Hochschule Karlsruhe
University of Applied Sciences

Fakultät für Informatik und Wirtschaftsinformatik
Wirtschaftsinformatik

BACHELORTHESIS

Sicherheitskonzepte für Anwendungen in der Cloud

von	Herr Julian Bücher
Matrikelnr.	61086
Arbeitsplatz	STP Informationstechnologie GmbH, Karlsruhe
Erstbetreuer	Prof. Dr. Udo Müller
Zweitbetreuer	Prof. Dr. Ingo Stengel
Abgabetermin	31.08.2021

Karlsruhe, 31.08.2021

Vorsitzender des Prüfungsausschusses

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit eigenständig und ohne fremde Hilfe angefertigt habe. Textpassagen, die wörtlich oder dem Sinn nach auf Publikationen oder Vorträgen anderer Autoren beruhen, sind als solche kenntlich gemacht. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

(Ort, Datum) (Unterschrift)

Inhaltsverzeichnis

Abkürzungsverzeichnis

API ..... *Anwendungsprogrammierschnittstelle*
AS ..... *Authorization Server*
BSI ..... *Bundesamt für Sicherheit in der Informationstechnik*
CLOUD ..... *Clarifying Lawful Overseas Use of Data*
CSP ..... *Cloud Service Provider*
DDoS ..... *Distributed-Denial-of-Service*
DoS ..... Denial-of-Service
IaaS ..... *Infrastructure-as-a-Service*
JWT ..... *JSON-Web-Token*
ML ..... *Machine Learning*
OS ..... *Operating System*
PaaS ..... *Platform-as-a-Service*
SaaS ..... *Software-as-a-Service*
SPA ..... *Single Page Applikation*

Abbildungsverzeichnis

Codeausschnitte

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

Gestützt durch Vorfälle aus jüngster Zeit ist die Sicherheit in der Informationstechnologie für Firmen auf der ganzen Welt zu einem immer mehr an Bedeutung gewinnenden Bestandteil geworden. Als Folge der aktuellen Angriffe wie zum Beispiel das unautorisierte Entwenden von mehr als 500 Millionen Nutzerdaten von Facebook [1] oder das Ausnutzen der Schwachstellen der Microsoft Exchange Server in zahlreichen Unternehmen [2] fokussieren immer mehr Firmen die IT-Sicherheit ihrer Produkte. Dies gilt nicht nur für Software wie native Applikationen auf Mobiltelefonen oder Desktopanwendungen, sondern auch für Anwendungen, die ihren Nutzern als Clouddienste zur Verfügung stehen und öffentlich über das Internet erreichbar sind. Jedoch sind Kriterien, die für die Absicherung dieser Dienste gedacht sind, wie zum Beispiel des National Institut for Standards and Technology (NIST), der Cloud Security Alliance (CSA) oder das Bundesamt für Sicherheit in der Informationstechnik (BSI) nur wenig verbreitet und werden von den betreffenden Firmen partiell umgesetzt. Des Weiteren ist festzustellen, dass sich die aufgezählten Kriterienkataloge immer wieder aufeinander beziehen, jedoch keine einheitliche Norm zur Regelung und Umsetzung der Sicherheitskriterien vorliegt. Als Richtlinie und Maßstab für zukünftige Entwicklungen an Cloudprodukten sollte jedes Unternehmen einen Katalog an bestehenden Sicherheitsmaßnahmen und Regularien entwerfen, der den Endnutzern ein gewisses Maß an Sicherheit garantiert. Im Kontext dieser Thesis wird auf Basis einer der bestehenden Kriterienkataloge innerhalb eines beispielhaften Szenarios eine Teilmenge an Kriterien selektiert. Ergebnis wird ein Bewertungskatalog speziell für dieses Szenario sein, der im Anschluss für einen Entwurf eines Sicherheitskonzeptes verwendet wird und Maßnahmen zur Optimierung des Produktes aufzeigt. Unter Verwendung dieses Mechanismus soll es dem Cloudprodukt ermöglicht werden, sich in diesem Szenario zu schützen.

1.2 Unternehmensvorstellung

Die STP Informationstechnologie GmbH (nachfolgend kurz: STP GmbH) ist ein in Karlsruhe gegründetes IT-Unternehmen. Der Schwerpunkt der angebotenen Softwarelösungen und Informationssysteme zielt auf die Anwendungen in den Berufsgruppen im Rechtssektor wie Anwälte, Justizverwaltungen und weiteren fachnahen Institutionen ab. Gegründet wurde das Unternehmen im Jahr 1993 von Gunter Thies und Ralph Suikat als „Suikat-Thies + Partner GmbH“. Im Jahre 2001 wurde die Unternehmensform in eine Aktiengesellschaft umgewandelt [3]. Die STP AG im Rahmen eines Zertifizierungsprogramms durch SGS-International Certification Services GmbH nach DIN ISO 9001 zertifiziert. Seit November 2011 gilt dieses Zertifikat für die komplette STP Gruppe: STP Informationstechnologie AG, STP Holding GmbH, STP Portal GmbH und STP Solution GmbH. Seit dem 05. März 2021 ist die STP von einer Aktiengesellschaft in eine GmbH umfirmiert [3].

Intern untergliedert sich die STP weiterhin in einzelne Abteilungen, Arbeitsbereiche und -gruppen, die mit verschiedenen Themen betraut sind. Der Schwerpunkt jeder einzelnen Abteilung liegt auf einem anderen Gebiet der Software- bzw. Produktentwicklung.

Der Fokus bei der Softwareentwicklung liegt jedoch primär auf der Umsetzung von Kundenlösungen mit dem .NET-Framework. Die Produktpalette umfasst neben On-Premise Lösungen, die lokal beim Kunden eingesetzt werden, auch Dienstleistungen wie Consulting- bzw. Beratungslösungen, die von internen Beratern bzw. Fachbearbeitern angeboten werden. Des Weiteren gehören auch Cloudlösungen, die über das Internet angeboten werden zu dem bestehenden Produktportfolio.

Der Wirkungsbereich der STP Informationstechnologie umfasst die gesamte DACH-Region. Dies bedeutet, es werden neben Kunden aus Deutschland auch Kunden aus der Schweiz, Standort der jüngsten Zweigstelle, und Österreich betreut. Die Niederlassung in Bulgarien fungiert in diesem Unternehmensverbund bisher als Zuarbeiter für spezielle Aufgaben der Entwicklung.

Das hier vorliegende Projekt wurde hauptsächlich in der Abteilung Produktentwicklung (PDE) mit Betreuung durch Manuel Naujoks durchgeführt und erarbeitet. Der Fokus dieser Abteilung liegt auf der Evaluation und Verwendung von neuen Technologien im Ökosystem .NET, die bei der Entwicklung von neuen hauseigenen Produkten verwendet werden sollen.

1.3 Grundlagen

1.3.1 Sicherheitsrelevante Begrifflichkeiten und Verfahren

1.3.1.1 Authentifizierung und Autorisierung

Aufgrund der Relevanz der Begriffe **Authentifizierung** und **Autorisierung** im Bereich der Informationssicherheit werden diese zunächst zum allgemeinen Verständnis in den Kontext eingeordnet und erläutert.

Die Definition des Begriffs der **Authentifizierung** wird anhand des nachfolgenden theoretischen Beispiels von einer Kommunikation zwischen einem menschlichen Nutzer mit einer Anwendung (Maschine) exemplarisch dargestellt.

Die Authentifizierung des Nutzers bei einer Anwendung ist in vielen Fällen der erste Schritt, wenn der Nutzer Zugriff auf geschützte Ressourcen, wie zum Beispiel die Daten seines Profils in einem sozialen Netzwerk haben möchte. Hierzu wird er vom System bzw. der Anwendung zuerst aufgefordert den Usernamen und sein Passwort (nachfolgend Zugangsdaten) einzugeben. Diesen Vorgang der Authentifizierung nutzt die Anwendung, um zu prüfen, ob es sich bei dem vorliegenden Nutzer wirklich um den Nutzer handelt, der er vorgibt zu sein. Stimmen die Zugangsdaten, die meistens via Abgleich von verschlüsselten Werten in der Datenbank überprüft werden, mit den angegebenen Werten überein, ist der Nutzer erfolgreich authentifiziert. Ist dies nicht der Fall, wird dem Nutzer der Zugriff verwehrt. Bei erfolgreicher Authentifizierung erhält der Nutzer in den meisten Fällen ein Token, der ihm als Beweis seiner Identität dient, um sich gegenüber dem System auszuweisen. Somit wird verhindert, dass bei erneuter Interaktion des Nutzers mit dem System, dieser sich erneut anmelden muss.

Bei der **Autorisierung** kommt der bereits genannte Token zum Einsatz. Möchte der Nutzer nun auf die Daten seines Profils zugreifen, sendet er neben dem Request noch seinen Token, im Header des Requests, mit. Dieser Token wird anschließend evaluiert und auf seine Gültigkeit geprüft. Stimmen die notwendigen Rollen mit denen im mitgelieferten Token überein, erhält der Nutzer den gewünschten Zugriff. Ansonsten wird ihm der Zugriff verwehrt.

Die Form und der Informationsgehalt dieser Tokens kann sich in den unterschiedlichen Authentifizierungs-Verfahren unterscheiden. In den meisten Fällen handelt es sich jedoch um einen sogenannten JSON-Web-Token (JWT), der im Bearer Schema vorliegt. Diese Art von Tokens wird zur Übermittlung von Informationen genutzt und sind in den meisten Fällen signiert und verschlüsselt.

1.3.1.2 JSON-Web-Token

Der grundlegende Aufbau des Tokens lässt sich in drei Bestandteile aufgliedern. Beginnend mit dem „Header“ oder auch Kopf, der den Typ des Tokens und den Algorithmus zur Signatur festlegt. Anschließend folgt durch einen Punkt getrennt der „Payload“, der userspezifische Informationen und die Berechtigungen des Users enthält. Als letzter Bestandteil folgt erneut durch einen Punkt getrennt die „Signature“ oder auch Signatur, mit der die Validität des Tokens überprüft werden kann.

Die gängigste Variante ist das Signieren der Tokens von Seiten des Identity Providers und eine Bereitstellung des Public Keys über einen öffentlichen Endpunkt. Mit Hilfe dieses Public Keys kann anschließend eine Client-Anwendung die Echtheit und die Validität des gesendeten Tokens überprüfen. Zugriff auf diese Informationen erhalten die Clients bzw. Anwendungen über spezielle Endpunkte, die die Identity Provider hierfür zur Verfügung stellen. Der derzeitige Standard zur Ausgabe dieser Informationen definiert eine URL in dem Format `<IdentityProviderDomain>/.well-known/openid-configuration`.

Das nachfolgende Beispiel zeigt einen JWT in seinem kodierten und dekodierten Zustand.

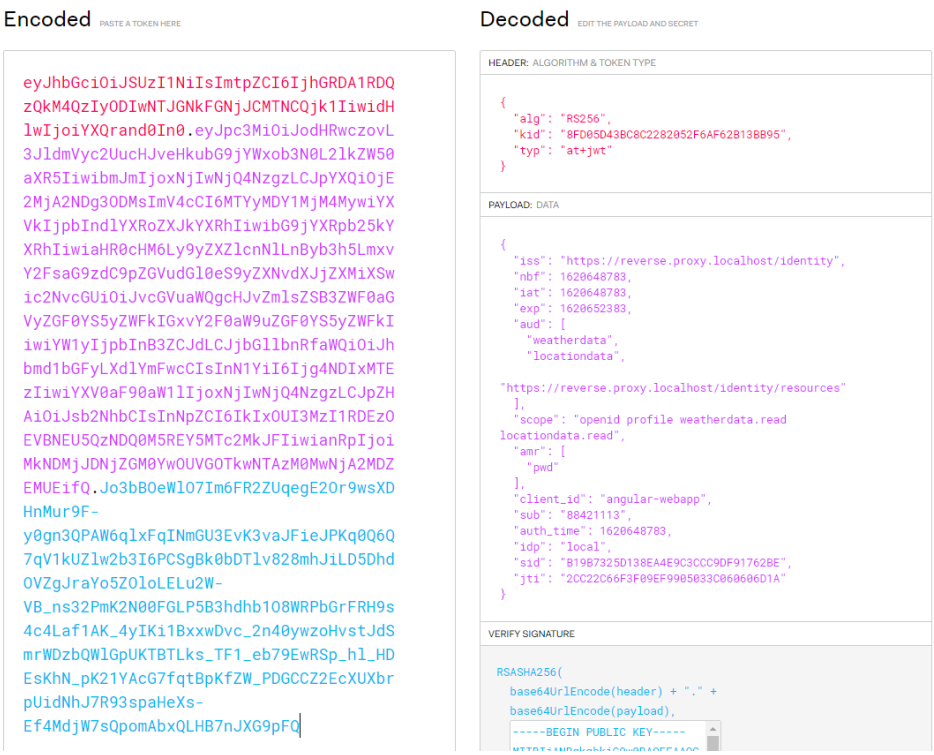

Abbildung 01 JSON-Web-Token in kodierten und dekodierten Zustand

1.3.1.3 Authorization Code Flow with Proof-Key-of-Code-Exchange

Zuerst erfolgt einer Erläuterung des "**Authorization Code Flow + Proof Key of Code Exchange (PKCE)**". Diese Authentifizierungsmethode ist derzeit der Standard zur Authentifizierung mittels Webapplikationen, wie einer SPA bei Identity Providern [4]. Der Vorgang besteht darin, dass der Client (hier eine Webapplikation) eine **Code_Challenge** an den Authorization Server (hier den Identity Provider, kurz AS genannt) schickt. Bei dieser **Code_Challenge** handelt es sich um eine Zeichenkette aus Base64-kodierten Werten. Diese Werte können Zeichen im Format von `a-z, A-Z, 0-9, . , ,,‘, ~` und - enthalten.

Zuerst wird diese Zeichenkette mittels SHA-256 verschlüsselt und an den AS geschickt. Nach Eingang der **Code_Challenge** speichert der AS den Wert ab und sendet eine **Code_Verification** zurück.

Möchte der Client nun einen Token zum Zugriff auf geschützte Bereiche des Ressource Server (kurz RS genannt), muss er neben dem ungehashten Wert der **Code_Challenge** zusätzlich die **Code_Verification** an den AS schicken, um sich zu authentifizieren [5]. Der Nachteil bei dieser Authentifizierungsvariante besteht darin, dass bei Entwendung des Tokens, dieser für die restliche Zeit seiner Gültigkeit weiterverwendet werden kann, ohne die Möglichkeit des Nutzers, dies zu verhindern.

1.3.2 Angriffsvektoren für Cloud-Applikationen

Zur besseren Einordnung von gängigen Angriffsvektoren auf cloudbasierte Softwarelösungen bzw. alle Anwendungen, die öffentlich über das Internet erreichbar sind, werden nachfolgend die im Blickpunkt dieser Thesis beleuchteten Attacken und die dazu notwendigen Vorkehrungen erläutert.

1.3.2.1 Denial-of-Service Angriff (DoS)

Der Denial-of-Service-Angriff oder auch DoS-Angriff beschreibt das Überlasten von Webanwendungen mit einer Flut von Anfragen. Ziel des Angriffs ist das vorübergehende oder gänzliche Blockieren eines Dienstes. Hieraus resultiert zum Beispiel eine längere Ladezeit für die Kunden bis hin zur Nicht-Erreichbarkeit. Diese Art von Attacken können auf unterschiedlichen Ebenen des OSI Modells ansetzen. Angriffe auf Ebene drei und vier des Netzwerkmodells haben gezielt das Überlasten von Firewalls und Load Balancer zum Ziel. Angriffe auf Ebene sieben, dem Application Layer, sind auf das Überlasten der Applikation an sich spezialisiert [6].

1.3.2.2 Bot-Netze

Bei einem Bot-Netz handelt es sich um eine Gruppe aus Computern oder Servern, die mit Schadcode infiziert und von einem zentralen Server gesteuert werden. Diese Computer bzw. Server werden auch als Bots bezeichnet und können innerhalb des Netzwerkes miteinander kommunizieren. Der Schadcode wird üblicherweise über Viren, Trojaner oder Würmer auf die Rechner gebracht und wird dort im Hintergrund ausgeführt. Bot-Netze bestehen zumeist aus mehreren tausend Rechnern und werden für die Ausübung von zum Beispiel Distributed-Denial-of-Service Angriffen verwendet [7, S. 75-76].

1.3.2.3 Distributed-Denial-of-Service Angriff

Bei einer Distributed-Denial-of-Service Attacke handelt es sich um die erweiterte Form des Denial-of-Service Angriffs. Hierbei greift der Angreifer auf die Funktionalität eines Bot-Netzes zurück, das er für das Senden der Anfragen auf das gewünschte Opfer instrumentalisiert. Mithilfe dieser Form des DoS-Angriffs umgeht der Angreifer die begrenzte Bandbreite einer Maschine und nutzt stattdessen die Rechenkapazität von mehreren verteilten Rechnern. Zusätzlich wird durch das Nutzen verschiedener Rechner eine größere Spanne an verwendeten IP-Adressen benutzt. Dies macht die Identifikation von normalem und schädlichem Traffic schwieriger [7, S. 121-123].

1.3.3 Einführung in die Cloud Taxonomie

Für die Einordung und das Verständnis der Cloud Taxonomie werden in diesem Unterabschnitt der Thesis alle notwendigen Begriffe und deren Definition eingeführt. Die Ressourcen eines Cloud Computing Dienstes reichen von Software-Diensten bis zu Datenspeichern, Betriebssystemen und ganzen Hardware-Infrastrukturen. Basierend auf dem Service-Modell des Dienstes kann in Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) und Software-as-a-Service (SaaS) unterschieden werden [8, S. 5]. Neben diesen bereits aufgeführten Service Modellen gibt es noch eine Bandbreite an weiteren Diensten, die über die Cloud angeboten werden können. Dazu zählen zum Beispiel Communication-as-a-Service (CaaS), Compute-as-a-Service (CompaaS), Data-Storage-as-a-Service (DSaaS) bzw. Database-as-a-Service (DaaS) oder Network-as-a-Service (NaaS) [9, S. 17]. In diesem Grundlagenkapitel bzw. im Rahmen der Thesis werden jedoch nur die ersteren drei Modelle genauer beschrieben. Sie sind im Rahmen der Cloud-Sicherheit am besten für die Einordnung der Verantwortung des Kunden bezüglich der Wartung und Absicherung der in Anspruch genommenen Cloud-Leistung geeignet.

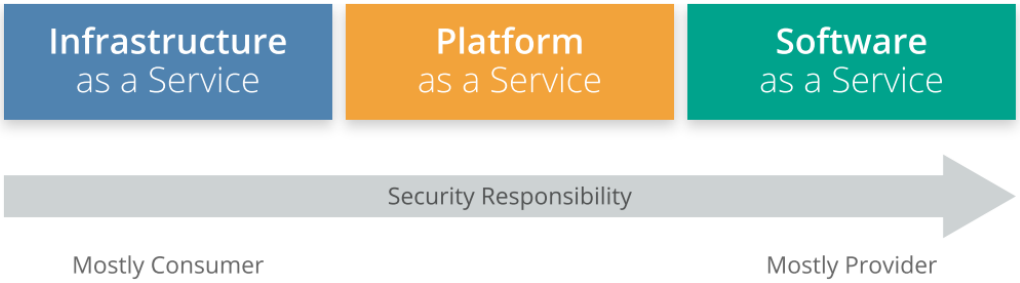

Abbildung 02 Verantwortung über die genutzten Cloud Services (Quelle: [10, S. 21])

1.3.3.1 Infrastructure-as-a-Service

Durch die Verwendung von Infrastructure-as-a-Service hat der Cloud Kunde den kompletten Zugriff auf alle Komponenten (wie z.B. Betriebssysteme, Middleware, Laufzeit, Daten und Applikationen) des gemieteten Cloud Servers. Hierauf kann über vordefinierte grafische Benutzerschnittstellen oder VPN Verbindungen zugegriffen werden [9, S. 17]. Dies garantiert neben einer Vielzahl an Konfigurationsmöglichkeiten eine große Verantwortung hinsichtlich der Wartung des Servers und den darauf ausgerollten Applikationen. Zusätzlich muss darauf geachtet werden, dass die neusten kritischen Sicherheits- und Betriebssystem-Updates installiert sind. Dies rundet die Konfiguration der Firewall, die den Server vor unbefugtem Zugriff schützt, ab.

1.3.3.2 Platform-as-a-Service

Im Rahmen von Platform-as-a-Service werden dem Cloud Kunden unterschiedliche Plattformen für das Betreiben seiner Anwendungen zur Verfügung gestellt. In diesem Rahmen kann dieser seine Applikationen mit Hilfe von vorhandenen Entwickler-Tools und Laufzeitumgebungen ohne größeren Aufwand hinsichtlich der Konfiguration platzieren [9, S. 16-17].

1.3.3.3 Software-as-a-Service

Der Cloud-Dienst Software-as-a-Service impliziert eine in der Cloud betriebene Anwendung, auf die der Kunde über einen Web-Browser zugreifen kann. Der Nutzer dieser Software hat somit den Vorteil, dass die Anwendung nicht lokal auf seinem System betrieben werden muss und somit keinerlei lokale Ressourcen verbraucht. Hiermit entfallen der Installationsprozess auf der lokalen Umgebung und der Kauf von Desktop- und Server-Lizenzen. Abgerechnet wird je nach Nutzungsdauer und der Anzahl der Nutzeraccounts, die für den jeweiligen Tenant angelegt wurden. Typische Anwendungen in diesem Service Modell sind E-Mail- und Dokumentenmanagement-Programme [9, S. 16].

Wie in der nachfolgenden Grafik nochmals genauer veranschaulicht, ist innerhalb der einzelnen Service Modelle ein Gefälle an Verantwortung zu erkennen, die der Cloud Kunde selbst übernehmen muss. Von der kompletten Verwaltung von eigenen On-Premise Systemen bis hin zur fremdverwalteten SaaS-Lösung.

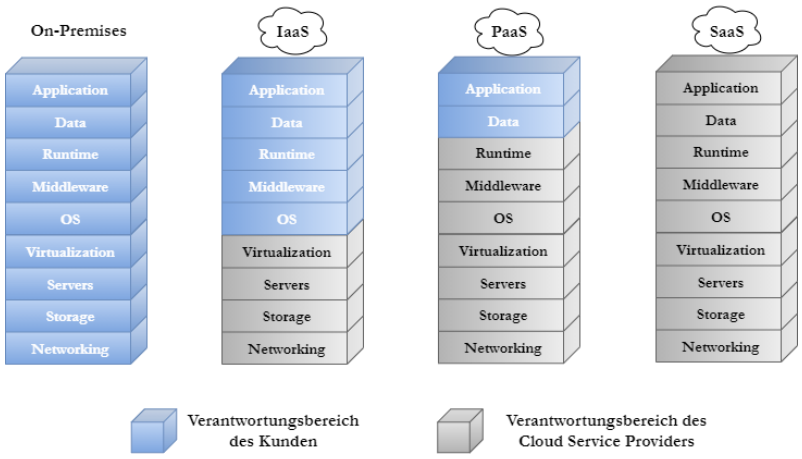

Abbildung 03 Verantwortung von Cloud Kunde und Cloud Betreiber (Quelle: Angelehnt an [9, S. 16])

1.3.3.4 Unterscheidung des verwendeten Deployment-Modells

Im nachfolgenden Unter-Unterabschnitt werden die einzelnen Deployment-Modelle, die potentiellen Cloud-Kunden für das Betreiben ihrer Anwendungen zur Verfügung stehen umschrieben und voneinander abgegrenzt.

1.3.3.4.1 Die öffentliche Cloud (Public Cloud)

Im Falle einer "Public Cloud" handelt es sich um eine geteilte Ressource, die dem Endkunden über eine öffentliche sichere IP zur Verfügung steht. Hierbei werden den Endkunden die verwendeten Leistungen des Modells nach dem Pay-as-you-go Prinzip berechnet. Verwendet der Kunde nur wenige Services innerhalb seines eigenen Produktes, fallen die Kosten meistens geringer aus, wie bei einem Kunden mit mehreren Services. Nachteile dieses Modells sind, dass durch gemeinsame Nutzung der virtuellen Maschinen von unterschiedlichen Kunden, keine Garantie auf die Sicherheit und ein Backup der Daten besteht. Hierum muss sich der Kunde selbst kümmern. Die Trennung zwischen den Verantwortungsbereichen der Kunden auf der VM basiert auf einer logischen Trennung, die in Form einer Multi-Tenant-Architektur umgesetzt wird. Typische Beispiele von Public Cloud Modellen sind die frei verfügbaren Speicher für digitale Inhalte von Providern wie Google, Amazon oder Facebook [9, S. 19].

1.3.3.4.2 Die private Cloud (Private Cloud)

Im Rahmen des Private Cloud Deployment Modells handelt es sich um ein Modell, welches innerhalb der eigenen IT-Infrastruktur des betreibenden Unternehmens ausgerollt ist. Hierbei besteht die Möglichkeit die notwendige Infrastruktur In-House zu betreiben oder an einen externen Dienstleister (Dritten) auszulagern und via VPN zuzugreifen. Bei diesem Modell wird im Vergleich zu einer Public Cloud ein höherer Grad an Sicherheit erreicht, da die hierauf betriebenen Anwendungen nicht über ein öffentliches Netzwerk zugänglich sind. Zusätzlich besitzt der Betreiber die Möglichkeit zur Wahl des Standortes für die Speicherung seiner Daten ("data at rest"). Dies ist zum Beispiel bei verteilten Public Cloud Modellen nicht möglich. Zusätzlich besteht die Option einer vorgeschalteten Firewall, die gegen gängige Angriffe aus dem Internet schützen kann [9, S. 19-20].

1.3.3.4.3 Die gemeinschaftliche Cloud (Community Cloud)

Die Eigenschaften einer Community Cloud lassen sich aus den bisher genannten Cloud Modellen der Public und der Private Cloud zusammensetzen. Hinsichtlich des Zugangs ist eine Community Cloud nur Mitgliedern der betreibenden Gemeinschaft zugänglich. Somit ist es wie bei einer Private Cloud nur einem bestimmten Nutzerkreis möglich dieses Modell zu nutzen. Die Ressourcen werden wie bei einer Public Cloud zwischen den Nutzern aufgeteilt und die Trennung der einzelnen Zuständigkeitsbereiche erfolgt nur auf einer logischen Basis. Es besteht hierbei jedoch die Möglichkeit in einem kontrollierten Rahmen Informationen zwischen den unterschiedlichen Mitgliedern auszutauschen. Dies macht dieses Modell für Unternehmen in der Gesundheitsbranche interessant. Das Verwalten der Cloud Infrastruktur kann über die Organisationen erfolgen, die die Dienste nutzen, oder über einen externen Dritten [9, S. 20].

1.3.3.4.4 Die hybride Cloud (Hybrid Cloud)

Eine Hybrid Cloud kann als Konzept für das Verwenden von mehreren einzelnen Cloud Deployment Modellen verstanden werden. Hierbei ist es möglich eine Private mit einer Public oder Community Cloud zu verbinden und die unterschiedlichen Vor- und Nachteile des jeweiligen Modells miteinander zu vereinen. Zum Beispiel wäre es möglich je nach Relevanz der verarbeiteten Daten, Daten mit hohem Sicherheitsniveau in einem privaten Teil der Cloud und Daten mit einem geringeren Niveau an Sicherheit in einem öffentlichen Teil zu speichern [9, S. 20].

1.3.4 Einführung in die ML Taxonomie

In diesem Unterabschnitt werden die im Rahmen der Thesis verwendeten Konzepte aus dem Bereich des maschinellen Lernens eingeführt und erläutert. Hierbei werden die verwendeten Techniken jedoch nicht mathematisch dargestellt, sondern nur exemplarisch für ihren Einsatzzweck in den Kontext der Arbeit eingeordnet.

1.3.4.1 Entscheidungsbäume

Bei einem Entscheidungsbaum, oder auch Decision Tree, handelt es sich um ein Verfahren des überwachten („supervised“) Lernens. Sie werden in den meisten Fällen für Regressionen oder wie im Rahmen dieser Thesis für eine binäre Klassifikation eingesetzt. Da es sich um ein überwachtes Lernverfahren handelt, müssen die Daten entsprechend vorbereitet werden, um dem Baum während des Trainings eine klare Vorstellung davon zu geben, welche Kriterien im Ergebnis ausschlaggebend sind. Ein solcher Baum besteht aus mehreren Knoten, Kanten und Blättern. In einem Knoten wird auf Basis der gelernten Daten und deren Attributen somit stets eine Ja-Nein-Frage beantwortet, die je nach Ergebnis in ein bestimmtes Blatt ableitet [11, S. 151-152]. Nachfolgend wird solch ein Baum beispielhaft dargestellt.

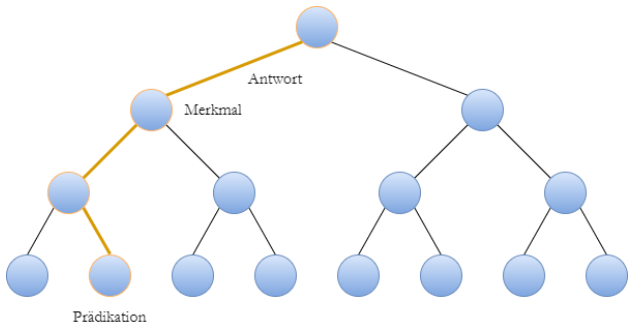

Abbildung 04 Beispielhafte Darstellung eines binären Entscheidungsbaumes (Quelle: Angelehnt an [11, S. 152])

1.3.4.2 Over-Fitting

Durch die Wahl von zu vielen Attributen ist es möglich, dass sich der Entscheidungsbaum so aufteilt, dass in den Blättern nur wenig repräsentative Daten vorhanden sind. Dies führt letztlich dazu, dass es zu einer Überanpassung (Over-Fitting) kommt. Resultat eines Over-Fitting ist, dass der Baum auf den gelernten Trainingsdaten eine gute Performance zeigt, jedoch bei den Testdaten oder neuen nicht-gelernten Daten keine aussagekräftigen Ergebnisse liefert. Zur Vermeidung von Überanpassung besteht die Möglichkeit, dass bestimmte Abbruchregeln eingeführt werden, die die Aufspaltung eines Knotens in weitere Blätter unterbinden [11, S. 161-162].

1.3.4.3 Random Forest

Aufgrund der Tatsache, dass Entscheidungsbäume häufig zu Over-Fitting neigen, hat sich in diesem Kontext eine neue Methode etabliert. Mit Decision Forests können diese Fehler durch die Verwendung von mehreren Entscheidungsbäumen in einem Modell (auch Komposition genannt) ausgeglichen werden. Innerhalb dieses Modells werden mehrere einzelne Entscheidungsbäume auf einer Untermenge des Trainingsdatensatzes trainiert und bilden im Nachhinein ein Plenum. Die Ergebnisfindung beruht abschließend darauf, dass jeder einzelne Baum ein eigenes Ergebnis präsentiert und das finale Ergebnis aus einer Mehrheitsbildung entsteht [11, S. 171-172]. Somit können Entscheidungen von überangepassten Bäumen gemittelt werden und fallen nicht mehr so stark ins Gewicht.

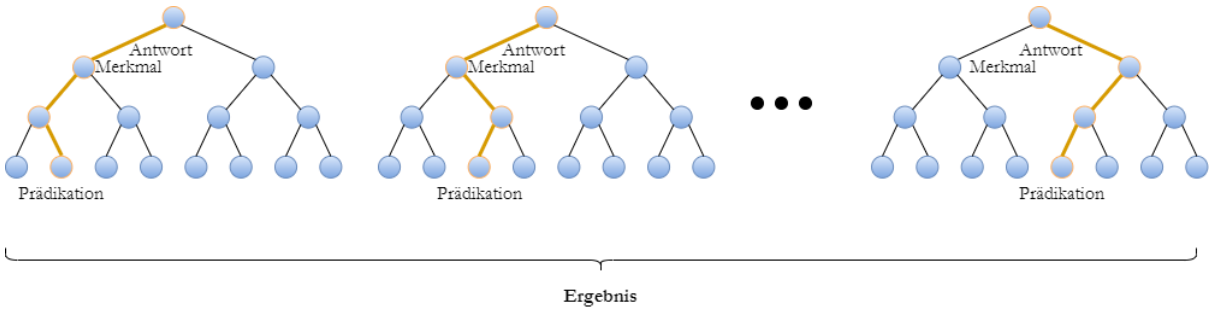

Abbildung 05 Darstellung eines Decision Forests, der aus n Bäumen besteht (Quelle: Angelehnt an [12])

2 Hauptteil

Inhalt dieses Kapitels ist die theoretische Ausarbeitung des Kriterienkataloges anhand eines vorher vorgegebenen operationalisierbaren Szenarios mit anschließender Konzeption eines Dienstes, der mit passenden Gegenmaßnahmen die negativen Effekte abmildern soll.

2.1 Beschreibung des operationalisierbaren Szenarios

Für die Erarbeitung des Kriterienkatalogs für die STP Cloud und dessen Anspruch an Vergleichbarkeit wird anhand eines der gängigsten Risiken für Anwendungen in der Cloud, eine Situation konstruiert, welche die Basis für die Auswahl der Kriterien für die Gefahrenanalyse bildet.

Die Auswahl des Angriffsvektors basiert auf der Liste der Cloud Security Alliance, die auch in der hier verwendeten Literatur [13] zur Veranschaulichung von Risiken für Cloudanwendungen herangezogen wird:

- Datenpannen
  Ausnutzen von schlecht konfigurierten Datenbanken, über die Angreifer Zugriff auf gespeicherte Kundendaten erhalten können.
- Datenverlust
  Durch Angreifer, unachtsame Service Provider oder Naturkatastrophen ausgelöste Verluste von Daten.
- Übernahme von Account oder Dienst Netzwerkverkehr (Traffic Hijacking)
  Diese Art von Risiko kann unterschiedliche Ursachen besitzen. Ein Angreifer könnte zum Beispiel durch Umleiten von Nutzern auf gefälschte Login-Seiten oder die Manipulation von Daten Zugriff auf Nutzerkonten des Dienstes erlangen und dies als Basis weiterer Angriffsvektoren verwenden.
- Unsichere APIs und Benutzeroberflächen
  Zur Verwaltung der Cloud-Infrastrukturen und deren Überwachung (auch Monitoring genannt) werden von den IT-Administratoren in den meisten Fällen APIs oder grafische Benutzeroberflächen verwendet. Falls diese ohne eine Absicherung frei aus dem öffentlichen Netzwerk adressiert werden können, ist hier eine kritische Sicherheitslücke vorhanden.

- Denial of Service (DoS) Attacken

  Im Rahmen von DoS Attacken können durch das ständige Anfragen der Dienste in der Cloud, Probleme in der Verfügbarkeit auftreten. Dies kann wiederum erhebliche Kosten für den Anbieter der Cloud-Dienste bedeuten, da viele Bezahlmodelle nach dem Pay-As-You-Go Prinzip verwaltet werden. Das bedeutet, dass durch die höhere Auslastung der virtuellen Maschinen, diese durch kostspielige Upgrades wie virtuellen Arbeitsspeicher erweitert werden, um die Lastspitzen abfangen zu können.
- Bösartige Insider

  Beschreibt das Risiko eines Angestellten des Cloud-Anbieters oder einer dritten Partei, der durch böswilliges Verhalten dem Cloud Kunden sowie dem Cloud-Anbieter Schaden zufügen kann.
- Missbrauch von Cloudinfrastrukturen (Cloud Abuse)

  Der Missbrauch von Cloudinfrastrukturen stellt das Ausnutzen der Rechenkapazität des verteilten Systems als Risiko dar. Anwendungsfälle wären zum Beispiel die Nutzung der Rechenleistung für das Knacken einer Verschlüsselung oder die Durchführung einer DoS-Attacke.
- Unbedachte Nutzung von Cloud Technologien durch Unternehmen

  Hier wird die unüberlegte Migration von Unternehmenssoftware in die Cloud thematisiert. Durch fehlendes Verständnis der Risiken, die durch eine Cloudmigration entstehen, die für zum Beispiel eine vorherige On-Premise Software nicht relevant waren, müssten sämtliche Kriterien der Cloud-Sicherheit bei einer derartigen Entscheidung miteinbezogen werden. In vielen Fällen wird dies wiederrum von Firmen nicht mit ins Kalkül gezogen.
- Sicherheitslücken auf technischer Ebene

  Dieses Risiko macht auf die Gefahren von Sicherheitslücken auf der Ebene der Software, Plattform oder Infrastruktur aufmerksam, auf der die Anwendungen des Cloud Kunden laufen. Hierdurch kann auch eine sichere Anwendung durch Kompromittierung der darunterliegenden Infrastruktur lahmgelegt werden.

Bei der Entscheidung für die Wahl des Angriffsvektors wurden die aktuellen Gegebenheiten der zu bewertenden Anwendung berücksichtigt. In Folge des Hostings der Lösung bei einem externen Anbieter und somit der Elimination von einigen der oben genannten Risiken fiel die Entscheidung auf die Denial of Service (DoS) Attacken. Dies lässt im späteren Teil der Implementierung von Gegenmaßnahmen die reine technische Behandlung des Problems, ohne das Einwirken von

dritten, wie zum Beispiel eines menschlichen Nutzers, zu (Social Engineering Attacken müssen somit nicht berücksichtigt werden).

2.2 Bewertungskriterien für die Analyse

Zur Bewertung der SaaS-Lösung wird anhand der Bewertungskriterien des Cloud Computing Compliance Criteria Catalogue des Bundesamtes für Sicherheit in der Informationstechnik C5:2020, der am 21.01.2020 neu aufgelegt wurde, ein Rahmen für das zugrundeliegende Szenario entworfen. Aufgrund seines Umfangs und der Komposition der Kriterien aus dem ISO/IEC 27001 und der Controls Matrix, entwickelt von der Cloud Security Alliance (CSA), stellt der C5 eine fundierte Grundlage für eine Selektion der nachfolgenden Kriterien dar [14, S. 11]. Hierfür wird der Kriterienkatalog zuerst auf die wichtigsten Bewertungskriterien heruntergebrochen. Der komplette Katalog besteht aus 17 unterschiedlichen Bereichen, die wiederum aus 121 Basiskriterien bestehen. Neben der Sicherheit von Softwareanwendungen, der Umsetzung von Sicherheitsstandards in den zu prüfenden Unternehmen werden auch allgemeine Vorkehrungen zur Absicherung bestimmter Bereiche innerhalb von Rechenzentren bewertet. Zu der Bearbeitung der hier vorliegenden Fragestellung ist dieses Sammelwerk zu umfangreich. Für die Bewertung der vorhandenen Sicherheitsmechanismen des Systems müssen einzelne Punkte gestrichen werden. Zur besseren Übersicht der abgedeckten Bereiche innerhalb des C5 werden diese nachfolgend aufgeführt und mit einer groben Definition ihrer Bewertungsgrundlage beschrieben. Die komplett gestrichenen bzw. gekürzten Bestandteile des Katalogs werden zusätzlich jeweils mit einer Begründung versehen, die den Grund für das Nichtverwenden innerhalb des eigenen Kriterienkatalogs aufzeigt.

2.2.1 Organisation der Informationssicherheit (OIS)

Der Bereich Organisation der Informationssicherheit (OIS) beschäftigt sich mit dem Ziel der "Planung, Umsetzung, Aufrechterhaltung und (der) kontinuierliche(n) Verbesserung eines Rahmenwerks zur Informationssicherheit innerhalb der (zu bewertenden) Organisation" [14, S. 16].

Bezüglich des anzuwendenden Szenarios ergibt sich für die sieben Basiskriterien folgendes Ergebnis:

Tabelle 01 Anwendbare Kriterien aus dem Bereich OIS

Anwendbare Kriterien	Nicht-anwendbare Kriterien	
• OIS-05	• OIS-01 • OIS-02 • OIS-03	• OIS-04 • OIS-06 • OIS-07

Die hier formulierten Kriterien bewerten, wie in der Definition bereits angedeutet, die Umsetzung eines Rahmenwerkes, also eines dokumentierten Prozesses innerhalb des Unternehmens, der den Umgang mit Sicherheitsrisiken beschreibt und mögliche Handlungsanweisungen an die betreffenden Personen überträgt. Sie sind somit außer des Kriteriums OIS-05, das den Umgang mit aktuellen Bedrohungen für Dienste in der Cloud beschreibt, nicht für den Einsatz innerhalb des Szenarios zu verwenden.

2.2.2 Sicherheitsrichtlinien und Arbeitsanweisungen (SP)

Im Rahmen der Sicherheitsrichtlinien und Arbeitsanweisungen (SP) steht im Mittelpunkt das "Bereitstellen von Richtlinien und Anweisungen bzgl. des Sicherheitsanspruchs und zur Unterstützung der geschäftlichen Anforderungen" [14, S. 16].

In Anlehnung an die Argumentation aus dem Bereich Organisation der Informationssicherheit (OIS) ist nach der Evaluation der Sicherheitsrichtlinien und Arbeitsanweisungen (SP) folgendes Resultat entstanden:

Tabelle 02 Anwendbare Kriterien aus dem Bereich SP

Anwendbare Kriterien	Nicht-anwendbare Kriterien
	• SP-01 • SP-02 • SP-03

Innerhalb dieser Kriterien wird Bezug auf die Festlegungen aus OIS genommen. Durch den Ausschluss eines großen Teils von OIS ist somit ein Einsatz von Kriterien aus dem Bereich SP nicht ohne weiteres möglich.

2.2.3 Personal (HR)

Das Personal (HR) verfolgt die Zielsetzung des "Sicherstellen(s), dass Mitarbeiter ihre Aufgaben verstehen, sich ihrer Verantwortung in Bezug auf Informationssicherheit bewusst sind und die Assets der Organisation bei Änderung der Aufgaben oder Beendigung geschützt werden" [14, S. 16].

Aufgrund des gewählten Szenarios, welches eine reine technische Vorkehrung zur Prävention von Angriffen behandelt, sind auch diese Kriterien nicht für eine Anwendung geeignet:

Tabelle 03 Anwendbare Kriterien aus dem Bereich HR

Anwendbare Kriterien	Nicht-anwendbare Kriterien	
	• HR-01 • HR-02 • HR-03	• HR-04 • HR-05 • HR-06

2.2.4 Asset Management (AM)

Die Zielsetzung des Asset Management (AM) verfolgt das "Identifizieren der organisationseigenen Assets gewährleisten und ein angemessenes Schutzniveau über deren gesamten Lebenszyklus sicherstellen" [14, S. 16].

Als Asset werden in diesem Kontext Objekte bezeichnet, die "während der Erstellung, Verarbeitung, Speicherung, Übermittlung, Löschung oder Zerstörung von Informationen benötigten Objekte im Verantwortungsbereich des Cloud-Anbieters, z.B. Firewalls, Loadbalancer, Webserver, Anwendungsserver und Datenbankserver." [14, S. 50] Hierbei kann nochmals in Hardware- und Software-Objekte unterteilt werden. Hardware-Objekte sind demnach physische und virtuelle Ressourcen, wie zum Beispiel Server, und Software-Objekte beschreiben Hypervisor, Container und Datenbanken [14, S. 50]. Da es sich bei dem gewählten Szenario um einen Angriff handelt, der speziell die Software-Objekte versiert, sind somit Kriterien aus dem Bereich AM eine gute Möglichkeit zur Bewertung der vorhandenen Konzepte.

Tabelle 04 Anwendbare Kriterien aus dem Bereich AM

Anwendbare Kriterien	Nicht-anwendbare Kriterien
• AM-01 • AM-02 • AM-06	• AM-03 • AM-04 • AM-05

2.2.5 Physische Sicherheit (PS)

Kernthema des Bereichs Nummer 5 Physische Sicherheit (PS) ist das "Verhindern von unberechtigtem physischen Zutritt und Schutz vor Diebstahl, Schaden, Verlust und Ausfall des Betriebs" [14, S. 16].

Aufgrund der „Carve-Out Methode“ [15, S. 4], durch die der Anbieter der Infrastruktur aus der Bewertung durch den kondensierten Katalog herausfällt, sind die Kriterien aus diesem Bereich nicht relevant. Sie behandeln die Sicherheitsvorkehrungen innerhalb des Gebäudes des Rechenzentrums und sind somit nicht Teil der Mechanismen für das SaaS-Produkt.

Tabelle 05 Anwendbare Kriterien aus dem Bereich PS

Anwendbare Kriterien	Nicht-anwendbare Kriterien	
	• PS-01 • PS-02 • PS-03 • PS-04	• PS-05 • PS-06 • PS-07

2.2.6 Regelbetrieb (OPS)

Innerhalb des Regelbetriebs (OPS) steht das "Sicherstellen eines ordnungsgemäßen Regelbetriebs einschließlich angemessener Maßnahmen für Planung und Überwachung der Kapazität, Schutz vor Schadprogrammen, Protokollierung und Überwachung von Ereignissen sowie den Umgang mit Schwachstellen, Störungen und Fehlern" im Mittelpunkt der Kontrolle [14, S. 16].

In diesem Szenario lässt sich zwischen Zuständigkeiten des SaaS-Anbieters und des Infrastruktur-Anbieters eine klare Linie ziehen. Aspekte wie die Sicherung und Wiederherstellung von Daten und die Härtung der verwendeten Komponenten, sind größtenteils Bestandteil des Hosting-Angebots und gehören somit in den Zuständigkeitsbereich des Infrastruktur-Anbieters. Jedoch besteht der Regelbetrieb auch aus Themen wie der Protokollierung und Überwachung von Schwachstellen und die erforderliche Prüfung. Somit kann eine Teilmenge der Regularien für die Bewertung verwendet werden.

Dies ermöglicht eine Verwendung eines Großteils der Analysekriterien.

Tabelle 06 Anwendbare Kriterien aus dem Bereich OPS

Anwendbare Kriterien		Nicht-anwendbare Kriterien	
• OPS-10 • OPS-11 • OPS-12 • OPS-13 • OPS-14 • OPS-15 • OPS-16	• OPS-17 • OPS-18 • OPS-19 • OPS-20 • OPS-21 • OPS-22	• OPS-01 • OPS-02 • OPS-03 • OPS-04 • OPS-05 • OPS-06	• OPS-07 • OPS-08 • OPS-09 • OPS-23 • OPS-24

2.2.7 Identitäts- und Berechtigungsmanagement (IDM)

Mit Hilfe von Identitäts- und Berechtigungsmanagement (IDM) wird das "Absichern der Autorisierung und Authentifizierung von Benutzern des Cloud-Anbieters (in der Regel privilegierte Benutzer) zur Verhinderung von unberechtigten Zugriffen" gewährleistet [14, S. 16].

Dies spielt als Basis für die Handhabung von unautorisierten Zugriffen auf die Funktionalitäten des restlichen Systems eine bedeutende Rolle bei der Bewertung. Falls durch gezieltes Ausschalten das Rechtevergabesystem nicht mehr funktionieren sollte, könnte ein Angreifer das gesamte System außer Gefecht setzen. Aus diesem Grund sind alle Basiskriterien in die Grundlage für die spätere Evaluation miteingeflossen.

Tabelle 07 Anwendbare Kriterien aus dem Bereich IDM

Anwendbare Kriterien		Nicht-anwendbare Kriterien
• IDM-01 • IDM-02 • IDM-03 • IDM-04 • IDM-05	• IDM-06 • IDM-07 • IDM-08 • IDM-09	

2.2.8 Kryptographie und Schlüsselmanagement (CRY)

Durch Kryptographie und Schlüsselmanagement wird das "Sicherstellen eines angemessenen und wirksamen Gebrauchs von Kryptographie zum Schutz der Vertraulichkeit, Authentizität oder Integrität von Informationen" gewährleistet [14, S. 16].

Die Gewährleistung der sicheren Übertragung von Daten spielt in diesem Szenario jedoch keine tragende Rolle und somit sind diese Bewertungskriterien nicht für die weitere Verwendung geeignet.

Tabelle 08 Anwendbare Kriterien aus dem Bereich CRY

Anwendbare Kriterien	Nicht-anwendbare Kriterien	
	• CRY-01 • CRY-02 • CRY-03	• CRY-04

2.2.9 Kommunikationssicherheit (COS)

Der Bereich der Kommunikationssicherheit (COS) widmet sich dem "Sicherstellen des Schutzes von Informationen in Netzen und den entsprechenden informationsverarbeitenden Systemen" [14, S. 17].

Neben den reinen Sicherheitsvorkehrungen innerhalb der Netze des Cloud-Anbieters werden hier auch technische Schutzmaßnahmen beschrieben, die z.B. im Fall von COS-01 Vorkehrungen zum Schutz vor Distributed-Denial-of-Service Attacken beschreiben.

Tabelle 09 Anwendbare Kriterien aus dem Bereich COS

Anwendbare Kriterien	Nicht-anwendbare Kriterien	
• COS-01	• COS-02 • COS-03 • COS-04 • COS-05	• COS-06 • COS-07 • COS-08

2.2.10 Portabilität und Interoperabilität (PI)

Das "Ermöglichen der Eigenschaft, den Cloud-Dienst über andere Cloud-Dienste oder IT-Systemen der Cloud-Kunden ansprechen zu können, die gespeicherten Daten bei Beendigung des Auftragsverhältnisses zu beziehen und beim Cloud-Anbieter sicher zu löschen" [14, S. 17] wird im Rahmen der Portabilität und Interoperabilität betrachtet.

Zusammenfassend handelt es sich hierbei um eine reine Dokumentation der vorhandenen Schnittstellen und die sichere Bereitstellung und Löschung von Daten. Für den Anwendungsfall der DoS-Angriffe sind diese Bewertungskriterien nicht relevant.

Tabelle 10 Anwendbare Kriterien aus dem Bereich PI

Anwendbare Kriterien	Nicht-anwendbare Kriterien
	• PI-01 • PI-02 • PI-03

2.2.11 Beschaffung, Entwicklung und Änderung von Informationssystemen (DEV)

Die Zielsetzung im Bereich der Beschaffung, Entwicklung und Änderung von Informationssystemen (DEV) ist das "Sicherstellen der Informationssicherheit im Entwicklungszyklus von Systemkomponenten des Cloud-Dienstes" [14, S. 17].

Die in diesem Kontext beschriebenen Kriterien erlauben die Qualität der Entwicklung der einzelnen Dienste zu bewerten. Letztlich wird geprüft, ob die entwickelten Features sicher gegenüber variablen Eingaben sind. Hier spielen zum Beispiel Angriffe wie die SQL-Injection eine tragende Rolle. Werden diese bösartigen Code-Schnipsel nicht vom System erkannt, bzw. als Plain-Text behandelt, ist es dem Angreifer möglich Daten in der Datenbank zu manipulieren, auszulesen oder im extremsten Fall zu löschen.

Es werden jedoch keine Entwicklungsarbeiten an dem SaaS-Produkt ausgelagert. Aus diesem Grund entfällt das Basiskriterium DEV-02.

Tabelle 11 Anwendbare Kriterien aus dem Bereich DEV

Anwendbare Kriterien		Nicht-anwendbare Kriterien
• DEV-01 • DEV-03 • DEV-04 • DEV-05	• DEV-06 • DEV-07 • DEV-08 • DEV-09 • DEV-10	• DEV-02

2.2.12 Steuerung und Überwachung von Dienstleistern und Lieferanten (SSO)

Die Steuerung und Überwachung von Dienstleistern und Lieferanten (SSO) konzentriert sich auf das "Sicherstellen des Schutzes von Informationen, auf die Dienstleister bzw. Lieferanten des Cloud-Anbieters (Subdienstleister) zugreifen können, sowie Überwachung der vereinbarten Leistungen und Sicherheitsanforderungen" [14, S. 17].

Die Überwachung und Bewertung von Dienstleistern, die zusätzliche Dienste zur Verfügung stellen und deren Überwachung, ist aufgrund des Carve-Out [15, S. 4] aller umliegenden Organisationen außer dem Anbieter der SaaS-Lösung ausgeschlossen. Diese Kriterien sind somit kein Bestandteil der Bewertung.

Tabelle 12 Anwendbare Kriterien aus dem Bereich SSO

Anwendbare Kriterien	Nicht-anwendbare Kriterien	
	• SSO-01 • SSO-02 • SSO-03	• SSO-04 • SSO-05

2.2.13 Umgang mit Sicherheitsvorfällen (SIM)

Mit dem "Gewährleisten eines konsistenten und umfassenden Vorgehens zur Erfassung, Bewertung, Kommunikation und Behandlung von Sicherheitsvorfällen" [14, S. 17] befasst sich der Bereich mit der Kennung „Umgang mit Sicherheitsvorfällen“.

Für die Implementierung einer allgemeinen Richtlinie für den Umgang mit Gefahren und Sicherheitsvorfällen bezüglich des Cloudproduktes sind diese Kriterien für eine Verwendung bei der Analyse geeignet. Sie ermöglichen Prozesse zu implementieren, die aus vergangenen Vorfällen neue Richtlinien für den Schutz des Produktes evaluieren.

Tabelle 13 Anwendbare Kriterien aus dem Bereich SIM

Anwendbare Kriterien	Nicht-anwendbare Kriterien
• SIM-01 • SIM-02 • SIM-03 • SIM-04 • SIM-05	

2.2.14 Kontinuität des Geschäftsbetriebs und Notfallmanagement (BCM)

Als Resultat der Domäne Kontinuität des Geschäftsbetriebs und Notfallmanagement steht das "Planen, Implementieren, Aufrechterhalten und Testen von Verfahren und Maßnahmen zur Kontinuität des Geschäftsbetriebs und für das Notfallmanagement" [14, S. 17].

Dies beschreibt somit im Groben die Handhabung von Sicherheitsvorfällen innerhalb des Rechenzentrums, die die Verfügbarkeit der Dienste beeinträchtigen könnten. Die Prävention dieser Gefahren liegt im Zuständigkeitsbereich des Infrastrukturanbieters.

Tabelle 14 Anwendbare Kriterien aus dem Bereich BCM

Anwendbare Kriterien	Nicht-anwendbare Kriterien
	• BCM-01 • BCM-02 • BCM-03 • BCM-04

2.2.15 Compliance (COM)

Innerhalb der Compliance (COM) steht das "Vermeiden von Verstößen gegen gesetzliche, regulatorische, selbstauferlegte oder vertragliche Anforderungen zur Informationssicherheit und [dem] Überprüfen der Einhaltung" [14, S. 17] im Mittelpunkt.

Dies betrifft größtenteils die Compliance bzw. die Einhaltung von Richtlinien im Informationsmanagementsystem (ISMS) aus dem Bereich OIS. Aus diesem Grund ist auch dieser Bereich nicht weiter für die nachfolgende Bewertung von Bedeutung.

Tabelle 15 Anwendbare Kriterien aus dem Bereich COM

Anwendbare Kriterien	Nicht-anwendbare Kriterien
	• COM-01 • COM-02 • COM-03 • COM-04

2.2.16 Umgang mit Ermittlungsanfragen staatlicher Stellen (INQ)

Das "Gewährleisten eines angemessenen Umgangs mit Ermittlungsanfragen staatlicher Stellen hinsichtlich juristischer Überprüfung, Information der Cloud-Kunden und Begrenzung des Zugriffs auf oder der Offenlegung von Daten" [14, S. 17] ist die Zielsetzung des Zuständigkeitsbereichs von Umgang mit Ermittlungsanfragen staatlicher Stellen.

Rechtliche Fragestellungen wie den Zugriff von staatlichen Stellen auf Daten von Cloud-Kunden werden in diesem Szenario nicht betrachtet. Aufgrund der Beheimatung der Dienste und deren Daten im Land des Kunden, sind Zugriffe von regierungsnahen Institutionen wie z.B. in Amerika in Folge des CLOUD Acts nicht in gleichem Maße zutreffend.

Tabelle 16 Anwendbare Kriterien aus dem Bereich INQ

Anwendbare Kriterien	Nicht-anwendbare Kriterien
	• INQ-01 • INQ-02 • INQ-03 • INQ-04

2.2.17 Produktsicherheit (PSS)

"Bereitstellen aktueller Informationen zur sicheren Konfiguration und über bekannte Schwachstellen des Cloud-Dienstes für Cloud-Kunden, geeigneter Mechanismen zur Fehlerbehandlung und Protokollierung sowie zur Authentisierung und Autorisierung von Benutzern der Cloud-Kunden" [14, S. 17] wird durch den letzten Bereich, die Produktsicherheit (PSS), garantiert.

Im Rahmen des letzten Bereiches, der Produktsicherheit, werden nochmals wichtige Kriterien für die Absicherung von Cloud-Diensten zusammengefasst, die über das ausgewählte Szenario hinaus Möglichkeiten für den sicheren Zugang der Cloud-Kunden sorgen.

Tabelle 17 Anwendbare Kriterien aus dem Bereich PSS

Anwendbare Kriterien		Nicht-anwendbare Kriterien
• PSS-01 • PSS-02 • PSS-03 • PSS-04 • PSS-05	• PSS-06 • PSS-07 • PSS-08 • PSS-09	• PSS-10 • PSS-11 • PSS-12

2.3 Kriterienkatalog für die STP Cloud

Auf Basis der vorangegangenen Untersuchung des Cloud Computing Compliance Criteria Catalogue – C5:2020 auf Verwendbarkeit der einzelnen Basiskriterien für die Beurteilung der Sicherheitsmechanismen im Falle des im Abschnitt 2.1 ausgewählten Szenarios, konnten 50 Basiskriterien aus den 17 unterschiedlichen Bereichen isoliert werden.

Die nachfolgende Tabelle dient zur Beurteilung und zeigt den Grad der Erfüllung der einzelnen Kriterien nach Einschätzung bzw. der Analyse im Rahmen dieser Thesis. Die Ergebnisse der Analyse wurden durch Recherche von internen Dokumenten, Befragung von qualifiziertem Personal, hier der „Lenkungskreis Sicherheit", der im Rahmen der Entwicklung des Cloud-Produktes dessen Sicherheit überwacht, und eigenen Untersuchungen der Anwendung gewonnen.

Tabelle 18 Bewertungskriterien und Erfüllungsgrad für die Bewertung der STP Cloud

Beurteilungskatalog für die Erfüllung der ausgewählten Basiskriterien für ein vordefiniertes Szenario			
Szenario	Denial-of-Service Attacke		
Anzahl an selektierten Basiskriterien	50		
	Erfüllung durch die SaaS-Lösung und deren Anbieter		
Ref.	Erfüllt	Teilweise erfüllt	Nicht erfüllt
OIS-05	✗	✓	✗
AM-01	✗	✓	✗
AM-02	✗	✓	✗
AM-06	✗	✗	✓
OPS-10	✗	✓	✗
OPS-11	✗	✓	✗
OPS-12	✓	✗	✗
OPS-13	✗	✗	✓
OPS-14	✓	✗	✗
OPS-15	✓	✗	✗
OPS-16	✓	✗	✗
OPS-17	✓	✗	✗
OPS-18	✗	✗	✓
OPS-19	✗	✗	✓
OPS-20	✗	✗	✓
OPS-21	✓	✗	✗
OPS-22	✗	✗	✓
IDM-01	✗	✓	✗
IDM-02	✓	✗	✗

IDM-03	✗	✗	✓
IDM-04	✗	✓	✗
IDM-05	✗	✗	✓
IDM-06	✗	✓	✗
IDM-07	✗	✗	✓
IDM-08	✗	✓	✗
IDM-09	✓	✗	✗
COS-01	✗	✗	✓
DEV-01	✓	✗	✗
DEV-03	✗	✗	✓
DEV-04	✗	✗	✓
DEV-05	✗	✗	✓
DEV-06	✗	✓	✗
DEV-07	✓	✗	✗
DEV-08	✓	✗	✗
DEV-09	✓	✗	✗
DEV-10	✓	✗	✗
SIM-01	✗	✓	✗
SIM-02	✓	✗	✗
SIM-03	✗	✗	✓
SIM-04	✓	✗	✗
SIM-05	✗	✗	✓
PSS-01	✗	✗	✓
PSS-02	✗	✓	✗
PSS-03	✗	✗	✓
PSS-04	✗	✓	✗
PSS-05	✓	✗	✗
PSS-06	✓	✗	✗
PSS-07	✓	✗	✗
PSS-08	✓	✗	✗
PSS-09	✗	✓	✗
Σ	19	14	17

Die genauen Definitionen der Basiskriterien und die Begründung für die Erfüllung der Inhalte liegen in einem umfassenden Bewertungsdokument und einem Interviewbogen zur Klärung offener Fragen im Anhang der Thesis bei. Des Weiteren werden die Eigenschaften aufgezählt, die von der Anwendung und dem Anbieter nicht programmatisch umgesetzt oder in Form von dokumentierten Prozessen oder Richtlinien dem Kunden für den Umgang mit der SaaS-Lösung zur Verfügung gestellt werden.

Im nachfolgenden Schaubild wird zur Illustration die prozentuale Verteilung des Ergebnisses nochmals mit Hilfe eines Diagramms grafisch dargestellt.

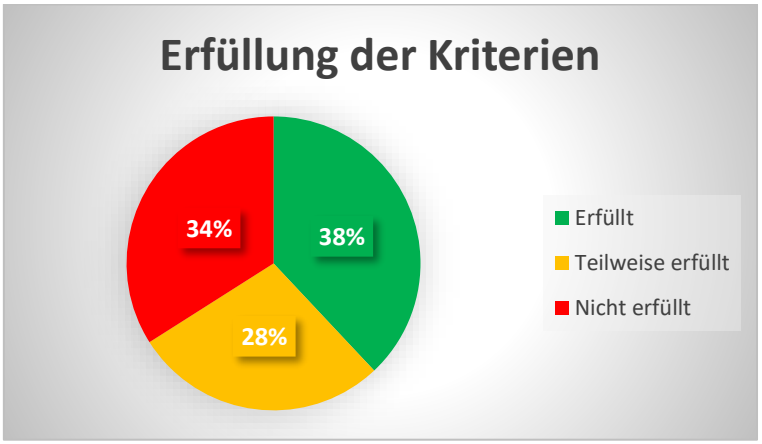

Abbildung 06 Grafische Darstellung mit prozentualer Verteilung des Erfüllungsgrades

2.4 Maßnahmen zur Optimierung des Ergebnisses

Basierend auf der Bewertung durch die zusammengestellten Basiskriterien aus dem Abschnitt 2.3 sind Lücken bzw. unvollständige Konzepte im Rahmen der C5-Kriterien für das Betreiben der Cloud-Anwendung aufgetreten. Diese Möglichkeiten zur Optimierung des Erfüllungsgrades des Kriterienkatalogs werden mit Hilfe der nachfolgend formulierten Maßnahmen adressiert.

Die Ergänzungen zur Vervollständigung der teilweise bis nicht erfüllten Kriterien orientieren sich hierbei eng an den Vorgaben aus dem C5 und stellen somit einen, nach Ansicht des BSI, Standard zur Umsetzung von Sicherheitskonzepten und Prozessen innerhalb des bewerteten Unternehmens bzw. der Anwendung dar. Hierbei liegt die Auslegung der einzelnen Maßnahmen und die Beispiele für eine Einsatzmöglichkeit bestimmter Werkzeuge auf den Annahmen und Empfehlungen des Autors und erhebt somit keinen Anspruch auf Vollständigkeit. Simultan gilt dies für den Umfang der vorangegangenen Analyse. Zur besseren Einordnung der einzelnen Maßnahmen in die jeweilig vordefinierten Bereiche, werden diese nach adressierter Thematik geordnet im Anschluss aufgeführt.

2.4.1 Organisation der Informationssicherheit (OIS)

Die aus dem Bereich OIS stammenden Kriterien (siehe OIS-05) empfehlen eine Etablierung eines (direkten) Kontaktes zu Organisationen, die sich mit der Problematik von Sicherheit für Anwendungen in der Cloud beschäftigen. Über diesen Kontakt können zukünftig aktuelle Informationen bezüglich Schwachstellen und Gefährdungen hinsichtlich der verwendeten Produkte und Technologien schneller ausgetauscht und somit in den aktuellen Entwicklungsprozess der Anwendung frühzeitig miteinfließen.

2.4.2 Asset Management (AM)

Im Rahmen der Bewertung hinsichtlich des Asset Management sind Dokumentationen der virtuellen Objekte wie z.B. der vorhandenen Microservices, aus denen die Anwendung besteht, zu führen. Darunter zählt auch eine Risikoeinschätzung dieser Güter. Solch eine Einschätzung ist nach Vorgaben des BSI auf der Brisanz der Daten, die von den Diensten in ihrer Funktion im System bearbeitet, gespeichert oder transferiert werden (siehe AM-01, AM-06) durchzuführen. Auf Basis der Risikoeinschätzung sind eine Klassifizierung und Kennzeichnung dieser Service-Objekte möglich, die nach dem Schutzbedarf der verarbeiteten Informationen die Objekte in unterschiedliche vorabdefinierte Schutzstufen einteilen. Durch Definition der Schutzstufen und Schutzbedarfe sind anschließend auch Themen wie die sichere Konfiguration der Anwendungen und den Umgang mit Anforderungen an Software- und Image-Versionen (hinsichtlich der Verwendung von Docker-Containern), sowie deren Aktualisierungen möglich (siehe AM-02).

2.4.3 Regelbetrieb (OPS)

Im Zusammenhang mit der Bewertung der OPS-Kriterien sind die nach dem BSI definierten Maßnahmen an einigen Stellen fraglich hinsichtlich ihres Einsatzes zur Steigerung der Sicherheit in der SaaS-Lösung. Darum werden Bemühungen im Hinblick auf die Vorgaben für das Aktivieren, Stoppen oder Pausieren der Protokollierung und dem Bereitstellen von Metadaten an den Cloud-Kunden nicht als sinnvoll erachtet. Im Kontext der Bewertung dieses Produktes wäre ein Stoppen der Aufzeichnungen der Aktivitäten in der Anwendung kontraproduktiv. Dies würde sich wiederum negativ in Bezug auf das Erkennen von Anomalien in den Anfragen und Protokolldaten auswirken und es würde eine Art „Black-Box“ entstehen. Aus diesem Grund werden beide Optimierungsmöglichkeiten nicht weiter von Seiten dieser Arbeit als mögliche Vorschläge betrachtet (siehe OPS-10, OPS-11).

Hingegen sind Konzepte wie ein automatisches Meldesystem von ungewöhnlichen Ereignissen bzw. Anomalien und die Weiterleitung von identifizierten Ereignissen an zuständiges Personal (siehe OPS-13) nicht im Rahmen der Anwendung vorhanden. Diesbezüglich sind weitere Maßnahmen wie Endpunkte (API's) für forensische Analysen von protokollierten Ereignissen (siehe OPS-15) Empfehlungen, die umgesetzt werden können.

Des Weiteren sind Dokumentationen für Prozesse und Anweisungen zur regelmäßigen (monatlichen) Identifikation von Schwachstellen, deren Beurteilung und Priorisierung für den späteren Entwicklungsprozess einen praktischen Leitfaden darstellen, eine durchaus zweckdienliche Grundlage für den weiteren Entwicklungsprozess. Mithilfe von Tools wie zum Beispiel dem OWASP ZAP Proxy [16] sind automatische Analysen von Web-Applikationen, wie

dieser SaaS-Lösung, auf bekannte Schwachstellen von Seiten der Web-Frameworks möglich. ZAP lässt sich entweder als eigenständige Anwendung in Form eines Docker-Containers in einer bestehenden Anwendung integrieren oder stellt ergänzend unterschiedliche APIs für die Verwendung in verschiedenen Programmiersprachen, u.a. C#, zur Verfügung (siehe OPS-18, OPS-22).

Ergänzend zu den automatisierten Analysen sind Penetrationstests von qualifiziertem internem oder externem Personal möglich, deren Ergebnisse und die daraus folgenden Nachbesserungen in definierten Zeiträumen durchzuführen sind (siehe OPS-19). Durch Etablieren eines Prozesses zur regelmäßigen Messung, Analyse und Bewertung der Verfahren zum Umgang mit Schwachstellen und Störungen ermöglicht dies in Kombination mit den automatischen Analysen einen Verfahrenskatalog, der die Aktualität der Anwendung hinsichtlich ihrer Sicherheitskonzepte kontinuierlich verbessert (siehe OPS-20).

2.4.4 Identitäts- und Berechtigungsmanagement (IDM)

Im Zuge des Rollen- und Rechtekonzeptes, sowohl Richtlinien zur Verwaltung von Zugangs- und Zugriffsberechtigungen für interne und externe Mitarbeiter des Cloud-Anbieters als auch Systemkomponenten (Services) im Autorisierungsprozess, sind basierend auf dessen Geschäfts- und Sicherheitsanforderungen Dokumentationen zur Vergabe von Benutzernamen, Zugangs- und Zugriffsberechtigungen und Funktionstrennungen empfehlenswert. Eine Komplettierung der bereits bestehenden Konzepte im weiteren Verlauf des Entwicklungsprozesses stellt für die Entwickler eine zusätzliche Möglichkeit zur Kontrolle der Rechtevergabe an maschinelle Clients dar (siehe IDM-01). Diese können auch Funktionen beschreiben, die zur Sperrung eines Nutzers bei Inaktivität oder mehrfach fehlgeschlagenen Anmeldeversuchen führen (siehe IDM-03). Zusätzlich wären Mechanismen denkbar, die Geo-Blocking ermöglichen. Somit besteht die Möglichkeit, im Voraus Zugriffe aus bestimmten Ländern und Regionen, die für die Verwendung des Cloud-Diensts nicht gedacht sind, zu verweigern. Ein Angreifer müsste in diesem Fall sich zuerst über ein VPN oder Ähnliches Zugang in ein ausländisches Netz für einen gezielten Angriff auf dieses Produkt verschaffen.

Bezüglich der Prozesse zur Überwachung der Nutzerrechte nach Veränderung des Aufgabengebietes oder im Allgemeinen der regelmäßigen Überprüfung der zugeordneten Rechte, sind Konzepte zur Einführung dieser Prozesse empfohlen. Hiermit soll durch qualifiziertes Personal eine Rechteprüfung in regelmäßigen Abständen stattfinden. Bei Änderung von Aufgabengebieten z.B. von privilegierten Nutzern (z.B. spätestens 48 Stunden) oder allen anderen Änderungen bzw. Abweichungen (z.B. spätestens 7-14 Tage) soll nach Inkrafttreten eine

Anpassung im System durch das qualifizierte Personal vorgenommen werden (siehe IDM-04, IDM-05).

Eine automatische Überwachung für verdächtige Ereignisse, die durch Zugriff von privilegierten Nutzern entstehen könnte und deren Meldung, sind denkbare Abläufe, die innerhalb des Systems für die Sicherheit vor unbefugten Zugriffen von innen heraus sorgen (siehe IDM-06). Dazu dient unter anderem die Alarmierung von Cloud-Kunden, auf deren Daten von Mitarbeitern des Cloud-Dienstes zugegriffen wurde (siehe IDM-07).

In Anlehnung an die Dokumentationen bezüglich der Zugangs- und Zugriffsberechtigungen sind Aufzeichnungen hinsichtlich der Richtlinien zur Passwort-Vergabe und den getroffenen Konventionen, für das Erstellen eines Passworts und dessen sichere serverseitige Speicherung, sicherheitstechnisch relevant (siehe IDM-09). Eine Umsetzung von Seiten des Identity Providers durch Funktionen wie zum Beispiel die Validierung der Gültigkeit von 14 Tagen für ein initial vergebenes Passwort ist hier denkbar (siehe IDM-08). Auch Abweichungen von diesem Vorgehen sind für die spätere Entwicklung in Form einer Dokumentation relevant.

2.4.5 Kommunikationssicherheit (COS)

Eine Implementierung von technischen Schutzmaßnahmen zur Identifikation von anomalen Eingangs- und Ausgangs-Traffic-Mustern oder DDoS-Attacken sind innerhalb der Anwendung eine sicherheitskritische wie auch notwendige Funktion (siehe COS-01). Selbst bei Verwendung von Infrastrukturkomponenten wie zum Beispiel Firewalls oder Loadbalancer, ist eine hundertprozentige Erkennung von Angriffen nicht immer möglich. Jedoch können durch speziell konzipierte Dienste ein vom System selbst herbeigeführtes Schutzverhalten eingeleitet werden, das unabhängig vom Menschen und nicht erst durch Weiterleiten von Anomalien an übergeordnete Security Information- and Event Management-Systeme (SIEM) agiert.

2.4.6 Beschaffung, Entwicklung und Änderung von Informationssystemen (DEV)

Für die sichere Entwicklung des Cloud-Diensts sind Richtlinien und Anweisungen mit technischen und organisatorischen Maßnahmen mit den Schwerpunkten Sicherheit in der Software-Entwicklung (u.a. Anforderungen, Design, Implementierung, Tests und Überprüfungen), Sicherheit in der Softwarebereitstellung und die Sicherheit im Betrieb (Reaktion auf identifizierte Fehler und Schwachstellen) notwendig (siehe DEV-01). Mithilfe dieser Anforderungen können zusätzliche organisatorische Maßnahmen zur Verwaltung von Änderungen an den Diensten der SaaS-Lösung im Rahmen der Software-Bereitstellung beschrieben werden (siehe DEV-03). Ohne diese Regularien ist eine Risikobeurteilung von Änderungen auf Basis ihrer potenziellen

Auswirkungen auf die restlichen Dienste nicht durchführbar und eine entsprechende Kategorisierung ist nicht möglich (siehe DEV-05).

Nicht nur Dokumentationen bezüglich der oben genannten Regularien sind im Bereich DEV essenziell. Auch Programme zur Weiterbildung der internen Mitarbeiter bezüglich der Sicherheit in der Software-Entwicklung und Bereitstellung sind ein grundlegender Bestandteil für das Ausliefern sicherer Softwareprodukte (siehe DEV-04). Es wird hiermit eine regelmäßige und zielgruppenorientierte Sicherheitsausbildung und Sensibilisierung empfohlen.

2.4.7 Umgang mit Sicherheitsvorfällen (SIM)

Die Dokumentation, Kommunikation und Bereitstellung von Richtlinien und Anweisungen mit technischen und organisatorischen Maßnahmen sind im Umgang mit auftretenden Sicherheitsvorfällen für das betroffene Personal unerlässlich. Hier werden Vorgaben definiert, die die Klassifizierung, Priorisierung und Eskalation von Sicherheitsvorfällen beschreiben. Diese Prozesse reichen bis hin zur Verarbeitung von Vorfällen mit anschließender Information des Kunden über einen ihn betreffenden Zwischenfall (siehe SIM-01, SIM-03).

Ein Mechanismus zur Messung und Überwachung von Art und Umfang der Sicherheitsvorfälle und deren Meldung an die notwendigen Stellen wird ebenfalls nahegelegt (siehe SIM-05).

2.4.8 Produktsicherheit (PSS)

Unterstützend werden im Rahmen der Produktsicherheit Leitlinien bzw. Anleitungen definiert, die dem Kunden bei der sicheren Konfiguration des Cloud-Produktes helfen. Diese umfassen neben der sicheren Einrichtung, Informationsquellen zu bekannten Schwachstellen, Fehlerbehandlungs- und Protokollierungsmechanismen, Authentisierungsmechanismen, Rollen- und Rechtekonzepte (inkl. riskanter Rechte-Kombinationen) und Dienste und Funktionen zur Administration (siehe PSS-01).

Zur Absicherung der Qualität bezüglich der Sicherheit des Programmcodes werden dynamische u.a. auch statische Code-Analysen für den Einsatz und zur Überprüfung vor dem Ausrollen auf die Produktivumgebungen empfohlen. Deren Ergebnisse sollten anschließend nach Schweregrad der identifizierten Schwachstelle nach vordefinierten Kriterien beurteilt werden (siehe PSS-02). Hierzu verwendete Tools aus dem .NET Core Umfeld sind zum Beispiel „Security Code Scan - static code analyzer for .NET“, der auf Basis der OWASP Top Ten der meistausgenutzten Sicherheitslücken die Code Fragmente eines .NET-Projektes scannt [17]. Dies kann auch automatisch im Rahmen einer Continuous Integration und Continuous Delivery Pipeline geschehen. Beispielhaft für den Einsatz dieses Tools für Static Application Security Testing (SAST)

ist die Verwendung bei GitLab, dem direkten Konkurrenten von GitHub, in seiner eigenen CI/CD Toolbar [18].

Die somit identifizierten Schwachstellen, die nicht sofort behoben werden können, sind gemäß den Richtlinien nach einer Kategorisierung, wie zum Beispiel des Common Vulnerability Scoring System (CVSS) eingestuft und dem Kunden über einen Verweis auf ein Online-Register oder in Form einer Dokumentation ausgehändigt zu werden (siehe PSS-03, PSS-09).

Bezüglich der Protokollierung von Informationen des Cloud-Diensts ist die Verwendung hinsichtlich der Schaffung besserer Sicherheitsstandards fraglich. Hier wird die Möglichkeit einer Einsicht des Kunden in die protokollierten Daten des Dienstes angestrebt. Mithilfe dieser Daten soll er anhand von Fehlerbehandlungsmechanismen auftretende Störungen selbst beheben können. Diese Funktion wird jedoch von Seiten eines vom Cloud-Anbieter betriebenen Support-Centers übernommen, das technische wie auch logische Problematiken entgegennimmt und als Teil des Service-Pakets für den Kunden bearbeitet. Aus diesem Grund wird diese Funktion nicht als Teil der Optimierung verstanden. Sie wird hier anderweitig abgedeckt.

2.5 Analyse der vorhandenen Sicherheitskonzepte

In den nachfolgenden Unterabschnitten wird der aktuelle Stand an Sicherheitskonzepten der Software-as-a-Service Lösung, auch „LEXolution.FLOW“ genannt, analysiert und einem möglichen Soll-Zustand gegenübergestellt. Dieser Soll-Zustand orientiert sich hierbei an den aus Abschnitt 2.4 benannten Maßnahmen zur Optimierung der bestehenden Sicherheitskonzepte. Explizit wird hierbei auf die technischen Schutzmaßnahmen der Cloud-Lösung eingegangen. Diese werden als Basis für die spätere Implementierung eines Mechanismus zum Schutz der Anwendung gelegt.

2.5.1 Ist-Zustand

Die bewertete SaaS-Lösung der STP wird über einen Cloud Service Provider (nachfolgend auch CSP) zur Verfügung gestellt, bedeutet die notwendige Hardware wird nicht In-House betrieben, sondern extern in Bezug auf den Bedarf hinzugebucht. Hierbei handelt es sich um ein deutsches Rechenzentrum, welches seine Daten ausschließlich in Standorten innerhalb von Deutschland speichert. In Folge des Beschlusses der amerikanischen Regierung ist es mittels des CLOUD Acts (Clarifying Lawful Overseas Use of Data) regierungsnahen Institutionen, wie zum Beispiel der NSA, CIA oder FBI möglich, sich ohne Einwilligung oder vorheriges Informieren der Cloud Nutzer, Zugang zu den gespeicherten Daten der Cloud Provider zu verschaffen [19]. So wäre ein Hosting bei Microsoft (Azure), Amazon Web Services (AWS) oder Google Cloud Platform (GCP)

mit der deutschen Rechtslage für die Zielgruppe der STP im Rahmen des Berufsträgergeheimnis § 203 StGB nicht vereinbar [20].

Bei der Analyse des Ist-Zustandes steht jedoch nicht die Bewertung der Regularien zum Betreiben einer Cloud Lösung bzw. die Sicherheitskriterien für das Rechenzentrum im Mittelpunkt, sondern die Resistenz vor gängigen Angriffen auf Cloud Softwarelösungen. Hiermit wird eine Grenze zwischen der Software und der Plattform beziehungsweise der darunterliegenden Infrastruktur gezogen („Carve-Out Methode“ [15, S. 4]). Diese Vorgehensweise orientiert sich hierbei an einem vom BSI in Kooperation mit der Beratungsfirma Pricewaterhouse Coopers (PwC) durchgeführten Projekt zur Bewertung einer SaaS-Lösung, die wie in diesem Fall nicht in einem eigenen Rechenzentrum betrieben wird.

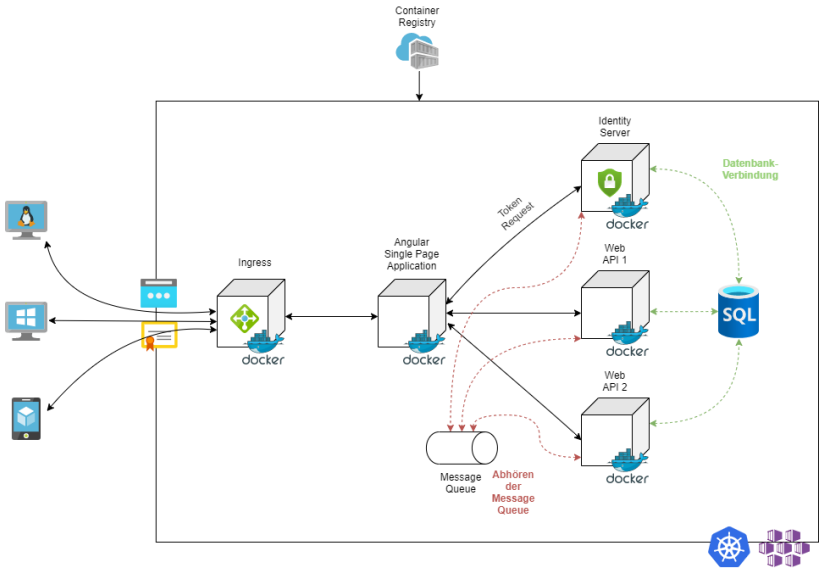

Abbildung 07 Vereinfachte Darstellung der STP Cloud SaaS-Lösung

Mithilfe der vorangegangenen vereinfachten Darstellung der STP Cloud Lösung soll nun die Bestandsaufnahme des Ist-Zustandes erfolgen. Der Anwendungsverbund besteht im übergeordneten Sinne aus unterschiedlichen Single Page Applikationen (nachfolgend auch SPA genannt), die die grafische Benutzeroberfläche der Anwendung repräsentieren. Hierbei wird je nach gewünschtem Menüpunkt über den vorgelagerten Ingress im Kubernetes Kluster die entsprechende SPA angesprochen. Somit verteilt sich die Last nicht nur auf eine einzelne Webanwendung. Über die Weboberfläche wird dem Kunden die Eingabe von für den Prozess relevanten Daten ermöglicht. Die Verarbeitung und Speicherung der Daten wird im Anschluss über Web-Programmierschnittstellen (nachfolgend auch API genannt) gewährleistet. Die Implementierung der APIs ist mit dem .NET Framework und C# als Programmiersprache umgesetzt worden. Für die sichere Übertragung der Daten vom Frontend an das Backend wird

mittels TLS 1.2 der ein- und ausgehende Datenstrom verschlüsselt. Dies verhindert im ersten Schritt das ungewünschte Mitschneiden und -lesen von Datenpaketen. Bezüglich der Verwaltung von Rechten wird im Backend ein Identity Provider (hier IdentityServer in der Version 4) bereitgestellt. Hierbei handelt es sich um einen Service, der das OpenID Connect und das OAuth2 Protokoll umsetzt. Die Nutzung dieses Diensts gewährleistet eine zentrale Nutzerverwaltung für alle Tenants und die Vergabe von Rechten unterschiedlicher Granularität. Bevor der Nutzer Zugriff auf die SaaS-Lösung erhält, muss er sich initial beim Identity Provider authentifizieren. Stimmen Nutzername und Passwort mit den hinterlegten Anmeldedaten überein, erhält der Nutzer, die ihm zugewiesenen Rechte, er wird somit autorisiert. Im Hintergrund des Anmeldeprozesses findet der Authorization Code Flow with Proof-Key-of-Code-Exchange (nachfolgend auch Authorization Code Flow + PKCE genannt) statt. Hierbei handelt es sich um das aktuellste und sicherste im OAuth2 Protokoll spezifizierte Verfahren für den Austausch von Tokens zwischen Identity Providern und clientseitigen Webanwendungen, wie zum Beispiel SPAs [4].

Für die Bewertung des aktuellen Sicherheitszustandes zur Erkennung und Abwendung von DoS-Angriffen wurden mithilfe von Oracle Virtualbox in der Version 6.1 [21], einer virtuellen Maschine und dem Betriebssystem Kali Linux in der Version 2021.1 [22], welches in der Industrie für Penetrationtesting von Business Anwendungen verwendet wird, im Rahmen dieser Thesis von Seiten des Autors gezielt Angriffe auf das System ausgeführt. Die Ausführung der Angriffe wurde in zwei Phasen durchgeführt. Die erste Angriffsphase erstreckte sich vom 01.06.2021 bis zum 04.06.2021, in der die Tools auf dem oben genannten Zielsystem installiert und prototypisch für kurze Zeit verwendet wurden. In der zweiten Phase vom 07.06.2021 bis zum 11.06.2021 wurden nochmals gezielte Services und Endpunkte angegriffen. Hierbei wurden die Open Source DoS- bzw. DDoS-Tools Low Orbit Ion Cannon (LOIC) [23], GoldenEye [24] und Slowloris [25] verwendet. Die Verwendung dieser Tools begründet sich auf dem weitverbreiteten Einsatz in der Community und der Referenz aus dem Paper „Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization“ [26], aus dem das verwendete Dataset für das maschinelle Lernmodell in Unterabschnitt 2.6.3.3 stammt. Hiermit soll gewährleistet werden, dass der erzeugte Netzwerkverkehr, der in den späteren Versuchen im Rahmen des programmatischen Teils erzeugt wird, auch wieder durch das Modell erkannt wird und somit ein Vergleich gezogen werden kann, ob das maschinelle Lernmodell funktionsfähig ist. Die hieraus gewonnenen Erkenntnisse werden nun in den nachfolgenden Unterabschnitten genauer beleuchtet.

2.5.1.1 DDoS-Angriffsszenario mit LOIC

Die Testreihe wurde mit dem Tool LOIC mit dem Ziel auf den Endpunkt des Identity Providers eingeleitet. Die Low Orbit Ion Cannon ist ein in C# geschriebenes Desktop-Programm, das in zwei unterschiedlichen Modi zur Verfügung steht. Es funktioniert einerseits im sogenannten „Manual Mode“, in dem nur eine Maschine (auf der das Programm läuft) mithilfe von Thread-Pools Anfragen an einen bestimmten Endpunkt oder an unterschiedliche Endpunkte des Services schicken kann. Für einen fortgeschrittenen DDoS-Angriff verfügt es auch über den „HiveMind“. Der „HiveMind“ erlaubt die Verbindung mit einem fremden Internet Relay Chat-Server (IRC), der remote von einem Administrator gesteuert wird und alle angeschlossenen LOIC als Bot-Netz instrumentalisiert. Hiermit hat der Endanwender, auf dessen System die LOIC ausgeführt wird, keinerlei Kontrolle über deren Verhalten mehr und kann Teil eines größer angelegten Angriffes werden. Innerhalb dieses Szenarios wurde jedoch nur der „Manual Mode“ verwendet, da kein eigener IRC-Server zur Verfügung stand und die Verbindung auf unbekannte Server als zu risikoreich eingestuft wurde. Nach dem Start der Anwendung begann diese den festgelegten Endpunkt mit einer Flut an Anfragen zu kontaktieren. Die Besonderheit dieser Anfragen ist, dass es sich hierbei um keine Keep-Alive Anfragen handelt, wie zum Beispiel beim Tool Slowloris, sondern hier wird versucht durch die immense Masse an Requests das System zum Überlasten zu bringen.

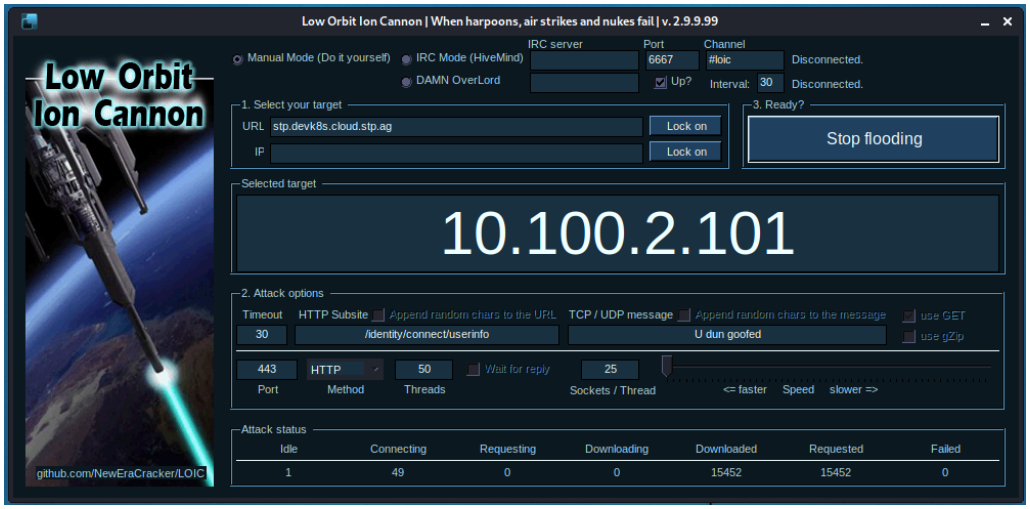

Abbildung 08 Angriff mit der LOIC auf den Login-Endpunkt des Identity Providers

Das Ergebnis des ersten Testlaufs war, dass der NGINX Ingress, der als Reverse Proxy an der Schnittstelle zwischen Kluster und dem Internet steht, die Anfragen nicht an den gewünschten Service weiterleitete. Wie im nachfolgenden Ausschnitt aus der Log-Datei des NGINX kenntlich wurde, hat er die eingehenden Requests mit dem Status Code „**400 Bad Request**“ gekennzeichnet und nicht in das dahinterliegende Kluster geleitet.

Abbildung 09 Access Log-Datei des NGINX Reverse Proxy vor dem Dev-Kluster

Dieses Verhalten wurde im späteren Verlauf der Testreihe an der selbstkonzipierten Testumgebung ebenfalls beobachtet. Somit wurde die dahinterliegende Anwendung nicht von dem Angriff getroffen und es wurde kein „Denial-of-Service“ herbeigeführt.

Die Analyse in der lokalen Testumgebung ergab folgendes Ergebnis. Mittels LOIC werden „Plain HTTP Requests“ verschickt. Durch einen Angriff auf Port 443, also HTTPS, des Reverse Proxy werden diese Anfragen von Seiten des Proxy somit stets geblockt, da es sich um keine auf TLS-Basis verschlüsselten Anfragen handelt. Diese Information konnte durch das Erhöhen des Logging-Levels von „Info“ auf „Debug“ innerhalb der „nginx.conf“-Datei und eines erneuten Testlaufes gewonnen werden.

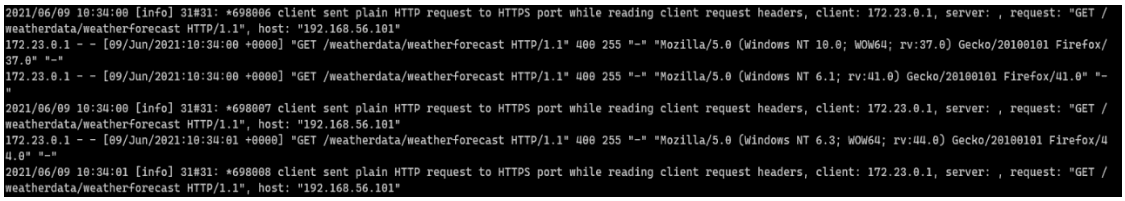

Abbildung 10 Ergebnis des erhöhten Logging-Levels in Kombination mit Angriff von „LOIC“ auf Port 443

Nach der Anpassung des Ports von 443 auf den HTTP Port 80 hat der NGINX in der lokalen Testumgebung die Anfragen an das entsprechende Backend weitergeleitet. Dasselbe Verhalten konnte nicht innerhalb des Development Klusters der STP beobachtet werden.

Hier wurde der Endpunkt des Identity Providers `/identity/connect/userinfo` angesprochen. Dieser stellt der Anwendung Informationen über den angemeldeten Nutzer gegen den Austausch eines Access Tokens zur Verfügung. Die Wahl dieses Endpunktes basierte auf der Abfrage des übergeordneten Endpunktes `/identity/.well-known/openid-configuration`. Hierbei handelt es sich um den „Discovery Endpoint“ des Identity Providers. Dieser ist nach Außen ohne vorherige Authentifizierung zugänglich und ermöglicht dem Angreifer spezifische Endpunkte des Identity Providers zu identifizieren und mit DoS- oder DDoS-Angriffen zu überlasten.

Bezüglich der fehlgeschlagenen Versuche über Port 80 Anfragen an diesen speziellen Endpunkt zu schicken, ist anzunehmen, dass dieser Port für Anfragen in das Development Kluster deaktiviert

wurde. Diese Tatsache konnte nach einem Blick in die Konfigurationsdatei im NGINX Reverse Proxy unter den `sites-enabled` bestätigt werden. Der Angriff von LOIC gegen die SaaS-Lösung blieb somit ohne Erfolg.

2.5.1.2 DoS-Angriffsszenario mit Slowloris

Im Anschluss an die Testreihe aus Unterabschnitt 2.5.1.1 kam das Tool Slowloris zum Einsatz. Bei Slowloris handelt es sich um ein in Python geschriebenes Konsolen-Programm, welches 150 Sockets auf dem Client öffnet und mit Keep-Alive Anfragen die Verbindung zum Ziel aufrechterhält. Dies funktioniert über das wiederholte Senden von kleinen Paketen an den entsprechenden Endpunkt. Somit wird die Verbindung vom Ziel aus nicht abgebaut und der Ressourcen-Pool mit der Zeit erschöpft.

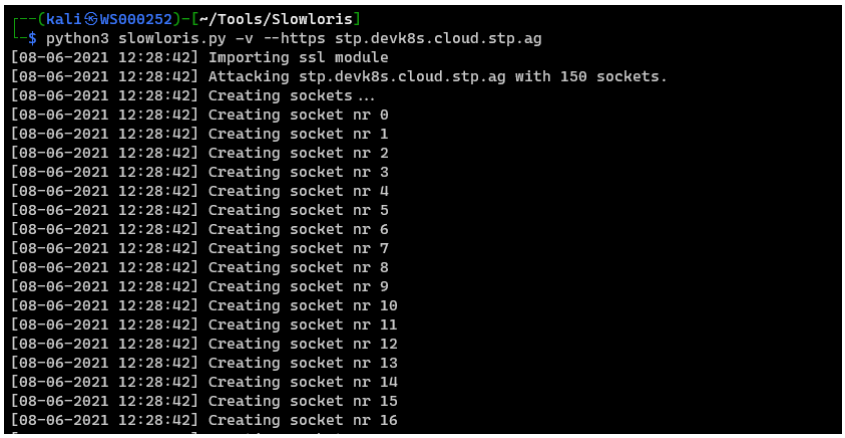

Abbildung 11 Angriff mittels „Slowloris" auf den NGINX Reverse Proxy

Auch dieser Angriffsvektor verlief ohne Erfolg. Der Reverse Proxy hat die Anfragen allesamt mit dem HTTP Status Code „**408 Connection Timeou**t" gekennzeichnet. Dies deutet darauf hin, dass die eingebettete Zeit für das Aufrechterhalten von Verbindungen in der Standard-Konfiguration des Proxies deutlich kürzer ist. Somit müssten von Seiten des Angreifers Änderungen am Tool vorgenommen werden, um diesen Angriff erfolgreich durchzuführen. Da dieser aber keine Möglichkeit zur Einsicht in die Log-Dateien des Proxies besitzt, wird der Angriff von Seiten des Proxies bereits abgewehrt und der „Denial-of-Service" bleibt aus.

Abbildung 12 Log-Datei nach dem Angriff mit „Slowloris"

2.5.1.3 DoS-Angriffsszenario mit GoldenEye

Die Testreihe wurde mit der Angriffs-Simulation des Tools GoldenEye abgeschlossen. Bei GoldenEye handelt es sich, wie bei Slowloris, um eine Konsolen-Applikation, die in Python geschrieben worden ist. Über vorgegebene Kommandozeilen-Argumente lassen sich verschiedene Parameter während der Ausführung des Programms anpassen. Von der Funktionsweise ähnelt es der LOIC und kann unterschiedliche Arten von HTTP Anfragen an ein gewünschtes Ziel schicken.

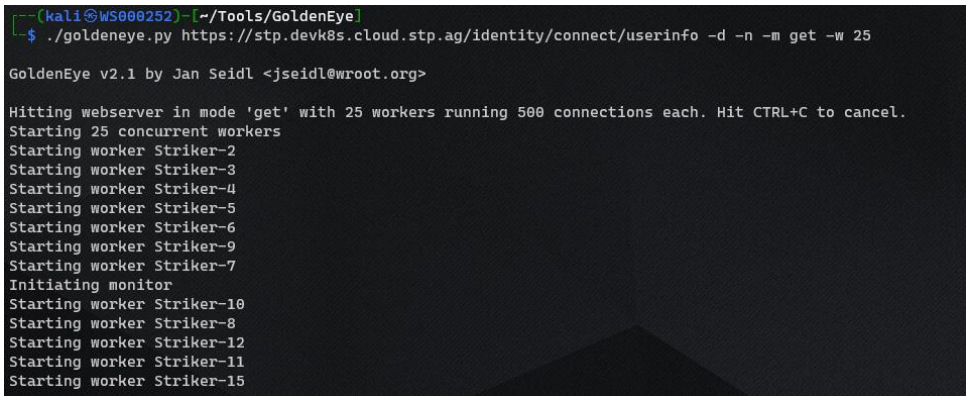

Abbildung 13 Angriff mit dem Tool „GoldenEye" auf die Dev-Cloud

Mithilfe dieses Tools konnte der gewünschte Endpunkt durch den Proxy hinweg angesprochen werden. Die Log-Dateien aus dem NGINX und dem IndentityServer wiesen die Vielzahl an produzierten Anfragen auf. Hierbei wurden von Seiten des NGINX wie auch von Seiten der dahinterliegenden Anwendung keinerlei Maßnahmen ergriffen, um diesen kontrollierten Angriff in einer Art einzudämmen, um eine spätere Überlastung auszuschließen.

Abbildung 14 Log-Nachrichten während des Angriffs mit "GoldenEye" auf den IdentityServer

2.5.2 Soll-Zustand

Resultierend aus der Analyse des Ist-Zustandes und der Bewertung mithilfe des Kriterienkatalogs und den daraus abgeleiteten Maßnahmen ist für den Soll-Zustand der SaaS-Lösung folgendes Konzept denkbar. Aufgrund des Fehlens eines internen Kontrollmechanismus für das Erkennen von netzbasierten Angriffen und Anomalien in Eingangs- und Ausgangs-Traffic-Mustern (siehe

2.4.5 Kommunikationssicherheit (COS)), wäre die Konzeption eines Dienstes mit solch einer Funktion als zusätzlicher Sicherheitsaspekt für den Anwendungsverbund geeignet.

Die Reaktion der Anwendung auf ein wiederholtes unautorisiertes Anfragen von öffentlich identifizierbaren Endpunkten ist wie zu erwarten ein Status-Code „**401 Unauthorized**“, auf den keine weiteren Maßnahmen in diesem Prozessmodell folgen.

Sollte hier ein Angreifer mittels eines Denial-of-Service oder des fortgeschritteneren Distributed-Denial-of-Service Angriffs versuchen das System über das Fluten von Endpunkten mit Anfragen zu überlasten, wird die Anwendung entweder auf Basis ihrer darunter liegenden Plattform, wie in diesem Fall Kubernetes, dementsprechend skalieren oder unter der Last zusammenbrechen. Als Folge entstehen Kosten für den Anbieter und Ausfallzeiten für den Kunden.

Nachfolgend soll nun auf Basis der vorliegenden Architektur und evaluierten Prinzipien zum Schutz von Software-as-a-Service-Lösungen ein mögliches Modell vorgestellt werden, das durch seine Funktion eine Lösung für diese Problematik darstellen könnte.

Auf Basis der Konfigurationen, die innerhalb des NGINX möglich sind, um einen DoS- oder DDoS Angriff abzumildern, ist nun die Frage, ob es möglich ist ein selbstagierendes System mit den Funktionen eines NGINX Reverse Proxy zu konzipieren. Bei NGINX müssen zum Beispiel gezielte Endpunkte mit Maßnahmen für das Gewährleisten von einer bestimmten Anzahl an zulässigen Anfragen („Throttling“) pro Nutzer manuell und per Hand durch den Administrator in die einzelnen Server-Blöcke in der Konfigurationsdatei eingegeben werden. Auch das Blockieren und Erlauben von bestimmten IP-Adressbereichen („IP-Blocking“), die sich als mögliche Angreifer durch die manuelle Analyse der Log-Dateien ergeben haben, muss per Hand erfolgen. Es gibt auf Basis des proprietären Produktes NGINX Plus Möglichkeiten über das Buchen von bestimmten Modulen diese Funktionen dynamisch in das Produkt einzubinden, jedoch sind diese Erweiterungen nur in dieser Version des Reverse Proxy zugänglich. Die freierhältliche Open Source Variante, die auch hier zum Einsatz kommt, muss durch eigene Konfigurationen mittels der Programmiersprache Lua oder anderen frei erhältlichen Modulen je nach Nutzen angepasst werden.

Ziel der nachfolgenden Abschnitts wird es sein, solch eine Anwendung zu konzipieren und im Anschluss umzusetzen. Hierbei wird der Schwerpunkt auf das Erkennen dieser speziellen Art von Angriffen und das Throttling der hiermit verbundenen Anfragen gelegt.

2.6 Erkennung und Prävention von Gefahren

Speziell für das Szenario des Denial-of-Service oder der weitaus fortgeschritteneren Variante des Distributed-Denial-of-Service Angriffs besteht das Ziel darin eine öffentlich erreichbare Webanwendung mittels einer immensen Masse an Anfragen so lange zu fluten, bis die Last an Anfragen das System zum Zusammenbruch bringt. Dies zu verhindern, wird im Rahmen des zweiten Teils der Thesis die Aufgabe eines eigenentwickelten Dienstes.

Das Konzept basiert darauf, dass der Dienst eine Art Reverse Proxy simuliert, der in seiner Funktionsweise gängigen Lösungen wie zum Beispiel dem NGINX Reverse Proxy [27] ähnelt. Jedoch wird dieser Service auf Basis eines in .NET 5 [28] implementierten Microservices aufbauen, der bei Erkennen von Anomalien (hier die DoS-Angriffe) die dahinterliegenden Services durch ein Throttling (Verlangsamen) der Anfragen vor einer Überlastung und somit eines Ausfalls schützt. Die zugrundeliegende Logik, ob es sich um solch eine Art Angriff handelt, wird mithilfe eines maschinellen Lernmodells erfolgen, das mit Anfragenmustern trainiert wurde, die diese Angriffe enthalten. Sollte es dem Modell möglich sein diese Anomalien zu erkennen, kann es mittels ML.NET [29], einer jungen Technologie aus dem Hause von Microsoft, in den Microservice integriert und dort das Throttling einleiten. Die Funktionsweise eines Proxies wird dem Service über die Open-Source Bibliothek YARP [30] („Yet Another Reverse Proxy“) verliehen, die auch aus dem Hause Microsoft stammt. Die Kombination dieser beiden Technologien stellt in diesem Abschnitt die Herausforderung bei der Programmierung dieses zusätzlichen Sicherheitskonzeptes dar.

2.6.1 Konzeption einer Testumgebung

Für die Evaluation und Umsetzung des ML.NET Proxy (nachfolgend auch ML.Proxy genannt) Dienstes wurde anhand der im Unterabschnitt 2.5.1 beschriebenen Architektur der Anwendung eine beispielhafte Testumgebung erstellt. Diese besitzt die grundlegenden Eigenschaften des Originals und kann somit als Substitut für Testzwecke verwendet werden, ohne auf die eigentliche Anwendungslandschaft zugreifen zu müssen.

Auf Basis des Originals besteht dieser Anwendungsverbund aus einem NGINX Reverse Proxy in der Version 1.21.0, der den im Kubernetes Kluster befindlichen NGINX Ingress auf der lokalen Testumgebung simuliert. Er übernimmt die Weiterleitung der Anfragen an die dahinterliegenden Services und führt ein Load Balancing durch. Hinter dem Proxy ist eine Single Page Applikation als Frontend für die Testanwendung geschalten. Sie wurde mit dem Web-Framework Angular von Google in der Version 8 aus dem Jahr 2019 [31] unter Zuhilfenahme der Bibliotheken „Angular Lib for OpenID Connect & OAuth2“ in der Version 11.6.7 [32] zur Authentifizierung gegen den

AS und „NG Bootstrap“ in der Version 9.1.0 [33] für das Gestalten der Komponenten im Rahmen dieser Thesis implementiert. Sie dient dem Authentifizieren eines Nutzers gegen den AS und der damit verbundenen Datenabfrage an den übrigen zwei Microservices. Für die Authentifizierung und Autorisierung kommt der IdentityServer in der Version 5.2.1 [34] als OpenID Connect und OAuth2 Provider zum Einsatz. Hierbei handelt es sich um ein Framework, welches die genannten Standards umsetzt und als Identity Provider eingesetzt werden kann. Hiermit lassen sich gezielt einzelne Anwendungen absichern und der Zugriff von maschinellen Clients wie auch Usern der Anwendung durch Vergabe von eigendefinierten Rechten steuern. Hierzu wurde der IdentityServer als Projekt-Template, zur Verfügung gestellt von den Entwicklern Duende Software [34], in eine .NET 5 [28] Webanwendung eingebunden. Anschließend wurden die mitgelieferten Standardkonfigurationen auf die Testumgebung angepasst, die vorhandenen Clients und Scopes für lesende und schreibende Zugriffe und Logik zur Scope-Validierung ergänzt. Die restlichen beiden Microservices stellen Dummy-Services zur Abfrage von generischen Standort- und Wetterdaten dar. Beide Services wurden mit dem AS verknüpft, sodass nur autorisierte Nutzer mit einem gültigen JWT die vorgehaltenen Daten abfragen können.

Neben einem lokalen Betreiben dieser Anwendung mittels Docker Compose [35], besteht die Möglichkeit, wie im nachfolgenden Schaubild exemplarisch dargestellt, die Anwendung auf einem in Microsoft Azure Kubernetes Service (AKS) betriebenen Kubernetes [36] Kluster auszuführen. Dies dient unter anderem der Simulation der Produktivbedingungen für die spätere Evaluation. Auf die Nutzung einer Message Queue und einer relationalen Datenbank für die Persistenz der Daten wurde verzichtet. Der Fokus wird ausschließlich auf die Manipulation bzw. das Throttling der Anfragen an das Backend gelegt.

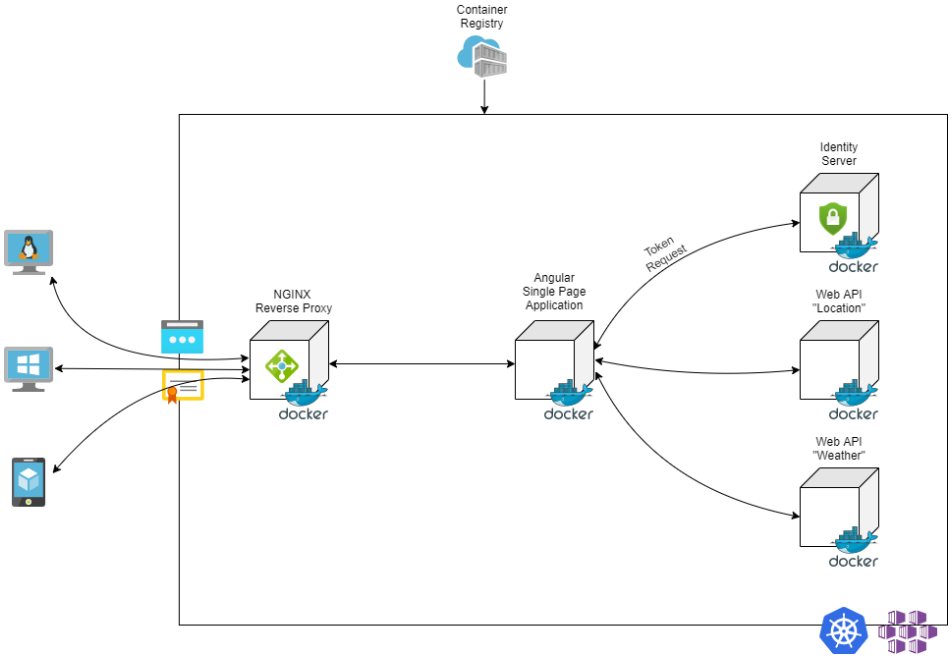

Abbildung 15 Testumgebung für die Simulation der realen Anwendungslandschaft

2.6.2 Mögliche Architektur mit ML.NET Proxy Service

Durch die Nutzung des ML.NET Proxy Service soll auf Auffälligkeiten in der Historie bzw. der aktuellen Anfragen von Außerhalb reagiert werden. Hierzu muss der Proxy jedoch zwischen Backend und dem Gateway des Klusters platziert werden. Eine mögliche Beispiel-Architektur ist in Abbildung 16 dargestellt.

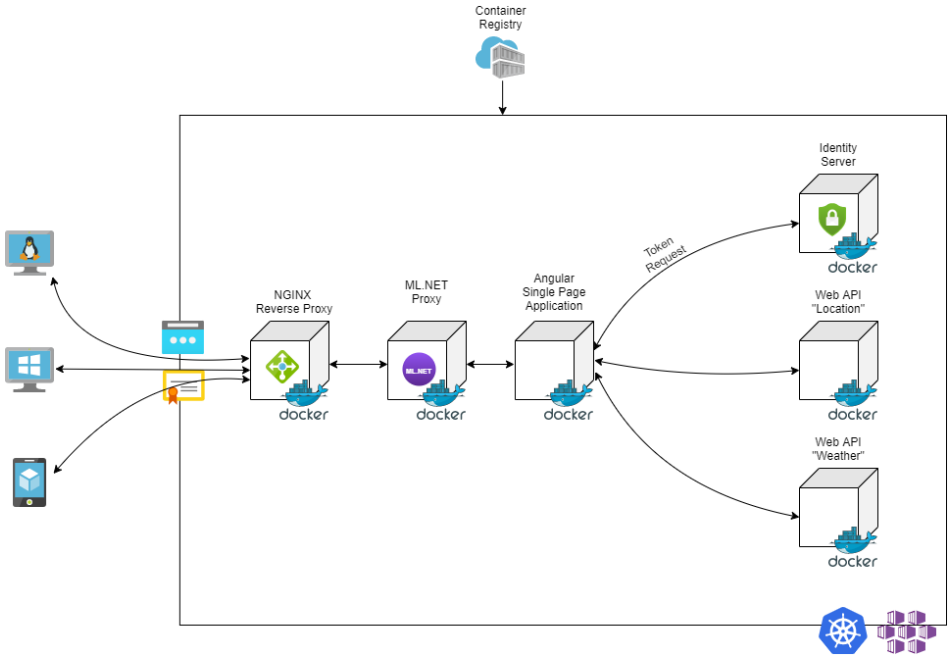

Abbildung 16 Beispiel-Architektur mit zusätzlichem Service für die Prüfung der eingehenden Requests mit Möglichkeit zum Throttling

2.6.3 Vorgehen bei der Implementierung

Die Entwicklung des kognitiven Dienstes zur Erkennung von DoS-Angriffen wird im Rahmen eines inkrementellen Prozesses durchgeführt. Arbeitspakete wurden jeweils für zwei Wochen ausgerichtet, um ein frühzeitiges Erkennen von Problemen und Inkompatibilitäten zu identifizieren und die Entwicklung auf einen anderen Teilaspekt zur Absicherung der Anwendung zu fokussieren.

2.6.3.1 Auswahl des Datensatzes für das Training des ML-Modells

Innerhalb des ersten Arbeitspaketes im Zeitraum vom 14.06.2021 bis zum 25.06.2021 stand die Auswahl und Evaluierung eines Datensatzes für das Trainieren des Machine Learning (ML) Modells im Mittelpunkt. Hierbei wurden verschiedene Datasets („Datensätze“) hinsichtlich ihrer Aktualität und der Verwendbarkeit geprüft. Hierbei fiel die Entscheidung auf das Dataset CSE-CIC-IDS2018, welches in Zusammenarbeit zwischen der Communication Security Establishment (CSE), der kanadischen nationalen Agentur für Kryptographie, und der University of New Brunswick entstand. Das CSE stellt hierbei eine regierungsnahe Institution dar, die die Versorgung der Regierung mit Informationen hinsichtlich Informationssicherheit und Aktionen ausländischer Geheimdienste im kanadischen Teil des Netzes [37] sicherstellen soll. Die Erstellung des oben genannten Datensatzes ging mit der gleichnamigen Arbeit „Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization“ [26] einher, die die Aktualität und Eignung der bisherigen öffentlichen Datensätze, wie zum Beispiel DARPA [38], KDD [39] und DEFCON [40] usw. prüfte und abschließend selbst ein maschinelles Lernmodell konzipierte, um die verwendeten Angriffe innerhalb von Log-Nachrichten und Mitschnitten von Netzwerkverkehr zu erkennen. Somit konnte neben dem reinen Datensatz auch auf die Erkenntnisse zur Auswahl und dem Training der Lern-Algorithmen zurückgegriffen werden. Bei den nachfolgenden Untersuchungen des Datenschemas werden im Rahmen dieser Arbeit die Ergebnisse dieses Papers zugrunde gelegt und durch eigene Experimente, beschrieben in dem nachfolgenden Unterabschnitt 2.6.3.2 und 2.6.3.3, hinsichtlich der Gültigkeit bzw. Validität der bereits existierenden Resultate ergänzt.

2.6.3.2 Untersuchung der Datensätze

Im nachfolgenden Abschnitt wird in Kürze die Zusammensetzung und Konzeption des ausgewählten Datensatzes erläutert. Der ausgewählte Datensatz CSE-CIC-IDS2018 untergliedert sich wiederum in vier einzelne Datensätze, die wiederum geordnet nach Tag und den an diesem Zeitpunkt ausgeführten Angriffen in mehreren Comma-Separated-Values (CSV) Dateien gespeichert wurden. Wie bereits in Unterabschnitt 2.6.3.1 erwähnt, war neben der Erstellung des hier verwendeten Datensatzes eine Evaluierung verschiedener maschineller Lernmodelle und

deren Fähigkeit zur Erkennung der verwendeten Angriffe Schwerpunkt der verwendeten Arbeit. Somit waren die CSV-Dateien bereits so weit aufbereitet, dass die Daten mit Überschriften (den sogenannten Features) und Label versehen waren. Im Rahmen des originalen Datasets wurde der normale von dem malignen Netzwerkverkehr durch die Label „Benign“ und der Art des Angriffs z.B. „Slowloris“ unterschieden. Diese Label wurden im späteren Verlauf dieser Arbeit in der Sektion zum Training des ML-Modells durch nummerische Werte, hier „0“ für „Benign“ und „1“ für die jeweilige Attacke, ersetzt. Die Erstellung der Datensätze fand von Seiten der Arbeitsgruppe mit einer selbstentwickelten Java-Anwendung namens CICFlowMeter in der Version 3 [41, 42] statt, die aus den gesammelten Log- und Packet Capture (PCAP)-Dateien eine Auswahl an 80 unterschiedlichen Eigenschaften des Netzwerkverkehrs extrahierte und die teilweise neu berechneten Werte in die unterschiedlichen CSV-Dateien auslagerte.

Die unterschiedlichen Arten von Angriffsmustern wurden jeweils separiert, während eines bestimmten Zeitfensters an einem Tag der Versuchswoche, durchgeführt. Nach der Sammlung der Daten wurde mittels des RandomForestRegressor, einer Funktion aus der Python-Bibliothek Scikit-Learn [43], die Relevanz der Features für die Erkennung eines bestimmten Angriffsmusters bestimmt. Für die Implementierung in Python wurde eine Referenzimplementierung [44] genutzt, die grundlegende Features bereits enthielt, jedoch für den Einsatzzweck im Rahmen dieser Arbeit noch angepasst wurde. Hieraus konnte auf Basis der Gewichtung der einzelnen Eigenschaften eine angriffsspezifische Teilmenge isoliert werden, die eindeutig für das Erkennen eines bestimmten Angriffsmusters war [26, S. 113]. Das Resultat dieser Untersuchung wird in der Tabelle im Anschluss zum Vergleich für die eigenen Berechnungen der Gewichtungen der einzelnen Eigenschaften („Feature Importance“) dargestellt:

Tabelle 19 Gewichtung der einzelnen Eigenschaften auf Basis des RandomForestRegressor (Quelle: Angelehnt an Tabelle aus [26, S. 113])

Label	Feature	Gewicht
DoS GoldenEye	Backward Packet Length Std	0.0479
	Flow IAT Min	0.0317
	Forward IAT Min	0.0257
	Flow IAT Mean	0.0214
DoS Slowloris	Flow Duration	0.0431
	Forward IAT Min	0.0378
	Backward IAT Mean	0.0300
	Forward IAT Mean	0.0265
DDoS LOIC	Backward Packet Length Std	0.1728
	Average Packet Size	0.0162
	Flow Duration	0.0137
	Flow IAT Std	0.0086

In den anschließenden Unterabschnitten werden die Analysen im Rahmen dieser Arbeit auf Basis der vorhandenen Datensätze aufgeführt und mit den bereits bestehenden Ergebnissen verglichen. Des Weiteren werden die hier aufgeführten Features für das bessere Verständnis in Unterabschnitt 2.6.3.3 nochmals erläutert.

2.6.3.2.1 Analyse des GoldenEye-Datensatzes

Die Analyse im Rahmen dieser Thesis begann mit dem Datensatz und den Werten für GoldenEye und Slowloris. Der Zeitraum für die DoS-Angriffe durch GoldenEye startete um 9:26 und endete um 10:09 Uhr am Donnerstag, den 15.02.2018. Für Slowloris wurde am selben Tag das Zeitfenster von 10:59 bis 11:40 Uhr ausgewählt. Der DDoS-Angriff unter Zuhilfenahme von LOIC fand am Dienstag, den 20.02.2018, von 10:12 bis 11:17 statt [45]. Auf die Eigenschaften und Auswertungen des LOIC- und des Slowloris-Datensatzes wird in den folgenden beiden Unterabschnitten eingegangen. Infolge des kurzen Abstandes der GoldenEye und Slowloris Angriffe war der protokollierte Netzwerkverkehr innerhalb einer CSV-Datei konkludiert und wurde von Seiten des Autors unter Zuhilfenahme eines eigen hierfür erstellten Python-Skriptes auf Basis der Label mit den Angriffsnamen voneinander getrennt. Bedingt durch die fehlenden Möglichkeiten im Bereich Data Science und der Datenauswertung von Seiten des .NET Frameworks und C# wurden die nachfolgenden Berechnungen der „Feature Importance“ mit Python-Skripten erstellt, die im Rahmen dieser Arbeit angefertigt wurden und in dem beigefügten ZIP-Archiv unter dem Verzeichnis „Implementierung“ im Ordner „ML.Proxy.FeatureImportance“ hinterlegt sind. Die verwendete Python Version war 3.9.5 [46] und zu den hier verwendeten Bibliotheken zählten neben Numpy in der Version 1.21.1 [47], Pandas in der Version 1.3.0 [48], Matplotlib in der Version 3.4.3 [49] unter anderem die Scikit-Learn Bibliothek [43] in der Version 0.24.2 für die

Verwendung des RandomForestRegressor. Zum Vergleich wurde mit der Bibliothek TensorFlow Decision Forests in der Version 0.1.7 [50] ausgeführt in Jupyter Notebooks unter der Version 6.4.3 [51] dieselbe Analyse nochmals durchgeführt, um einen Vergleich zwischen den erhaltenen Werten zu erstellen. Die Verwendung der Notebooks war hierbei unumgänglich, da sich der Decision Forest nach mehreren Anläufen nicht innerhalb eines Skriptes ausführen ließ. Mittels der Scikit-Learn Bibliothek wurden drei Analysen der „Feature Importance" für die einzelnen Eigenschaften des Datensatzes durchgeführt. Die dargestellten Diagramme sind Erzeugnisse der oben aufgeführten Matplotlib [49] Bibliothek und wurden während der Laufzeit des Skriptes generiert. Beginnend mit dem mitgelieferten Funktionsumfang des Random Forest Algorithmus, der die Gewichtung basierend auf dem sogenannten „Gini-Index" berechnet. „Der Gini-Index misst, wie oft etwas falsch klassifiziert würde, wenn die Kennzeichnung zufällig wäre." [11, S. 161]

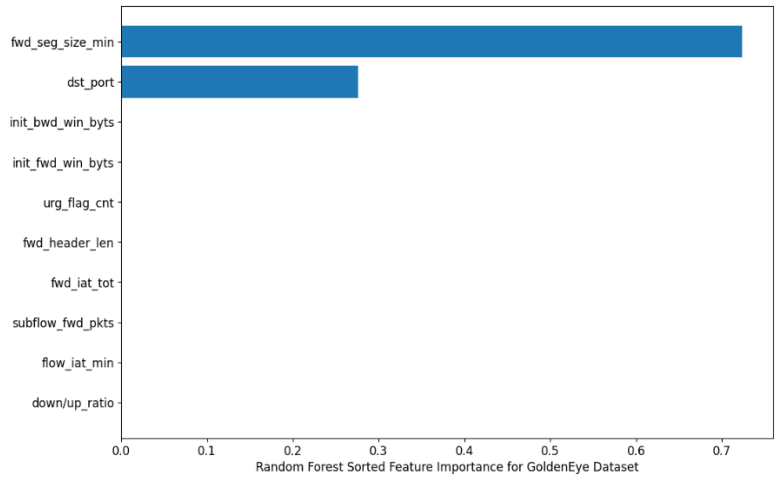

Abbildung 17 Auswahl der obersten zehn Eigenschaften auf Basis der „Feature Importance" des RandomForestRegressor für das GoldenEye-Dataset

Wie bereits aus der Abbildung zu erkennen ist, befinden sich die Eigenschaften aus dem Paper nicht in den selbstberechneten Gewichtungen. Zum Vergleich wurde mit der TensorFlow Decision Forest [50] Bibliothek eine ergänzende Evaluierung der Eigenschaften durchgeführt:

```
1   ("fwd_seg_size_min" (1; #51), 43.0),
2   ("init_fwd_win_byts" (1; #58), 39.0),
3   ("flow_pkts/s" (1; #33), 35.0),
4   ("flow_iat_mean" (1; #30), 31.0),
5   ("fwd_header_len" (1; #37), 21.0),
6   ("fwd_pkts/s" (1; #48), 21.0),
7   ("fwd_iat_max" (1; #38), 19.0),
8   ("flow_iat_max" (1; #29), 18.0),
9   ("timestamp" (4; #73), 12.0),
10  ("fwd_iat_tot" (1; #42), 11.0)
```

Abbildung 18 „Feature Importance" für das GoldenEye-Dataset basierend auf den Berechnungen des TensorFlow Decision Forests

Auch hier sind Abweichungen zu den ausgewählten Features in der obigen Referenz zu erkennen. Einzig die Eigenschaft „flow_iat_mean" in Zeile 4, die die „Interarrival Time" zwischen dem Start

von zwei „Flows“ angibt, stimmt mit dem vierten selektierten Kriterium des Originals überein. Ein „Flow“ stellt in diesem Szenario den Austausch einer Sequenz von Paketen zwischen zwei unterschiedlichen Diensten dar. Hierauf folgte als nächstes die Auswertung mit der „Permutation Feature Importance“ [52], die auf Basis eines beispielhaft trainierten Modells die Eigenschaften des von dem Training-Datensatzes abgespaltenen Test-Datensatzes verwendet und durch zufälliges Einsetzen die Auswirkungen auf die Performanz des Modells analysiert.

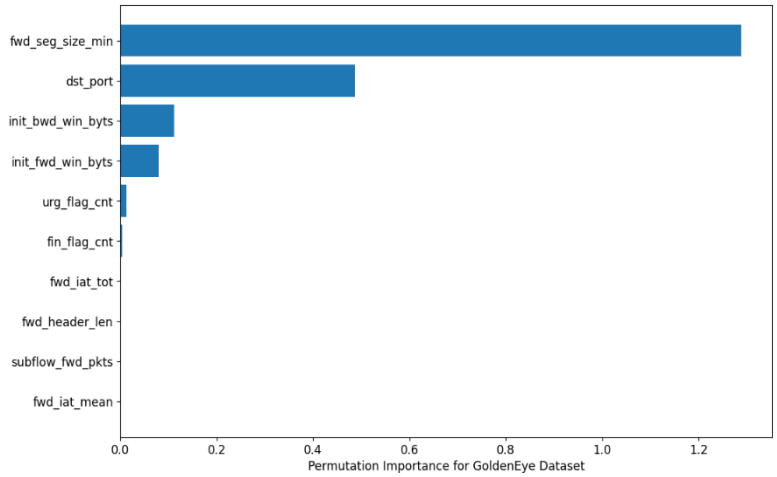

Abbildung 19 Die zehn am stärksten gewichteten Eigenschaften auf Basis der "Permutation Feature Importance" für das GoldenEye-Dataset

Auch in diesem Fall sind wie bei den vorherigen Analysen starke Abweichungen zu den ursprünglichen Werten aufgetreten. Durch die fehlenden Beschreibungen in den Aufzeichnungen zur Berechnung der „Feature Importance“ in der zugrundeliegenden Arbeit, lassen sich die festgestellten Unterschiede nur schwer konkretisieren. Die möglichen Variationen, die von Seiten der Bibliotheken geboten werden, die ein spezifisches Gewichten der Eigenschaften oder eine erhöhte Anzahl der verwendeten Bäume innerhalb des „Random Forest“ Modells ermöglichen, sind nicht im Paper der originalen Veröffentlichung enthalten. Somit sind Rückschlüsse auf Basis der eigenen Untersuchungen nur schwer durchführbar. Hierdurch ergeben sich weitere Aspekte zur Evaluation der Kriterien und Optionen, die eine Optimierung des Modells ermöglichen. Jedoch ist eine Verbesserung des Modells nicht Teil dieser Arbeit. Aus diesem Grund werden für das Training des später verwendeten Modells die selektierten Eigenschaften aus der Tabelle 19 verwendet. Hinsichtlich der Vollständigkeit werden in den anschließenden zwei Unterabschnitten die Gewichtung der Eigenschaften des Slowloris- und des LOIC-Angriffs dennoch berechnet und in Bezug zu den verwendeten Ergebnissen gesetzt.

2.6.3.2.2 Analyse des Slowloris-Datensatzes

Die in diesem Abschnitt verwendeten Ansätze und Analysen sind identisch zu den im Unterabschnitt 2.6.3.2.1 und werden aus diesem Grund nicht nochmals in kompletter Ausführlichkeit erläutert. Nachfolgend werden nun die Ergebnisse der Berechnungen mit dem bereits vorgestellten Python-Skript und einem zusätzlichen Jupyter Notebook [51] dargestellt.

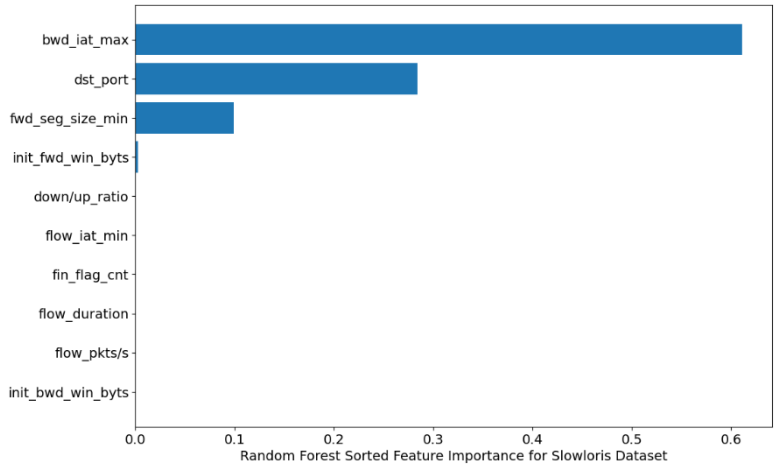

Abbildung 20 Auswahl der obersten zehn Eigenschaften auf Basis der „Feature Importance" des RandomForestRegressor für das Slowloris-Dataset

Wie im Fall des GoldenEye-Datensatzes sind in der Bewertung der Wichtigkeit der Eigenschaften für das Erkennen eines Slowloris-Angriffs unterschiedliche Ergebnisse im Vergleich zu der zugrundeliegenden Arbeit entstanden.

```
1   ("bwd_iat_max" (1; #8), 44.0),
2   ("bwd_iat_mean" (1; #9), 33.0),
3   ("bwd_iat_min" (1; #10), 32.0),
4   ("fwd_seg_size_min" (1; #51), 31.0),
5   ("bwd_iat_std" (1; #11), 25.0),
6   ("init_fwd_win_byts" (1; #58), 23.0),
7   ("flow_iat_std" (1; #32), 20.0),
8   ("active_max" (1; #1), 14.0),
9   ("active_mean" (1; #2), 14.0),
10  ("idle_std" (1; #56), 13.0)
```

Abbildung 21 „Feature Importance" für das Slowloris-Dataset basierend auf den Berechnungen des TensorFlow Decision Forests

Wie das Ergebnis der „Feature Importance" der TensorFlow Decision Forest [50] Bibliothek zeigt, konnte weder die Scikit-learn [43] Bibliothek noch die Implementierung von TensorFlow übereinstimmende Ergebnisse liefern.

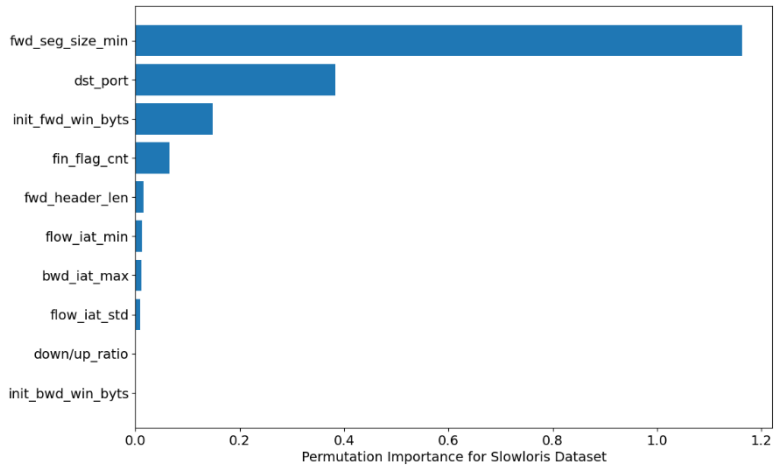

Abbildung 22 Die zehn am stärksten gewichteten Eigenschaften auf Basis der "Permutation Feature Importance" für das Slowloris-Dataset

Diese Differenzen setzen sich in der alternativen Analyse mit der „Permutation Feature Importance“ fort. Im Vergleich zu der Auswertung bei GoldenEye [24], gibt es hier keinerlei Übereinstimmungen mit dem Original. Wie in der Beurteilung des Ergebnisses aus dem vorherigen Unterabschnitt, wird auch hier für das spätere Training eines Modells die vordefinierten Feature-Sets aus dem herangezogenen Paper verwendet.

2.6.3.2.3 Analyse des LOIC-Datensatzes

Abschließend erfolgte die Prüfung der „Feature Importance“ für das LOIC-Dataset. Jedoch waren auch hier die Ergebnisse vergleichbar mit den Feststellungen aus den vorherigen beiden Unterabschnitten. Weder die Scikit-learn [43] Bibliothek noch die TensorFlow Decision Forests [50] Bibliothek konnten dieselben Ergebnisse liefern.

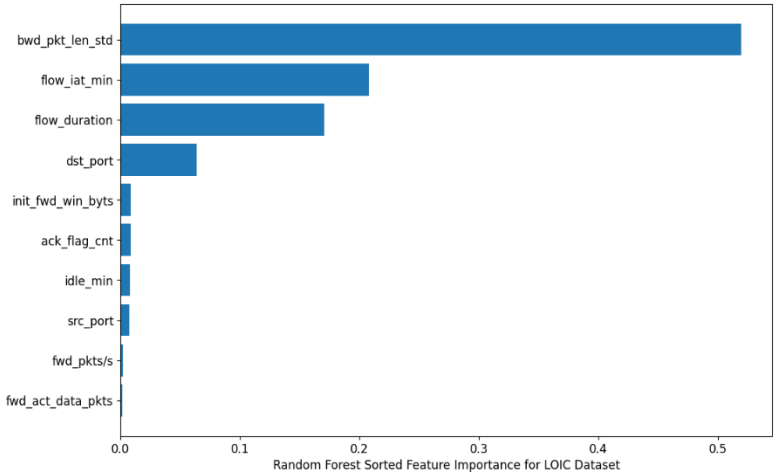

Abbildung 23 Auswahl der obersten zehn Eigenschaften auf Basis der „Feature Importance“ des RandomForestRegressor für das LOIC-Dataset

Es konnten zwar einzelne Übereinstimmungen wie z.B. aus Abbildung 23 ersichtlich die „bwd_pkt_len_std“, also die Standardabweichung der Größe von Paketen, die wieder vom angesteuerten Server zurückkommen und die „flow_duration“, als Dauer des einzelnen Flows, erzielt werden, jedoch sind dies nur zwei von vier Eigenschaften, die für eine valide Aussage somit nicht ausreichen.

```
("totlen_fwd_pkts" (1; #81), 52.0),
("dst_ip" (4; #24), 38.0),
("subflow_fwd_byts" (1; #74), 36.0),
("fwd_pkt_len_max" (1; #45), 28.0),
("fwd_seg_size_avg" (1; #52), 22.0),
("fwd_pkt_len_mean" (1; #46), 20.0),
("fwd_pkts/s" (1; #50), 18.0),
("flow_iat_max" (1; #30), 13.0),
("flow_pkts/s" (1; #35), 13.0),
("flow_duration" (1; #29), 12.0)
```

Abbildung 24 „Feature Importance“ für das LOIC-Dataset basierend auf den Berechnungen des TensorFlow Decision Forests

Keine Kongruenzen wurden von Seiten des TensorFlow Frameworks erzielt. Dieses konnte nach der Evaluierung des Datasets keine Übereinstimmung mit den gegebenen Feature-Sets erzielen.

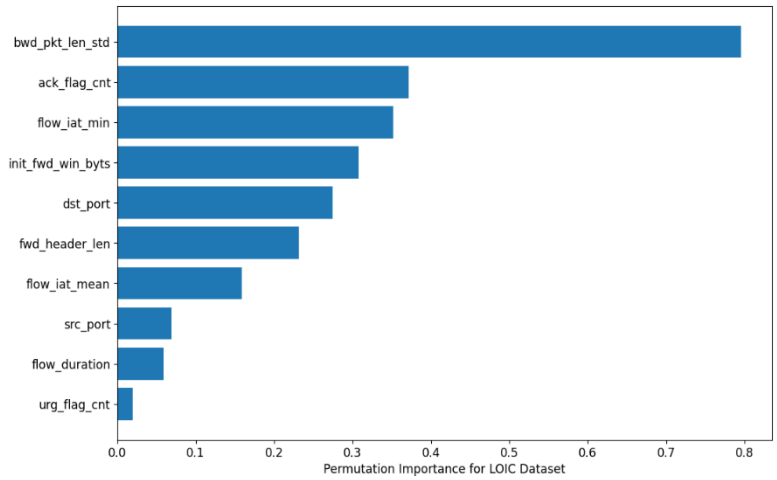

Abbildung 25 Die zehn am stärksten gewichteten Eigenschaften auf Basis der "Permutation Feature Importance" für das LOIC-Dataset

Wiederum vergleichbar war das Ergebnis der „Permutation Feature Importance“ Messung. Wie aus Abbildung 25 ersichtlich, konnte als markantestes Merkmal, wie auch bei der ersten Messung eine Konformität mit dem Kriterium „bwd_pkt_len_std“ erzielt werden. Trotz einer höheren Übereinstimmung mit den gegebenen Sets, werden auch in diesem Fall für das Modell die vorgegebenen Werte verwendet. Die Definition der nun ausgewählten Kriterien und das anschließende Training des Machine Learning Modells werden im folgenden Unterabschnitt genauer behandelt.

2.6.3.3 Trainieren des ML-Modells mit den Daten

Das Ergebnis basierend auf der Gewichtung der einzelnen Eigenschaften in den separierten Datensätzen konnte somit für die Auswahl der Features für das Training des Modells genutzt werden. Aufgrund der abweichenden Ergebnisse in Bezug auf das Ursprungsprojekt wurden im nachfolgenden Training die vorgegebenen Features verwendet. Im Anschluss folgen die selektierten Eigenschaften pro Angriff nochmals in einer Übersicht mit kurzer Definition. Diese wiederum basiert auf der Legende in der zugrundeliegenden Dokumentation der Datensätze [26]. Hierbei sind die in Klammern gehaltenen Spaltenüberschriften das Produkt eines weiteren Python-Skriptes, welches die Überschriften vereinheitlicht und sie somit für das ML-Framework und die verwendete Bibliothek lesbar macht. Zum Beispiel ist es für die Bibliothek TensorFlow Decision Forests [50] und das Framework ML.NET [29] nicht möglich eine Überschrift bzw. Feature in den Trainingsdaten mit enthaltenen Leerzeichen zu erkennen und zu verarbeiten. Bei Testversuchen mit ML.NET [29] war das Training zwar erfolgreich, jedoch hat die Validierung mit Testdaten nicht funktioniert. Grund für dieses Versagen könnte sein, dass die Funktion für das Einlesen der Spalten jedes Leerzeichen als separierte Spalte erkannt hat und die nachfolgenden Werte anschließend falsch zugeordnet hat. Somit resultierte kein Fehler im Prozess des Trainings, sondern in der nachfolgenden Validierung. Aus diesem Grund wurde eine allgemeine Standardisierung eingeführt. Der Code für die Standardisierung und Selektion der Eigenschaften der nachfolgenden Features aus den ursprünglichen Datasets ist in dem beigefügten ZIP-Archiv der hier vorliegenden Arbeit unter dem Verzeichnis „Implementierung“ im Ordner „ML.Proxy.Python.DataWrangler“ zu finden.

- DoS GoldenEye
    - Backward (Bwd) Packet Len Std („bwd_pkt_len_std“)
      Beschreibt die Standardabweichung der Länge eines Netzwerkpaketes in der Richtung von Server zu Client.
    - Flow IAT Min („flow_iat_min“)
      Beschreibt die minimale vergangene Zeit zwischen zwei Flows.
    - Forward (Fwd) IAT Min („fwd_iat_min“)
      Kennzeichnet die minimale vergangene Zeit zwischen dem Senden von zwei Paketen in der Richtung Client zu Server.
    - Flow IAT Mean („flow_iat_mean“)
      Beschreibt die durchschnittliche Zeit zwischen zwei Flows.

- DoS Slowloris
    - Flow Duration („flow_duration“)
      Kennzeichnet die Dauer eines Flows.
    - Forward (Fwd) IAT Min („fwd_iat_min“)
      Kennzeichnet die minimale vergangene Zeit zwischen dem Senden von zwei Paketen in der Richtung Client zu Server.
    - Backward (Bwd) IAT Mean („bwd_iat_mean“)
      Beschreibt die durchschnittliche vergangene Zeit zwischen dem Versenden von zwei Paketen in der Richtung Server zu Client.
    - Forward (Fwd) IAT Mean („fwd_iat_mean“)
      Beschreibt die durchschnittliche vergangene Zeit zwischen dem Versenden von zwei Paketen in der Richtung Client zu Server.
- DDoS LOIC
    - Backward (Bwd) Packet Len Std („bwd_pkt_len_std“)
      Bezeichnet die Standardabweichung der Größe eines Paketes in der Richtung von Server zu Client.
    - Avg Packet Size („pkt_size_avg“)
      Kennzeichnet die durchschnittliche Paketgröße.
    - Flow Duration („flow_duration“)
      Beschreibt die Dauer eines Flows.
    - Flow IAT Std („flow_iat_std“)
      Kennzeichnet die Standardabweichung der Zeit zwischen zwei Flows.

Diese Konstellation von Features mit den dazugehörigen Labels wurde in drei unterschiedliche CSV-Dateien für das nachfolgende Training ausgelagert, die jeweils einem Angriffsmuster entsprachen. Für das Training der ML.NET Modelle wurden die Features, die Größen in Richtung Server zu Client beschreiben, vernachlässigt. Diese Entscheidung wurde aufgrund der Tatsache gefällt, da der Proxy bzw. das Modell den Angriff bereits in der „Forward“-Richtung erkennen sollte, und so keine direkte Kommunikation zwischen Angreifer und den zu schützenden Diensten hinter dem Proxy zu ermöglichen. Aufgrund der unterschiedlichen Eigenschaften jedes Angriffs und den damit verbundenen abweichenden „Feature Importance“-Kriterien fiel die Entscheidung im Rahmen dieser Arbeit, anstatt einem Modell für die Erkennung aller drei Angriffe anzufertigen, auf die Konzeption dreier unterschiedlicher Modelle, die jeweils spezifisch für eine Attacke trainiert werden. Ein weiterer Grund für diese Entscheidung war die prognostizierte steigende Genauigkeit für die Erkennung eines speziellen Angriffs. Das jeweilige Modell kann somit separiert

von den zwei anderen spezifisch nach dem für ihn gedachten Angriffsmuster selektieren. Hiermit wird auch die Möglichkeit geboten unterschiedliche Services zur Erkennung von Angriffen auf die Cloud-Umgebung zu konzipieren. Eine mögliche Implementierung im ML.Proxy wäre dann z.B. eine Reihenschaltung der drei Modelle, die jede Anfrage in einer Art „Pipeline“ verarbeiten und auf Basis der Werte eine Einschätzung abgeben, ob es sich um einen zu blockenden oder einen validen Request handelt.

Das anschließende Training der Modelle wird in den nächsten beiden Unterabschnitten behandelt. Es wird einerseits ein Modell-Training mit dem ML-Framework von Microsoft, das sogenannte ML.NET [29], in der Programmiersprache C# durchgeführt und als Referenzimplementierung ein Training mit der Bibliothek TensorFlow Decision Forests [50] in der Programmiersprache Python. Ein Impuls für die Verwendung des Random Forest Modells ist die Analyse der Performance unterschiedlicher maschineller Lernmethoden, u.a. KNN, ID3 und dem hier verwendeten Random Forest, in der zugrundeliegenden Arbeit [26]. Als Resultat für die Zeit hinsichtlich des Trainings und dem Testen der Modelle, auch Ausführungszeit genannt, war das Ergebnis bei dem Modell KNN [(K-Nearest Neighbors)] mit 1.908,23 Sekunden am schlechtesten. Danach kam an zweiter Stelle der ID3 mit einer Zeit von 235,02 Sekunden. Das schnellste Modell mit einer Ausführungszeit von 74,39 Sekunden war der Random Forest [26, S. 114]. Aufgrund der somit festgestellten Genauigkeit und Geschwindigkeit hinsichtlich des Trainings, fiel die Auswahl des Modells auf die beiden oben genannten Implementierungen des Random Forest Modells. Die Nutzung von zwei unterschiedlichen Ansätzen der Programmierung im Rahmen dieser Thesis basierte auf der Entscheidung, dass bei Problemen oder Inkompatibilitäten eines der genutzten Modelle die Möglichkeit besteht im Nachhinein bei Bedarf auf ein anderes Modell umzustellen. Zusätzlich werden innerhalb der Lern-Algorithmen verschiedene Implementierungen hinsichtlich der Konstruktion der Entscheidungsbäume verwendet. Durch die Verwendung zweier unterschiedlicher Ansätze der Implementierung kann bei schlechter Performanz eines der Modelle wie im Fall einer Inkompatibilität zwischen den Modellen gewechselt werden.

2.6.3.3.1 Training der ML.NET Random Forest Modelle

Das Thema des nachfolgenden Unterabschnitts ist die Implementierung der als .NET Konsolen-Anwendung programmierten Trainingspipeline zur Erstellung und Evaluation der in Unterabschnitt 2.6.3.3 angesprochenen Modelle. In diesem Zusammenhang liegt der Schwerpunkt nicht auf der Entwicklung der Trainings- und Testpipeline, sondern auf den Erkenntnissen, die über den Prozess des Modell-Trainings gesammelt wurden. Der verwendete Code orientiert sich an einer Referenzimplementierung für die Auswertung eines Datasets, welches auf Basis unterschiedlicher Eigenschaften von Pinguinen, diese in ihrer Art unterscheiden soll [53]. Die

Struktur und Vorgehensweisen sind dabei in den meisten Machine Learning Projekten, die mit dem eingesetzten Framework erstellt wurden, gleich. Einzig die Projektstruktur unterscheidet sich in den einzelnen Beispielen und Referenzrepositories des Frameworks und anderen Projekten. Beginnend mit der Initialisierung eines `MLContexts` wird eine Pipeline erstellt, die zuerst die Daten in CSV-Format einliest und sie anschließend auf entsprechende C#-Klassen abbildet.

Genutzt wurden in der Eigenimplementierung die grundlegenden Methoden und die Projektstruktur, die für spätere Verwendungszwecke eine Erweiterbarkeit der bereits existierenden Funktionen vereinfachen sollen. Änderungen fanden hinsichtlich der verwendeten Klassen und bezüglich der Rückgabetypen der Funktionen statt. Hinsichtlich der Rückgabetypen wurden die Funktionen in generische Funktionen mit einem beliebigen Parameter `<T>` als Output und Input umgeschrieben. Dadurch wurde garantiert, dass dieselben Methoden mit allen verwendeten Klassen für das Training der drei Modelle operieren konnten. Ergänzungen ergaben sich im Kontext der Konvertierung der im ZIP-Format abgespeicherten Modelle in das Open Neural Network Exchange (ONNX) Format [54] und der anschließenden Evaluierung der gespeicherten Modelle mit Testdaten. Resultierend konnten drei Modelle entwickelt werden, die die notwendigen Fähigkeiten besaßen, Anfragen, die als Angriff gekennzeichnet waren, wiederum als Gefahr zu identifizieren. Zur Erkennung der Attacken benötigte jedes Modell eine unterschiedliche Anzahl an vorhandenen Bäumen und Blättern. Nachfolgend werden nun die minimal notwendigen Anforderungen und die erzeugten Metriken für jedes Modell aufgeführt. Der Source Code und die abgespeicherten Modelle befinden sich in dem beigefügten ZIP-Archiv unter dem Verzeichnis „Implementierung“ im Ordner „ML.Proxy.ModelTrainer“.

Das Modell für den GoldenEye-Datensatz war bereits mit einer Anzahl von 20 Bäumen und 10 Blättern je Baum im Stande Netzwerkverkehr zu erkennen, der als Angriff gekennzeichnet wurde. Eine Validierung dieses Ergebnisses wurde durch ein weiteres Experiment mit zufällig ausgewählten Daten aus dem Originaldatensatz nochmals überprüft. Ablauf des Tests war die Auswahl vier zufälliger Datenpunkte, von denen zwei jeweils normale Anfragen und zwei Gefahren darstellten. Das Modell konnte mit einer 100%igen Übereinstimmung für den jeweiligen Datenpunkt eine richtige Prognose aufstellen und hatte somit die notwendige Genauigkeit erreicht. Trotz der Verwendung von Trainingsdaten und einer somit bestehenden Gefahr, dass das Modell nur für bekannte Daten richtige Einschätzungen stellt, ist dieses Risiko, wie es sich im Fall des GoldenEye-Modells zeigt, zu vernachlässigen.

```
************************************************************
 Random Forest: 10-20 for GoldenEyeTrafficData-Attack
************************************************************
Accuracy: 0.99
AUC: 0.99
F1 Score: 0.87
Negative Precision: 0.99
Negative Recall: 1.00
Positive Precision: 0.93
Positive Recall: 0.82

TEST POSITIVE RATIO: 0.0398 (12368.0/(12368.0+298473.0))
Confusion table
          ||======================
PREDICTED || positive | negative | Recall
TRUTH     ||======================
 positive ||   10,088 |    2,280 | 0.8157
 negative ||      797 |  297,676 | 0.9973
          ||======================
Precision ||   0.9268 |   0.9924 |

************************************************************
Prediction: True
************************************************************
```

Abbildung 26 Metriken für das Modell zur Identifikation von GoldenEye-Angriffen

Im Fall des Slowloris-Modells wurde der bereits beschriebene Testlauf mit vier beliebigen Datenpunkten aus dem Slowloris-Originaldatensatz wiederholt. Hier konnten jedoch nur drei von vier Punkten richtig zugeordnet werden. Das Modell hat auch nach wiederholten Versuchen einen Datenpunkt nicht korrekt zuordnen können. Trotz der Verwendung von bekannten Trainingsdaten, ist dieses bereits angesprochene Wagnis auch in diesem Fall mit einer Genauigkeit von 75% einzugehen. Dieses Ergebnis lässt sich zusätzlich mit den erreichten Werten während der Testphase des Modells in Verbindung bringen. Selbst mit 20 Bäumen und 40 Blättern kann das Slowloris-Modell den zur Validierung herausgegriffenen Angriffsvektor nicht korrekt zuordnen.

```
************************************************************
 Random Forest: 20-40 for SlowlorisTrafficData-Attack
************************************************************
Accuracy: 1.00
AUC: 0.98
F1 Score: 0.88
Negative Precision: 1.00
Negative Recall: 1.00
Positive Precision: 0.95
Positive Recall: 0.81

TEST POSITIVE RATIO: 0.0107 (3216.0/(3216.0+298473.0))
Confusion table
          ||======================
PREDICTED || positive | negative | Recall
TRUTH     ||======================
 positive ||    2,618 |      598 | 0.8141
 negative ||      124 |  298,349 | 0.9996
          ||======================
Precision ||   0.9548 |   0.9980 |

************************************************************
Prediction: False
************************************************************
```

Abbildung 27 Metriken für das Modell zur Identifikation von Slowloris-Angriffen

Das LOIC-Modell hat sich im Test wie das GoldenEye-Modell mit 100%iger Genauigkeit bei den Zuordnungen ausgezeichnet. Somit wurde durch das Training mit ML.NET [29] in relativ kurzer Zeit drei Modelle erstellt, die mit einer überaus hohen Präzision Vorhersagen tätigen können. Diese Aussage wird pro Modell durch die aufgeführten Metriken, wie der „Accuracy“ und dem

Wert der „Area Under Curve“ (AUC) nochmals bestätigt. Alle Werte nahe eins sind in diesen Fällen positiv für die Genauigkeit eines Modells zu interpretieren.

```
************************************************************
 Random Forest: 5-10 for LOICTrafficData-Attack
************************************************************
Accuracy: 0.99
AUC: 0.99
F1 Score: 0.94
Negative Precision: 0.99
Negative Recall: 1.00
Positive Precision: 0.99
Positive Recall: 0.90

TEST POSITIVE RATIO: 0.0723 (172398.0/(172398.0+2210636.0))
Confusion table
          ||======================
PREDICTED || positive | negative | Recall
TRUTH     ||======================
 positive ||  155,375 |   17,023 | 0.9013
 negative ||    1,513 | 2,209,123 | 0.9993
          ||======================
Precision ||   0.9904 |   0.9924 |

************************************************************
Prediction: True
************************************************************
```

Abbildung 28 Metriken für das Modell zur Identifikation von LOIC-Attacken

Zur Kontrolle und zur Inspektion des Innenlebens der angefertigten Prototypen wurde mit der Webapplikation Netron [55] beispielhaft das ONNX-Modell für GoldenEye geöffnet und die Operationen betrachtet, die innerhalb des ONNX-Modells ausgeführt werden.

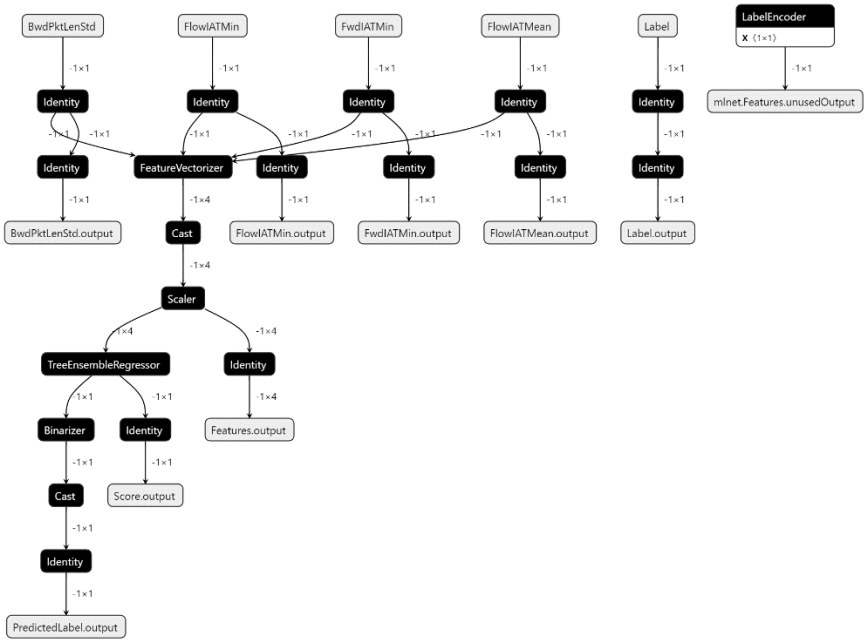

Abbildung 29 Inspektion des Innenlebens des GoldenEye-Modells mittels Netron

Beginnend mit den Features, die als Eingabedaten dienen und deren Transformation über den „FeatureVectorizer“ bis hin zu dem „TreeEnsembleRegressor“, in dem die Vorhersage bzw. die Klassifikation des eingegebenen Wertes bestimmt wird, lassen sich die einzelnen Wege, die bis hin

zur Entscheidungsfindung führen genauer inspizieren. Der Aufbau ist hier nur exemplarisch für das .NET-Modell und kann mit dem TensorFlow-Modell aufgrund der Verwendung von anderen Operationen nicht verglichen werden.

Der nächste Unterabschnitt stellt die Vorgehensweise zur Erstellung eines ML-Modells mit der Bibliothek TensorFlow Decision Forests [50] genauer dar.

2.6.3.3.2 Training der TensorFlow Decision Forests Modelle

Im Vergleich zur Implementierung der Trainingspipeline im vorherigen Unterabschnitt 2.6.3.3.1 ist der Ablauf zur Erstellung des Modells mit Python [46] und TensorFlow [56] interaktiver. Für die Programmierung wurde anstatt eines Skriptes ein Jupyter Notebook [51] verwendet, das sich an der Referenzimplementierung des offiziellen Entwicklerhandbuches [57] orientiert. Bei diesen Notebooks handelt es sich um eine Art Python-Server, die entweder über den Browser und einer grafischen Oberfläche oder über eine Integrated Development Environment (IDE) für Data Science angesteuert werden können. Im Rahmen dieser Thesis wurde das Notebook in einer virtuellen Linux Umgebung, dem Windows Subsystem for Linux (besser bekannt unter dem Namen WSL 2 [58]) auf dem Windows Host Rechner ausgeführt. Grund für die Verwendung der WSL Umgebung war, dass die Bibliothek TensorFlow Decision Forests [50] nur auf UNIX basierten Betriebssystemen zur Verfügung gestellt wird. Eine spätere Konvertierung für Windows steht im Rahmen des Entwicklungsprozesses noch aus. Alternativ wäre eine Kompilierung des Source Code nötig gewesen. In Anbetracht der zweiwöchigen Arbeitspakete hätte diese Variante den zeitlich angesetzten Rahmen für die Erstellung der Modelle jedoch überschritten. Die fertiggestellten Modelle und die Notebooks sind als Referenz in dem beigefügten ZIP-Archiv unter dem Verzeichnis „Implementierung“ im Ordner „ML.Proxy.Python.ModelTrainer“ hinterlegt. Äquivalent wie im vorherigen Unterabschnitt konnten am Ende des Trainingsprozesses resultierend drei Modelle für jeweils einen Angriff angefertigt werden. Die herbei entstandenen Ergebnisse unterscheiden sich in ihrem Aufbau grundlegend von den Modellen aus Unterabschnitt 2.6.3.3.1. Abgesehen von der Anzahl der erstellten Bäume, die bei allen TensorFlow-Modellen auf eine Stückzahl von 300 Bäumen festgesetzt ist, unterscheiden sich die Prototypen in ihrer erzeugten Menge an Knoten. Beginnend mit einer Knotenmenge von insgesamt 586.286 hat das LOIC-Modell am meisten Knoten von allen drei Modellen erzeugt. Gefolgt von 285.772 Knoten im Falle des GoldenEye-Modells und schließlich das Slowloris-Modell mit 124.818 Knoten. Alle drei Modelle wiesen eine Genauigkeit von ungefähr 99% auf den abgespaltenen Testdatensätzen auf und sind somit den Resultaten von ML.NET [29] ähnlich. Es folgt eine Darstellung des ersten binären Baumes des GoldenEye-Modells, der auf Basis der unterschiedlichen Eingabe-Features pro Knoten eine Entscheidung fällt, die wiederum in nachfolgende Knoten oder Blätter mündet.

Innerhalb der Abbildung kennzeichnen die roten den normalen Netzwerkverkehr und die blauen Flächen einen Angriff. Aufgrund der wenigen Visualisierungstools des C# Frameworks, konnte solch eine Abbildung nicht im Falle der anderen Modelle erzeugt werden.

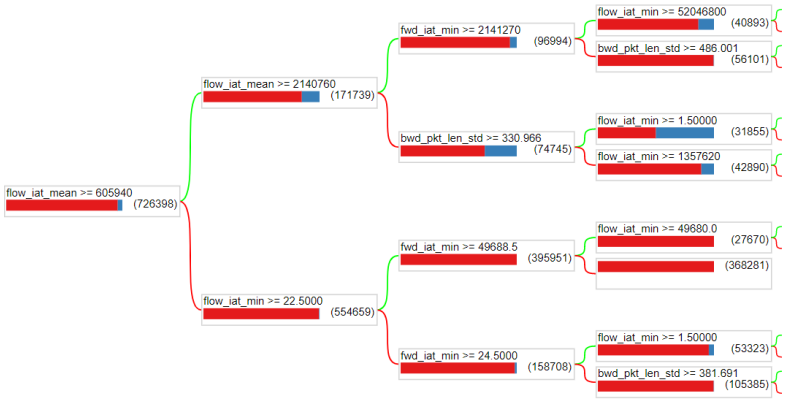

Abbildung 30 Abbildung der ersten drei Ebenen des ersten Entscheidungsbaumes im Random Forst Modell für GoldenEye von TensorFlow (Quelle: Eigene Abbildung)

Eine Abbildung des Modells bezüglich der verwendeten Operationen mit Netron [55] ist aus platztechnischen und Gründen der Übersichtlichkeit nicht möglich.

Anlässlich einiger Schwierigkeiten hinsichtlich der Konvertierungsmöglichkeiten in ONNX-Formate und der Verwendung des CPU anstatt der GPU für den Trainingsprozess mit TensorFlow Decision Forests [50] und den Missinterpretationen der CSV-Dateien im Falle von ML.NET [29] konnte die Frist des ersten Arbeitspaketes nicht eingehalten werden. Die Fertigstellung der Modelle erfolgte abschließend in der Woche vom 28.06.2021 bis zum 02.07.2021.

Zusammenfassend bleibt als Fazit zur Implementierung eines maschinellen Lernmodells mit C# oder herkömmlichen Methoden wie zum Beispiel der Verwendung von Python [46] und TensorFlow [56], dass in Bezug auf Zugänglichkeit der bereitgestellten Funktionen die herkömmlichen Data Science Tools aus dem Bereich der Skriptsprachen in Punkten wie zum Beispiel dem vorgelagerten Bearbeiten der Daten („Data Preprocessing“) den Bibliotheken aus der .NET-Welt weitaus überlegen sind. Ohne die Verwendung der im Rahmen dieser Thesis angefertigten Python-Skripte zur Bereinigung und Aufbereitung der Daten wäre ein Trainieren der Modelle nur mit Bordmitteln des .NET-Frameworks nicht möglich gewesen.

2.6.3.4 Implementierung eines Reverse Proxy

Der nachfolgende Unterabschnitt stellt im Rahmen des zweiten Arbeitspaketes für den Zeitraum vom 05.07.2021 bis zum 16.07.2021 die Implementierung des Reverse Proxy mit den in Abschnitt 2.6 beschriebenen Bibliotheken vor. Der Dienst für die Weiterleitung der Anfragen, wurde hierbei bereits in der Konzeptions- und Implementierungsphase der Testumgebung aus Unterabschnitt 2.6.1 gefertigt. Dennoch wird anschließend auf die grundlegenden Konzepte der Programmierung anhand von Beispielen aus dem Code in Kürze eingegangen. Am Beispiel der Codeausschnitte wird in Bezug auf das Thema, der Erstellung einer Referenzimplementierung zur Simulation eines NGINX Reverse Proxy [27] und dessen grundlegenden Funktionen mit einer zusätzlichen Erkennung von Angriffen, pro Abschnitt ein Vergleich zu den Konfigurationen innerhalb des NGINX [27] gezogen. Einerseits soll dies die bisherigen Möglichkeiten des bestehenden Open Source Produktes aufzeigen und auf der anderen Seite die zusätzlichen Features hervorherben, die mit dem programmatischen Ansatz im Rahmen dieser Thesis möglich sind.

Die Grundlage zur Einbindung der YARP-Bibliothek [30] stellt eine .NET-Webanwendung dar, die als Standardtemplate in der Basisinstallation von .NET 5 mitgeliefert wird. Die anschließende Konfiguration des Proxys bedarf der Einbindung der YARP-Middleware, die in der Startup-Klasse der Anwendung in der `ConfigureServices`-Methode eingefügt und in der `Configure`-Methode auf die bereitgestellten Endpunkte abgebildet wird.

```
1  public void ConfigureServices(IServiceCollection services)
2  {
3      ...
4      var proxyBuilder = services.AddReverseProxy();
5      proxyBuilder.LoadFromConfig(Configuration.GetSection("ML.Proxy"));
6      ...
7  }
8
9  public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
10 {
11     ...
12     app.UseEndpoints(endpoints =>
13     {
14         endpoints.MapReverseProxy();
15     });
16 }
```

Codeausschnitt 01 Basis-Implementierung in der Startup-Klasse zur Implementierung des Reverse Proxy

Die bereits angesprochenen Endpunkte werden anschließend zur Laufzeit mittels des `proxyBuilder` in Zeile 5 über das Laden der Konfiguration aus der `appsettings.json`, die zur Anpassung von Funktionen der Anwendung wiederum aus mehreren Sektionen bestehen kann, realisiert. Hierbei wird die Sektion mit der Bezeichnung „ML.Proxy“ verwendet, die ein „Mapping“ der selbstdefinierten Pfade auf die vorhandenen Services hinter dem Proxy definiert. Hierbei besteht die Möglichkeit diese Einstellungen in Form von ConfigMaps innerhalb eines Kubernetes [36] Klusters zu definieren und zur Laufzeit der Anwendung zu verändern. Dies hat den Vorteil,

dass die Anwendung somit nicht jedes Mal bei Änderungen an den dahinterliegenden Diensten neu gestartet werden muss. Jedoch besitzt die Implementierung innerhalb von .NET bezüglich des dynamischen Erkennens von Änderungen an dieser Datei in der Version 5 noch Fehler. Die ConfigMap, die als Datei in das Verzeichnis des Containers gemountet wird, ändert sich zwar zur Laufzeit, jedoch die Anwendung nimmt dies nicht zur Kenntnis. Der hiermit verbundene Fehler ist jedoch in Form eines „Pull Requests“ in der im Winter 2021 erscheinenden Version von .NET 6 adressiert und die Alarmierung der Anwendung soll in Bezug auf Änderungen in der Konfiguration stattfinden. Im nachfolgenden Codeausschnitt ist als Beispiel die Weiterleitung der Anfragen, die über den Pfad „`/weatherdata/…`“ den Proxy kontaktieren, an den Dienst zu Abfrage von zufälligen Wetterdaten.

```
1 {
2    "ML.Proxy": {
3      "Routes": {
4        "weatherApi": {
5          "ClusterId": "weatherCluster",
6          "Match": {
7            "Path": "/weatherdata/{*any}"
8          },
9          "Transforms": [
10           { "PathRemovePrefix": "/weatherdata" }
11         ]
12       },
13       ...
14     },
15     "Clusters": {
16       "weatherCluster": {
17         "Destinations": {
18           "weatherApi": {
19             "Address": "http://weatherapi"
20           }
21         }
22       },
23       ...
24     }
25   }
26 }
```

Codeausschnitt 02 Abbilden der zur Verfügung gestellten Pfade für das Weiterleiten von Anfragen

Nach diesem Muster sind die restlichen Endpunkte für die anderen Dienste im Anwendungsverbund definiert. Als Äquivalent ist im nächsten Codeausschnitt die Referenzimplementierung innerhalb eines NGINX Reverse Proxy [27] gezeigt. Einziger Unterschied ist die Definition der Ziele für die weitergeleiteten Anfragen. Diese werden bei YARP [30] in Form von Klustern definiert, hiermit lassen sich zum Beispiel mehrere Ziele definieren, die auf Basis unterschiedlicher Load-Balancing-Regeln angesprochen werden können. Jedoch sind die tiefgreifendere Konfiguration und die vorhandenen Einstellungen, die mit dieser Bibliothek zum jetzigen Zeitpunkt möglich sind, nicht Inhalt dieses Unterabschnitts.

```
upstream weatherapi {
    server weatherapi:80;
}
...
server {
...
    location /weatherdata {
        rewrite ^/weatherdata(.*)$ $1 break;
        proxy_pass http://weatherapi;
        include common_location.conf;
    }
...
}
```

Codeausschnitt 03 Konfiguration der Endpunkte innerhalb eines NGINX Reverse Proxy

Resultierend aus der vorangegangenen Implementierungsarbeit wurde mit wenigen Zeilen Code ein Substitut für den aus der Testumgebung bekannten NGINX Proxy [27] erstellt, der grundlegend dieselben Funktionen aufweist. Somit wurde die Weiterleitung der Anfragen an die restlichen Dienste, die vorher über den NGINX [27] ausgeführt wurden an diesen Proxy übergeben. Nach den vollzogenen Anpassungen werden die Anfragen, die über den NGINX [27] an den ML.Proxy übergeben werden, von diesem an die hinter ihm liegenden Dienste weitergeleitet. Basierend auf dieser Grundlage ist es nun möglich, in den Proxy eine Funktion zum Throttling von Anfragen zu integrieren.

2.6.3.5 Integration des Throttling-Mechanismus in den ML.Proxy

Aufbauend auf den Ergebnissen aus dem Unterabschnitt 2.6.3.4 bestand die nächste Aufgabe in der Implementierung eines Throttling-Mechanismus, der abhängig von vordefinierten Regeln wie im Fall des NGINX Reverse Proxy [27], ein Throttling der eingehenden Anfragen durchführt. Die verwendete YARP Bibliothek [30] stellt für die Nutzung eines Throttling-Mechanismus bzw. eines „Concurrency Limiters“ bisher nur wenige Funktionen bereit, die über die angesprochene Konfigurationsdatei gesetzt werden können. Jedoch hinsichtlich der Dynamik, die durch das ML-Modell im späteren Verlauf der Implementierung in die Entscheidungsfindung einfließen soll, sind diese starren vorabdefinierten Kriterien nicht zielführend. In Folge eines Austausches mit den Entwicklern über deren GitHub-Forum wurde auf die Existenz einer Open Source Bibliothek eines Throttling-Mechanismus für .NET-Webanwendungen mit dem Namen ThrottlR [59] hingewiesen, die näherungsweise Eigenschaften eines NGINX Proxy [27] bereitstellt. Die Implementierung dieser Bibliothek ermöglicht, entweder auf Basis einzelner Endpunkte gesteuert durch Annotationen auf der gewünschten Controller-Klasse oder generell über alle Endpunkte hinweg die Anwendung mit Throttling zu versehen. Einsatz und Nutzung der Funktionen ähneln dabei der Verwendung von YARP [30]. Durch Registrieren der bereitgestellten Middleware in der Pipeline der `Startup`-Klasse einer bestehenden Anwendung, werden die enthaltenen Eigenschaften aktiviert. Anschließend kann durch Setzen von vordefinierten Richtlinien der Mechanismus so eingestellt werden, dass er ein bestimmtes Throttling-Verhalten realisiert. Hierbei

reicht die Bandbreite von Einschränken der Anzahl an Anfragen innerhalb eines gewissen Zeitraums (siehe Zeile 9), über das Zulassen einer bestimmten Anzahl an Anfragen von spezifischen IP-Adressen innerhalb eines Zeitraums (siehe Zeile 10-11), dem Safelisting bestimmter IP-Adressen (siehe Zeile 12) und dem Safelisting von vordefinierten Nutzerrollen (siehe Zeile 13) und Headern (siehe Zeile 14) auf den eingehenden Anfragen. Der `InMemoryCounterStore` in Zeile 16 wird für das Zählen der eingehenden Requests verwendet und blockt wie in diesem Fall jeden vierten Request, der alle 10 Sekunden den Proxy erreicht. Zusätzlich anzumerken ist in Zeile 27 die `Throttle`-Methode, die auf die von YARP [30] generierten Endpunkte angewendet wird. Aufgrund der fehlenden Controller-Klassen für diese Endpunkte werden in diesem Szenario alle Endpunkte mit Throttling belegt. Zur Spezifizierung einzelner Endpunkte mit und ohne dieses Mechanismus, müssten die in der `appsettings.json` deklarierten Regeln in Form von Endpunkten in der `Startup`-Klasse deklariert werden.

```
1  public void ConfigureServices(IServiceCollection services)
2  {
3      ...
4      services.AddThrottlR(options =>
5      {
6          options.AddDefaultPolicy(policy =>
7          {
8              policy
9                  .WithGeneralRule(TimeSpan.FromSeconds(10), 3)
10                 .WithSpecificRule("10.20.10.47",
11                     TimeSpan.FromSeconds(10), 60)
12                 .SafeList.IP("127.0.0.1", "::1")
13                 .SafeList.User("Admin-User")
14                 .SafeList.Host("WeatherAPI.Local");
15         });
16     }).AddInMemoryCounterStore();
17 }
18
19 public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
20 {
21       ...
22       app.UseThrottler();
23       ...
24       app.UseEndpoints(endpoints =>
25       {
26           endpoints.MapReverseProxy()
27               .Throttle();
28       });
29 }
```

Codeausschnitt 04 Registrieren des Throttle-Mechanismus und mögliche Beispielkonfiguration

Diese als „Policy“ gekennzeichneten Regeln orientieren sich stark an den Möglichkeiten von Seiten des NGINX [27]. Lediglich ein „Blocklisting“ von IP-Adressen, die im Rahmen eines Log-Reviews entdeckt und denen verdächtige Aktivitäten zugeschrieben werden, ist nicht möglich. Die kongruenten Funktionen des NGINX [27] werden nachfolgend nicht weiter ausgeführt.

Diese zugrundeliegenden Eigenschaften werden als Fundament für die nachfolgenden Schritte verwendet. Beginnend mit der Ergänzung einer neuen Policy oder Middleware, die im Stande ist, die Modelle aus Unterabschnitt 2.6.3.3 von ihrem Speicherort zu laden, und die empfangenen

Anfragen auf Grundlage der Prognosen bzw. Klassifikationen der Modelle zu blocken oder weiterzuleiten.

2.6.3.6 Integration der Modelle in den Proxy

Nachfolgend werden nun die einzelnen Implementierungsschritte zur Integration der bereits trainierten ML.NET-Modelle beschrieben. Zeitraum für die Aufnahme der Funktionen dieses und der nachfolgenden Unterabschnitts in den vorhandenen Proxy-Dienst stellt die Zeitspanne vom 19.07.2021 bis 30.07.2021 dar. Die Aufgabe besteht darin, die Modelle von ihrem Speicherort, hier das GitHub-Repository des Autors, zur Laufzeit in die Applikation zu laden und somit keine zusätzliche Speicherkapazität innerhalb des Containers für das Vorhalten aller verwendeten Modelle zu belegen. Hierfür bietet ML.NET [29] standardmäßig die Methode `AddPredictionEngine<TIn, TOut>` an, die es ermöglicht das Modell entweder aus einem lokalen ZIP-Archiv oder aus einem Verzeichnis, welches Remote auf einem anderen Server (zum Beispiel einem Webserver) gespeichert sein kann, herunterzuladen und zu verarbeiten. Aufgrund der Verwendung der Modelle in einer .NET-Webanwendung und der somit einhergehenden Restriktion einer Umgebung, die nicht „Thread-Safe“ ist, wird zusätzlich eine erweiterte Form der bereits genannten Methode angeboten. Über die Deklaration `AddPredictionEnginePool<TIn, TOut>` lassen sich verschiedene ML-Modelle in den Speicher der Anwendung transferieren und an beliebiger Stelle über den vergebenen Modell-Namen adressieren. Zur besseren Übersichtlichkeit und Kapselung der Funktionen, sind diese Arbeitsschritte in einer für .NET-Anwendungen typischen ServiceExtension-Methode zusammengeführt, die mehrere Aufgaben zusammenfasst und als eine zentrale Methode (hier `AddMachineLearningAttackPrediction`) innerhalb der `ConfigureServices`-Methode in der `Startup`-Klasse registriert.

```
1  public static class AttackPredictionServiceExtension
2  {
3      public static void AddMachineLearningAttackPrediction(
4          this IServiceCollection services,
5          IConfiguration _configuration)
6      {
7          services
8              .AddPredictionEnginePool<GoldenEyeTrafficData,
9                  NetworkAttackPrediction>()
10             .FromUri(
11                 modelName: "GoldenEyeAttackModel",
12                 uri: _configuration.GetValue<string>(
13                     "ML.Proxy.ML-Modell:Optimized:GoldenEye:Model-Path"
14             ),
15                 period: TimeSpan.FromMinutes(10));
16         ...
17     }
18 }
```

Codeausschnitt 05 ServiceExtension-Methode zur Registrierung der Machine Learning Modelle

Hier wird beispielhaft das Laden des GoldenEye-Modells über eine injizierte URL gezeigt. Das Modell wird mit dem Namen „GoldenEyeAttackModel“ im PredictionEnginePool registriert und

kann über die Methode `Predict` in einem beliebigen Teil der Applikation über den vergebenen Namen angesprochen und somit eine Prognose für den vorgelegten Datensatz erfragt werden.

```
1 private static (bool IsGoldenEyeAttack, ...) PredictAttack(
2     PredictionEnginePool<GoldenEyeTrafficData,
3         NetworkAttackPrediction> goldenEyeModel,
4     ...)
5 {
6     var goldenEyePrediction = goldenEyeModel
7         .Predict(modelName: "GoldenEyeAttackModel", example: goldenEyeAttack);
8     ...
9 }
```

Codeausschnitt 06 Nutzen des GoldenEye-Modells in der PredictionMiddleware zur Prognose von Angriffen

Dieses Vorgehen kann anhand der hier eingeführten statischen Methode `PredictAttack`, die Teil der `PredictionMiddleware` ist (wird später noch genauer erläutert), nachvollzogen werden. Über Dependency Injection wird der `PredictionEnginePool` mit dem GoldenEye-Modell in die `PredictionMiddleware` des ML.Proxy induziert. Von dort aus wird das Modell in der `PredictAttack`-Methode über den Namen adressiert und ein Datensatz, der eine eingehende Anfrage darstellt, übergeben. Als Rückgabewert wird in der Variable `goldenEyePrediction` ein Objekt vom Typ `NetworkAttackPrediction` gespeichert. Hierbei handelt es sich um eine selbstdefinierte Klasse mit einem Feld namens `PredictedLabel` vom Typ `Boolean`. Ist das Modell der Meinung, es handelt sich bei der vorliegenden Anfrage um einen Angriff, dann wird das Feld mit `true` besetzt, ansonsten `false`. Dabei ist die Einschätzung des GoldenEye-Modells nur ein Teil des Rückgabewertes der hier gezeigten Methode. Innerhalb von `PredictAttack` werden alle drei vorhandenen Modelle (GoldenEye, LOIC und Slowloris) einzeln um eine Einschätzung gebeten und abschließend ein `Tuple` zusammengesetzt aus drei booleschen Werten zurückgegeben. Hierauf basiert anschließend die Entscheidung des Throttlings im ML.Proxy. Der genauere Aufbau dieses Mechanismus ist Teil des nächsten Unterabschnitts, das sich mit der Implementierung der Anfragenverarbeitung und dem eigentlichen Throttling beschäftigt.

2.6.3.7 Dynamisches Throttling auf Basis der ML-Prognosen

Inhalt dieses Unterabschnitts ist die Konzeption der unterschiedlichen `Service`-Klassen, die die Funktionen für das Abhören des Netzwerkverkehrs, dessen Verarbeitung und dem anschließenden Vorhalten in einem Cache beinhalten. Im Folgenden wird die Funktionsweise des `CaptureTrafficService` beschrieben, der das Abhören des Netzwerkverkehrs, der Datengrundlage für die Erstellung der unterschiedlichen Klassen für die Prognose der Attacken, gewährleistet. Das Mitschneiden von Netzwerkpaketen in einer Applikation im Betrieb und vor allem einer Webapplikation, deren Hauptfunktion im Regelfall nicht für diese Zwecke ausgelegt ist, stellte sich vorerst als potenzielles Problem bei der Implementierung heraus. Aus dem Hause Microsoft sind bisher keine Bibliotheken oder Frameworks veröffentlicht worden, die diese Art

von Funktionen für Anwendungen im Web zur Verfügung stellen. Jedoch sind von Seiten der Community zwei Bibliotheken, SharpPcap [60] und Packet.Net [61] in der jeweils aktuellsten Version, veröffentlicht worden, die es ermöglichen innerhalb von Konsolen-Anwendungen auf dem ausführenden Rechner interaktiv den Netzwerkverkehr mitzuschneiden. Hierfür bedienen sie sich der Software Npcap [62] unter Windows oder libpcap [63] unter Linux bzw. Unix, um die vorhandenen Netzwerkschnittstellen ansprechen zu können. Da es sich bei der späteren Laufzeitumgebung der Anwendung um Docker Container [64] mit einer Debian-Distribution als Betriebssystem bzw. Operating System (nachfolgend auch OS genannt) handelt und diese letztlich Linux-Serverumgebungen darstellen, war es möglich mittels dieser beiden Bibliotheken und `libpcap-dev`, das beim Bauen des Images auf dem OS installiert wurde, innerhalb des `CaptureTrafficService` den Netzwerkverkehr auf dem Standardinterface (`eth0`) des Containers mitzuschneiden. Ergebnis ist ein Objekt der Klasse `RawPacketCapture`, die alle Daten des mitgeschnittenen Verkehrs pro Anfrage enthält. Dieser Output dient wiederum als Input für den `RequestProcessingService`. In dieser Service-Klasse werden die Daten des Pakets in eine `NetworkAttack`-Klasse überführt. Bei den Werten, die zur Erstellung der verwendeten Datensätze für das Training der Modelle eingesetzt wurde, kam wie bereits in Unterabschnitt 2.6.3.2 angesprochen, die Anwendung CICFlowMeter in der Version 3 [41, 42] zum Einsatz. Mithilfe dieser Java-Anwendung wurden die Daten, die abschließend in den CSV-Dateien abgespeichert wurden, mittels der Werte aus den Log- und Pcap-Dateien kalkuliert. Die nächste Aufgabe bestand nun darin, die Daten der Pakete annäherungsweise so zu bestimmen, dass sie den ursprünglichen Eingabedaten von der Form entsprachen und so einen validen Input für das ML-Modell darstellen. In Anbetracht der Tatsache, dass es sich bei CICFlowMeter um eine Open Source Software handelt und der Quellcode öffentlich einsehbar ist, konnte aufgrund dessen die Berechnung basierend auf den genutzten Java-Funktionalitäten auf eine äquivalente Kalkulation in C# adaptiert werden.

```
1  var actualPacketBinaryTimestamp = request.Timeval.Date.ToBinary();
2  var lastPacketBinaryTimestamp = timestampLastPacket.ToBinary();
3  var initialPacketBinaryTimestamp = initialTimestamp.ToBinary();
4
5  var iatCalculation = actualPacketBinaryTimestamp - lastPacketBinaryTimestamp < 0 ?
6      0 : actualPacketBinaryTimestamp - lastPacketBinaryTimestamp;
7  var flowCalculation = lastPacketBinaryTimestamp - initialPacketBinaryTimestamp < 0 ?
8      0 : actualPacketBinaryTimestamp - initialPacketBinaryTimestamp;
9
10 var networkAttack = new NetworkAttack()
11 {
12     FlowIATMin   = iatCalculation,
13     FwdIATMin    = iatCalculation,
14     FlowIATMean  = iatCalculation,
15     PktSizeAvg   = packet.PayloadPacket.Bytes.Length,
16     FlowDuration = flowCalculation,
17     FlowIATStd   = iatCalculation,
18     FwdIATMean   = iatCalculation,
19 };
```

Codeausschnitt 07 Berechnung der Größen für das Mapping auf die NetworkAttack-Klasse

Letztlich konnte bis auf `PktSizeAvg` und `FlowDuration` für die restlichen Properties dieselben Werte verwendet werden. Dies ist legitim, da es sich bei den ursprünglichen Werten um historische Werte handelt, die auf Basis von Minima, Maxima oder Durchschnitten berechnet werden. Ähnelt ein eingehendes Paket diesem errechneten Wert, dann kann sich das Modell sicher sein, dass es sich um einen validen Angriff handelt. Aus diesem Grund ist die Verwendung einer Historie diesbezüglich nicht notwendig. Einzig die Speicherung der initialen Anfrage eines Flows muss für die Berechnung der `FlowDuration` persistiert werden. Hierzu wurde der `RedisCacheService` implementiert, der eine Abstraktionsschicht mit Lese und Schreiboperationen zur Nutzung eines verteilten Caches, in diesem Szenario Redis [65], darstellt. Die Entscheidung für den Gebrauch eines Cache und keiner Datenbank basiert primär auf der Flüchtigkeit der benötigten Daten. Es ist anzunehmen, dass Anfragen von unterschiedlichen Nutzern nur in bestimmten Zeitintervallen und vereinzelt auftreten und kein Nutzer für die Dauer von über einer Stunde im Sekundenrhythmus Anfragen an den Dienst schickt, wie es zum Beispiel ein Bot tun würde. Somit sind Daten, die vor drei Stunden für diesen Nutzer erfasst wurden für den nächsten Flow nicht mehr relevant. Wird ein Angriff erkannt, dann wird der ML-Dienst frühzeitig Gegenmaßnahmen einleiten. Hieraus resultierte die Entscheidung die „gecachten" Daten nur für einen Zeitraum von maximal 5 Minuten im Cache vorrätig zu halten. Werden dieselben Einträge mehrmals verwendet, dann wird die Ablaufzeit bei jedem Zugriff erneuert (zum Beispiel im Fall eines Angriffs). Dadurch wird neben einer höheren Performanz des Zugriffs auch eine schnellere Verarbeitung der eintreffenden Anfragen ermöglicht. Ein weiterer tragender Aspekt für die Verwendung von Redis [65] ist die spätere Zielinfrastruktur. Da es sich hierbei um die Architektur eines verteilten Systems und die Nutzung von replizierten Einheiten, um ein und desselben Dienstes handelt, benötigt eine Anwendung einen skalierbaren Speicher, der sich den Gegebenheiten anpassen kann. Mittels des verwendeten Cache sind eine spätere Skalierbarkeit und das verteilte Speichern von Daten und den Zugriff darauf bereits gegeben. Anlässlich des `IDistributedCache`-Interface, das Seitens .NET zur Verfügung gestellt und mittels der Bibliothek `StackExchangeRedis` [66] gekapselt wird, kann ein Höchstmaß an Leistung garantiert werden. Der `RedisCacheService` wird wiederum innerhalb des `PacketService` verwendet. Innerhalb des Proxy-Dienstes übernimmt der `PacketService` die Verwaltung der eingehenden Pakete und schreibt diese mit einem personalisierten Cache-Key, welcher aus binärem Zeitstempel des Eingangs am Proxy und der IP-Adresse des Absenders besteht, und dem Value, der die `RawPacketCapture`-Klasse serialisiert als JSON-String, enthält in den Zwischenspeicher. Das letzte Paket und das initiale Paket für jeden anfragenden Nutzer wird mit „`First-Packet+IP`" und „`Last-Packet+IP`" hervorgehoben. „`Last-Packet`"-Einträge werden mit jedem neuen Eingang eines weiteren Pakets von derselben Adresse

überschrieben und mit der bereits genannten Kombination aus Zeitstempel und IP-Adresse überschrieben. Folglich werden die eingehenden Anfragen auf Basis der IP-Adresse vor der Auswertung durch das ML-Modell separiert und eine Fehlverteilung der Pakete zu den Anfragenden ist somit ausgeschlossen. Die Extraktion der IP-Adresse findet durch die statische Methode `GetClientIPAddress` statt, die innerhalb der unterschiedlichen `Service`-Klassen implementiert ist und die Adresse aus dem „`X-Forwarded-For`"-Header liest und als `string` zurückgibt. Gesetzt wird dieser Header durch den vorgelagerten Reverse Proxy bzw. Ingress im Kluster und garantiert folglich die richtige Zuordnung des Response an den Initiator. Konkludiert in der bereits genannten `PredictionMiddleware` werden alle eingehenden Anfragen mittels der Services bearbeitet und schlussendlich durch die Modelle bewertet. Wird ein Angriff erkannt, wird innerhalb der Middleware auf den bearbeitenden Request ein zusätzlicher Header in der Form „`Attack-Prediction-Header`" mit dem Wert „`true`" gesetzt. Andernfalls enthält er den Wert „`false`". Grund für die Entscheidung des Setzens eines spezifischen Headers ist, dass innerhalb der Request-Pipeline die Weitergabe von Werten, wie zum Beispiel eines Boolean, nicht möglich ist. Einzig das Request-Objekt in Form der Klasse `HttpContext` wird von einer Middleware an die nächste weitergegeben. Infolgedessen wurde die ThrottleR-Middleware im Rahmen dieser Thesis so angepasst, dass sie auf das Vorhandensein des neu eingeführten Request-Headers reagiert und daraufhin das Throttling bzw. ein Blockieren des weiteren Netzwerkverkehrs einleitet.

```
1 if (context.Request.Headers.TryGetValue("Attack-Prediction-Header",
2     out StringValues prediction) && Convert.ToBoolean(prediction))
3 {
4     LogDetectedAttack(scope);
5
6     await BlockIncomingAttack(context);
7
8     return;
9 }
```

Codeausschnitt 08 Logik innerhalb der ThrottlR-Middleware für das Erkennen von Angriffen

Mithilfe dieser Prüfung wird am Ende der `ThrottlerMiddleware` ein Angriff auf die dahinterliegenden Anwendungen abgewehrt. Die Middleware für die Prognose des Angriffs wird aus diesem Grund vor der `UseThrottler`-Extension in der `Configure`-Methode in der `Startup`-Klasse durch die `MiddlewareBuilderExtension` `UseAttackPredictionMiddleware` registriert und ist somit vor der `ThrottlerMiddleware` bezüglich der Anfragenbearbeitung geschaltet. Nachfolgend werden nun Ergänzungen zu der in Unterabschnitt 2.6.1 definierten Konzeption des Diensts in Form von Safe- und automatisierten Blocklisting vorgestellt, die dem Dienst weitere Fähigkeiten verleihen, die ein NGINX Reverse Proxy [27] nur hinreichend erfüllen kann.

2.6.3.8 Integration von Safe- und automatisierten Blocklisting

Anlässlich der Bereitstellung von „Safeflisting“ durch die ThrottlR-Bibliothek [59] ist durch die Verwendung des bereits angesprochenen Zwischenspeichers (Cache) die Möglichkeit für den Einsatz einer eigenen Safe- und Blocklist für den ML-Teil des Dienstes entstanden. Hiermit soll es dem Dienst möglich sein, erkannte Angreifer und deren IP-Adressen frühestmöglich in eine Blocklist (neue Deklaration der Blacklist, die aufgrund ethnischer und politischer Gründe von „black“ in „block“ umbenannt wurde) zu verschieben, um somit die erneute Verarbeitung der eingehenden Anfragen durch die ML-Modelle und die hiermit verbundenen und vermeidbaren Lastspitzen zu verhindern. Im Gegenzug soll es jedoch auch möglich sein durch Einwirkung von außen an abgesicherten Endpunkten innerhalb des ML.Proxy IP-Adressen gezielt zu „safelisten“ und anerkannte Nutzer des Systems als sichere Quellen von Anfragen zu registrieren. Dasselbe gilt für das manuelle Entfernen von Nutzern, die aufgrund vereinzelt auftretender Falschzuordnung in die Blocklist geraten sind. Dies ermöglicht zur Laufzeit der SaaS-Anwendung über hierfür konzipierte Oberflächen gezielt das Verhalten des ML.Proxy zu steuern und nicht wie im Falle von NGINX [59] den Proxy durch eine SSH-Verbindung auf den betreffenden Server und das Editieren der einzelnen Konfigurationsdateien auf die Bedürfnisse der Anwendung anzupassen.

Zur Umsetzung dieser Funktionalitäten sind neben den bereits genannten `Service`-Klassen aus den vorherigen Unterabschnitten noch der `IPSafelistService` und der `IPBlocklistService` ergänzt worden. Beide Services bedienen sich dem `RedisCacheService` und bearbeiten mit Lese- und Schreiboperationen die als eine Liste von `strings` gespeicherten und dem Cache-Key „`Safelist`“ und „`Blocklist`“ versehenen Einträge im Cache. Diese Überprüfung findet somit zu Beginn der `PredictionMiddleware` statt und soll die bereits angesprochenen Lastspitzen bei der Evaluierung der Anfragen durch die Modelle verhindern. Das Vorhandensein der IP-Adressen in einer der beiden Listen führt dazu, dass das Ausführen der Middleware an dieser Stelle abgebrochen wird, sodass anschließend der „`Attack-Prediction-Header`“ mit dem Wert „`true`“ im Blocklist-Fall oder „`false`“ im Fall der Safelist hinzugefügt wird und die nachgeschaltete `ThrottlerMiddleware` die Anfrage an die dahinterliegenden Dienste weiterreicht oder blockiert.

Für das manuelle Bearbeiten der beiden Listen sind zwei Webcontroller, der `Safelist`- und der `BlocklistController`, der Anwendung hinzugefügt worden. Die beiden Controller-Klassen stellen die nachfolgenden Operationen bereit.

Tabelle 20 Auflistung der vorhandenen Endpunkte zur Bearbeitung der Safe- und Blocklist

Pfad	Methode	Parameter	Beschreibung
SafelistController			
`/api/safelist/`	**GET**	-	Auslesen aller vorhandenen Einträge der Safelist
`/api/safelist/add?ip=<param>`	**POST**	IP-Adresse	Hinzufügen einer IP-Adresse zur Safelist
`/api/safelist/add?ip=<param>`	**DELETE**	IP-Adresse	Löschen einer IP-Adresse von der Safelist
BlocklistController			
`/api/blocklist/`	**GET**	-	Auslesen aller vorhandenen Einträge in der Blocklist
`/api/blocklist/add?ip=<param>`	**POST**	IP-Adresse	Hinzufügen einer IP-Adresse zur Blocklist
`/api/blocklist/remove?ip=<param>`	**DELETE**	IP-Adresse	Löschen einer IP-Adresse von der Blocklist

Diese Endpunkte sind im Rahmen dieser Thesis mittels des bereits vorhandenen AS abgesichert worden und lassen sich nur mittels eines vorhandenen Bearer-Tokens mit den entsprechenden Scopes ansteuern. Eine Auflistung der Scopes wird hier nicht erfolgen, aufgrund der Möglichkeit einer späteren Anpassung oder Abänderung hinsichtlich existierender Rechte in der gewünschten Zielarchitektur. Zusätzlich ist anzumerken, dass die Implementierung einer Authentifizierung und Autorisierung am ML.Proxy nicht Bestandteil dieser Arbeit ist und nur beispielhaft für eine mögliche spätere Verwendung eingefügt wurde. Nach den abschließenden Implementierungsschritten ist die Anwendung als Docker Container [64] gebaut in der Container Registry des Azure Kubernetes Kluster [36] zusammen mit den restlichen Anwendungen platziert und zur Simulation des Produktivbetriebes auf dem Kluster ausgerollt worden. Diese Simulationsumgebung wird nun nochmals zur Analyse mit den bereits vorgestellten DoS- bzw. DDoS-Tools aus Unterabschnitt 2.5.1 verwendet und die Implementierung auf ihre Wirksamkeit und den dadurch entstandenen Nutzen hin im Rahmen dieser Arbeit ausgewertet.

2.6.4 Evaluation und Validierung der Implementierung

Mit Abschluss der Implementierungsarbeiten und dem Platzieren des Anwendungsverbundes auf der Testumgebung, folgt in diesem Unterabschnitt die abschließende Evaluation des erarbeiteten Sicherheitskonzeptes. Auf Basis der Ergebnisse aus Unterabschnitt 2.5.1 stellte sich heraus, dass die DoS- bzw. DDoS-Tools Low Orbit Ion Cannon (LOIC) [23] und Slowloris [25] nicht im Stande waren eine Verbindung mit den Diensten hinter dem Proxy aufzunehmen. Dies resultierte aus den Analysen der einzelnen Log-Nachrichten, die während der Angriffe aus dem vorgeschalteten Reverse Proxy gesammelt wurden. Aufgrund der fehlenden Funktionalität zum Absetzen von Requests über TLS 1.2 mittels der LOIC und der Reaktion des Proxys auf den unverhältnismäßig langen Timeout des Slowloris Clients mit einem Verbindungsabbruch, war nur noch GoldenEye [24] als potentielle Gefahrenquelle für die SaaS-Applikation zu identifizieren. Dementsprechend wurde auch nur dieses Tool für die Analyse des Nutzens des ML.Proxy für den Schutz der Anwendung herangezogen. Zuvor war jedoch die Überprüfung der Anwendung und der grundsätzlichen Funktionen wie Login, Logout und das Abfragen der Wetter- und Standortdaten der zwei Beispiel-APIs vorangestellt. Diese Auswertung, durchgeführt durch den Verfasser dieser Thesis, fand im Zeitraum des letzten Arbeitspaketes in der Woche vom 26.07.2021 bis 30.07.2021 statt und erstreckten sich zur Absicherung der Ergebnisse und der Protokollierung innerhalb der Arbeit in die Woche vom 02.08.2021 bis 06.08.2021. Ergebnis war, dass mit der bisherigen Architektur, die in Unterabschnitt 2.6.2 vorgestellt wurde, das Modell auf das initiale Laden der SPA in den Browser des Clients bereits mit einer Abwehrreaktion antwortete. Eine mögliche Erklärung hierfür sind die im Vergleich zur späteren Nutzung unverhältnismäßig großen Datenpakete, die über den Proxy an den anfragenden Client gesendet werden. Da es sich bei der SPA um eine Anwendung handelt, die im Browser des Clients verwendet wird und somit nicht mittels des Proxys im späteren Verlauf der Verwendung geschützt werden kann, wurde die Architektur nochmals überarbeitet. Letztlich erfolgte eine Umplatzierung der SPA vor den ML.Proxy. Demnach kommuniziert diese nun über den Proxy-Service mit den dahinterliegenden Diensten, die in diesem Fall die zu schützenden Ressourcen darstellen. Nachfolgend wird die aktualisierte Version des Anwendungsverbundes nochmals beispielhaft dargestellt.

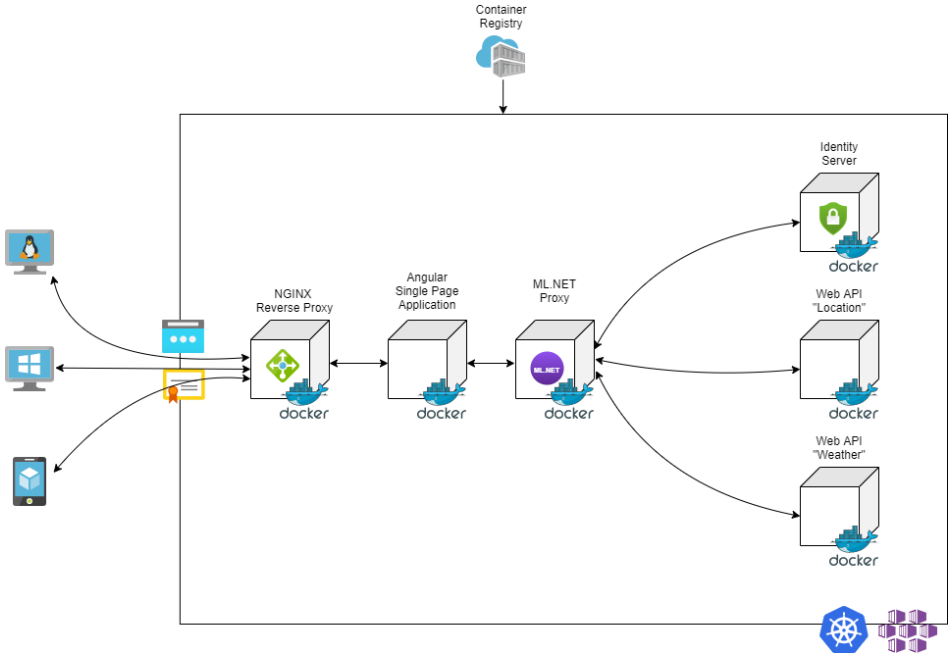

Abbildung 31 Aktualisierte Version der Anwendungsarchitektur

Mittels dieser kleinen Änderung trat das bisherige Kennzeichnen von nicht-malignen Angriffen hinsichtlich des Ladens der SPA nicht mehr auf. Einzig der Fall einer gespeicherten validen Session im Browser, die die SPA automatisch am AS mit den vorgehaltenen Session-Cookies authentifiziert und autorisiert, ruft in vereinzelten Fällen eine falsche Einschätzung des Modells hervor. Auch dieses Verhalten lässt sich in ähnlicher Weise wie das Laden der SPA in den Browser erklären. Durch das Setzen der Session-Cookies und dem mehrmaligen Redirect zwischen den Anwendungen, wächst die initiale Paketgröße in kurzer Zeit auf einen Wert, der dem eines validen Angriffs ähnelt. Ob diese Art von Netzwerkverkehr Anwendung in den verwendeten Datensätzen gefunden hat, ist aus der beiliegenden Dokumentation nicht ersichtlich. Aus diesem Grund muss davon ausgegangen werden, dass die Durchführung von Authentifizierungs- und Autorisierungsverfahren nicht Teil der Netzwerkmitschnitte innerhalb des Testszenarios des Datensatzes war. Hingegen sind diese Anomalien nur vereinzelt aufgetreten und lassen sich durch Mechanismen wie zum Beispiel einer Safelist, beschrieben in Unterabschnitt 2.6.3.8 präventieren. Im Übrigen werden die restlichen Anfragen als Nicht-Angriffe richtig identifiziert und an das Backend der Anwendung weitergeleitet.

Darauffolgend schloss die Auswertung der Reaktion des ML.Proxy auf einen validen Angriff mittels GoldenEye [24] an. Aufbau und Durchführung des Versuches ähneln den Tests aus den vorherigen Unterabschnitten. Nur wird in diesem Fall nicht der Endpunkt des AS zur Abfrage von Informationen zu dem angemeldeten Nutzer attackiert, sondern der Endpunkt der Wetter-API. Zur Auswertung der Anzahl der abgesendeten Anfragen, die über das DoS-Tool abgesetzt

wurden und vom ML.Proxy als Angriff erkannt und anschließend geblockt werden, wurde der Mechanismus des automatischen Blocklistings für diesen Teil des Tests deaktiviert. Ansonsten setzt das Blocklisting zu früh ein und die nachfolgenden Anfragen erscheinen nicht mehr als Nachrichten in den Logs der Dienste. Zur Analyse der Log-Nachrichten, die die Einschätzung der ML-Modelle wiedergeben, wurde hierfür die Kombination der Anwendungen Elasticsearch [67] für das Sammeln der Logs im Kluster und zur Visualisierung Kibana [68] verwendet, die im Zusammenspiel mit Serilog [69] einer Logging-Bibliothek für .NET-Anwendungen häufig zum Einsatz kommen. Die Testreihe ohne Blocklisting wird in der Tabelle 21 kurz dargestellt.

Tabelle 21 Ergebnisse der Versuchsreihe von Angriffen ohne die Blocklisting-Funktion

Testlauf	Falsche Prognose	Richtige Prognose	Anfragen gesamt	Präzision
1	12.202	2.900	12.492	23,21 %
2	12.267	2.347	14.614	16,06 %
3	13.049	1.612	14.661	10,99 %
4	12.394	2.714	15.108	17,96 %
5	12.659	2.447	15.106	16,20 %
Ø	12.514,2	2.404	14.396,2	16,69 %

Das Ergebnis der Testreihe ohne Blocklisting war somit, dass das Modell mit einer Präzision von rund 20% der eingehenden Angriffe, diese als valide Angriffe kennzeichnet und somit von der implementierten Logik abgewehrt werden. Die hier erreichte Rate ist zwar nicht besonders hoch, jedoch kann sie mit dem nachfolgenden Test, der mit dem automatischen Blocklisting arbeitet, neue Erkenntnisse bezüglich des Zeitraums der Erkennung eines Angriffs liefern.

Nachfolgend nun die Testreihe mit Blocklisting. Hierbei liegt der Schwerpunkt nicht darauf wie viel Anfragen von den eingegangenen Requests als valide Angriffe gekennzeichnet wurden, sondern nach wie vielen Anfragen der ML.Proxy reagiert und die IP-Adresse des Angreifers blockiert.

Tabelle 22 Werte für die Versuchsreihe von Angriffen mit Blocklisting-Funktion

Testlauf	Erfolgreiche Angriffe	Angriffe geblockt	Angriffe gesamt	Quote
1	42	13.608	13.650	99,69 %
2	118	29.100	29.218	99,60 %
3	585	28.316	28.901	97,98 %
4	384	27.594	27.978	98,63 %
5	15	30.296	30.311	99,95 %
Ø	228,8	25.782,8	26.011,6	99,12 %

Resultierend ergab sich eine Quote von rund 99 % aller Angriffe, die an das Backend gerichtet waren und nicht ihr Ziel erreichen konnten. In Bezug auf die These am Ende des letzten Tests, liegt die Vermutung nahe, dass der ML.Proxy schon in einem relativ frühen Stadium eines Denial of Service mit dem verwendeten Tool auf diesen Angriff reagiert. Im Vergleich zu der erreichten Präzision von rund 20 % am Ende des vorherigen Tests können somit trotzdem effektiv Angriffe erkannt und abgewehrt werden.

Nach Abschluss der Evaluation und Validierung der bestehenden Implementierung konnte nach Betrachtung aller gesammelten Informationen folgendes Fazit gezogen werden. Auf Basis von maschinellen Lernverfahren und geeigneten Datensätzen für deren Training kann bereits eine funktionsfähige Detektion und Prävention von gängigen Angriffsvektoren, wie hier zum Beispiel DoS, auf Anwendungen in der Cloud etabliert werden. In Anbetracht der formulierten wissenschaftlichen Frage, ob eine vergleichbare Implementierung zu einem NGINX Reverse Proxy [27] unter Verwendung von Technologien aus dem Bereich .NET möglich sei, ist das Ergebnis dieser Arbeit eine mögliche Alternative, die mit fortschreitender Entwicklung der verwendeten Bibliotheken eine Ergänzung zu bereits existierenden Architekturen darstellt. Aufgrund der Tatsache, dass es sich bei YARP [30] noch um eine Bibliothek im Preview-Status handelt und diese noch nicht den Umfang von Funktionen eines bereits etablierten Produktes hat, sind hinsichtlich Performanz im direkten Vergleich noch keine konkreten Aussagen möglich. Nur unter Einsatz von Drittbibliotheken aus der Community, wie zum Beispiel ThrottlR [59], sind Konfigurationen ähnlich des Referenzmodells durchführbar. Aus der Produkt-Roadmap ist jedoch ersichtlich, dass der Funktionsumfang noch weit nicht ausgeschöpft ist und durch Unterstützung der bisherigen Community Vorschläge zu neuen Features und Verbesserungen zu den bereits Existierenden die Entwicklungsarbeit unterstützen.

3 Reflexion und Ausblick

Auf der Grundlage der Bewertung durch den Kriterienkatalog C5 [14] des BSI und den daraus isolierten Kriterien, die im Rahmen des vorliegenden Szenarios selektiert wurden, konnte im Kontext dieser Thesis der bisherige Stand der vorhandenen Sicherheitskonzepte der STP-Cloudanwendung „LEXolution.FLOW“ skizziert werden. Die hierdurch entstandene Perspektive auf die Entwicklungsarbeit und den Entwicklungsprozess des Produktes kann in Zukunft zur Optimierung des Arbeitsflusses und der Bearbeitung von sicherheitskritischen Aspekten der Anwendung genutzt werden. Unter Einsatz des zusammengestellten Maßnahmenkatalogs zur Verbesserung der erfüllten Standards, wird dem Anwendenden die Möglichkeit geboten einzelne Schwachstellen oder fehlende Richtlinien im zukünftigen Arbeitsprozess zu korrigieren oder neu zu konzipieren. Des Weiteren wird mit den übrigen Kriterien, die nicht integraler Bestandteil der Beurteilung waren, die Möglichkeit geboten ergänzende Bewertungen durchzuführen und Axiome zu den bereits bestehenden Bedingungen hinzuzufügen. Aus dem Ergebnis der Bewertung konnte ein mögliches Sicherheitskonzept für den Schutz von SaaS-Lösungen abgeleitet werden, welches speziell Nutzer von fremder bzw. angemieteter Infrastruktur für den Betrieb deren Anwendungen in die Lage versetzt, die konzipierte Lösung mit einem sich selbst regulierenden Schutzmechanismus auszustatten. In Anbetracht der unterschiedlichen Angriffsmuster stellt dieser Schutz jedoch nur einen Teilaspekt der möglichen Sicherheitsvorkehrungen dar. Neben automatisierten Angriffen, wie zum Beispiel der hier behandelten DoS-Angriffe, ist das Spektrum zur Infiltration und Beeinträchtigung der Funktionsweise von im Internet betriebener Software facettenreicher. Zudem stellte sich heraus, dass die Erkennung dieser Art von Angriffen durch die Verwendung der aktuellsten Techniken der Authentifizierung und Autorisierung und Frameworks zur Entwicklung von Webanwendungen, nicht immer fehlerfrei durchführbar ist. Infolge der Menge an Daten, die durch das initiale Laden von Anwendungen in den Browser der Nutzer und die Anzahl an Umleitungen (Redirects), veranlasst durch die verwendeten Authentifizierungsmechanismen, von unterschiedlichen Teilen der Servicelandschaft, sind maschinelle Lernmethoden als einziges Mittel zur Detektion und Prävention von Angriffen anhand des Beispiels der hier vorliegenden Arbeit nicht ausreichend. In diesem Zusammenhang ist eine Synthese unterschiedlicher Konzepte der IT-Sicherheit zur Abwehr von Angriffen effizienter. Wohingegen das im Rahmen dieser Arbeit entstandene Konzept eine Möglichkeit bietet, durch Weiterentwicklung der verwendeten Features und Rekonstruktion der eingesetzten Modelle auf Basis aktualisierter Datensätze für deren Training und eine somit noch genauere Einschätzung der ML-Modelle für das vorliegende Szenario zu ermöglichen. Dies lässt darauf schließen, dass auch die Detektion anderer Angriffe mittels der Kombination der hier verwendeten

Techniken innerhalb von Anwendungen im Produktivbetrieb etabliert werden kann. Hiermit werden Perspektiven geschaffen, Attacken, die nicht auf Basis der Firewall eines Rechenzentrums aufgedeckt werden, durch ein sogenanntes „Self-Healing“ innerhalb einer Anwendung bzw. Infrastruktur zu behandeln. Im Zuge dessen kann der Betreiber für seine Anwendung neben einer Kostenersparnis zusätzlich eigene Sicherheitskriterien implementieren, die unabhängig und in Koexistenz mit den Sicherheitsniveaus des gemieteten Rechenzentrums Anwendung finden. Das Konzept des „Self-Healing“ setzt dabei herkömmlicherweise an der Infrastrukturebene an und schafft durch die Verwendung von zum Beispiel Machine Learning, wie im Rahmen dieser Arbeit, eine Alternative zur manuellen Fehlerbehandlung. Für diesen Zweck sind die Modelle so trainiert, dass sie auf Anomalien während des Betriebs von zum Beispiel Serverlandschaften, auf den Ausfall von einzelnen Komponenten geeignet reagieren. Dies kann beispielsweise durch gezieltes Herunterfahren einzelner Serverblöcke oder durch ein Umleiten von Netzwerkverkehr in andere Bereiche der Infrastruktur erfolgen. Dennoch funktioniert das hier umgesetzte Konzept nach demselben Prinzip. Neben einer Selbstregulation können demzufolge auch alarmierende Funktionen übernommen werden, die bei unüblichem Verhalten unter anderem verantwortliche Mitarbeiter für den betreffenden Bereich auf dem schnellsten Wege alarmieren und eine effektivere und zielgerichtetere Reaktion der Betreiber ermöglichen. Hinsichtlich dieser Herangehensweise könnten weitere Positiv-Beispiele für den Einsatz eines solchen Mechanismus aufgeführt werden, andererseits sind die Nachteile, die mit der Verwendung dieser Techniken einhergehen, nicht zu vernachlässigen. Wie es aus den Beispielen der Evaluation der Implementierung hervorgegangen ist, sind durchaus unerwartete Fälle aufgetreten, die nur unter Verwendung von Machine Learning zu einer Fehlfunktion der Anwendung durch den Benutzer geführt hätten. Diese Fälle gilt es, wie bereits angesprochen, durch eine Kombination aus unterschiedlichen Praktiken der IT-Sicherheit, wie beispielsweise das verwendete Safe- und Blocklisting, soweit zu regulieren, dass in der Mehrheit der Fälle keine Beeinträchtigung der Funktion für den Anwender entsteht. Diese Abschätzung gilt es im Rahmen der Konzipierung bei dieser Art von Diensten zu beachten. Schließlich darf die gesteigerte Sicherheit nicht zur Minderung der Verwendbarkeit führen.

4 Literaturverzeichnis

[1] C. Duffy, *So you're one of 533 million in the Facebook leak. What now?* [Online]. Verfügbar unter: https://edition.cnn.com/2021/04/06/tech/facebook-data-leaked-what-to-do/index.html (Zugriff am: 8. April 2021).

[2] o.V., *Mehrere Schwachstellen in MS Exchange.* [Online]. Verfügbar unter: https://www.bsi.bund.de/SharedDocs/Cybersicherheitswarnungen/DE/2021/2021-197772-1132.pdf?__blob=publicationFile&v=4 (Zugriff am: 8. April 2021).

[3] o.V., *STP Informationstechnologie AG.* [Online]. Verfügbar unter: https://ka.stadtwiki.net/STP_Informationstechnologie_AG (Zugriff am: 6. Mai 2021).

[4] okta Developer, *OAuth 2.0 and OpenID Connect Overview.* [Online]. Verfügbar unter: https://developer.okta.com/docs/concepts/oauth-openid/#recommended-flow-by-application-type (Zugriff am: 9. August 2021).

[5] Nat Sakimura, John Bradley und Naveen Agarwal, *Proof Key for Code Exchange by OAuth Public Clients,* Request for Comments. RFC Editor. Verfügbar unter: https://rfc-editor.org/rfc/rfc7636.txt.

[6] o.V., *Was ist ein DDoS-Angriff?* [Online]. Verfügbar unter: https://www.cloudflare.com/de-de/learning/ddos/what-is-a-ddos-attack/ (Zugriff am: 28. Mai 2021).

[7] C. Eckert, *IT-Sicherheit, 10th Edition,* 10. Aufl. De Gruyter, 2018.

[8] A. Squicciarini, D. Oliveira und D. Lin, „Cloud Computing Essentials“ in *Cloud computing security: Foundations and challenges*, J. R. Vacca, Hg., Boca Raton, FL: CRC Press, Taylor & Francis Group, 2021, S. 3–11.

[9] W. Stallings, „An Overview of Cloud Computing“ in *Cloud computing security: Foundations and challenges*, J. R. Vacca, Hg., Boca Raton, FL: CRC Press, Taylor & Francis Group, 2021, S. 13–29.

[10] R. Mogull *et al., Security Guidance: For Critical Areas of Focus In Cloud Computing v4.0.* [Online]. Verfügbar unter: https://cloudsecurityalliance.org/artifacts/security-guidance-v4/ (Zugriff am: 6. Mai 2021).

[11] P. Wilmott, *Grundkurs Machine Learning,* 1. Aufl. Bonn: Rheinwerk Computing, 2020.

[12] M. Guillame-Bert, S. Bruch, J. Gordon und J. Pfeifer, *Introducing TensorFlow Decision Forests.* [Online]. Verfügbar unter: https://blog.tensorflow.org/2021/05/introducing-tensorflow-decision-forests.html (Zugriff am: 6. Juli 2021).

[13] P. Pandya und R. Rahmo, „Cloud Computing Architecture and Security Concepts“ in *Cloud computing security: Foundations and challenges*, J. R. Vacca, Hg., Boca Raton, FL: CRC Press, Taylor & Francis Group, 2021, S. 214–223.

[14] Bundesamt für Sicherheit in der Informationstechnik, *Cloud Computing Compliance Criteria Catalogue - C5:2020: Kriterienkatalog Cloud Computing.* [Online]. Verfügbar unter: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Empfehlungen-nach-Angriffszielen/Cloud-Computing/Kriterienkatalog-C5/C5_AktuelleVersion/C5_AktuelleVersion_node.html (Zugriff am: 6. Mai 2021).

[15] Bundesamt für Sicherheit in der Informationstechnik, *C5:2020: SaaS-Fallstudie: Anwendung des Community Draft C5:2020 bei SaaS-Anbietern ohne eigene Infrastruktur im Vergleich zu C5:2016.* [Online]. Verfügbar unter: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Empfehlungen-nach-Angriffszielen/Cloud-Computing/Kriterienkatalog-C5/SaaS-Studie/Studie_node.html (Zugriff am: 27. Juli 2021).

[16] *OWASP Zed Attack Proxy (ZAP).* 2.10.0, 2021. [Online]. Verfügbar unter: https://www.zaproxy.org/

[17] *Security Code Scan - static code analyzer for .NET.* 5.1.0, 2021. [Online]. Verfügbar unter: https://security-code-scan.github.io/

[18] *Static Application Security Testing (SAST).* [Online]. Verfügbar unter: https://docs.gitlab.com/ee/user/application_security/sast/index.html

[19] Congress Government, *H.R.200 – 117th Congress (2021-2022): National Intersection and Interchange Safety Construction Program Act of 2021.* [Online]. Verfügbar unter: https://www.congress.gov/bill/115th-congress/house-bill/4943/text (Zugriff am: 10. Mai 2021).

[20] o.V., *Qualitätsmanagement STP Cloud.* [Online]. Verfügbar unter: https://www.stp-online.de/qualitaetsmanagement/ (Zugriff am: 27. Mai 2021).

[21] *Oracle VirtualBox.* 6.1. Virtualisierungssoftware. Oracle, 2019. [Online]. Verfügbar unter: https://www.virtualbox.org/

[22] *Kali Linux.* 2021.1. OffSec Services Limited, 2013. [Online]. Verfügbar unter: https://www.kali.org/

[23] *Low Orbit Ion Cannon (LOIC).* 2.0.0.4-1, 2009. [Online]. Verfügbar unter: https://github.com/NewEraCracker/LOIC

[24] *GoldenEye.* 1.0, 2012. [Online]. Verfügbar unter: https://github.com/jseidl/GoldenEye

[25] *slowloris.py - Simple slowloris in Python.* 0.2.3, 2015. [Online]. Verfügbar unter: https://github.com/gkbrk/slowloris

[26] I. Sharafaldin, A. Habibi Lashkari und A. A. Ghorbani, „Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization“ in *4th International Conference on Information Systems Security and Privacy*, Funchal, Madeira, Portugal, 1222018, S. 108–116, doi: 10.5220/0006639801080116.

[27] *NGINX*. 1.21.0. Reverse Proxy Webserver. F5 Inc., 2004. [Online]. Verfügbar unter: https://www.nginx.com/

[28] *.NET Core*. 5.0.7. Framework zur Erstellung von Applikationen mit C#. Microsoft, 2018. [Online]. Verfügbar unter: https://github.com/dotnet/core

[29] *ML.NET*. 1.5.5. Maschinelles Lern-Framework. Microsoft, 2019. [Online]. Verfügbar unter: https://github.com/dotnet/machinelearning

[30] *Yet Another Reverse Proxy*. 1.0.0-preview12. Reverse Proxy Bibliothek für .NET-Applikationen. Microsoft, 2020. [Online]. Verfügbar unter: https://github.com/microsoft/reverse-proxy

[31] *Angular*. 12.1.1. Web-Framework für die Erstellung von SPA. Google, 2016. [Online]. Verfügbar unter: https://github.com/angular/angular

[32] *Angular Lib for OpenID Connect & OAuth2*. 11.6.7, 2017. [Online]. Verfügbar unter: https://github.com/damienbod/angular-auth-oidc-client

[33] *NG Bootstrap: Angular powered Bootstrap widgets*. 9.1.0, 2017. [Online]. Verfügbar unter: https://ng-bootstrap.github.io/#/home

[34] *IdentityServer*. 5.2.1. OpenID Connect und OAuth2 Provider. Duende Software, 2020. [Online]. Verfügbar unter: https://github.com/DuendeSoftware/IdentityServer

[35] *Docker Compose*. 1.29.2. Tool für das Betreiben von Multi-Container Anwendungen mit Docker. Docker, 2013. [Online]. Verfügbar unter: https://github.com/docker/compose

[36] *Kubernetes*. 1.21.2. Container Orchestrierungssoftware. Google, 2014. [Online]. Verfügbar unter: https://github.com/kubernetes/kubernetes

[37] o.V., *Communications Security Establishment*. [Online]. Verfügbar unter: https://www.cse-cst.gc.ca/en (Zugriff am: 2. Juli 2021).

[38] Richard Lippmann, *1999 DARPA Intrusion Detection Evaluation*. [Online]. Verfügbar unter: https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset (Zugriff am: 4. Juli 2021).

[39] D. Dua und C. Graff, *UCI Machine Learning Repository*. Verfügbar unter: http://archive.ics.uci.edu/ml.

[40] The Shmoo Group, *DEFCON 8, 10 und 11*. [Online]. Verfügbar unter: http://cctf.shmoo.com (Zugriff am: 4. Juli 2021).

[41] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun und A. A. Ghorbani, „Characterization of Encrypted and VPN Traffic using Time-related Features“ in *2nd International Conference on Information Systems Security and Privacy*, Rome, Italy, 2192016, S. 407–414, doi: 10.5220/0005740704070414.

[42] A. Habibi Lashkari, G. Draper Gil, M. S. I. Mamun und A. A. Ghorbani, „Characterization of Tor Traffic using Time based Features“ in *3rd International Conference on Information Systems Security and Privacy*, Porto, Portugal, 2192017, S. 253–262, doi: 10.5220/0006105602530262.

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

[44] P. Płoński, *Random Forest Feature Importance Computed in 3 Ways with Python.* [Online]. Verfügbar unter: https://mljar.com/blog/feature-importance-in-random-forest/ (Zugriff am: 6. August 2021).

[45] I. Sharafaldin, A. Habibi Lashkari und A. A. Ghorbani, *CSE-CIC-IDS2018 on AWS*. [Online]. Verfügbar unter: https://www.unb.ca/cic/datasets/ids-2018.html (Zugriff am: 2. Juli 2021).

[46] *Python.* 3.9.5. Python Software Foundation. [Online]. Verfügbar unter: https://www.python.org/

[47] Charles R. Harris *et al.,* „Array programming with NumPy“, *Nature*, Jg. 585, Nr. 7825, S. 357–362, 2020, doi: 10.1038/s41586-020-2649-2.

[48] Jeff Reback *et al., pandas-dev/pandas: Pandas 1.3.0*. Zenodo.

[49] J. D. Hunter, „Matplotlib: A 2D graphics environment“, *Computing in Science & Engineering*, Jg. 9, Nr. 3, S. 90–95, 2007, doi: 10.1109/MCSE.2007.55.

[50] *TensorFlow Decision Forests.* 0.1.7. Maschinelle Lern-Bibliothek für das Random Forest Modell. TensorFlow, 2021. [Online]. Verfügbar unter: https://github.com/tensorflow/decision-forests

[51] F. Loizides *et al., Hg., Jupyter Notebooks - a publishing format for reproducible computational workflows*, 2016.

[52] F. Pedregosa *et al., Permutation feature importance.* [Online]. Verfügbar unter: https://scikit-learn.org/stable/modules/permutation_importance.html#permutation-importance (Zugriff am: 4. Juli 2021).

[53] N. M. Zivkovic, *Machine Learning with ML.NET: Random Forest.* [Online]. Verfügbar unter: https://rubikscode.net/2021/03/01/machine-learning-with-ml-net-random-forest/ (Zugriff am: 6. Juli 2021).

[54] *ONNX: Open standard for machine learning interoperability.* 1.9.0. Microsoft; Facebook, 2017. [Online]. Verfügbar unter: https://onnx.ai/

[55] *Netron.* 5.0.0, 2012. [Online]. Verfügbar unter: https://github.com/lutzroeder/netron

[56] *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, 2015. [Online]. Verfügbar unter: https://www.tensorflow.org/

[57] TensorFlow Team, *Build, train and evaluate models with TensorFlow Decision Forests.* [Online]. Verfügbar unter: https://www.tensorflow.org/decision_forests/tutorials/beginner_colab (Zugriff am: 6. August 2021).

[58] *Windows Subsystem for Linux (WSL).* 2. Microsoft, 2018. [Online]. Verfügbar unter: https://docs.microsoft.com/en-us/windows/wsl/

[59] *ThrottlR: A Throttling middleware for ASP.NET Core.* 0, 2020. [Online]. Verfügbar unter: https://github.com/Kahbazi/ThrottlR

[60] *sharppcap.* 6.0.0, 2008. [Online]. Verfügbar unter: https://github.com/chmorgan/sharppcap

[61] *packetnet.* 1.3.0, 2010. [Online]. Verfügbar unter: https://github.com/chmorgan/packetnet

[62] *NPcap.* 1.50. Nmap, 2013. [Online]. Verfügbar unter: https://github.com/nmap/npcap

[63] *libpcap-dev.* 1.10.1. Debian, 2009. [Online]. Verfügbar unter: http://www.tcpdump.org/index.html

[64] *Docker Desktop.* 3.5.1. Containerisierungssoftware. Docker Inc., 2013. [Online]. Verfügbar unter: https://github.com/docker/

[65] *Redis: Remote Dictionary Server.* 6.2.4. In-Memory Datenspeicher für das Caching von großen Datenmengen. Redis Labs, 2009. [Online]. Verfügbar unter: https://github.com/redis/redis

[66] *Microsoft.Extensions.Caching.StackExchangeRedis: Distributed cache implementation of Microsoft.Extensions.Caching.Distributed.IDistributedCache using Redis.* 5.0.1. Microsoft, 2018. [Online]. Verfügbar unter: https://www.nuget.org/packages/Microsoft.Extensions.Caching.StackExchangeRedis

[67] *Elasticsearch.* 7.13.4. Elastic, 2010. [Online]. Verfügbar unter: https://github.com/elastic/elasticsearch

[68] *Kibana.* 7.12.0. Elastic, 2019. [Online]. Verfügbar unter: https://github.com/elastic/kibana

[69] *Serilog.* 2.10.0. Community Project, 2016. [Online]. Verfügbar unter: https://github.com/serilog/serilog

5 Anhang

Im nachfolgenden Kapitel ist das Bewertungsdokument, das im Rahmen dieser Thesis aus den gegebenen Basiskriterien des C5 [14] konzipiert wurde und das Protokoll des Interviews mit dem Chefarchitekten der Abteilung PDE und dem DevOps Engineer mit Fragen, die im Kontext der Untersuchungen des Autors nicht selbstständig beantwortet werden konnten. Die beiden Formulare sind innerhalb des Dokumentes „Anhang.pdf“ im beigefügten ZIP-Archiv im Ordner „Dokumente“ konkludiert. Die Nummerierung der Seiten setzt sich aufgrund der besseren Übersichtlichkeit auf Basis des hier vorliegenden Dokumentes fort.

5.1 Verzeichnisstruktur des Archivs

Verzeichnis „Dokumente“

Verzeichnis „Implementierung“

Praktischer Teil und Quellcode der Anwendung