

Aufbau eines ML-basierten Intrusion Detection Systems (IDS)

Bachelorarbeit
OST - Ostschweizer Fachhochschule

Moritz Bättig & Christoph Landolt

Verfasser:	Moritz Bättig	moritz.baettig@ost.ch
	Christoph Landolt	christoph.landolt@ost.ch
Referent:	Prof. René Pawlitzek	rene.pawlitzek@ost.ch
Korreferent:	Prof. Dr. Klaus Frick	klaus.frick@ost.ch
Industriepartner:	Noser Engineering AG	
Institute:	INF Institut für Ingenieurinformatik ICE Institut für Computational Engineering	
Studiengang:	Systemtechnik mit Vertiefung Ingenieurinformatik	
Jahr:	2022	
Projekteingabe:	26.08.2022	

Zusammenfassung

Das Erkennen von Attacken auf HTTP-fähige Systeme stellt für IT-Sicherheitsverantwortliche eine erhebliche Schwierigkeit dar, da in der Praxis jeder Zugriff mit einer Liste von statischen Regeln abgeglichen werden muss, um einen Angriff zu erkennen. Damit diese regelbasierten Intrusion Detection Systeme zuverlässig funktionieren, muss die Liste mit statischen Regeln in immer kürzeren Zeitintervallen auf neue bekannte Angriffe und Angriffsmuster aktualisiert werden. Dies versetzt den IT-Sicherheitsverantwortlichen in eine reaktive Rolle und macht die zu überwachenden Systeme verwundbar auf neue Attacken, welche noch nicht in der Intrusion Detection Regelliste aufgeführt sind.

Bestehende Arbeiten beschäftigen sich überwiegend mit der Erkennung von Angriffen auf der Transportebene des TCP/IP-Stapels oder erfordern regelmässige Regelupdates mit den neusten Angriffen. Es gibt bereits auf dem Markt erhältliche Systeme, welche maschinelles Lernen zur Erkennung von Angriffen einsetzen. Diese Systeme benötigen aber eine grosse Menge an Trainingsdaten, sehr viel Rechenleistung und eine Anbindung an das Security Operations Center des Herstellers, um die eingegangenen Daten in Angriff und Nicht-Angriff einzuteilen.

In der vorliegenden Arbeit wird untersucht, wie HTTP-Zugriffe quantifiziert werden können, um mit begrenzter Rechenleistung ein Machine Learning Modell zu trainieren, mit welchem Angriffe über das HTTP-Protokoll effizient erkennen zu können. Dazu wurde eine Software entwickelt, mit welcher HTTP-Zugriffe aufgezeichnet, analysiert und an das Zielsystem weitergeleitet werden können. Um verschiedene Machine Learning Algorithmen zu testen, wurde die Pipeline modular mittels Plugin-Entwurfsmuster implementiert. Damit die Algorithmen neben akademischen Datensätzen ebenfalls mit realen Daten getestet werden können, wurde ein Honeypot, also ein Computersystem als Ziel für Cyberangriffe, aufgebaut, um möglichst viele Angriffe aufzuzeichnen. Die Zuverlässigkeit der Erkennung wird in verschiedenen öffentlichen Datensätzen und mit den realen Daten getestet.

Das in der Arbeit vorgestellte Modell konnte in den realen Daten eine Korrektklassifikationsrate von über 96.6% und in den öffentlichen Datensätzen, je nach Datensatz, über 99.7% erreichen. Im schlechten Datensatz wurde lediglich eine Korrektklassifikationsrate von über 85.9% erreicht. Die Arbeit zeigt somit, dass eine zuverlässige Intrusion Detection mittels maschinellen Lernens möglich ist, und bildet dank der modularen Softwarestruktur eine gute Grundlage für weitere Optimierungen und akademische Untersuchungen im Bereich der Merkmalsextraktion in HTTP-Anfragen und im Bereich des maschinellen Lernens zur Unterstützung der Intrusion Detection.

Abstract

Detecting attacks on HTTP-based systems poses a significant difficulty for IT security professionals because, in practice, each access must be checked against a list of static rules to detect an attack. To function reliably, the list of static rules of these rule-based intrusion detection systems must be updated at increasingly shorter intervals to detect recent published attacks and attack patterns. This puts the IT security professional in a reactive role and makes the monitored systems vulnerable to new and yet unincluded attacks in the intrusion detection rule list.

Existing papers are predominantly concerned with detecting attacks at the transport level of the TCP/IP stack or require periodic updates with the latest attacks. There are already systems available on the market that use machine learning to detect attacks. However, these systems require a large amount of training data, a lot of computing power and a connection to the vendor's security operations center to classify the incoming data into attack and non-attack.

In this thesis, it is investigated how HTTP requests can be quantified to train a machine learning model with limited computational power to efficiently detect attacks over the HTTP protocol. For this purpose, a data pipeline was developed that can be used to record HTTP requests, analyze them, and forward them to the target system. To test different machine learning algorithms, the pipeline was implemented with a modular architecture using the plugin design pattern. To ensure that the algorithms can also be tested with real data in addition to academic datasets, a honeypot, i.e. a computer system as a target for cyberattacks, was built to record as many attacks as possible. The reliability of the detection was tested in different public datasets and with real data.

The model presented in the thesis was able to achieve an accuracy of over 96.6% in the real data and over 99.7% in the public datasets, depending on the dataset. In the weakest dataset, only an accuracy of over 85.9% was achieved. The thesis therefore shows that reliable intrusion detection using machine learning is possible and thanks to the modular software structure, forms a reliable basis for further optimization and academic research in the area of feature extraction in HTTP requests and machine learning to support intrusion detection.

Danksagung

An dieser Stelle möchten wir unserem Referenten Prof. René Pawlitzek und unserem Korreferenten Prof. Dr. Klaus Frick für ihre fachliche Unterstützung zum Gelingen der vorliegenden Arbeit danken. Ohne ihre Bereitschaft, sich in das Thema Instrusion Detection mittels maschinellem Lernen einzuarbeiten und uns mit zahlreichen technischen Artikeln und Hinweisen zu unterstützen, wäre diese Arbeit nicht möglich gewesen.

Ein weiterer Dank gilt Dr. Simon Härdi, der uns von Seiten unseres Industriepartners, Noser Engineering AG, mit zahlreichen Hinweisen zu den aktuellen industriellen Herausforderungen im Bereich Cyber-Sicherheit und mit der Verbesserung der Softwarearchitektur zur Implementierung der Datenpipeline unterstützt hat.

Inhaltsverzeichnis

Abbildungsverzeichnis	VII
Tabellenverzeichnis	VIII
1 Einleitung	1
2 Datenakquirierung	2
2.1 Honeypot	2
2.1.1 Aufbau Honeypot	2
2.1.2 Angriffsagent	4
2.1.3 Labeling der Daten	4
2.1.4 Beurteilung der Daten	5
2.2 Öffentliche Datensätze	7
3 Data-Mining	8
3.1 Aufbau einer HTTP-Anfrage	8
3.2 Quantifizierung einer HTTP-Anfrage	9
3.2.1 Typisierung der Ressource	9
3.2.2 Metadaten	11
3.2.3 N-Gramme	14
3.2.4 Zeichenhäufigkeiten in einer Anfrage	17
3.2.5 Zeichenklassen	19
3.2.6 Erkennen von Hashwerten	21
3.2.7 Doppelt codierte Werte	22
3.3 Featurequalität	23
4 Implementierung	28
4.1 Grundlagen	28
4.2 Architektur	29
4.2.1 Anforderungen an die Architektur	29
4.2.2 Bausteinsicht	29
4.2.3 Laufzeitsicht	31
4.3 Pipeline: Inversion of Control	33
4.4 Data Transfer Objects	33
4.5 Alerting	34
4.6 Datenbank	35
4.7 Komponenten	36

4.7.1	Akquirierung	36
4.7.2	Filter	37
4.7.3	Typisierung	38
4.7.4	Merkmalsextraktion	41
4.7.5	Modell	42
4.8	Trainingshypothese	44
5	Experimente und Ergebnisse	45
5.1	Geschwindigkeit der Pipeline	45
5.2	Zuverlässigkeit der Intrusion Detection	46
5.2.1	Seq2Seq Datensatz	46
5.2.2	HTTP DATASET CSIC 2010	55
5.2.3	Honeypot Datensatz	61
5.2.4	Diskussion	68
6	Schlussfolgerungen und Ausblick	70
Literatur		76
Anhang		A-1
A	Programmcode	A-1
B	Klassendiagramm	A-2
C	Risikoplan	A-3
D	Work Breakdown Structure	A-4
E	Projektplan	A-5
F	Anforderungsliste	A-6

Abbildungsverzeichnis

1	Aufbau des Netzwerks mit Honeypot	3
2	Aufbau eines HTTP-Requests	8
3	Hierarchie einer HTTP-Anfrage	10
4	Länge der Anfragen im [5] Datensatz.	11
5	Summe der ASCII-Zeichen pro Anfrage im [5] Datensatz.	12
6	Scatter-Plot der Anzahl Features gegen die Anfragenlänge im [5] Datensatz.	12
7	Histogramm der Hexagramme der normalen Anfragen im [5] Datensatz. . .	14
8	Histogramm der Hexagramme bösartiger Anfragen im [5] Datensatz. . . .	15
9	Histogramme verschiedener N-Gramme im [5] Datensatz.	16
10	Korrelationsmatrix verschiedener N-Gramme im [5] Datensatz.	17
11	Mahalanobis-Distanz der einzelnen Zeichen normal gegen anomal im [5] Datensatz.	18
12	Mahalanobis-Distanz der einzelnen Zeichen normal gegen anomal im Honeypot Datensatz.	19
13	Klassen von Zeichen im Path-Query String im [5] Datensatz.	20
14	Klassen von Zeichen im Body des [5] Datensatzes.	21
15	Korrelationsmatrix, eingeschränkt auf eine spezifische Ressource ohne Path-Query im [5] Datensatz.	23
16	Korrelationsmatrix, eingeschränkt auf alle Ressourcen mit einer Path-Query im [5] Datensatz.	24
17	PCA mit zwei Komponenten, eingeschränkt auf eine Ressource ohne N-Gramme im [5] Datensatz.	25
18	PCA mit zwei Komponenten, eingeschränkt auf eine Ressource mit N-Grammen im [5] Datensatz.	25
19	PCA mit drei Komponenten, eingeschränkt auf eine Ressource mit N-Grammen im [5] Datensatz.	26
20	PCA mit drei Komponenten, Projektionen 1-3 im [5] Datensatz.	27
21	Komponentendiagramm des ML-IDS	30
22	Sequenzdiagramm des ML-IDS	32
23	Klassendiagramm der Inversion of Control Struktur	33
24	Klassendiagramm der Data Transfer Objects	34
25	Klassendiagramm des Alerting	35
26	Klassendiagramm des Datenbank Strategy-Patterns	36
27	Reverse Proxy	37
28	Klassendiagramm der Filterkomponente	38
29	Baumstruktur der Typisierungskomponente	39

30	Config-Datei zur Definierung vertrauenswürdiger Pfade	40
31	Klassendiagramm der Typisierungskomponente	41
32	Klassendiagramm der Merkmalsextraktion	42
33	Klassendiagramm der Modellkomponente	43
34	Visualisierung der Trainingshypothese	44
35	Pfad-Alerting der Typisierungskomponente im [5] Datensatz.	47
36	Visualisierung der Anzahl Alerts, geordnet nach generierender Stage im [5] Datensatz	47
37	Lernkurve für die URL /vulnbank/online/api.php im [5] Datensatz	48
38	Konfusionsmatrix im [5] Datensatz.	49
39	Optimale Anzahl Cluster im [5] Datensatz	51
40	Zuverlässigkeit des k-Means-Modells im [5] Datensatz	52
41	Veranschaulichung der Gütfunktion	53
42	Gütfunktion im [5] Datensatz.	54
43	Leverage Points im [5] Datensatz.	55
44	Pfad-Alerting der Typisierungskomponente im [8] Datensatz.	56
45	Visualisierung der Anzahl Alerts, geordnet nach generierender Komponente im [8] Datensatz	56
46	Lernkurve für die URL /tienda1/miembros/editar.jsp im [8] Datensatz	57
47	Konfusionsmatrix im [8] Datensatz.	58
48	Optimale Anzahl Cluster im [8] Datensatz	59
49	Zuverlässigkeit des k-Means-Modells im [8] Datensatz	59
50	Gütfunktion im [8] Datensatz.	60
51	High Leverage Points im [8] Datensatz.	61
52	Pfad-Alerting der Typisierungskomponente im Honeypot Datensatz.	62
53	Visualisierung der Anzahl Alerts, geordnet nach generierender Komponente im HoneyPot Datensatz	63
54	Lernkurve für die URL /web/cgi-bin/ptzctrl.cgi im Honeypot Datensatz	63
55	Konfusionsmatrix im Honeypot Datensatz	64
56	Optimale Anzahl Cluster im Honeypot Datensatz	65
57	Zuverlässigkeit des k-Means-Modells im Honeypot Datensatz	66
58	Gütfunktion im Honeypot Datensatz.	67
59	High Leverage Points im Honeypot Datensatz.	68

Tabellenverzeichnis

1	Honeypot-Systeme	3
2	Häufigste Scan-Zugriffe	6
3	Häufigste Brute-Force-Angriffe	6
4	Häufigste Exploits	7
5	Architekturanforderungen	29
6	Beschreibung der Softwarekomponenten	31
7	Interpretation der Konfusionsmatrix	49
8	Zusammenfassung wichtigster Kenngrössen	68

1 Einleitung

Durch die steigende Vernetzung in der Industrie und Diversität an eingesetzten Systemen steigt auch das Risiko durch Cyberangriffe. In der Schweiz werden beim nationalen Zentrum für Cybersicherheit mehrere Hackerangriffe pro Woche auf Unternehmen gemeldet. Damit ein Unternehmen langfristig erfolgreich am Markt bestehen kann, ist es wichtig, die Cyber-Risiken zu minimieren. Zu diesem Zweck werden neben herkömmlichen Sicherheitssystemen wie Firewalls und Antiviren-Programmen auch Intrusion Detection Systeme (IDS) eingesetzt, um mögliche Angriffe frühzeitig zu erkennen und diesen entgegenzuwirken. Aufgrund der Diversität und der stetigen Veränderung der modernen IT-Umgebungen ist es sehr schwer, mittels statischen Signaturen die IT-Umgebung und damit auch die Produktionsressourcen in der modernen Fabrik effektiv zu schützen. Diese Arbeit beschäftigt sich mit der Frage, wie Cyber-Angriffe auf HTTP-fähige IT-Systeme effektiv erkannt werden können und wie durch stetige Analyse der HTTP-Anfragen Machine Learning Modelle zur Angriffserkennung geschaffen werden können.

Zur Beantwortung der Frage wurde ein Honeypot aufgebaut, um reale Angriffe auf HTTP-Services aufzuzeichnen. Diese Angriffe wurden zusammen mit weiteren öffentlichen Datensätzen analysiert und quantifiziert. Zur Klassifizierung der HTTP-Anfragen in Angriff und Nicht-Angriff wurde eine Datenpipeline erstellt, die durch eine Plugin-Struktur dynamisch verschiedene Machine Learning Modelle laden und anwenden kann. Mithilfe dieser Pipeline wurden die Zuverlässigkeit der verschiedenen Modelle anhand der Honeypot Daten und der öffentlichen Datensätze analysiert.

Im Bericht wird als Einführung die Datenakquirierung der öffentlichen Daten und der Aufbau des Honeypots erläutert. Anschliessend wird erläutert, wie die akquirierten Daten mittels Data-Mining analysiert und wenn die extrahierten Merkmale aussagekräftig sind, quantifiziert wurden. Danach folgt die Implementierung der Datenpipeline und der Machine Learning Modelle. Als letzter Schritt wird die Performance und Zuverlässigkeit der Implementierung anhand der Daten aus der Datenakquirierung analysiert und diskutiert. Zum Schluss wurden die Schlussfolgerungen und der Ausblick auf weiterführende mögliche Forschungsbereiche aufgeführt.

Dieser Bericht wurde im Rahmen einer Bachelorarbeit am Institut für Ingenieurinformatik (INF) und Institut für Computational Engineering (ICE) in Zusammenarbeit mit der Noser Engineering AG an der Ostschweizer Fachhochschule OST durchgeführt und dient als Grundlage zur Implementierung von Machine Learning Algorithmen zur Analyse von HTTP-Anfragen in Intrusion Detection Systemen.

2 Datenakquirierung

Als kritischer Faktor für die Arbeit wurde bereits im Fachmodul [29] die Qualität der Daten identifiziert. Als Grundlage zur Analyse konnten die öffentlichen Datensätze [5] und [8] verwendet werden. Um jedoch ein möglichst realitätsnahes Szenario zu schaffen, wurde zusätzlich zu den beiden Datensätzen ein Honeypot erstellt, mit dem realer Internetverkehr und somit auch aktuelle Scan-Methoden und Angriffe aufgezeichnet wurden.

2.1 Honeypot

Nachfolgend wird der Aufbau und der Betrieb des Honeypots beschrieben.

2.1.1 Aufbau Honeypot

Der Honeypot besteht aus einer Philips Hue-Bridge, über die drei Philips-Lampen angesteuert werden können. Der Zugriff auf die Bridge erfolgt über REST. Um die Lampen beobachten zu können, wird zusätzlich die webfähige Kamera upCam Cyclone HD S+ installiert. Sowohl die Philips Hue-Bridge, als auch die Kamera sollen sowohl physisch als auch als Simulation bereitgestellt werden. Das Ziel ist es, über die physischen Ressourcen die Attraktivität des Honeypots zu erhöhen und so durch die grosse Anzahl an Zugriffen über die Dauer des Projektes Angriffe immer akkurate zu erkennen. Die simulierten Ressourcen werden auf einem Hypervisor bereitgestellt und sollen über die gesamte Dauer des Projektes angreifbar bleiben. Durch die Bereitstellung auf einem Hypervisor kann das System bei Bedarf mittels Snapshot zurückgesetzt werden.

Netzwerk

Der Honeypot wurde wie folgt aufgebaut:

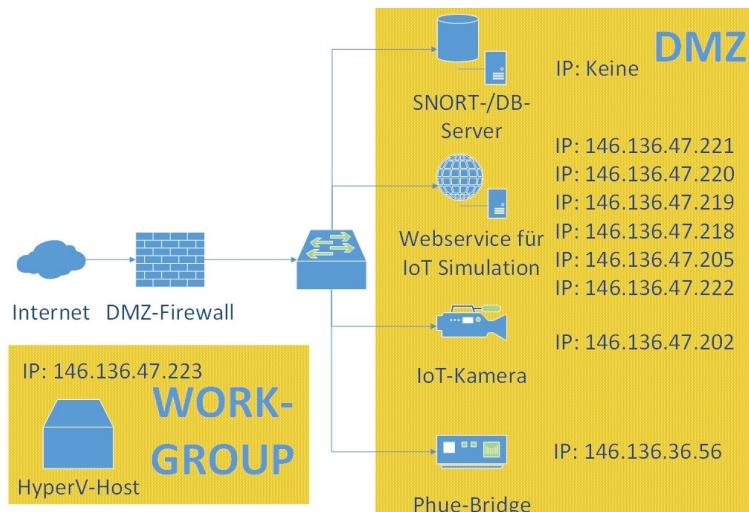


Abbildung 1: Aufbau des Netzwerks mit Honeypot

System	IP	Name	Beschreibung
Virtuelle Maschine	146.136.47.221	PhilipsHUE01Sim	Simulation der Philips HUE Bridge-Funktionalität
Virtuelle Maschine	146.136.47.220	UPCam01Sim	Simulation der upCam-Funktionalität
Virtuelle Maschine	146.136.47.219	Webservice01	Reserve
Virtuelle Maschine	146.136.47.218	Webservice02	Reserve
Virtuelle Maschine	146.136.47.205	Webservice03	Reserve
Philips HUE Bridge	146.136.36.56	PhilipsHUE01	Physische HUE Bridge
upCam Cyclone HD S+	146.136.47.202	UPCam01	Physische upCam
Virtuelle Maschine	146.136.47.222	IoTSecurityChecker01	Von diesem System gehen Zugriffe, Attacken und Scans aus.
HyperV-Host	146.136.47.223	hn700687	HyperV-Host zur Virtualisierung der Systeme.
Snort / DB-Server	keine	hn700686	Das System zeichnet alle Netzwerkpakete im Honeypot auf und klassifiziert diese mittels Snort in Angriffe und Nicht-Angriffe.

Tabelle 1: Honeypot-Systeme

Alle Systeme sind über einen Hub mit dem Internet verbunden. Da der Hub die eingehenden Netzwerk-Pakete immer an alle Ausgänge sendet, kann Snort als Netzwerk-Sniffer alle Zugriffe vom Internet auf die Zielsysteme aufzeichnen und analysieren.

2.1.2 Angriffsagent

Da zum Zeitpunkt des Aufbaus des Honeypots noch nicht klar war, ob für das Training eines Machine Learning Modells genügend qualitativ hochwertige Angriffe mit ausreichender Varianz aufgezeichnet werden können, wurden zusätzliche Methoden evaluiert, mit denen Security-Scans und Angriffe auf den Honeypot simuliert werden können. Zu diesem Zweck wurden zwei Software-Lösungen auf einer virtuellen Maschine bereitgestellt. Der Nessus Scanner ist ein weitverbreitetes Vulnerability Assessment Tool und erlaubt es, ein System auf bekannte Sicherheitslücken zu testen [36]. Mittels dieses Scanners kann der Honeypot auf Schwachstellen geprüft werden und ein typisches Scanszenario, welches einem Angriff vorausgeht, simuliert werden.

Als weiteres Tool wurde der IoT-SecurityChecker eingesetzt, welcher im Rahmen einer Masterarbeit entwickelt wurde und neben klassischen Scans auch Exploits auf Webcams ausführen kann [25].

Da die beiden Tools jedoch immer nach demselben Muster vorgehen und dadurch die aufgezeichneten Daten nur wenig Varianz aufweisen, ist es wichtig, dass das Machine Learning Modell nicht bloss aufgrund der selbst generierten Daten trainiert wird, sondern ausreichend reale Daten aufgezeichnet werden, welche das Verhalten eines realen Angreifers besser widerspiegeln. Ansonsten besteht die Gefahr, dass sich das Machine Learning Modell lediglich auf die Trainingsdaten einlernt und keine realen Angriffe erkennt.

2.1.3 Labeling der Daten

Um Angriffe im aufgezeichneten Internetverkehr zu erkennen, wurde das Intrusion Detection System Snort 3.0 eingesetzt [10]. Snort verfügt über eine grosse Community, welche bei neuen Angriffen die neu erstellten Signaturen zeitnah veröffentlicht. Dadurch ist es möglich, mit Snort auch aktuelle Angriffe effektiv und zuverlässig zu detektieren. Damit Snort über die aktuellsten Rules verfügt, können diese automatisch mittels der Software PulledPork aktualisiert werden [31]. Um die Trainingsdaten aus den aufgezeichneten Netzwerkpaketen zu extrahieren und mittels der aufgezeichneten Snort-Alerts zu klassifizieren, wurden innerhalb der aufgezeichneten Netzwerkdaten die Pakete anhand der Zielports 80 und 8080 gefiltert. Da im Honeypot alle HTTP-Services auf diese Ports hören und kein HTTPS aktiviert ist, kann so der gesamte HTTP-Internetverkehr extrahiert werden. Danach werden die verbleibenden Pakete mittels der Zeitstempel mit den Snort-Alerts abgeglichen und dadurch in die Klassen „Angriff“ oder „Kein Angriff“ eingeteilt.

2.1.4 Beurteilung der Daten

Mit dem Honeypot konnte eine Vielzahl an Zugriffen und Angriffen auf die bereitgestellten Systeme aufgezeichnet werden. Der grösste Teil der Zugriffe stammt jedoch von Bots, die das System scannen, Brute-Force-Angriffe ausführen oder bekannte Exploits durchprobieren. In der gesamten Betriebsphase des Honeypots konnten nur vereinzelte Zugriffe aufgezeichnet werden, welche realen Zugriffen auf die Systeme von ausserhalb des OST-Netzwerks zugeordnet werden können. Anbei wurden die häufigsten Zugriffe nach Klasse aufgelistet:

Scan-Zugriffe

Methode	Pfad	Beschreibung
GET	/ .env	Viele Applikationen speichern diverse Informationen wie beispielsweise die Anmeldeinformationen in einer .env-Datei. Dieser Scan prüft, ob diese Datei existiert und zugreifbar ist [11].
OPTIONS	/RTSP	Dieser Zugriff prüft, ob der Service das Real Time Streaming Protocol unterstützt und ob das System über dieses Protokoll weiter auf Schwachstellen gescannt werden soll [7].
GET	/sitemap.xml	In der Datei sitemap.xml stellt ein Betreiber einer Webseite Informationen über die Seite für Suchmaschinen bereit [16].
GET	/robots.txt	Die Datei robots.txt regelt, ob und wie eine Webseite von Webcrawlern besucht werden kann [26].
GET	/ .well-known/security.txt	Die Datei security.txt gibt die Kontaktangaben des Systemadministrators an, welcher von gutartigen Hackern bei aufgedeckten Sicherheitslücken kontaktiert werden kann [12].

Methode	Pfad	Beschreibung
GET	/	Viele Bots greifen initial auf das Wurzelverzeichnis zu, um Informationen über den Service zu erhalten. Häufig werden diese Anfragen von einem User-Agent „python-requests“ ausgeführt.
POST	/onvif/devices	Dieser Zugriff prüft mit der Aktion action=„http://www.onvif.org/ver10/device/wsdl/GetSystemDateAndTime“ ob das Device das ONVIF Network Interface implementiert, welches von vielen Webcams verwendet wird [32].
GET	/sql/sqladmin/index.php	Es konnten sehr viele Scans auf der Suche nach bekannten Software-Produkten aufgezeichnet werden.

Tabelle 2: Häufigste Scan-Zugriffe

Zusätzlich zu den aufgeführten Scans konnten alle im Artikel [43] dokumentierten Scans ebenfalls aufgezeichnet werden. Neben den in der Tabelle 2 aufgeführten Scans, konnten viele weitere Angriffe aufgezeichnet werden.

Brute-Force-Angriffe

Methode	Pfad	Beschreibung
GET	/tempfs/auto.jpg?usr=user&pwd=user123	Über diesen Brute-Force-Angriff wird versucht, sich bei Camertin IP Cameras anzumelden [21].
POST	/goform/webLogin	Der Post versucht sich mittels „User=admin&Passwd=admin“ auf einem Linksys Router anzumelden [41]. Bei den Zugriffen wird mittels Wortlisten der Benutzername und das Passwort variiert.
POST	/boaform/admin/formLogin	Dieser Bot versucht sich mit unterschiedlichen Anmeldeinformationen wie username=admin&psd=Feeffif anzumelden [27].

Tabelle 3: Häufigste Brute-Force-Angriffe

Exploits

Methode	Pfad	Beschreibung
GET	/	Dieser Zugriff versucht als User-Agent eine Log4j-Schwachstelle [17] auszunutzen. Dazu wird folgende URI als User-Agent angegeben: \${jndi:ldap://168.61.94.169:443/TomcatBypass/Command/Base64/Base64EncodedString}. Der mit Base64 kodierte String enthält folgenden Aufruf: wget http://212.192.241.35/lshboot && chmod 777 lshboot && ./lshboot lshboot && rm lshboot

Tabelle 4: Häufigste Exploits

Die Aufzählung der Zugriffe ist nicht abschliessend. Es wurden lediglich die häufigsten Zugriffe dokumentiert. Bei den meisten im Honeypot aufgezeichneten Daten handelt es sich um Scans. Trotzdem eignen sich die Daten sehr gut, die realen Attacken und Zugriffe bei mit dem Internet verbundenen Systemen aufzuzeigen.

2.2 Öffentliche Datensätze

Für das Projekt wurden zusätzlich zu den aufgezeichneten und den mit dem Angriffsagenten selbst generierten Datensätzen noch zwei öffentlich zugängliche Datensätze verwendet. Der Seq2Seq-Datensatz beinhaltet 21'991 gutartige und 1'097 bösartige HTTP-Zugriffe auf eine E-Banking-Applikation [5]. Die Daten sind bereits klassifiziert und beinhalten nur eine geringe Anzahl an unterschiedlichen Ressourcen. Daher wurde dieser Datensatz auch massgeblich für das Data-Mining im Kapitel 3 verwendet. Der HTTP DATASET CSIC 2010 beinhaltet Anfragen auf eine E-Commerce-Plattform und beinhaltet 36'000 normale und 25'000 bösartige HTTP-Zugriffe [8]. Beide öffentlichen Datensätze wurden zum Data-Mining und zur Überprüfung des Machine Learning Modells verwendet. Dadurch, dass neben den vom Angriffsagenten generierten und den Honeypot-Daten auch noch diese beiden öffentlichen Datensätze verwendet wurden, lässt sich die Genauigkeit des Modells sehr gut bewerten. Zusätzlich wurde der HTTP DATASET CSIC 2010 wie bereits im Fachmodul beschrieben [29] in vielen wissenschaftlichen Arbeiten und Artikeln zum Thema Intrusion Detection zitiert. Mittels dieses Datensatzes kann überprüft werden, wie zuverlässig das eigene Modell im Vergleich zu den bereits existierenden Arbeiten funktioniert.

3 Data-Mining

Das HTTP-Protokoll ist ein generisches, zustandsloses Applikationsebene-Protokoll, welches die Kommunikation zwischen verteilten, kollaborativen oder Hypermedia-Informationsystemen ermöglicht [40]. Das Protokoll kann durch Adaptieren der Anforderungsmethoden und des Headers in einer Vielzahl von Kommunikationssystemen eingesetzt werden [40]. Dieser einfach adaptierbare Aufbau hat massgeblich zur Verbreitung des HTTP-Protokolls beigetragen. Jedoch stellt diese Vielfalt beim Aufbau einer HTTP-Anfrage beim Zugriff auf verschiedene Endpunkte eine erhebliche Schwierigkeit bei der Quantifizierung der Anfrage für ein Machine Learning Modell dar. In diesem Kapitel wird beschrieben, wie aus den gegebenen HTTP-Anfragen die benötigten Merkmale gewonnen werden.

3.1 Aufbau einer HTTP-Anfrage

Eine HTTP-Anfrage ist wie folgt aufgebaut:

Method	Request-URI?Query-String	HTTP-Version
	HTTP-Header	
	HTTP-Body	

Abbildung 2: Aufbau einer HTTP-Anfrage [40]

Method

Ursprünglich war das HTTP-Protokoll als Schnittstelle zu verteilten Objektsystemen gedacht und erlaubte eine Vielzahl an Method-Tokens [40]. Durch die Einführung von REST-Systemen konnten die Methoden im RFC7231 [39] standardisiert werden. Lediglich die Methoden GET und HEAD müssen von jedem System unterstützt werden. Alle anderen Methoden sind optional. Die standardisierten HTTP-Methoden müssen bei IANA registriert werden [20]. Da das ML-IDS-System auf RESTful Services ausgelegt ist, werden in dieser Arbeit lediglich die Methoden POST, GET, PUT, PATCH, DELETE und HEAD berücksichtigt.

Request-URI und Query-String

Die Request-URI (Uniform Resource Identifier) identifiziert die Ressource, auf welche die Anfrage zugreifen soll [40]. Der Query-String ist optional und im RFC3986 standardisiert [47]. Über den Query-String können Schlüssel-Werte-Paare an den Webserver übermittelt werden. Der Query String ist mittels eines ? von der URI abgegrenzt. Die Schlüssel-Werte-Paare können mittels & aneinander gereiht werden.

Beispiel: test.php?key1=value1&key2=value2

HTTP-Header

Im Anfragenkopf kann ein Client dem Server Metadaten über sich selbst oder über die Anfrage mitteilen. Die Standardfelder sind im RFC2616 [40] beschrieben, können aber beliebig erweitert werden. Wird ein Eintrag im Header vom Server nicht unterstützt, wird dieser ignoriert. Ein Feld im Anfragenkopf hat immer den Aufbau: Key: SP Value CRLF.

HTTP-Body

Der Request-Body ist mittels CRLF vom Header getrennt. Im Body werden die Daten des Requests übermittelt. Das Format der Daten (String, XML, JSON, ...) hängt dabei vom jeweiligen Endpunkt ab und wird im Header-Feld Content-Type festgelegt [40].

3.2 Quantifizierung einer HTTP-Anfrage

Das HTTP-Protokoll kann als Datenübertragungsprotokoll für verschiedene Anwendungen eingesetzt werden. Deswegen ist die Reduktion von Eigenschaften einer HTTP-Anfrage auf eine definierte Anzahl von Merkmalen sehr anspruchsvoll.

In diesem Abschnitt wird beschrieben, welche Kennzahlen zur Quantifizierung im ML-IDS betrachtet wurden.

3.2.1 Typisierung der Ressource

Aufgrund der im Abschnitt 3.1 beschriebenen Informationen wurde ein Modell entwickelt, welches eine hierarchische Typisierung einer HTTP-Anfrage zulässt. Ziel der Typisierung ist es, unterschiedliche Anfragen an einen Webservice effektiv zu trennen, um einen bestmöglichen Schutz der jeweiligen Ressource zu ermöglichen. Zu diesem Zweck wurde folgende Baumstruktur entwickelt:

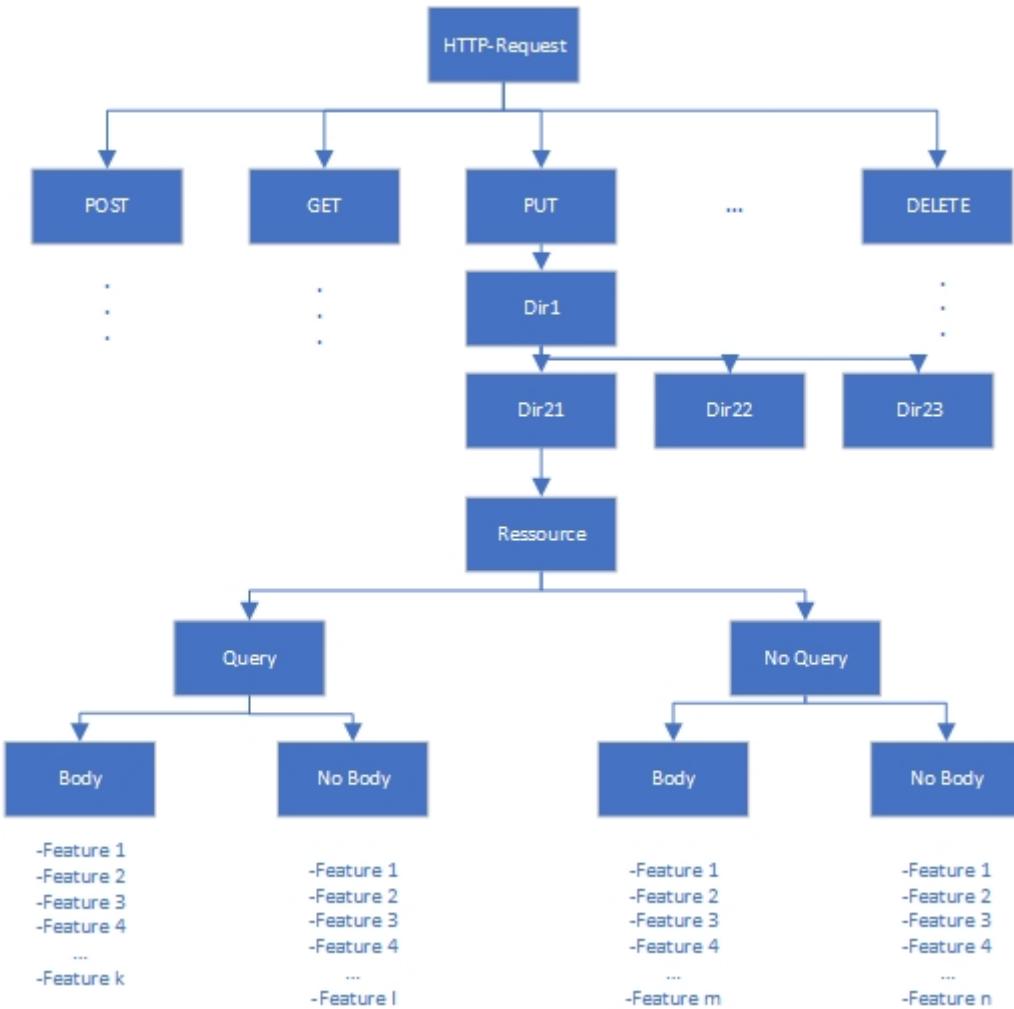


Abbildung 3: Hierarchie einer HTTP-Anfrage

Dabei wird eine HTTP-Anfrage aufgrund der ersten Zeile (z.B. „GET /Dir1/Dir21 /Ressource.php?key1=value1“) in der Baumstruktur angeordnet. Als Erstes wird die Anfragenmethode betrachtet. Danach wird die URI betrachtet. Alle Werte, welche in der Form Dir/ dargestellt werden können, werden als Verzeichnis betrachtet. Folgt kein weiterer Wert in der Form Dir/, wird dies als Ressource interpretiert. Folgt auf die Ressource ein Query-String wird dies als Ressource mit Query betrachtet. Zum Schluss wird noch geprüft, ob die Anfrage über einen Body verfügt oder nicht.

Aufgrund dieser Struktur lassen sich Anfragen effizient typisieren und entsprechend aussagekräftige Merkmale für die jeweilige Anfrage extrahieren. Wird beispielsweise eine Anfrage ohne Query an eine Ressource gesendet, müssen keine Merkmale aus dem Query-String erstellt werden.

3.2.2 Metadaten

Die Quantifizierung von Metadaten kann ohne grossen Rechenaufwand durchgeführt werden und eignet sich um DOS oder Overflow-Attacken zu erkennen. Die Metadaten können auch ohne Deep-Package Inspection [29] quantifiziert werden. Im vorliegenden Projekt wurden folgende Metadaten analysiert:

Anfragenlänge

In der Arbeit [44] wird die Länge einer Anfrage als signifikanter Faktor zur Erkennung von Buffer-Overflow-Attacken beschrieben. Zusätzlich konnte in [24] festgestellt werden, dass auch Cross-Site-Scripting (XSS) Attacken oftmals länger als die normalen Anfragen sind. Um die Qualität der Länge als Feature zu analysieren, wurde der [5] Datensatz betrachtet:

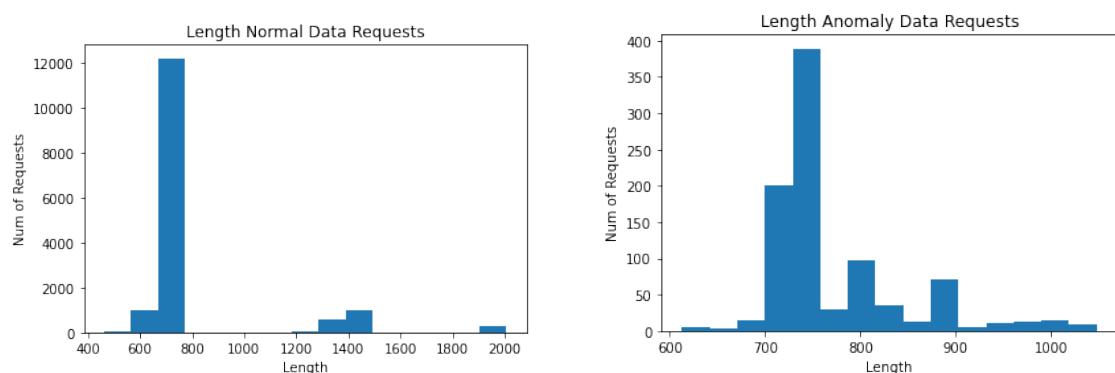


Abbildung 4: Länge der Anfragen im [5] Datensatz.

Es konnte festgestellt werden, dass die normalen Anfragen je nach Typ in abgrenzbaren Gruppen auftreten, während ein Grossteil der Angriffe auch in diesen Klassen auftreten, aber einige Anfragen stärker streuen. Daher kann vermutet werden, dass es sich bei der Länge um ein aussagekräftiges Merkmal zur Charakterisierung der HTTP-Anfrage handelt.

Summe der erweiterten ASCII-Codes

Bei dieser Analyse wurden alle Zeichen in der Anfrage in den zugehörigen ASCII-Code umgewandelt und daraus die Summe berechnet:

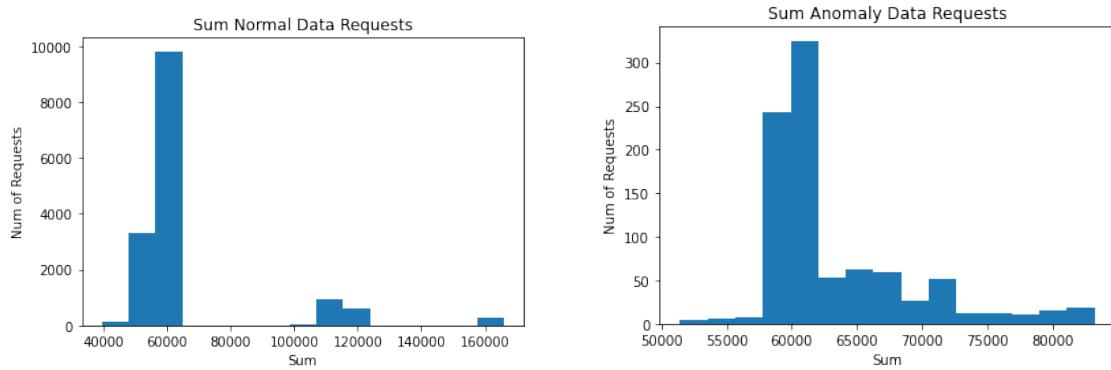


Abbildung 5: Summe der ASCII-Zeichen pro Anfrage im [5] Datensatz.

Die Verteilung der Summe der ASCII-Zeichen korreliert stark mit der Länge und liefert somit keine weiteren Erkenntnisse. Da die Summierung der ASCII-Zeichen rechenintensiv ist, wird auf dieses Feature verzichtet und der Ansatz wurde nicht weiterverfolgt.

Anzahl der Merkmale

Bei diesem Feature wurden alle Kandidaten für die Erstellung eines Merkmals in der Anfrage gezählt. Konkret wurden folgende Daten gezählt:

- Länge der Anfrage
- Anzahl Schlüssel-Werte-Paare im Query-String
- Anzahl Felder im Header
- Anzahl Schlüssel-Werte-Paare im Body, wenn sich diese aufgrund der Struktur extrahieren lassen

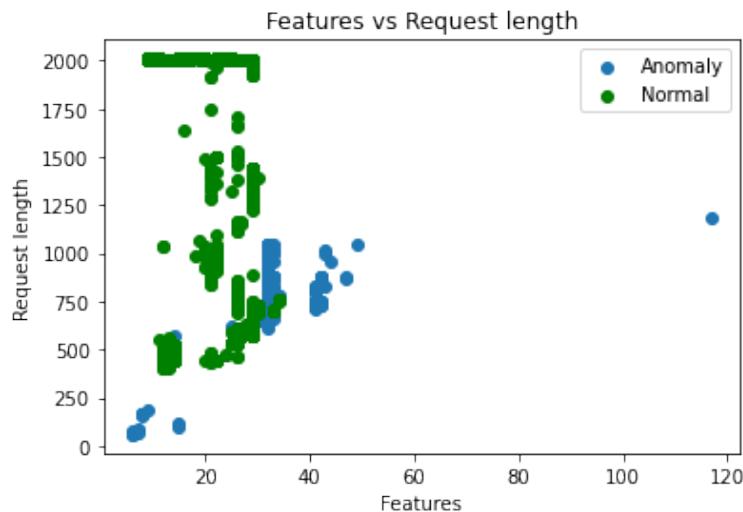


Abbildung 6: Scatter-Plot der Anzahl Features gegen die Anfragenlänge im [5] Datensatz.

Die Analyse veranschaulicht, dass bei den normalen Anfragen die Anzahl Features in einem bestimmten Bereich liegen und bei den Anomalien die Anzahl der Merkmale stärker streuen können. Dadurch wurde dieser Ansatz weiterverfolgt.

Ländercode des Anfragenursprungs

Über die IP-Adresse des Senders der HTTP-Anfrage kann mittels des Dienstes [15] die Geo-Location des Senders ermittelt werden. Der Dienst liefert zur überprüften IP-Adresse unter anderem den Ländercode. Da es sich beim Ländercode (z.B. CH für Schweiz) um nominale Daten handelt, müsste für jeden existierenden Wert eine Dummy-Variable erstellt werden, was sehr hochdimensionale Daten liefert. Daher wurde der Ländercode des Anfragenursprungs als eigenes Merkmal nicht weiter berücksichtigt.

Zeitpunkt der Anfrage

Aufgrund menschlicher Gewohnheiten lässt sich vermuten, dass die Interaktion eines Benutzers mit einem Webservice einem zeitlichen Muster folgt. Da es sich bei der Uhrzeit des Zugriffs um ordinale Daten handelt, könnte der Machine Learning Algorithmus 23:59 Uhr höher gewichtet als 00:01 Uhr. Für die Codierung solcher zyklischer Features wird in [37] folgendes Vorgehen empfohlen:

Die Uhrzeit wird in Sekunden umgerechnet und der Winkel der Sekunden innerhalb eines 24h-Kreises wird gebildet [19]. Als Merkmale werden vom dadurch entstandenen Winkel der Sinus- und Cosinus-Wert berechnet und verwendet [19]. Es wird sowohl der Sinus- als auch der Cosinus-Wert benötigt, um die Uhrzeit eindeutig zu identifizieren und um eine höhere Gewichtung eines grösseren trigonometrischen Wertes durch das Machine Learning Modell zu verhindern.

Diese Methode hat gemäss [37] folgende Nachteile:

- Es werden von der Uhrzeit zwei Merkmale erstellt. Dadurch kann die Uhrzeit im Modell übergewichtet werden.
- Alle Algorithmen, die auf dem Decision-Tree basieren, können immer nur ein Merkmal nach dem anderen betrachten. Dadurch kann der Sinus- und Cosinus-Wert nicht zur selben Zeit betrachtet werden und die Uhrzeit kann nicht mehr eindeutig identifiziert werden. Eine höhere Gewichtung eines grösseren trigonometrischen Wertes ist möglich.

Aufgrund der aufgeführten Nachteile wird der Zeitpunkt der Anfrage als Merkmal nicht weiter berücksichtigt.

Zeitpunkt der Anfrage aus Sicht des Senders

Obwohl sich die Menschen oft an feste Gewohnheiten halten, können sich diese je nach Zeitzone doch sehr stark unterscheiden. Daher wurden die Merkmale „Ländercode des Anfragenursprungs“ und „Zeitpunkt der Anfrage“ kombiniert indem mithilfe der Zeitzone des Senders der Zeitpunkt aus Sicht des Senders berechnet wurde. Die dadurch erhaltene Uhrzeit lässt sich wie im Abschnitt 3.2.2 Zeitpunkt der Anfrage beschrieben quantifizieren.

3.2.3 N-Gramme

Bei dieser Analyse wurden wie in [24] beschrieben, die Zeichen der HTTP-Anfrage in Hexagramme eingeteilt und die Häufigkeit bestimmter Zeichenfolgen analysiert. Das Ziel ist es, für einen Webservice, ähnlich wie bei einer natürlichen Sprache, bestimmte charakteristische Zeichenfolgen zu identifizieren und über diese Verteilung „Fremdsprachen“ also Angriffe zu identifizieren.

Die normalen Anfragen weisen eine charakteristische Verteilung auf:

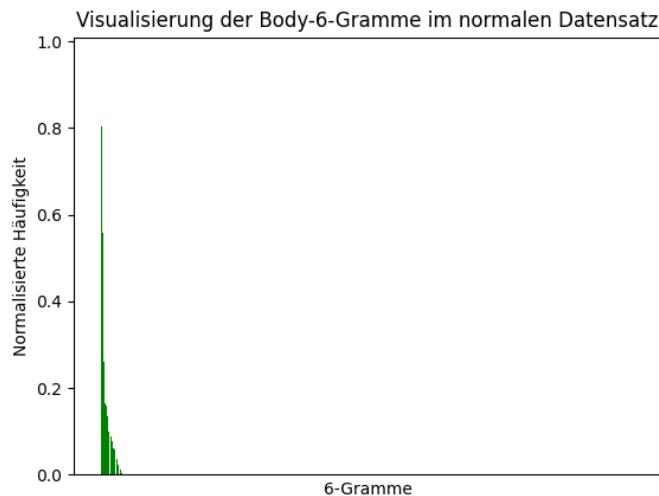


Abbildung 7: Histogramm der Hexagramme der normalen Anfragen im [5] Datensatz.

Bei den bösartigen Anfragen ist ein grosser Teil der Zeichenfolgen identisch, da nur minimale Änderungen für einen Angriff benötigt werden. Es zeigt sich jedoch, dass es bei den Anomalien bestimmte Häufigkeiten gibt, die bei den normalen Anfragen fehlen. Diese sind jedoch schwer als Angriffe zu charakterisieren, da sich das ML-Modell nicht sicher sein kann, ob es ein Angriff oder lediglich eine zufällige Häufung ist.

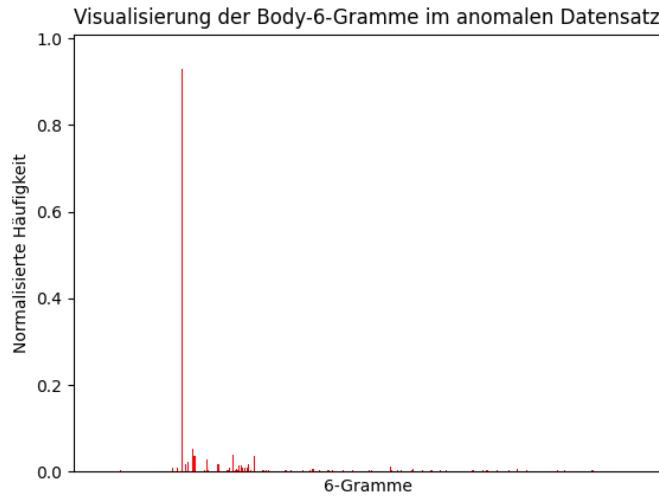


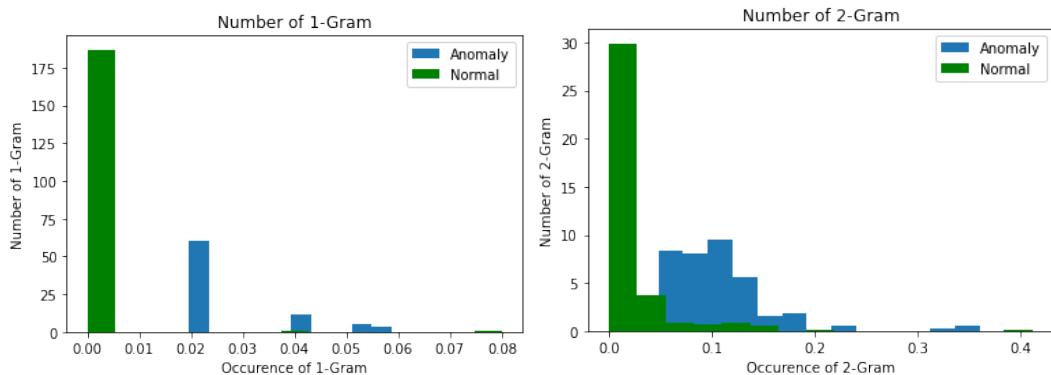
Abbildung 8: Histogramm der Hexagramme bösartiger Anfragen im [5] Datensatz.

Da die Anzahl an N-Grammen sehr stark variieren kann und eine Vielzahl von kombinatorischen Möglichkeiten besteht, entsteht durch die N-Gramm-Analyse ein sehr hochdimensionaler Raum, welcher viel Rechenleistung für die Analyse benötigt. Des Weiteren kommt hinzu, dass die Anzahl an Merkmalen in diesem Raum variiieren kann. Daher wird in [24] folgender Ansatz vorgeschlagen:

$$s = \frac{\#(N - \text{Gramme}) \text{ der Anfrage, welche auch in den Trainingsdaten vorkommen}}{\#(N - \text{Gramme}) \text{ in der HTTP - Anfrage}} \in [0, 1] \quad (1)$$

Die Schwierigkeit dieser Methode ist es, die Trainingsdaten sinnvoll zu wählen, da im betrachteten Szenario keine klassifizierten Daten zur Verfügung stehen.

Im Folgenden wurde die Qualität der N-Gramm-Merkmale für verschiedene Längen im [5] Datensatz betrachtet:



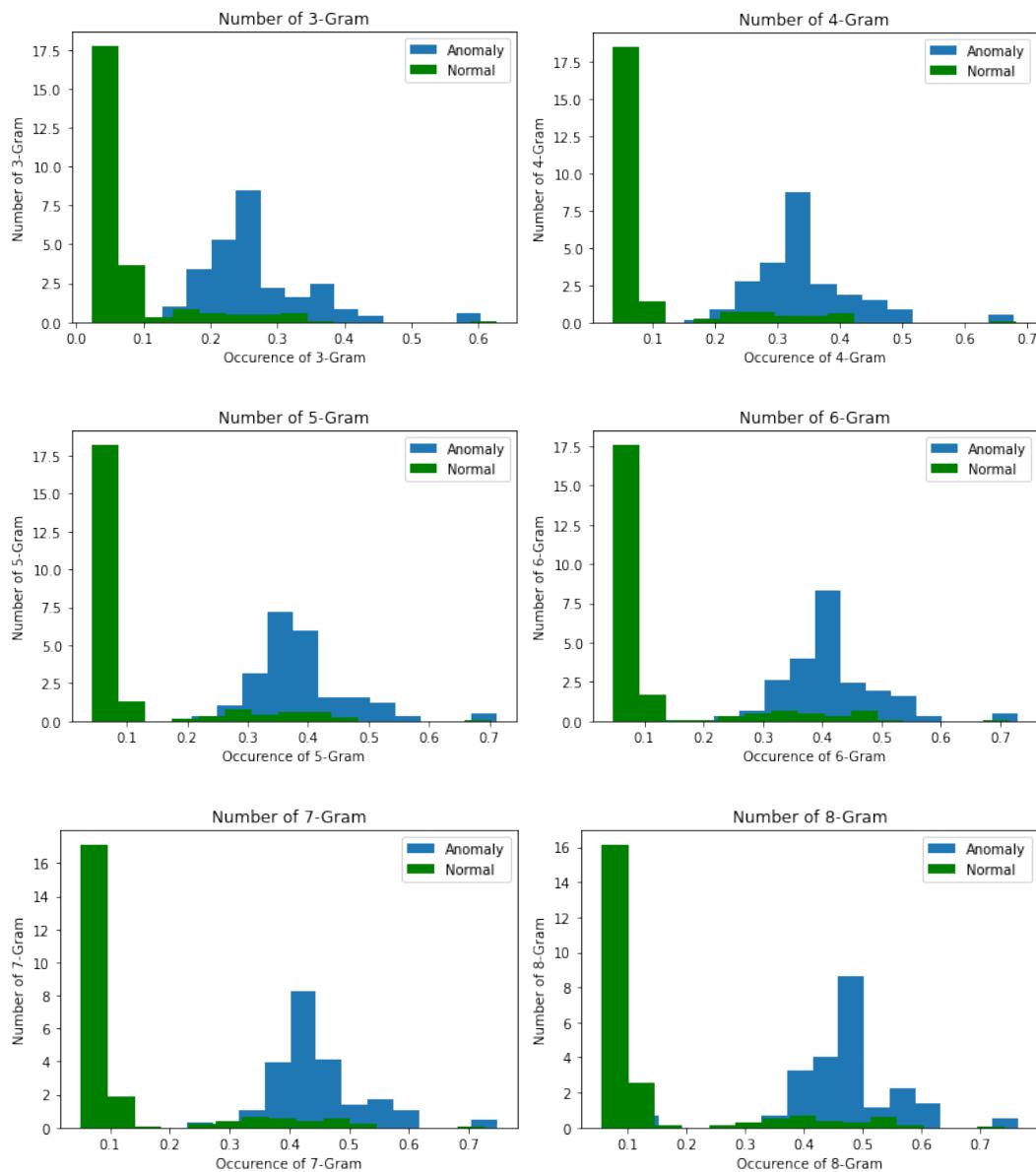


Abbildung 9: Histogramme verschiedener N-Gramme im [5] Datensatz.

Die verschiedenen Visualisierungen zeigen deutlich, dass die N-Gramm-Analyse eine Trennung der normalen und anomalen Anfragen erlaubt.

Zur Bewertung der Merkmalsqualität wurde eine Korrelationsanalyse durchgeführt:

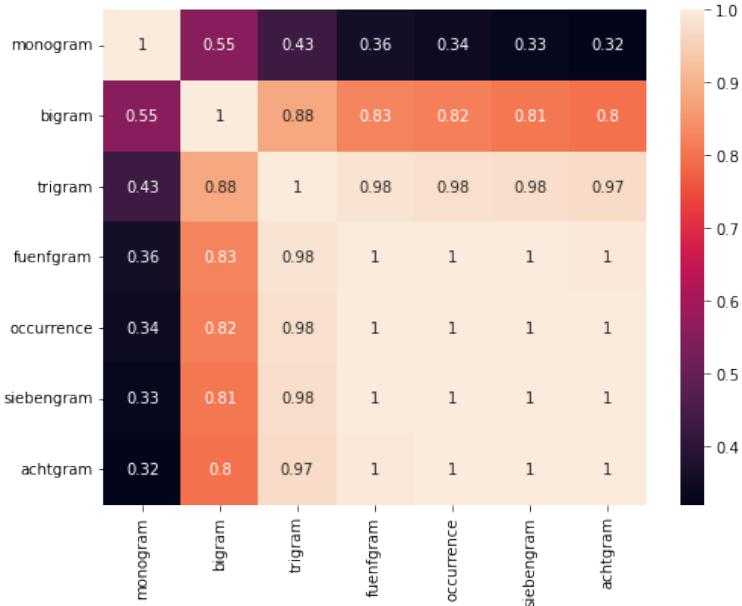


Abbildung 10: Korrelationsmatrix verschiedener N-Gramme im [5] Datensatz.

Die Korrelationsmatrix zeigt, dass für alle Buchstabengruppen, welche grösser als zwei Buchstaben sind, die Korrelation nahezu bei eins liegt. Daher wurden als Merkmale folgende N-Gramme gewählt:

- 1-Gramm (Monogramm)
- 2-Gramm (Bigramm)
- 6-Gramm (Hexagramm)

3.2.4 Zeichenhäufigkeiten in einer Anfrage

In der Abbildung 9 ist ersichtlich, dass die Monogramme sich sehr gut zur Charakterisierung einer HTTP-Anfrage eignen könnten. Daher wurde weiter untersucht, ob mittels der Auftretenswahrscheinlichkeit gewisser Zeichen die Daten besser charakterisiert werden können. Zu diesem Zweck wurde die Auftretenshäufigkeit der 256 Zeichen der erweiterten ASCII-Tabelle in der HTTP-Anfrage gezählt. Danach wurde wie in [34] vorgeschlagen, die Mahalanobis-Distanz d zwischen den 256 Merkmalen wie folgt berechnet:

$$d(x, \bar{y}) = \sqrt{(x - \bar{y})^T C^{-1} (x - \bar{y})} \quad (2)$$

x bezeichnet den aktuellen Feature-Vektor, \bar{y} ist der gemittelte Eigenschaftsvektor und C^{-1} ist die Kovarianzmatrix. In der Praxis zeigte sich, dass die Berechnung der Mahalanobis-Distanz sehr ressourcenintensiv ist und in hochdimensionalen Räumen viel Rechenzeit benötigt.

Bei einer Analyse mit den Trainingsdaten aus [5] konnten keine Anomalien bei der Verteilung der einzelnen Zeichen festgestellt werden:

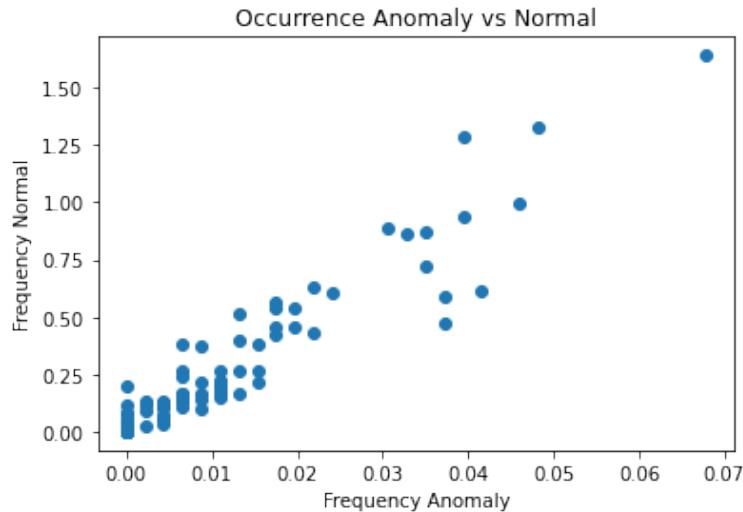


Abbildung 11: Mahalanobis-Distanz der einzelnen Zeichen normal gegen anomal im [5] Datensatz.

Um die Qualität der Mahalanobis-Distanz als Merkmal besser beurteilen zu können, wurde die Analyse nochmals mit den Honeypot-Daten durchgeführt:

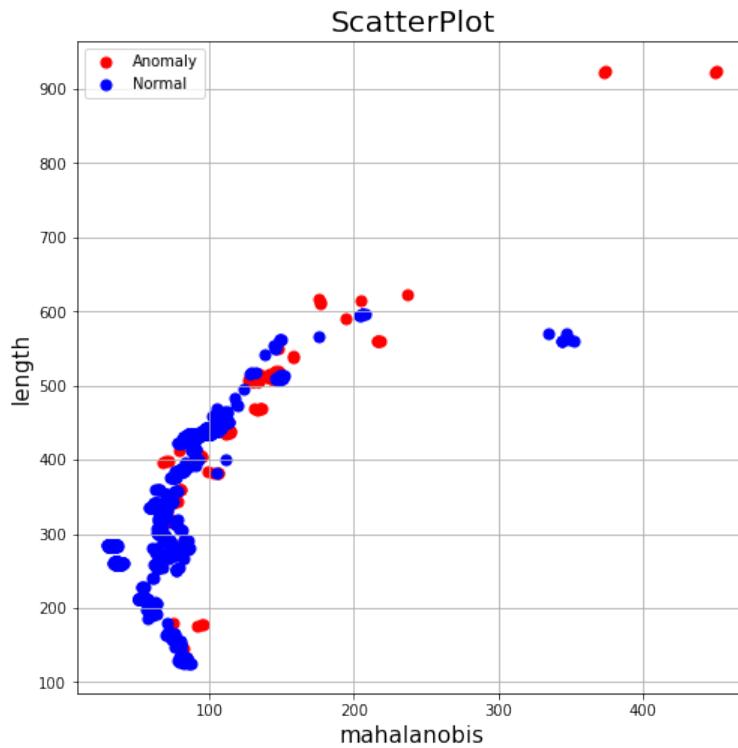


Abbildung 12: Mahalanobis-Distanz der einzelnen Zeichen normal gegen anomal im Honeypot Datensatz.

Da weder mit dem [5] Datensatz noch mit den Honeypot Daten die Mahalanobis-Distanz als qualitativ hochwertiges Merkmal nachgewiesen werden konnte, wurde auf dieses Feature bei der weiteren Analyse verzichtet.

3.2.5 Zeichenklassen

In [3] wird das Vorgehen vorgestellt, in einem Stream Buffer von fester Länge die jeweiligen Zeichen in Klassen einzuteilen. Dazu wurden in dieser Analyse folgende Klassen verwendet:

- Zahlen
- Grossbuchstaben
- Kleinbuchstaben
- Leerzeichen
- Sonderzeichen

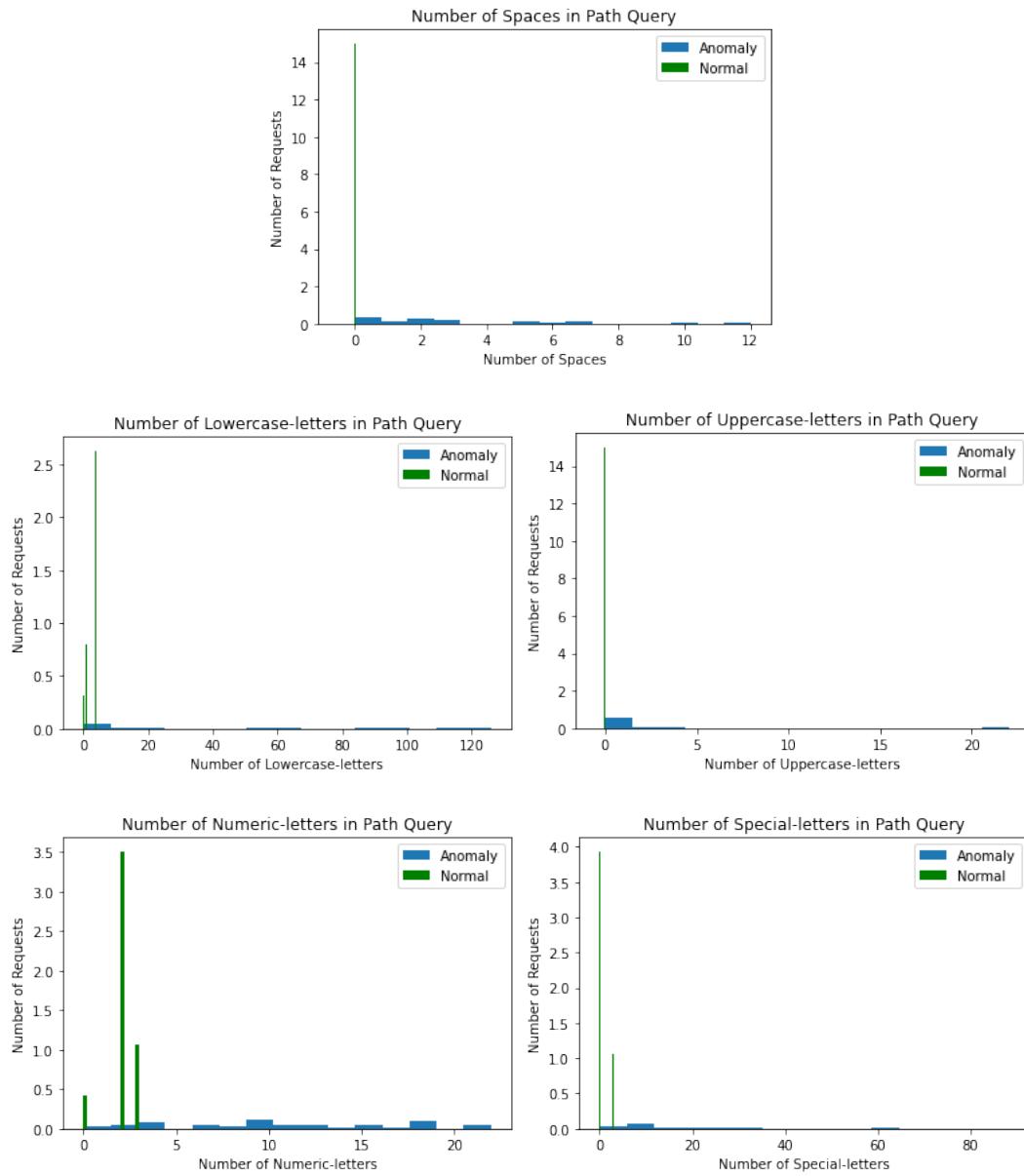


Abbildung 13: Klassen von Zeichen im Path-Query String im [5] Datensatz.

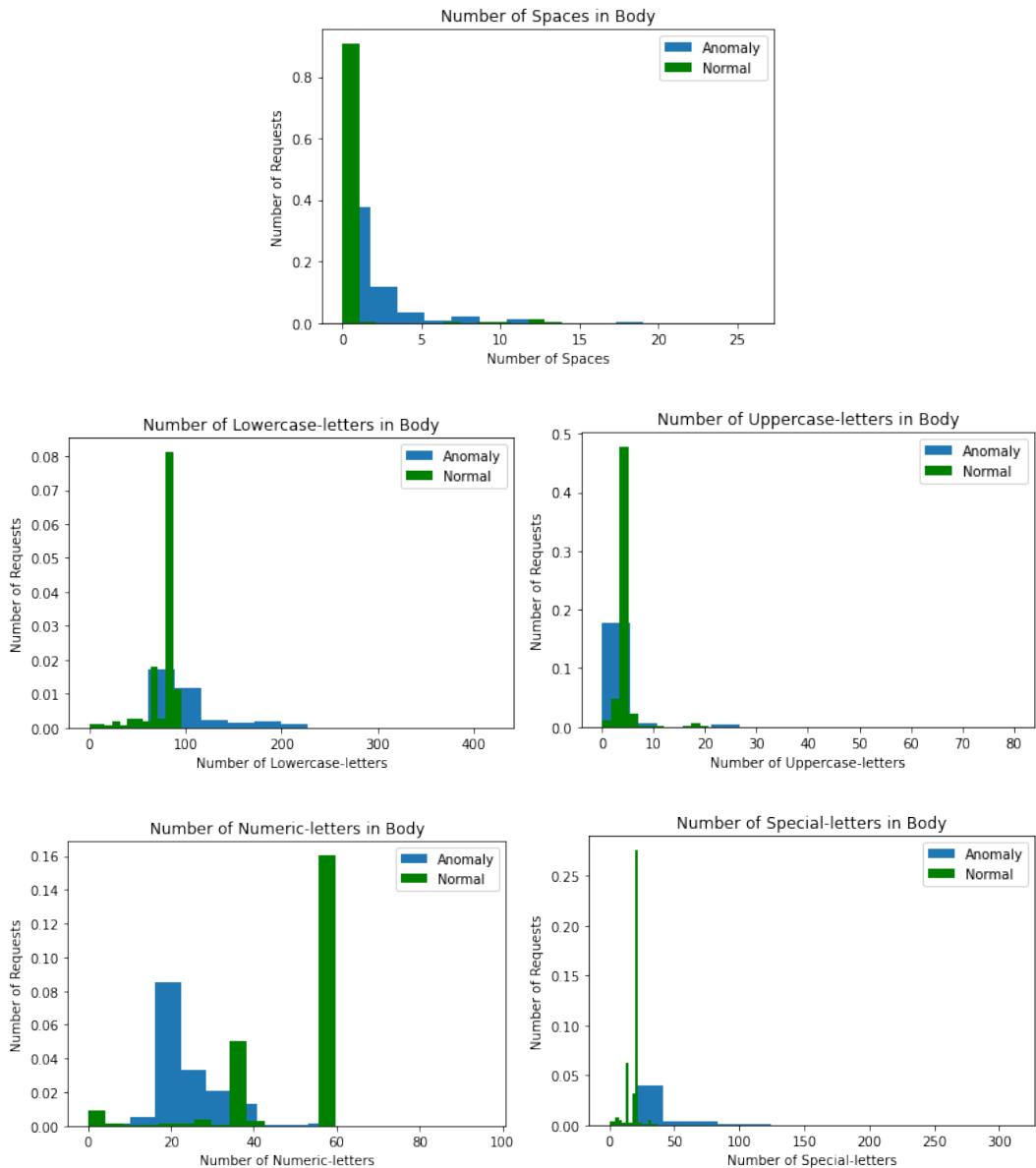


Abbildung 14: Klassen von Zeichen im Body des [5] Datensatzes.

3.2.6 Erkennen von Hashwerten

Alle Analysen, welche die Verteilung von Zeichen betrachten, diese in Klassen einordnen oder mittels Natural Language Processing versuchen Muster in den Daten zu erkennen, funktionieren nur in Zeichenfolgen, die auch eine erkennbare Bedeutung aufweisen [24]. Bei Hashfunktionen wird für Daten beliebiger Länge ein zugehöriger Wert von fester Länge erstellt [45]. Bei kryptographischen Hashfunktionen wird darüber hinaus versucht die Muster in den Ursprungsdaten zu verbergen. Zu diesem Zweck wird die Entropie in den gehashten Daten erhöht, damit nicht mehr auf die ursprünglichen Zeichen geschlossen werden kann. Dies ist besonders bei Passwörtern von Bedeutung.

Da sich in Zeichenfolgen mit einer grossen Entropie keine Muster mehr erkennen lassen, könnten solche Werte das ML-Modell der Angriffserkennung verschlechtern. Aus diesem Grund müssen Hashwerte erkannt und herausgefiltert werden.

Die Hashwerte können anhand folgender Eigenschaften erkannt werden:

- Die Hash-Strings haben immer eine feste Länge und variieren nicht
- Die Hash-Strings können nur in der Path-Query und im Body auftreten
- Die Hash-Strings weisen eine hohe Entropie auf

Gemäss [46] ist die Entropie wie folgt definiert:

Definition 1 (Shannon-Entropie) *Sei Q eine Informationsquelle mit $|Q| = n$ und seien $p_i \in Q$, $1 \leq i \leq n$ die Wahrscheinlichkeiten des Auftretens der Einzelereignisse in Q . Dann ist die Shannon-Entropie einer Quelle Q gegeben durch:*

$$H_1(Q) = -\sum_i p_i * \log_2(p_i)$$

3.2.7 Doppelt codierte Werte

Die URL kann nur im ASCII-Code übermittelt werden. Alle anderen Zeichen werden in den ASCII-Code übersetzt [48]. Damit die Methoden des Natural Language Processings funktionieren, ist es erforderlich, dass alle verarbeiteten Strings auf dieselbe Weise codiert sind. Mittels doppelt codierten Werten können Angreifer versuchen das System auszutricksen, indem sie die Zeichen in der Path-Query oder im Body doppelt codieren[33]. So wird aus `<script>` mittels HTML URL Encoding `%3Cscript%3E` und mittels doppeltem Encoding `%253Cscript%253E`. Zeichenfolgen, welche doppelt codiert sind, können einfach erkannt werden, indem die URL zweimal entschlüsselt wird und mit der URL nach der ersten Entschlüsselung verglichen wird. Falls diese beiden Zeichenfolgen nicht identisch sind, handelt es sich um eine Double-Encoding-Attacke und die Anfrage kann verworfen werden [9].

3.3 Featurequalität

Nachdem die Merkmale gewählt und einzeln betrachtet wurden, konnten diese noch gesamtheitlich betrachtet werden. Zu diesem Zweck wurde die Korrelation zwischen den erstellten Merkmalen betrachtet.

Zuerst wurden die Merkmale für eine spezifische Ressource ohne Query-String in einer Korrelationsmatrix analysiert:

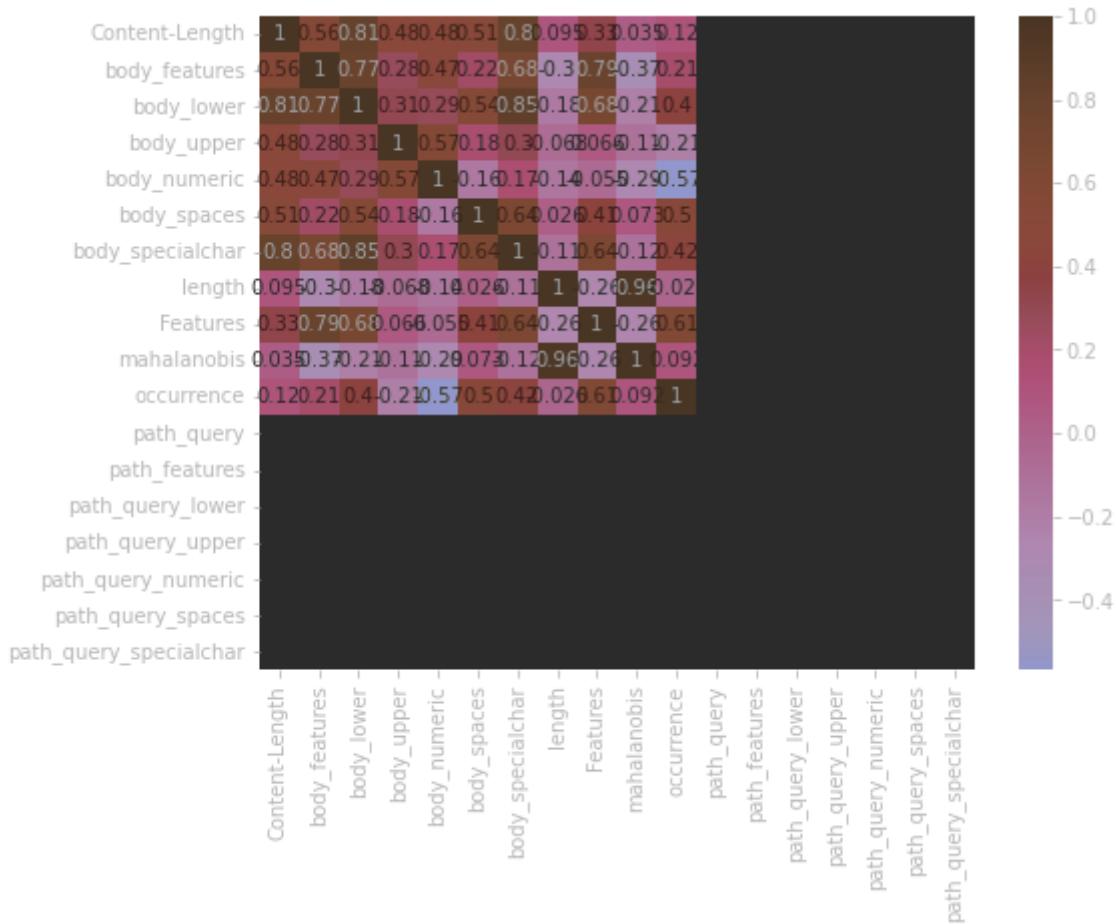


Abbildung 15: Korrelationsmatrix, eingeschränkt auf eine spezifische Ressource ohne Path-Query im [5] Datensatz.

Es ist zu erkennen, dass die einzelnen Klassen von Zeichen teilweise stark mit der Länge korrelieren. Die Ursache dafür ist, dass alle Klassen summiert genau die Länge ergeben. Um dies zu verhindern, wurden die Klassen später mit der Länge normiert.

Als Nächstes wurde die Korrelation für alle Ressourcen mit den jeweiligen Path-Queries betrachtet:

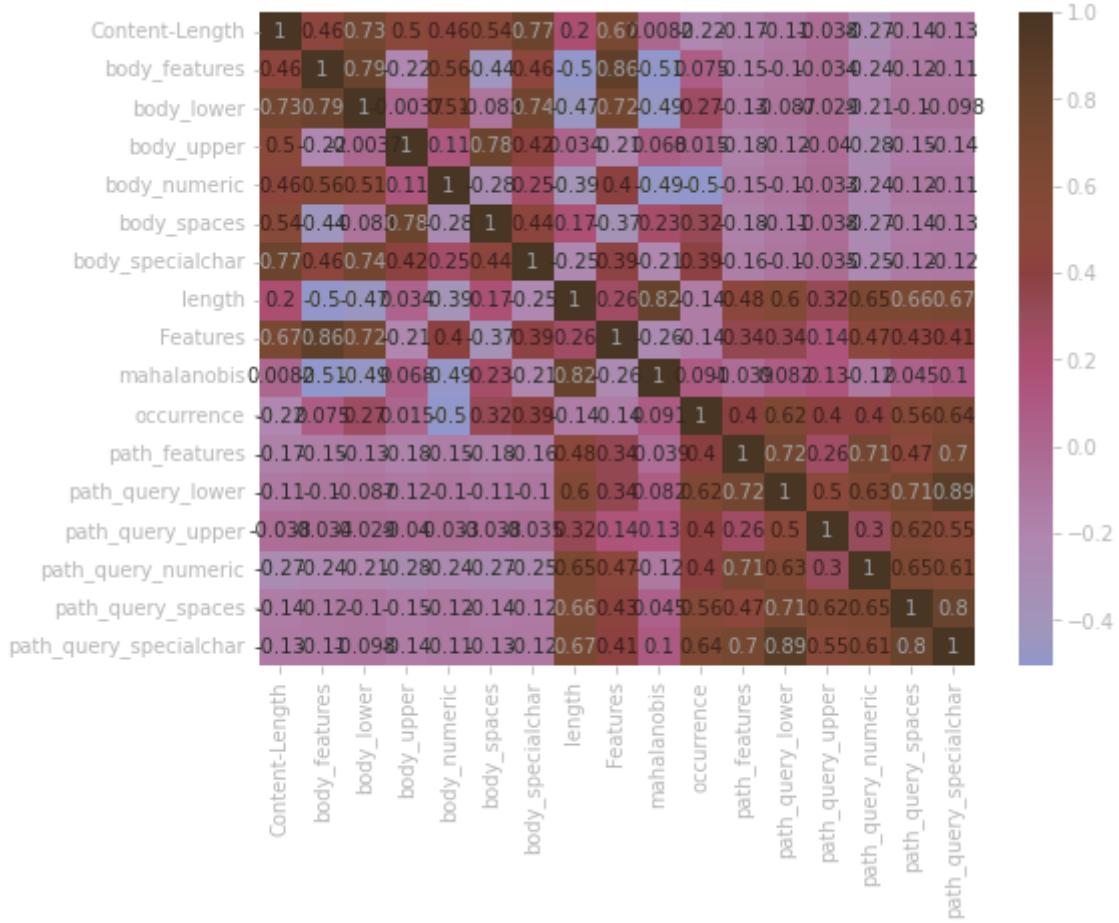


Abbildung 16: Korrelationsmatrix, eingeschränkt auf alle Ressourcen mit einer Path-Query im [5] Datensatz.

Die Analyse zeigt, dass auch beim Query-String die Zeichenklassen teilweise untereinander stark korrelieren. Daher wurden auch diese Features mit der Länge normiert.

Daraufhin wurde mit den jeweiligen Merkmalen eine PCA mit zwei Komponenten erstellt.

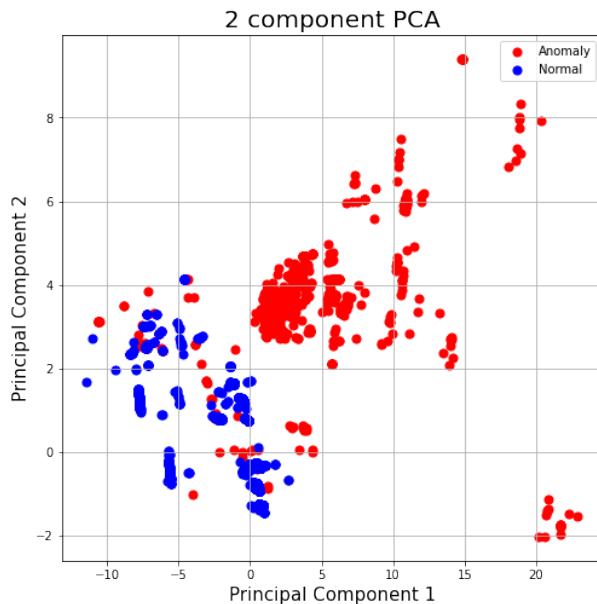


Abbildung 17: PCA mit zwei Komponenten, eingeschränkt auf eine Ressource ohne N-Gramme im [5] Datensatz.

Die PCA verdeutlicht, dass mit den gewählten Merkmalen die normalen und die anomalen Anfragen bereits gut separiert werden können.

Zusätzlich wurde eine PCA mit zwei Komponenten und den N-Gramm-Merkmalen erstellt:

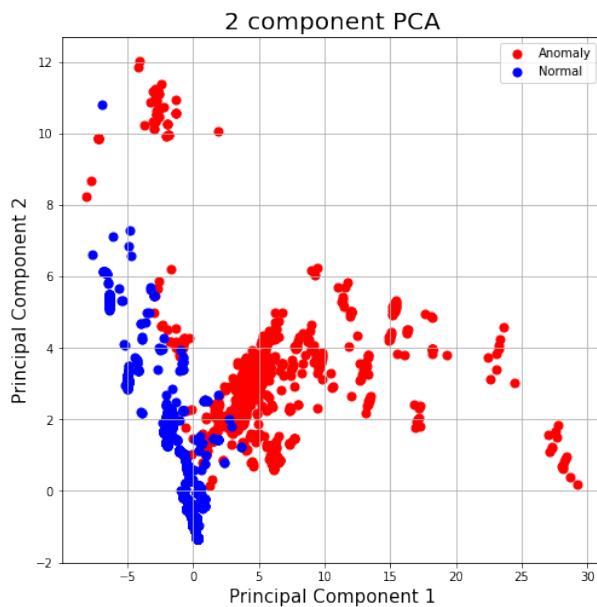


Abbildung 18: PCA mit zwei Komponenten, eingeschränkt auf eine Ressource mit N-Grammen im [5] Datensatz.

Die Visualisierung zeigt, dass mit den N-Gramm-Merkmalen die Daten in der zweidimensionalen PCA noch besser separiert werden können. Zur besseren Analyse wurde zusätzlich eine weitere PCA mit drei Komponenten erstellt:

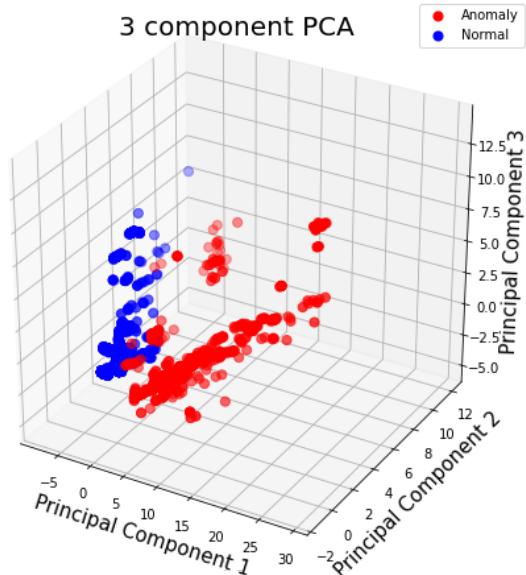


Abbildung 19: PCA mit drei Komponenten, eingeschränkt auf eine Ressource mit N-Grammen im [5] Datensatz.

Die Visualisierung zeigt eine erfolgreiche Trennung der normalen Anfragen von den Anomalien.

Zur besseren Visualisierung wurden noch die drei Projektionen der Drei-Komponenten-PCA erstellt:

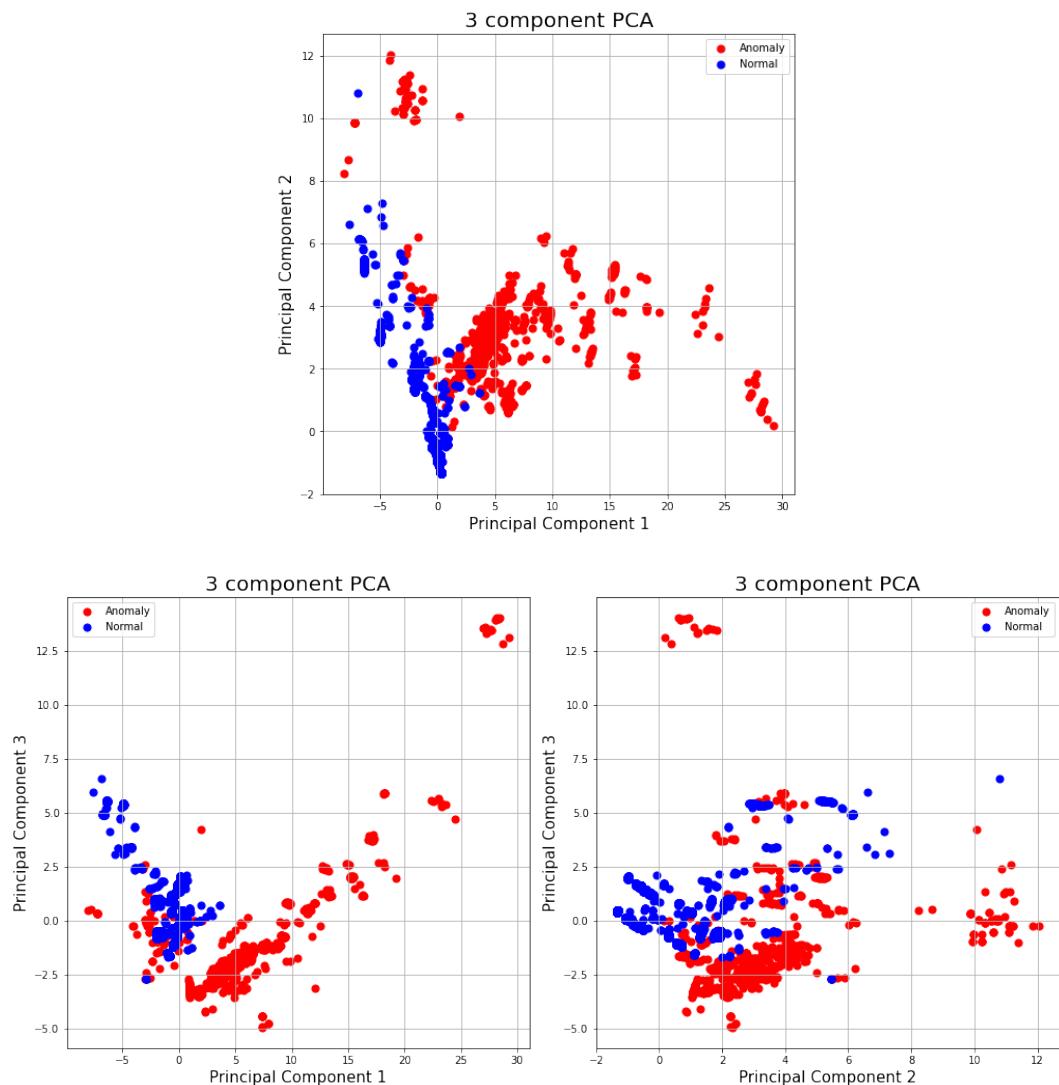


Abbildung 20: PCA mit drei Komponenten, Projektionen 1-3 im [5] Datensatz.

4 Implementierung

Auf Basis der erlangten Erkenntnisse aus dem Data Mining konnte im nächsten Schritt eine Software erstellt werden, die als Prototyp des entwickelten Intrusion Detection Systems dienen soll und für einen ersten Use-Case eingesetzt werden soll. Da die Aufgabe eines Intrusion Detection System auf mehreren nacheinanderfolgenden Komponenten basiert, wird im Folgenden von einer Daten-Pipeline gesprochen, in der Daten (hauptsächlich HTTP-Nachrichten) von einer Komponente zur Nächsten weitergereicht werden. In diesem Kapitel werden auf die Architektur und die Eigenschaften der erstellten Software eingegangen.

4.1 Grundlagen

Die Software wurde in der Programmiersprache Python geschrieben, da somit viele Data Science Bibliotheken verwendet werden konnten, die bei der Erstellung und Benutzung verschiedenster Machine Learning Modelle von grossem Nutzen waren. Um die Übersichtlichkeit und Modularität der Software zu gewährleisten wurde der objektorientierte Ansatz gewählt.

Die Software kann entweder im Trainings- oder im Testmodus gestartet werden. Befindet sich die Pipeline im Trainingsmodus, lernt das System und nimmt immer mehr Daten in die Analyse mit auf. Falls sich das System im Testmodus befindet, werden keine neuen Daten in die Analyse mit einbezogen und es werden bloss die eintreffenden Anfragen auf potenzielle Angriffe untersucht.

4.2 Architektur

Zu Beginn der Implementierung wurden die Anforderungen an die Software analysiert und die daraus resultierende Architektur abgeleitet.

4.2.1 Anforderungen an die Architektur

Mit der Softwarearchitektur sollen folgende Ziele erreicht werden:

Anforderung	Beschreibung	Architekturentscheid
Adaptierbarkeit auf neue Technologien	Die Software soll als akademischer Prototyp für Testzwecke dienen und daher einfach an neue Datenbanken und Reverse-Proxy-Server angepasst werden können.	Logisch zusammenhängende Schritte sollen in Containern zusammengefasst werden. Die einzelnen Container sollen über klar definierte Interfaces kommunizieren.
Adaptierbarkeit auf unterschiedliche ML-Algorithmen	Die Software soll einfach auf neue Machine Learning Algorithmen und neue Features angepasst werden können.	Alle Funktionalitäten, welche mit der Erkennung von Angriffen zusammenhängen, sollen als Plugin implementiert werden.
Einfache Erweiterung auf Intrusion Prevention	In der Praxis sollen Angriffe nicht nur erfolgreich erkannt, sondern auch erfolgreich abgewehrt werden. Die erstellte Architektur soll mit geringen Anpassungen auf Intrusion Prevention angepasst werden können.	Der Dataflow soll als Pipeline implementiert werden. Diese soll mit geringem Aufwand erweitert werden können, um Angriffe zu blockieren.
Software soll performant auf Systemen mit limitierten Ressourcen laufen	Die gewählte Architektur soll mit möglichst wenig Arbeitsspeicher und Datenbank-Speicher zuverlässig arbeiten.	Wachsende Daten welche im Arbeitsspeicher gehalten werden, sollen periodisch aggregiert werden. Die Trainingsdaten für das Training des ML-Modells aus der Datenbank sollen auf eine representative Untermenge reduziert werden.

Tabelle 5: Architekturanforderungen

4.2.2 Bausteinsicht

Zur Visualisierung der aus der Tabelle 5 abgeleiteten Architekturentscheidungen wurde ein UML-Komponentendiagramm erstellt:

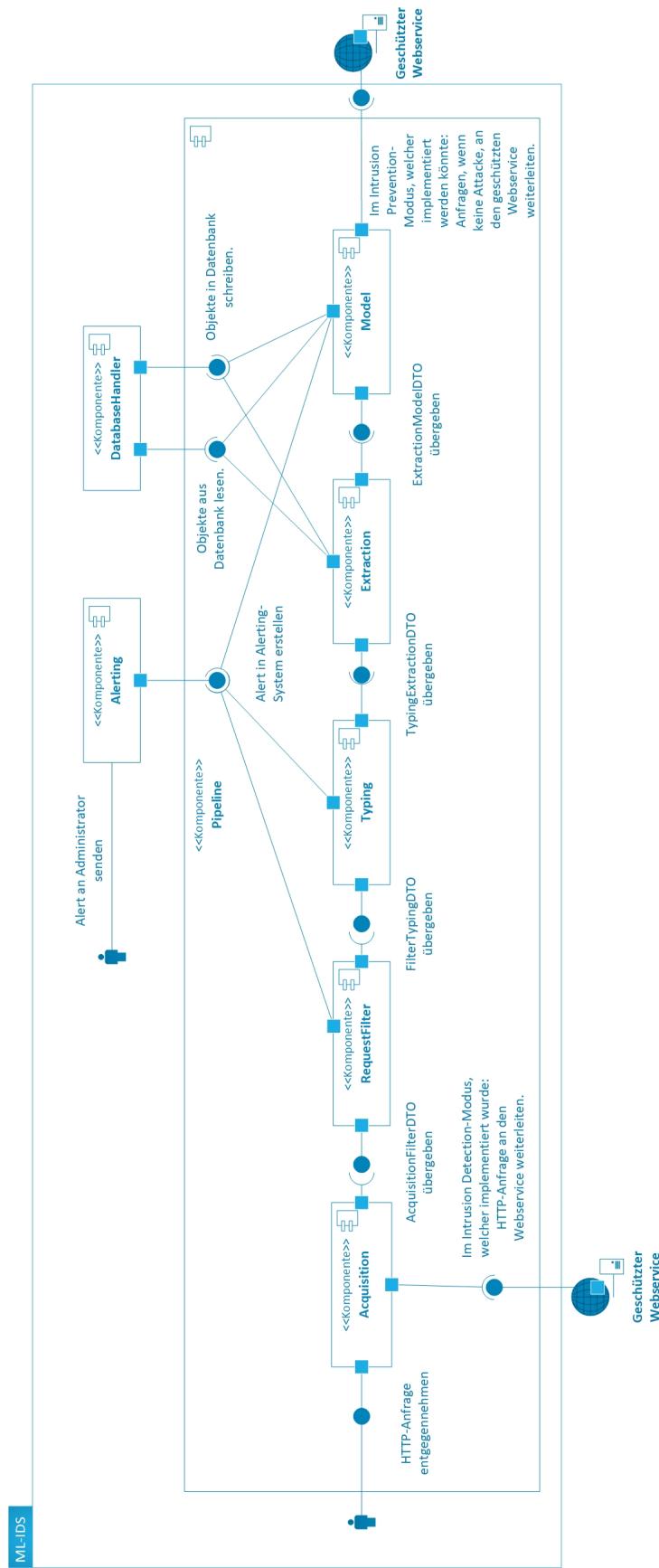


Abbildung 21: Komponentendiagramm des ML-IDS

Die Software besteht aus folgenden Komponenten:

Komponente	Beschreibung
Pipeline	Die Pipeline ist der Eintrittspunkt der Software und steuert die Erstellung der einzelnen Komponenten/Stages/Container.
Acquisition	Die Akquirierungskomponente nimmt die HTTP-Anfragen entgegen.
RequestFilter	Die Filterkomponente prüft die HTTP-Anfrage anhand statischer Regeln. Bei einem Angriff benachrichtigt diese Komponente die Komponente Alerting.
Typing	Die Typisierungskomponente stellt fest, um welchen Typ von Anfrage es sich handelt und bestimmt dadurch, welche Merkmale aus der Anfrage extrahiert werden. Bei einem unbekannten Typ von Anfrage, benachrichtigt diese Komponente die Komponente Alerting.
Extraction	Die Extraktionskomponente erstellt aus der Anfrage den Feature-Vector und speichert diesen in der Datenbank.
Model	Die Modellkomponente prüft anhand eines ML-Modells, ob es sich bei der Anfrage um einen Angriff handelt. Bei einem Angriff benachrichtigt diese Komponente die Komponente Alerting.
Alerting	Die Komponente Alerting wird benachrichtigt, wenn ein Angriff detektiert wurde und erstellt einen Alert.
DatabaseHandler	Die Komponente DatabaseHandler übernimmt das Lesen und Schreiben aus und in die Datenbank. Durch diese Komponente sind die anderen Komponenten von der Datenbank entkoppelt.

Tabelle 6: Beschreibung der Softwarekomponenten

4.2.3 Laufzeitsicht

In der Laufzeitsicht wurde mittels der Abbildung 22 beschrieben, wie die unterschiedlichen Komponenten erstellt werden und miteinander interagieren:

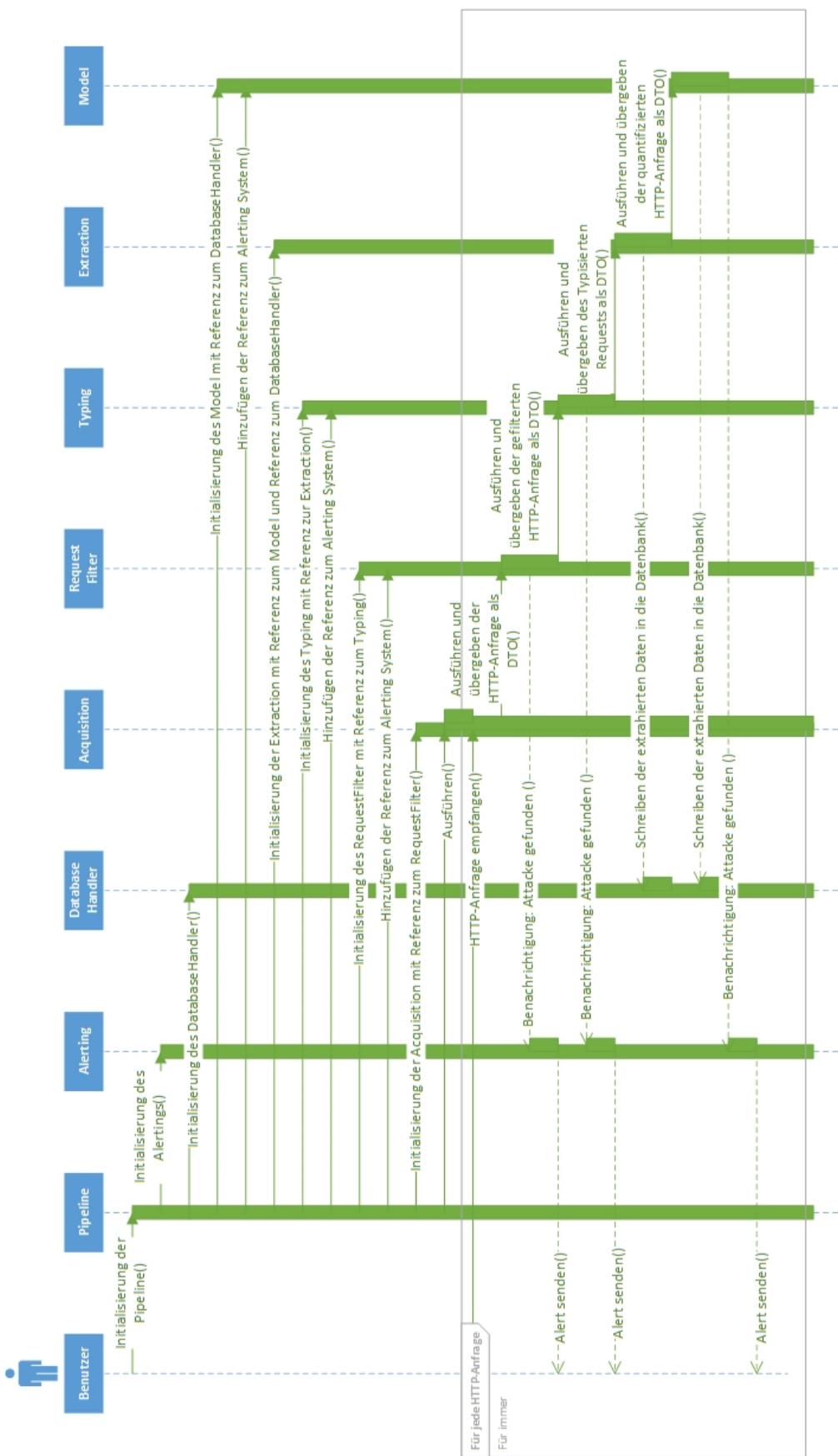


Abbildung 22: Sequenzdiagramm des ML-IDS

4.3 Pipeline: Inversion of Control

Um die Architektur der nacheinander folgenden IDS-Komponenten sinnvoll umzusetzen, wurde das Inversion of Control Pattern gewählt [28]. Dabei werden alle Instanzen der einzelnen Pipeline-Komponenten von einer kontrollierenden Stelle - dem sogenannten Inversion of Control Container - erstellt und verwaltet. Bei der Initialisierung der Software wird dann jeder Komponente die Instanz der nachfolgenden Komponente übergeben, so dass jede weiß, wohin die Daten weitergeleitet werden müssen. Dabei implementieren alle Komponenten das Interface „Stage“, in der die Methode „run“ deklariert ist, welche vom jeweiligen Vorgänger aufgerufen werden kann. Dieser Methode werden auch die zu verarbeitenden Daten übergeben.

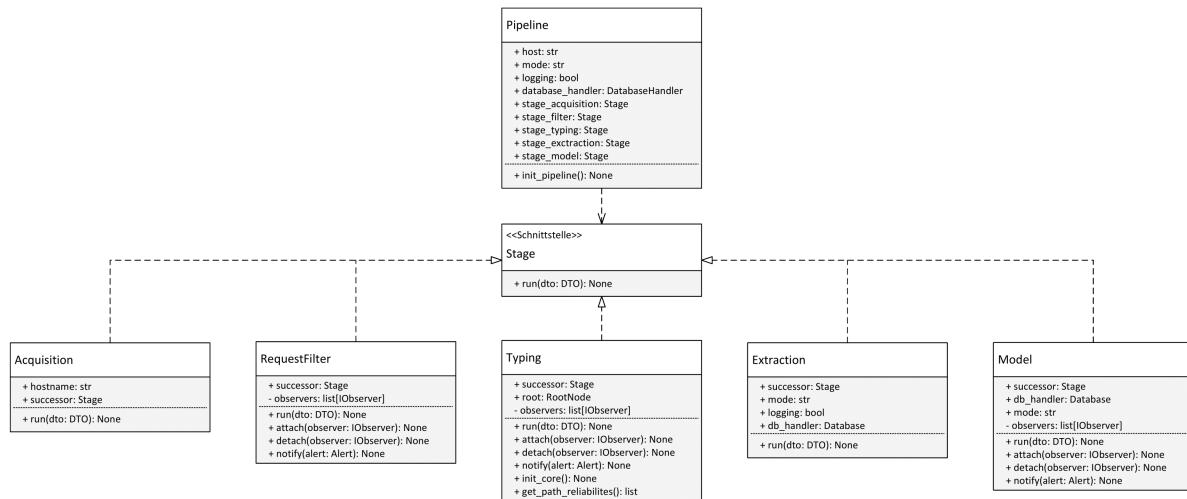


Abbildung 23: Klassendiagramm der Inversion of Control Struktur

4.4 Data Transfer Objects

Um die Daten in einer einheitlichen Form von einer Komponente zur Nächsten zu übergeben, wurden sogenannte Data Transfer Objects - auch DTOs - eingesetzt [13]. Jedes konkrete DTO wird durch eine Python-Dataclass repräsentiert [38]. Eine Dataclass ist eine Klasse, die nur Attribute und keine Methoden beinhaltet. Somit kann für jeden Datenaustausch zwischen zwei Pipeline-Komponenten ein konkretes DTO mit spezifischen Attributen definiert werden, welche alle das Interface „DTO“ implementieren.

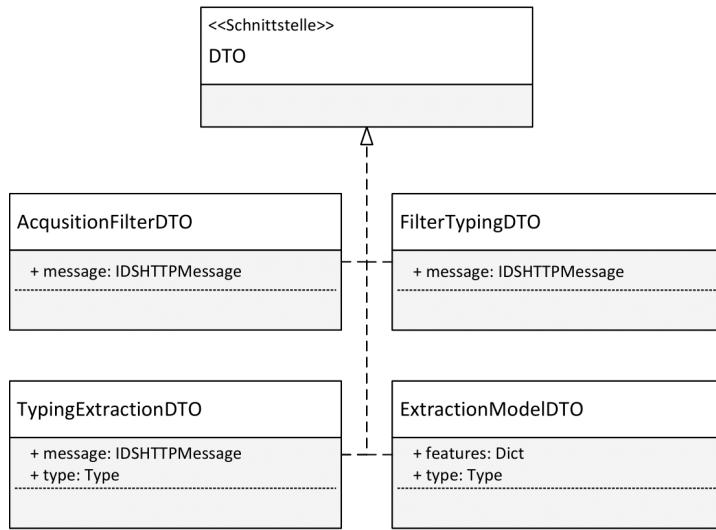


Abbildung 24: Klassendiagramm der Data Transfer Objects

4.5 Alerting

Die Hauptaufgabe eines Intrusion Detection Systems ist es, bei einer verdächtigen Nachricht, die die Sicherheit des zu schützenden Systems gefährdet, eine Art Alarm auszugeben. Da in dieser Software mehrere Pipeline-Stufen einen solchen Alert auslösen können, wurde ein Observer-Pattern implementiert [14]. Das Alerting-Modul ist dabei der Observer und implementiert somit das „*IObserver*“-Interface. Alle Komponenten, die einen Alert auslösen können, implementieren wiederum das „*IObservable*“-Interface. Bei der Initialisierung der Software instanziert der Inversion of Control Container das Alerting-Modul und fügt dieses in der Rolle des Observers an jede Pipeline-Komponente an, die das Observable-Interface implementiert. Somit kann durch jede dieser Komponenten allein durch das Aufrufen der „*notify*“-Methode ein Alert ausgelöst werden.

Wird ein Alert ausgelöst, wird dieser mit dem logging-Modul von Python auf der Konsole ausgegeben und in eine Logdatei geschrieben. Die Schreibzugriffe auf die Datei finden in einer Warteschlange statt, die asynchron abgearbeitet wird. Somit wird der Programmfluss nicht durch das Logging unterbrochen. Protokolliert wird jeweils der Grund und die Quelle des Alerts.

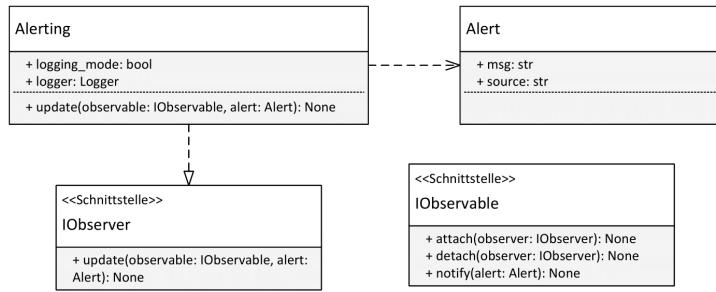


Abbildung 25: Klassendiagramm des Alerting

4.6 Datenbank

Die meisten Berechnungen und Analysen der Software laufen im Hauptspeicher ab. Je- doch gibt es einige Vorgänge, wie die N-Gramm-Analyse oder das Trainieren des Machine Learning Modells, die von älteren Daten abhängig sind. Auch die spätere Datenanalyse und Diskussion der Ergebnisse setzen eine Sammlung von abgespeicherten Daten voraus. Somit musste eine persistente Datenbank eingesetzt werden. Um die Verbindung zwischen Software und Datenbank möglichst einfach zu halten, fiel die Entscheidung auf eine native Objektdatenbank namens ZODB [49]. Der Vorteil hierbei ist es, dass Python-Objekte als sogenannte Pickles, also serialisiert, in die Datenbank unter einem Namespace abgelegt und wieder ausgelesen werden konnten. Der dazu benötigte Code beschränkte sich auf ein Minimum.

Um die benötigten Datenbankfunktionen möglichst einfach und ohne Konflikte zur Verfügung zu stellen wurde ein Strategy Pattern [14] implementiert, da die verschiedenen Komponenten auf unterschiedliche Art und Weise auf die Datenbank zugreifen können müssen. Die Klasse „DatabaseHandler“ ist dabei die Kontextklasse, mit der auf eines der verschiedenen Strategies zugegriffen werden kann. Dank der Implementierung dieses Patterns ist es für verschiedenste Komponenten oder auch Plugins des IDS möglich, mit einem einheitlichen Interface auf die Datenbank zuzugreifen und Objekte persistent zu schreiben bzw. zu lesen.

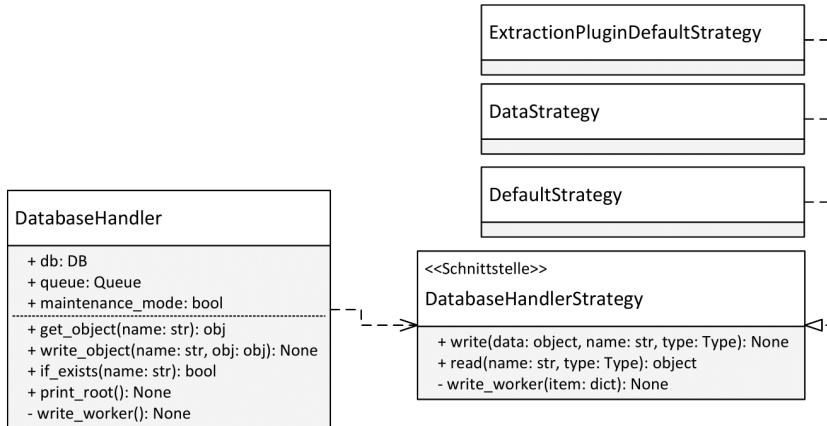


Abbildung 26: Klassendiagramm des Datenbank Strategy-Patterns

Da die Datenbank von Haus aus atomare Transaktionen unterstützt, können die Lesezugriffe synchron und von mehreren Threads aus ohne Probleme durchgeführt werden. Um Wartezeiten und Konflikte bei Schreibzugriffen zu verhindern, wurde entschieden, diese asynchron und mithilfe eines Daemon Threads, welcher eine Warteschlange mit dem FIFO-Prinzip implementiert, durchzuführen. Somit wird garantiert, dass alle Schreibzugriffe von demselben Thread aus geschehen und die restliche Programmausführung nicht unterbrochen wird, wenn gerade ein anderes Objekt geschrieben wird. Die Pipeline hat somit die Möglichkeit, Objekte, welche in die Datenbank geschrieben werden müssen, über den Database Handler in die Warteschlange einzureihen und mit dem Programm sofort fortzufahren. Der Daemon Thread arbeitet nach und nach alle zu schreibenden Objekte ab.

Der Database Handler wird vom Inversion of Control Container instanziert und an die Komponenten weitergegeben, die einen Datenbankzugriff benötigen.

4.7 Komponenten

Im Folgenden werden die fünf Stufen der Pipeline genauer dargestellt.

4.7.1 Akquirierung

Die erste Aufgabe eines Intrusion Detection Systems ist das Abfangen bzw. das Abhören der zu untersuchenden Nachrichten. Da sich diese Arbeit nur mit der Untersuchung der HTTP-Nachrichten beschäftigt, wurde die Akquirierungs-Komponente durch einen HTTP-Reverse-Proxy-Server realisiert. Beim Einsatz eines Reverse-Proxys nimmt dieser stellvertretend die HTTP-Anfragen an und leitet sie an das eigentliche Backend weiter.

Auch die Antwort wird über den Reverse-Proxy zum jeweiligen Client umgeleitet. Der Client merkt dabei nicht, dass er eigentlich mit dem Reverse-Proxy-Server kommuniziert und nicht direkt mit dem eigentlichen Service. Somit kann das Intrusion Detection System die HTTP-Anfrage vor dem Erreichen des eigentlichen Services analysieren. Im Falle des Einsatzes eines Intrusion Prevention Systems könnten sogar spezifische Sicherheitsaktionen vorgenommen werden, zum Beispiel könnte die Anfrage gar nicht erst an den Service zur Bearbeitung weitergeleitet werden.

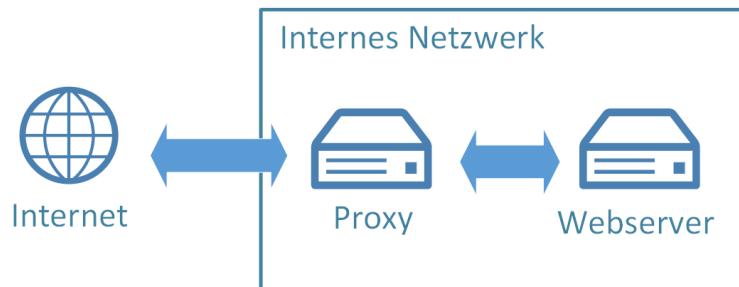


Abbildung 27: Reverse Proxy

Im Rahmen dieser Software wurde ein einfacher HTTP-Server von Python verwendet, der die ankommenden HTTP-Anfragen entgegennimmt und diese an die folgende Komponente zur weiteren Analyse weitergibt. Direkt anschliessend wird die Anfrage an den Service weitergeleitet und die Antwort wird zurück an den Client gesendet. Da nur ein IDS und kein IPS entwickelt wurde, hat das Ergebnis der Analyse keinen Einfluss auf die Weiterleitung der Anfrage an den Service. Damit das System mehrere Anfragen innerhalb kürzester Zeit entgegennehmen kann und die Ausführung nicht durch die Nachrichtenanalyse blockiert wird, wird für jede Anfrage ein eigener Thread gestartet.

4.7.2 Filter

Die zweite Stufe des Systems ist die Filterkomponente, deren Aufgabe es ist, bestimmte Anfragen bereits auf dieser Stufe zu verwerfen und gar nicht erst zur weiteren Analyse an die nächste Komponente weiterzuleiten. Zu filternde Nachrichten könnten zum Beispiel solche sein, die durch ihre hohe Länge oder hohe Frequenz eine Overflow-Attacke darstellen. Durch die Filterung werden die nachfolgenden Komponenten wie zum Beispiel das ML-Modell nicht überfordert. Weiter könnten Nachrichten gefiltert werden, die auf Ressourcen zugreifen möchten, die auf dem eingesetzten System gar nicht existieren. Da für jeden Use-Case und für jedes zu schützende System andere Filter Sinn ergeben, wurde bei dieser Stufe der Pipeline eine Plugin-Architektur realisiert [14]. Ein Nutzer hat dabei die Möglichkeit, einen eigenen Filter einzubinden, der das Interface „FilterPluginInterface“ implementiert. Alle Plugins, die in einem bestimmten Ordner liegen, werden von

der Software dynamisch eingebunden und nacheinander für jede Nachricht verwendet. Jeder konkrete Filter hat die Möglichkeit durch den Rückgabewert der Filter-Methode der Komponente mitzuteilen, ob das Paket ausgefiltert wird. Ist dies der Fall, wird von der Komponente, welche das „*IObservable*“-Interface implementiert, ein Alert ausgelöst.

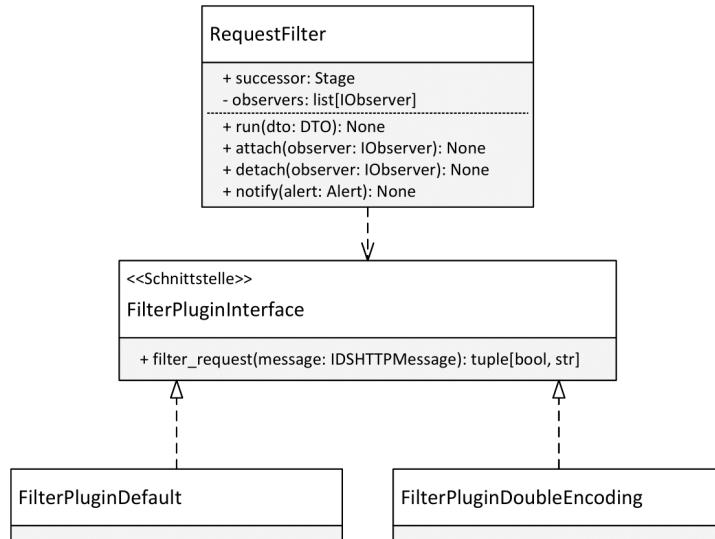


Abbildung 28: Klassendiagramm der Filterkomponente

4.7.3 Typisierung

In der Data-Mining Phase des Projektes wurde festgestellt, dass HTTP-Anfragen extrem unterschiedlich sein können. Je nach Art der Anfrage und je nach der zuzugreifenden Ressource unterscheiden sich die Merkmale der Anfrage stark. Es ist somit unmöglich ein Machine Learning Modell zu trainieren, das solch unterschiedliche HTTP-Anfragen als eine Datenquelle betrachtet. Es wurde festgestellt, dass das Modell nur dann eine hohe Genauigkeit erzielt, wenn nur HTTP-Anfragen eines bestimmten Typs isoliert betrachtet werden. Dazu teilt die Typisierungskomponente der Software jeder ankommenden HTTP-Anfrage einen bestimmten Typ zu. Dieser besteht aus folgenden Attributten:

- Zugriffsmethode: z.B. GET, POST, etc.
- Zugriffspfad: z.B. /web/index.php
- Ob Body enthalten
- Ob Path-Query enthalten

Dieser Typ wird zusammen mit der zugehörigen HTTP-Anfrage an die nächste Komponente weitergegeben.

Die Typisierungskomponente hat jedoch noch eine zweite Aufgabe: Durch die Aufzeichnung jeder HTTP-Anfrage wird nach und nach eine baumartige Struktur erstellt, in der die Zugriffsmethode (GET, POST, usw.) die Wurzel, die Unterverzeichnisse des Zugriffspfades die Zwischenknoten und die Ressource das Blatt des Baumes darstellt.

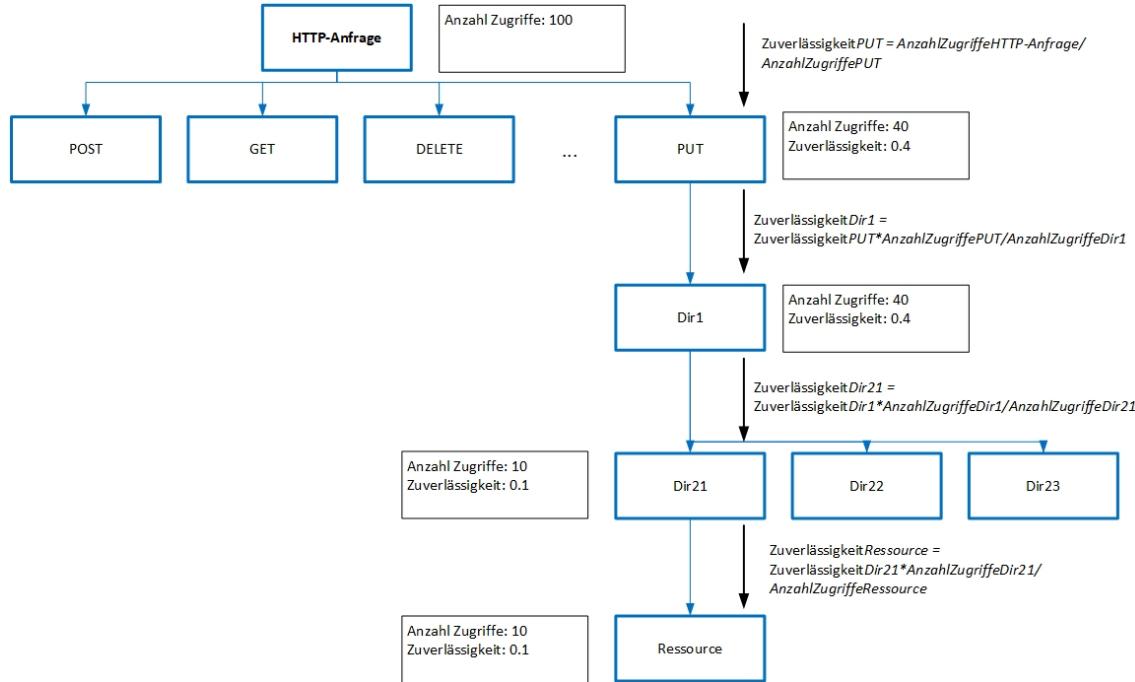


Abbildung 29: Baumstruktur der Typisierungskomponente

Das Ziel dabei ist es, zu erkennen welche Zugriffe üblich und welche eher unüblich sind. Bei jeder HTTP-Anfrage wird in allen Knoten des Baumes, die sich entlang des jeweiligen Pfades befinden, der aktuelle Zeitstempel abgespeichert. Somit wird festgehalten, auf welche Pfade wie häufig zugegriffen wird. In jedem Knoten wird eine Zuverlässigkeit mit folgender Formel berechnet:

$$r = \frac{\text{Anzahl Zugriffe}}{\text{Anzahl Zugriffe auf den Elternknoten}} \quad (3)$$

Um die Zuverlässigkeit des gesamten Pfades zu berechnen, werden die jeweiligen Zuverlässigkeitswerte der Knoten miteinander multipliziert. Wenn jetzt beispielsweise sehr häufig ein GET auf die Ressource /web/index.html ausgeführt wird und dann plötzlich versucht wird auf die Datei /web/admin.html zuzugreifen, erhält dieser neu angelegte Knoten und somit dieser Pfad eine sehr geringe Zuverlässigkeit. Die aufgezeichneten Zeitstempel, die für die Zuverlässigungsberechnung nötig sind, werden automatisch auf einen mittelfristigen und auf einen langfristigen Zeitraum aggregiert, indem alle Zugriffe, die älter als eine Stunde bzw. einen Tag sind, auf eine Anzahl im Stunden- bzw. Tageintervall

heruntergebrochen werden. Damit die Zuverlässigkeitseinformationen immer aktuell bleiben, werden die Zugriffe, die älter als sieben Tage sind, aus der Aufzeichnung entfernt. Bei der Berechnung des Zuverlässigkeitswertes eines Knotens wird je nach Alter des Knotens die Anzahl der kurzfristigen, mittelfristigen oder langfristigen Zugriffe berücksichtigt. Anhand des resultierenden Pfad-Zuverlässigkeitswertes, welcher immer zwischen 0 und 1 liegt, kann bei Unterschreitung eines bestimmten Grenzwertes ein Alert geworfen werden. Ein sinnvoller Grenzwert wird im Folgenden durch Tests ermittelt. Um zu verhindern, dass bei erstmaligem oder seltenem Zugriff auf eine eigentlich gültige Ressource ein Alert ausgelöst wird, hat der Nutzer die Möglichkeit sogenannte Kernpfade in einer config-Datei zu definieren. Dessen Zuverlässigkeit betragen immer 1.

```

1  {
2      "paths": [
3          {
4              "methods": [
5                  "GET"
6              ],
7              "path": "/"
8          },
9          {
10             "methods": [
11                 "GET"
12             ],
13             "path": "/index.html"
14         },
15         {
16             "methods": [
17                 "GET",
18                 "POST"
19             ],
20             "path": "/var/www/endpoint1"
21         }
22     ]
23 }
```

Abbildung 30: Config-Datei zur Definierung vertrauenswürdiger Pfade

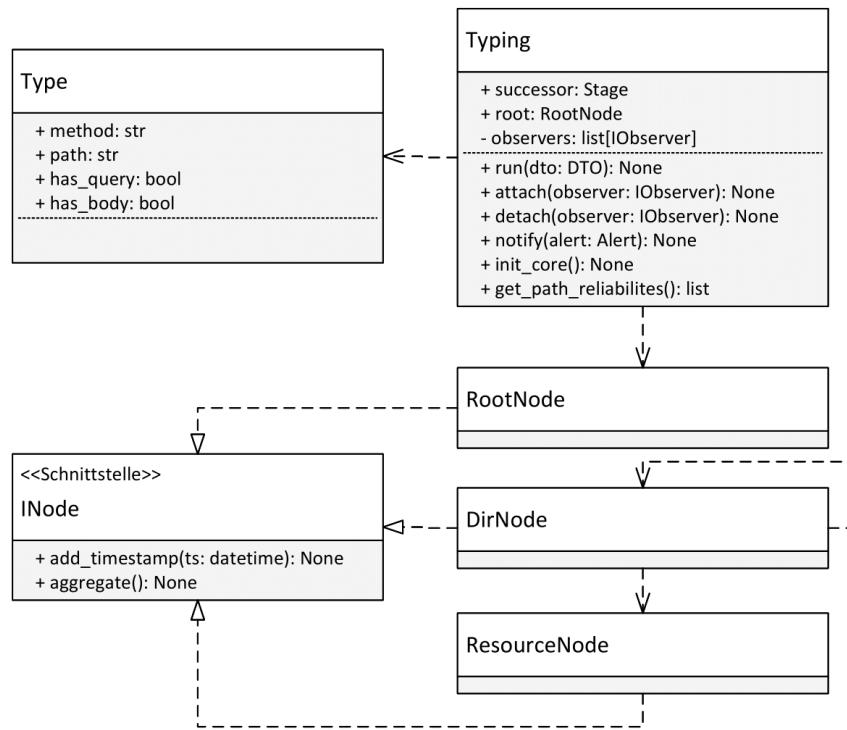


Abbildung 31: Klassendiagramm der Typisierungskomponente

4.7.4 Merkmalsextraktion

Die Aufgabe der vierten Komponente der Pipeline ist die Extraktion und die Quantifizierung der Merkmale aus einer HTTP-Anfrage, welche dann an das Machine Learning Modell weitergegeben werden können. Basierend auf dem in der vorherigen Komponente ermittelten Typs werden andere Merkmale extrahiert. Beispielsweise werden aus Anfragen, welche einen Body beinhalten, mehr Merkmale extrahiert, wie aus Anfragen ohne Body. Wie in der Filter-Komponente wurde hier eine Plugin-Struktur implementiert, sodass für verschiedene Einsatzgebiete weitere, spezielle Merkmale extrahiert werden können.

Ein Standard-Extraktions-Plugin beinhaltet die Merkmale, welche im Kapitel 3 Data-Mining bereits beschrieben wurden. Für die N-Gramm-Analyse werden in dieser Komponente die gewünschten N-Gramme berechnet und falls sich das System im Trainingsmodus befindet, zusammen mit einem Zeitstempel in die Datenbank abgelegt. Der damit entstehende N-Gramm-Referenzpool wird periodisch aggregiert, das heisst die ältesten aufgezeichneten N-Gramme werden verworfen, sodass eine Veränderung des Pools über die Zeit möglich ist. Gleichzeitig wird mithilfe des Pools für jede Anfrage der im Kapitel 3 Data-Mining beschriebene Wert s berechnet und im Feature-Dictionary abgelegt. Das Dictionary mit den Merkmalsnamen als Schlüssel und den quantifizierten Daten als Werte

wird zusammen mit dem Typ der Anfrage an die Modell-Komponente weitergegeben. Falls sich das System im Trainingsmodus befindet, wird in dieser Komponente ausserdem das ermittelte Merkmals-Dictionary in die Datenbank abgelegt, damit das folgende Machine Learning Modell periodisch neu trainiert werden kann.

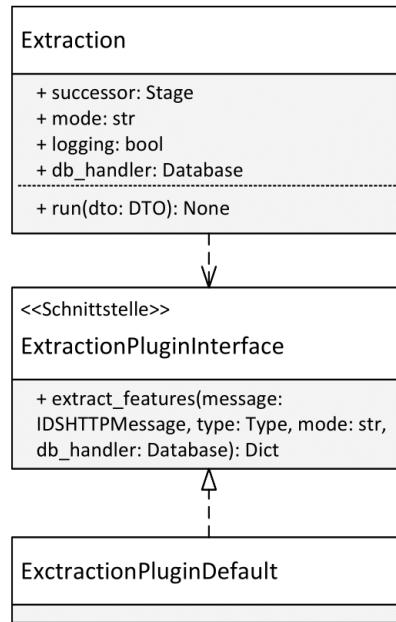


Abbildung 32: Klassendiagramm der Merkmalsextraktion

4.7.5 Modell

In der letzten Stufe des Systems ist die Model Stage, welche zu jedem Typ ein trainiertes ML-Modell enthält. Die Modellkomponente erhält von der Merkmalsextraktion einen Feature-Vector und macht anhand des trainierten ML-Modells eine Einschätzung, ob es sich bei diesem Zugriff um einen Angriff handelt oder nicht. Die Modellkomponente wurde als Plugin-Architektur implementiert, um verschiedene ML-Modelle zu testen.

Jedes Modell-Plugin beinhaltet ein Factory-Pattern [14], welches zu jedem Typen eine neue Instanz der jeweiligen ML-Implementierung speichert und zurückgibt. Das Plugin stellt die Schnittstelle, um das ML-Modell zu trainieren oder ein Feature-Vektor auszuwerten zur Verfügung.

Das Machine Learning Modell selbst ist in einer eigenen Klasse implementiert. Das Standard-Plugin nutzt logistische Regression [6] als ML-Algorithmus, um einen Angriff zu erkennen. Neben der logistischen Regression, welche zu den überwachten Lernmethoden gehört [29], wurde mit dem k-Means-Clustering Algorithmus [30] noch eine Methode

implementiert, welche zu den unüberwachten Lernmethoden gehört [29]. Dabei erstellt der k-Means-Algorithmus anhand der gespeicherten Feature-Vektoren aus der Datenbank verschiedene Cluster und weist diese Cluster anhand von gelabelten Daten einem Angriffscluster oder Nicht-Angriffscluster zu. Zur Bestimmung, ob eine Anfrage ein Angriff ist, wird geprüft, welchem Cluster diese Anfrage zugeordnet werden kann. Ziel der Implementierung einer unüberwachten Lernmethode ist, dass im Trainingsmode der Pipeline neue Anfragen automatisch gelabelt werden können und so dem Administrator die Arbeit beim Labeln während des Trainingsvorgangs erleichtern.

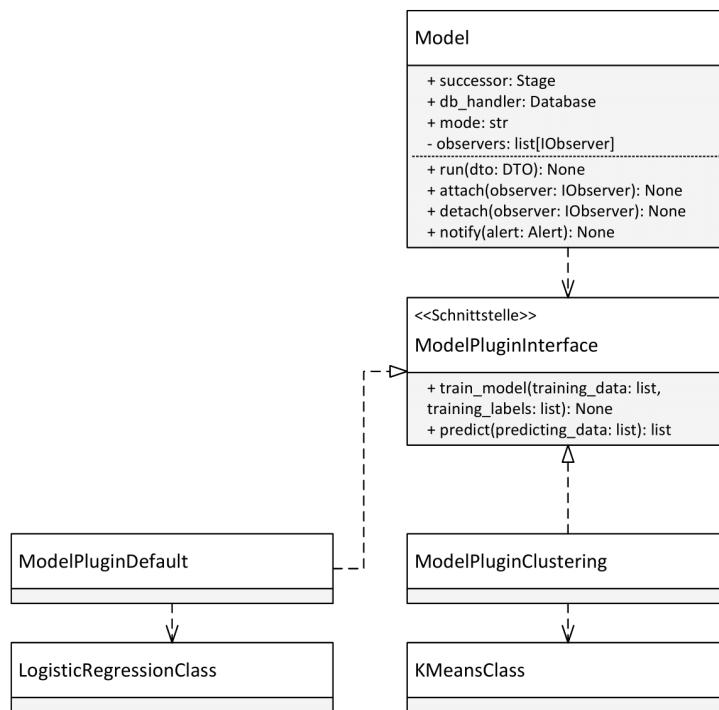


Abbildung 33: Klassendiagramm der Modellkomponente

4.8 Trainingshypothese

Beim Start der Pipeline kann der Administrator den Modus „train“ oder „test“ verwenden. Im Modus „train“ werden die eingehenden Anfragen in der Datenbank gespeichert und müssen durch den Administrator gelabelt werden. Im Modus „test“ wird anhand der gelabelten Daten in der Datenbank eine Voraussage gemacht, ob ein Zugriff ein Angriff ist oder nicht. In diesem Modus wird das ML-Modell nicht mehr verändert. Dadurch kann im Modus „test“ ein Angreifer nicht durch eine Flut von gezielten Zugriffen das ML-Modell zu den eigenen Gunsten verändern.

In dieser Trainingshypothese wird angenommen, dass ein ML-Modell beim Training mit realen Daten sich laufend verbessert, bis die Zuverlässigkeit des Modells konvergiert. Danach kann die Pipeline im Modus „test“ gestartet werden. Nach einiger Zeit werden jedoch neue Schwachstellen entdeckt. Daher wird mit der Zeit die Zuverlässigkeit des Modells abnehmen, da diese neuen Angriffe unter Umständen noch nicht durch das Modell erkannt werden. Daher sollte die Pipeline periodisch wieder im „train“ Modus gestartet werden, um das Model zu aktualisieren. Bei jeder Aktualisierung muss der Administrator die Zugriffe neu labeln.

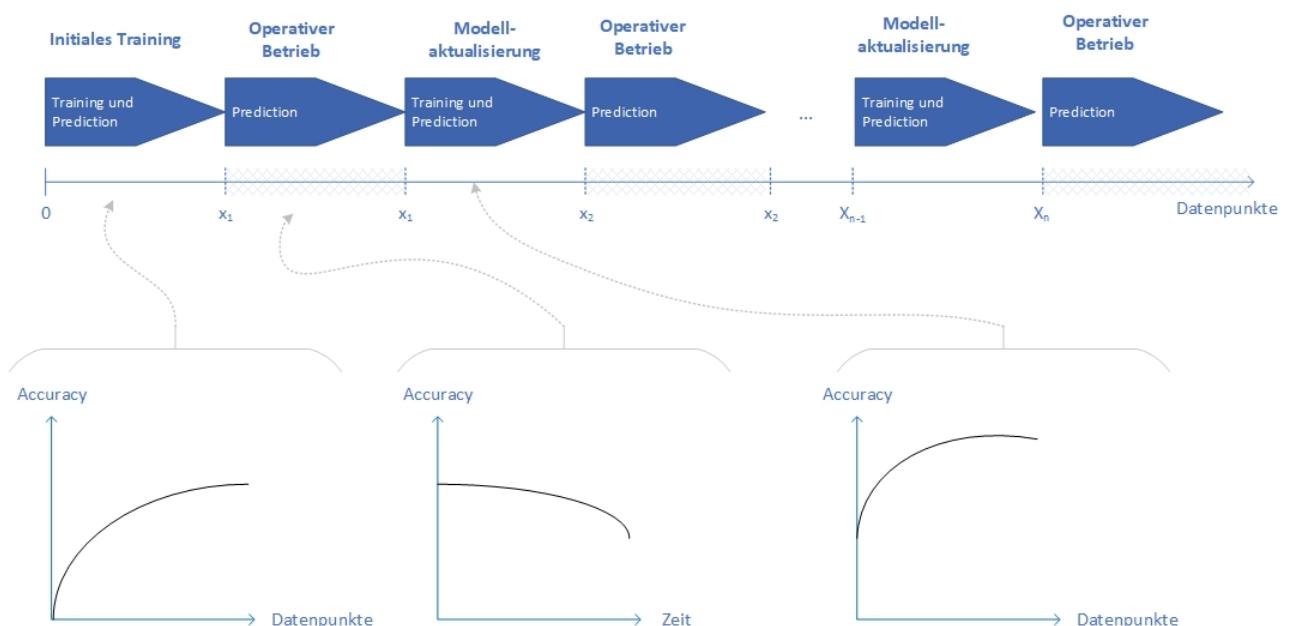


Abbildung 34: Visualisierung der Trainingshypothese

5 Experimente und Ergebnisse

In diesem Kapitel wird die implementierte Software auf die Geschwindigkeit sowie auf die Zuverlässigkeit der Intrusion Detection geprüft und ausgewertet.

5.1 Geschwindigkeit der Pipeline

Um die Geschwindigkeit der implementierten Software zu beurteilen, wurden Zeitmessungen mit dem Python time-Modul in folgender Konfiguration durchgeführt:

- Zu schützender Service: IP Kamera upCam Cyclone HD S+
- Start der Pipeline mit vorerst leerer Datenbank
- Senden von n HTTP POST-Anfragen an die Kamera über das IDS, welches auf demselben Host läuft, von dem auch die Anfrage gesendet wird
- Ermittlung der Durchschnittsverzögerungen der n Anfragen

Die Verzögerung ergibt sich durch Abzug von 10ms der gemessenen Zeit, da ein direkter Zugriff auf die Kamera in dieser Konfiguration ca. 10ms benötigt.

Vorerst wurde das IDS im Trainingsmodus gestartet. Dabei konnten über die ersten 10 Zugriffe im Schnitt eine Verzögerung von 36ms festgestellt werden. Nach 100 Zugriffen konnte bereits eine leicht erhöhte durchschnittliche Verzögerung von 63ms gemessen werden. Nach 1'000 Zugriffen belief sich die Verzögerung wieder auf durchschnittlich 52ms, womit festgestellt werden konnte, dass sich die Verzögerung mit der Anzahl Zugriffe nicht kontinuierlich erhöht. Wenn jedoch die asynchronen Schreibzugriffe auf die Datenbank im Daemon Thread miteinbezogen werden, dauert der Trainingsvorgang deutlich länger. Nach dem Training mit den 1'000 Zugriffen, bei dem 500 als Angriff und 500 als normale Zugriffe gelabelt waren, dauerte es nochmals rund 2 Minuten bis alle Schreibvorgänge auf die Datenbank abgeschlossen waren.

Mit den eingetragenen 1'000 Zugriffen in der Datenbank wurde nun das IDS im Testmodus gestartet. Nach 1'000 gesendeten und geprüften Zugriffen beläuft sich die durchschnittliche Verzögerung auf nur 36ms, wobei keine Schreibzugriffe ausgeführt wurden, da dies im Testmodus nicht nötig ist.

5.2 Zuverlässigkeit der Intrusion Detection

Zur Beurteilung der Zuverlässigkeit, mit welcher Attacken detektiert werden, wurden die Datensätze aus 2.1 und 2.2 mittels der Pipeline, welche in 4 erstellt wurde, analysiert. Dazu wurde folgender Testablauf definiert:

1. Einlesen des Datensatzes
2. Erstellen eines Histogramms der Pfad-Zuverlässigkeit aus der Typisierungskomponente und Festlegen des Alerting-Schwellwertes bei 60%.
3. Analyse des Verhaltens des Typisierungsbaumes im Datensatz. Dabei wird analysiert, wie viele Pfad-Zuverlässigkeits-Alerts pro Anzahl Anfragen generiert werden und ob der Wert wie erwartet konvergiert.
4. Analyse, wie viele Alerts in welcher Komponente generiert werden. Dazu werden 100% bösartige Anfragen aus dem jeweiligen Datensatz gesendet.
5. Erstellen der Lernkurve und der Konfusionsmatrix des logistischen Regressionsmodells eines bestimmten Anfragen-Typs.
6. Evaluation der optimalen Anzahl Cluster und des Scores beim k-Means Modell eines bestimmten Anfragen-Typs.
7. Detektion der Clustergüte und der Leverage Points für ein verbessertes Labeling eines bestimmten Anfragen-Typs.

5.2.1 Seq2Seq Datensatz

Als Erstes wurde die Zuverlässigkeit der Intrusion Detection im Seq2Seq Datensatz [5] betrachtet, welcher auch für das Data-Mining 3 grösstenteils verwendet wurde.

Von der Verteilung der Pfad-Zuverlässigkeiten im [5] Datensatz wurde das untere 60%-Quantil ausgelesen und somit der Schwellwert für die Alarmierung der Zuverlässigkeiten in der Typisierungskomponente auf kleiner gleich 0.2 gesetzt. Wie in der Abbildung 35 links ersichtlich, werden alle Ressourcen in der roten Säule des Histogramms als unzuverlässig gekennzeichnet.

In der Abbildung 35 sind rechts die mit der Anzahl Requests normalisierten Alerts, welche die Typing Stage generiert, ersichtlich. In dieser Abbildung ist gut ersichtlich, dass nach einer gewissen Anzahl Anfragen, die wichtigsten Ressourcen im Baum bekannt sind und dadurch auch die Anzahl Alerts abnimmt und gegen einen bestimmten Wert konvergiert.

Sollte die Anzahl an Scans oder Angriffen auf das System zunehmen, würde sich auch die Anzahl Alerts erhöhen.

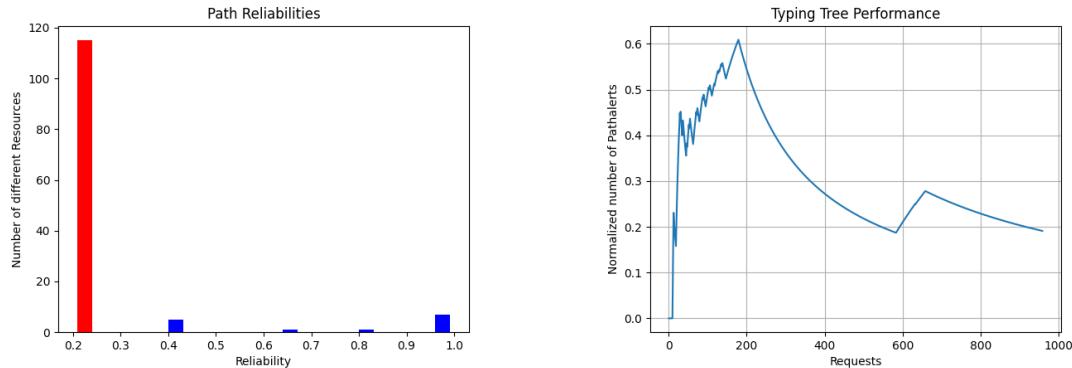


Abbildung 35: Pfad-Alerting der Typisierungskomponente im [5] Datensatz.

Die Abbildung 36 zeigt, welche Komponente welchen Anteil an den vom System generierten Alerts hat. Dabei ist zu erkennen, dass die Filterkomponente den kleinsten Anteil an den Alerts hat. Der Grund dafür ist, dass in diesem Test alle Angriffe von einem Testsystem aus gesendet wurden und die Filterkomponente lediglich eine gewisse Anzahl an Bots und Double-Encoding-Angriffe erkennt. Da das Testsystem nicht auf der Liste der schädlichen Bots war und der Datensatz sehr wenige Double-Encoding-Angriffe enthält, ist der Anteil der Filterkomponente mit lediglich 1.8% der Angriffe eher gering. Die Typisierungskomponente macht dagegen bereits 11.8% der Alerts aus und erkennt Scans und Zugriffe auf nicht existierende Ressourcen zuverlässig. Der grösste Anteil an den Alerts im [5] Datensatz hat die Modellkomponente mit 86.4% der Alerts.

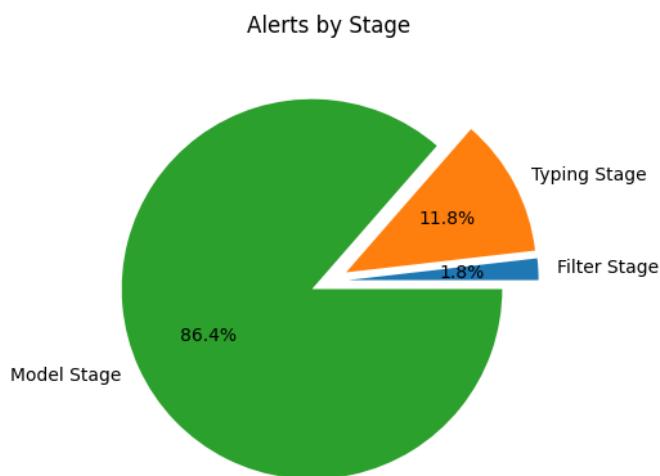


Abbildung 36: Visualisierung der Anzahl Alerts, geordnet nach generierender Stage im [5] Datensatz

Die folgenden Auswertungen wurden auf Zugriffe der HTTP-Methode POST auf die Resource /vulnbank/online/api.php eingeschränkt.

Die nachfolgende Abbildung 37 zeigt die Lernkurve für einen Test-Train Split von 20% zu 80% in den Trainingsdaten mit einer 5-Fold Kreuzvalidierung [22]. Die Lernkurve zeigt, dass durch die Zuteilung der Anfragen in der Typisierungskomponente bereits ab kleinen Datenmengen von weniger als 200 Anfragen sehr präzise Vorhersagen mit dem ML-Modell getroffen werden können, ob es sich um einen Angriff handelt oder nicht.

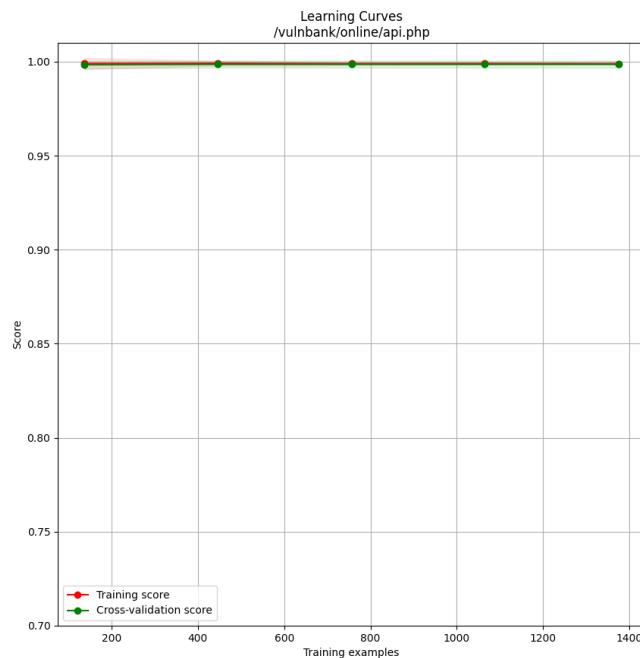


Abbildung 37: Lernkurve für die URL /vulnbank/online/api.php im [5] Datensatz

Diese Ergebnisse konnten mit den Testdaten aus dem Test-Train-Split der Lernkurve in der Konfusionsmatrix in Abbildung 38 verdeutlicht werden:

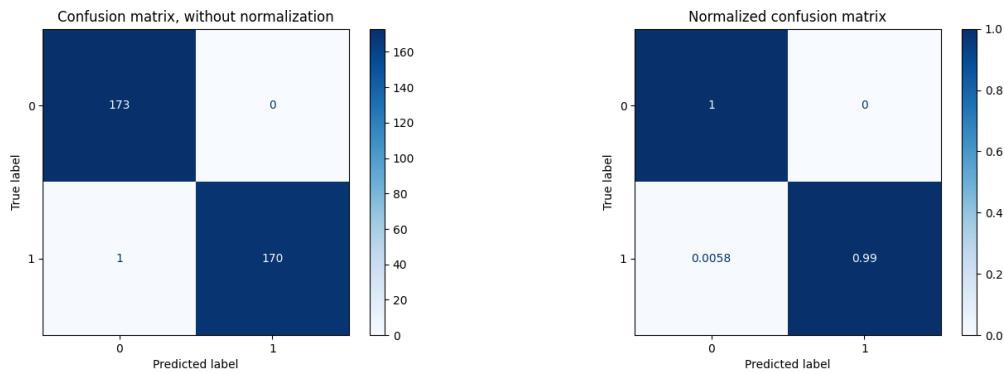


Abbildung 38: Konfusionsmatrix im [5] Datensatz.

Die Konfusionsmatrix kann wie in der Tabelle 7 dargestellt interpretiert werden:

		Vorhersage des Modells	
		Normal	Attacke
Label der Daten	Normal	richtig negativ	falsch positiv
	Attacke	falsch negativ	richtig positiv

Tabelle 7: Interpretation der Konfusionsmatrix

Aus dieser Konfusionsmatrix können die wichtigsten Metriken für die Performance des Machine Learning Modells berechnet werden. Die Metriken basieren auf den vier Werten der Konfusionsmatrix:

- i Richtig negativ r_n : Anfragen, die korrekt als normale Zugriffe klassifiziert wurden.
- ii Falsch positiv f_p : Anfragen, die fälschlicherweise als Angriffe klassifiziert wurden.
- iii Falsch negativ f_n : Anfragen, die fälschlicherweise als normale Zugriffe klassifiziert wurden.
- iv Richtig positiv r_p : Anfragen, die korrekt als Angriffe klassifiziert wurden.

Für die Beurteilung des Modells wurden folgende in [2] beschriebenen Metriken verwendet:

- **Genauigkeit/Precision:** Bezeichnet das Verhältnis der richtig vorhergesagten Angriffe zu allen als Angriffe vorhergesagten Zugriffen:

$$Genauigkeit = \frac{r_p}{r_p + f_p} \quad (4)$$

Im Datensatz [5] beträgt die Genauigkeit $\frac{170}{170+0} = 1.0$.

- **Erkennungsrate/Recall:** Bezeichnet das Verhältnis der korrekt als Angriff klassifizierten Zugriffen und allen Zugriffen, die als Angriffe gelabelt sind:

$$Erkennungsrate = \frac{r_p}{r_p + f_n} \quad (5)$$

Im Datensatz [5] beträgt die Erkennungsrate $\frac{170}{170+1} = 0.9942$.

- **Fehlalarm-Rate/Falsch-positiv-Rate:** Bezeichnet das Verhältnis von falsch vorhergesagten Angriffen zu allen normalen Zugriffen:

$$Fehlalarm - Rate = \frac{f_p}{f_p + r_n} \quad (6)$$

Im Datensatz [5] beträgt die Falsch Alarm Rate $\frac{0}{0+173} = 0.0$.

- **Richtig-negativ-Rate:** Bezeichnet das Verhältnis zwischen den korrekt klassifizierten normalen Zugriffen und der Gesamtheit von allen normalen Zugriffen:

$$Richtig - negativ - Rate = \frac{r_n}{r_n + f_p} \quad (7)$$

Im Datensatz [5] beträgt die Richtig-negativ-Rate $\frac{173}{173+0} = 1.0$.

- **Korrektklassifikationsrate/Accuracy:** Bezeichnet das Verhältnis der richtig klassifizierten Zugriffen zu allen Zugriffen:

$$Korrektklassifikationsrate = \frac{r_p + r_n}{r_p + f_p + r_n + f_n} \quad (8)$$

Im Datensatz [5] beträgt die Korrektklassifikationsrate $\frac{170+173}{170+0+173+1} = 0.9971$.

Als Nächstes wurde die Zuverlässigkeit des k-Means-Modells betrachtet. Dazu wurde als Erstes die optimale Anzahl Cluster im Datensatz ermittelt. Zur Evaluation dieser Daten wurden folgende drei Metriken verwendet:

- In der Abbildung 39 zeigt das Bild links den in [22] beschriebenen „elbow“ Plot. Dabei wird die Summe der quadrierten Fehler als Liniengraph gegen die Anzahl Cluster ausgegeben. Am Punkt an dem der Graph einen Knick hat, liegt ein guter Wert für die Anzahl Cluster in diesem Datensatz.
- Der in Abbildung 39 in der Mitte dargestellte Silhouettenwert beschreibt die Kohäsion innerhalb eines Clusters im Vergleich zur Separation zu anderen Clustern. Der Wert wird gemäss [35] wie folgt berechnet:
 $a(i)$ = mittlere Distanz von i zu allen anderen Punkten im Cluster.
 $b(i)$ = mittlere Distanz von i zu allen Punkten im Nachbarcluster.

$$Silhouette - Score(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \quad (9)$$

Je grösser der $Silhouette - Score(i)$ ist, desto besser lassen sich die Daten in die Cluster trennen.

- In der Abbildung 39 zeigt das Bild rechts die in [42] vorgestellte Gap-Statistik-Methode zur Bestimmung der Anzahl Cluster in einem Datensatz. Bei dieser Methode wird die Streuung W_k innerhalb der Cluster standardisiert, indem der Wert $\log(W_k)$ mit seiner Erwartung unter einer geeigneten Null-Referenzverteilung verglichen wird. Zur Berechnung dieses Wertes wurde die Software-Bibliothek OptimalK verwendet, welche für die Null-Referenzverteilung Zufallswerte über die Ausdehnung des Merkmalsraums verwendet.

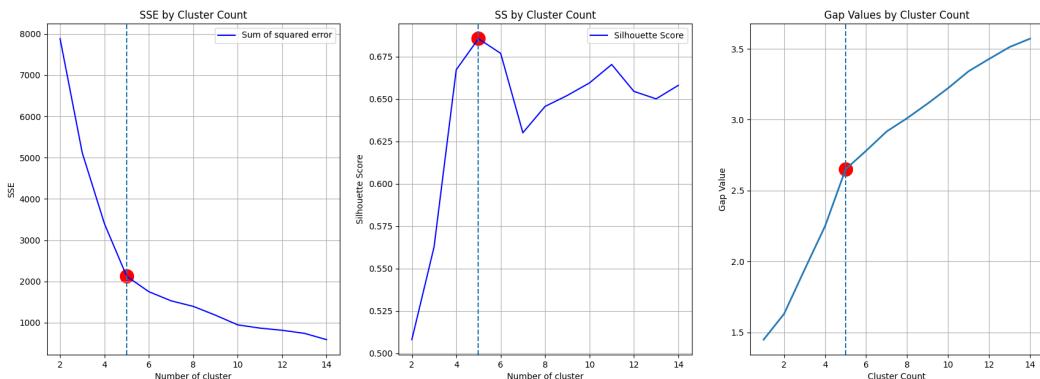


Abbildung 39: Optimale Anzahl Cluster im [5] Datensatz

Alle drei Visualisierungen zeigen ein Optimum bei 5 Clustern. Dieser Wert wurde mit der Gütfunktion, welche in der Abbildung 42 links visualisiert wurde, verglichen. Da in allen Referenzgrafiken der Wert 5 ermittelt werden konnte, wurde dieser Wert für alle weiteren Betrachtungen gewählt.

Als Nächstes wurde eine Konfusionsmatrix für das k-Means-Modell erstellt, welche die normalisierten Werte gegenüber der Anzahl Cluster ausweist. Diese Darstellung konnte mit den vorhandenen Labels im Datensatz erstellt werden. Die Interpretation der Werte ist analog der Tabelle 7 durchzuführen.

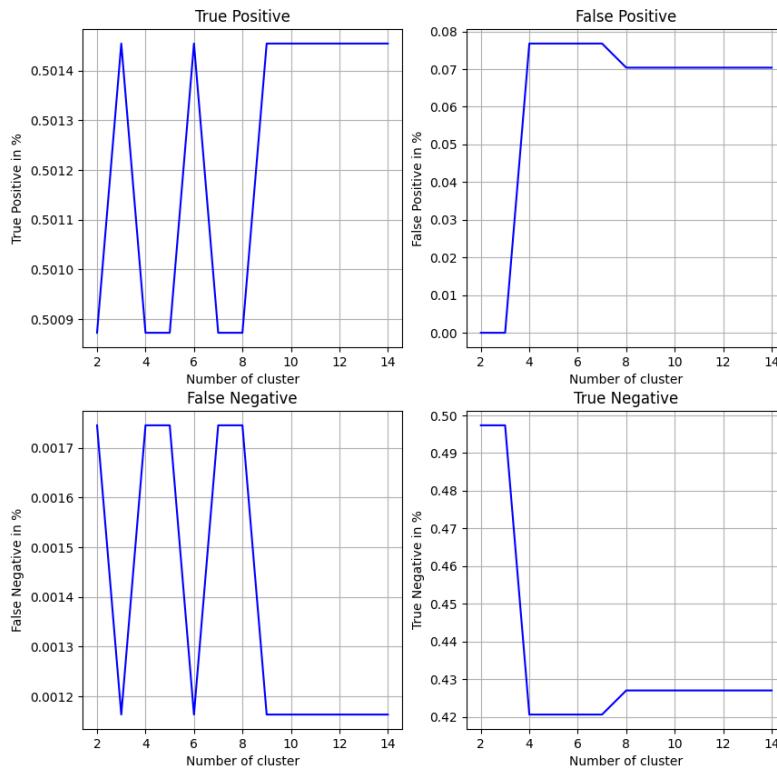


Abbildung 40: Zuverlässigkeit des k-Means-Modells im [5] Datensatz

Die Abbildung 42 links zeigt die Gütfunktion, welche angibt, wie homogen die Daten in den Clustern im Mittel gelabelt sind.

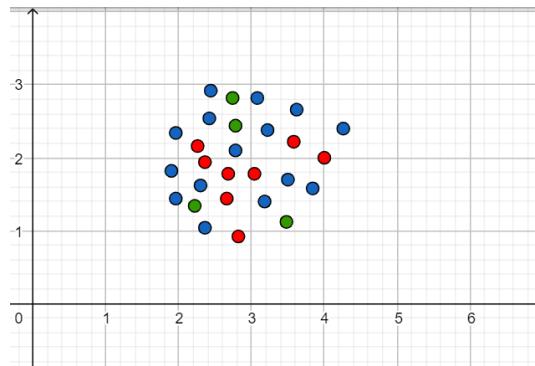


Abbildung 41: Veranschaulichung der Gütefunktion

Zur Berechnung der Homogenität innerhalb eines Clusters wird folgende Formel angewendet:

$$h(a, n, k) = \frac{|a - n|}{a + n + k}, \quad a, n, k \in \mathbb{N} \quad (10)$$

Daraus lässt sich wie folgt die Güte des Clusters ij bestimmen:

$$g_{ij} = h(|A_{ij}|, |N_{ij}|, |K_{ij}|) \quad (11)$$

wobei:

- A_{ij} die Menge der Datenpunkte im Cluster j bei insgesamt i Cluster, die als Angriff gelabelt sind (In Abbildung 41 rot) bezeichnet.
- N_{ij} die Menge der Datenpunkte im Cluster j bei insgesamt i Cluster, die als Normal gelabelt sind (In Abbildung 41 grün) bezeichnet.
- K_{ij} die Menge der Datenpunkte im Cluster j bei insgesamt i Cluster, die über kein Label verfügen (In Abbildung 41 blau) bezeichnet.

Um die Güte zur Bestimmung der Anzahl Cluster als Referenzwert in teilweise gelabelten Datensätzen zu verwenden, kann der Wert g_i für jede Anzahl Cluster $i = 1, \dots, n$ wie folgt berechnet werden:

$$g_i = \frac{1}{i} \cdot \sum_{j=1}^i g_{ij} \quad (12)$$

Die Werte g_i in Abhängigkeit der Anzahl Cluster i wurden in Abbildung 42 links dargestellt. Die optimale Anzahl Cluster i aus Sicht der Homogenität der bereits gelabelten Datenpunkte liegt beim maximalen Wert g_i .

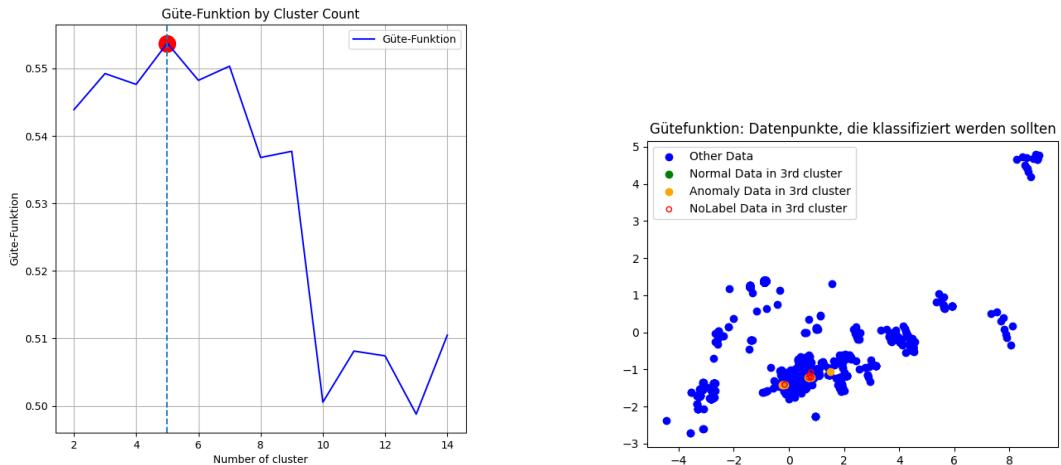


Abbildung 42: Gütefunktion im [5] Datensatz.

Zum Schluss stellte sich die Frage, wie der Aufwand für den Administrator für das Labeling im Trainingsmodus gemäss dem Kapitel 4.8 verringert werden könnte. Dazu wurden per Zufall im Datensatz [5] 60% der Label entfernt und alle Cluster mit der Gütefunktion 11 bewertet. Der Cluster mit der schlechtesten Güte bei $i = 5$ Clustern wurde, wie in Abbildung 42 rechts veranschaulicht, dem Administrator zum Labeln vorgeschlagen. Durch mehrfaches Wiederholen des Vorgangs, kann die Qualität der Referenzdaten für das ML-Modell verbessert werden.

Neben der Anwendung der Gütefunktion zur Verbesserung der Referenzdaten wurde ein ähnlicher Ansatz mit der logistischen Regression verfolgt. Wie in [1] beschrieben, können mittels der logistischen Regression Ausreisser, sogenannte High Leverage Points, erkannt werden. Der Einfachheit halber wurden in dieser Arbeit die High Leverage Points durch den Abstand zur Entscheidungsgrenze (decision boundary) detektiert. Wie in Abbildung 43 links ersichtlich ergibt sich eine charakteristische Verteilung mit zwei Gruppen, den Angriffen und den normalen Zugriffen. Als Schwellwert für die Detektion der High Leverage Points wurde das untere 5%-Quantil und das obere 5%-Quantil verwendet. Danach wurde zur Visualisierung eine PCA auf zwei Komponenten durchgeführt und die High Leverage Points wurden in Abbildung 43 rechts zusammen mit der Entscheidungsgrenze dargestellt.

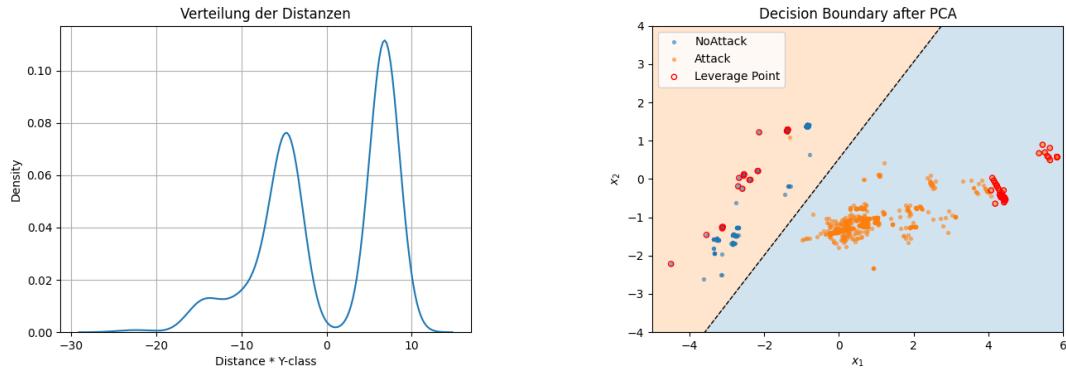


Abbildung 43: Leverage Points im [5] Datensatz.

Der Datensatz [5] wurde in der Publikation [4] untersucht. Bei dieser Untersuchung wurde ein Seq2Seq Autoencoder und nicht ein Pipeline-Aufbau wie in dieser Arbeit verwendet. Die Ergebnisse aus [4] decken sich mit einem Wert von nahezu 0.99 bei der Genauigkeit/Precision und bei der Erkennungsrate/Recall mit diesen.

5.2.2 HTTP DATASET CSIC 2010

Als Nächstes wurde die Zuverlässigkeit der Pipeline mit dem CSIC 2010 HTTP Datensatz [8] analysiert. Dieser Datensatz beinhaltet generierte HTTP-Zugriffe und Angriffe auf eine E-Commerce Plattform. Entgegen dem Seq2Seq-Datensatz sind die Ähnlichkeiten zwischen den Angriffen und den normalen Zugriffen etwas höher und dadurch ist die Schwierigkeit der Intrusion Detection in diesem Datensatz auch entsprechend höher.

Als Erstes wurde auch in diesem Datensatz, wie in Abbildung 44 links dargestellt, die Verteilung der Pfad-Zuverlässigkeiten betrachtet und der Schwellwert beim 60% Quantil gesetzt. Danach wurden die mit der Anzahl Zugriffe normalisierten Alerts in der Abbildung 44 rechts dargestellt. Die Darstellung zeigt entgegen dem Seq2Seq-Datensatz, dass bis etwa zum 2'500-sten Zugriff fast keine Alerts ausgelöst werden. Dies liegt daran, dass die ersten Zugriffe alle auf eine als zuverlässig gekennzeichnete Adresse gesendet wurden. Erst bei ca. 3'300 Zugriffen beginnt der Wert der Alerts wie im Seq2Seq Datensatz zu konvergieren.

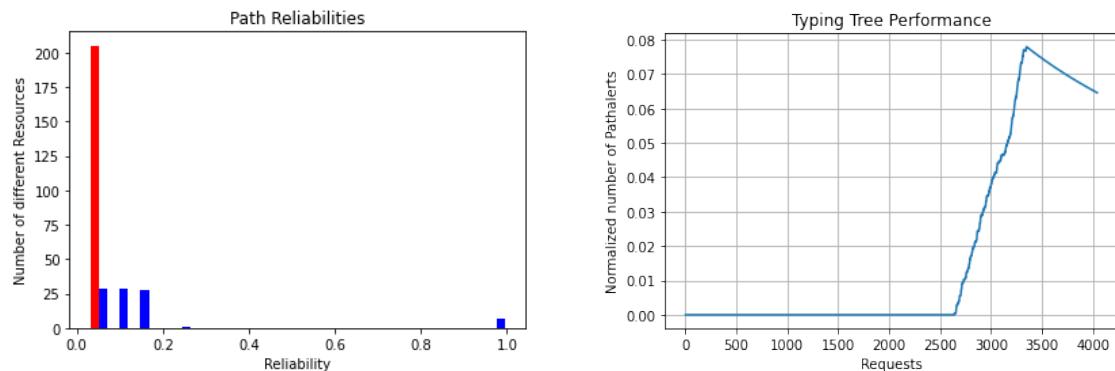


Abbildung 44: Pfad-Alerting der Typisierungskomponente im [8] Datensatz.

Auch die in Abbildung 45 visualisierte Anzahl Alerts nach Komponente, zeigt ein leicht anderes Bild als beim Seq2Seq-Datensatz. Beim HTTP DATASET CSIC 2010 konnten bereits 12.9% der Angriffe mit der Filterkomponente erkannt werden, obwohl auch hier alle Zugriffe vom selben Testrechner zur Pipeline gesendet wurden und daher auch bei diesem Test die Bot-Erkennung in der Filterkomponente deaktiviert war. Dagegen wurden nur 9.1% der Alerts von der Typisierungskomponente generiert. Dies erklärt sich damit, dass die Varianz nach verschiedenen Endpunkten im Datensatz eher gering ist. Wie auch im Seq2Seq-Datensatz konnten mit 61.0% die meisten Angriffe von der Modellkomponente erkannt werden. Leider konnten bei diesem Test 16.9% der im Datensatz als Angriff gekennzeichneten Zugriffe nicht erkannt werden.

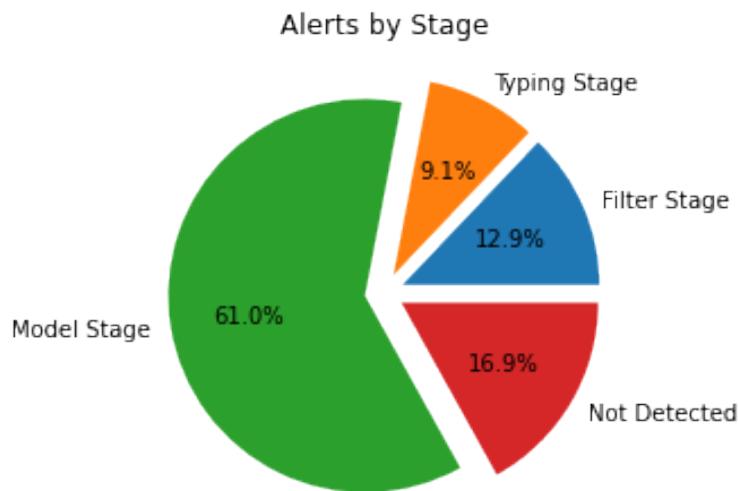


Abbildung 45: Visualisierung der Anzahl Alerts, geordnet nach generierender Komponente im [8] Datensatz

Die Lernkurve zeigt, dass durch die Zuteilung der Zugriffe in der Typisierungskomponente wie im Seq2Seq-Datensatz auch hier bereits ab kleinen Datenmengen von weniger als 100 Zugriffen präzise Vorhersagen mit dem ML-Modell getroffen werden können, ob es sich um einen Angriff handelt oder nicht. Jedoch liegt hier der Score mit 0.85-0.9 bedeutend tiefer als beim Seq2Seq-Datensatz, was sich mit der Komplexität der Angriffe im [8] Datensatz erklären lässt. Auch in diesem Datensatz wurde ein Test-Train-Split von 20% zu 80% mit einer 5-Fold Kreuzvalidierung verwendet.

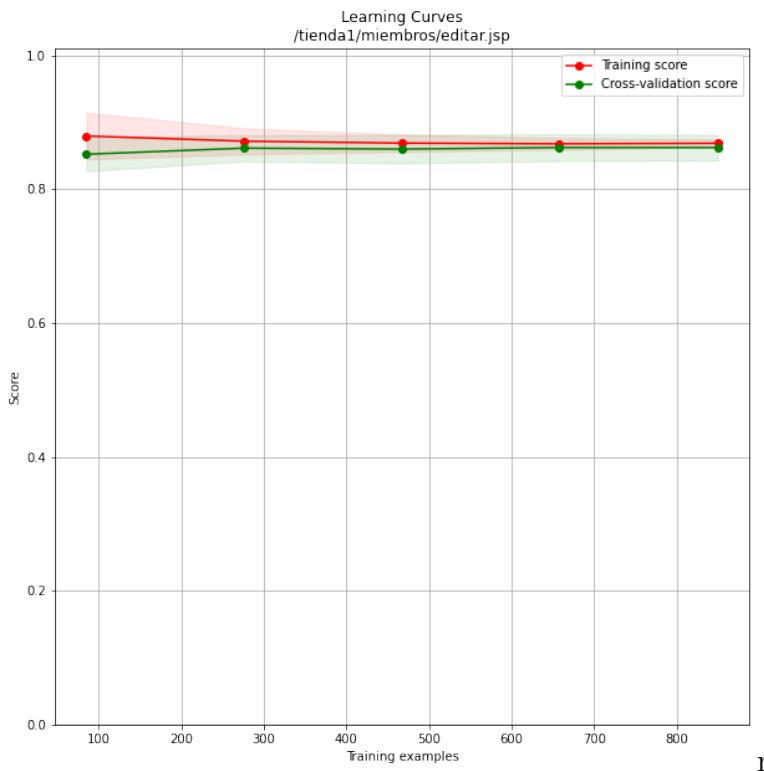


Abbildung 46: Lernkurve für die URL /tienda1/miembros/editar.jsp im [8] Datensatz

Die Konfusionsmatrix in 47 kann gemäss Tabelle 7 interpretiert werden. Die wichtigsten Kenngrössen sind:

- **Genauigkeit/Precision:** Im Datensatz [8] beträgt die Genauigkeit $\frac{52}{52+9} = 0.8525$.
- **Erkennungsrate/Recall:** Im Datensatz [8] beträgt die Erkennungsrate $\frac{52}{52+21} = 0.7123$.
- **Fehlalarm-Rate/Falsch-positiv-Rate:** Im Datensatz [8] beträgt die Fehlalarm-Rate $\frac{9}{9+131} = 0.0643$.
- **Richtig-negativ-Rate:** Im Datensatz [8] beträgt die Richtig- negativ-Rate $\frac{131}{131+9} = 0.9357$.

- **Korrektklassifikationsrate/Accuracy:** Im Datensatz [8] beträgt die Korrektklassifikationsrate $\frac{52+131}{52+9+131+21} = 0.8592$.

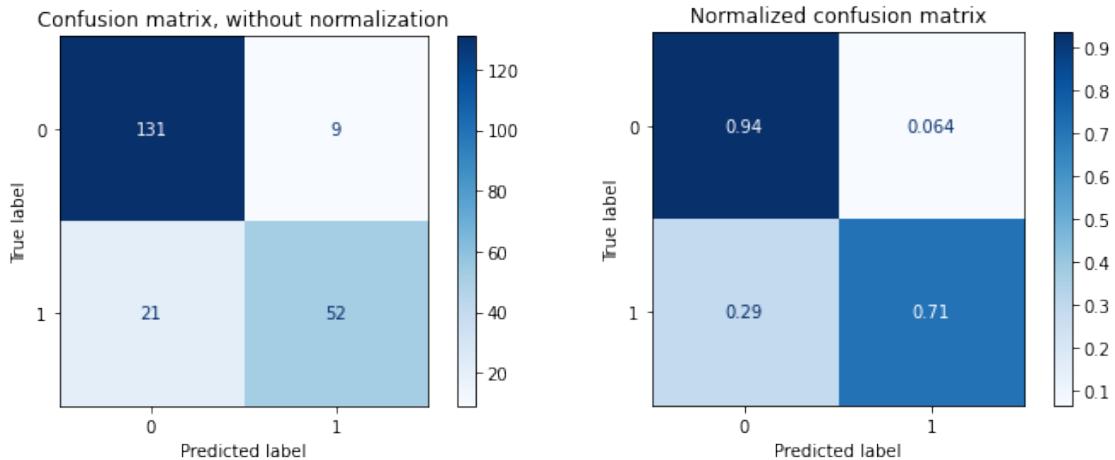


Abbildung 47: Konfusionsmatrix im [8] Datensatz.

Als Nächstes wurde, wie auch im Seq2Seq-Datensatz, die Zuverlässigkeit des k-Means-Modells betrachtet. Dazu wurde als Erstes die optimale Anzahl Cluster im Datensatz ermittelt. Zur Evaluation dieser Daten wurden dieselben drei Metriken wie im Seq2Seq-Datensatz 5.2.1 verwendet:

- In der Abbildung 48 rechts ist der „elbow“ Plot dargestellt der die quadrierte Fehlersumme gegen die Anzahl an Clustern visualisiert. In diesem Datensatz ist der Knick in der Kurve weniger deutlich ersichtlich, als im Seq2Seq-Datensatz, da im Datensatz [8] die Streuung grösser ist. Es lässt sich ein leichter Knick bei $n = 5$ Clustern erkennen.
- In der Abbildung 48 in der Mitte ist der Silhouette-Score gegen die Anzahl Cluster aufgezeichnet. Dieser Plot ist der Aussagekräftigste der Dreien aus Abbildung 48. Daher wurde $n = 6$ auch als optimale Anzahl Cluster gewählt.
- In der Abbildung 48 rechts ist noch die Gap-Statistik gegen die Anzahl Cluster ersichtlich. Wie auch bei der quadrierten Fehlersumme ist das Optimum nicht klar erkennbar. Es ist aber auch hier ein leichter Knick bei $n = 5$ ersichtlich.

Aufgrund der Deutlichkeit des Silhouette-Scores und der Verifikation mit der Gütfunktion aus Abbildung 50 links, welche bei $n = 6$ Cluster zwar kein Maximum aber den zweithöchsten Wert ausweist, wurden sechs Cluster als optimale Clusteranzahl für das Modell angenommen.

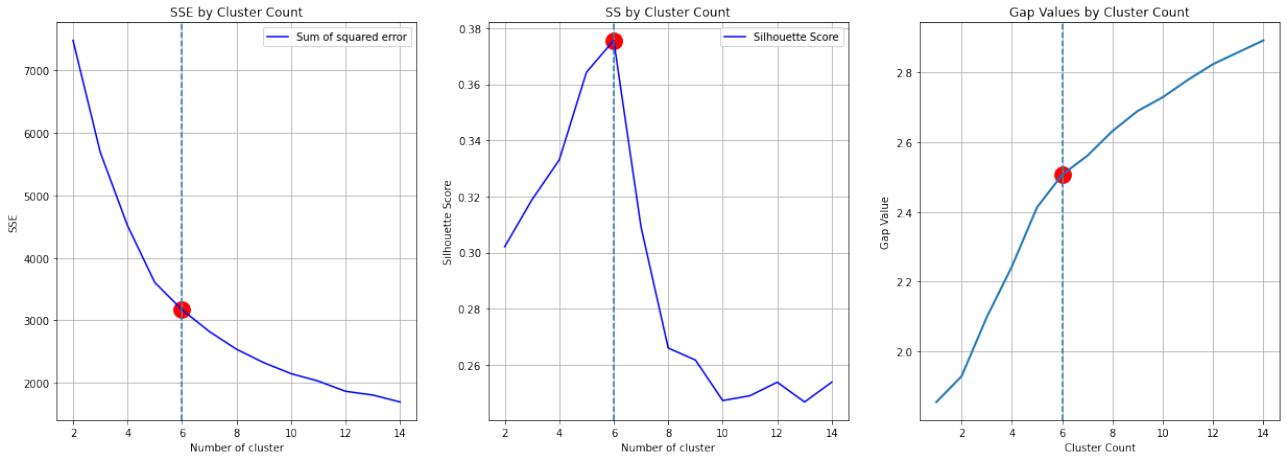


Abbildung 48: Optimale Anzahl Cluster im [8] Datensatz

Danach wurden identisch zum Seq2Seq-Datensatz die Kofusionsplots des k-Means-Modells im Datensatz [8] erstellt. Die Abbildung 49 zeigt, dass das k-Means-Modell etwa 80% der Zugriffe korrekt klassifiziert und sich in etwa 20% der Fälle irrt. Bei $n = 12$ Cluster werden mehr Zugriffe als negativ gekennzeichnet. Da jedoch der falsch negativ und der richtig negativ Wert korrelieren und der richtig positiv und falsch positiv Wert ebenfalls bei $n = 12$ Clustern korrelieren, ist bei dieser Clusteranzahl keine Verbesserung des Modells ersichtlich.

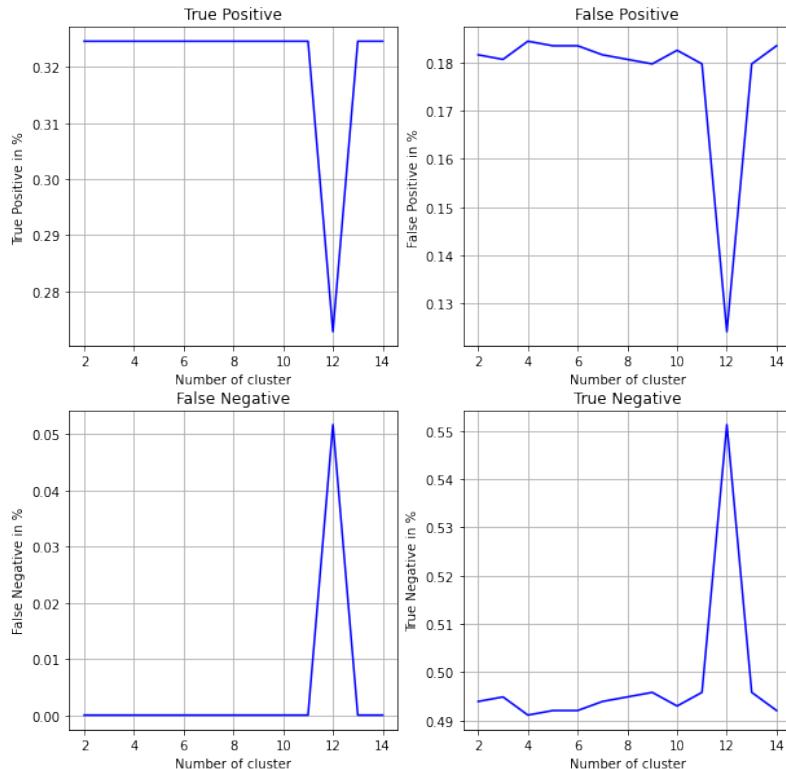


Abbildung 49: Zuverlässigkeit des k-Means-Modells im [8] Datensatz

Die Abbildung 50 links zeigt die Gütefunktion, welche angibt, wie homogen die Daten in den Clustern im Mittel gelabelt sind. Die Interpretation wurde unter Abbildung 41 beschrieben. Die Funktion zeigt ein Optimum bei $n = 7$ Clustern und weist den zweithöchsten Wert bei der gewählten Anzahl von $n = 6$ Clustern auf.

Wie im Seq2Seq-Datensatz wurde auch im CSIC HTTP 2010 Datensatz versucht, den Cluster bei $n = 6$ mit der schlechtesten Güte zum Labeln ausfindig zu machen, um das Training für den Administrator zu erleichtern. Die in Abbildung 50 rechts dargestellten Punkte, sollten gemäss dem in 42 vorgeschlagenen Verfahren, klassifiziert werden:

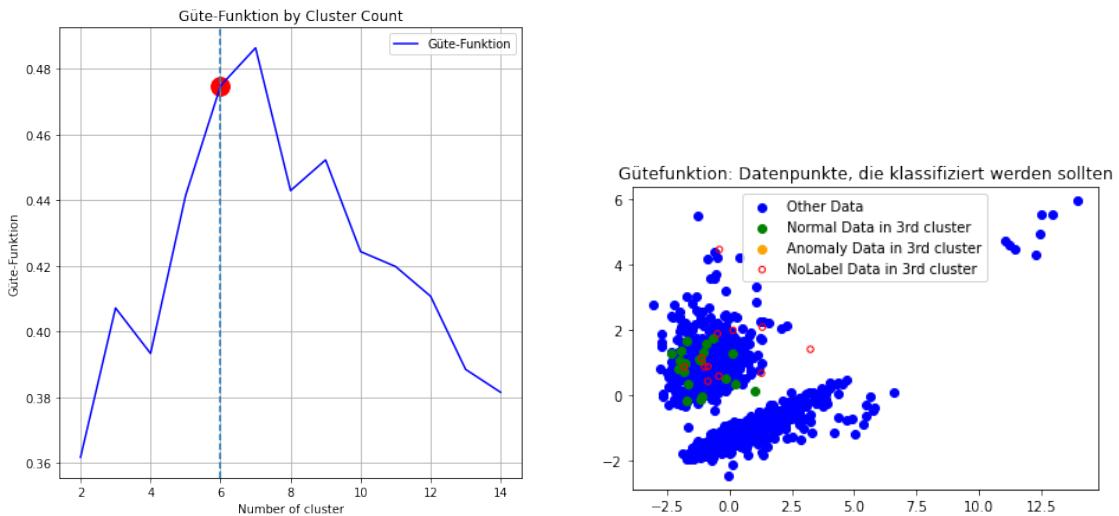


Abbildung 50: Gütefunktion im [8] Datensatz.

Analog zum Seq2Seq-Datensatz wurden auch im Datensatz [8] mit der logistischen Regression die High Leverage Points zum Klassifizieren gesucht. In Abbildung 51 rechts ist die Verteilung der Distanzen zur Entscheidungsgrenze ersichtlich. Es sind auch hier wie im Seq2Seq-Datensatz die beiden Klassen von normalen Zugriffen und Attacken ersichtlich. Wie im Seq2Seq-Datensatz wurde auch hier das obere 5%-Quantil und das untere 5%-Quantil als Grenze für die High Leverage Points gesetzt. Die Punkte wurden in Abbildung 51 rechts sichtbar gemacht. Die Visualisierung zeigt ebenfalls, dass nach einer PCA im CSIC HTTP 2010 Datensatz, die beiden Klassen nicht mehr gleich zuverlässig von der Entscheidungsgrenze getrennt werden können. Dies lässt sich darauf zurückführen, dass im Datensatz [8] auch unbeabsichtigte, unerlaubte Abfragen enthalten sind, die keine böswillige Absicht enthalten. Diese Anfragen haben oft eine hohe Ähnlichkeit zu den normalen Anfragen und sind daher sehr schwer zu erkennen.

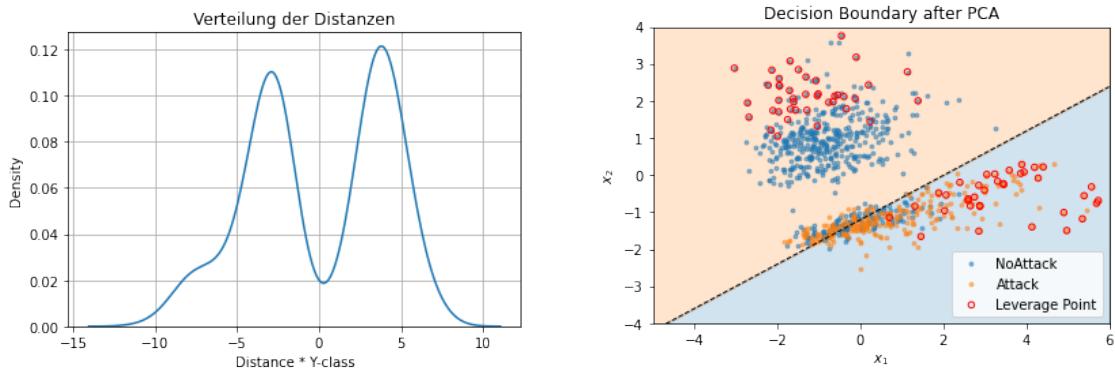


Abbildung 51: High Leverage Points im [8] Datensatz.

Der HTTP CSIC 2010 Datensatz wurde auch bereits im Artikel [18] mittels verschiedenen Machine Learning Methoden untersucht. Die Arbeit weist eine Korrektklassifikationsrate/Accuracy von 90-99% aus, während die eigene Implementierung lediglich knapp 86% erreichte. Betrachtet man jedoch die in [18] gewählten Merkmale etwas genauer, fällt auf, dass eines der gewählten 14 Merkmale die Anzahl Schlüsselwörter im Zugriff sind. Dabei handelt es sich um eine statische Liste von Wörtern, die laufend aktualisiert werden muss. In der Schlussfolgerung von [18] wird darauf hingewiesen, dass erst durch Hinzufügen der charakteristischen Schlüsselwörter für den HTTP CSIC 2010 Datensatz die Korrektklassifikationsrate von über 90% erreicht wurde. Da die hier vorgestellte Implementierung als Plugin-Architektur umgesetzt wurde, könnte in der Extraktionskomponente einfach ein entsprechendes Merkmal hinzugefügt werden, wodurch die Korrektklassifikationsrate dieses Systems dem in [18] vorgeschlagenen Verfahren vermutlich angeglichen werden würde.

5.2.3 Honeypot Datensatz

Als Nächstes wurde die Zuverlässigkeit der Pipeline im Honeypot Datensatz analysiert. Dieser Datensatz beinhaltet echte HTTP-Zugriffe und Angriffe, die mit dem in Kapitel 2.1 beschriebenen System aufgezeichnet wurden. Da mit dem Honeypot überwiegend Scans und fast keine normalen Zugriffe und echte Angriffe aufgezeichnet wurden, wurden dem Datensatz zur Beurteilung der Baum-Performance ca. 14% normale, selbst generierte Zugriffe beigemischt. Für die Analyse der Zuverlässigkeit des ML-Modells wurden ca. 40% normale, selbst generierte Zugriffe und 40% bösartige, selbst generierte Zugriffe auf eine isolierte Ressource zu 20% Scans beigemischt. Bei der isolierten Ressource handelt es sich um den Endpunkt der IP-Kamera, mit dem über die Argumente im Query-Path die Motoren der Kamera gesteuert werden können.

Ein gutartiger Zugriff sieht beispielsweise folgendermassen aus: „<http://146.136.47.202:80/web/cgi-bin/hi3510/ptzctrl.cgi?-step=0&-act=right&-speed=45>“.

Als Erstes wurde wie bei den beiden vorhergegangenen Datensätzen auch in diesem Datensatz, wie in Abbildung 52 links dargestellt, die Verteilung der Pfad-Zuverlässigen betrachtet und der Schwellwert beim 60%-Quantil gesetzt. Danach wurden die mit der Anzahl Zugriffe normalisierten Alerts in der Abbildung 52 rechts dargestellt. Die Darstellung zeigt, dass nach ca. 700 Zugriffen die normalisierte Anzahl Alerts abzunehmen beginnt, jedoch nach ca. 6'800 Requests nochmals ein Peak erreicht wird und der Wert erst bei 8'000 Zugriffen erneut abnimmt. Dies liegt daran, dass die ersten Zugriffe von überwiegend gutartigen Webcrawlern ausgeführt wurden und nur sehr wenige Anomalien bei den Zugriffen festzustellen ist. Nach ca. 6'800 Requests startet ein Bot mit dem Senden von den im Artikel [43] dokumentierten Scans, welche eine grosse Streuung ausweisen und auf eine bösartige Absicht hinweisen. Somit ist das in Abbildung 52 dargestellte Verhalten nachvollziehbar und sinnvoll.

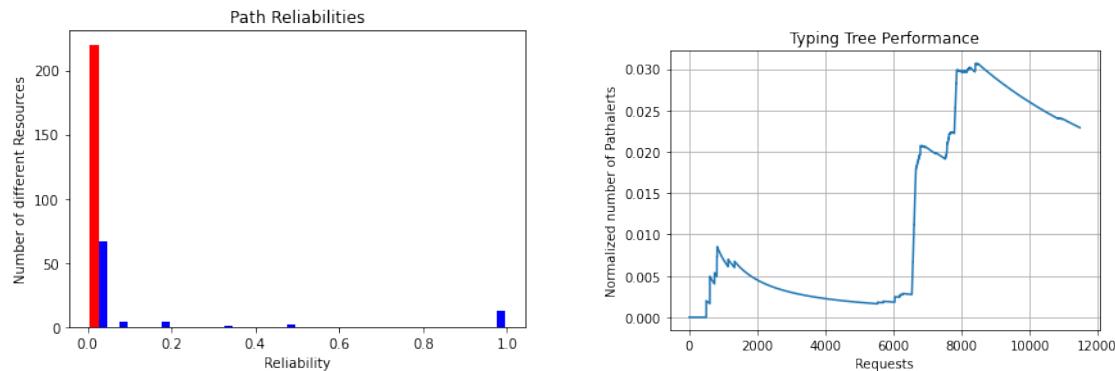


Abbildung 52: Pfad-Alerting der Typisierungskomponente im Honeypot Datensatz.

Die in Abbildung 53 visualisierte Anzahl Alerts nach Komponente zeigt, dass bei 12'542 gesendeten Requests 0.9% der Alerts bereits mit der Filterkomponente erkannt werden. Auch hier war, wie bei den vorhergegangenen Datensätzen, die Bot-Erkennung in der Filterkomponente deaktiviert. Weiter wurden 2.1% der Alerts von der Typisierungskomponente und 96.5% von der Modellkomponente generiert. Bei diesem Test konnten nur 0.4% der Angriffe nicht detektiert werden. Wie beim CSIC HTTP 2010 Datensatz handelt es sich bei den nicht erkannten Zugriffen überwiegend um illegale Anfragen, die eine grosse Ähnlichkeit mit den normalen Zugriffen aufweisen. Der geringe Anteil von Alerts in der Typisierungskomponente lässt sich damit begründen, dass der Anteil an gesendeten, unzuverlässigen Pfaden lediglich 3% betrug, da die Referenzdaten überwiegend gutartige Webcrawler beinhalteten.

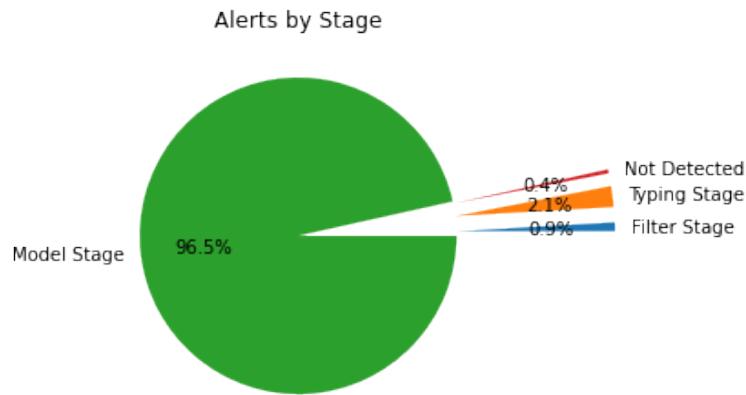


Abbildung 53: Visualisierung der Anzahl Alerts, geordnet nach generierender Komponente im HoneyPot Datensatz

Wie in den beiden vorhergegangenen Datensätzen weist auch hier das ML-Modell bereits ab kleinen Datenmengen von weniger als 100 Zugriffen präzise Vorhersagen, ob es sich um einen Angriff handelt oder nicht, aus. Der Score ist bei dieser Implementierung mit über 95% deutlich über dem CSIC HTTP 2010 Datensatz. Auch in diesem Datensatz wurde ein Test-Train-Split von 20% zu 80% mit einer 5-Fold Kreuzvalidierung verwendet.

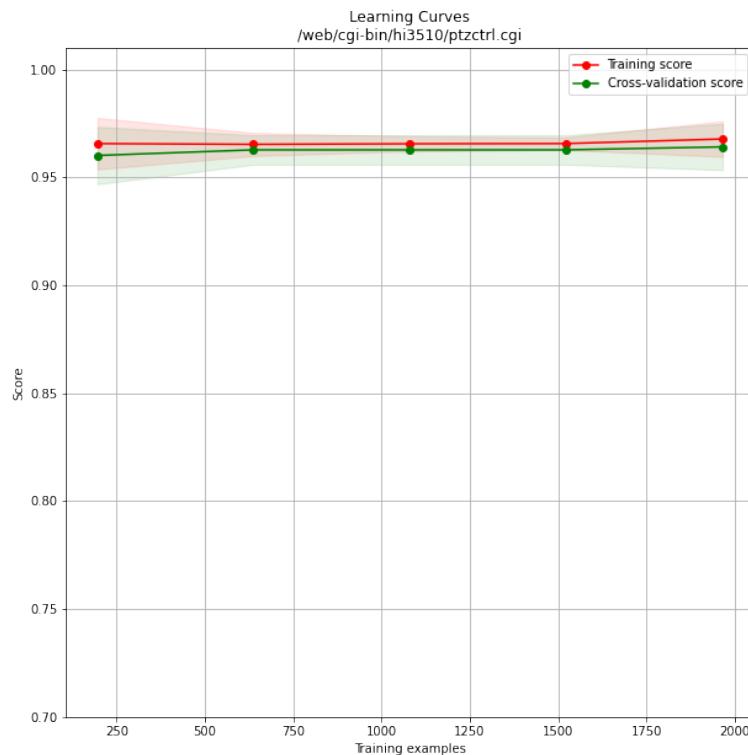


Abbildung 54: Lernkurve für die URL /web/cgi-bin/ptzctrl.cgi im Honeypot Datensatz

Die Konfusionsmatrix in Abbildung 55 kann gemäss der Tabelle 7 interpretiert werden. Die wichtigsten Kenngrössen sind:

- **Genauigkeit/Precision:** Im Honeypot Datensatz beträgt die Genauigkeit $\frac{373}{373+0} = 1.0$.
- **Erkennungsrate/Recall:** Im Honeypot Datensatz beträgt die Erkennungsrate $\frac{373}{373+17} = 0.9564$.
- **Fehlalarm-Rate/Falsch-positiv-Rate:** Im Honeypot Datensatz beträgt die Fehlalarm-Rate $\frac{0}{0+102} = 0.0$.
- **Richtig-negativ-Rate:** Im Honeypot Datensatz beträgt die Richtig- negativ-Rate $\frac{102}{102+0} = 1.0$.
- **Korrektklassifikationsrate/Accuracy:** Im Honeypot Datensatz beträgt die Korrektklassifikationsrate $\frac{373+102}{373+0+102+17} = 0.9654$.

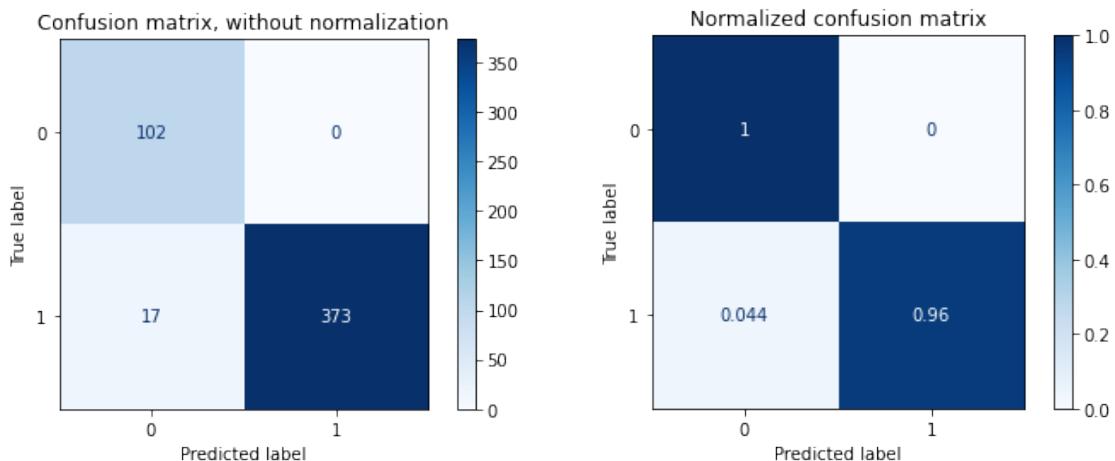


Abbildung 55: Konfusionsmatrix im Honeypot Datensatz

Als Nächstes wurde wie auch mit den vorherigen Datensätzen die Zuverlässigkeit des k-Means-Modells betrachtet. Dazu wurde als Erstes die optimale Anzahl an Clustern im Datensatz ermittelt. Zur Evaluation dieser Daten wurden dieselben drei Metriken wie im Seq2Seq-Datensatz 5.2.1 verwendet:

- In der Abbildung 56 links ist der „elbow“ Plot dargestellt, der die quadrierte Fehlersumme gegen die Anzahl Cluster visualisiert. In diesem Datensatz ist ein Knick in der Kurve bei $n = 7$ Clustern zu erkennen.

- In der Abbildung 56 in der Mitte ist der Silhouette-Score gegen die Anzahl Cluster aufgezeichnet. Der Plot weist bei $n = 8$ ein lokales Maximum auf.
- In der Abbildung 56 rechts ist die Gap-Statistik gegen die Anzahl Cluster ersichtlich. Der Graph weist einen „Knick“ bei $n = 8$ auf.

Aufgrund des lokalen Maximums des Silhouette-Scores, des „Knicks“ bei $n = 8$ Clustern und der Verifikation mit der Gütfunktion aus Abbildung 58 links, welche bei $n = 8$ Clustern einen geeigneten Wert ausweist, wurden acht Cluster als optimale Clusteranzahl für das Modell angenommen.

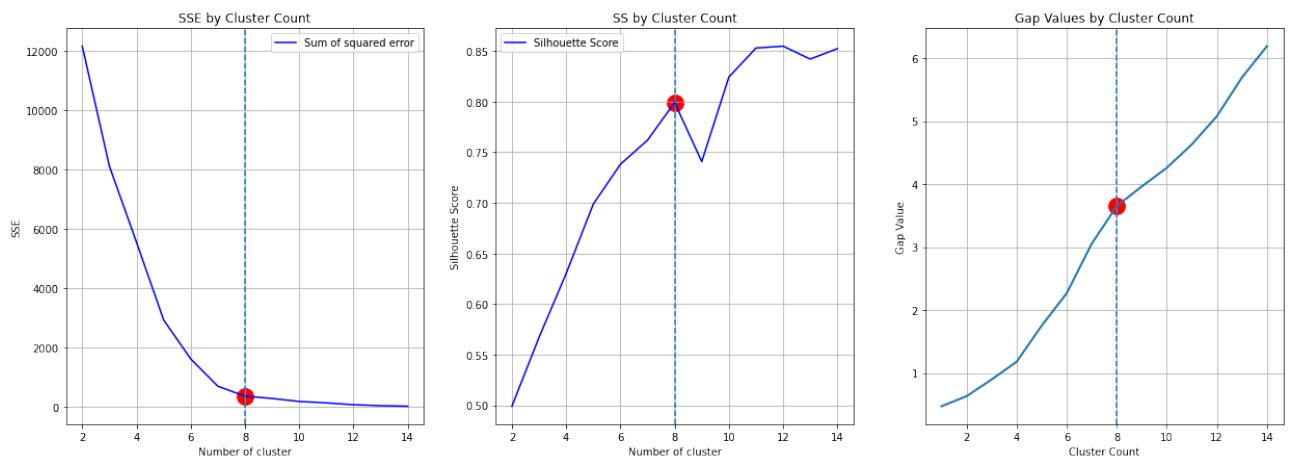


Abbildung 56: Optimale Anzahl Cluster im Honeypot Datensatz

Wie in den beiden anderen Datensätzen, wurde auch in der Abbildung 57 die Konfusionsmatrix des k-Means-Modells gegen die Anzahl Cluster ausgegeben. Jedoch werden mit diesem Modell 100% der Anfragen als Angriffe ausgewiesen, was k-Means für den Honeypot nicht anwendbar macht.

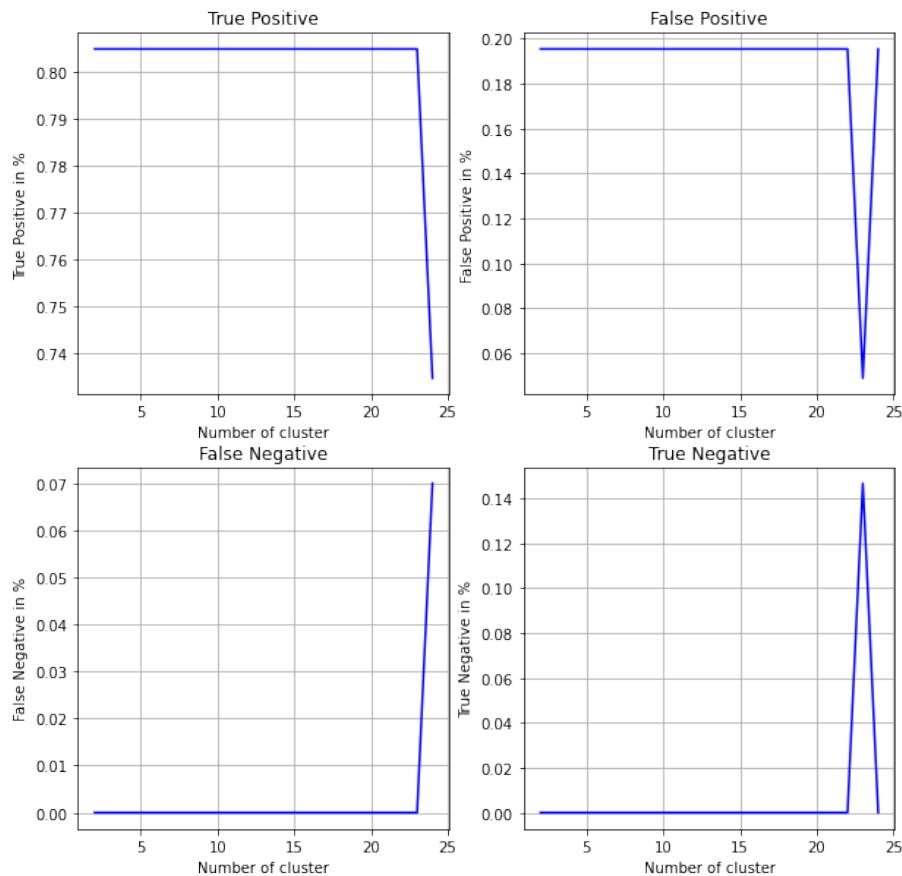


Abbildung 57: Zuverlässigkeit des k-Means-Modells im Honeypot Datensatz

Wie in den beiden anderen Datensätzen wurde auch im Honeypot Datensatz versucht, den Cluster bei $n = 8$ mit der schlechtesten Güte zum Labeln ausfindig zu machen, um das Training für den Administrator zu erleichtern. Die in Abbildung 58 rechts dargestellten Punkte, sollten gemäss dem in 42 vorgeschlagenen Verfahren, klassifiziert werden:

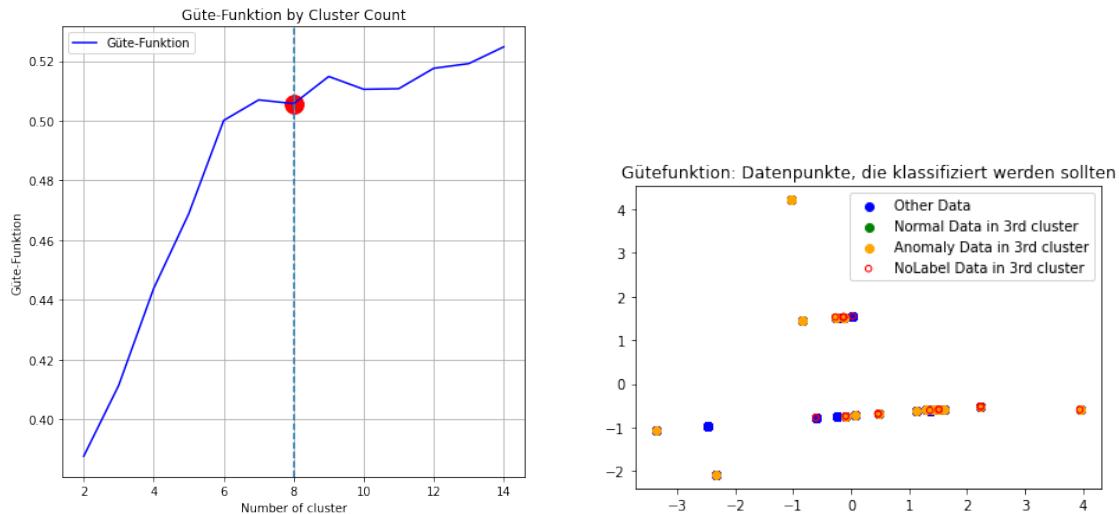


Abbildung 58: Gütfunktion im Honeypot Datensatz.

Die Visualisierung in 58 rechts, sollte mit Vorsicht interpretiert werden, da die Vorhersagewerte der Cluster sehr schlecht sind, was auf eine schlechte Unterstützung beim Labeling hindeuten kann.

Analog zu den anderen Datensätzen wurden auch im Honeypot-Datensatz mit der logistischen Regression die High Leverage Points zum Klassifizieren gesucht. In Abbildung 59 links ist die Verteilung der Distanzen zur Entscheidungsgrenze ersichtlich. Es sind auch hier wie in den beiden anderen Datensätzen die beiden Klassen von normalen Zugriffen und Attacken ersichtlich. Wie bei den anderen Daten wurde auch hier das obere 5%-Quantil und das untere 5%-Quantil als Grenze für die High Leverage Points gesetzt. Die Punkte wurden in Abbildung 59 rechts sichtbar gemacht. Die Visualisierung zeigt, dass nach der PCA die Attacken von den normalen Daten nur noch schwer zu unterscheiden sind, und daher in diesem Datensatz zwingend mehr Dimensionen zur Trennung benötigt werden.

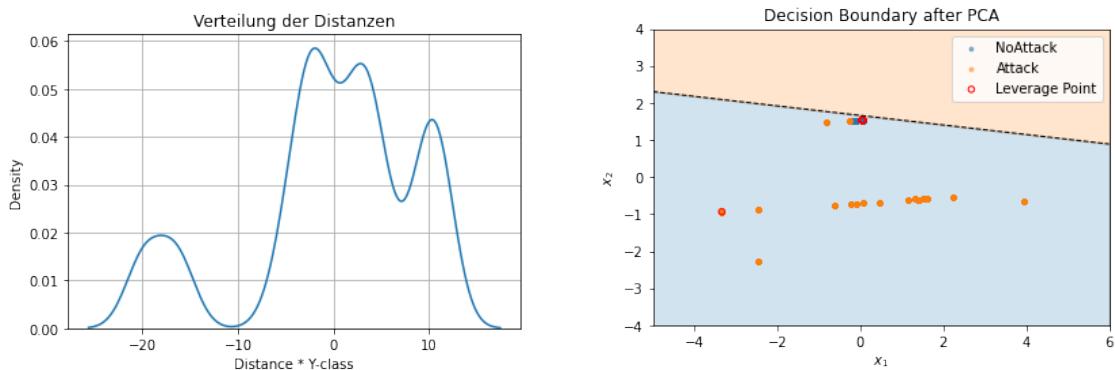


Abbildung 59: High Leverage Points im Honeypot Datensatz.

5.2.4 Diskussion

Die im Kapitel 5.2 analysierten Daten konnten aufzeigen, dass die Intrusion Detection in allen drei Datensätzen sehr gut funktioniert. Die wichtigsten Kenngrößen und Ergebnisse sind in der Tabelle 8 ersichtlich.

	Seq2Seq	CSIC HTTP 2010	Honeypot
Genauigkeit	1.0	0.8525	1.0
Erkennungsrate	0.9942	0.7123	0.9564
Fehlalarm-Rate	0.0	0.0643	0.0
Richtig-negativ-Rate	1.0	0.9357	1.0
Korrektklassifikationsrate	0.9971	0.8592	0.9654

Tabelle 8: Zusammenfassung wichtigster Kenngrößen

Im Seq2Seq-Datensatz entsprachen die Resultate der hier vorgeschlagenen Datenpipeline den Ergebnissen aus vorhergegangenen Arbeiten. Im CSIC HTTP 2010 Datensatz lag die Korrektklassifikationsrate hinter den Ergebnissen aus vergleichbaren Arbeiten. Dieser Umstand konnte auf für den CSIC HTTP 2010 Datensatz zugeschnittene Features in der Referenzarbeit zurückgeführt werden. In den eigenen vom Honeypot aufgezeichneten Daten konnte mittels der logistischen Regression ein sehr gutes Ergebnis erreicht werden. Lediglich der Ansatz des unüberwachten Lernens funktionierte im Honeypot Datensatz nicht. Die normalen Zugriffe konnten nicht gut genug in separate Cluster unterteilt werden. Der k-Means-Algorithmus hat jedoch in gewissen Einsatzgebieten erhebliche Vorteile gegenüber der logistischen Regression. So werden für die logistische Regression Label der Klasse Attacke und Normal benötigt. Es gibt jedoch Ressourcen, die aufgrund ihres statischen Inhaltes für Angreifer kein lohnendes Ziel sind oder noch nicht ausreichend Angriffe aufgezeichnet werden konnten und daher nicht ausreichend Datenpunkte beider Klassen für die logistische Regression vorhanden sind. Für diese Ressourcen bietet

der k-Means-Algorithmus eine ausreichend gute Alternative, da mittels des Clusterings Veränderungen in den Daten festgestellt werden können. Ein weiterer Vorteil bietet der k-Means-Algorithmus beim Unterstützen des Labelings für den Administrator. Da mit der logistischen Regression lediglich die Punkte mit einem ungewöhnlich grossen Abstand zur Entscheidungslinie identifiziert werden, können mit dem k-Means-Verfahren, zusammen mit dem eingeführten Gütemass, auch Veränderungen in Datenstrukturen, die näher bei der Entscheidungslinie liegen, identifiziert werden. Dies kann insbesondere dann von Interesse sein, wenn sich der zu schützende Endpunkt durch ein Update verändert hat und sich dadurch die Merkmalsausprägungen in den HTTP-Anfragen verändert haben. Die Voraussetzung zur Verwendung des k-Mean-Algorithmus zur Erkennung solcher Veränderungen ist jedoch, dass lediglich eine repräsentative Teilmenge mit gleichbleibendem Verhältnis von Datenpunkten mit Label und ungelabelten Datenpunkten analysiert wird, da sonst das Gütemass mit steigender Anzahl normaler Anfragen mit grosser Similarität, die Güte negativ beeinflussen. Die Wahl von solchen repräsentativen Teilmengen wurde bereits in [23] untersucht und ist nicht Bestandteil dieser Arbeit. Diese Teilmenge könnte auch das Wachstum der Datenbank begrenzen, ohne das Machine Learning Modell negativ zu beeinflussen.

Abschliessend lässt sich feststellen, dass mit der vorgeschlagenen Implementierung eine gute Grundlage zur Intrusion Detection geschaffen wurde, die dank dem gewählten modularen Aufbau stetig erweitert und verbessert werden kann.

6 Schlussfolgerungen und Ausblick

Die zunehmende Vernetzung und Komplexität von Computernetzwerken führt zu einer wachsenden Anzahl an Risiken im Bereich der Cybersicherheit. Die Arbeit konnte aufzeigen, dass für eine rechtzeitige Identifikation und Abwehr von Cyber-Angriffen moderne Machine Learning Algorithmen ein gutes Werkzeug zur Verfügung stellen, um solche Anomalien zu erkennen.

Als Hauptherausforderung wurde bereits zu Beginn die Qualität an aussagekräftigen Daten identifiziert. Da das HTTP-Protokoll weitverbreitet und vielfältig einsetzbar ist, stellte sich die Extraktion von aussagekräftigen Merkmalen als ein Schlüsselkriterium heraus. Daher war es wichtig, unterschiedliche Datensätze daraufhin zu untersuchen, ob die extrahierten Merkmale auch wirklich in verschiedenen Praxisszenarien einsetzbar sind oder nicht. Um neben den öffentlich zugänglichen Daten auch noch ein eigenes, sehr praxisnahes Szenario zu untersuchen, wurde der Honeypot erstellt. Rückblickend wäre es besser gewesen, Geräte und Services mit dynamischeren und vielfältigeren Endpunkten einzusetzen und zu untersuchen, da die jetzigen Daten aufgrund der gewählten, relativ simplen Ressourcen mit wenig Angriffsfläche eine kleine Varianz aufweisen. Durch die Bereitstellung von komplexeren Webservices als Endpunkte im Honeypot, hätten neben zahlreichen Bot-Scans auch komplexere Attacken wie Cross-Site-Scripting oder SQL-Injections aufgezeichnet werden können. Somit hätte das Machine Learning Modell mit mehr realen und vielfältigeren Angriffen auf die Endpunkte und weniger mit selbst generierten Angriffen trainiert und getestet werden können.

Eine weitere Schwierigkeit dieser Arbeit war, dass die Merkmalsausprägung für jeden HTTP-Endpunkt sehr unterschiedlich ausfiel. Aus diesem Grund wurde zur Implementierung des Intrusion Detection Systems auch eine Datenpipeline mit unterschiedlichen aufeinanderfolgenden Komponenten gewählt. Dadurch konnten die HTTP-Zugriffe gezielt für jede Ressource einzeln verarbeitet werden, was die Zuverlässigkeit der Vorhersagen verbesserte. Zudem erlaubte der modulare Aufbau der Pipeline mit der Plugin-Architektur es, das System laufend zu erweitern und mit unterschiedlichen Plugins auch unterschiedliche Lösungsansätze zu verfolgen.

Eine weitere Herausforderung waren die Schreibzugriffe auf die Datenbank. Sobald die Anzahl an Zugriffen im Trainingsmodus sehr hoch war, wurden die Schreibvorgänge in die Datenbank sehr langsam. Dieses Problem wurde damit gelöst, dass eine Warteschlange für die Schreibzugriffe auf die Datenbank eingerichtet wurde, welche asynchron von der Datenpipeline in die Datenbank schrieb. Rückblickend wäre es sinnvoller gewesen, eine

Datenbanklösung einzusetzen, die deutlich ressourcenschonender arbeitet und ein sinnvolleres Locking-System mit sich bringt. Des weiteren erfordert der modulare Aufbau mit Plugin-Architektur ebenfalls einen klar definierten Aufbau und eine Namenskonvention in der Datenbank, um bei wachsender Datenmenge und Updates von einzelnen Plugins die Konsistenz der gespeicherten Daten beizuhalten. Dies wurde im aktuellen Aufbau nicht ausreichend berücksichtigt.

Da die Arbeit in den drei Themen IT-Sicherheit, maschinelles Lernen und Software Engineering angesiedelt war, konnte auch in allen drei Themen das Wissen stetig ausgebaut werden. Im Bereich IT-Sicherheit wurde gelernt, wie trickreich Angreifer vorgehen können und mit welchen grossen Herausforderungen dadurch die Sicherheitsverantwortlichen zu kämpfen haben. Im Bereich des maschinellen Lernens konnte selbst erfahren werden, wie wichtig aussagekräftige Daten und gute Merkmale sind, um Daten erfolgreich zu klassifizieren. Im Fachbereich Software Engineering wurde sehr schnell realisiert, dass ein logischer und durchdachter Softwareaufbau und die ständige Erweiterbarkeit und Adapтивierbarkeit der Software die Arbeit der Entwickler erheblich erleichtert.

In einem weiterführenden Forschungsprojekt könnten anhand der bestehenden Software-Implementierung weitere Plugins entwickelt werden, um die Qualität der Vorhersagen zu verbessern. So könnte beispielsweise in der Extraktionskomponente ein Autoencoder eingesetzt werden, um vom HTTP-Body weitere Merkmale zu gewinnen. Zudem könnten zusätzliche Merkmale vom HTTP-Header gewonnen werden.

Ein weiteres Themengebiet für eine vertiefende Forschungsarbeit stellt die Wahl einer aussagekräftigen Teilmenge an Trainingsdaten für das ML-Modell dar. In der vorliegenden Arbeit wird die Anzahl an Trainingsdaten stetig erhöht, was in der Praxis nicht sinnvoll ist.

Die Arbeit konnte aufzeigen, wie mittels High Leverage Points und Clustern die Qualität der Trainingsdaten verbessert werden kann. Eine weitere Arbeit könnte untersuchen, wie die Anzahl an Trainingsdaten effektiv reduziert werden könnte.

Literatur

- [1] Abdul Awal Md Nurunnabi, A. H. M. Rahmatullah Imon, M. Nasser. "Identification of multiple influential observations in logistic regression". In: (2010). URL: https://www.researchgate.net/publication/258141262_Identification_of_multiple_influential_observations_in_logistic_regression.
- [2] Zeeshan Ahmad u. a. "Network intrusion detection system: A systematic study of machine learning and deep learning approaches". In: *Transactions on Emerging Telecommunications Technologies* 32.1 (2021). ISSN: 2161-3915. DOI: 10.1002/ett.4150.
- [3] Jonathan Andersson, Josiah Hagen und Brandon Niemczyk. "Intrusion prevention system with machine learning model for real-time inspection of network traffic". Pat. US11128664 (B1). 2021.
- [4] Arseny Reutov, Fedor Sakharov, Irina Stepanyuk, Alexandra Murzina. *Detecting Web Attacks with a Seq2Seq Autoencoder*. 2019. URL: <https://habr.com/en/company/pt/blog/441030/>.
- [5] Arseny Reutov, Fedor Sakharov, Irina Stepanyuk, Alexandra Murzina. *Seq2Seq for Web Attack Detection*. URL: <https://github.com/jasonng17/seq2seq-web-attack-detection>.
- [6] Joachim Behnke. *Logistische Regressionsanalyse: Eine Einführung*. 1. Aufl. Methoden der Politikwissenschaft. s.l.: Springer VS, 2014. ISBN: 978-3-658-05081-8. URL: <https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=1967370>.
- [7] Carlos Polop. *554,8554 - Pentesting RTSP*. 2022. URL: <https://book.hacktricks.xyz/network-services-pentesting/554-8554-pentesting-rtsp>.
- [8] Carmen Torrano Giménez, Alejandro Pérez Villegas, Gonzalo Álvarez Marañón. *HTTP DATASET CSIC 2010*. 2010. URL: <https://www.isi.csic.es/dataset/>.
- [9] Chris Brown. *How do I avoid double URL encoding when rendering URLs in my website?* 2013. URL: <https://stackoverflow.com/questions/16094265/how-do-i-avoid-double-url-encoding-when-rendering-urls-in-my-website>.
- [10] Cisco. *Snort*. 2022. URL: <https://www.snort.org/>.
- [11] Danila Vershinin. *Do these .env GET requests from localhost indicate an attack? [closed]*. 2021. URL: <https://stackoverflow.com/questions/64109005/do-these-env-get-requests-from-localhost-indicate-an-attack>.
- [12] Yakov Shafranovich EdOverflow. 2022. URL: <https://securitytxt.org/>.

-
- [13] Martin Fowler. *Patterns of Enterprise Application Architecture*. Harlow: Pearson Education, 2002. ISBN: 0-321-12742-0. URL: <https://elibrary.pearson.de/book/99.150005/9780133065206>.
 - [14] Erich Gamma u. a. *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley professional computing series. Reading, Mass.: Addison-Wesley, 1995. ISBN: 0-201-63361-2.
 - [15] geoPlugin. *geoPlugin*. 2022. URL: <https://www.geoplugin.com/>.
 - [16] Google Developers. *Allgemeine Informationen zu Sitemaps*. 2022. URL: <https://developers.google.com/search/docs/advanced/sitemaps/overview?hl=de>.
 - [17] GovCERT.ch. *Zero-Day Exploit Targeting Popular Java Library Log4j*. 2021. URL: <https://govcert.ch/blog/zero-day-exploit-targeting-popular-java-library-log4j/>.
 - [18] Ayush Gupta und Avani Modak. "Anomaly Detection in HTTP Requests Using Machine Learning". In: *Machine Learning and Information Processing*. Hrsg. von Debabala Swain, Prasant Kumar Pattnaik und Tushar Athawale. Bd. 1311. Advances in Intelligent Systems and Computing. Singapore: Springer Singapore, 2021, S. 445–455. ISBN: 978-981-33-4858-5. DOI: 10.1007/978-981-33-4859-2{\textunderscore}44.
 - [19] Ian London. *Encoding cyclical continuous features - 24-hour time*. IAN LONDON'S BLOG, 2022. URL: <https://ianlondon.github.io/blog/encoding-cyclical-features-24hour-time/>.
 - [20] IANA. *Hypertext Transfer Protocol (HTTP) Method Registry*. 2014. URL: <https://www.iana.org/assignments/http-methods/http-methods.xhtml>.
 - [21] iSpyConnect.com. *Kamtron IP camera URL: Connecting to your Kamtron IP camera**. 2022. URL: <https://www.ispyconnect.com/camera/kamtron>.
 - [22] Gareth James u. a. *An Introduction to Statistical Learning: with Applications in R (Springer Texts in Statistics)*. 2013 Edition. Springer, 2013. ISBN: 978-1-4614-7137-0.
 - [23] Javier Tejada, Mikhail Alexandrov, Gabriella Skitalinskaya, Dmitry Stefanovskiy. *Selection of Statistically Representative Subset from a Large Data Set*. 2017. URL: https://link.springer.com/content/pdf/10.1007/978-3-319-52277-7_58.pdf.

-
- [24] Kenneth LeRoy Ingham III. "Anomaly Detection for HTTP Intrusion Detection: Algorithm Comparisons and the Effect of Generalization on Accuracy: DISSERTATION". In: (). URL: <http://agl.cs.unm.edu/~forrest/dissertations/kenneth-ingham-dissertation.pdf>.
 - [25] Lorenzo Comi. *IoT-SecurityChecker*. 2018. URL: <https://github.com/c0mix/IoT-SecurityChecker>.
 - [26] M. Koster, Stalworthy Computing, G. Illyes, H. Zeller, L. Sassman, Google. *Robots Exclusion Protocol*. 2022. URL: <https://datatracker.ietf.org/doc/html/draft-koster-rep>.
 - [27] Mark Taylor. *Skeleton of a site attack*. 2022. URL: <https://marktaylor.synology.me/?p=1021>.
 - [28] Martin Fowler. *InversionOfControl*. martinfowler.com, 2005. URL: <https://www.martinfowler.com/bliki/InversionOfControl.html>.
 - [29] Christoph Landolt Moritz Bättig. "Aufbau eines ML-basierten Intrusion Detection Systems (IDS): Fachmodul". Diss. Eastern Switzerland University of Applied Sciences, 2022.
 - [30] Annalyn Ng und Kenneth Soo. *Data Science - was ist das eigentlich?! Algorithmen des maschinellen Lernens verständlich erklärt*. Berlin und Heidelberg: Springer, 2018. ISBN: 978-3-662-56775-3.
 - [31] Noah Dietrich, Colin Grady, Michael Shirk. *PulledPork3*. 2021. URL: <https://github.com/shirkdog/pulledpork3>.
 - [32] *ONVIF Technical Specification Development*. GitHub, 2022. URL: <https://github.com/onvif/specs/>.
 - [33] owasp. *Double Encoding*. 2022. URL: <https://owasp.org/www-community/Double-Encoding>.
 - [34] R. Padmashani, M. Nivaashini und R. Vidhyapriya. "RAkEL Algorithm and Mahalanobis Distance-Based Intrusion Detection System Against Network Intrusions". In: *Proceedings of International Conference on Artificial Intelligence, Smart Grid and Smart City Applications*. Hrsg. von L. Ashok Kumar, L. S. Jayashree und R. Manimegalai. Cham: Springer International Publishing, 2020, S. 689–696. ISBN: 978-3-030-24050-9. DOI: [10.1007/978-3-030-24051-6_63](https://doi.org/10.1007/978-3-030-24051-6_63).
 - [35] Peter J.Rousseeuw. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: (1986). URL: [https://doi.org/10.1016/0377-0427\(87\)90125-7](https://doi.org/10.1016/0377-0427(87)90125-7).

-
- [36] PhD. Kushe R. "COMPARATIVE STUDY OF VULNERABILITY SCANNING TOOLS: NESSUS vs RETINA". In: (2017). URL: <https://stumejournals.com/journals/confsec/2017/2/69.full.pdf>.
 - [37] Pierre-Louis Bescond. *Cyclical features encoding, it's about time!* Towards Data Science, 2020. URL: <https://towardsdatascience.com/cyclical-features-encoding-its-about-time-ce23581845ca>.
 - [38] Python Software Foundation. *dataclasses — Data Classes*. 2022. URL: <https://docs.python.org/3/library/dataclasses.html>.
 - [39] R. Fielding, Ed., Adobe, J. Reschke, Ed., greenbytes. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content: Request for Comments: 7231*. 2014. URL: <https://datatracker.ietf.org/doc/html/rfc7231>.
 - [40] R. Fielding, UC Irvine, J. Gettys, Compaq/W3C, J. Mogul, Compaq, H. Frystyk, W3C/MIT, L. Masinter, Xerox, P. Leach, Microsoft, T. Berners-Lee, W3C/MIT. *Hypertext Transfer Protocol – HTTP/1.1: Request for Comments: 2616*. 1999. URL: <https://datatracker.ietf.org/doc/html/rfc2616#ref-47>.
 - [41] RE-Solver. *Linksys RE6500 1.0.11.001 - Unauthenticated RCE*. 2020. URL: <https://www.exploit-db.com/exploits/49270>.
 - [42] Robert Tibshirani, Guenther Walther, Trevor Hastie. "Estimating the Number of Clusters in a Data Set Via the Gap Statistic". In: (2001). URL: <https://hastie.su.domains/Papers/gap.pdf>.
 - [43] Sagar Makwana. *Internet Bot Attack Test Experiment*. 2020. URL: <https://medium.com/@sagarmakwana093/internet-bot-attack-test-experiment-e2caacd1321e>.
 - [44] Sara Althubiti, Xiaohong Yuan, Albert Esterline. "Analyzing HTTP requests for web intrusion detection". In: (). URL: <https://digitalcommons.kennesaw.edu/cgi/viewcontent.cgi?referer=https://www.google.com/&httpsredir=1&article=1053&context=ccerp>.
 - [45] Stephan Spitz, Michael Pramateftakis und Joachim Swoboda. *Kryptographie und IT-Sicherheit: Grundlagen und Anwendungen*. 2., überarb. Aufl. Studium. Wiesbaden: Vieweg + Teubner, 2011. ISBN: 978-3-8348-1487-6.
 - [46] Stefan Lucks. *Kryptographie und Mediensicherheit (2017): Passwörter und Entropie*. 2017. URL: <https://www.uni-weimar.de/fileadmin/user/fak/medien/professuren/Mediensicherheit/Teaching/SS17/Krypto/krypto03.pdf>.

- [47] T. Berners-Lee, W3C/MIT, R. Fielding, Day Software, L. Masinter, Adobe Systems, January 2005. *Uniform Resource Identifier (URI): Generic Syntax: Request for Comments: 3986*. 2005. URL: <https://datatracker.ietf.org/doc/html/rfc3986>.
- [48] w3schools. *HTML URL Encoding Reference*. 2022. URL: https://www.w3schools.com/tags/ref_urlencode.asp.
- [49] ZODB - a native object database for Python. 2022. URL: <https://zodb.org/en/latest/index.html> (besucht am 22.07.2022).

Anhang

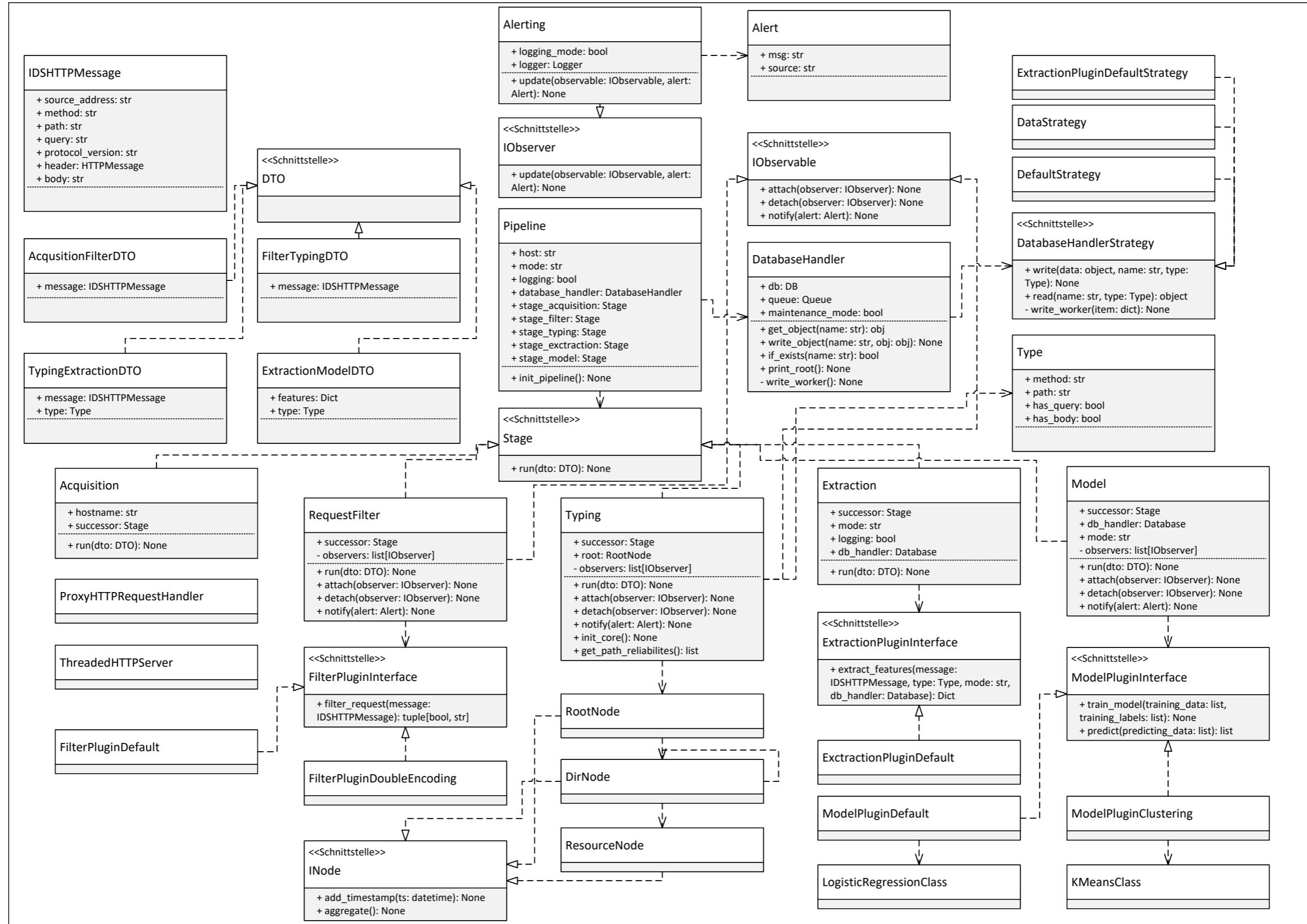
A Programmcode

Die implementierte Software befindet sich auf folgendem GitHub-Repository:

<https://github.com/moritzbaettig-ost/bachelorarbeit-pipeline/>



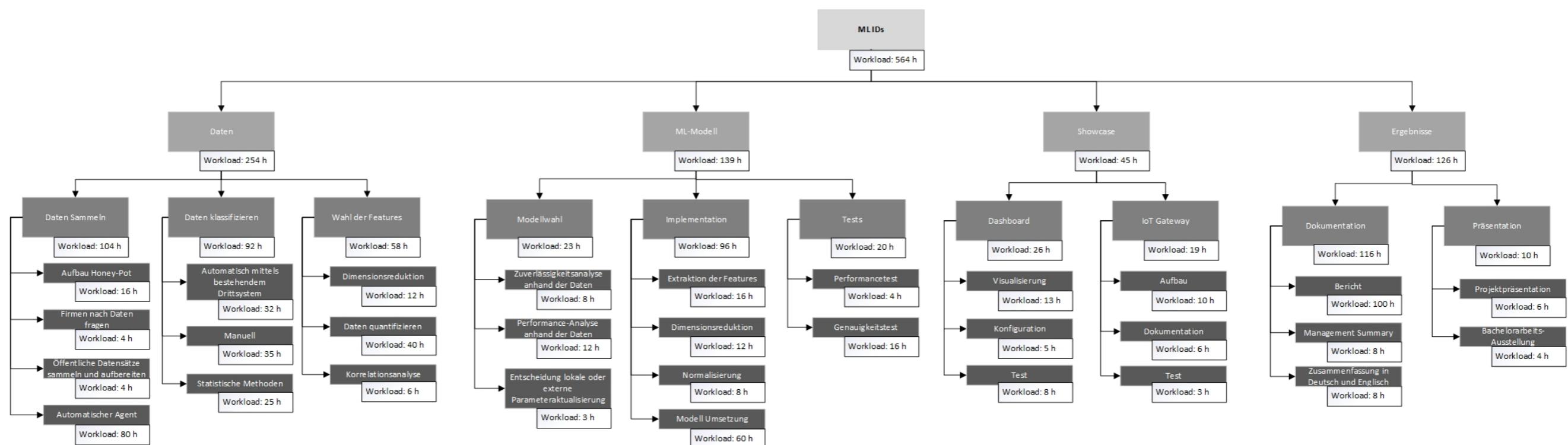
B Klassendiagramm



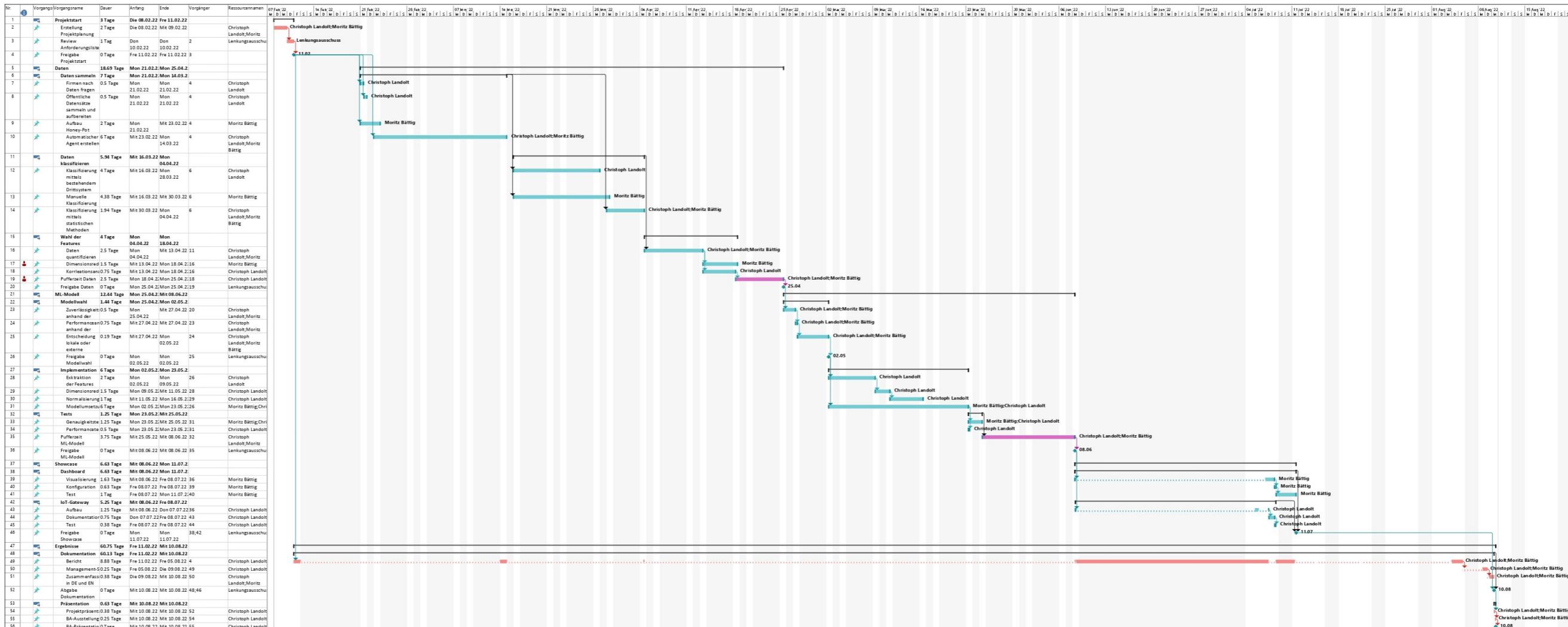
C Risikoplan

Risk List													
Rev 0.1		Project Name		Aufbau eines ML-basierten Intrusion Detection Systems (IDS)				Total Risk Exposure Sum:		20.8			
		Project Manager		Christoph Landolt / Moritz Bättig				Top 10 Risk Exposure Average:		2.0			
Risk Analysis													Action Plan
#	Risk description	Impact on project (cost, time, feature, quality, profitability)	Category Optional field for grouping of risks.	Prob. 0...100%	Impact 1...10	Risk Exposure product of probability and impact	Mitigation Steps Actions to reduce risk - these actions are included in project plan.	Contingency Plan What will be done if the mitigation steps fail (agreed with product management).	Resp. Person (only 1)	Status	Date by when the risk will be reassessed	Risk History	
1	nicht genügend qualitativ hochwertige Daten	quality	Daten	50%	8	4.00	Firmen aktiv angehen und nach Daten fragen, Aufbau eines Honey-Pots, Konzept eines Angriffs-Agenten im Bereich des RL ausarbeiten.	Honey-Pot-Daten manuell klassifizieren, Daten mittels Angriffsagenten erzeugen, vorhandene öffentlich zugängliche Datensätze verwenden.	Christoph Landolt	Open	14-Mar-22		
6	zu hohe Falsch-Negativ-Rate in Produktion	quality	Modell	50%	6	3.00	Verschiedenste Modellansätze werden anhand der vorhandenen Daten evaluiert. Die Performanz der Modelle werden dokumentiert.	Anpassung des Modells anhand der Modellwahl.	Christoph Landolt	Open	25-May-22		
5	zu hohe Falsch-Positiv-Rate in Produktion	quality	Modell	50%	6	3.00	Verschiedenste Modellansätze werden anhand der vorhandenen Daten evaluiert. Die Performanz der Modelle werden dokumentiert.	Anpassung des Modells anhand der Modellwahl.	Christoph Landolt	Open	25-May-22		
8	zu hoher Speicherbedarf auf Endgerät	quality	Modell	50%	5	2.50	Ressourcenschonender Aufbau der Pipeline, weitere Dimensionsreduktion	Ein Teil der Daten wird auf Cloud ausgelagert.	Christoph Landolt	Open	23-May-22		
4	zu hohe Performanceauslastung durch ML-Algorithmen	quality	Modell	50%	5	2.50	Ressourcenschonender Aufbau der Pipeline, weitere Dimensionsreduktion	Ein Teil der Berechnung wird auf Cloud ausgelagert.	Christoph Landolt	Open	23-May-22		
3	zu niedrige Genauigkeit der ML-Algorithmen bei der Evaluation des Modells	quality	Daten	25%	6	1.50	Quantifizierung aller möglicher Features, Korrelationsanalyse und Dimensionsreduktion, Hyperparameter-Tuning	Erneute Iteration der Wahl der Features und des Modells	Christoph Landolt	Open	2-May-22		
228	Projektverzögerungen durch falsche Framework- / Technologiewahl	time	Projekt	20%	5	1.00	Orientierung an bestehenden Projekten, Einsatz von ausreichend dokumentierten Frameworks	Reorientierung vor nächstem Meilenstein	Christoph Landolt	Open	laufend		
230	Projektverzögerungen durch erhöhten Aufwand bei Grundlagenforschung	time	Projekt	20%	5	1.00	Orientierung an bestehenden Forschungsergebnissen	Reorientierung bei der Modellwahl	Christoph Landolt	Open	2-May-22		
9	System funktioniert auf Showcase-Aufbau nicht	feature	Showcase	10%	8	0.80	Realitätsnahes Erfassen von Daten	Ermittlung der Ursache, zusätzliche Trainingsdaten für Showcase aufzeichnen	Christoph Landolt	Open	11-Jul-22		
229	Projektverzögerungen durch Krankheit/Unfall von Projektmitgliedern	time	Projekt	10%	7	0.70	Gesunde Ernährung, Sport, Impfung	Information an Lenkungsausschuss, Anpassung des Projektumfangs, Reorientierung des Projektplans	Christoph Landolt	Open	laufend		
227	Unverhagesehene Kosten für Datenbeschaffung und/oder Showcase-	cost	Projekt	10%	5	0.50	Budget von Kleinkosten vorhanden	Absprache mit Sponsoren und Partnern	Christoph Landolt	Open	laufend		
7	zu hohe Latenzzeit der Anfragen in Produktion	quality	Modell	10%	3	0.30	Perfomerter Aufbau der Pipeline, weitere Dimensionsreduktion	Ermittlung der Ursache	Christoph Landolt	Open	23-May-22		

D Work Breakdown Structure



E Projektplan



F Anforderungsliste

Anforderungsliste Bachelorarbeit 2022

Erstelltdatum	07.02.2022
Aenderungsdatum	08.02.2022
Version	0.1

1 0 0 Anwendungsumfeld

1 1 0 Use Cases

1 1 1 Szenario 1	Schutz von Smart-Home-Gateway	F
1 1 2 Szenario 2	Schutz von ConnectedDrive	F
1 1 3 Szenario 3	Schutz von mobilen Robotern	F

1 2 0 Hardwareanforderungen

1 2 1 RAM	max. 4 GB	F
1 2 2 Speicher	max. 8 GB	F
1 2 3 Prozessor	64 bit, 1.5 GHz, Quad Core	F
1 2 4 Grafikprozessoreinheit	keine	F

1 3 0 Kommunikation

1 3 1 Entschlüsselung und Zertifikatsverwaltung 1	Mittels bestehender Technologie	F
1 3 2 Entschlüsselung und Zertifikatsverwaltung 2	Mittels eigenem System	W
1 3 3 Socket-Verwaltung des Webservers 1	Mittels bestehender Technologie	F
1 3 4 Socket-Verwaltung des Webservers 2	Mittels eigenem System	W
1 3 5 Untersuchte Kommunikationsform 1	HTTP REST	F
1 3 6 Untersuchte Kommunikationsform 2	HTTP allgemein	S
1 3 7 Untersuchte Kommunikationsform 3	weitere Protokolle	W
1 3 8 Schutz in Schichten unterhalb der Applikationsschicht	nicht implementiert	F

1 4 0 Softwareanforderungen

1 4 1 Betriebssystem	Linux Ubuntu	S
1 4 2 Programmiersprache	Python	S
1 4 3 Programmiersprache	C++	W

2 0 0 Daten

2 1 0 Trainingsdaten

2 1 1 Quelle 1	öffentliche zugängliche klassifizierte Datensätze	S
2 1 2 Quelle 2	öffentliche zugängliche unklassifizierte Datensätze	F
2 1 3 Quelle 3	reale Daten von OST	W
2 1 4 Quelle 4	reale Daten von IBM	W
2 1 5 Quelle 5	reale Daten von Dritten	W
2 1 6 Quelle 6	selbstgenerierte Daten mittels Honey-Pot	W
2 1 7 Quelle 7	selbstgenerierte Daten mittels Angriffs-Agent	W

2 2 0 Produktionsdaten

2 2 1 Batch-Vearbeitung zur Erkennung des Concept Drifts	Berechnung in der Cloud	W
2 2 2 Batch-Vearbeitung zur Erkennung des Concept Drifts	Berechnung auf Endgerät	F
2 2 3 Reinforcement Learning Agenten	Berechnung in der Cloud	W
2 2 4 Reinforcement Learning Agenten	Berechnung auf Endgerät	W

3 0 0 Algorithmus

3 1 0 Performance

3 1 1 Latenzzeit pro Anfrage	max. 200ms	W
3 1 2 Häufigkeit der Anfragen	max. 2/Sekunde	W
3 1 3 Arbeitsspeicherbedarf	max. 4GB	F

3 2 0 Wartungsfreundlichkeit

3 2 1 Log-Aufbewahrungszeit auf Device	min. 1 Tag	W
3 2 2 Log-Events	Information (Zugriff), Warning (Angriff), Error (Systemfehler)	W
3 2 3 Modularität	Softwarekomponenten modular aufgebaut	W
3 2 4 Backup/Restore	Sicherung und Wiederherstellung der Modellparameter	W
3 2 5 Error Handling	Verhindern von blockierenden Systemzugriffen (Timer)	W

3 3 0 Qualität

3 3 1 Accuracy 1	90%	F
------------------	-----	---

3 3 2 Accuracy 2 95% S

4 0 0 Show Case

4 1 0 Demoanwendung

- 4 1 1 Zu schützendes System
 - 4 1 2 Darstellung
 - 4 1 3 Hands-on Demonstration
- Smarthome Gateway zu Philips Hue
Dashboard mit Systemzustand und Einstellungen erwünscht

W
W
W

5 0 0 Diverses

5 1 0 Finanzierung

- 5 1 1 Budget Industriepartner
- 3'000.- CHF

F

5 2 0 Eigentum

- 5 2 1 Geistiges Eigentum an der Entwicklung
- 5 2 2 Eigentumsverhältnisse bei unbeanspruchten Mitteln
- 5 2 3 Eigentumsverhältnisse bei beanspruchten Mitteln

Bei den Studierenden
Die Studierenden sind Eigentümer des Produktes.
Der Industriepartner Eigentümer des Produktes.

F
F
F

5 3 0 Organisation

- 5 3 1 Dokumentation des Projektes
 - 5 3 2 Information an Lenkungsausschuss
- Pflicht
Bei jedem Meilenstein

F
F

Definition

F	Festforderung	Muss erfüllt sein
S	Sollforderung	Sollte erfüllt sein
W	Wunschforderung	Kann erfüllt sein

Eigenständigkeitserklärung

Hiermit bestätigen wir eidesstattlich, dass die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt wurden. Diese schriftliche Arbeit wurde noch an keiner Stelle vorgelegt.

Ort, Datum

Moritz Bättig

Ort, Datum

Christoph Landolt