Grundprojekt Winter 2020
Master Informatik

Simulationsbasierter Vergleich von Methoden zur Anomalieerkennung im Netzwerk des Fahrzeugs

Wilhelm Schumacher

Eingereicht am 7.6.2020

Zusammenfassung Mit dem immer weiter voranschreitenden Ausbau der Kommunikation im internen Netzwerkes des Autos steigt auch gleichzeitig der Bedarf an IT-Security-Konzepten im Auto. Zum Beispiel verändert der Einsatz von Echtzeit-Ethernet mit TSN im zentralen Backbone des Netzes und auf lange Sicht die Entwicklung hin zu einem flachen Ethernet-Netzwerk die Kommunikationsstrukturen des Netzes. Zusätzlich erfolgt im Moment eine Öffnung des internen Netzes hinzu vermehrter Kommunikation mit der Außenwelt. Ein Security-Konzept, das in diesem Kontext im Moment wachsende Aufmerksamkeit erhält, ist die Entdeckung von Angriffen durch Anomalieerkennung. Unterschiedliche Verfahren zur Anomalieerkennung werden bereits in anderen Domänen wie zum Beispiel bei der Erkennung von Kreditkartenbetrug, im medizinischen Bereich, für Fehlererkennungen im Raumschiff (Safety) oder klassisch für Intrusion Detection Systeme, die Angriffe auf Netzwerke entdecken sollen, erfolgreich eingesetzt. Das langfristige Ziel ist es Herauszufinden, wie praktikabel der Einsatz von Anomalieerkennung im Fahrzeug ist und welche der bekannten Algorithmen besonders gut für die Erkennung von Anomalien innerhalb der Automotive Security Domäne geeignet sind. Diese Ausarbeitung gibt erstmal einen geordneten Überblick zu den Methoden der Anomalieerkennung. Anschließend ist das Ziel dieser Ausarbeitung innerhalb des Simulations-Framework OMNeT++ (Objective Modular Network Testbed in C++) in einem beispielhaften modellierten Fahrzeugnetzwerk Basisalgorithmen zur Anomalieerkennung aus unterschiedlichen Kategorien miteinander zu vergleichen. Dafür werden nach einer Lernphase mit normalen Daten verschiedene Angriffsszenarien durchgeführt werden, die anschließend durch die Algorithmen korrekt erkannt werden müssen.

Schlüsselwörter Automotive Security · Anomaly Detection · Fahrzeugnetzwerk · Anomalieerkennung · Vergleich von Methoden zur Anomalieerkennung · OMNeT++

Wilhelm Schumacher
E-Mail: Wilhelm.Schumacher@haw-hamburg.com
Department Informatik
Fakultät Technik und Informatik
Hochschule für Angewandte Wissenschaften Hamburg

Inhaltsverzeichnis

1 Einleitung 3
2 Überblick Anomalieerkennung 4
3 Methoden der Anomalieerkennung 7
4 Aufbau der Simulationsumgebung und verwendete Algorithmen 12
5 Ergebnisse des Vergleiches und Evaluierung 21
6 Ausblick und Zusammenfassung 29

1 Einleitung

Moderene Fahrzeugnetzwerke bestehen aus einer Vielzahl verteilter Steuereinheiten (ECUs) [vgl. SNA+13, Seite 9], die über verschiedene Kommunikationsmedien systemweit viele Informationen über Betriebszustände und weitere relevante Daten miteinander austauschen. Typischerweise enthält ein modernes Auto über 70 verschiedene ECUs. Zur Verbindung der ECUs werden als Kommunikationsmedien entweder verschiedene Systembusse wie CAN, LIN, MOST, FlexRay oder vor allem in neueren Fahrzeugen Ethernet-Netzwerke [HMVVDK13], die Systembusse schrittweise ersetzen sollen, eingesetzt. Auch die Angriffsoberfläche (attack surface) des Fahrzeuges hat sich durch neue Schnittstellen, wie einer Internetverbindung zur Außenwelt, Car-To-X, OBD-II, Bluetooth usw., mit der Zeit immer mehr erweitert [vgl. CMK+11, Seite 2-4]. Diese Interfaces bieten das Potential für die Umsetzung einer Vielzahl von neuen Funktionen, mit Updates over the Air kann zum Beispiel das Kartenmaterial des autonomen Fahrzeuges aktualisiert werden, mit Car-To-X Kommunikation die Motorsteuerung an die aktuelle Verkehrslage angepasst werden oder mit Hilfe der Cloud der nächste Ladezyklus des Elektroautos inklusive Reservierung und Einbeziehung des Kalenders besser geplant werden.

Jedoch führt die Öffnung des internen Kommunikationsnetzes nach Außen auch zu einer erhöhten Verwundbarkeit der Informationssicherheit (Integrität, Vertraulichkeit, Verfügbarkeit) im internen Netzwerk des Fahrzeuges [vgl. MA11, Seite 1]. Denn durch die Einbindung der neuen Interfaces wird das Auto zu einem attraktiven Angriffsziel für Hacker. Hacker könnten potentielle Schwachstellen in einem der externen Interfaces ausnutzen, um böswillige Pakete in das Netzwerk einzuschleusen, die den normalen Betriebsablauf des Fahrzeuges stören sollen. Zum Beispiel sind die Steuergeräte (ECUs) angreifbar und können nach Kompromittierung genutzt werden, um die gesamte Kommunikation zu manipulieren. Deswegen sind IT-Security-Konzepte zum Schutz des Fahrzeuges notwendig. Ein Security-Konzept, das in diesem Kontext im Moment wachsende Aufmerksamkeit erhält, ist die Entdeckung von Angriffen durch Anomalieerkennung. Unterschiedliche Verfahren zur Anomalieerkennung werden bereits in anderen Domänen wie zum Beispiel bei der Erkennung von Kreditkartenbetrug, im medizinischen Bereich, für Fehlererkennungen im Raumschiff oder klassisch für Intrusion Detection Systeme, die Angriffe auf Netzwerke entdecken sollen, erfolgreich eingesetzt. Einige Algorithmen wurden dabei speziell für diese bestimmten Anwendungsbereiche entwickelt, abhängig von Art der Daten, der Verfügbarkeit von gelabelten Trainingsdaten, den Typen von Anomalien und dem gewünschten Outputformat [vgl. CBK09, Seite 6-7]. In der automotiven Domäne könnte sich der Einsatz von Anomalieerkennung lohnen, da die Verarbeitung auf Grund der Menge an Daten automatisiert durchgeführt werden muss und die Nachrichtenmuster genug Ähnlichkeiten und Zusammenhänge aufweisen um sie von anormalen Daten zu unterscheiden. Weiterhin sind die unterschiedlichen Ebenen, wie z.B. auf Anwendungsebene, im Netzwerk, in den Sensoren oder im GPS Signal, auf denen die Erkennung der Anomalien durchgeführt werden kann, eine Besonderheit der Anomalieerkennung aus dem automotiven Bereich.

Dieses Grundprojekt soll einen Beitrag zu dem langfristigen Ziel Herauszufinden, wie praktikabel der Einsatz von Anomalieerkennung im Fahrzeug ist und welche der bekannten Algorithmen besonders gut für die Erkennung von Anomalien innerhalb der Automotive Security Domäne geeignet sind, leisten. Dazu gibt diese Ausarbeitung erstmal einen geordneten Überblick zu den Methoden der Anomalieerkennung aus der Literatur und zeigt auf, wie diese in Kategorien unterteilt werden können. Anschließend ist das Ziel dieser Ausarbeitung innerhalb des Simulations-Framework OMNeT++ (Objective Modular Network Testbed in C++) in einem beispielhaften modellierten Fahrzeugnetzwerk Basisalgorithmen zur Anomalieerkennung aus unterschiedlichen Kategorien miteinander zu vergleichen. Die Simulation basiert auf Daten aus einer echten Kommunikationsmatrix, die korrekt abbildet welche Komponente mit welcher anderen Komponente kommuniziert. Anhand dieser Daten können die Algorithmen in einer Lernphase das normale Verhalten des Systems lernen. Darauf folgend werden verschiedene Angriffsszenarien durchgeführt, an denen die Algorithmen dann evaluiert werden. Dabei wird betrachtet, ob die jeweiligen Angriffe korrekt erkannt werden und inwiefern Daten als falsch positiv eingestuft werden. Die Arbeit findet innerhalb des SecVI Forschungsprojekt [siehe Sec] statt. Das SecVI Projekt wird in Verbindung mit Industriepartnern durchgeführt und vom Federal Ministry of Education und Research gefördert und hat als Ziel eine sichere Netzwerkarchitektur für das Auto zu entwickeln.

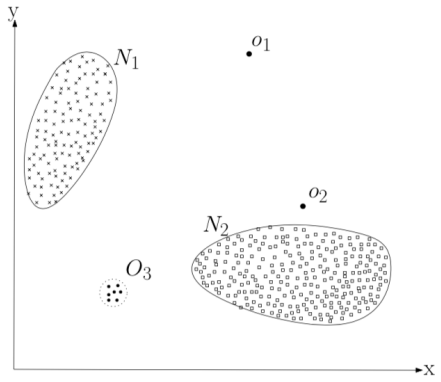
Die folgende Ausarbeitung ist so strukturiert, dass als Erstes das Kapitel **Überblick Anomalieerkennung** relevante Grundlagen zum Thema Anomalieerkennung beschreibt. Anschließend betrachtet das Kapitel **Methoden der Anomalieerkennung** die Zuordnung von Verfahren der Anomalieerkennung in Kategorien und erklärt die wichtigsten Charakteristika der einzelnen Kategorien. Das Kapitel **Aufbau der Simulationsumgebung und verwendete Algorithmen** gibt einen Überblick zu der Funktionsweise der verwendeten Algorithmen, dem Netzwerk, Metriken, Parametern und den Angriffsszenarien. Daraufhin stellt das Kapitel **Ergebnisse des Vergleiches und Evaluierung** die Ergebnisse der Durchführung dar, die dann im folgenden genauer betrachtet und interpretiert werden. Das letzte Kapitel **Ausblick und Zusammenfassung** schließt die Ausarbeitung dann mit einer kurzen Zusammenfassung der wichtigsten Punkte der Ausarbeitung und einem Ausblick hinsichtlich des Hauptprojektes ab.

2 Überblick Anomalieerkennung

Dieses Kapitel erklärt die Grundlagen zum Thema Anomalieerkennung, die für das Verständnis der folgenden Kapitel benötigt werden. Dazu erläutert das Unterkapitel 2.1 **Ziele der Anomalieerkennung** die Begriffe Anomalie und Anomalieerkennung und fasst deren Funktionsweise, Anwendungsbereiche und Ziele kurz zusammen. Anschließend folgen die Kapitel 2.2 **Input Daten**, 2.3 **Typen von Anomalien** und 2.4 **Lernmethoden** mit weiteren wichtigen Aspekten zur Anomalieerkennung.

2.1 Ziele der Anomaly Detection

Die Anomalieerkennung hat als Aufgabe unregelmäßige Muster innerhalb von normalen Daten festzustellen. Diese Unregelmäßigkeiten entsprechen dabei nicht dem normalen Verhalten des Systems, treten seltener auf und unterscheiden sich signifikant vom Rest der Daten. Sie werden hauptsächlich als **Anomalien** oder **Outlier** (Ausreißer) [CBK09] bezeichnet. Anomalieerkennung wird bereits in einer Vielzahl von unterschiedlichen Anwendungsbereichen eingesetzt. Zum Beispiel bei dem Erkennen von Kreditkartenbetrug, zum Entdecken von Tumoren in MRT-Bilder (Magnetresonanztomographie), im Cyber Security Bereich, dem Erkennen von Cheatern in Videospielen [Pat] oder für die Fehlererkennung bei sicherheitskritischen Systemen [FYM05]. In vielen dieser Anwendungsbereiche spielt die Identifikation von Anoma-



Quelle: http://doi.acm.org/10.1145/1541880.15418822 [vgl. CBK09, Seite 2]

Abb. 1 Abbildung 1 stellt die normalen Datenbereiche N1 und N2 den anomalen Beobachtungen O1,O2 und O3 gegenüber.

lien eine große Rolle, denn sie ermöglicht es kritische Situationen zu erkennen und daraufhin die notwendigen Gegenmaßnahmen zu veranlassen. Zum Beispiel können ungewöhnliche Nachrichtenmuster in einem Computernetzwerk darauf hinweisen, dass von einem gehackten Computer aus, sensitive Daten in das Netzwerk an einen unautorisierten Host geschickt werden oder anomales Verhalten in einem Sensor eines Raumschiffes könnte auf Fehler in einer Komponente hinweisen.

Ein konkretes Beispiel für eine Anomalie ist in Abbildung 1 dargestellt. Hier können innerhalb eines zweidimensionalen Datensatz sowohl normale Bereiche (N_1 und N_2) als auch Anomalien (O_1, O_2 und O_3), die weiter von den restlichen Daten entfernt liegen, erkannt werden. Häufig ist es aber schwierig präzise Grenzen zwischen den normalen und anomalen Daten, wie in Abbildung 1, zu definieren und stellt eine der großen Herausforderungen der Anomalieerkennung dar. Zum Beispiel können Anomalien, die dicht an der Grenze eines normalen Datenbereiches, nur sehr schwierig als Anomalie oder normaler Datensatz eingeordnet werden.

2.2 Input Daten

Abhängig vom Anwendungsbereich kann sich die Art der Input Daten deutlich voneinander unterscheiden. Im Allgemeinen sind die Input Daten erstmal eine Sammlung von Daten (bezeichnet als Objekte, Vektoren, Beobachtungen, Punkte etc.), die wiederum aus einer Menge von Attributen (bezeichnet als Variablen, Felder, Feature etc.) bestehen [vgl. CBK09, Seite 6-7]. Ein Attribut stammt dabei aus einer von drei Kategorien **binär**, **kategorisch** oder **kontinuierlich**. Wenn die Daten jeweils nur aus einem Attribut bestehen, werden sie als **Univariate** bezeichnet und wenn die Daten aus mehreren Attributen bestehen als **Multivariate**. Je nachdem welche Arten von Attributen oder Kombinationen von Attributen innerhalb der Daten vorhanden sind, können andere Algorithmen angewendet werden oder die Daten müssen entsprechend angepasst werden. Zum Beispiel benötigen einige Algorithmen (Nearest Neighbor) ein Verfahren zur Distanzbestimmung, das für binäre Attribute anders funktioniert als für kategorische Attribute.

2.3 Typen von Anomalien

Eine Anomalie kann in eine von drei Kategorien [Gmb18] eingeordnet werden. Die erste und einfachste Kategorie wird als **Punktanomalie** bezeichnet. Bei einer Punktanomalie wird ein einziger Datenpunkt in Anbetracht zu den restlichen Daten als Anomalie eingestuft. Ein Beispiel aus der realen Welt für eine Punktanomalie wäre zum Beispiel die Höhe des ausgegebenen Betrages bei Bezahlung mit Kreditkarte. Wenn über einen längeren Zeitraum nur sehr kleine Beträge bezahlt wurden und dann ein im Vergleich zu diesen Beträgen sehr großer Betrag auftritt, handelt es sich um eine Punktanomalie. In der zweiten Kategorie **kollektive Anomalien** verhält sich eine Reihe von Datenpunkten abweichend vom Rest der Daten. Dabei sind die individuellen Datenpunkte aus der kollektiven Anomalie nicht unbedingt anormal, sondern nur deren gemeinsames Auftreten. Ein Beispiel für eine kollektive Anomalie ist das Signal eines Elektrokardiogramms [vgl. CBK09, Seite 9] . Wenn das Signal über eine längeren Zeitraum stabil bleibt und keine steilen Anstiege enthält wie in den anderen Intervallen, handelt es sich um eine kollektive Anomalie. Die dritte Kategorie von Anomalien sind die **kontextuellen** Anomalien. Bei einer kontextuellen Anomalie wird ein Wert nur in einem spezifischen Kontext als Anomalie betrachtet. Zum Beispiel könnte eine kontextuelle Anomalien in Temperatur-Zeitreihen auftreten. Ein sehr niedriger Temperaturwert im Winter wäre nicht ungewöhnlich, aber im Sommer wäre es eine kontextuelle Anomalie.

2.4 Lernmethoden

Bei der Implementierung von Anomalieerkennungen wird zwischen den drei Methoden überwachtes Lernen, semiüberwachtes Lernen und unüberwachtes Lernen unterschieden [vgl. BBK14, Seite 7]. Welche Methode angewendet werden kann, hängt hautsächlich davon ab, ob gelabelte Trainingsdaten oder nur ungelabelte Trainingsdaten zur Verfügung stehen. Bei gelabelten Trainingsdaten sind die Datenpunkte

entweder als normal oder anormal markiert. Häufig ist es aber sehr schwierig und aufwendig, genug korrekt gelabelte Trainingsdaten zu erhalten, weil diese oft manuell von menschlichen Experten erarbeitet werden müssen. Beim überwachten Lernen wird vorausgesetzt das Datenpunkte sowohl für die normale Klasse als auch für die anormale Klasse verfügbar sind. Auf Basis dieser Eingabedaten soll das System dann ein Model erstellen, dass es ermöglicht möglichst zielsicher vorauszusagen, ob ein neuer Datenpunkt eine Anomalie darstellt oder nicht. Dagegen wird beim semiüberwachten Lernen nur auf Grundlage von Daten aus der normalen Klasse bestimmt, ob eine neue Dateninstanz anormal ist oder nicht. Ein Vorteil des semiüberwachten Lernens ist die bessere Anwendbarkeit, weil keine Daten für die Klasse der Anomalien benötigt werden. Für das unüberwachte Lernen werden überhaupt keine gelabelten Trainingsdaten benötigt. Es wird davon ausgegangen das normale Dateninstanzen in den Testdaten deutlich häufiger auftreten als Anomalien. Anomalien können dann dadurch identifiziert, indem Datenpunkte gefunden werden, die von den am häufigsten auftretenden Mustern abweichen. Ein großer Nachteil dieser Technik ist, dass wenn die Annahme nicht stimmt, das System sehr viel falsch positive Rückmeldungen bekommt.

3 Methoden der Anomalieerkennung

Das Kapitel Methoden der Anomalieerkennung blickt auf Methoden, die zur Anomalieerkennung verwendet werden und ordnet diese in Kategorien ein. Dazu betrachtet es verschiedene Ansätze zur Kategorisierung aus der Literatur. Es erläutert für die Kategorien Classification, Statistisch, Clustering und Nearest Neighbor die grundlegende Funktionsweise und nennt Beispiele für Algorithmen aus diesen Kategorien. Weiterhin betrachtet das Kapitel die Vor- und Nachteile der einzelnen Kategorien und gibt einen Ausblick auf weitere Algorithmen, die zur Anomalieerkennung benutzt werden können.

3.1 Klassen von Algorithmen

Verfahren zur Anomalieerkennung klar voneinander abzugrenzen ist ziemlich schwierig. Aus diesem Grund findet man auch verschiedene Ansätze von Kategorisierungen in der Literatur wieder. Drei Beispiele für verschiedene Kategorien, in die die Algorithmen eingeordnet werden können, sind in der Tabelle 1 dargestellt. In der Tabelle werden die Kategorien aus den Arbeiten von [BBK14], [CBK09] und [GTDVMFV09] einander gegenübergestellt. Gemeinsam ist bei allen drei Ausarbeitungen, dass jeweils eine Kategorie für die statistischen Verfahren existiert. Unterschiede finden sich vor allem in der Kategorie Clustering. Während bei [GTDVMFV09] Clustering-Verfahren zu der Kategorie Machine Learning zählen, haben [BBK14] und [CBK09] diese Art von Algorithmen in eine separate Kategorie eingeordnet. Zusätzlich werden bei [CBK09] aus den Clustering-Verfahren nochmal die Nearest Neighbor Based Verfahren in eine extra Kategorie abgetrennt. Die Trennung basiert auf der Annahme, dass Clustering-Verfahren den Fokus auf die Bildung des Clusters und die anschließende

Einordnung legen. Im Gegensatz dazu fokussieren sich die Nearest Neighbor Based Verfahren auf die lokale Nachbarschaft der Dateninstanzen. Eine weitere Kategorie die in allen drei Arbeiten zu finden ist, sind die Klassifikationsalgorithmen (Machine Learning bei [GTDVMFV09]). Ergänzend dazu sind noch weitere Kategorien wie Combination Learner, Spectral, Information Theoretic und Knowledge Based vorhanden, die speziellere Verfahren zusammenfassen und einzig in jeweils einer Ausarbeitung als Kategorie vorkommen.

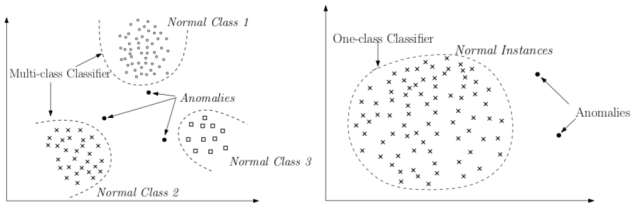
Bhuyan et al.: Network Anomaly Detection: Methods, Systems and Tools	Chandola et al.: Anomaly Detection : A Survey	P. García-Teodoro et al.: Anomaly-based network intrusion detection: Techniques, systems and challenges
1. Statistical	1. Statistical	1. Statistical
2. Classification Based	2. Classification Based	2. Machine Learning Based
3. Clustering and Outlier Based	3. Clustering Based	3. Knowledge Based
4. Soft Computing	4. Nearest Neighbor Based	-
5. Knowledge Based	5. Information Theoretic	-
6. Combination Learners	6. Spectral	-

Tabelle 1 zeigt beispielhaft Kategorien in die Algorithmen/Methoden zur Anomalieerkennung eingeordnet werden können.

3.2 Classification Based Verfahren

Die erste Kategorie von Verfahren stellen die Classification Based Verfahren dar. Classification Based Verfahren versuchen neu auftretende Dateninstanzen anhand eines zuvor gelernten Models in verschiedene Klassen einzuordnen [vgl. BBK14, Seite 12]. Dies geschieht in 2 Phasen. Zuerst erfolgt eine Trainingsphase in der gelabelte Trainingsdaten benutzt werden, um das Model zu lernen. Anschließend wird in einer Testphase durch einen Klassifikator mit Hilfe des Models entschieden, ob es sich um eine normale oder anormale Instanz handelt. Dabei wird zwischen Verfahren mit nur einer normalen Klasse (one-class) und Multi-Klassen (multi-class) unterschieden. Bei Multi-Klassen Verfahren wird angenommen, dass er mehrere normale Klassen gibt, in welche die Dateninstanzen eingeordnet werden können. Wenn der Klassifikator der Dateninstanz keine der normalen Klassen sicher zuordnen kann, wird diese Dateninstanz als anormal eingestuft. In Abbildung 2 ist die Unterscheidung zwischen den beiden Verfahren dargestellt und es wird gezeigt was in dem jeweiligen Kontext eine Anomalie ist.

Beispiele für Klassifikationsalgorithmen sind verschiedene Varianten von neuronalen Netzen, bayessche Netze, Support Vector Machines (SVM) oder regelbasierte Systeme. Bei Klassifikationsalgorithmen besitzt die Trainingsphase abhängig vom konkreten Algorithmus zwar eine hohe Rechenkomplexität, dafür ist die Testphase



(a) Multi-class Anomaly Detection (b) One-class Anomaly Detection

Quelle: http://doi.acm.org/10.1145/1541880.15418822 [vgl. CBK09, Seite 21]

Abb. 2 Abbildung 2 verdeutlicht die Funktionsweise von one-class Classification und multi-class Classification und hebt hervor, wo die Anomalien jeweils zu finden sind.

zum Einstufen neuer Beobachtungen wiederum sehr schnell [vgl. CBK09, Seite 21]. Dies stellt einen großen Vorteil der Klassifikationsalgorithmen dar. Ein weiterer Vorteil ist, dass zum Beispiel die multi-class Verfahren es erlauben zwischen verschiedenen normalen Klassen zu unterscheiden. Nachteile von Klassifikationsalgorithmen sind, dass viele Verfahren sehr abhängig von korrekt gelabelten Trainingsdaten sind und als Output nur angeben, ob eine Dateninstanz zu einer Klasse gehört anstatt einen Score anzugeben, der verdeutlicht wie wahrscheinlich es sich um eine anormale Instanz handelt.

3.3 Nearest Neighbor Based Verfahren

Nearest Neighbor Based Verfahren [vgl. CBK09, Seite 28] benutzen die Distanz bzw. die Ähnlichkeit (bei kategorischen Attributen) zwischen zwei Datenpunkten als Grundlage zur Anomalieerkennung. Abhängig von den Distanzen/Ähnlichkeiten ihrer Attribute liegen Datenpunkte verschieden weit auseinander. Mit der Annahme, dass normale Daten gesammelt in kurzer Distanz zueinander liegen und anormale Daten weit von ihren Nachbarn entfernt liegen, sollen Anomalien identifiziert werden. Welche Verfahren dabei zur Distanzbestimmung verwendet werden können, hängt von der Art der Attribute der Daten ab und ob es sich um Univariate oder Multivariate (siehe 2.2) handelt. Zum Beispiel ist ein einfaches Verfahren zur Distanzbestimmung der Euklidische Abstand. Der Euklidische Abstand kann aber nur auf kontinuierliche Daten angewendet werden. Daher müssen bei kategorischen Datentypen oder gemischten Attributtypen andere Verfahren oder Kombinationen von Verfahren benutzt werden. Die Funktionsweise von Nearest Neighbor Based Verfahren kann in die zwei Kategorien **k^{th} Nearest Neighbor** und **Relative Density** unterteilt werden. Bei k^{th} Nearest Neighbor Verfahren wird für eine Dateninstanz deren Distanz zu den nächsten k-Nachbarn als Anomalie-Score betrachtet. Anhand eines Schwellenwertes (threshold) wird die Dateninstanz dann als normal oder anormal eingestuft. Diese Basisvariante wurde von vielen Forschern noch erweitert, indem zum Beispiel nur die Nachbarn einer Dateninstanz, die nicht weiter als Distanz x entfernt liegen, gezählt werden, um zu Ermitteln bei welchen Daten es sich um Anomalien handelt [KNT00]. Bei Relative Density Verfahren wird eine Anomalie an der Dichte der Nachbarschaft

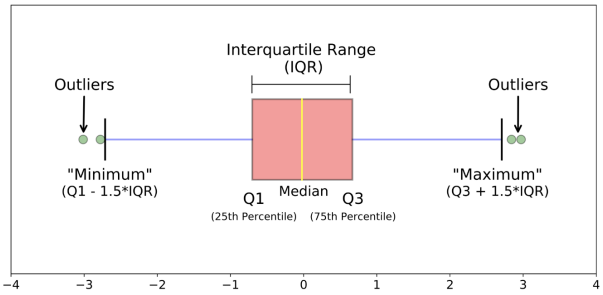
erkannt. Es wird davon ausgegangen, dass Anomalien in Bereichen mit geringer Dichte an Nachbarn liegen, während normale Daten eine hohe Nachbarschaftsdichte besitzen. Der Abstand einer Dateninstanz zu seinem k entferntesten Nachbar stellt eine Schätzung der Inversen der Dichte im Datensatz dar. Wenn Datensätze mit variierenden Dichten vorliegen, wird es notwendig auch die Lokalität des Datensatzes in Relation zu seinen Nachbarn zu berücksichtigen. Ein Beispiel für ein Verfahren, das eine lokale Dichte anstatt einer globalen Dichte verwendet, ist der Local Outlier Factor [KKSZ09].

Ein Vorteil von Nearest Neighbor Based Verfahren ist, dass sie von Natur aus unsupervised sind und keine direkten Annahmen über die Verteilung der Daten treffen [vgl. CBK09, Seite 28]. Außerdem sind Nearest Neighbor Based Algorithmen gut anpassbar in Bezug auf verschiedene Datentypen, denn es wird nur ein anderes Verfahren zur Distanzbestimmung benötigt und Nearest Neighbor Based Verfahren können auch im semi-supervised Modus genutzt werden. Nachteile dieser Verfahren sind, dass normale Ausreißer ohne direkte Nachbarn als Anomalien eingeordnet werden könnten und Anomalien, die in Mitten von normalen Daten liegen, als normal eingestuft werden. Auch die Komplexität der Verfahren in der Testphase ist höher als beispielsweise bei Classification-based Verfahren und weiterhin hängt die Anomalieerkennungsrate davon ab, wie gut die benutzte Methode zur Distanzbestimmung zwischen normalen und anormalen Daten differenzieren kann.

3.4 Statistische Verfahren

Auch statistische Methoden können zur Erkennung von Anomalien eingesetzt werden. Als Grundlage dazu wird ein statistisches Model benutzt [vgl. BBK14, Seite 9-11]. Im ersten Schritt muss das Model dann mit normalen Daten befüllt werden. Mit Hilfe dieses Models können dann Anomalien erkannt werden, indem geschaut wird, wie wahrscheinlich es ist, dass eine neue Dateninstanz von diesem Model generiert wurde. Bei statistischen Anomalieerkennungsverfahren wird zwischen **parametrisierten** und **nicht-parametrisierten** Varianten unterschieden. Bei der parametrisierten Variante ist die Verteilungsfunktion der Daten schon bekannt und die Parameter werden durch die gegebenen normalen Daten eingestellt. Der Anomalie-Score wird durch das Inverse einer Wahrscheinlichkeitsdichtefunktion $f(x,\Theta)$ mit Parameter Θ und neuer Dateninstanz x bestimmt. Weiterhin unterscheiden sich die parametrisierten Verfahren, darin welche Verteilungsfunktion benutzt wird (z.B. Gaußsche Verteilung oder Regressionsmodell) und der Art und Weise wie Anomalien erkannt werden. Zum Beispiel können in einer einfachen Variante alle Dateninstanzen die mehr als die dreifache Standardabweichung von Durchschnitt der Verteilung entfernt sind, als Anomalien eingeordnet werden oder es wird die Box Plot Rule [LJK00] (siehe Abbildung 3) verwendet. Bei der Box Plot Rule werden durch einen minimalen und einen maximalen Wert Grenzen definiert und der restliche Bereiche wird in unteren, mittleren und oberen Quarter unterteilt. Bei den nicht-parametrisierten Methoden werden statistische Modele ohne Parameter benutzt, wie zum Beispiele Methoden, die auf Histogrammen basieren. Dabei wird zuerst durch die Daten ein Histogramm

aufgebaut, das abhängig von den Werten eines Features verschiedene Bereiche mit unterschiedlichen Höhen (stellen Häufigkeiten dar) enthält. Für neue Testinstanzen wird dann überprüft in welchen Bereich des Histogramms der Datensatz fällt. Die Höhe des Bereiches bestimmt dann den Anomalie-Score der Dateninstanz.



Quelle: https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51

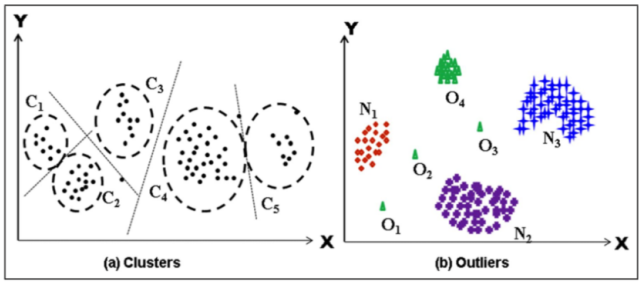
Abb. 3 Abbildung 3 stellt dar wie mit Hilfe eines Boxplots Anomalien identifiziert werden können. Er ist durch die Grenzen Minimum, Q1, Median, Q2 und Maximum in 4 ”Quarter” unterteilt.

Ein Hauptvorteil von statistischen Verfahren ist das ein Anomalie-Score häufig auch eine Wahrscheinlichkeitsangabe als extra Information bereitstellt [vgl. CBK09, Seite 39-40], die angibt wie sicher es sich um eine Anomalie handelt. Weiterhin bieten statistische Verfahren eine statistisch richtige Lösung bei der Erkennung von Anomalien an, wenn die zugrunde liegende Verteilung stimmt. Es ist auch möglich statistische Verfahren in einem unüberwachten Modus auszuführen. Nachteile sind hingegen, dass statistische Verfahren davon ausgehen, dass die Verteilung der realen Daten auf einer bestimmten Verteilung basiert und diese Annahme nicht immer stimmen muss. Ein weiterer Nachteil ist das Techniken die Histogramme benutzten Schwierigkeiten bei der Erkennung von Anomalien in Daten mit multivariaten Attributen haben.

3.5 Clustering-Verfahren

Eine weitere Möglichkeit zur Anomalieerkennung bieten Clustering-Verfahren. Das Ziel von Clustering ist es eine Menge von ähnlichen Objekten in gemeinsame Cluster einzuordnen [vgl. CBK09, Seite 30]. Abbildung 3 zeigt beispielhaft wie eine Menge von Datenpunkten in fünf verschiedene Cluster eingeordnet werden könnte. Die meisten Clustering-Verfahren operieren im unsupervised Modus. Zur Erkennung von Anomalien können unterschiedliche Annahmen genutzt werden. Eine einfache Annahme ist zum Beispiel, dass nur normale Daten zu den Clustern gehören und Anomalien nicht zu einem Cluster gehören. Algorithmen dieser Kategorie ordnen nicht zwangsläufig jeder Dateninstanz ein Cluster zu und die übriggebliebenen Daten werden als Anomalien eingestuft. Ein weitere Annahme besagt, dass normale Daten dichter am

Mittelpunkt des Clusters (cluster centroid) liegen, während anormale Daten weit vom Mittelpunkt entfernt liegen. Ein Algorithmus aus dieser Kategorie wäre zum Beispiel K-Means Clustering. Die letzte Annahme geht davon aus, dass normale Dateninstanzen sehr große und dichte Cluster bilden und daher nicht mit kleinen Clustern aus Anomalien zu verwechseln sind.



Quelle: https://ieeexplore.ieee.org/document/6524462 [vgl. BBK14, Seite 14]

Abb. 4 Auf der linken Seite der Abbildung werden insgesamt 5 verschiedene Cluster in einem zweidimensionalen Raum dargestellt. Der rechte Teil verdeutlicht, wo Ausreißer zwischen den Clustern zu finden sind.

Die Vorteile von Clustering-Verfahren bei der Anomalieerkennung sind zum einen die Möglichkeit im unsupervised Modus zu operieren [vgl. CBK09, Seite 32] und zum anderen sind Clustering-Algorithmen bei der Einordnung von neuen Dateninstanzen sehr schnell, da sie die neue Dateninstanz nur mit einer kleinen Menge von Clustern vergleichen müssen. Ein weiterer Vorteil besteht darin, wenn die Anzahl k an Clustern vorab bekannt ist. Dadurch wird die Bildung der Cluster und die Zuordnung sehr einfach. Ein Nachteil von Clustering-Verfahren ist, dass einige Algorithmen Probleme bekommen, wenn Anomalien selber wieder ein Cluster bilden. Ein weiterer Nachteil stellt die hohe Rechenkomplexität $\mathcal{O}(n^2)$ einiger Clustering-Algorithmen dar und die Effektivität der Clustering-Verfahren hängt stark davon ab, ob die Cluster die Struktur der normalen Daten abbilden können.

4 Aufbau der Simulationsumgebung und verwendete Algorithmen

Dieses Kapitel erläutert die Versuchsumgebung, die für den Vergleich der Algorithmen verwendet wurde. Dazu verdeutlicht es den Aufbau des verwendeten Netzwerkes, stellt die Parameter der Simulation wie verwendete Metriken, Trainingsdauer und Ungenauigkeiten dar. Weiterhin erklärt es die grundlegende Funktionsweise aller eingesetzten Algorithmen und zum Abschluss listet das Kapitel noch die sechs durchgeführten Angriffsszenarien auf.

4.1 Aufbau des RecBarToZones Netzwerkes

Die Basis des Vergleiches der Algorithmen bildet eine Simulation in OMNeT++. Die Simulation modelliert ein Fahrzeugnetzwerk eines bekannten Fahrzeugherstellers. Die Daten der Simulation stammen aus einer echten Kommunikationsmatrix, die korrekt abbildet welche Komponente mit welcher anderen Komponente kommuniziert. Weiterhin werden nach IEEE 802.1Q [LHC12] verschiedene Prioritätsklassen innerhalb des Netzwerkes verwendet.

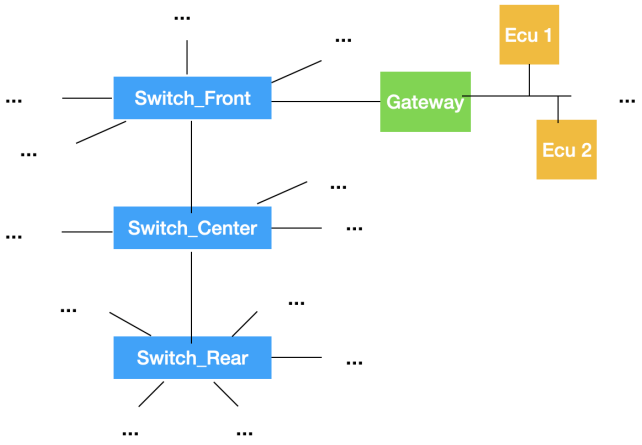


Abb. 5 Diese Abbildung stellt vereinfacht den Aufbau des Netzes, das innerhalb der Simulation verwendet wird, dar.

Die Abbildung 5 zeigt vereinfacht den grundlegenden Aufbau des Netzwerkes. Das Netzwerk besteht ingesamt aus drei Switches. Diese befinden sich an der Frontseite, in der Mitte des Netzwerkes und an der Rückseite. Weiterhin verfügt jeder Switch über eine bestimmte Anzahl an Ports. Diese Ports stellen über Ethernet entweder eine Verbindung zu einem der anderen Switche dar oder eine Verbindung zu einem Gateway (in der Abbildung grün gekennzeichnet). Jede dieser Verbindungen verfügt über eine maximale Bandbreite von 100 MBit/s. An den Gateways sind mehrere CAN-Bus Systeme angeschlossen, die alle eine bestimmte Anzahl an ECUs enthalten. Weiterhin befinden sich im Netzwerk auch noch weitere Komponenten wie zum Beispiel Kameras oder eine Modul für die Internetverbindung.

4.2 Metriken und weitere Parameter

Im folgenden wird erläutert wie das Netzwerk konfiguriert wurde, um den Versuch durchzuführen.

4.2.1 Welche Daten werden betrachtet und wo wird die Anomalieerkennung durchgeführt?

In diesem Vergleich wird ein Datenstrom einer spezifischen ECU betrachtet. Die Anomalieerkennung wird an einem der Eingangsport im zentralen Switch durchgeführt. Die Entscheidung nicht den gesamten einkommenden Netzwerkverkehr zu betrachten hat den Grund, dass z.B. minimale Änderungen von Frequenzen einer einzelnen ECU im gesamten Netzwerkverkehr untergehen würden. Daher werden nur Pakete mit einer spezifischen ID (106) von der ECU 125 an das Modul zur Anomalieerkennung weitergeleitet. In Abbildung 6 wird ein Ausschnitt des Netzwerkes gezeigt. Es ist zu erkennen, dass mehrere ECUs an das gleiche CAN-Bus System wie die ECU 125 angeschlossen sind und diese alle über ein Gateway mit dem zentralen Switch verbunden sind.

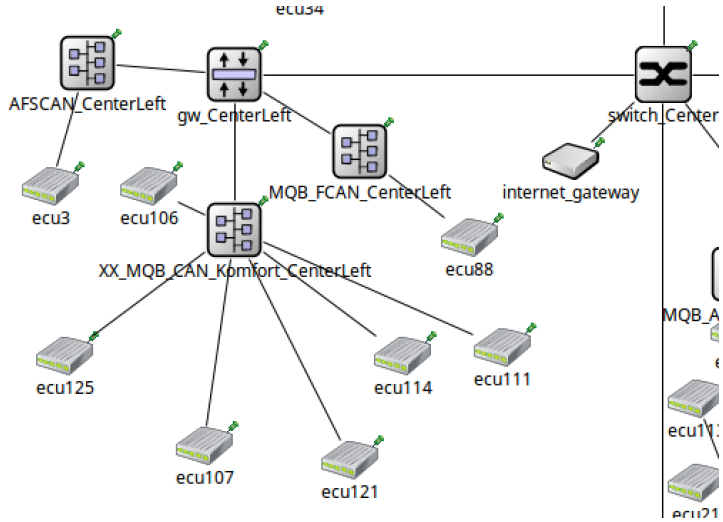


Abb. 6 Diese Abbildung zeigt einen kleinen Ausschnitt aus dem RecBarToZones-Netzwerk. Die Anomalieerkennung wird am Eingang der zentralen Switches mit Paketen der ECU 125 durchgeführt.

4.2.2 Metriken

Zur Anomalieerkennung innerhalb des Versuches werden die Metriken **Bandbreite** und **Jitter** betrachtet. Die Bandbreite wird berechnet durch die Anzahl und Größe der Pakete in einem Zeitintervall. Der Jitter ergibt sich aus der Differenz des größten Abstandes zwischen zwei Paketen und dem geringsten Abstand zwischen zwei Paketen in einem Zeitintervall.

4.2.3 Training

Damit die Algorithmen das normale Verhalten des Netzwerkes lernen können, erfolgt zuerst eine Trainingsphase. Insgesamt lernt jeder Algorithmus über eine Dauer von **480 s**. Dazu wird ein Trainingsintervall von **0.4 s** verwendet. Aus der Trainingsdauer und dem Trainingsintervall ergeben sich 1200 Datenpunkte zum Lernen des normalen Verhaltens für den Algorithmus. In Anbetracht der vorliegenden relativ zyklischen Daten und des sehr zeitaufwendigen Trainings ist diese Anzahl an Daten/Trainingsdauer festgelegt worden. Eine größere Anzahl an Trainingspunkten könnte möglicherweise die Qualität des Trainings noch verbessern. Das Trainingsintervall wurde auf die Dauer von 0.4 s festgelegt, weil in diesem Intervall immer genug Pakete vorhanden sind für einen vernünftigen Datenwert und gleichzeitig ist das Intervall nicht zu lange, so dass kleine Auffälligkeiten nicht im Durchschnitt untergehen. Im Durchschnitt erhält das Modul zur Anomalieerkennung immer ca. 10 Pakete in einem Intervall, da die ECU 125 Pakte mit der ID 106 in einem Intervall von 0.04 s versendet.

4.2.4 PeriodInaccurracy

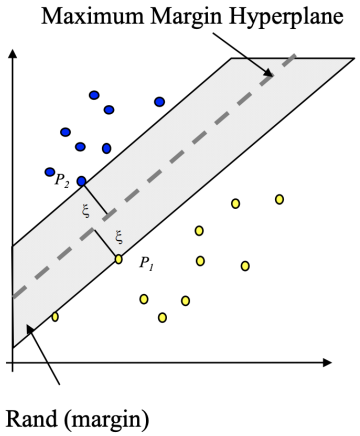
Im Netzwerk wird an jeder ECU eine PeriodInaccurracy von 5 % konfiguriert. Dadurch soll simuliert werden, dass Taktzyklen im Fahrzeug durch z.B. bestimmte physikalische Bedingungen wie die Temperatur abweichen können.

4.3 Verwendete Algorithmen

Die folgenden sechs Unterkapitel beschreiben die grundlegende Funktionsweise der Algorithmen, die innerhalb der Simulation benutzt werden.

4.3.1 Support Vector Machines

Ein Algorithmus zur Erkennung von Anomalien ist Support Vector Machines (SVM). Dieser Algorithmus ist der Kategorie der Klassifikationsalgorithmen 3.2 zuzuordnen. In seiner Grundform hat der Algorithmus als Ziel zwei verschiedene Klassen voneinander zu trennen [JA03]. Die Trennung wird durch Bildung einer Hyperebene vorgenommen (siehe Abbildung 7). Diese Hyperebene wird durch lineare Separation mit dem „Maximum Margin Hyperlane“ Verfahren gebildet. Das Verfahren bestimmt die Trennlinie (Hyperlane) so, dass der Abstand zu Punkten aus beiden Mengen maximal ist. Neue unbekannte Punkte werden der Seite der Ebene zugeordnet, auf der sie sich befinden. Zur Anomalieerkennung mit SVM wird eine leicht modifizierte Variante mit One-Class SVM [sl] benutzt. One-Class SVM ist eine unüberwachte Variante, die nur eine Grenze um die normalen Daten bildet und alle Punkte außerhalb als Ausreißer deklariert.



Quelle: https://www.dbs.ifi.lmu.de/Lehre/KDD/WS1011/skript/kdd-3-klassifikation7.pdf [JA03]

Abb. 7 Die Abbildung demonstriert die Trennung von Punkten aus zwei verschiedenen Klassen via Maximum Margin Hyperlane.

4.3.2 Isolation Forest

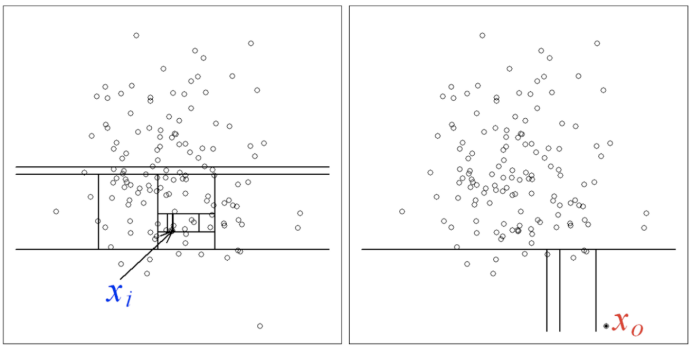


Figure 1 Identifying normal vs. abnormal observations

Quelle: https://towardsdatascience.com/outlier-detection-with-isolation-forest-3d190448d45e

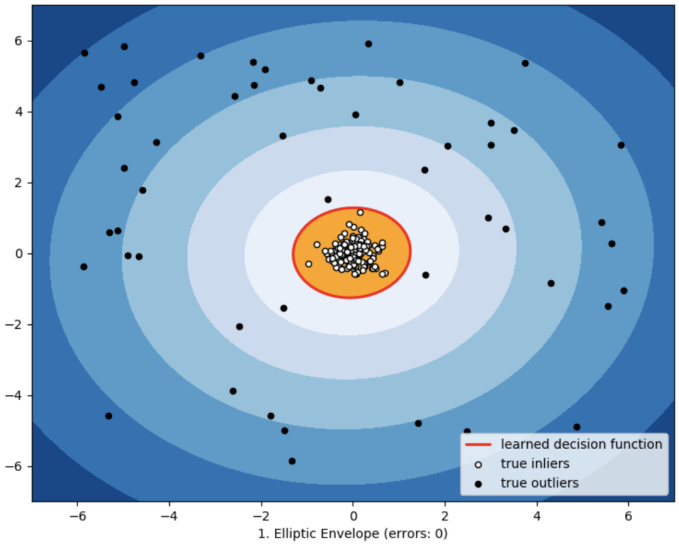
Abb. 8 Diese Abbildung repräsentiert auf der linken Seite das Isolieren eines normalen Datenpunktes und im Gegensatz dazu auf der rechten Seite die Isolation eines anormalen Datenpunktes in weniger Schritten.

Das Erkennen von Anomalien mit dem Algorithmus Isolation Forest funktioniert nach dem Prinzip des Isolieren von anormalen Daten. Im Jahre 2008 wurde er das erste Mal in der Arbeit Isolation Forest [LTZ08] publiziert. Der Algorithmus ist ein Vertreter der Kategorie der Klassifikationsalgorithmen (siehe 3.2) und stellt einen unüberwachten Algorithmus dar. Anomalien lassen sich erkennen durch die Annahme,

dass anormale Datenpunkte schneller vom Rest der Daten getrennt werden können [HK18]. Die Trennung basiert auf Entscheidungsbäumen (decision trees). Die Entscheidungsbäume werden aufgebaut, indem ein zufälliges Feature (Attribut) ausgewählt wird und genau bei diesem Feature ein zufälliger Trennwert aus dem gesamten Wertebereich bestimmt wird. Anschließend kann an dieser Stelle eine Linie gezeichnet werden und dieser Prozess wird so lange wiederholt bis der Baum komplettiert ist. Durch den Aufbau mehrerer Bäume entsteht ein „Forest“ (Menge an Bäumen). Im Durchschnitt werden in jedem Baum mehr Schritte benötigt um normale Daten von allen anderen Daten zu trennen, während Anomalien in der Regel in wenigen Schritten isoliert werden können. In Abbildung 8 ist beispielhaft dargestellt, wie viele Schritte für die Isolation eines normalen Datensatzes und für eine Anomalie benötigt werden.

4.3.3 Elliptic Envelope

Ein weiterer Algorithmus zur Erkennung von Anomalien ist Elliptic Envelope. Elliptic Envelope ist ein Beispiel für einen parametrisierten statistischen Algorithmus (siehe 3.4). Mit Hilfe einer hochdimensionalen Normal- oder Gauß-Verteilung [vgl. HRP+15, Seite 2] wird eine elliptische Grenze um den Großteil der Daten modelliert. Alle Daten außerhalb dieser Grenze (siehe auch Abbildung 9) werden als Anomalien eingestuft. Zur Bestimmung der Größe und Form der Ellipse wird die Methode FAST-Minimum Covariance Determinant [Alp16] benutzt.

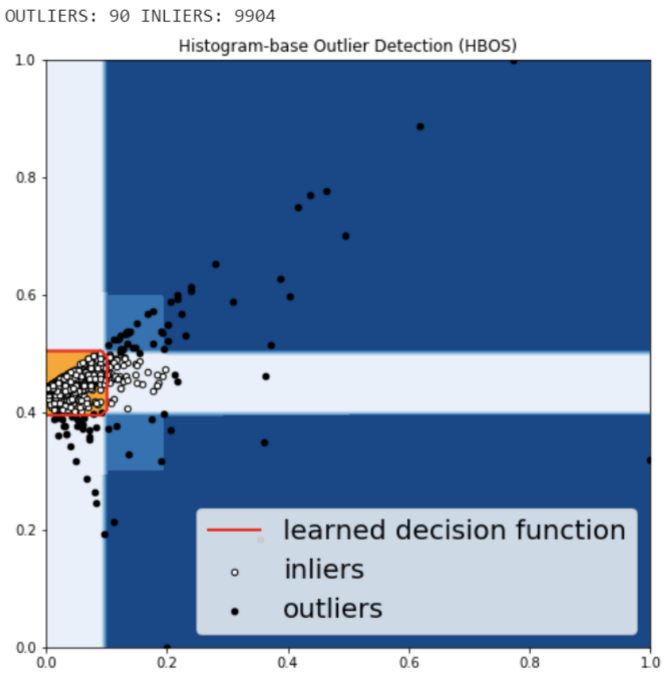


Quelle: https://stackoverflow.com/questions/50006318/i-want-to-detect-outliers-only-using-elliptic-envelope-using-python-so-how

Abb. 9 Diese Abbildung zeigt beispielhaft wie der Algorithmus Elliptic Envelope mit Hilfe einer Ellipse die normalen von den anormalen Daten trennt.

4.3.4 Histogram-based Outlier Detection

Ein unparametrisiertes statistisches Verfahren zur Anomalieerkennung ist die Erkennung von Anomalien mit Hilfe von Histogrammen. Bei der Anomalieerkennung mit Histogrammen wird für jedes einzelne Feature/Dimension ein Histogramm erstellt [vgl. GD12, Seite 3]. Dadurch ist es aber nicht möglich Beziehungen von Features untereinander abzubilden, da die Histogramme alle unabhängig voneinander sind. Die einzelnen Histogramme werden in Behälter unterteilt. Jeder neue unbekannte Datenpunkt wird einem der Behälter zugeordnet. Zum Beispiel sind die Behälter bei numerischen Features ein Teil des Wertebereiches breit oder bei kategorischen Features bilden die Kategorien die Behälter. Jeder Behälter erhält durch die Menge an Daten in diesem Behälter eine bestimmte Höhe, die auch eine Wahrscheinlichkeitsangabe darstellt, inwiefern neue Daten Ausreißer sind.



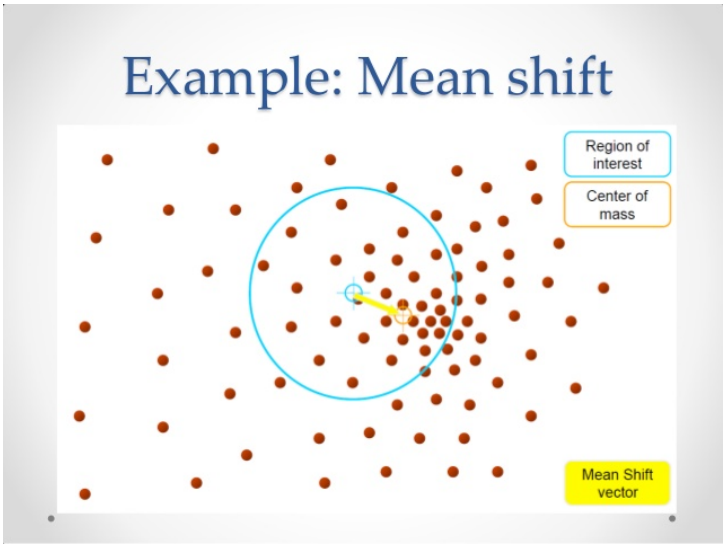
Quelle: https://towardsdatascience.com/anomaly-detection-for-dummies-15f148e559c1

Abb. 10 Diese Abbildung stellt dar, wie Daten bei der Anomalieerkennung mittels Histogrammen einegordnet werden können.

4.3.5 Mean Shift

Der Mean Shift Algorithmus ist ein Beispiel für die Kategorie der Clustering-Algorithmen (siehe 3.5). Mean Shift [CM02] basiert auf einem Sliding-Window-Verfahren.

Ausgehend von mehreren zufälligen Startpunkten wird um den Startpunkt herum ein Kreis mit einem bestimmten Radius (bandwidth of the kernel) gebildet. Der Mean Shift Algorithmus verschiebt diesen Punkt nun iterativ in Bereiche mit größerer Dichte bis zur Konvergenz des Algorithmus. Innerhalb jeder Iteration wird der Mittelpunkt so verschoben (siehe Abbildung 11), dass der neue Mittelpunkt den Mittelwert aller Punkte im Sliding-Window darstellt. Das Verschieben des Mittelpunktes wird solange fortgesetzt, bis es keine Richtung mehr gibt, in die das Sliding-Window verschoben werden kann um mehr Punkte aufzunehmen. Anschließend können in einer Nachverarbeitungsphase identische Kreise zusammengefasst werden und die Anzahl der übriggebliebenen Kreise stellt die Anzahl der Cluster dar. Der große Vorteil des Mean Shift Algorithmus ist, dass die Anzahl der Cluster vorab nicht bekannt sein muss und nur der Radius des Sliding-Windows konfiguriert werden muss.



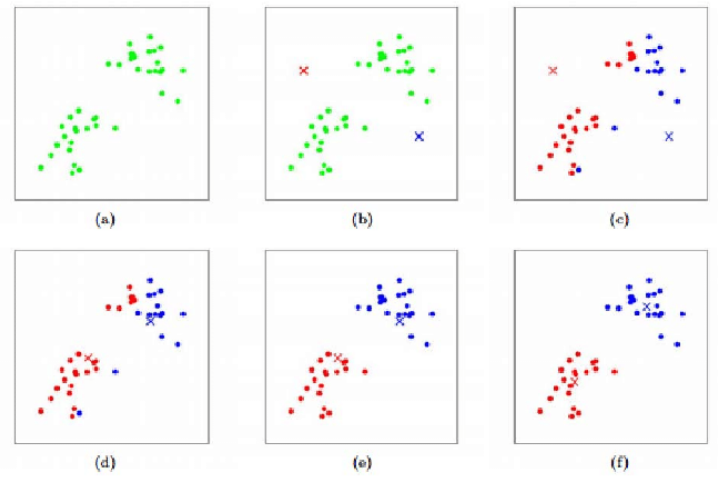
Quelle: https://www.slideshare.net/xuyangela/mean-shift-and-hierarchical-clustering

Abb. 11 Diese Abbildung stellt dar, wie beim Mean Shift Verfahren der Mittelpunkt aktualisiert wird, indem der alte Mittelpunkt auf den Mittelwert der Punkte im Sliding-Window verschoben wird.

4.3.6 K-Means

Ein weiterer Algorithmus aus der Kategorie der Clustering Algorithmen (siehe 3.5) ist K-Means. Das Ziel des K-Means Algorithmus [WCRS01] ist es eine bestimmte Anzahl an Clustern zu bilden, in die gleiche Datenpunkte eingeordnet werden können. Die Funktionsweise des Algorithmus ist in Abbildung 12 genauer dargestellt. Zuerst werden zufällige Startpunkte initialisiert. Die Anzahl der Startpunkte hängt von der Anzahl der Cluster ab, die gebildet werden sollen. Im nächsten Schritt wird jeder

Datenpunkt anhand des Abstandes einem der initialen Startpunkte zugeordnet. Anschließend wird der Startpunkt auf den Mittelpunkt aller dem Startpunkt zugeordneten Punkte verschoben. Dort liegt dann das neue Clusterzentrum. Dieses Vorgehen wird solange wiederholt bis eine bestimmte Anzahl von Iterationen erreicht worden ist oder bis keine Änderungen zwischen den Gruppenzentren der Iterationen mehr auftreten.



Quelle: https://stanford.edu/ cpiech/cs221/handouts/kmeans.html

Abb. 12 In dieser Abbildung wird in sechs Schritten verdeutlicht wie K-means die Cluster innerhalb der Daten ermittelt. Im ersten Schritt werden zufällige Startpunkte gewählt und diese anschließend in mehreren Iterationen verschoben werden, bis die fertigen Cluster gebildet worden sind.

4.4 Angriffsszenarien

Die folgende Liste gibt einen Überblick über die Szenarien, an denen die Algorithmen verglichen werden sollen. Das erste Szenario bildet dabei auch ab, inwiefern der Algorithmus falsch positive Rückmeldungen bei ausschließlich normalen Daten abgibt. Drei weitere Szenarien betrachten eine Veränderung der Sendefrequenz. Im Paper **Entropy-based anomaly detection for in-vehicle networks** [vgl. MA11, Seite 4] wird die Erhöhung der Frequenz als ein häufiges und bekanntes Merkmal bei Angriffen eingestuft. Ein weiteres Szenario soll einen Offset, wie er bei der Kompromittierung einer ECU entstehen könnte, simulieren. Das letzte Szenario betrachtet Interferenzen mit anderer Kommunikation. Es wird untersucht, ob „Flooding“ anderer Nachrichtenstreams auch mit der Anomalieerkennung sichtbar ist.

1. Erkennung normaler Daten
2. Frequenz * 10 (deutlich erhöhte Frequenz)
3. Frequenz * 1.5 (leicht erhöhte Frequenz)

4. Frequenz * 0.5 (halbierte Frequenz)
5. Nur Offset bei Angriff
6. Interferenzen mit anderer Kommunikation

5 Ergebnisse des Vergleiches und Evaluierung

Diese Kapitel zeigt die Ergebnisse des Vergleiches und stellt die Algorithmen einander gegenüber. Dabei stellt das Kapitel zuerst die Trainingsdaten dar und anschließend folgen die Angriffsszenarien nacheinander.

5.1 Abbildung der Trainingsdaten

Die Ergebnisse des insgesamt 480 s dauernden Trainings sind in Abbildung 13 abgebildet. Es ist eine Gesamtzahl von ca. 1200 Datenpunkten sichtbar. Dabei stellt die X-Achse den Jitter in einem Intervall dar und die Y-Achse die Bandbreite. Bei den Beschriftungen der Achsen handelt es sich um skalierte Werte, die für die Algorithmen zur Anomalieerkennung notwendig waren. Weiterhin auffällig ist ein einzelner Datenpunkt in der rechten oberen Ecke, der eine deutlich höhere Bandbreite aufweist. Dieser Datenpunkt ist ein fehlerhafter Startwert (der erste Wert in der OMNeT++ Simulation). Er wurde für die weiteren Versuche aussortiert und wird auch bei den Testdaten übersprungen.

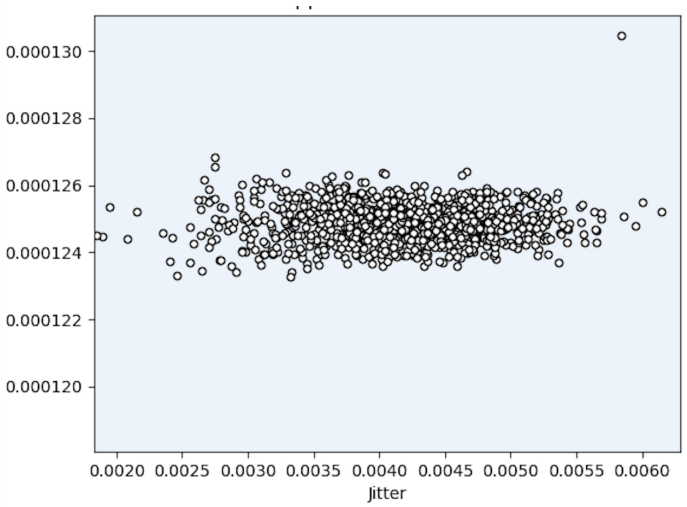


Abb. 13 Diese Abbildung stellt die Trainingsdaten dar, die die Basis für das Training aller Algorithmen bilden.

5.2 Erkennung normaler Daten

Beim ersten Angriffsszenario erfolgt in der Simulation kein Angriff. Das Ziel dieses Szenarios ist es, zu erkennen welche Konfigurationen von Algorithmen viele falsch positive Rückmeldungen geben und damit ungeeignet für weitere Untersuchungen sind. Tabelle 2 stellt die Ergebnisse des Versuches dar. Dabei enthält die Spalte „Richtig“ die Anzahl der korrekt als normale Daten erkannten Punkte. Die Spalte „Fehler“ gibt an, wie viele der normalen Daten fälschlicherweise als Anomalien eingestuft worden sind. Weiterhin stellt die Spalte „Contamination“ den Wert des Contamination Parameters dar, der das Verhältnis an Ausreißern im Datensatz angibt. Die Spalte „BorderEnlargementFactor“ bildet einen Faktor ab, der den maximalen Abstand zum Cluster repräsentiert. Der K-Means Algorithmus enthält zusätzlich zum Namen noch eine Angabe zu der Anzahl an Clustern, die gebildet werden sollen. Zum Beispiel KM2 für 2 Cluster.

Algorithmus	Contamination	BorderEnlarge-mentFactor	Richtig	Fehler
SVM	0.5	-	46	54
SVM	0.4	-	60	40
SVM	0.3	-	**100**	0
SVM	0.1	-	2	98
SVM	0	-	0	100
IF	0.3	-	72	28
IF	0.1	-	90	10
IF	0.01	-	99	1
IF	0	-	**100**	0
EE	0.01	-	99	1
EE	0	-	**100**	0
HBO	0.1	-	90	10
HBO	0.01	-	**100**	0
MS	-	1.1	**100**	0
MS	-	0.8	**100**	0
MS	-	0.6	96	4
KM1	-	0.8	96	4
KM2	-	0.8	96	4
KM3	-	0.8	96	4

Tabelle 2 zeigt die Ergebnisse der Algorithmen bei der Erkennung von Anomalien in komplett normalen Daten.

Das erste Szenario zeigt, dass es für fünf der sechs Algorithmen mindestens eine Konfiguration gibt mit der alle normalen Daten korrekt als normal eingestuft werden (siehe Tabelle 2). Alle Konfigurationen, die mehr als eine Anomalie erkannt haben, werden in den folgenden Versuchen nicht mehr berücksichtigt. Die folgenden Abbildungen zeigen als Beispiel einmal ein Ergebnis (siehe Abbildung 14) ohne Fehler, ein Ergebnis mit einem Ausreißer (siehe Abbildung 15) und ein Ergebnis mit vielen Ausreißern (siehe Abbildung 16).

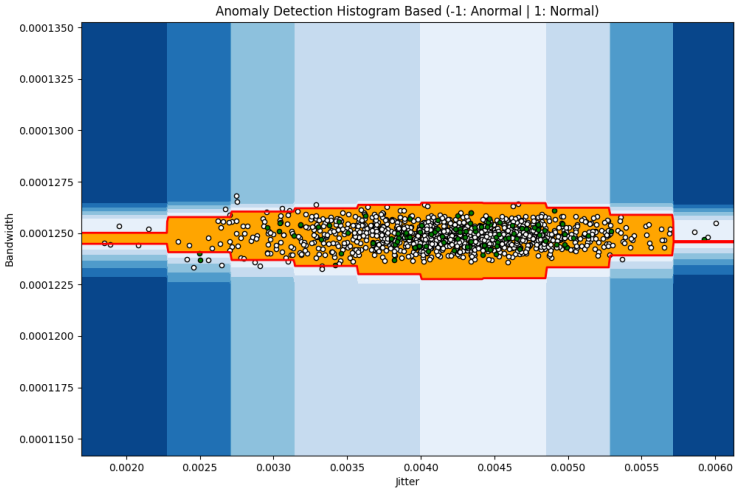


Abb. 14 Die Abbildung zeigt die Erkennung von normalen Daten mit dem HBO (Histogram-based Outlier Detection) Algorithmus. Dabei stuft der Algorithmus alle Datenpunkte korrekt als normal ein.

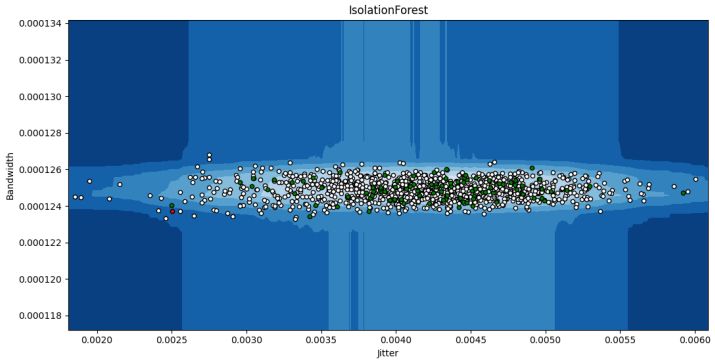


Abb. 15 Diese Abbildung stellt die Anomalieerkennung mit dem Isolation Forest Algorithmus dar. Ein Datenpunkt wird fälschlicherweise als Anomalie eingeordnet.

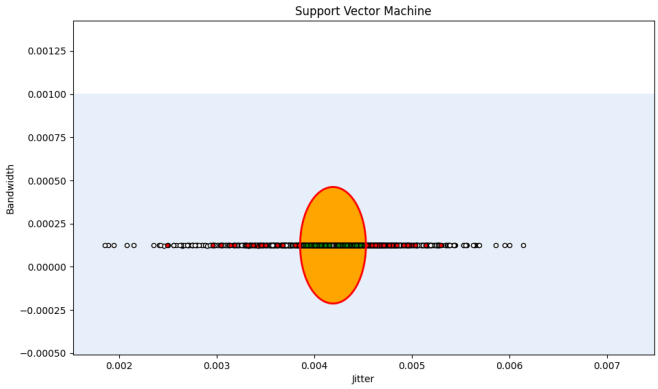


Abb. 16 Diese Abbildung zeigt die Anomalieerkennung von Support Vector Machines bei normalen Daten mit einem Contamination Parameter von 0.5. Dieses Beispiel enthält zu viele falsch positive Rückmeldungen.

5.3 Frequenz * 10 (deutlich erhöhte Frequenz)

Das Ziel des zweiten Angriffsszenarios ist zuschauen, welche der Algorithmen in der Lage sind offensichtliche Angriffe zu erkennen. Dabei wird die Frequenz der Pakete nach einer kurzen Anlaufzeit um den Faktor 10 erhöht. Die Ergebnisse des Versuches sind in der Tabelle 3 aufgelistet. Jeder der Algorithmen konnte den Angriff erfolgreich erkennen. Die Anzahl der normalen Daten und Anomalien ist in der Spalte „Normal/Ausreißer“ abgebildet. Da der Angriff nicht direkt mit dem Start der Simulation begonnen hat, ist es korrekt, dass alle Algorithmen mindestens 2-3 normale Daten erkannt haben.

Algorithmus	Contamination	BorderEnlarge-mentFactor	Angriff erkannt	Normal/Ausreißer
SVM	0.3	-	Ja	12/15
IF	0.01	-	Ja	2/25
IF	0	-	Ja	10/17
EE	0.01	-	Ja	2/25
EE	0	-	Ja	2/25
HBO	0.01	-	Ja	2/25
MS	-	1.1	Ja	16/11
MS	-	0.8	Ja	7/20
KM1	-	0.8	Ja	7/20
KM2	-	0.8	Ja	7/20
KM3	-	0.8	Ja	3/24

Tabelle 3 zeigt die Ergebnisse der Algorithmen bei der Erkennung von Anomalien im Angriffsszenario 2. Dabei erfolgt ein Angriff, der die Frequenz der Pakete um den Faktor 10 erhöht.

Die Abbildung 17 stellt beispielhaft das Ergebnis des K-Means Algorithmus dar. Dabei ist zu erkennen, dass die Ausreißer in erhöhter Position (höhere Bandbreite) verteilt um die Grenze des Clusters liegen. Daher stuft der Algorithmus einige der Daten fälschlicherweise als normale Daten ein. Eine weitere Besonderheit ist die vereinzelte Anomalie auf der linken Seite der Abbildung. Dieser Datenpunkt weicht ab, da der Angriff im ersten Intervall versetzt gesendet wird. Deshalb ist der Wert des Jitters größer und deutlich als Anomalie identifizierbar. Dieser Wert wird in den folgenden Versuchen übersprungen, da Angreifer den Angriff auch zum exakt richtigen Zeitpunkt starten könnten (siehe Kapitel 5.6). In diesem Fall ist der Jitterwert nicht auffällig.

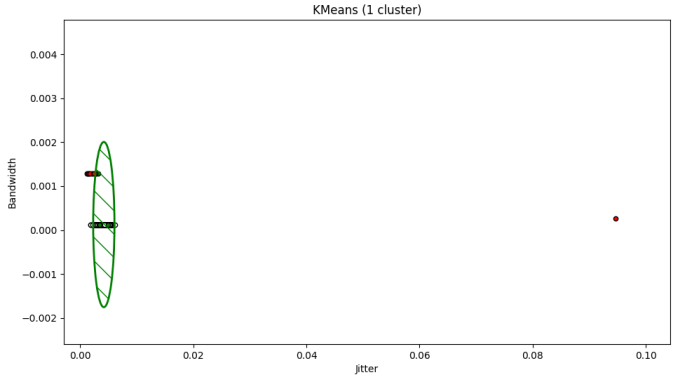


Abb. 17 Diese Abbildung zeigt die Anomalieerkennung mit dem K-Means Algorithmus. Bei diesem Angriffsszenario wurde die Frequenz der Pakete verzehnfacht.

5.4 Frequenz * 1.5 (leicht erhöhte Frequenz)

Das Ziel des dritten Angriffsszenarios ist ein Angriff mit einer leicht erhöhten Frequenz zu erkennen. Die Sendefrequenz der Pakete ist um den Faktor 1.5 beschleunigt. Die Ergebnisse des Versuches sind in der Tabelle 4 abgebildet. Der Mean Shift Algorithmus war mit einem BorderEnlargementFactor von 1.1 nicht in der Lage diesen Angriff zu identifizieren. Alle anderen Algorithmen konnten mindestens einen anormalen Datenpunkt feststellen. Das optimale Ergebnis in diesem Versuch wäre, wenn ein Algorithmus nur Anomalien und keinen einzigen normalen Datenpunkt gefunden hätte. Abbildung 18 zeigt ein Beispiel für dieses Szenario an dem Isolation Forest Algorithmus. In dem Beispiel ist zu erkennen, dass die anormalen Datenpunkte eine deutliche höhere Bandbreite aufweisen und der Jitter im Durchschnitt leicht geringer ist. Der Algorithmus hat trotzdem 3 Datenpunkte als normal eingestuft.

Algorithmus	Contamination	BorderEnlarge-mentFactor	Angriff erkannt	Normal/Ausreißer
SVM	0.3	-	Ja	28/1
IF	0.01	-	Ja	3/26
IF	0	-	Ja	19/10
EE	0.01	-	Ja	0/29
EE	0	-	Ja	0/29
HBO	0.01	-	Ja	0/29
MS	-	1.1	Nein	29/0
MS	-	0.8	Ja	21/8
KM1	-	0.8	Ja	21/8
KM2	-	0.8	Ja	24/5
KM3	-	0.8	Ja	25/4

Tabelle 4 zeigt die Ergebnisse der Algorithmen bei der Erkennung von Anomalien im Angriffsszenario 3. Dabei erfolgt ein Angriff, der die Frequenz der Pakete um den Faktor 1.5 erhöht.

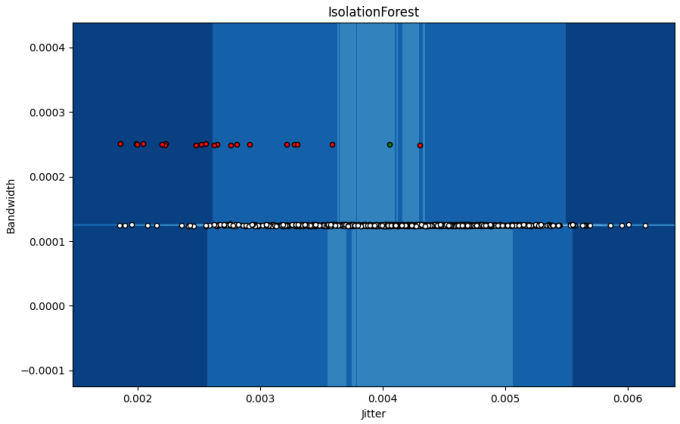


Abb. 18 Diese Abbildung zeigt die Anomalieerkennung mit dem Isolation Forest Algorithmus. Bei diesem Angriffsszenario wurde die Frequenz der Pakete um den Faktor 1.5 erhöht.

5.5 Frequenz * 0.5 (halbierte Frequenz)

Das dritte Angriffsszenario halbiert die Sendefrequenz der Pakete. Die Ergebnisse des Versuches sind in der Tabelle 5 zu sehen. Wie im vorherigen Szenario 3 mit leicht erhöhter Frequenz war nur der Mean Shift Algorithmus mit BorderEnlargementFactor 1.1 nicht in der Lage den Angriff zu erkennen. Die weiteren Algorithmen haben alle den Angriff entdeckt. Das optimale Ergebnis wäre in diesem Versuch, wenn ein Algorithmus nur Anomalien erkannt hätte und keine normalen Daten. Abbildung 19 demonstriert am Beispiel des K-Means Algorithmus die Lage der Anomalien. Im Gegensatz zur erhöhten Frequenz der Pakete ist diesmal zu erkennen, dass die Bandbreite kleiner geworden ist und der durchschnittliche Jitterwert angestiegen ist.

Algorithmus	Contamination	BorderEnlarge-mentFactor	Angriff erkannt	Normal/Ausreißer
SVM	0.3	-	Ja	22/5
IF	0.01	-	Ja	6/21
IF	0	-	Ja	14/13
EE	0.01	-	Ja	0/27
EE	0	-	Ja	0/27
HBO	0.01	-	Ja	0/27
MS	-	1.1	Nein	27/0
MS	-	0.8	Ja	18/9
KM1	-	0.8	Ja	15/12
KM2	-	0.8	Ja	13/14
KM3	-	0.8	Ja	13/14

Tabelle 5 zeigt die Ergebnisse der Algorithmen bei der Erkennung von Anomalien im Angriffsszenario 4. Dabei erfolgt ein Angriff, der die Frequenz der Pakete halbiert.

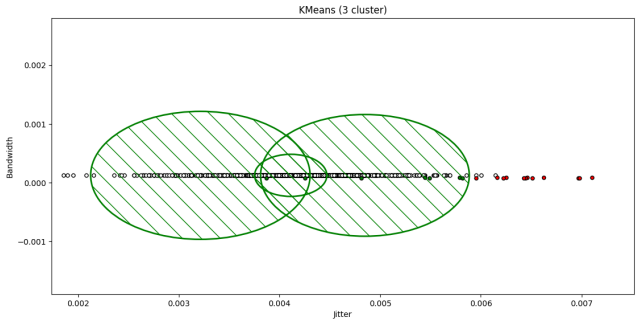


Abb. 19 Diese Abbildung zeigt die Anomalieerkennung mit dem K-Means Algorithmus mit 3 gebildeten Clustern (KM3). Bei diesem Angriffsszenario wurde die Frequenz der Pakete halbiert.

5.6 Nur Offset bei Angriff

Das Ziel des fünften Angriffsszenarios ist es zu schauen, ob die Algorithmen in der Lage sind einen Angriff, der mit gleicher Frequenz weitersendet, nur an einem vorhandenen Offset zu entdecken. Die Tabelle 6 zeigt die Ergebnisse des Versuches. Jeder der Algorithmen war in der Lage diesen Angriff zu entdecken. Bei einem Angriff, der einen Offset im Gegensatz zur normalen Sendefrequenz enthält, kann ein auffälliger Jitterwert entstehen. Ein Beispiel für diesen Jitterwert ist in Abbildung 20 dargestellt. Je nach Offset/Startzeitpunkt des Angriffes liegt der Jitterwert irgendwo im Bereich der Sendefrequenz der Pakete. Der erste Jitterwert bei einem Angriff kann sowohl noch größer als auch kleiner als der durchschnittliche Jitterwert sein oder sogar exakt im Bereich des durchschnittlichen Jitters liegen. Zum Beispiel bei einem Start des Angriffes nach 2,015 s liegt der Jitterwert bei 1,701 ms während der Jitterwert bei einem Start des Angriffes nach 2 s bei 13,677 ms liegt. Dabei befindet sich der Jitterwert von 1,701 ms in Mitten der anderen normalen Jitterwerte. Ein Angriff zu diesem Zeitpunkt wäre sehr schwer zu identifizieren.

Algorithmus	Contamination	BorderEnlarge-mentFactor	Angriff erkannt	Normal/Ausreißer
SVM	0.3	-	Ja	18/1
IF	0.01	-	Ja	18/1
IF	0	-	Ja	18/1
EE	0.01	-	Ja	18/1
EE	0	-	Ja	18/1
HBO	0.01	-	Ja	18/1
MS	-	1.1	Ja	18/1
MS	-	0.8	Ja	17/2
KM1	-	0.8	Ja	17/2
KM2	-	0.8	Ja	17/2
KM3	-	0.8	Ja	17/2

Tabelle 6 zeigt die Ergebnisse der Algorithmen bei der Erkennung von Anomalien im Angriffsszenario 5. Dabei erfolgt ein Angriff, der die Frequenz der Pakete beibehält und nur einen Offset bei dem Senden der Pakete enthält.

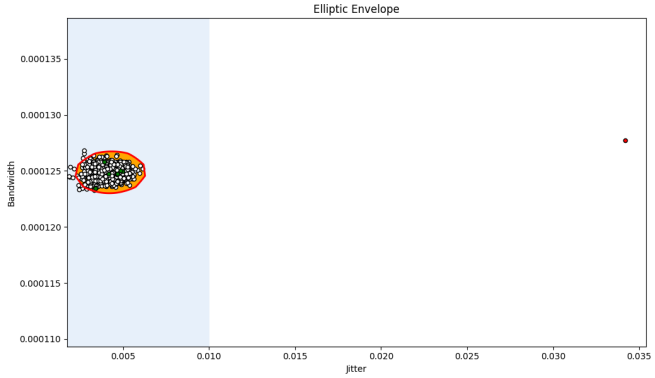


Abb. 20 Diese Abbildung zeigt die Anomalieerkennung mit dem Elliptic Envelope Algorithmus. Bei diesem Angriffsszenario wurde die Frequenz der Pakete beibehalten und der Angriff enthält nur einen Offset beim Senden der Pakete.

5.7 Interferenzen mit anderer Kommunikation

Dieses Angriffsszenario beschreibt, ob es möglich ist, DoS-Angriffe (Denial of Service) auch in einem Paketstrom zu beobachten, der nicht angegriffen wird. Dazu wurde in einem Fall die Sendefrequenz einer ECU am selbem Gateway und im anderen Fall die Sendefrequenz einer ECU in gleichen CAN-Bus System um einen sehr großen Wert erhöht. In beide Fällen konnten keine Auffälligkeiten beobachtet werden. Solange die Priorität der Pakete (auf Basis des Identifiers) nicht die des beobachteten Nachrichtenstroms übersteigt, entsteht keine Verzögerung im beobachteten Nachrichtenstrom. Also ist es nicht möglich Anomalien zu erkennen. Wenn der angreifende Nachrichtenstrom eine höhere Priorität besitzen würde, wäre es vermutlich möglich Verzögerungen im beobachteten Nachrichtenstrom zu identifizieren und den Angriff damit zu entdecken.

5.8 Evaluation der Szenarien

Die Ergebnisse der durchgeführten Szenarien haben gezeigt, dass vor allem die beiden statistischen Algorithmen HBO und Elliptic Envelope Anomalien bei einem Angriff in großer Anzahl erkennen können und auch bei normalen Daten nicht viele bzw. gar keine falsch positive Rückmeldungen geben. Auch der Algorithmus Isolation Forest hat bei allen Szenarien gute Ergebnisse hervorgebracht. SVM und die beiden Clustering-Algorithmen hatten mehr Probleme bei der Erkennung von Angriffen und waren zum Teil nahe an der Grenze einen Angriff nicht zu erkennen bzw. haben Angriffe nicht erkannt. Ein Problem der beiden Clustering-Algorithmen war, dass die kreisförmige Clusterform schwierig auf die Struktur der normalen Daten abzubilden war. Auch die Bildung von mehreren Clustern war nur bedingt sinnvoll. Der SVM (OneClassSvm) Algorithmus hatte auch große Schwierigkeiten gute Grenzen zu definieren, ohne zu viele falsch positive Rückmeldungen zu geben. Eine andere bessere Konfiguration könnte möglicherweise noch bessere Ergebnisse erzielen. Ein Grund für die guten Ergebnisse der statistischen Algorithmen sind vermutlich die Daten, die mit einen normal verteilten zufälligen PeriodInaccurracy Parameter generiert wurden. Dadurch war die Annahme, dass eine statistische Verteilung zugrunde liegt korrekt.

6 Ausblick und Zusammenfassung

Die Entdeckung von Angriffen mit Anomalieerkennung ist ein mögliches Security Konzept zum Schutz des Fahrzeugnetzwerkes. Denn durch die schrittweise Öffnung des Netzes und vermehrte Kommunikation mit der Außenwelt sind Security Konzepte notwendig, die ein sicherheitskritisches System wie das Fahrzeug effektiv vor Angriffen schützen können. Diese Ausarbeitung hat grundlegende Aspekte der Anomalieerkennung erläutert und einen Überblick zu unterschiedlichen Verfahren der Anomalieerkennung gegeben. Anschließend wurde in mehreren Szenarien ein Vergleich zwischen verschiedenen Algorithmen auf Basis einer OMNeT++ Simulation umgesetzt. Dabei wurde die Anomalieerkennung in einem zentralen Switch an den Paketen einer spezifischen CAN-ID durchgeführt. Die Ergebnisse haben gezeigt, dass die statistischen Algorithmen (HBO und EE) sowie IsolationForest gute Ergebnisse erzielt haben, während die Clustering-Algorithmen und der SVM Algorithmus leichte Probleme bei der Erkennung aufgewiesen haben. Im weiteren Verlauf meines Masters (Hauptprojekt) werde ich auf Basis meiner Erkenntnisse aus dem Grundprojekt mich mit Anomalieerkennung in realer Hardware befassen und untersuchen, ob sich die Ergebnisse aus dem Grundprojekt in realer Hardware ähnlich ausschauen. Weiterhin werde ich auch mit einem Kamerastream eine andere Ebene, auf der die Entdeckung von Anomalien durchgeführt werden kann, betrachten.

Literatur

Alp16. Özlem Alpu. Using fast minimum covariance determinant estimators for factor analysis in the presence of outliers, 2016.

BBK14. M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. Network anomaly detection: Methods, systems and tools. *IEEE Communications Surveys Tutorials*, 16(1):303–336, First 2014.

CBK09. Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

CM02. D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

CMK+11. Stephen Checkoway, Damon Mccoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. Comprehensive Experimental Analyses of Automotive Attack Surfaces. *USENIX Security*, 2011.

FYM05. Ryohei Fujimaki, Takehisa Yairi, and Kazuo Machida. An approach to spacecraft anomaly detection problem using kernel feature space. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 401–410, New York, NY, USA, 2005. ACM.

GD12. Markus Goldstein and Andreas Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. 09 2012.

Gmb18. Wirecard Technologies GmbH. Data and analytics – anomalieerkennung in zeitreihen, 2018.

GTDVMFV09. P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers Security*, 28(1):18 – 28, 2009.

HK18. S. Hariri and M. C. Kind. Isolation forest for anomaly detection. *LSST Workshop 2018*, 06 2018.

HMVVDK13. Peter Hank, Steffen Müller, Ovidiu Vermesan, and Jeroen Van Den Keybus. Automotive ethernet: In-vehicle networking and smart mobility. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '13, pages 1735–1739, San Jose, CA, USA, 2013. EDA Consortium.

HRP+15. Ben Hoyle, Markus Michael Rau, Kerstin Paech, Christopher Bonnett, Stella Seitz, and Jochen Weller. Anomaly detection for machine learning redshifts applied to SDSS galaxies. *Monthly Notices of the Royal Astronomical Society*, 452(4):4183–4194, 08 2015.

JA03. Karsten Borgwardt Martin Ester Eshref Januzaj Karin Kailing Peer Kröger Jörg Sander und Matthias Schubert Johannes Aßfalg,

Christian Böhm. Kapitel 3: Klassifikation. pages 123–151, 2003.

KKSZ09. Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Loop: Local outlier probabilities. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 1649–1652, New York, NY, USA, 2009. ACM.

KNT00. Edwin M. Knorr, Raymond T. Ng, and Vladimir Tucakov. Distance-based outliers: Algorithms and applications. *The VLDB Journal*, 8(3-4):237–253, February 2000.

LHC12. H. Lim, D. Herrscher, and F. Chaari. Performance comparison of ieee 802.1q and ieee 802.1 avb in an ethernet-based in-vehicle network. In *2012 8th International Conference on Computing Technology and Information Management (NCM and ICNIT)*, volume 1, pages 1–6, 2012.

LJK00. Jorma Laurikkala, Martti Juhola, and Erna Kentala. Informal identification of outliers in medical data. *Fifth International Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, 07 2000.

LTZ08. F. T. Liu, K. M. Ting, and Z. Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008.

MA11. M. Müter and N. Asaj. Entropy-based anomaly detection for in-vehicle networks. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 1110–1115, June 2011.

Pat. Andrew Patterson. Outlier detection methods for detecting cheaters in mobile gaming.

Sec. Projekt SecVI. Security for vehicular information - research project for secure automotive network architectures.

sl. scikit learn.

SNA+13. I. Studnia, V. Nicomette, E. Alata, Y. Deswarte, M. Kaâniche, and Y. Laarouchi. Survey on security threats and protection mechanisms in embedded automotive networks. In *2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*, pages 1–12, June 2013.

WCRS01. K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. 2001.