HAW HAMBURG

SEMINARARBEIT
Wilhelm Schumacher

Modellbasierter Vergleich von Methoden zur Anomalieerkennung im Fahrzeugnetzwerk des SecVI Demonstrators

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

Betreuung durch: Prof. Dr. Franz Korf
Eingereicht am: 22. September 2021

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Inhaltsverzeichnis

Moderne Fahrzeugnetzwerke bilden die Grundlage für eine Vielzahl neuer hochentwickelter Funktionen im zukünftigen Auto (“Connected Vehicle”). Die Nutzung dieser Funktionen, wie etwa Fahrerassistenzsysteme, Infotainmentsysteme oder bis hin zum automatisierten und autonomen Fahren, führt gleichzeitig auch zu einer Öffnung des internen Netzes und zu vermehrter Kommunikation mit der Außenwelt. Dies ermöglicht die Durchführung einer Vielzahl neuer Angriffe auf das Auto, für die neue Sicherheitskonzepte entwickelt werden müssen. Ein Bestandteil des zukünftigen Security-Konzept des Autos könnte die Entdeckung von Angriffen durch die Anomalieerkennung sein. 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. Das Ziel dieser Ausarbeitung (Hauptprojekt) ist es, verschiedene Algorithmen zur Anomalieerkennung in realer Hardware innerhalb des SecVI Demonstrators eimzusetzen und miteinander zu vergleichen. Dieses Projekt baut auf der theoretischen Ausarbeitung zu den Methoden und dem Vergleichen von Basisalgorithmen zur Anomalieerkennung in einer OMNeT++ Simulationsumgebung aus dem Grundprojekt auf. Der Vergleich der Algorithmen im Netzwerk wird auf der Ebene eines CAN-Bus Streams und auf der Ebene eines Videostreams durchgeführt. In verschiedenen Angriffsszenarien wird das Verhalten und die Erkennungsrate der Algorithmen untersucht.

Keywords: Anomalieerkennung, Automotive Security, Anomaly Detection, Network Anomaly Detection, Fahrzeugnetzwerk

1 Einleitung

Die Grundlage für eine Vielzahl neuer, hochentwickelter Funktionen im zukünftigen Auto ("Connected Vehicle"), wie etwa Fahrerassistenzsysteme, Infotainmentsysteme oder bis hin zum automatisierten und autonomen Fahren bilden in Zukunft moderne Fahrzeugnetzwerke. Ein fundamentaler Bestandteil dieser Netzwerke bildet die Technologie **Ethernet** [17] als Hochgeschwindigkeits-Backbone im Auto. Bisher wurden im Netzwerk vor allem Systembusse wie CAN, LIN, MOST und FlexRay als Kommunikationsmedien verwendet [8], die nun schrittweise durch Ethernet-Netzwerke ersetzt werden. Viele Funktionen erfordern auch eine Öffnung des internen Fahrzeugnetzwerkes nach außen und erweitern somit die Angriffsoberfläche (attack surface) des Fahrzeuges [4]. Gleichzeitig ermöglicht dies auch die Durchführung einer Vielzahl neuer Angriffe auf das Auto und führt dadurch zu einer erhöhten Verwundbarkeit der Informationssicherheit (Integrität, Vertraulichkeit, Verfügbarkeit) im internen Netzwerk des Fahrzeuges [13].

Zum Schutz des Autos durch diese Angriffe müssen neue Sicherheitskonzepte und -mechanismen eingeführt werden. Ein Bestandteil des zukünftigen Sicherheitskonzept des Autos könnte die Entdeckung von Angriffen durch die Anomalieerkennung sein. 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 der Art der Daten, der Verfügbarkeit von gelabelten Trainingsdaten, den Typen von Anomalien und dem gewünschten Outputformat [3]. Das Netzwerk des Fahrzeugs ermöglicht die Durchführung der Anomalieerkennung auf unterschiedlichen Ebenen (Layern). Zum Beispiel betrachtet [6] die Erkennung von Anomalien innerhalb des CAN-Bus-System oder [18] führt die Anomalieerkennung mit den Daten aus der Sensorik durch. Weiterhin könnte sich der Einsatz von Anomalieerkennung in der automotiven Domäne 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. Trotz allem stellt die Anomalieerkennung nur einen Teil eines gut funktionierenden Sicherheitskonzeptes dar und muss durch weitere Elemente wie zum Beispiel SDN (Software-defined Networking), Cyber Defense Center in der Cloud oder V2X Application-Level Gateways ergänzt werden.

Dieses Hauptprojekt 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. Dabei baut es auf der theoretischen Ausarbeitung der Kategorien von Methoden zur Anomalieerkennung und dem Vergleich von Algorithmen innerhalb des Simulations-Framework OMNeT++ (Objective Modular Network Testbed in C++) aus dem Grundprojekt auf. Das Ziel dieser Ausarbeitung (Hauptprojekt) ist es, verschiedene Algorithmen zur Anomalieerkennung in realer Hardware innerhalb des SecVI Demonstrators einzusetzen und miteinander zu vergleichen. Der Vergleich der Algorithmen im Netzwerk wird auf der Ebene eines CAN-Bus Streams und auf der Ebene eines Videostreams durchgeführt. In verschiedenen Angriffsszenarien wird das Verhalten und die Erkennungsrate der Algorithmen untersucht. Die Arbeit findet innerhalb des SecVI Forschungsprojekt [16] 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 beginnt mit dem Kapitel **2 Grundlagen**, dass ein paar fundamentale Konzepte zur Anomalieerkennung erläutert und in der Arbeit verwendete Algorithmen hervorhebt. Das Kapitel **3 Aufbau des SecVI Demonstrator** gibt einen Überblick über die Versuchsumgebung SecVI Demonstrator und beschreibt alle wichtigen Elemente des Netwerkes. Anschließend betrachten die Kapitel **4 Anomalieerkennung in einem CAN-Bus Stream** und **5 Anomalieerkennung in einem Videostream** den Vergleich von 6 Algorithmen in verschiedenen Angriffsszenarien innerhalb eines CAN-Bus Stream und in einem Videostream. Beide Kapitel umfassen dabei jeweils die Versuchsdurchführung, die Ergebnisse und eine Einordnung der Algorithmen. Abschließend fasst das Kapitel **6 Ausblick und Zusammenfassung** die wichtigsten Punkte der Ausarbeitung zusammen, betrachtet noch offene Punkte und gitb einen Ausblick auf die folgende Masterarbeit.

2 Grundlagen

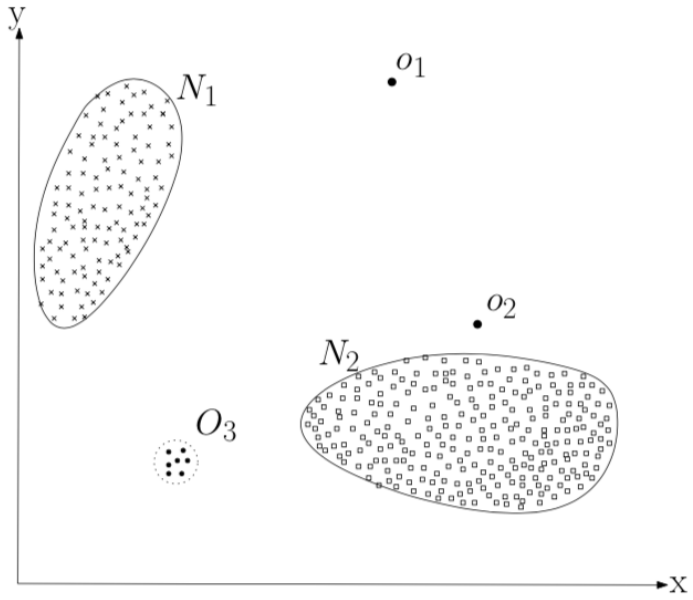
Das Kapitel Grundlagen beschreibt kurz die Funktionsweise der Anomalieerkennung in **2.1 Grundlagen der Anomalieerkennung** und hebt anschließend die fundamentalsten Konzepte der Anomalieerkennung in **2.2 Aspekte der Anomalieerkennung** hervor. Das dritte Teil des Kapitels **2.3 Algorithmen der Anomalieerkennung** erläutert kurz die Funktionsweise der 6 Algorithmen, die in den folgenden Kapiteln zur Anomalieerkennung verwendet werden.

2.1 Grundlagen der Anomalieerkennung

Die Anomalieerkennung hat als Aufgabe unregelmäßige Muster innerhalb von normalen Daten festzustellen [3]. 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)** bezeichnet. Ein konkretes Beispiel für eine Anomalie ist in Abbildung 1 dargestellt. Hier können innerhalb eines zweidimensionalen Datensatz sowohl normale Bereiche (N1 und N2) als auch Anomalien (O1, O2 und O3), 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.

2.2 Aspekte der Anomalieerkennung

Bei der Anomalieerkennung wird beim Einordnen der Datensätze zwischen **False Positive** und **False Negative** unterschieden. Ein False Positive liegt vor, wenn ein Datensatz zu Unrecht als Ausreißer eingestuft wird (falscher Alarm) und ein False Negative liegt vor, wenn ein Datensatz als normal eingestuft wird, obwohl es sich dabei um einen Ausreißer handelt. Ein gutes System zur Anomalieerkennung zeichnet sich dadurch aus, dass es sowohl wenige bis zu keine False Positive erkennt und trotzdem keine Ausreißer übersieht (False Negative).
Ein weiterer wichtiger Aspekt der Anomalieerkennung sind die drei unterschiedlichen Lernmethoden [1]. Beim **überwachten Lernen** erhält der Algorithmus einen Datensatz, der die Datenpunkte bereits als normal oder anormal makiert hat. Auf Basis dieses



Quelle: https://dl.acm.org/doi/10.1145/1541880.15418822 [3]

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

Datensatzes wird das System trainiert, um anschließend 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 ein neuer Datenpunkt anormal ist oder nicht. Beim **unüberwachten Lernen** lernt der Algotithmus selbständig Muster und Zusammenhänge zu erkennen, ohne dass ein bearbeiteter Datensatz vorgegeben wird. Welche Methode angewendet werden kann, hängt hautsächlich davon ab, ob gelabelte Trainingsdaten oder nur ungelabelte Trainingsdaten zur Verfügung stehen.

2.3 Algorithmen der Anomalieerkennung

In den später folgenden Versuchen werden 6 Algorithmen aus unterschiedlichen Kategorien zur Anomalieerkennung verwendet.
IsolationForest [2] basiert auf dem Prinzip des Isolieren von anormalen Daten und ist der Kategorie der Klassifikationsalgorithmen zuzuordnen. Der Algorithmus erstellt mehrere unterschiedliche Entscheidungsbäume (decision trees), die zusammen einen „Forest“

(Menge an Bäumen) bilden. Ein einzelner Baum wird aufgebaut, indem bis zur Kompletierung immer wieder ein zufälliger Trennwert aus einer zufälligen Feature ausgewählt wird. Der Baum ist komplettiert bis entweder jeder Datenpunkt isoliert dargestellt wird oder eine durchschnittliche Tiefe erreicht wird. Anomalien werden erkannt, da diese in der Regel in wenigen Schritten isoliert werden können.

Support Vector Machines [14] basiert auf dem Prinzip des Trennen von zwei verschiedenen Klassen voneinander und ist der Kategorie der Klassifikationsalgorithmen zuzuordnen. In seiner Grundform bildet der Algorithmus durch das „Maximum Margin Hyperplane“ Verfahren eine Hyperebene zur Trennung der Daten. Das Verfahren bestimmt die Trennlinie (Hyperplane) so, dass der Abstand zu den Punkten aus beiden Mengen maximal ist. Zur Anomalieerkennung mit SVM wird eine leicht modifizierte Variante mit One-Class SVM [15] benutzt. One-Class SVM trennt eine Klasse von Daten ab, indem das Volumen eines Hyperballs um die Trainingsdaten minimiert wird.

Elliptic Envelope [9] basiert auf der Modellierung einer elliptischen Grenze um den Großteil der Daten und ist der Kategorie der parametrisierten statistischen Algorithmen zuzuordnen. Die elliptische Grenze wird mit Hilfe einer hochdimensionalen Normal- oder Gauß-Verteilung erstellt und alle Daten außerhalb dieser Grenze werden als Anomalien eingestuft.

Histogram-based Outlier Detection [7] basiert auf der Erkennung von Anomalien mit Hilfe von Histogrammen und ist der Kategorie der unparametrisierten statistischen Algorithmen zuzuordnen. Bei der Anomalieerkennung mit Histogrammen wird für jedes einzelne Feature ein Histogramm erstellt und die einzelnen Histogramme werden in Behälter unterteilt. Jeder Behälter erhält durch die Menge an Daten in diesem Behälter eine bestimmte Höhe, die auch eine Wahrscheinlichkeitsangabe darstellt, inwiefern neue Daten Ausreißer sind.

Mean Shift [5] basiert auf der Bildung von Clustern über das Sliding-Window-Verfahren und ist ein Beispiel für die Kategorie der Clustering-Algorithmen. Ausgehend von mehreren zufälligen Startpunkten iteriert der Algorithmus durch Verschieben des Cluster-Mittelpunktes in Bereiche mit größerer Dichte bis zur Konvergenz des Algorithmus. Nach Abschluss werden identische Kreise zusammengefasst. Der Mean Shift Algorithmus bestimmt die Anzahl der Cluster selbst.

K-Means [11] hat als Ziel eine vorher festgelegte Anzahl an Clustern zu bilden, in die gleiche Datenpunkte eingeordnet werden können und ist der Kategorie der Clustering-Algorithmen zuzuordnen. In mehreren Iterationen werden Datenpunkte, ausgehend von zufälligen Startpunkten, wiederholt einem Clusterzentrum zugeordnet und anschließend wird das Clusterzentrum jedes Mal verschoben. Dieses Vorgehen wird so lange wiederholt

bis eine bestimmte Anzahl von Iterationen erreicht worden ist oder bis keine Änderungen zwischen den Zentren der Iterationen mehr auftreten.

3 Aufbau des SecVI Demonstrators

Das Kapitel Aufbau des SecVI Demonstrator hat als Ziel die Versuchsumgebung für die später folgenden Experimente darzustellen. Dazu erklärt das erste Kapitel **3.1 SecVI Demonstrator** erstmal allgemein, was unter dem Begiff SecVI Demonstrator zu verstehen ist. Anschließend erläutert das Kapitel **3.2 Topologie** die Komponenten und den Aufbau des Netzwerkes. Das letzte Kapitel **3.3 Durchführung der Anomalieerkennung** beschreibt konkret, wie die Anomalieerkennung technisch durchgeführt wird.

3.1 SecVI Demonstrator

Beim SecVI Demonstrator handelt es sich um einen Tischaufbau aus echter Hardware, der als Ziel hat ein modernes internes Fahrzeugnetzwerk eines Seat Atecas zu modellieren [12]. Der Tischaufbau wurde im Rahmen des SecVI Projektes entwickelt und wird bereits für Vielzahl weiterer Versuche aus den Bereichen SDN (Software-defined Networking), Anomalieerkennung, Monitoring (Cyber Defense Center) und allgemein zum Schutz des Netzwerkes verwendet. Dabei basiert das Fahrzeugnetzwerk vor allem auf der Ethernet Technologie und verwendet diese für das interne Backbone.

3.2 Topologie

Abbildung 2 gibt einen Überblick über die Topologie des SecVI Demonstrators. Im Zentrum der Abbildung befindet sich ein auf OpenFlow basierender Edgecore 24 Port **SDN Switch**. Dieser stellt das Ethernet Backbone des Netzes dar. Umgeben ist der Switch in einer Zonenarchitektur von **4 Zonencontrollern** (ZC_FR, ZC_RR, ZC_FL und ZC_RL), die jeweils eine Verbindung zu den CAN Bus-Systemen bilden. Die CAN Bus-Systeme werden von dem core-laptop-2 emuliert, der in Dauerschleife auf Zonen aufgeteilte Traces auf die 4 CAN Busse CAN 1-4 aufspielt. Da die CAN Busse funktional getrennt sind, werden Nachrichten über das zentrale Ethernet Backbone an die anderen CAN Bus-Systeme versendet. Weiterhin befindet sich im Netzwerk noch eine hochauflösende **Kamera**, die über eine ethernetbasierte Kommunikation Bilder an einen **Monitor**

verschickt. Mit Hilfe eines **SDN-Controllers**, der mit dem zentralen Switch verbunden ist, kann das Weiterleiten von Pakten mittels Durchflusstabelle (flow table) gesteuert werden. Für die Anomalieerkennung enthält das Netzwerk zwei **NUC** (Next Unit of Computing) Geräte, die an dem zentralen Switch angeschlossen sind. Auch noch befinden sich im Netzwerk ein ACDC_EDGE (Automotive Cloud Defense Center) und ein HPC_I (High-Performance Controller Cluster).

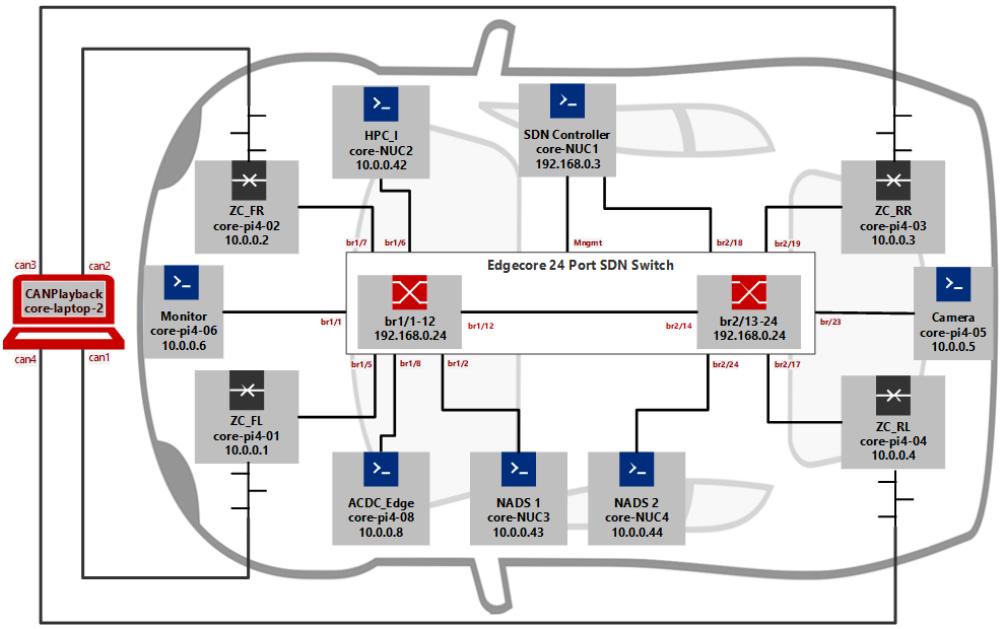


Abbildung 2: Abbildung 2 stellt die Topologie des SecVI Demonstrators dar. Das Netzwerk ist in eine Ethernet Zonenarchitektur aufgeteilt.

3.3 Durchführung der Anomalieerkennung

Die Anomalieerkennung für die folgenden Versuche wird auf dem NUC Gerät, das im Netzwerk als NADS 1 (Network Anomaly Detection System) bezeichnet wird, durchgeführt. Der NUC ist mit dem Switch über einen Port verbunden, auf den der komplette eingehende Datenverkehr des Switches gespiegelt wird. Auf dem NUC läuft eine Python Anwendung, die mit Hilfe von PyShark [10] via Socket bestimmten Netzwerkverkehr aufzeichnet und diesen an ein Modul zur Anomalieerkennung weiterleitet. Ein großer Vorteil von PyShark gegenüber anderen Modulen zum Aufzeichnen von Netzwerkpaketen ist, dass PyShark nur ein Wrapper der Kommandozeilenanwendung Tshark ist und die notwendige Präzision in den Zeitstempeln der Pakete bietet. Aus den aufgezeichneten Netz-

werkdaten erstellt die Python Anwendung dann die folgenden 4 Metriken **Bandbreite, Paketabstand, Paketgröße und Jitter** (Differenz des größten und kleinsten Paketabstandes) in Intervallen von festgelegter Länge. Weiterhin ist es für die Algorithmen erforderlich in einer **Trainingsphase** zuerst das normale Verhalten des Netzwerkverkehrs zu lernen. Die Trainingsphase wurde für alle Algorithmen auf eine Dauer von **30000 Sekunden** mit Intervallen von **1 Sekunde** festgelegt, um in angemessener Trainingszeit genug Repräsentanten des Verhaltens zu erhalten.

4 Anomalieerkennung im CAN-Bus Stream

Das Kapitel Anomalieerkennung im CAN-Bus Stream erläutert die Durchführung sowie die Ergebnisse der Anomalieerkennung in verschiedenen Angriffsszenarien im CAN-Bus Stream und ordnet am Ende des Kapitels die Ergebnisse ein. Zuerst beschreibt **4.1 Training** den Ablauf des Trainings und **4.2 False Positve Test an normalen Daten** betrachtet die Anzahl an False Positives bei Erkennung von normalen Daten, um bestimmte Konfigurationen der Algorithmen für die folgenden Versuche auszusortieren. Anschließend schaut das Kapitel **4.3 Offensichtlicher Angriff**, ob alle Algorithmen in der Lage sind einen offensichtlichen Angriff zuerkennen und dann folgen in **4.4 Packet Injection**, **4.5 Packet Elimination** und **4.6 Packet Modification** spezifische Angriffsszenarien zum Vergleichen der Algorithmen. Das letzte Kapitel **4.7 Weitere Angriffsmöglichkeiten und Bewertung der Algorithmen** diskutiert weitere Angriffsmöglichkeiten, die von den Algorithmen möglicherweise nicht erkannt werden können und bewertet die Algorithmen.

4.1 Training

Die folgende **Abbildung 3** zeigt die Ergebnisse des durchgeführten Trainings. Das Training wurde über eine Gesamtdauer von 30000s in Intervallen von 1s durchgeführt. Deshalb ist eine Gesamtmenge von 30000 weißen Punkten in der Abbildung erkennbar. Die Pakete des CAN-Bus werden in Zyklen von 80ms gesendet und daher empfängt die Anwendung zur Anomalieerkennung im Durchschnitt 12-13 Pakete in einem Intervall. Die empfangenen Frames haben eine Paketgröße von 72 Bytes. Auf der Y-Achse wird ein von 0 bis 1 skalierter Wert der Bandbreite in einem Intervall dargestellt. Das Maximum 1 entspricht dabei 1000 MBit/s. Auf der X-Achse wird der Jitter des Intervalls von 0 bis 1 skaliert dargestellt. Das Maximum 1 beim Jitter entspricht der Intervalllänge von 1s. Die Abbildung verdeutlicht, dass die Jitterwerte vor allem in den Bereichen von 0 ms bis zu 10 ms vorzufinden sind und die Bandbreite durchschnittlich 0.0072 MBit/s beträgt.

4.2 False Positve Test an normalen Daten

Das Ziel des ersten Szenarios ist es verschiedene Konfigurationen aller Algorithmen an der Erkennung von normalen Daten zu testen. Dabei sollte ein guter Algorithmus möglichst keine Daten als Ausreißer einstufen, da in diesem Szenario kein Angriff stattfindet.

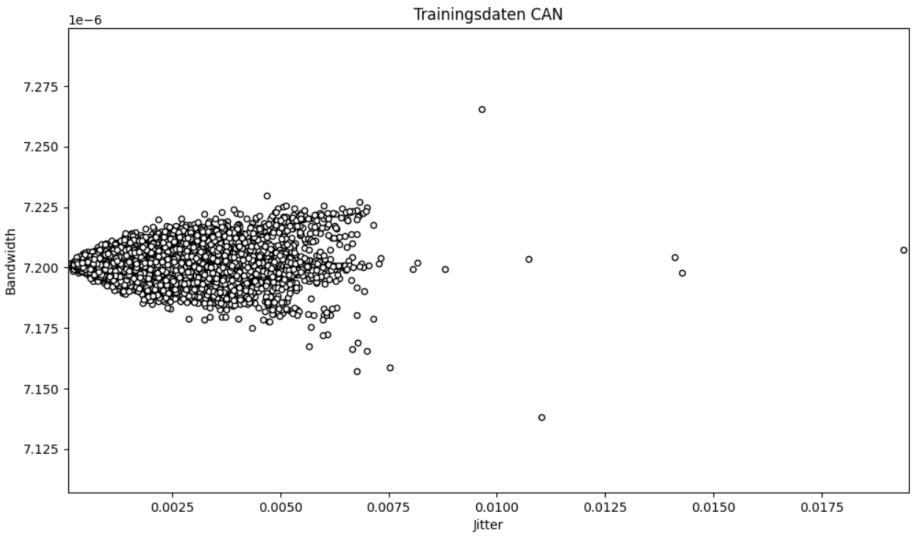


Abbildung 3: Abbildung 3 bildet die 30000 Trainingsdaten, die in den folgenden Versuchen verwendet werden, in weißen Punkten ab. Auf der Y-Achse wird ein skalierter Bandbreitenwert dargestellt und auf der X-Achse ein skalierter Jitterwert.

Insgesamt 3001 Datensätze erhält ein Algorithmus zum Einstufen als Anomalie oder normale Daten in diesem Szenario. Die Konfigurationen der Algorithmen die größere Zahlen an Ausreißern erkennen, werden in den folgenden Versuchen aussortiert. Die Ergebnisse des Szenarios sind in **Tabelle 1** abgebildet. Die Konfigurationen der Algorithmen, die besonders gute Resultate erzielt haben, sind durch fettgedruckte Schrift hervorgehoben. Jeder der 6 Algorithmen war in der Lage mit einer Konfiguration keine Daten als False Postive einzustufen oder im Falle von SVM nur 5 False Positives einzustufen.

4.3 Offensichtlicher Angriff

Dieses Szenario testet, welche Konfigurationen der Algorithmen in der Lage sind, einen offensichtlichen Angriff zuerkennen. Beim offensichtlichen Angriff werden zusätzlich zu den normalen Paketen noch eine große Anzahl weiterer Pakete mit gleichen Paketformat über einen bestimmten Zeitraum gesendet (Packet Injection). In diesem Szenario werden insgesamt 1000 Pakete über einen Zeitraum von 4 Sekunden zusätzlich versendet. Die Ergebnisse des Versuches sind in **Tabelle 2** abgebildet. In der Tabelle ist zuerkennen, dass alle Konfigurationen außer einer (IF Contamination = 0) zumindest einen Ausreißer

Algorithmen	Contamination	BEF	Nbins	Gamma	Normale Daten	Anomalien
SVM	0.5				1590	1411
SVM	0.3				2117	884
SVM	0.2				2418	583
SVM	0.1				2717	284
SVM	0.05				2860	141
SVM	0.01				2323	678
SVM	0			0.05	2993	8
SVM	0			0.01	2989	12
SVM	**0**			**0.0025**	**2995**	**5**
SVM	**0**			**0.0015**	**2995**	**5**
IF	0.01				2971	30
IF	**0.001**				**3000**	**1**
IF	**0.0001**				**3001**	**0**
IF	**0**				**3001**	**0**
EE	0.01				2977	24
EE	0.005				2987	14
EE	**0.001**				**2997**	**4**
EE	**0.0001**				**3001**	**0**
EE	**0**				**3001**	**0**
HBO	0.01		10		2972	29
HBO	**0.001**		**10**		**3000**	**1**
HBO	**0.0001**		**10**		**3001**	**0**
HBO	0.01		20		2980	21
HBO	**0.001**		**20**		**3000**	**1**
MS		0.5			2830	171
MS		0.7			2942	59
MS		0.9			2985	16
MS		**1**			**3000**	**1**
MS		**1.1**			**3001**	**0**
KM1		**0.4**			**3000**	**1**
KM1		**0.5**			**3001**	**0**
KM1		**0.6**			**3001**	**0**
KM1		**0.7**			**3001**	**0**
KM2		**0.5**			**3000**	**1**
KM2		**0.7**			**3001**	**0**

Tabelle 1: Die Tabelle zeigt die Ergebnisse des **False Positive Tests**. Das optimale Ergebnis für einen Algorithmus wäre es keinen einzigen Ausreißer zuerkennen, da kein Angriff in diesem Szenario stattgefunden hat. Die Konfigurationen der Algorithmen, die besonders gute Resultate erzielt haben, sind durch fettgedruckte Schrift hervorgehoben. (Beschreibung der Tabellenparameter: 7)

Algorithmen	Contamination	BEF	Nbins	Gamma	Normale Daten	Anomalien
SVM	**0.5**				**900**	**101**
SVM	**0.05**				**952**	**49**
SVM	**0**			**0.0015**	**990**	**11**
IF	**0.01**				**984**	**17**
IF	**0.001**				**996**	**5**
IF	**0.0001**				**999**	**1**
IF	0				1001	0
EE	**0.01**				**984**	**17**
EE	**0.001**				**996**	**5**
EE	**0.0001**				**997**	**4**
EE	**0**				**997**	**4**
HBO	**0.01**		**10**		**978**	**23**
HBO	**0.001**		**10**		**995**	**6**
HBO	**0.0001**		**10**		**998**	**3**
MS		**1**			**999**	**2**
MS		**1.1**			**999**	**2**
KM1		**0.4**			**999**	**2**
KM1		**0.5**			**998**	**3**
KM1		**0.6**			**999**	**2**

Tabelle 2: Die Tabelle zeigt die Ergebnisse der Algorithmen bei der Erkennung eines **offensichtlichen Angriffs**. Das optimale Ergebnis für einen Algorithmus wäre es 3 bis 5 Ausreißer zuerkennen. Die Konfigurationen der Algorithmen, die den Angriff erkannt haben, sind durch fettgedruckte Schrift hervorgehoben. (Beschreibung der Tabellenparameter: 7)

erkannt haben und somit auch den Angriff entdeckt haben. Das optimale Ergebnis eines Algorithmus wäre es 3-5 Ausreißer erkannt zu haben. Ein Beispiel für den Algorithmus, der gar nicht in der Lage war den Angriff zuerkennen, ist in **Abbildung 4** dargestellt. Die Konfiguration Isolation Forest mit der Contamination von 0 hat zwar zuvor keine False Positives erkannt, ist hier aber nicht in der Lage einen offensichtlichen Angriff zu identifizieren.

4.4 Packet Injection

Das Ziel dieses Angriffsszenarios ist es herauszufinden, welche Konfigurationen der Algorithmen in der Lage sind, Angriffe mit nur wenigen hinzugefügten Paketen zuerkennen

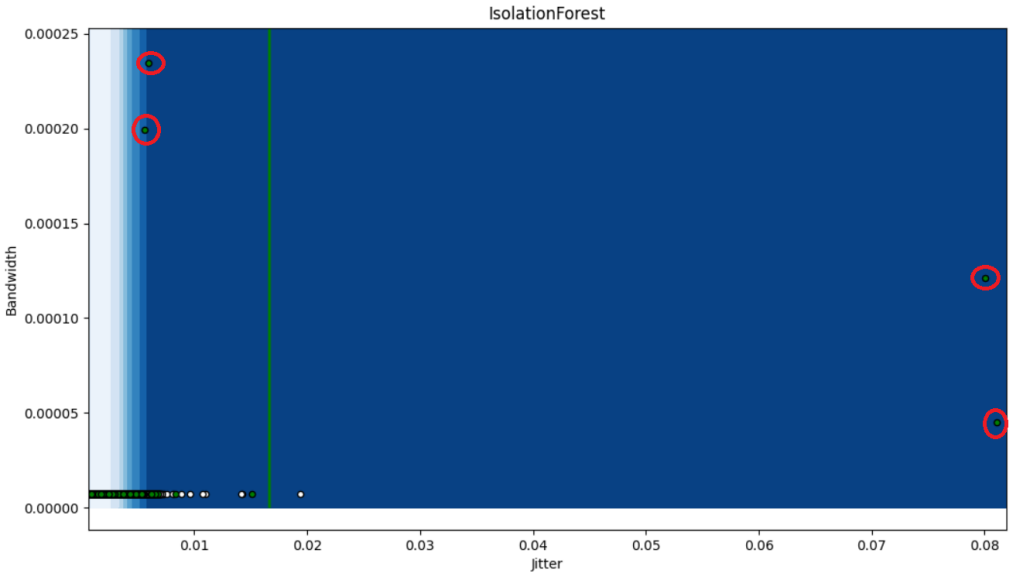


Abbildung 4: Abbildung 4 zeigt den Algorithmus Isolation Forest (IF) mit der Konfiguration Contamination 0. Die Abbildung ist ein Beispiel für eine Konfiguration in der ein offensichtlicher Angriff nicht erkannt wurde. Am oberen Rand der Abbildung und am rechten Rand der Abbildung sind jeweils zwei grüne Punkte (rote Kreise) zuerkennen, bei denen es sich klar um Ausreißer handelt. Diese Ausreißer liegen weit entfernt von den restlichen normalen grünen Punkten.

(Packet Injection). Es wurde Angriffe mit 50 hinzugefügten, 4 hinzugefügten und 1 hinzugefügten Paketen durchgeführt. Da alle verwendeten Konfigurationen der Algorithmen die Angriffe mit 50 und 4 hinzugefügten Paketen erkannt haben, werden in **Tabelle 3** nur die Ergebnisse des Angriffs mit 1 hinzugefügten Paket dargestellt. Auch bei diesem Angriff war jede Konfiguration der Algorithmen in der Lage den Angriff zu entdecken. Das optimale Ergebnis eines Algorithmus wäre dementsprechend genau 1 Ausreißer erkannt zu haben. In **Abbildung 5** ist zuerkennen, dass das einzelne hinzugefügte Paket in einem Datenpunkt, der auf der X-Achse weit entfernt von den restlichen normalen Datenpunkten liegt, resultiert. Deshalb waren alle Konfigurationen der Algorithmen in der Lage diesen Datenpunkt als Ausreißer zu identifizieren und damit den Angriff zu entdecken.

Algorithmen	Contamination	BEF	Nbins	Gamma	Normale Daten	Anomalien
SVM	**0**			**0.0015**	**994**	**7**
IF	**0.01**				**985**	**16**
IF	**0.001**				**997**	**4**
IF	**0.0001**				**1000**	**1**
EE	**0.001**				**999**	**2**
EE	**0.0001**				**1000**	**1**
EE	**0**				**1000**	**1**
HBO	**0.001**		**10**		**1000**	**1**
HBO	**0.0001**		**10**		**999**	**2**
MS		**1**			**1000**	**1**
MS		**1.1**			**1000**	**1**
KM1		**0.4**			**1000**	**1**
KM1		**0.5**			**1000**	**1**
KM1		**0.6**			**1000**	**1**

Tabelle 3: Die Tabelle zeigt die Ergebnisse der Algorithmen bei der Erkennung eines Angriffs, bei dem nur ein einzelnes Paket zusätzlich zu den normalen zyklischen Daten versendet wurde. Das optimale Ergebnis für einen Algorithmus wäre es genau 1 Ausreißer zuerkennen. Die Konfigurationen der Algorithmen, die den Angriff erkannt haben, sind durch fettgedruckte Schrift hervorgehoben. (Beschreibung der Tabellenparameter: 7)

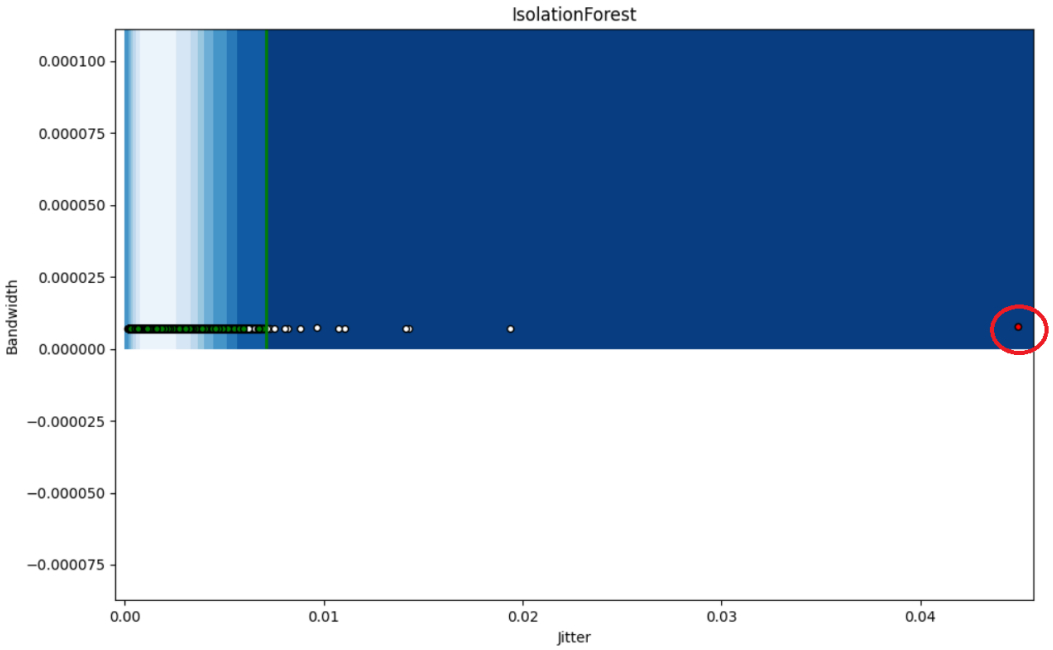


Abbildung 5: Abbildung 5 zeigt den Algorithmus Isolation Forest (IF) mit der Konfiguration Contamination 0.0001. Die Abbildung ist ein Beispiel für einen Angriff mit einem einzelnen Paket und soll verdeutlichen, dass der anomale Datenpunkt (roter Kreis) auf der X-Achse weit entfernt von den anderen normalen Datenpunkten liegt. Deshalb waren alle Algorithmen in der Lage den Angriff zu entdecken.

4.5 Packet Elimination

Der nächste Versuch betrachtet das Verhalten der Algorithmen, wenn ein Paket aus den normalen zyklischen Daten entfernt wird (Packet Elimination). Die Ergebnisse des Angriffs sind in **Tabelle 4** dargestellt. Jede einzelne Konfiguration der Algorithmen war in der Lage mindestens einen Ausreißer zu identifizieren, so dass der Angriff immer erkannt wurde. Ähnlich wie beim Hinzufügen eines einzelnen Paketes resultiert auch das Entfernen eines einzelnen Paketes in einem anomalen Datenpunkt, der weit von den normalen Datenpunkten entfernt liegt (siehe auch **Abbildung 6**).

Algorithmen	Contamination	BEF	Nbins	Gamma	Normale Daten	Anomalien
SVM	**0**			**0.0015**	**995**	**6**
IF	**0.01**				**981**	**20**
IF	**0.001**				**996**	**5**
IF	**0.0001**				**1000**	**1**
EE	**0.001**				**999**	**2**
EE	**0.0001**				**1000**	**1**
EE	**0**				**1000**	**1**
HBO	**0.001**		**10**		**997**	**4**
HBO	**0.0001**		**10**		**1000**	**1**
MS		**1**			**1000**	**1**
MS		**1.1**			**1000**	**1**
KM1		**0.4**			**1000**	**1**
KM1		**0.5**			**1000**	**1**
KM1		**0.6**			**1000**	**1**

Tabelle 4: Die Tabelle zeigt die Ergebnisse der Algorithmen bei der Erkennung eines Angriffs, bei dem ein einzelnes Paket aus den normalen zyklischen Daten entfernt wurde **(Packet Elimination)**. Das optimale Ergebnis für einen Algorithmus wäre es genau 1 Ausreißer erkannt zu haben. Die Konfigurationen der Algorithmen, die den Angriff erkannt haben, sind durch fettgedruckte Schrift hervorgehoben. (Beschreibung der Tabellenparameter: 7)

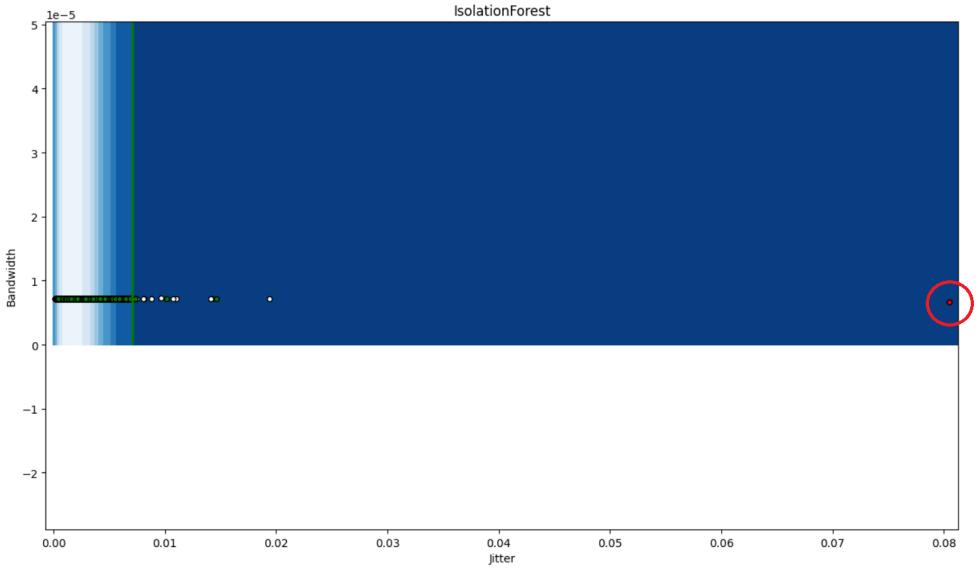


Abbildung 6: Abbildung 6 zeigt den Algorithmus Isolation Forest (IF) mit der Konfiguration Contamination 0.0001. Die Abbildung ist ein Beispiel für einen Angriff bei dem ein einzelnes Paket entfernt wird. Es ist deutlich zuerkennen, dass das entfernte Pakete in einem anomalen Datenpunkt (roter Kreis) resultiert.

4.6 Packet Modification

Das Angriffsszenario Packet Modification verändert die Payload der Pakete. Jede Payload wird um zusätzliche 8 Bytes erweitert. Die Ergebnisse des Versuches werden in **Tabelle 5** dargestellt. Nicht jeder Algorithmus war in der Lage den Angriff zu entdecken. Im Optimalfall hätte ein Algorithmus 989-990 Ausreißer erkennen müssen, da der Angriff erst nach kurzer Verzögerung gestartet ist. Nur die Konfigurationen der Algorithmen Elliptic Envelope und Histogram-based Outlier Detection haben das optimale Ergebnis erreicht. Die Algorithmen Support Vector Machines und IsolationForest waren zwar in der Lage den Angriff an ein paar Ausreißern zuerkennen, haben diesen aber an zu wenigen Ausreißern und den falschen Aspekten erkannt. Die beiden Clustering Algorithmen Mean Shift und K-Means haben bei diesem Angriff keinen einzigen Ausreißer identifiziert. **Abbildung 7** zeigt beispielhaft an dem Algorithmus Histogram-based Outlier Detection, dass die Ausreißer (rote Punkte) deutlich an einer erhöhten Bandbreite oberhalb der normalen Trainingsdaten (weiße Punkte) zu identifizieren sind. Im dem Beispiel wurde der Angriff von dem Algorithmus erfolgreich erkannt.

Algorithmen	Contamination	BEF	Nbins	Gamma	Normale Daten	Anomalien
SVM	**0**			**0.0015**	**996**	**5**
IF	**0.01**				**983**	**18**
IF	**0.001**				**998**	**3**
IF	0.0001				1001	0
EE	**0.001**				**12**	**989**
EE	**0.0001**				**11**	**990**
EE	**0**				**12**	**989**
HBO	**0.001**		**10**		**12**	**989**
HBO	**0.0001**		**10**		**12**	**989**
MS		1			1001	0
MS		1.1			1001	0
KM1		0.4			1001	0
KM1		0.5			1001	0
KM1		0.6			1001	0

Tabelle 5: Die Tabelle 5 zeigt die Ergebnisse des Angriffsszenarios **Packet Modification**, dass die Payload von einem Großteil der Pakete um 8 Byte vergrößert hat. Das optimale Ergebnis für einen Algorithmus wäre es 989-990 Ausreißer erkannt zu haben, da der Angriff erst startet, nachdem ein paar normale Pakete verschickt worden sind. Die Konfigurationen der Algorithmen, die den Angriff erfolgreich erkannt haben, sind durch fettgedruckte Schrift hervorgehoben. Ein paar der Algorithmen haben den Angriff nur durch falsche Aspekte bzw. wenige Ausreißer erkannt. (Beschreibung der Tabellenparameter: 7)

4.7 Weitere Angriffsmöglichkeiten und Bewertung der Algorithmen

Nach Durchführung aller Versuche konnten **zwei Konfigurationen vom dem Algorithmus Elliptic Envelope und 1 Konfiguration von dem Algorithmus Histogram-based Outlier Detection** in jedem Versuch das optimale Ergebnis erzielen. Denn diese Konfigurationen waren in der Lage sowohl keine False Positves in den normalen Daten zuerkennen, als auch bei jedem Angriff die richtige Anzahl an Ausreißern an den richtigen Kriterien zuerkennen. Durch die Metrik des Jitters ist es für einen Angreifer nur schwer möglich ein einzelnes Paket hinzuzufügen oder zu entfernen, ohne das einer der beiden Algorithmen den Angriff erkennen würde. Ein präzises Timing und mehrere anomale Pakete wären notwendig, um keine Auffäligkeit in der Metrik des Jitters zu zeigen. Deswegen waren auch in allen Versuchen bei dem Pakete hinzugefügt oder entfernt wurden,

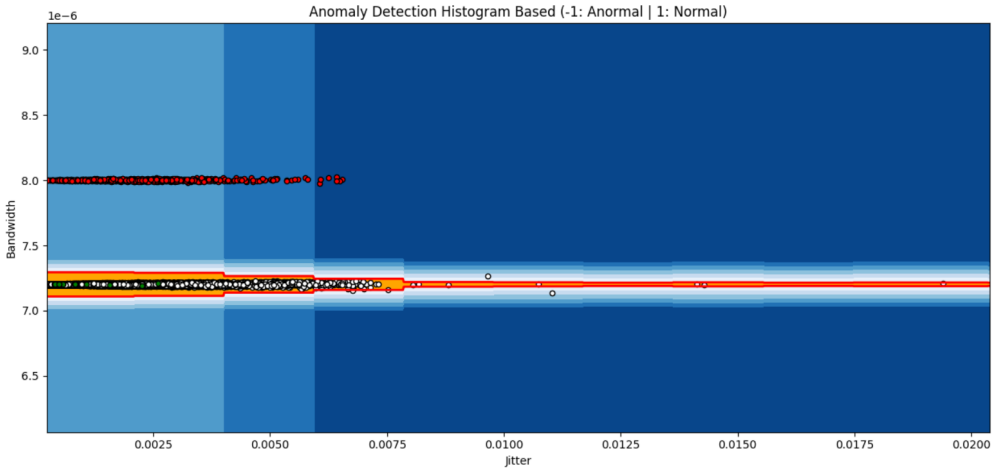


Abbildung 7: Abbildung 7 zeigt den Algorithmus Histogram-based Outlier Detection (HBO) mit der Konfiguration Contamination 0.0001. Die Abbildung ist ein Beispiel für einen Angriff bei dem die Payload der Pakete jeweils um 8 Byte erweitert wird. Es ist deutlich zuerkennen, dass die Ausreißer oberhalb der normalen Daten angeordnet sind. Der Algorithmus Histogram-based Outlier Detection ist in der Lage diese Ausreißer erfolgreich zu identifizieren.

jeder einzelne Algorithmus in der Lage diesen Angriff zuerkennen. Bei einem Angriff der eine Veränderung der Paketgröße beinhaltet, ergaben sich Auffäligkeiten in der Metrik der Bandbreite. Diese Auffäligkeiten waren auch für die meisten Algorithmen gut erkennbar. Dagegen wären Angriffe, die den Inhalt der Pakete verändern würden ohne die Payload zu vergrößern und keine Pakete hinzufügen würden, für keinen der Algorithmen mit diesen Metriken erkennbar.

Der Algorithmus IsolationForest war nicht in der Lage alle Ausreißer bei einer Veränderung der Paketgröße zuerkennen. Der Angriff konnte nur noch durch anomale Datenpunkte, die auch einen erhöhten Jitteranteil besaßen, entdeckt werden. Auch die beiden Clustering Algorithmen Mean Shift und K-Means waren nicht in der Lage den Angriff mit veränderter Paketgröße überhaupt zuerkennen. Das Hauptproblem der Algorithmen war, dass eine kreisförmige Form mit gleichem Abstand in beiden Dimensionen zu viel Toleranz im Bereich der Bandbreite bietet. Der Algorithmus Support Vector Machines war als einziger Algorithmus nicht der Lage keine False Positives an nur normalen Daten zu erkennen.

5 Anomalieerkennung im Videostream

Das Kapitel Anomalieerkennung im Videostream fokussiert sich auf die Durchführung und Ergebnisse bei der Anomalieerkennung in verschiedenen Angriffsszenarien im Videostream und ordnet am Ende des Kapitels die Ergebnisse ein. Zuerst beschreibt **5.1 Training** den Ablauf des Trainings und **5.2 False Positve Test an normalen Daten** betrachtet die Anzahl an False Positives bei Erkennung von normalen Daten, um bestimmte Konfigurationen der Algorithmen für die folgenden Versuche auszusortieren. Anschließend schaut das Kapitel **5.3 Offensichtlicher Angriff**, ob alle Algorithmen in der Lage sind einen offensichtlichen Angriff zuerkennen und dann folgen in **5.4 Packet Injection**, **5.5 Packet Elimination** und **5.6 Packet Modification** spezifische Angriffsszenarien zum Vergleichen der Algorithmen. Das letzte Kapitel **5.7 Weitere Angriffsmöglichkeiten und Bewertung der Algorithmen** diskutiert weitere Angriffsmöglichkeiten, die von den Algorithmen möglicherweise nicht erkannt werden können und bewertet die Algortihmen.

5.1 Training

Die folgende **Abbildung 8** zeigt die Ergebnisse des durchgeführten Trainings. Das Training wurde wie bei der Anomalieerkennung im CAN-Bus Stream über eine Gesamtdauer von 30000s in Intervallen von 1s durchgeführt. Die 30000 Trainingsdaten sind in der Abbildung als weiße Punkte dargestellt. Auf der Y-Achse wird ein von 0 bis 1 skalierter Wert der Bandbreite in einem Intervall dargestellt. Das Maximum 1 entspricht dabei 1000 MBit/s. Auf der X-Achse wird beim Videostream die durchschnittliche Paketgröße des Intervalls von 0 bis 1 skaliert dargestellt. Das Maximum 1 bei der Paketgröße entspricht mit 1518 Bytes der maximalen Gesamtgröße eines Ethernet-Frames. Aus der Abbildung ist zu erkennen, dass die durchschnittliche Paketgröße der Trainingsdaten ausnahmelos bei 1362 Bytes liegt. Die Metrik der Paketgröße wurde anstatt der Metrik des Jitters gewählt, weil der Jitter beim Videostream zu verteilt über den gesamten Wertebereich liegt, um damit Ausreißer von normalen Daten abzugrenzen. Die Bandbreite liegt in dem Wertebetreich von 0,1 MBit/s (10 Pakete pro Intervall) bis zu 2,5 MBit/s (230 Pakete pro Intervall).

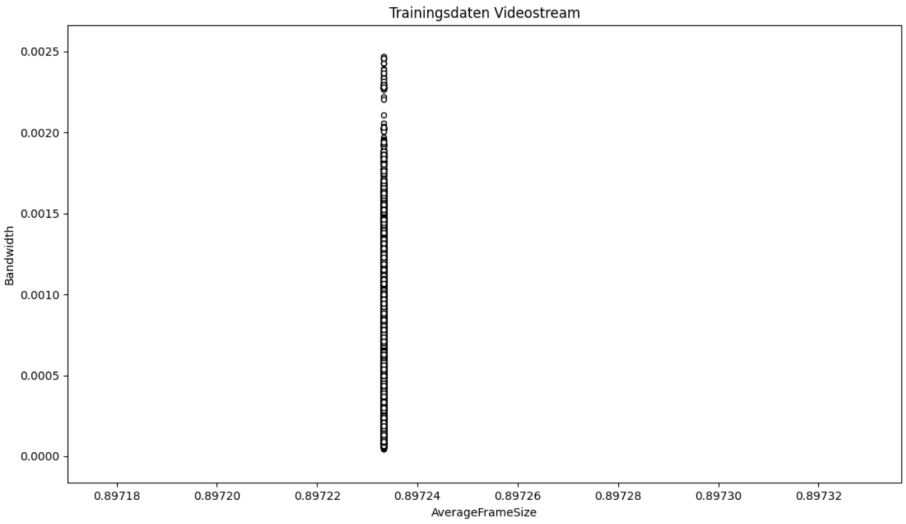


Abbildung 8: Abbildung 8 bildet die 30000 Daten, die in den folgenden Versuchen für das Training der Algorithmen verwendet werden, in weißen Punkten ab. Auf der Y-Achse wird ein skalierter Bandbreitenwert dargestellt und auf der X-Achse ein skalierter Wert der Paketgröße.

5.2 False Positive Test an normalen Daten

Das Ziel des ersten Szenarios ist es verschiedene Konfigurationen aller Algorithmen an der Erkennung von normalen Daten zu testen. Dabei sollte ein guter Algorithmus möglichst keine Daten als Ausreißer einstufen, da in diesem Szenario kein Angriff stattfindet. In diesem Szenario erhält jeder Algorithmus insgesamt 1001 Daten, die er richtig als keine Anomalien einstufen muss. Die Konfigurationen der Algorithmen die größere Zahlen an Ausreißern erkennen, werden in den folgenden Versuchen aussortiert. Die Ergebnisse des Szenarios sind in **Tabelle 6** abgebildet. Die Konfigurationen der Algorithmen, die besonders gute Resultate erzielt haben, sind durch fettgedruckte Schrift hervorgehoben. Jeder der 6 Algorithmen war in der Lage mit einer oder mehrerer Konfigurationen keine Daten als False Postive einzustufen.

5.3 Offensichtlicher Angriff

Dieses Szenario testet, welche Konfigurationen der Algorithmen in der Lage sind, einen offensichtlichen Angriff zuerkennen. Beim offensichtlichen Angriff werden zusätzlich zu

Algorithmen	Contamination	BEF	Nbins	Gamma	Normale Daten	Anomalien
SVM	0.5			scale	250	751
SVM	0.3			scale	737	264
SVM	0.1			scale	898	103
SVM	0.01			scale	854	147
SVM	**0.01**			**5**	**993**	**8**
SVM	0.01			50	989	12
SVM	**0.001**			**5**	**998**	**3**
SVM	0.0001			5	95	906
SVM	**0.0001**			**50**	**1001**	**0**
SVM	**0.0001**			**100**	**1001**	**0**
IF	0.01				991	10
IF	**0.001**				**999**	**2**
IF	**0.0001**				**1001**	**0**
IF	**0**				**1001**	**0**
EE	0.01				991	10
EE	**0.001**				**1000**	**1**
EE	**0.0001**				**1001**	**0**
EE	**0**				**1001**	**0**
HBO	**0.001**		**10**		**999**	**2**
HBO	**0.0001**		**10**		**1001**	**0**
HBO	**0.001**		**20**		**1001**	**0**
HBO	**0.0001**		**20**		**1001**	**0**
MS		0.5			910	91
MS		0.7			972	29
MS		0.8			982	19
MS		0.9			992	9
MS		**1**			**1001**	**0**
KM1		0.5			992	9
KM1		**0.6**			**997**	**4**
KM1		**0.7**			**999**	**2**
KM1		**0.8**			**1000**	**1**
KM1		**0.9**			**1000**	**1**
KM1		**1**			**1001**	**0**

Tabelle 6: Die Tabelle zeigt die Ergebnisse des **False Positive Tests**. Das optimale Ergebnis für einen Algorithmus wäre es keinen einzigen Ausreißer zuerkennen, da kein Angriff in diesem Szenario stattgefunden hat. Die Konfigurationen der Algorithmen, die besonders gute Resultate erzielt haben, sind durch fettgedruckte Schrift hervorgehoben. (Beschreibung der Tabellenparameter: 7)

Algorithmen	Contamination	BEF	Nbins	Gamma	Normale Daten	Anomalien
SVM	**0.1**			**5**	**915**	**86**
SVM	**0.01**			**5**	**983**	**18**
SVM	**0.001**			**5**	**990**	**11**
SVM	**0.0001**			**50**	**994**	**7**
SVM	**0.0001**			**100**	**993**	**8**
IF	**0.01**				**984**	**17**
IF	**0.001**				**982**	**9**
IF	0.0001				1001	0
IF	0				1001	0
EE	**0.001**				**991**	**10**
EE	**0.0001**				**993**	**8**
EE	**0**				**992**	**9**
HBO	**0.001**		**20**		**991**	**10**
HBO	**0.0001**		**20**		**993**	**8**
MS		**0.8**			**978**	**23**
MS		**0.9**			**989**	**12**
MS		**1**			**994**	**7**
KM1		**0.7**			**991**	**10**
KM1		**0.8**			**990**	**11**
KM1		**0.9**			**993**	**8**
KM1		**1**			**993**	**8**

Tabelle 7: Die Tabelle zeigt die Ergebnisse der Algorithmen bei der Erkennung eines **offensichtlichen Angriffs**. Das optimale Ergebnis für einen Algorithmus wäre es 7-9 Ausreißer zuerkennen. Die Konfigurationen der Algorithmen, die den Angriff erkannt haben, sind durch fettgedruckte Schrift hervorgehoben. (Beschreibung der Tabellenparameter: 7)

den normalen Paketen noch eine große Anzahl weiterer Pakete mit gleichen Paketformat über einen bestimmten Zeitraum gesendet (Packet Injection). Der Angriff läuft über einen Zeitraum von 8 Sekunden ab und verschickt insgesamt 2400 anomale Pakete. Die Ergebnisse des Versuches sind in **Tabelle 7** abgebildet. Aus der Tabelle ist abzulesen, dass alle Algorithmen außer zwei Konfigurationen von IsolationForest in der Lage waren den Angriff zuerkennen. Das optimale Ergebnis wäre es insgesamt 7-9 Ausreißer erkannt zu haben. Ein Beispiel für das erfolgreiche und optimale Entdecken des Angriffs ist in **Abbildung 9** an dem Algorithmus Support Vector Machines dargestellt. Die roten Punkte in der Abbildung stellen die anomalen Daten, die eine deutlich erhöhte Bandbreite im Vergleich zu den normalen Daten aufweisen, dar.

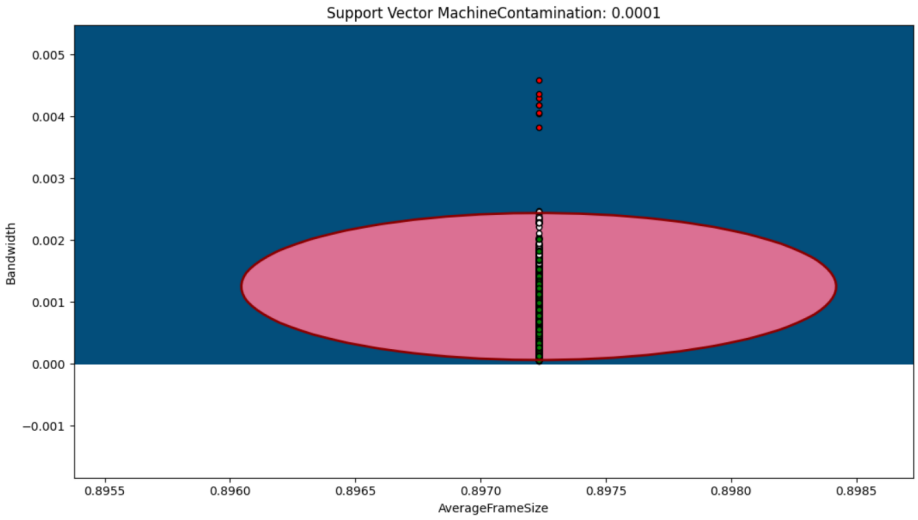


Abbildung 9: Abbildung 9 visualisiert ein Ergebnis des Angriffsszenarios Offensichtlicher Angriff. Die Abbildung stellt den Algorithmus Support Vector Machines mit der Contamination von 0.0001 dar. Weiterhin ist zuerkennen, dass die anomale Daten (rote Punkte) beim offensichtlichen Angriff eine deutlich erhöhte Bandbreite gegenüber den normalen Daten (grüne Punkte) aufweisen und der Angriff von diesem Algorithmus erfolgreich erkannt wurde.

5.4 Packet Injection

Das Ziel dieses Angriffsszenarios ist es zu untersuchen, welche Konfigurationen der Algorithmen in der Lage sind, Angriffe mit nur wenigen hinzugefügten Paketen zuerkennen (Packet Injection). Beim Videostream wurden zwei verschiedene Angriffsvarianten von Packet Injection durchgeführt. Bei der ersten Variante wurden insgesamt 100 anomale Pakete über einen sehr kurzen Zeitraum von 1 Sekunde hinzugefügt. Die Ergebnisse dieses Versuches sind in **Tabelle 8** abgebildet. Das optimale Ergebnis bei diesem Angriff wäre es genau 1 Ausreißer erkannt zu haben. Die Konfigurationen der Algorithmen, die den Angriff erkannt haben, sind durch fettgedruckte Schrift hervorgehoben. Viele Konfigurationen der Algorithmen waren gar nicht in der Lage diesen Angriff zu erkennen. Weitere Algorithmen haben zusätzlich zu dem anomalen Datenpunkt weitere False Positives erkannt und nur dadurch den Angriff identifiziert. Durch die große Varianz und Anzahl der normalen Pakete und die Intervalllänge, zeigen die 100 hinzugefügten Pakete trotzdem häufig keine Auffälligkeit in der Bandbreite beim Videostream. Das Hinzufügen nur eines einzelnen Paketes wie beim CAN-Bus Stream ist für keinen Algorithmus an der

Bandbreite erkennbar. Die zweite Variante des Packet Injection Angriffes hat insgesamt 1000 anomale Pakete über einen Zeitraum von 10 Sekunden versendet. Die Ergebnisse dieses Angriffes sind in **Tabelle 9** dargestellt. Das optimale Ergebnis für einen Algorithmus wäre es 10-11 Ausreißer erkannt zu haben. Auch in diesem Szenario waren die meisten Konfigurationen nicht in der Lage die anomalen Pakete zu erkennen. Das beste Ergebnis wurde von zwei Konfigurationen von Support Vector Machines und einer Konfiguration von Mean Shift erreicht. In **Abbildung 10** ist am Beispiel von SVM mit der Contamination von 0.001 das Entdecken der anomalen Datenpunkte dargestellt. Es ist zu erkennen, dass beim Videostream selbst eine größere Anzahl von Paketen mit identischen Paketformat nur eine sehr geringe Auffälligkeit in der Metrik der Bandbreite besitzt. Dadurch ist es sehr schwierig Angriffe, bei denen Pakete hinzugefügt werden, zu erkennen.

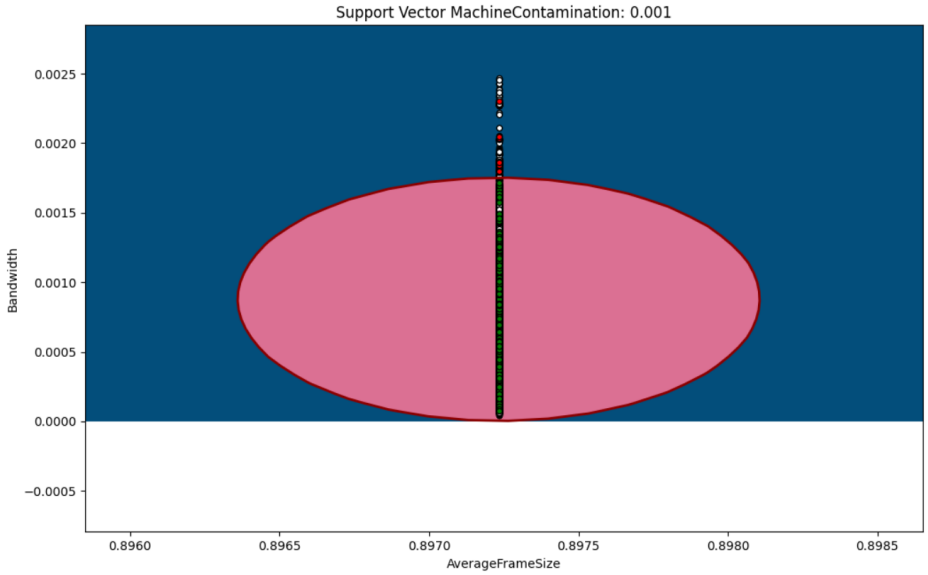


Abbildung 10: Abbildung 10 visualisiert ein Ergebnis des Angriffsszenarios Packet Injection. Es wurden 1000 anomale Pakete über einen Zeitraum von 10 Sekunden hinzugefügt. Die Abbildung stellt den Algorithmus Support Vector Machines mit der Contamination von 0.001 dar. Weiterhin ist zuerkennen, dass die anomale Daten (rote Punkte) in der Metrik der Bandbreite kaum eine Auffälligkeit besitzen und im oberen Bereich der Trainingsdaten angeordnet sind.

Algorithmen	Contamination	BEF	Nbins	Gamma	Normale Daten	Anomalien
SVM	**0.1**			**5**	**897**	**104**
SVM	**0.01**			**5**	**992**	**9**
SVM	**0.001**			**5**	**994**	**7**
SVM	0.0001			50	1001	0
SVM	0.0001			100	1001	0
IF	**0.01**				**993**	**8**
IF	**0.001**				**1000**	**1**
IF	0.0001				1001	0
IF	0				1001	0
EE	**0.001**				**1000**	**1**
EE	0.0001				1001	0
EE	0				1001	0
HBO	0.001		20		1001	0
HBO	0.0001		20		1001	0
MS		**0.8**			**981**	**20**
MS		**0.9**			**994**	**7**
MS		**1**			**1001**	**0**
KM1		**0.8**			**1000**	**1**
KM1		**0.9**			**1000**	**1**
KM1		1			1001	0

Tabelle 8: Die Tabelle zeigt die Ergebnisse der Algorithmen bei der Erkennung eines Angriffs, der **100 anomale Pakete über sehr kurzen Zeitraum** versendet. Das optimale Ergebnis für einen Algorithmus wäre es genau 1 Ausreißer zuerkennen. Die Konfigurationen der Algorithmen, die den Angriff erkannt haben, sind durch fettgedruckte Schrift hervorgehoben. (Beschreibung der Tabellenparameter: 7)

Algorithmen	Contamination	BEF	Nbins	Gamma	Normale Daten	Anomalien
SVM	**0.1**			**5**	**884**	**17**
SVM	**0.01**			**5**	**991**	**10**
SVM	**0.001**			**5**	**992**	**9**
SVM	**0.0001**			**50**	**1000**	**1**
SVM	**0.0001**			**100**	**1000**	**1**
IF	**0.01**				**977**	**24**
IF	**0.001**				**1000**	**1**
IF	0.0001				1001	0
IF	0				1001	0
EE	**0.001**				**1000**	**1**
EE	0.0001				1001	0
EE	0				1001	0
HBO	0.001		20		1001	0
HBO	0.0001		20		1001	0
MS		**0.8**			**978**	**23**
MS		**0.9**			**992**	**9**
MS		1			1001	0
KM1		**0.8**			**998**	**3**
KM1		0.9			1001	0
KM1		1			1001	0

Tabelle 9: Die Tabelle zeigt die Ergebnisse der Algorithmen bei der Erkennung eines Angriffs, der **1000 anomale Pakete über einen Zeitraum von 10 Sekunden** hinzugefügt. Das optimale Ergebnis für einen Algorithmus wäre es 10-11 Ausreißer zuerkennen. Die Konfigurationen der Algorithmen, die den Angriff erkannt haben, sind durch fettgedruckte Schrift hervorgehoben. (Beschreibung der Tabellenparameter: 7)

5.5 Packet Elimination

Der nächste Versuch betrachtet das Verhalten der Algorithmen, wenn mehrere Pakete aus den normalen zyklischen Daten entfernt werden (Packet Elimination). Dazu wird ein Angriff ausgeführt, der über eine Dauer von 1 Minute jedes zweite Paket aus den zyklischen normalen Daten entfernt. Die Ergebnisse dieses Versuches sind in der **Tabelle 10** dargestellt. Das optimale Ergebnis für einen Algorithmus wäre es 60-61 Ausreißer zu entdecken. Die Tabelle zeigt, dass der Großteil der Algorithmen nicht der Lage war den Angriff zu erkennen. Support Vector Machines mit der Contamination von 0.0001 und Gamma von 100 war die einzige Konfiguration, die in der Lage war, in der Dauer des Angriffs 4 Anomalien zu entdecken ohne einen weiteren False Positive zu erkennen. Das Ergebnis dieses Algorithmus ist der **Abbildung 11** dargestellt. Der rote Kreis umrandet die 4 identifizierten Anomalien am unteren Rand der Abbildung.

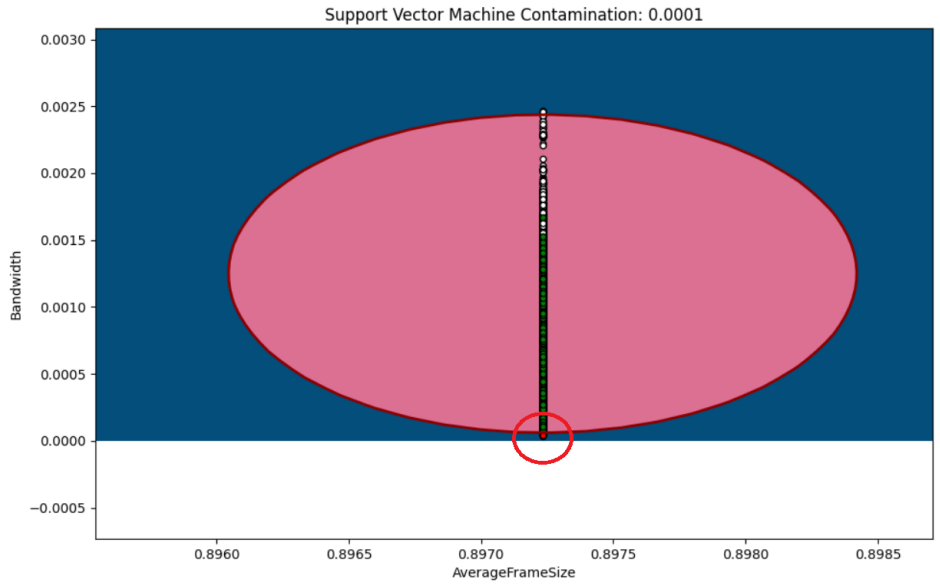


Abbildung 11: Abbildung 11 zeigt am Beispiel des Algorithmus Support Vector Machines die 4 anomalen Datenpunkte (rote Kreise) aus dem Angriffsszenario Packet Elimination. Der Algorithmus war erfolgreich in der Lage ein Teil der anomalen Datenpunkte zu identifizieren und den Angriff zu entdecken.

Algorithmen	Contamination	BEF	Nbins	Gamma	Normale Daten	Anomalien
SVM	**0.1**			**5**	**916**	**85**
SVM	**0.01**			**5**	**994**	**7**
SVM	**0.001**			**5**	**997**	**4**
SVM	0.0001			50	1001	0
SVM	**0.0001**			**100**	**997**	**4**
IF	**0.01**				**992**	**9**
IF	**0.001**				**1000**	**1**
IF	0.0001				1001	0
IF	0				1001	0
EE	**0.001**				**999**	**2**
EE	0.0001				1001	0
EE	0				1001	0
HBO	0.001		20		1001	0
HBO	0.0001		20		1001	0
MS		**0.8**			**990**	**11**
MS		**0.9**			**992**	**9**
MS		1			1001	0
KM1		**0.8**			**999**	**2**
KM1		0.9			1001	0
KM1		1			1001	0

Tabelle 10: Die Tabelle zeigt die Ergebnisse der Algorithmen bei der Erkennung eines Angriffs, der **über einen Zeitraum von 1 Minute jedes zweite Paket entfernt (Packet Elimination)**. Das optimale Ergebnis für einen Algorithmus wäre es 60-61 Ausreißer zuerkennen. Die Konfigurationen der Algorithmen, die den Angriff erkannt haben, sind durch fettgedruckte Schrift hervorgehoben. (Beschreibung der Tabellenparameter: 7)

5.6 Packet Modification

Das Angriffsszenario Packet Modification verändert die Payload der Pakete. Der Angriff dauert 1 Minute an und reduziert jede Sekunde die Payload mehrerer Pakete auf 8 Bytes. Die Ergebnisse des Versuches werden in **Tabelle 11** dargestellt. Im Optimalfall hätte ein Algorithmus 60-61 Ausreißer erkennen müssen. Konfigurationen der drei Algorithmen Support Vector Machines, Mean Shift und K-Means waren erfolgreich in der Lage den Angriff zu identifizieren und haben auch die optimale Anzahl an Ausreißern erkannt. Die anderen drei Algorithmen waren nicht in der Lage den Angriff an der veränderten Paketgröße zu erkennen und haben höchstens an False Positives den Angriff überhaupt erkannt. In der **Abbildung 12** ist am Beispiel vom Algorithmus Support Vector Machines die Anordnung der anomalen Daten (rote Kreise) zuerkennen. Die Trainingsdaten (weiße Punkte) und die normalen Daten (grüne Punkte) sind im rechten Teil der Abbildung vom Kreis des Algorithmus Support Vector Machines umrandet. Die roten Punkte besitzen alle unterschiedliche Paketgrößen, da der durchschnittliche Wert von der gesamten Anzahl an Paketen in einem Intervall abhängt. In diesem Beispiel war der Algorithmus erfolgreich in der Lage den Angriff zu entdecken. Ein paar weitere anomale Datenpunkte befinden sich noch im kleineren Bandbreitenbereich und sind in der Abbildung nicht dargestellt.

5.7 Weitere Angriffsmöglichkeiten und Bewertung der Algorithmen

Nach Durchführung aller Versuche konnte keine Konfiguration eines Algorithmus in jedem Szenario das optimale Ergebnis erzielen und jeden Angriff an den richtigen Aspekten entdecken. Der Algorithmus **Support Vector Machines mit der Konfiguration Contamination 0.0001 und Gamma 100** hat die besten Ergebnisse bei der Anomalieerkennung im Videostream gezeigt. Diese Konfiguration hat als einzige einen Angriff, in dem viele Pakete entfernt wurden (Packet Elimination), erkannt. Weiterhin wurden in keinem Versuch Daten fälschlicherweise als False Positives eingeordnet und nur die Packet Injection von wenigen Paketen war für diese Konfiguration nicht zu identifizieren. Die beiden Clustering Algorithmen K-Means und Mean Shift haben im Vergleich zu anderen Algorithmen auch gute Ergebnisse bei dem Angriff mit veränderter Paketgröße gezeigt. Die anderen drei Algorithmen IsolationForest, Histogram-based

Algorithmen	Contamination	BEF	Nbins	Gamma	Normale Daten	Anomalien
SVM	**0.1**			**5**	**861**	**140**
SVM	**0.01**			**5**	**930**	**71**
SVM	**0.001**			**5**	**933**	**68**
SVM	**0.0001**			**50**	**940**	**61**
SVM	**0.0001**			**100**	**940**	**61**
IF	**0.01**				**995**	**6**
IF	**0.001**				**999**	**2**
IF	0.0001				1001	0
IF	0				1001	0
EE	**0.001**				**1000**	**1**
EE	0.0001				1001	0
EE	0				1001	0
HBO	0.001		20		1001	0
HBO	0.0001		20		1001	0
MS		**0.8**			**924**	**77**
MS		**0.9**			**931**	**70**
MS		**1**			**940**	**61**
KM1		**0.8**			**938**	**63**
KM1		**0.9**			**939**	**62**
KM1		**1**			**940**	**61**

Tabelle 11: Die Tabelle zeigt die Ergebnisse der Algorithmen bei der Erkennung eines Angriffs, der **die Paketgröße der Pakete modifiziert**. Das optimale Ergebnis für einen Algorithmus wäre es 60-61 Ausreißer zuerkennen, da der Angriff eine Minute lang die Paketgröße modifiziert. Die Konfigurationen der Algorithmen, die den Angriff erkannt haben, sind durch fettgedruckte Schrift hervorgehoben. (Beschreibung der Tabellenparameter: 7)

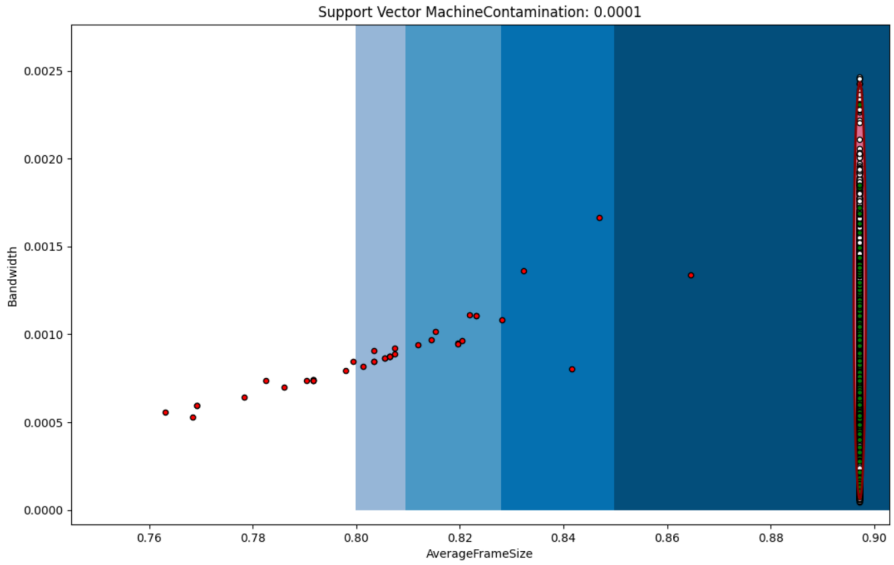


Abbildung 12: Abbildung 12 zeigt am Beispiel des Algorithmus Support Vector Machines die Anordnung der anomalen Datenpunkte (rote Punkte) im Angriffsszenario Packet Modification. Die Trainingsdaten (weiße Punkte) und die normalen Daten (grüne Punkte) sind im rechten Teil der Abbildung vom Kreis des Algorithmus Support Vector Machines umrandet. Der Algorithmus war erfolgreich in der Lage die anomalen Datenpunkte zu identifizieren und den Angriff zu entdecken.

Outlier Detection und Elliptic Envelope haben trotz sehr deutlich veränderter Paketgröße keine Ausreißer und damit zu viel Toleranz im Bereich der Paketgröße aufgewiesen.

Die Metrik der Bandbreite bietet beim Videostream für die Algorithmen kaum die Möglichkeit das Hinzufügen einzelner Pakete zu entdecken. Denn der Videostream weist zu viel Varianz in der Anzahl der Pakete in einem Intervall auf und Auffälligkeiten in der Metrik der Bandbreite bei der Packet Injection und Packet Elimination basieren dadurch vor allem auch auf Glück/Timing. Die Metrik der Paketgröße ist wiederum gut geeignet um anomale Pakete zu identifizieren. Angriffe, die den Inhalt der Pakete verändern würden und die gleiche Paketgröße besitzen, wären für alle Algorithmen nicht zu entdecken.

6 Ausblick und Zusammenfassung

Die Ausarbeitung hat grundlegende Aspekte der Anomalieerkennung erläutert und anschließend verschiedene Algorithmen zur Anomalieerkennung in realer Hardware innerhalb des SecVI Demonstrators eingesetzt und miteinander verglichen. Der Vergleich der Algorithmen wurde im Netzwerk auf der Ebene eines CAN-Bus Streams und auf der Ebene eines Videostreams durchgeführt. In verschiedenen Angriffsszenarien wurde das Verhalten und die Erkennungsrate der Algorithmen untersucht. Bei der Anomalieerkennung im CAN-Bus Streams hat sich gezeigt, dass die beiden statistischen Algorithmen Histogram-based Outlier Detection und Elliptic Envelope bessere Ergebnisse als die anderen Algorithmen in der Erkennung von anomalen Daten erreicht haben. Weiterhin hat sich die Metrik des Jitters als sehr gut geeignet für die Anomalieerkennung herausgestellt. Dagegen war bei der Anomalieerkennung im Videostream auffällig, dass die Metrik der Bandbreite eher ungeeignet für die Anomalieerkennung war und mit Support Vector Machines ein völlig anderer Algorithmus die besten Ergebnisse erreicht hat. Die beiden besten Algorihtmen Histogram-based Outlier Detection und Elliptic Envelope aus der Anomalieerkennung beim CAN-Bus Stream hatten bei der Erkennung im Videostream größere Schwierigkeiten beim Identifizieren der anoamlen Daten. Abhängig von den gewählten Metriken und der Anordnung der Daten variieren die Ergebnisse der Algorithmen stark und die Wahl des Algorithmus muss darauf angepasst werden. Durch eine bessere Konfiguration der Algorithmen und eine Durchführung der Angriffe ohne zu viele normale Daten mit anschließendem präzisen Prozentwert für die Erkennung hätte die Qualität der Durchführung der Versuche noch erhöht werden können. Im weiteren Verlauf meines Masters (Masterarbeit) werde ich auf Basis meiner Erkenntnisse aus diesem Hauptprojekt mich mit Anomalieerkennung in realer Hardware im Rahmen eines echten TSN-Netzwerkes inklusive TSN-Switch befassen.

7 Anhang: Beschreibung der Tabellenparameter

Contamination beschreibt den prozentualen Anteil an Ausreißern im Datensatz. Der Parameter wird im Training benutzt um die Grenzen für die normalen Daten festzulegen.
BEP ist die Abkürzung für BorderEnlargementFactor und legt den Radius des Clusters ausgehend vom Clusterzentrum fest.
Nbins gibt bei dem Algorithmus Histogram-based Outlier Detection die Anzahl an Behältern, in die Trainingsdaten eingeordnet werden, an.
Gamma beschreibt beim Algorithmus Support Vector Machines den Einfluss eines einzelnen Trainingsdatenpunkt. Je größer Gamma ist, desto dichter müssen andere Daten für die Beeinflussung sein.
KM1 gibt mit der Zahl 1 die Anzahl der Cluster, die gebildet werden sollen, an.

Literatur

[1] Bhuyan, Monowar H. ; Bhattacharyya, D. K. ; Kalita, J. K.: Network Anomaly Detection: Methods, Systems and Tools. In: *IEEE Communications Surveys Tutorials* 16 (2014), Nr. 1, S. 303–336

[2] Buschjäger, Sebastian ; Honysz, Philipp-Jan ; Morik, Katharina: Generalized Isolation Forest: Some Theory and More Applications Extended Abstract. In: *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, 2020, S. 793–794

[3] Chandola, Varun ; Banerjee, Arindam ; Kumar, Vipin: Anomaly Detection: A Survey. In: *ACM Comput. Surv.* 41 (2009), 07

[4] Checkoway, Stephen ; Mccoy, Damon ; Anderson, Danny ; Kantor, Brian ; Shacham, Hovav ; Savage, Stefan ; Koscher, Karl ; Czeskis, Alexei ; Roesner, Franziska ; Kohno, Tadayoshi: Comprehensive Experimental Analyses of Automotive Attack Surfaces, 08 2011

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

[6] Gmiden, Mabrouka ; Gmiden, Mohamed H. ; Trabelsi, Hafedh: An intrusion detection method for securing in-vehicle CAN bus. In: *2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, 2016, S. 176–180

[7] Goldstein, Markus ; Dengel, Andreas: Histogram-based Outlier Score (HBOS): A fast Unsupervised Anomaly Detection Algorithm, 09 2012

[8] Hank, Peter ; Müller, Steffen ; Vermesan, Ovidiu ; Van Den Keybus, Jeroen: Automotive Ethernet: In-vehicle networking and smart mobility. In: *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2013, S. 1735–1739

[9] Hoyle, Ben ; Rau, Markus M. ; Paech, Kerstin ; Bonnett, Christopher ; Seitz, Stella ; Weller, Jochen: Anomaly detection for machine learning redshifts applied to SDSS galaxies. In: *Monthly Notices of the Royal Astronomical Society* 452 (2015), 08, Nr. 4, S. 4183–4194. – URL https://doi.org/10.1093/mnras/stv1551. – ISSN 0035-8711

[10] KimiNewt: *PyShark*. https://kiminewt.github.io/pyshark/. 2021

[11] Lima, Moisés F. ; Zarpelão, Bruno B. ; Sampaio, Lucas D. H. ; Rodrigues, Joel J. P. C. ; Abrão, Taufik ; Proença, Mario L.: Anomaly detection using baseline and K-means clustering. In: *SoftCOM 2010, 18th International Conference on Software, Telecommunications and Computer Networks*, 2010, S. 305–309

[12] Meyer, Philipp ; Hackel, Timo ; Langer, Falk ; Stahlbock, Lukas ; Decker, Jochen ; Eckhardt, Sebastian A. ; Korf, Franz ; Schmidt, Thomas C. ; Schüppel, Fabian: Demo: A Security Infrastructure for Vehicular Information Using SDN, Intrusion Detection, and a Defense Center in the Cloud. In: *2020 IEEE Vehicular Networking Conference (VNC)*, 2020, S. 1–2

[13] Müter, Michael ; Asaj, Naim: Entropy-based anomaly detection for in-vehicle networks. In: *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, S. 1110–1115

[14] Noble, William S.: What is a support vector machine? In: *Nature Biotechnology* 24 (2006), Dec, Nr. 12, S. 1565–1567. – URL https://doi.org/10.1038/nbt1206-1565. – ISSN 1546-1696

[15] Pedregosa, F. ; Varoquaux, G. ; Gramfort, A. ; Michel, V. ; Thirion, B. ; Grisel, O. ; Blondel, M. ; Prettenhofer, P. ; Weiss, R. ; Dubourg, V. ; Vanderplas, J. ; Passos, A. ; Cournapeau, D. ; Brucher, M. ; Perrot, M. ;

Duchesnay, E.: Scikit-learn: Machine Learning in Python. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830

[16] SecVI: *Security for Vehicular Information*. 2018. – URL https://secvi.inet.haw-hamburg.de/. – Zugriffsdatum: 2021-06-25

[17] Steinbach, Till: *Ethernet-basierte Fahrzeugnetzwerkarchitekturen für zukünftige Echtzeitsysteme im Automobil*. Wiesbaden : Springer Vieweg, Oktober 2018. – ISBN 978-3-658-23499-7

[18] Wyk, Franco van ; Wang, Yiyang ; Khojandi, Anahita ; Masoud, Neda: Real-Time Sensor Anomaly Detection and Identification in Automated Vehicles. In: *IEEE Transactions on Intelligent Transportation Systems* 21 (2020), Nr. 3, S. 1264–1276