

Effektivität von KI in der Erkennung von Cyber-Sicherheitsvorfällen: Ein Vergleich mit traditionellen Firewalls

Hendrik Merz (IU14073316)

Wirtschaftsinformatik (B.Sc.)
Bachelorthesis 2. September 2024

Tutor: Prof. Dr. Alexander Lawall

EIDESSTATTLICHE ERKLÄRUNG



Hiermit versichere ich an Eides statt, dass ich die Abschlussarbeit selbstständig und ohne Inanspruchnahme fremder Hilfe angefertigt habe. Ich habe dabei nur die angegebenen Quellen und Hilfsmittel verwendet und die aus diesen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen. Ich erkläre mich damit einverstanden, dass die Arbeit mit Hilfe eines Plagiatserkennungsdienstes auf enthaltene Plagiäte überprüft wird.

Niederau, 01.09.24

Ort, Datum

H. Hoffmann

Unterschrift

Abstract

Diese Forschungsarbeit untersucht die Effektivität von künstlicher Intelligenz (KI) im Vergleich zu traditionellen Firewalls bei der Erkennung von Cyber-Sicherheitsvorfällen. Angesichts der zunehmenden Komplexität und Dynamik von Cyber-Bedrohungen wird die Notwendigkeit flexibler und anpassungsfähiger Abwehrmechanismen immer wichtiger. Traditionelle signaturbasierte Systeme stoßen hierbei an ihre Grenzen. Die Arbeit vergleicht eine KI-basierte Firewall mit einer traditionellen Firewall anhand der Erkennungsgenauigkeit, Ressourceneffizienz und Implementierungskomplexität. Dazu wurden zwei Prototypen entwickelt und anhand von simulierten Angriffen auf Serverumgebungen getestet. Die Ergebnisse zeigen, dass die KI-basierte Firewall insbesondere bei der Erkennung von Bedrohungen überlegen ist, während die traditionelle Firewall durch eine höhere Verarbeitungsgeschwindigkeit und geringere Ressourcennutzung punktet. Die Studie liefert wertvolle Erkenntnisse für die Wahl der optimalen Cyber-Security-Strategien und gibt Empfehlungen für die Implementierung und Wartung KI-gestützter Firewalls in der Praxis.

Inhaltsverzeichnis

Abbildungsverzeichnis	V
Tabellenverzeichnis	VI
Abkürzungsverzeichnis	VII
1 Einleitung	1
1.1 Problemstellung und Motivation	1
1.2 Zielsetzung der Arbeit	2
1.3 Aufbau der Arbeit	2
1.4 Formulierung der Forschungsfrage	3
2 Beschreibung des Themas	5
2.1 OSI-Modell	5
2.2 Firewalls	6
2.3 Künstliche Intelligenz - Machine Learning	7
2.3.1 Künstliche Intelligenz in einer Firewall	10
2.4 Definition und Überblick über Cyber-Security-Bedrohungen	10
2.5 Angriffsarten	12
2.6 Klassifizierung von Firewall Ergebnissen	13
2.7 Aktueller Stand der Forschung	13
3 Methodik	15
3.1 Erstellung der Prototypen	15
3.2 Betrachtung der Metriken	15
4 Implementierung	17
4.1 Erstellung der Anfragen	17
4.2 Backend traditionelle Firewall	19

4.3 Backend KI-Firewall	22
4.4 Senden der Anfragen	25
5 Vergleichende Analyse	26
5.1 Ergebnis traditioneller Firewall	27
5.2 Ergebnis KI-Firewall	30
6 Diskussion	33
6.1 Interpretation der Ergebnisse	33
6.2 Limitationen der Studie	34
7 Schlussfolgerung und Ausblick	36
7.1 Zusammenfassung der wichtigsten Ergebnisse	36
7.2 Empfehlungen für die Praxis	36
7.3 Vorschläge für zukünftige Forschungen	37
Literaturverzeichnis	39
A Anhang	41
A.1 Code Firewall KI	41
A.2 Main Code Traditionelle Firewall	42
A.3 Code Path Traversal Traditionelle Firewall	43
A.4 Code SQL Injektion Traditionelle Firewall	44
A.5 Code XSS Injection Traditionelle Firewall	45
A.6 Code Command Injection Traditionelle Firewall	46
A.7 Code Logger Traditionelle Firewall	47

Abbildungsverzeichnis

1	Modell Ablauf überwachtes Lernen	9
2	CSV-Datei mit allen Anfragen	18
3	Aufteilung Anfragen nach Angriffsart und Anfragemethode	18
4	Einbindung Middleware in das Backend	19
5	regulärer Ausdruck zur Erkennung von SQL-Injektionen	19
6	Funktion zur Überprüfung einer SQL-Injektionen	20
7	Schleife zum iterieren über Anfrage-Parameter	21
8	Funktion zur Weiterleitung des Codes	22
9	Abfangen aller Anfragen in Flask	23
10	Basis Funktion KI-Backend	23
11	KI-Bewertung von Anfragen	25
12	Skript zum senden der Anfragen an den Server	26
13	Ergebnis traditionelle Firewall	27
14	Zuordnung von Angriffen traditionelle Firewall	28
15	Kategorisierung des Ergebnis	28
16	Zuordnung von Angriffen traditionelle Firewall	29
17	Durchlaufzeit traditionelle Firewall	29
18	Erkennungsgenauigkeit KI	30
19	Erkennungsgenauigkeit KI nach Angriffsart	31
20	Kategorisierung Ergebnis KI	31
21	Durchlaufzeit KI	32

Tabellenverzeichnis

1	Tabelle Angriffsarten und Anzahl	26
2	Ergebnis traditionelle Firewall	27

Abkürzungsverzeichnis

BSI Bundesamt für Sicherheit in der Informationstechnik

DDoS Distributed Denial of Service

KI Künstliche Intelligenz

LLM Large Language Model

LLMs Large Language Models

OSI Open System Interconnection Model

OWASP Open Worldwide Application Security Project

XSS Cross-Site-Scripting

1 Einleitung

1.1 Problemstellung und Motivation

"Für die große Mehrheit der Bürger*innen ist Digitalisierung ein fester Bestandteil des eigenen Lebens. 55 Prozent gehören zur Digitalen Mitte, 30 Prozent sogar zu den Digitalen Profis."(„D21-Digital-Index 2022/2023“, 2023). Auch in der Arbeitswelt ist die Digitalisierung nicht aufzuhalten. Dementsprechend muss nahezu jeder Mensch in der Lage sein mit Computern oder weiteren technischen Mitteln zu interagieren.

Dies bietet eine große Angriffsfläche für kriminelles Verhalten. SStraftaten im Bereich Cybercrime liegen in Deutschland weiter auf einem sehr hohen Niveau. Im vergangenen Jahr registrierte die Polizei 136.865 Fälle von Cybercrime. [...] nahm die Zahl jener Taten, die aus dem Ausland heraus begangen werden und in Deutschland einen Schaden verursachen, weiter zu, nämlich um 8 Prozent im Vergleich zum Vorjahr. [...] So erwarten rund zwei Drittel (63 Prozent) der befragten Unternehmen einen Cyberangriff in den kommenden 12 Monaten."(„BKAListenseiteFur2023“, 2023). Eine Technologie, welche mehr als alles andere Gegenwärtig polarisiert, ist das Thema Künstliche Intelligenz (KI). "KI ist der Trend, der im Jahr 2024 die IT-Branche am stärksten prägen wird, davon sind rund drei Viertel (73,6 Prozent) der IT-Entscheiderinnen und -Entscheider überzeugt."(„eco Branchenmonitor Internetwirtschaft 2023“, 2023).

Seit Ende 2022 sind durch das US-amerikanische Startup OpenAI Large Language Models (LLMs), die auf maschinellem Lernen und KI basieren, relevant geworden. Durch die kostenlose Nutzung von LLMs, wie beispielsweise ChatGPT, sind die Anwender in der Lage scheinbar echte Gespräche mit einem Computer zu führen (Altman, 2023). Diese Programme sind allerdings auch in der Lage Code und Malware zu generieren. Eine Studie zweier Wissenschaftler der ETH Zürich bewies nun, wie ChatGPT verwendet werden kann, dass Antivirenprogramme die Malware nicht mehr erkennen („KI und Cyberbedrohungen“, 2024).

Das dadurch implizierte Sicherheitsrisiko ist enorm. Daher planen laut einer Studie von PWC "[...] in den nächsten zwölf Monaten 75% der Befragten, GenAI-Tools für die Cyberabwehr einzusetzen."(PricewaterhouseCoopers, 2024). GenAI-Tools bezeichnet dabei ähnlich, wie LLMs Softwareanwendungen, die generative künstliche Intelligenz nutzen, um automatisch Inhalte, wie z.B. Texte, Bilder oder Code zu generieren und dadurch kreative oder produktive Aufgaben zu unterstützen.

Traditionelle Methoden zur Erkennung von Cyber-Security-Bedrohungen, wie signaturbasierte Systeme und regelbasierte Ansätze, stoßen zunehmend an ihre Grenzen, da sie oft nicht in der Lage sind, neue und raffinierte Angriffsmethoden effektiv zu identifizieren. Die dynamische Natur von Cyber-Bedrohungen erfordert flexible und anpassungsfähige Abwehrmechanismen. Hier bieten maschinelles Lernen und andere KI-Technologien signifikante Vorteile. Sie ermöglichen es, aus großen

Datenmengen Muster zu erkennen und Anomalien in Echtzeit zu identifizieren, die auf mögliche Sicherheitsverletzungen hindeuten könnten. Dies hat das Potenzial, sowohl die Erkennungsrate als auch die Reaktionszeit auf Bedrohungen erheblich zu verbessern (Jenis Nilkanth Welukar & Gagan Prashant Bajoria, 2021).

Die Motivation der Arbeit liegt in der wissenschaftlichen Neugier, die Wirksamkeit und Effizienz von KI-basierten Ansätzen im Vergleich zu traditionellen Methoden zu erforschen. Dabei treibt vor allem die Neugier herauszufinden, ob künstliche Intelligenz in ihrer Effektivität überbewertet ist. Besonders interessant ist die Untersuchung der Limitationen, die beide Lösungsansätze zurückhalten.

1.2 Zielsetzung der Arbeit

Ziel dieser Forschungsarbeit ist es, eine fundierte Bewertung der verschiedenen Ansätze zur Implementierung von Firewalls durchzuführen, um herauszufinden, welcher Ansatz sich als effektiver erweist. Im Mittelpunkt steht dabei der Vergleich zwischen einer Firewall, die auf traditionellen Filterregeln basiert, und einer Firewall, die mithilfe künstlicher Intelligenz Entscheidungen trifft.

Im Rahmen dieser Untersuchung wird insbesondere die Erkennungsgenauigkeit der beiden Firewall-Modelle analysiert. Darüber hinaus werden zentrale Gütekriterien wie Reliabilität, Verallgemeinbarkeit und Objektivität detailliert betrachtet und in die Beurteilung einbezogen, um eine fundierte und umfassende Entscheidung über den besseren Lösungsansatz treffen zu können. Ziel ist es, nicht nur die technischen Merkmale der Ansätze zu beleuchten, sondern auch deren Anwendbarkeit und Zuverlässigkeit in unterschiedlichen Szenarien kritisch zu hinterfragen und zu bewerten.

1.3 Aufbau der Arbeit

Die Struktur dieser Bachelorarbeit ist so aufgebaut, dass sie eine umfassende, systematische und nachvollziehbare Untersuchung der zentralen Forschungsfrage ermöglicht. Zu Beginn der Arbeit steht die Einleitung, in der die Problemstellung sowie die Motivation für die Untersuchung dargelegt werden. Hier wird auch die Zielsetzung der Arbeit definiert und ein Überblick über das methodische Vorgehen gegeben.

Im Anschluss folgt eine ausführliche Darstellung der grundlegenden Konzepte und Technologien, die für das Verständnis der weiteren Ausführungen unerlässlich sind. Dieser theoretische Teil dient dazu, das notwendige Fachwissen zu vermitteln und die Leserinnen und Leser in die Thematik einzuführen. Dabei werden sowohl klassische Firewall-Technologien als auch moderne Ansätze wie künstliche Intelligenz erläutert.

Der Methodik-Abschnitt der Arbeit beschreibt detailliert das wissenschaftliche Vorgehen, das zur Beantwortung der Forschungsfrage gewählt wurde. Hier werden die ausgewählten Methoden und ihre

Anwendung im Kontext der Untersuchung ausführlich erklärt. Dieser Abschnitt schafft die Grundlage für die nachfolgende praktische Umsetzung.

Im darauf folgenden Kapitel zur Implementierung wird detailliert beschrieben, wie der Code für die verschiedenen Firewalls erstellt und getestet wurde. Dabei werden die spezifischen Herausforderungen und Lösungsansätze, die im Verlauf der Entwicklung aufgetreten sind, transparent dargestellt. Dieser Abschnitt gibt einen tiefen Einblick in die technische Umsetzung der Forschungsarbeit.

Anschließend werden in der Auswertung die Ergebnisse der durchgeführten Tests und Experimente präsentiert. Jede Firewall wird hierbei gesondert betrachtet, und die erzielten Resultate werden anschaulich beschrieben und analysiert.

In der folgenden Diskussion werden die Ergebnisse der Auswertung kritisch hinterfragt und im Kontext der Forschungsfrage interpretiert. Hier wird insbesondere darauf eingegangen, wie die verschiedenen Technologien im Vergleich zueinander abschneiden und welche Vor- und Nachteile sie mit sich bringen.

Abschließend fasst die Schlussfolgerung die wesentlichen Erkenntnisse der Arbeit zusammen. Hier wird ein abschließendes Urteil über die untersuchten Technologien gefällt, das sowohl die praktischen Ergebnisse als auch die theoretischen Überlegungen berücksichtigt. Dieses Fazit dient dazu, die Relevanz der Ergebnisse für die Praxis zu unterstreichen und mögliche Implikationen für zukünftige Forschungen aufzuzeigen.

1.4 Formulierung der Forschungsfrage

"Künstliche Intelligenz (KI) ist der Trend, der im Jahr 2024 die IT-Branche am stärksten prägen wird, davon sind rund drei Viertel (73,6 Prozent) der IT-Entscheiderinnen und -Entscheider überzeugt." („eco Branchenmonitor Internetwirtschaft 2023“, 2023). Dabei werden häufig die Limitationen und Nachteile von KI missachtet oder gar nicht in Betracht gezogen. Daher lautet die zentrale Forschungsfrage dieser Bachelorarbeit: "Wie effektiv sind KI-basierte Ansätze im Vergleich zu traditionellen Methoden zur Erkennung von Cyber-Security-Bedrohungen? Um diese Frage umfassend zu beantworten, müssen mehrere Teilespekte betrachtet werden:

1. Erkennungsgenauigkeit: Wie genau identifizieren KI-basierte Systeme im Vergleich zu traditionellen Methoden verschiedene Arten von Cyber-Bedrohungen? Dabei soll untersucht werden, ob KI-Modelle insbesondere bei unbekannten oder neuartigen Bedrohungen überlegen sind.
2. Ressourceneffizienz: Welche Anforderungen stellen die verschiedenen Methoden an Ressourcen wie Rechenleistung und Speicher? Hier soll untersucht werden, ob die möglicherweise höhere Erkennungsrate von KI-Systemen durch einen signifikant höheren Ressourcenverbrauch erkauft wird und wie sich dies auf die Praktikabilität der Implementierung auswirkt.

3. Implementierung und Wartung: Wie unterscheiden sich die Aufwände für Implementierung und laufende Wartung zwischen KI-basierten und traditionellen Systemen? Dies umfasst sowohl die initiale Einrichtung als auch die Notwendigkeit regelmäßiger Updates und Anpassungen.

Diese Untersuchung zielt darauf ab, fundierte Empfehlungen für die Wahl der optimalen Cyber-Security-Strategien in verschiedenen Anwendungsszenarien zu geben. Dabei wird nicht nur die theoretische Wirksamkeit der Ansätze betrachtet, sondern auch deren praktische Anwendbarkeit.

2 Beschreibung des Themas

2.1 OSI-Modell

Netzwerke bilden das Rückgrat der modernen Informationstechnologie und ermöglichen die Kommunikation und den Datenaustausch zwischen verschiedenen Geräten und Systemen. Um die Komplexität und die vielfältigen Aspekte von Netzwerken verständlich zu machen, wurde das Open System Interconnection Model (OSI) entwickelt.

Das OSI unterteilt die Netzwerk-Architektur in sieben Schichten. Jede dieser Schichten hat eine dedizierte Funktion und erweitert die Funktion der Schicht direkt unter sich. Dadurch wird sichergestellt, dass die Aufgabenbereiche klar sind. Je nach Übersetzung sind die Bezeichnungen der einzelnen Schichten unterschiedlich. Im Rahmen dieser Arbeit werden die Schichten wie folgt bezeichnet:

- Bitübertragungsschicht
- Sicherungsschicht
- Vermittlungsschicht
- Transportschicht
- Sitzungsschicht
- Darstellungsschicht
- Anwendungsschicht

Schicht 1: Bitübertragungsschicht Im Layer 1 werden die elektrischen und physikalischen Spezifikationen der Datenübertragung definiert. Darunter fallen die Beziehung zwischen einem Gerät und dem physischen Übertragungsmedium. Des Weiteren definiert dieses Layer das Protokoll zur Herstellung und Beendigung der Verbindung zwischen den verbundenen Knoten.

Schicht 2: Sicherungsschicht Die Sicherungsschicht sorgt für die zuverlässige Übertragung von Daten zwischen den Knoten. Diese Schicht erweitert den rohen und unzuverlässigen Bitübertragungsdienst, der von Schicht eins bereitgestellt wird. Um die zuverlässige und sichere Übertragung zu garantieren führt die Sicherungsschicht Fehlererkennung und -kontrollen durch. Dabei kann die Sicherungsschicht nur die Relation zwischen zwei direkt miteinander verbundenen Systemen absichern.

Schicht 3: Vermittlungsschicht Während sich die Sicherungsschicht damit beschäftigt, wie die physikalische Schicht zur Datenübertragung genutzt wird, befasst sich die Netzwerkschicht mit der Organisation dieser Daten zur Übertragung und Zusammenführung. Dabei bietet die Netzwerkschicht

die Funktionalität und die prozeduralen Mittel um variabel lange Datensequenzen von einem Knoten zu einem anderen Knoten im Netzwerk zu übertragen.

Schicht 4: Transportschicht Durch die Transportschicht wird die sichere Ende-zu-Ende Kommunikation zwischen den Prozessen der verschiedenen Maschinen sichergestellt. Zwar ist der Service, den die Transportschicht bietet ähnlich zu dem der Sicherungsschicht, jedoch gibt es signifikante Unterschiede. Die größte Unterscheidung ist, dass bei der Transportschicht der Anwender direkt mit der Schicht interagiert, daher sollten die Protokolle hier mehr Richtung Anwender entwickelt werden.

Schicht 5: Sitzungsschicht Um auch längerfristige Kommunikation über das Netzwerk abilden zu können, gibt es die Sitzungsschicht. Diese ist verantwortlich für das Errichten der Sitzung, den Austausch der Daten und das Beenden der Sitzung.

Schicht 6: Darstellungsschicht Die Darstellungsschicht kann als Übersetzer für die Anwendungsschicht angesehen werden. Sie ist dafür zuständig, die Daten vorzubereiten. Darunter fällt zum Beispiel die Übersetzung von Zeichen oder die Änderung an der Bitreihenfolge.

Schicht 7: Anwendungsschicht Die letzte Schicht und auch die Schicht mit der der Anwender interagiert ist die Anwendungsschicht. Hierüber laufen viele Protokolle, wie z.B. DNS, HTTP und SSH.

Kumar et al., 2014

2.2 Firewalls

"Die durch den Digitalverband Bitkom errechneten Cybercrime-Schäden in Deutschland beliefen sich laut Wirtschaftsschutzbericht 2022 auf 203 Mrd. Euro und sind rund doppelt so hoch wie noch im Jahr 2019." („BKAListenseiteFur2023“, 2023). Mit der steigenden Anzahl von Cyberattacken und Schwere der Auswirkungen beschäftigen sich immer mehr Firmen und Personen mit der Absicherung ihres Netzwerks.

Auf der Suche nach Schwachstellen sind Firewalls und deren Probleme immer mehr in den Fokus von Sicherheitsforschern gekommen. Durch eine falsch konfigurierte Firewall kann ein komplettes Netzwerk infiltriert und zerstört werden.

Eine Firewall ist ein Router, ein Computer oder ein anderes technisches Gerät, welches den Zugang zu einer Website, einer App oder einem Netzwerk filtert. Dabei inspizieren Firewalls die Netzwerkpakete und setzen somit an der OSI-Schicht 3 und 4, also der Netzwerk- und Transportschicht an. Es werden Informationen aus dem IP-Header sowie dem Payload der jeweiligen Datenpakete betrachtet.

Ein Paketfilter wird über verschiedene Filterregeln konfiguriert. Diese sind in sogenannten Listen abgespeichert. Die Regeln bestehen aus einer Bedingung und einer Folgerung. Man unterscheidet zwischen sogenannten Blacklisting und Whitelisting. Die Firewall gleicht eingehende Datenpakete mit

allen Filterregeln ab. Beim Ansatz des Blacklisting werden alle Datenpakete abgelehnt, welche der Filterregel entsprechen. Whitelisting bedeutet, dass nur Verbindungen und Datenpakete zugelassen werden, die einer Filterregeln entsprechen. Dadurch lassen sich Systeme besser abschirmen, jedoch ist dieser Ansatz deutlich wartungsintensiver.

Bei Paketfilter kann zwischen zustandslosen und zustandsbehafteten Filtern unterschieden werden. Ersteres ist deutlich simpler. Dabei wird jedes Datenpaket als einzelne Instanz betrachtet und Entscheidung lediglich anhand des Pakets und den Filterregeln getroffen. Bei zustandsbehafteten Filtern ist der Entscheidungsprozess deutlich umfangreicher. Hier werden Informationen über die einzelnen Pakete abgespeichert und die Filterentscheidung wird ebenfalls anhand zuvor gesendeter Datenpakete getroffen. So kann bei zustandsbehafteten Paketfiltern z.B. erkannt werden, ob es sich bei dem Paket um eine Antwort handelt und dementsprechend akzeptiert werden.

Wie bereits erwähnt, fungieren Firewalls als Barriere zwischen internen Netzwerken oder Webseiten / Apps und externen Bedrohungen, indem sie den Datenverkehr kontrollieren und unerwünschte Verbindungen blockieren.

Traditionelle Firewalls lassen sich in mehrere Kategorien unterteilen, die jeweils unterschiedliche Methoden der Bedrohungserkennung und -abwehr nutzen.

Paketfilter-Firewalls untersuchen einzelne Datenpakete und entscheiden auf vordefinierte Regeln, ob das Paket durchgelassen oder blockiert wird. Diese arbeiten hauptsächlich auf der Netzwerkschicht (Layer 3) und der Transportschicht (Layer 4).

Stateful Inspection Firewalls also zustandsbehaftete, sind als dynamische Paketfilter bekannt und agieren ebenfalls auf der Netzwerkschicht (Layer 3) und der Transportschicht (Layer 4).

2.3 Künstliche Intelligenz - Machine Learning

Datenverarbeitung und maschinelles Lernen sind zwei Bereiche, die eng miteinander verbunden sind. Die traditionelle Datenverarbeitung in der Informatik bildet das Rückgrat der modernen Informations-technologie, indem sie die Grundlage für Software und Hardwaresysteme schafft, die die Verarbeitung und Speicherung von Daten ermöglichen.

Maschinelles Lernen ist ein Teilbereich der künstlichen Intelligenz, der sich auf die Entwicklung von Algorithmen und Modellen konzentriert, die aus Daten lernen und eigenständig Vorhersagen oder Entscheidungen treffen können, ohne dass diese explizit programmiert werden müssen (Li et al., 2023).

Hauptsächlich wird versucht, dass der Computer das menschliche Gehirn bestmöglich nachahmt. Der Computer soll also in der Lage sein anhand von Algorithmen selbstständig zu lernen. Lernen bezieht sich dabei auf das Erkennen von statistischen Regelmäßigkeiten und anderen Muster in den Daten.

Der Algorithmus muss in der Lage sein multidimensionale Daten zu verarbeiten und die Genauigkeit und Effizienz der Verarbeitung und der Ausgabe zu optimieren. Abhängig vom gewünschten Ergebnis werden die maschinellen Lernalgorithmen in folgende Gruppen unterteilt:

- Überwachtes Lernen (Supervised Learning) - viele verschiedene Algorithmen erzeugen eine Funktion, welche die Eingaben auf die gewünschte Ausgaben abbildet. Eine gängiger Anwendungsbereich im überwachten Lernen ist das Klassifizierungsproblem. Hierbei lernt der Algorithmus das Verhalten seiner Funktion so anzunähern, indem es sich verschiedene Eingabe- und Ausgabebeispiele ansieht.
- Unüberwachtes Lernen (Unsupervised Learning) - Im Gegensatz zum überwachten Lernen wird das Modell mit unbeschrifteten Daten trainiert. Es gibt also keine Labels, das Modell muss selbstständig Strukturen, Muster oder Zusammenhänge in den Daten erkennen. Das Modell erkennt in den meisten Fällen Cluster oder Anomalien. Dadurch ist die Ergebnisbewertung oftmals schwieriger als beim überwachten Lernen.
- Bestärktes Lernen (Reinforcement Learning) - Das Modell lernt durch Interaktionen mit der Umgebung. Es trifft eine Entscheidung und sollte diese richtig sein wird das Modell belohnt bei einer falschen Entscheidung bestraft. Das Modell passt daraufhin seine Strategie dementsprechend an, um die Belohnung zu maximieren.
- Semi-überwachtes Kernen (Semi-supervised Learning) - Bei dieser Methode wird eine Kombination aus einem kleinen Teil beschrifteter Daten und einer Menge unbeschrifteter Daten zum trainieren des Modell verwendet.
- Transfer Learning - Das Modell wird für eine Aufgabe trainiert und für eine andere Aufgabe verwendet. Dies ist besonders nützlich, wenn nicht genügend Daten für die neue Aufgabe vorhanden sind.
- Aktives Lernen - Hier wird das Modell aktiv in den Datenbeschriftungsprozess einbezogen. Es identifiziert dabei die Datenpunkte, von denen es am meisten lernen kann und fordert diese spezifischen Labels an.

Grundsätzlich wird maschinelles Lernen also in überwachtes und nicht überwachtes Lernen unterteilt. Im Rahmen des überwachten Lernens ist es des Weiteren hilfreich zwischen Klassifikationsmodellen und Regressionsmodellen zu unterscheiden.

Klassifikationsmodelle sind Algorithmen, die darauf abzielen, Eingabedaten einer von mehreren vordefinierten Kategorien oder Klassen zuzuordnen. Diese Modelle werden verwendet, wenn die Zielvariable diskrete Werte annimmt, also eine endliche Anzahl von Kategorien hat. Ein Beispiel dafür ist

ein Modell, dass E-Mails entweder als Spam oder "Nicht-Spam" klassifiziert. Typische Algorithmen für diese Modelle sind:

- Logistische Regression - wird für binäre Klassifizierungsprobleme verwendet.
- Entscheidungsbäume - sind baumartige Modelle, welche die Entscheidungen auf Basis von Attributen treffen.
- Random Forest - ein Konglomerat von Entscheidungsbäumen, welche Zusammenarbeiten, um die Genauigkeit zu erhöhen.
- Support Vector Machines (SVM) - ein Modell, welches Klassen durch das Finden einer optimalen Trennlinie (Hyperplane) trennt.
- Neuronale Netze - besonders gut geeignet für komplexe Klassifizierungsproblemen, wie z.B. Bilderkennung.

Überwachtes Lernen ist die am häufigsten verwendete Technik bei Klassifizierungsproblemen und ein exemplarischer Ablauf eines solchen Modells sieht wie folgt aus:

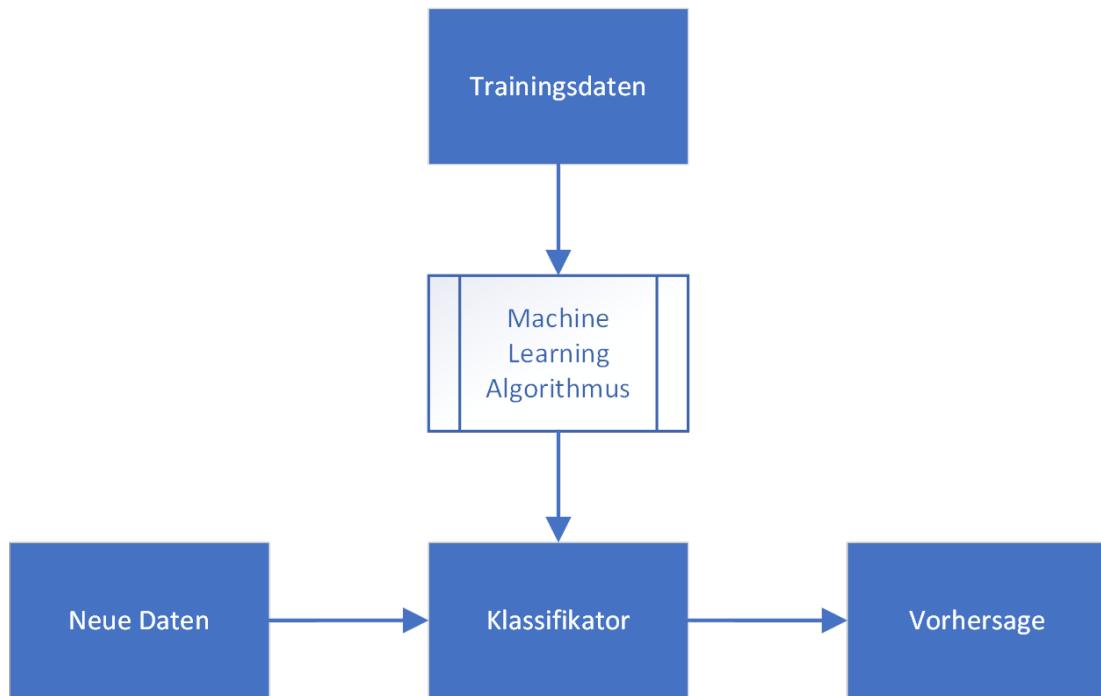


Abbildung 1: Modell Ablauf überwachtes Lernen

Der Lernprozess in einem einfachen maschinellen Lernmodell ist in zwei Schritte unterteilt: Training und Testen. Im Trainingsprozess werden die Stichproben aus den Trainingsdaten als Eingabe verwendet, wobei der Lernalgorithmus oder "Learner" die Merkmale lernt und das Lernmodell aufbaut [4]. Im Testprozess verwendet das Lernmodell die Ausführungsumgebung, um Vorhersagen für die Test-

oder Produktionsdaten zu treffen. Die markierten Daten sind das Ergebnis des Lernmodells, das die endgültige Vorhersage oder klassifizierte Daten liefert.

Regressionsmodelle werden verwendet, um kontinuierliche, numerische Werte vorherzusagen. Sie kommen zum Einsatz, wenn die Zielvariable ein kontinuierlicher Wert ist, also nicht diskret oder kategorisch, sondern fließend (Nasteski, 2017).

2.3.1 Künstliche Intelligenz in einer Firewall

Die Integration von künstlicher Intelligenz in Firewalls ermöglicht eine dynamische und proaktive Verteidigung gegen Cyberbedrohungen. Durch die Möglichkeit große Datenmengen zu verarbeiten bietet sich künstliche Intelligenz wie prädestiniert dafür an Bedrohungen zu identifizieren und sich an neue Angriffsmuster anzupassen.

Die Implementierung von KI in Firewalls erfordert sorgfältige Überlegungen bezüglich der Wahl der richtigen Lernmethoden und Algorithmen. Für diesen Anwendungsfall kommen besonders das überwachte Lernen und das unüberwachte Lernen in Frage.

Der Ansatz des überwachten Lernens verspricht durch das Training mit klassifizierten Beispielen eine hohe Genauigkeit bei der Erkennung bekannter Bedrohungen. Jedoch ist das Modell auf die Aktualität und Qualität der Trainingsdaten angewiesen. Es kann Schwierigkeiten haben, unbekannte oder neue Bedrohungen zu erkennen, wenn diese im Trainingsdatensatz nicht vorhanden sind.

Beim unüberwachten Lernen werden wie zuvor aufgegriffen keine klassifizierten Daten benötigt. Das Modell identifiziert Muster und Anomalien im Datenverkehr, was es besonders nützlich macht um Zero-Day-Angriffe, also Angriffe welche zum ersten mal auftreten, oder unbekannte Bedrohungen zu erkennen. Dadurch ist das Modell allerdings fehleranfälliger und es kann zu einer höheren Rate an Fehlalarmen kommen, im Vergleich zum überwachten Lernen.

2.4 Definition und Überblick über Cyber-Security-Bedrohungen

In der modernen IT-Welt gibt es viele Arten von Cyber-Bedrohungen. Das Bundesamt für Sicherheit in der Informationstechnik (BSI) veröffentlicht jedes Jahr einen Überblick über die Lage der IT-Sicherheit in Deutschland. Darin werden die größten Bedrohungen aufgelistet und erklärt. In dem Bericht von 2023 wird die Bedrohungslage als angespannt bis kritisch eingeordnet. Dabei bleibt laut BSI Ransomware die Hauptbedrohung für IT-Systeme. Bei einem Ransomware-Angriff versuchen Angreifer alle oder wichtige Informationen auf einem Computer zu verschlüsseln und bieten die Entschlüsselung der Dateien nur gegen eine Zahlung von Lösegeld an.

Durch den russischen Angriffskrieg gegen die Ukraine haben laut BSI allerdings auch die Distributed Denial of Service (DDoS)-Angriffe zugenommen. Ein solcher Angriff zielt auf die Verfügbarkeit von

Internetdiensten ab. Häufig sind Webseiten Ziel solcher Angriffe. Die zugehörigen Webserver werden dabei so mit Anfragen überflutet, dass die Webseiten nicht mehr erreichbar sind. Wird ein solcher Angriff mittels mehrerer Systeme parallel ausgeführt, spricht man von einem DDoS-Angriff. Ziel kann es dabei sein Lösegeld zu fordern oder die Dienste von unter anderem systemkritischen Anbietern direkt zu schaden, indem diese nicht mehr erreichbar sind. Dadurch können finanziellen Schäden bei den betroffenen Betreibern entstehen, sowie Image Schäden entstehen.

Eine weitere Institution zur Veröffentlichung von Cyber-Gefährdungen ist Open Worldwide Application Security Project (OWASP). Die Non-Profit-Organisation hat das Ziel, die Sicherheit von Anwendungen, Diensten und Software im Allgemeinen zu verbessern. OWASP veröffentlicht in regelmäßigen Abständen eine Top-Ten mit den größten Bedrohungen für IT-Systeme.

Derzeit schätzt OWASP die Sicherheitsrisiken wie folgt in absteigender Reihenfolge ein:

- Unzureichende Zugriffskontrollen - können dazu führen, dass Benutzer unberechtigte Aktionen ausführen oder auf Daten zugreifen können, die sie nicht sehen sollten.
- Kryptografische Fehler - Unzureichende oder unsichere Implementierung von kryptografischen Verfahren, wie schwache Verschlüsselung, kann die Vertraulichkeit und Integrität von Daten gefährden.
- Injektion - treten auf, wenn ungeprüfte Eingaben in einen Interpreter gesendet werden, wodurch Angreifer unerwünschte Befehle ausführen können.
- Unsichere Designentscheidungen - Unsichere Einstellungen oder Konfigurationen der Anwendungen oder Server, wie das Deaktivieren wichtiger Sicherheitsfeatures oder die Verwendung unsicherer Standards.
- Falsche Sicherheitskonfiguration - Unsichere Einstellungen oder Konfigurationen der Anwendungen oder Server, wie das Deaktivieren wichtiger Sicherheitsfeatures oder die Verwendung unsicherer Standards.
- Verletzung der Authentizität - Probleme in der Authentifizierungs- und Session-Verwaltung, wie schwache Passwortrichtlinien oder schlecht geschützte Session-IDs.
- Fehlerkonfiguration von Sicherheitsmaßnahmen - Unsachgemäße oder unzureichende Implementierung von Sicherheitsmechanismen, die die Anwendung gegen Angriffe schützen sollen.
- Fehlerhafte Software und Datenintegrität - ermöglicht es den Angreifern Manipulationen oder Schadsoftware einzuschleusen.
- Unsichere Komponenten - die Verwendungen von unsicheren Komponenten, sowie veralteten Bibliotheken, die bekannte Schwachstellen aufweisen.

- Unsichere Protokollierungs- und Überwachungsmechanismen - Mangelhafte Protokollierung und Überwachung von sicherheitsrelevanten Ereignissen, die es erschweren, Sicherheitsvorfälle zu erkennen oder darauf zu reagieren.

2.5 Angriffsarten

SQL-Injektionen versuchen Informationen aus einer SQL-Datenbank, welche im Hintergrund agiert zu erhalten, indem die spezifische Abfragesprache in die Anfrage inkludiert wird. In den meisten Fällen lässt sich eine SQL-Injektion verhindern, indem die Eingabe des Anwenders nochmal beim Server in Anführungszeichen gepackt wird. Dabei spricht man davon die Anfrage zu „escapen“.

Bei Cross-Site-Scripting (XSS)-Angriffen handelt es sich um eine Sicherheitslücke in Webanwendungen, bei der es Angreifern ermöglicht wird, schädlichen Code in eine Website einzuschleusen. Dieser Code wird dann im Browser des Anwenders ausgeführt, welcher die komprimierte Seite besuchen. Es gibt drei Hauptkategorien von XSS-Angriffen:

- Reflektiertes XSS - tritt auf, wenn ein Angreifer schädlichen Code an einen Webserver sendet, der diesen dann in der Antwort unverändert an den Benutzer zurückgibt. D.h. der Code wird in der URL oder einem anderen Anfrage-Parameter eingebettet und direkt im Browser des Anwenders ausgeführt.
- Gespeichertes XSS - tritt auf, wenn der schädliche Code dauerhaft auf dem Server gespeichert wird, also z.B. in einer Datenbank. Dieser Code wird mit jedem Aufruf der betroffenen Webseite ausgeführt und ist daher besonders gefährlich.
- DOM-basiertes XSS - hierbei liegt die Schwachstelle nicht auf dem Server, sondern im Client. Der schädliche Code wird durch Manipulation des Document Object Models (DOM) des Website ausgeführt.

Im Rahmen der Forschungsarbeit wird sich auf reflektierte- und gespeicherte-XSS-Angriffe konzentriert.

Command Injection Angriffe versuchen bösartigen Code in die Anwendung einzuschleusen, welcher daraufhin auf dem Server oder Computer ausgeführt wird. Der Code konzentriert sich dabei auf Befehle, welche durch das zugrundeliegende Betriebssystem ausgeführt werden. Da die meisten Server auf Linux laufen, richten sich diese Angriffe meist gezielt auf Linux-Systeme ab. Dadurch kann der Angreifer Zugriff auf vertrauliche Daten erhalten oder im schlimmsten Fall die Kontrolle über den Server erlangen.

Authentication ByPass-Angriffe versuchen sich als ein legitimer Anwender auszugeben oder höhere Rechte als geplant zu erhalten, basieren auf unterschiedlichen Ansätzen.

2.6 Klassifizierung von Firewall Ergebnissen

Im Rahmen der Forschungsarbeit werden die Firewalls und deren Ergebnisse auf mehrere Metriken untersucht. Als erstes wird die Erkennungsgenauigkeit der beiden Firewalls verglichen. Im Rahmen der Cyber-Security werden normalerweise vier verschiedene Szenarien betrachtet:

- True Positive
- True Negative
- False Positive
- False Negative

Sollte der Proxy-Server einen Angriff als Angriff identifizieren und abblocken, so würde man in diesem Fall die Entscheidung als true Positive bezeichnen. Ein false Positive ist, wenn die Firewall eine legitime Anfrage als Angriff eingestuft und diese blockiert. Beim true Negative wird eine legitime Anfrage als diese eingestuft und weitergeleitet. Die gefährlichste dieser Ergebnisse fallen in die Kategorie false Negative. Bei diesen schätzt die Firewall die Anfrage als legitim ein und leitet diese an den Server weiter. Bei der Anfrage handelt es sich jedoch um einen Angriff. Die Ergebnisse der jeweiligen Firewall werden nach true Positives, false Positives und false Negatives unterteilt und veranschaulicht werden.

2.7 Aktueller Stand der Forschung

Die Forschung im Bereich der künstlichen Intelligenz und Firewalls hat in den letzten Jahren erhebliche Fortschritte gemacht. Mit der Explosion an Popularität von Large Language Model (LLM)s wird künstliche Intelligenz immer mehr als die Zukunft in der Technikbranche betrachtet. Es wird vermehrt nach Möglichkeiten gesucht um Hardware herzustellen, welche mit den speziellen Anforderungen von künstlicher Intelligenz umgehen können, wie im Artikel aus dem Jahre 2022 beschreiben (Saha et al., 2022). Mit den technologischen Verbesserungen in Bereich der Datenverarbeitung wird es immer leichter möglich viele Daten auf einmal zu inspizieren, ohne dabei viel an Leistung einzubüßen. Daher wird es für viele Unternehmen immer attraktiver möglichst viele Daten und Zustände in den Entscheidungsprozess einer Firewall einzubeziehen. Diese neumodischen Firewalls wurden 2010 von "Palo Alto Networks als Next-Generation Firewalls (NGFW) bezeichnet. Dabei bezieht sich der Begriff NGFW auf Netzwerk-Technik Lösungen, welche die Funktionen einer traditionellen Firewall mit zusätzlichen Funktionen erweitert. Der Fokus dabei lag zuerst auf sogenannter "Deep Package Inspection"(DPI). Damit ist gemeint, dass die Firewall in der Lage sein soll Netzwerktechnologien aus "tieferen SSchichten des OSI-Modells zu verarbeiten. Mit Unterstützung von DPI sind Next-Generation

Firewalls (NGFWs) in der Lage, fortschrittliche Sicherheitsfunktionen bereitzustellen. Dazu gehören Intrusion Prevention Systeme (IPS), die Kontrolle des Netzwerkverkehrs basierend auf Netzwerk-anwendungen sowie eine erhöhte Sichtbarkeit und Sicherheit durch die Entschlüsselung von TLS-Verkehr. Der Begriff „Netzwerkanwendung“ umfasst dabei nicht nur verschiedene Webdienste wie Google oder Facebook, sondern auch die Identifizierung von Protokollen auf der Anwendungsschicht. NGFWs können zudem Sicherheitsrichtlinien basierend auf Benutzer- und Gruppen-IDs implementieren. Neben der Möglichkeit, den Datenverkehr auf Basis von Transport- und Anwendungsschicht-eigenschaften zu steuern, bieten NGFWs auch grundlegende Funktionen wie das Routing und NATting von Datenverkehr. (Heino et al., 2022) Der Funktionsumfang von Firewalls wächst also stetig. Und es wird stetig daran gearbeitet die Technologien zur Erkennung von Angriffen und Gefahren zu verbessern.

3 Methodik

3.1 Erstellung der Prototypen

Um den Versuchsaufbau so realitätsnah wie möglich zu gestalten, wird als Server der Juice-Shop von OWASP verwendet. Der Juice-Shop ist ein fiktiver Online-Shop, der speziell entwickelt wurde, damit angehende Cybersecurity-Spezialisten verschiedene Schwachstellen entdecken und testen können. Durch die Verwendung des Juice-Shop als Server können reale Routen für die Angriffe simuliert werden. Dies veranschaulicht die Angriffe besser und macht sie leichter nachvollziehbar, ohne jedoch die Ergebnisse im Vergleich zu einem Mock-Server zu verfälschen.

Vor dem Juice-Shop wird ein Proxy-Server zwischengeschaltet. Ein Proxy-Server fungiert als Vermittler, der Anfragen von Clients entgegennimmt und diese an den eigentlichen Server weiterleitet. Der Proxy-Server wartet anschließend auf die Antwort des Servers und leitet diese wiederum an den Client weiter. Diese technische Implementierung hat den Vorteil, dass Serveranfragen abgefangen werden können, bevor sie den Server erreichen.

Im Proxy-Server wird eine Firewall in Form einer sogenannten Middleware implementiert. Eine Middleware besteht aus einer oder mehreren Funktionen, die ausgeführt werden, bevor die Anfrage an den Server weitergeleitet wird. In diesem Fall wird die Firewall als Middleware integriert. Erhält der Proxy-Server eine Anfrage, so überprüft er mit Hilfe der Middleware bzw. der Firewall, ob es sich um eine legitime Anfrage handelt. Alle legitimen Anfragen werden daraufhin an den Juice-Shop weitergeleitet. Anfragen, die als Angriffe identifiziert werden, werden hingegen blockiert, und der Client erhält eine „Forbidden Request“-Statusmeldung.

Jede Anfrage wird eine ID als Header enthalten. Der Proxy-Server protokolliert jede Anfrage zusammen mit der ID, der URL und dem Ergebnis in einer Log-Datei. Diese Log-Datei dient nach Abschluss der Tests als Grundlage für die Auswertung.

3.2 Betrachtung der Metriken

In der Versuchsauswertung wird daraufhin eine umfassende Analyse der Erkennungsgenauigkeit der beiden untersuchten Firewall-Ansätze durchgeführt. Dabei wird detailliert betrachtet, wie effektiv die Firewalls sowohl mit bekannten als auch mit unbekannten Bedrohungen umgehen. Insbesondere wird der Fokus auf die Fähigkeit der Firewalls gelegt, schädliche Aktivitäten korrekt zu identifizieren. Das Ergebnis wird in die zuvor genannten Metriken zur Einstufung einer Firewall-Meldung kategorisiert.

Ein weiterer wichtiger Aspekt dieser Untersuchung ist die Analyse der Ressourceneffizienz der jeweiligen Firewall-Implementierungen. Hierbei werden verschiedene Systemressourcen wie CPU-, GPU- und RAM-Auslastung während der Ausführung der beiden Server intensiv betrachtet. Dies gibt Auf-

schluss darüber, wie ressourcenintensiv die Firewalls im Betrieb sind und wie sie sich auf die Leistungsfähigkeit des Gesamtsystems auswirken. Zudem wird der benötigte Speicherplatz sowohl für den Code als auch für die Modelle, die zur Schulung der künstlichen Intelligenz verwendet werden, untersucht. Dies ist besonders relevant, da eine effiziente Ressourcennutzung in Form einer kurzen Laufzeit eine wesentliche Voraussetzung für den Einsatz in realen Szenarien darstellt.

Ein weiterer zentraler Punkt der Analyse ist die Untersuchung der Antwortzeiten der Firewalls. Die Reaktionsgeschwindigkeit ist ein kritischer Faktor, insbesondere in realen Anwendungsszenarien, in denen schnelle Reaktionen auf Bedrohungen erforderlich sind, um Schäden zu minimieren.

Abschließend werden die Kriterien Objektivität, Verallgemeinerbarkeit und Reliabilität der Ergebnisse untersucht. Dabei wird analysiert, inwieweit die Ergebnisse unabhängig von äußeren Einflüssen sind (Objektivität), ob sie auf andere Szenarien übertragen werden können (Verallgemeinbarkeit) und ob die Ergebnisse bei wiederholter Anwendung konsistent bleiben (Reliabilität). Diese Betrachtungen sind entscheidend, um die wissenschaftliche Robustheit und den praktischen Nutzen der beiden Firewall-Ansätze fundiert bewerten zu können.

4 Implementierung

4.1 Erstellung der Anfragen

Vor der Implementierung der Server und der Firewall müssen die Anfragen definiert werden. Wichtig ist, dass jede Anfrage eine eindeutige ID erhält, über die nachträglich eine Zuordnung möglich ist, um die Art der Anfrage zu bestimmen. Die einfachste Umsetzung besteht darin, jeder Anfrage im Header ein zusätzliches Feld hinzuzufügen. Dieses Header-Feld trägt den Namen **id**, und der tatsächliche Wert der ID wird dort eingetragen. Die Server können dieses Feld aus der Anfrage extrahieren und in die Log-Datei schreiben. Um die Testmöglichkeiten einzuschränken, werden nur GET- und POST-Anfragen untersucht. Über diese beiden Anfragetypen kann theoretisch jede gewünschte Funktionalität einer Website abgebildet werden. Anfragetypen wie UPDATE oder DELETE können durch eine POST-Anfrage abgebildet werden, und die Einbeziehung dieser Typen in die Testreihe würde einen erhöhten Aufwand bei gleichbleibendem Ergebnis bedeuten.

Die Anfragen an den Server werden mittels eines Python-Skripts ausgeführt, das eine CSV-Datei ausliest. Jede Zeile in der CSV-Datei entspricht einer Anfrage. Die Datei enthält vier Spalten: eine ID, damit jede Anfrage später identifiziert werden kann, die Methode, damit das Skript weiß, welche Art von Anfrage gesendet werden soll (POST oder GET), den Payload, also den Inhalt der Anfrage, die an den Server geschickt werden soll, und die Angriffskategorie. Die Datei enthält ebenfalls legitime Serveranfragen, die in der Spalte Angriffskategorie mit „None“ gekennzeichnet sind. Zur Erstellung der legitimen Anfragen wurde ein Programm namens Burp Suite verwendet. Burp Suite ist ein Programm der Firma PortSwigger und wird hauptsächlich im Bereich der Cyber-Sicherheit eingesetzt. Unter anderem bietet Burp Suite die Möglichkeit, als Proxy für einen Browser zu agieren. Es zeichnet den gesamten Datenverkehr auf und macht ihn zugänglich. Um möglichst realitätsnahe legitime Anfragen zu sammeln, wurde der Juice-Shop verwendet, bei dem jede Funktionalität einmal aufgerufen und ausgeführt wurde. Diese Historie wurde dann aus Burp Suite exportiert und per Skript in ein passendes Format umgewandelt.

Die Angriffscontent wurden mithilfe von PayloadAllTheThings erstellt („Swisskyrepo/PayloadsAllTheThings: A List of Useful Payloads and Bypass for Web Application Security and Pentest/CTF“, 2024). Dabei handelt es sich um ein Open-Source-GitHub-Repository, das sich darauf spezialisiert hat, den Inhalt verschiedenster Web-Angriffe aufzulisten und zu kategorisieren. Wie bereits erwähnt, schätzt OWASP Injektionen als die dritthöchste Gefährdung im Internet ein. Besonders Injektionen lassen sich durch eine Firewall gut überprüfen und abfangen, da diese die Anfragedaten untersucht. Daher wird sich im Rahmen dieser Forschungsarbeit auf Injektionen konzentriert. Im Detail sollen die beiden Firewalls in der Lage sein, SQL-Injektionen, XSS-Angriffe, Command-Execution-Angriffe, Authentication-Bypass-Angriffe und Path-Traversal-Angriffe zu erkennen und zu verhindern.

Die resultierende Liste der Anfragen sieht wie folgt aus:

id	url	method	payload	attack category
0	http://localhost:5050/rest/products/search?q=	GET	==	Auth Bypass
1	http://localhost:5050/rest/products/search?q=	GET	=	Auth Bypass
2	http://localhost:5050/rest/products/search?q=	GET	'	Auth Bypass
3	http://localhost:5050/rest/products/search?q=	GET	' --	Auth Bypass
4	http://localhost:5050/rest/products/search?q=	GET	' #	Auth Bypass
5	http://localhost:5050/rest/products/search?q=	GET	' -	Auth Bypass

Abbildung 2: CSV-Datei mit allen Anfragen

Sie enthält insgesamt 3071 Anfragen, welche wie folgt aufgeteilt sind:

method	GET	POST	PUT	Gesamtsumme
attack category	id (Eindeutig)			
Auth Bypass	121	121		242
Command Execution	448	448		896
Directory Traversal	140	140		280
None	283	26	2	311
SQL Injection	68	68		136
Union Select	424	424		848
XSS	179	179		358
Gesamtsumme	1663	1406	2	3071

Abbildung 3: Aufteilung Anfragen nach Angriffsart und Anfragemethode

4.2 Backend traditionelle Firewall

Für die Implementierung der traditionellen Firewall wird als Backend, also als Server, Express.js verwendet. Dieser Server nimmt alle Anfragen entgegen und hat Funktionen zur Angriffserkennung als Middleware implementiert. Wenn keine der Funktionen einen Fehler ausgibt, wird die Anfrage an den Juice-Shop-Server weitergeleitet.

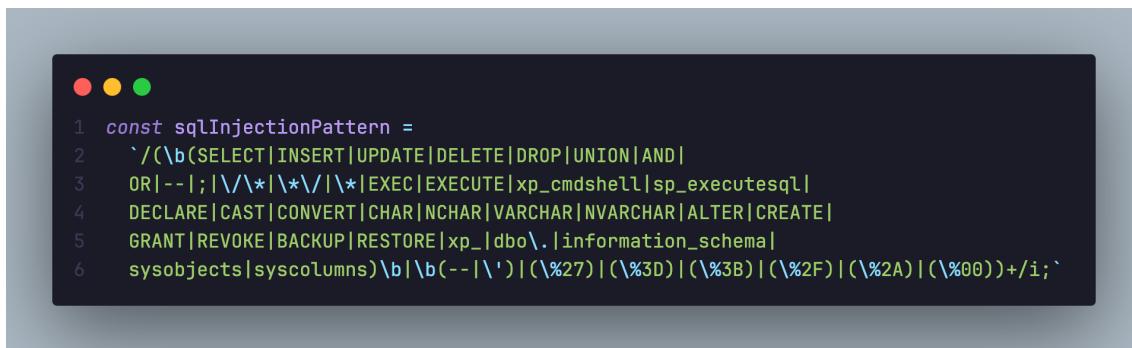


```
● ● ●
1 var app = express();
2
3 app.use(detectSQLInjection);
4 app.use(detectCommandInjection);
5 app.use(detectPathTraversal);
6 app.use(detectXSS);
```

Abbildung 4: Einbindung Middleware in das Backend

Der Aufbau der jeweiligen Methoden zur Angriffserkennung ist grundsätzlich gleich. Im Folgenden wird am Beispiel der Funktion zur Erkennung einer SQL-Injektion der Aufbau und Ablauf des Moduls erklärt. Dieses Vorgehen lässt sich auf die weiteren Module übertragen. Jedes Modul besitzt eine konstante Variable, die einen regulären Ausdruck enthält. Dieser reguläre Ausdruck umfasst alle wichtigen Wörter und Zeichenkombinationen für den jeweiligen Angriff.

Beispielhaft ist der reguläre Ausdruck zur Erkennung einer SQL-Injektion wie folgt aufgebaut.



```
● ● ●
1 const sqlInjectionPattern =
2 `^/(b(SELECT|INSERT|UPDATE|DELETE|DROP|UNION|AND|
3 OR|--|;|\\/*|/*|EXEC|EXECUTE|xp_cmdshell|sp_executesql|
4 DECLARE|CAST|CONVERT|CHAR|NCHAR|VARCHAR|NVARCHAR|ALTER|CREATE|
5 GRANT|REVOKE|BACKUP|RESTORE|xp_|dbo\.|information_schema|
6 sysobjects|syscolumns)\b|\b(--|\')|(\%27)|(\%3D)|(\%3B)|(\%2F)|(\%2A)|(\%00))+/i;`
```

Abbildung 5: regulärer Ausdruck zur Erkennung von SQL-Injektionen

Die Variable `sqlInjectionPattern` enthält die Wörter, die verwendet werden, um eine SQL-Suchabfrage an die Datenbank zu starten. Diese Wörter sollten nicht vom Benutzer gesendet werden, da diese Schlüsselwörter bereits in der Suchabfrage im Backend enthalten sind. Die Nutzereingabe sollte lediglich die Variablen enthalten, nach denen gesucht werden muss.

Als Middleware-Funktion eines Express.js-Servers müssen die Module die Anfrage (`req`), die Antwort an den Client (`res`) und den Parameter `next` als Parameter entgegennehmen. Das hier beispielhaft betrachtete Modul heißt `detectSQLInjection`. Die Funktion `checkForSQLInjection` überprüft, ob die gegebene Eingabe mit einem der vordefinierten Muster im regulären Ausdruck übereinstimmt. Ein regulärer Ausdruck ist ein Muster, das verwendet wird, um Zeichenketten zu beschreiben, zu suchen, oder zu manipulieren. Sollte die Eingabe mit einem der Muster übereinstimmen, erhält die Variable den Wert "true", andernfalls erhält sie den Wert "false".



```
● ● ●
1 const checkForSQLInjection = (value) => {
2   if (typeof value === "string" && sqlInjectionPattern.test(value)) {
3     return true;
4   }
5   return false;
6 }
```

Abbildung 6: Funktion zur Überprüfung einer SQL-Injektionen

Im Programmablauf wird in der Funktion die URL sowie die ID aus der Anfrage extrahiert. Die ID wird aus dem Header der Anfrage entnommen. Daraufhin werden in der Funktion drei Schleifen ausgeführt, die in ihrer Funktionalität identisch sind, sodass das Prinzip auf alle übertragen werden kann. Die erste Schleife extrahiert alle Query-Parameter aus der Anfrage, die typischerweise bei einer GET-Anfrage befüllt sind. Für jeden einzelnen Parameter wird mit der Funktion `checkForSQLInjection` überprüft, ob es sich um eine potenzielle SQL-Injektion handelt. Sollte die Funktion den Rückgabewert `true` liefern, wird in die Log-Datei geschrieben, dass eine SQL-Injektion erkannt wurde. Um die Anfrage in der Auswertung zuordnen zu können, werden zusätzlich die ID und die URL der Anfrage in die Log-Datei geschrieben. Anschließend sendet die Funktion eine Antwort mit dem Statuscode 400 (Bad Request) an den Client und verwirft die Anfrage, ohne sie an den eigentlichen Server weiterzuleiten.

```
1 // Check query parameters
2 for (const param in req.query) {
3     if (checkForSQLInjection(req.query[param])) {
4         fileLogger.log({
5             level: "info",
6             message: "SQL-Injection detected",
7             additional: [id, url],
8             are: "Request Denied",
9         });
10        return res.status(400).send("Bad Request");
11    }
12 }
```

Abbildung 7: Schleife zum iterieren über Anfrage-Parameter

Diese Logik existiert im Middleware-Modul noch zweimal: einmal zur Überprüfung des Anfragekörpers und einmal zur Überprüfung der URL-Parameter. Damit sind alle gängigen Methoden zur Datenübertragung durch eine Anfrage abgedeckt. Sollte in keiner der Schleifen eine Bedrohung erkannt werden, wird die Anfrage über den Funktionsaufruf next() an das nächste Middleware-Modul weitergeleitet. Dieses könnte beispielhaft die Anfrage auf eine Command-Injektion überprüfen. Wenn keine der Middleware-Module eine Bedrohung erkennt, leitet der Server die Anfrage weiter und schreibt die ID sowie die URL in die Log-Datei mit dem Vermerk *valider Request*.

```
1 app.all(
2   "★",
3   proxy("http://localhost:3000", {
4     filter: function (req, res) {
5       var id = req.header("id");
6       var url = req.url;
7
8       if (url.includes("/socket.io")) {
9         return true;
10      }
11
12      logValidRequest(id, url);
13      return true;
14    },
15  }),
16 );
```

Abbildung 8: Funktion zur Weiterleitung des Codes

Beim Testen fiel auf, dass das Backend viele Socket.io-Anfragen erhielt. Diese werden vom Browser unter anderem zum Cachen von Ressourcen verwendet und hätten die Ergebnisse in den Logfiles verfälscht. Um die Auswertung zu vereinfachen und da diese Anfragen keinen Einfluss auf das Ergebnis haben, werden sie in den Zeilen 8-10 herausgefiltert und an den Server weitergeleitet.

4.3 Backend KI-Firewall

Das Backend für die KI-gestützte Firewall ist in seiner Funktionalität identisch mit dem Backend der traditionellen Firewall. Um auf eine dateibasierte Verarbeitung zu verzichten, wurde das Backend in Python implementiert. Somit können die Daten direkt an den Algorithmus weitergegeben werden. Umgesetzt wurde das Backend in Flask. Dies ist ein leichtgewichtiges Web-Framework, das alle Anforderungen erfüllt, ohne dabei zu viele Ressourcen zu benötigen.

```

1 # A single function to handle multiple routes and methods
2 @app.route('/<path:path>', methods=['GET', 'POST'])
3 def catch_all(path):
4     # Extracting the request method
5     method = request.method
6     id = request.headers.get('id')
7     query_params = request.args
8     url_params = request.full_path

```

Abbildung 9: Abfangen aller Anfragen in Flask

Der Server nimmt, wie bei der traditionellen Firewall, Anfragen über jeden Pfad entgegen. Erlaubt sind dabei nur GET- und POST-Anfragen. Initial werden bei eingehenden Anfragen die Methode, die ID aus dem Header und die Anfrageparameter extrahiert. Bei einer POST-Anfrage wird zusätzlich der Anfragekörper extrahiert.

```

1 if len(query_params) > 0:
2     result = injection_test_get(url_params, classifier)
3     app.logger.error('%s, %s', id, result)
4     if result == "NOT_MALICIOUS":
5         full_url = f'{TARGET_URL}{path}'
6         resp = requests.get(full_url, params=request.args)
7         return Response(resp.content, status=resp.status_code, headers=dict(resp.headers))
8
9     return Response("Forbidden Request", status=400)
10 else:
11     print("empty request")
12     app.logger.error('%s, %s', id, "NOT_MALICIOUS")
13
14 full_url = f'{TARGET_URL}{path}'
15 resp = requests.get(full_url, params=request.args)
16 return Response(resp.content, status=resp.status_code, headers=dict(resp.headers))

```

Abbildung 10: Basis Funktion KI-Backend

Das Backend sendet daraufhin die extrahierten Anwendereingaben an den Machine-Learning-Algorithmus und leitet je nach Ergebnis die Anfrage weiter oder sendet dem Anwender eine Forbidden-Request-Antwort.

Der Machine-Learning-Algorithmus orientiert sich am Machine-Learning-Web-Application-Firewall-and-Dataset (fgranqvist, 2017, 5. April/2024). In dem GitHub-Repository wird Schritt für Schritt die Datenaufbereitung zum Trainieren eines Modells beschrieben. Das Repository stellt auch ein bereits

trainiertes Modell zur Verfügung. Dieses wird aufgrund der zeitlichen Dauer und des hohen Ressourcenaufwands, der benötigt wird, um ein Modell zu trainieren, in dieser Forschungsarbeit verwendet. Da es sich um ein Open-Source-Repository handelt, kann jedoch nachvollzogen werden, wie das Modell trainiert wurde und ob es den Forschungsansprüchen genügt.

Das Modell wurde mit über 2929 verschiedenen Angriffen trainiert, darunter 286 verschiedene SQL-Injektionen und 1115 XSS-Angriffe. Beim Trainieren wurde unter anderem darauf geachtet, dass die Angriffe durchgemischt werden, damit der Algorithmus nicht anhand von Anfragemustern eine Entscheidung trifft. Für eine präzisere Entscheidungsfindung bezieht das Modell weitere Parameter mit ein, wie z. B. die Anzahl einzigartiger Bytes innerhalb der Anfrage, den maximalen Byte-Wert und den minimalen Byte-Wert sowie die Anfragelänge.

Zum Trainieren des Modells wurde die Python-Bibliothek scikit-learn (auch als sklearn) verwendet. Der TfidfVectorizer ist eine Klasse zur Umwandlung von Textdaten in numerische Merkmale. Diese Umwandlung muss durchgeführt werden, damit der maschinelle Lernalgorithmus die Daten verarbeiten kann. Die Klasse besteht aus zwei Hauptkomponenten: TF (Term Frequency), das misst, wie oft ein Wort in einem Dokument vorkommt, und Inverse Document Frequency, das die Wichtigkeit des Wortes misst. Für das Treffen der Vorhersagen wird der RandomForestClassifier ebenfalls aus der scikit-learn verwendet. Dieses Lernverfahren erstellt eine Ansammlung von Entscheidungsbäumen während des Trainings, die daraufhin zusammenarbeiten, um eine Vorhersage zu treffen. Der RandomForestClassifier zeichnet sich vor allem durch seine hohe Genauigkeit und Effizienz aus.

Für POST- und GET-Anfragen werden zwei verschiedene Methoden implementiert, die funktional identisch sind. Sie unterscheiden sich lediglich darin, wie die Anfragezeichenkette aufgeteilt wird. Die Funktion teilt die Zeichenkette auf und fügt sie in eine Liste hinzu. Danach iteriert sie über die Liste und übergibt die einzelnen Werte an das Modell zur Einschätzung, ob es sich um eine legitime Anfrage handelt oder nicht. Das Ergebnis wird daraufhin als Rückgabewert zurückgegeben. Der Server schreibt dann das Ergebnis mit der ID in die Log-Datei.

```
● ● ●
```

```
1 def injection_test_get(inputs, classifier):
2     variables = inputs.split('&')
3     values = []
4     for variable in variables:
5         values.append(variable.split("="))
6     print(values)
7     mal = False
8     for value in values:
9         if classifier.predict(value).sum() > 0:
10             mal = True
11             break
12
13     if mal:
14         return "MALICIOUS"
15
16     return "NOT_MALICIOUS"
```

Abbildung 11: KI-Bewertung von Anfragen

4.4 Senden der Anfragen

Zum Senden der Anfragen an den Server wurde ein Python-Skript implementiert, das die Anfragenliste durchläuft, die Anfragen zusammenstellt und sie anschließend absendet. Mit dem Start des Skripts wird die Zeit gemessen, und nach Erhalt der letzten Antwort ebenfalls. Auf diese Weise kann gemessen werden, wie lange der Server benötigt hat, um die 3070 Anfragen zu verarbeiten.

```

● ● ●
1 start = time.time()
2
3 with open("final_payloads.csv", "r") as csvfile:
4     reader = csv.DictReader(csvfile, delimiter=";")
5
6     for line in reader:
7         method = line["method"]
8         url = line["url"]
9         payload = line["payload"]
10        id = line["id"]
11
12        if method == "GET":
13            final_url = "".join([url, payload])
14            res = requests.get(url=final_url, headers={"id": id})
15            print(res.text)
16
17        if method == "POST":
18            res = requests.post(url=url, json=payload, headers={"id": id})
19            print(res.text)
20
21 csvfile.close()
22
23 end = time.time()
24
25 print("Time taken: ", end - start)

```

Abbildung 12: Skript zum senden der Anfragen an den Server

5 Vergleichende Analyse

Wie bereits beschrieben, wurden insgesamt 3070 Anfragen an die jeweiligen Firewalls gesendet. Davon waren 301 Anfragen legitim. Die Firewalls mussten daher 2769 Angriffe erkennen. Die Aufteilung der Angriffsarten sieht wie folgt aus:

Angriffsart	Anzahl
Authentication Bypass	242
Command Execution	896
Path Traversal	280
SQL Injection	984
XSS	358

Tabelle 1: Tabelle Angriffsarten und Anzahl

5.1 Ergebnis traditioneller Firewall

Das Ergebnis der traditionellen Firewall sieht wie folgt aus:

Ergebnis	Command Injection erkannt	Path Traversal erkannt	SQL-Injection erkannt	Legitime Anfrage erkannt	XSS erkannt	Gesamtsumme
Authentication Bypass			108	134		242
Command Execution	209		80	607		896
Path Traversal	17	83		180		280
Legitime Anfrage				310		310
SQL Injection	6		452	526		984
XSS	152		22	179	5	358
Gesamtsumme	384	83	662	1936	5	3070

Tabelle 2: Ergebnis traditionelle Firewall

Durch die gewählte Implementierung ist es möglich, zu erkennen, welche Art von Bedrohung die Firewall einer Anfrage zugeordnet hat. Insgesamt hat die traditionelle Firewall 1936 Anfragen als legitim eingeordnet. Bei 384 Anfragen ging die Firewall von einer Command Injection aus. 83 Anfragen wurden als Path Traversal kategorisiert, und ein vermeintlicher XSS-Angriff wurde 5-mal erkannt. 662 Anfragen wurden als SQL-Injektion markiert. Insgesamt wurden also 1444 Anfragen korrekt als Angriff oder legitime Anfrage eingeordnet. Bei 1626 Anfragen war sich die Firewall sicher, dass es sich um keinen Angriff handelt, obwohl es sich tatsächlich um einen Angriff gehandelt hätte.

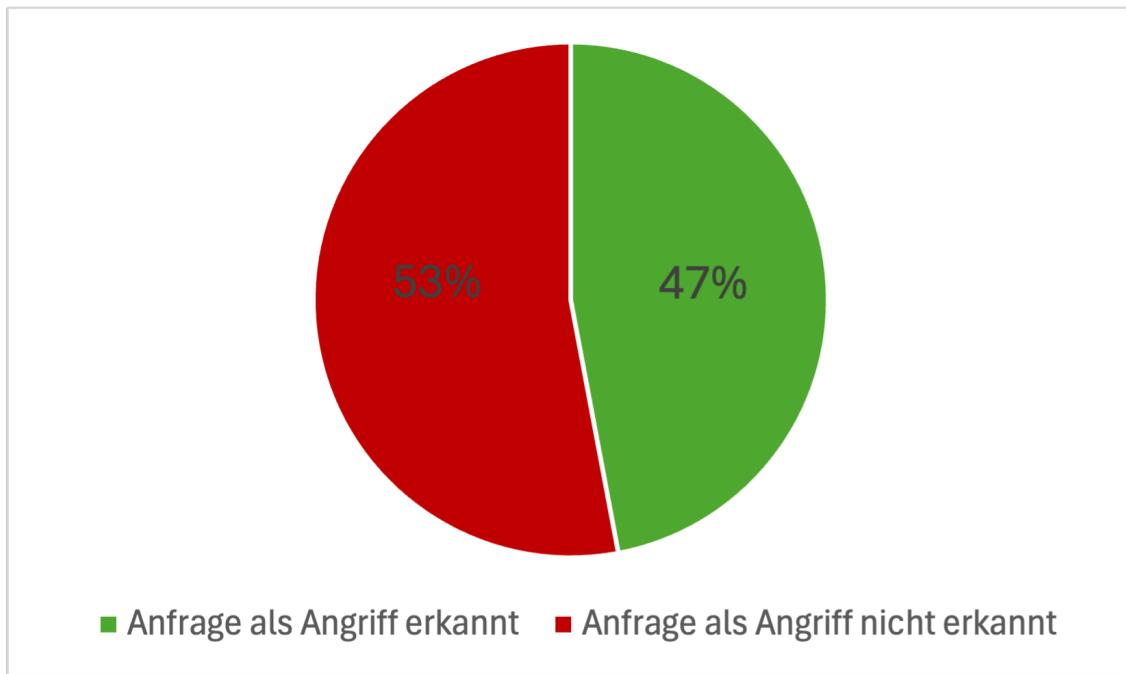


Abbildung 13: Ergebnis traditionelle Firewall

Bei einer tiefergehenden Betrachtung lässt sich erkennen, dass die traditionelle Firewall zwar viele Anfragen als Angriff erkannt hat, jedoch oft der falschen Angriffskategorie zugeordnet hat. Von den 1444 als Angriff gekennzeichneten Anfragen wurden 1059 der richtigen Angriffskategorie zugeordnet, während 390 Anfragen falsch kategorisiert wurden.

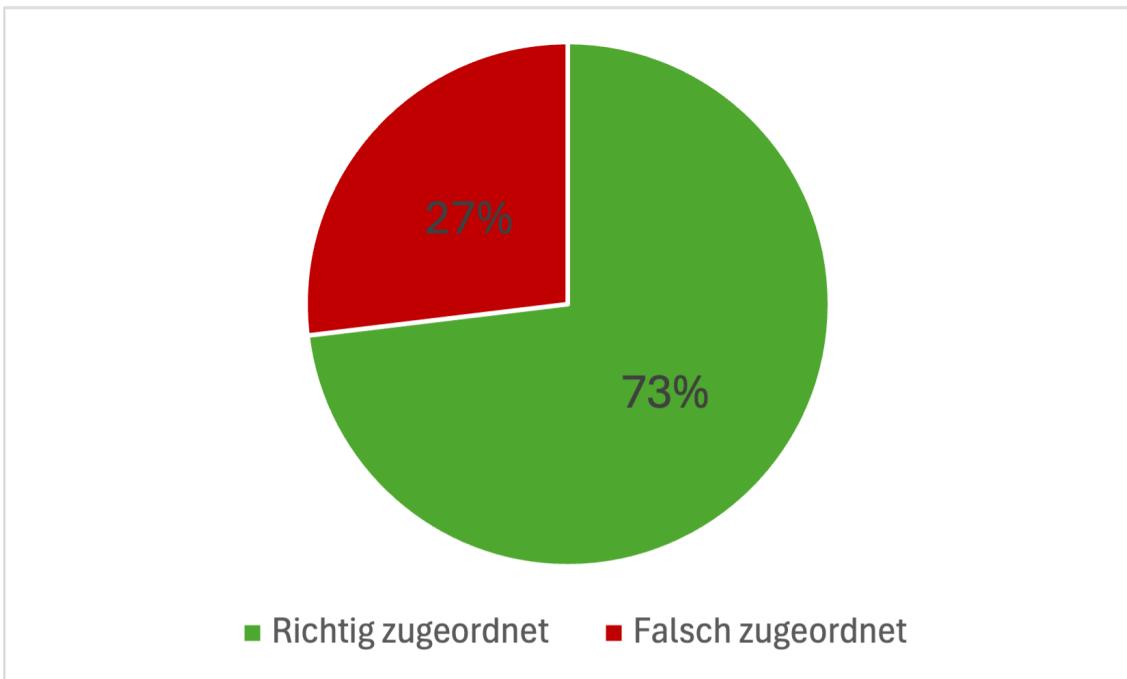


Abbildung 14: Zuordnung von Angriffen traditionelle Firewall

Bei der Kategorisierung des Ergebnis der traditionellen Firewall lässt sich festhalten, dass die keine Anfrage als False Positive eingestuft wurde. Die Menge an False Positives ist wie bereits zu erkennen war zu hoch.

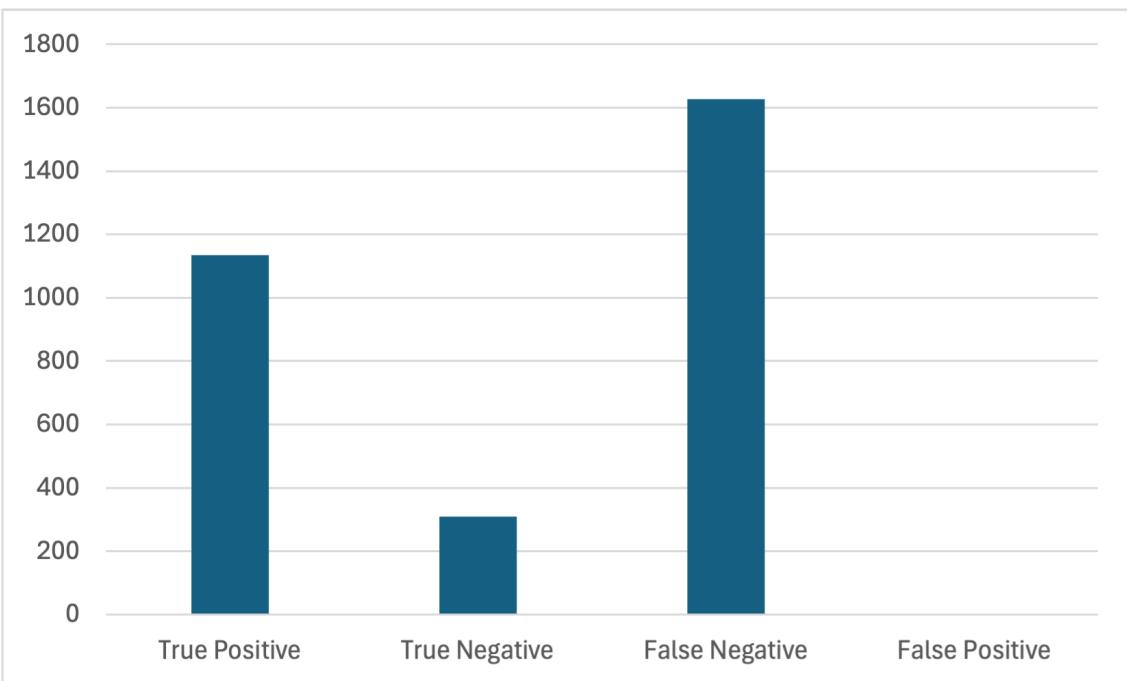


Abbildung 15: Kategorisierung des Ergebnis

Eine Aufteilung nach den Angriffstypen zeigt, mit welcher Angriffsart die traditionelle Firewall am meisten Probleme hatte.

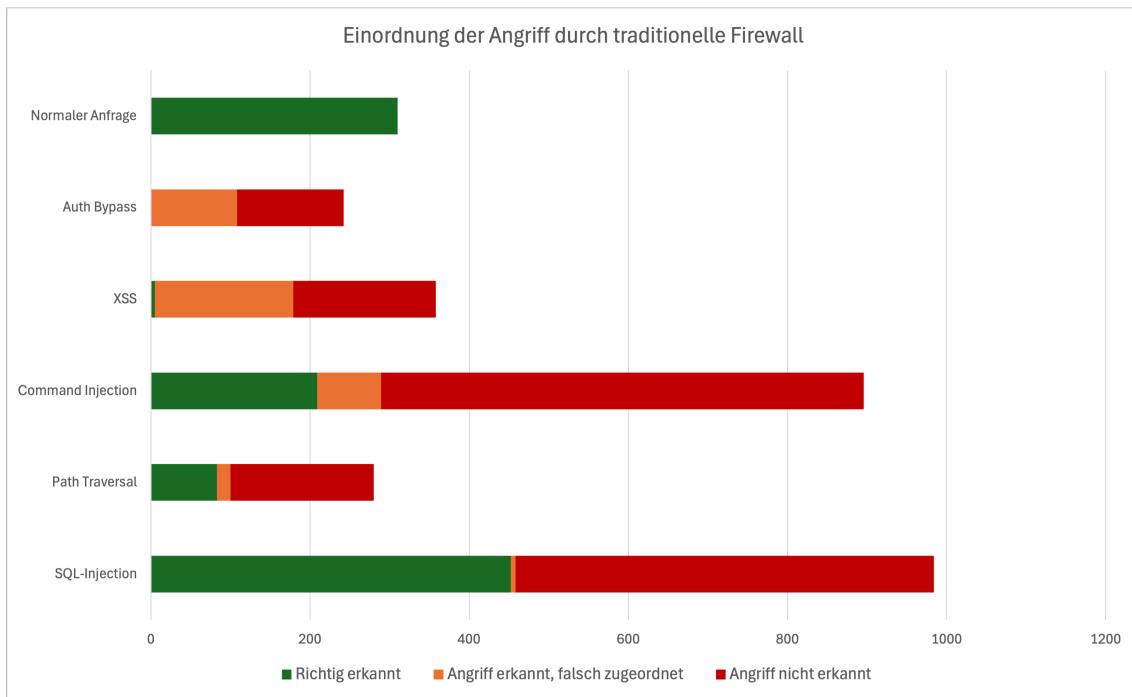


Abbildung 16: Zuordnung von Angriffen traditionelle Firewall

Um die Laufzeit zu bestimmen wurde das Anfragen-Skript 10-mal hintereinander ausgeführt. Die Erkennungsrate blieb dabei gleich. Die Laufzeit variierte dabei im geringen Maße. Die durchschnittliche Dauer zur Beantwortung und Kategorisierung der 3070 Anfragen dauerte 16.5 Sekunden.

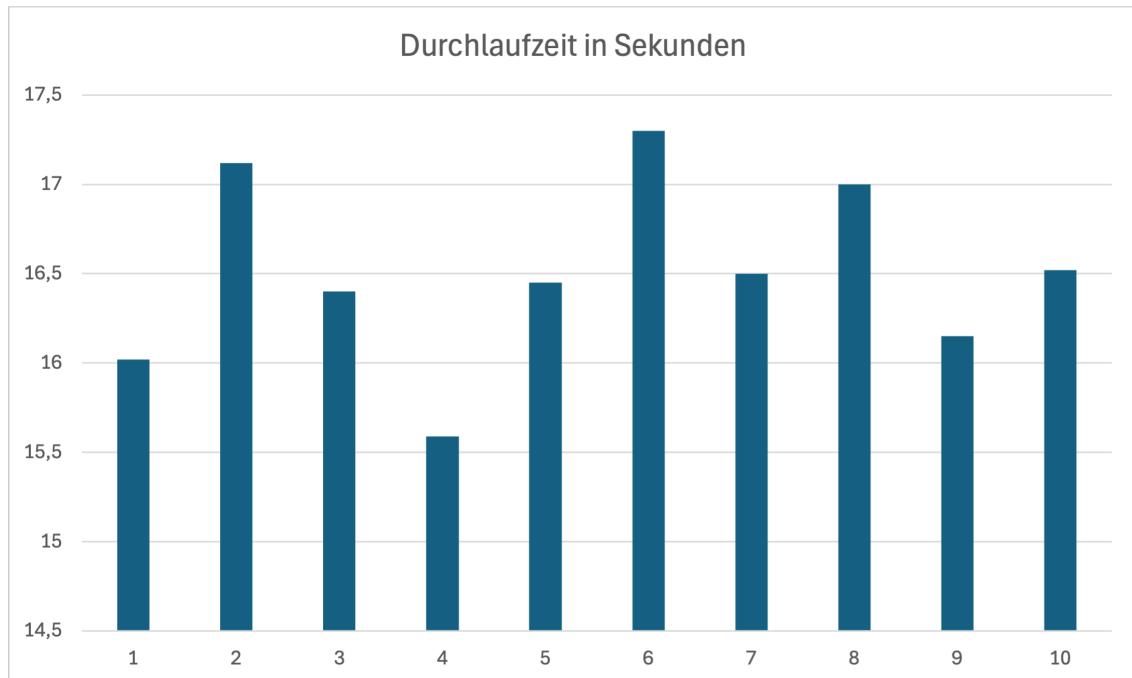


Abbildung 17: Durchlaufzeit traditionelle Firewall

Ohne die Filterregeln der Firewall weichte die Durchlaufzeit nicht signifikant von der Durchlaufzeit mit den Filterregeln ab.

5.2 Ergebnis KI-Firewall

Aufgrund der unterschiedlichen Implementierung und der Gestaltung des Modells hat der maschinelle Lernalgorithmus, wie bereits beschrieben, nur zwei Ausgabevariablen. Eine so detaillierte Auswertung wie bei der traditionellen Firewall ist daher nicht möglich. Wenn ein Angriff erkannt wurde, kann nicht ausgewertet werden, als welchen Angriffsvektor die KI diesen kategorisiert hat. Es kann lediglich gesagt werden, dass ein Angriffsversuch stattgefunden hat.

Das Ergebnis sieht wie folgt aus:

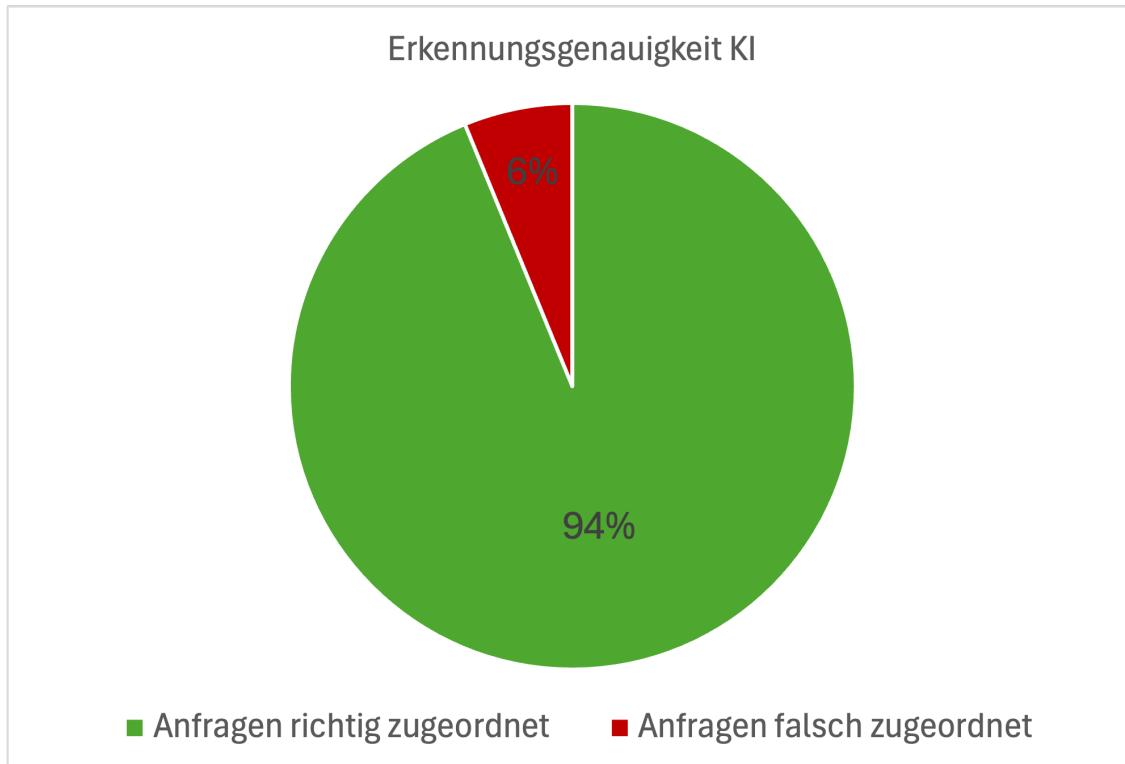


Abbildung 18: Erkennungsgenauigkeit KI

Eine Aufteilung nach Angriffsart zeigt, dass der künstliche Lernalgorithmus nahezu alle Angriffsarten sehr gut erkennen konnte. Wie in Abbildung 19 zu erkennen ist, schnitt der Algorithmus bei der Command Injection mit einer Erkennungsrate von 83,88% ab.

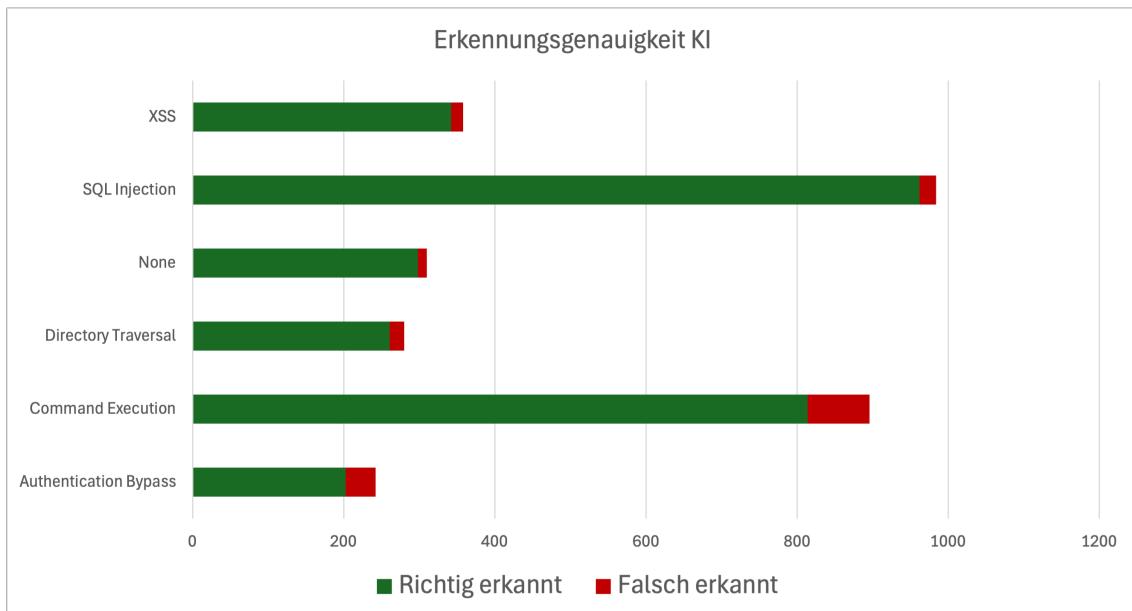


Abbildung 19: Erkennungsgenauigkeit KI nach Angriffsart

Im Vergleich zur traditionellen Firewall schneidet der künstliche Lernalgorithmus bei der Klassifizierung nach False-Negatives schlechter ab. Die Menge an True-Positives ist allerdings deutlich besser.

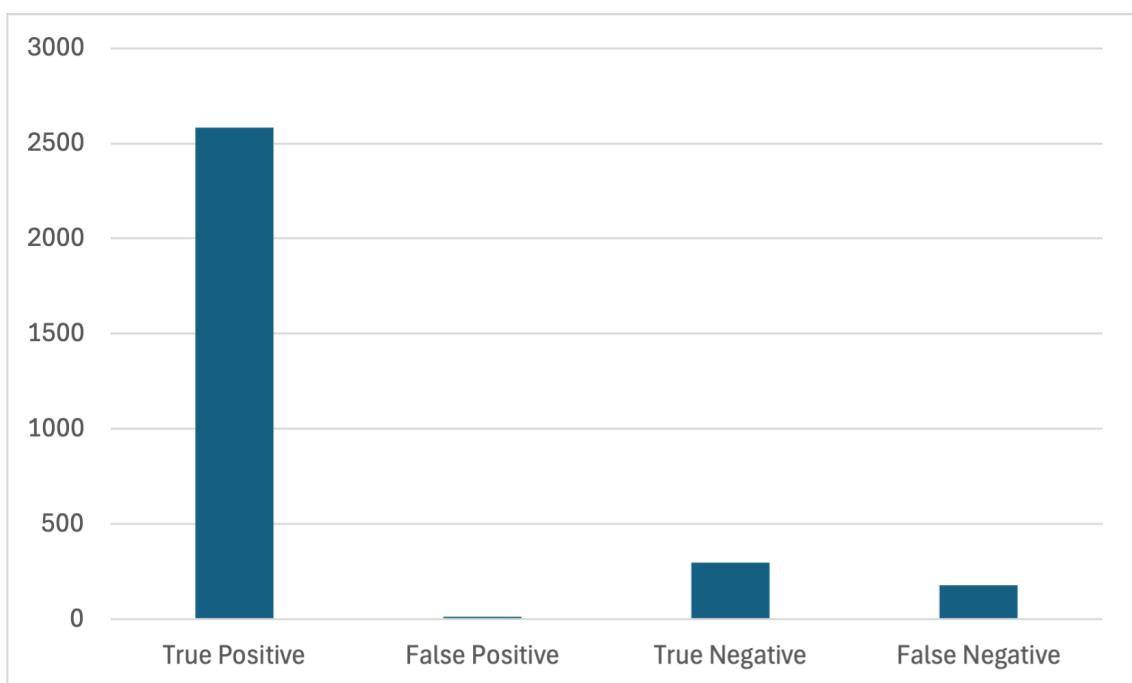


Abbildung 20: Kategorisierung Ergebnis KI

Auffällig ist dass die Durchlaufzeit mit dem künstlichen Lernalgorithmus deutlich höher war, als bei der traditionellen Firewall. Die durchschnittliche Durchlaufzeit bei zehn gemessenen Versuchen betrug 57,59 Sekunden.

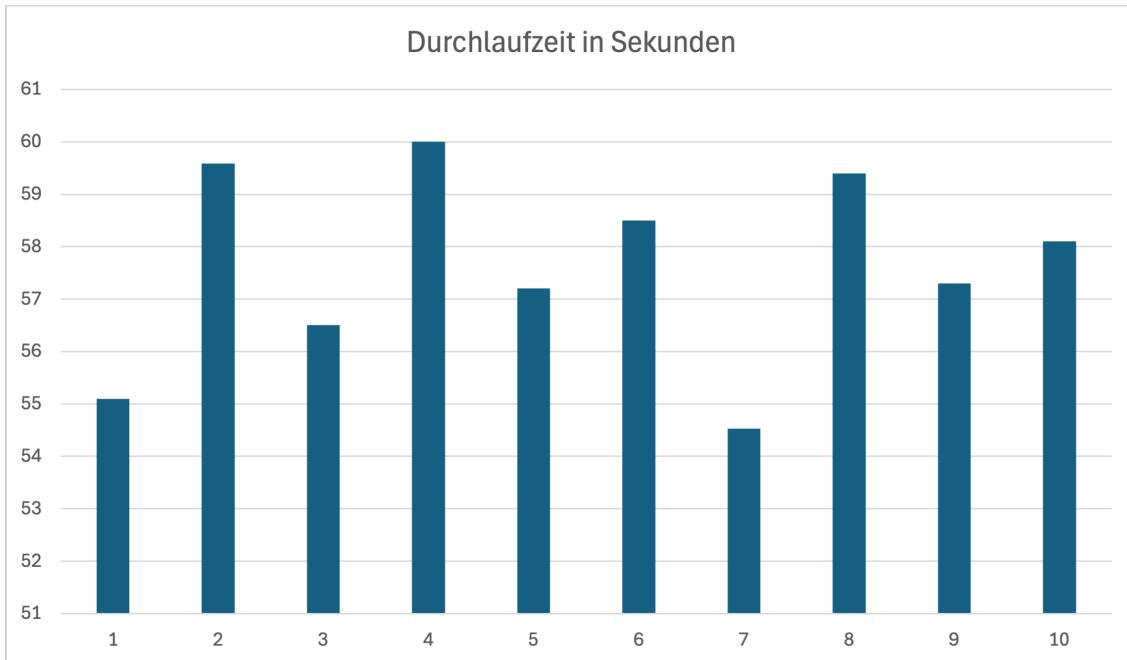


Abbildung 21: Durchlaufzeit KI

Das bedeutet, dass die Firewall mit dem künstliche Lernalgorithmus ca. 249% langsamer war, als die traditionelle Firewall.

6 Diskussion

6.1 Interpretation der Ergebnisse

Die Messungen haben viele interessante Ergebnisse geliefert. Offensichtlich ist, dass die traditionelle Firewall mit einer Erkennungsgenauigkeit von lediglich 47% sehr schlecht abgeschnitten hat. Besonders Command- und SQL-Injections bereiteten der Firewall Probleme.

Problematisch für die traditionelle Firewall sind ebenfalls die verschiedenen Encodings. Zwar wurde in den jeweiligen Pattern unter anderem URL-Encoding inkludiert, jedoch lässt sich auf diese Art und Weise vieles Abfangen, z.B. die automatische URL-Verschlüsselung durch den Browser, aber eben nicht alle Fälle. Die Problematik mit so genannter Obfuscation lässt sich nur schwer umgehen. Bei Obfuscation verschlüsselt der Angreifer die Anfrage z.B. indem er Zeichenketten bewusst trennt. Die getrennten einzelnen Silben treffen demnach nicht mehr auf einen der Filter zu und werden von der Firewall als legitime Anfrage durchgelassen. Bei der Verarbeitung des Codes im Backend würden die Zeichenketten daraufhin zusammengesetzt werden und der Server würde den Angriff ausführen. Eine gesamtheitliche Betrachtung des übermittelten Angriffs fällt der traditionellen Firewall also sehr schwer.

Eine weitere Limitation durch die reine Filterung nach Stichwörtern und Zeichen ist, dass legitime Anfragen dadurch als False Positive gekennzeichnet werden. Sollte ein Anwender beispielsweise 'Select' oder 'Insert' als Nutzernamen verwenden wollen, würde die Firewall dies als Angriff einstufen. Es müsste also eine recht komplexe Logik mit vielen Ausnahmefällen implementiert werden um eine flüssige Nutzererfahrung bieten zu können.

Eine weitere interessante Beobachtung bei der isolierten Betrachtung des Ergebnisses der traditionellen Firewall ist, dass zwar ca. ein Drittel der Anfragen zwar richtiger Weise als Angriff eingeordnet wurden, sie jedoch der falschen Angriffsart zugeordnet wurden. Dies kann daran liegen, dass sich manche Stichwörter der einzelnen Angriffsarten ähneln.

Durch das reinen Vergleichen von Zeichenketten ist die traditionelle Firewall von der reinen Bearbeitungszeit her sehr effizient. Dies kann bei zeitkritischen Anwendungsfällen von Vorteil sein. Jedoch kann diese Art der Datenverarbeitung bei sehr langen Zeichenketten ebenfalls an seine Grenzen stoßen. Damit hat die traditionelle Firewall dennoch einen großen Vorteil gegenüber der Firewall mit künstlicher Intelligenz. Die durchschnittliche Laufzeit war signifikant besser. Die Laufzeit der Firewall mit künstlicher Intelligenz ließe sich mit besserer und schnellere Hardware zwar noch verbessern, jedoch hätte dies auch eine drastische Auswirkung auf die Kosten der jeweiligen Firewall.

Die künstliche Intelligenz schnitt über jede Angriffsart hinweg besser ab, als die traditionelle Firewall. Jedoch gibt es auch hier interessante Teilergebnisse. Beispielsweise ordnete die KI die Anfragen

3050 und 3052 als Angriff ein. Dabei handelte es sich um legitime Anfragen, die die Konversation mit dem Chat-Bot des Juice-Shops imitierten. Vermutlich sah die KI diese Anfragen als Bedrohung, da der Parameter für den Chat-Bot vom Entwickler als "query" bezeichnet wurde. Solche Anwendungsfälle würden eine enge Abstimmung mit dem Entwicklungsteam voraussetzen beziehungsweise ein gezieltes Training des Models erfordern. In beiden Fällen ist der Entwicklungsaufwand sehr hoch.

Ebenfalls interessant ist die Betrachtung der Anfrage mit der ID 1994, diese wurde von der KI als eine legitime Anfrage eingestuft, obwohl es sich dabei um eine Command-Execution handelt. Am Stichwort "echo" kann es nicht liegen, denn eine ähnliche Anfrage mit der ID 1855 wurde von der KI als Angriff eingestuft. Das bedeutet, dass es vermutlich an der Länge der Anfrage liegen musste. Diese war mit 63 "XBuchstaben sehr lang und deutet darauf hin, dass besonders lange Anfragen durch die Gewichtung tendenziell als unproblematisch eingestuft werden. Dies zeigt, wie schwierig es ist die Gewichtung der zu betrachtenden Parameter beim Trainieren eines Models für die KI gut zu wählen.

Die Reliabilität der Ergebnisse bezieht sich darauf, wie konsistent und wiederholbar die Messungen sind. Um diese Sicherzustellen wurde das Anfragenskript zehnmal hintereinander ausgeführt, wobei die Erkennungsrate gleich blieb. Dies deutet auf eine hohe Reliabilität der Messungen hin, da die Ergebnisse bei wiederholten Tests konsistent waren. Für zukünftige Studien könnte es interessant sein wiederholte Messungen, bei gleichbleibender Angriffsart Verteilung, allerdings mit unterschiedlichen Payload durchzuführen.

Die Verallgemeinerbarkeit bezieht sich darauf, inwieweit die Ergebnisse der Studie auf andere Szenarien übertragen werden können. Die Verallgemeinerbarkeit ist eingeschränkt, da die Firewalls nicht in unterschiedlichen Umgebungen getestet werden konnten. Insbesondere aufgrund der spezifischen Implementierungsdetails und der Verwendung eines vortrainierten Modells, das nicht optimal für die spezifischen Anforderungen dieser Studie angepasst wurde.

Die Studie wurde objektiv durchgeführt, da die Messungen auf klar definierten Kriterien beruhen, wie z.B. der Erkennungsgenauigkeit von Angriffen und der Durchlaufzeit.

6.2 Limitationen der Studie

Wie bereits in der Ergebnisauswertung angedeutet wurde aufgrund der zeitlichen Limitation ein vortrainiertes Model verwendet. Um ein optimales Ergebnis mit künstlicher Intelligenz zu erzielen, sollte eigenständig ein Model trainiert werden. Dadurch könnte eine Auswertung, wie bei der traditionellen Firewall erreicht werden, indem die Ausgabeparameter bestimmt werden. Im Rahmen der Arbeit war es jedoch nicht möglich genügend Trainingsdaten zu generieren und Ressourcen zur Verfügung zu haben um ein solches Model zu trainieren. Dabei ist vor allem die Auswahl und das damit verbundene Ausreichende testen des richtigen KI-Modells eine Herausforderung. Des weiteren hätte die traditio-

nelle Firewall mit ihren strickten Filtern nach Stichwörtern teilweise die Nutzererfahrung eines echten Anwenders eingeschränkt. Wie in der Interpretation besprochen, müsste mit den jeweiligen Firewalls ein Test in einer Produktionsnahen Umgebung stattfinden, um sicherzustellen dass alle Funktionalitäten der Webseite oder Applikation noch vorhanden sind.

Aufgrund der zeitlichen Limitationen und der unzureichenden Ressourcen war es nicht möglich, ein eigenes Modell zu trainieren und eine ausreichende Menge an Trainingsdaten zu generieren. Deshalb konnte der Wartungsaufwand, insbesondere in Bezug auf das kontinuierliche Training des Modells auf neu auftretende Bedrohungen, nicht evaluiert werden. Weiterhin ist es nicht möglich den Wartungsaufwand für die traditionelle Firewall zu evaluieren, da die Integration in ein bestehendes System und die damit verbundenen Komplikationen im Rahmen der Forschungsarbeit nicht getestet werden konnte.

7 Schlussfolgerung und Ausblick

7.1 Zusammenfassung der wichtigsten Ergebnisse

Zusammenfassend bietet diese Forschungsarbeit einen umfassenden Überblick über die Themen künstliche Intelligenz und Cybersicherheit sowie die damit verbundenen Herausforderungen. Es konnte festgestellt werden, dass beide Ansätze zum Aufbau einer Firewall ihre Berechtigung haben. Die von künstlicher Intelligenz unterstützte Firewall zeichnete sich insbesondere durch eine höhere Erkennungsgenauigkeit bei Angriffen aus. Im Gegensatz dazu konnte die traditionelle Firewall Anfragen wesentlich schneller bearbeiten und auf weniger leistungsfähiger Hardware betrieben werden.

Zur Beantwortung der Forschungsfrage „Wie genau identifizieren KI-basierte Systeme im Vergleich zu traditionellen Methoden verschiedene Arten von Cyber-Bedrohungen?“ zeigt sich, dass die KI-unterstützte Firewall insbesondere bei neuartigen oder nicht in den Filtern der traditionellen Firewall abgebildeten Angriffsmethoden signifikant bessere Ergebnisse liefert.

Die zweite Forschungsfrage „Ressourceneffizienz“ lässt sich nicht pauschal beantworten, da dies stark von den jeweiligen Anwendungsfällen abhängt. Obwohl KI-Systeme in der Regel mehr Ressourcen benötigen und nicht auf allen Systemen ausgeführt werden können, auf denen traditionelle Firewalls betrieben werden, liefern sie im Gegenzug eine wesentlich bessere Leistung. Es lässt sich jedoch eindeutig bestätigen, dass die höhere Erkennungsrate von KI-Systemen mit einem signifikant höheren Ressourcenverbrauch erkauft wird.

Die dritte Forschungsfrage „Implementierung und Wartung“ kann objektiv nicht vollständig beantwortet werden. Der zeitliche Aufwand für die Implementierung der traditionellen Firewall war deutlich geringer. Der Wartungsaufwand konnte jedoch aufgrund der Limitationen der Studie nicht abschließend bewertet werden.

7.2 Empfehlungen für die Praxis

Bei der praktischen Umsetzung einer Firewall, die auf künstlicher Intelligenz basiert, gibt es mehrere wesentliche Empfehlungen, die berücksichtigt werden sollten, um die Effektivität und Effizienz der Lösung sicherzustellen:

- Datenmanagement - ein effektives Datenmanagement ist von entscheidender Bedeutung für den Erfolg einer KI unterstützten Firewall. Die Effektivität der Firewall ist direkt abhängig von der Qualität und Integrität der gesammelten Daten. Daher ist es ratsam die Daten regelmäßig zu bereinigen und zu aktualisieren.
- Sorgfältige Modell Auswahl - die Auswahl des Modells ist entscheidend für die Leistung der Firewall und jedes Modell bietet unterschiedliche Stärken und Schwächen. Es empfiehlt sich

daher verschiedene Modelle zu evaluieren, um herauszufinden, welches Modell am besten zu den bestehenden Sicherheitsbedürfnissen passt.

- Überwachung und Wartung - auch nach der Implementierung ist eine kontinuierliche Überwachung und Wartung der Firewall unerlässlich. Eine proaktive Überwachung ermöglicht es, potenzielle Schwachstellen oder Fehlalarme frühzeitig zu erkennen und zu beheben.

7.3 Vorschläge für zukünftige Forschungen

Wie bereits in der Limitation der Arbeit aufgefasst konzentrierte sich die Arbeit auf eine relativ schmale Problemstellung. Im Gesamtkontext der Netzwerktechnik gibt es noch viele Forschungsmöglichkeiten in diesem Themengebiet. Daher liefert diese Forschungsarbeit einen Grundstein für weitere Forschungsthemen.

Ein ähnlicher Versuchsaufbau zur Absicherung eines gesamten Netzwerks könnte interessante Erkenntnisse liefern und sollte untersucht werden. Wie bereits beschrieben sollte der gleiche Versuch mit besserer Hardware und mit einem eigens dafür trainierten Modell durchgeführt werden. Dabei wäre vor allem interessant zu sehen ob das eigens trainierte Modell signifikant besser abschneidet. Weiterhin sollte untersucht werden inwiefern eine Optimierung in der Verarbeitung von Daten durch künstliche Intelligenz die Bearbeitungszeit erheblich verbessern kann.

Die vorliegende Forschungsarbeit hat sich auf eine spezifische Problematik im Bereich der künstlichen Intelligenz und Cybersicherheit konzentriert und bietet somit eine solide Basis für weiterführende Studien. Die Limitationen der Arbeit verdeutlichen, dass es im umfassenderen Kontext der Netzwerktechnik noch zahlreiche offene Fragestellungen und Forschungsansätze gibt. Im Folgenden werden einige zentrale Vorschläge für zukünftige Forschungen dargelegt, die auf den Erkenntnissen dieser Arbeit aufbauen und neue Perspektiven eröffnen könnten:

- Erweiterung des Versuchsaufbau auf das gesamte Netzwerk - Die vorliegende Studie fokussierte sich auf den Einsatz von KI in einer isolierten Firewall-Umgebung. Zukünftige Forschungen könnten den Versuch auf die Absicherung eines gesamten Netzwerks ausweiten. Ein solcher Ansatz würde es ermöglichen, die Wirksamkeit von KI-unterstützten Sicherheitslösungen in einem realistischeren und komplexeren Umfeld zu evaluieren. Hierbei könnten insbesondere Fragen zur Integration von KI-basierten Firewalls in bestehende Netzwerksicherheitsarchitekturen untersucht werden. Auch der Einfluss auf die Netzwerkperformance und die potenziellen Wechselwirkungen mit anderen Sicherheitstechnologien wären von Interesse.
- Einsatz verbesserter Hardware und maßgeschneiderter Modelle - Die Studie hat gezeigt, dass die Hardwareleistung und die Qualität der Modelle einen erheblichen Einfluss auf die Effektivität der KI-gestützten Firewall haben können. Zukünftige Forschungen könnten darauf abzielen, die

Experimente unter Einsatz leistungsfähigerer Hardware und maßgeschneiderter, speziell trainierter KI-Modelle zu wiederholen. Es wäre besonders aufschlussreich zu analysieren, ob diese verbesserten Modelle signifikante Leistungssteigerungen in der Erkennung und Verarbeitung von Bedrohungen bieten. Hierbei könnten auch fortgeschrittene Machine-Learning-Techniken und neuartige Algorithmen zum Einsatz kommen, um die Effizienz und Genauigkeit der Modelle weiter zu optimieren.

- Langfristige Auswirkungen und Skalierbarkeit - Langfristige Studien könnten sich auf die dauerhaften Auswirkungen der Integration von KI in Cybersicherheitslösungen konzentrieren. Dabei könnten Fragen zur langfristigen Skalierbarkeit und Anpassungsfähigkeit von KI-Systemen an sich verändernde Bedrohungslandschaften untersucht werden. Ebenso wäre es wichtig zu erforschen, wie sich solche Systeme über längere Zeiträume hinweg bewähren und welche Anpassungen erforderlich sind, um ihre Effektivität aufrechtzuerhalten.
- Verwendung von KI-gestützten Systemen mit traditionellen Ansätzen - Ein Studie könnte sich darauf konzentrieren zu erforschen, ob ein besseres Ergebnis erzielt werden kann, wenn die beiden Ansätze in Kombination miteinander verwendet werden.

Diese Vorschläge bieten zahlreiche Ansatzpunkte für zukünftige Forschungen, die dazu beitragen könnten, die Anwendung von künstlicher Intelligenz in der Cybersicherheit weiter zu verbessern und zu optimieren.

Literaturverzeichnis

- Altman, S. (2023). „The reason to develop AI at all, in terms of impact on our lives and improving our lives and upside, this will be the greatest technology humanity has yet developed.“
- BKA - Listenseite Für Pressemitteilungen 2023 - Veröffentlichung Bundeslagebild: Über 130.000 Fälle von Cybercrime in 2022.* (2023, 31. Dezember). Verfügbar 26. Juli 2024 unter https://www.bka.de/DE/Presse>Listenseite_Pressemitteilungen/2023/Presse2023/230816_PM_BLB_Cybercrime.html
- D21-Digital-Index 2022/2023.* (2023, 31. Dezember). Initiative D21. Verfügbar 9. Juli 2024 unter <https://initiatived21.de/publikationen/d21-digital-index/2022-2023>
- eco Branchenmonitor Internetwirtschaft 2023.* (2023, 31. Dezember). eco. Verfügbar 29. Juni 2024 unter <https://www.eco.de/themen/eco-studien-und-whitepaper/eco-branchenmonitor-internetwirtschaft-2023/>
- fgranqvist. (2024, 9. August). *Grananqvist/Machine-Learning-Web-Application-Firewall-and-Dataset.* Verfügbar 29. August 2024 unter <https://github.com/grananqvist/Machine-Learning-Web-Application-Firewall-and-Dataset>
- Heino, J., Hakkala, A. & Virtanen, S. (2022). Study of Methods for Endpoint Aware Inspection in a next Generation Firewall. *Cybersecurity*, 5(1), 25. <https://doi.org/10.1186/s42400-022-00127-8>
- Jenis Nilkanth Welukar & Gagan Prashant Bajoria. (2021). Artificial Intelligence in Cyber Security - A Review. *International Journal of Scientific Research in Science and Technology*, 488–491. <https://doi.org/10.32628/IJSRST218675>
- KI und Cyberbedrohungen: Neue Risiken durch Malware-Nutzung von ChatGPT.* (2024, 24. Juni). Verfügbar 9. Juli 2024 unter <https://www.mind-verse.de/news/ki-und-cyberbedrohungen-neue-risiken-durch-malware-nutzung-chatgpt>
- Kumar, S., Dalal, S. & Dixit, V. (2014). THE OSI MODEL: OVERVIEW ON THE SEVEN LAYERS OF COMPUTER NETWORKS. 2(3).
- Li, X., Lin, W. & Guan, B. (2023). The Impact of Computing and Machine Learning on Complex Problem Solving. *Engineering Reports*, 5. <https://doi.org/10.1002/eng2.12702>
- Nasteski, V. (2017). An overview of the supervised machine learning methods. *HORIZONS.B*, 4, 51–62. <https://doi.org/10.20544/HORIZONS.B.04.1.17.P05>
- PricewaterhouseCoopers, .-. (2024, 24. Juni). *Digital Trust Insights 2024.* PwC. Verfügbar 27. Juni 2024 unter <https://www.pwc.de/de/cyber-security/digital-trust-insights.html>
- Saha, S. S., Sandha, S. S. & Srivastava, M. (2022). Machine Learning for Microcontroller-Class Hardware: A Review. *IEEE Sensors Journal*, 22(22), 21362–21390. <https://doi.org/10.1109/JSEN.2022.3210773>

Swisskyrepo/PayloadsAllTheThings: A List of Useful Payloads and Bypass for Web Application Security and Pentest/CTF. (2024). Verfügbar 27. August 2024 unter
<https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master>

A Anhang

A.1 Code Firewall KI



```
1 import pickle
2 from sklearn.feature_extraction.text import TfidfVectorizer
3 from sklearn.ensemble import RandomForestClassifier
4 from flask import Flask, request, jsonify, Response
5 import logging
6 import requests
7
8 def injection_test_post(inputs, classifier):
9     variables = inputs.split('&')
10    values = []
11    for variable in variables:
12        values.append(variable.split("=:"))
13    print(values)
14    mal = False
15    for value in values:
16        if classifier.predict(value).sum() > 0:
17            mal = True
18            break
19
20    if mal:
21        return "MALICIOUS"
22
23    return "NOT_MALICIOUS"
24
25 def injection_test_get(inputs, classifier):
26    variables = inputs.split('&')
27    values = []
28    for variable in variables:
29        values.append(variable.split("=:"))
30    print(values)
31    mal = False
32    for value in values:
33        if classifier.predict(value).sum() > 0:
34            mal = True
35            break
36
37    if mal:
38        return "MALICIOUS"
39
40    return "NOT_MALICIOUS"
41
42 classifier = pickle.load( open("data/tfidf_2grams_randomforest.p", "rb"))
43 logging.basicConfig(filename="ai_log.log", level=logging.ERROR)
44
45 TARGET_URL = 'http://localhost:3000/'
46 app = Flask(__name__)
47
48 # A single function to handle multiple routes and methods
49 @app.route('<path>', methods=['GET', 'POST'])
50 def handle_all(path):
51     # Extracting the request method
52     method = request.method
53     id = request.headers.get('id')
54     query_params = request.args
55     url_params = request.full_path
56
57     # Handling GET request
58     if method == 'GET':
59
60         if len(query_params) > 0:
61             result = injection_test_get(url_params, classifier)
62             app.logger.error('%s %s %s', result)
63             if result == "NOT_MALICIOUS":
64                 full_url = f'{TARGET_URL}{path}'
65                 resp = requests.get(full_url, params=request.args)
66                 return Response(resp.content, status=resp.status_code, headers=dict(resp.headers))
67
68             return Response("Forbidden Request", status=400)
69         else:
70             print("empty request")
71             app.logger.error('%s %s', id, "NOT_MALICIOUS")
72
73             full_url = f'{TARGET_URL}{path}'
74             resp = requests.get(full_url, params=request.args)
75             return Response(resp.content, status=resp.status_code, headers=dict(resp.headers))
76
77     # Handling POST request
78     elif method == 'POST':
79
80         request_data = request.data
81
82         if len(request_data) > 0:
83             request_value = request_data.decode("utf-8")
84             result = injection_test_post(request_value, classifier)
85             app.logger.error('%s %s %s', result)
86             if result == "NOT_MALICIOUS":
87                 full_url = f'{TARGET_URL}{path}'
88                 resp = requests.get(full_url, params=request.args)
89                 return Response(resp.content, status=resp.status_code, headers=dict(resp.headers))
90
91             return Response("Forbidden Request", status=400)
92         else:
93             app.logger.error('%s %s', id, "NOT_MALICIOUS")
94
95             full_url = f'{TARGET_URL}{path}'
96             resp = requests.get(full_url, params=request.args)
97             return Response(resp.content, status=resp.status_code, headers=dict(resp.headers))
98
99 app.run(debug=True, port=5050, host='0.0.0.0')
```

Abbildung 22: Code Firewall KI

A.2 Main Code Traditionelle Firewall

```
 1 import { fileLogger } from "./Logger/logger";
 2 import { checkForAttack } from "./traditional_firewall/index";
 3 import { detectSQLInjection } from "./traditional_firewall/sqlinjection";
 4 import { detectXSS } from "./traditional_firewall/xss";
 5 import { detectPathTraversal } from "./traditional_firewall/pathtraversal";
 6 import { detectCommandInjection } from "./traditional_firewall/commandInjection";
 7 const express = require("express");
 8 const proxy = require("express-http-proxy");
 9
10 var app = express();
11
12 app.use(detectSQLInjection);
13 app.use(detectCommandInjection);
14 app.use(detectPathTraversal);
15 app.use(detectXSS);
16
17 app.all(
18   "*",
19   proxy("http://localhost:3000", {
20     filter: function(req, res) {
21       var id = req.header("id");
22       var url = req.url;
23
24       logValidRequest(id, url);
25       return true;
26     },
27   }),
28 );
29
30 function logValidRequest(id, url) {
31   fileLogger.log({
32     level: "info",
33     message: "Valid request",
34     additional: [id, url],
35     are: "passed along",
36   });
37 }
38
39 app.listen(5050, () => {
40   console.log("Listening to Port 5050");
41 });
42
```

Abbildung 23: Main Code Traditionelle Firewall

A.3 Code Path Traversal Traditionelle Firewall

Abbildung 24: Code Path Traversal Traditionelle Firewall

A.4 Code SQL Injektion Traditionelle Firewall

Abbildung 25: Code SQL Injection Traditionelle Firewall

A.5 Code XSS Injection Traditionelle Firewall

Abbildung 26: Code XSS Injection Traditionelle Firewall

A.6 Code Command Injection Traditionelle Firewall

Abbildung 27: Code Command Injection Traditionelle Firewall

A.7 Code Logger Traditionelle Firewall

```
1 'use strict'  
2  
3 import * as winston from "winston"  
4  
5  
6  
7 export const fileLogger = winston.createLogger({  
8   level: 'info',  
9   format: winston.format.combine(  
10     winston.format.timestamp({ format: 'YYYY-MM-DD HH:mm:ss' }),  
11     winston.format.errors({ stack: true }),  
12     winston.format.splat(),  
13     winston.format.json()  
14   ),  
15   transports: [  
16     new winston.transports.Console(),  
17     new winston.transports.File({ filename: 'error.log', level: 'error' }),  
18     new winston.transports.File({ filename: 'log.log' }),  
19   ],  
20 });  
21  
22
```

Abbildung 28: Code Logger Traditionelle Firewall