, welche insbesondere bei der verwendeten Quarkus-Distribution eingesetzt wird. Standardmäßig benötigt man hierfür entsprechende Administrator-Rollen, um sensible Informationen wie User-Event-Logs und Admini -Event-Logs abrufen zu können. Zunächst müssen die korrekten Authentifizierungsdaten vorliegen, um die erforderlichen Zugriffsrechte zu erhalten.

Für die Authentifizierung wurde der sogenannte *Client Credentials* Grant-Typ gewählt. Dabei erfolgt die Anmeldung über einen registrierten Client, wobei neben dem Client-Secret weitere Zugangsdaten übermittelt werden.

```
def get_token():
    data = {
        'client_id': client_id_local,
        'client_secret': client_secret_local,
        'grant_type': 'client_credentials'
```

Abbildung 10: Daten, die an den Keycloak-Server gesendet werden müssen

Durch diesen Vorgang wird ein Zugriffstoken generiert, welches für den Abruf der Event-Logs benötigt wird. Das Token enthält Informationen zu den Zugriffsrechten des jeweiligen Admins. So müssen beispielsweise die Rechte zum Verwalten von Benutzern enthalten sein (also `manage-users`), damit diese auch im Token ausgewiesen werden. Ebenfalls benötigt amn die Berechtigung überhaupt auf Events zugreifen zu dürfen. Dafür wurde dem Admin die Rolle `view-events` zugeteilt. Anschließend wird mit diesem Token eine weitere Anfrage an die

API gestellt, um die Event-Logs abzurufen. Die erhaltenen Daten werden in einer separaten JSON-Datei gespeichert.

Weitere Details zur Keycloak-API sind in der offiziellen Dokumentation zu finden³⁰.

9.2 Datenbeschaffung durch Selenium

Selenium³¹ ist ein Testframework zur automatisierten Ausführung von Webbrowser-Aktionen und als Python-Paket verfügbar. Es ermöglicht, Interaktionen mit der Benutzeroberfläche realitätsnah zu simulieren, sodass der Eindruck entsteht, ein Mensch bediene den Browser. Die wesentlichen Komponenten sind:

- **Selenium WebDriver:** Steuert den Browser programmgesteuert und stellt das zentrale Element der Tests dar.
- **Selenium Grid:** Ermöglicht die parallele Ausführung von Tests auf verschiedenen Maschinen und Browsern.
- **Selenium IDE:** Ein Recording-Tool im Browser, das jedoch für produktive oder komplexe Tests nicht empfohlen wird.

Für die Erstellung der Logs wird ausschließlich der WebDriver verwendet, der automatisch Chrome als Browser startet. Da Selenium anwendungsagnostisch ist, war das Steuern von Keycloak damit besonders leicht. Die Einrichtung gestaltet sich einfach: Es wird der zu verwendende Browser („Driver“) definiert und die Tests werden auf dem Port ausgeführt, auf dem die lokale Keycloak-Instanz erreichbar ist.

Ein Nachteil dieser Methode liegt in der konstanten IP-Adresse, da alle Anfragen vom selben Host ausgehen. Dies kann dazu führen, dass das Modell lernt, mehrere Benutzer mit einer gemeinsamen IP-Adresse zu assoziieren, was in realen Szenarien als Anomalie interpretiert werden könnte. Daher wird die IP-Adresse vor der Weiterverarbeitung aus den Events entfernt. Diese Maßnahme reduziert zwar eine potenzielle Störquelle, verringert jedoch auch die Datenvielfalt für das Modell. Jedoch bestehen weiterhin genug Features, mit denen die Modelle arbeiten können, wie bspw. dem Event-Typen.

Zudem erfolgt die Log-Erzeugung in kurzen zeitlichen Abständen, wodurch die zeitliche Streuung reduziert wird. Dies kann die Fähigkeit der Modelle zur Anomalieerkennung einschränken. Zwar werden wenige Tests zu ungewöhnlichen Uhrzeiten durchgeführt (z.B. abends), jedoch wurden davon nicht zu viele durchgeführt, damit die Modelle nicht lernen, dass eine Uhrzeit außerhalb der gewöhnlichen Arbeitszeit „normalist“.

Trotz dieser Einschränkungen überwiegt der Vorteil, dass die durch Selenium erzeugten Logs den realen Keycloak-Logs sehr nahekommen – ein zentraler Aspekt, insbesondere wenn wirtschaftlich verwertbare Erkenntnisse aus dieser Arbeit gewonnen werden sollen. Da die Logs

direkt durch die definierten Tests erzeugt und gespeichert werden, entfällt die Notwendigkeit einer separaten manuellen Log-Generierung größtenteils.

Zur Generierung der normalen Trainingsdaten wurden zuvor vier Realms erstellt:

- ubisoft
- Nintendo
- Sega
- aws

Pro Realm erfolgt die Benutzeranmeldung über einen jeweils definierten Client. Absichtlich wurden keine Benutzer und Clients im Master-Realm erzeugt, da auf diesen standardmäßig andere Admins aus den Nicht-Master-Realms nicht zugreifen können. Durch das Weglassen des Master-Realms bei der Generierung normaler Benutzerlogs wurde sichergestellt, dass die aufgezeichneten Logs ausschließlich reale Benutzeraktivitäten aus den einzelnen Realms widerspiegeln, ohne dass administrative Operationen oder systeminterne Aktionen die Analyse verfälschen. Dies erhöht die Klarheit und Vergleichbarkeit der Anomalieerkennung, da nur typische Nutzerinteraktionen untersucht werden und auch nur die der internen Admins in den einzelnen Realms.

Im Realm *ubisoft* wurden 25 synthetische Benutzer erstellt, deren Namen alphabetisch generiert wurden (jeder Buchstabe des Alphabets außer „X“ ist vertreten, da dafür kein Name gefunden wurde). Für diesen Realm wurde der Client *ps3* angelegt. Dieser ist erforderlich, damit sowohl Admins Zugriff auf die User-Events erhalten als auch Benutzer sich über diesen Client authentifizieren können.

In den weiteren Realms – *Sega*, *Nintendo* und *aws* – wurden jeweils zehn Benutzer angelegt. Die zugehörigen Clients lauten: *megadrive* (Sega), *wii* (Nintendo) und *aws-console* (aws).

Jeder Realm verfügt über genau einen Admin mit der Rolle *realm-admin*. Ein Admin, der für einen bestimmten Client Benutzer anlegen oder verwalten möchte, benötigt darüber hinaus die Rollen *manage-users* und *view-events*. Wichtig ist hierbei, dass diese Rollen nicht direkt den Benutzern, sondern den jeweiligen Clients zugewiesen werden. Dies liegt in der Architektur von Keycloak begründet: Clients besitzen eigene Zugriffsrechte, die ihnen bestimmte Operationen ermöglichen. So erlaubt die Rolle *manage-users* dem Client, Benutzer im Realm programmatisch zu verwalten, während *view-events* erforderlich ist, um Zugriff auf die Ereignisprotokolle zu erhalten – essenziell für Logging und Fehlersuche.

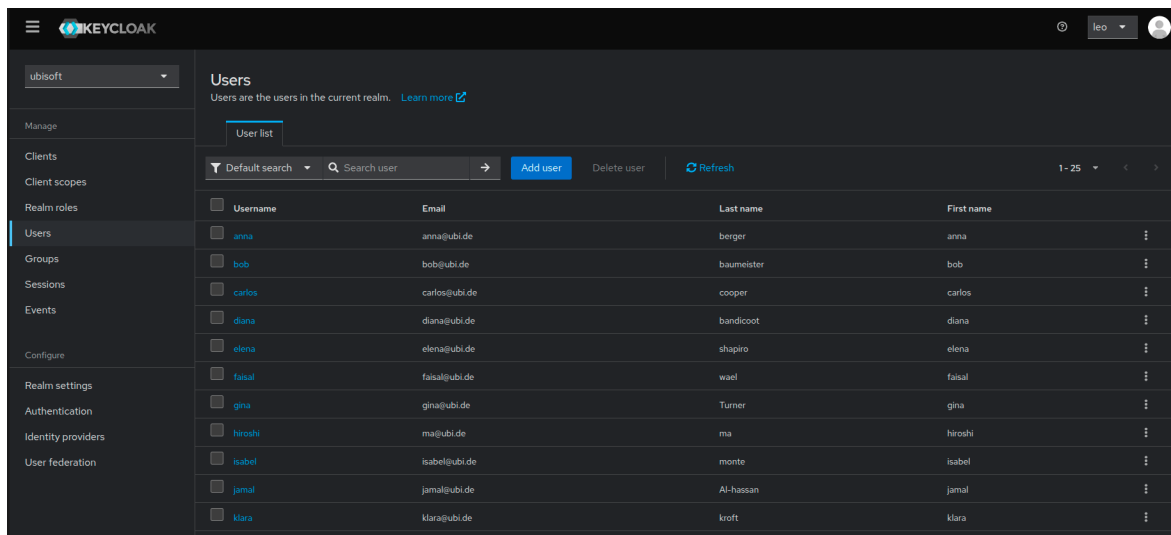


Abbildung 11: Beispielhafte Darstellung der generierten Benutzer für den Testfall

Im Rahmen dieser Arbeit erzeugt ein Selenium-Test automatisch neue Benutzerkonten in Keycloak und weist ihnen jeweils ein Passwort zu. Diese Daten werden pro Realm einmalig in einer JSONL-Datei gespeichert, wobei Benutzername und Passwort als Schlüssel-Wert-Paar abgelegt werden.

Damit sich die erzeugten Benutzer über die automatisierten Tests in Keycloak anmelden können, muss in der Konfiguration festgelegt werden, dass ein vollständiges Profil (Vorname, Nachname, E-Mail) erforderlich ist und vorhanden ist– andernfalls würde die Authentifizierung fehlschlagen.

Darüber hinaus verfügen Clients dann über Service Accounts. Die Service-Account-Rollen bestimmen dabei die verfügbaren Rechte. Ohne diese Rollen könnte der Client zwar eine Verbindung herstellen, hätte jedoch keinen Zugriff auf Benutzerverwaltung oder Logdaten. Die korrekte Konfiguration der Service-Account-Rollen ist daher zwingend notwendig, um automatisierte Vorgänge im Hintergrund zu ermöglichen.

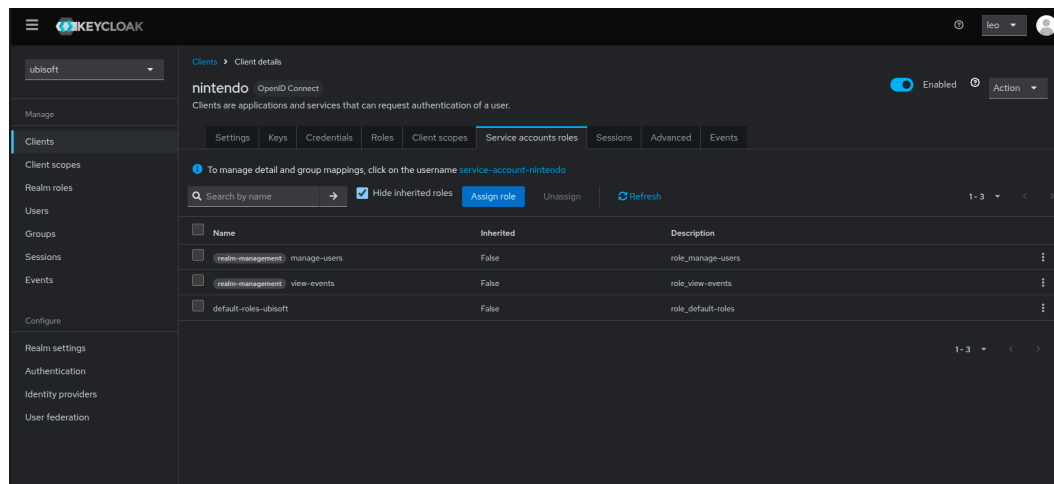


Abbildung 12: Benötigte Rollen für den Client Nintendo

Ab einer bestimmten Menge an Trainingsdaten werden die generierten Benutzer im Rahmen des Selenium-Tests dazu gebracht, sich regulär anzumelden. Die dabei entstehenden sogenannten „gewöhnlichen“ Logs umfassen unter anderem Anmeldevorgänge und sind wie bereits erwähnt, inspiriert von realen firmeninternen Protokollen. Diese bestehen typischerweise aus den Event-Typen LOGIN, CLIENT_LOGIN und CODE_TO_TOKEN.

Folgende Aktivitäten gelten laut interner Log-Analyse als „typisch“ nach einen Vergleich mit den firmeninternen Keycloak-Logs:

- Anmeldung über einen Client
- Token-Aktualisierung (Refresh Token)
- Abmeldung (Logout), auch nach Ablauf einer Session
- Fehlgeschlagene Login-Versuche (kommen häufiger vor)

Weitere Aktionen – wie das Anpassen des Benutzerprofils (z. B. E-Mail-Änderung, Passwort-Zurücksetzung), das Löschen eines Clients oder das Zuweisen von Rollen an andere Benutzer beziehungsweise Clients – wurden in der Log-Generierung bewusst manuell durchgeführt. Der Grund hierfür liegt in der vergleichsweise hohen Komplexität und der geringen Häufigkeit dieser Vorgänge. Da Keycloak primär als SSO-Dienst genutzt wird, wäre eine regelmäßige Durchführung solcher Aktionen im automatisierten Testkontext unrealistisch und würde die Aussagekraft der „normalen“ Logs verzerren.

Stattdessen werden solche seltenen Aktionen manuell während des Testbetriebs von einzelnen Benutzern durchgeführt. Um dabei in die „Rolle“ eines anderen Benutzers zu schlüpfen, wird der Event-Typ Impersonate verwendet. Die Verwendung von Impersonation während der automatisierten Testausführung wird vermieden, da sie eine potenzielle Anomalie darstellt. Ziel ist es, lediglich gelegentlich und gezielt ungewöhnliche Aktionen zu erzeugen – etwa

eine Passwort- oder E-Mail-Änderung, das Anlegen oder Löschen eines Benutzers oder das Zuweisen von Rollen. Dies reduziert den manuellen Aufwand bei der Erzeugung seltener Ereignisse, ohne die Gesamtdatenbasis zu verzerren.

Pro Durchlauf werden insgesamt 30 Sessions (Bei den ersten 10000 Loggs waren es noch 15, wurde aber erhöht, um schneller Logs zu erzeugen) durchgeführt, wobei jede Session durch den häufigen Einsatz von Refresh Tokens mindestens zehn Einträge im Log erzeugt. Die Sitzungsdauer wird dabei durch die Konfiguration des jeweiligen Realms bestimmt. Aus praktischer Erfahrung ergibt sich eine durchschnittliche Dauer von etwa 35 Minuten pro Session – realistisch im Rahmen von 10 Minuten bis zu einer Stunde. So können am Tag zwischen 2.500 und 5.000 Log-Einträge erzeugt werden.

Ereignisse, die typischerweise erst nach längerer Zeit auftreten, fehlen somit in der Testbasis. Eine spätere Erweiterung ist möglich, jedoch wurde aufgrund der Bearbeitungsfrist beschlossen, die Generierung auf 12 aufeinanderfolgende Tage zu konzentrieren, ausgeschlossen ist dabei das Wochenende. Die Sessions werden dabei nacheinander mit variablen Abstand ausgeführt.

9.3 Mögliche Angriffsszenarien basierend auf CVE und CWE

CVE stellt eine umfassende Datenbank bekannter Sicherheitslücken dar, die in verschiedenen Softwaresystemen dokumentiert sind. Für Keycloak existieren ebenfalls zahlreiche veröffentlichte Schwachstellen³².

CVE-Einträge liefern jedoch keine Lösungen, sondern dienen der Information über Art, Ursache und potenzielle Auswirkungen der jeweiligen Sicherheitslücke. Jeder Eintrag erhält eine eindeutige Identifikationsnummer, eine Kurzbeschreibung sowie (sofern verfügbar) Angaben zur betroffenen Softwareversion. Die Pflege und Dokumentation erfolgt weltweit durch Organisationen, Forschungseinrichtungen sowie unabhängige Sicherheitsexperten.

Im Gegensatz dazu beschreibt die CPunkte in DBSCAN, die keinem Cluster zugeordnet werden können, systematisch bekannte Schwachstellenmuster (Weaknesses), die potenziell zu Sicherheitslücken führen können. Während CVEs konkrete, bereits gefundene oder ausgenutzte Schwachstellen auflisten, klassifiziert CWE abstrakte Fehlerklassen, wie beispielsweise „unzureichende Rechteprüfung“ oder „unsichere Tokenverarbeitung“. Beide Kataloge sind essentiell für die systematische Entwicklung und Interpretation von Sicherheitstests.

Obwohl viele der bekannten Probleme bereits behoben wurden, zeigen sich Schwachstellen in Keycloak zunehmend wiederkehrend und entwickeln sich mit der Zeit weiter. In letzten Versionen bspw. können zwar frühere Schwachstellen behoben worden sein, in späteren jedoch neue Schwachstellen und Lücken auftreten.

Im Rahmen dieser Arbeit wurden mittels Selenium gezielt sicherheitsrelevante Angriffssze-

narien nach CVE und CWE und anderen Quellen (die schon im vorherigen Kapiteln bzgl. des ersten Entwurfes der Angriffsszenarien beschrieben wurden), gegen Keycloak simuliert und getestet. Es wird dabei folgendes Angriffsszenario zur Orientierung genommen:

CVE-2024-3656 - Privilege Escalation in Keycloak Bei CVE-2024-3656³³ handelt es sich um eine Sicherheitslücke in Keycloaks Admin REST API, bei der unzureichende Berechtigungsprüfungen es einem Nutzer mit niedrigen Rechten ermöglichen, administrative Endpunkte aufzurufen. Dadurch können sensible Daten wie Benutzerlisten eingesehen und administrative Aktionen durchgeführt werden, ohne dass entsprechende Rechte vorliegen. Die Ursache liegt in fehlerhaften oder fehlenden Zugriffskontrollen auf kritischen API-Endpunkten. Die Schwachstelle wurde in neueren Keycloak-Versionen behoben, darunter auch die Version 26.1.0, welche in dieser Arbeit verwendet wird. Andere Angriffsfälle konnten nicht umgesetzt werden, da manche Features die dafür verlangt werden in aktuellen oder früheren Versionen nicht mehr erhältlich sind. Auch wenn dieser Angriff fehlschlägt, sorgt er dafür, dass viele Logs mit anomalen Charakter generiert werden.

Da CVE-Datenbanken keine vollständigen Exploitbeschreibungen liefern, wurde im Rahmen der Simulation eine technische Interpretation der Schwachstellen vorgenommen. Zudem wurden noch weitere Angriffe ausgeführt, welche typisch sind für einen Angriff und auch schon in den bekannten Insiderangriffsfällen genannt wurde:

- **Account Sabotage-Test:** Löschen oder Verändern wichtiger Benutzerinformationen.
- **Dictionary Angriff**

9.4 Angriffsszenarien

Die im vorherigen Abschnitt beschriebenen Angriffsszenarien wurden im Rahmen dieser Arbeit mithilfe von Selenium-Tests implementiert und in den generierten Logs dokumentiert. Für jedes Szenario wurde ein konkreter Ablauf definiert, um die jeweilige Sicherheitslücke zu simulieren.

Hier werden die Angriffsfälle aufgelistet welche direkt in der Keycloak-Konsole ausgeführt wurden, also nicht als Selenium-Test- außer das erste Szenario des Testfalls *Privilege Escalation*, dieses wird auch als Selenium-Test ausgeführt.

9.4.1 Privilege Escalation

In diesem Szenario wird untersucht, ob ein technisch privilegierter, aber nicht vollständig administrativer Benutzer in Keycloak durch gezielte API-Aufrufe seine eigenen Berechtigungen ausweiten kann. Ziel ist es, das Verhalten von Keycloak bei einer möglichen Rechteausweitung über die REST-Schnittstelle zu evaluieren.

Ziel: Unautorisierte Erweiterung der Benutzerrechte durch Ausnutzung der REST API

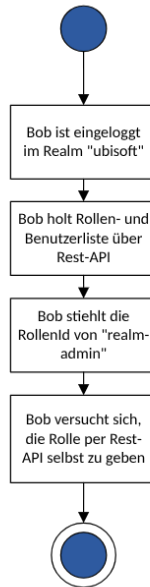


Abbildung 13: Aktivitätsdiagramm Privilege Escalation

Betroffener Realm: ubisoft

Angreiferrolle: Benutzer mit der Rolle `manage-users`

Betroffener Benutzer: bob (versucht sich selbst administrative Rechte zu geben)

Ablauf:

1. Der Benutzer bob meldet sich regulär im System an und verfügt über ein gültiges Access Token.
2. Mit diesem Token führt er einen GET-Request auf `/admin/realms/ubisoft/users` aus, um die Benutzerliste des Realms zu erhalten.
3. Die Abfrage ist erfolgreich – bob erhält Zugriff auf die vollständige Liste und identifiziert seine eigene Benutzer-ID.
4. Anschließend versucht er über einen POST-Request an `/admin/realms/ubisoft/users/{id}/role-mappings/realm`, sich selbst die Rolle `realm-admin` zuzuweisen.
5. Der Request schlägt fehl – die Serverantwort ist HTTP 403 Unauthorized.

Technische Umsetzung: Die Requests wurden mithilfe von curl direkt an die REST API gesendet. Dabei wurde ein korrekt signiertes Access Token verwendet, das bob durch einen Login mit seiner `manage-users`-Rolle erhalten hatte.

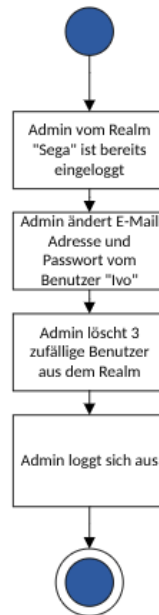


Abbildung 14: Aktivitätsdiagramm von Account-Sabotage

Erwartetes Verhalten:

- Die Benutzerliste kann durch Rollen wie `manage-users` teilweise eingesehen werden, was ein potenzielles Datenschutzrisiko darstellt.
- Die unautorisierte Zuweisung privilegierter Rollen wird vom Server korrekt abgelehnt.
- Der Zugriff auf administrative Rollen über die REST API ist in Keycloak Version 26.1 restriktiver als in früheren Versionen, was diese Art von Privilege Escalation wirksam verhindert.

9.4.2 Angriffsszenario: Account-Sabotage durch privilegierten Benutzer

In diesem Szenario wird untersucht, inwieweit ein privilegierter Benutzer (z. B. ein Administrator) absichtlich oder versehentlich Schaden an Benutzerkonten anrichten kann. Ziel ist es, das Verhalten von Keycloak bei kritischen administrativen Operationen zu beobachten und zu analysieren, welche Aktionen protokolliert werden.

Ziel: Simulierte Account-Sabotage durch einen legitimen Admin im Realm

Betroffener Realm: Sega

Angreiferrolle: Benutzer mit Administratorrechten (z. B. `realm-admin` oder `manage-users`)

Betroffene Benutzer: Admin in Sega (Profiländerung) sowie drei zufällige Benutzer (Löschung)

Ablauf:

1. Ein Angreifer kennt die Credentials des Admins im Realm Sega und ist bereits mit dessen Credentials eingeloggt.
2. Der Angreifer ändert das Passwort des Admins in Sega.
3. Anschließend löscht der Admin drei beliebige Benutzerkonten im selben Realm.
4. Der Angreifer meldet sich ab.

Technische Umsetzung: Die Ausführung erfolgt entweder manuell über die Keycloak Admin Console oder automatisiert mit Hilfe von Selenium.

Erwartetes Verhalten:

- Die Profiländerungen und das Zurücksetzen des Passworts sollten als `UPDATE_PASSWORD` im **User-Event-Log** erscheinen.
- Die Löschung der Benutzerkonten wird im **Admin-Event-Log** als `DELETE` Operation registriert.
- Alle Aktionen sind über den Zeitstempel, Benutzer und IP-Adresse nachvollziehbar.
- *LOGOUT* wird ebenfalls in den Logs angezeigt

Das einzige Szenario, welches durch ein Selenium-Test erzeugt wurde ist folgendes:

9.4.3 Angriffsszenario: Dictionary-Angriff über die Web-Oberfläche

In diesem Szenario wird ein klassischer Dictionary-Angriff simuliert, bei dem ein Benutzername fest vorgegeben ist und systematisch unterschiedliche Passwörter ausprobiert werden. Ziel ist es zu prüfen, ob ein erfolgreiches Login durch mehrfaches Raten von Passwörtern möglich ist und ob Keycloak entsprechende Fehlversuche protokolliert.

Ziel: Test der Account-Sicherheit gegenüber einfachen Dictionary-Angriffen über das Login-Formular

Betroffener Benutzer: Admin

Realm: Dynamisch gewählt (aus mehreren Realms auswählbar)

Passwortliste: Eine Liste gängiger schwacher Passwörter (z. B. admin123, 123456, password, qwerty, welcome, etc.)

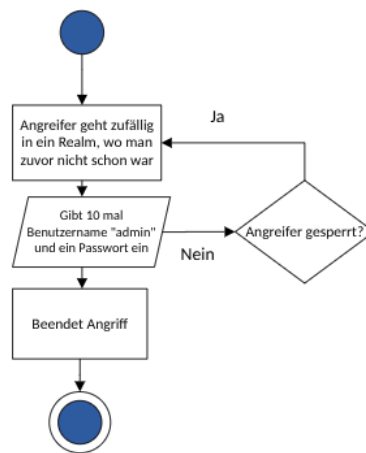


Abbildung 15: Aktivitätsdiagramm von Dictionary

Ablauf:

1. Es wird eine Verbindung zur Keycloak-Login-Seite des jeweiligen Realms hergestellt.
2. Der Benutzername wird in das Formular eingetragen (Admin).
3. Es wird nacheinander jedes Passwort aus der vorgegebenen Liste eingegeben und die Login-Schaltfläche betätigt. Wenn Keycloak angibt, dass zu viele fehlgeschlagene Anmeldeversuche versucht wurden, wechselt der Angreifer die Login-Startseite des jeweiligen Realms (Pro Realm gibt es unterschiedliche Login-Startseiten, bei der in der URL ein anderes Realm steht)
4. Das Skript prüft, ob eine Fehlermeldung erscheint oder ob der Login erfolgreich war.
5. nach 10 gescheiterten versuchen wird das Szenario, bzw. Skript gestoppt.

Technische Umsetzung: Die Automatisierung erfolgt mittels Selenium WebDriver in Kombination mit einer Python-Testfunktion. Die Login-Seite wird jeweils mit einem neuen Passwort geladen und automatisiert ausgefüllt.

Erwartetes Verhalten:

- Keycloak sollte nach mehreren Fehlversuchen Gegenmaßnahmen ergreifen (z. B. Account-Sperrung, Rate Limiting).
- Im Event-Log sollten die fehlgeschlagenen Login-Versuche als LOGIN_ERROR mit dem jeweiligen Benutzer verzeichnet sein.

Dictionary und Privilege Escalation sind Angriffe, die fehlschlagen. Nur Account- Sabotage ist erfolgreich. Dies liegt an der Umsetzung (Dictionary Angriff durch Selenium-Tests) und daran, dass der erwähnte CVE Fall für neuere Keycloak Versionen nicht mehr funktioniert durch neue Sicherheitsfunktionen- dennoch stellen sie erkennbare, anomale Logs dar.

9.5 Änderung der Timestamps durch Unix Timestamp

Die Angriffsszenarien wurden am Ende der 12 Tage, also am 12. Tag erzeugt. So durch wirkt es, als ob alle Angriffe am letzten Tag erzeugt worden sind. Dies verringert die zufällige Verteilung der Angriffe. Daher müssen die Timestamps manuell geändert werden. Es muss gewährleistet sein, dass diese zufällig vom 22.07, ca. 14 Uhr bis zum 08.08, 17 Uhr erzeugt wurden- und demnach verändert werden. Es wurden folgende Zeitpunkte pro Angriffsszenario modelliert:

- *Dictionary Angriff*: Umstellung auf den 06.08 um 13 Uhr Abends
- *Privilege Escalation*: Umstellung auf den 07.08 um 16 Uhr
- *Account Sabotage*: Umstellung auf den 08.08 um 11 Uhr Morgens

Diese Zeitpunkte werden auf der Webseite Unix Timestamp³⁴ verändert.

The Current Epoch Unix Timestamp

Enter a Timestamp

1754567250

Supports Unix timestamps in seconds, milliseconds, microseconds and nanoseconds.

Convert →

1754567596

SECONDS SINCE JAN 01 1970. (UTC)

1:53:18 PM

Copy

Related Tools

- Binary Converter

Enter a Date & Time

Year	Month	Day	Hour (24 hour)	Minutes	Seconds
2025	08	07	11	47	30

Convert →

The current epoch translates to

Abbildung 16: Aktuelle Uhrzeit nach dem Unix-System [Online]. Verfügbar unter: <https://www.unixtimestamp.com/>, abgerufen am 18. August 2025.

Die erste Attacke dauert ca. 1-2 Minuten, die zweite ca. 5-10 Minuten und die letzte im ca. 20 Minuten. Dies Zeit wurde so bestimmt, dass sie sich für den jeweiligen Angriff eignen. Die Angriffe werden mit Absicht an Zeitpunkte durchgeführt, welche zwischen den normalen Logs mit Abstand sind, damit das LSTM-AE auch seinen Einsatz daran zeigen kann.

10 Durchführung des Trainings

Die Logs werden separat je nach Training, Test und Validierung in Dateien gepackt und dann zu Beginn ausgelesen. Alle drei Dateien, also Evaluierungs-, Test- und Trainingsdateien werden dabei einzeln bereinigt und zusammengefügt. Darauf hin werden alle Daten mit der standardisiert und skaliert, wie im Abschnitt zur Implementierung bereits erklärt. Da der LSTM-AE nur mit sequentiellen Daten arbeiten kann, wurden diese Dateien in Sequenzen aufgeteilt.

Über einen Zeitraum von 12 Tagen (ca. 2.5 Wochen, nur Werktage) wurden ausschließlich „normale“ Logs generiert. An den darauffolgenden Tagen fanden gezielte Angriffsszenarien statt. Mithilfe eines Skripts wurden anschließend sämtliche Event-Logs aus allen Realms zusammengeführt.

Das Skript funktioniert in etwas wie folgt:

Die kombinierten Logs werden zunächst im Verhältnis 70 % zu Trainingsdaten und jeweils 15 % zu Test- und Validierungsdaten aufgeteilt. Die Anomalien werden zufällig auf die Test- und Validierungsdaten verteilt, dabei jedoch in chronologischer Reihenfolge einsortiert. Da die Anomalien erst nachträglich erzeugt wurden, mussten die Timestamps wie bereits diskutiert, entsprechend angepasst werden.

Eine rein zufällige Vergabe der Timestamps war dabei nicht möglich, da bestimmte Angriffsszenarien einen klaren zeitlichen Ablauf erfordern. Die Timestamps wurden daher gezielt so angepasst, dass sie diesen inhaltlichen Zusammenhang korrekt abbilden, ohne bspw. am Ende des Datensatzes zu kumulieren. Aufgrund der überschaubaren Anzahl an Anomalien (ca. 31 Einträge) konnte dieser Schritt manuell umgesetzt werden. Die Logs wurden dennoch in chronologischer Reihenfolge angeordnet, sodass der zeitliche Ablauf der Ereignisse konsistent abgebildet wird.

Die Anomalien sind zuvor in eine separate Datei ausgelagert. Bevor die anomalen Logs mit den normalen vermischt werden, fügt das Skript Labelinformationen hinzu: In der Datei *attacks_logs.py* wurden die anomalen Ereignisse aus Keycloak extrahiert und mit dem Label "1" versehen, während in der Datei *normal_test_logs.py* die regulären Log-Einträge mit dem Label "0" markiert wurden.

Abschließend wird die finale Datei in zwei Hälften aufgeteilt; in eine Validierung- und einer Testdatei.

Das LSTM-AE wird ausschließlich mit normalen, nicht gelabelten Daten trainiert. Dies liegt an der Funktionsweise des Autoencoders, der lediglich versucht, die Eingangsdaten zu rekonstruieren. Nach ausreichendem Training lernt er, nur normale Daten zuverlässig zu rekonstruieren. Enthalten die Testdaten Anomalien, werden diese schlechter rekonstruiert, was zum sogenannten Rekonstruktionsfehler führt.

Jedes Hybridmodell wurde pro Sequenzlänge fünfmal trainiert. Dies ist notwendig, um die durch die zufällige Initialisierung der Parameter in neuronalen Netzen bedingte stochastische Variabilität angemessen abzubilden.

10.1 Die Anzahl der Log-Einträge

Wie viele Trainingsdaten an sich auch nötig sind, ist ein weitgehend offenes Problem. Es gibt Studien, die sich systematisch mit der Frage beschäftigt haben, welche Datenmenge „angemessen“ ist, um ein neuronales Netzwerk zuverlässig zu trainieren.

Nach der Studie von Golestaneh et al. [71] wurde eine theoretische Fehlerabschätzung für bestimmte Netzwerke abgeleitet. Demnach skaliert der Fehler ε eines solchen Netzwerks nicht mit der klassischen „parametrischen“ Rate von $\varepsilon \sim 1/n$, sondern mit einer langsameren Rate:

$$\varepsilon(n) \sim \frac{1}{\sqrt{n}}$$

Dabei ist n die Anzahl der Trainingsbeispiele. Diese Formel bedeutet, dass eine Verdopplung der Datenmenge den Fehler nicht halbiert, sondern nur um den Faktor $1/\sqrt{2}$ verringert. Dennoch erlaubt diese Abschätzung eine grobe Einschätzung der notwendigen Datenmenge.

Wendet man diese Formel auf eine Trainingsmenge von $n = 10.000$ Log-Zeilen an, ergibt sich ein theoretischer Fehler von etwa

$$\varepsilon(10.000) \approx \frac{1}{\sqrt{10.000}} = 0,01$$

Dieser Wert gilt zwar nicht als absolute Fehlermarke (da der tatsächliche Fehler von Modellarchitektur, Datenqualität und Regularisierung abhängt), er ist jedoch in vielen praktischen Fällen bereits hinreichend gering, um eine sinnvolle Modellierung zu ermöglichen. Somit kann ein LSTM-Autoencoder bei geeigneter Modellwahl und sauberem Datenvorverarbeitung bereits mit etwa 10.000 normalisierten Log-Zeilen ein brauchbares Modell für Anomalieerkennung liefern.

Auch eine weitere Studie von Götz et al. [72] beschäftigt sich mit der Relation zwischen der Komplexität eines neuronalen Netzwerks (gemessen an der Anzahl der trainierbaren Parameter N_p) und der Anzahl der notwendigen Trainingsbeispiele N_{tr} . Dabei wurde gezeigt, dass kein starres Verhältnis wie $N_{tr} \geq N_p$ erforderlich ist, um gute Ergebnisse zu erzielen. Stattdessen weisen die Autoren nach, dass auch mit deutlich weniger Trainingsbeispielen eine stabile Modellleistung möglich ist.

In ihren Experimenten mit Convolutional Neural Networks (CNNs) konnten selbst bei einem Verhältnis $N_{tr}/N_p < 1$ noch vergleichbare Klassifikationsleistungen erzielt werden, solange die Reduktion der Netzwerkkomplexität kontrolliert erfolgt. Übertragen auf den vorliegen-

den Fall bedeutet dies: Bei einem moderat komplexen LSTM-Autoencoder mit z. B. einigen zehntausend Parametern können bereits 10.000 Trainingsbeispiele (z. B. Log-Zeilen) ausreichen, um ein funktionierendes Modell zu erhalten – sofern geeignete Regularisierung und Modellarchitektur verwendet werden.

Die Studie stützt damit ebenfalls die Annahme, dass in vielen praktischen Szenarien bereits Datensätze von 10.000 Einträgen als ausreichend gelten können, um eine robuste Anomalieerkennung durchzuführen- wobei man natürlich beachten muss, dass in jeder Studie der Fokus auf einem anderen Netzwerk lag. Dennoch will man diese Studien als Hinweis dafür wahrnehmen, möglichst viele Daten zu generieren, auch wenn es im Rahmen dieser Arbeit die Anzahl der wahrscheinlich nur im fünf-stelligen Bereich liegen wird.

Auf Grundlage dieser Erkenntnisse wird eine Testdatei mit ca. 5.850 Logs erstellt.

In dieser Testdatei befinden sich in den Testdaten wie zuvor angedeutet ca. 31 Logs, welche durch alle der Angriffsszenarien entstanden sind (also ca. 10 Logs pro Angriff).

10.2 Testfälle

Aufgrund der ungleichmäßigen Verhältnis zwischen Anomalien und normalen Logs und der geringen Anzahl an Logs wurde beschlossen Testfälle mit unterschiedlichen Architekturen durchzugehen. Es werden folgende Sequenzlängen validiert:

- Sequenzlänge 25
- Sequenzlänge 45

Da es sich in dieser Arbeit um eine vergleichsweise kleine Testdatenmenge von rund 39.000 Log-Einträgen handelt und das LSTM-Netzwerk dadurch tendenziell unterfordert wäre, musste die Sequenzlänge gering gehalten werden. Zudem zeigt auch eine Studie von Toor et Al., dass eine kleine Sequenzlänge von 24–48 eine sehr effektive Wahl ist für Bi-LSTM-basierte Netze [57].

Bei sehr großen Datensätzen mit beispielsweise einer Million Log-Einträgen ist eine Sequenzlänge von 100 nachvollziehbar: Hier liegen viele Ereignisse dicht beieinander, und Anomalien können sich über mehrere Log-Einträge hinweg erstrecken. In einem kleineren Datensatz hingegen ist es notwendig, die Sequenzlänge zu verkürzen, um sowohl genügend Trainingsbeispiele zu erzeugen als auch relevante Muster erfassen zu können.

Die Wahl der unterschiedlichen Sequenzlängen dient dazu, die Sensitivität des Modells auf verschiedene zeitliche Zusammenhänge zu testen. Kürzere Sequenzen erfassen lokale Muster, während längere Sequenzen komplexere zeitliche Abhängigkeiten abbilden können. Dadurch kann das Modell besser an die unterschiedlichen Charakteristika der Log-Daten angepasst werden.

Zur Übersicht wurden Tabellen erstellt, welche das Verhältnis zu den abhängigen und unabhängigen Variablen zeigen soll:

Testvariable- Unabhängige Variable	Konstanten
<ul style="list-style-type: none"> - Sequenzlänge 	<ul style="list-style-type: none"> - Tiefe des Netzwerkes - Breite des Netzwerkes - Batch-Size = 32 - Lernrate = 0.001 - Art, wie der Schwellenwert berechnet wird - Art, wie der Rekonstruktionsfehler berechnet wird - Skalierung (StandardScaler) - Numerische Vorverarbeitung (Kein One Hot Encoding) - Epochenanzahl = 50 - Optimierer-Typ (Adam) - Verlustfunktion (MSE) - Patience = 2

Abbildung 17: Überblick der Konstanten Werte und der unabhängigen Variable

Alle genannten Konstanten wie die Breite, Tiefe und Batch-Size bleiben dabei konstant. Weitere Faktoren wie die Berechnung des Rekonstruktionsfehlers, der Schwellenwert, die Skalierung (StandardScaler), die Anzahl von 50 der Trainings-Epochen, der Optimierer (Adam) und die Verlustfunktion (MSE) werden konstant gehalten, um einen fairen Vergleich der Modelle zu gewährleisten. Die gewählten Konstanten können zwar die erzielbaren Höchstwerte im Vergleich zu anderen Studien begrenzen, gewährleisten jedoch, dass die Analyse vergleichbar bleibt.

Zusammenfassend ergibt sich, dass die Sequenzlänge als unabhängige Variable und die Metriken F1-Score, Balanced Accuracy sowie MCC als abhängige Variablen betrachtet werden.

11 Auswertung

11.1 Berechnung der Mittelwerte

Um herauszufinden, wie hoch die Werte vom F1-Score, MCC und Balanced Accuracy haben sind, werden die Mittelwerte aller Runs je Sequenzlänge ausgerechnet.

Tabelle 2: Vergleich der Modelle bei Sequenzlänge 25

Modell	Sequenzlänge	F1-Score	MCC	Balanced Accuracy
IsolationForest	25	0.7541	0.7587	0.8188
One-Class SVM	25	0.7430	0.7411	0.8271
DBSCAN	25	0.1928	0.3170	0.5538

Tabelle 3: Übersicht der Ergebnisse für Sequenzlänge 45

Modell	Sequenzlänge	F1-Score	MCC	Balanced Accuracy
IsolationForest	45	0.6172	0.5904	0.7937
One-Class SVM	45	0.6367	0.6185	0.7895
DBSCAN	45	0.1237	0.2453	0.5332

Es ist erkennbar, dass die Ergebnisse stark von der Sequenzlänge abhängen. Sequenzen der Länge 25 erzielen generell bessere Ergebnisse als Sequenzen der Länge 45, vermutlich weil kürzere Sequenzen für diese Modelle besser geeignet sind. In Zukunft sollte auch versucht werden, deutlich kürzere Sequenzen zu verwenden, zum Beispiel mit einer Länge von 1 bis 10. Dies könnte neue Erkenntnisse insbesondere bei kleinen Datensätzen liefern.

DBSCAN erzielt unter allen Metriken und Sequenzlängen die niedrigsten Werte. Dies könnte auf die ungleiche Verteilung der Daten zurückzuführen sein. DBSCAN ist grundsätzlich ein Clustering-Verfahren – eine stark ungleiche Datenverteilung kann daher dazu führen, dass Anomalien falsch geclustert werden.

OCSVM erreicht bei der Sequenzlänge 45 eine höhere MCC sowie einen höheren F1-Score. Isolation Forest erzielt hingegen bei der Sequenzlänge 25 bessere Werte in beiden Metriken sowie eine höhere Balanced Accuracy. OCSVM zeigt dafür bei der Sequenzlänge 25 eine höhere Balanced Accuracy als bei längeren Sequenzen.

Die Mittelwerte zeigen klar, dass Sequenzlänge 25 für IF optimal ist, während OCSVM bei Sequenzlänge 45 besser abschneidet. Damit lässt sich die Hypothese aber weder verifizieren, noch falsifizieren.

Wenn man alle Metriken zusammenrechnet, unabhängig von der Sequenzlänge erhält man ein anderes Bild:

Tabelle 4: Übersicht der Ergebnisse

Modell	Sequenzlänge	F1-Score	MCC	Balanced Accuracy
IsolationForest	25 und 45	0.6857	0.6746	0.8063
One-Class SVM	25 und 45	0.6899	0.6798	0.8083
DBSCAN	25 und 45	0.1583	0.2812	0,5435

Bei der Summierung der Werte ergibt sich ein neues Bild: Unter allen Sequenzlängen erzielt OCSVM in allen drei Metriken die höchsten Ergebnisse, auch wenn sie nur knapp etwas höher sind als IF. Die Hypothese kann man hier schon als widerlegt ansehen, jedoch wäre dies vorurteilend. Unter anderem kann man hier bei der Berechnung der Mittelwerte zwar nur aussagen, wie "hoch" die Ergebnisse sind, aber nicht wie stabil sie sind. Es werden deshalb weitere Evaluierungsmetriken benötigt, um ein genaues Bild davon zu bekommen, wie die Werte verteilt sind und wie sie zustande gekommen sind.

11.1.1 Berechnung der Standardabweichung und Varianz

Tabelle 5: Varianz der Modelle bei Sequenzlänge 25

Modell	F1-Score	MCC	BA
IF	0.000124	0.000220	0.000047
OCSVM	0.000568	0.000944	0.000087
DBSCAN	0.003704	0.003892	0.000319

Tabelle 6: Varianz der Modelle bei Sequenzlänge 45

Modell	F1-Score	MCC	BA
IF	0.000318	0.000383	0.000073
OCSVM	0.000318	0.000382	0.000073
DBSCAN	0.001706	0.002560	0.000130

Tabelle 7: Standardabweichung der Modelle bei Sequenzlänge 25

Modell	F1-Score	MCC	BA
IF	0.01113	0.01484	0.00683
OCSVM	0.02382	0.03072	0.00933
DBSCAN	0.06086	0.06239	0.01786

Tabelle 8: Standardabweichung der Modelle bei Sequenzlänge 45

Modell	F1-Score	MCC	BA
IF	0.01783	0.01955	0.00857
OCSVM	0.01783	0.01955	0.00857
DBSCAN	0.04131	0.05060	0.01138

Zur Beurteilung der Stabilität der Modelle wurden die Varianz und Standardabweichung der F1-Score, Matthews Correlation Coefficient (MCC) und Balanced Accuracy (BA) über fünf Testläufe berechnet. Die Berechnungen erfolgten für Sequenzlängen von 25 und 45.

Für Sequenzlänge 25 zeigt IF sehr geringe Varianzen und Standardabweichungen (F1-Score: Varianz 0.000124, Std 0.0111; MCC: Varianz 0.000220, Std 0.0148; BA: Varianz 0.000047, Std 0.0068). Auch bei längeren Sequenzen (45) bleibt die Streuung niedrig (F1-Score: Varianz 0.000318, Std 0.0178; MCC: Varianz 0.000382, Std 0.0196; BA: Varianz 0.000073, Std 0.0086). Dies zeigt, dass IF über die Testläufe hinweg sehr stabile Ergebnisse liefert.

OCSVM zeigt bei Sequenzlänge 25 eine leicht höhere Streuung (F1-Score: Varianz 0.000568, Std 0.0238; MCC: Varianz 0.000944, Std 0.0307; BA: Varianz 0.000087, Std 0.0093), stabilisiert sich jedoch bei Sequenzlänge 45 (F1-Score: Varianz 0.000318, Std 0.0178; MCC: Varianz 0.000382, Std 0.0196; BA: Varianz 0.000073, Std 0.0086). Damit zeigt OCSVM eine gute Konsistenz, insbesondere bei längeren Sequenzen.

DBSCAN weist im Vergleich zu IF und OCSVM deutlich höhere Varianz und Standardabweichung auf (z. B. F1-Score: Varianz 0.003704, Std 0.0609 bei Sequenzlänge 25; F1-Score: Varianz 0.001706, Std 0.0413 bei Sequenzlänge 45). Diese Werte deuten auf eine hohe Sensitivität gegenüber den Parametern und eine geringere Stabilität über die Testläufe hin.

Kurz gesagt: IF ist über alle Testläufe am zuverlässigsten, OCSVM variiert ein wenig bei kurzen Sequenzen, DBSCAN ist am wenigsten zuverlässig. Hier zeigt sich, dass IF die stabilsten Werte also erzielte.

11.1.2 Friedman-Test

Der Friedman-Test [73] ist ein nicht-parametrischer Test, der prüft, ob es systematische Unterschiede zwischen mehreren verbundenen Gruppen gibt, z.B., ob verschiedene Modelle über mehrere Datensätze oder Sequenzlängen hinweg unterschiedliche Leistungen zeigen.

Dabei liefert der Test zwei zentrale Kenngrößen: die Teststatistik χ^2 und den p-Wert.

χ^2 ist die Teststatistik, die aus den Rängen der Beobachtungen berechnet wird und angibt, wie stark sich die Gruppen systematisch unterscheiden.

Der p-Wert gibt an, wie wahrscheinlich es ist, dass die beobachteten Unterschiede zwischen den Gruppen zufällig entstanden sind. Ein kleiner p-Wert (z.B. größer als 0,05) deutet darauf

hin, dass die Unterschiede statistisch signifikant sind.

Für den Vergleich der Leistungsmetriken der drei Hybridmodelle über verschiedene Sequenzlängen wurde der Friedman-Test verwendet. In diesem Experiment entsprechen die verschiedenen Sequenzlängen den verbundenen Messungen, da jede Sequenzlänge für alle Modelle vorliegt.

Ein weiterer Vorteil des Friedman-Tests ist, dass er keine Annahmen über die Normalverteilung der Daten macht. Parametrische Tests wie die ANOVA³⁵ setzen voraus, dass die Messwerte innerhalb der Gruppen normalverteilt sind. Da dies bei den betrachteten Metriken (F1-Score, MCC, Balanced Accuracy) nicht garantiert werden kann, stellt der Friedman-Test eine robuste Alternative dar.

Der Test basiert auf Rangplätzen innerhalb der Sequenzlängen und ist dadurch weniger anfällig für Ausreißer. Gleichzeitig liefert er einen p-Wert, der es erlaubt zu prüfen, ob mindestens ein Modell systematisch von den anderen abweicht. Somit eignet sich der Friedman-Test ideal, um die signifikanten Unterschiede in den Leistungsmetriken zwischen den Modellen nachzuweisen.

Anbei sind die Ergebnisse des Tests:

Tabelle 9: Friedman-Test Ergebnisse (seq. 25)

Metrik	Chi ²	p-Wert	Signifikanz	Seq.
F1-Score	10.000	0.0067	Signifikante Unterschiede zwischen den Modellen	25
MCC	8.400	0.0150	Signifikante Unterschiede zwischen den Modellen	25
Balanced Accuracy	10.000	0.0067	Signifikante Unterschiede zwischen den Modellen	25

Tabelle 10: Friedman-Test Ergebnisse (seq. 45)

Metrik	Chi ²	p-Wert	Signifikanz	Seq.
F1-Score	10.000	0.0067	Signifikante Unterschiede zwischen den Modellen	45
MCC	10.000	0.0067	Signifikante Unterschiede zwischen den Modellen	45
Balanced Accuracy	10.000	0.0067	Signifikante Unterschiede zwischen den Modellen	45

Da die p-Werte alle unter 0.05 liegen, ist es sehr unwahrscheinlich, dass die beobachteten Unterschiede zufällig entstanden sind. Die Chi²-Werte sind größer als 0, was zusätzlich auf Abweichungen von der Nullhypothese hinweist. Somit wird die Nullhypothese abgelehnt.

Wichtig ist zu beachten, dass der Friedman-Test nur eine abhängige Variable untersucht, weshalb für jede Metrik ein separater Test durchgeführt wurde. Er stellt also nicht den gesamten

Zusammenhang zwischen den einzelnen Metriken dar, sondern die signifikanten Unterschiede einer Metrik je Modell.

Dies ist gerechtfertigt, da die Stichprobengröße pro Modell klein ist und keine Normalverteilung der Metriken vorausgesetzt wird. Eine multivariate Analyse über alle Metriken hinweg könnte theoretisch Korrelationsstrukturen berücksichtigen, ist jedoch aufgrund der geringen Stichprobengröße hier nicht sinnvoll. Die Stichprobe wurde bewusst klein gehalten, da bereits erste Tests zeigten, dass bei sehr hohen Sequenzlängen die Modellgüte in allen Metriken deutlich abnimmt. Ziel war es daher, den Einfluss unterschiedlicher Fenstergrößen gezielt zu analysieren, ohne den Rahmen der Arbeit zu sprengen. Mit den beiden gewählten Sequenzlängen (25 und 45) lässt sich die Diskrepanz zwischen kürzeren und längeren Kontexten ausreichend abbilden und validieren. Ein Vergleich mit zwei Sequenzlängen ist für die Fragestellung dieser Arbeit daher angemessen. In zukünftigen Arbeiten ist ein Vergleich mit mehreren Sequenzlängen wie erwähnt von Bedeutung.

Um zudem noch zu zeigen welche Modelle genau nun sich gering unterscheiden ist ein sogenannter Post-Hoc-Test notwendig.

11.1.3 Nemenyi-Test

Der Nemenyi-Test ist ein nichtparametrischer Post-hoc-Test³⁶, der nach einem Friedman-Test angewendet wird, wenn signifikante Unterschiede zwischen mehreren verbundenen Gruppen festgestellt wurden. Er dient dazu, paarweise Vergleiche zwischen den Gruppen durchzuführen und zu prüfen, welche Paare sich statistisch signifikant unterscheiden.

Im Gegensatz zu parametrischen Tests setzt der Nemenyi-Test keine Normalverteilung der Daten voraus, sondern arbeitet auf den Rängen der Beobachtungen. Die Ergebnisse werden in einer Matrix von p-Werten dargestellt, die für jedes Gruppenpaar angibt, ob der Unterschied signifikant ist.

In diesem Fall wurden drei Hybridmodelle hinsichtlich verschiedener Leistungsmetriken (F1-Score, MCC, Balanced Accuracy) bewertet. Da die Modelle auf denselben Datensätzen getestet wurden, handelt es sich um verbundene (abhängige) Stichproben, und die Daten sind nicht notwendigerweise normalverteilt. Nach dem Friedman-Test, der einen globalen Unterschied zwischen den Modellen zeigte, eignet sich der Nemenyi-Test ideal, um konkret zu bestimmen, welche Modellpaare sich signifikant unterscheiden.

Tabelle 11: Nemenyi Post-hoc Test: F1-Score, Sequenzlänge 25

	IF	OCSVM	DBSCAN
IF	1.0000	0.2538	0.0045
OCSVM	0.2538	1.0000	0.2538
DBSCAN	0.0045	0.2538	1.0000

Tabelle 12: Nemenyi Post-hoc Test: MCC, Sequenzlänge 25

	IF	OCSVM	DBSCAN
IF	1.0000	0.6094	0.0123
OCSVM	0.6094	1.0000	0.1394
DBSCAN	0.0123	0.1394	1.0000

Tabelle 13: Nemenyi Post-hoc Test: Balanced Accuracy, Sequenzlänge 25

	IF	OCSVM	DBSCAN
IF	1.0000	0.2538	0.2538
OCSVM	0.2538	1.0000	0.0045
DBSCAN	0.2538	0.0045	1.0000

Tabelle 14: Nemenyi Post-hoc Test: F1-Score, Sequenzlänge 45

	IF	OCSVM	DBSCAN
IF	1.0000	1.0000	0.0466
OCSVM	1.0000	1.0000	0.0466
DBSCAN	0.0466	0.0466	1.0000

Tabelle 15: Nemenyi Post-hoc Test: MCC, Sequenzlänge 45

	IF	OCSVM	DBSCAN
IF	1.0000	1.0000	0.0466
OCSVM	1.0000	1.0000	0.0466
DBSCAN	0.0466	0.0466	1.0000

Tabelle 16: Nemenyi Post-hoc Test: Balanced Accuracy, Sequenzlänge 45

	IF	OCSVM	DBSCAN
IF	1.0000	1.0000	0.0466
OCSVM	1.0000	1.0000	0.0466
DBSCAN	0.0466	0.0466	1.0000

Die Nemenyi-Post-hoc-Tests wurden durchgeführt, um nach den signifikanten Ergebnissen des Friedman-Tests die paarweisen Unterschiede zwischen den drei Modellen (IF, OCSVM, DBSCAN) für jede Metrik und Sequenzlänge zu identifizieren.

Für Sequenzlänge 25 zeigt der F1-Score, dass IF und OCSVM keinen signifikanten Unterschied aufweisen ($p = 0.2538$) und somit ähnlich performen, während IF im Vergleich zu

DBSCAN signifikant besser ist ($p = 0.0045$) und OCSVM vs DBSCAN keinen signifikanten Unterschied zeigt ($p = 0.2538$).

Beim MCC unterscheiden sich IF und OCSVM nicht signifikant ($p = 0.6094$), IF ist jedoch signifikant besser als DBSCAN ($p = 0.0123$), während OCSVM vs DBSCAN keinen signifikanten Unterschied zeigt ($p = 0.1394$). Bei der Balanced Accuracy zeigen IF vs OCSVM ($p = 0.2538$) und IF vs DBSCAN ($p = 0.2538$) keine signifikanten Unterschiede, während OCSVM vs DBSCAN signifikant ist ($p = 0.0045$), wobei OCSVM besser abschneidet und IF im Mittelfeld liegt.

Für Sequenzlänge 45 zeigt der F1-Score, dass IF und OCSVM keinen signifikanten Unterschied aufweisen ($p = 1.000$), während sowohl IF vs DBSCAN als auch OCSVM vs DBSCAN signifikant sind ($p = 0.0466$), wodurch DBSCAN deutlich schlechter performt und IF und OCSVM vergleichbar bleiben.

Beim MCC zeigt sich dasselbe Muster: DBSCAN unterscheidet sich signifikant von IF und OCSVM ($p = 0.0466$), IF und OCSVM unterscheiden sich nicht ($p = 1.000$), DBSCAN ist also auch im MCC schwächer. Bei der Balanced Accuracy gilt dasselbe: DBSCAN zeigt signifikant niedrigere Werte als IF und OCSVM ($p = 0.0466$), IF und OCSVM unterscheiden sich nicht ($p = 1.000$).

Insgesamt zeigt die Post-hoc-Analyse, dass insbesondere DBSCAN im Vergleich zu IF und OCSVM deutlich schlechter abschneidet, während IF und OCSVM weitgehend vergleichbar sind, was die globalen Unterschiede bestätigt, die bereits im Friedman-Test festgestellt wurden.

Somit wird die Nullhypothese, dass alle Modelle gleich gut sind, verworfen, und die Unterschiede zwischen den Modellen sind statistisch signifikant und nicht zufällig entstanden.

12 Diskussion

Die Ergebnisse zeigen deutliche Unterschiede in der Klassifikationsleistung zwischen den getesteten Verfahren und den Sequenzlängen. Die Tests für die Sequenzlängen 25 und 45 verdeutlichen, dass insbesondere Isolation Forest (IF) und One-Class SVM (OCSVM) insgesamt gute Ergebnisse liefern, während DBSCAN deutlich schlechter abschneidet (s. Tabellen und Post-hoc-Analyse).

IF und OCSVM erreichen konstant hohe Werte in F1-Score, MCC und Balanced Accuracy. Das bedeutet, dass die Modelle die meisten Anomalien korrekt erkennen, gleichzeitig aber nur wenige normale Logs fälschlicherweise als anomal klassifizieren. DBSCAN weist dagegen deutlich niedrigere Werte auf und ist instabiler über die Testläufe.

Für OCSVM zeigt sich, dass die Leistung bei längeren Sequenzen (45) leicht besser in F1

und MCC ist, während IF bei kürzeren Sequenzen (25) besonders stabile Ergebnisse liefert. DBSCAN bleibt unabhängig von der Sequenzlänge deutlich schlechter, was mit seiner Sensitivität gegenüber Parameterwahl und ungleich verteilten Daten zusammenhängt.

Weitere Einflussfaktoren auf die Ergebnisse könnten darin liegen, dass die Tiefe und Breite der LSTM-Autoencoder-Komponente möglicherweise zu gering oder zu hoch für die datenhungrigen Hybridmodelle war und das Training nur über 50 Epochen unter Einsatz von EarlyStopping erfolgte.

Dies deutet darauf hin, dass der LSTM-Autoencoder die auffälligsten Muster normaler Logs sehr schnell gelernt hat. Gleichzeitig bedeutet dies, dass die Modelle möglicherweise nicht alle Feinheiten der normalen Daten erfasst haben und daher empfindlicher gegenüber kleinen Variationen oder Anomalien in den Testdaten sind. Das frühe Stoppen verhindert extremes Overfitting, kann aber dazu führen, dass die Rekonstruktion subtiler normaler Muster unvollständig bleibt.

Außerdem wurde das Verhältnis der Features nur unzureichend berücksichtigt. Die abhängige Variable stellt zwar das Label dar, das den Log als anomal oder normal klassifiziert, jedoch wurden IP-Adressen und Features wie Standort und Zeit entfernt. Die Angriffsszenarien mussten daher innerhalb normaler Zeiträume, beispielsweise mittags, stattfinden, um die Fähigkeiten des LSTM-Autoencoders wirklich zu testen. Einflussreicher sind nun Features wie Name, Eventtyp und andere Faktoren. Die Frage, in welchem Ausmaß einzelne Features die Klassifikation beeinflussen, bleibt Gegenstand zukünftiger Arbeiten.

Die Modelle wurden auf einer CPU trainiert, da keine GPU verfügbar war. Aufgrund der begrenzten Rechenressourcen könnten sich Trainingsdauer, Batch-Größe und numerische Präzision auf Modellstabilität und Varianz der Ergebnisse ausgewirkt haben.

Ein weiterer Grund für die beobachteten Unterschiede ist, dass OCSVM sich besonders für kleinere Datensätze eignet, während IF tendenziell bei größeren Datenmengen stärker performt. Die 39.000 Logs waren für OCSVM ausreichend, während IF bei noch größeren Datensätzen potenziell höhere Werte erzielen könnte.

Nach allen metrischen Messungen lässt sich die Hypothese nur im Vergleich zu DBSCAN verifizieren; gegenüber OCSVM hingegen nicht. Somit ist die Hypothese in dieser Arbeit weder vollständig verifizierbar noch falsifizierbar. Änderungen an Parametern, Regularisierungstechniken oder der Trainingskonfiguration könnten zu anderen Ergebnissen führen. Die Resultate hängen stark von Faktoren wie den Hyperparametern, dem Datensatz und der verfügbaren Rechenleistung ab. Unter anderen Bedingungen könnten daher abweichende Ergebnisse erzielt werden. Weitere Studien mit einem alternativen Testsetup, anderen Modellarchitekturen oder zusätzlichen Daten könnten dazu beitragen, die Aussagekraft für diesen Fall zu erhöhen.

13 Fazit

Insgesamt zeigen die Ergebnisse, dass Hybridmodelle in der Lage sind, Anomalien in Keycloak-Logs zuverlässig zu erkennen. Dabei spielen sowohl die Architektur des LSTM-Autoencoders als auch die gewählte Sequenzlänge eine entscheidende Rolle für die Erkennungsleistung, insbesondere angesichts des stark unbalancierten Datensatzes.

Die Methodik stellt sicher, dass sensible Unternehmensdaten nicht verwendet werden: Die Trainings- und Testdaten basieren auf generierten Logs, während durch Selenium-basierte Tests und die Keycloak-API authentische Log-Muster abgebildet werden. Insgesamt unterstreichen die Ergebnisse das Potenzial von LSTM-Autoencodern in Kombination mit klassischen Anomalieerkennungsverfahren für die automatisierte Überwachung von Keycloak-Logs und demonstrieren, dass die Wahl der Modellkombination und der Hyperparameter maßgeblich für die Performance ist.

Die Ergebnisse haben gezeigt, dass die Verifizierung der Hypothese stark von der Wahl der Parameter der LSTM-AE-Architektur, der Größe des Datensatzes und von der Rechenleistung der Grafikkarte abhängig ist, sowie der Verteilung und Anzahl der Anomalien im Datensatz.

Für weitere Studien ist es wichtig, andere unabhängige Variablen oder auch mehreren wie mehreren Datensätzen zu achten, sowie eventuell andere Metriken mehr mit einzubeziehen, wie bspw. die AUC-PR-Werte.

14 Ausblick

14.1 Wirtschaftliche Anwendung

Das Unternehmen intension kann mit weiteren Tests in naher Zukunft und mit mehr Forschung ein Anomalieerkennungstool entwickeln. Mit zusätzlichen kombinierten Produkten wie z.B. ELK³⁷ könnte das Anomalieerkennungstool Insider-Angriffe, Intrusionsangriffe und generelle Anomalien erkennen. Es ist wichtig, dass das Produkt richtig integriert wird und getestet wird, welche Schnittstellen mit welchen Produkten integrierbar sind. Dies erfordert jedoch weiterhin Zeit, Forschung und ein größeres Team, um eine professionelle Umsetzung zu gewährleisten.

14.2 Wissenschaftlicher Ausblick

In Zukunft könnten weitere Algorithmen oder alternative Architekturen in die Untersuchung aufgenommen werden. Mit einer größeren Menge an Trainings- und Testdaten ließe sich die Architektur weiter optimieren und generalisieren. Auf Basis der gewonnenen Erkenntnisse wäre zudem die Entwicklung eines praxistauglichen Anomalieerkennungstools denkbar. Die

konkrete Leistungsfähigkeit hängt jedoch stark vom jeweiligen Testszenario ab. In dieser Arbeit wurden rund 39.000 Log-Einträge verwendet, wobei zwei feste Sequenzlängen und eine spezifische Architektur untersucht wurden. Unter anderen Bedingungen könnten die Erkennungsergebnisse deutlich abweichen. So könnte beispielsweise ein LSTM-Autoencoder kombiniert mit Isolation Forest in einem anderen Szenario bessere Resultate erzielen als OCSVM oder DBSCAN. Daher erscheint es sinnvoll, weitere Tests mit komplexeren Angriffsszenarien, zusätzlichen Trainingsdaten und alternativen Methoden zur Schwellenwertbestimmung durchzuführen.

14.3 Angriffe

Die hier verwendeten Angriffsszenarien waren vergleichsweise einfach gehalten. Angriffe über Drittanbieter im SSO (z.B. eine Attacke auf ein Google-Konto mit anschließendem Zugriff auf Keycloak) oder über das Netzwerk wurden nicht berücksichtigt. Auch Szenarien wie Zertifikatsfälschungen in Keycloak sollten in zukünftigen Studien einbezogen werden. Durch die Einbeziehung realistischerer und variablerer Angriffsmuster könnten die Modelle lernen, mit komplexeren Angriffen umzugehen.

Literatur

- [1] Bundesamt für Sicherheit in der Informationstechnik (BSI), “Lagebericht 2023,” 2023, [Online]. Verfügbar: <https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2023.pdf>. [Zugriff am: Jun. 10, 2025].
- [2] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009, [Online]. Verfügbar: <https://dl.acm.org/doi/10.1145/1541880.1541882>. [Zugriff am: 19. August 2025].
- [3] R. Foorthuis, “On the nature and types of anomalies: A review of deviations in data,” *International Journal of Data Science and Analytics*, vol. 11, no. 2, pp. 127–157, 2021, [Online]. Verfügbar: https://www.researchgate.net/publication/353701611_On_the_nature_and_types_of_anomalies_a_review_of_deviations_in_data. Zugriff am 21. August 2025.
- [4] B. Sanni, “Anomaly detection and user frustration prediction using machine learning in mobile app ux,” ResearchGate Preprint, Dec. 2024. [Online]. Available: https://www.researchgate.net/publication/386375210_Anomaly_Detection_and_User_Frustration_Prediction_Using_Machine_Learning_in_Mobile_App_UX
- [5] M. G. P. Legg, O. Buckley and S. Creese, “Visualizing the insider threat: challenges and tools for identifying malicious user activity,” in *Proc. IEEE Symp. Vis. Cyber Secur. (VizSec)*, 2015, pp. 1–8, [Online]. Verfügbar: <https://ieeexplore.ieee.org/document/7312772>. [Zugriff am: 19. August 2025].
- [6] A. Dahal, P. Bajgai, and N. Rahimi, “Analysis of zero day attack detection using mlp and xai,” *Communications in Computer and Information Science*, vol. 2254, pp. 53–66, 2024, [Online]. Verfügbar: <https://arxiv.org/abs/2501.16638>. [Zugriff am: 19. August 2025].
- [7] G. Yan, Y. Zheng, and C. Lin, “Detecting anomalous user behavior using an extended isolation forest algorithm: An enterprise case study,” in *2021 International Conference on Computer Engineering and Artificial Intelligence (ICCEAI)*. IEEE, 2021, pp. 115–121, [Online]. Verfügbar: https://www.researchgate.net/publication/308409790_Detecting_Anomalous_User_Behavior_Using_an_Extended_Isolation_Forest_Algorithm_An_Enterprise_Case_Study. [Zugriff am 19. August 2025].
- [8] T. Arjunan, “A comparative study of deep neural networks and support vector machines for unsupervised anomaly detection in cloud computing environments,” *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 12, p. 9, 2024, [Online]. Verfügbar: https://www.researchgate.net/publication/378403423_A_Comparative_Study_of_

[Deep_Neural_Networks_and_Support_Vector_Machines_for_Unsupervised_Anomaly_Detection_in_Cloud_Computing_Environments](#). [Zugriff am: 19. August 2025].

- [9] T. M. Thang and J. Kim, “The anomaly detection by using dbscan clustering with multiple parameters,” in *2011 International Conference on Information Science and Applications*. Jeju, Korea (South): IEEE, 26-29 April 2011, pp. 1–6, [Online]. Verfügbar: <https://ieeexplore.ieee.org/document/5772437>. [Zugriff am: 19. August 2025].
- [10] E. Demir and C. Karaoğlu, “A comparative performance analysis of lstm autoencoder and timegpt models in time series anomaly detection,” [Online]. Verfügbar: https://www.researchgate.net/publication/385301485_A_Comparative_Performance_Analysis_of_LSTM_Autoencoder_and_TimeGPT_Models_in_Time_Series_Anomaly_Detection. [Zugriff am: 13. Juni 2025], 2024.
- [11] S. Iseed and M. M. N. Hamarsheh, “Performance evaluation of lstm autoencoder for behavioral-based insider threat detection,” in *2025 International Conference on Smart Applications, Communications and Networking (SmartNets)*. Istanbul, Türkiye: IEEE, July 22-24 2025.
- [12] Y. Wei, J. J. Jaccard, W. Xu, F. Sabrina, S. Camtepe, and M. Boulic, “Lstm-autoencoder-based anomaly detection for indoor air quality time series data,” *IEEE Sensors Journal*, vol. 23, no. 4, pp. 3787–3800, 2023.
- [13] P. H. Tran, C. Heuchenne, and S. Thomassey, “An anomaly detection approach based on the combination of lstm autoencoder and isolation forest for multivariate time series data,” in *Developments of Artificial Intelligence Technologies in Computation and Robotics: Proceedings of the 14th International FLINS Conference (FLINS 2020)*. World Scientific, 2020, pp. 589–596, [Online]. Verfügbar: https://www.worldscientific.com/doi/10.1142/9789811223334_0071. [Zugriff am: 19. August 2025].
- [14] R. J. Z. Ghrib and R. Romdhane, “Hybrid approach for anomaly detection in time series data,” in *2020 IEEE Int. Conf. Intell. Syst. Comput. Vis. (ISCV)*, 2020, pp. 1–7, [Online]. Verfügbar: <https://ieeexplore.ieee.org/document/9207013>. [Zugriff am: 19. August 2025].
- [15] P.-F. Marteau, S. Soheily-Khah, and N. Béchet, “Hybrid isolation forest - application to intrusion detection,” *arXiv preprint arXiv:1705.03800*, pp. 1–24, 2017, [Online]. Verfügbar: <https://arxiv.org/abs/1705.03800>. Zugriff am 21. August 2025.
- [16] T. Ergen and S. S. Kozat, “Unsupervised and semi-supervised anomaly detection with lstm neural networks,” [Online]. Verfügbar: <https://arxiv.org/abs/1710.09207>. [Zugriff am: 20. Juni 2025], 2017.

- [17] D. T. Ha, N. X. Hoang, N. V. Hoang, N. H. Du, T. T. Huong, and K. P. Tran, "Explainable anomaly detection for industrial control system cybersecurity," *Computers in Industry*, vol. 135, p. 103736, 2022. [Online]. Available: <https://doi.org/10.1016/j.compind.2022.103736>
- [18] J. Zhou, S. Zhou, and P. Wang, "Improved lstm-ae-dbscan motor controller temperature abnormality label recognition based method," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 2024, [Online]. Verfügbar: <https://journals.sagepub.com/doi/abs/10.1177/09544070241308017>. Zugriff am 21. August 2025.
- [19] K. Prabu, S. Periyasamy, M. Periyasamy, and A. Alagarsamy, "Harnessing dbscan and auto-encoder for hyper intrusion detection in cloud computing," *Bulletin of Electrical Engineering and Informatics*, vol. 13, no. 5, pp. 3345–3354, 2024, [Online]. Verfügbar: <https://www.beei.org/index.php/EEI/article/viewFile/8135/3880>. Zugriff am 21. August 2025.
- [20] F. Liu, M. Cai, L. Wang, and Y. Lu, "An ensemble model based on adaptive noise reducer and over-fitting prevention lstm for multivariate time series forecasting," *IEEE Access*, vol. 7, pp. 26 102–26 115, 2019, [Online]. Verfügbar: <https://ieeexplore.ieee.org/document/8648338>. Zugriff am 21. August 2025.
- [21] W. Chua, A. L. D. Pajas, C. S. Castro, S. P. Panganiban, A. J. Pasuquin, M. J. Purganan, R. Malupeng, D. J. Pingad, J. P. Orolfo, H. H. Lua, and L. C. Velasco, "Web traffic anomaly detection using isolation forest," *Informatics*, vol. 11, no. 4, p. 83, 2024, [Online]. Verfügbar: https://www.researchgate.net/publication/385531295_Web_Traffic_Anomaly_Detection_Using_Isolation_Forest. [Zugriff am: 19. August 2025].
- [22] D. Ahmadzadeh, M. Jalali, R. Ghaemi, and M. Kheirabadi, "Graphdbscan: Optimized dbscan for noise-resistant community detection in graph clustering," *Future Internet*, vol. 17, no. 4, p. 150, 2025, [Online]. Verfügbar: <https://www.thefreelibrary.com/GraphDBSCAN%3a+Optimized+DBSCAN+for+Noise-Resistant+Community+Detection...-a0838235929>. Zugriff am 21. August 2025. [Online]. Available: <https://doi.org/10.3390/fi17040150>
- [23] J.-E. Moussaoui, M. Kmiti, K. E. Gholami, and Y. Maleh, "A systematic review on hybrid ai models integrating machine learning and federated learning," *J. Cybersecur. Priv.*, vol. 5, no. 3, p. 41, 2025, [Online]. Verfügbar: <https://www.mdpi.com/2624-800X/5/3/41>. [Zugriff am: 19. August 2025].
- [24] H. Lu, "Evaluating the performance of SVM, isolation forest, and DBSCAN for anomaly detection," *ITM Web of Conferences*, vol. 70, p. 04012, 2025, [Online]. Verfügbar:

- https://www.researchgate.net/publication/388323510_Evaluating_the_Performance_of_SVM_Isolation_Forest_and_DBSCAN_for_Anomaly_Detection. Zugriff am 21. August 2025.
- [25] D. N. Tran and S. Thomassey, “Forecasting and anomaly detection approaches using lstm and lstm autoencoder techniques with the applications in supply chain management,” *International Journal of Information Management*, vol. 57, p. 102282, 2021, [Online]. Verfügbar: <https://www.sciencedirect.com/science/article/pii/S026840122031481X>.
- [26] F. Provost, T. Fawcett, and R. Kohavi, “The case against accuracy estimation for comparing induction algorithms,” in *Proceedings of the 15th International Conference on Machine Learning (ICML)*. Madison, WI, USA: Morgan Kaufmann Publishers, 1998, [Online]. Verfügbar: https://www.researchgate.net/publication/2373067_The_Case_Against_Accuracy_Estimation_for_Comparing_Induction_Algorithms. [Zugriff am 28. Juni 2025].
- [27] N. Japkowicz and S. Stephen, “The class imbalance problem: A systematic study,” *Intelligent Data Analysis*, vol. 6, no. 5, pp. 429–449, 2002, [Online]. Verfügbar: <https://dl.acm.org/doi/10.5555/1293951.1293954>. [Zugriff am 28. Juni 2025].
- [28] D. Fourure, M. Mazurowski, J. Amar, and A. Derreumaux, “Anomaly detection: How to artificially increase your f1-score with a biased evaluation protocol,” *arXiv preprint arXiv:2106.16020*, 2021, [Online]. Verfügbar unter: <https://arxiv.org/abs/2106.16020>. Zugriff am: 23. Juni 2025.
- [29] J. Davis and M. Goadrich, “The relationship between precision-recall and roc curves,” in *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 233–240, [Online]. Verfügbar: <https://dl.acm.org/doi/10.1145/1143844.1143874>. [Zugriff am 28. Juni 2025].
- [30] D. Chicco and G. Jurman, “The matthews correlation coefficient (mcc) is more informative than cohen’s kappa and brier score in binary classification assessment,” *BMC Genomics*, vol. 21, no. 1, p. 6, 2020, [Online]. Verfügbar: <https://bmcbgenomics.biomedcentral.com/articles/10.1186/s12864-019-6413-7>. [Zugriff am 28. Juni 2025].
- [31] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, “The balanced accuracy and its posterior distribution,” in *2010 20th International Conference on Pattern Recognition*, 2010, pp. 3121–3124, [Online]. Verfügbar: <https://ieeexplore.ieee.org/document/5597285>. [Zugriff am 28. Juni 2025].
- [32] Y. Acikmese and S. E. Alptekin, “Prediction of stress levels with lstm and passive mobile sensors,” *Procedia Computer Science*, vol. 158, pp. 221–228, 2019, [Online]. Verfügbar:

- <https://www.sciencedirect.com/science/article/pii/S187705091931405X>. Zugriff am 21. August 2025. [Online]. Available: <https://doi.org/10.1016/j.procs.2019.09.221>
- [33] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997, [Online]. Verfügbar: https://www.researchgate.net/publication/13853244_Long_Short-Term_Memory. [Zugriff am: 19. August 2025].
- [34] R. C. Staudemeyer and E. R. Morris, “Understanding lstm – a tutorial into long short-term memory recurrent neural networks,” *arXiv preprint arXiv:1909.09586*, 2019, [Online]. Verfügbar: <https://arxiv.org/abs/1909.09586>. [Zugriff am 25. Juni 2025].
- [35] J. L. X. Wei and L. Zhang, “Lstm autoencoder-based anomaly detection for indoor air quality data,” *arXiv preprint arXiv:2204.06701*, 2022, [Online]. Verfügbar: <https://arxiv.org/abs/2204.06701>. [Zugriff am: 13. Juni 2025].
- [36] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994, [Online]. Verfügbar: <https://doi.org/10.1109/72.279181>. [Zugriff am 28. Juni 2025].
- [37] J. Chung, S. Ahn, and Y. Bengio, “Hierarchical multiscale recurrent neural networks,” *arXiv preprint arXiv:1609.01704*, 2017, [Online]. Verfügbar: <https://arxiv.org/abs/1609.01704>. [Zugriff am 28. Juni 2025].
- [38] N. El-Naggar, P. Madhyastha, and T. Weyde, “Exploring the long-term generalization of counting behavior in rnns,” in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022, [Online]. Verfügbar: <https://arxiv.org/abs/2211.16429>. [Zugriff am 28. Juni 2025].
- [39] U. Michelucci, “An introduction to autoencoders,” Jan. 2022, [Online]. Verfügbar: <https://arxiv.org/abs/2201.03898>. [Zugriff am 28. Juni 2025].
- [40] F. Laakom, J. Raitoharju, A. Iosifidis, and M. Gabbouj, “Reducing redundancy in the bottleneck representation of autoencoders,” *Information Sciences*, 2022, [Online]. Verfügbar: <https://www.sciencedirect.com/science/article/pii/S0167865524000126>. Abgerufen am 28. Juni 2025. [Zugriff am: 19. August 2025].
- [41] E. Pantelidis, G. Bendiab, S. Shiaeles, and N. Kolokotronis, “Insider detection using deep autoencoder and variational autoencoder neural networks,” in *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2021, pp. 1–7. [Online]. Available: <https://arxiv.org/abs/2109.02568>

- [42] W. Skaf and T. Horváth, “Denoising architecture for unsupervised anomaly detection in time-series,” in *arXiv preprint arXiv:2208.14337*, 2022. [Online]. Available: <https://arxiv.org/abs/2208.14337>
- [43] Y. Wei, K.-P. Chow, and S.-M. Yiu, “Insider threat detection using multi-autoencoder filtering and unsupervised learning,” in *Advances in Digital Forensics XVI, IFIP Advances in Information and Communication Technology*, vol. 589. Springer, 2020, pp. 273–290. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-56223-6_15
- [44] F. T. Liu, K. M. Ting, and Z. H. Zhou, “Isolation forest,” in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422, [Online]. Verfügbar: https://www.researchgate.net/publication/224384174_Isolation_Forest. [Zugriff am: 19. August 2025].
- [45] K. Yang, S. Kpotufe, and N. Feamster, “An efficient one-class svm for anomaly detection in the internet of things,” *arXiv preprint arXiv:2104.11146*, 2021. [Online]. Available: <https://doi.org/10.48550/arXiv.2104.11146>
- [46] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, “Estimating the support of a high-dimensional distribution,” *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.
- [47] K.-L. Li, H.-K. Huang, S.-F. Tian, and W. Xu, “Improving one-class svm for anomaly detection,” in *Proceedings of the Second International Conference on Machine Learning and Cybernetics*, 2003, pp. 2–5, [Online]. Verfügbar: <https://ieeexplore.ieee.org/document/1260106>. [Zugriff am: 25. Juni 2025].
- [48] A. Bounsiar and M. G. Madden, “One-class support vector machines revisited,” in *2014 International Conference on Information Science & Applications (ICISA)*. Seoul, Korea: IEEE, 2014, [Online]. Verfügbar: <https://ieeexplore.ieee.org/document/6847442>. [Zugriff am: 19. August 2025].
- [49] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*. AAAI Press, 1996, pp. 226–231, [Online]. Verfügbar: <https://dl.acm.org/doi/10.5555/3001460.3001507>. [Zugriff am: 19. August 2025].
- [50] R. J. Campello, D. Moulavi, A. Zimek, and J. Sander, “Hierarchical density estimates for data clustering, visualization, and outlier detection,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 10, no. 1, pp. 1–51, 2015, [Online]. Verfügbar: <https://doi.org/10.1145/2733381>. [Zugriff am: 05. August 2025].

- [51] B. V. S. Vines, E. Schubert, A. Zimek, and R. L. F. Cordeiro, “A comparative evaluation of clustering-based outlier detection,” *Data Mining and Knowledge Discovery*, 2025, [Online]. Verfügbar: <https://doi.org/10.1007/s10618-024-01086-z>. [Zugriff am 05. August 2025].
- [52] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “Lof: Identifying density-based local outliers,” in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: ACM, May 2000, pp. 93–104, [Online]. Verfügbar: <https://doi.org/10.1145/342009.335388>. [Zugriff am: 05.August 2025].
- [53] E. H. Budiarto, A. E. Permanasari, and S. Fauziati, “Unsupervised anomaly detection using k-means, local outlier factor and one class svm,” Department of Electrical Engineering and Information Technology, Universitas Gadjah Mada, Yogyakarta, Indonesia, Tech. Rep., 2025, [Online]. Verfügbar: <https://ieeexplore.ieee.org/document/9166366>. [Zugriff am 05. August 2025].
- [54] A. Zimek, E. Schubert, and H.-P. Kriegel, “A survey on unsupervised outlier detection in high-dimensional numerical data,” *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 5, no. 5, pp. 363–387, 2012, [Online]. Verfügbar: <https://doi.org/10.1002/sam.11161>. [Zugriff am 05. August 2025].
- [55] R. Mussabayev and R. Mussabayev, “Comparative analysis of optimization strategies for k-means clustering in big data contexts: A review,” *Journal of Big Data*, 2023, [Online]. Verfügbar: <https://journalofbigdata.springeropen.com/articles/10.1186/s40537-023-00799-6>. [Zugriff am 05. August 2025].
- [56] A. N.-B. C. Horyń, “Analysis of the effectiveness and efficiency of lof algorithm for anomaly detection in large datasets,” in *Proceedings in SpringerLink*, 2023, [Online]. Verfügbar: https://link.springer.com/chapter/10.1007/978-3-031-50959-9_43. [Zugriff am 05. August 2025].
- [57] A. A. Toor, J.-C. Lin, and E. G. Gran, “Exploring the impact of optimised hyperparameters on bi-lstm-based contextual anomaly detector,” in *Proc. Int. Conf. Inf. Security and Communication Technology*, Gjøvik, Norway, 2022, [Online]. Verfügbar: <https://arxiv.org/abs/2501.15053>. [Zugriff am: 19. August 2025].
- [58] B. Santoso, W. Anggraeni, H. Pariaman, and M. H. Purnomo, “Rnn-autoencoder approach for anomaly detection in power plant predictive maintenance systems,” *Int. J. Intell. Eng. Syst.*, vol. 15, no. 4, pp. 422–433, 2022, [Online]. Verfügbar: https://www.researchgate.net/publication/361525909_RNN-Autoencoder_Approach_for_Anomaly_Detection_in_Power_Plant_Predictive_Maintenance_Systems. [Zugriff am: 19. August 2025].

- [59] D. Masters and C. Luschi, “Revisiting small batch training for deep neural networks,” arXiv preprint arXiv:1804.07612, 2018, [Online]. Verfügbar: <https://arxiv.org/abs/1804.07612>. [Zugriff am: 18. August 2025].
- [60] B. M. Hussein and S. M. Shareef, “An empirical study on the correlation between early stopping patience and epochs in deep learning,” in *ITM Web of Conferences, ICACS24*, vol. 64. EDP Sciences, 2024, p. 01003, open access under CC BY 4.0.
- [61] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, “Lstm-based encoder-decoder for multi-sensor anomaly detection,” *arXiv preprint arXiv:1607.00148*, 2016, [Online]. Verfügbar: <https://arxiv.org/abs/1607.00148>. [Zugriff am 25. Juni 2025].
- [62] T. Acharya, A. Annamalai, and M. F. Chouikha, “Efficacy of bidirectional lstm model for network-based anomaly detection,” in *2023 IEEE 13th Symposium on Computer Applications & Industrial Electronics (ISCAIE)*. Penang, Malaysia: IEEE, May 2023, pp. 1–6, [Online]. Verfügbar: <https://ieeexplore.ieee.org/document/10165336>. Zugriff am 20. August 2025.
- [63] S. P. K. Gudla and S. K. Bhoi, “A study on effect of learning rates using adam optimizer in lstm deep intelligent model for detection of ddos attack to support fog based iot systems,” in *Proceedings of the International Conference on Computing, Communication and Learning (CoCoLe 2022)*, ser. Communications in Computer and Information Science (CCIS), vol. 1729. Springer, 2023, pp. 27–38, [Online]. Verfügbar: https://link.springer.com/chapter/10.1007/978-3-031-21750-0_3. Zugriff am 20. August 2025.
- [64] H. Torabi, S. L. Mirtaheri, and S. Greco, “Practical autoencoder based anomaly detection by using vector reconstruction error,” *Journal of Big Data*, vol. 10, no. 1, pp. 1–21, 2023, [Online]. Verfügbar: <https://link.springer.com/content/pdf/10.1186/s42400-022-00134-9.pdf>. [Zugriff am: 19. August 2025].
- [65] A. Jablonski and K. Mendrok, “Automatic threshold setting for anomaly detection,” *AGH University of Krakow, Faculty of Mechanical Engineering and Robotics*, 2023, [Online]. Verfügbar: <https://www.sciencedirect.com/science/article/pii/S0888327025001633>. [Zugriff am: 19. August 2025].
- [66] K. Wilkinghoff and K. Imoto, “F1-ev score: Measuring the likelihood of estimating a good decision threshold for semi-supervised anomaly detection,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 256–260, [Online]. Verfügbar: <https://ieeexplore.ieee.org/document/10446011>. [Zugriff am: 19. August 2025].

- [67] C. Y. Priyanto, Hendry, and H. D. Purnomo, "Combination of isolation forest and lstm autoencoder for anomaly detection," in *2021 International Conference on Artificial Intelligence and Mechatronics Systems (AIMS)*. IEEE, 2021, [Online]. Verfügbar: <https://doi.org/10.1109/AIMS53203.2021.9588259>. [Zugriff am 04. August 2025].
- [68] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778, [Online]. Verfügbar: <https://ieeexplore.ieee.org/document/7780459>. [Zugriff am 04. August 2025].
- [69] S. A. Najafi, M. H. Asemani, and P. Setoodeh, "Attention and autoencoder hybrid model for unsupervised online anomaly detection," *arXiv preprint*, 2024, [Online]. Verfügbar: <https://arxiv.org/abs/2401.03322>. [Zugriff am: 19. August 2025].
- [70] A. P. Singh and A. Sharma, "A systematic literature review on insider threats," *arXiv preprint*, 2022.
- [71] P. Golestaneh, M. Taheri, and J. Lederer, "How many samples are needed to train a deep neural network?" *arXiv preprint arXiv:2101.01235*, 2021, [Online]. Verfügbar: <https://arxiv.org/abs/2101.01235>. [Zugriff am 31. Juni 2025].
- [72] T. I. Götz, S. Göb, S. Sawant, X. F. Erick, T. Wittenberg, C. Schmidkonz, A. M. Tomé, E. W. Lang, and A. Ramming, "Number of necessary training examples for neural networks with different number of trainable parameters," *Neurocomputing*, 2023, [Online]. Verfügbar: <https://doi.org/10.1016/j.neucom.2023.02.087>. [Zugriff am 31. Juni 2025].
- [73] D. G. Pereira, A. Afonso, and F. M. Medeiros, "Overview of friedman's test and post-hoc analysis," *Journal of Statistics Education*, vol. 23, no. 3, pp. 2636–2653, 2015. [Online]. Available: <https://www.tandfonline.com/doi/abs/10.1080/03610918.2014.931971>

Fußnotenverzeichnis

- 1 Beleg hier im "Pendleratlas: <https://statistik.arbeitsagentur.de/DE/Navigation/Statistiken/Interaktive-Statistiken/Pendleratlas/Pendleratlas-Nav.html>, letzter Zugriff am 28.August.2025
- 2 Siehe die Doku dazu: Keycloak Project: <https://www.keycloak.org/>, letzter Zugriff am 28.August.2025
- 3 IAM steht für *Identity and Access Management* und beschreibt Systeme zur Verwaltung digitaler Identitäten und Zugriffskontrollen
- 4 Single Sign-On ermöglicht einem Nutzer den Zugriff auf mehrere Anwendungen mit nur einer Anmeldung
- 5 Genauere Informationen zu Timing-Angriffe im National Vulnerability Database (NVD) unter „CVE-2024-4629 Vulnerability Details“, <https://nvd.nist.gov/vuln/detail/CVE-2024-4629>, letzter Zugriff am 28.August.2025
- 6 Entdeckt in CVEDetails, „CVE-2025-3501 Details“: <https://www.cvedetails.com/cve/CVE-2025-3501>, letzter Zugriff am 28.August.2025
- 7 Erklärung der Modellbewertung: https://scikit-learn.org/stable/modules/cross_validation.html, letzter Zugriff am 28.August.2025
- 8 Definition unter: <https://www.sciencedirect.com/topics/engineering/regularization>, letzter Zugriff am 28.August.2025
- 9 Mehr Informationen zu Scikit sind hier nachzulesen: <https://scikit-learn.org/stable/>, letzter Zugriff am 28.August.2025
- 10 Offizielle Dokumentation zu Keras ist verfügbar unter: <https://keras.io/>, letzter Zugriff am 28.August.2025
- 11 TensorFlows offizielle Dokumentation ist verfügbar unter: <https://www.tensorflow.org/>, letzter Zugriff am 28.August.2025
- 12 <https://www.nvidia.com/en-us/data-center/dgx-systems/>, letzter Zugriff am 28.August.2025
- 13 <https://docs.nvidia.com/dgx/dgxm100-user-guide/introduction-to-dgxm100.html>, letzter Zugriff am 28.August.2025
- 14 Also ein Token, welches im Prinzip für die Autorisierung notwendig ist, um dem Server mitzuteilen, dass er Berechtigungen hat auf eine Ressource zuzugreifen
- 15 Mehr Informationen zu Authentifizierungsprozess in Keycloak siehe: <https://www.oauth.com/oauth2-servers/server-side-apps/authorization-code/>, letzter Zugriff am 28.August.2025
- 16 <https://www.redhat.com/de/technologies/jboss-middleware/application-platform>, letzter Zugriff am 28.August.2025
- 17 Mehr Infos unter: <https://www.wildfly.org/>, letzter Zugriff am 28.August.2025
- 18 <https://www.keycloak.org/docs-api/25.0.6/javadocs/org/keycloak/events/EventListenerProvider.html>, letzter Zugriff am 28.August.2025
- 19 <https://www.keycloak.org/server/logging>, letzter Zugriff am 28.August.2025
- 20 <https://github.com/keycloak/keycloak/blob/main/model/jpa/src/main/java/org/keycloak/events/jpa/JpaAdminEventQuery.java>, letzter Zugriff am 28.August.2025
- 21 <https://de.python-3.com/?p=45278>, letzter Zugriff am 28.August.2025

- 22 <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html>, letzter Zugriff am 28.August.2025
- 23 <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>, letzter Zugriff am 28.August.2025
- 24 <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html>, letzter Zugriff am 28.August.2025
- 25 Definition des MSE: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html
- 26 Definition des Adam-Optimierers: <https://docs.pytorch.org/docs/stable/generated/torch.optim.Adam.html>, letzter Zugriff am 18. August 2025
- 27 Viele Modelle geben keine direkten Wahrscheinlichkeiten, sondern numerische Scores aus, die anzeigen, wie stark das Modell zu einer Klasse tendiert. Dies ist dann ein Rohscore
- 28 Detaillierte Dokumentation: <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html>
- 29 Internet Assigned Numbers Authority (IANA), „IANA IPv4 Special Registry“, <https://www.iana.org/assignments/iana-ipv4-special-registry/iana-ipv4-special-registry.xhtml>, letzter Zugriff am 28.August.2025
- 30 Keycloak Documentation, „REST API Reference“, <https://www.keycloak.org/docs-api/latest/rest-api/index.html>, letzter Zugriff am 28.August.2025
- 31 <https://www.selenium.dev/>, letzter Zugriff am 28.August.2025
- 32 MITRE Corporation, „CVE Search Results for Keycloak“, <https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=keycloak>, letzter Zugriff am 28.August.2025
- 33 CVE-Ticket: <https://www.cve.news/cve-2024-3656/>, letzter Zugriff am 28.August.2025
- 34 Unix Timestamp Umwandler: <https://www.unixtimestamp.com/>, letzter Zugriff am 28.August.2025
- 35 <https://www.sciencedirect.com/science/article/abs/pii/S0169743989800954>, letzter Zugriff am 28.August.2025
- 36 Ein post-hoc-Test ist ein statistisches Verfahren, das nach einem Test durchgeführt wird, um genau zu bestimmen, welche Gruppen sich signifikant voneinander unterscheiden.
- 37 Definition von ELK: <https://www.elastic.co/elastic-stack/>, letzter Zugriff am 28.August.2025

Anhang

Ergebnisse mit der Sequenzlänge 25

14.3.1 Sequenzlänge 25, Testlauf 1

Tabelle 17: Ergebnisse von LSTM-AE-IF (Sequenzlänge 25, Testlauf 1)

Metrik	Wert
Best Threshold by F1	-0.1938
Precision	0.9461
Recall	0.6345
F1-Score	0.7596
ROC-AUC	0.9295
AUC-PR	0.6159
Matthews Correlation Coefficient (MCC)	0.7673
Balanced Accuracy	0.8165
Accuracy	0.9829

Tabelle 18: Ergebnisse von LSTM-AE-OCSVM (Sequenzlänge 25, Testlauf 1)

Metrik	Wert
Best Threshold by F1	-0.0033
Precision	0.8956
Recall	0.6546
F1-Score	0.7564
ROC-AUC	0.7532
AUC-PR	0.6010
Matthews Correlation Coefficient (MCC)	0.7572
Balanced Accuracy	0.8256
Accuracy	0.9821

Tabelle 19: Ergebnisse von LSTM-AE-DBSCAN(Sequenzlänge 25, Testlauf 1)

Metrik	Wert
Passender eps-Wert	0.07
Anomalien	32
Precision	1.0000
Recall	0.1285
F1-Score	0.2278
Matthews Correlation Coefficient (MCC)	0.3517
Balanced Accuracy	0.5643
ROC-AUC	0.5643
AUC-PR	0.1656
Accuracy	0.9630

14.3.2 Sequenzlänge 25, Testlauf 2

Tabelle 20: Ergebnisse von LSTM-AE-IF (Sequenzlänge 25, Testlauf 2)

Metrik	Wert
Best Threshold by F1	-0.1885
Precision	0.9689
Recall	0.6265
F1-Score	0.7610
ROC-AUC	0.9391
AUC-PR	0.6229
Matthews Correlation Coefficient (MCC)	0.7720
Balanced Accuracy	0.8128
Accuracy	0.9833

Tabelle 21: Ergebnisse von LSTM-AE-OCSVM (Sequenzlänge 25, Testlauf 2)

Metrik	Wert
Best Threshold by F1	-0.0023
Precision	0.9573
Recall	0.6305
F1-Score	0.7603
ROC-AUC	0.7528
AUC-PR	0.6193
Matthews Correlation Coefficient (MCC)	0.7696
Balanced Accuracy	0.8146
Accuracy	0.9831

Tabelle 22: Ergebnisse von LSTM-AE-DBSCAN (Sequenzlänge 25, Testlauf 2)

Metrik	Wert
Passender eps-Wert	0.08
Anomalien	31
Precision	1.0000
Recall	0.1245
F1-Score	0.2214
Matthews Correlation Coefficient (MCC)	0.3462
Balanced Accuracy	0.5622
ROC-AUC	0.5622
AUC-PR	0.1617
Accuracy	0.9628

14.3.3 Sequenzlänge 25, Testlauf 3

Tabelle 23: Ergebnisse von LSTM-AE-IF (Sequenzlänge 25, Testlauf 3)

Metrik	Wert
Best Threshold by F1	-0.1852
Precision	0.8564
Recall	0.6466
F1-Score	0.7368
ROC-AUC	0.9315
AUC-PR	0.5688
Matthews Correlation Coefficient (MCC)	0.7346
Balanced Accuracy	0.8209
Accuracy	0.9804

Tabelle 24: Ergebnisse von LSTM-AE-OCSVM (Sequenzlänge 25, Testlauf 3)

Metrik	Wert
Best Threshold by F1	-0.0036
Precision	0.8482
Recall	0.6506
F1-Score	0.7364
ROC-AUC	0.7511
AUC-PR	0.5667
Matthews Correlation Coefficient (MCC)	0.7332
Balanced Accuracy	0.8227
Accuracy	0.9802

Tabelle 25: Ergebnisse von LSTM-AE-DBSCAN (Sequenzlänge 25, Testlauf 3)

Metrik	Wert
Precision	1.0000
Recall	0.0442
F1-Score	0.0846
Matthews Correlation Coefficient (MCC)	0.2059
Balanced Accuracy	0.5221
ROC-AUC	0.5221
AUC-PR	0.0848
Accuracy	0.9594

14.3.4 Sequenzlänge 25, Testlauf 4

Tabelle 26: Ergebnisse von LSTM-AE-IF (Sequenzlänge 25, Testlauf 4)

Metrik	Wert
Best Threshold by F1	-0.1754
Precision	0.9235
Recall	0.6305
F1-Score	0.7494
ROC-AUC	0.9323
AUC-PR	0.5980
Matthews Correlation Coefficient (MCC)	0.7550
Balanced Accuracy	0.8141
Accuracy	0.9821

Tabelle 27: Ergebnisse von LSTM-AE-OCSVM (Sequenzlänge 25, Testlauf 4)

Metrik	Wert
Best Threshold by F1	-0.0025
Precision	0.7265
Recall	0.6827
F1-Score	0.7039
ROC-AUC	0.7498
AUC-PR	0.5095
Matthews Correlation Coefficient (MCC)	0.6916
Balanced Accuracy	0.8357
Accuracy	0.9756

Tabelle 28: Ergebnisse von LSTM-AE-DBSCAN (Sequenzlänge 25, Testlauf 4)

Metrik	Wert
Passender eps-Wert	0.06
Anomalien	31
Precision	1.0000
Recall	0.1245
F1-Score	0.2214
Matthews Correlation Coefficient (MCC)	0.3462
Balanced Accuracy	0.5622
ROC-AUC	0.5622
AUC-PR	0.1617
Accuracy	0.9628

14.3.5 Sequenzlänge 25, Testlauf 5

Tabelle 29: Ergebnisse von LSTM-AE-IF (Sequenzlänge 25, Testlauf 5)

Metrik	Wert
Best Threshold by F1	-0.2036
Precision	0.9016
Recall	0.6627
F1-Score	0.7639
ROC-AUC	0.9354
AUC-PR	0.6118
Matthews Correlation Coefficient (MCC)	0.7647
Balanced Accuracy	0.8297
Accuracy	0.9826

Tabelle 30: Ergebnisse von LSTM-AE-OCSVM (Sequenzlänge 25, Testlauf 5)

Metrik	Wert
Best Threshold by F1	-0.0030
Precision	0.8579
Recall	0.6787
F1-Score	0.7578
ROC-AUC	0.7523
AUC-PR	0.5959
Matthews Correlation Coefficient (MCC)	0.7540
Balanced Accuracy	0.8369
Accuracy	0.9816

Tabelle 31: Ergebnisse von LSTM-AE-DBSCAN (Sequenzlänge 25, Testlauf 5)

Metrik	Wert
Passender eps-Wert	0.07
Anomalien	29
Precision	1.0000
Recall	0.1165
F1-Score	0.2086
Matthews Correlation Coefficient (MCC)	0.3348
Balanced Accuracy	0.5582
ROC-AUC	0.5582
AUC-PR	0.1540
Accuracy	0.9624

Ergebnisse mit der Sequenzlänge 45

14.3.6 Sequenzlänge 25, Testlauf 1

Tabelle 32: Ergebnisse von LSTM-AE-IF (Sequenzlänge 45, Testlauf 1)

Metrik	Wert
Best Threshold by F1	-0.2616
Precision	0.6174
Recall	0.6015
F1-Score	0.6094
ROC-AUC	0.7344
AUC-PR	0.3980
Matthews Correlation Coefficient (MCC)	0.5819
Balanced Accuracy	0.7875
Accuracy	0.9486

Tabelle 33: Ergebnisse von LSTM-AE-OCSVM (Sequenzlänge 45, Testlauf 1)

Metrik	Wert
Best Threshold by F1	-0.0214
Precision	0.6786
Recall	0.6350
F1-Score	0.6560
ROC-AUC	0.6910
AUC-PR	0.4552
Matthews Correlation Coefficient (MCC)	0.6328
Balanced Accuracy	0.8067
Accuracy	0.9556

Tabelle 34: Ergebnisse von LSTM-AE-DBSCAN (Sequenzlänge 45, Testlauf 1)

Metrik	Wert
Passender eps-Wert	0.04
Anomalien	29
Precision	1.0000
Recall	0.0746
F1-Score	0.1388
Matthews Correlation Coefficient (MCC)	0.2644
Balanced Accuracy	0.5373
ROC-AUC	0.5373
AUC-PR	0.1362
Accuracy	0.9383

14.3.7 Sequenzlänge 25, Testlauf 2

Tabelle 35: Ergebnisse von LSTM-AE-IF (Sequenzlänge 45, Testlauf 2)

Metrik	Wert
Best Threshold by F1	-0.2639
Precision	0.6489
Recall	0.6272
F1-Score	0.6379
ROC-AUC	0.8038
AUC-PR	0.4319
Matthews Correlation Coefficient (MCC)	0.6126
Balanced Accuracy	0.8015
Accuracy	0.9525

Tabelle 36: Ergebnisse von LSTM-AE-OCSVM (Sequenzlänge 45, Testlauf 2)

Metrik	Wert
Best Threshold by F1	-0.0234
Precision	0.6464
Recall	0.6298
F1-Score	0.6380
ROC-AUC	0.6873
AUC-PR	0.4318
Matthews Correlation Coefficient (MCC)	0.6126
Balanced Accuracy	0.8026
Accuracy	0.9524

Tabelle 37: Ergebnisse von LSTM-AE-DBSCAN (Sequenzlänge 45, Testlauf 2)

Metrik	Wert
Precision	1.0000
Recall	0.0257
F1-Score	0.0501
Matthews Correlation Coefficient (MCC)	0.1550
Balanced Accuracy	0.5129
ROC-AUC	0.5129
AUC-PR	0.0906
Accuracy	0.9351

14.3.8 Sequenzlänge 25, Testlauf 3

Tabelle 38: Ergebnisse von LSTM-AE-IF (Sequenzlänge 45, Testlauf 3)

Metrik	Wert
Best Threshold by F1	-0.2698
Precision	0.6305
Recall	0.6272
F1-Score	0.6289
ROC-AUC	0.8627
AUC-PR	0.4203
Matthews Correlation Coefficient (MCC)	0.6024
Balanced Accuracy	0.8005
Accuracy	0.9507

Tabelle 39: Ergebnisse von LSTM-AE-OCSVM (Sequenzlänge 45, Testlauf 3)

Metrik	Wert
Best Threshold by F1	-0.0147
Precision	0.6604
Recall	0.6350
F1-Score	0.6474
ROC-AUC	0.6867
AUC-PR	0.4437
Matthews Correlation Coefficient (MCC)	0.6229
Balanced Accuracy	0.8058
Accuracy	0.9539

Tabelle 40: Ergebnisse von LSTM-AE-DBSCAN (Sequenzlänge 45, Testlauf 3)

Metrik	Wert
Passender eps-Wert	0.05
Anomalien	31
Precision	1.0000
Recall	0.0797
F1-Score	0.1476
Matthews Correlation Coefficient (MCC)	0.2735
Balanced Accuracy	0.5398
ROC-AUC	0.5398
AUC-PR	0.1410
Accuracy	0.9387

14.3.9 Sequenzlänge 25, Testlauf 4

Tabelle 41: Ergebnisse von LSTM-AE-IF (Sequenzlänge 45, Testlauf 4)

Metrik	Wert
Best Threshold by F1	-0.2671
Precision	0.5581
Recall	0.6298
F1-Score	0.5918
ROC-AUC	0.7926
AUC-PR	0.3762
Matthews Correlation Coefficient (MCC)	0.5619
Balanced Accuracy	0.7971
Accuracy	0.9421

Tabelle 42: Ergebnisse von LSTM-AE-OCSVM (Sequenzlänge 45, Testlauf 4)

Metrik	Wert
Best Threshold by F1	-0.0076
Precision	0.8356
Recall	0.4704
F1-Score	0.6020
ROC-AUC	0.6848
AUC-PR	0.4284
Matthews Correlation Coefficient (MCC)	0.6087
Balanced Accuracy	0.7319
Accuracy	0.9585

Tabelle 43: Ergebnisse von LSTM-AE-DBSCAN (Sequenzlänge 45, Testlauf 4)

Metrik	Wert
Passender eps-Wert	0.05
Anomalien	30
Precision	1.0000
Recall	0.0771
F1-Score	0.1432
Matthews Correlation Coefficient (MCC)	0.2690
Balanced Accuracy	0.5386
ROC-AUC	0.5386
AUC-PR	0.1386
Accuracy	0.9385

14.3.10 Sequenzlänge 25, Testlauf 5

Tabelle 44: Ergebnisse von LSTM-AE-IF (Sequenzlänge 45, Testlauf 5)

Metrik	Wert
Best Threshold by F1	-0.2633
Precision	0.6533
Recall	0.5861
F1-Score	0.6179
ROC-AUC	0.7442
AUC-PR	0.4105
Matthews Correlation Coefficient (MCC)	0.5932
Balanced Accuracy	0.7820
Accuracy	0.9517

Tabelle 45: Ergebnisse von LSTM-AE-OCSVM (Sequenzlänge 45, Testlauf 5)

Metrik	Wert
Best Threshold by F1	-0.0181
Precision	0.6568
Recall	0.6247
F1-Score	0.6403
ROC-AUC	0.6866
AUC-PR	0.4353
Matthews Correlation Coefficient (MCC)	0.6155
Balanced Accuracy	0.8007
Accuracy	0.9532

Tabelle 46: Ergebnisse von LSTM-AE-DBSCAN (Sequenzlänge 45, Testlauf 5)

Metrik	Wert
Passender eps-Wert	0.05
Anomalien	29
Precision	1.0000
Recall	0.0746
F1-Score	0.1388
Matthews Correlation Coefficient (MCC)	0.2644
Balanced Accuracy	0.5373
ROC-AUC	0.5373
AUC-PR	0.1362
Accuracy	0.9383