HAW HAMBURG

BACHELORTHESIS
Seyed Mohammad Ehsan Sajadi

Simulation und Evaluation ausgewählter Ethernet Angriffe in Echtzeitfahrzeugnetzwerken

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Seyed Mohammad Ehsan Sajadi

Simulation und Evaluation ausgewählter Ethernet Angriffe in Echtzeitfahrzeugnetzwerken

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Informatik Technischer Systeme*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Franz Korf
Zweitgutachter: Prof. Dr. Martin Hübner

Eingereicht am: 07. Juni 2024

Seyed Mohammad Ehsan Sajadi

Thema der Arbeit

Simulation und Evaluation ausgewählter Ethernet Angriffe in Echtzeitfahrzeugnetzwerken

Stichworte

TSN, Fahrzeugnetz, Echtzeit-Ethernet, Ethernet-Angriffe, OMNeT++, Simulation

Kurzzusammenfassung

Moderne Fahrzeugnetze basieren auf Echtzeit-Ethernet, welches eine hohe Zuverlässigkeit und Zeitgenauigkeit für die Kommunikation zwischen kritischen Fahrzeugsystemen gewährleistet. Dadurch können Daten effizient und sicher zwischen den Komponenten eines Fahrzeugs übertragen werden. Andererseits werden diese Fahrzeugnetze durch die potenziellen Risiken von Ethernetangriffen bedroht. Diese Bedrohungen können in Echtzeitnetzwerken ungewöhnliche Effekte haben und Störungen in der Kommunikation zwischen Fahrzeugkomponenten verursachen. In dieser Arbeit werden solche möglichen Netzwerkangriffe vorgestellt und mittels verschiedener Angriffsszenarien in der Simulationsumgebung OMNeT++ simuliert und evaluiert. . . .

Seyed Mohammad Ehsan Sajadi

Title of Thesis

Simulation and evaluation of selected Ethernet attacks in real-time vehicle networks

Keywords

TSN, vehicle network, real-time Ethernet, Ethernet attacks, OMNeT++, simulation

Abstract

Modern vehicle networks are based on real-time Ethernet, which guarantees high reliability and time accuracy for communication between critical vehicle systems. This allows data to be transferred efficiently and securely between the components of a vehicle. On the other

hand, these vehicle networks are threatened by the potential risks of Ethernet attacks. These threats can have unusual effects in real-time networks and cause disruptions in communication between vehicle components. In this thesis, such possible network attacks are presented and simulated and evaluated using various attack scenarios in the OMNeT++ simulation environment. . . .

Inhaltsverzeichnis

Abbildungsverzeichnis **vii**

Tabellenverzeichnis **ix**

Abkürzungen **x**

1 Einleitung **1**
1.1 Motivation 1
1.2 Zielstellung 2
1.3 Aufbau der Arbeit 2

2 Grundlagen **3**
2.1 Echtzeit Ethernet 3
2.1.1 Time Sensitive Networking 3
2.1.2 IEEE802.1Q-2018 4
2.2 Netzwerkangriffe 5
2.2.1 DoS Angriff 5
2.2.2 Man-in-the-Middle Angriff 5
2.2.3 MAC Flooding 6
2.2.4 EtherOops 7

3 IST-Zustand **11**
3.1 Simulationsumgebung 11
3.2 Simuliertes Fahrzeugnetzwerk 13

4 Verwante Arbeiten **15**
4.1 Network Anomaly Detection in Cars: A Case for Time-Sensitive Stream Filtering and Policing 15
4.2 Integration realer Angriffe in simulierte Echtzeit- Ethernet-Netzwerke . . . 15

4.3 Automotive Ethernet Real-Time Observer for Anomaly Detection in In-Vehicle Networks . . . 16

5 Simulation von Angriffen in Fahrzeugnetzen 18

5.1 Denial-of-Service Angriff . . . 18

5.2 Man-in-the-Middle Angriff . . . 26

5.2.1 Manipulation . . . 26

5.2.2 Redirection . . . 37

5.2.3 Elimination . . . 42

5.2.4 Rescheduling . . . 47

5.3 MAC Flooding Angriff . . . 52

5.4 EtherOops . . . 61

6 Fazit und Ausblick 71

6.1 Fazit . . . 71

6.2 Ausblick . . . 72

Literaturverzeichnis 73

A Anhang 75

Glossar 76

Selbstständigkeitserklärung 77

Abbildungsverzeichnis

2.1 Standard Ethernet Frame-Format mit der Erweiterung von IEEE802.1Q (Quelle: T. Steinbach[13]) 4
2.2 Man-in-the-Middle Angriffsmuster 6
2.3 Die Struktur eine Ethernetframe 8
2.4 Ethernetframe in Paket-in-Paket Angriff 9

3.1 Die erweiterte Frameworks in Simulationsumgebung OMNeT++ 12
3.2 Die Topologie des verwendeten Fahrzeugnetzwerks 13

5.1 DoS Angriff in RecBarToZone Netzwerk 19
5.2 Ergebnisse für die Ausführung DoS-Angriff im RecBarToZone-Netzwerk . 21
5.3 Die Latenzzeit der Ethernetframes in Switch-Center 23
5.4 Manipulation in Gateway für Szenario 1 in RecBarToZone Netzwerk . . . 27
5.5 Manipulation im Switch für Szenario 2 in RecBarToZone Netzwerk 28
5.6 Manipulation in Gateway in RecBarToZone Netzwerk 33
5.7 Manipulation in Switch in RecBarToZone Netzwerk 34
5.8 Redirection in RecBarToZone Netzwerk 38
5.9 Grafische Darstellung der Anzahl der gesendeten, empfangenen und komprimierten Pakete durch Ridirection-Simulation 41
5.10 Elimination in RecBarToZone Netzwerk 44
5.11 Grafische Darstellung der Anzahl der gesendeten, empfangenen und eliminiereten Pakete durch Elimination 45
5.12 Reschaduling in RecBarToZone Netzwerk 47
5.13 Die Latenzzeit empfangener Nachrichten bei der Sensorfusion unter Normalbedingungen und im Falle eines Angriffs 50
5.14 MAC Flooding in RecBarToZone Netzwerk 52
5.15 Modifizierte Module im Switch bei CoRE4INET 54
5.16 Anzahl der gesendeten/empfangenen Pakete pro Ports im Switch Front . . 56
5.17 Paket-in-Paket Angriff in RecBarToZone Netzwerk 61

5.18 Die Paket-in-Paket-Anwendung eines Moduls im Rahmen von CoRE4INET 62
5.19 Die BitFlip-Anwendung eines Moduls im Rahmen von CoRE4INET 63
5.20 Die Route der Paket-in-Paket-Frames im Netzwerk 67
5.21 Berechnung der Wahrscheinlichkeit eines Bitflips im SFD-Byte 69
5.22 Berechnung der Wahrscheinlichkeit eines Bitflips 69
5.23 Die Route der Paket-in-Paket-Frames im Netzwerk 69

Tabellenverzeichnis

2.1 Armis-Umfrageergebnisse von Bitfehlerrate in Ethernetkabeln 10

5.1 Die durchschnittliche und maximale Latenzzeit von Port 3 am Switch-Front 24
5.2 Die durchschnittliche und maximale Latenzzeit von Port 4 am Switch-Center/ von Port 5 am Switch-Rear 25
5.3 Die Jitter und maximale Latenzzeit von Port 1 am Switch-Center 36
5.4 Die maximale empfangene Latenzzeit von Left Lidarsensor 42
5.5 Die maximale Latenzzeit der Fusi-Komponente 51
5.6 Die maximale Latenzzeit am Port 4 des Switch_Center 51
5.7 Die Differenz der Paketanzahl pro Port von Switch-Front 57
5.8 Anzahl der gesendeten und empfangenen Paket-in-Paket 67

Abkürzungen

AVB Audio Video Bridging.

CAN Controller Area Network.

CoRE Communication over Realtime Ethernet.

DEI Drop Eligible Indicator.

DoS Denial-of-Service.

ECU Electronic Control Units.

FAS Fahrerassistenzsysteme.

FIB Forwarding-Informationsbasis.

HAW Hochschule für Angewandte Wissenschaften.

MitM-Angriff Man-in-the-Middle Angriff.

PCP Priority Code Point.

PSFP Per-Stream Filtering and Policing.

SFD Start Frame Delimiter.

SRP Stream Reservation Protocols.

TCI Tag Control Information.

TPID Tag Protocol Identifier.

TSN Time Sensitive Networking.

TTE Time-Triggered Ethernet.

VID VLAN-Identifier.

VLAN virtuelle lokale Netzwerke.

1 Einleitung

1.1 Motivation

Die Fahrzeugtechnologie hat sich erheblich weiterentwickelt und moderne Fahrzeuge sind heutzutage mit verschiedenen elektronischen Systemen ausgestattet. Diese Fahrzeugsysteme erfordern eine größere Datenmenge. Die Traditionelle Bussysteme wie das Controller Area Network (CAN) sind oft nicht mehr in der Lage, die notwendigen Anforderungen an Bandbreite und Latenzzeit zu erfüllen. Deshalb bietet sich Ethernet als eine robuste und sichere Netzwerklösung für diese Herausforderungen an. Diese Fahrzeugnetzwerke basieren auf Echtzeit-Ethernet, das eine hohe Zuverlässigkeit und Zeitgenauigkeit für die Kommunikation zwischen kritischen Fahrzeugsystemen gewährleistet. Der Einsatz von Echtzeit-Ethernet unterstützt die modernen Fahrzeugsysteme wie z.B. Fahrerassistenzsysteme (FAS), die eine geringe Latenz benötigen. Daher können Datenmengen über Echtzeit-Ethernet zwischen den Komponenten zeitgerecht und mit minimaler Verzögerung übertragen werden. Angesichts der Motivation für die Erforschung und Weiterentwicklung der Fahrzeugsysteme mit Echtzeit-Ethernet entsteht zunehmend die Bedrohung von Ethernet-Angriffe, die die Sicherheit und Funktionalität dieser Fahrzeugnetze gefährden können.

Die Ethernet-Angriffe können in Echtzeitnetzwerken ungewöhnliche Effekte auslösen und Störungen in der Kommunikation zwischen Fahrzeugkomponenten verursachen. Das Ziel solcher Angriffe besteht darin, das Fahrzeugnetz zu überlasten oder wichtige Fahrzeugkomponenten lahmzulegen, indem sie Netzwerkressourcen wie Bandbreite aufbrauchen oder die Übertragung zeitkritischer Daten verzögern. Andererseits können diese Angriffe die Daten manipulieren und zu falschen Informationen führen, die die Entscheidungsfindung von Fahrzeugsystemen wie FAS-System stören. Dies kann besonders gefährlich sein, wenn solche Angriffe den Angreifern Zugriff auf die Kontrolle über Fahrzeugfunktionen verschaffen.

1.2 Zielstellung

Das Ziel dieser Arbeit ist es, exemplarische Auswirkungen ausgewählter Netzwerkangriffe zu evaluieren. Diese Angriffe werden in verschiedenen Szenarien durchgeführt und schrittweise beschrieben. Sie werden mithilfe des diskreten Ereignissimulators OMNeT++ implementiert und simuliert. Diese Arbeit entstand im Rahmen der Communication-over-Realtime-Ethernet Research Group (CoRE RG) der HAW Hamburg, die sich mit der Erforschung von Echtzeit-Ethernet-Kommunikation in Fahrzeugen beschäftigt.

1.3 Aufbau der Arbeit

Diese Arbeit ist in sechs Kapitel untergliedert. Kapitel 2 vermittelt das grundlegende Wissen, das für diese Arbeit erforderlich ist. Es behandelt Themen wie Time-Sensitive Networking und Netzwerkangriffe. In Kapitel 3 wird ein umfassender Überblick über die technische und analytische Umgebung gegeben, die für die Durchführung, Analyse und Bewertung genutzt wird. Kapitel 4 befasst sich mit vorangegangenen Arbeiten, die Netzwerkangriffe thematisieren. In Kapitel 5 werden die praktischen Herausforderungen im Kontext der Netzwerksimulation illustriert. Der Aufbau eines Angriffs wird anhand eines Szenarios beschrieben. Dabei werden alle relevanten Schritte erläutert, die zur Realisierung des Angriffsszenarios erforderlich waren. Details zur Implementierung, Konfiguration sowie Durchführung werden in diesem Kapitel dargelegt. Abschließend wird der Angriff validiert und seine Ergebnisse werden evaluiert. Zum Schluss bietet Kapitel 6 einen Ausblick über den Inhalt dieser Arbeit hinaus und diskutiert zukünftige Erweiterungsmöglichkeiten sowie Anwendungen.

2 Grundlagen

In diesem Kapitel wird das grundlegende Wissen vermittelt, das dieser Arbeit zugrunde liegt. Zu Beginn erfolgt eine kurze Einführung in das aktuelle Echtzeit-Ethernet und das Time-Sensitive Networking (TSN). Des Weiteren wird auf die Echtzeiterweiterung von Ethernet eingegangen. Abschließend werden die Grundlagen und Definitionen zu möglichen Netzwerkangriffen erläutert.

2.1 Echtzeit Ethernet

Der Standard-Ethernet bietet keine Garantie für die Übertragung, da er nach dem Best-Effort-Prinzip arbeitet und keine Annahmen über die zu übertragenden Daten macht. In den oberen Schichten des OSI-Modells werden derartige Dienste eingeführt. Ethernet ist aufgrund seiner Flexibilität, Skalierbarkeit und Kosteneffizienz besonders attraktiv für Echtzeitkommunikation[13]. Aus dieser Motivation heraus stellt Echtzeit-Ethernet eine revolutionäre Entwicklung im Bereich der Kommunikation von Fahrzeugsystemen dar. Im Vergleich zu anderen Technologien wie CAN ermöglicht Echtzeit-Ethernet eine höhere Zuverlässigkeit und Zeitgenauigkeit. In den folgenden Abschnitten wird näher auf Time Sensitive Networking (TSN) sowie den Standard IEEE 802.1Q-2018 eingegangen.

2.1.1 Time Sensitive Networking

Die folgenden Erläuterungen zu Time Sensitive Networking (TSN) stützen sich auf die Arbeit der TSN-Task-Group [15]. TSN ist ein Protokoll für Echtzeit-Ethernet und umfasst eine Reihe von IEEE802-Standards, die deterministische Dienste über IEEE802-Netzwerke bereitstellen. Das Hauptziel von TSN ist es, einen Pakettransport zu gewährleisten, der durch begrenzt niedrige Latenzzeiten, minimale Paketverzögerungen und geringen Paketverlust gekennzeichnet ist. TSN definiert zudem eine Erweiterung des Ethernets für

die Echtzeitdatenübertragung, bekannt unter dem Standard IEEE802.1Q-2018, der viele der TSN-Substandards beinhaltet.

2.1.2 IEEE802.1Q-2018

Die folgende Erklärung zum IEEE802.1Q-2018 basiert auf der Arbeit von T. Steinbach[13]. Der Standard IEEE802.1Q-2018 ist auch als 802.1Q bekannt und stellt eine Erweiterung des Ethernet dar, um Echtzeitfähigkeit zu ermöglichen. Wie in der Abbildung 2.1 zu sehen ist, wird bei 802.1Q der Header des Ethernet-Frames um ein zusätzliches Feld erweitert. Dieses Feld vergrößert den Ethernet-Frame um 4 Bytes, wodurch die Ethernet-Frames priorisiert und in virtuelle lokale Netzwerke (VLAN) eingeordnet werden können.
Mithilfe der Abbildung 2.1 ist zu erkennen, dass dieses Feld in 2 Bytes für den Tag Protocol Identifier (TPID) sowie 2 Bytes für die Tag Control Information (TCI) aufgeteilt ist. Das TPID-Feld wird konstant mit dem Wert 0x8100 belegt und zeigt an, dass es sich um ein Q-Frame handelt. Das TCI-Feld ist in 3 Bits für den Priority Code Point (PCP), ein Bit für den Drop Eligible Indicator (DEI) und 12 Bits für den VLAN-Identifier (VID) unterteilt. Das PCP-Feld ist 3 Bits groß und ermöglicht es, jedem Frame eine von 8 Prioritätsklassen zuzuweisen. Das DEI-Feld ist 1 Bit groß und gibt an, ob das Ethernet-Frame im Falle eines Staus verworfen werden darf. Das VID-Feld ist 12 Bits groß und erlaubt es, jedem Frame eines von 4094 möglichen virtuellen Netzwerken zuzuweisen. Diese Werte sind reserviert.

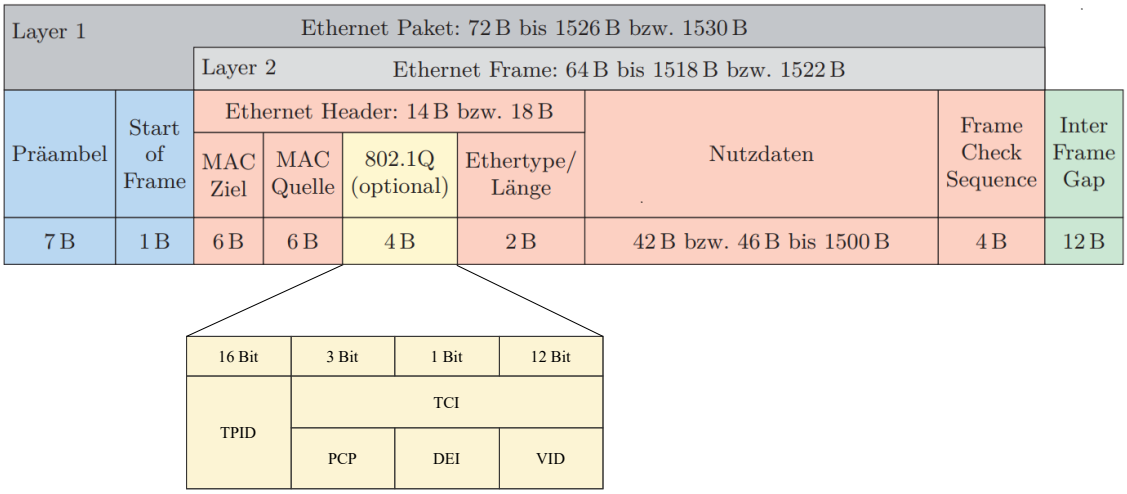

Abbildung 2.1: Standard Ethernet Frame-Format mit der Erweiterung von IEEE802.1Q (Quelle: T. Steinbach[13])

In Netzwerkgeräten wie Switches funktioniert das Priorisierunsverfahren von 802.1Q so, dass ein Frame mithilfe von PCP-Fled eine von acht verschiedenen Prioritätklassen erhält. Diese Prioritätsklasse bestimmt für den Scheduler, in welcher Reihenfolge der nächste Frame verschickt werden soll. Die Frames mit derselben Prioritätsklasse werden nach dem FIFO-Prinzip (First In, First Out) behandelt. Diese Erweiterung hat den Nachteil, dass Frames mit niedrigen Prioritäten durch Frames mit höheren Prioritäten verzögert oder verdrängt werden können.

2.2 Netzwerkangriffe

Ein Netzwerkangriff ist ein gezielter Versuch, um ein Netzwerk einzudringen, es zu manipulieren oder zu stören. Das Ziel solcher Angriffe besteht darin, Daten zu stehlen, Schadsoftware zu verbreiten, das Netzwerk lahmzulegen oder andere Schaden anzurichten. Netzwerkangriffe können in verschiedenen Formen durchgeführt werden. In diesem Abschnitt werden die Angriffe vorgestellt, die für diese Arbeit verwendet wurden.

2.2.1 DoS Angriff

Die folgende Erklärungen basiert auf den Informationen von BSI und CrowdStrike-Unternehmen [4, 5]. Ein Denial-of-Service (DoS) beschreibt im Wesentlichen das Unzugänglichmachen oder die Außerbetriebsetzung eines Dienstes. Technisch gesehen, zielt ein DoS-Angriff darauf ab, einen Computer oder ein Netzwerk derart zu beeinträchtigen, dass die legitimen Nutzer keinen Zugriff mehr auf Informationssysteme, Geräte oder Netzwerkressourcen haben. Das Hauptziel eines DoS-Angriffs besteht darin, die Kapazität des angezielten Rechners oder Netzwerks mit einer Flut von Anfragen zu überlasten. Dies führt dazu, dass der normale Datenverkehr nicht mehr verarbeitet werden kann oder einfach das Ziel abstürzt. Charakteristisch für einen DoS-Angriff ist, dass er von einem einzigen Computer und über eine einzige Verbindung ausgeführt wird.

2.2.2 Man-in-the-Middle Angriff

Die folgenden Erklärungen basieren auf Informationen von Security Insider[8]. Es gibt viele Arten von Bedrohungen, die ein Angreifer für seine Zwecke nutzen kann. Einige dieser Angriffe können sowohl mit automatisierter Software als auch mit physischen

Geräten durchgeführt werden. In diesem Abschnitt wird die Grundidee eines Man-in-the-Middle-Angriffs(MitM-Angriff) erläutert und in Kapitel 4.2 werden weiterführende Angriffstechniken vorgestellt.

Ein Man-in-the-Middle-Angriff ist eine Form des Lauschangriffs, bei dem ein Angreifer eine laufende Kommunikation oder Datenübertragung unterbricht. Nachdem er sich in die Übertragung eingeschaltet hat, gibt sich der Angreifer als die beiden legitimen Teilnehmer oder Systeme aus. Auf diese Weise ist es ihm möglich, Informationen und Daten von beiden Seiten abzufangen und gleichzeitig schädliche Links oder andere Daten an die legitimen Teilnehmer oder Systeme zu senden. Bei einem MitM-Angriff fängt der Angreifer eine Datenübertragung zwischen einem Client und einem Server ab, indem er sowohl dem Client einredet, dass die Kommunikation mit dem Server weiterhin besteht, als auch dem Server einredet, dass er weiterhin Informationen vom Client erhält(sieh Abbildung 2.2). Der Angreifer kann Daten abfangen und eigene, gefälschte Informationen in die zukünftigen Übertragungen einfügen.

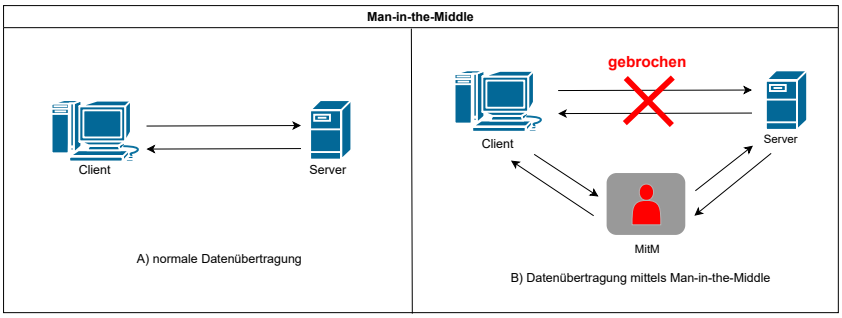

Abbildung 2.2: Man-in-the-Middle Angriffsmuster

2.2.3 MAC Flooding

In einem Netzwerk hat jeder Layer2-Switch einen Mechanismus zur Speicherung des Ports, an den eine bestimmte MAC-Adresse angeschlossen ist. Dadurch wird sichergestellt, dass Unicast-Kommunikation zwischen zwei Hosts stattfinden kann, ohne dass andere Hosts den Datenverkehr sehen. Laut RFC3222[14] ist die Forwarding-Informationsbasis (FIB) oder Routing-Tabelle eine wichtige Komponente in Netzwerkgeräten wie Routern und Switches. Die FIB ist eine dynamische Tabelle mit fester Größe. Sie enthält mindestens die Schnittstellenkennung und Informationen über den nächsten Hop für jedes erreichbare Ziel. Im Switch speichert sie die MAC-Adressen und die zugehörigen VLAN-IDs der verschiedenen Hosts, die an den Switch angeschlossen sind.

Die Technik, die es einem Angreifer ermöglicht, den gesamten Datenverkehr abzufangen, wird als MAC-Flooding bezeichnet. Das Ziel von MAC-Flooding ist es, einen Konstruktionsfehler einiger Switches auszunutzen, nämlich die Tatsache, dass FIB-Tabelle von Switch nur über eine begrenzte Menge an Speicher verfügen. Bei dieser Technik wird der Switch mit verschiedenen MAC-Quelladressen überflutet, bis diese Tabelle zusammenbricht [17]. Ein MAC-Flooding-Angriff ist eine Art von Denial-of-Service-Angriff. Dabei sendet der Angreifer eine große Anzahl von Paketen, jede mit einer zufällige Quell-MAC-Adresse, um diese Tabelle schnell zu füllen. Da die FIB-Tabelle nur eine begrenzte Größe hat, hat sie irgendwann keinen Platz mehr für neue MAC-Adressen. Wenn dieser Speicher voll wird, gehen einige Switches in einen Fail-Open-Zustand über. In diesem Zustand wandelt sich der Switch funktional zu einem Hub. Dies ermöglicht der Angreifer den gesamten Datenverkehr ausspionieren[11, 17].

2.2.4 EtherOops

In diesem Abschnitt wird Bezug auf das technische Paper des Forschungsteams von Armis genommen[1]. Die Armis-Forscher hat eine neue Methode entdeckt, um eine Sicherheitslücke in Ethernet-Kabeln auszunutzen. Diese Technik kann dazu verwendet werden, perimetrische Sicherheitsvorrichtungen wie Firewalls und NATs zu umgehen.

Der EtherOops-Angriff ist im Grunde ein Paket-in-Paket-Angriff, bei dem mehrere Netzwerkpakete ineinander verschachtelt sind. Während die äußere Schicht gutartige Pakete enthält, tragen die inneren Schichten Schadcode oder Befehle. Das äußere Paket ermöglicht es der bösartige Nutzlast, die Netzwerkverteidigung, wie Firewalls oder andere Sicherheitsprodukte, zu umgehen, während das innere Paket Geräte innerhalb des Netzwerks angreift. Dabei sollte jedoch berücksichtigt werden, dass Netzwerkpakete normalerweise nicht ihre Struktur ändern oder ihre äußere Schicht verlieren.Dies kann passieren, wenn ein Fehler im Kabel auftritt. Die Forscher von Armis sind der Auffassung, dass fehlerhafte Kabel entweder aufgrund von unvollkommener Verkabelung oder beschädigter Kable unter unerwünschten elektrischen Störunge leiden. Dies führt zur Zerstörung der äußeren Hülle und zur Aktivierung interner Nutzlasten.

Paket-in-Paket Angriff in Ethernet

Um ein Verständnis für den Paket-in-Paket-Angriff in einem Ethernet-Netzwerk zu entwickeln, werfen wir einen genaueren Blick auf die Struktur eines Ethernet-Frames. Die

untere Abbildung 2.3 zeigt, welche Daten tatsächlich als Teil eines Ethernet-Packet auf der Leitung erscheinen und wie sie verarbeitet werden. Zunächst wird eine Präambel(Bytes mit dem Wert 0x55) und ein Start Frame Delimiter (SFD)(Byte mit dem Wert 0xD5) vorangestellt, die beide Teil des Ethernet-Pakets auf der physikalischen Schicht sind.

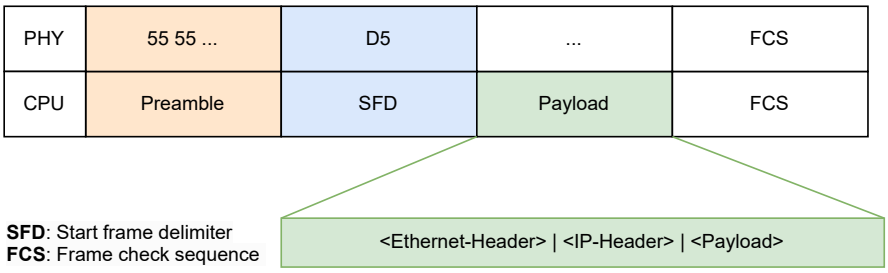

Abbildung 2.3: Die Struktur eine Ethernetframe

Die CPU wartet auf ein SFD-Byte und alle Bytes vor diesem Byte werden als Präambel behandelt. Nach diesen ersten Bytes folgt die bekannte Nutzlast der Schicht 2, das ist der Ethernet-Rahmen mit Header und Nutzlast. Der Frame endet mit einer Rahmenprüfsequenz (FCS), einer zyklischen 32-Bit-Redundanzprüfung, die dazu dient, eine eventuelle Beschädigung der Daten während der Übertragung zu erkennen. Da die MAC-Schicht für das MAC-Framing (Präambel, SFD und FCS) verantwortlich ist, die die Gültigkeit des FCS überprüft, und nicht die PHY-Schicht. Das bedeutet, dass potenziell beschädigte Frames den ganzen Weg bis zur CPU zurücklegen können, ohne von der PHY-Schicht verworfen zu werden.

Bei einem Paket-in-Paket-Angriff durchlaufen die Verarbeitungsstufen eines Ethernet-Pakets verschiedene Schritte. In der Abbildung 2.4 ist es zu zeigen, dass beim Ethernet-Paket das SFD Byte auf der Leitung verfälscht wurde.In diesem Fall wurde das 0xD5-Byte beschädigt und z. B. in ein 0xD4 umgewandelt. Aus der Sicht der MAC-Schicht der CPU wartet darauf, bis ein SFD-Byte empfangen wird. Da das erste SFD-Byte verfälscht wird, wartet die MAC-Schicht weiterhin darauf.

Wenn das Paket speziell erstellt wurde, könnte der Angreifer ein 0xD5 irgendwo innerhalb der Nutzlast des Pakets platzieren. Wenn dies das erste 0xD5-Byte im Rahmen ist, wird es als SFD aufgefasst. Alle vorherigen Bytes werden als Präambelsequenz betrachtet Wie oben erwähnt wurde, ist die MAC-Schicht-Aufgabe die Gültigkeit des FCS zu überprüfen. Trotz der Bit-Flip des SFD-Bytes hat es geschafft, den Anfang des Pakets in die Nutzlast des Angreifers zu verschieben, können das Ende des Pakets und der erwartete CRC-Wert des Pakets nicht vom Angreifer kontrolliert werden.

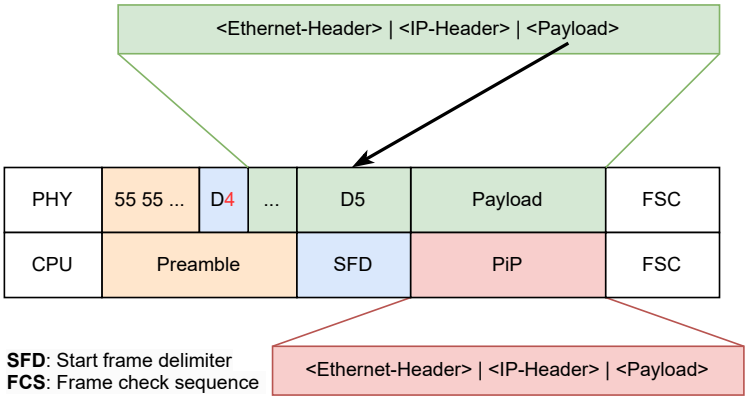

Abbildung 2.4: Ethernetframe in Paket-in-Paket Angriff

Damit eine SFD-Beschädigung in ein Paket-in-Paket-Szenario umgewandelt werden kann, müssen daher zwei Bedingungen erfüllt sein:

- Einige SFD-Byte-Beschädigungen führen nicht zu einem gültigen Datensymbol, aber einige davon schon, und da Bitflops zufällig auftreten, landen sie schließlich auf gültigen Datensymbolen.
- Der CRC32 am Ende des Rahmens muss sowohl für das ursprüngliche Paket als auch für das neue innere Paket korrekt sein. Der Angreifer kann die FCS(CRC32) des ursprünglichen Pakets im Voraus berechnen und eine CRC-Kollision zwischen dem äußeren und dem inneren Paket verursachen, wenn er den genauen Inhalt der Kopfzeilen des ursprünglichen Pakets kennt.

Bit-Fehler in Ethernet

Damit dieser Angriff funktioniert, ist es erforderlich, dass auf den Ethernetkabeln zufällige Bitfehler stattfinden. Zudem besteht ein wesentlicher Unterschied zwischen der von der Industrie festgelegten akzeptablen Bitfehlerrate (Bit-Error-Rate, BER) und der tatsächlichen Fehlerhäufigkeit in realen Ethernetkabeln. Laut Industriestandard ist die maximale Bitfehlerrate bei Gigabit-Ethernetkabeln 1-Bitfehler in 10 Milliarden Bits. Im Gegensatz dazu haben Forscher von Armis in ihrer Studie festgestellt, dass Bitfehler in Ethernetkabeln bei einer Vielzahl von Unternehmen tatsächlich häufiger auftreten. Ihre Daten, die rund 500.000 aktive Switch-Ports in Unternehmensnetzwerken umfassen, zeigen, dass etwa 1,8% dieser Ports eine signifikant höhere Bitfehlerrate aufweisen als

vom Industriestandard vorgesehen sind. Die von Armis identifizierten Kabel mit hoher Fehlerrate überschreiten die festgelegte maximale Bitfehlerrate deutlich. Bei solchen Kabeln kann ein Packet-in-Packet-Angriff innerhalb einer angemessenen Zeitspanne, d.h. Stunden oder weniger, durchgeführt werden.

Bereich	% der High-Error-Kabel
Medical	1,75%
OT	1,20%
Others	2,26%
Grand Total	1,79%

Tabelle 2.1: Armis-Umfrageergebnisse von Bitfehlerrate in Ethernetkabeln

Die Tabelle 2.1 zeigt die variierenden Bitfehlerquoten je nach verschiedenen Bereichen innerhalb der Installationsbasis von Armis. In Gigabit-Ethernet-Kabeln mit einer hohen Bitfehlerrate kann ein Bit etwa einmal alle 10 Sekunden umkippen. Ein Packet-in-Packet-Angriff kann unter diesen Umständen innerhalb von Stunden oder sogar weniger erfolgen.

Ein Bitfehler in einem Ethernet-Kabel kann durch folgende mögliche Ursachen hervorgerufen werden:

- **Physische Schäden:** Dies ist eine häufige Ursache. Risse, Schnitte, Knicke oder übermäßiges Biegen des Kabels können die internen Drähte beschädigen und Signalstörungen verursachen.
- **Elektromagnetische Interferenz (EMI):** Externes elektrisches Rauschen von Quellen wie Stromleitungen, Motoren oder Funkwellen kann das Signal im Kabel stören und Bitfehler verursachen.
- **Crosstalk (Nebensprechen):** Dies tritt auf, wenn sich elektrische Signale von benachbarten Kabeln (innerhalb desselben Bündels) überlagern und unerwünschtes Rauschen erzeugen, das die Datenintegrität in Ihrem Ethernet-Kabel beeinträchtigt. Crosstalk kann zwischen den einzelnen Adernpaaren eines Kabels oder zwischen benachbarten Kabeln auftreten.

3 IST-Zustand

3.1 Simulationsumgebung

Die folgende Erläuterung der Simulationsumgebung OMNeT++ orientiert sich am Simulationshandbuch von OMNeT++[9]. OMNeT++ steht für Objective Modular Network Testbed und ist ein objektorientiertes Framework für Netzwerksimulationen. Dieses Framework ist ereignisgesteuert und ermöglicht das Erstellen von Modulen durch die Verwendung einer speziellen Sprache namens NED (Network Description). Ein Modul kann beispielsweise eine Queue, ein Port oder eine Anwendung sein. Ein Modell wird in OMNeT++ als Netzwerk definiert, das aus verschiedenen zusammengesetzten Modulen besteht. Diese Module kommunizieren untereinander mittels Message Passing. Das Verhalten der Module wird in der Programmiersprache C++ implementiert.

OMNeT++ selbst ist kein spezialisierter Simulator für einen bestimmten Zweck, sondern bietet eine modulare und erweiterbare Simulationsbibliothek, die für die Erstellung verschiedener Simulationen dient. Erweitert wird diese Basis mit weiteren Simulationsframeworks:

- **CoRE4INET** ist ein Framework, das von der Arbeitsgruppe Communication over Realtime Ethernet (CoRE) entwickelt wurde[2]. Es stellt eine Erweiterung des INET-Frameworks dar und umfasst Implementierungen von Audio Video Bridging (AVB) und Time-Triggered Ethernet (TTE), mit deren Hilfe bereits TSN-Technologien (Time-Sensitive Networking) simuliert wurden.

- **INET**[10] ist ein spezielles Framework für OMNeT++, das für die Simulation von Kommunikationsnetzwerken konzipiert wurde. Es implementiert die Standard-Ethernet- und Internetprotokoll-Technologien.

- **INEFiCo4OMNeTT**ist ein Framework, das von der CoRE Arbeitsgruppe entwickelt wurde. Es ist ein Open-Source-Simulationsmodell für die ereignisbasierte

Simulation von Feldbustechnologien in OMNeT++. Es ermöglicht die Simulation von CAN- und Flexray-Technologien. Dies Framework ist in der Core Arbeitsgruppe entstanden.

- **SignalsAndGateways**[3] ist ein Framework, das von der Arbeitsgruppe entwickelt wurde. Es verwendet CoRE4INET, INET und FiCo4OMNeT. Dies ermöglicht eine heterogene Netzwerksimulation in OMNeT++ Simulationssystem mit Hilfe von Gateway-Komponenten, die die Kommunikation zwischen Ethernet- und Bus-Technologien zu ermöglichen.

Die Abbildung 3.1 stellt die Abhängigkeiten in der Struktur der Simulationsumgebung dar. Das INET-Framework basiert auf OMNeT++. CoRE4INET nutzt die Standard-Ethernet-Implementierung aus INET und erweitert sie um die Realisierung von Echtzeit-Ethernet-Protokollen. Für die Simulationen werden die Versionen OMNeT++ 5.5.1, INET-Framework 3.7.1 und das CoRE4INET-, SignalsAndGateways- und FiCo4OMNeT-Framework vom 15.Oktober 2021 verwendet.

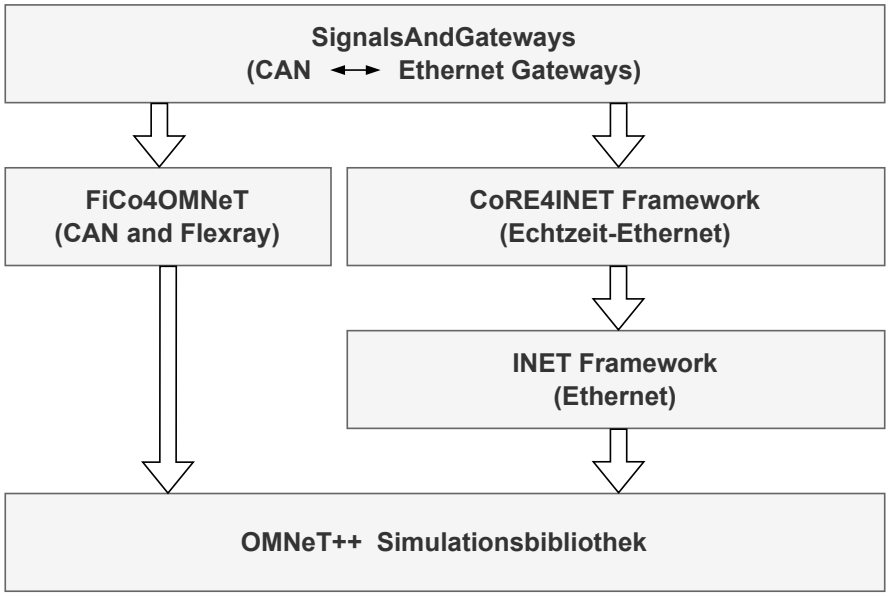

Abbildung 3.1: Die erweiterte Frameworks in Simulationsumgebung OMNeT++

3.2 Simuliertes Fahrzeugnetzwerk

Der folgende Abschnitt orientiert sich aud die Arbeit von S. Reider[12]. Die Simulation in dieser Arbeit dient als Grundlage für die Integration von Angriffen in Echtzeit-Ethernet-Netzwerke. Das RECBAR-Projekt ist ein Forschungsprojekt von Hochschule für Angewandte Wissenschaften (HAW) und steht als Akronym für Ethernet-Backbone für Autos. Ziel des RECBAR-Projekts ist es, intelligente Backbone-Architekturen zu entwerfen, zu entwickeln und zu testen, die in zukünftigen Automobilkommunikationsinfrastrukturen zum Einsatz kommen. RECBAR entwickelt ein neues Simulationsframework, das auf OMNeT++ und der INET-Erweiterung für Time Triggered Ethernet-Infrastrukturen basiert. Es umfasst alle verbauten Steuergeräte sowie die zwischen ihnen ausgetauschten Nachrichten. Die Abbildung 3.2 zeigt die Topologie des verwendeten Fahrzeugnetzwerks.

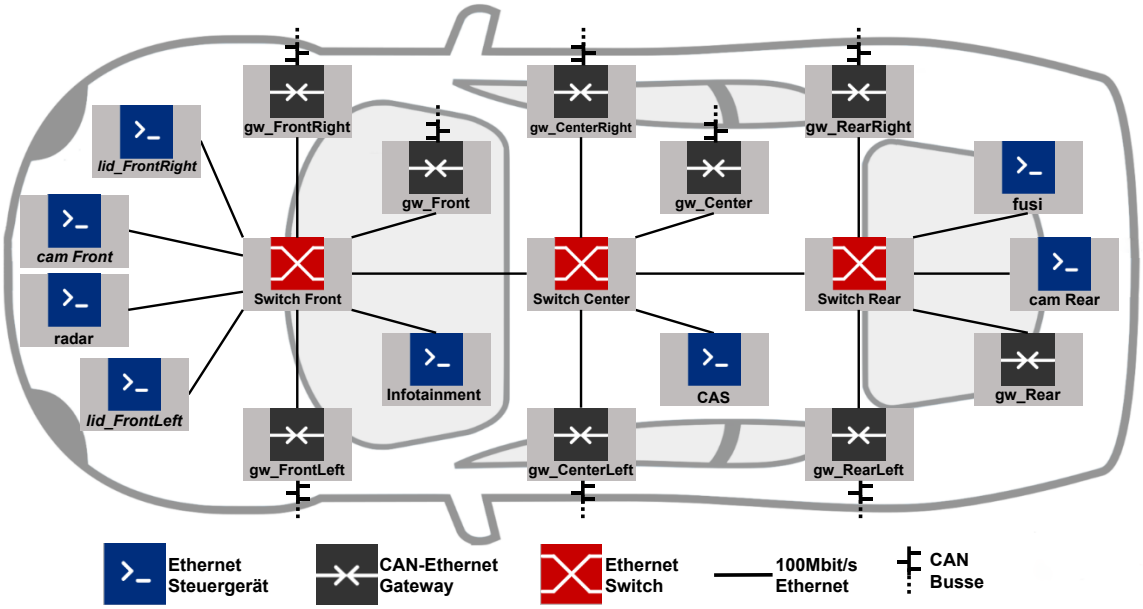

Abbildung 3.2: Die Topologie des verwendeten Fahrzeugnetzwerks

Die Fahrzeugtopologie ist in drei Gruppen unterteilt. Diese Gruppen bilden Zonen (z. B. vorne links), die durch ein Ethernet-Backbone miteinander verbunden sind. Das Ethernet-Backbone, bestehend aus 100Mbit/s Links und verbindet die Steuergeräte mit drei Switches, die sich im vorderen, mittleren und hinteren Teil des Fahrzeugs befinden. Zudem werden die Steuergeräte nach ihrer physischen Position im Fahrzeug eingeteilt. Steuergeräte wie beispielsweise Radar und Kameras sind direkt an das Backbone angebunden. Insbesondere sendet das Radar über ein Zeitschlitzverfahren Nachrichten zur

Kollisionsvermeidung (CAS). Die vorderen und hinteren Kameras sowie Lasersensoren (Lidar) übertragen ihre Daten mittels reservierter Netzwerkströme an die Sensorfusion(Fusi) im hinteren Teil des Fahrzeugs.

An jedem dieser drei Switches sind jeweils drei Gateways angeschlossen, die die Steuergeräte der linken, mittleren und rechten Bereiche im Fahrzeug verbinden. Dadurch entstehen insgesamt neun Zonen im Fahrzeugnetzwerk. Hinter jedem Gateway befinden sich CAN-Busse, an die die CAN-Steuergeräte angeschlossen sind. Diese Steuergeräte sind bekannt als Electronic Control Units (ECU). Jede ECU tauscht zahlreiche unterschiedliche Botschaften oder Signale mit jeder anderen ECU aus. Insgesamt werden 208 CAN-Nachrichten mit verschiedenen Prioritäten zwischen den Gateways über das Ethernet-Backbone ausgetauscht. Die CAN-Steuergeräte sowie ECUs sind in der dargestellten Abbildung 3.2 nicht eingeführt.

4 Verwante Arbeiten

4.1 Network Anomaly Detection in Cars: A Case for Time-Sensitive Stream Filtering and Policing

In der Arbeit von P. Meyer[7] wurde die Technologie namens Per-Stream Filtering and Policing (PSFP) zur Erkennung von Anomalien in Fahrzeugnetzwerken vorgestellt, die auf Time-Sensitive Networking (TSN) basiert. Diese Technologie wurde als Kerntechnologie eingesetzt, um Verkehrsströme im Fahrzeugnetzwerk zu überwachen und anomales Verhalten zu identifizieren. Es wurde festgestellt, dass Angriffe auf das Netzwerk wirksam erkannt werden können, ohne dass normale Aktivitäten fälschlicherweise als bösartig eingestuft werden. In der Arbeit werden der gesamte eingehende Verkehr mithilfe von PSFP-Muster gefiltert. Dieser Muster umfasst die zeitlichen und bandbreitenbezogenen Anforderungen sowie die Paketgröße. P. Meyer untersuchte das Anomalieerkennungssystem durch verschiedene Arten von Netzwerkangriffen. Angriffe werden in sieben Kategorien klassifiziert: Eliminierung, Injection, Inspektion, Manipulation, Redirection, Reordering und Rescheduling. Jede Klasse wird hinsichtlich ihrer Auswirkungen auf die Linkschicht untersucht. Es wurde festgestellt, dass PSFP die meisten dieser Angriffe gut erkennen konnte. Der Einsatz von Filtern und Regeln ermöglicht eine präzise und zuverlässige Erkennung von Netzanomalien. Das bedeutet, dass verdächtige Aktivitäten erkannt und blockiert werden können, bevor sie Schaden anrichten. Dadurch wird die Sicherheit von Fahrzeugen gegen potenzielle Angriffe erhöht.

4.2 Integration realer Angriffe in simulierte Echtzeit-Ethernet-Netzwerke

S. Reider[12] befasst sich in ihrer Arbeit mit einem wichtigen Konzept der Cybersicherheit in der Automobilindustrie, indem sie untersucht, wie Angriffe aus der realen Welt in

simulierte Echtzeit-Ethernet-Netzwerke integriert und evaluiert werden können. Um ihren Ansatz zu demonstrieren, wurde ein DoS-Angriff innerhalb einer Simulation nachgespielt. Das verwendete Fahrzeugnetzwerk kann Echtzeit-Ethernet-Kommunikation und Fahrzeugbusse simulieren, wodurch ein realistisches Modell der Fahrzeugarchitektur ermöglicht wird. Zuerst wurde der ausgehende Datenverkehr am Interface eines Kamerasteuergerätes aufgezeichnet und daraus wurden pcapng-Dateien erzeugt. Ein Modul als Paketgenerator wurde entwickelt, das den aufgezeichneten Ethernet-Datenverkehr aus pcapng-Dateien liest und in die Simulation einspielt, um eine realitätsnahe Nachbildung von Angriffen in der Simulationsumgebung zu ermöglichen. Die Simulation wurde in zwei verschiedenen Formen durchgeführt, einmal im normalen Betrieb und einmal mit einem DoS-Angriff. Der DoS-Angriff wurde mithilfe eines Paket-Injektors generiert und über ein Kamerasteuergerät in das Netzwerk eingespielt. Der Vergleich der Netzwerkstatistiken vor und nach dem Angriff zeigten einen spürbaren Anstieg des Netzwerkverkehrs und der Latenzzeit, insbesondere bei Datenpaketen mit niedriger oder gleicher Priorität. Trotz des Angriffs wurden keine Paketverluste aufgrund unzureichend genutzter Bandbreite beobachtet, was darauf hindeutet, dass bestimmte Systeme weiterhin ohne Datenverlust arbeiten konnten. Die Integration von realen Angriffsmustern in Simulationen ermöglicht es, die Auswirkungen von Angriffen zu untersuchen und die Schwachstellen eines Netzwerks zu identifizieren.

4.3 Automotive Ethernet Real-Time Observer for Anomaly Detection in In-Vehicle Networks

S. Jeong[6] stellt in seiner Arbeit ein System namens Automotive Ethernet Real-Time Observer (AERO) zur Erkennung von Anomalien in fahrzeuginternen Netzwerken vor. Dieses System soll die interne Kommunikation von Fahrzeugnetzwerken vor Angriffen oder Anomalien schützen. Im Fahrzeugnetzwerk überwacht ein Intrusion Detection System (IDS) den Datenverkehr, der zwischen verschiedenen Komponenten des Fahrzeugs gesendet wird, und überprüft ihn auf ungewöhnliche oder verdächtige Aktivitäten. AERO ist ein unüberwachtes IDS, was bedeutet, dass es keine vordefinierten Beispiele von Angriffen benötigt, um zu lernen. Es kann neue und unbekannte Arten von Angriffen erkennen, indem es Abweichungen vom normalen Verhalten identifiziert. AERO umfasst drei Komponenten: einen Merkmale-Extraktor, ein neuronales Netzwerk zur Verarbeitung der extrahierten Merkmale und einen Online-Anomalie-Erkennung. Der Merkmale-Extraktor sammelt

verschiedene Arten von Informationen aus den Datenpaketen, die durch das Netzwerk des Fahrzeugs laufen. Er untersucht Aspekte wie die Reihenfolge der Nachrichten, den Inhalt der Nachrichten und das Timing dieser Nachrichten. Das neuronale Netzwerk verarbeitet und analysiert die extrahierten Merkmale, um zu lernen, wie normale Netzwerkmuster aussehen. Die Anomalie-Erkennung läuft in Echtzeit und vergleicht aktuelle Daten mit den erlernten Mustern, um ungewöhnliche Muster zu erkennen, die als potenzielle Anomalien oder Angriffe gekennzeichnet werden. AERO arbeitet somit in einem einfachen dreistufigen Prozess. S. Jeong hat die Leistung von AERO anhand des TOW-IDS Automobil-Ethernet-Datensatzes getestet und evaluiert. Die Ergebnisse zeigen, dass AERO fünf verschiedene Angriffsarten effektiv und mit hoher Genauigkeit erkennt. Der Merkmale-Extraktor konnte 9638 Merkmale pro Sekunde verarbeiten. Diese entspricht den Anforderungen an die Echtzeit-Erkennung in Fahrzeugnetzwerken. AERO errichte hohe Erkennungsrate für verschiedene Angriffstypen wie CAN Denial of Service (DoS), CAN Replay, Content Addressable Memory (CAM) Table Overflow, AVTP Frame Injection und PTP Sync Attack. Im Vergleich zu anderen Methoden zeigt AERO eine bessere oder vergleichbare Leistung bei geringerer Komplexität und höherer Effizienz. AERO kann Anomalien in Echtzeit erkennen und bietet somit eine zuverlässige Lösung für die Sicherheit von Fahrzeugnetzwerken.

5 Simulation von Angriffen in Fahrzeugnetzen

5.1 Denial-of-Service Angriff

In den folgenden Unterkapiteln werden zunächst der Angriff und sein Aufbau im Kontext eines Szenarios beschrieben. Anschließend werden die Konfigurationen erläutert, die zur Realisierung des Angriffs notwendig sind, sowie die Umsetzung des Angriffs im Simulationsnetzwerk dargelegt. Zum Abschluss wird der simulierte Angriff validiert und werden seine exemplarischen Auswirkungen auf das Netzwerk erörtert.

Konzept und Aufbau des Angriffs

Ethernet wird in Fahrzeugen nicht nur zur Unterstützung von Infotainment-Anwendungen verwendet, sondern auch zur Übertragung von Echtzeitdaten, die für die Steuerung des Fahrzeugs benötigt werden. Dies ermöglicht eine minimale Verzögerung bei der Übertragung. Ein bekannter Cyberangriff, der diese Echtzeitdatenübertragung beeinträchtigen kann, ist der sogenannte "DoS-Angriff". Hierbei wird ein Dienst absichtlich mit vielen Anfragen bombardiert, bis er nicht mehr in der Lage ist, Anfragen zu bearbeiten oder im schlimmsten Fall zusammenbricht. Ein DoS-Angriff ermöglicht es einem Angreifer, kritische Komponenten des Fahrzeugs zu beschädigen und die Zuverlässigkeit des Systems zu beeinträchtigen. Insbesondere kann dies auf einem Echtzeit-Ethernet dazu führen, dass die Übertragung von Echtzeitdaten gestört wird.

In diesem Abschnitt werden die Auswirkungen eines DoS-Angriffs in einem Ethernet-basierten Fahrzeugnetzwerk untersucht. Dies wird durch das simulierte Szenario eines DoS-Angriffs erreicht. Eine Sensorfusion (Fusion) hat im Fahrzeug die Aufgabe, alle Sensordaten im Fahrzeugnetzwerk zu sammeln. Abbildung 5.1 zeigt, dass der Angreifer per Software in einer kompromittierten Sensorkomponente (z.B. Frontkamera) platziert

ist. Dadurch kann er die Sensorfusion mit einer großen Anzahl von Nachrichten von dieser Sensorkomponente bombardieren.

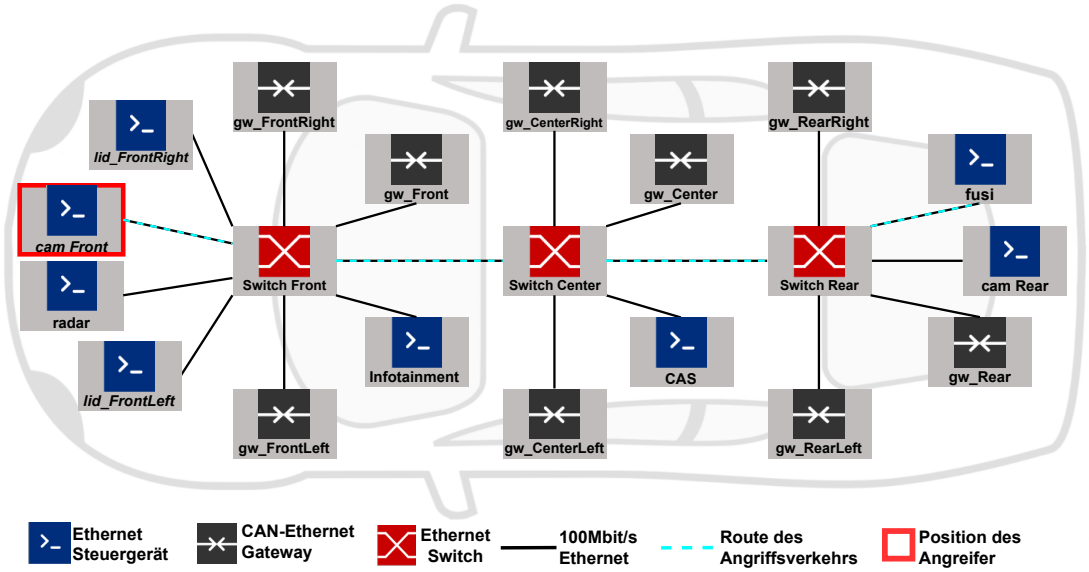

Abbildung 5.1: DoS Angriff in RecBarToZone Netzwerk

Vorbreitung des Angriffs

Um die Grundlage für die Simulation des Angriffs zu schaffen, sollen einige Vorbereitungen für die genannten Komponenten getroffen werden, um das beschriebene Konzept umzusetzen. Zur Erfassung von Statistiken über die Nachrichten, die von der Frontkamera an Fusi gesendet wurden, wird in der INI-Datei der Fusi-Komponente eine Sinkapp eingerichtet. Diese ermöglicht es, das Nachrichtenaufkommen später zu analysieren. Die folgende Konfiguration wurde in der INI-Datei hinzugefügt:

Listing 5.1: Die Erstellung eine Sink-App für empfangenen Nachrichten von Camera Front

```
1 **.fusi.numApps = 12
2 # Sink app for message "camFrontToFusi":
3 **.fusi.app[11].typename = "IEEE8021QTrafficSinkApp"
4 **.fusi.app[11].srcAddress = "F6-0E-A2-E0-49-14"
5 **.fusi.app[11].replyFirst = false
```

In der ersten Zeile von Listing 5.1 wird eine neue Anwendung in der Fusi-Komponente instanziiert. Diese Zahl zeigt die Anzahl der Anwendungen, die auf der Fusi-Komponente zu instanziiert sind. In der dritten Zeile wird der Typ der Anwendung als $IEEE8021QTraffic-SinkApp$ festgelegt, um den Verkehr mit Q-Prioritäten zu erfassen. Die vierte Zeile legt fest, von welcher Quelladresse die empfangenen Nachrichten stammen sollen. Hier soll die Adresse der Frontkamera-Komponente eingetragen werden. Die fünfte Zeile dient dazu, den ersten Frame zur Aktualisierung der Tabellen in den Switches zu wiederholen.

Umsetzung des Angriffs in der Simulation

Für die Durchführung des DoS-Angriffs wurden in der Cam_Front-Komponente einige Änderungen vorgenommen, die durch den Konfigurationscode erläutert werden.
Im Netzwerk (siehe Abbildung 5.1) befinden sich eine Komponente namens Sensorfusion und eine Komponente namens Frontkamera. Der Angreifer infiltriert das Fahrzeugnetzwerk und infiziert die Kamerakomponente mithilfe der installierten Software. Alle 1 ms sendet der Angreifer eine IEEE802.1Q-Nachricht mit einer Größe von 1500 Bytes an die Unicast-MAC-Adresse der Sensorfusion, und das mit einer Priorität von 6. Dadurch überlastet er die Sensorfusion mit einer Flut von Nachrichten. Um sowohl die Position des Angreifers als auch den Angriff selbst zu simulieren, werden die folgenden Konfigurationen in der INI-Datei von Cam_Front umgesetzt:

Listing 5.2: Die Konfiguration für die Erstellung ein Nachricht von Camera zu Fusi

```
1  **.cam_Front.numApps = 3
2  #Type der Nachrichtprotokoll und das Zeil des Angriff
3  **.cam_Front.app[2].typename = "IEEE8021QTrafficSourceApp"
4  **.cam_Front.app[2].destAddress = "36-67-5A-E4-EE-91"
5  #Eigenschaften der Nachricht
6  **.cam_Front.app[2].payload = intWithUnit(uniform(1500Byte, 1500Byte)
     ↪ )
7  **.cam_Front.app[2].priority = 6
8  **.cam_Front.app[2].vid = 0
9  #Angriff Eigenschaften
10 **.cam_Front.app[2].sendInterval = uniform(0.0010s, 0.0010s)
11 **.cam_Front.app[2].startTime = 1.2 s
```

In der ersten Zeile von Listing 5.2 wird eine neue Anwendung in der Kamerakomponente instanziiert. Diese Zahl zeigt die Anzahl der Anwendungen, die auf der Kamerakomponente

zu instanziiert sind. In der dritten Zeile wird das Protokoll IEEE802.1Q ausgewählt, welches den Typ der Nachricht für den Angriff bestimmt. In der vierten Zeile wird die Unicast-MAC-Adresse der Fusi-Komponente eingetragen. In den Zeilen 6-8 werden die Eigenschaften des Pakets, wie Nutzlastgröße, Priorität und die zugehörige VLAN-ID angegeben. In der zehnten Zeile wird das Intervall angegeben, in dem ein Paket gesendet wird. In der elften Zeile wird der Startzeitpunkt des Angriffs angegeben, was bedeutet, wann der Angriff beginnen soll.

Qualitätsicherung des Angriffs

In diesem Abschnitt wird der DoS-Angriff validiert. Das beschriebene Konzept wurde 10 Sekunden lang simuliert und die Ergebnisse sind in der Grafik dargestellt. Um festzustellen, ob die Zielkomponente mit vielen Anfragen bombardiert wurde, sollte die Anzahl der ankommenden Pakete an der Fusion-Komponente betrachtet werden. Für die Erstellung der Abbildung 5.2 wurden nur Pakete mit der Priorität 6 ausgewählt, da die DoS-Pakete mit dieser Priorität gesendet wurden. In der Abbildung stammen die Daten für empfangene Pakete im normalen Fall sowie im Fall eines DoS-Angriffs aus der Parametrisierung der Sinkapp.

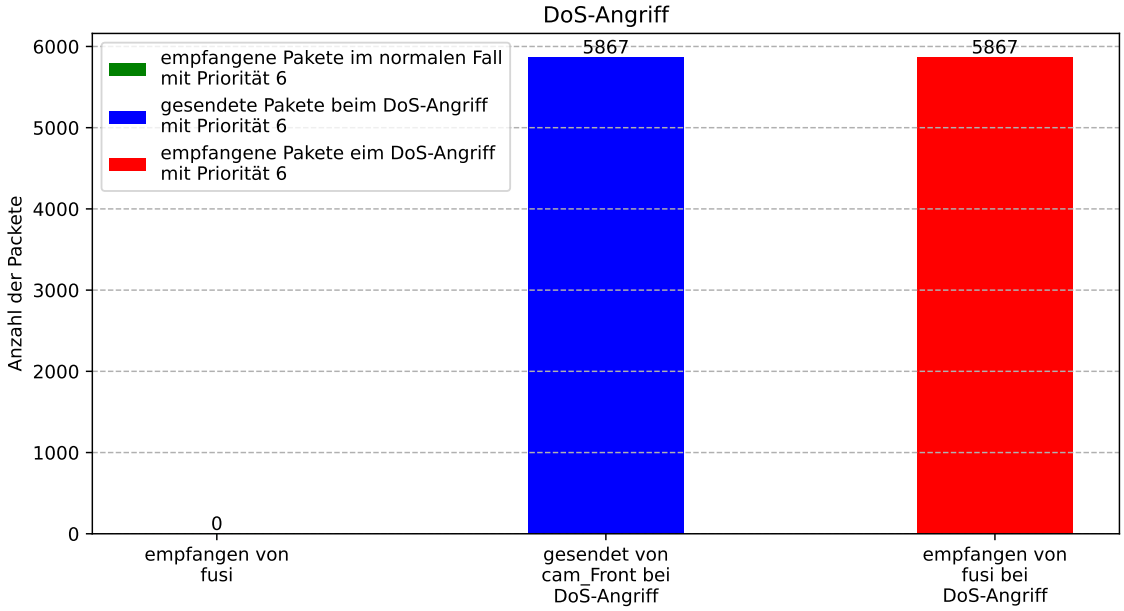

Abbildung 5.2: Ergebnisse für die Ausführung DoS-Angriff im RecBarToZone-Netzwerk

Die Abbildung 5.2 zeigt deutlich ersichtlich, dass während des Normalbetriebs die Anzahl der empfangenen Pakete durch die Fusi-Komponente null beträgt. Dies bedeutet, dass keine Pakete mit Priorität 6 von der Frontkamera gesendet wurden. Im Gegensatz dazu zeigt die blaue Spalte während des Angriffs, dass die Frontkamera insgesamt 5867 Pakete mit der Priorität 6 an die Fusion gesendet hat. Die rote Spalte zeigt die identische Anzahl von Paketen, die während des Angriffs von der Fusion empfangen wurden. Hierdurch lässt sich sicherstellen, dass der DoS-Angriff die Fusion-Komponente mit zahlreichen Anfragen bombardiert hat.

Evaluation der Angriffsergebnisse

In diesem Abschnitt werden die Auswirkungen des DoS-Angriffs sowohl auf Anwendungs- als auch auf Netzwerkebene beschrieben. Im Konfigurationscode (sieh Listing 5.2, Zeile 3) ist erkennbar, dass der verwendete Protokolltyp IEEE 802.1Q ist. Der Angreifer nutzt den Priorisierungsmechanismus dieses Protokolls aus, um das Fahrzeugnetzwerk oder die Anwendungen in den Fahrzeugkomponenten daran zu hindern, ordnungsgemäße Dienste bereitzustellen. Dieser Angriff kann die Rechenleistung der Fahrzeugkomponente beeinträchtigen und die Netzwerkbandbreite aufbrauchen.
Im Allgemeinen wird die Fusi-Komponente durch den DoS-Angriff mit einer Vielzahl von Anfragen bombardiert. Dies wirkt sich auf die Anwendungsebene derart aus, dass die Fusi-Komponente aufgrund dieser Überlastung nicht in der Lage ist, alle Anfragen korrekt zu verarbeiten, was zu einer Beeinträchtigung oder zum Absturz der Software führen kann.
Die Auswirkungen auf die Netzwerkebene werden als Charakterisierung der Netzwerkleistung betrachtet. Einerseits führt die Überlastung zu einer Reduzierung der Bandbreite von der Front-Kamera zur Fusi-Komponente. Andererseits entsteht die Netzüberlastung aus zahlreichen ankommenden Paketen mit hoher Priorität. Dies bedeutet, dass aufgrund der hohen Priorität dieser Pakete alle anderen Pakete mit niedrigerer Priorität blockiert werden, bis die hoch priorisierten Pakete übertragen sind. Dies führt wiederum zu Verzögerungen bei diesen Ethernet-Paketen.

Um die Verzögerung im Fahrzeugnetzwerk besser zu verstehen, wurde beispielsweise der Datenverkehr von Port 4 des Switch-Centers ausgewählt. Die Abbildung 5.3 zeigt die Latenz der Queue 5 mit Priorität 5 des Switch-Centers. Jeder Punkt in der Abbildung 5.3 repräsentiert einen Ethernet-Frame, der an der Queue des Switch-Centers angekommen wurde.

Die hellgrüne gestrichelte Linie in der Abbildung 5.3 dient dazu, die Ausführungszeit des DoS-Angriffs zu veranschaulichen, nachdem die Simulation gestartet wurde. Das obere Diagramm zeigt die Latenzzeit im normalen Netzwerkbetrieb. Es ist ersichtlich, dass am Anfang einige Pakete eine höhere Latenz aufweisen, die sich jedoch im Laufe der Zeit verringert haben. Hier kann angenommen werden, dass diese Pakete durch Prioritäten 6 oder 7 blockiert wurden. Es ist klar erkennbar, dass die Latenzzeit von Queue 5 meist zwischen 0,2ms und 1,0ms liegt. Die höchste Latenzzeit beträgt 3,5ms.
Das untere Diagramm der Abbildung 5.3 zeigt die Latenzzeit bei einem DoS-Angriff im Netzwerk. Es ist unbestreitbar, dass die Latenzzeit in den ersten 1,2ms gleich wie in normalem Netzwerkbetrieb ist. Im Vergleich zum oberen Diagramm ist jedoch deutlich zu erkennen, dass die Latenzzeit jedes Ethernet-Frames durch die Belastung des DoS-Angriffs stark ansteigt. Es ist ebenfalls offensichtlich, dass die maximale Latenzzeit überschritten wurde. In diesem Fall beträgt die höchste Latenzzeit 4,5ms.

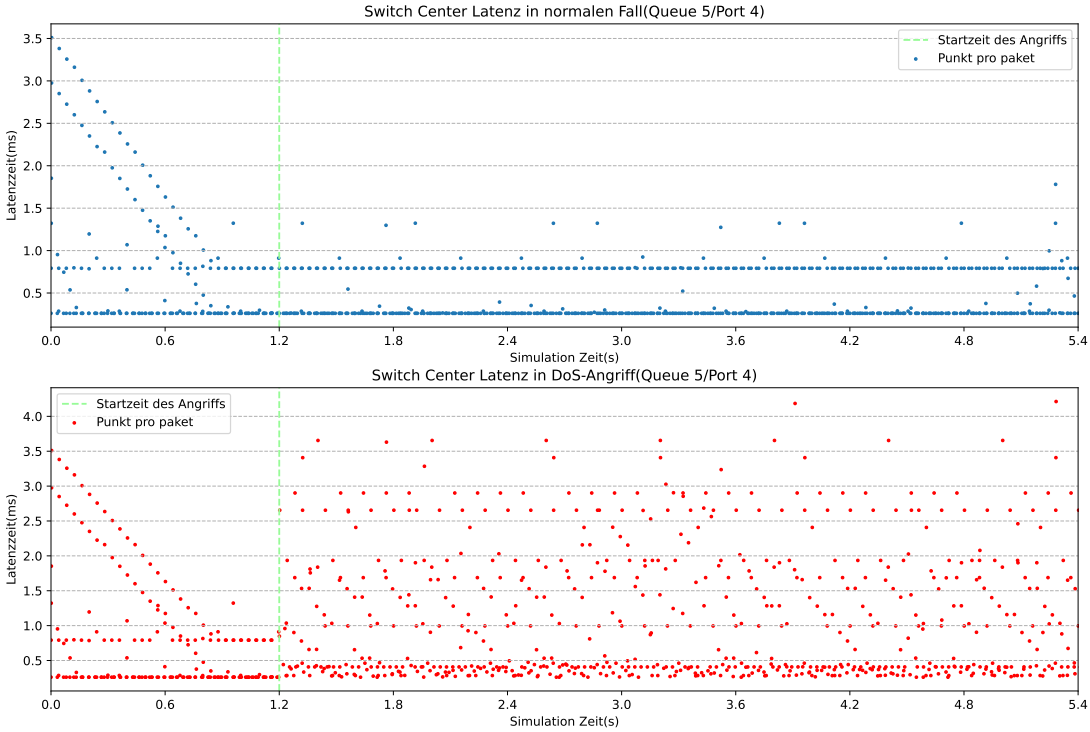

Abbildung 5.3: Die Latenzzeit der Ethernetframes in Switch-Center

Durch den Angriff werden alle Switches im Netzwerk beeinträchtigt. Daher werden die Auswirkungen auf jeden einzelnen Switch mithilfe von Statistiken genauer untersucht. Die folgenden Tabellen zeigen die durchschnittliche und maximale Latenzzeit der Ethernet-

Pakete, die von der Kamera front über die Switches an die Fusion weitergeleitet wurden - sowohl unter normalen Bedingungen als auch bei einem DoS-Angriff.

	Switch Front (Port 3)			
	Normal		DoS	
Prior.	Durchschn.	Max	Durchschn.	Max
0	0,0462ms	0,10784ms	0,0462ms	0,10784ms
1	0,05364ms	0,10112ms	0,05364ms	0,10112ms
2	-	-	-	-
3	0,01977ms	0,03392ms	0,01977ms	0,03392ms
4	-	-	-	-
5	0,17246ms	2,84536ms	0,20120ms	2,84537ms
6	0,03146ms	0,09440ms	0,10474ms	0,25176ms
7	0,01384ms	0,01384ms	0,01384ms	0,01384ms

Tabelle 5.1: Die durchschnittliche und maximale Latenzzeit von Port 3 am Switch-Front

Zunächst werden die Statistiken der Switch-Front betrachtet. In der Tabelle 5.1 sind die Latenzzeiten jeder Priorität des Ports 3 dargestellt. Im Falle eines DoS-Angriffs ist keine Veränderung bei der Latenzzeit der 7. Priorität erkennbar. Der Grund dafür ist, dass die DoS-Frames eine niedrigere Priorität haben und durch die Priorität 7 blockiert wurden. Es besteht jedoch möglich, dass ein DoS-Frame ganz kurz von dem Zeitpunkt eintreffen, zu dem ein 7ter Prioritätsframe verschickt wird. Warum hat dieser DoS-Frame keinen Einfluss auf 7ten Prioritätsframe? Der Grund ist, dass der hoch priorisierte Frame mit einem größeren Zeitabstand verschickt wird, wodurch genügend Bandbreite für die Übertragung des DoS-Frames zur Verfügung steht. Wenn jedoch der hoch priorisierte Frame sehr eng verschickt wird, könnte der DoS-Frame den Frame beeinflussen. Dieser Grund gilt auch für Prioritäten 0 bis 3.
Im Gegensatz zu anderen Prioritäten ist die Latenzzeit von Priorität 5 durch den Angriff beeinträchtigt und ihre maximale Latenzzeit hat um 0,1us erhöht(sieh Tabelle 5.1). Das bedeutet, dass die Frames mit Priorität 5 durch den DoS-Angriff blockiert wurden. Der Effekt des DoS-Angriffs besteht nur in der Erhöhung der durchschnittlichen Latenzzeit der Priorität 5 des Ports 3 an der Switch-Front.

Anschließend betrachten wir die Statistiken des Switch-Centers und des Switch-Rears. In Tabelle 5.2 sind die Latenzzeiten jeder Priorität des Ports 4 am Switch-Center und des Ports 5 am Switch-Rear dargestellt.
Die Latenzzeiten des Switch-Centers unterscheiden sich von denen des Switch-Fronts. Die

Prior.	Switch Center (Port 4)				Switch Rear (Port 5)			
	Normal		DoS		Normal		DoS	
	Durchschn.	Max	Durchschn.	Max	Durchschn.	Max	Durchschn.	Max
0	0,083ms	0,299ms	2,238ms	11,436ms	-	-	-	-
1	0,072ms	0,231ms	0,810ms	5,347ms	-	-	-	-
2	-	-	-	-	-	-	-	-
3	0,045ms	0,260ms	0,118ms	2,879ms	-	-	-	-
4	-	-	-	-	-	-	-	-
5	0,492ms	3,509ms	1,081ms	4,212ms	0,491ms	3.640ms	0,920ms	4,342ms
6	0,062ms	0,209ms	0,270ms	0,560ms	0	0	0,422ms	0,788ms
7	-	-	-	-	-	-	-	-

Tabelle 5.2: Die durchschnittliche und maximale Latenzzeit von Port 4 am Switch-Center/ von Port 5 am Switch-Rear

maximale Latenzzeit jeder Priorität wurde durch den Angriff überschritten(sieh Switch-Center in Tabelle 5.2), insbesondere bei den Prioritäten 0 und 1. Dieser große Unterschied hängt von zwei Faktoren ab. Erstens spielt die Netzwerkstruktur eine Rolle. Der meisten Datenverkehr fließen durch das Switch-Center, und dann kommen massive Mengen von hoch priorisierten Frames an. Der Angriff führt dazu, dass die Latenzzeit jeder Priorität überschritten wird, was zu einem Zusammenbruch des Konzepts der Echtzeitübertragung führt.

Ein weiterer Punkt, auf den eingegangen werden soll, ist das Sende-Intervall der Frames im DoS-Angriff. Das Sende-Intervall steht in direktem Zusammenhang mit der Latenzzeit. Je kleiner das Sende-Intervall ist, desto mehr Frames werden verschickt. Dadurch steigt die Latenzzeit. Zum Beispiel beträgt in Zeile 10 von Listing 5.2 das Sende-Intervall 1 ms. Wenn das Sende-Intervall auf **1 us** geändert wird, erhöht sich sowohl die maximale als auch die durchschnittliche Latenzzeit.

Es lässt sich annehmen, dass dieser Angriff nicht nur die Funktionalität der Fusi-Komponente beeinträchtigt, sondern auch Störungen im Netzwerk verursachen kann.

5.2 Man-in-the-Middle Angriff

Der Kern von Man-in-the-Middle Angriff (MitM-Angriff) basiert auf der Arbeit von P. Meyer [7]. Heutigen Echtzeit-Ethernet-Fahrzeuge nutzen fortschrittliche Technologien, um Fahrzeuge sicher und effizient zu steuern. Eine potenzielle Bedrohung für dieses Netzwerk ist der Man-in-the-Middle-Angriff. Ein MitM-Angriff ermöglicht es dem Angreifer, den Datenverkehr zwischen zwei Systemen im Netzwerk abzufangen und zu verändern und kritische Systeme in Fahrzeug zu beeinträchtigen. Dem Angreifer stehen verschiedene Arten von Techniken zur Verfügung.
Dieses Kapitel befasst sich mit diesen Techniken sowie der Funktionsweise des MitM-Angriffs und seinen Auswirkungen auf ein Ethernet-basiertes Fahrzeugnetzwerk. Die angewendeten Techniken für diesen Angriff sind:

- Manipulation von Frames
- Redirection von Frames
- Elimination von Frames
- Rescheduling von Frames

5.2.1 Manipulation

In den folgenden Unterkapiteln werden zunächst der Angriff und sein Aufbau im Kontext eines Szenarios beschrieben. Anschließend werden die Konfigurationen erläutert, die zur Realisierung des Angriffs notwendig sind, sowie die Umsetzung des Angriffs im Simulationsnetzwerk dargelegt. Zum Abschluss wird der simulierte Angriff validiert und wedern seine exemplarischen Auswirkungen auf das Netzwerk erörtert.

Konzept und Aufbau des Angriffs

Ein häufiges Ziel des Angreifers ist es, gefälschte Informationen zu generieren oder gezielt Fahrzeugdaten zu manipulieren. Eine Technik, um dieses Ziel zu erreichen, ist die Manipulation. Durch die Manipulationstechnik können die Daten verändert bzw. verfälscht werden, was zu gefährlichen Situationen für den Fahrer und die Umgebung führen kann. Zum Beispiel, wenn ein Motorstörungsalarm, ein Bremsfunktionsfehler oder Sensordaten im Infotainmentsystem nicht korrekt angezeigt werden. Oder wenn im Kombiinstrument

die empfangenen Daten nicht richtig dargestellt werden. In diesem Abschnitt werden die exemplarischen Auswirkungen der Manipulation in einem Ethernet-basierten Fahrzeugnetzwerk untersucht. Dies wird anhand von zwei simulierten Szenarien eines MitM-Angriffs verdeutlicht. Die Szenarien erläutern zwei bestimmte Arten von Manipulationen, da mit dieser Technik verschiedene gezielte Veränderungen durchgeführt werden können. Diese Probleme werden in zwei Szenarien dargestellt.
Im ersten Szenario gehen wir davon aus, dass ein Steuergerät (ECU3), das die Drehgeschwindigkeit der Reifen misst, die Grunddaten für die Anzeige der Geschwindigkeit im Kombiinstrument (ECU91) liefert. In der Abbildung 5.4 wird ein kompromittiertes Gateway dargestellt, das den CAN-Frame in Ethernet-Frames umwandelt und sendet. Der Angreifer platziert sich durch Software auf dem Gateway-Center-Left (gw_CenterLeft) und positioniert sich somit zwischen dem Steuergerät und dem Kombiinstrument. Das Ziel des Angreifers besteht darin, die hoch priorisierten Ethernetframes zu verzögern, die vom Steuergerät zum Kombiinstrument gesendet werden. In diesem Fall manipuliert der Angreifer die Daten, die von ECU3 über das Netzwerk an ECU91 übertragen werden, was zur Folge hat, dass dem Fahrer ein falscher Wert angezeigt wird.

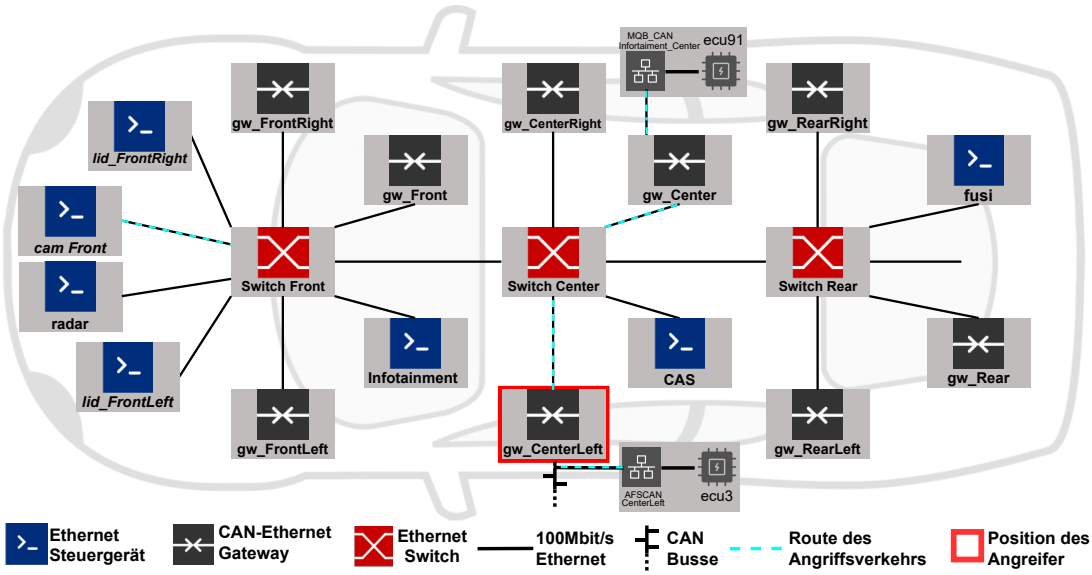

Abbildung 5.4: Manipulation in Gateway für Szenario 1 in RecBarToZone Netzwerk

Im zweiten Szenario gehen wir davon aus, dass eine Sensorfusion (Fusi), die Daten von ausgewählten Sensoren im Netzwerk sammelt und aufbereitet, diese anschließend an das Infotainment weiterleitet. Der Angreifer platziert sich durch Software auf dem Switch und positioniert sich somit zwischen der Sensorfusion und dem Infotainment,

um die von der Sensorfusion gesendeten Daten zu manipulieren. In der Abbildung 5.5 wird ein kompromittierter Switch (Switch-Center) dargestellt, in dem der Angreifer die Möglichkeit hat, sowohl die Priorität als auch die Größe und den Inhalt der Payload eines Ethernetframes zu ändern, der von der Sensorfusion an das Infotainment gesendet wird. Das Ziel des Angreifers besteht darin, die gesendeten Daten zu verändern.

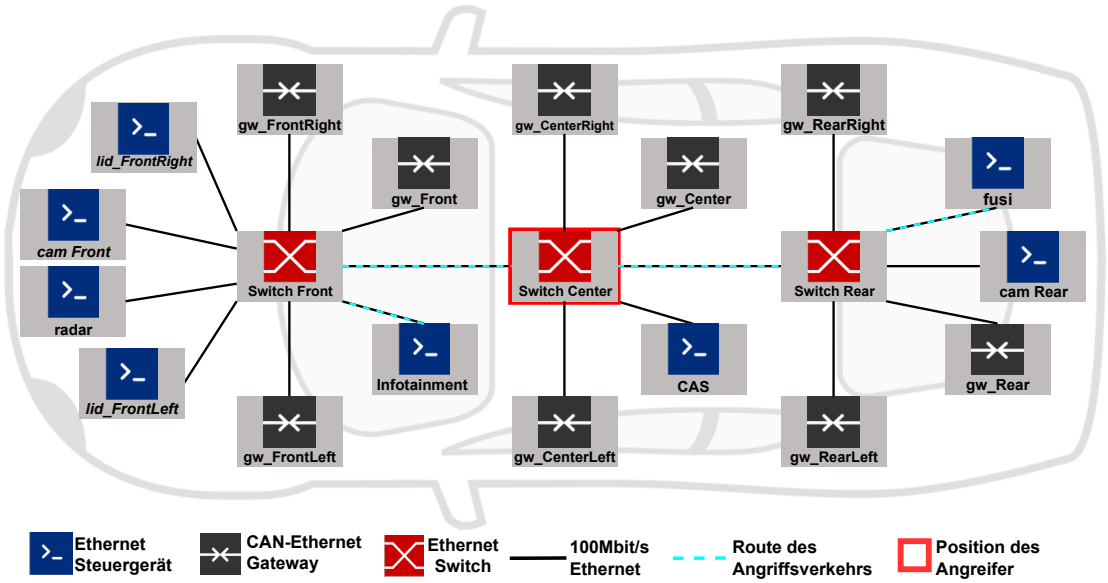

Abbildung 5.5: Manipulation im Switch für Szenario 2 in RecBarToZone Netzwerk

Vorbreitung des Angriffs in der Simulation

Zur Realisierung der Angriffe werden zusätzliche Funktionen in den zuvor genannten Komponenten des Netzwerks eingerichtet. Für das erste Szenario muss zunächst der Verkehr, der später Ziel des Angriffs sein wird, konfiguriert werden. Das ECU3 agiert als CAN-Gerät im Netzwerk (siehe sieh Abbildung 5.4) und sendet mehrere Datenströme an CAN-Komponenten. Die ausgewählte CAN-Nachricht von ECU3 ist eine Nachricht, die periodisch alle 100 ms mit einer Größe von 8 Bytes an verschiedene CAN-Komponenten, insbesondere an ECU91, gesendet wird. Im Gateway-Center-Left wird eine bestimmte Menge dieser CAN-Botschaften in einem Ethernetframe gekapselt und erhalten eine neue Multicast-MAC-Adresse sowie eine Priorität. Aus diesem Grund, um den Datenverkehr

zwischen ECU3 und ECU91 isoliert betrachten zu können, wird diese ausgewählte CAN-Nachricht in einem neuen Ethernetframe verpackt. Diese Änderungen werden in der XML-Datei des Gateway-Center-Left konfiguriert:

Listing 5.3: Verkapselung der CAN-Nachricht zu Ethernet Nachricht

```
1 # neue Multicast Mac Adresse mit VLAN-Id und Prioritaet
2 <ethernetqframe dst="01-AA-EE-AA-00-01" vid="0" pcp="6">
3     <canframe canId="347"/>
4 </ethernetqframe>
```

Der Code aus Listing 5.3 kapselt die CAN-Nachricht von ECU3 in einem Ethernetframe. In Zeile 2 wird dem Ethernetframe eine neue Multicast-MAC-Adresse, eine VLAN-ID und Priorität 6 zugewiesen. In Zeile 3 wird die ID der CAN-Nachricht angegeben. Es wird nur eine einzige CAN-Nachricht in diesem Ethernetframe gekapselt.
Im nächsten Schritt muss die Multicast-MAC-Adresse im Switch-Center bekannt gemacht werden, zusammen mit der zugehörigen VLAN-ID und dem Port, zu dem sie gehört. Dafür wird die angegebene Multicast-MAC-Adresse zusammen mit der VLAN-ID und dem Port in die Adresstabelle der Switch-Center XML-Datei eingetragen. Dieser Eintrag ist in der XML-Datei von Switch-Center.
Um sicherzustellen, dass diese Ethernet-Nachricht mit ihrer neuen MAC-Adresse an ECU91 weitergeleitet wird, muss die Weiterleitung (Routing) der Nachricht in der XML-Datei des Gateway-Centers definiert sein. Dazu muss der Ethernetframe der Nachricht wieder in einen CAN-Frame entkapselt werden, da ECU91 eine CAN-Komponente ist. Für diese Anpassungen wird derselbe Code wie in Listing 5.3 verwendet.
Um der Nachrichtstrom des zweiten Szenarios in der Simulation zu modellieren, wird eine zusätzliche Anwendung in der Sensorfusion(Fusi) erstellt. Diese Anwendung stellt den Nachrichtenstrom zum Infotainment her, indem sie alle 9875ms periodisch eine Nachricht mit einer Größe von 1500 Bytes und einer Priorität von 3 an die MAC-Adresse des Infotainments sendet. Die INI-Datei der Fusi-Komponente wird mit dem folgenden Code konfiguriert:

Listing 5.4: Die Konfiguration für die Erstellung ein Nachricht von Fusi zu Infotainment

```
1 [Config Selection_Manipulation_ImSender]
2 # Source app for Q message "msg_fusiToInfortaiment":
3 **.fusi.app[11].typename = "IEEE8021QTrafficSourceApp"
4 **.fusi.app[11].sendInterval = uniform(0.009875s, 0.009875s)
5 **.fusi.app[11].destAddress = "1F-1F-1F-1F-1F-1F"
6 **.fusi.app[11].payload = intWithUnit(uniform(1500Byte, 1500Byte))
7 **.fusi.app[11].priority = 3
```

```
8  **.fusi.app[11].vid = 0
9  **.fusi.app[11].startTime = 0 s
```

Die Zeile 3 des Listing 5.4 spezifiziert den Nachrichtentyp gemäß IEEE802.1Q. Die 4te Zeile definiert das Übertragungsintervall, wobei alle 9875 ms eine Nachricht gesendet wird. Die Zeilen 5 bis 8 bestimmen die Eigenschaften der Nachricht, einschließlich der Unicast-MAC-Adresse der Infotainment-Komponente, der Nachrichtengröße, der zugehörigen Priorität und der VLAN-ID. In Zeile 8 wird der Startzeitpunkt festgelegt, ab dem die Anwendung beginnen soll, die Nachrichten zu versenden.

Listing 5.5: Die Erstellung einer Sinkapp für empfangenen Nachrichten von Fusi

```
1  **.infotainment.numApps = 1
2
3  # Sink app for message "msg_fusiToInfortaimentt":
4  **.infotainment.app[0].typename = "IEEE8021QTrafficSinkApp"
5  **.infotainment.app[0].srcAddress = "36-67-5A-E4-EE-91"
6  **.infotainment.app[0].replyFirst = false
```

Um Statistiken über die empfangenen Nachrichten im Infotainment zu sammeln, wird in der INI-Datei der Infotainment-Komponente eine Sinkapp mit der Konfiguration, wie im Listing 5.5 dargestellt, eingerichtet. In der ersten Zeile wird eine neue Anwendung instanziiert. Die Zeile 4 legt den Typ der Sinkapp-Anwendung als IEEE8021QTrafficSinkApp fest, da der Nachrichtentyp bereits im Listing 5.4 angegeben wurde, muss die Konfiguration der Sink-App denselben Typ aufweisen. Die Zeile 5 definiert, von welcher Quelladresse die empfangenen Nachrichten sein sollen. Die Zeile 6 dient dazu, den ersten Frame zur Aktualisierung der Tabellen in den Switches zu wiederholen.

Umsetzung des Angriffs in der Simulation

In diesem Abschnitt wird die Umsetzung der Manipulation für jedes Szenario erläutert. Im ersten Szenario erfolgt der Angriff im Ethernet Gateway, welches der Sender des CAN-Verkehrs in Ethernet ist. Im zweiten Szenario hingegen erfolgt der Angriff im Switch. Der Grund dafür ist, zu zeigen, dass der Angreifer an verschiedenen Stellen im Netzwerk platziert werden kann.

Manipulation im Gateway:
Um einen Angriff durch Manipulation durchzuführen, wurden im Gateway-Center-Left des

RecBarToZone-Netzwerks einige Änderungen vorgenommen. Diese Änderungen werden durch den Konfigurationscode erläutert.
Nach dem Einschleusen infiziert der Angreifer das Gateway-Center-Left mithilfe der installierten Software. Über dieses kompromittierte Gateway(siehe Abbildung 5.4) kann er den Datenverkehr auf dem Kommunikationsweg manipulieren. In der Simulation setzt sich das TSNPort-Modul aus verschiedenen Komponenten zusammen. Dieses Modul beinhaltet ein Queueing-Modul, acht Queues, ein Transmission-Algorithmus-Modul, Transmission-Gate und Transmission-Selection. Das Transmission-Selection-Modul ist für das Entnehmen einer Nachricht aus einer Queue zur weiteren Übertragung zuständig. Im Rahmen des Frameworks von P. Meyer[7] wurde das Transmission-Selection-Modul modifiziert, um zusätzliche Funktionen bereitzustellen.
Zur Simulation des Angriffs werden mithilfe dieses modifizierten Moduls Nachrichten gezielt manipuliert. Diese Nachrichten werden nach Manipulation weitergesendet. Um sowohl die Position des Angreifers als auch den Angriff selbst zu simulieren, werden folgende Änderungen in der INI-Datei von Gateway-Center-Left vorgenommen:

Listing 5.6: Die Konfiguration von Manipulation Im Gateway

```
1  [Config Selection_Manipulation_ImSender]
2  #ersetzen des Moduls durch kompromittierter Transmission Selection
3  **.gw_CenterLeft.tte[*].**.transmissionSelection.typename = "
       ↪ CorruptIEEE8021QbvSelectionManipulation"
4  #Filter,mit der die Nachricht kompromittiert wird
5  **.gw_CenterLeft.tte[*].**.transmissionSelection.corruptionDestAddress = "01-AA-EE
       ↪ -AA-00-01"
6  **.gw_CenterLeft.tte[*].**.transmissionSelection.corruptionPriority = 6
7  **.gw_CenterLeft.tte[*].**.transmissionSelection.corruptionVid = 0
8  **.gw_CenterLeft.tte[*].**.transmissionSelection.corruptionProbability = 0.8
9  #Modifizierung der Prioritaet des Pakets
10 **.gw_CenterLeft.tte[*].**.transmissionSelection.manipulationPriority = 2
```

In Zeile 3 des Listing 5.6 wird die modifizierte Transmission-Selection festgelegt. Diese ersetzt die standardmäßige Transmission-Selection von Gateway-Center-Left und stellt den Angreifer dar. In den Zeilen 5-7 von Listing 3.10 werden die Eigenschaften der Nachricht eingetragen, wie die MAC-Adresse, die Priorität und zugehörige VLAN-ID der zu kompromittierenden Nachricht. In Zeile 8 wird die Wahrscheinlichkeit eingegeben, mit der eine Nachricht manipuliert werden kann. Wenn dieser Wert auf 1 gesetzt wird, bedeutet dies, dass jede Nachricht manipuliert wird. In Zeile 10 von Listing 3.10 wird die Priorität der Nachricht auf 2 gesetzt, was den Angriff auf die Nachricht darstellt.

Manipulation im Switch

Im Netzwerk (siehe Abbildung 5.5) befindet sich eine Komponente namens Switch-Center. Nach dem Einschleusen infiziert der Angreifer dieses Switch mithilfe der installierten Software. Über das kompromittierte Switch kann er den Datenverkehr auf dem Kommunikationsweg manipulieren. Zur Simulation des Angriffs wird die modifizierte Transmission-Selection von dem gleichen Framework verwendet. Um sowohl die Position des Angreifers als auch den Angriff selbst zu simulieren, werden folgende Änderungen in der INI-Datei von Switch-Center umgesetzt:

Listing 5.7: Die Konfiguration von Manipulation Im Switch

```
1  [Config Selection_Manipulation_ImSwitch_From_Fusi]
2  #ersetzen des Moduls durch kompromittierte Transmission-Selection
3  **.switch_Center.phy[*].**.transmissionSelection.typename = "
       ↪ CorruptIEEE8021QbvSelectionManipulation"
4  #Filter,mit der die Nachricht kompromittiert wird
5  **.switch_Center.phy[*].**.transmissionSelection.corruptionDestAddress = "1F-1F-1F
       ↪ -1F-1F-1F"
6  **.switch_Center.phy[*].**.transmissionSelection.corruptionPriority = 3
7  **.switch_Center.phy[*].**.transmissionSelection.corruptionVid = 0
8  **.switch_Center.phy[*].**.transmissionSelection.corruptionProbability = 0.8
9  #Modifizierung der Nachrichtengroesse
10 **.switch_Center.phy[*].**.transmissionSelection.manipulationPayload = intWithUnit
       ↪ (uniform(1000Byte, 1000Byte))
```

In Zeile 3 des Listing 5.7 wird die modifizierte Transmission-Selection festgelegt. Dieser ersetzt die standardmäßige Transmission-Selection von Switch-Center und stellt den Angreifer dar. In den Zeilen 5-7 von Listing 3.10 werden die Eigenschaften der Nachricht eingetragen, wie die MAC-Adresse, die Priorität und zugehörige VLAN-ID der zu kompromittierenden Nachricht. In Zeile 8 wird die Wahrscheinlichkeit eingegeben, mit der eine Nachricht manipuliert werden kann. Wenn dieser Wert auf 1 gesetzt wird, bedeutet dies, dass jede Nachricht manipuliert wird. In Zeile 10 wird die ursprüngliche Nachrichtengröße von 1500 Bytes zu 1000 Bytes reduziert, welche der Inhalt der Nachricht repräsentiert, was den Angriff auf die Nachricht darstellt.

Qualitätsicherung des Angriffs

In diesem Abschnitt wird der Manipulationsangriff für jedes Szenario validiert. Die beschriebenen Szenarien wurden jeweils 120 Sekunden lang simuliert, und die Ergebnisse sind in verschiedenen Grafiken dargestellt.

Für das **erste Szenario** wird zur Feststellung möglicher Änderungen an den Nachrichten im Gateway-Center-Left das Ziel des Angreifers betrachtet. In diesem Szenario beabsichtigt der Angreifer, die Priorität der gesendeten Frames zu kompromittieren bzw. zu reduzieren. Die Abbildung 5.6 enthält die Anzahl der gesendeten und empfangenen Nachrichten. Zudem wird die Anzahl der Nachrichten betrachtet, die in den Queues mit der zweiten und sechsten Priorität an den Ports des Switch-Centers angekommen sind. Diese Daten stammen aus den Statistiken der Simulation.
Die linke Grafik in Abbildung 1 zeigt, dass während der Simulation von ECU3 insgesamt 1200 Pakete gesendet wurden, welche auch von ECU91 empfangen wurden. Die Anzahl der Pakete, die im kompromittierten Gateway-Center-Left manipuliert wurden, lässt sich mithilfe der im Switch-Center empfangenen Pakete bestimmen.

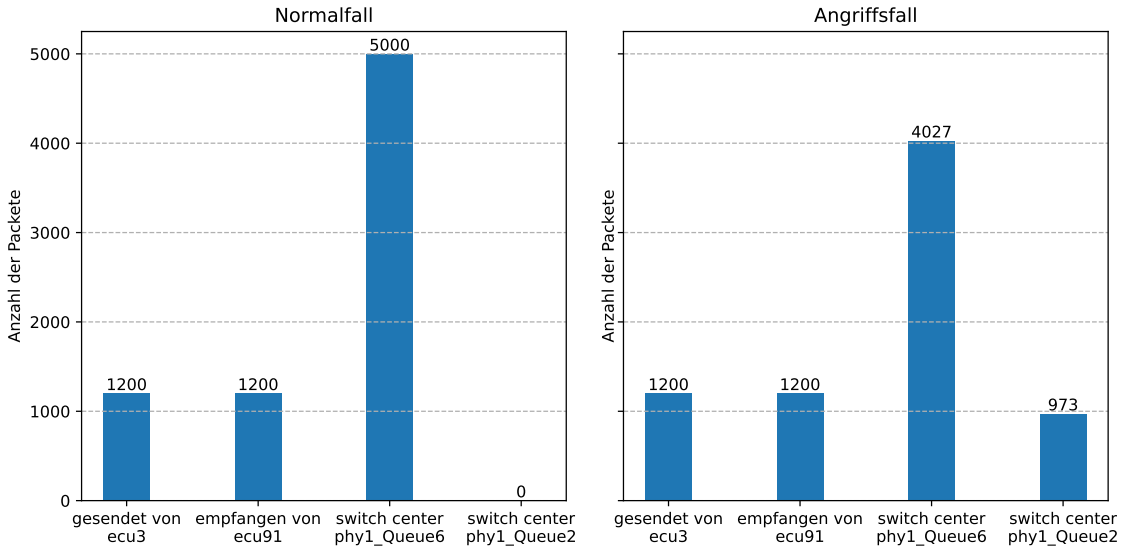

Abbildung 5.6: Manipulation in Gateway in RecBarToZone Netzwerk

Im Switch-Center werden die Pakete, die in den Queues mit der 2. und 6. Priorität des Portes 1 zugeordent sind, betrachtet. Wie aus der linken Grafik in Abbildung 5.6 scheinbar ist, sind in der Queue mit Priorität 6 im Switch-Center während der Simulation 5000[1] Pakete im Normalzustand angekommen und weitergeleitet worden. Diese Zahl ist inklusive anderer Datenverkehrsinformationen. Ebenso lässt sich erkennen, dass während der Simulation keine Pakete mit einer Priorität von 2 in die Queue des Switch-Centers im Normalfall angekommen sind.

[1] Für eine ansprechendere Darstellung der Grafik wurde diese Zahl reduziert. Die tatsächliche Anzahl beträgt 82680

Im Vergleich zum Normalzustand ist in der rechte Grafik der Abbildung zu sehen, dass die Anzahl der Pakete in der Queue von Priorität 6 auf 4027[2] reduziert wurde . Aufgrund des Angriffs wurde die Priorität einiger Pakete auf 2 geändert. Nun zeigt die rechte Grafik in Laufe des Angriffs, dass bei der Priorität 2 die Anzahl der Pakete von 0 auf 973 gestiegen ist. Dies bedeutet, dass 973 Pakete manipuliert wurden. Hier lässt sich sicherstellen, dass der Angriff im ersten Szenario erfolgreich war.

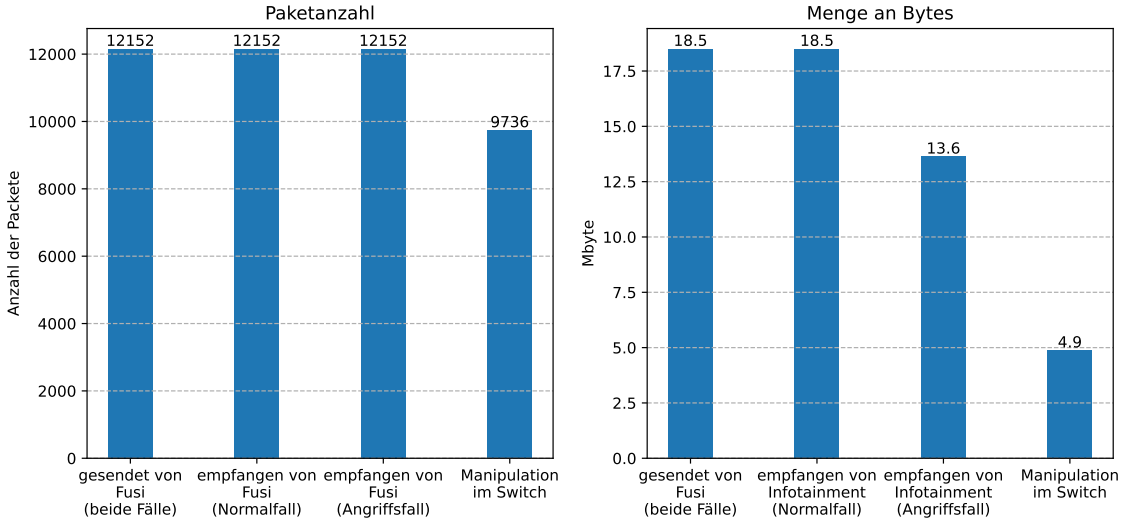

Abbildung 5.7: Manipulation in Switch in RecBarToZone Netzwerk

Im **zweiten Szenario** beabsichtigt der Angreifer, die Inhaltsdaten der Nachrichten zu kompromittieren oder zu verändern. Die Abbildung 5.7 zeigt die Anzahl der gesendeten und empfangenen Nachrichten sowie die übermittelten Datenmengen in Megabyte sowohl im Normalzustand als auch im Angriffsfall. Diese Daten stammen aus den Statistiken der Simulation.

Die linke Grafik in Abbildung 5.7 zeigt, dass während der Simulation in beiden Fällen insgesamt 12152 Pakete von Fusi gesendet wurden, die auch von Infotainment empfangen wurden. Laut den Statistiken wurden 9736 Pakete durch den Angriff beeinträchtigt und ihre Payload-Größe wurde manipuliert. Anhand dieser Daten kann man die folgende Formel verwenden, um zu berechnen, wie viele Bytes von Infotainment empfangen wurden.

$$Summe\,der\,Pakete\,in\,Byte = Ethernetframe\,Groesse\,in\,Byte * Anzahl\,der\,Pakete \tag{5.1}$$

[2]Für eine ansprechendere Darstellung der Grafik wurde diese Zahl reduziert. Die tatsächliche Anzahl beträgt 81707

In der Formel ist die Ethernetframe die maximale Größe des Frames. Insbesondere ist zu beachten, dass bei Verwendung des IEEE 802.1Q-Protokolls(Q-Tags) der Header des Frames um 4 Byte erweitert wird, wodurch der Frame maximal 1522 Byte groß wird.Damit kann man in beiden Fällen nachrechnen, dass bei einer Anzahl von 12152 Paketen, von denen jedes 1522 Byte groß ist, insgesamt 18.495.344 Byte von Fusi gesendet wurden. Diese Daten sind in der ersten Spalte der rechten Grafik der Abbildung 5.7 dargestellt. Die mittlere Säule der rechten Grafik zeigt, dass insgesamt 18.495.344 Byte (18,4 MByte) empfangen wurden. Nach der Simulation des Angriffs wurden die empfangenen Bytes durch Infotainment auf 13.627.344 Byte (13,6 MByte) reduziert. Wie in der rechten Grafik zu sehen ist, beträgt die Größe der Manipulation insgesamt 4.868.000 Byte (4,9 MByte). Hier lässt sich sicherstellen, dass der Angriff im zweiten Szenario erfolgreich war.

Evaluation der Angriffsergebnisse

In diesem Abschnitt werden die Auswirkungen einer MitM-Manipulation sowohl auf Anwendungs- als auch auf Netzwerksebene beschrieben. Der entscheidende Aspekt dieses Angriffs liegt darin, an welcher Stelle sich der Angreifer im Kommunikationsweg befindet. Im ersten Szenario ist er im Gateway-Center-Left positioniert und kann lediglich den übermittelten Datenverkehr von diesem Gateway ausspähen oder manipulieren. Im Gegensatz dazu, im zweiten Szenario, wo der Angreifer im Switch-Center platziert ist, hat er Zugriff auf unterschiedliche Datenströme aus verschiedenen Komponenten im Netzwerk, die er dann manipulieren kann.
Im ersten Szenario verfolgt der Angreifer das Ziel, dem Fahrer falsche Informationen anzuzeigen. Im Gateway-Center-Left wurden die Pakete mit hoher Priorität von der Tachometereinheit (ECU3) in eine niedrigere Priorität umgewandelt. Dies hat zur Folge, dass die tatsächliche Geschwindigkeit im Kombiinstrument nicht korrekt angezeigt wird und Störungen auf der Anwendungsebene verursacht oder im schlimmsten Fall zu einem Absturz der Anwendung führen kann. Anhand der Statistiken aus der Simulation zeigt die Tabelle 5.3 den Jitter und maximale Latenzzeit von Port1 am Switch-Center sowohl im Normalfall als auch im Angriffsfall. Im Angriffsfall ist in der Tabelle 5.3 zu erkennen, dass nach der Manipulation der Priorität diese Pakete einen Jitter von etwa 0,48 ms aufweisen. Diese Pakete werden durch die Prioritäten 6 und 3 blockiert und mit Verzögerung weitergeleitet.

Darüber hinaus hat diese Prioritätsänderung einen Einfluss auf die 0te und 1te Prioritäten. Im Normalfall gibt es keinen Datenverkehr in der 2ten Priorität des Ports1, und die 0te

	Switch Center (Port 1)			
	Normal		Manipulation	
Prior.	Jitter	Max	Jitter	Max
0	0,39384 ms	0,4076 ms	0,39384 ms	0,4076 ms
1	0,488246 ms	0,501877 ms	0,488352 ms	0,502112 ms
2	-	-	0,03248 ms	0,07312 ms
3	0,266035 ms	0,279803 ms	0,266020ms	0,279780 ms
4	-	-	-	-
5	-	-	-	-
6	0,212101 ms	0,225861 ms	0,212062 ms	0,225822 ms
7	-	-	-	-

Tabelle 5.3: Die Jitter und maximale Latenzzeit von Port 1 am Switch-Center

und 1te Prioritäten werden lediglich durch zwei höhere Prioritäten blockiert. Wie in der Tabelle 5.3 zu sehen ist, zeigt sich im Angriffsfall, dass die maximale Latenzzeit der 1ten Priorität um etwa 0,2 us gestiegen ist. Im Vergleich dazu wurden derJitter und die maximale Latenzzeit der 0ten Priorität nicht überschritten, obwohl diese auf hohe Prioritäten warten musste. Der Grund dafür ist, dass trotz der Überschreitung der Latenzzeit der 1ten Priorität die gelagerten Pakete in der 0ten Priorität nicht so lange warten mussten und ausreichend Bandbreite vorhanden war, um die Pakete in Zeitabständen zu versenden, in denen die 0te Priorität nicht blockiert war.
Bei den Prioritäten 3 und 6 ist zu erkennen, dass im Gegensatz zum Normalfall der Jitter und die maximalen Latenzzeiten im Angriffsfall reduziert wurden. Nach der Änderung der Priorität im Gateway-Center-Left wurden diese Pakete in der Queue mit der Priorität 2 zwischengespeichert. Dies bewirkt, dass andere Datenpakete mit der Priorität 6 vorangestellt werden. Aus diesem Grund sinkt die Latenzzeit der 6ten Priorität. Deshalb werden auch die Datenpakete mit der 3ten Priorität nicht länger durch die Pakete mit der 6ten Priorität blockiert, und sie können ebenfalls vorangestellt werden, was wiederum zu einer Verringerung der Latenzzeit der 3ten Priorität führt. Es steht nun fest, dass im ersten Szenario die manipulierten Pakete verzögert wurden, da sie eine niedrige Priorität aufweisen und auf Pakete mit hoher Priorität warten müssen. Dies beeinträchtigt nicht nur die Funktionalität der Switch-Center, sondern erschwert auch das Datenmapping in der Anwendungsebene bei ECU91.
Im zweiten Szenario hat der Angreifer das Ziel, falsche Daten an die Sensorfusion zu senden. In der Switch-Center wurden die Inhaltsdaten eines Pakets geändert. Dadurch

kann die Sensorfusion falsche Daten verarbeiten und den Fahrer nicht vor einer kritischen Situation warnen. Da die manipulierten Pakete eine geringere Größe haben, werden sie schneller übertragen, und es steht ausreichend Bandbreite im Netzwerk zur Verfügung. Diese Gelegenheit kann der Angreifer ausnutzen, um seine modifizierten Pakete zwischendurch zu senden.

5.2.2 Redirection

In den folgenden Unterkapiteln wird zunächst der Angriff und sein Aufbau im Kontext eines Szenarios beschrieben. Anschließend werden die Konfigurationen erläutert, die zur Realisierung des Angriffs notwendig sind, sowie die Umsetzung des Angriffs im Simulationsnetzwerk dargelegt. Zum Abschluss wird der simulierte Angriff validiert und werden seine exemplarischen Auswirkungen auf das Netzwerk erörtert.

Konzept und Aufbau des Angriffs

Die Fahrerassistenzsysteme ist zur Unterstützung des Fahrers in spezifischen Fahrsituationen, indem sie den Fahrer vor potenziellen Gefahren warnen. Diese Systeme verwenden Daten aus der Sensorfusion. Bei der Sensorfusion werden Informationen verschiedener Umgebungssensoren, wie Radar-, Kamera- und Lidar-Sensoren, zusammengeführt. Eine Möglichkeit, die es dem Angreifer erlaubt, in solchen Systemen Störungen zu erzeugen, ist die Redirection. Die Redirection oder Paketumleitung ist eine der verschiedenen Techniken von MitM-Angreiff, bei der der Angreifer in der Lage ist, Pakete an ein bestimmtes Ziel in einem Netzwerk umzuleiten.
In diesem Abschnitt werden die exemplarischen Auswirkungen der Redirection in einem Ethernet-basierten Fahrzeugnetzwerk untersucht. Dies wird durch das simulierte Szenario eines MitM-Angriffs erreicht. In diesem Szenario gehen wir davon aus, dass die Sensorfusion periodisch Werte von rechten und linken Lidarsensoren abfragt. In der Abbildung 5.8 wird eine kompromittierte Komponente namens Camera_Front(Cam_Front) dargestellt, auf dem der Angreifer per Software platziert ist. In dieser Komponente ist der Angreifer in der Lage, die Zieladresse der Nachrichten zu verändern, die von Cam_Front zum Infotainment gesendet werden. In der Abbildung 5.8 veranschaulichen die grünen Pfeile die ursprüngliche Richtung des Datenverkehrs von Cam_Front zum Infotainment. Innerhalb der kompromittierten Komponente verändert der Angreifer die Zieladresse der Ethernet-Nachrichten, sodass diese nun an den linken Lidarsensor (Lid_FrontLeft)

gesendet werden. Der rote Pfeil in der Abbildung symbolisiert diese Umleitung und zeigt somit die veränderte Datenverkehrs.

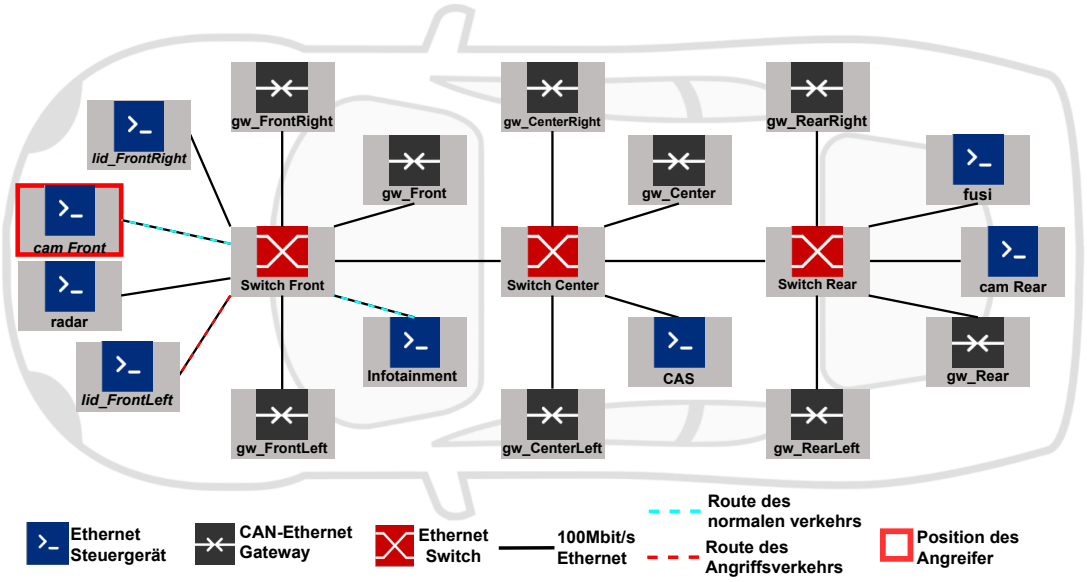

Abbildung 5.8: Redirection in RecBarToZone Netzwerk

Vorbreitung des Angriffs in der Simulation

Um das Nachrichtenverhalten des Szenarios in der Simulation zu modellieren, wird eine zusätzliche Anwendung in der Cam_Front-Komponente erstellt. Diese Anwendung stellt den Nachrichtenstrom zum Infotainment her, indem sie alle 9 ms periodisch eine Nachricht mit einer Größe von 1500 Bytes und einer Priorität von 3 an die MAC-Adresse des Infotainments sendet. Die INI-Datei der Cam_Front-Komponente wird mit dem folgenden Code konfiguriert:

Listing 5.8: Die Konfiguration zur Erstellung eines Nachrichtenstroms zum Infotainment

```
# Source app for Q message "msg_cam_FrontToInfotainment":
**.cam_Front.app[2].typename = "IEEE8021QTrafficSourceApp"
**.cam_Front.app[2].destAddress = "1F-1F-1F-1F-1F-1F"
**.cam_Front.app[2].sendInterval = uniform(0.009s, 0.009s)
**.cam_Front.app[2].payload = intWithUnit(uniform(1500Byte, 1500Byte))
**.cam_Front.app[2].priority = 3
**.cam_Front.app[2].vid = 0
**.cam_Front.app[2].startTime = 0 s
```

Die Zeile 2 des Listing 5.8 spezifiziert den Nachrichtentyp gemäß IEEE802.1Q. Die 3te Zeile definiert das Übertragungsintervall, wobei alle 9 ms eine Nachricht gesendet wird. Die Zeilen 4 bis 7 bestimmen die Eigenschaften der Nachricht, einschließlich der Unicast-MAC-Adresse der Infotainment-Komponente, der Nachrichtengröße, der zugehörigen Priorität und der VLAN-ID. In Zeile 8 wird der Startzeitpunkt festgelegt, ab dem die Anwendung beginnen soll, die Nachrichten zu versenden.

Listing 5.9: Die Konfiguration zur Erstellung einer Sinkapp für empfangenen Nachrichten von Cam_Fornt

```
1 # Sink app for message "msg_camFrontToinfotainment":
2 **.infotainment.app[0].typename = "IEEE8021QTrafficSinkApp"
3 **.infotainment.app[0].srcAddress = "F6-0E-A2-E0-49-14"
4 **.infotainment.app[0].replyFirst = false
```

Um Statistiken über die empfangenen Nachrichten im Infotainment zu sammeln, wird in der INI-Datei der Infotainment-Komponente eine Sinkapp mit der Konfiguration, wie im Listing 5.9 dargestellt, eingerichtet. Da der Nachrichtentyp bereits im Listing 5.8 angegeben wurde, sollte die Konfiguration der Sink-App denselben Typ aufweisen. Zeile 2 legt den Typ der Sinkapp-Anwendung als IEEE8021QTrafficSinkApp fest.Die Zeile 3 definiert, von welcher Quelladresse die empfangenen Nachrichten sein sollen. Die Zeile 4 dient dazu, den ersten Frame zur Aktualisierung der Tabellen in den Switches zu wiederholen.

Umsetzung des Angriffs in der Simulation

Um einen Angriff durch Redirection durchzuführen, wurden in der Cam_Front-Komponente des RecBarToZone-Netzwerks einige Änderungen vorgenommen. Diese Änderungen werden durch den Konfigurationscode erläutert.
Im Netzwerk, wie in der Abbildung 5.8 dargestellt, sendet die Cam_Front-Komponente alle 9 ms eine IEEE802.1Q-Nachricht mit einer Größe von 1500 Bytes an die Unicast-MAC-Adresse des Infotainment, und zwar mit einer Priorität von 3. In der Simulation setzt sich das TSNPort-Modul aus verschiedenen Komponenten zusammen. Dieses Modul beinhaltet ein Queueing-Modul, acht Queues, ein Transmission-Algorithmus-Modul, Transmission-Gate und Transmission-Selection. Das Transmission-Selection-Modul ist für das Entnehmen eines Pakets aus einer Queue zur weiteren Übertragung zuständig. Im Rahmen des Frameworks von P. Meyer[7] wurde das Transmission-Selection-Modul

modifiziert, um zusätzliche Funktionen bereitzustellen. Zur Simulation des Angriffs werden mithilfe dieses modifizierte Modul Nachrichten gezielt verändert. Diese Nachrichten werden nach der Veränderung weitergesendet.
Um sowohl die Position des Angreifers als auch den Angriff selbst zu simulieren, werden folgende Änderungen in der INI-Datei von Cam_Front umgesetzt:

Listing 5.10: Redirection-Konfigurationscode im Camera Front

```
1  [Config Selection_Redirection]
2  #ersetzen des Moduls durch kompromittierte Transmission-Selection
3  **.cam_Front.phy[*].**.transmissionSelection.typename = "
       ↪ CorruptIEEE8021QbvSelectionManipulation"
4  #Filter,mit der die Nachricht kompromittiert wird
5  **.cam_Front.phy[*].**.transmissionSelection.corruptionDestAddress = "1F-1F-1F-1F
       ↪ -1F-1F"
6  **.cam_Front.phy[*].**.transmissionSelection.corruptionPriority = 3
7  **.cam_Front.phy[*].**.transmissionSelection.corruptionVid = 0
8  **.cam_Front.phy[*].**.transmissionSelection.corruptionProbability = 0.85
9  #Modifizierung der Zieladresse
10 **.cam_Front.phy[*].**.transmissionSelection.manipulationDestAddress = "DA-54-3A
       ↪ -52-D2-5F"
```

In Zeile 3 des Listing 5.10 wird die modifizierte Transmission-Selection festgelegt. Diese ersetzt die standardmäßige Transmission-Selection von Cam_Front und stellt den Angreifer dar. In den Zeilen 5-7 von Listing 5.10 werden die Eigenschaften der Nachricht eingetragen, wie die MAC-Adresse, die Priorität und zugehörige VLAN-ID der zu kompromittierenden Nachricht. In Zeile 8 wird die Wahrscheinlichkeit eingegeben, mit der eine Nachricht manipuliert werden kann. Wenn dieser Wert auf 1 gesetzt wird, bedeutet dies, dass jede Nachricht manipuliert wird. In Zeile 10 von Listing 5.10 wird die ursprüngliche Ziel-MAC-Adresse der Nachricht auf die Unicast-MAC-Adresse von Infotainment ersetzt, was den Angriff auf die Nachricht darstellt.

Qualitätsicherung des Angriffs

In diesem Abschnitt wird der Redirectionsangriff validiert. Das beschriebene Szenario wurde 20 Sekunden lang simuliert und die Ergebnisse sind in Grafiken dargestellt. Um mögliche Änderungen an den Nachrichten im Cam_Front festzustellen, wird das Ziel des Angreifers betrachtet. In diesem Szenario beabsichtigt der Angreifer gezielt, die Nachrichten zu linke Lidarsensor(lid_FrontLeft) umzuleiten. Daher werden sowohl die Anzahl der

von Fusi und Cam_Front gesendeten Nachrichten als auch die Anzahl der von lid_Front-Left und Infotainment empfangenen Nachrichten berücksichtigt. Die Abbildung 5.9 stellt die genannten Metriken aus den Statistiken der Simulation dar.

Die linke Grafik der Abbildung 5.9 zeigt gesendete und empfangene Nachrichten im

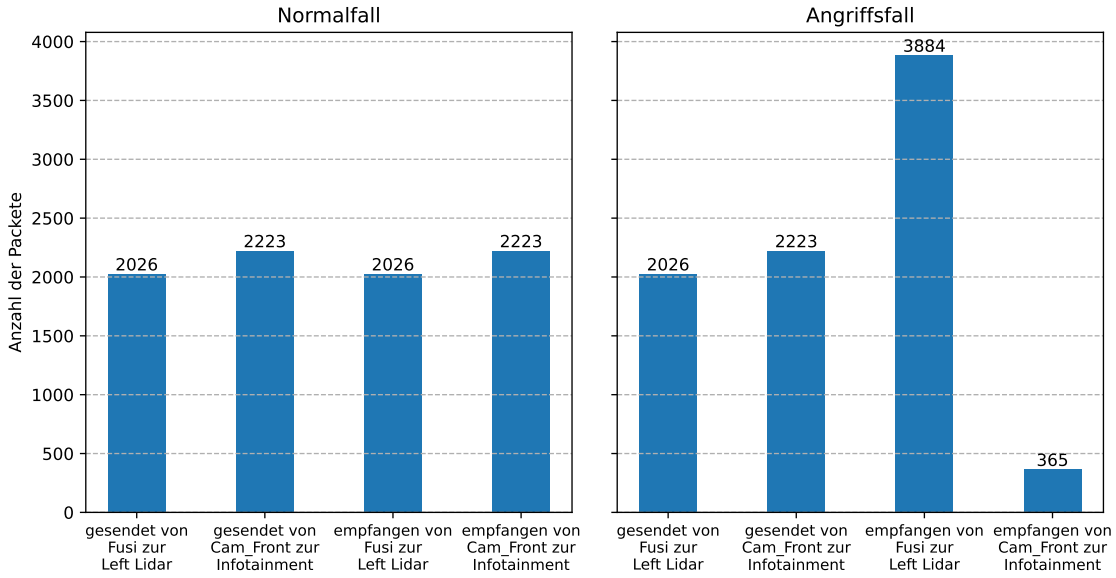

Abbildung 5.9: Grafische Darstellung der Anzahl der gesendeten, empfangenen und komprimierten Pakete durch Ridirection-Simulation

Normalfall. Im Verlauf der Simulation hat Fusi 2026 Paketnachrichten an den linken Lidarsensor gesendet, die von lid_FrontLeft empfangen wurden. Parallel dazu hat Cam_-Front 2223 Paketnachrichten an das Infotainment gesendet, welche ebenfalls erfolgreich empfangen wurden. Im Angriffsfall zeigt die rechte Grafik der Abbildung 5.9 andere Metriken. Wie im Normalfall wurden gleiche Anzahl von Nachrichten gesendet. Die Anzahl der empfangenen Nachrichten von Infotainment ist auf 365 Pakete gesunken, während im Vergleich dazu die empfangenen Nachrichten vom linken Lidarsensor auf 3884 Pakete gestiegen sind. Es ist offensichtlich, dass die Umleitung (Redirection) der Pakete der Grund für diese Veränderung ist. Laut Statistiken wurden 1858 Pakete korrumpiert und wurden an Infotainment umgeleitet.

Evaluation der Angriffsergebnisse

In diesem Abschnitt werden die exemplarischen Auswirkungen eines MitM-Angriffs beschrieben. Im Netzwerk ist der Angreifer in der Komponente Cam_Front positioniert und hat die Unicast-MAC-Adresse der Nachrichten, die von Cam_Front an das Infotainment gesendet werden, auf den linken Lidarsensor umgeändert. Die Sensorfusion sendet Kontrollnachrichten an beide Lidarsensoren mit einer Priorität von 3. Die komprimierten Nachrichten weisen dieselbe Priorität wie die Kontrollnachrichten auf. Deshalb werden die Umleitungseffekte auf die Netzwerkleistung des linken Lidarsensors untersucht. Diese Auswirkungen werden durch die Statistiken aus der Simulation analysiert. Die Tabelle 5.4 zeigt die maximale Latenzzeit der angekommenen Nachrichten beim linken Lidarsensor. Laut der Tabelle 5.4 liegt die maximale Latenzzeit im Normalfall bei 0,545 ms. Nach dem Angriff überschreitet diese Latenzzeit den Normalwert und erhöht sich um 0,08 ms. Nach der Kompromittierung wurden Nachrichten der Cam_Front in der Queue des Portes 5 des Switch-Front zugeordent, was dazu führte, dass die ursprünglichen Nachrichtenpakete verzögert und erst zu einem späteren Zeitpunkt weitergeleitet wurden. Aus diesem Grund ist die Latenzzeit bei linken Lidarsensor von 0,545 ms auf 0,625 ms gestiegen und hat somit maximale Latenzzeit überschritten.

	maximale latenzzeit	
Name der Komponente	**Normalfall**	**Angirffsfall**
Left Liedarsensor	0,54555ms	0,62616 ms

Tabelle 5.4: Die maximale empfangene Latenzzeit von Left Lidarsensor

Die Ergebnisse verdeutlichen, dass auf Anwendungsebene die Verzögerungen beim linken Lidarsensor zu einer erhöhten Verarbeitungszeit dieser Nachrichten und somit zum Entzug von Ressourcen führen. Andererseits resultieren aus den Verzögerungen fehlerhafte Daten, die an die Sensorfusion zurückgesendet werden, wodurch die Fahrerassistenzsysteme kritische Situationen nicht erkennen können.

5.2.3 Elimination

In den folgenden Unterkapiteln wird zunächst der Angriff und sein Aufbau im Kontext eines Szenarios beschrieben. Anschließend werden die Konfigurationen zur Umsetzung des Angriffs im Simulationsnetzwerk dargelegt. Zum Abschluss wird der simulierte Angriff validiert und werden seine exemplarischen Auswirkungen auf das Netzwerk erörtert.

Konzept und Aufbau des Angriffs

Ein Kollisionsvermeidungssystem (CAS) stellt ein fortschrittliches Fahrerassistenzsystem dar, das darauf abzielt, Unfallrisiken zu verhindern oder zu minimieren. Es überwacht kontinuierlich sowohl die Geschwindigkeit des eigenen Fahrzeugs als auch die des vorausfahrenden Fahrzeugs sowie den Abstand zwischen beiden. Dadurch kann es den Fahrer rechtzeitig warnen, wenn sich die Fahrzeuge stark annähern, und somit dabei helfen, potenzielle Unfälle zu vermeiden. Für die Erkennung drohender Unfallsituationen werden Technologien wie Radar, Lidar-Sensoren und Kameras eingesetzt.
Eine Möglichkeit, die es einem Angreifer ermöglicht, Störungen in CAS zu erzeugen, ist die Elimination. Dies ist eine von verschiedenen Techniken eines MitM-Angriff, bei der der Angreifer die Ethernetframes auf dem Kommunikationsweg in einem Netzwerk eliminieren kann. In diesem Abschnitt werden die exemplarischen Auswirkungen der Elimination in einem Ethernet-basierten Fahrzeugnetzwerk anhand eines Szenarios untersucht.
Wie bereits erwähnt, setzt das CAS verschiedene Technologien und Sensoren ein, unter anderem Kameras mit Bilderkennung. In diesem Szenario gehen wir davon aus, dass die Frontkamera(Cam_Front) Daten an die CAS-Komponente sendet. Wie in der Abbildung 5.10 dargestellt, platziert der Angreifer per Software auf dem Switch-Front. Innerhalb dieses kompromittierten Switches kann der Angreifer die Ethernetframes, die von der Cam_Front zum CAS gesendet werden, löschen, um eine drohende Gefahr zu verhindern. Die grünen Pfeile in der Abbildung 5.10 zeigen die Richtung des Datenverkehrs an.

Umsetzung des Angriffs in der Simulation

Um einen Angriff durch Elimination durchzuführen, wurden im Switch-Front des RecBarToZone-Netzwerks einige Änderungen vorgenommen. Diese Änderungen werden durch den Konfigurationscode erläutert. Im Netzwerk, wie in der Abbildung 5.10 dargestellt, sendet die Cam_Front alle 100 ms eine IEEE802.1Q-Nachricht mit einer Größe von 256 Bytes an die Unicast-MAC-Adresse der CAS-Komponente, und zwar mit einer Priorität von 6. Nach dem Einschleusen infiziert der Angreifer das Switch-Front mithilfe der installierten Software. Über dieses kompromittierte Switch(siehe Abbildung 5.10) kann er den Ethernetframe auf dem Kommunikationsweg löchen. In der Simulation setzt sich das TSNPort-Modul aus verschiedenen Komponenten zusammen. Dieses Modul beinhaltet ein Queueing-Modul, acht Queues, ein Transmission-Algorithmus-Modul, Transmission-Gate und Transmission-Selection. Das Transmission-Selection-Modul ist für das Entnehmen

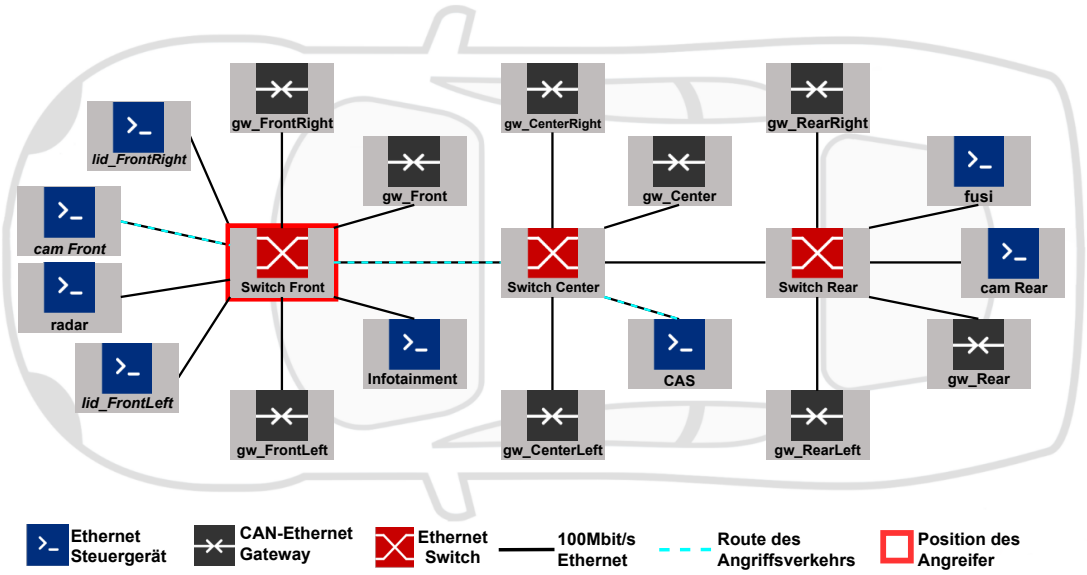

Abbildung 5.10: Elimination in RecBarToZone Netzwerk

eines Nachrichtspakets aus einer Queue zur weiteren Übertragung zuständig. Im Rahmen des Frameworks von P. Meyer[7] wurde das Transmission-Selection-Modul modifiziert, um zusätzliche Funktionen bereitzustellen. Zur Simulation des Angriffs werden Nachrichten mithilfe dieses modifizierten Moduls gezielt eliminiert, bevor sie übertragen werden.
Um sowohl die Position des Angreifers als auch den Angriff selbst zu simulieren, werden folgende Änderungen in der INI-Datei von Switch-Front umgesetzt:

Listing 5.11: Elimination-Konfiguration Code in Switch Front

```
1 [Config Selection_Loss_CAS]
2 #ersetzen des Moduls durch kompromittierte Transmission Selection
3 **.switch_Front.phy[3].**.transmissionSelection.typename = "
    ↪ CorruptIEEE8021QbvSelectionLoss"
4 #Filter,mit der die Nachricht kompromittiert wird
5 **.switch_Front.phy[3].**.transmissionSelection.corruptionDestAddress = "CA-CA-CA-
    ↪ CA-CA-CA"
6 **.switch_Front.phy[3].**.transmissionSelection.corruptionPriority = 6
7 **.switch_Front.phy[3].**.transmissionSelection.corruptionVid = 0
8 **.switch_Front.phy[3].**.transmissionSelection.corruptionProbability = 0.85
```

In Zeile 3 von Listing 5.11 wird die modifizierte Transmission-Selection festgelegt. Diese ersetzt die standardmäßige Transmission-Selection des Ports 3 von Cam_Front und stellt den Angreifer dar. In den Zeilen 5-7 von Listing 5.11 werden die Eigenschaften der zu

eliminierenden Nachricht eingetragen, wie die MAC-Adresse, die Priorität und zugehörige VLAN-ID der Nachricht. In Zeile 8 wird die Wahrscheinlichkeit eingegeben, mit der eine Nachricht eliminiert werden kann. Wenn dieser Wert auf 1 gesetzt wird, bedeutet dies, dass jede Nachricht eliminiert wird.

Qualitätsicherung des Angriffs

In diesem Abschnitt wird der Eliminationsangriff validiert. Das beschriebene Szenario wurde über einen Zeitraum von 5 Sekunden simuliert, und die Ergebnisse sind in Grafiken dargestellt. Im Szenario beabsichtigt der Angreifer gezielt, die Nachrichten, die an die CAS-Komponente gesendet werden, zu eliminieren. Anhand von Statistiken aus der Simulation werden die Anzahl der gesendeten und empfangenen Nachrichten sowie die Anzahl der eliminierten Nachrichten ermittelt, um festzustellen, dass diese Nachrichten tatsächlich im Switch-Front eliminiert wurden. Die Abbildung 5.11 wurde anhand der genannten Metriken erstellt.

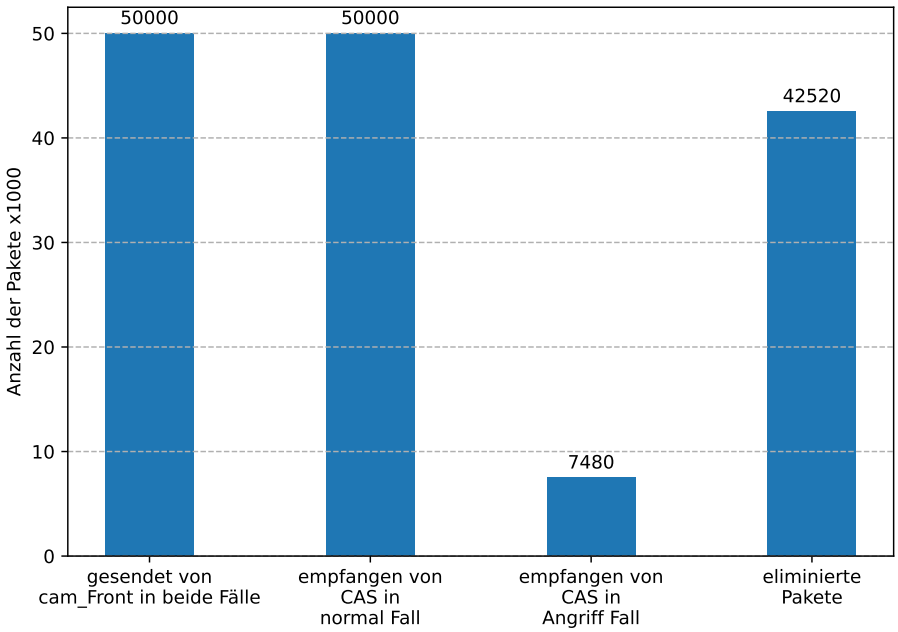

Abbildung 5.11: Grafische Darstellung der Anzahl der gesendeten, empfangenen und eliminiereten Pakete durch Elimination

In der Abbildung 5.11 ist zu erkennen, dass sowohl im normalen als auch im Angriffsfall 50.000 Pakete von Cam_Front an die Can-Komponente gesendet wurden. Im Normalfall wurden diese Pakete vollständig von der CAS-Komponente empfangen. Im Gegensatz dazu zeigt die Abbildung 5.11 für den Angriffsfall, dass 42.520 Pakete im Switch-Front eliminiert wurden. Das bedeutet, dass von den 50.000 gesendeten Paketen lediglich 7.480 Pakete von der CAS-Komponente empfangen wurden.

Evaluation der Angriffsergebnisse

In diesem Abschnitt werden die exemplarischen Auswirkungen eines Elimination-Angriffs beschrieben. Aus der Abbildung 5.11 lässt sich ableiten, dass die Elimination den Datenstrom zwischen Cam_Front und der CAS-Komponente beeinträchtigt, indem sie die Ethernetpakete entfernt.
Die Eliminierung von Ethernetpaketen kann in Zusammenhang dieses Szenarios die Funktionsweise des Kollisionsvermeidungssystems (CAS) beeinträchtigen oder dazu führen, dass bestimmte Funktionen fehlschlagen, wodurch das Verhalten der CAS-Anwendungen manipuliert wird. Wie bereits erwähnt, nutzt das CAS verschiedene Technologien, darunter Sensoren und Kameras mit Bilderkennung. Da der Angreifer einige Pakete des Datenstroms eliminiert, ist davon auszugehen, dass Schwankungen bei den Übertragungszeiten auftreten. Dies kann zu einem ungleichmäßigen Datenstrom oder möglicherweise zu unterbrochener Kommunikation führen.
Auf der Netzwerkebene kann durch diese Eliminierung zusätzliche Bandbreite entstehen. Diese könnte verbraucht werden, wenn einige Komponenten im Netzwerk versuchen, die verlorenen Pakete durch erneute Übertragungen zu kompensieren. Dies könnte zur Folge haben, dass die verfügbare Gesamtbandbreite für die Dienste anderer Komponenten verringert wird.

5.2.4 Rescheduling

In den folgenden Unterkapiteln wird zunächst der Angriff und sein Aufbau im Kontext eines Szenarios beschrieben. Anschließend werden die Konfigurationen zur Umsetzung des Angriffs im Simulationsnetzwerk dargelegt. Zum Abschluss wird der simulierte Angriff validiert und werden seine exemplarischen Auswirkungen auf das Netzwerk erörtert.

Konzept und Aufbau des Angriffs

Bei der Sensorfusion werden Daten von verschiedenen Sensoren, wie beispielsweise Radarsensoren oder Lidar, zusammengeführt, um Umgebungsbedingungen präzise zu interpretieren und somit eine sichere Erkennung zu gewährleisten. In Fahrzeugen scannen Lidarsensoren kontinuierlich die Umgebung, um detaillierte Informationen über Position, Größe und Bewegung von Objekten in ihrer Nähe zu liefern. Eine Methode, um Störungen bei der Zusammenführung solcher Daten zu erzeugen, ist das Rescheduling. Rescheduling, oder auch Umplanung, zählt zu den verschiedenen Techniken eines Man-in-the-Middle-Angriffs (MitM), bei dem der Angreifer in der Lage ist, Datenpakete an ein spezifisches Ziel innerhalb eines Netzwerks mit Verzögerung zu versenden.

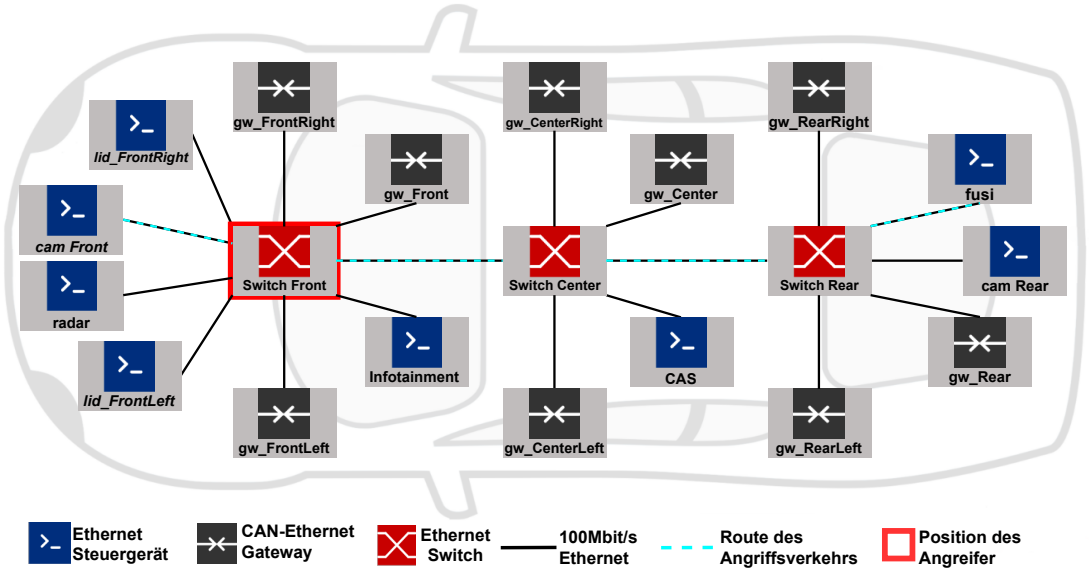

Abbildung 5.12: Reschaduling in RecBarToZone Netzwerk

In diesem Abschnitt werden die exemplarischen Auswirkungen der Rescheduling in einem Ethernet-basierten Fahrzeugnetzwerk untersucht. Dies wird anhand eines simulierten

Szenarios eines Man-in-the-Middle (MitM)-Angriffs erreicht. Dabei nehmen wir an, dass die Lidarsensoren auf der rechten und linken Seite des Fahrzeuges kontinuierlich Daten über die gescannte Umgebung an die Sensorfusion senden. Wie in der Abbildung 5.12 dargestellt, platziert der Angreifer per manipulative Software auf dem Switch-Front an der Netzwerk. Durch diese Positionierung im Netzwerk ist es dem Angreifer möglich, jeden Datenverkehr gezielt zu verzögern. Sein Ziel ist es, innerhalb dieses kompromittierten Switches die Daten, die vom rechten Lidarsensor an die Sensorfusion gesendet werden, zu verzögern. Eine solche Verzögerung kann die Berechnungen der Sensorfusion für die Ermittlung der optimalen Fahrstrategie beeinflussen.

Umsetzung des Angriffs in der Simulation

Um einen Angriff durch Rescheduling durchzuführen, wurden einige Änderungen in dem Switch-Front des RecBarToZone-Netzwerks vorgenommen. Diese Änderungen werden durch den Konfigurationscode erläutert.
Im Netzwerk, wie in der Abbildung 5.12 dargestellt, sendet der rechte Lidarsensor (lid_-Right) alle 0.039875 Sekunden eine IEEE802.1Q-Nachricht mit einer Größe von 1500 Bytes an die Unicast-MAC-Adresse der Fusi-Komponente, und zwar mit einer Priorität von 5. Zur Simulation des Angriffs wird das Framework von P. Meyer[7] verwendet. Dieses Framework bietet eine modifizierte Transmission-Selection, um die Übertragung von Nachrichten in dem Port gezielt zu verzögern. Um sowohl die Position des Angreifers als auch den Angriff selbst zu simulieren, werden folgende Änderungen in der INI-Datei von Switch-Front umgesetzt:

Listing 5.12: Rescheduling Konfiguration Code in Switch Front

```
1  [Config Selection_Rescheduling]
2  #ersetzen des Moduls durch kompromittierte Transmission Selection
3  **.switch_Front.phy[3].**.transmissionSelection.typename = "
       ↪ CorruptIEEE8021QbvSelectionTiming"
4  #Filter,mit der die Nachricht kompromittiert wird
5  **.switch_Front.phy[3].**.transmissionSelection.corruptionDestAddress = "AB-AA
       ↪ -00-00-00-05"
6  **.switch_Front.phy[3].**.transmissionSelection.corruptionPriority = 5
7  **.switch_Front.phy[3].**.transmissionSelection.corruptionVid = 0
8  **.switch_Front.phy[3].**.transmissionSelection.corruptionProbability = 1
9  #Modifizierung fuer Verzoegerung einer Nachricht
10 **.switch_Front.phy[3].**.transmissionSelection.corruptionMinInterval = 60ms
11 **.switch_Front.phy[3].**.transmissionSelection.delayTime = uniform(5ms, 5ms)
12 **.switch_Front.phy[3].**.transmissionSelection.allowOtherTrafficDuringDelay =
       ↪ true
```

In Zeile 3 des Listing 5.12 wird die modifizierte Transmission-Selection festgelegt. Diese ersetzt die standardmäßige Transmission-Selection von Switch_Front und stellt den Angreifer dar. In den Zeilen 5-7 von Listing 5.12 werden die Eigenschaften der Nachricht eingetragen, wie die Ziel-MAC-Adresse, die Priorität und zugehörige VLAN-ID der zu kompromittierenden Nachricht. In Zeile 8 wird die Wahrscheinlichkeit eingegeben, mit der eine Nachricht manipuliert werden kann. In diesem Fall ist die Verzögerung einer Nachricht. Wenn dieser Wert auf 1 gesetzt wird, bedeutet dies, dass jede Nachricht verzögert wird. In Zeile 10 wird das minimale Intervall von 60 ms angegeben, in dem der Angriff stattfindet. In Zeile 11 von Listing 5.12 wird die Verzögerungszeit eingegeben, mit der eine Nachricht verzögert werden muss. In Zeile 12 wird bestimmt, dass während der Verzögerung andere Datenverkehre übertragen werden können und nicht von der verzögerung betroffen werden. Die Zeilen 10-12 stellen den Angriff auf die Nachricht dar.

Qualitätsicherung des Angriffs

In diesem Abschnitt wird der Rescheduling-Angriff validiert. Das zuvor beschriebene Szenario wurde über einen Zeitraum von 20 Sekunden simuliert, und die Ergebnisse sind in Form von Grafiken visualisiert.
Um eine mögliche Verzögerung auf der Nachrichten im Switch-Front festzustellen, fokussieren wir uns auf die Ziele des Angreifers. Im Rahmen des Szenarios zielt der Angreifer darauf ab, die an die Sensorfusion gesendeten Nachrichten gezielt zu verzögern. Folglich nehmen wir die Latenzzeit der Nachrichten in den Blick, die vom rechten Lidarsensor empfangen wurden. Die Abbildung 5.13 veranschaulicht die entsprechenden Metriken, basierend auf den statistischen Daten der Simulation.

Die Grafik in der Abbildung 5.13 zeigt eine geringe Latenzzeit für empfangene Nachrichten in der Sensorfusion unter Normalbedingungen. Gemäß den Simulationstatistiken beträgt die maximale Latenzzeit 2,64 ms. Dies gewährleistet, dass die Sensorfusion umfassendere und genauere Daten aus rechtem Lidarsensor rechtzeitig erhält und verarbeitet.
In der unteren Grafik der Abbildung 5.13, die den Zustand nach einem Angriff darstellt, sind Schwankungen in den Latenzzeiten ersichtlich. Aufgrund der erzeugten Verzögerung liegt die Latenzzeit mehrerer Datenpakete zwischen 6,0 ms und 7,5 ms. Gemäß den Simulationstatistiken beträgt die maximale Latenzzeit 7,4 ms.

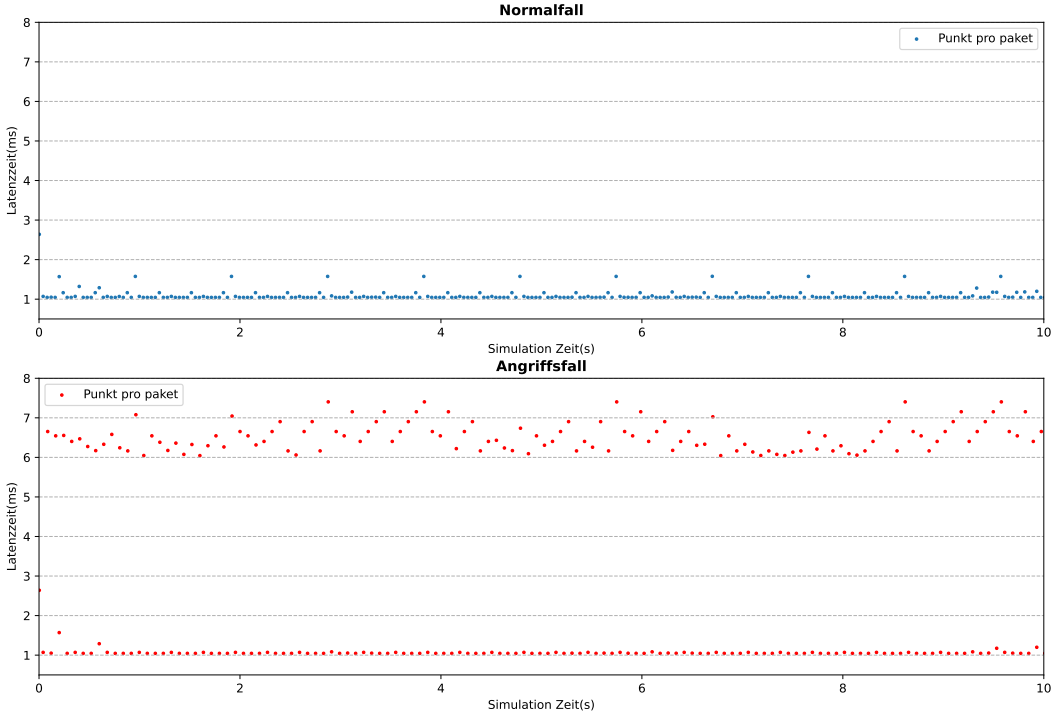

Abbildung 5.13: Die Latenzzeit empfangener Nachrichten bei der Sensorfusion unter Normalbedingungen und im Falle eines Angriffs

Evaluation der Angriffsergebnisse

In diesem Abschnitt werden die exemplarischen Auswirkungen eines Rescheduling-Angriffs beschrieben. Der Angreifer hat einige Pakete im Datenstrom, die von der Cam_Front zur Fusi-Komponente gesendet wurden, verzögert. Mithilfe von Statistiken aus der Simulation wurde die folgende Tabelle erstellt.
Die Tabelle 5.5 zeigt die maximale Latenzzeit bei der Fusi-Komponente. Im Normalfall beträgt die maximale Latenzzeit 2,63 ms. Nach dem Angriff wurden einige Pakete verzögert, wodurch die maximale Latenzzeit auf 7,40 ms anstieg. Diese Verzögerung bei der Datenübertragung kann zeitkritische Anwendungen in der Fusi-Komponente beeinträchtigen. Da die Fusi-Komponente die Kameradaten nicht rechtzeitig erhält, kann dies nicht nur zu einer Fehlfunktion dieser Komponente führen, sondern auch dazu, dass diese unbrauchbaren Daten zu falschen Entscheidungen führen.

Weiterhin kann eine durch Rescheduling verursachte Verzögerung dazu führen, dass Pakete außerhalb ihrer geplanten Zeitfenster gesendet werden. Dies kann Kollisionen mit anderen

Name der Komponente	Normalfall	Angirffsfall
Fusi	2,63912 ms	7,4035 ms

Tabelle 5.5: Die maximale Latenzzeit der Fusi-Komponente

Paketen verursachen und das Netzwerk überlasten. Die Tabelle 5.6 zeigt die maximale Latenzzeit des Ports 4 von Switch_Center.
In der Tabelle 5.6 ist ein Anstieg der maximalen Latenzzeit der sechsten Priorität im Angriffsfall zu sehen. Der Grund dafür ist, dass die verzögerten Pakete die Bandbreite aufgebraucht haben, was zu einer Blockierung der Pakete mit der Priorität 6 führt. Auf der anderen Seite wurden die Pakete mit der Priorität 3 durch diese Verzögerung verdrängt und weisen daher im Vergleich zum Normalfall eine niedrigere Latenzzeit auf.

	Normalfall	Angriffsfall
Priorität	max	max
3	0,26069 ms	0,26039 ms
5	2,9758 ms	7,15077 ms
6	0,23152 ms	0,27888 ms

Tabelle 5.6: Die maximale Latenzzeit am Port 4 des Switch_Center

5.3 MAC Flooding Angriff

In den folgenden Unterkapiteln werden zunächst der Angriff und sein Aufbau im Kontext eines Szenarios beschrieben. Anschließend werden die Konfigurationen erläutert, die zur Realisierung des Angriffs notwendig sind, sowie die Umsetzung des Angriffs im Simulationsnetzwerk dargelegt. Zum Abschluss wird der simulierte Angriff validiert und werden seine exemplarischen Auswirkungen auf das Netzwerk erörtert.

Konzept und Aufbau des Angriffs

Eine der primären Funktionen eines Ethernet-Switches besteht darin, die Weiterleitung von Ethernetpaketen innerhalb eines Netzwerks zu steuern. Dabei lernt der Switch die MAC-Adressen der angeschlossenen Geräte, indem er die Quell-MAC-Adressen von eingehenden Paketen erlernt. Diese Adressen werden mit der zugehörigen Portnummer zusammen in einer Tabelle, die als MAC-Adresstabelle bekannt ist, gespeichert. Sobald der Switch die Quell-MAC-Adresse in seiner Tabelle gespeichert hat, verwendet er die Ziel-MAC-Adresse, um zu bestimmen, an welchen spezifischen Port das Paket weitergeleitet werden soll. Diese Funktion verhindert, dass der Switch das Paket an alle Ports weiterleitet. Stattdessen sendet er das Paket nur an den Port, an dem das Zielgerät angeschlossen ist.

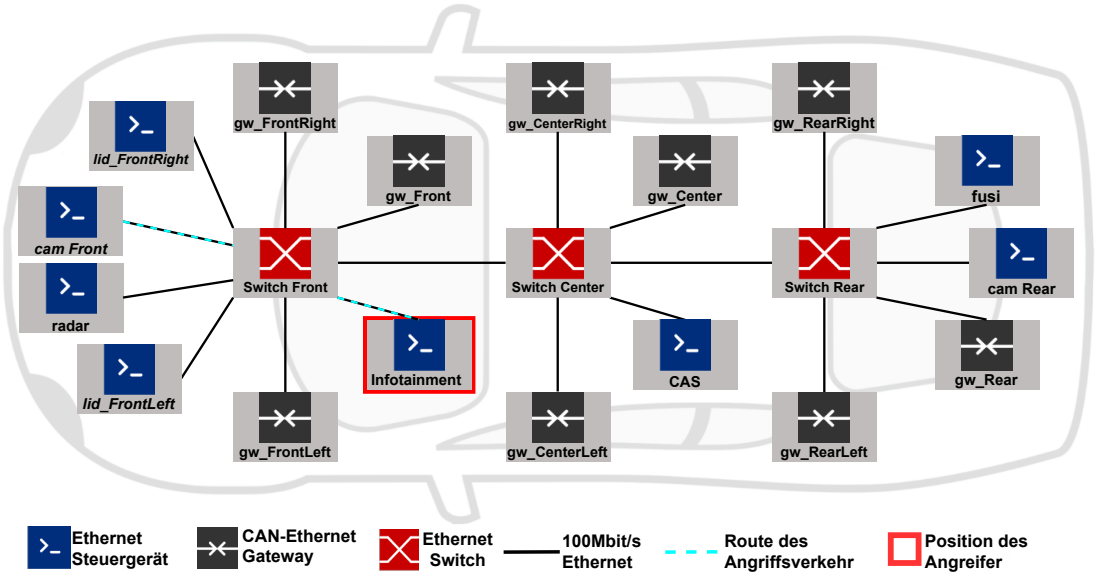

Abbildung 5.14: MAC Flooding in RecBarToZone Netzwerk

In diesem Szenario gehen wir davon aus, dass ein Angreifer versucht, Daten innerhalb des Fahrzeugnetzwerks zu kompromittieren. Wie in der Abbildung 5.14 dargestellt, platziert er per Software auf der Infotainment-Komponente. Bei diesem Angriff wird die Switch-Front mit einer Vielzahl von Datenpaketen von der kompromittierten Infotainment-Komponente bombardiert. Das Ziel besteht darin, die MAC-Adresstabelle von Switch-Front mit einer großen Menge gefälschter Ethernetpakete zu überfluten. Jedes Ethernetpaket enthält eine zufällige Quell-MAC-Adresse. Der Switch lernt und speichert jede dieser gefälschten MAC-Adressen, bis seine Quelladresstabelle voll ist. Nachdem die MAC-Adresstabelle überläuft, schaltet der Switch in seiner Konfiguration in einen sogenannten "Fail-Open-Modus". In diesem Zustand verhält sich der Switch wie ein Netzwerk-Hub und leitet erhebliche Mengen eingehender Pakete an alle Ports weiter. Dies führt zur Übertragung sensibler Daten an andere Komponenten des Netzwerks.

Vorbreitung des Angriffs in der Simulation

Um das Switchverhalten für das vorgegebene Szenario in der Simulation zu modellieren, wurden bestimmte Module des Switches modifiziert. Die Abbildung 5.15 präsentiert die Komponenten eines TSN-Switches innerhalb der Simulation. Die modifizierten Module sind in der Abbildung durch rote Kreise markiert.

Zunächst wurde eine angepasste MAC-Adresstabelle erstellt, um dem Switch einen begrenzten Speicherplatz für das Erlernen neuer MAC-Adressen zur Verfügung zu stellen. Wie in der Abbildung 5.15 zu sehen ist, wurde ein Modul mit dem Namen CustomMACAddressTable erstellt. Dieses Modul erbt von der tatsächlichen MACAddressTable des INET-Framworks und verfügt über einen Parameter sowie eine Funktion. Der Parameter 'maxSize' legt die maximale Anzahl der MFunktionAC-Adressen fest, die in der Tabelle des Switches gespeichert werden können. Die isMACAddressTableOverflow-Funktion dient zur Überprüfung, ob die MAC-Adresstabelle voll ist. Sie wurde so implementiert, dass sie die Gesamtanzahl aller MAC-Adressen aus allen VLAN-Tabellen des Switches summiert. Wenn diese Summe größer oder gleich maxSize ist, gibt die Funktion den Wert $'True'$ zurück.

Im Anschluss daran wurde das SRP-Relay-Modul des Switches modifiziert. Das SRP-Relay-Modul ist dafür zuständig, Etherframes auf Basis ihrer Ziel-MAC-Adressen an die jeweils zugeordneten Ports weiterzuleiten. Wie in der Abbildung 5.15 zu sehen ist, wurde dieses Modul um eine Funktion erweitert: CustomHandleAndDispatchFrame. Diese

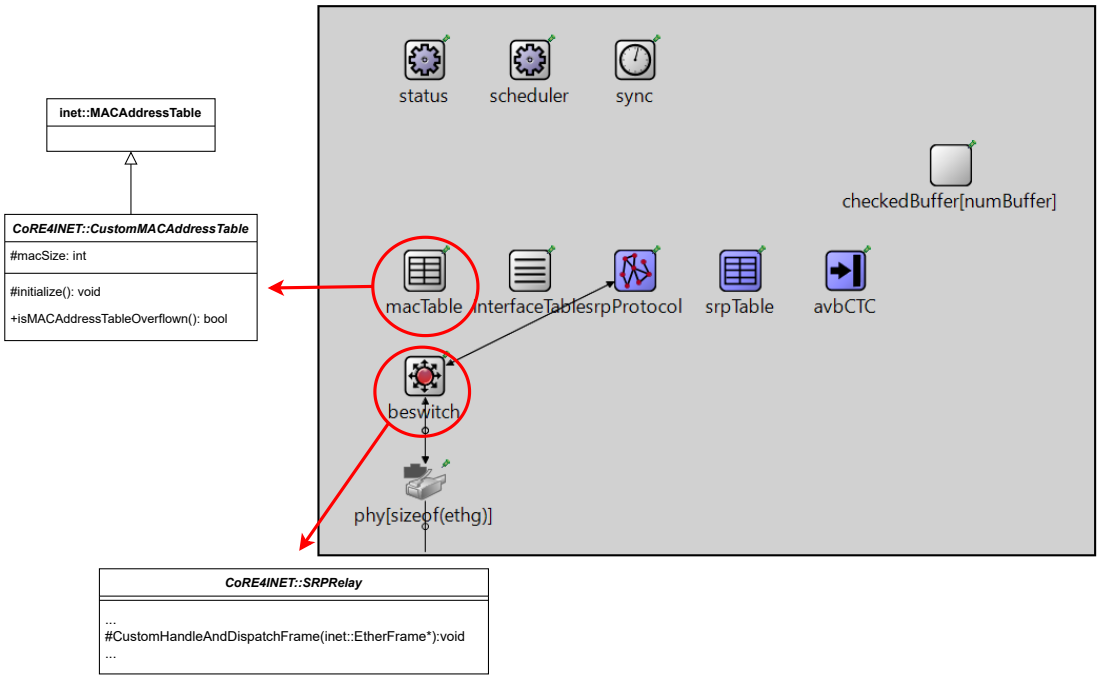

Abbildung 5.15: Modifizierte Module im Switch bei CoRE4INET

angepasste Funktion entscheidet, basierend auf dem Rückgabewert der Funktion des CustomMACAddressTable-Moduls, über das Verhalten des Switches. Sollte die MAC-Adresstabelle überflutet sein, werden die Ethernetframes broadcastet. Andernfalls wird die Adresstabelle aktualisiert, und der Frame wird an die zugehörigen Ports weitergeleitet.

Umsetzung des Angriffs in der Simulation

Um einen Angriff durch MAC-Flooding durchzuführen, wurde zunächst die Kapazität der MAC-Adresstabelle des Switch-Front im RecBarTo-Zonen-Netzwerk auf 500 Einträge festgelegt. Im Netzwerk, wie in Abbildung 5.14 dargestellt, soll der Switch-Front mit einer Vielzahl von Ethernetpaketen bombardiert werden. Der Angreifer sendet alle 200 Mikrosekunden eine IEEE802.1Q-Nachricht mit einer Größe von 16 Bytes an die Unicast-MAC-Adresse der Radar-Komponente, und zwar mit einer Priorität von 6. Hierbei ist entscheidend, dass jedes Paket eine zufällige Quell-MAC-Adresse enthält.
In der Simulation setzt sich das TSNPort-Modul aus verschiedenen Komponenten zusammen. Dieses Modul beinhaltet ein Queueing-Modul, acht Queues, ein Transmission-Algorithmus-Modul, Transmission-Gate und Transmission-Selection. Das Queueing-Modul

ordnet die eingehenden Pakete den entsprechenden Queues zu. Im Rahmen des Frameworks von P. Meyer[7] wurde das Queueing-Modul modifiziert, um zusätzliche Funktionen bereitzustellen. Zur Simulation des Angriffs werden mithilfe dieses modifizierte Modul Pakete erzeugt und leitet es an die passende Queue weiter. Um sowohl die Position des Angreifers als auch den Angriff selbst zu simulieren, werden folgende Änderungen in der INI-Datei der Infotainment-Komponente vorgenommen:

Listing 5.13: Konfigurationscode von MAC-Flooding-Angriff in Infotaiment

```
1  [Config MAC_Flooding]
2  #ersetzen des Moduls durch kompromittierten Queueing
3  **.infotainment.**.queueing.typename = "CorruptIEEE8021QbvQueueingMACFlooding"
4  #sende Indervall und Angriffsdauer
5  **.infotainment.**.queueing.injectionInterval = uniform(200us,200us)
6  **.infotainment.**.queueing.stopTime = 0.093s
7  #Filter, mit der die Nachricht kompromittiert wird
8  **.infotainment.**.queueing.corruptionDestAddress = "4A-4A-4A-4A-4A-4A"
9  **.infotainment.**.queueing.corruptionPayload = intWithUnit(uniform(16Byte,16Byte)
       ↪ )
10 **.infotainment.**.queueing.corruptionPriority = 6
11 **.infotainment.**.queueing.corruptionVid = 0
```

In Zeile 3 des Listing 5.13 3.10 wird der angepasste Queueing-Mechanismus festgelegt, der den Standardmechanismus der Infotainment-Komponente ersetzt und stellt den Angreifer dar. Das Sendeintervall der Nachrichten wird in Zeile 5 festgelegt und in der Zeile 6 die Dauer des Sendevorgangs spezifiziert. Die Zeilen 8 bis 11 des Listing 5.13 enthalten die Eigenschaften der Nachricht, einschließlich der Ziel-MAC-Adresse, der Nachrichtengröße, der Priorität und der zugehörigen VLAN-ID der zu kompromittierenden Nachricht.

Qualitätsicherung des Angriffs

In diesem Abschnitt wird der MAC-Flooding-Angriff validiert. Das beschriebene Szenario wurde über einen Zeitraum von 5 Sekunden simuliert, und die Ergebnisse sind in Grafiken dargestellt.
Um einen erfolgreichen Angriff festzustellen, ist es wichtig, das Ziel des Angriffs zu beachten. Das Ziel von MAC-Flooding besteht darin, den Switch_Front dazu zu bringen, sich wie ein Netzwerk-Hub zu verhalten, was bedeutet, dass die Switch-Front Ethernet-Pakete an alle ihre angeschlossenen Netzwerkknoten als Broadcast weiterleiten soll. Um das Verhalten des Switches zu bestimmen, wird die Anzahl der Pakete berücksichtigt, die pro Sekunde von jedem physischen Port des Switches gesendet wurden.

In der Abbildung 5.16 wird die physikalische Portnummer des Switches auf der X-Achse dargestellt, während die Y-Achse die Anzahl der Pakete pro Sekunde zeigt. Wie aus der Abbildung 5.16 ersichtlich ist, zeigen die blauen Spalten ein deutliches Muster der Paketverteilung, die von jedem physikalischen Port gesendet wird, solange der Switch sich in normalem Betriebszustand befindet.

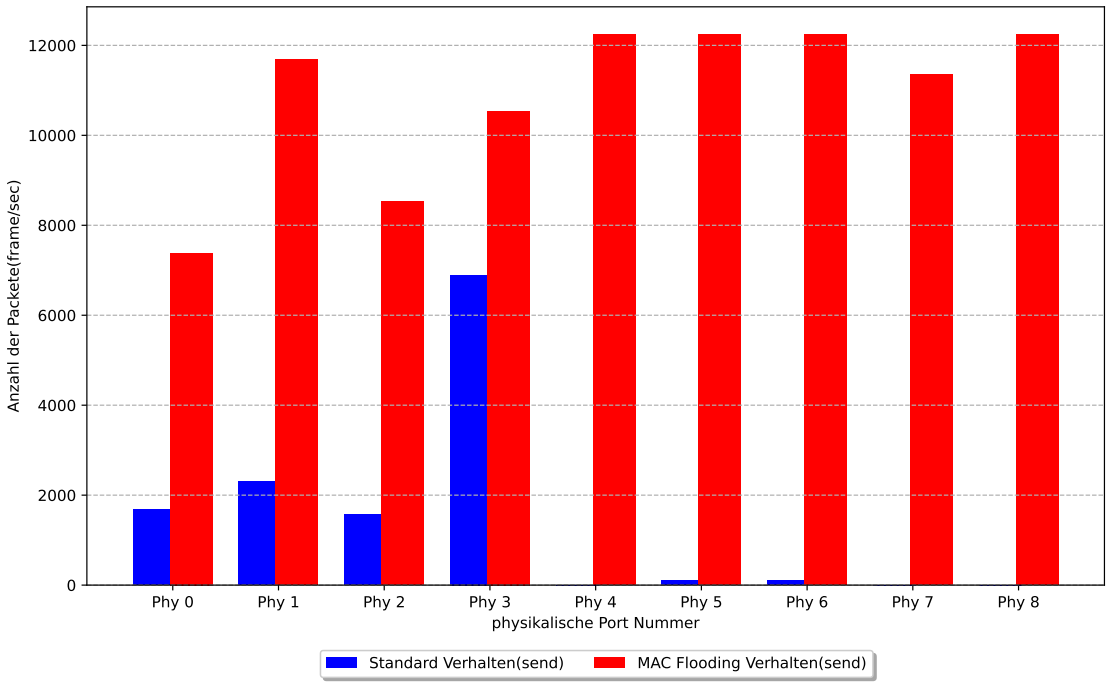

Abbildung 5.16: Anzahl der gesendeten/empfangenen Pakete pro Ports im Switch Front

Nachdem der Angriff stattgefunden hatte, wurde der Switch_Front mit einer enormen Menge an Paketen bombardiert, die jeweils zufällige Quell-MAC-Adressen aufwiesen. Diese Flut an Daten überlastete die MAC-Adresstabelle des Switch_Fronts. Dies bedeutete, dass der Switch_Front jede MAC-Adresse erlernen und in seine Tabelle aufnehmen musste. Infolgedessen begann der Switch_Front sich wie ein Netzwerk-Hub zu verhalten und jedes Paket als Broadcast zu versenden. Die roten Spalten in der Grafik veranschaulichen deutlich den starken Anstieg der Paketzahl, die pro Sekunde über die Ports gesendet wurden. Im Vergleich zu normalem Betriebszustand ist zu erkennen, dass unter normalen Bedingungen keine Pakete über die Ports 4, 7 und 8 gesendet oder weitergeleitet wurden. Nach dem Angriff hingegen wurden über 12.000 Pakete pro Sekunde über diese Ports gesendet.

Betrachtet man die Situation aus einer anderen Perspektive, werden im Hub-Modus der eingehende Datenverkehr eines Ports an alle anderen Ports des Switch_Front weitergeleitet. Theoretisch sollte daher die Anzahl der Pakete, die von einem Port gesendet wurden, gleich der Summe aller Pakete sein, die bei anderen Ports empfangen wurden.
Um dies zu verdeutlichen, bezeichnen wir die empfangenen Pakete als $\mathbf{Pkt_{in}}$ und die gesendeten Pakete als $\mathbf{Pkt_{out}}$. Zur Identifizierung von Abweichungen vom erwarteten Verhalten eines Ports und zur Bewertung des Erfolgs eines Angriffs, muss die Differenz zwischen allen anderen Pkt_{in} und Pkt_{out} an diesem Port gleich null sein. Aus dem oben genannten Satz kann die folgende Gleichung herangezogen werden:

$$\begin{aligned}\text{Differenz der Paketanzahl am Port } P = \\ &\text{Summe aller } \mathrm{Pkt_{in}} \text{ pro Port, exklusive Port } P \\ &- \mathrm{Pkt_{out}} \text{vom Port } P\end{aligned} \tag{5.2}$$

Diese Gleichung 5.2 hilft dabei, spezifische Unregelmäßigkeiten im Verhalten des Switch_-Front unter Angriffsbedingungen zu identifizieren, indem sie die Differenz zwischen der erwarteten und der tatsächlichen Anzahl von Paketen, die von einem Port gesendet und empfangen wurden, berechnet. Zur Veranschaulichung wird eine Tabelle mit Daten aus einer Simulation herangezogen, nachdem der Switch_Front in den Fail-Open-Modus versetzt wurde. Diese Tabelle zeigt, wie die Gleichung 5.2 auf die Daten angewandt wird, und liefert Einblicke in das Verhalten des Switch_Front.

Port	Gegenstelle	empfangen	gesendet (G)	Summe Empfangene Exkl. (EE)	**Differenz (EE - G)**	Pakete in SRP empfangen (SE)	Pakete in SRP gesendet (SG)	Summe SRP Empfangene Exkl. (SEE)	**Differenz ohne SRP (EE - G -SEE+SG)**
Phy0	gw_FrontLeft	24470	36749	37536	787	0	0	787	0
Phy1	gw_Front	3047	58172	58959	787	0	0	787	0
Phy2	gw_FrontRight	18777	42442	43229	787	0	0	787	0
Phy3	Switch_Center	10014	51992	51992	0	0	787	787	0
Phy4	Cam_Front	297	61219	61709	490	297	0	490	0
Phy5	lid_FrontLeft	245	61219	61761	542	245	0	542	0
Phy6	lid_FrontRight	245	61219	61761	542	245	0	542	0
Phy7	radar	4907	56313	57099	786	0	0	787	1
Phy8	infotainment	4	61215	62002	787	0	0	787	0

Tabelle 5.7: Die Differenz der Paketanzahl pro Port von Switch-Front

Die Tabelle 5.7 zeigt die Differenz in der Paketanzahl für jeden Port P des Switch_Front. Die erste und zweite Spalte geben die Portnummer und die an diesem Port angeschlossene Komponente an. Die dritte Spalte zeigt die Anzahl der Pakete, die an diesem Port P angekommen sind. Die vierte Spalte gibt die Anzahl der Pakete an, die von Port P

gesendet bzw. weitergeleitet wurden. Die fünfte Spalte ist die Summe aller empfangenen Pakete pro Port, außer dem Port P. Die sechste Spalte zeigt die Ergebnisse der oben genannten Gleichung 5.2.

Angenommen, in der Tabelle 5.7 zeigt der Port 8, an den Infotainment angeschlossen ist, dass im Laufe der Simulation insgesamt 4 Pakete empfangen und 61215 Pakete an Infotainment gesendet wurden. Die Summe aller von den Ports 0 bis 7 empfangenen Pakete beträgt 62002 Pakete. Unter Anwendung der ursprünglichen Gleichung 5.2 wird deutlich, dass eine Differenz von 787 Paketen besteht. Dies deutet darauf hin, dass 787 Pakete weniger von Port 8 gesendet wurden, als es unter der Annahme der Fall sein sollte, dass der Switch_Front korrekt alle Pakete an alle Ports broadcastet.
Hier führt die Berücksichtigung von Stream Reservation Protocols (SRP) zu einer wichtigen Anpassung in dieser Analyse. SRP ermöglicht eine Reservierung von Bandbreite für bestimmte Datenströme, was bedeutet, dass die Pakete unter SRP unabhängig vom Zustand des Switch_Front nicht broadcastet werden. Daher lautet die erweiterte Gleichung wie folgt:

$$\begin{aligned}\text{Differenz der Paketanzahl am Port } P = \\ &\text{Summe aller } \mathrm{Pkt_{in}} \text{ pro Port, exklusive Port } P \\ &- \mathrm{Pkt_{out}} \text{ vom Port } P \\ &- \text{Summe aller } \mathrm{Pkt_{in}} \text{ druch SRP} \\ &+ \mathrm{Pkt_{out}} \text{ druch SRP vom Port } P\end{aligned} \tag{5.3}$$

Die Anwendung von Gleichung 5.3, unter Einbeziehung der Pakete mit SRP ermöglicht eine differenziertere Analyse des Netzwerkverkehrs. Der Switch_Front hat im Netzwerk von den Komponenten Cam_Front, Lid_FrontLeft und Lid_FrontRight jeweils 297, 245 und 245 Pakete mittels SRP empfangen. Diese Pakete werden nicht aufgrund des aktuellen Zustands des Switch_Front an alle Ports weitergeleitet, sondern gezielt an ihren Zielport. Deshalb muss bei jedem Port P des Switch_Front die Summe der durch SRP empfangenen Pakete abgezogen werden. Am Zielport werden diese Pakete gesendet, weshalb die Anzahl der durch SRP gesendeten Pakete hinzugefügt werden muss. Zum Beispiel wurden vom Port 8 keine Pakete durch SRP gesendet. Wenn man die empfangenen SRP-Pakete von der Differenz am Port 8 abzieht, wird deutlich, dass die Gesamtsumme aller am Port 8 empfangenen und gesendeten Pakete null beträgt. Dies zeigt, dass der Switch_Front effektiv alle anderen Pakete, wie erwartet, broadcastet hat.

Evaluation der Angriffsergebnisse

In diesem Abschnitt werden die exemplarischen Auswirkungen eines MAC-Flooding-Angriffs beschrieben. Ein MAC-Flooding-Angriff ist eine Methode, bei der ein Switch in einem Netzwerk dazu gezwungen wird, sich wie ein Hub-Gerät zu verhalten. Der Angreifer zielt darauf ab, die MAC-Adresstabelle des Switches mit gefälschten Quell-MAC-Adressen zu füllen. Sobald der Tabellenplatz erschöpft ist, geht der Switch in einen öffenen-Modus über und verhält sich wie ein Netzwerk-Hub. Die erste Konsequenz ist, dass die Kapazitäten der MAC-Adresstabelle des Switches erschöpft werden. Infolgedessen kann der Switch-Front keine gültigen Einträge mehr in der Tabelle speichern. Gültige Ethernetpakete werden daher als Broadcasts anstatt als Unicasts gesendet. Dies führt dazu, dass gültige Ethernetpakete über alle Ports des Switches, einschließlich des vom Angreifer genutzten Ports, geflutet werden. Dies erleichtert es dem Angreifer, die benötigten Daten mithilfe eines Packet-Sniffers abzufangen. Zudem verbraucht die große Anzahl unnötiger Broadcasts schnell die Bandbreite und die Ressourcen des gesamten Switches.

Darüber hinaus sollte hier berücksichtigt werden, ob die Frames von einem VLAN in ein anderes übertragen werden, nachdem es überfüllt ist. E. Vyncke[16] verdeutlicht in seinem Buch, sobald die MAC-Tabelle auf dem Switch voll wird, wird der Verkehr ohne FIB-Eintrag über jeden Port in lokalen VLAN geflutet. Dadurch wird ein VLAN auf einem Switch praktisch zu einem Hub. Genauer gesagt, die MAC-Adresstabelle ist eine Tabelle, die innerhalb eines begrenzten Speicherbereichs verwaltet wird. Sobald diese Tabelle ihre Kapazitätsgrenze erreicht, ist sie für jedes VLAN voll. Es spielt keine Rolle, wie viele Einträge von z.B. VLAN 1 und von VLAN 2 stammen.
Es gibt keine Zuordnung von so viel Platz für VLAN 1 und so viel Platz für VLAN 2. Wenn VLAN 1 den verfügbaren Platz aufgebraucht hat, wird VLAN 2 dadurch beeinträchtigt, dass keine Einträge in die Tabelle aufgenommen werden können.Dies führt zur Überflutung der Datenverkehr in VLAN 2 sowie in 1. Im Endeffekt führt ein MAC-Flooding-Angriff nicht dazu, dass Frames von einem VLAN in ein anderes verschoben werden. Die Grenze von VLAN bleibt unverändert.

Es sollte auch nicht unerwähnt bleiben, dass sich ein Switch je nach Konfiguration unterschiedlich verhalten kann, wenn es zu einem Überlauf der FIB-Tabelle kommt[3]:

[3] Diese Begriffe werden in Sicherheitskontexten verwendet, um zu beschreiben, wie sich ein System beim Auftreten eines Fehlers verhält.

- **Fail Open:** In diesem Modus lässt der Switch den Datenverkehr auch bei Überlauf weiterfließen. Er leitet den gesamten Datenverkehr an alle Ports weiter. Dies führt dazu, dass jedes Gerät im Netzwerk den gesamten Datenverkehr einsehen kann, wodurch sich der Switch praktisch wie ein Hub verhält.
- **Fail Close:** Hier unterbricht der Switch den Datenverkehr. Er behält die vorhandenen MAC-Adressen in seiner FIB-Tabelle bei und weigert sich jedoch, neue MAC-Adresse hinzuzufügen.
- **Ringspeicher-Ansatz:** Dies ist ein Mechanismus, bei dem der Switch die MAC-Adressen löcht, die am längsten nicht verwendet wurden.

Theoretisch könnte eine FIB-Tabelle wie ein Ringspeicher aufgebaut sein. Ein Ringspeicher ist ein Ringspeicher, bei dem neue Daten die ältesten Daten überschreiben. Aus diesem Grund ist es bei diesem Angriff wichtig zu wissen, dass der Angreifer eine gewisse Zeitlang die Datenverkehr abfangen kann, wenn die FIB-Tabelle voll ist.
In jedem Switch gibt es für FIB-Einträge eine sogenannte Aging-Time, die als Alterungszeit bezeichnet wird. Diese sorgt dafür, dass nicht mehr verwendete MAC-Adressen nach einer bestimmten Zeit automatisch aus der FIB-Tabelle entfernt werden. Die Switches im RecBarToZone-Netzwerk haben standardmäßig eine Aging-Time von 1000 Sekunden. Diese lange Zeit gibt dem Angreifer die Chance, den Switch-Front so lange zu bombardieren, bis die FIB-Tabelle aller Switches voll ist. Der Angreifer kann dann alle sensiblen Daten im gesamten Netzwerk abfangen, solange die Switches alle Datenverkehr broadcasten. Darüber hinaus führt MAC-Flooding zur Verschlechterung der Netzwerkleistung. Die Überflutung des Datenverkehrs kann zu Verzögerungen oder zum Datenverlust für den Netzwerkverkehr führen. Zusammenfassend wurde es festgestellt, dass MAC-Flooding den Betrieb eines Netzwerks beeinträchtigt, die Sicherheit des Netzwerks gefährdet und einen Zugriff oder Abfangen von Daten zur Folge hat.

5.4 EtherOops

In den folgenden Unterkapiteln werden zunächst der Angriff und sein Aufbau Anhand eines Szenarios beschrieben. Anschließend werden die Konfigurationen erläutert, die zur Realisierung des Angriffs notwendig sind, sowie die Umsetzung des Angriffs im Simulationsnetzwerk dargelegt. Zum Abschluss wird der simulierte Angriff validiert und werden seine exemplarischen Auswirkungen auf das Netzwerk erörtert.

Konzept und Aufbau des Angriffs

Etheroops ist eine Art von Cyberangriff, der Schwachstellen in fehlerhaften Ethernet-Kabeln ausnutzt. Dieser Angriff ermöglicht es einem Angreifer, Netzwerksicherheitsgeräte wie Firewalls und NATs zu umgehen. Unter bestimmten Voraussetzungen kann Etheroops dazu verwendet werden, ein bösartiges Ethernet-Paket in scheinbar harmlosen Paketen zu verstecken und zu versenden.

In diesem Abschnitt werden diese Angriffsmethode und ihre spezifischen Auswirkungen auf Fahrzeugnetzwerk anhand eines Szenarios ausführlich beschrieben. In diesem Szenario gehen wir davon aus, dass der Angreifer versucht, ein schädliches Ethernet-Paket in anderem Paket zu verstecken und in das Fahrzeugnetz zu senden.

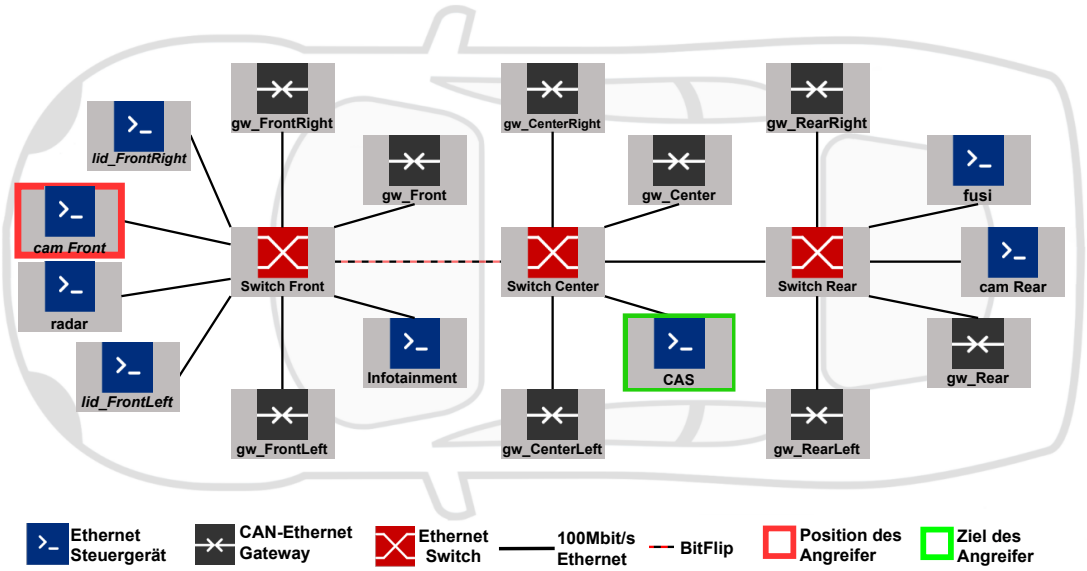

Abbildung 5.17: Paket-in-Paket Angriff in RecBarToZone Netzwerk

Die Abbildung 5.17 stellt das Fahrzeugnetzwerk dar. Der Angreifer positioniert sich per Software auf der vorderen Kamerakomponente (Cam_Front). Sein Ziel ist es, ein schädliche Ethernet-Paket zur CAS-Komponente zu senden. Dies geschieht, wenn ein Bitfehler in den Kabeln zwischen dem vorderen Switch(Switch-Front) und dem Mittleren Switch(Switch-Center) auftritt, woraufhin das innere Paket im Switch-Center als harmloses Paket angesehen und an die CAS-Komponente weitergeleitet wird.
Der Angreifer sendet zahlreiche verschachtelte Pakete an die hintere Kamerakomponente (Cam_Rear). Diese Pakete enthalten in ihrer Nutzlast ein weiteres Paket, das die MAC-Adresse der CAS-Komponente aufweist.

Vorbreitung des Angriffs in der Simulation

Beim Etheroops-Angriff besteht die Idee darin, dass ein vom Angreifer erstelltes Ethernet-Paket in ein anderes Paket eingebettet wird. Um diesen Packet-in-Packet-Angriff für das vorgegebene Szenario in der Simulation umzusetzen, sind folgende Schritte erforderlich. Zunächst muss ein Paketgenerator implementiert werden, der ein Ethernetframe in seine Nutzlast (Payload) einbettet. Anschließend ist es notwendig, ein Modul zu entwickeln, das Bitfehler in Ethernet-Kabeln simuliert.

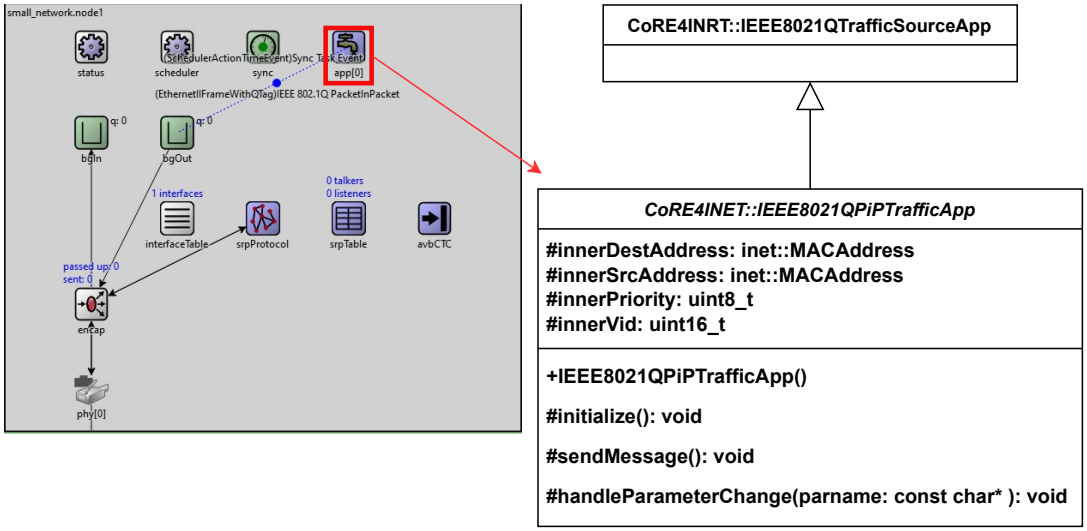

Abbildung 5.18: Die Paket-in-Paket-Anwendung eines Moduls im Rahmen von CoRE4INET

Zunächst wurde in CoRE4INET ein Modul implementiert, das einen Ethernetframe erzeugt, welcher einen anderen Frame in ihrer Nutzlast (Payload) einfügt. Die Abbildung 5.18

illustriert die Datenverkehr-Anwendung des Packet-in-Packet-Frames (PiP-Frame). Der Modul von PiP-Frame baut auf einem anderen Modul des CoRE4INET-Frameworks auf, das unter dem Namen IEEE8021QTrafficApp bekannt ist. Die Ethernetframes werden gemäß dem IEEE802.1Q-Standard generiert. Das PiP-Frame-Modul bietet zusätzliche Parameter für einen Ethernetframe, die sich auf die Eigenschaften des inneren Frames beziehen. In der Funktion $sendMessage()$ wird zuerst der äußere Frame erzeugt. Anschließend wird ein weiterer Frame mit eigenen Eigenschaften wie Quell- und Ziel-MAC-Adresse, Priorität und VLAN-ID in die Nutzlast des äußeren Frames gekapselt und versendet.

Im nächsten Schritt wurde in CoRE4INET ein Modul namens IEEE8021QPacketBitFlip implementiert. Dieses Modul simuliert das BitFlip-Verhalten und wurde in das Modul des physikalischen Ports integriert. Der Grund für die Integration ist, dass alle eingehenden Pakete von diesem Modul durchgehen müssen. Das TSNPort-Modul stellt einen physischen Anschluss für IEEE802.1Q-kompatible Ethernet-Geräte wie Switches oder Hosts in der Simulation dar.

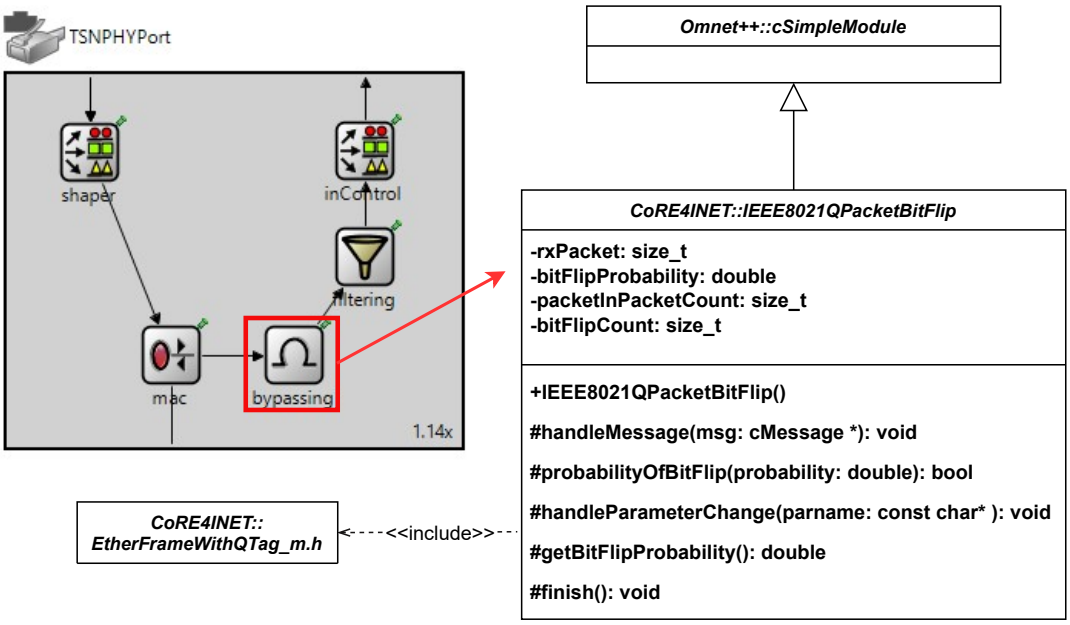

Abbildung 5.19: Die BitFlip-Anwendung eines Moduls im Rahmen von CoRE4INET

Wie in der Abbildung 5.19 zu sehen ist, umfasst das Modul vier Parameter, die wie folgt beschrieben sind:

- **rxPacket:** Diese Variable zählt die Anzahl der auf dem Port eingegangenen Pakete.

- **bitFlipProbability:** Diese Variable definiert die Wahrscheinlichkeit eines Bitfehlers. Der Wert liegt zwischen 0 und 1 und gibt den Grad der Wahrscheinlichkeit von BitFlip auf dem Kabel an. Je höher der Wert, desto größer ist die Wahrscheinlichkeit eines BitFlip.
- **bitFlipCount:** Dieser Zähler wird erhöht, wenn ein Bitfehler auftritt.
- **PacketInPacketCount:** PacketInPacketCount: Dieser Zähler wird speziell dann erhöht, wenn ein Bitfehler ausschließlich bei einem Packet-in-Packet-Frame auftritt.

Die getBitFlipProbability()-Funktion setzt den eingegebenen Wahrscheinlichkeitswert. Die Funktion probabilityOfBitFlip() liefert als Rückgabewert True, wenn die generierte Zufallsvariable kleiner oder gleich dem eingestellten Wahrscheinlichkeitswert ist. Ein Rückgabewert von True bedeutet, dass ein BitFlip auftritt.

Die grundlegende Logik der handleMessage()-Funktion lässt sich folgendermaßen im Pseudocode beschreiben:

Algorithm 1 Pseudocode für die handleMessage-Funktion

```
ERSTELLE qframe als ein dekapsuliertes Message
if probabilityOfBitFlip(getBitFlipProbability()) = True then
    if qframe->getName() = "IEEE 802.1Q PacketInPacket" then
        packetInPacketCount ← packetInPacketCount + 1
        // Entkapsle der innere Frame und sende er weiter
        ERSTELLE innerframe als ein dekapsulierter Frame von qframe
        SENDE innerframe
    else
        LÖSCHE qframe
else
    SENDE Message
```

Zur Erfassung von Statistiken über die Nachrichten, die nach BitFlip an CAS-Komponente angekommen sind, wird in der INI-Datei der CAS-Komponente eine Sinkapp eingerichtet. Diese ermöglicht es, das Nachrichtenaufkommen später zu analysieren. Die folgende Konfiguration wurde in der INI-Datei hinzugefügt:

Listing 5.14: Die Erstellung einer Sinkapp für empfangenen Paket-in-Paket Nachrichten von vordere Kamera

```
1 # Sink app for message "msg_camFront_PiP":
2 **.cas.numApps = 3
3 **.cas.app[2].typename = "IEEE8021QTrafficSinkApp"
4 **.cas.app[2].srcAddress = "AA-00-00-00-00-11"
5 **.cas.app[2].replyFirst = false
```

In der ersten Zeile von Listing 5.14 wird eine neue Anwendung in der CAS-Komponente instanziiert. Diese Zahl zeigt die Anzahl der Anwendungen, die auf der CAS-Komponente zu instanziiert sind. In der dritten Zeile wird der Typ der Anwendung als $IEEE8021QTraffic-SinkApp$ für SinkApp festgelegt, um den Verkehr mit Q-Prioritäten zu erfassen. Die vierten Zeile legt fest, von welcher Quelladresse die empfangenen Nachrichten stammen sollen. Die fünften Zeile dient dazu, den ersten Frame zur Aktualisierung der Tabellen in den Switches zu wiederholen.

Umsetzung des Angriffs in der Simulation

Für die Durchführung des Packet-in-Packet-Angriffs wurden einige Änderungen an der vorderen Kamera vorgenommen, die durch den Konfigurationscode erläutert werden. Im Netzwerk (siehe Abbildung 5.17) platziert der Angreifer sich per Software in der vorderen Kamera(Cam_Front).

Zunächst wird die Wahrscheinlichkeit eines BitFlips für jedes Ethernet-Kabel im Netzwerk auf 0,0653 festgelegt. Diese Einstellung wird in der INI-Datei jeder Netzwerkkomponente, ausgenommen CAN-Komponenten und Gateways, konfiguriert. Die Berechnung dieses Wertes und der Grund für die Auswahl dieser Zahl werden im weiterem Abschnitten erläuert.
Um mögliche Bitfehler für Packet-in-Packet-Frame(PiP-Frame) auf dem Ethernet-Kabel zwischen Switch_Front und Switch_Center zu haben, muss der Angreifer eine Vielzahl vom PiP-Frame versenden. Aus diesem Grund sendet er aus Cam_Front alle 250us einen PiP-Frame zur hinteren Kamera (Cam_Rear). Der äußere Frame hat die Unicast-MAC-Adresse von Cam_Rear mit einer Priorität von 6, während der innere Frame die Unicast-MAC-Adresse der CAS-Komponente sowie eine eigene Quell-MAC-Adresse besitzt und eine Größe von 1500 Bytes mit einer Priorität von 5 hat. So erfolgt die Weiterleitung des inneren Frames zur CAS-Komponente durch Switch_Center nach dem BitFlip. Um sowohl die Position des Angreifers als auch den Angriff selbst zu simulieren, werden die folgenden Konfigurationen in der INI-Datei von Cam_Front umgesetzt:

Listing 5.15: Konfigurationscode für Paket-in-Paket-Angriff in vordere Kamera

```
1  [Config PacketInPacket]
2  **.cam_Front.numApps = 3
3  #Type der Nachrichtprotokoll
4  **.cam_Front.app[2].typename = "IEEE8021QPiPTrafficApp"
5  #Eigenschaften des aeusseren Frames
6  **.cam_Front.app[2].destAddress = "EE-43-02-AD-85-D5"
7  **.cam_Front.app[2].priority = 6
8  **.cam_Front.app[2].vid = 0
9  #Eigenschaften des inneren Frames
10 **.cam_Front.app[2].innerDestAddress = "CA-CA-CA-CA-CA-CA" #CAS
11 **.cam_Front.app[2].innerSrcAddress = "AA-00-00-00-00-11" #Cam_Front
12 **.cam_Front.app[2].payload = intWithUnit(uniform(1478Byte, 1478Byte))
13 **.cam_Front.app[2].innerVid = 0
14 **.cam_Front.app[2].innerPriority = 5
15 #Angriff Eigenschaften
16 **.cam_Front.app[2].sendInterval = uniform(250us, 250us)
```

In der zweiten Zeile von Listing 5.15 wird eine neue Anwendung in der Kamerakomponente instanziiert. Die Anzahl der zu instanziierenden Anwendungen auf der Kamerakomponente wird durch diese Zahl angezeigt. In der vierten Zeile wird das Protokoll IEEE802.1Q angegeben, das den Nachrichtentyp für den Angriff festlegt. In den Zeilen 6 bis 8 werden die Eigenschaften des äußeren Frames festgelegt, einschließlich Ziel-MAC-Adresse, Priorität und zugehöriger VLAN-ID. In Zeile 10 wird die Quell-MAC-Adresse des inneren Frames angegeben, die die Adresse des Angreifers sein könnte. In Zeile 11 wird die Ziel-MAC-Adresse des Zielkomponenten des Angreifers spezifiziert. In den Zeilen 12 bis 14 werden die Eigenschaften des inneren Frames, wie Nutzlastgröße, Priorität und die entsprechende VLAN-ID definiert. In Zeile 16 wird das Intervall festgelegt, in dem ein Frame gesendet wird.

Qualitätsicherung des Angriffs

In diesem Abschnitt wird der Packet-in-Packet-Angriff (PiP-Angriff) validiert. Das vorgestellte Szenario wurde über einen Zeitraum von zehn Sekunden simuliert und die Ergebnisse sind in Grafik und Tabelle dargestellt.

Um einen erfolgreichen Angriff festzustellen, ist es wichtig, das Ziel des Angriffs zu berücksichtigen. Bei einem PiP-Angriff besteht das Ziel darin, dass der Angreifer eine Vielzahl von Packet-in-Packet-Frames (PiP-Frames) sendet und auf das Auftreten eines Bitfehlers wartet, um durch der innere Frame CAS-Komponente zu kompromittieren.

Zunächst wird die Anzahl der gesendeten und empfangenen Frames anhand von Statistiken betrachtet. Die Tabelle 5.8 zeigt die Anzahl der von der Frontkamera (Cam_Front) gesendeten sowie die Anzahl der empfangenen Frames der CAS-Komponente nach einem Bitfehler. Während der Simulation wurden von Cam_Front insgesamt 40.000 Frames gesendet. Von diesen sind 30.489 Frames normal an ihr Ziel(Cam_Rear) angekommen. Nach dem Auftreten von Bitfehlern wurden 4.584 Frames von der CAS-Komponente empfangen. Diese empfangenen Frames sind die inneren Frames der Packet-in-Packet-Frames.

	Cam_Front	CAS	Cam_Rear
Anzahl der gesendeten Pakete	40000	-	-
Anzahl der empfangenen Pakete	-	4584	30489

Tabelle 5.8: Anzahl der gesendeten und empfangenen Paket-in-Paket

Der Unterschied in der Anzahl der von beiden Komponenten empfangenen Frames lässt sich durch die Orte der Bitfehler erklären. Die Darstellung in der Abbildung zeigt die Route sowie Anzahl der Packet-in-Packet-Frames im Netz und zeigt der Kabeln, an denen Bitfehler aufgetreten sind. Da das Ziel der inneren Frames die CAS-Komponente ist, führt jeder Bitfehler, der auf dem Segment zwischen Cam_Front und Switch_Center auftritt, zum erfolgreichen Packet-in-Packet-Angriff. Sollten Bitfehler auf anderen Kabelsegmenten im Netzwerk auftreten, werden die betroffenen Frames in der Regel verworfen, was die Integrität weiterer Netzwerkkomponenten schützt.

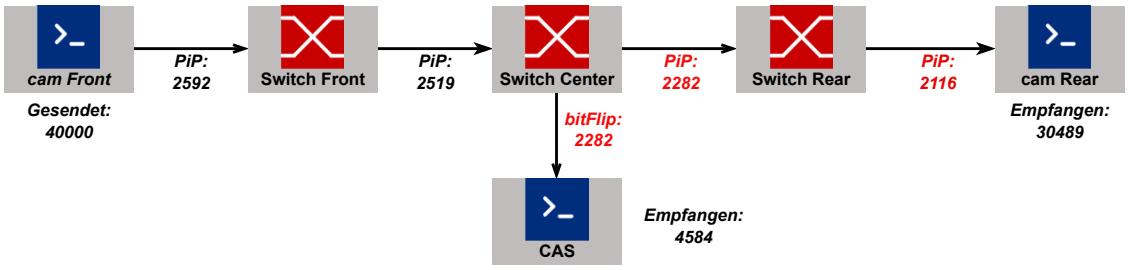

Abbildung 5.20: Die Route der Paket-in-Paket-Frames im Netzwerk

Der Grund für den Unterschied zwischen den von beiden Komponenten empfangenen Frames liegt im Ort des Bitfehlers. Die Abbildung illustriert die Route der Packet-in-Packet-Frames. Wie in der Abbildung ersichtlich, tritt an verschiedenen Stellen ein Bitfehler auf. Da das Ziel der inneren Frames die CAS-Komponente ist, führt jeder Bitfehler auf dem PiP-Frame zwischen Cam_Front und Switch_Center zum Packet-in-Packet-Angriff. Die Summe der erfolgreich durchgeführten PiP-Angriff zwischen Cam_Front und Switch_-

Center betrifft sich auf 5111 Frames. Wenn ein Bitfehler auf anderen Kabeln im Netzwerk auftritt, werden die betroffenen Frames verworfen. Die Anzahl der Bitfehler zwischen Switch_Center und der CAS-Komponente, die auf alle Frames einwirken, beträgt 2.282. Aus diesem Grund sind lediglich 4.584 Frames erfolgreich bei der CAS-Komponente angekommen.

Evaluation der Angriffsergebnisse

In diesem Abschnitt werden die exemplarischen Auswirkungen eines Packet-in-Packet-Angriffs(PiP-Angriff) beschrieben. Bei einem PiP-Angriff handelt es sich um das Einfügen eines modifizierten Ethernetframe innerhalb eines harmlosen Frames. Durch die Ausnutung eine Schwachstelle in Ethernet-Kabeln Der Packet-in-Packet-Angriff nutz eine Schwachstelle in Ethernet-Kabeln aus, sodass ein Frame während der Übertragung geändert werden kann und dennoch als ein gültiger Frame verarbeitet wird.

Der Angreifer sendet eine Vielzahl modifizierter Frames ins Netzwerk. Tritt ein Bitfehler auf, kann der innere Frame des modifizierten Frames in Netzwerkkomponenten wie dem Switch_Center als neuer, eigenständiger Frame reinterpretiert werden. Die Herausforderung dieses Angriffs liegt in der Häufigkeit von Bitfehlern. Die Wahrscheinlichkeit eines Bitfehlers (Bit Error Rate , BER) beträgt bei einem 100 Mbit\s Ethernet-Kabel gemäß den IEEE-Normen $\mathbf{10^{-9}}$. Dies bedeutet, dass statistisch gesehen ein Bitfehler pro neun Milliarden übertragener Bits auftritt.
Für einen erfolgreichen Angriff muss der Bitfehler an einem spezifischen Byte im Kabel auftreten. Auf der Layer 2 wird das erste Byte nach den Präambel-Bytes als Beginn des Frames (SFD) interpretiert. Wenn die SFD-Byte durch Bitfehler andere Symbol bildet, wird als Preämble-Bytes integriert. So kann der SFD-Byte des inneren Frame für Beginn des Frames interpretiert werden. Aus diesem Grund muss der Bitfehler im SFD-Byte auftreten. Die Wahrscheinlichkeit eines Bitfehlers im SFD-Byte lässt sich mit folgender Gleichung 5.4 berechnen.

aus dieser Berechnung ergibt sich, dass die Wahrscheinlichkeit von $6,535 \times 10^{-4}$ für einen Bitfehler genau in dem SFD-Byte ist. Um die Gesamtwahrscheinlichkeit eines Bitfehlers im Kabel (sieh Gleichung 5.5) zu berechnen, multipliziert man die Bitfehlerwahrscheinlichkeit des SFD mit der Basis-BER, die in diesem Fall 10^{-9} beträgt. Das resultierende Produkt zeigt die Wahrscheinlichkeit eines Bitfehlers auf dem Kabel von $6,535 \times 10^{-13}$. Diese Wahrscheinlichkeit kann durch Faktoren wie physische Schäden,

$$
\begin{aligned}
P_{\text{BitFlip in SFD}} &= \left(\frac{SFD_{\text{byte}} \cdot 8_{\text{bit}}}{\text{Framegröße}_{\text{byte}} \cdot 8_{\text{bit}}} \right) \\
&= \frac{1_{\text{Byte}} \cdot 8_{\text{bit}}}{1530_{\text{Byte}} \cdot 8_{\text{bit}}} = 6{,}5359 \cdot 10^{-4}
\end{aligned}
\tag{5.4}
$$

Abbildung 5.21: Berechnung der Wahrscheinlichkeit eines Bitflips im SFD-Byte

elektromagnetische Interferenzen (EMI) und Nebensprechen(Crosstalk) erhöht werden. Solche Faktoren wurden für die Simulation nicht berücksichtigt. Insbesondere bei niedrigen Werten könnte die Simulation stundenlang dauern. Daher wurde die Wahrscheinlichkeit, dass alle Netzwerkkabel in der Simulation 0,0653 festgelegt.

$$
\begin{aligned}
P_{\text{BitFlip}} &= (P_{\text{BitFlip in SFD}}) \cdot \text{BER-Basis} \\
&= (6{,}5359 \cdot 10^{-4}) \cdot 10^{-9} = 6{,}5359 \cdot 10^{-13}
\end{aligned}
\tag{5.5}
$$

Abbildung 5.22: Berechnung der Wahrscheinlichkeit eines Bitflips

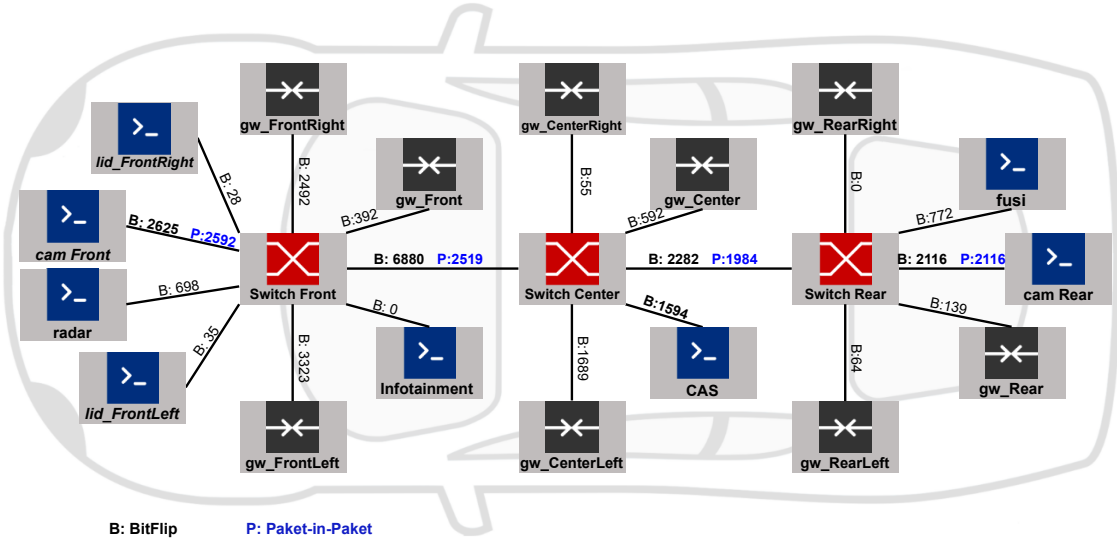

Abbildung 5.23: Die Route der Paket-in-Paket-Frames im Netzwerk

Die Abbildung 5.23 verdeutlicht die Anzahl der Bitfehler in den Netzwerkkabeln sowie die Anzahl der Packet-in-Packet-Frames. Es ist erkennbar, dass Bitfehler in bestimmten Kabeln des Netzwerks häufiger auftreten. Für das beschriebene Szenario ist es für den Angreifer entscheidend, dass der Bitfehler spezifisch auf dem Kabel zwischen Switch_Front und Switch_Center auftritt, um den gewünschten Effekt des Angriffs zu erzielen.

In der durchgeführten Simulation jedoch führen auch Bitfehler zwischen Cam_Front und Switch_Front zu einem Packet-in-Packet-Angriff. In solchen Fällen wird der innere Frame durch den Switch_Front als ein neuer Frame interpretiert und an das Netzwerk broadcastet. Allerdings besteht die Möglichkeit, dass dieser innere Frame verworfen wird, falls weitere Bitfehler in den nachfolgenden Kabeln auftreten. Trotz der hohen Anzahl von PiP-Frames, die durch die Kabeln gesendet werden, sind nur 4584 innere Frames die CAS-Komponente(siehe Tabelle 5.8).

6 Fazit und Ausblick

Die vorliegende Arbeit endet in diesem Abschnitt mit einem Fazit und einer Ausblick. Zuerst wird die Arbeit kurz zusammengefasst. Abschließend werden im Ausblick potenzielle zukünftige Arbeiten vorgestellt.

6.1 Fazit

Die modernen Fahrzeuge sind heutzutage mit verschiedenen elektronischen Systemen ausgestattet. Die traditionellen Bussysteme sind oft nicht mehr in der Lage, die notwendigen Anforderungen an Bandbreite und Latenzzeit zu erfüllen. Ethernet bietet daher eine sichere Netzwerklösung. Der Einsatz von Echtzeit-Ethernet verbessert die Kommunikation zwischen den Komponenten. Diese Weiterentwicklung der Fahrzeugsysteme hat zur Folge, dass Echtzeit-Ethernet durch Ethernet-Angriffe gefährdet werden kann.
In dieser Arbeit wurden die möglichen Angriffe auf Fahrzeugnetzwerke vorgestellt und in der Simulationsumgebung von OMNet++ implementiert. Jeder Angriff wurde anhand eines Szenarios in einem simulierten Fahrzeugnetzwerk umgesetzt. Die Angriffe wurden durch ihre Charakterisierung validiert, um zu überprüfen, ob sie ihre Anforderungen erfüllen können. Die exemplarischen Auswirkungen jedes Angriffs wurden evaluiert.
Der DoS-Angriff hat Auswirkungen wie erhöhte Latenzen und Netzwerkauslastung. Die Latenzen für die Nachrichten mit gleicher oder niedrigerer Priorität nahmen zu. Dies zeigte sich vor allem bei Nachrichten von der vorderen Kamera zur Sensorfusion, deren durchschnittliche und maximale Latenzen durch diesen Angriff gestiegen sind. Der Man-in-the-Middle-Angriff kann in verschiedenen Angriffsformen im Fahrzeugnetzwerk durchgeführt werden.Diese Angriffe haben die Fähigkeit, Daten abzufangen oder zu manipulieren oder zu löschen. Dies kann zu Schäden führen und die Kommunikation zwischen kritischen Fahrzeugkomponenten stören. Ein MAC-Flood-Angriff zielt darauf ab, die MAC-Adresstabelle eines Switches mit gefälschten MAC-Adressen zu füllen. Sobald der Speicherplatz in der Tabelle erschöpft ist, geht der Switch in einen offenen-Modus über

und verhält sich wie ein Netzwerk-Hub. Zu diesem Zeitpunkt wird der Datenverkehr nicht in VLANs aufgeteilt sondern werden broadcastet und der Angreifer kann den gesamten Datenverkehr des Switches abfangen. Der Kern des Etheroops-Angriffs besteht darin, dass der Angreifer zahlreiche Paket-in-Paket-Frames sendet und auf das Auftreten eines Bitfehlers wartet, um die Netzwerksicherheit zu kompromittieren.

6.2 Ausblick

Die vorliegende Arbeit stellt mögliche Bedrohungen im Bereich der Sicherheit von Echtzeit-Ethernet-Netzwerken in Fahrzeugen vor. Als Fortsetzung dieser Arbeit können die vorgestellten Angriffe weiterentwickelt und angepasst werden, um die Sicherheit und Zuverlässigkeit des Netzwerks zu erhöhen. Ein möglicher Ansatz ist die detaillierte Analyse und Implementierung erweiterter Angriffstechniken. Weiterhin können zusätzliche Angriffsszenarien untersucht und simuliert werden. Dazu gehören unter anderem Angriffe auf höhere Protokollebenen. Die Untersuchung der Auswirkungen kombinierter Angriffe, bei denen mehrere Angriffstechniken gleichzeitig eingesetzt werden, kann ebenfalls wertvolle Erkenntnisse liefern. Ein weiterer Aspekt könnte die Evaluation von Maßnahmen zur Erkennung und Abwehr solcher Angriffe in Fahrzeugnetzwerken sein.

Literaturverzeichnis

[1] Ben Seri, Gregory V. ; Yusepovsky, Yevgeny: *EtherOops-Bypassing Firewalls and NATs By Exploiting Packet-in-Packet Attacks in Ethernet*. 2020. – URL https://info.armis.com/rs/645-PDC-047/images/Armis-EtherOops-TWP-20200805-1.pdf

[2] CoRE-Arbeitsgruppe: *Communication over Real-time Ethernet*. – URL https://core-researchgroup.de/. – Zugriffsdatum: 2024-04-20

[3] CoRE-Arbeitsgruppe: *CoRE Simulation Models for Real-time Networks.*. – URL https://core.informatik.haw-hamburg.de/projects/simulation/. – Zugriffsdatum: 2024-04-20

[4] crowdstrike: *denial-of-service-dos-attacks*. – URL https://www.crowdstrike.de/cybersecurity-101/denial-of-service-dos-attacks/

[5] Bundesamt für Sicherheit in der Informationstechnik(BSI): *DoS- und DDoS-Attacken*. – URL https://www.bsi.bund.de/dok/6599510

[6] Jeong, Seonghoon ; Kim, Huy K. ; Han, Mee L. ; Kwak, Byung I.: AERO: Automotive Ethernet Real-Time Observer for Anomaly Detection in In-Vehicle Networks. In: *IEEE Transactions on Industrial Informatics* 20 (2024), Nr. 3, S. 4651–4662

[7] Meyer, Philipp ; Häckel, Timo ; Reider, Sandra ; Korf, Franz ; Schmidt, Thomas C.: Network Anomaly Detection in Cars: A Case for Time-Sensitive Stream Filtering and Policing. (2021), Dezember

[8] security-insider.de: *Man-in-the-Middle-Angriff*. – URL https://www.security-insider.de/was-ist-ein-man-in-the-middle-angriff-a-775391/

[9] OMNeT++ Simulation Manual. – Forschungsbericht. Zugriffsdatum 20.04.2024

[10] OpenSim Ltd.: *INET Framework*. – URL https://inet.omnetpp.org/

[11] Oriyano, Sean-Philip: *Hacker Techniques, Tools, and Incident Handling* -. Burlington, MA : Jones & Bartlett Publishers, 2014. – ISBN 978-1-284-03171-3

[12] Reider, Sandra ; Meyer, Philipp ; Häckel, Timo ; Korf, Franz ; Schmidt, Thomas C.: Integration realer Angriffe in simulierte Echtzeit- Ethernet-Netzwerke. In: *Echtzeit 2020*. Wiesbaden : Springer Vieweg, Januar 2021 (Informatik aktuell), S. 51–60. – ISBN 978-3-658-32818-4

[13] Steinbach, Till: Ethernet-basierte Netzwerkarchitekturen für künftige Echtzeitsysteme im Automobil. In: *ATZ - Automobiltechnische Zeitschrift* 121 (2019), July, Nr. 7, S. 86–91. – URL https://doi.org/10.1007/s35148-019-0071-6

[14] Trotter, Guy M.: *Terminology for Forwarding Information Base (FIB) based Router Performance*. RFC 3222. Dezember 2001. – URL https://www.rfc-editor.org/info/rfc3222

[15] IEEE 802.1 TSN Task Group: *Time-Sensitive Networking (TSN) Task Group*. – URL https://1.ieee802.org/tsn/

[16] Vyncke, Eric ; Paggen, Christopher: *LAN Switch Security - What Hackers Know About Your Switches*. Amsterdam : Pearson Education, 2007. – ISBN 978-0-134-43360-8

[17] Ziegler, Paul S.: *Netzwerkangriffe von innen* -. Köln : O'Reilly Germany, 2008. – ISBN 978-3-897-21778-2

A Anhang

Glossar

DoS-Angriff Denial of Service Attack.

INI-Datei Eine INI-Datei ist eine Konfigurationsdatei, die von Programm zur Initialisierung von Programmeinstellungen verwendet wird. Sie enthält Abschnitte für Einstellungen und Präferenzen, wobei jeder Abschnitt einen oder mehrere Namens- und Wertparameter enthält.

Jitter die Differenz zwischen maximaler und minimaler Latenzzeit.

MitM-Angriff Man-in-the-Middle Attack.

Sinkapp Traffic Sink-Anwendung für die statistische Erfassung des Verkehrs mit Q-Prioritäten.

TSN Time Sensitive Networking.

VLAN-ID Ein Virtual Local Area Network (VLAN) stellt ein logisches Teilnetz innerhalb eines Switches oder eines gesamten physischen Netzwerks dar. Es dient dazu, physische Netzwerke in Teilnetze zu unterteilen. Dabei werden den einzelnen Ethernet-Pakete Tags hinzugefügt, die die VLAN-ID enthalten, um anzuzeigen, zu welchem VLAN das Frame gehört. Dies gewährleistet, dass Datenpakete nicht in ein anderes VLAN weitergeleitet werden..

XML-Datei Eine XML-Datei ist Extensible Markup Language. Sie enthält einen formatierten Datensatz, der für die Verarbeitung durch ein Softwareprogramm bestimmt ist. XML-Dateien kann man sich als textbasierte Datenbanken vorstellen.

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original