HAW HAMBURG

# Bachelorarbeit

Santa Rudevica

## Evaluation von Werkzeugen zur Einbruchserkennung in Fahrzeugnetzwerken

*Fakultät Technik und Informatik*
*Department Informatik*

*Faculty of Engineering and Computer Science*
*Department Computer Science*

Santa Rudevica

# Evaluation von Werkzeugen zur Einbruchserkennung in Fahrzeugnetzwerken

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 Becke

Eingereicht am: 19. Januar 2023

**Santa Rudevica**

**Thema der Arbeit**

Evaluation von Werkzeugen zur Einbruchserkennung in Fahrzeugnetzwerken

**Stichworte**

IT-Sicherheit, Einbruchserkennung, Fahrzeugnetzwerke, NIDS, Automotive, Ethernet

**Kurzzusammenfassung**

Die Sicherheit in Fahrzeugnetzwerken spielt bei einem steigenden Vernetzungsgrad innerhalb des Fahrzeuges eine größere Rolle. Immer mehr Funktionalitäten werden in Fahrzeugen durch Software, die eine verteilte Kommunikation in Fahrzeugnetzwerken voraussetzt, gesteuert. Auch sicherheitskritische Anwendungen sind Teil des Netzwerks und haben aufgrund der Kritikalität entsprechend hohe Anforderungen an die Netzwerksicherheit. Ein weiterer Aspekt, der die Angriffsoberfläche der Netzwerke in Fahrzeugen der Zukunft erhöht, ist die globale Vernetzung, denn die Anzahl der Schnittstellen nach außen erhöht sich ebenfalls. Dadurch stellt ein Auto ein offenes, sicherheitskritisches System dar. Für die Netzwerksicherheit führt das zu einer großen Herausforderung, da mögliche Angriffe ein hohes Maß an Kritikalität erreichen können. Um Angriffe auf ein Netzwerk festzustellen, gibt es verschiedene Ansätze. Ziel dieser Arbeit ist es die Eignung von bereits auf dem Markt existierenden Werkzeugen zur Einbruchserkennung, welche nicht für den Einsatz in Fahrzeugnetzwerken ausgerichtet sind, zu evaluieren. Diese werden anhand verschiedener Kriterien, welche allgemein für die Überwachung des Netzwerkverkehrs relevant sind, ausgewählt. Nach der Vorauswahl werden Werkzeuge einzeln vorgestellt und evaluiert. Für die Evaluation werden potenziell geeignete Werkzeuge ausgewählt. Diese werden auf die Funktionsweise und mögliche Ansätze im Bereich Einbruchserkennung in Fahrzeugnetzwerken untersucht.

**Santa Rudevica**

**Title of Thesis**

Evaluation of tools for intrusion detection in vehicle networks

**Keywords**

IT-Security, Intrusion Detection, Vehicle networks, NIDS, Automotive, Ethernet

**Abstract**

With an increasing degree of networking within a vehicle, security in vehicle networks is playing an increasingly important role. More and more functionalities in vehicles are controlled by software that requires distributed communication in vehicle networks. Safety-critical applications are also part of the network and have correspondingly high network security requirements due to their criticality. Another aspect that increases the attack area of networks in vehicles of the future is global networking, as the number of interfaces to the outside world also increases. This makes a car an open safety-critical system. For network security, this leads to a major challenge, as possible attacks can reach a high level of criticality. In order to detect attacks on a network, there are different approaches. The aim of this work is to evaluate the suitability of intrusion detection tools already available on the market, which are not designed for use in vehicle networks. These are selected on the basis of various criteria that are generally relevant for monitoring network traffic. After the pre-selection, tools are presented and evaluated individually. Potentially suitable tools are selected for evaluation. These are examined for their functionality and possible approaches in the area of intrusion detection in vehicle networks.

# Inhaltsverzeichnis

**Abbildungsverzeichnis** **viii**

**Tabellenverzeichnis** **x**

**Abkürzungen** **xi**

**1 Einleitung** **1**

**2 Grundlagen** **3**

2.1 Einbruchserkennung . . . . . 3

2.1.1 Intrusion Detection System (IDS) . . . . . 3

2.1.2 Host-basierte IDS . . . . . 4

2.1.3 Netzwerk-basierte IDS . . . . . 4

2.1.4 Erkennungsmodelle und Datenverarbeitung . . . . . 5

2.1.5 Intrusion Prevention Systems . . . . . 7

2.2 Fahrzeugnetzwerke . . . . . 7

**3 Verwandte Arbeiten** **14**

3.1 Allgemeiner IDS Vergleich . . . . . 14

3.1.1 IDS: A comprehensive review[23] . . . . . 14

3.1.2 Performance Comparison and Detection Analysis in Snort and Suricata Environment[31] . . . . . 16

3.2 IDS in Fahrzeugnetzwerken . . . . . 19

3.2.1 Ein Spezifkations-basierter IDS Ansatz im Automotive Ethernet [7] 20

3.2.2 Intrusion detection system for automotive Controller Area Network (CAN) bus system: a review [24] . . . . . 21

3.3 Implikationen . . . . . 23

**4 Ist Analyse** **24**
4.1 IDS Tools . . . 24
4.1.1 Übersicht der IDS auf dem Markt . . . 24
4.2 Fahrzeugnetzwerke . . . 27
4.2.1 Evolution: Topologien in Fahrzeugnetzwerken . . . 27
4.2.2 Domänen-Zonen-Topologie . . . 29
4.2.3 Zonenarchitektur . . . 29
4.2.4 Zusammenfassung . . . 30
4.3 SecVI Demonstrator . . . 31
4.3.1 Allgemein . . . 31
4.3.2 Topologie . . . 31
4.4 Problemstellung . . . 32

**5 Anforderungsanalyse** **34**
5.1 Systemeigenschaften . . . 35
5.2 Datenanforderungen . . . 35
5.3 Zeitliche Anforderungen . . . 36
5.4 Sonstige technische Anforderungen . . . 37
5.5 Sonstige nichttechnische Anforderungen . . . 37
5.6 Vorauswahl der Werkzeuge . . . 38

**6 Evaluation** **40**
6.1 Allgemeine Funktionsweise . . . 40
6.2 Aufbau der Signatur-basierten Regeln . . . 42
6.2.1 Snort . . . 42
6.2.2 Suricata . . . 46
6.2.3 Zeek . . . 47
6.3 Einsatz in Fahrzeugnetzwerken . . . 49
6.3.1 Protokollstack . . . 50
6.3.2 Payload . . . 50
6.3.3 Paketgröße . . . 51
6.3.4 Zeitverhalten . . . 51
6.3.5 Regeln in Fahrzeugnetzwerken . . . 52
6.4 Evaluation . . . 53

**7 Fazit und Ausblick** **55**
7.1 Fazit . . . 55

7.2 Ausblick . . . . . 56

**Literaturverzeichnis** **57**

**A Anhang** **61**

Selbstständigkeitserklärung . . . . . 62

# Abbildungsverzeichnis


2.1 Funktionsweise Signatur-basierte IDS (Quelle: [19] Abbildung 1) . . . . . 6
2.2 Standard Ethernet-Frame Format (Quelle: [36] Abbildung 2-6) . . . . . 8
2.3 Der Inhalt eines 802.1Q-Header Segments . . . . . 9
2.4 Aufbau eines 802.1Q Ethernet-Switches (Quelle: [18]) . . . . . 10
2.5 Die Übetragungskontrolle nach 802.1Qbv (Quelle: [18] Abbildung 8-14) . . 10
2.6 Drei Funktionalitätsschichten im Netzwerk (Quelle: [21] Abbildung 3) . . . 12
2.7 Konventionelles Netzwerk und Software-Defined Networking (SDN) (Quelle: [25] Abbildung 1) . . . . . 13

3.1 Übersicht der IDS Taxonomie(Quelle: [23] Abbildung 1) . . . . . 17
3.2 Übersicht Erkennungsrate von Snort und Suricata in einer Single-Core Umgebung (Quelle: [31] Abbildung 7) . . . . . 19
3.3 Übersicht Erkennungsrate von Snort und Suricata in einer Multi-Core Umgebung (Quelle: [31] Abbildung 10) . . . . . 19
3.4 IDS Trend in Controller Area Network (CAN)-basierten Netwerken (Quelle: [24] Abbildung 5) . . . . . 22

4.1 Feldbusbasierte Topologie in Serienfahrzeugen - Farben repräsentieren Anwendungsdomänen, Gateway (G) in schwarz, mehrfarbige Steuergeräte sind ohne Gateway Funktionalität an mehreren Bussen angebunden (Quelle: [36] Abbildung 3-8) . . . . . 28
4.2 Domänen-Gateway-Topologie - Vormals zentrales Gateway (schwarz) wird ersetzt durch mehrere Domänen-Gateways (G) und Ethernet Backbone, mehrfarbige Steuergeräte sind ohne Gateway-Funktionalität an mehreren Bussen angebunden (Quelle: [36] Abbildung 3-9) . . . . . 28
4.3 Domänen-Zonen-Topologie - Ursprüngliche Feldbusse werden zerteilt und lokal über Gateways (G) an den Ethernet Backbone angebunden; mehrfarbige Steuergeräte und Gateways besitzen Funktionalität verschiedener Domänen (Quelle: [36] Abbildung 3-10) . . . . . 29

4.4 Zonenarchitektur (Quelle: [15] Abbildung 1-c) . . . . . 30
4.5 2016' Seat Ateca Prototyp (Quelle: [26] Abbildung 1a) . . . . . 32
4.6 Topologie des SecVI Demonstrators . . . . . 33

6.1 Snort Regel (Quelle: [20] Abbildung 1) . . . . . 42
6.2 Snort Regel mit Adressliste und Negationsoparator (Quelle: [12] Abbildung 3.3) . . . . . 43
6.3 Suricata Regel bestehend aus einer Action (rot), Header (grün) und Option (blau) (Quelle: [28]) . . . . . 46
6.4 Eine einfache Zeek Signatur(Quelle: [29]) . . . . . 48
6.5 Ein Zeek generiertes Event zu Abbildung 6.4 (Quelle: [29]) . . . . . 48
6.6 Snort detection_filter Syntax(Quelle: [12]) . . . . . 52
6.7 Suricata detection_filter Syntax(Quelle: [28]) . . . . . 52
6.8 Signatur-basierte Regel Snort . . . . . 53

# Tabellenverzeichnis

4.1 Rechercheergebnis IDS-Tools auf dem Markt *Anomalie-basierte Funktionsweise **Signatur-basierte Funktionsweise . . . . . 25

5.1 Anforderungen an Systemeigenschaften . . . . . 35
5.2 Anforderungen an Daten . . . . . 36
5.3 Zeitliche Anforderungen . . . . . 36
5.4 Sonstige technische Anforderungen . . . . . 37
5.5 Sonstige nichttechnische Anforderungen . . . . . 38
5.6 Anforderungsmatrix Priorität „hoch“ . . . . . 38

# Abkürzungen

**CAN** Controller Area Network

**CIDR** Classless Inter-Domain Routing

**CoRE AG** Communication- over-Realtime-Ethernet Arbeitsgruppe

**DEI** Drop Eligible Indicator

**DoIP** Diagnostics over IP

**FIFO** First In - First Out

**GUI** Graphical User Interface

**HAW** Hochschule für Angewandte Wissenschaften

**HIDS** Host-based Intrusion Detection System

**IBM** International Business Machines Corporation

**ICMP** Internet Control Message Protocol

**IDS** Intrusion Detection System

**IP** Internet Protocol

**IPS** Intrusion Prevention System

**NIDS** Network-based Intrusion Detection System

**OISF** Open Information Security Foundation

**OSI** Open Systems Interconnection

**PCP** Priority Code Point

**SDN** Software-Defined Networking

**SecVI** Security for Vehicular Information

**SOME/IP** Scalable Service-Oriented Middleware over IP

**TCI** Tag Control Information

**TCP** Transmission Control Protocol

**TG** Transmission Gate

**TPID** Tag Protocol Identifier

**TS** Transmission Selection

**TSA** Transmission Selection Algorithm

**TSN** Time-Sensitive Networking

**UDP** User Datagram Protocol

**VID** VLAN-Identifier

**VM** Virtual Machine

# 1 Einleitung

Immer mehr gerät das Thema Sicherheit in IT-Systemen in den Fokus. Durch die steigende Komplexität der Vernetzung kann jedes System zu einem potentiellen Angriffsziel werden. Es ist eine Herausforderung moderner IT-Systeme Gefahrenpotenzial zu erkennen und Sicherheitslücken bestmöglich zu schließen. „Es ist nicht nur die Anzahl von Sicherheitsvorfällen, die besorgniserregend ist, es ist auch die rasante Entwicklung neuer und angepasster Angriffsmethoden, die massenhafte Ausnutzung schwerwiegender Software-Schwachstellen und die teilweise gravierenden Folgen, die erfolgreiche Cyber-Angriffe auslösen.[6]“

Die Digitalisierung spielt in heutiger Welt eine große Rolle, dies gilt genauso für den Automobilbereich. Immer mehr Funktionen im Fahrzeug werden von Software gesteuert und immer mehr soll automatisiert erfolgen. Perspektivisch sollen Fahrzeuge global vernetzt sein.

Vor allem die globale Vernetzung und die Übernahme der Steuerung von sicherheitskritischen Funktionen stellen ein großes Risiko dar. Somit können Cyberangriffe durchaus katastrophale Folgen haben, zum Beispiel indem ein Fahrzeug durch einen Angriff manipuliert wird und als Folge einen Unfall verursacht wird. Für die Fahrzeuge der Zukunft ist es maßgeblich auf Angriffe vorbereitet zu sein, potenzielle Gefahren rechtzeitig zu erkennen und bei einem Angriff aktiv zu reagieren, um schwerwiegende Folgen zu verhindern.

In dieser Arbeit geht es darum die Sicherheit eines Fahrzeugnetzwerkes mithilfe von Intrusion Detection Systems zu evaluieren. Intrusion Detection ist ein möglicher Weg Angriffe zu erkennen und somit die Grundlage für eine entsprechend notwendige Reaktion zu schaffen.

Diese Arbeit ist in der Communication- over-Realtime-Ethernet Arbeitsgruppe (CoRE AG) der Hochschule für Angewandte Wissenschaften (HAW) Hamburg entstanden. Das Ziel dieser Arbeit ist herauszufinden, welche Anforderungen Fahrzeugnetze an ein IDS stellen und ob der Einsatz von so einer Art Sicherheitsmechanismus überhaupt sinnvoll wäre. Dafür werden bereits bestehende Werkzeuge zur Einbruchserkennung ausgewählt, auf das Erfüllen unterschiedlicher Anforderungen für Fahrzeugnetzwerke geprüft und eva-

luiert. Das Netzwerkdesign richtet sich an dem SecVI Demonstrator, welcher im Kapitel 2.3. vorgestellt wird.
Die Arbeit ist in sieben Kapitel gegliedert. Im zweiten Kapitel werden die benötigten Grundlagen vermittelt. Dabei wird auf Themen wie Intrusion Detection Systems, welche Unterschiede es in dem Bereich gibt und wie diese grundätzlich funktionieren, eingegangen. Ausßerdem werden die Grundlagen in den Bereichen Fahrzeugnetzwerke, für die Evaluation benötigte Protokolle und Security for Vehicular Information (SecVI) Demonstrator vermittelt.
Das dritte Kapitel befasst sich mit verwandten Arbeiten sowohl im Bereich von IDS allgemein als auch fahrzeugspezifisch. Außerdem wird am Ende des Kapitels eine Problemstellung vorgestellt. In dem Kapitel 4 wird eine Ist-Analyse durchgeführt. Dabei werden die wichtigsten IDS Werkzeuge auf dem Markt und der aktuelle Stand im Bereich Fahrzeugnetzwerke vorgestellt. Außerdem wird in dem vierten Kapitel der SecVI Demonstrator beschrieben.
Des Weiteren werden im fünften Kapitel sowohl die technischen als auch die nichttechnischen Anforderungen an ein potenzielles IDS definiert. Basierend auf den definierten Anforderungen wird eine Vorauswahl der Werkzeuge getroffen. Diese IDS werden im Kapitel 6 genauer untersucht. Dabei wird die allgemeine Funktionsweise solcher Werkzeuge vorgestellt. Außerdem folgen eine detaillierte Beschreibung der ausgewählten IDS und eine Analyse, ob die Anforderungen für die Anwendung in Fahrzeugnetzwerken erfüllt werden können. Am Ende des Kapitels werden die Werkzeuge evaluiert.
Zum Schluss folgt in Kapitel 7 das Fazit über den Einsatz von Werkzeugen zur Einbruchserkennung anhand der Evaluation. Außerdem wird die Relevanz von IDS für Fahrzeugnetzwerke evaluiert und ein Ausblick über mögliche Alternativen und fortführende Untersuchungen gegeben.

# 2 Grundlagen

In diesem Kapitel werden die Grundlagen für das weitere Verständnis dieser Arbeit erläutert. Zuerst werden im Abschnitt 2.1 die wichtigsten Informationen im Bereich Einbruchserkennung vermittelt. Dabei werden die wichtigsten Unterschiede möglicher Architektur-Ansätze, die grundsätzliche Funktionsweise und die Abgrenzung zu Intrusion Prevention System (IPS) erklärt. Im Abschnitt 2.2 werden die Grundlagen zu Kommunikation in zukünftigen Fahrzeugnetzwerken vermittelt. Dabei wird der Begriff Time-Sensitive Networking und die Relevanz in Fahrzeugnetzwerken erläutert und das Konzept von Software-Defined Networking vorgestellt.

## 2.1 Einbruchserkennung

Bei Intrusion Detection bzw. Einbruchserkennung geht es darum, eine Komponente oder einen Bereich im IT-System zu überwachen und anhand gegebener Methoden auffällige Ereignisse, die auf einen Angriff hindeuten, zu erfassen. In diesem Kontext ist ein Angriff ein unauthorisierter Zugriff auf das System. „Als Intrusion-Detection wird die aktive Überwachung von Computersystemen und/oder -netzen mit dem Ziel der Erkennung von Angriffen und Missbrauch bezeichnet[3].“

### 2.1.1 IDS

„Ein Intrusion Detection System bezeichnet eine Zusammenstellung von technischen Werkzeugen, die den gesamten Prozess der Intrusion-Detection unterstützen.[35]“
Grundsätzlich bestehen Network-based Intrusion Detection System (NIDS) aus drei Hauptkomponenten, welche miteinander interagieren. Diese sind ein Datenmonitor, ein Analyzer und eine Komponente zur Ergebnisdarstellung. Der Datenmonitor kann als ein Sniffer, welcher für das Mitlesen der Pakete zuständig ist, implementiert sein. Der mitgelesene Netzwerkverkehr wird anhand verschiedener Methoden und Kriterien von der Analyzer

Komponente paketweise ausgewertet. Mögliche Gefahren bzw. verdächtige Aktivitäten werden über die Ergebnisdarstellung gemeldet. Es gibt verschiedene Ansätze, um Angriffe und potentielle Bedrohungen mithilfe von IDS und IPS zu erkennen und ggf. darauf zu reagieren. Grundsätzlich werden IDS in zwei Kategorien eingeteilt: Host-basierte und Netzwerk-basierte Intrusion Detection Systems [23]. Außerdem gibt es spezielle Formen der Einbruchserkennungssysteme: hybride IDS, welche sich aus Host-based Intrusion Detection System (HIDS) und NIDS zusammensetzen, und Wireless IDS, welche für drahtlose Netzwerke besonders geeignet sind.

### 2.1.2 Host-basierte IDS

Bei einem Host-basierten IDS geht es um die Überwachung einzelner Server. Dazu wird ein Agent, welcher die überwachende Komponente darstellt und zentral die Informationen sammelt, platziert. Es werden verschiedene Systeminformationen wie Log-Dateien oder Kernel-Daten ausgewertet. Um die Einbruchserkennung zu gewährleisten, muss an jedem Host ein eigenes IDS installiert werden. Durch die Positionierung direkt am Host ist es möglich spezifische Angriffe zu erkennen, sodass eine feine Granularität gewährleistet werden kann [37].
Im weiteren Verlauf dieser Bachelorarbeit werden HIDS nicht berücksichtigt. Der Schwerpunkt liegt auf Netzwerküberwachung und somit auf NIDS.

### 2.1.3 Netzwerk-basierte IDS

Aufgrund der steigenden Komplexität der Vernetzung und der vermehrten Einsätze verteilter Systeme reicht die Überwachung einzelner Hosts nicht mehr aus.
Das Ziel eines Netzwerk-basierten IDS ist, wie der Name schon vermuten lässt, ein ganzes Netzwerk zu überwachen, um potentielle Angriffe zu erkennen. Dabei werden Sensoren an mindestens einem Netzsegment positioniert. Die Netzwerktopologie spielt somit eine wichtige Rolle. Bereits vor dem einsetzen eines NIDS ist es sinnvoll strategisch zu überlegen, an welcher Stelle die Netzwerküberwachung stattfinden soll und welches Ziel dabei verfolgt wird. Mithilfe der Platzierung der Sensoren können verschiedene Aussagen über die erkannten Angriffe getroffen werden. Sofern eine Firewall im Netzwerk als Schutzmechanismus eingesetzt wird, kann man zusätzlich ein IDS verwenden. Eine Platzierung beispielsweise vor der Firewall könnte für eine statistische Auswertung tatsächlicher Angriffe hilfreich sein, andererseits wird ein IDS hinter der Firewall platziert, könnte dies

als ein weiterer Schutzmechanismus dienen, falls die Firewall eine Fehlkonfiguration beinhaltet [37].

### 2.1.4 Erkennungsmodelle und Datenverarbeitung

Es gibt zahlreiche Ansätze der Methodik und der Funktionsweise von IDS. An erster Stelle ist es wichtig zu wissen, welche Art von Daten überwacht werden sollen. Die wichtigsten Datenquellen bei IDS sind jegliche Art von Logs (zum Beispiel Host- oder Applikationslogs) und der Paketverkehr im Netzwerk. Einige Tools bieten auch die Möglichkeit der Überwachung von TLS/SSL-Zertifikaten, HTTP- und DNS-Anfragen und die File Extraction als zusätzliche Optionen für die Umsetzung der Intrusion Detection[27].
Das Sammeln von Daten kann sowohl zentral als auch verteilt erfolgen. Für diesen Prozess sind üblicherweise Agenten bei HIDS und Sensoren bei NIDS zuständig. Die Datenanalyse und -verarbeitung kann ebenfalls zentral und verteilt erfolgen.
IDS werden hauptsächlich in drei Kategorien der Funktionsweise unterschieden:

- Anomalie-basiert
- Signatur-basiert
- Spezifikations-basiert

Im folgenden wird die grunsdsätzliche Funktionsweise der drei Ansätze weiter erläutert und die wichtigsten Unterschiede dargestellt.

#### Anomalie-basierte IDS

Anomalie-basierte IDS implementieren ein statistisches Modell. Um einen diesen Ansatz zu verwenden, ist es erforderlich das IDS auf den „Normalzustand“ zu trainieren. So kann das akzeptable Verhalten vom ungewöhnlichen Verhalten unterschieden werden. Die Anomalie Erkennung basiert darauf, dass eine vordefinierte Abweichung vom Normalzustand erkannt wird. Die Schwierigkeit liegt darin einen optimalen Schwellwert zu finden, sodass zwar anomales Verhalten erkannt wird, aber eine möglichst geringe Anzahl an False Positives erzeugt wird. Diese Schwellwertfindung stellt einen Nachteil des Anomalie-basierten IDS dar, da es dadurch fehleranfällig konfiguriert werden kann [8].
Aufgrund der Schwächen der statistischen Erkennung werden immer mehr Anomalie-basierte IDS Ansätze, welche auf Basis von Machine Learning arbeiten, entwickelt [13].

### Signatur-basierte IDS

Signatur-basierte IDS überwachen den Netzwerkverkehr und verwenden eine Datenbank mit Paketsignaturen, welche für Angriffe typisch sind und auf ungewöhnliches Verhalten hindeuten könnten. Dieser Ansatz basiert auf Musterekennung. Wie in der Abbildung 2.1 dargestellt, werden die Signaturen aus der Signaturdatenbank mit den Paketsignaturen im Netzwerk verglichen.
Durch diese Methodik werden die Fehlerrate und die Anzahl der False Positives gering gehalten. Genauso ist es unproblematisch Angriffe aufgrund der charakteristischen Signaturen zu klassifizieren.
Jedoch ist der Nachteil von Signatur-basierten IDS, dass die Einbruchserkennung nur dann funktioniert, wenn sich die entsprechenden Signaturen in der Datenbank befinden. Auch geringe Signaturabweichungen können für die Nichterkennung der Angriffe sorgen. Um diese Problematik zu minimieren, ist es wichtig, dass die Datenbanken regelmäßig, sobald neue Angriffsmöglichkeiten bekannt sind, aktualisiert werden [8].

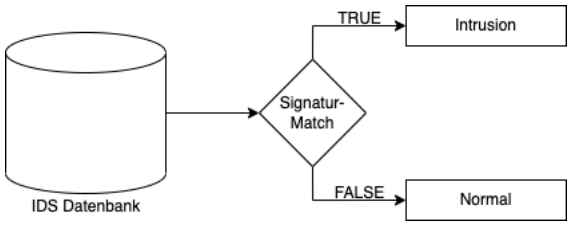



Abbildung 2.1: Funktionsweise Signatur-basierte IDS (Quelle: [19] Abbildung 1)

### Spezifikations-basierte IDS

Ähnlich wie bei dem Anomalie-basierten IDS geht es bei dem Spezifikations-basierten IDS um die Festlegung des „Normalzustandes“, welcher als Spezifikation verstanden wird. Anders als bei Anomalie-basierten IDS wird dieser Zustand nicht erlernt. „Normale“ Verhaltensweisen werden in einer Dokumentation ausführlich beschrieben.
Spezifikations-basierte IDS implementieren kein statistisches Modell, sondern es baut darauf auf, von der Dokumentation bzw. Spezifikation abweichendes Verhalten als einen Einbruch zu erkennen. Diese Vorgehensweise kann sowohl einen Vorteil, aber auch einen Nachteil darstellen. Es werden alle nicht definierten Verhaltensweisen außerhalb der Spezifikation als Angriffe erkannt. Jedoch ist es erforderlich die Spezifikation möglichst korrekt zu dokumentieren, auf die Genauigkeit zu achten und immer zu aktualisieren, damit bereits „normales“ Verhalten nicht als ein Angriff identifiziert wird [5].

### 2.1.5 Intrusion Prevention Systems

IDS sind üblicherweise darauf ausgelegt Netzwerke (bei NIDS) zu überwachen, Einbrüche zu erkennen und auf diese passiv zu reagieren zum Beispiel indem ein Alarm ausgelöst wird. Dies geschieht frühestens in Echtzeit oder erst nachdem ein Angriff erfolgt ist. Ein IDS greift nicht aktiv in das System ein. Um aktive Maßnahmen aufgrund der Ergebnisse eines IDS einzuleiten, ist entweder menschliche Interaktion oder ein weiteres System erforderlich.
Aus diesem Grund bietet sich die Möglichkeit IDS-Tools auf Einbruchsprävention auszuweiten. Ein IPS ist ein Kontrollsystem, und somit in der Lage Pakete zurückzuweisen und aktiv in das System einzugreifen.
Die Gemeinsamkeit von IDS und IPS besteht darin, dass beide Systeme Netzwerkverkehr überwachen [4].

## 2.2 Fahrzeugnetzwerke

Schon länger gilt das Fahrzeug nicht mehr als reines Fortbewegungsmittel. Die Anzahl der elektronischen Komponenten und Steuergeräte, die miteinander kommunizieren und Informationen austauschen, steigt. Und somit steigt auch die Komplexität der Vernetzung, die nicht nur innerhalb eines Fahrzeugs, sondern auch global stattfindet. Dies bringt einige Herausforderungen, welche die heutigen Fahrzeugnetzwerkstrukturen an ihre Grenzen bringen, mit sich.
Die Netzwerkteilnehmer im Fahrzeug erfüllen unterschiedliche Funktionen, nutzen aber gemeinsame Kommunikationskanäle. Daher weisen die Komponenten unterschiedliche Wichtigkeit und Relevanz für die Funktionsfähigkeit des Fahrzeugs auf. Dies führt dazu, dass die Teilnehmer nicht immer die gleichen Anforderungen an die Nachrichtenübertragung erfordern. Zu dieser Problematik wird in diesem Abschnitt Time-Sensitive Networking, eine Technologie, die es ermöglicht Ethernet-basierte Pakete nicht gleichberechtigt zu behandeln, vorgestellt und näher erläutert.
Bei zunehmender Netzwerkkomplexität werden Forderungen nach Skalierbarkeit und Reduzierung der Komplexität immer bedeutungsvoller [36]. Außerdem spiegelt sich die Netzwerkkomplexität in der Anzahl der Netzwerkgeräte, welche meistens einzeln aufwendig konfiguriert bzw. nach Ausfällen rekonfiguriert werden müssen, wider. Daher wird in diesem Kapitel wird der Ansatz des Software-Defined Networking, der diesen Herausforderungen in Fahrzeugnetzwerken entgegenwirkt, beschrieben.

### Time-Sensitive Networking

Eine Ethernet-basierte Kommunikation bringt bedeutsame Vorteile wie Skalierbarkeit, Flexibilität und Kosteneffizienz mit sich, weshalb dieser Ansatz für die unterschiedlichsten Anwendungsbereiche optimal geeignet ist. Jedoch entspricht dies nicht vollständig den Kommunikationsanforderungen für Fahrzeugnetzwerke der Zukunft, da Kriterien wie Robustheit, garantierte Übertragungszeit und Verfügbarkeit nicht umgesetzt sind und somit keine Echtzeitanforderungen erfüllt werden können. Funktionen wie Bestätigungen (Acknowledge) oder ein erneutes Versenden (Retransmit), um eine gewisse Quality of Service zu gewährleisten, müssen in höheren Schichten des Open Systems Interconnection (OSI)-Referenzmodells implementiert werden [36]. Time-Sensitive Networking (TSN) stellt eine Technologie, welche die fehlenden, zeitlichen und sicherheitsrelevanten Anforderungen des Ethernets erfüllen, dar [14]. TSN stellt an erster Stelle eine Sammlung von Standards, welche von der Time-Sensitive Networking Task Gruop definiert werden, dar. In diesem Abschnitt werden die wichtigsten TSN Standards wie IEEE 802.1Q-2018 und IEEE 802.1Qbv-2015 erläutert. Für eine technisch detaillierte Beschreibung der Funktionsweise wird auf die Standards von IEEE verwiesen.

### IEEE 802.1Q-2018

<table>
<tr><td colspan="2" rowspan="2">Layer 1</td><td colspan="6">Ethernet Paket: 72 B bis 1526 B bzw. 1530 B</td><td rowspan="2"></td></tr>
<tr><td colspan="6">Layer 2 Ethernet Frame: 64 B bis 1518 B bzw. 1522 B</td></tr>
<tr><td rowspan="2">Präambel</td><td rowspan="2">Start of Frame</td><td colspan="4">Ethernet Header: 14 B bzw. 18 B</td><td rowspan="2">Nutzdaten</td><td rowspan="2">Frame Check Sequence</td><td rowspan="2">Inter Frame Gap</td></tr>
<tr><td>MAC Ziel</td><td>MAC Quelle</td><td>802.1Q (optional)</td><td>Ethertype/ Länge</td></tr>
<tr><td>7 B</td><td>1 B</td><td>6 B</td><td>6 B</td><td>4 B</td><td>2 B</td><td>42 B bzw. 46 B bis 1500 B</td><td>4 B</td><td>12 B</td></tr>
</table>

Abbildung 2.2: Standard Ethernet-Frame Format (Quelle: [36] Abbildung 2-6)

In dem IEEE Standard 802.1Q-2018, welcher im weiteren Verlauf der Arbeit als 802.1Q bezeichnet wird, geht es um eine 4 Byte Ethernet-Header Erweiterung. Diese ist in der Abbildung 2.2 gelb markiert. In der Abbildung 2.3 werden die konkreten Inhaltsvorgaben der 802.1Q-Erweiterung dargestellt. Dieses Segment setzt sich aus dem Tag Protocol Identifier (TPID) und der Tag Control Information (TCI), welche jeweils 2 Byte groß sind, zusammen. Die TCI besteht aus drei Untersegmenten:

- Priority Code Point (PCP): Der PCP ist ein 3 Bit großes Feld, welches die Priorisierung des Ethernet-Paketes ermöglicht. Dabei wird eine Priorität von 0 bis 7 angegeben, wobei 7 die höchste und 0 die niedrigste Priorität darstellt.
- Drop Eligible Indicator (DEI): Der DEI ist ein 1 Bit großes Feld und gibt an, was im Fall eines Paketstaus mit diesem Paket geschehen soll, also ob dieses Paket verworfen werden darf. Der DEI kann auch gesetzt werden, wenn das PCP-Feld nicht benutzt wird.
- VLAN-Identifier (VID): Der VID ist 12 Bit groß und identifiziert das zum Paket gehörige VLAN.

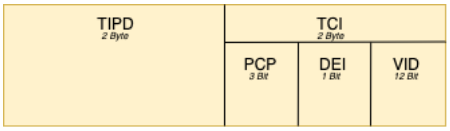



Abbildung 2.3: Der Inhalt eines 802.1Q-Header Segments

Die 802.1Q Standardisierung ermöglicht somit eine Priorisierung der Ethernet-Pakete. Dieser Aspekt hat den Vorteil, dass Pakete mit einer höheren Priorität eine geringere Latenz aufweisen kann. Außerdem können Pakete bei einem Stau mithilfe des DEI-Feldes verworfen werden, was der Verzögerung bei hohem Paketaufkommen entgegenwirkt und das Verarbeiten der höher priorisierten Pakete entsprechend beschleunigt.
In der Abbildung 2.4 ist der Aufbau eines 802.1Q-Switches dargestellt. Dabei werden die ankommenden Pakete nach Prioritäten sortiert. Für jede Priorität 0 bis 7 gibt es eine First In - First Out (FIFO) Warteschlange. Der Scheduler entscheidet anhand der Priorisierung, welches Paket als nächstes verarbeitet werden soll. Die Pakete innerhalb einer Warteschlange werden gleichberechtigt behandelt[18].

### IEEE 802.1Qbv-2015

In der Abbildung 2.4 dargestellten Paketverarbeitung kann es zu Scheduling Anomalien kommen, indem zum Beispiel große, niedrig priorisierte Pakete neu dazugekommene, höher priorisierte Pakete blockieren. Dies resultiert daraus, dass Ethernet-Pakete bei der Übertragung in dem Fall nicht unterbrochen werden können [36].
Der IEEE 802.1Qbv-2015 Standard [17], im Folgenden 802.1Qbv genannt, ist ein weiterer Substandard von TSN und wirkt dem beschriebenen Scheduling Problem entgegen.

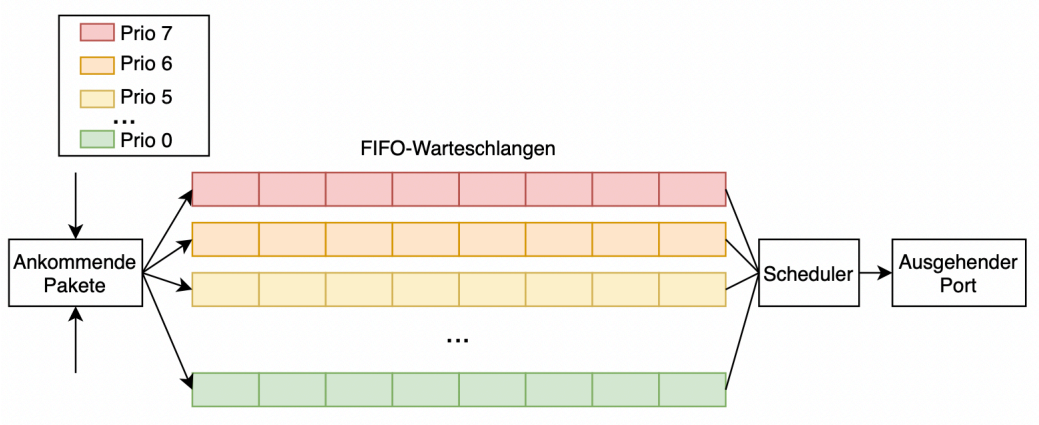



Abbildung 2.4: Aufbau eines 802.1Q Ethernet-Switches (Quelle: [18])

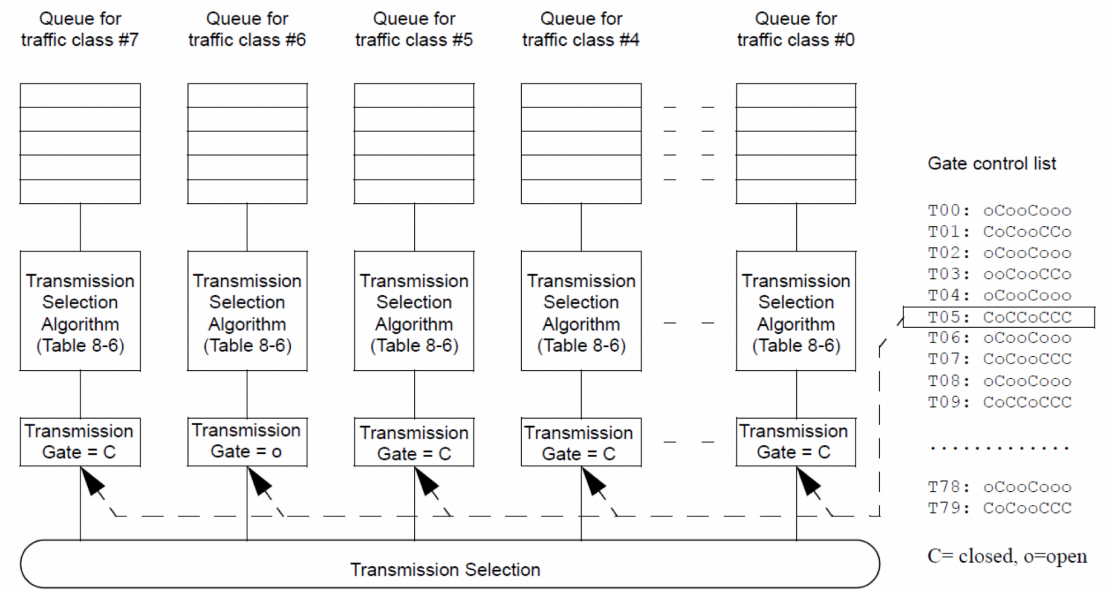



Abbildung 2.5: Die Übetragungskontrolle nach 802.1Qbv (Quelle: [18] Abbildung 8-14)

Zusätzlich zu den acht priorisierten Warteschlangen, wie in der Abbildung 2.4 dargestellt, setzt der 802.1Qbv Standard eine zeitgesteuerte Kommunikation um. Die Warteschlangen können durch eine Übetragungskontrolle synchronisiert, zeitgesteuert aktiviert und deaktiviert werden. Die Übetragungskontrolle wird in der Abbildung 2.5 visualisiert, diese besteht aus drei zusätzlichen (vgl. Abbildung 2.4) Komponenten:

- Transmission Selection Algorithm (TSA): Der Übertragungsauswahlalgorithmus trifft eine Auswahl an Ethernet-Frames, welche aus der Warteschlange als nächstes weiter verarbeitet werden.
- Transmission Gate (TG): Der TG ist eine weitere Instanz, welche beschreibt, ob die ausgewählten Ethernet-Frames in die Übertragungskontrolle weitergeleitet werden.
- Transmission Selection (TS): Die Übertragungsauswahl entscheidet darüber, welches der Ethernet-Frames über den Port des Switches versendet wird.

Nach 802.1Qbv [17] kann ein Übertragungsauswahlalgorithmus entweder selbst implementiert oder von den in 802.1Qbv vorgegebenen Algorithmen gewählt werden.
Es stehen folgende Übertragungsauswahlalgorithmen zur Verfügung:

- Algorithmus Credit-based Shaper (CBS)
- Enhanced-Transmission-Selection Algorithmus (ETS-Algorithmus)
- Strikte Priorität

Nach der Vorauswahl durch den Algorithmus kommen die Ethernet-Frames in das Transmission Gate, welches zwei Zustände annehmen kann:

- OPEN
- CLOSED

Das Transmission Gate ermöglicht einen zeitgesteuerten Datenverkehr. Die Zustände der Gates werden in der Gate Control List (siehe Abbildung 2.5) verwaltet.
Falls mehrere Gates im Zustand OPEN sind, entscheidet die Transmission Selection über die Weiterleitung der Ethernet-Frames an den Ausgangsport. Wenn die Warteschlangen überlaufen und der Gate im Zustand CLOSED ist, werden Ethernet-Frames verworfen.
Dieser Mechanismus unterstützt die zeitlichen Anforderungen der Kommunikation, indem die Pakete nicht wie üblich gleichberechtigt einfach weiter geleitet werden.

### Software-Defined Networking

Computernetzwerke können, wie in der Abbildung 2.6 dargestellt, in drei Schichten unterteilt werden: Daten-, Kontroll- und Verwaltungsschicht. Die Datenschicht hat als Aufgabe die Weiterleitung der Daten. Die Kontrollschicht ist für die Steuerung des Weiterleitungsverhaltens der Netzwerkgeräte zuständig. Auf der Verwaltungsschicht wird das Netzwerk überwacht und konfiguriert [21].

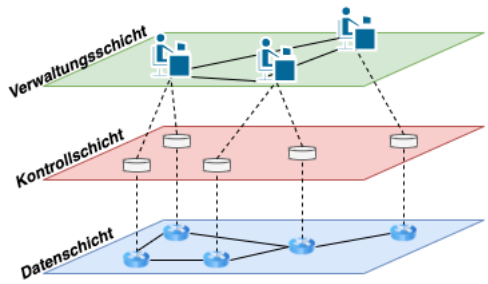



Abbildung 2.6: Drei Funktionalitätsschichten im Netzwerk (Quelle: [21] Abbildung 3)

In konventionellen Netzwerken sind Netzwerkgeräte wie zum Beispiel Switches oder Router sowohl für die Daten- als auch für die Kontrollschicht zuständig. Diese beiden Schichten sind sehr eng gekoppelt und die gesamte Struktur ist dezentralisiert. Dadurch wird das Anpassen definierter Richtlinien oder das Rekonfigurieren nach Ausfällen und Änderungen in größeren Netzwerken erschwert [21].
SDN hingegen entkoppelt die Kontrollschicht von der Datenschicht. Dieser Unterschied wird in der Abbildung 2.7 dargestellt. Die Netzwerkgeräte sind weiterhin für die effektive Weiterleitung der Daten zuständig. Die Richtung der Weiterleitung wird jedoch von dem SDN-Controller, welcher die Kontrollschicht implementiert, gesteuert. Außerdem ermöglicht SDN eine zentrale Konfigurierung und Verwaltung des Netzwerks, sodass die o.g. Nachteile konventioneller Netzwerke aufgehoben werden.

### Openflow

Durch die Trennung der Kontrollebene von Netzwerkgeräten wird ein Kommunikationsprotokoll zwischen der Datenebene auf den Weiterleitungsgeräten (SDN-Switches) und dem SDN-Contoller benötigt. Dies wird durch das Openflow Protokoll sichergestellt. Es wird von der Open Network Foundation standardisiert.

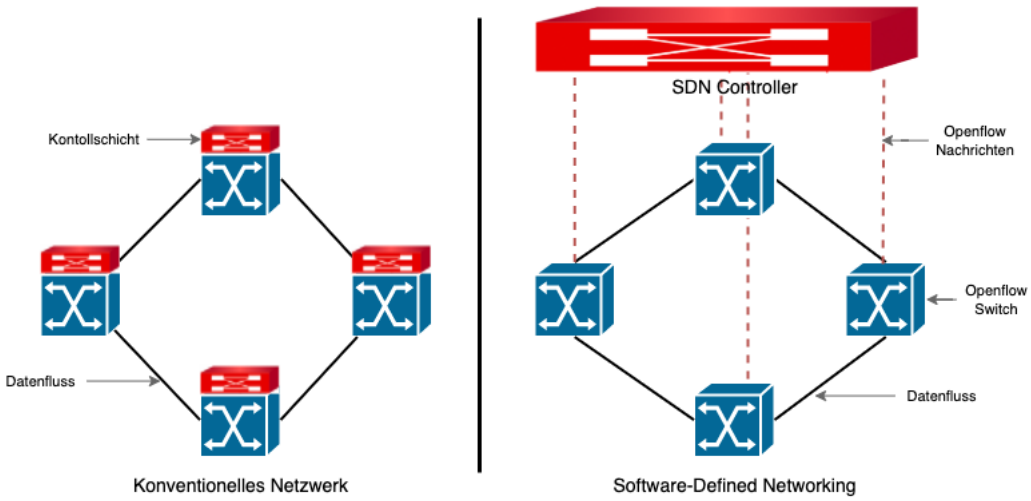



Abbildung 2.7: Konventionelles Netzwerk und SDN (Quelle: [25] Abbildung 1)

Openflow ermöglicht den Zugriff auf die Weiterleitungsebene eines Switches oder Routers über das Netzwerk [22]. Es besteht hauptsächlich aus drei Komponenten:

- Openflow Controller
- Openflow Channel
- Openflow Switch

Der SDN-Controller ist für die Verteilung, Verwaltung und die Anweisungen an die Netzwerkgeräte zuständig. Er kann bestimmen, was passiert, wenn Pakete beispielsweise keine gültigen Flow Einträge haben und wie diese zu behandeln sind. Der Openflow Channel stellt die Kommunikationsschnittstelle zwischen den Switches und dem SDN-Controller dar. Darüber empfängt der Controller die Ereignisse und sendet Pakete an die Switches. Die Openflow Switches werden von dem SDN-Controller über das Openflow Protokoll verwaltet. Ein Switch hält mindestens eine Weiterleitungstabelle (Flow Table), bei welcher ermittelt werden kann, wie ein ankommendes Paket zu behandeln ist. Sollte es keinen Eintrag in der Flow Table geben, leitet der Switch die Paketinformationen an den SDN-Contoller weiter. Daraufhin teilt der Controller dem Switch eine Weiterleitungsregel (Flow Rule) mit. Die Flow Rule enthält Informationen darüber, wie die Pakete zu behandeln sind und welche Aktionen ausgeführt werden sollen [22].

# 3 Verwandte Arbeiten

In diesem Kapitel werden verwandte Arbeiten im Bereich Intrusion Detection aufgegriffen. Es werden vier unterschiedliche Arbeiten vorgestellt. Diese weisen verschiedene Herangehensweisen im Bezug auf Evaluation von IDS auf. Die Auswahl der verwandten Arbeiten erfolgte basierend auf zwei Hauptbereichen: allgemeiner Vergleich der IDS und IDS als Teilbereich möglicher Security Lösungen in Fahrzeugnetzwerken. Wichtig ist es hierbei anzumerken, dass die Anzahl der Arbeiten, die sich mit Intrusion Detection im Automotive Bereich klein ist.
Zum Schluss wird im Abschnitt 4.4 die Problemstellung, mit der sich diese Arbeit beschäftigt, erläutert.

## 3.1 Allgemeiner IDS Vergleich

In der Informatik werden seit vielen Jahren neben Firewalls auch IDS eingesetzt, um mögliche Angriffe an IT-Systeme zu identifizieren und diese zu schützen. In diesem Abschnitt werden Arbeiten, welche gängige Werkzeuge miteinander anhand verschiedener Kriterien vergleichen, vorgestellt. Außerdem spielt die Kategorisierung und ein allgemeiner Vergleich zwischen den verschiedenen Arten der Einbruchserkennung eine wichtige Rolle, denn die möglichen Ansätze der IDS im Bezug auf Performance in der Praxis entscheidend sein können.

### 3.1.1 IDS: A comprehensive review[23]

Die Arbeit von L. Hung-Jen et al. [23] stellt einen allgemeinen und umfassenden Überblick über IDS dar. Der Schwerpunkt liegt bei dieser Arbeit auf die detaillierte Erläuterung der verschiedenen IDS Ansätze und deren Funktionsweisen. Die Abbildung 3.1 visualisiert die Kategorien der IDS Eigenschaften und teilt die Bewertung der IDS in vier Bereiche ein (Abbildung 3.1 gelbe Kästen).

- Bereitstellung des Systems (System Deployment): Dabei wird nach Art der Technologie, des Netzwerks und der Netzwerkarchitektur unterschieden. Die Art der Technologie beschreibt, ob es sich zum Beispiel um ein HIDS, NIDS oder sonstige Einbruchserkennungs-Technologien handelt. Der Netzwerktyp kann entweder „verkabelt“, „drahtlos“, oder „gemischt“ sein. Bei der Netzwerkarchitektur wird definiert, ob es sich um ein zentralisiertes, verteiltes oder hybrides System handelt. Bei einem verteilten System werden mehrere Systeme in einem Netzwerk überwacht und analysiert, wogegen bei einer zentralisierten Netzwerkarchitektur ein IDS für ein einzelnes System eingesetzt wird.

- Datenquelle (Data Source): Dieser Bereich wird in Komponente zur Datensammlung, Art der Datenquelle und Vorgehensweise bei der Sammlung der Daten unterteilt. Mögliche Erfassungskomponenten wären je nach Art der Technologie entweder Sensoren oder Agenten. Dies kann entweder zentralisiert (Agent) oder verteilt (Sensoren) erfolgen. Außerdem ist es relevant welche Art von Daten gesammelt werden sollen, dies können zum Beispiel verschiedene Protokolldaten, Systembefehle oder Logdaten sein.

- Zeitverhalten (Timeliness): Das Zeitverhalten wird anhand der Erkennungszeit, Granularität und der Erkennungsreaktion beschrieben. Der Erkennungszeitpunkt kann entweder in Echtzeit bzw. online oder offline sein. Von diesem Aspekt hängt auch die mögliche Erkennungsreaktion ab. IDS können entweder passiv oder aktiv reagieren, wobei die aktive Reaktion ein online IDS erfordert, da dies einen Eingriff in das Systemverhalten bedeuten kann. Bei einer passiven Erkennungsreaktion werden keine Gegenmaßnahmen ergriffen. Zusätzlich wird beim Zeitverhalten die Granularität betrachtet. Diese beschreibt, in welchen Abständen Daten erfasst werden. Die Zeitgranularität kann zum Beispiel kontinuierlich oder periodisch definiert werden.

- Verfahren zur Einbruchserkennung (Detection Strategy): Die Bewertung anhand der Erkennungsstrategie erfolgt mithilfe der Erkennungsmethodik, Erkennungsdisziplin und der Verarbeitungsstrategie. Bei der Erkennungsmethodik geht es darum, ob es sich um ein Anomalie-, Signatur- oder Spezifikations-basiertes IDS handelt. Außerdem kann die Erkennungsstrategie entweder Zustands- oder Transitionsbasiert sein und die Verarbeitung entweder zentralisiert oder verteilt erfolgen.

Zusätzlich werden in dieser Arbeit die Methodiken, welche die grundsätzliche Funktionsweise IDS wiedergeben, näher erläutert und mithilfe einer tabellarischen Übersicht

verglichen. Außerdem werden die Vor- und Nachteile der Einbruchserkennungsmöglichkeiten anhand der Erkennungsmethodiken aufgeführt. Als Ergänzung werden die Herausforderungen beim IDS Einsatz im Virtual Machine (VM) Kontext thematisiert, da die Netzwerkvirtualisierung, welche virtuelle Netzwerke und somit auch Fehler und Angriffsauswirkungen in einem Netzwerk isoliert, eventuell eine Möglichkeit zur Aufdeckung neuer Sicherheitsschwachstellen bietet und welche Aspekte berücksichtigt werden müssen.
Es werden zwei Werkzeuge, die sich auf Signatur-basierte Einbruchserkennung spezialisieren, vorgestellt: Snort[11] und ClamAV[10], während es sich bei Snort um eins der bekanntesten IDS handelt, ist ClamAV als ein Antiviren-Programm auf dem Markt bekannt. Grundsätzlich befasst sich die Arbeit von L. Hung-Jen et al. [23] nicht mit konkreten Werkzeugen, welche für spezifische Anwendungsfälle ausgewählt werden, sondern beschreibt und analysiert unabhängig vom Einsatzgebiet die unterschiedlichen Optionen im Bereich Einbruchserkennung.

### 3.1.2 Performance Comparison and Detection Analysis in Snort and Suricata Environment[31]

Die Arbeit [31] von W. Park und S. Ahn ist aus der Motivation entstanden, dass IDS aufgrund eines steigenden Vernetzungsgrades der IT-Systeme immer mehr Pakete verarbeiten müssen. Zu dem Zeitpunkt des Papers (2016) ist Snort zwar ein IDS mit einer langjährigen Geschichte, jedoch kann das Werkzeug vielen Anforderungen nicht mehr gerecht werden. Suricata ist ein Werkzeug zur Einbruchserkennung, welches entwickelt wurde unter anderem die Nachteile von Snort auszugleichen. Aus diesen Gründen wird in der Arbeit von W. Park und S. Ahn zwei IDS Werkzeuge - Snort und Suricata - bezüglich der Performance miteinander verglichen. In diesem Fall bedeutet Performance die Erkennungs- und Verarbeitungsrate. Außerdem wird die benötigte Rechenleistung berücksichtigt und mit analysiert. Zuerst werden die Werkzeuge einzeln vorgestellt. Da diese Arbeit im Jahr 2016 veröffentlicht wurde, handelt es sich bei Snort um die Version 2.x. Diese Snort Version arbeitet in einem Single-Threading Verfahren, wobei Suricata in der aktuellen Version das Multi-Threading bereits unterstützt. Außerdem wird Security Onion als ein weiteres Werkzeug, in diesem Fall als ein Framework für Einbruchserkennung, vorgestellt. Security Onion ist eine Linux Distribution, welche verschiedene Tools für IT Sicherheit zur Verfügung stellt [32] und in dieser Arbeit als Umgebung für eine vereinfachte Installation und Konfiguration von Snort und Suricata verwendet wurde.

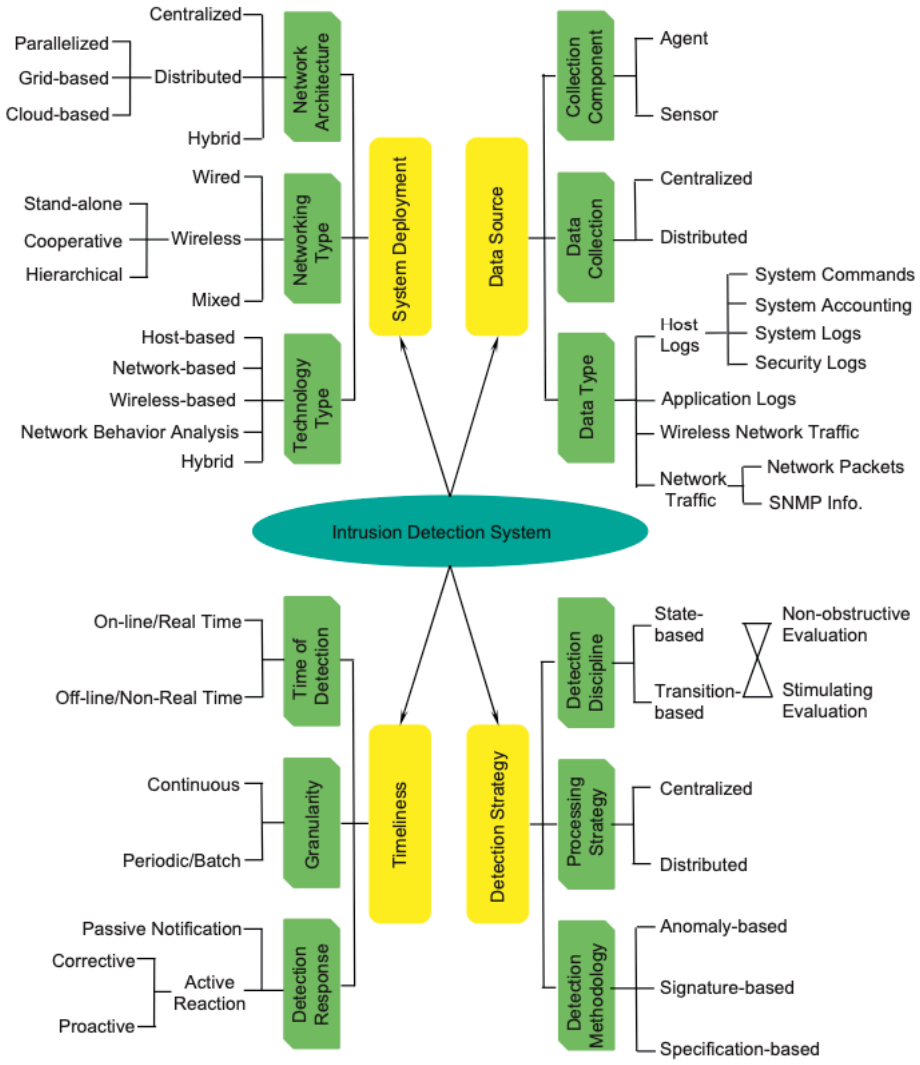


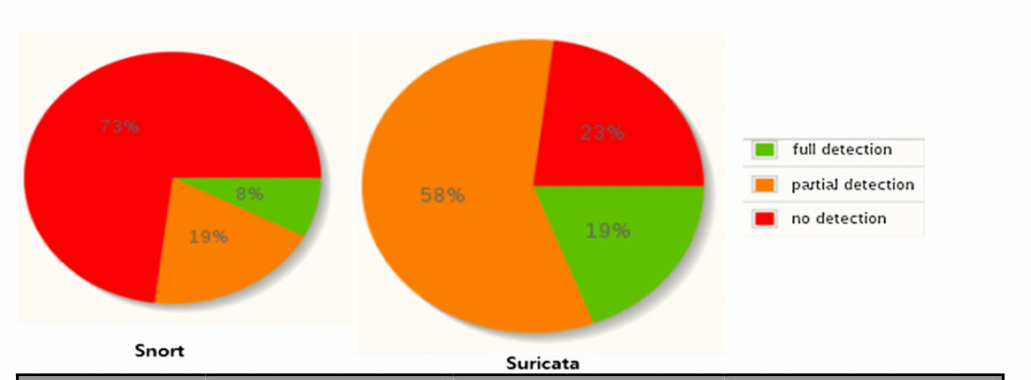

Abbildung 3.1: Übersicht der IDS Taxonomie(Quelle: [23] Abbildung 1)

Um die zwei Werkzeuge zu testen, wurde in dieser Arbeit ein Testing Framework „Pytbull“verwendet, welches speziell dafür gedacht ist IDS und IPS zu testen, indem man zum Beispiel Angriffe simuliert und daraus resultierende Statistiken generiert. In dieser Arbeit wurden folgende Pytbull Module verwendet[31]:

- badTraffic: Das Senden von nicht RFC konformen Paketen, um zu testen, wie diese verarbeitet werden.

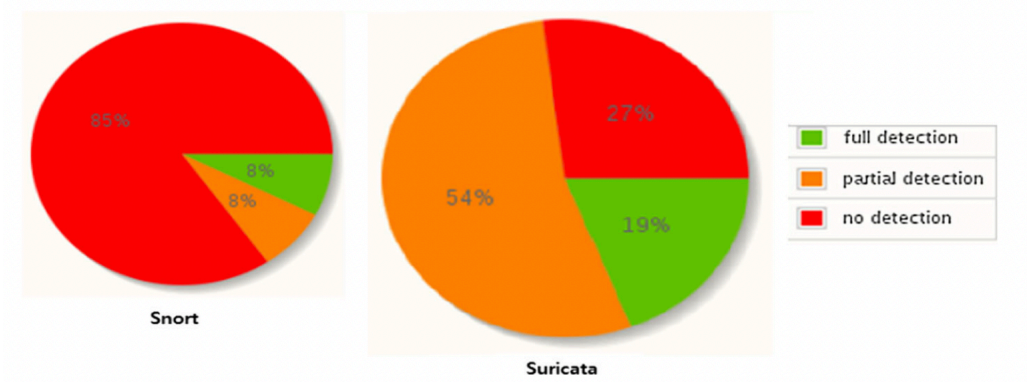
- bruteForce: Das Modul prüft die Erkennung und Verfolgung von Brute-Force-Angriffen.
- denialOfService: DoS ist eine Serverüberflutung mit speziellen Paketen. Dieses Modul testet die Fähigkeit des IDS/IPS vor DoS Angriffsversuchen zu schützen.
- evasionTechinques: Es werden verschiedene verschleierte Angriffe bzw. Techniken verwendet, um zu prüfen, ob das IDS/IPS sie erkennen kann.
- fragmentedPackets: Das Fragmentieren von Angriffen auf kleinere Pakete, um die Fähigkeit zu testen diese wieder zusammen zu setzen und die Angriffe zu erkennen.
- pcapReplay: Dieses Modul prüft, ob eine pcap-Datei rekonfiguriert werden kann.
- shellCodes: Das Senden verschiedener Shellcodes an den Server auf Port 21/tcp, um die Fähigkeit des Servers zu testen diese zu erkennen bzw. abzulehnen.
- testRules: Das Testen der Grundregeln. Diese Angriffe sollen von den mit dem IDS/IPS gelieferten Rule Sets erkannt werden.

Die Performanceanalyse wird in zwei Kategorien eingeteilt: single CPU und multi CPU. Zuerst wurden beide Werkzeuge in einer Single-Core Umgebung mithilfe von Pytbull und den dazugehörigen Modulen analysiert. Die Abbildung 3.2 zeigt eine Übersicht der Ergebnisse von Snort und Suricata in einer Single-Core Umgebung. Die abgebildeten Diagramme sind Ausschnitte aus der Pytbull Auswertung. Es ist deutlich, dass Snort in einer Single-Core-Umgebung nur wenige Angriffe, nämlich 8% vollständig erkannt hat. Dies ist auf die Schwierigkeit die Anzahl der Pakete in einer Single-Core-Umgebung zu verarbeiten zurückzuführen. Wogegen Suricata eine wesentlich höhere Erkennungsrate von 19% aufweist.
Die Abbildung 3.3 zeigt eine Übersicht der Analyse Ergebnisse jeweils von Snort und Suricata in einer Multi-Core Umgebung. Hier ist zu erkennen, dass die Anzahl der nicht erkannten Angriffsversuche bei Snort von 73% auf 85% erhöht hat, während bei Suricata die Verschlechterung nur 4% beträgt. Diese Ergebnisse zeigen, dass Snort nur Single-Threading unterstützt und die Vorteile der Multi-Core Umgebung nicht vollumfänglich nutzen kann. Zusammenfassend lässt sich sagen, dass Suricata die besseren Analyseergebnisse bezüglich der Erkennungsrate und/oder Rechenleistung zu liefert und Snort somit einige Schwachstellen diesbezüglich aufweist. Jedoch kann es sinnvoll sein, bei einem geringen CPU Einsatz, Snort als IDS zu nutzen.




| IDS | High | Medium | Low |
|---|---|---|---|
| Snort | 137 | 165 | 2 |
| Suricata | 298 | 289 | 2212 |

Abbildung 3.2: Übersicht Erkennungsrate von Snort und Suricata in einer Single-Core Umgebung (Quelle: [31] Abbildung 7)




| IDS | High | Medium | Low |
|---|---|---|---|
| Snort | 137 | 166 | 2 |
| Suricata | 297 | 292 | 2220 |

Abbildung 3.3: Übersicht Erkennungsrate von Snort und Suricata in einer Multi-Core Umgebung (Quelle: [31] Abbildung 10)

## 3.2 IDS in Fahrzeugnetzwerken

Die bekanntesten Werkzeuge zur Einbruchserkennung auf dem Markt wurden für klassische IT-Systeme entwickelt und waren somit nicht für den Einsatz im Bereich Fahrzeugnetzwerke gedacht. Jedoch ist im Laufe der Jahre der Bedarf nach Sicherheitsmechanis-

men gegen potenzielle Angriffe gestiegen[24]. In diesem Abschnitt werden zwei Arbeiten, die sich mit IDS im Automotive Umfeld beschäftigen, vorgestellt.

### 3.2.1 Ein Spezifkations-basierter IDS Ansatz im Automotive Ethernet [7]

Bei der Arbeit von F. Baumann [7] handelt es sich um eine Evolution von einem neuen Spezifikations-basierten IDS Ansatz, der neben der Einbruchserkennung vor allem einen möglichst geringen Ressourcenverbrauch als Ziel definiert. Es werden keine weiteren Verfahren oder auf dem Markt verfügbaren IDS evaluiert. Da es sich um ein Spezifikationsbasiertes Verfahren handelt, wird eine Spezifikation für das normale Verhalten in Form einer Whitelist festgelegt. Folgende Daten können mögliche Parameter in der Spezifikation für die Einbruchserkennung darstellen: IP-Adressen, Payloadlängen, Protokollflags oder andere protokollspezifische Parameter. In der Arbeit evaluiert F. Baumann zusätzlich zwei Hash-Verfahren, nämlich den Bloom- und Cuckoo-Filter, die für die Implementierung des IDS verwendeten Algorithmus relevant sind, da diese unterschiedlichen Ressourcenverbrauch aufweisen. Die Hash-Verfahren werden für die Whitelist Speicherung der Spezifikationen bzw. der Spezifikations relevanten Daten verwendet. Letztendlich wird auf den Cuckoo-Filter zurückgegriffen, da dieser im praktischen Einsatz bessere Performance erzielt.
Auch wenn die Arbeit sich mit dem IDS Einsatz im Automotive Ethernet Umfeld auseinander setzt, handelt es sich um eine prototypische Implementierung, die nicht in einem Fahrzeugumfeld evaluiert wird. Stattdessen wird ein Raspberry Pi 3 als Testsystem verwendet, da die Rechenleistung ähnlich wie bei den Steuergeräten im Fahrzeug ist. Als eine Simulation des Netzwerkverkehrs wurde eine mittgeschnittene PCAP-Datei als Testdatensatz, welcher anomales Verhalten simuliert, verwendet. Die Testdaten beinhalten zwei Automotive-spezifische Protokolle: Scalable Service-Oriented Middleware over IP (SOME/IP) und Diagnostics over IP (DoIP).
Grundsätzlich lässt sich sagen, dass F. Baumann in [7] die mögliche Verwendung einer Spezifikations-basierten Einbruchserkennung im Automotive Ethernet anhand der Testergebnisse validiert und einen Ausblick für weitere Umsetzung gibt, wobei alternative Lösungen nicht berücksichtigt werden.

### 3.2.2 Intrusion detection system for automotive Controller Area Network (CAN) bus system: a review [24]

Die Arbeit von Lokman et al. [24] ist eine Überprüfung und Zusammenfassung darüber, welche Schwachstellen CAN als fahrzeuginternes Kommunikationsprotokoll aufweist und welche Angriffe auf das Netzwerk möglich wären. Außerdem wird IDS als eine Sicherheitslösung in einem CAN-basierten Fahrzeugnetzwerk diskutiert. Dies wird im Bezug auf folgende Aspekte untersucht:

- Erkennungsansätze
- Einsatzstrategien
- Angriffstechniken
- CAN-spezifische, technische Herausforderungen

In der Arbeit wird auf mögliche Schwachstellen des Protokolls hingewiesen. Die Schwachstellen von CAN basieren darauf, dass die CAN-Pakete grundsätzlich keine Information über die Sender- und Empfängeradresse enthalten. Die Pakete werden anhand des Arbitration Identifier Field an alle CAN-Knoten gesendet. Dadurch, dass die Herkunft der Pakete nicht angegeben wird, können die Empfänger nicht erkennen, ob die Pakete für sie bestimmt sind. Somit kann auch nicht festgestellt werden, ob die empfangenen Pakete legitim sind. Außerdem kommt es vor, dass die Steuergeräte über keine Mechanismen zur Nachrichten Authentifizierung, um die zu übertragenen Pakete entsprechend zu sichern, verfügen. Solche Steuergeräte können potenziell genutzt werden, um gefälschte Pakete zu versenden.
Des Weiteren werden potenzielle Angriffsmöglichkeiten auf ein Fahrzeugnetzwerk vorgestellt. Durch die Überflutung des CAN-Busses mit einer großen Anzahl an gefälschten CAN-Paketen wird ein Angriff auf verschiedene Bereiche des Fahrzeugs ermöglicht. So kann die Tachometeranzeige manipuliert, die Tür blockiert oder der Motor abgeschaltet werden. [24] Als eine weitere Angriffsmöglichkeit wird das Senden manipulierter CAN-Pakete aus der Ferne vorgestellt. Dadurch können kritische Komponenten im Fahrzeug wie zum Beispiel das Bremssystem bzw. Lenksystem deaktiviert oder kontrolliert werden.
Der Einsatz von IDS in CAN-basierten Fahrzeugnetzwerken bringt folgende Herausforderungen mit sich[24]:

- Begrenzte Ressourcen: Die Steuergeräte verfügen über stark eingeschränkte Speicherkapazität, Rechenleistung und Bandbreite.

- Echtzeitanforderungen: Dadurch, dass die Kommunikation zwischen den Knoten in Echtzeit erfolgt, können mögliche Verzögerungen nicht toleriert oder benötigte Puffermechanismen umgesetzt werden. Dies ist erforderlich um die Fahrzeugfunktionen korrekt und in Echtzeit ausführen zu können.
- Verkehrsverhalten: Die Muster des CAN-Protokolls bei Fahrzeugnetzwerken unterscheiden sich von denen herkömmlicher Internetprotokoll Netze. Ein Beispiel hierfür ist die Übertragung der CAN-Pakete per Broadcast.
- Instabile Konnektivität: Durch die wechselnde geographische Lage der Fahrzeuge kann eine stabile Internetverbindung nicht immer gewährleistet werden. Dieser Aspekt ist bei der IDS zu berücksichtigen, da gegebenenfalls ein benötigter Verbindugsaufbau nicht möglich ist.

Des Weiteren werden verschiedene Ansätze von IDS vorgestellt: Anomalie-basiert, Signatur-basiert, Spezifikations-basiert und hybrid. Diese Erkennungsmethoden werden in Unterabschnitt 2.1.4 erläutert.

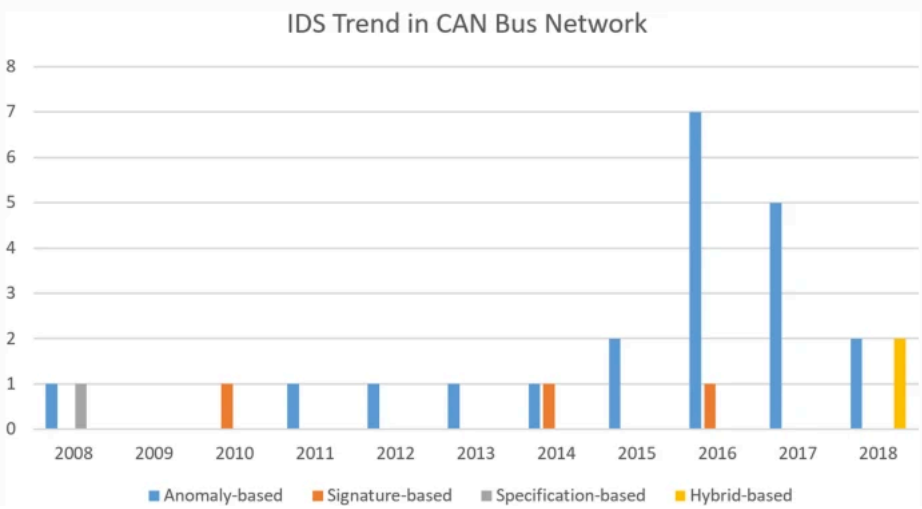



Abbildung 3.4: IDS Trend in CAN-basierten Netwerken (Quelle: [24] Abbildung 5)

Die Abbildung 3.4 veranschaulicht die Entwicklung darüber, welche IDS Ansätze in CAN-basierten Fahrzeugnetzwerken über die Jahre 2008 bis 2018 eingesetzt wurden. Es ist deutlich, dass Anomalie-basierte IDS am populärsten im CAN Umfeld waren. Die Untersuchung in der Arbeit von Lokman et al. [24] begründet den sehr häufigen Einsatz von Anomalie-basierten Verfahren damit, dass sowohl signatur- als auch Spezifikations-basierte IDS semantisches Wissen über CAN-Pakete erfordern. Dagegen stellt die Anomalie-

basierte Methodik aufgrund der Anpassungsfähigkeit und Flexibilität im Bezug auf Protokolle und Semantik eine geeignete Schutzmethode für Fahrzeugnetzwerke dar. Zusätzlich beinhaltet das Erkennen von anomalen Verhalten auch neuartige Angriffe, welche die Signatur- und Spezifikations-basierten IDS ohne zusätzliche Updates nicht berücksichtigen können.

## 3.3 Implikationen

Bei der Suche nach verwandten Arbeiten im Automotive Umfeld ist es deutlich geworden, dass CAN-Busse seit vielen Jahren ein Thema ist. Dagegen haben Ethernet-basierte Architekturen in Fahrzeugnetzwerken einen geringen Anteil. Dass die Netzwerksicherheit vor allem bei globaler Vernetzung eine große Rolle spielt, ist bereits deutlich geworden. Die CoRE Arbeitsgruppe forscht viel im Bereich Anomalieerkennung und Netzwerksicherheit in Fahrzeugnetzwerken. Eine Anomalie-basierte Einbruchserkennung muss immer den Normalzustand des Systems beziehungsweise Netzwerkes kennen, um anomales Verhalten festzustellen. Dabei kann es dazu kommen, dass entweder Anomalien nicht erkannt werden (False Negatives) oder als solche nicht gewertet werden, obwohl es sich um Anomalien handelt (False Postives). Die Anzahl der falschen Ergebnisse kann je nach Algorithmus oder Methode der Anomalieerkennung variieren. Bereits die Möglichkeit, dass es falsche Ergebnisse geben kann, erfordert ein Konzept, welches die konkrete Herangehensweise für die Verarbeitung und Behandlung solcher Werte definiert.
Vor allem Signatur-basierte Intrusion Detection hat den Vorteil, dass tatsächlich verdächtiges Verhalten als solches erkannt wird, sofern es in der IDS Datenbank oder als Regel definiert ist. Die Anzahl der False Positives und False Negatives ist vergleichsweise gering, weil die Signatur-basierte Einbruchserkennung nicht den Normalzustand mit dem Ist-Zustand vergleicht, sondern einen generischen Ansatz, der bereits bekannte Szenarien mit dem Ist-Zustand wie in Unterabschnitt 2.1.4 beschrieben abgleicht, implementiert.
Die Tatsache, dass die IDS Werkzeuge nicht für den Einsatz in Fahrzeugnetzwerken entwickelt wurden, stellt eine Herausforderung dar. Anders als in herkömmlichen Netzwerken spielen weitere Aspekte wie Echtzeitanforderungen oder begrenzte Ressourcen eine ausschlaggebende Rolle.

# 4 Ist Analyse

Dieser Abschnitt befasst sich mit der Ausgangssituation bezüglich den aktuellen Stand der verwendeten Technologien, die für diese Arbeit relevant sind. Zuerst werden im Abschnitt 4.1 IDS Tools vorgestellt und grob kategorisiert. Dies dient als eine Vorbereitung für die spätere Bewertung und Auswahl. Weiter im Abschnitt 4.2 wird der aktuelle Stand der Topologien und Architekturen in Bezug auf Fahrzeugnetzwerke erläutert und im Abschnitt 4.3 in die für diese Arbeit relevante Versuchsumgebung eingeordnet.

## 4.1 IDS Tools

In diesem Kapitel wird der aktuelle Stand an IDS Tools auf dem Markt vorgestellt. Es werden sowohl Netzwerk- aber auch Host-basierte IDS und deren wichtigsten Eigenschaften erläutert.

### 4.1.1 Übersicht der IDS auf dem Markt

Eins der wichtigsten Ziele dieser Arbeit besteht darin, etablierte Werkzeuge zur Einbruchserkennung zu evaluieren. Der erste Schritt der Marktrecherche basiert auf einer Internetsuche. Für die erste Übersicht wurden alle gefundenen Tools berücksichtigt, die erste Auswahl für die Evaluation wird im Abschnitt 5.6 getroffen. In der Tabelle 4.1 sind die recherchierten Tools aufgelistet. Außerdem werden diese Tools entsprechend kategorisiert. Grundsätzlich gibt es zwei Möglichkeiten die IDS zu clustern:

- Anhand des Überwachungsziels (Host oder Netzwerk)
- Anhand der Funktionsweise (Signatur- oder Anomaliebasiert)

| | Open-Source | IPS | Kategorie | Ab* | Sb** | Letzte Aktualisierung |
|---|---|---|---|---|---|---|
| **Snort[11]** | Ja | ✓ | NIDS | | ✓ | aktuell |
| **Security Onion[32]** | Ja | ✓ | NIDS | | ✓ | aktuell |
| **OSSEC[30]** | Teilweise | ✓ | HIDS | ✓ | ✓ | aktuell |
| **Suricata[27]** | Ja | ✓ | NIDS | | ✓ | aktuell |
| **Zeek[38]** | Ja | | NIDS | ✓ | ✓ | aktuell |
| **IBM Security QRadar NDR[16]** | Nein | ✓ | NIDS | ✓ | ✓ | aktuell |
| **OpenWIPS-NG[2]** | Ja | ✓ | Wireless IDS | | ✓ | 2012 |
| **Sagan[33]** | Ja | | Log-Analyse | | ✓ | aktuell |
| **NetRanger[9]** | Nein | | NIDS | | ✓ | 2005 |
| **Haystack[34]** | Nein | | HIDS | ✓ | | 1988 |
| **NetStat[39]** | Prototyp | | NIDS | | ✓ | 1999 |

Tabelle 4.1: Rechercheergebnis IDS-Tools auf dem Markt
*Anomalie-basierte Funktionsweise **Signatur-basierte Funktionsweise

Zusätzlich wird in der Tabelle 4.1 die Information, ob es sich um ein Open-Source oder um ein propriotäres Tool handelt, angegeben. Außerdem bieten einige der Tools neben Intrusion Detection auch die Möglichkeit der Intrusion Prevention an. In der Tabelle wird dies ebenfalls in der Spalte „IPS“ gekennzeichnet.

**Netzwerk-basierte IDS**

Es gibt auf dem ersten Blick eine Vielzahl von NIDS auf dem Markt. Die bekanntesten davon sind Snort von Cisco und Suricata von Open Information Security Foundation (OISF). Zeek (ehemals Bro) ist die weltweit führende Plattform für die Überwachung der Netzwerksicherheit und bietet ebenfalls ein integriertes NIDS. Im Bereich der proprietären Software gibt es die International Business Machines Corporation (IBM) als Anbieter für eine Lösung mit integrierter Intrusion Detection: IBM Security QRadar. Diese ist aber genau so wie die open-source Lösung Security Onion von Security Onion Solutions eine Sammlung verschiedener Tools für IT-Sicherheitsmanagement. Der Einbruchserkennungsteil bei Security Onion wird von Suricata zur Verfügung gestellt.
Bei der Recherche wurde festgestellt, dass einige der Tools zwar in der Entwicklungsphase oder bereits auf dem Markt waren, sich aber nicht durchsetzen konnten. So wird zum Beispiel NetRanger von Cisco seit 2000 nicht mehr vermarktet und seit 2005 nicht mehr unterstützt. Stattdessen wird auf Snort verwiesen. Zu dem Intrusion Detection Tool NetStat gibt es seit 2003 keine neuen Publikationen mehr.

**Host-basierte IDS**

Auf dem Markt der Host-basierten IDS ist die Auswahl nicht so zahlreich wie bei NIDS. OSSEC ist ein Open-Source HIDS, welches aber auch kostenpflichtige Erweiterungen für kommerzielle Zwecke zur Verfügung stellt. Es ist ein aktuelles Tool, welches sowohl für einzelne Hosts, als auch ganze Server Landschaften, aber auch für Cloud Anwendungen geeignet ist.
Bei der Recherche ist es aufgefallen, dass es nicht viele Alternativen im Bereich HIDS existieren. Haystack ist zum Beispiel ein Tool, welches Ende der achtziger Jahre für den Einsatz in Militär Systemen entwickelt wurde.

**Sonstiges**

Neben den klassischen NIDS und HIDS gibt es auch andere Ansätze für Intrusion Detection Lösungen. So bietet zum Beispiel Sagan eine Log-Analyse Engine, die eine ähnliche Funktionsweise wie IDS und IPS aufweist. Der Unterschied zu IDS liegt darin, dass nicht der Netzwerkverkehr sondern die Logs unterschiedlicher Quellen überwacht werden [33]. Aber auch hierbei handelt es sich um eine regelbasierte Einbruchserkennung, ähnlich wie bei den Signatur-basierten IDS Tools.
Außerdem gibt es das Tool OpenWIPS-NG, welches speziell für drahtlose Netzwerke entwickelt wurde. Bei OpenWIPS-NG handelt es sich um ein aus drei Komponenten bestehendes IPS Tool[2]:

- Sensor: Erfassung und Weiterleiten des Netzwerkverkehrs an den Server
- Server: Analyse der Sensordaten, Reaktion auf Angriffe
- Graphical User Interface (GUI): Informationsdarstellung über Angriffe an den Benutzer, Serververwaltung

Beide Tools sind zwar als Open-Source Lösungen verfügbar, jedoch wurde bei OpenWIPS-NG die offizielle Website seit 2012 nicht mehr aktualisiert und bei dem verfügbaren Download handelt es sich um eine 0.1 Beta Version. Dagegen wird das GitHub Repository [1] von Sagan regelmäßig aktualisiert.

## 4.2 Fahrzeugnetzwerke

In diesem Kapitel werden verschiedene Netzwerktopologie Ansätze, also die möglichen Anordnungen zwischen den Teilnehmern im Netzwerk, für Fahrzeugnetzwerke vorgestellt. Es wird sowohl der feldbusbasierte Ansatz in früheren Serienfahrzeugen, als auch die Topologie Evolution, also die Entwicklung von einer busbasierten bis hin zu einer switchbasierten Topologie, für die Fahrzeugnetzwerke der Zukunft vorgestellt. Vor allem werden die Aspekte der Domänen-Zonen-Topologie im Abschnitt 4.2.2 erläutert.

### 4.2.1 Evolution: Topologien in Fahrzeugnetzwerken

Ursprünglich wurde in Serienfahrzeugen eine feldbusbasierte, meist CAN-basierte, Topologie umgesetzt, dies wird in der Abbildung 4.1 veranschaulicht. CAN ist ein serielles Bussystem, welches für die Kommunikation zwischen verschiedenen Steuergeräten in Fahrzeugnetzwerken eingesetzt wird. Die Steuergeräte des Fahrzeugs sind in unterschiedliche Anwendungsdomänen unterteilt. Zu Beginn haben die Anwendungsdomänen in Serienfahrzeugen voneinander getrennte Feldbusse. Für die domänenübergreifende Kommunikation wird ein zentrales Gateway, welches die Nachrichten von dem Quellbus an das Zielbus weiterleitet, benötigt. Einige Steuergeräte, die vor allem einen hohen Bandbreitenbedarf aufweisen, haben von den Anwendungsdomänen entkoppelte Feldbusse, sogenannte private Busse und kommunizieren nicht über das zentrale Gateway.
Dieser feldbusbasierte Ansatz einer Netzwerktopologie besitzt eine beschränkte Flexibilität, da die Integration neuer Steuergeräte eine Abhängigkeit von den physikalischen Eigenschaften der Anwendungsdomänen aufweisen. So kann es zum Beispiel bei neuen Steuergeräten der Fall sein, dass an der Einbauposition keine Busse der benötigeten Anwendungsdomäne vorhanden sind und somit neu verlegt werden müssen. Außerdem muss die Kapazität des zentralen Gateways bei steigendem Vernetzungsgrad berücksichtigt werden [36]. Um dieser Problematik entgegenzuwirken wurde die herkömmliche Topologie in die Domänen-Gateway-Topologie überführt.

Die Domänen-Gateway-Topologie stellt eine höhere Evolutionsstufe der CAN-basierten Netwerktopologie in Serienfahrzeugen dar. Dabei wird das zentrale Gateway durch mehrere Domänen-Gateways abgelöst. Wie in der Abbildung 4.2 dargestellt wird die Gateway Funktionalität von einem Steuergerät der Anwendungsdomäne übernommen. Diese Topologie hat den Vorteil gegenüber der vorherigen Evolutionsstufe, dass die Last über

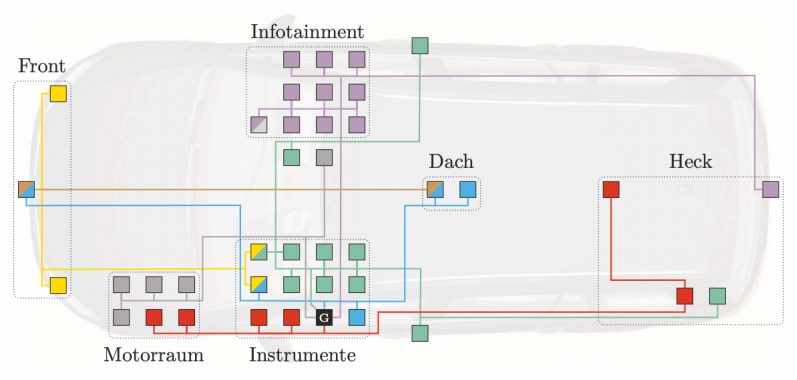



Abbildung 4.1: Feldbusbasierte Topologie in Serienfahrzeugen - Farben repräsentieren Anwendungsdomänen, Gateway (G) in schwarz, mehrfarbige Steuergeräte sind ohne Gateway Funktionalität an mehreren Bussen angebunden (Quelle: [36] Abbildung 3-8)

mehrere Gateways verteilt werden kann. Die Steuergeräte, vor allem die mit einer hohen Bandbreite haben die Möglichkeit die Daten an mehrere Domänen Gateways des Fahrzeugs parallel zu senden [36]. Jedoch wirkt diese Topologie nicht dem Problem der physikalischen Abhängigkeit entgegen, da die Verortung der Feldbusse immernoch berücksichtigt und gegebenenfalls neue Busse verlegt werden müssen.

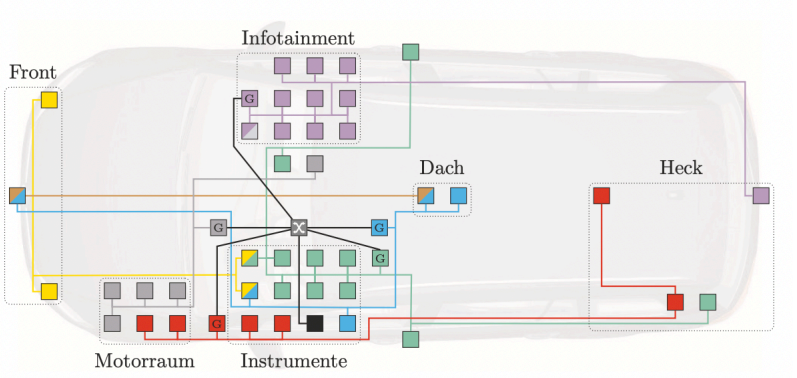



Abbildung 4.2: Domänen-Gateway-Topologie - Vormals zentrales Gateway (schwarz) wird ersetzt durch mehrere Domänen-Gateways (G) und Ethernet Backbone, mehrfarbige Steuergeräte sind ohne Gateway-Funktionalität an mehreren Bussen angebunden (Quelle: [36] Abbildung 3-9)

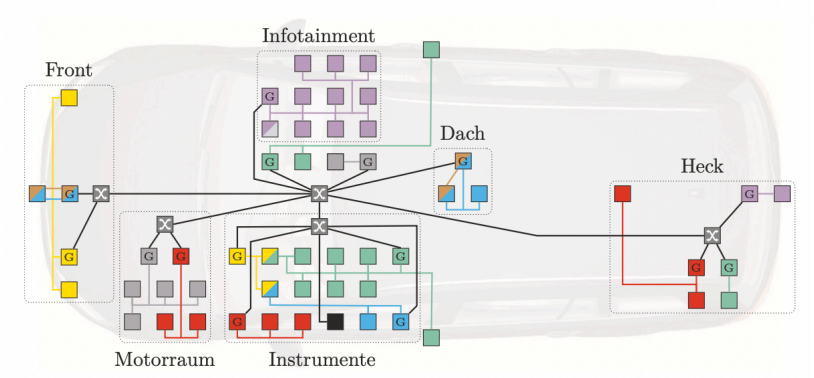



Abbildung 4.3: Domänen-Zonen-Topologie - Ursprüngliche Feldbusse werden zerteilt und lokal über Gateways (G) an den Ethernet Backbone angebunden; mehrfarbige Steuergeräte und Gateways besitzen Funktionalität verschiedener Domänen (Quelle: [36] Abbildung 3-10)

### 4.2.2 Domänen-Zonen-Topologie

Durch die immer steigenden Anforderungen bezüglich der Datenübertragung in Fahrzeugnetzwerken geht die Entwicklung im Bereich Topologien immer mehr Richtung Ethernet-basierte Kommunikation [15]. Die nächste Stufe nach der Domänen-Gateway-Topologie stellt die Domänen-Zonen-Topologie dar. Diese Netzwerkarchitektur verringert den Kommunikationsanteil über Busse und fördert einen Übergang zu Ethernet basierter Kommunikation. Die Domänenbusse werden in Zonen aufgeteilt, wobei eine Zone mehrere Anwendungsdomänen beinhalten kann. Die Kommunikation innerhalb der Zonen erfolgt weiterhin über Busse. Dabei erhält jede Anwendungsdomäne innerhalb einer Zone, wie in der Abbildung 4.3 dargestellt, ein eigenes Gateway. Diese Aufteilung ermöglicht eine Kommunikation über Ethernet Backbone innerhalb einer Domäne. Außerdem ist es durch die Domänen-Zonen-Topologe nicht mehr notwendig beim Einbau neuer Steuergeräte die Verortung der Busse in dem Ausmaß wie bei den vorangegangenen Topologie Stufen zu berücksichtigen. Diese müssen nicht zwingend an die eigene Anwendungsdomäne, sodass die Busse möglicherweise das gesamte Fahrzeug umspannen, angebunden werden [36].

### 4.2.3 Zonenarchitektur

Ein möglicher Ansatz für eine zukünftige Architektur für Fahrzeugnetzwerke stellt die Zonenarchitektur dar. Diese basiert auf den Ansatz der Domänen-Zonen-Topologie, teilt

aber das Fahrzeugnetzwerk in physikalische Zonen ein. Dabei werden die Anwendungsdomänen durch Zonen vollständig abgelöst, sodass die Steuergeräte direkt an Zonen Controller, wie in der Abbildung 4.4 veranschaulicht, angebunden sind[15]. Die Anbindung der Steuergeräte hängt somit nicht mehr von der jeweiligen Domäne, sondern von der räumlichen (z.B. vorne links) Nähe zu dem Zonen Controller ab.

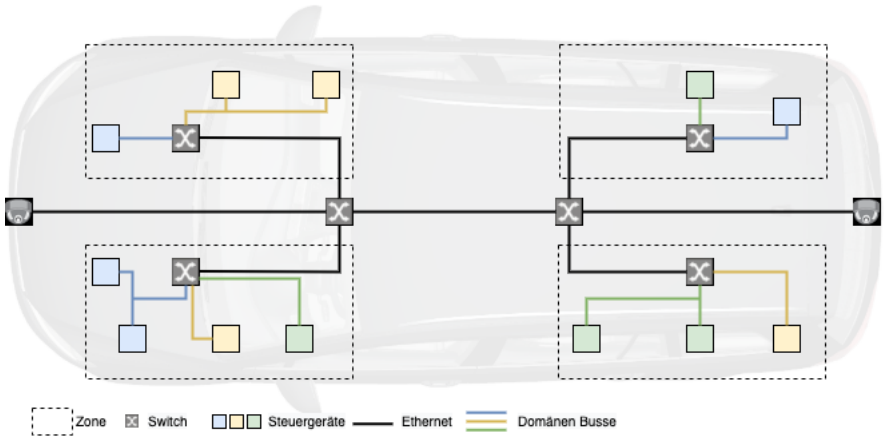



Abbildung 4.4: Zonenarchitektur (Quelle: [15] Abbildung 1-c)

### 4.2.4 Zusammenfassung

Die im Unterabschnitt 4.2.2 und Unterabschnitt 4.2.3 vorgestellten Ansätze stellen eine denkbare Möglichkeit für die Umsetzung der zukünftigen Fahrzeugnetzwerke dar. Diese Architekturansätze bieten zahlreiche Chancen, um die Herausforderungen an die Kommunikation in Fahrzeugnetzwerken angehen zu können. So kann die Zonenarchitektur durch die verteilte Rechenleistung eine bedarfsgerechte Nutzung ermöglichen. Auch die Tendenz zur Ablösung der CAN Busse durch Ethernet-basierte Kommunikation bringt zahlreiche Vorteile mit sich, darunter die Flexibilität im Bezug auf die Komponenten. Neue Komponenten können bei zukünftigen Architekturen problemlos in das bestehende Fahrzeugnetzwerk integriert und umgekehrt genauso unproblematisch entfernt werden. Dies wird zusätzlich durch Technologien wie das SDN unterstützt.
Aber auch wenn Probleme wie Skalierbarkeit und zuverlässige Kommunikationsmechanismen mit zukünftigen Architekturen angegangen werden können, so darf man die dabei entstehenden Risiken nicht vernachlässigen. Dadurch, dass immer mehr Funktionalitäten mithilfe von Software, angefangen bei einzelnen Steuergeräten bis hin zu SDN, umgesetzt

werden, steigt die Anfälligkeit für Bedrohungen durch eine vergrößerte Angriffsfläche und mögliche Sicherheitslücken in Fahrzeugnetzwerken.

## 4.3 SecVI Demonstrator

In Kapitel 5 werden die IDS Anforderungen für den Einsatz im Fahrzeugnetzwerk basierend auf dem SecVI Demonstrator definiert und dementsprechend verschiedene Werkzeuge ausgewählt. Anschließend folgt eine Evaluation im Kapitel 6. In diesem Kapitel wird im Unterabschnitt 4.3.1 der SecVI Demonstrator vorgestellt und im Unterabschnitt 4.3.2 die entsprechende Netzwerktopologie näher erläutert.

### 4.3.1 Allgemein

Ethernet-basierte Fahrzeugnetzwerke sind die Zukunft der Fahrzeugnetzkommunikation. Sie bilden die Grundlage für viele neue Technologien wie IoT-Dienste, fortschrittliche Assistenzsysteme und sogar autonomes Fahren [26].
Bei dem SecVI Demonstrator handelt es sich um einen Hardwareaufbau, der ein internes Fahrzeugnetzwerk und die im Unterabschnitt 4.2.2 vorgestellte Domänen-Zonen-Topologie darstellt. Der Aufbau basiert auf dem Seat Ateca, welcher in der Abbildung 4.5 dargestellt ist. Darin werden Funktionen wie Überwachung, Erkennung und Verwaltung von Vorfällen und entsprechende Gegenmaßnahmen umgesetzt und erforscht. Die Implementierung beinhaltet SDN, Anomalieerkennung, sichere Gateways, Cloud-Dienste und weitere Technologien [26]. Bei der Evaluation wird die Relevanz der IDS für die genannten Technologien geprüft. Zusätzlich wird die bereits vorhandene Anomalieerkennung berücksichtigt, da die zu evaluierenden Werkzeuge zur Einbruchserkennung gegebenenfalls eine mögliche Erweiterung oder Alternative darstellen können.

### 4.3.2 Topologie

In der Abbildung 4.6 ist die Netzwerktopologie des SecVI Demonstrators visualisiert. Es handelt sich um eine Domänen-Zonen-Topologie mit einem Openflow basierenden (siehe **??**)SDN Switch, welcher als Ethernet-Backbone des Fahrzeugnetzwerks fungiert.


Abbildung 4.5: 2016' Seat Ateca Prototyp (Quelle: [26] Abbildung 1a)

Dieser wird von dem SDN Controller, der das Weiterleiten von Paketen anhand der Weiterleitungstabellen steuert, verwaltet. Die Steuergeräte des Fahrzeugs sind in fünf Anwendungsdomänen, die jeweils einen CAN-Bus für die interne Kommunikation innerhalb der Domäne beinhalten, unterteilt. Die Funktionen, die eine Kombination aus mehreren Domänen darstellen, benötigen ein zentrales Gateway für die Domänen übergreifende Kommunikation [26].
Die Topologie ist in vier Zonen eingeteilt: vorne rechts, vorne links, hinten rechts und hinten links. In diesen Zonen fungiert jeweils ein Zonen Kontroller als Gateway zwischen den CAN-Geräten und dem Ethernet-Backbone. In der Abbildung 4.6 haben die Zonen Gateways folgende Bezeichnungen: ZC_FL, ZC_FR, ZC_RR und ZC_RL [26].
Außerdem befinden sich in diesem Fahrzeugnetzwerk ein Monitor und eine Kamera, welche über eine Ethernet-basierte Kommunikation Daten an den Monitor sendet. Die Anomalieerkennung dieses Demonstrators wird über die Network Anomaly Detection System (NADS) 1 und 2 durchgeführt.

## 4.4 Problemstellung

Das Ziel dieser Arbeit ist es bereits bestehende NIDS Werkzeuge auszuwählen, diese bezüglich Anforderungen in Fahrzeugnetzwerken zu untersuchen und anschließend zu evaluieren. Um eine Vergleichbarkeit herzustellen wird der Fokus auf einen Signatur-basierten Ansatz gelegt, da in der Tabelle 4.1 deutlich wird, dass diese Eigenschaft eine Gemeinsamkeit der IDS darstellt. Im Laufe dieser Arbeit wird untersucht, ob die ausgewählten

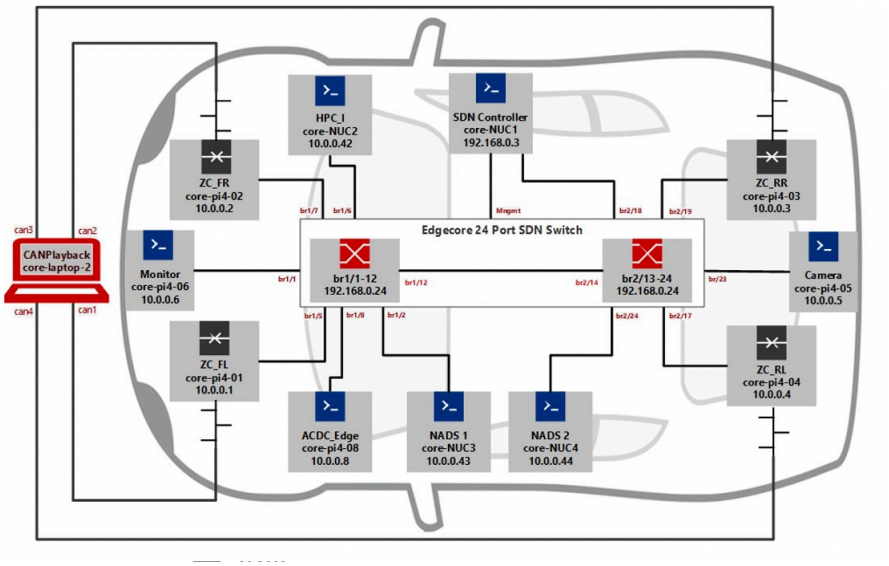



Abbildung 4.6: Topologie des SecVI Demonstrators

Werkzeuge zur Einbruchserkennung sowohl für den Einsatz auf dem SecVI Demonstrator, als auch für einen allgemeinen Einsatz in Fahrzeugnetzwerken sinnvoll wären. Um die einzelnen IDS zu evaluieren, wird zuerst eine Marktrecherche mit einem anschließenden Auswahlprozess stattfinden. Darauf folgend werden die ausgewählten Werkzeuge tiefergehend vorgestellt.
Die Schwierigkeit besteht darin, die Angriffe zu erkennen, welche zu dem Netzwerkverkehr nicht passen, also in Bezug auf Merkmale wie Paketinhalte, Paketgröße und Bandbreite unauffällig sind. Das Erkennen der Angriffe, welche auffällige Merkmale aufweisen sind grundsätzlich deutlich leichter zu entdecken. Auß müssen fahrzeugspezifische Protokolle und Anwendungsfälle beachtet werden. Zusätzlich spielen Metriken wie False Positives und False Negatives bei der Evaluation eine Rolle, da es beim praktischen Einsatz mit gegebenenfalls gravierenden Folgen verbunden wäre. Es wird im Laufe der Arbeit geprüft, inwiefern die ausgewählten IDS diese Aspekte berücksichtigen und umsetzen können. Der aktive Eingriff in das Netzwerk als Reaktion auf erkannte Angriffe ist kein Teil dieser Arbeit. Es geht lediglich um die Feststellung, ob die Nutzung der ausgewählten IDS innerhalb eines Fahrzeugnetzwerks perspektivisch sinnvoll wäre, ob es sich um eine Alternative zu Anomalieerkennung oder gegebenenfalls eine zusätzliche Ergänzung handelt.

# 5 Anforderungsanalyse

In diesem Kapitel werden anhand einer Anforderungsanalyse Kriterien an ein Intrusion Detection System definiert. Die Anforderungen werden in fünf Kategorien eingeteilt: Anforderungen an das System (A), an die Datenverarbeitung (B), an das Zeitverhalten (C) und sonstige technische (D) bzw. nichttechnische (E) Anforderungen. In diesem Kapitel sind die Anforderungskategorien jeweils in einer Tabelle zusammengefasst und in den Abschnitten 5.1 bis 5.5 näher erläutert. Die Kategorisierung der Anforderungen ist an die Arbeit von L. Hung-Jen et al. [23] angelehnt. Diese Kategorien finden sich ebenfalls in der Abbildung 3.1 wieder.
Jede Anforderung hat eine Priorität. Diese gibt an, wie ausschlaggebend die Anforderung für die Auswahl eines IDS ist. Die Priorität kann niedrig, mittel oder hoch sein. Eine niedrige Priorität beeinträchtigt die Funktionalität des IDS nicht, bietet aber beim Einsatz gewisse Vorteile oder erweitert das IDS mit sinnvollen Features. Die Prioritätsstufe „mittel“ impliziert, dass die Anforderung unter bestimmten Voraussetzungen vernachlässigt werden darf. Dies darf das Systemverhalten nicht negativ beeinflussen. Hohe Prioritäten sind ausschlaggebend für die Auswahl eines IDS. Wenn eine Anforderung mit hoher Priorität nicht erfüllt werden kann, ist das IDS für die Evaluation ungeeignet.
Nach der Anforderungsanalyse folgt eine Vorauswahl der Werkzeuge. Dies stellt die erste Stufe des zweistufigen Auswahlprozesses dar. Die Werkzeuge, welche im Abschnitt 4.1 vorgestellt wurden, werden anhand der typischen Eigenschaften geprüft und für die spätere Evaluation ausgewählt. Außerdem wird im Abschnitt 5.6 die Vorauswahl für das Fahrzeugnetzwerk erläutert. Die Vorauswahl basiert auf die in diesem Kapitel beschriebenen Anforderungen mit der Priorität „hoch“, welche die erforderlichen Anforderungen an ein IDS definiert. Dies hat zur Folge, dass nur die Werkzeuge, welche diese Anforderungen erfüllen, in den darauffolgenden Kapiteln evaluiert werden.

| ID | Anforderung | Priorität | Beschreibung |
|---|---|---|---|
| **A1** | NIDS | hoch | Netzwerk-basiertes IDS |
| **A2** | Verteiltes System | hoch | Topologie besteht aus mehr als einem Computer |
| **A3** | Wired | hoch | Kabel als Übertragungsmedium im Netzwerk |

Tabelle 5.1: Anforderungen an Systemeigenschaften

## 5.1 Systemeigenschaften

Die Anforderungen an die Systemeigenschaften sind in der Tabelle 5.1 dargestellt. Das Ziel des IDS ist mögliche Hinweise auf Einbrüche in ein Fahrzeugnetzwerk zu entdecken. Somit stellt sich die Anforderung, dass es sich um ein NIDS (A1) mit hoher Priorität handeln muss. Beim Betrachten der Topologie des Fahrzeugnetzwerks in der Abbildung 4.6 wird deutlich, dass der Einsatz des IDS in einem verteilten System (A2) stattfindet. Die Anforderungen A1 und A2 schließen ein HIDS als Werkzeug zur Einbruchserkennung im Netzwerk eines Fahrzeugs, welches mehrere Steuergeräte und weitere miteinander interagierende Softwarekomponenten beinhaltet, aus.
Als Ergänzung und als Abgrenzung zu IDS, welche speziell für den Einsatz in kabellosen Netzwerken entwickelt wurden, wird die Anforderung A3 definiert. Diese hat ebenfalls die Priorität „hoch“, da es sich um eine Ethernet-basierte Topologie handelt und damit der Einsatz von wireless-geeigneten IDS nicht sinnvoll wäre.

## 5.2 Datenanforderungen

Die Tabelle 5.2 fasst alle Anforderungen an die zu überwachenden Daten bzw. Datenquellen zusammen. Bei der Auswahl geeigneter Werkzeuge zur Einbruchserkennung ist es von hoher Bedeutung zu definieren, welche Daten überhaupt überwacht und analysiert werden sollen. In dieser Arbeit liegt der Fokus auf Fahrzeugnetzwerke mit der in der Abbildung 4.6 dargestellten Topologie. Dies führt zu der Anforderung möglichst viel von dem gesamten Netzwerkverkehr als Überwachungsziel mit hoher Priorität festzulegen. Es ist erforderlich, dass es sich um ein NIDS (B1) handelt.
Zusätzlich ist die Auswertung der Ethernet Frames sowohl für diese Arbeit als auch für Ethernet-basierte Fahrzeugnetzwerke allgemein interessant.

| ID | Anforderung | Priorität | Beschreibung |
|---|---|---|---|
| **B1** | Network Traffic | hoch | Gesamtes Netzwerk als Überwachungsziel |
| **B2** | Ethernet | mittel | Datenauswertung auf OSI-Schicht 2 |

Tabelle 5.2: Anforderungen an Daten

| ID | Anforderung | Priorität | Beschreibung |
|---|---|---|---|
| **C1** | Online | mittel | Erkennung während der Laufzeit |
| **C2** | Passive Reaktion | mittel | Kein aktives Eingreifen in das System |
| **C3** | Echtzeit | niedrig | Echtzeiterkennung |

Tabelle 5.3: Zeitliche Anforderungen

## 5.3 Zeitliche Anforderungen

In der Tabelle 5.3 sind zeitliche Anforderungen aufgelistet. Grundsätzlich lässt sich sagen, dass diese Anforderungen stark von dem gewünschten Effekt der IDS Nutzung abhängig sind.
Basierend auf der Kategorisierung aus der Abbildung 3.1 kann es sich bei einem IDS entweder um eine online oder offline Einbruchserkennung handeln. Eine Anforderung mittlerer Priorität stellt somit die online Erkennung (C1), welche während der Laufzeit stattfindet, dar. Eine offline Erkennung wäre für die Evaluation der Werkzeuge zur Einbruchserkennung zwar möglich, aber für den praktischen Einsatz in einem straßentauglichen Fahrzeug nicht sinnvoll.
Außerdem soll das ausgewählte IDS eine passive Reaktion (C2) auf mögliche Einbrüche liefern. Dies bedeutet, dass die Ergebnisse als Information zur Verfügung gestellt wird, aber das IDS keine aktiven Systemeingriffe vornimmt. Das weitere Vorgehen bei erkannten Einbrüchen kann somit individuell ggf. von anderen Systemkomponenten gesteuert werden.
Eine Echtzeiterkennung (C3) der Einbrüche stellt eine Anforderung mit einer niedrigen Priorität dar. Grundsätzlich bietet das keinen beachtlichen Vorteil für die Evaluation der IDS. Außerdem erfordert eine Echtzeiterkennung möglicher Angriffe ein entsprechendes Konzept zur Verarbeitung der Ereignisse.

| ID | Anforderung | Priorität | Beschreibung |
|---|---|---|---|
| **D1** | Signatur-basiertes IDS | mittel | Tool arbeitet mit Signatur-basierten Erkennungsmethoden |
| **D2** | Erweiterbare Regeln | niedrig | Regeln der IDS konnen anwendungsspezifisch erweitert werden |

Tabelle 5.4: Sonstige technische Anforderungen

## 5.4 Sonstige technische Anforderungen

Die Tabelle 5.4 beschreibt alle technischen Anforderungen, welche den System-, Daten- oder zeitlichen Anforderungen nicht zugeordnet werden können.
Um für die spätere Evaluation eine Vergleichbarkeit gewährleisten zu können, ist es von Vorteil, dass die ausgewählten IDS eine ähnliche Erkennungsmethodik verwenden. Daraus resultiert die Anforderung der Signatur-basierten Erkennung (D1) mit einer mittleren Priorität. Zusätzlich besteht die Möglichkeit bei Signatur-basierten Werkzeugen die vorhandenen Regeln zu erweitern (D2). Dies würde bedeuten, dass die Regeln bei Bedarf individuell angepasst oder neu entwickelt werden können. Diese Anforderung hat jedoch eine niedrige Priorität, da dies beim Einsatz der Werkzeuge für die Evaluation nicht vorgesehen ist.

## 5.5 Sonstige nichttechnische Anforderungen

In diesem Abschnitt werden Anforderungen, welche keine technische Relevanz darstellen, jedoch eine bedeutsame Rolle spielen können, vorgestellt. Eine Übersicht der Anforderungen ist in der Tabelle 5.5 gegeben.
Dadurch, dass ein Signatur-basiertes IDS nur Einbrüche erkennen kann, die bereits bekannt und die Regeln entsprechend implementiert sind, stellt sich die Anforderung der regelmäßigen Updates der IDS Regelsätze bzw. Datenbanken (E1). Zusätzlich muss das Werkzeug vom Hersteller unterstützt (E2) werden. Diese Anforderungen haben eine hohe Priorität, da ansonsten der Einsatz der Werkzeuge nicht langfristig ihren Zweck der Einbruchserkennung erfüllen kann.
Eine weitere Anforderung stellt eine aktuelle Systemdokumentation (E3) dar. Diese trägt zwar zur Leistungsfähigkeit der IDS nicht bei, erleichtert jedoch die Installation, Konfiguration und Anwendung in der entsprechenden Systemumgebung. Somit können eventuelle

| ID | Anforderung | Priorität | Beschreibung |
|---|---|---|---|
| **E1** | Regelmäßige Updates | hoch | IDS Datenbanken werden auf dem aktuellen Stand gehalten |
| **E2** | Support | hoch | IDS wird vom Hersteller unterstützt |
| **E3** | Aktuelle Systemdokumentation | niedrig | umfangreiche und einfach geschriebene Tool- und Konfigurationsbeschreibung |

Tabelle 5.5: Sonstige nichttechnische Anforderungen

| | A1 | A2 | A3 | B1 | E2 | E3 |
|---|---|---|---|---|---|---|
| **Snort** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Security Onion** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **OSSEC** | | | | | ✓ | ✓ |
| **Suricata** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Zeek** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **IBM Security QRadar NDR** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **OpenWIPS-NG** | | | | | | |
| **Sagan** | | | | | ✓ | ✓ |
| **NetRanger** | ✓ | ✓ | ✓ | ✓ | | |
| **Haystack** | | | | | | |
| **NetStat** | ✓ | ✓ | ✓ | ✓ | | |

Tabelle 5.6: Anforderungsmatrix Priorität „hoch“

Fehlkonfigurationen vorgebeugt werden.

## 5.6 Vorauswahl der Werkzeuge

Die Tabelle 5.6 stellt eine Matrix für die Vorauswahl der zu evaluierenden IDS dar. Diese besteht aus einer Auflistung aller im Abschnitt 4.1 vorgestellten IDS Werkzeuge und den oben definierten Anforderungen mit der Priorität „hoch“ .

Wie in der Tabelle 5.6 ablesbar, gibt es grundsätzlich folgende Werkzeuge zur Auswahl: Snort, Security Onion, Suricata, Zeek und IBM Security QRadar NDR. Die anderen Werkzeuge sind entweder nicht für die Einbruchserkennung auf Netzwerkebene geeignet oder die notwendige Aktualität ist nicht gegeben.

**Security Onion**

Bei Security Onion handelt es sich grundsätzlich zwar um ein mögliches Werkzeug zur Einbruchserkennung, jedoch ist es kein klassisches Tool, sondern eine Linux-Distribution, die Tools von Drittanbietern zur Verfügung stellt[32]. Da Zeek und Suricata bereits in Security Onion integriert sind, ist es nicht notwendig diese Distribution in die Evaluation mit einzubeziehen.

**IBM Security QRadar NDR**

Security QRadar ist eine von IBM vermarktete Lösung für den Bereich Cyber Secuirity. IBM Security QRadar Network Detection and Response (NDR) ist ein Produkt, welches Unternehmen dabei unterstützen soll, die unternehmensinterne Netzwerksicherheit zur überwachen. Somit steht Security QRadar nicht Open-Source zur Verfügung und die Nutzung dieses Produkts ist ebenfalls mit IBM als Dienstleistungsunternehmen verbunden[16].
Auch wenn es sich um ein Werkzeug zur Einbruchserkennung handelt, so ist es für den Einsatz am SecVI Demonstrator im Kontext dieser Arbeit ungeeignet. Daher wird IBM Security QRadar NDR bei der Evaluation nicht berücksichtigt.

**Ergebnis**

Somit werden folgende IDS Werkzeuge für die Evaluation im Rahmen dieser Arbeit ausgewählt:

- Snort
- Suricata
- Zeek

Diese Werkzeuge zur Einbruchserkennung stellen marktführende Open-Source Lösungen für Netzwerküberwachung und Sicherheit in Netzwerken dar.

# 6 Evaluation

In diesem Kapitel werden die im Abschnitt 5.6 ausgewählten NIDS näher erläutert und später evaluiert. Im Abschnitt 6.1 wird die allgemeine Funktionsweise solcher IDS - und wie diese eingesetzt werden können - vorgestellt. Daraufhin wird auf die einzelnen Werkzeuge zur Einbruchserkennung eingegangen, der Aufbau der Signatur-basierten Regeln und die dazugehörige Syntax erklärt. Um die Werkzeuge evaluieren zu können, wird zuerst im Abschnitt 6.3 auf wichtige Aspekte für die Einbruchserkennung in Fahrzeugnetzwerken eingegangen und die Möglichkeiten von Snort, Suricata und Zeek in dem Kontext vorgestellt. Zum Schluss wird der Einsatz von den ausgewählten IDS evaluiert.

## 6.1 Allgemeine Funktionsweise

Auch wenn es sich bei den ausgewählten Werkzeugen um IDS handelt, können diese vielfältig benutzt werden.
So bietet Snort die Möglichkeit als ein Sniffer, ein Paket Logger und ein NIDS eingesetzt zu werden. Diese Option ermöglicht Snort in drei weiteren Modi zu benutzen[12].

- Inline
- Passive
- Inline-Test

Im Inline Modus funktioniert Snort wie ein IPS. Damit wird ein aktives Eingreifen in den Netzwerkverkehr ermöglicht. Um Snort als ein IDS einzusetzen, ist der Passive Modus geeignet. Eine abgewandelte Form des Inline Modus ist der Inline-Test Modus. Hierbei funktioniert Snort als ein IDS, jedoch werden die Events, welche im Inline Modus den Netzwerkverkehr beeinflusst hätten, mit aufgezeichnet. Dies hat keine Auswirkungen auf den Verkehr, ermöglicht aber eine Evaluation über das Verhalten von Snort als ein IPS. Suricata kann auch sowohl als IDS oder IPS eingesetzt werden. Zusätzlich sind drei

verschiedene Modi zur Ausführung (Runmodes), die das Threading beeinflussen, von Suricata konfigurierbar[28]:

- single: Die Pakete werden in einem einzigen Thread verarbeitet. Dieser Modus wird für Test- oder Entwicklungszwecke empfohlen.
- workers: Die Pakete werden ausbalanciert über N Paketverarbeitungsthreads verteilt. Dieser Modus hat die beste Performance.
- autofp: Vor der Verarbeitung der Pakete werden diese in zusätzlichen Capture-Threads erfasst und dekodiert. Dies hat den Zweck PCAP Files zu erstellen und ist bei einigen IPS Funktionalitäten sinnvoll.

Bei Zeek handelt es sich um ein Framework zur Netzwerkanalyse, welches neben der Funktion als NIDS viele weitere Komponenten zur Verfügung stellt. Es kann unter anderem für Monitoring, Logging von Events, Fehler- oder Paketanalyse im Netzwerkverkehr eingesetzt werden [29].
In dieser Arbeit werden alle Werkzeuge aufgrund der Vergleichbarkeit als NIDS betrachtet.
Alle dieser Werkzeuge beinhalten ein Signatur-basiertes IDS. Eine Signatur wird als mindestens ein charakteristisches Merkmal, welches für ein bestimmtes Verhalten oder in diesem Fall einen Angriff spezifisch ist, verstanden. Der Nachteil bei Signatur-basierter Angriffserkennung ist, wie bereits im Unterabschnitt 2.1.4 thematisiert, dass die Signaturen erst nach dem Stattfinden des ersten Angriffs in eine Signaturdatenbank gepflegt werden können. Dies wird als „the day after“ Erkennung bezeichnet. Auch wenn es sich um Signatur-basierte Angriffserkennung handelt, ist es wichtig den Unterschied zwischen Signaturen und Regeln zu verstehen. Die Regeln stellen eine erweiterte, Signatur-basierte Erkennungsmethode, welche eine „zero day“ Erkennung unterstützt, dar. Anders als bei einer Signatur ist das Ziel einer Regel die tatsächliche Schwachstelle zu erkennen, und nicht nur nach einzelnen, verdächtigen Datenelementen zu suchen. Die Voraussetzung für das Entwickeln einer Regel ist das genaue Verständnis dafür wie die Schwachstelle funktioniert[12].
An erster Stelle sind Snort und Suricata marktführende Open-Source IDS und stellen auch ein Set an Community Regeln, welche seit Jahren weiter entwickelt und auf dem aktuellen Stand gehalten werden, zur Verfügung. Diese können verschiedene Angriffe identifizieren. Ein Regel-Set besteht aus mindestens einer, meistens aber aus mehreren Regeln, welche nach Angriffsart oder Einsatzbereich geordnet sind[12].
Neben der Community Regeln bieten alle drei Werkzeuge die Möglichkeit Regeln selbst zu

entwickeln, entsprechend zu konfigurieren und einzusetzen. In dem Abschnitt 6.2 werden der Aufbau und die Syntax der jeweiligen Regeln vorgestellt.

## 6.2 Aufbau der Signatur-basierten Regeln

In diesem Abschnitt wird erläutert, wie die Entwicklung der Regeln für die ausgewählten Werkzeuge funktioniert. Dafür wird spezifisch auf den Aufbau einer Regel und dazugerhöriger Syntax eingegangen. Außerdem werden abseits von Paketsignaturen weitere Optionen, welche bei der Umsetzung neuer Regeln relevant sind, vorgestellt.

### 6.2.1 Snort

Die Snort Regeln bestehen aus einem Rule Header und einer Rule Option [20]. In der Abbildung 6.1 ist der Aufbau einer Snort Regel dargestellt. Bei dieser Beispielregel wird ein Alarm ausgelöst, wenn die Payload eines TCP Pakets den String „Bob“ enthält. Dies gilt für alle Transmission Control Protocol (TCP) Pakete, die an einen beliebigen Host in einem vorher definierten $HOME_NET über Port 80 gesendet werden.

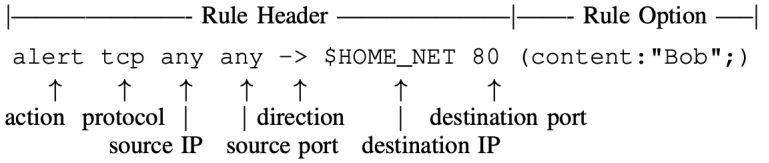



Abbildung 6.1: Snort Regel (Quelle: [20] Abbildung 1)

Der Rule Header beinhaltet folgende, erforderliche Parameter(Abbildung 6.1): eine Aktion, ein Protokoll, eine Quell- und Ziel-Adresse, die jeweils aus einer IP-Adresse und einem Port besteht. Außerdem wird ein Operator für die Richtung benötigt.
Der Parameter „action“ definiert genau eine Aktion, welche bei der Übereinstimmung mit der definierten Regel auftreten soll. Bei dem Einsatz von Snort als ein IDS, stehen die Aktionen alert, log und pass zur Verfügung. Bei der Nutzung als IPS im Snort inline Modus können die Aktionen drop, reject und sdrop zusätzlich verwendet werden[12].

- alert: Ein Alert wird ausgelöst und das Paket wird geloggt.
- log: Das Paket wird geloggt.

- pass: Das Paket wird ignoriert.
- drop: Das Paket wird blockiert und geloggt.
- reject: Das Paket wird blockiert und geloggt. Zusätzlich wird ein TCP-Reset bei TCP Paketen oder eine Internet Control Message Protocol (ICMP) Port unreachable Meldung bei User Datagram Protocol (UDP) gesendet.
- sdrop: Das Paket wird blockiert.

Aktuell ist Snort in der Lage folgende Protokolle zu analysieren: TCP, UDP, ICMP und Internet Protocol (IP)[12].
Nach der Definition der Aktion und des Protokolls muss die Quell- und Zieladresse angegeben werden. Um eine beliebige Adresse zu verwenden kann das Schlüsselwort any verwendet werden. Die Adressen werden als numerische IP-Adressen mit einem Classless Inter-Domain Routing (CIDR)-Block angegeben. So können bestimmte Adressbereiche oder eine konkrete Rechneradresse definiert werden. Bei der Angabe der IP Adresse kann ein Negationsoperator „!“ angewendet werden. In der Abbildung 6.2 werden die Quelladressbereiche 192.168.1.0/24 und 10.1.1.0./24 für die Regel ausgeschlossen. Dies sorgt dafür, dass der angegebene Adressbereich bei der Regel nicht berücksichtigt wird[12]. Außerdem können Listen von Adressen als Quell- oder Zieladresse, indem sie mit Komma getrennt werden, verwendet werden. Dies wird in der Abbildung 6.2 sowohl bei der Quell- als auch bei der Zieladresse veranschaulicht.

```
alert tcp ![192.168.1.0/24,10.1.1.0/24] any -> \
    [192.168.1.0/24,10.1.1.0/24] 111 (content:"|00 01 86 a5|"; \
    msg:"external mountd access";)
```

Abbildung 6.2: Snort Regel mit Adressliste und Negationsoparator (Quelle: [12] Abbildung 3.3)

Die Ports der Quell- und Zieladresse können mithilfe verschiedener Ansätze definiert werden. Zum einen können Ports statisch als eine einzelne Port Nummer wie Port 80 in der Abbildung 6.1 bei der Zieladresse angegeben werden. Zum anderen können Portbereiche mithilfe eines Bereichsoperators „:“ definiert werden. So wäre die Angabe 6000:6010 für jeden Port zwischen 6000 und 6010, wobei die Portnummern 6000 und 6010 auch berücksichtigt werden. Die Bereichsdefinition kann ebenso in eine Richtung offen gelassen werden. Die Angabe 6000: würde jede Portnummer, die größer oder gleich 6000

ist, beinhalten. Außerdem kann auch hier das Schlüsselwort any für beliebige Ports und der Negationsoperator für den Ausschluss bestimmter Ports verwendet werden[12]. Der Richtungsoperator gibt die Richtung des Datenverkehrs an. Snort stellt dafür zwei Möglichkeiten zur Verfügung. Der Operator -> definiert die Adresse auf der linken Seite von dem Operator als Quelladresse und die Adresse auf der rechten Seite als Zieladresse, somit wird die Regel nur bei der Kommunikation in die entsprechende Richtung berücksichtigt. Zusätzlich kann der Richtungsoperator <> für eine bidirektionale Kommunikation angewendet werden. Dabei werden beide Adress- und Portpaare als Quell- und Zieladresse in beide Richtungen betrachtet.
Neben der Rule Header muss auch die Rule Option definiert werden. Snort stellt vier Hauptkategorien der Regeloptionen zur Verfügung[20].
Die Snort Rule Options werden in Klammern hinter der Rule Header geschrieben. Viele Schlüsselwörter stehen hierfür zur Auswahl. Das wichtigste Schlüsselwort ist „msg“, welches für die Payload Überprüfung zuständig ist. Zusätzlich gibt es protokollspezifische Schlüsselwörter sowohl zu den Layer 3 und 4 als auch zu den Anwendungsspezifischen Protokollen.

- general
- payload
- non-payload
- post-detection

Die Rule Option „general“ gibt die Informationen über die Regel an, ohne in den Erkennungsprozess einzugreifen. Für diese Option stehen acht Schlüsselbegriffe zur Verfügung[12].

- msg: Eine Textnachricht, welche bei dem Eintreten der Regel ausgegeben werden soll.
- reference: Ein Verweis auf externe Systeme, welche weitere Informationen über bestimmte Angriffe beinhalten und somit die Identifizierung solcher Angriffe unterstützen.
- gid: Die Generator ID (gid) bestimmt welcher Teil von Snort die im Rule Header definierte Aktion auslöst. Allgemein wird von Snort empfohlen diesen optionalen Wert nicht zu setzen. Der standartmäßige Wert ist auf 1 gesetzt und mit dem Regel Subsystem verbunden. Gid sollte zusammen mit sid verwendet werden.

- sid: Die Snort-Regel ID identifiziert die Regeln eindeutig. Sid sollte zusammen mit rev verwendet werden.
- rev: Die Revisions ID gibt die Revision von einer in sid angegebenen Regel an.
- classtype: Der Klassentyp kategorisiert die Regeln in bestimmte Klassen.
- priority: Die Priorität stellt eine Möglichkeit zur Priorisierung der Regeln dar.
- metadata: Das Keyword metadata ermöglicht zusätzliche Informationen über die Regel in einem key-value Format einzubetten.

Die Rule Option „payload“ ermöglicht die Überprüfung der Paketinhalte. Snort stellt 49 Schlüsselwörter für diesen Zweck zur Verfügung[20]. Die gängigsten payload Schlüssel sind:

- content: Der Inhalt der Pakete wird auf Übereinstimmung geprüft.
- rawbytes: Die Paketdaten werden als Rohdaten behandelt.
- depth: Die Tiefe gibt an, wie weit in einem Paket nach content gesucht wird.
- offset: Der Offset gibt an, wo die Suche nach content gesucht wird.
- distance: Die Distanz gibt an, in welchen Abständen nach content gesucht wird.

Im Gegensatz zu der payload Rule Option befasst sich die non-payload Rule Option nicht mit den Paketinhalten, sondern mit sonstigen Paketinformationen. Dazu gehören die Protokoll Header Daten. Für diese Option stellt Snort 22 protokollspezifische (TCP, ICMP, IP) und allgemeine Schlüsselwörter zur Verfügung. Die post-detection Rule Option befasst sich mit den Aktionen, die nach der Erkennung der Regel zusätzlich ausgeführt werden[20]. Diese Optionen verfeinern die im Rule Header definierte action. Es gibt zehn Schlüsselwörter für diese Option, darunter logto, session, resp, tag, replace und detection_filter[12].

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET TROJAN Likely Bot Nick in IRC (USA
+..)"; flow:established,to_server; flowbits:isset,is_proto_irc; content:"NICK "; pcre:"/NICK
.*USA.*[0-9]{3,}/i"; reference:url,doc.emergingthreats.net/2008124; classtype:trojan-
activity; sid:2008124; rev:2;)
```

Abbildung 6.3: Suricata Regel bestehend aus einer Action (rot), Header (grün) und Option (blau) (Quelle: [28])

### 6.2.2 Suricata

Ähnlich wie bei Snort besteht eine Regel bei Suricata aus einer Rule Header und einer Rule Option, jedoch gehört die Aktion nicht zu der Header. Der Aufbau einer Suricata Regel wird in der Abbildung 6.3 deutlich.

Suricata bietet bei der Entwicklung von Regeln folgende Aktionen:

- alert: Eine Warnung wird erzeugt.
- pass: Das Paket wird nicht weiter geprüft.
- drop: Das Paket wird verworfen und eine Warnung wird erzeugt.
- reject: Je nach Protokoll wird ein TCP-Reset oder eine ICMP unreachable error an den Absender des Pakets gesendet.
- rejectsrc: Je nach Protokoll wird ein TCP-Reset oder eine ICMP unreachable error an den Absender des Pakets gesendet.
- rejectdst: Je nach Protokoll wird ein TCP-Reset oder ein ICMP Error Paket an den Empfänger des Pakets gesendet.
- rejectboth: Je nach Protokoll wird ein TCP-Reset oder ein ICMP Error Paket sowohl an den Sender als auch Empfänger des Pakets gesendet.

Bei dem Einsatz von Suricata als ein IPS beinhalten alle Aktionen, welche die Pakete ablehnen (reject), zusätzlich die drop Aktion.
Eine Suricata Rule Header definiert das Protokoll, die Quell- und Ziel-IP-Adressen, die dazugehörigen Ports und die Richtung.
Suricata stellt folgende Protokolle zur Verfügung: TCP, UDP, ICMP und IP. Zusätzlich werden viele Protokolle auf der Anwendungsschicht unterstützt darunter HTTP, FTP, TLS, SMB, DNS, DCERPC, SSH, SMTP, IMAP, NFS, IKEv2, krb5, NTP, DHCP, RFB,

RDP, SNMP, TFTP, SIP, HTTP2.
Nach der Festlegung des Protokolls müssen die Quell- und Zieladressen definiert werden. Üblicherweise, wenn der Richtungsoperator „->“ angegeben wird, steht auf der linken Seite der Sender und auf der rechten Seite der Empfänger des Pakets. Die IP-Adresse kann sowohl als eine IPv4 als auch IPv6 Adresse, oder als ein Adressbereich in der CIDR Notation angegeben werden. Mehrere IP-Adressen können in eckigen Klammern, mit Komma getrennt definiert werden. Das Schlüsselwort „any“ erlaubt jede beliebige IP-Adresse. Außerdem kann der Negationsoperator „!“ , um bestimmte Adressen oder Adressbereiche auszuschließen, verwendet werden. Dies ist nicht zulässig, wenn „any“ die Adresse definiert.
Ähnlich wie bei der IP-Adresse kann der Port als ein einzelner Wert, mehrere Werte oder ein Wertebereich angegeben werden. Und auch hier kann der Negationsoperator verwendet werden. So wäre der Ausdruck [6000:6020,![6010,6011]] für alle Portnummern von 6000 bis 6020, außer die Portnummern 6010 und 6011.
Die Richtungsangabe der Suricata Rules wird genau so wie bei Snort umgesetzt[28]. Für die Rule Options bietet Suricata vielfältige Möglichkeiten die Signaturen zu spezifizieren. Die Meta Keywords sind rein informativ und geben eine Auskunft über die Signatur und die daraus resultierenden Aktionen. Dies hat keinen Einfluss auf das von der Signatur definierte Verhalten. Für die Inhaltsanalyse der Pakete werden Payload Keywords mit verschiedenen optionalen Schlüsselwörtern, um die gesuchten Inhalte zu spezifizieren, zur Verfügung gestellt. Abhängig von dem Protokoll, welches in der Rule Header festgelegt wird, stehen protokollspezifische Schlüsselwörter zur Verfügung. Diese sind die IP, TCP UDP und ICMP Keywords. Zusätzlich ermöglicht Suricata die Regeln in Bezug auf die unterstützten Protokolle der Anwendungsschicht zu spezifizieren[28].

### 6.2.3 Zeek

An erster Stelle bietet Zeek als eine Netzwerkanalyse Plattform eine Skriptsprache zur Definition von Events und zur Analyse von Netzwerkverhalten. Der Fokus von Zeek liegt nicht bei der Nutzung als ein NIDS, jedoch ist eine Signatur-basierte Einbruchserkennung ein Teil davon. Zeek stellt eine Signatur-basierte Sprache sowohl für die Analyse von Netzwerkverkehr als auch für File Content zur Verfügung. In der Abbildung 6.4 ist es deutlich zur erkennen, dass die Regeln von Zeek eine andere Struktur als die von Snort und Suricata aufweisen. Außerdem fällt auf, dass bei Zeek nicht von Regeln, sondern von Signaturen geschrieben wird.

Ein Beispiel für eine einfache Signatur wird in der Abbildung 6.4 dargestellt. Dabei wird ein Event, welches in der Abbildung 6.5 definiert wird, ausgelöst, wenn der reguläre Ausdruck „.*root“ über den TCP Port 80 gesendet wird.

```
signature my-first-sig {
    ip-proto == tcp
    dst-port == 80
    payload /.*root/
    event "Found root!"
}
```

Abbildung 6.4: Eine einfache Zeek Signatur(Quelle: [29])

Das dazugehörige Event in der Abbildung 6.5 beinhaltet ein Parameter state, welches zusätzliche Informationen über den Auslöser der Signaturübereinstimmung beinhaltet, und ein weiteres Parameter msg, welches die Nachricht von der Signatur ausgibt. Bei diesem Beispiel wäre die msg „Found root!“ . Das Feld data beinhaltet die Payloadnachricht von dem Paket, welches die Übereinstimmung der Signatur ausgelöst hat.

```
event signature_match(state: signature_state, msg: string, data: string)
```

Abbildung 6.5: Ein Zeek generiertes Event zu Abbildung 6.4 (Quelle: [29])

Grundsätzlich hat eine Zeek Signatur dieses Format: signature <id> { <attributes> }. Die Signaturen bestehen aus zwei Hauptbereichen: Konditionen (Conditions) und Aktionen (Actions).
Die Konditionen sind in vier weitere Kategorien aufgeteilt:

- Header Conditions
- Content Conditions
- Dependency Conditions
- Context Conditions

Die Header Konditionen erfüllen den Zweck den zu überwachenden Datenverkehr einzuschränken, indem diese mit den Paket Header Informationen abgeglichen werden. Dieser Vorgang findet nur bei dem ersten Paket einer Verbindung statt. Die Syntax kann wie folgt beschrieben werden: <keyword> <cmp> <value-list>. Bei „keyword“ handelt es sich um ein Schlüsselwort, welches mit der „value-list“ verglichen werden soll. Die gängigsten Schlüsselwörter sind src-ip, src-port, dst-ip und dst-port. Diese werden dann mit

den tatsächlichen Quell- oder Ziel-Adressen oder Adressbereichen auf Übereinstimmung geprüft. Der Parameter „cmp“ stellt einen Vergleichsoperator dar. Außerdem kann mit dem Schlüsselwort ip-proto ein konkretes Protokoll festgelegt werden. Zeek unterstützt folgende Protokolle TCP, UDP, ICMP, ICMP6, IP und IPv6.
Die Content Konditionen werden durch reguläre Ausdrücke definiert. Weiterhin werden diese in zwei Arten eingeteilt. Zum einen können die Payload Informationen der Pakete verarbeitet werden. Hierfür wird das Schlüsselwort „payload“ verwendet. Zum anderen können mithilfe spezifischer Content Schlüsselwörter einige Informationen der Anwendungsschicht (vor allem HTTP Request und Reply) ausgelesen werden.
Die Dependency Konditionen stellen eine Abhängigekit zu anderen Signaturen dar. Diese Abhängigkeit gilt nur innerhalb einer Verbindung. Hierfür stehen zwei Möglichkeiten zur Auswahl:

- requires-signature [!] <id>: Die aktuelle Signatur stimmt nur dann überein, wenn die mit der Signatur mit der angegebenen id übereinstimmt.
- requires-reverse-signature [!] <id>: Die aktuelle Signatur stimmt nur dann überein, wenn die mit der Signatur mit der angegebenen id für die entgegengesetzte Richtung der Verbindung übereinstimmt.

Nachdem eine Signatur anschlägt, werden die Context Konditionen überprüft. Diese sind in der Lage weitere Paketinformationen zu evaluieren (z.B. payload-size), Informationen an andere Komponenten von Zeek weiter zu leiten oder bestimmte Funktionalitäten auszuführen. Weiterhin stellt Zeek für die Signaturdefinition zwei Aktionen zur Verfügung[29]:

- event <string>: Ein Event mit einer Nachricht (string) wird generiert.
- enable <string>: Eine in string definierte Zeek Komponente zur Protokollanalyse wird aktiviert.

## 6.3 Einsatz in Fahrzeugnetzwerken

In diesem Abschnitt werden Kriterien, welche für die Angriffserkennung in Fahrzeugnetzwerken eine besondere Relevanz haben, vorgestellt und geprüft, inwiefern die ausgewählten IDS diese erfüllen. Die IDS wurden für herkömmliche IT-Systeme und somit nicht primär für den Einsatz in Fahrzeugnetzwerken entworfen. Dieser Abschnitt soll zeigen,

ob und wie die Werkzeuge die Kriterien abbilden. Um Angriffe auf Fahrzeugnetzwerke zu erkennen, sind folgende Aspekte relevant: Protokollstack, Payload der zu analysierenden Pakete, Paketgröße und zeitliche Aspekte. Diese vier Kriterien werden einzeln in den folgenden Unterabschnitten in Bezug auf die Angriffserkennung in Fahrzeugnetzwerken erläutert und auf die IDS abgebildet.

### 6.3.1 Protokollstack

Die Aufgabe von NIDS ist Netzwerkverkehr zu überwachen und gegebenenfalls weitere Maßnahmen einzuleiten. Für die Analyse der Pakete ist es somit von großer Wichtigkeit, welche Schichten des OSI-Modells beim Auswerten der Daten abgedeckt werden.
Wie bereits im Abschnitt 4.2 erläutert, wird in dieser Arbeit eine Ethernet-basierte Topologie für Fahrzeugnetzwerke als ein potenzielles Angriffsziel evaluiert. Darauf basierend spielt das OSI-Layer 2 bereits eine große Rolle, da die Ethernet-Header die Informationen über Datenströme und somit eine Filtermöglichkeit beinhalten. Ob diese Informationen auf höheren OSI-Schichten ebenfalls verfügbar sind, hängt jedoch vom Netzwerkdesign ab.
Keins der ausgewählten IDS ist in der Lage das Layer 2 (Ethernet) auf der Signatur Ebene abzubilden. Zeek bietet zwar ein Modul für die Datenverarbeitung auf Layer 2, jedoch handelt es sich dabei um eine Monitoring Komponente, die keine Informationen an das IDS weiter leiten kann.
Die Syntax der Signaturen bietet die Möglichkeit die Protokolle IP, ICMP auf Layer 3 und UDP, TCP explizit anzugeben. Außerdem unterstützt Zeek zusätzlich IPv6 und ICMP6. Wie im Unterabschnitt 6.2.2 beschrieben, ist Suricata zwar in der Lage viele Protokolle der Anwendungsschicht zu verarbeiten, dies ist aber für den Einsatz in Fahrzeugnetzwerken nicht ausschlaggebend.

### 6.3.2 Payload

Für die Einbruchserkennung in Fahrzeugnetzwerken ist der Dateninhalt der Pakete ein grundlegender Aspekt und somit spielt das Auslesen der Payload eine zentrale Rolle. Dadurch, dass die Signatur-basierte Einbruchserkennung auf Payload Inhalte angewiesen ist, wird diese Möglichkeit von allen ausgewählten Werkzeugen zur Verfügung gestellt. Dies erfolgt jedoch in unterschiedlichem Ausmaß.

Während Zeek die Payload anhand eines regulären Ausdrucks definieren lässt, bieten Snort und Suricata eine Vielzahl an zusätzlichen Schlüsselwörtern, welche die Datenanalyse erleichtern. Neben der Inhaltsanalyse ist es möglich zu konfigurieren, um was für eine Art von Inhalt es sich handelt (pcre, Snort: uricontent, ftpbounce), an welcher Stelle dieser lokalisiert ist (offset, depth, distance) oder sonstige Funktionen (byte_test). Dies erlaubt eine gezielte Inhaltssuche und eine genaue Datenanalyse, welche für Einbruchserkennung in Fahrzeugnetzwerken entsprechend konfigurierbar ist.

### 6.3.3 Paketgröße

Eine weitere Grundlage der Einbruchserkennung in Fahrzeugnetzwerken stellt das Auslesen der Paketgröße dar. Bei allen Werkzeugen kann die Paketgröße mit unterschiedlichen Vergleichsopearatoren angegeben. Außerdem können auch zu erkennende Größenbereiche definiert werden. Die Payloadgröße wird wie folgt angegeben:

- Zeek: payload-size <cmp> <integer> (cpm ist einer der Vergleichsoperatoren ==, !=, <, <=, >, >=)
- Suricata: dsize:<number>;
- Snort: dsize:min<>max; oder dsize:[<|>]<number>;

### 6.3.4 Zeitverhalten

Viele Angriffe lassen sich anhand einzelner Pakete nicht erkennen und erfordern eine Möglichkeit die Bandbreite oder die zeitlichen Abstände zwischen den Paketen zu messen. Dieser Aspekt macht die Entwicklung einer Signatur-basierten Regel deutlich komplexer als eine reine Datenauswertung basierend auf einzelnen Paketinformationen.
Snort lässt mithilfe der Post-detection Regeloptionen die Zeit- und Bandbreitenanalyse in die Regeln integrieren. Dafür steht das Schlüsselwort detection_filter wie in der Abbildung 6.6 dargestellt. Die track by_src|by_dst definiert auf welche Quell- oder Zieladresse sich diese Regel bezieht. Count c gibt an, wie oft diese Regel im Zeitraum von s Sekunden (s seconds) positiv sein muss, sodass die vordefinierte Aktion ausgeführt wird. Pro Regel darf maximal ein detection_filter definiert werden[12].

Auch bei Suricata steht ein detection_filter Schlüsselwort zur Verfügung. Die Syntax ist in der Abbildung 6.7 dargestellt und hat eine große Ähnlichkeit zu Snort. Der einzige

```
detection_filter: \
    track <by_src|by_dst>, \
    count <c>, seconds <s>;
```

Abbildung 6.6: Snort detection_filter Syntax(Quelle: [12])

Unterschied zu Snort ist die Erweiterung der track Variable. Bei Suricata ist es neben der Angabe der Quell- oder Zieladresse zusätzlich möglich beide Adressen gleichzeitig (by_both) innerhalb einer Regel zu analysieren oder sogar eine andere Regel (by_rule) als Referenz für die Messung zu nutzen[28].

```
detection_filter: track <by_src|by_dst|by_rule|by_both>, count <N>, seconds <T>
```

Abbildung 6.7: Suricata detection_filter Syntax(Quelle: [28])

Das IDS von Zeek bietet keine Optionen für eine zeitliche Analyse des Netzwerkverkehrs[29].

### 6.3.5 Regeln in Fahrzeugnetzwerken

Um einen sinnvollen Einsatz von IDS wie Snort in Fahrzeugnetzwerken zu ermöglichen, ist vor allem eine Flexibilität in Bezug auf das Entwickeln von Regeln notwendig. Ein möglicher Anwendungsfall auf dem SecVI Demonstrator wäre die Inhalte der Pakete auf SOME/IP Informationen zu prüfen und diese entsprechend zu validieren oder anderweitig auszuwerten. Die Werkzeuge unterstützen zwar SOME/IP nicht als Protokoll, bieten aber die Möglichkeit die Payload der Pakete auszulesen. Somit könnte der UDP Paketinhalt auf SOME/IP spezifische Daten geprüft und für eine Angriffserkennung genutzt werden. Dies erfodert Kenntnisse über den zu analysierenden Netzwerkverkehr und potenzielle Angriffsmöglichkeiten. Eine andere Option wäre die Bestimmung der Bandbreite oder die Analyse einer auffälligen Anzahl der Pakete in vordefinierten Datenströmen.
Um den Einsatz zu erproben wurde auf dem SecVI Demonstrator auf dem NADS1 (Abbildung 4.6) Snort installiert und entsprechend konfiguriert. Die aktuelle Konfiguration wendet die Regeln aus dem File: /etc/snort/rules/secvi.rules an. In der Abbildung 6.8 wurde eine Beispielregel umgesetzt. Diese Regel überwacht den UDP Verkehr von der Kamera (core-pi4-05) mit der IP-Adresse 10.0.0.5 an den Monitor (core-pi4-06) mit der IP-Adresse 10.0.0.6 und den Port 1112 (Abbildung 4.6) und meldet ein Alert, sobald mehr als 100 Pakete mit der entsprechenden Quell- und Zieladresse in dem Zeitraum von einer Sekunde gesendet werden.

```
alert udp 10.0.11.5 any -> \
        10.0.11.6 1112 (msg: "Packet Injection Videostraeam"; \
                                        detection_filter:track by_src, count 100, seconds 1; \
                                        sid:1000002; rev:1)
```

Abbildung 6.8: Signatur-basierte Regel Snort

## 6.4 Evaluation

Das Hauptziel dieser Arbeit ist das Evaluieren von Werkzeugen zur Einbruchserkennung in Fahrzeugnetzwerken. Für diesen Zweck wurden im Kapitel 5 drei marktführende NIDS ausgewählt: Snort, Suricata und Zeek. Diese Tools wurden ursprünglich nicht für den Einsatz in Fahrzeugnetzwerken mit einer Ethernet-basierten Architektur entwickelt. Die IDS sind für den Einsatz in herkömmlichen IT-Systemen gedacht. Somit war die Hauptaufgabe zu prüfen, ob die ausgewählten Werkzeuge in diesem Kontext der Fahrzeugnetzwerke eine gute Lösung für Angriffserkennung wäre.
Die Kommunikation innerhalb eines Fahrzeugnetzwerks wird durch ein Subnetz realisiert. Die Verteilung der Pakete an die Steuergeräte erfolgt anhand der MAC-Adresse. Dies erfordert aufgrund der ansonsten fehlenden Filterungsmöglichkeiten eine Einbruchserkennung bereits auf dem OSI-Layer 2 (Ethernet), sofern die Stream Informationen nicht auf höheren Schichten verfügbar sind. Dies hängt jedoch von dem Netzwerkdesign ab. Wie in dem Abschnitt 6.3 beschrieben, bieten die ausgewählten Werkzeuge keine Optionen die Regeln basierend auf der MAC-Adresse zu entwickeln. Im Abschnitt 6.2 wird deutlich, dass die Regeln der Werkzeuge erst ab OSI-Layer 3 entwickelt und angewendet werden können.
Snort und Suricata sind jahrelange Marktführer im Bereich Open-Source NIDS und verfügen somit über große Communities, welche ständig an der weiter Entwicklung der Tools arbeiten. Ein weiterer Aspekt, welcher für die Benutzung dieser Werkzeuge relevant ist, ist das Vorhandensein von Community Regeln. Beide IDS bieten jeweils eine Auswahl an Regeln, welche für die Einbruchserkennung optional genutzt werden können. Bevor die Community Regeln konfiguriert werden, müssen diese Regeln analysiert werden, inwiefern diese für das zu überwachende System relevant sind. Jeder aktivierte Regelsatz verbraucht Ressourcen, welche stark begrenzt sein können und somit gezielt eingesetzt werden sollen. Die Community Regeln wurden so wie die Tools an sich für klassische IT-Systeme, und nicht für den Einsatz in Fahrzeugnetzwerken, entwickelt.
Neben der Möglichkeit Community Regeln anzuwenden, können bei allen drei Werkzeugen Signatur-basierte Regeln individuell entwickelt werden. Die allgemeine Syntax der

Regeln wurde bereits in diesem Kapitel beschrieben. Zusätzlich wurde die Syntax der IDS auf vier Aspekte, welche für die Angriffserkennung in Fahrzeugnetzwerken relevant sind, im Abschnitt 6.1 untersucht. Das Ergebnis zeigt, dass, sofern die überwachten Pakete anhand einer Quell- und Zieladresse gefiltert werden können, das Entwickeln von Regeln für den Einsatz in Ethernet-basierten Fahrzeugnetzwerken theoretisch möglich wäre. Dies erfordert jedoch Aufwand für die Analyse, welche Angriffe erkannt werden sollen, für die Umsetzung, welche je nach Angriffsart sehr komplex sein kann, und ein Konzept zum Warten der Regeln, da Angriffe meist dynamisch sind und regelmäßige Updates benötigen.
Ein weiterer relevanter Aspekt für die Nutzung der IDS in Fahrzeugnetzwerken ist die Zweckmäßigkeit. Neben der Machbarkeit Angriffe zu erkennen ist die Verarbeitung dieser Erkenntnisse ein wichtiges Thema. IDS sind nicht in der Lage aktiv in das System einzugreifen und somit weitreichende Folgen direkt zu verhindern, sondern lediglich von den Signatur-basierten Regeln definierten Netzwerkverkehr zu erkennen und passiv zu reagieren. Daraus stellt sich die Frage, inwiefern ein passives Monitoring der Angriffe im praktischen Einsatz sinnvoll ist. Das aktive Verhindern der Angriffe, durch ein Eingriff in den Netzwerkverkehr, wird nicht von IDS sondern von IPS realisiert.

# 7 Fazit und Ausblick

Dieses Kapitel stellt einen Abschluss dieser Arbeit mit einem Fazit und einem Ausblick dar. Im Fazit werden die wichtigsten Erkenntnisse zusammengefasst und der Ausblick dient als eine Grundlage für mögliche, weiterführende Untersuchungen im Bereich der Einbruchserkennung in Fahrzeugnetzwerken.

## 7.1 Fazit

Das Hauptziel dieser Arbeit war das Herausfinden, ob die marktüblichen NIDS als Werkzeuge zur Einbruchserkennung für den Einsatz in Fahrzeugnetzwerken sinnvoll sind. Außerdem war das Prüfen der Anwendbarkeit für eine Evaluation notwendig. Dies erforderte einerseits eine Analyse der Kriterien, welche für Ethernet-basierte Fahrzeugnetzwerke relevant sind. Andererseits war eine Analyse der Tools, und was diese mitbringen, erforderlich.
Zuerst wurde eine Marktanalyse für mögliche Werkzeuge für Einbruchserkennung durchgeführt und eine Vorauswahl anhand definierter Kriterien getroffen. Dabei wurden drei NIDS ausgewählt: Snort, Suricata und Zeek. Des Weiteren war die Aufgabe zu prüfen, wie diese konkreten Tools funktionieren und inwiefern diese für den Einsatz in Fahrzeugnetzwerken geeignet sind. Dabei wurde die Architektur und die Topologie des SecVI Demonstrators als Referenz für ein Fahrzeugnetzwerk verwendet.
Ein ausschlaggebendes Hindernis für den Einsatz der ausgewählten IDS ist die fehlende Filtermöglichkeit auf der OSI-Schicht 2. Die Tools arbeiten ab der OSI-Schicht 3 aufwärts. Die Angabe der Quell- und Zieladresse als MAC-Adresse kann im Bereich Fahrzeugnetzwerke, je nach Netzwerkdesign, ausschlaggebend sein. Sofern die Stream Informationen in den höheren Schichten nicht zur Verfügung stehen, ist der Einsatz von NIDS nicht sinnvoll.
Snort und Suricata bieten zwar eine Sammlung an Community Regeln, diese wurden

aber für den Einsatz in klassischen Netzwerken entwickelt und bilden somit kaum Anwendungsfälle für Angriffe in Fahrzeugnetzwerken ab. Des Weiteren wurde die Möglichkeit evaluiert Signatur-basierte Regeln individuell zu schreiben. Auch hierfür stellt die fehlende Filtermöglichkeit der Quell- und Zieladresse nach der MAC-Adresse eine Schwierigkeit dar. Zusätzlich wurde die Möglichkeit für das Auslesen der Paketdaten (Payload, Größe) und für die Analyse der zeitlichen Aspekte (Bandbreite) untersucht. Diese werden von den ausgewählten Tools unterstützt. Jedoch setzt das Schreiben solcher Regeln ein tiefgehendes Verständnis über Angriffe, welche spezifisch für Fahrzeugnetzwerke sind, voraus. Außerdem bringt das Entwickeln solcher Regeln eine Verantwortung diese Regeln auf dem aktuellen Stand zu halten mit sich.
Somit lässt sich sagen, dass der Einsatz von NIDS in Fahrzeugnetzwerken differenziert betrachtet werden muss. Vor dem Einsatz sollte eine individuelle Machbarkeitsstudie oder ein Konzept und eine Kosten-Nutzen Analyse erstellt werden. Außerdem sollten weitere Alternativen in Betracht gezogen werden.

## 7.2 Ausblick

Die Werkzeuge Snort, Suricata und Zeek können nicht nur als reine IDS genutzt werden. Alle drei Tools können für das Überwachen von Netzwerkverkehr eingesetzt werden. Zeek Stellt eine vielseitige Plattform mit verschiedenen Modulen für die Sicherheit in Netzwerken zur Verfügung, dieser Aspekt könnte ebenfalls für den Einsatz in Fahrzeugnetzwerken evaluiert werden.
Weiterhin bleibt die Frage offen, ob aufgrund der zeitlichen Kritikalität der Fahrzeugnetzwerke der Einsatz von IPS, was ebenfalls von Snort und Suricata angeboten wird, sinnvoll wäre. Diese Option hätte neben der Erkennung von Einbrüchen den Vorteil auf diese sofort reagieren zu können. Aber auch dafür wäre eine Evaluation und ein Konzept mit Maßnahmen erforderlich, da es im praktischen Einsatz weitreichende Folgen haben kann.
Ergänzend sollten weitere Alternativen für Angriffs- und Anomalieerkennung, welche andere Erkennungsmethoden implementieren, evaluiert werden.

# Literaturverzeichnis


[1]

[2] : *OpenWIPS-NG*. – URL https://openwips-ng.org. – Zugriffsdatum: 2022-07-08

[3] : *Einführung von Intrusion-Detection-Systemen*. 2002. – URL https://www.bsi.bund.de/DE/Service-Navi/Publikationen/Studien/IDS02/index_htm.html?nn=520750

[4] *The Trends of Intrusion Prevention Network*, Sriwijaya University, Dissertation, 2010

[5] *Specification-based Intrusion Detection for Home Area Networks in Smart Grids*, The University of British Columbia, Dissertation, 2011

[6] : *Die Lage der IT-Sicherheit in Deutschland 2021*. 2021. – URL https://www.bsi.bund.de/DE/Service-Navi/Publikationen/Lagebericht/lagebericht_node.html

[7] Baumann, Felix: *Ein Spezifikations-basierter Intrusion Detection Ansatz im Automotive Ethernet*, Dissertation, 08 2019

[8] Bolzoni, D.: *Revisiting Anomaly-based Network Intrusion Detection Systems*, University of Twente, Dissertation, 2009

[9] CISCO: *Cisco NetRanger Sensor*. – URL https://www.cisco.com/c/en/us/obsolete/security/cisco-netranger-sensor.html. – Zugriffsdatum: 2022-07-08

[10] CISCO: *ClamAV*. – URL https://www.clamav.net

[11] CISCO: *Snort*. – URL https://snort.org. – Zugriffsdatum: 2022-07-08

[12] CISCO: *Snort Users Manual*. – URL https://snort-org-site.s3.amazonaws.com/production/document_files/files/000/000/249/original/snort_manual.pdf

[13] Damopoulos, Dimitrios: *Evaluation of anomaly-based IDS for mobile devices using machine learning classifiers*, Stevens Institute of Technology, Dissertation, 2011

[14] Farkas, Janos ; Bello, Lucia L. ; Gunther, Craig: Time-Sensitive Networking Standards. In: *IEEE Communications Standards Magazine* 2 (2018), Nr. 2, S. 20–21

[15] Haeckel, Timo ; Meyer, Philipp ; Korf, Franz ; Schmidt, Thomas C.: Secure Time-Sensitive Software-Defined Networking in Vehicles. (2022), Januar

[16] IBM: *IBM Security QRadar NDR*. – URL https://www.ibm.com/de-de/qradar/security-qradar-ndr. – Zugriffsdatum: 2022-07-08

[17] IEEE 802.1 Working Group: IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic / IEEE. March 2016 (Std 802.1Qbv-2015). – Forschungsbericht. – 1–57 S. ()

[18] IEEE 802.1 Working Group: IEEE Standard for Local and Metropolitan Area Network–Bridges and Bridged Networks / IEEE. Juli 2018 (Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014)). – Standard. – 1–1993 S

[19] Khraisat, Gondal I. Vamplew P. et a.: Survey of intrusion detection systems: techniques, datasets and challenges. In: *Cybersecur* (2019). – URL https://cybersecurity.springeropen.com/articles/10.1186/s42400-019-0038-7#citeas. – ISSN 2523-3246

[20] Khurat, Assadarat ; Sawangphol, Wudhichart: An Ontology for SNORT Rule. In: *2019 16th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2019, S. 49–55

[21] Kreutz, Diego ; Ramos, Fernando M. V. ; Verissimo, Paulo E. ; Rothenberg, Christian E. ; Azodolmolky, Siamak ; Uhlig, Steve: Software-Defined Networking: A Comprehensive Survey. In: *Proceedings of the IEEE* 103 (2015), Nr. 1, S. 14–76

[22] Li, Wenjuan ; Meng, Weizhi ; Kwok, Lam F.: A survey on OpenFlow-based Software Defined Networks: Security challenges and countermeasures.

In: *Journal of Network and Computer Applications* 68 (2016), S. 126–139. – URL https://www.sciencedirect.com/science/article/pii/S1084804516300613. – ISSN 1084-8045

[23] Liao, Hung-Jen ; Richard, Chun-Hung ; Lin, Ying-Chih ; Tung, Kuang-Yuan: Intrusion detection system: A comprehensive review. In: *Journal of Network and Computer Applications* (2012)

[24] Lokman, SF. ; Othman, A.T. ; Abu-Bakar, MH.: Intrusion detection system for automotive Controller Area Network (CAN) bus system: a review, 2019

[25] Maaloul, Rihab ; Taktak, Raouia ; Chaari, Lamia ; Cousin, Bernard: Energy-Aware Routing in Carrier-Grade Ethernet Using SDN Approach. In: *IEEE Transactions on Green Communications and Networking* 2 (2018), Nr. 3, S. 844–858

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

[27] OISF: *Suricata*. – URL https://suricata.io/documentation/. – Zugriffsdatum: 2022-07-08

[28] OISF: *Suricata Documentation*. – URL https://suricata.readthedocs.io/en/suricata-6.0.0/rules/intro.html

[29] OISF: *Suricata Documentation*. – URL https://docs.zeek.org/en/master/frameworks/signatures.html#signature-language-for-network-traffic

[30] OSSEC Project Team: *OSSEC*. – URL https://www.ossec.net. – Zugriffsdatum: 2022-07-08

[31] Park, W. and Ahn, S.: *Performance Comparison and Detection Analysis in Snort and Suricata Environment*

[32] Security Onion Solutions: *Security Onion*. – URL https://securityonionsolutions.com. – Zugriffsdatum: 2022-07-08

[33] Security Operations Center: *Sagan Log-Analysis Engine.* – URL https://sagan.readthedocs.io/en/latest/what-is-sagan.html. – Zugriffsdatum: 2022-07-08

[34] Smaha, Stephen E.: *Haystack: An Intrusion Detection System.* – URL https://homeostasis.scs.carleton.ca/~soma/id-2007w/readings/smaha-haystack.pdf

[35] Stein, Thomas: *Intrusion Detection System Evasion Durch Angriffsverschleierung in Exploiting Netzworks.* Diplomica Verlag, 2010

[36] Steinbach, Till ; Korf, Franz ; Schmidt, Thomas C.: Real-time Ethernet for automotive applications: A solution for future in-car networks. In: *2011 IEEE International Conference on Consumer Electronics -Berlin (ICCE-Berlin)*, 2011, S. 216–220

[37] Strobel, Stefan: *Firewalls und IT-Sicherheit.* dpunkt.verlag, 2003

[38] The Zeek Project: *Zeek.* – URL https://zeek.org. – Zugriffsdatum: 2022-07-08

[39] Vigna, G. ; Kemmerer, R.A.: NetSTAT: a network-based intrusion detection approach. In: *Proceedings 14th Annual Computer Security Applications Conference (Cat. No.98EX217)*, 1998, S. 25–34

# A Anhang

## 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 |
|---|---|---|