Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterthesis

Stephan Phieler

Misuse Detection für Echtzeit Ethernet basierte Kommunikationssysteme

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Stephan Phieler

Misuse Detection für Echtzeit Ethernet basierte Kommunikationssysteme

Masterthesis eingereicht im Rahmen der Masterprüfung

im Studiengang Master of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

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

Eingereicht am: 5. Mai 2017

Stephan Phieler

Thema der Arbeit
Misuse Detection für Echtzeit Ethernet basierte Kommunikationssysteme

Stichworte
Time-Triggered Ethernet, Informationssicherheit, Intrusion Detection, Misuse Detection, Signaturnetz

Kurzzusammenfassung
Das Time-Triggered-Ethernet (TTE) ist eine fehlertolerante Netzwerktechnologie, welche fast alle Anforderungen zukünftiger Bordnetze erfüllt. Die Informationssicherheit ist eine Anforderung, die nicht erfüllt wird. Analysen der Entwickler haben die Schwachstellen bereits benannt. Harte Echtzeitanforderungen und limitierte Ressourcen machen den Einsatz kryptographischer Verfahren zur Signierung oder Verschlüsselung aber teilweise unmöglich. Um trotzdem auf Angriffe auf die Informationssicherheit reagieren zu können, kommt eine Überwachung durch ein Intrusion Detection System (IDS) in Betracht. In dieser Arbeit wird ein Konzept für die Überwachung des TTEs mit Hilfe einer Misuse Detection (MD) vorgestellt, welche für die Beschreibung der Angriffe Signaturnetze verwendet.
Stephan Phieler

Title of the paper
Misuse detection for realtime-ethernet based communication

Keywords
Time-Triggered Ethernet, Security, Intrusion Detection, Misuse Detection, Signature net

Abstract
The Time-Triggered-Ethernet (TTE) is an upcoming network technology with fault tolerance in mind. It covers almost all requirments for the automotive network except for security. Analysis have already identified the security vulnerabilities. Hard real-time requirements and limited ressources lead to restricted usage of cryptographic security functions due to processing time. Network monitoring is another option to increase the security of a network, without affecting the communication. Hence in this work a concept for network monitoring based on misuse detection via an intrusion detection system is presented, which relies on signature nets for attack modeling.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1. Einleitung und Motivation

Die Elektrifizierung des Autos schreitet immer mehr voran. So besteht ein Auto heute aus einer Vielzahl an Computern (Electronic Control Units (ECUs)), Sensoren und Aktoren. Mehr und mehr Systeme werden durch Software unterstützt, durch diese ersetzt oder neu hinzugefügt. Diese Systeme werden durch ein Netzwerk (Bordnetz), wie dem Controller Area Network (CAN)-Bus oder Flexray, untereinander verbunden, um Informationen austauschen zu können. Im Laufe der Zeit wurden, bei unzureichender Kapazität der vorhandenen Netzwerke, immer neue Netzwerke hinzugefügt, welche über Gateways an die bestehenden Netzwerke angeschlossen wurden. Somit entstand ein großes heterogenes Netzwerk. Da der Bedarf an Bandbreite immer noch ansteigt, ausgelöst durch neue Assistenzsysteme, dem Einzug des Internets in das Auto und der aufkommenden Car-to-Car (C2C)- und Car-to-Environment (C2E)-Kommunikation, kommen die aktuell eingesetzten Netzwerktechnologien an ihr Limit. Die zunehmende Vernetzung und das steigende Interesse an den Daten, führt auch zu der Notwendigkeit die Daten abzusichern. Nicht nur der Schutz der Nutzerdaten, sondern auch die sichere Kommunikation von Steuerdaten, ist wichtig. Aktuelle Bordnetze verfügen oft nur über rudimentäre oder gar keine Sicherheitsvorkehrungen (vgl. Wolf, 2009; Henniger u. a., 2009; Koscher u. a., 2010; Checkoway u. a., 2011; Tillich und Wójcik, 2012; Studnia u. a., 2013a,b).

Mit dem Ethernet gibt es eine aus den Computernetzwerken bekannte Technologie, welche über eine sehr hohe Bandbreite verfügt. Dadurch kann nicht nur der zukünftige Bedarf an Bandbreite gedeckt, sondern auch das heterogene, bestehende Bordnetz aufgelöst werden, da alle Teilnetzwerke zu einem Einzigen zusammengefasst werden können. Im Gegensatz zum CAN-Bus oder Flexray, können beim Ethernet aber keine Garantien über das rechtzeitige, geschweige denn über das eigentliche Eintreffen einer Nachricht abgegeben werden. Diese Eigenschaft ist aber eine Grundvoraussetzung für den Einsatz als Bordnetz im Auto. Denn auch kritische Daten, wie die für das Auslösen des Airbags, werden über das Bordnetz übertragen. Daher wurden verschiedene Lösungen entwickelt, um das Ethernet echtzeitfähig zu machen. Darunter fallen das TTE (vgl. SAE - AS-2D Time Triggered Systems and Architecture Committee, 2009), das Time Sensitive Network (TSN) (vgl. TSNTaskGroup) und weitere in der

Automatisierung im Industriebereich vertretene Lösungen. Bei allen lag der Schwerpunkt vor allem auf der Funktionalität der Echtzeitfähigkeit, nicht aber auf der Datensicherheit.

Verschiedene Forscher haben sich daher dem Thema angenommen und neben den Schwachstellen und Problemen (vgl. Hirschler und Treytl, 2011; Studnia u. a., 2013a; Steiner, 2013) auch erste Lösungsansätze vorgestellt (vgl. Wasicek, 2011; Wasicek u. a., 2011; Isakovic, 2011a) um die Informationssicherheit zu gewährleisten. Harte Echtzeitanforderungen und die limitierten Ressourcen der verwendeten Recheneinheiten erschweren aber den Einsatz von kryptographischen Verfahren. Zudem müsste die Spezifikation des TTEs für die Sicherheitsanforderungen angepasst werden. Daher sollten auch andere Herangehensweisen untersucht werden. Eine davon ist die Überwachung des Netzwerkes, um, ähnlich wie bei einem Anti-Virus-Programm, Angriffe zu identifizieren und anschließend darauf zu reagieren.

Für LANs und WANs gibt es dahingehend schon eine große Menge an Erkenntnissen durch wissenschaftliche Arbeiten. Für das Bordnetz werden, wenn überhaupt, nur rudimentäre Überwachungsmöglichkeiten eingesetzt, welche zudem nicht in wissenschaftlichen Arbeiten veröffentlicht wurden (vgl. dragTimes, 2014). Daher soll diese Arbeit untersuchen, ob eine Überwachung des Bordnetzes realisierbar ist und ob sie zur Steigerung der Sicherheit beitragen kann.

Umgesetzt wird die Überwachung mit einer Intrusion Detection (ID). Diese gibt es bereits seit den 1980er Jahren (vgl. Rittinghouse und Hancock, 2003). Sie wird für das automatisierte Erkennen von Sicherheitsverletzungen eingesetzt. Dazu wird das Verhalten eines elektronischen Systems überwacht. Man unterscheidet dabei zwischen der Anomaly Detection und der Misuse Detection. Bei der Anomaly Detection (AD) wird das Normalverhalten des Systems definiert und alles, was sich nicht wie in der Definition beschrieben verhält, wird als Angriff oder Fehler interpretiert. Die MD geht den umgekehrten Weg. Hier wird das Fehlverhalten beschrieben. Alles andere wird als Normalverhalten eingestuft. Beide Ansätze haben Vor- und Nachteile, die in Kapitel 2 beschrieben werden.

Eine Auflistung von Angriffen und Schwachstellen für das TTE wurde von Steiner (2013) veröffentlicht. Diese werden als Grundlage für die Beschreibung des Fehlverhaltens genutzt. Dazu wird in dieser Arbeit der Einsatz der MD untersucht, da die Schwachstellen bekannt sind und sich diese für ein System mit Echtzeitanforderungen und begrenzten Ressourcen in einer ersten Betrachtung besser eignet, als eine AD.

Zielsetzung

In dieser Arbeit wird untersucht, ob ein Intrusion Detection System mit dem Schwerpunkt auf der MD zur Sicherung des Netzwerkverkehrs in einem Time-Triggered-Ethernet beitragen kann. Es wird diskutiert, welche Daten erhoben werden müssen und wo diese gespeichert werden, wie die Daten zeitnah in verwertbare Informationen umgewandelt werden können und wie das IDS in das Bordnetz integriert werden kann. Zudem werden mögliche Reaktionen auf einen erkannten Angriff vorgestellt.

Aufbau der Arbeit

Die Grundlagen zum TTE und zur ID werden in Kapitel 2 beschrieben. Anschließend werden in Kapitel 3 die Schwachstellen und die zu schützenden Informationen vorgestellt. Dann wird, anhand der aufgestellten Anforderungen, dass in dieser Arbeit entwickelte IDS diskutiert. Zum Abschluss wird eine Zusammenfassung über die erlangten Erkenntnisse dieser Arbeit gegeben.

2. Grundlagen

In diesem Kapitel werden die Grundlagen für Kapitel 3 vermittelt. Als erstes wird das Time-Triggered-Ethernet (TTE) näher erläutert. Es werden die Bereiche behandelt, welche Schwachstellen nach Steiner (2013) aufweisen, bzw. welche für die Erstellung der MD benötigt werden. Die Schwachstellen selbst werden im anschließenden Kapitel behandelt. Der zweite Teil der Grundlagen befasst sich mit der Intrusion Detection (ID), mit dem Schwerpunkt auf der Misuse Detection (MD). Dabei wird eine kurze strukturelle Einführung in die ID gegeben. Anschließend werden Verfahren zur Signaturbeschreibung für eine MD vorgestellt, mit denen die Signaturen zu den Schwachstellen des TTE beschrieben werden können. Aus diesen Verfahren wurden die Signaturnetze zur Beschreibung ausgewählt, deren Syntax im letzten Abschnitt behandelt wird.

2.1. Time-Triggered Ethernet

In diesem Abschnitt wird das TTE (vgl. SAE - AS-2D Time Triggered Systems and Architecture Committee, 2009) näher vorgestellt. Neben der Fähigkeit zur Echtzeitkommunikation, wird auch besprochen, welchen Schutzbedarf das TTE hat und welche Maßnahmen dahingehend bereits ergriffen oder vorgestellt wurden.

Das TTE ist vollständig kompatibel zum IEEE 802.3 Ethernet. Es nutzt die Sterntopologie, bei dem alle Endknoten über Switches miteinander verbunden werden. Das TTE besitzt eine statische Konfiguration, in der Informationen über Sende- und Empfangszeiten sowie die Routen der Nachrichten hinterlegt sind. Die Nachrichten werden dabei in drei Kategorien unterschieden, welche in Abschnitt 2.1.1 vorgestellt werden.

Das TTE besitzt einen High-Integrity-Modus, bei dem gewährleistet wird, dass nur gültige Aktionen ausgeführt werden, so dass fehlerhafte Komponenten keinen Schaden anrichten können. Dazu wird die Commander/Monitor (COM/MON)-Funktion verwendet. Mit dieser Funktion wird der Commander, der für die Ausführung von Aktionen im TTE-Stack zuständig ist, daraufhin überwacht, dass er so arbeitet, wie es in der Konfiguration hinterlegt wurde. Dafür führt der Monitor, welcher sich auf dem selben Teilnehmer befindet, die gleichen Aktionen

aus wie der Commander und vergleicht anschließend die Ergebnisse. Treten Differenzen auf, deaktiviert der Monitor den Commander.

Eine typische Aktion ist das Empfangen oder Senden einer Nachricht. Dafür müssen, ausgenommen vom Payload, alle Informationen über die Nachricht bekannt sein. Dazu zählen z.B. der Sende-/Empfangsport, die Zeit sowie die Route der Nachricht. Der Monitor ist aber nur in der Lage zwischen zulässigen und unzulässigen Aktionen zu unterscheiden. Angriffe bestehend aus zulässigen Aktionen findet er hingegen nicht.

2.1.1. Nachrichtenklassen

Das TTE beinhaltet drei Nachrichtenklassen, die unterschiedliche Anforderungen erfüllen und welche nachfolgend vorgestellt werden.

Time-Triggered-Nachrichten

Die Time-Triggered-Nachrichten haben die höchste Priorität im TTE und werden nach dem Time Division Multiple Access (TDMA)-Verfahren zu festgelegten Zeitpunkten versendet und empfangen. Die Nutzung von Time-Triggered (TT)-Nachrichten garantiert sowohl eine konstante Latenz als auch einen minimalen Jitter, welcher im Mikro- oder Nanosekundenbereich liegt. Damit die Garantien eingehalten werden können, muss sichergestellt werden, dass exklusiver Zugriff auf das Übertragungsmedium besteht. Die Voraussetzung dafür ist, dass jedem Teilnehmer bekannt sein muss, wann eine Nachricht gesendet wird und welchen Weg sie nehmen wird. Dazu muss das Netzwerk entsprechend konfiguriert und die Zeit aller Teilnehmer synchronisiert sein.

TT-Nachrichten besitzen einen eigenen Header, welcher in Abbildung 2.1 zu sehen ist. Die Nachrichten werden durch den Eintrag $0x88d7$ im Type Field gekennzeichnet. Das Routing der Nachricht erfolgt anhand der eindeutigen Virtual Link Identifier (VL ID). Da alle Routen und alle Zeitfenster bereits in der statischen Konfiguration ausgewiesen sind, können nur Nachrichten mit der richtigen VL ID im zugehörigen Zeitfenster versendet oder empfangen werden. Alle anderen Nachrichten werden spätestens beim Empfänger verworfen. TT-Nachrichten eignen sich vor allem für periodisch zu sendende Steuerdaten. Zur Zeit besteht kein Schutz zur Wahrung der Integrität oder Authentizität.

Aufgrund von zeitlichen Einschränkungen in der Verarbeitung der Nachrichten, können diese nicht einfach durch Verschlüsselungstechniken geschützt werden. Wasicek u. a. (2011) weist in seiner Arbeit darauf hin, dass ein sicherer Schlüsselaustausch und die Verwendung von sicheren Schlüsseln zum Schutz der Integrität und Authentizität unverhältnismäßig viel Zeit in Anspruch

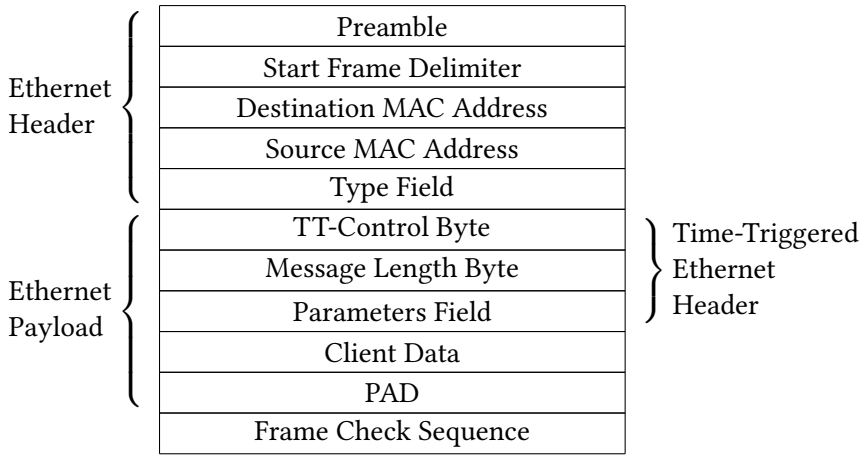

Abbildung 2.1.: Format einer Time-Triggered-Nachricht

nimmt. In seiner Lösung werden Nachrichten erst nachträglich einer Sicherheitsüberprüfung unterzogen. Für manche Anwendungen kann dies aber zu spät sein. Zum Beispiel, wenn ein Angreifer den Airbag auslöst um einen Unfall zu forcieren.

Auch ein IDS wird die Nachrichten nicht schneller auswerten können, als die Applikation. Sie erlaubt aber zumindest die Untersuchung dessen.

Rate-Constrained Nachrichten

Rate-Constrained (RC)-Nachrichten entsprechen dem ARINC664-7 Standard. Da die Nachrichten ereignisbasiert auftreten und priorisierbar sind, kann zwar die maximale Latenz bestimmt werden, nicht aber ein minimaler Jitter. Eine Bandwitdh Allocation Gap (BAG) sorgt dafür, dass nur eine bestimmte Menge an Nachrichten übertragen werden kann, so dass genügend Bandbreite für alle Teilnehmer zur Verfügung steht. Wird die BAG nicht eingehalten, wird die Nachricht beim Empfänger verworfen. Da RC-Nachrichten von höher priorisierten RC-Nachrichten verdrängt werden können, kann keine genaue Aussage über die Latenz oder den Jitter getroffen werden. Berechnet werden, können nur die maximalen Obergrenzen. Anders als bei den TT-Nachrichten ist keine Synchronisation der Teilnehmer notwendig.

Best-Effort Nachrichten

Die dritte und am niedrigsten priorisierte Klasse bilden die Best Effort-Nachrichten, welche den Standard-Ethernet-Nachrichten entsprechen. Sie nutzen die übrige Bandbreite, welche nicht von TT- oder RC-Nachrichten verwendet wird. Da bei Best Effort (BE)-Nachrichten keine Garantien über eine erfolgreiche Zustellung gegeben werden können, sind auch keine Aussagen

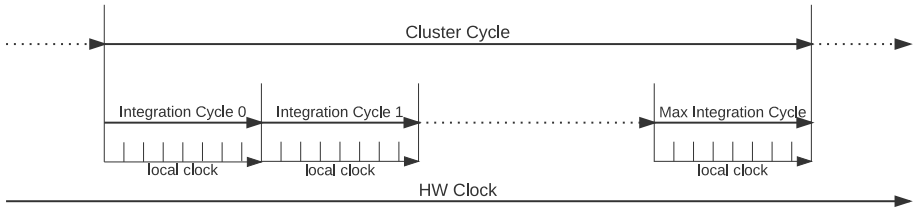

Abbildung 2.2.: TTE Cluster Cycle

über die Latenz oder den Jitter möglich. Daher werden diese Nachrichten ausschließlich für Daten ohne Echtzeitanforderungen genutzt.

2.1.2. Synchronisation

Für den Einsatz von TT-Kommunikation muss die Zeit der Teilnehmer synchronisiert werden, so dass die konfigurierten Zeitpunkte für das Senden und Empfangen eingehalten werden können. Das TTE beinhaltet zwar ein eigenes Protokoll zur Synchronisierung der Teilnehmer, es kann aber auch ein beliebiges anderes Protokoll genutzt werden. Nachfolgend wird das im TTE enthaltene Protokoll beschrieben. Jeder Teilnehmer nimmt eine bestimmte Rolle im Synchronisationsprozess ein. Er ist entweder ein Synchronisation Client (SC), ein Synchronisation Master (SM) oder ein Compression Master (CM). Alle Rollen werden in der Konfiguration statisch festgelegt.

Synchronisation Master sind Teilnehmer mit einer sehr genauen Uhr. Sie stellen die Zeitbasis in der Synchronisation dar und senden ihre lokale Zeit an die CM.

Compression Master berechnen anhand der Zeit der SM eine globale Zeit und propagieren sie anschließend an alle Teilnehmer. Diese Funktion wird von Teilnehmern mit einer guten topologischen Lage[1] und ausreichend Ressourcen übernommen.

Synchronisation Clients nehmen nicht aktiv an der Synchronisation teil. Sie konsumieren lediglich die vom CM erhaltenen Nachrichten und passen ihre lokale Uhr an die neue Zeit an.

[1] Zentral platzierte Teilnehmer haben eine topologisch gute Lage.

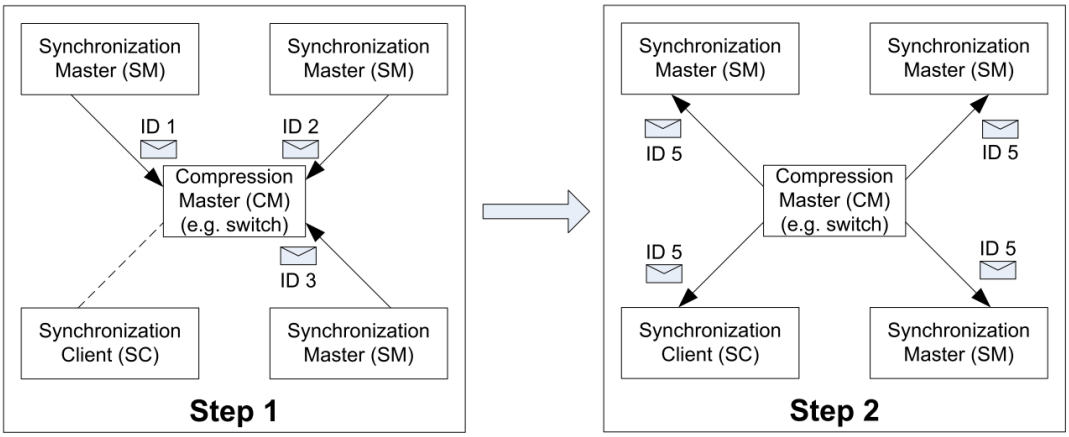

Abbildung 2.3.: Synchronisation im TTE (Quelle: SAE - AS-2D Time Triggered Systems and Architecture Committee (2009))

Synchronisierungsprotokoll

Der Ablauf des Synchronisierungsprotokolls ist in Abbildung 2.3 dargestellt. In Schritt 1 senden alle SM eine Nachricht an den CM. Dieser ermittelt mit Hilfe der Compression-Funktion die neue Zeit. Anschließend sendet der CM eine Nachricht an alle Teilnehmer, welche sich daraufhin mit dem CM synchronisieren. Im ersten Durchlauf verwendet der CM die Zeit, welche am weitesten fortgeschritten ist. In jedem weiteren Durchlauf berechnet er einen Durchschnittswert der erhaltenen Zeiten (siehe Formel 2.2).

Repräsentation der Zeit

Im TTE werden Zeitpunkte durch den Integration Cycle und die Local Clock abgebildet und unter dem Cluster Cycle zusammengefasst (siehe Abbildung 2.2). Die Local Clock ist ein einfacher Zähler und teilt den Integration Cycle in gleich große Zeitabschnitte auf. Ist der maximale Wert für einen Integration Cycle erreicht, beginnt dieser wieder bei Null. In der Konfiguration ist festgelegt, zu welchem Zeitpunkt (Local Clock) im Integration Cycle ein Teilnehmer TT-Nachrichten verschicken, bzw. empfangen darf.

Protocol Control Frames

Für die Synchronisierung steht mit den Protocol Control Frames (PCFs) eine eigene Nachrichtenklasse zur Verfügung. Sie basiert auf Standard-Ethernet-Nachrichten, hat eine Größe von 46 Byte und wird durch den Type $0x891d$ gekennzeichnet (vgl. SAE - AS-2D Time Trigge-

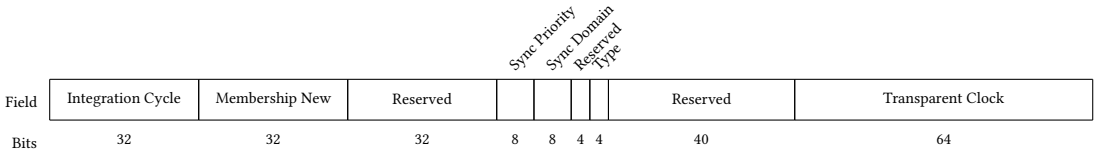

Abbildung 2.4.: Protocol Control Frame

red Systems and Architecture Committee, 2009, S. 30). Sie besitzen eine höhere Priorität als RC-Nachrichten, damit die Aufrechterhaltung der TT-Kommunikation gewährleistet werden kann. Abbildung 2.4 zeigt den Aufbau eines PCF. Wie zu sehen ist, gibt es kein Feld, in dem ein Zeitwert abgelegt werden kann, welcher einen bestimmten Zeitpunkt repräsentiert. Stattdessen wird die Übertragungszeit, mit Hilfe des Transparent Clock-Feldes, und der aktuelle Zyklus des Teilnehmers, im Integration Cycle-Feld, übermittelt.

Um die Übertragungszeit zu ermitteln, wird die aus IEEE 1588 Precision Time Protocol (PTP) bekannte Funktion der Transparent Clock (TC) verwendet. Die TC ermittelt aus den dynamischen und statischen Verzögerungszeiten die Gesamtverzögerung. Dieser Wert wird im TC-Feld des PCF abgelegt. Immer wenn ein PCF von einem Teilnehmer weitergeleitet werden soll, speichert dieser den Empfangs- und den Sendezeitpunkt. Aus der Differenz berechnet er die dynamische Aufenthaltsdauer. Diese addiert er mit der statischen Empfangsverzögerung, der statischen Übertragungszeit sowie mit der Zeit aus dem TC-Feld des empfangenen PCF und schreibt diese Zeit anschließend in das TC-Feld des neuen PCFs.

$$t_{send} = t_{receive} + (t_{maximum\ delay} - t_{delay}) \tag{2.1}$$

Der Empfänger des PCFs kann den Sendezeitpunkt mit Hilfe der Formel 2.1 ermitteln. Eine kleine Ungenauigkeit bleibt aber, da nicht der genaue Absendezeitpunkt des PCFs des Senders zur Berechnung der dynamischen Aufenthaltsdauer verwendet werden kann, da das TC-Feld vor dem Absenden befüllt werden muss. Das Prinzip setzt voraus, dass jeder Teilnehmer, der PCFs weiterleitet, das TC-Feld bearbeiten kann.

Neben der Übertragungszeit werden in einem PCF auch diejenigen SM gespeichert, die zu dem jeweiligen CM synchron sind. Dazu wird das Membership New-Feld genutzt. In diesem ist ein Bit-Vektor enthalten, bei dem alle synchronen SM durch ein positives Bit repräsentiert werden. Die Zuordnung, welches Bit zu welchem SM gehört, ist jedem Teilnehmer bekannt. Bekommt ein SM oder SC mehrere PCF von verschiedenen CM, wird der PCF zur Synchronisation genutzt, der die meisten positiven Einträge enthält. Es ist auch möglich PCFs zu priorisieren, so dass bestimmte CM für bestimmte Domänen im Netzwerk zuständig sind. Jeder

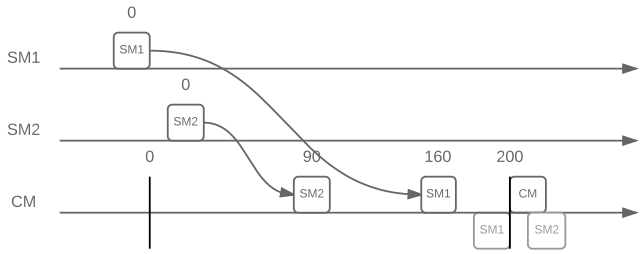

Abbildung 2.5.: Zeitberechnung TTE

Teilnehmer hat eine lokale Konfiguration, in der die Priorität der PCFs hinterlegt ist, welche von dem Teilnehmer verarbeitet werden dürfen. Teilnehmer haben die Möglichkeit zur Laufzeit zu entscheiden, ob Sie auch PCFs verarbeiten, die eine höhere Priorität haben als die lokal konfigurierte.

Berechnung der globalen Zeit

$$\begin{aligned}
\text{Ein SM: } & Compression_correction = input_1 \\
\text{Zwei SM: } & Compression_correction = \frac{input_1 + input_2}{2} \\
\text{Drei SM: } & Compression_correction = input_2 \\
\text{Mehr als drei SM: } & Compression_correction = \frac{input_k + input_{pcfquantity-k}}{2}
\end{aligned} \tag{2.2}$$

In der Konfiguration ist festgelegt, wann PCFs im Integration Cycle versendet werden. Aus diesem Grund kann der CM davon ausgehen, dass die Subtraktion der Übertragungszeit von der Empfangszeit gleich die Sendezeit ist. Das Beispiel aus Abbildung 2.5 und Formel 2.2 zeigen, wie die Zeitberechnung funktioniert.

Über den Nachrichtenblöcken ist der Sende- bzw. Empfangszeitpunkt abgebildet. Das k aus Formel 2.2 kann frei gewählt werden. Es gibt an, welche PCFs von dem CM zur Berechnung genutzt werden, wobei der $k$te PCF von vorn und der $k$te PCF von hinten genutzt werden. Die Liste mit den PCFs ist nach den Empfangszeitpunkten sortiert. Bei k = 3 und 10 SM fließen somit $input_3$ und $input_8$ als Parameter in die Formel ein. Damit wird sichergestellt, dass Ausreißer keinen Einfluss auf die Synchronisation haben. Für einen Angreifer bedeutet das, dass er mindestens k PCFs manipulieren muss um Einfluss auf die Synchronisation zu haben.

Da jeder Teilnehmer den Sendezeitpunkt des PCFs des CMs bzw. der SMs kennt und die statische Laufzeit der Nachrichten bekannt ist, öffnet jeder empfangsbereite Teilnehmer um den antizipierten Empfangszeitpunkt herum ein Observation Window. Außerhalb eintreffende

PCFs werden nicht zur Synchronisierung benutzt, da dies darauf hinweist, dass der Teilnehmer nicht mehr synchron ist und sich komplett neu synchronisieren muss.

Zudem steht im TTE eine Clique Detection zur Verfügung. Sie ist dafür zuständig, Gruppen von Teilnehmern zu identifizieren, die untereinander synchron sind, aber nicht zum Rest des Netzwerkes. Dies kann bei dem Einsatz von mehreren CM auftreten. Ist eine Clique identifiziert worden, werden entweder nur die betroffenen Teilnehmer oder das gesamte Netzwerk neugestartet.

2.1.3. Sicherheitseigenschaften

TTE ist ein fehlertolerantes Kommunikationsprotokoll, welches allerdings keine Regeln zur Nutzung und Integration von Sicherheitsfunktionen und zum Schutz der Informationssicherheit bestimmt hat (vgl. Steiner, 2013). Es hat aber durch seine statische Struktur einige sicherheitsrelevante Eigenschaften. So ist es nicht möglich unbekannte TT-Nachrichten zu versenden. Es sind nur solche Nachrichten zulässig, welche vorher in der statischen Konfiguration festgelegt wurden. Alle anderen Nachrichten werden beim Empfänger sofort verworfen. Auch das Nutzen von freien Ports an einem Switch ist nicht möglich, da auch hier nur die in der statischen Konfiguration festgelegten Ports zum Empfang und Versand benutzt werden können. Somit kann das Netzwerk auch nicht mit TT-Nachrichten geflutet werden.

Für RC-Nachrichten bestehen ähnliche Restriktionen. Nur Nachrichten mit einer bekannten VL ID werden verarbeitet. Nachrichten die ihr BAG nicht einhalten, werden vom Empfänger abgewiesen. Bei PCFs sorgt der Membership New-Vektor dafür, dass sich Teilnehmer nur mit CMs synchronisieren, die eine gewisse Güte vorweisen können. Trotzdem sind Angriffe auf das TTE und dessen Synchronisation möglich. Ein Angreifer kann z.B. die Informationen der Nachrichten beliebig manipulieren oder Nachrichten unerkannt löschen. Genaue Szenarien werden in Kapitel 3 vorgestellt.

2.1.4. Zusammenfassung

Das TTE ist kompatibel zum Standard Ethernet 802.3 und verwendet die Stern-Topologie. Es besitzt zwei neue Nachrichtenklassen für die zeitgesteuerte (Time-Triggered (TT)) und die ereignisgesteuerte (Rate-Constrained (RC)) Kommunikation. Für die RC-Kommunikation, welche auf dem ARINC664-7 Standard basiert, wird eine festgelegte Bandbreite garantiert. Bei der TT-Kommunikation wird eine freie Übertragungsstrecke zu einem vorgegebenen Zeitpunkt garantiert. Die Routen der Nachrichten und die zeitlichen Grenzwerte sind in einer statischen Konfiguration festgelegt. Die Synchronität der Teilnehmer bildet die Grundlage für die TT-

Kommunikation. Dafür enthält das TTE ein Synchronisationsprotokoll mit einer eigenen Nachrichtenklasse, den Protocol Control Frames (PCFs). Das TTE enthält keinen Schutz für die Authentizität und Integrität der Nachrichten, hat durch seine statische Struktur aber einige Restriktionen, was das Senden und Empfangen von Nachrichten betrifft. Zur Überwachung der Teilnehmer setzt das TTE einen Monitor ein, der die Aktionen des Commanders überwacht (Commander/Monitor (COM/MON)). Der Monitor kann aber nur Aktionen erkennen, welche nicht den Vorgaben der statischen Konfiguration entsprechen. Für die Überwachung des Netzwerkes zur Erkennung von Angriffen, bietet das TTE keine Funktionen an.

2.2. Intrusion-Detection-Systeme

Intrusion Detection (ID) ist das Überwachen und die Analyse von Ereignissen in einem Computersystem oder Netzwerk, um ein Eindringen feststellen zu können, welches unter der Absicht durchgeführt wird, die Schutzziele der Informationssicherheit (Vertraulichkeit, Integrität, Verfügbarkeit) anzugreifen (vgl. Bace und Mell, 2001). Sie besteht grob aus drei Abschnitten: Das Sammeln der Daten (Abschnitt 2.2.2), das extrahieren von Informationen aus den Daten (Abschnitt 2.2.3) und die Reaktion auf einen Angriff (Abschnitt 2.2.4). Diese Vorgehensweise wird auch unter dem Begriff Angriffssprache (Attack Language) zusammengefasst.

2.2.1. Angriffssprache

Vigna u. a. (2000b) unterteilt die Angriffssprache noch spezifischer, in die folgenden sechs Kategorien (siehe Abbildung 2.6):

- Event Language
- Response Language
- Reporting Language
- Correlation Language
- Exploit Language
- Detection Language

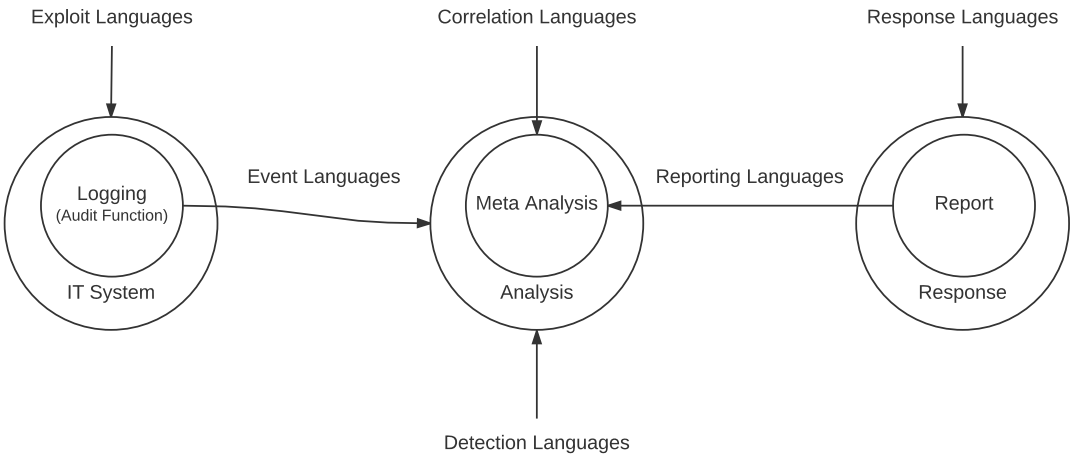

Abbildung 2.6.: Angriffssprachen nach Meier (2008) S. 25

Tabelle 2.1.: Standard-Nachrichtenklassen im IDMEF RFC4765

Nachricht	Feld	Feld
Alert	messageid	
	Analyzer	
	AnalyzerTime	
	CreateTime	
	DetectTime	
	Source	Node
		User
		Process
		Service
	Target	Node
		User
		Process
		Service
		File
	Classification	
	Assessment	
	AdditionalData	
Heartbeat	messageid	
	Analyzer	
	CreateTime	
	AdditionalData	

Tabelle 2.2.: Eventstruktur Microsoft Windows

Typ	Beschreibung
EventType	
BinaryEventData	Contains the event data as a binary blob.
DebugData	Contains the data that can be logged for Windows software trace preprocessor events.
EventData	Contains the event data.
ProcessingErrorData	Contains details of the error that occurred while trying to render the event.
RenderingInfo	Contains the rendered message strings for the event (includes the event's message string and the message strings for any of the event's properties such as level, task and opcode).
System	Contains information that identifies the provider and how it was enabled, the event, the channel to which the event was written and system information such as the process and thread IDs.
UserData	Contains the event data.
EventDataType	
Binary	A binary data blob for events that are written using Event Logging.
ComplexData	A structure that is defined in the template for the event.
Data	A top-level data item that is defined in the template for the event.
RenderingInfoType	
Channel (message)	The rendered message string of the channel specified in the event.
Keywords (list)	A list of rendered keywords.
Level	The rendered message string of the level specified in the event.
Message	Contains the event message that is rendered for the event.
Opcode	The rendered message string of the opcode specified in the event.
Provider	The rendered message string for the provider.
Task	The rendered message string of the task specified in the event.
SystemPropertiesType	
Channel	The channel to which the event was logged.
Computer	The name of the computer on which the event occurred.
Correlation	The activity identifiers that consumers can use to group related events together.
EventID	The identifier that the provider used to identify the event.
EventRecordID	The record number assigned to the event when it was logged.
Execution (process)	Contains information about the process and thread that logged the event.
Keywords (list)	A bitmask of the keywords defined in the event.
Level	Contains the severity level of the event.
Opcode	The opcode defined in the event.
Provider	Identifies the provider that logged the event.
Security (user)	Identifies the user that logged the event.
Task	The task defined in the event.
TimeCreated	The time stamp that identifies when the event was logged.
Version	Contains the version number of the event's definition.
DebugDataType	

Weiter auf der nächsten Seite

Tabelle 2.2: Eventstruktur Microsoft Windows - Fortsetzung

Typ	Beschreibung
Component	The name of the component that logged the trace message.
FileLine	The name of the source file and the line within the source file that logged the trace message.
FlagName	The flag value passed to the provider when it was enabled.
Function	The name of the function that logged the trace message.
LevelName	The level value passed to the provider when it was enabled.
Message	The message string. The XML contains this element if the WPP event specified the FormattedString field.
SequenceNumber	The local or global sequence number of the trace message.
SubComponent	Specifies the SubComponentName WPP debug tracing field that is used in debug events in debug channels.
ProcessingErrorDataType	
DataItemName	Contains the name of the event data item that caused an error when the event data was processed.
ErrorCode	Contains the error code that was raised when there was an error processing event data.
EventPayload	Contains binary event data for the event that caused an error when the event data was processed.
ComplexDataType	
Data (list)	The list of data items in the structure. The list of items are in the same order as defined in the template.
Keywords	
Keyword	Contains a rendered keyword.

Tabelle 2.3.: Eventstruktur IBM Datenbank

Element	Beschreibung
Timestamp	Date and time of the audit event.
Category	Category of audit event.
Audit Event	Specific Audit Event.
Event Correlator	Correlation identifier for the operation being audited. Can be used to identify what audit records are associated with a single event.
Event Status	Status of audit event
Database Name	Name of the database for which the event was generated.
User ID	User ID at time of audit event.
Authorization ID	Authorization ID at time of audit event.
Origin Node Number	Node number at which the audit event occurred.
Coordinator Node Number	Node number of the coordinator node.
Application ID	Application ID in use at the time the audit event occurred.
Application Name	Application name in use at the time the audit event occurred.
Package Schema	Schema of the package in use at the time of the audit event.

Weiter auf der nächsten Seite

Tabelle 2.3: Eventstruktur IBM Datenbank - Fortsetzung

Element	Beschreibung
Package Name	Name of package in use at the time the audit event occurred.
Package Section Number	Section number in package being used at the time the audit event occurred.
Object Schema	Schema of the object for which the audit event was generated.
Object Name	Name of object for which the audit event was generated.
Object Type	Type of object for which the audit event was generated.
Access Approval Reason	Indicates the reason why access was approved for this audit event.
Access Attempted	Indicates the type of access that was attempted.
Package Version	Version of the package in use at the time that the audit event occurred.
Checked Authorization ID	Authorization ID is checked when it is different than the authorization ID at the time of the audit event.
Local Transaction ID	The local transaction ID in use at the time the audit event occurred.
Global Transaction ID	The global transaction ID in use at the time the audit event occurred.
Client User ID	The value of the CURRENT CLIENT USERID special register at the time the audit event occurred.
Client Workstation Name	The value of the CURRENT CLIENT_WRKSTNNAME special register at the time the audit event occurred.
Client Application Name	The value of the CURRENT CLIENT_APPLNAME special register at the time the audit event occurred.
Client Accounting String	The value of the CURRENT CLIENT_ACCTNG special register at the time the audit event occurred.
Trusted Context Name	The name of the trusted context associated with the trusted connection.
Connection Trust Type	-
Role Inherited	The role inherited through a trusted connection.

Event Language

Die Event Language beschreibt Informationen zu eingetretenen Ereignissen. Zur Zeit gibt es keine einheitliche Definition, wie diese Ereignisse abgebildet werden sollen, stattdessen besitzt jedes System (Operating System (OS), Programm, ...) ein eigenes Format. Die beiden Tabellen 2.2 und 2.3 sind zwei Beispiele für die unterschiedlichen Ansätze zur Umsetzung einer Event Language.

Bei Microsoft Windows wird ein komplettes Abbild des Systemzustands (Snapshot) zu einem eingetretenen Ereignis erstellt. Dies würde sehr viel Speicherplatz beanspruchen, wenn alle Ereignisse des Windows Systems aufgezeichnet würden. Auch bei einem Bordnetz mit vielen Teilnehmern kann eine hohe Anzahl an Events erwartet werden. Deswegen werden nur für vorher festgelegte Ereignisse Snapshots des Systems erstellt. Ein Beispiel dafür ist die Fehleraufzeichnung (Error Logging).

Die zweite Tabelle zeigt die Eventstruktur für eine IBM Datenbank. Auch hier wird eine große Menge an Daten erhoben, so dass detailliert nachvollzogen werden kann, wer welches

Ereignis zu welcher Zeit in welchem Systemzustand ausgelöst hat. Auch hier werden nur zu ausgewählten Ereignissen Snapshots erstellt.

Das Aufzeichnen von Ereignissen in einem Automobilnetzwerk muss neben der Aufzeichnung aller relevanten Informationen auch dahingehend optimiert werden, dass nur wichtige Ereignisse detailliert aufgezeichnet werden. Die Schwierigkeit besteht darin diese Ereignisse zu bestimmen. Es bietet sich zudem an, die Snapshots für eine offline Analyse vorzuhalten. Denn die Auswertung der erhobenen Informationen kann je nach Komplexität viel Zeit in Anspruch nehmen und muss ggf. von einem Experten durchgeführt werden.

Es gab das Bestreben, ein einheitliches Format zu definieren, welches zum Ziel die Entkopplung von System und IDS hatte. So hat Bishop (1995) in seiner Arbeit ein Format für die Abbildung beschrieben und auch die Internet Engineering Task Force (IETF) hatte mit dem Intrusion Detection Message Exchange Format (IDMEF) (RFC4765), aus Tabelle 2.1, ein eigenes Format im Standardisierungsprozess (vgl. Feinstein u. a., 2007). Das IDMEF wurde zwar bis heute nicht standardisiert, kann aber als Referenz für ein unabhängiges Format genutzt werden. Die IDMEF ist generischer als die beiden Beispiele von Microsoft und IBM und kann genau auf ein spezifisches Einsatzgebiet angepasst werden, womit sich wiederum Speicherplatz sparen lässt.

Response Language

Reaktionen auf Angriffe werden mit der Response Language beschrieben. Dies können Meldungen oder Gegenmaßnahmen sein. Auch hier gibt es kein einheitliches Format für eine solche Sprache. Daher werden die Reaktionen auf einen Angriff oft in den Code des Programms eingebettet und bieten keine Möglichkeit zum Austausch oder zum editieren an (vgl. Vigna u. a., 2000b).

Reporting Language

Die Reporting Language wird dafür genutzt, einer ausgewählten Entität, Meldung über einen erkannten Angriff zu machen. Sie enthält alle Informationen zum Angriff, wie dessen Art, das Ziel und die Quelle. Auch hier kann die IDMEF verwendet werden.

Sie besteht im wesentlichen aus zwei Klassen: Die Alert-Klasse beinhaltet Informationen zu einer gefundenen Angriffsinstanz. Sie kann durch beliebige Unterklassen weiter spezifiziert werden. Um z.B. einen Buffer-Overflow-Angriff zu melden, kann eine Overflow-Alert-Klasse definiert werden, welche die Alert-Klasse um Felder ergänzt wie: Programm, Größe und Buffer (vgl. Feinstein u. a., 2007).

Die Heartbeat-Klasse hat die gleiche Funktion wie ein Watchdog. Ein Analyzer kann damit einem anderen Analyzer mitteilen, dass er noch funktionstüchtig ist. Hierzu sendet er periodisch Heartbeat-Nachrichten. Fehlende Nachrichten deuten auf Inaktivität des Analyzers oder Verbindungsprobleme hin. Als Austauschformat hat man Extensible Markup Language (XML) gewählt, da es ein freies, weit verbreitetes und von Menschen lesbares Format ist.

Correlation Language

Mit der Correlation Language versucht man Angriffe und Ereignisse miteinander in Relation zu stellen. Dazu nutzt man nicht nur die in den Logdateien enthaltenen Informationen, sondern kann auch die durch die Response Language erzeugten Meldungen einbeziehen. So lassen sich auf verschiedenen Abstraktionsebenen Relationen bilden.

Exploit Language

Die Exploit Language beschreibt jene Schritte, die für ein Eindringen in das System durchlaufen werden (vgl. Vigna u. a., 2000b). Sie kann für die Generierung von Signaturen herangezogen werden.

Detection Language

Die letzte Sprache ist die Detection Language. Sie ist dafür zuständig, die Daten der Event Language auszuwerten und Angriffe zu lokalisieren. In Bild 2.6 sind noch einmal die Zusammenhänge der einzelnen Sprachen dargestellt. Damit die bei der Überwachung erhobenen Daten ausgewertet werden können, bedarf es einer allgemein gültigen Datenstruktur. Damit lassen sich auch Daten aus unterschiedlichen Quellen vergleichen und Informationen können besser extrahiert werden.

2.2.2. Auditing

Ein IDS analysiert das Verhalten eines Systems. Um dies zu ermöglichen, müssen dem IDS Informationen zu Ereignissen und Aktionen, welche aus den Ereignissen resultieren, bereitgestellt werden. Dies wird mit Hilfe der Event Language realisiert. Das System muss Ereignisse und Aktionen speichern können, um diese für das IDS zugänglich zu machen.

Als Audit wird eine Datenstruktur bezeichnet, welche zur Protokollierung von Systemmeldungen eingesetzt wird. Dazu zählen z.B. Dateizugriffe, Systemaufrufe oder Sicherheitsereignisse. Das Auditing ist das sammeln dieser Daten durch ein Programm. Linuxsysteme bieten für das Überwachen und Filtern von Nachrichten der Ethernet-Schnittstelle das Programm

ebtables an. Mit den Programmen können Regeln bestimmt werden, wie mit einem Datenpaket verfahren werden soll und es können Einträge im Systemlog generiert werden.

Für das Auditing ist wichtig, dass gefiltert werden kann, welche Daten gespeichert werden sollen. Somit kann man sich auf wichtige Teilsysteme konzentrieren und Ressourcen einsparen. Trotzdem können die Logdateien sehr groß werden. Man unterscheidet das Auditing grob in zwei Kategorien: zustandsbasiertes und aktionsbasiertes Auditing.

Zustandsbasiertes Auditing

Bei dem zustandsbasierten Auditing werden periodisch Daten über den Gesamtzustand oder über Teilzustände eines Systems erhoben. Bei großen Systemen führt das unter Umständen zu sehr großen Datenmengen, weshalb man sich oft auf ausgesuchte Teilzustände konzentriert. Das System kann sich entweder in einem sicheren oder in einem unsicheren Zustand befinden, was Unterschiede bei der Erhebung der Daten macht. So können in einem unsicheren Zustand mehr Daten erhoben werden als in einem sicheren. Diese Vorgehensweise eignet sich z.B. für die Überwachung von Systemressourcen oder der Prüfung, welche Prozesse gerade ausgeführt werden.

Aktionsbasiertes Auditing

Aktionsbasiertes Auditing speichert bei jeder Aktion Informationen ab, wie: wer, wann, welche Aktion, erfolgreich oder erfolglos (vgl. Meier, 2008, S. 11). In den Tabellen 2.2 und 2.3 sind aktionsbasierte Audits dargestellt. Wichtig ist vor allem, den Kontext des Systems zu kennen, wenn ein aktionsbasierter Audit aufgezeichnet wird. Befindet sich das System schon in einem unsicheren Zustand, muss eine Aktion ggf. anders bewertet werden. Daher bietet es sich an, beide Verfahren zu kombinieren. Zum Beispiel wird bei Microsoft Windows der Systemzustand im EventType System festgehalten. Üblicherweise werden die Audits in ihrer zeitlichen Abfolge gespeichert, was als Audit-Trail bezeichnet wird.

In Bezug auf eingebettete Systeme muss abgewägt werden, welche Daten wie lange zur Verfügung stehen müssen, so dass die Sicherheitsanforderungen noch erfüllt werden. Da in einem Bordnetz unterschiedlichste Systeme, vom einfachen Sensor bis hin zum vollwertigen OS, eingesetzt werden, gibt es auch unterschiedlich aufgebaute Logdateien und Datenbanken oder es existieren diese erst gar nicht. Damit die extrahierten Informationen vergleichbar werden, sollte eine einheitliche Datenstruktur und ein Austauschformat festgelegt werden. Die wichtigste Voraussetzung für ein zuverlässiges IDS ist vor allem die Qualität der Audits. Es muss trotz einer eventuellen Limitierung der Ressourcen möglich sein, aus einem Audit oder einer Audit-Trail einen Angriff zu erkennen.

2.2.3. Intrusion-Detection

„Ziel des Einsatzes von Intrusion-Detection-Systemen ist eine möglichst frühzeitige Erkennung von Angriffen, um den Schaden zu minimieren und Angreifer identifizieren zu können“ (Meier, 2008, S. 9). Das IDS ist dabei die Soft- oder Hardware, welche den Erkennungsprozess automatisiert (vgl. Bace und Mell, 2001). Dazu verwendet es eine Detection Language. In modernen Fahrzeugen, wie z.B. dem Tesla, werden solche Überwachungsmaßnahmen bereits eingesetzt. Als Diagnoseschnittstelle wird im Tesla ein Standard-Ethernet-Port verwendet, der zu einer Telemetrieeinheit gehört, welche auf Ubuntu basiert (vgl. dragTimes, 2014).

Daher können die vom Betriebssystem mitgelieferten Auditing-Funktionen genutzt werden, um die Diagnoseschnittstelle zu überwachen. Die durch ein IDS erhobenen Daten können entweder sofort oder später ausgewertet werden. Auch wenn zur Laufzeit keine bekannten Angriffe festgestellt werden, können durch eine forensische Datenanalyse neue Angriffe oder Sicherheitslücken entdeckt werden. Bei der Auswertung der Daten kann das Ergebnis des IDS in vier verschiedene Kategorien eingeteilt werden:

- **True-Positive**: Richtig erkanntes Normalverhalten (kein Angriff)
- **True-Negative**: Richtig erkannte Gefahr (Angriff)
- **False-Positive**: Falsch erkanntes Normalverhalten (Angriff)
- **False-Negative**: Falsch erkannte Gefahr (kein Angriff)

Die Güte eines IDS wird anhand der Falscherkennungsrate gemessen (False-Positive und False-Negative)(vgl. Liao u. a., 2013). Ein falsch erkanntes Normalverhalten täuscht nicht vorhandene Sicherheit vor. Daher sollte dieser Wert so gering wie möglich sein. Falsch erkannte Gefahren sind erst einmal nichts Schlimmes. Sie erhöhen aber die Rate der nachträglichen Auswertung, damit das IDS sensibilisiert werden kann. Je nachdem welche Konsequenzen ein gefundener Angriff hat, können falsch erkannte Angriffe auch die Funktion des Systems beeinträchtigen. Daher sollte auch dieser Wert so gering wie nur möglich sein.

IDS werden in zwei Kategorien unterschieden. Zum einen in die Anomaly Detection und zum anderen in die Misuse Detection. Bei der AD wird jede Abweichung vom Normalverhalten als Anomalie angesehen. Die MD dagegen nutzt die Kenntnis über Angriffe dazu, diese zu erkennen. Beide Herangehensweisen gehen also den jeweils umgekehrten Weg, um Angriffe zu erkennen. In Abschnitt 2.2.3 und 2.2.3 werden die Herangehensweisen genauer vorgestellt. In neueren Forschungsberichten wird eine alternative Kategorisierung anhand der eingesetzten Algorithmen vorgeschlagen, anstatt auf Basis der Herangehensweise. So teilen Liao u. a. (2013)

die ID in fünf Kategorien auf: statistische, musterbasierte, regelbasierte, zustandsbasierte und heuristische IDS. Alle fünf Kategorien finden in der AD sowie in der MD Anwendung. In dieser Arbeit wird der konservative Ansatz über eine MD betrachtet.

Integration

Die Integration eines IDSs kann hostbasiert, netzwerkbasiert oder hybrid sein. Manche unterscheiden bei einem netzwerkbasierten IDS zusätzlich in kabellos und kabelgebunden (vgl. Stavroulakis und Stamp, 2010), da kabellose netzwerkbasierte IDS, z.B. in Ad-Hoc-Netzen, zusätzliche Anforderungen mit sich bringen. Denn Funkverbindungen können z.B. leichter gestört werden.

In hostbasierten IDS werden Informationen direkt auf dem Host gesammelt. Nur so ist es möglich auch eine sichere Ende-zu-Ende-Kommunikation zu untersuchen. Es kann aber zu Konflikten mit bereits bestehenden Sicherheitsfunktionen kommen, wenn auf sicherheitskritische Informationen zugegriffen werden muss. Dazu kommt, dass die Ressourcen des Hosts für die Verarbeitung genutzt werden müssen. Netzwerkbasierte IDS greifen, mit Hilfe von Sensoren, Daten innerhalb des Netzwerkes ab, ähnlich wie bei einem Man-In-The-Middle (MITM)-Angriff. Diese werden typischerweise an strategisch wichtigen Punkten wie Gateways oder Switchen platziert. Die Sensoren sind leichter zu verwalten als der Host, da sie normalerweise ein einheitliches System bieten. Allerdings können verschlüsselte (Ende-zu-Ende) Nachrichten nicht analysiert werden.

Anomaly Detection

Die AD verfolgt den Ansatz, dass jegliche Abweichung vom Normalverhalten einen Angriff darstellt. Dazu muss das Normalverhalten eines Systems vorher in einem Profil spezifiziert werden. Entweder das Profil wird vorgegeben oder von dem IDS, mit Hilfe von maschinellem Lernen, erlernt. Bei großen und komplexen Systemen stellt dies eine Herausforderung dar, denn wann ist ein System komplett erfasst, bzw. wann ist der Lernprozess abgeschlossen? Das führt zu einer verschwommenen Linie, zwischen normalem und anormalem Verhalten, was wiederum dazu führt, dass manche Angriffe nicht erkannt werden (False-Negative-Rate), da sie sich zu nah am ermittelten Normalverhalten befinden. Ist eine hohe Sicherheit gefordert, wird dieser Schwellwert verkleinert. Damit steigt zwar auch die False-Positive-Rate, dafür sinkt aber die False-Negative-Rate ab.

Der große Vorteil der AD ist, dass auch unbekannte Angriffe und Sicherheitslücken entdeckt werden können. Hat man ein dynamisches System, muss das Profil der AD stetig angepasst werden, was eine zusätzliche Fehlerquelle bei der Spezifizierung des Normalverhaltens darstellt.

Zudem ist die AD während dieses Vorgangs nicht verfügbar (vgl. Corchado und Herrero, 2011; Wu und Banzhaf, 2010). Für die AD werden, neben den von Liao u. a. (2013) vorgestellten Kategorien, z.B. auch Algorithmen und Techniken aus den Bereichen der neuronalen Netze oder der Immunsystem-Modelle verwendet.

Die AD benötigt für eine korrekte Funktionsweise ein genaues Verhaltensprofil. Das Bordnetz wird hauptsächlich durch computergesteuerte Elektronik genutzt. Der kritische Datenverkehr sowie alle nötigen Funktionen zum Bedienen des Autos sind bekannt. Auch Funktionen die sehr selten genutzt werden, wie das Auslösen des Airbags, werden berücksichtig. Daher kann hier leicht ein Profil über das Normalverhalten erstellt, bzw. darauf zurückgegriffen werden. Es gibt aber auch einen Teil, der in der statischen Konfiguration nur teilweise berücksichtigt werden kann, da dieser durch die Insassen bestimmt wird: Zum Beispiel der Anschluss von externen Geräten, die Nutzung von Internetdiensten oder das Verhaltensprofil des Fahrers.

Selbst wenn ein Verhaltensprofil des Fahrers erstellt werden würde, kann sich dieses über die Zeit ändern. Bei CarSharing-Plattformen wäre dies eine große Herausforderung. Zudem wird eine C2C- bzw. C2E-Kommunikation angestrebt, bei der Informationen externer Quellen zur Zustandsanalyse verwendet werden. Daher müssen gerade die Stellen überwacht werden, an denen kritische sowie unkritische Daten und Funktionen aufeinander Einfluss haben.

Misuse Detection

Im Gegensatz zur AD, bei der das Normalverhalten als Referenz zur Erkennung genutzt wird, wird bei der MD das anormale Verhalten zur Erkennung von Angriffen genutzt. Das setzt voraus, dass die Angriffe bereits bekannt und vollständig und eindeutig spezifiziert sind. Theoretisch geht die False-Positive-Rate damit gegen Null. In der Realität wird das aber noch nicht erreicht, da Angriffe entweder nicht richtig oder auf Verdacht spezifiziert werden (vgl. Debar und Morin, 2002; Cheng u. a., 2012). Aktuelle Anti-Virus-Programme basieren hauptsächlich auf MD-Systemen (vgl. Sanok, 2005). Sie durchsuchen Dateien nach bekannten Angriffsmustern, was bei einem hohen Datenaufkommen auch sehr viele Ressourcen beansprucht.

Die Spezifizierung eines Angriffs ist nicht trivial. Wird z.B. eine Variante eines bekannten Angriffs verwendet, kann es sein, dass das MD-System diesen nicht erkennt, da die Signatur zu spezifisch ist. Zustandsbasierte Erkennungsalgorithmen können dabei Abhilfe schaffen (vgl. Bace und Mell, 2001), worauf später näher eingegangen wird. Wie die AD muss auch die MD stetig aktualisiert werden und zwar dann, wenn neue Angriffe und Sicherheitslücken entdeckt werden. Denn im Gegensatz zur AD, werden neue Angriffe nicht automatisch erkannt.

Die Datenbank mit bekannten Angriffen wächst so stetig weiter an, was dazu führt, dass die MD gegen immer mehr Angriffe prüfen muss. Entweder die MD muss dahingehend optimiert

werden, auch mit großen Datenbeständen effektiv umgehen zu können oder es muss zwischen Aufwand und Ertrag abgewogen werden, was eine Ungenauigkeit in der Erkennung zur Folge hat.

Da die AD und die MD beide unterschiedliche Vor- und Nachteile haben, bietet es sich an, beide gemeinsam einzusetzen. Das kann zu einer besseren False-Negative-Rate führen, da die AD nur eine Teilmenge der bekannten Angriffe abdeckt, welche im Idealfall von der MD komplettiert wird. Eine Verbesserung der False-Positive-Rate ist hingegen nicht zu erwarten, denn diese wird nur durch die AD bestimmt (vgl. Meier, 2008, S. 18).

Trotzdem eignet sich die MD dafür, die bekannten Schwachstellen des TTEs zu überwachen. Zum einen sind diese in verschiedenen Arbeiten beschrieben worden (vgl. Steiner, 2013; Isakovic, 2011a), zum anderen ist die MD genauer bei der Auswertung der Audits, da bekannt ist, welche Aktionen und Ereignisse beobachtet werden müssen. Überwacht werden nur die Schwachstellen des Protokolls, nicht aber eventuelle Fehler der Implementation.

Als erstes müssen die Angriffsmöglichkeiten (Aktionen und Ereignisse) ermittelt werden. Jeder Angriff folgt einem eigenen Muster in seiner Vorgehensweise. Aus den Mustern werden anschließend Signaturen generiert. Die Angriffe auf das TTE werden in Kapitel 3 näher beschrieben.

Signaturen

Um eine Signatur für einen Angriff erstellen zu können, muss zu erst festgelegt werden, wie die Ereignisse aus dem Audit semantisch aufgearbeitet und einem Angriff zugeordnet werden können. Im einfachsten Fall ist ein Ereignis gleich ein Angriff (Einzelschritt-Attacken). Das ist aber nicht immer der Fall. Daher müssen auch lose Ereignisse in eine Reihenfolge gebracht und einem Angriff zugeschrieben werden können (Mehrschritt-Attacken). Bei gleichzeitig eingetretenen Ereignissen, muss es möglich sein, diese in unterschiedliche Sequenzen einordnen zu können.

Manche Angriffe verlangen auch das Zählen von Versuchen oder Schritten, z.B. wenn ein Angreifer einen Brute-Force-Angriff[2] durchführt. Da Angriffe auch gleichzeitig von verschiedenen Angreifern durchgeführt werden können (Distributed Denial of Service (DDoS)) muss es möglich sein, dass Instanzen einer Signatur (Signaturinstanz) betrachtet werden können, um die Angriffe unterscheidbar zu machen.

Eine Signaturinstanz ist vollständig, wenn alle ihre Ereignisse eingetreten sind oder wird abgebrochen, wenn sie einen Zustand erreicht, der nicht mehr als Angriff gewertet wird. Vorher gilt sie als partielle Signaturinstanz. Es kann vorkommen, dass Signaturinstanzen nie

[2] Lösungsfindung für ein Problem (z.B. ein Passwort für einen Login) durch Probieren.

vervollständigt oder abgebrochen werden. Ereignisse, welche zu einer Signaturinstanz zählen, werden Instanzschritte genannt. Man unterscheidet zwischen initialen, inneren, finalen und abbrechenden Instanzschritten. Ein Schritt kann zudem konsumierend oder nicht konsumierend sein. Somit kann ein Ereignis einem oder mehreren Angriffen zugeordnet werden. All diese Eigenschaften müssen durch eine Signaturbeschreibungssprache abgedeckt werden.

Nachfolgend werden drei Kategorien vorgestellt, mit denen sich Signaturen beschreiben lassen: automatenbasierte, graphenbasierte oder netzbasierte Signaturen.

Automatenbasierte Signaturen

Die State Transition Analysis Technique Language (STATL) ist eine Signaturbeschreibungssprache auf Basis von endlichen deterministischen Automaten Vigna u. a. (2000a). Neben den bekannten Komponenten, wie Zuständen und Transitionen, kann sie auch Konsumeigenschaften über die Transitionen abbilden. Ein Zustand repräsentiert immer den Systemzustand, wobei nur sicherheitskritische Zustände des Systems abgebildet werden. Die Ereignisse lösen die Zustandswechsel aus.

Zustände und Transitionen können durch ihre Eintrittsbedingungen kontextabhängig betrachtet werden. Zudem können Variablen verwendet werden, was z.B. die Verwendung von Schleifen erlaubt, mit denen Häufigkeiten abgebildet werden können. Damit können z.B. die Anfragen von einem Client an einen Server gezählt werden, um z.B. einen Denial of Service (DoS)-Angriff zu erkennen. Die Konjunktion von Zuständen wird nicht unterstützt. Das bedeutet, dass eine Signatur die mehrere Initialschritte besitzt, nicht abgebildet werden kann oder das Sequenzen von gleichzeitig aufgetretenen Ereignissen, nicht abgebildet werden können. Da auch immer das gesamte System betrachtet wird, ist es nicht möglich, einzelne Signaturinstanzen abzubilden. Andere Signaturbeschreibungssprachen auf Basis von Automaten bieten entweder die gleichen oder weniger Eigenschaften.

Graphenbasierte Signaturen

Ning u. a. (2001) hat mit den Misuse Signatures (MuSigs) einen graphenbasierten Ansatz zur Beschreibung von Signaturen vorgestellt. Jeder Knoten in einem Graphen ist ein dedizierter Systemzustand, welcher mit Hilfe von Variablen kontextabhängig betrachtet werden kann. Das bedeutet aber auch, dass keine Signaturinstanzen betrachtet werden können, da immer das gesamte System betrachtet wird. Im Gegensatz zu den automatenbasierten Signaturen können graphenbasierte Signaturen mehrere Elternknoten besitzen, so dass eine Konjunktion von Ereignisketten möglich ist. Ein Abbruch kann durch einen dazu erstellten Zustand abgebildet werden.

Die Kanten der Knoten können mit zeitlichen Eigenschaften belegt werden. So können sie aussagen, dass Ereignisse gleichzeitig oder in einer bestimmten Reihenfolge eintreten müssen. Bei den automatenbasierten Signaturen hätte man dafür alle möglichen Sequenzen modellieren müssen, wobei gleichzeitige Ereignisse durch die fehlende Konjunktion gar nicht möglich sind. Nicht spezifiziert sind Schleifen und Konsumeigenschaften.

Netzbasierte Signaturen

Der größte Unterschied zu den ersten beiden Ansätzen ist, dass netzbasierte Signaturen, welche auf Petri-Netzen basieren, Token verwenden, um Zustände von Angriffen abzubilden. Verwendet man z.B. Coloured Petri Nets (CPNs) lassen sich auch Variablen an die Token binden, womit sich Signaturinstanzen unterscheiden lassen. Eine Realisierung für netzbasierte Signaturen ist der Coloured Petri Net Automaton (CPA), welcher von Kumar und Spafford (1994) vorgestellt wurde.

Konjunktionen von Ereignissen sind bei CPAs durch die Eigenschaften von Petri-Netzen gegeben. Abbruchplätze sind nicht explizit vorgesehen. Sie werden indirekt durch ein invariantes, nebenläufiges Netz abgebildet. Erreicht ein Token dort einen Finalplatz, wird sein korrespondierenden Token aus dem Signaturnetz gelöscht. Erreicht ein Token aus dem Signaturnetz einen Finalplatz, wird der Token im Invariantennetz zurückgesetzt. Die Variablenbindung ist immer final, was den Einsatz von Schleifen unmöglich macht.

Bein einem CPA können den Plätzen Eigenschaften für das Schalten von Token, ähnlich den geforderten Konsumeigenschaften, zugeordnet werden. Bei einem mit dem Schlüsselwort dup versehenen Platz, bleibt der Token solange auf diesem Platz, bis er durch das Invariantennetz zurückgesetzt wurde. Währenddessen werden Kopien des Tokens geschalten. Dies hat Auswirkungen auf alle ausgehenden Transitionen.

Meier (2008) hat den Ansatz netzbasierter Signaturen aufgegriffen und um die Anforderungen an die Beschreibung aktueller Signaturen erweitert. In seinen Signaturnetzen hat er Abbruchplätze definiert und Variablen eingeführt, welche mehrfach beschrieben werden können. Konsumeigenschaften sind an Transitionen gebunden. Somit kann für jede Sequenz entschieden werden, ob ein Token konsumiert wird oder nicht. Ein Invariantennetz, wie bei CPAs, ist nicht mehr nötig. Die Plätze wurden erweitert, so dass Bedingungen für das Schalten von Transitionen definiert werden können.

Fazit

Da automatenbasierte- und graphenbasierte Signaturen, neben fehlender Konjunktion und beschränkter Sequentialisierung, nicht in der Lage sind, Signaturinstanzen abzubilden, kommen

sie für die Beschreibung von Signaturen, nach aktuellem Anforderungsprofil, nicht in Betracht. Netzbasierte Signaturen hingegen sind in der Lage die Anforderungen zu erfüllen. Daher wird nachfolgend die Arbeit von Meier (2008) genau vorgestellt. Seine Signaturnetze werden anschließend genutzt, um die Signaturen für die Erkennung der Ausnutzung der Schwachstellen des TTEs zu beschreiben.

2.2.4. Signaturnetze

Die Arbeit von Meier (2008) basiert auf dem Modell von aktiven Datenbanken. Datenbanken reagieren normalerweise nur dann mit einer eigenen Aktion auf ein Ereignis, wenn dies von außen angestoßen wird. Aktive Datenbanken können bei Ereignissen selbstständig reagieren, sobald alle Vorbedingungen erfüllt sind (vgl. McCarthy und Dayal, 1989). Das Schema enthält die folgenden drei Parameter:

e: Ereignis (Event)

c: Bedingung (Condition)

a: Reaktion (Action)

Alle drei Parameter werden auch einfach unter dem Begriff Ereignis zusammengefasst. In aktiven Datenbanken können mehrere Ereignisse ein komplexes Ereignis bilden, was für die Bildung von Signaturen, welche aus mehreren Ereignissen bestehen, interessant ist. Um dies Abbilden zu können wird, für jeden der drei Parameter, eine Menge definiert: Ereignisse (e_1, e_2, …, e_n), Bedingungen (c_1, c_2, …, c_n) und Reaktionen (a_1, a_2, …, a_n) (n $\in \mathbb{N}$). Allerdings tritt das komplexe Ereignis erst dann ein, wenn alle Teilereignisse eingetreten sind, alle Bedingungen erfüllt und alle Reaktionen durchgeführt wurden (späte Instanziierung).

Für Signaturen bedeutet das, dass alle möglichen Ereignisstränge eines Angriffs in einer eigenen Signatur abgebildet werden müssen, obwohl sie denselben Angriff darstellen, z.B. dann, wenn Teilereignisse mehrfach ausgeführt werden. Meier (2008) hat dies an die Anforderungen für eine Signatur angepasst. Bei ihm wird schon bei dem Eintreten des ersten Teilereignisses das komplexe Ereignis erzeugt (frühe Instanziierung). Dafür müssen die Mengen anders gebildet werden: Teilereignis 1 (e_1, c_1, a_1), Teilereignis 2 (e_2, c_2, a_2), …, Teilereignis n (e_n, c_n, a_n) (n $\in \mathbb{N}$). Das ermöglicht zudem bereits auf Teilereignisse reagieren zu können.

Ereignisordnung

Neben der Erkennung der Ereignisse ist es wichtig, dass im Audit auch der Eintrittszeitpunkt eines Ereignisses festgehalten wird, da Signaturen von Angriffen anhand der Abfolge der

erzeugten Ereignisse definiert werden. Wenn eine Signatur z.B. drei mal ein Ereignis a benötigt, um vervollständigt zu werden, aber zwischendurch von einem Ereignis b abgebrochen werden kann, ist es wichtig zu wissen, in welcher Reihenfolge die Ereignisse aufgetreten sind. Dazu ist die Ordnungsrelation $<_{time}$ definiert als:

$$\{(e, e') | e, e' \in E \text{ und der Zeitstempel von } e \text{ ist kleiner als der von } e'\} \tag{2.3}$$

Wobei E die Menge aller Ereignisse ist. Somit können die Ereignisse in der Reihenfolge ihres Auftretens verarbeitet werden. Wenn Ereignisse gleichzeitig auftreten, ist keine totale Ordnung auf Basis der Zeit mehr möglich und es ergeben sich mehrere Sequenzen von Ereignissen, welche separat betrachtet werden. Diese werden durch $<_{trail}$ in eine Abfolge gebracht.

Komponenten

Das von Meier (2008) vorgestellte Signaturnetz enthält, genau wie bei Petri-Netzen, Plätze, Transitionen, Kanten und Token. Es hat aber auch einige Unterschiede, welche nachfolgend beschrieben werden. Ein Signaturnetz schaltet immer dann, wenn ein externes Ereignis eintritt. Dabei werden vor dem Schalten die Schaltbedingungen geprüft. Anschließend werden die Aktionen ausgeführt. Dies gleicht dem Prinzip der aktiven Datenbanken. Das Signaturnetz beschreibt ein komplexes Ereignis, welches aus Teilereignissen besteht. Tritt dass erste Teilereignis ein, wird eine Instanz des komplexen Ereignis angelegt, in dem ein repräsentativer Token erzeugt wird. Erreicht der Token einen Endplatz, ist das komplexe Ereignis eingetreten. Nachfolgend werden die Bestandteile und die Funktionsweise eines Signaturnetzes vorgestellt und beschrieben.

Platz

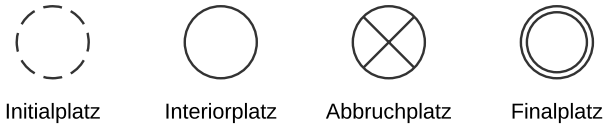

Abbildung 2.7.: Platzarten im Signaturnetz

Plätze stellen, zusammen mit den Token, den aktuellen Zustand des komplexen Ereignis dar. Es werden vier Arten von Plätzen unterschieden, welche in Abbildung 2.7 zu sehen sind. Die

Initialplätze sind die Eintrittspunkte in das Signaturnetz. Sie beinhalten immer mindestens einen Token. Jedes Signaturnetz hat genau einen Finalplatz, welcher das Eintreten des komplexen Ereignis darstellt. Der Abbruchplatz dient dazu, eine Instanz eines komplexen Ereignis zu beenden, bevor dieses eingetreten ist. Alle weiteren Plätze werden Interiorplätze gennant.

Transition

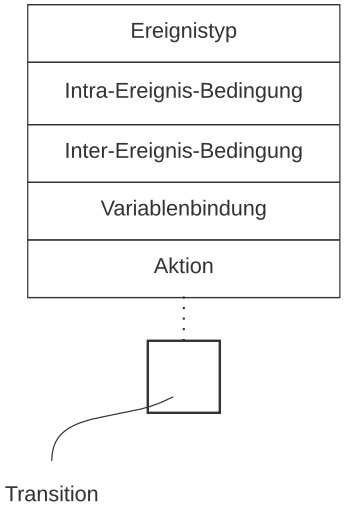

Abbildung 2.8.: Transition mit Bedingungen und Aktionen

Mit Transitionen schalten Signaturnetze von einen Zustand in einen anderen. Ihr Aufbau ist in Abbildung 2.8 dargestellt und entspricht der Definition von Ereignissen = (e, c, a). Der Ereignistyp repräsentiert das Ereignis, welches die Transition schalten lässt. Dabei wird zwischen spontanen und nicht-spontanen Transitionen unterschieden. Spontane Transitionen schalten sobald die Schaltbedingungen erfüllt sind. Sie werden mit dem Ereignistyp ε gekennzeichnet. Es muss also keine Aktion eintreten um das Schalten zu triggern. Das Prinzip stammt aus den aktiven Datenbanken. Im Gegensatz dazu muss bei nicht-spontanen Transitionen eine Aktion eintreten.

Es wird bei den Transitionen zwischen zwei Bedingungsarten unterschieden um den Kontext des Ereignisses abzubilden. Intra-Ereignis-Bedingungen gleichen die Werte eines Teilereignis mit Konstanten ab. Zum Beispiel kann somit ein Zähler realisiert werden. Die Variable eines

Token wird mit jedem Schleifendurchlauf erhöht und gegen eine Konstante geprüft. Inter-Ereignis-Bedingungen gleichen die Werte von verschiedenen Teilereignissen untereinander ab. Zum Beispiel können so Werte, welche in einem Token gespeichert wurden, mit aktuellen Werten eines aufgetretenen Ereignisses verglichen werden.

Bei dem Schalten einer Transition können Informationen über die Teilzustände erhoben und in den Variablen der Token gespeichert werden (Variablenbindung). Zum Abschluss kann eine Aktion festgelegt werden, welche bei dem Schalten der Transistion ausgeführt wird.

Kante

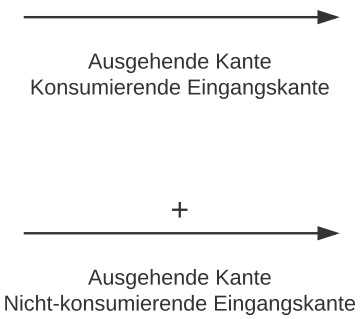

Abbildung 2.9.: Kanten im Signaturnetz

Kanten verbinden Plätze mit Transition und vice versa. Sie haben entweder die Eigenschaft konsumierend oder nicht-konsumierend (siehe Abbildung 2.9). Konsumierende Kanten schalten einen Token von einem Platz über die Transition auf den Nachfolgeplatz. Bei nicht-konsumierenden Kanten wird eine Kopie des Tokens von dem Ausgangsplatz geschalten. Hat ein Platz zwei schaltbereite Transitionen und hat eine der Transitionen eine konsumierende Eingangskante, wird der Token geschalten und nicht seine Kopie.

Token

Ein Token beschreibt immer eine Instanz eines Angriffs. Er enthält Informationen, gespeichert in Variablen, über den Teilzustand seiner Instanz und ermöglicht die Darstellung dynamischer Abläufe. Alle Token eines Signaturnetzes bilden zusammen den Systemzustand. Ein Platz kann $n \in \mathbb{N}$ Token enthalten. Befinden sich identische Token auf einem Platz, werden diese zusammengelegt. Sind Token mit unterschiedlicher Variablenbindung(Beispielsweise

Token_1:$\{a = 0, b = 1\}$ und Token_2:$\{c = 0\}$) auf einem Platz, können diese nach dem Schalten zusammengelegt (Token_{12}:$\{a = 0, b = 1, c = 0\}$) werden.

Das kann z.B. bei Angriffen interessant sein, welche unterschiedliche Sequenzen von Ereignissen und Aktionen nutzen. Beide Token gehören dann zu dem selben Angriff. Nicht belegte Tokenvariablen werden nicht dargestellt. Dies kann unter anderem auch ein Indiz dafür sein, dass unnötige Elemente vorhanden sind. Einen besonderen Token stellt der Token auf dem Initialplatz dar. Dieser kann nicht verbraucht werden und wird auch nicht im Netz angezeigt.

Schaltregel

Im Gegensatz zu Petri-Netzen besitzen Signaturnetze eine deterministische Schaltregel. Alle aktivierten Transitionen müssen schalten. Haben alle Vorplätze einer Transition genau ein Token welches die Bedingungen der Transition erfüllt, dann gilt sie als aktiviert. Bei konkurrierenden Transitionen werden Token ggf. dupliziert. Wie bereits erwähnt, können Transitionen auch ohne Ereignis schalten. Ist das der Fall, gilt der Zustand des Netzes als instabil. Alle spontanen Transitionen schalten dann solange, bis das Netz stabil ist, es also keine schaltbereite spontane Transition mehr gibt.

Reaktion

Die Erkennung von Angriffen kann online oder offline geschehen. Bei der Online-Variante werden die Logdateien des Systems im laufenden Betrieb analysiert. Eine Offline-Analyse wird dann durchgeführt, wenn ein Angriff festgestellt wurde und nachträglich weitere Informationen gesammelt werden sollen oder wenn die Analyse zur Laufzeit nicht möglich ist. Ist kein Angriff festgestellt worden, kann eine Offline-Auswertung trotzdem nützlich sein, um z.B. neue Angriffe zu finden. Hat das IDS einen Angriff lokalisiert, kann es entweder aktiv oder passiv darauf reagieren. Die Maßnahmen werden mit Hilfe der Response Language definiert.

Aktive Maßnahmen sind z.B. das weitere Beobachten des Angriffs um mehr Informationen über den Angreifer und das Ziel zu bekommen. Eine andere Maßnahme ist das Aussperren des Angreifers, z.B. über seine IP-Adresse. Das System kann auch in einen sicheren Zustand überführt werden, so dass der Angreifer keinen weiteren Schaden anrichten kann. Das kann man mit weiteren Maßnahmen, wie dem Honeypot kombinieren. Dazu wird dem Angreifer vorgegaukelt, dass er immer noch Zugang zu dem System oder Netzwerk hat. Wobei er sich eigentlich in einer Sandbox befindet, in der er keinen Schaden anrichten kann. Passive Maßnahmen beinhalten ausschließlich die Benachrichtigung darüber, dass ein Angriff erkannt wurde.

2.2.5. Open Source-Projekte

Zur Zeit gibt es einige aktive Projekte, die sich mit ID beschäftigen. Ein netzwerkbasiertes Open Source IDS ist z.B. SNORT (vgl. Roesch, 1999). Es enthält einen Paketsniffer sowie einen Logger. Mit Hilfe von Pattern können Anomalien beschrieben werden auf die die Pakete überprüft werden. In der aktuellen Umsetzung kann es allerdings nur zur Erkennung von Einzelschritt-Attacken genutzt werden. SNORT lässt sich aber als Basis für eine Umsetzung zur Erkennung von Mehrschritt-Attacken nutzen. Es kann sowohl auf Unix- als auch auf Windows-Hosts ausgeführt werden und kann die jeweilige Struktur für Systemmeldungen nutzen. Mit Gnort Vasiliadis u. a. (2008) wurde eine Optimierung von Snort vorgestellt, bei der anstatt einer CPU eine GPU zum Pattern Matching benutzt wurde. Damit waren sie in der Lage 2.3 Gbit/s an Daten zu verarbeiten.

Ein hostbasiertes Open Source Projekt ist ClamAV, welches als Antivirus-Programm eingesetzt wird. Es enthält eine ständig aktualisierte Virus-Datenbank, sowie einen Virusscanner. Zudem lassen sich über die integrierten Schnittstellen eigene Virusscanner implementieren. Auch ClamAV ist für Unix- und Windowssysteme verfügbar.

2.2.6. Zusammenfassung

Die Misuse Detection ist bei der Erkennung von bekannten Angriffen besser geeignet als die Anomaly Detection. Da die Schwachstellen des TTE-Protokolls in verschiedenen Arbeiten beschrieben wurden, bietet es sich an, diese mit Hilfe einer MD zu überwachen. Damit dies überhaupt möglich ist, müssen Ereignisse ausgewertet werden, welche mit Hilfe des Auditings aufgezeichnet und aufgearbeitet wurden. Da jedes System ein eigenes Vorgehen dafür hat, muss eine allgemein gültige Angriffssprache spezifiziert werden.

Auf Basis der aufbereiteten Ereignisse der Audits müssen als nächstes Signaturen der Angriffe erstellt werden. Es gibt viele Darstellungsformen, aber nur die vorgeschlagenen Signaturnetze von Meier (2008) erfüllen alle gestellten Anforderung an die Abbildung von Signaturen. Vor allem die Möglichkeit Signaturinstanzen und Mehrschritt-Attacken abbilden zu können ist von großer Wichtigkeit. Daher werden diese zur Erstellung der Signaturen für die Angriffe auf die Schwachstellen im TTE genutzt.

Für die Integration in das Bordnetz kann auf die Wissensbasis verschiedener Open Source-Projekte zurückgegriffen werden.

3. Network Intrusion Detection im Time-Triggered-Ethernet

In Kapitel **??** wurden Sicherheitsfunktionen beschrieben, welche aufgrund der Struktur des TTE bereits gegeben sind. Darüberhinaus gibt es aber noch Schwachstellen, durch die ein Angreifer die Kommunikation zu seinen Gunsten beeinflussen kann. Diese wurden unter anderem von den Entwicklern des TTE, in der Arbeit von Steiner (2013), beschrieben. Dieses Kapitel stellt die durch Steiner (2013) ermittelten Schwachstellen vor und untersucht, ob und wie mit Hilfe eines IDS diese Schwachstellen überwacht werden können. Da die Schwachstellen bereits bekannt sind, wird eine Misuse Detection verwendet und die dazu nötigen Signaturen erstellt. Dies wird mit Hilfe der vorgestellten Signaturnetze durchgeführt.

3.1. Schwachstellen im Time-Triggered-Ethernet

Die nachfolgenden Abschnitte beschreiben die von Steiner (2013) vorgestellten Schwachstellen und bewerten ihr Risiko für das Netzwerk. Daher werden als erstes die möglichen Zugänge für einen Angreifer dargestellt und die Risikostufen abgegrenzt. Es mögliche Angriffsszenarien untersucht und ggf. bereits vorhandene Lösungswege vorgestellt. Als erstes wird die Synchronisation betrachtet. Sie ist der essentielle Bestandteil der TT-Kommunikation. Anschließend folgt der Time-Triggered- und der Event-Datenverkehr.

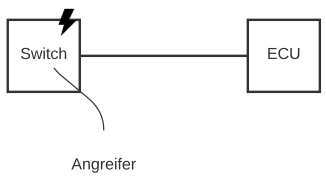

Abbildung 3.1.: Verzögerung einer Nachricht mit Hilfe eines kompromittierten Switches

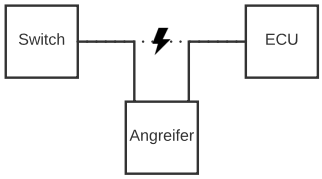

Abbildung 3.2.: Verzögerung einer Nachricht durch eine man-in-the-middle-Attacke

Zugang zum Netzwerk

Ein Angreifer kann auf das Netzwerk entweder über einen kompromittierte Teilnehmer zugreifen (siehe Abbildung 3.1) oder über die Verbindung zwischen zwei Teilnehmern, in dem er sich mit einer eigenen Komponente zwischen die Teilnehmer setzt (Man-In-The-Middle)(siehe Abbildung 3.2).

Risikoklassen

Tabelle 3.1.: Kategorisierung von Eintrittswahrscheinlichkeiten nach Bundesamt für Sicherheit in der Informationstechnik (2016) S. 22

selten	Ereignis könnte nach heutigem Kenntnisstand höchstens alle 5 Jahre eintreten.
mittel	Ereignis tritt einmal alle 5 Jahre bis einmal im Jahr ein.
häufig	Ereignis tritt einmal im Jahr bis einmal pro Monat ein.
sehr häufig	Ereignis tritt mehrmals im Monat ein.

Tabelle 3.2.: Kategorisierung von Schadensauswirkungen nach Bundesamt für Sicherheit in der Informationstechnik (2016) S. 22

vernachlässigbar	Die Schadensauswirkungen sind gering und können vernachlässigt werden.
begrenzt	Die Schadensauswirkungen sind begrenzt und überschaubar
beträchtlich	Die Schadensauswirkungen können beträchtlich sein.
existenzbedrohend	Die Schadensauswirkungen können ein existentiell bedrohliches, katastrophales Ausmaß erreichen.

Tabelle 3.3.: Definition der Risikokategorien nach Bundesamt für Sicherheit in der Informationstechnik (2016) S. 23

gering	Die bereits umgesetzten oder zumindest im Sicherheitskonzept vorgesehenen Sicherheitsmaßnahmen bieten einen ausreichenden Schutz. In der Praxis ist es üblich, geringe Risiken zu akzeptieren und die Gefährdung dennoch zu beobachten.
mittel	Die bereits umgesetzten oder zumindest im Sicherheitskonzept vorgesehenen Sicherheitsmaßnahmen reichen möglicherweise nicht aus.
hoch	Die bereits umgesetzten oder zumindest im Sicherheitskonzept vorgesehenen Sicherheitsmaßnahmen bieten keinen ausreichenden Schutz vor der jeweiligen Gefährdung.
sehr hoch	Die bereits umgesetzten oder zumindest im Sicherheitskonzept vorgesehenen Sicherheitsmaßnahmen bieten keinen ausreichenden Schutz vor der jeweiligen Gefährdung. In der Praxis werden sehr hohe Risiken selten akzeptiert.

Das Bundesamt für Sicherheit in der Informationstechnik (BSI) teilt die Risikokategorien in *gering*, *mittel*, *hoch* und *sehr hoch* ein. Die Beschreibung der einzelnen Kategorien ist in Tabelle 3.3 zu sehen. Das Risiko ergibt sich durch die Kombination der Eintrittswahrscheinlichkeit (siehe Tabelle 3.1) mit der Schadenshöhe (siehe Tabelle 3.2).

Viele bekannte Angriffe stammen von Wissenschaftlern, welche an Fahrzeugen unter Laborbedingungen durchgeführt worden sind. Es muss davon ausgegangen werden, dass wenn die Zahl der Fahrzeuge mit einem Internetzugang weiter steigt, die Angriffe auf das Bordnetz zunehmen werden und somit weitere Schwachstellen aufkommen. So, wie es zur Zeit bei dem Internet of Things (IoT)zu beobachten ist (vgl. Gan u. a., 2011; Böck, 2017).

Die Schadensauswirkungen reichen bei einem fahrenden Fahrzeug von begrenzt bis existenzbedrohlich. Die Angriffsziele können vielfältig sein. Zum Beispiel Spionage, Tuning oder absichtlich herbeigeführte Unfälle, bei denen Menschenleben gefährdet werden. Kommt es zu einer Beeinträchtigung, so dass das Fahrzeug erst gar nicht benutzt werden kann, ist der Schaden, bezogen auf den Menschen, aber vernachlässigbar. Nachfolgend werden die einzelnen Schwachstellen beschrieben und einer Risikoklasse zugeordnet.

3.1.1. Synchronisation

Ein Angriff auf die Synchronisation hat direkte Auswirkung auf den TT-Datenverkehr, da dieser ohne die Synchronisation nicht stattfinden kann. Beschrieben wurde die Angriffe unteranderem von Steiner (2013) und Isakovic (2011a). Ein Angriff kann dabei auf einen einzelnen Teilnehmer oder auf das gesamte Netzwerk abzielen. Der Angriff auf einen einzelnen Teilnehmer kann

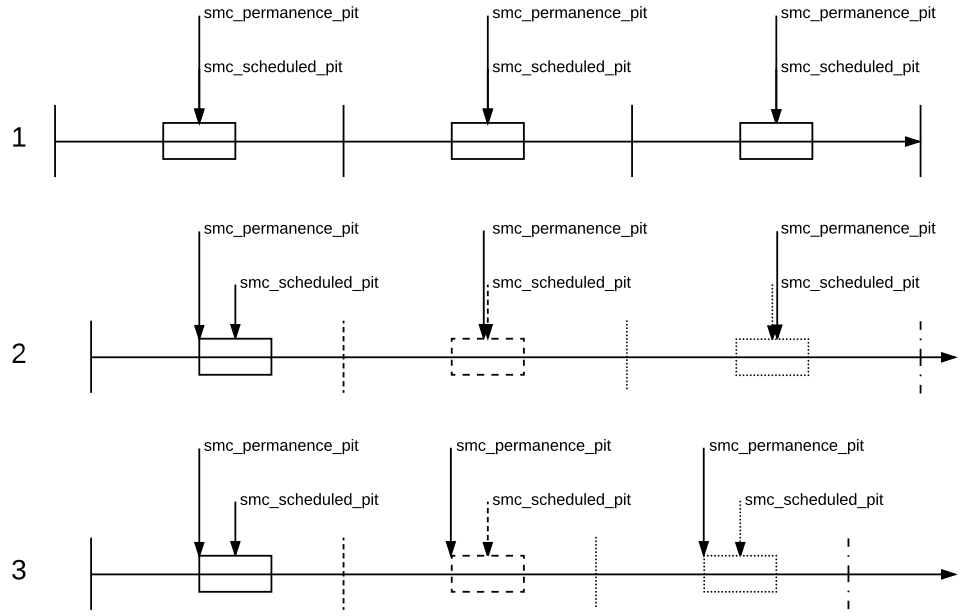

Abbildung 3.3.: Angriff auf die Synchronisation

z.B. zum Ziel haben, eine Funktion des Fahrzeuges zu deaktivieren. Der Angreifer hat mehrere Möglichkeiten die Synchronisation anzugreifen. Hat er direkten Zugang zum Netzwerk, kann er die Kommunikation der Synchronisation manipulieren. Die dafür verwendeten PCFs kann er löschen, verzögern oder beschleunigen.

Beschleunigen kann er einen PCF, indem er einen eigenen PCF generiert und an den Empfänger sendet, bevor der eigentliche PCF beim Empfänger ankommt. Verzögern kann er einen PCF in dem er ihn konsumierend liest und zu einem späteren Zeitpunkt wieder abschickt. So kann er einen Teilnehmer in einen asynchronen Zustand überführen. Will der Angreifer den PCF löschen, liest er ihn konsumierend ein und sendet ihn dann nicht mehr. Treffen die PCFs das Acceptance Window nicht mehr, wird sich der Teilnehmer ggf. neu synchronisieren. Es kann aber auch sein, dass ein Angriff dazu führt, dass ein Teilnehmer seine Asynchronität erst einmal nicht feststellen kann.

Zum Beispiel wenn ein Angreifer die PCFs so manipuliert, dass diese das Acceptance Window einhalten aber die lokale Uhr des Teilnehmers mit jedem Integration Frame immer weiter verschoben wird, was in Bild 3.3 dargestellt ist. Zeile 1 ist die Zeit des CMs zu denen er die PCFs sendet. Zeile 2 zeigt einen normalen SC, welcher sich zu dem CM synchronisiert. In Zeile

3 ist ein SC, dessen lokale Uhr durch einen Angreifer immer weiter verschoben wird, ohne das der SC sich seiner Asynchronität bewusst ist, da die Korrektur innerhalb des Acceptance Window liegt. Das Ergebnis des Angriffs ist, dass TT-Nachrichten ihre Zeitfenster nicht mehr einhalten.

Der Teilnehmer erkennt zwar, dass die TT-Nachrichten nicht mehr im richtigen Zeitfenster eintreffen, aber er reagiert nicht darauf, sondern verwirft die Nachrichten nur. Bei einem IDS könnte das Verwerfen als Symptom für einen Angriff angesehen werden. Allerdings nur, wenn der Teilnehmer selbst TT-Nachrichten empfängt. Ist der Teilnehmer hingegen ein reiner Sender von TT-Nachrichten, kann er daran nicht erkennen, dass er asynchron ist. Ein wesentlich komplexerer Angriff muss durchgeführt werden, wenn ein Angreifer die globale Zeit, bzw. die lokalen Zeiten aller Teilnehmer beeinflussen will.

Dazu muss er entweder PCFs von ausreichend vielen SM oder alle ausgehenden PCFs vom CM manipulieren. Damit erreicht er entweder, dass das gesamte Netzwerk desynchronisiert wird oder, dass die globale Zeit des gesamten Netzwerks beschleunigt oder verzögert wird. So, dass z.B. eine Sekunde Realzeit nicht mehr mit einer Sekunde Netzwerkzeit übereinstimmt. Mit der Formel aus 2.2 kann er bestimmen, wie viele PCFs er manipulieren muss, um die Synchronisation für seine Zwecke zu manipulieren. Greift der Angreifer über den Backbone, mit Hilfe eines Man-In-The-Middle-Angriffs an, kann es sein, dass dieser eine Angriffspunkt reicht, um ausreichend viele PCFs von SM zu manipulieren.

Um einen Angriff auf die Synchronisation in einem TTE zu verhindern, hat Isakovic (2011a) in seiner Arbeit eine Middleware, denSecure Clock Synchronization Algorithm (SCSA), vorgestellt, welche optional eingesetzt werden kann. Somit ist der Betrieb auch möglich, wenn nicht alle Teilnehmer den SCSA implementiert haben. Die Funktionen des SCSA sind ausreichend um die Schwachstellen der TTE-Synchronisation zu schließen. Eine weitere Sicherheitserweiterung ist Annex K für das kurz beschriebene PTP. Diese ist aber wesentlich umfangreicher als die vorgestellte Lösung von Isakovic (2011a) und bedarf weiteren Untersuchungen (vgl. Treytl und Hirschler, 2009; Hirschler und Treytl, 2011; Önal und Kirrmann, 2012), weshalb sie für diese Arbeit außenvorgelassen wird.

Der Secure Clock Synchronization Algorithm

Isakovic (2011a) hat für das TTE eine Middleware entworfen, welche einem Teilnehmer einen zusätzlichen Status (vertrauenswürdig/nicht vertrauenswürdig) hinzufügt. Der Ablauf des Algorithmus sieht vor, das sich die Teilnehmer (Slaves) regelmäßig bei einer zentralen Trusted Authentication Authority (TAA) (Master) melden. Das Schema ist in Abbildung 3.4 zu sehen. In der Nachricht an den Master (Global Time Message (GTM)) ist die lokale Zeit des Slaves

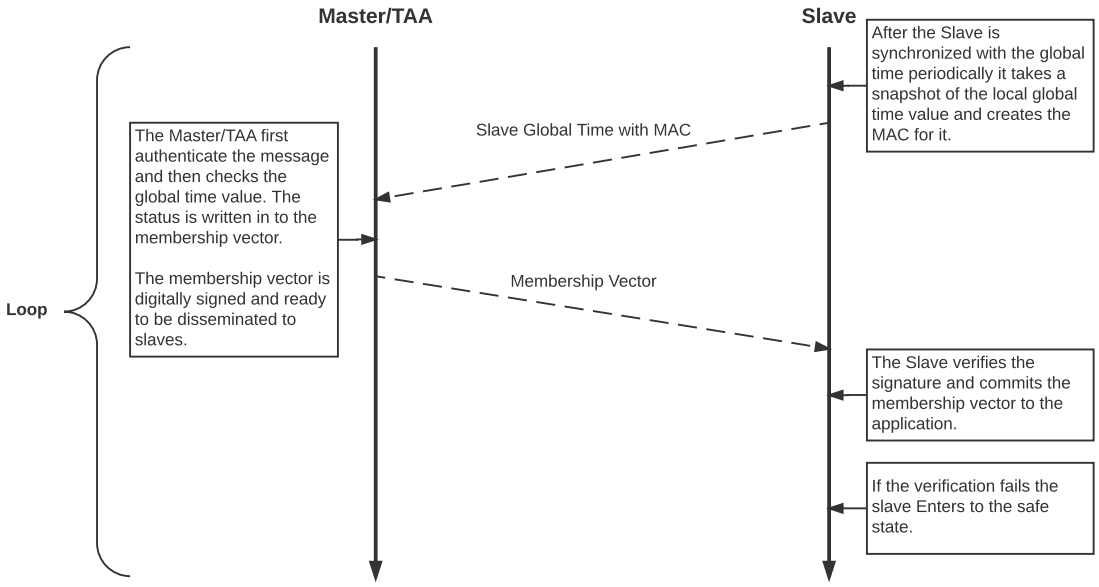

Abbildung 3.4.: Sequenzdiagramm des SCSA nach Isakovic (2011a) S. 38

gespeichert. Deren Integrität wird durch einen Message Authentication Code (MAC)[1] sichergestellt. Der Master sammelt alle Nachrichten und wertet diese aus. Ist der MAC nicht korrekt oder ist die lokale Zeit des Slaves nicht synchron zur Zeit des Masters, wird der Slave als nicht vertraulich eingestuft.

Die Zustände der Slaves bildet der Master in einem Bit-Vektor ab, bei dem eine *0* für nicht vertraulich und eine *1* für vertraulich stehen. Jeder Slave weiß, welches Bit welchem Slave zugeordnet ist. Anschließend sendet der Master eine Nachricht (Membership Vector Message (MVM)) an alle Slaves, welche er mit einer digitalen Signatur vor unbemerkten Änderungen schützt. Somit muss der Master n symmetrische Schlüssel für die Auswertung der MACs der Slaves und mindestens seinen privaten asymmetrischen Schlüssel speichern. Wobei n die Anzahl der Slaves ist. Die Slaves müssen einen symmetrischen Schlüssel für den MAC und den öffentlichen Schlüssel des Masters speichern.

Der SCSA ermöglicht es Slaves zu identifizieren, bei denen ein Angreifer die Zeit mit zulüssigen Werten so manipuliert, dass die Zeit immer weiter abdriftet. Dazu vergleicht der Master die Zeiten der Slaves mit seiner und den Zeiten der anderen Slaves. Wird ein festgelegter

[1] Mit dem Message Authentication Code (MAC) kann die Integrität und Authentizität einer Nachricht festgestellt werden. Gebildet wird er aus einem symmetrischen Schlüssel und den Daten.

Wert für die maximale Abweichung von der globalen Zeit überschritten, wird der Slave als nicht vertrauenswürdig eingestuft.

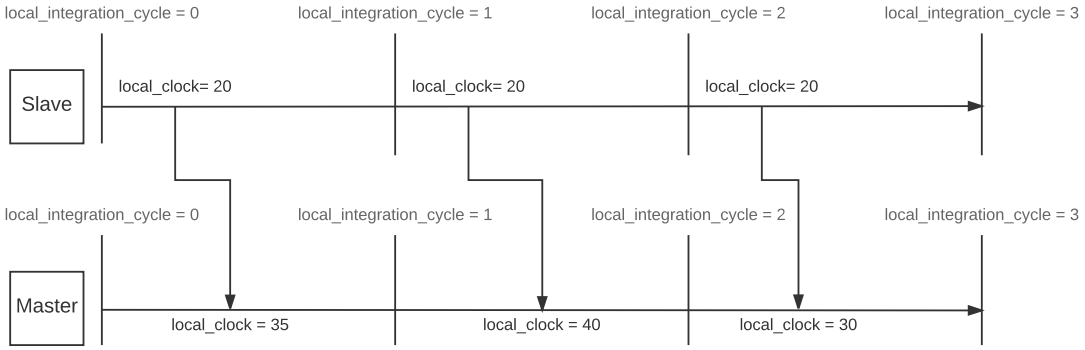

Abbildung 3.5.: Unterschiedliche Empfangszeiten von Global Time Messages

Im TTE setzt sich die aktuelle Uhrzeit aus dem local_integration_cycle und der local_clock zusammen. Bekommt ein Master eine GTM eines Slaves, repräsentieren diese beiden Werte dessen lokale Zeit. Genau wie bei den PCFs kann damit allein aber nicht festgestellt werden, ob ein Slave synchron ist. Denn dafür fehlt die Information, wie lange die Nachricht gebraucht hat, um den Master zu erreichen. Nur dann kann der Master bestimmen, wie groß der Unterschied zwischen den Uhren ist. Bild 3.5 verdeutlicht das Problem.

Die Übertragungszeit hängt zu einem Teil von den variablen Aufenthaltszeiten der Nachrichten auf den Switches, den Sende- und den Empfangsverzögerungen ab (vgl. SAE - AS-2D Time Triggered Systems and Architecture Committee, 2009). Wird die Übertragungszeit nicht bereinigt, erhält der Master falsche Werte. So kann es passieren, dass der Slave einmal als synchron und einmal als asynchron eingestuft wird. Bild 3.5 verdeutlicht dies daran, dass der Master die Nachricht des Slaves in jedem Integration Cycle zu einer anderen Zeit zugestellt bekommt. Beträgt der Unterschied am Anfang nur 5 Zeiteinheiten sind es im nachfolgenden z.B. 20. Für die MVMs ist die Übertragungszeit allerdings nicht wichtig, da hier keine Zeiten enthalten sind.

Für den SCSA wurde kein Schutz vor dem Wiedereinspielen von alten Nachrichten vorgesehen[2]. Für die Nachrichten der Slaves an den Master ist das auch nicht nötig, da dies dort über den local_integration_cycle realisiert wird. Für die MVM gilt das aber nicht und es besteht somit kein Schutz vor dem Wiedereinspielen. Daher kann ein Angreifer die Nachrichten auf-

[2]Der Schutz vor dem Wiedereinspielen (Replay-Schutz) wird durch eine fortlaufende Nummer realisiert, welche bei jeder neuen Nachricht hochgezählt wird. Sie kann durch eine digitale Signatur oder einen MAC vor Manipulation geschützt werden

zeichnen und ggf. so einen asynchronen Slave vertuschen. Der SCSA startet, sobald sich das Netzwerk das erste mal synchronisiert hat.

Zusammenfassung

Beschreibung:

Ein PCF wird durch einen Angreifer verzögert oder beschleunigt.

oder Ein PCF wird durch einen Angreifer gelöscht.

oder Die Integrität eines PCFs ist verletzt.

oder Die Authentizität eines PCFs kann nicht sichergestellt werden.

Konsequenz:

- Eine Komponente oder das System funktionieren nicht mehr oder nur noch eingeschränkt:
  - TT-Nachrichten werden vom Empfänger verworfen.
  - Das Netzwerk oder ein einzelner Teilnehmer müssen sich neu synchronisieren.

Risiko:

- Die Synchronisation bietet einen guten Angriffspunkt, um Teilnehmer zu deaktivieren. Ein Angreifer kann zwar nicht gezielt einzelne Funktionen deaktivieren, dafür aber mit wenig Aufwand einen kompletten Teilnehmer oder das gesamte Netzwerk instabil machen. Im schlimmsten Fall führt er einen Angriff während der Fahrt aus, wodurch Menschenleben gefährdet werden könnten. Somit ist das Risiko für den Eintritt und der daraus folgende Schaden als *sehr hoch* einzuschätzen, da noch keine Maßnahmen zur Verhinderung oder Erkennung eingeführt wurden.

Umsetzung des Angriffs:

- Ein Angreifer hat einen Teilnehmer kompromittiert und kann TT-Nachrichten senden (siehe Bild 3.1).
- Ein Angreifer hat Zugang zum Netzwerk mit Hilfe einer man-in-the-middle-Attacke uns kann TT-Nachrichten senden (siehe Bild 3.2).

3.1.2. Time-Triggered-Datenverkehr

Um sicherheitskritische Funktionen im Auto zu manipulieren oder zu deaktivieren, kann ein Angreifer den TT-Nachrichtenverkehr angreifen. Dazu löscht, verzögert, beschleunigen oder

manipuliert er den Inhalt der TT-Nachrichten, so dass diese nicht mehr im geplanten Zeitfenster eintreffen oder falsche Daten enthalten. Der Empfänger reagiert auf TT-Nachrichten, welche außerhalb des Receive Windows empfangen werden, mit dem verwerfen der TT-Nachricht. Manipulation können nicht erkannt werden. Ein IDS kann dabei helfen, Angriffe zu erkennen, welche mit Hilfe von TT-Nachrichten ausgeführt werden.

Das TTE verwirft alle TT-Nachrichten, welche keine gültige ID besitzen oder nicht im passenden Zeitfenster eintreffen. Immer wenn dies passiert, deutet es auf einen Angriff hin. Vor allem dann, wenn es stetig auftritt. Wobei dies auch auf einen Angriff auf die Synchronisation hinweisen kann. Schwieriger ist es zu erkennen, ob der Payload einer TT-Nachricht manipuliert worden ist. Dazu muss dem IDS bekannt sein, was gültige oder ungültige Werte sind (vgl. Wang u. a., 2006). So können bei Unregelmäßigkeiten der Werte passende Audits erzeugt werden. Wird der Wert aber innerhalb seines gültigen Wertebereich verändert, zum Beispiel Licht an oder aus, dann kann dies nicht als Angriff erkannt werden. Dazu sind weitere Informationen über den Kontext nötig, z.B. wann das Licht an sein darf und wann nicht. Oder es werden korrelierende Nachrichten in die Analyse eingebunden (Parekh u. a., 2006).

Beschreibung:

Eine TT-Nachricht wird durch einen Angreifer verzögert oder beschleunigt.

oder Eine TT-Nachricht wird durch einen Angreifer gelöscht.

oder Die Integrität einer TT-Nachricht ist verletzt.

oder Die Authentizität einer TT-Nachricht kann nicht sichergestellt werden.

Konsequenz:

Eine sicherheitskritische Funktion, Komponente oder System funktionieren nicht mehr oder nur noch eingeschränkt.

Eine sicherheitskritische Funktion, Komponente oder System stehen unter Kontrolle eines Angreifers.

Risiko:

Ein Angriff auf TT-Nachrichten ist für einen Angreifer leicht durchzuführen, wenn er direkten Zugang zum Bordnetz hat. Der resultierende Schaden kann für die Insassen lebensbedrohlich sein, z.B. wenn die Scheinwerfer während der Nacht plötzlich ausgeschalten werden . Es sind bisher keine Maßnahmen zum Schutz der Integrität, Authentizität und Verfügbarkeit ergriffen worden. Daher wird das Risiko als *sehr hoch* eingestuft.

Umsetzung des Angriffs:

Ein Angreifer hat einen Teilnehmer kompromittiert und kann TT-Nachrichten senden oder abfangen (siehe Bild 3.1).

oder Ein Angreifer hat Zugang zum Netzwerk mit Hilfe einer man-in-the-middle-Attacke und kann TT-Nachrichten senden oder abfangen (siehe Bild 3.2).

3.1.3. Event-Datenverkehr

Steiner (2013) beschreibt in seiner Arbeit einen DoS-Angriff auf RC-Nachrichten. „[...]an internal attacker can flood the network with RC messages, leading to a denial of service attack for RC and BE message transmissions on the attacked channel(s)“ (Steiner, 2013, S. 8). Dies ist aber nur deshalb möglich, da der Einsatz des Leaky-Bucket Algorithmus nicht verpflichtend ist, sondern nur dessen Funktionalität vorhanden sein muss (vgl. SAE - AS-2D Time Triggered Systems and Architecture Committee, 2009, S. 21). Da diese Sicherheitsfunktion bereits in allen TTE-kompatiblen Geräten integriert ist, kann diese Schwachstelle von dem IDS vernachlässigt werden.

Das Nichteinhalten der BAG führt dazu, dass Nachrichten beim Switch oder dem Empfänger verworfen werden. Dieses Verhalten weißt auf einen Fehler oder einen Angriff hin und sollte vom IDS erkannt und gemeldet werden, auch wenn solche Angriffe ohne Aussicht auf Erfolg sind. Treffen RC-Nachrichten nicht ein, kann auch das auf einen Angriff hinweisen, obwohl dies für das TTE im Bereich des normalen Verhaltens liegt. Ein weiterer Angriffspunkt ist die Manipulation des Payloads.

Auch bei RC-Nachrichten ist kein Schutz der Integrität vorgesehen. Das IDS kann hier, wie bei den TT-Nachrichten, aber nur bedingt eingesetzt werden. Ohne Kenntnisse über die Anwendung hat das IDS keine Möglichkeit Manipulationen zu erkennen.

Beschreibung:

Eine RC-Nachricht hält das BAG nicht ein.

oder Eine RC-Nachricht wird durch einen Angreifer gelöscht.

oder Die Integrität einer RC-Nachricht ist verletzt.

oder Die Authentizität einer RC-Nachricht kann nicht sichergestellt werden.

Konsequenz:

Fehlende Sicherheitsmaßnahmen machen einen Angriff auf die RC-Kommunikation genauso leicht, wie ein Angriff auf die TT-Kommunikation. Je nachdem welche

Funktionen den RC-Datenverkehr nutzen, kann auch hier der Schaden lebensbedrohlich für die Insassen sein. Daher wird auch hier das Risiko als *sehr hoch* eingestuft.

Risiko:

Ein Angriff auf RC-Nachrichten ist für einen Angreifer leicht durchzuführen, wenn er direkten Zugang zum Bordnetz hat. Der resultierende Schaden kann für die Insassen lebensbedrohlich sein, wenn man in Betracht zieht, dass Funktionen wie der Airbag oder die Lenkung plötzlich nicht mehr funktionieren. Es sind bisher keine Maßnahmen zum Schutz der Integrität, Authentizität und Verfügbarkeit ergriffen worden. Daher wird das Risiko als *sehr hoch* eingestuft.

Umsetzung des Angriffs:

Angreifer hat einen Teilnehmer kompromittiert und kann RC-Nachrichten senden oder abfangen (siehe Bild 3.1)

oder Angreifer hat Zugang zum Netzwerk mit Hilfe einer man-in-the-middle-Attacke und kann RC-Nachrichten senden oder abfangen (siehe Bild 3.2)

3.2. Signaturen - Misuse Detection

Das Netzwerk und deren Teilnehmer sollen durch ein IDS überwacht werden, so dass ein Angriff auf das Netzwerk erkannt werden kann. Dazu wird eine MD eingesetzt, welche Signaturen von Angriffen zur Erkennung nutzt. Anhand der vorgestellten Schwachstellen wird diskutiert, ob das IDS einen Angriff erkennen kann. Anschließend werden die Signaturen zu deren Erkennung vorgestellt.

3.2.1. Synchronisation

Das Manipulieren eines PCFs kann, wie in 3.1.1 beschrieben, eine Störung der Synchronisation zur Folge habe. Mit Hilfe eines IDS kann ein Angriff auf die Synchronisation erkannt werden. Dazu wird der PCF-Verkehr beobachtet und ausgewertet. Nachfolgend wird beschrieben, wie die Signaturen für die Erkennung aufgebaut sind.

Der vorgestellte SCSA kann schematisch in ein IDS übertragen werden, um Angriffe auf die Synchronisation zu erkennen. Dabei kann auch die Schwachstelle eines fehlenden Replay-Schutzes der MVMs ausgemerzt werden. Um die Signatur erstellen zu können, müssen die

Symptome bzw. Schritte eines Angriffs bekannt sein. Aus der Analyse der Schwachstellen ergeben sich folgende Auswirkungen und die dazugehörigen Angriffe auf die PCFs:

PCF nicht empfangen: Der Angreifer löscht PCFs, damit ein Teilnehmer sich nicht mehr synchronisieren kann.

PCF nicht im Acceptance Window empfangen: Der Angreifer hat einen PCF beschleunigt oder verzögert gesendet, so dass der Teilnehmer sich nicht synchronisieren kann. Dazu hat er entweder die TC des PCF manipuliert, den PCF zurückgehalten oder eine PCF wiedereingespielt. Dabei kann er entweder aufgezeichnete PCF verwenden oder selbst welche erstellen.

PCF nicht im richtigen Integration Cycle empfangen: Der Angreifer hat entweder einen aufgezeichneten PCF wiedereingespielt, verzögert oder manipuliert.

PCF-Korrektur lässt Uhr driften: Der Angreifer sendet valide PCF, welche über die Zeit die lokale Uhr des Teilnehmers driften lassen.

Nachfolgend werden alle Symptome einzeln in ihre Signaturnetze überführt. Abbildung 3.6 zeigt die Signatur für das Löschen eines PCFs. Mit jedem neuen local_integration_cycle wird ein Token auf Platz *p2* erzeugt. Wird ein PCF empfangen, erreicht der Token den Abbruchplatz. Bricht ein neuer Integration Cycle an und wurde inzwischen kein PCF empfangen, wird Finalplatz *p3* erreicht und es wird eine Nachricht für das IDS erzeugt. In diesem Signaturnetz befindet sich immer nur ein Token zur selben Zeit.

In Bild 3.7 ist das Signaturnetz für PCFs zu sehen, welche außerhalb des Acceptance Windows empfangen wurden. Um erkennen zu können, ob ein PCF außerhalb des Acceptance Window empfangen wurde, muss der Bereich zwischen dem Schließen und dem öffnen des Acceptance Windows beobachtet werden. Dies kann unabhängig vom aktuellen Integration Cycle betrachtet werden.

Wird ein PCF vor dem öffnen des Acceptance Windows empfangen, gelangt dieser direkt auf den Finalplatz *p3*. Ist das Acceptance Window geöffnet, bleibt der Token solange auf Platz *p2* liegen, bis das Acceptance Window wieder geschlossen ist. Somit stellt *p2* den Zustand innerhalb und Platz *p1* den Zustand außerhalb des Acceptance Window dar. Wird nach dem Schließen des Acceptance Windows ein PCF empfangen, gelangt der Token auch auf den Finalplatz *p3*. Immer wenn die Fork-Transition *t3* schaltet, wird jeweils ein Token auf Platz *p1*

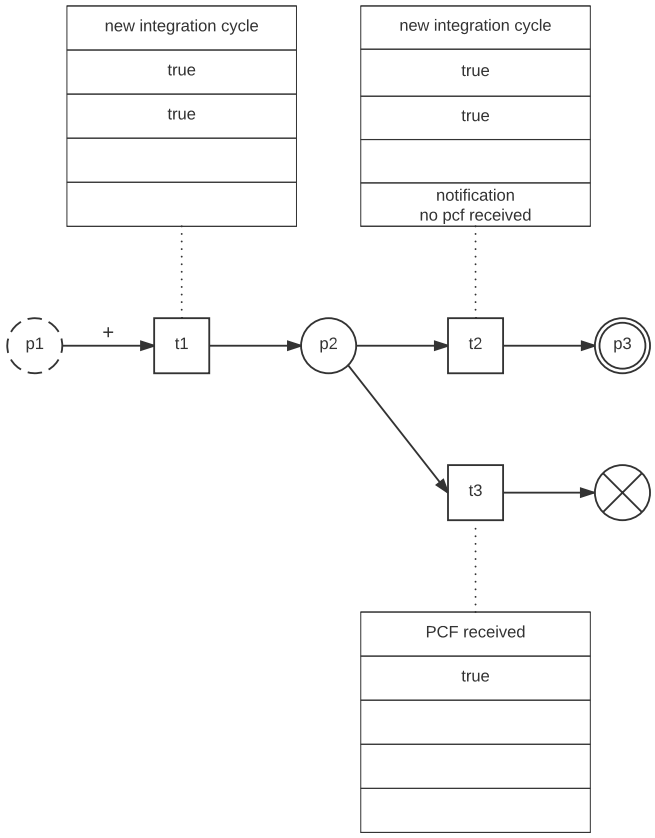

Abbildung 3.6.: Signaturnetz zur Erkennung eines gelöschten PCF

und *p3* abgelegt. Auch hier befindet sich immer nur ein PCF gleichzeitig im Signaturnetz. Es wird nur nach dem Start des IDS und bei einem erkannten Angriff ein neuer Token erzeugt.

Das Signaturnetz aus Abbildung 3.8 prüft beim Empfangen eines PCFs ob dieser den passenden Integration Cycle besitzt. Nur bei einem falschen Integration Cycle wird ein Token transferiert und eine Nachricht generiert. Somit befindet sich auch hier maximal ein Token zur selben Zeit im Signaturnetz.

Das letzte Signaturnetz dient zur Erkennung einer driftenden Uhr. Der Ansatz ist dem Secure Clock Synchronization Algorithm von Isakovic (2011a) entnommen. Unter der Annahme, dass mit dem erreichen des Stable-Zustands die lokale Uhr eingepegelt ist, wird die Änderung durch die clock_corr beobachtet. Nachdem eine driftende Uhr durch das Signaturnetz festgestellt worden ist, sendet das IDS des Teilnehmers eine Nachricht an das zentrale IDS.

Bild 3.9 zeigt das genannte Signaturnetz. Bei Empfang eines neuen PCFs wird einmalig ein Token von Platz *p2* erzeugt. Dabei wird dem Token der Wert der clock_corr als correction

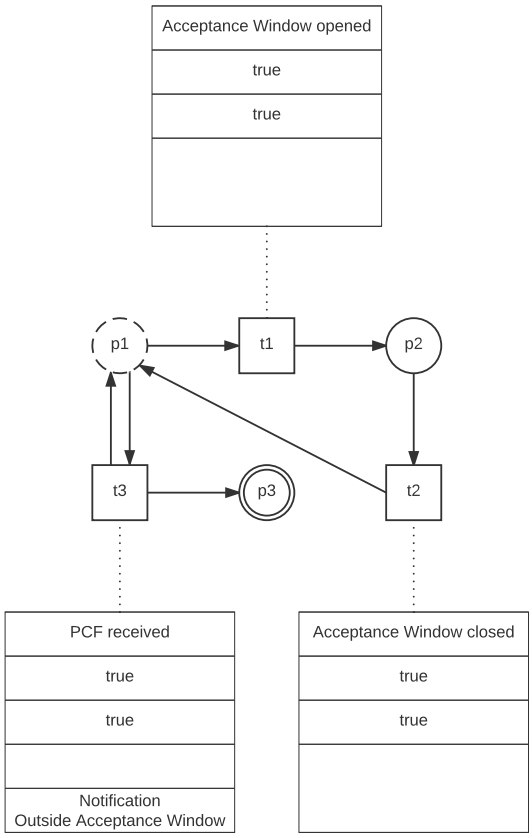

Abbildung 3.7.: Signaturnetz zur Erkennung von PCFs außerhalb des Acceptance Window

hinzugefügt. Anschließend wird die clock_corr von jedem weiteren PCF auf den Wert der correction im Token addiert. Da die clock_corr sowohl positiv als auch negativ sein kann, sollte sich die correction bei einer stabilen Synchronisation um Null herum einpendeln.

Überschreitet die akkumulierte correction einen kritischen Wert (correction_threshold), wird eine Benachrichtigung an das zentrale IDS gesendet. Wird anschließend ein neuer PCF empfangen, wird die correction in dem Token mit der aktuellen clock_corr überschrieben und ein neuer Zyklus beginnt. Somit befindet sich immer nur ein Token im Signaturnetz.

Die vier vorgestellten Teilnetze lassen sich zu einem Signaturnetz zusammenfassen, da Sie unterschiedliche Bereiche von Angriffszielen abdecken. Dadurch hat man nur noch ein Signaturnetz mit einem einzigen Token. Die Anzahl der benötigten Plätze reduziert sich dabei von dreizehn auf vier, die Anzahl der Transitionen bleibt gleich. Dafür müssen in dem Token drei (correction, pcf_received, acc_window) anstatt einer (correction) Variablen gespeichert

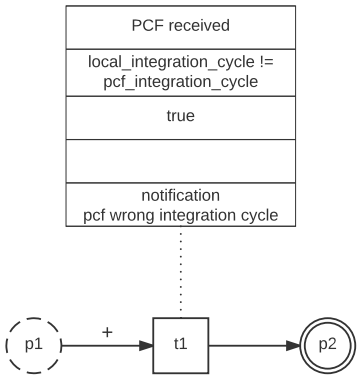

Abbildung 3.8.: Signaturnetz zur Erkennung von PCFs mit falschem Integration Cycle

werden, wobei die beiden zusätzlichen Variablen jeweils in mit Hilfe eines Bits kodiert werden können.

Abbildung 3.10 zeigt das resultierende Signaturnetz. Für die Übersichtlichkeit wurden der Initial- und der Finalplatz jeweils doppelt abgebildet. Mit dem Erreichen des CM_STABLE-Zustands wird das Signaturnetz initialisiert. Anschließend wird, durch einen neuen Integration Cycle, der Token auf Platz *p2* transferiert. Wird nun ein PCF empfangen, liegt dieser außerhalb des Acceptance Windows (*t7*). Anschließend öffnet das Acceptance Window und *t2* schaltet.

Mit dem Empfangen eines PCFs wird entweder *t7* (falscher Integration Cycle) oder *t3* (valider PCF) geschalten. Schaltet *t3* wird die errechnete Clock Correction (clock_corr) auf den gespeicherten Wert (corr) auf akkumuliert. Wird kein PCF empfangen, solange das Acceptance Window geöffnet ist, schaltet *t4* und der Token gelangt, wie bei Transition *t3*, auf Platz *p3*. Wird erst nach dem Schließen des Acceptance Windows ein PCF empfangen, schaltet Transition *t10* und der Token gelangt auf den Finalplatz.

Startet ein neuer Integration Cycle, wird unterschieden, ob bereits ein valider *PCF* empfangen wurde (*t8*) oder nicht (*t9*). Schaltet Transition *t8*, wird beim Empfangen eines neuen validen PCFs die Clock Correction wieder auf den bestehenden Wert akkumuliert. Erreicht der Wert corr einen festgelegten Grenzwert (correction_threshold), schaltet Transition *t5*. Immer wenn ein Token auf den Finalplatz *p4* gelangt wird immer auch ein neuer Token auf *p1* erzeugt, so dass eine erneute Überwachung starten kann.

Eine Situation, welche nicht von dem Signaturnetz erfasst wird ist, wenn mehr als ein PCF innerhalb des Acceptance Windows empfangen wird (siehe *p3*). Zur Zeit wird nur der erste gültige PCF untersucht. Damit alle PCFs vom IDS untersucht werden können, muss für jeden Sender ein eigener Token angelegt werden, welche sich durch die Source Port

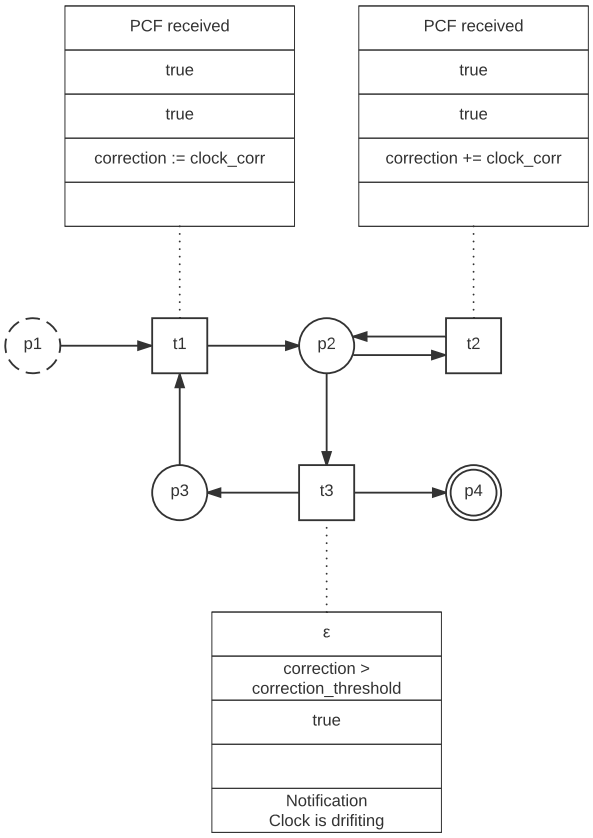

Abbildung 3.9.: Signaturnetz zur Erkennung einer driftenden Uhr

Identity unterscheiden. Initialisiert wird das Signaturnetz mit erreichen des Stable-Zustands der Synchronisation. Im Anhang befinden sich Beispiele mit Token und Werten für alle Angriffe (Abbildung A.1 bis A.11).

3.2.2. Time-Triggered-Nachrichten

Wie beschrieben, kann die Verzögerung und Beschleunigung von TT-Nachrichten oder die Verwendung einer falschen Virtual Link ID mit Hilfe eines IDSs erkannt werden. Dazu muss überprüft werden, ob eine TT-Nachricht rechtzeitig eintrifft und ob sie eine gültige Virtual Link ID besitzt. Abbildung 3.11 zeigt das resultierende Signaturnetz.

Startplatz *p1* enthält so viele Token, dass alle zu empfangenen TT-Nachrichten durch jeweils einen eigenen Token abgebildet werden. Jeder Token enthält bereits seine Virtual Link ID, welche im Feld VL-ID abgelegt ist. Mit dem Start eines neuen Integration Cycles werden auf Platz *p2* Kopien aller Token abgelegt. Öffnet sich ein neues Receive Window, wird der

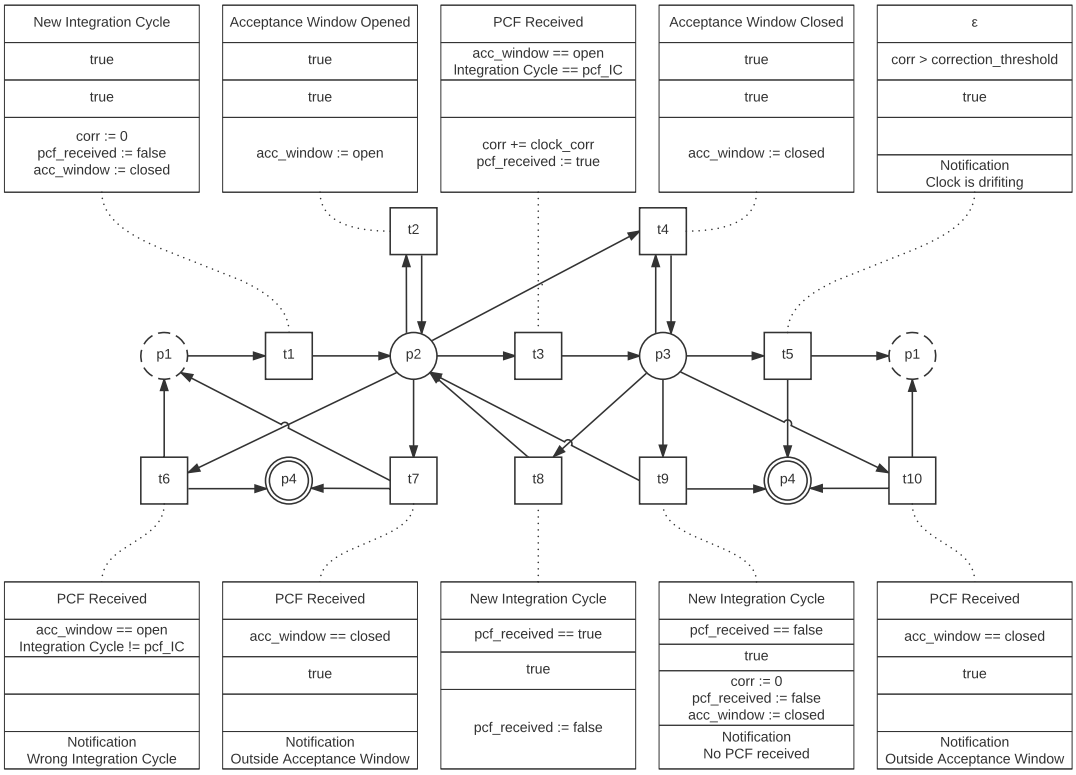

Abbildung 3.10.: Signaturnetz zur Erkennung aller Angriffe auf die Synchronisation

entsprechende Token auf Platz *p3* transferiert. Bei Empfang einer passenden TT-Nachricht gelangt der Token auf den Abbruchplatz. Wird eine TT-Nachricht empfangen, welche nicht die passende Virtual Link ID besitzt, gelangt der Token auf Finalplatz *p4* und es wird die Nachricht *Wrong Virtual Link ID* erzeugt.

Schließt sich das Receive Window wieder und wurde bis dahin keine TT-Nachricht empfangen, wird der Token zurück auf Platz *p2* kopiert. Wird erst jetzt eine TT-Nachricht mit der passenden Virtual Link ID empfangen, schaltet Transition *t5* und der Token gelangt auf den Finalplatz. Dabei wird die Nachricht *TT-Message out of Window* generiert. Sollten TT-Nachrichten gelöscht worden sein, wird dies mit dem Start eines neuen Integration Cycles bekannt. Dann schaltet Transition *t4* und dem IDS wird mitgeteilt, welche TT-Nachrichten nicht empfangen wurden. Obwohl dies kein Fehlverhalten ist, kann dies auf eine Manipulation hinweisen.

TT-Nachrichten, welche nicht in ihrem Receive Window empfangen wurden, können auch auf einen asynchronen Teilnehmer hinweisen. Dies sollte das IDS in seine Analyse mitein-

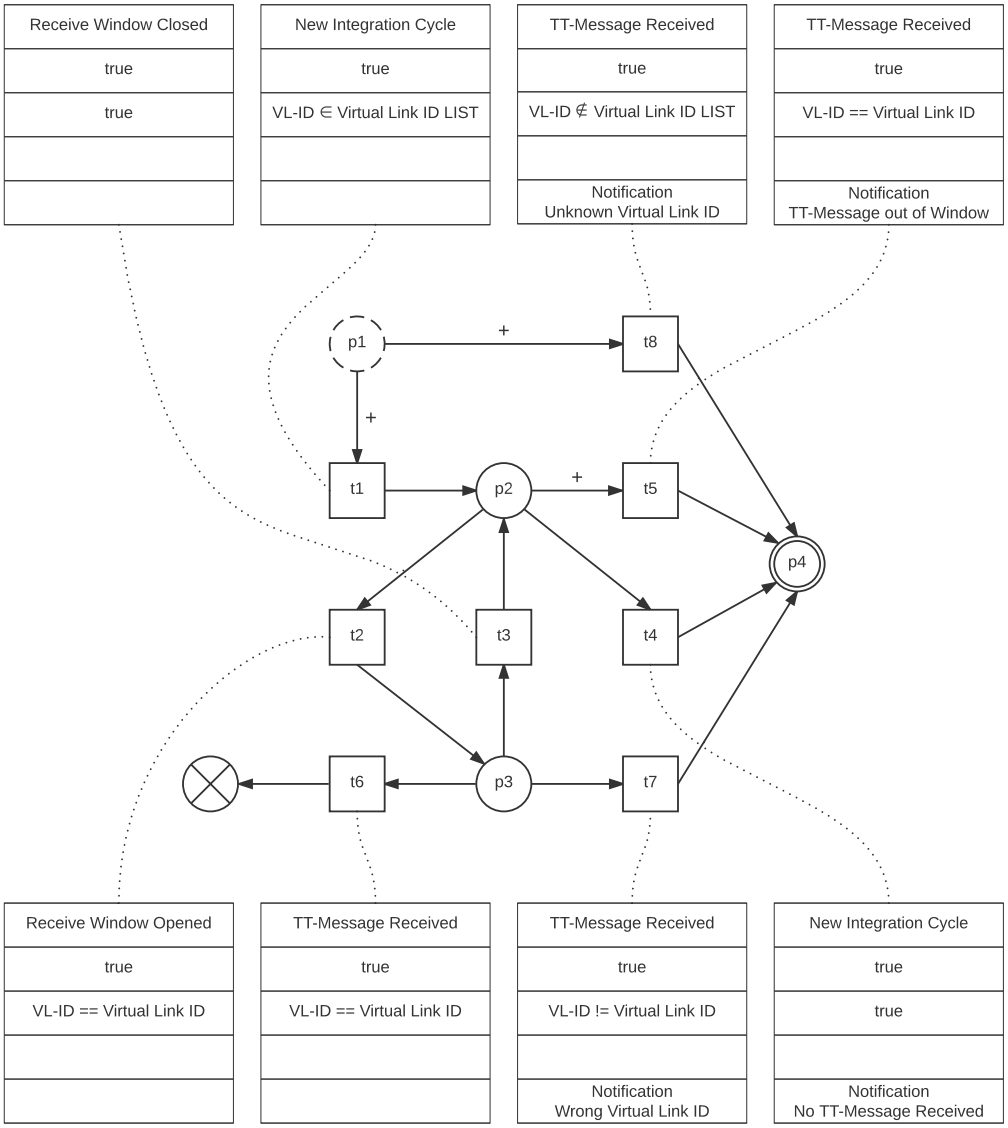

Abbildung 3.11.: Signaturnetz zur Erkennung von Angriffen auf TT-Nachrichten

beziehen. Auch hierzu gibt es im Anhang Abbildungen mit Beispielen zu allen Angriffen (Abbildung A.12 bis A.22). Wie in der Schwachstellenanalyse bereits erwähnt, kann ein IDS auch dazu verwendet werden Manipulation des Payloads zu erkennen, wenn es hinreichende Informationen über die Daten besitzt.

Somit können unter anderem Sensorwerte untersucht werden (vgl. Meier, 2008, S. 61). Die einzelnen Werte werden dafür auf maximale und minimale Grenzen überprüft sowie miteinander in Relation gesetzt. übersteigt z.B. die Änderung eines Wertes innerhalb einer definierten Zeit einen bestimmten Schwellwert, wird ein Ereignis ausgelöst. Hier bietet sich alternativ auch die AD an, da hier nicht alle Fälle von anormalen Verhalten abgebildet werden müssen. Die AD kann in einer sicheren Umgebung angeleitet werden und ist in der Lage Angriffe zu finden, die der MD nicht bekannt sind. Alle anormalen Werte oder Wertänderungen werden auch hier durch ein Ereignis repräsentiert.

3.2.3. Rate-Constrained-Nachrichten

Ein Kontrolle des Payloads von RC-Nachrichten unterliegt denselben Restriktionen wie bei TT-Nachrichten. Es müssen das Verhalten und die Schwellwerte bekannt sein, welche das IDS zu überprüfen hat. Das protokollabhängige Verhalten kann aber beobachtet werden. Auch, wenn ein Angreifer mit einem Angriff darauf keinen Erfolg hat, kann zumindest eine Warnung generiert werden. Ein Angreifer kann versuchen mehr als die in der Konfiguration festgeschriebene Bandbreite zu nutzen. Die eintreffenden Nachrichten werden dann bei dem Switch durch den Leaky Bucket-Algorithmus verworfen.

In dem Signaturnetz in Abbildung 3.12 wird die Überwachung der BAG durch Transition *t3* für die jeweilige Virtual Link ID mit dem Eintreten des Events *Frame Disabled* aktiviert. Wird nun eine RC-Nachricht mit dieser Virtual Link ID empfangen, schaltet *t5* den Token auf den Finalplatz. Wird keine passende Nachricht empfangen bis das Event *Frame Enabled* eingetreten ist, wird der Token über *t4* auf den Abbruchplatz geschalten.

Genau wie bei den periodischen TT-Nachrichten ist das Fernbleiben von Nachrichten nicht gleich ein Angriff. Werden aber Nachrichten erwartet, sollte das IDS dies überwachen. Dafür kann ein applikationsabhängiger Schwellwert (X) festgelegt werden. Übersteigt die Anzahl der Zyklen ohne eine Nachricht diesen Wert, meldet das IDS einen Angriff. Obwohl das Senden von RC-Nachrichten von den Zyklen entkoppelt ist, können die Zyklen als Zeitbasis herangezogen werden.

Startet ein neuer Zyklus, werden so viele Token nach *p2* kopiert, wie es Virtual Link IDs für diesen Teilnehmer gibt. Mit jedem neuen Zyklus in dem keine RC-Nachricht mit passender Virtual Link ID eintrifft, wird der *Integration Cycle Count* hochgezählt. Sobald er den

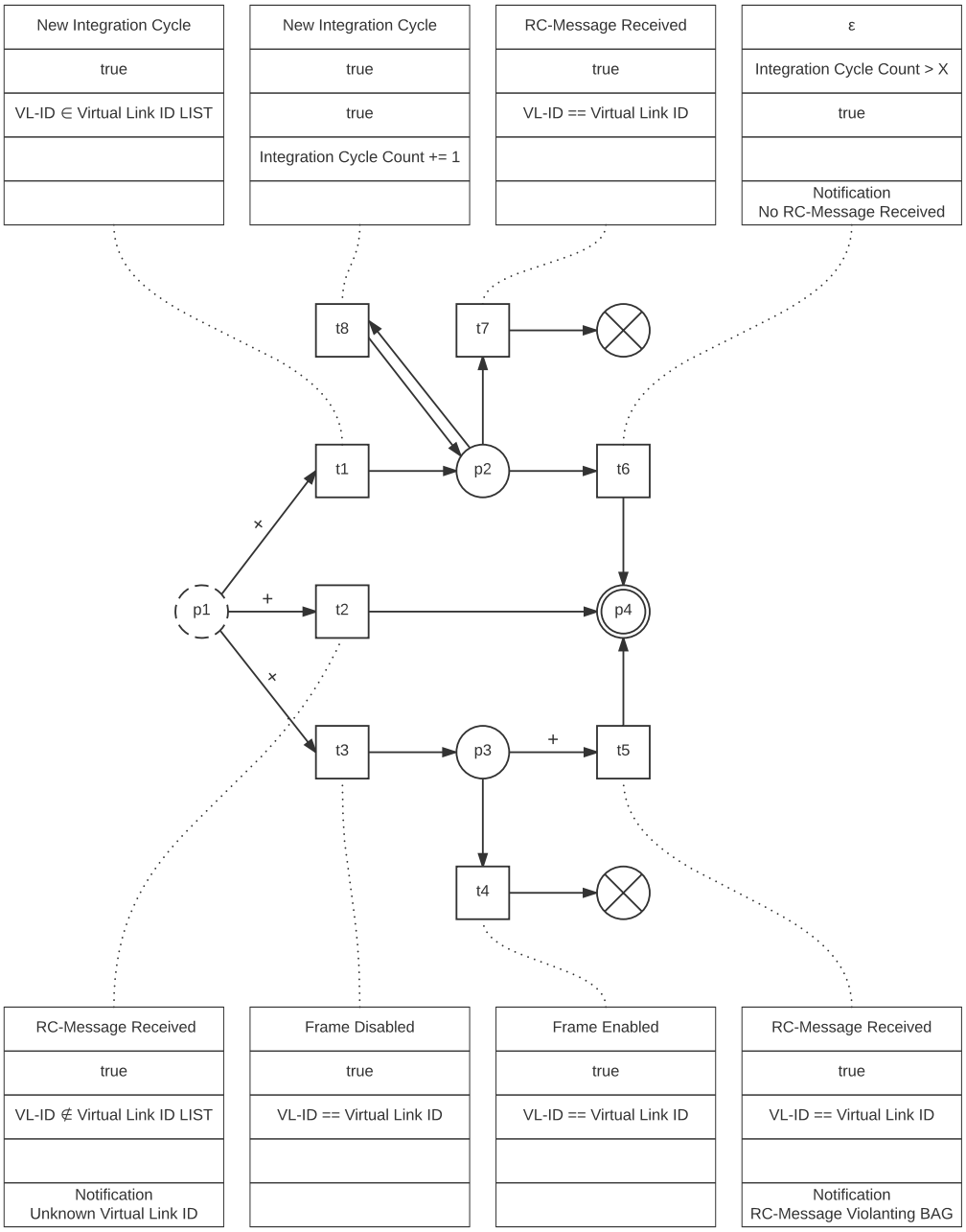

Abbildung 3.12.: Signaturnetz zur Erkennung von Angriffen auf RC-Nachrichten

eingestellten Wert X erreicht, schaltet Transition *t5*. Über Transition *t7* gelangen Token, dessen RC-Nachrichten eingetroffen sind auf den Abbruchplatz. Der Wert X sollte für jeden RC-Nachrichtentyp separat eingestellt werden können.

Wird eine RC-Nachricht mit einer ungültigen Virtual Link ID empfangen, schaltet *t2*. Im Anhang wird für jeden Angriff ein Beispiel mit Token und dessen Variablenbindung durchgespielt (Abbildung A.23 bis A.30). Es befinden sich maximal so viele Token im Signaturnetz, wie es konfigurierte Virtual Link IDs für den Teilnehmer gibt, plus die zur Zeit gesperrten Virtual Link IDs.

3.3. Integration in des Bordnetz

Für die Integration des IDS sind vor allem zwei Punkte wichtig. Zum einen, wo die Sensoren für die Überwachung installiert werden und zum anderen, wie die Kommunikation der Sensoren untereinander realisiert werden muss. Darüberhinaus muss es eine zentrale ID-Komponente geben, welche zur Koordination der Reaktionen auf einen erkannten Angriffe eingesetzt wird, sowie für eine Kommunikation mit externen Teilnehmern wie dem Hersteller. Alle IDS welche nicht das zentrale IDS sind werden weiter als lokales IDS bezeichnbet. Die zentrale Komponente wird zudem für die Überprüfung der Echtheit und Plausibilität der eintreffenden Befunde herangezogen.

Wahl und Platzierung der Sensoren

Es bieten sich drei unterschiedliche Punkte für die Integration der Sensoren an Sie können zum einen auf der Ebene des Stacks der TTE-Komponente integriert werden. Das hat den Vorteil, das keine weitere Hardware verwendet werden muss und die Überwachung der vom Controller erzeugten Events sehr nah durchgeführt werden kann. Das IDS kann nach einem Befund direkt Einfluss auf die Nachfolgenden Applikationen nehmen und Angriffe weiter Unsteruchen, z.B. mit dem Einsatz eines Honeypots. Das für das Auditing festgelegte Austauschformat muss dafür aber in den Stack implementiert werden.

Eine weitere Möglichkeit ist die Integration des Sensors in die Firmware der TTE-Komponente. Auch hier muss das Austauschformat implementiert werden. Das Auditing ist dann auf ein Logging der Events des TTE-Controllers durch die Firmware angewiesen. Diese Variante lässt den TTE-Stack unverändert. Bei Änderungen am diesem muss somit keine eigene Version nebenher gepflegt werden.

Die dritte Möglichkeit ist die Verwendung eines netzwerkbasierten Sensors. Das hat den Vorteil, das in jedem Fall ausreichend Ressourcen für die Überwachung zur Verfügung stehen.

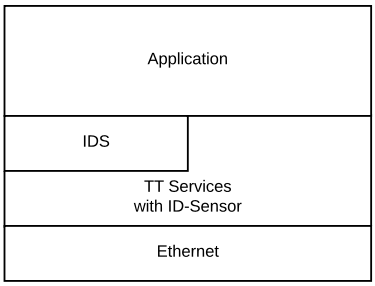

Abbildung 3.13.: Sensor und IDS im TTE-Stack

Weder der Stack noch die Firmware müssen verändert werden. Dafür muss der Sensor TTE-fähig sein. Denn ob z.B. ein PCF rechtzeitig angekommen ist, kann der Sensor nur beurteilen, wenn er selbst synchron ist. Es wird damit aber euch eine weitere Fehlerquelle integriert, welche bei einem Ausfall im schlimmsten Fall für die Stilllegung des Fahrzeugs sorgen kann.

Für die Umsetzung wird die Verwendung von hostbasierten Sensoren innerhalb des TTE-Stacks vorgeschlagen, deren Schema in den Abbildung 3.13 dargestellt ist. Somit lassen sich große Logdateien vermeiden, da die Events direkt verarbeitet werden können. Unbrauchbare Events können vom IDS einfach verworfen werden. Die Anpassung des TTE-Stacks beinhaltet keine Kompatibilitätsprobleme mit unveränderten TTE-Teilnehmern, da die Überwachung keine Funktionen verändert. Zudem hat sich der Standard des Protokolls seit 2011 nicht geändert und eine Anpassung, z.B. zur Verbesserung der Sicherheitsfunktionen, ist offiziell nicht in Arbeit. Es braucht keine neue Hardware hinzugefügt werden, was allerdings voraussetzt, dass die TTE-Komponente über ausreichend Ressourcen verfügt.

Es wird zudem vorgeschlagen, die Sensoren des IDSs in alle TTE-Komponenten zu integrieren, denn eine einzelne attackierte TTE-Komponente reicht aus, um einen hohen Schaden anzurichten.

Auditing

Da der Stack um das Auditing erweitert wird, muss festgelegt werden, wie das Austauschformat strukturiert wird. Dazu wird sich an dem IDMEF orientiert, welches in seinen Grundzügen übernommen wird (siehe Tabelle 3.4). Ersetzt werden muss die AnalyzerTime (Zeit des Sensors), CreateTime und die DetectTime, da diese nun den Integration Cycle und die Locale Clock beinhalten müssen. Die AnalyzerTime ist dabei nicht mehr relevant, da sich der Sensor auf der selben Ebene befindet wie der TTE-Stack. Zudem sind alle Teilnehmer synchron und haben

Tabelle 3.4.: Auditing-Nachrichtenklassen

Nachricht	Feld	Feld
Alert	messageid	
	Analyzer	
	AnalyzerTime	
	CreateTime	Integration Cycle
		Locale Clock
	DetectTime	Integration Cycle
		Locale Clock
	AdditionalData	

die gleiche lokale Zeit. Die CreateTime und die ClockTime werden erweitert. Die Source und das Target werden nicht weiter benötigt.

Der Analyzer (Sensor) gibt schon Auskunft darüber, auf welchem Teilnehmer das Ereignis generiert wurde. Die messageid identifiziert die auslösende Nachricht. Damit ist dem IDS genau bekannt welche Route die Nachricht hatte und an welchem Punkt der Route der Angriff erkannt wurde. Classification und Assessment werden nicht verwendet und können daher auch weggelassen werden. AdditionalData wird dazu verwendet, um Informationen über bereits eingeleitete oder einzuleitende Maßnahmen zu kommunizieren.

Für jedes Ereignis wird eine Unterklasse von Alert angelegt. Das beinhaltet: New Integration Cycle, Acceptance Window Opened, Acceptance Window Closed, PCF Received, Receive Window Opened, Receive Window Closed, TT-Message Received, RC-Message Received, Frame Enabled und Frame Disabled. Zudem werden die nötigen Parameter integriert. New Integration Cycle enthält ein weiteres Feld für den Integration Cycle, ist der Wert nicht gesetzt, wird der Integration Cycle vom IDS um eins erhöht. Die Klasse PCF Received bekommt den Parameter clock_corr hinzugefügt. TT-Message Received, RC-Message Received, Receive Window Opened und Receive Window Closed haben ein Feld Virtual Link ID.

Kommunikation der Sensoren

In Kapitel 2 wurden bereits das zustandsbasierte und das aktionsbasierte Auditing vorgestellt. Beide Varianten haben spezielle Vorteile, welche für das IDS verwendet werden. Die Sensoren müssen einer dem zentralen IDS berichten, wenn diese einen Angriff erkannt haben. Das IDS muss dann den Bericht bewerten und Maßnahmen veranlassen. Zum Beispiel die Benachrichtigung der Insassen und des Herstellers. Damit Angreifer keine Falschmeldungen an das IDS senden können, müssen die Nachrichten geschützt werden. In der Projektarbeit von Phieler

(2016) wurde eine mögliche Realisierung einer durch Zertifikate geschützten Kommunikation vorgestellt. Es kann aber auch ein anderes Konzept genutzt werden. Da die Nachrichten nicht den zeitlichen Restriktionen der TT-Kommunikation unterliegen, kann eine Verzögerung, welche durch das Erstellen und Prüfen der Signaturen entsteht, akzeptiert werden.

Das Modell für die Kommunikation zwischen den Sensoren und des zentralen IDS besteht aus der Kombination von zustandsbasierten und aktionsbasierten Meldungen. Zustandsbasierte Meldungen werden von dem zentralen IDS verwendet, um feststellen zu können, ob die Sensoren noch funktionsfähig sind. Dies wird mit der Heartbeat-Klasse der IDMEF realisiert. In einem definierten Intervall senden die Sensoren kurze signierte Nachrichten an das zentrale IDS. Dieses quittiert den Eingang, die Aktualität und die Authentizität der Nachrichten. Da die Sensoren Informationen über den aktuellen Integration Cycle besitzen, kann dieser für den Replay-Schutz verwendet werden.

Die aktionsbasierten Meldungen werden von den Sensoren genutzt, um dem zentralen IDS mit Hilfe der Reporting Language Bericht über eine Entdeckung zu machen. Sie enthalten neben dem Bericht auch einen Replay-Schutz und eine Signatur. Es sollte dem zentralen IDS nach der Analyse des Berichts auch möglich sein, weitere Daten, auch von anderen Sensoren, anzufordern, um den Bericht überprüfen zu können. Für die MD ist das nicht unbedingt nötig, da die Schwachstellen und die Durchführung der Angriffe klar definiert sind und unbekannte Angriffe nicht erkannt werden können. Für eine AD ist dies aber von Vorteil, um z.B. neue Angriffe mit Hilfe eines Expertensystems analysieren zu können.

Dazu hat Zhang und Lee (2000) ein Konzept für kabellose Ad-Hoc-Netzwerke vorgestellt, welches adaptiert werden kann. Jeder Teilnehmer übersendet auf Anfrage der zentralen IDS seinen aktuellen Zustand. Da die lokalen IDS Angriffe selbst diagnostizieren können, befinden sie sich entweder im Zustand attackiert oder nicht attackiert. Wird zusätzlich eine AD verwendet, kann noch hinzugefügt werden, auf welcher Basis die Annahme getroffen wurde. Gerade bei der AD kann somit festgestellt werden, wie viele Teilnehmer eine Anomalie erkannt haben und ob dies plausibel ist.

3.4. Zusammenfassung

Es ist möglich die bekannten Schwachstellen des TTE mit Hilfe von Signaturen abzubilden. Die verwendeten Events sind ausschließlich Events, welche im TTE-Standard beschrieben werden. Dies vereinfacht unter anderem das Logging der Events. Die vorgestellten Signaturnetze sind übersichtlich und nutzen eine geringe Zahl an Token (1 Token je Virtual Link ID). Der

Payload von PCFs kann nicht auf Integrität und Authentizität überprüft werden, daher wird das Verhalten der Nachrichten und der Synchronisation beobachtet.

Solange sich die Werte für die Anpassung der lokalen Zeit im festgelegten Rahmen befinden, sind keine weiteren Maßnahmen durch das IDS nötig. Fehlende Nachrichten oder solche welche sich nicht nach der Spezifikation verhalten, werden von der Signatur abgebildet. Für TT- und RC-Nachrichten gilt, für den Payload und den Header der Nachrichten, dasselbe wie für die PCFs. Das IDS wäre in der Lage den Payload anhand von Signaturen zu überwachen (vgl. Meier, 2008). Dafür braucht es aber Kenntnisse über die Applikation. Protokollspezifisches Fehlverhalten wird von den Signaturnetzen erkannt und es werden Benachrichtigungen erzeugt.

Die Sensoren zur Überwachung der Ereignisse werden innerhalb des Stacks platziert. Zum einen, um Ressourcen zu schonen, da durch die direkte Verarbeitung der Ereignisse keine großen Logdateien angelegt werden müssen, und zum anderen, um direkt auf Angriffe reagieren zu können, in dem diese z.B. in einen Honeypot umgeleitet werden. Lokale IDS melden jeden Befund an das zentrale IDS. Dort werden die Befunde nochmals ausgewertet und können mit den Daten von weiteren lokalen IDS verglichen werden. So können Angriffe eingegrenzt und Reaktionen besser koordiniert werden. Zudem stellt es die Daten für Externe Teilnehmer, wie den Hersteller, bereit.

Für die Kommunikation unter den lokalen und zentralen IDS werden signierte und vor Wiedereinspielung geschützte Nachrichten eingesetzt. Die Schlüssel für die Signierung können mit Hilfe von Zertifikaten erstellt werden. Im Gegensatz zu TT-Nachrichten, bei denen die Verwendung von Signaturen zu Problemen führen kann, können hier Signaturen eingesetzt werden.

4. Fazit und Ausblick

Da das TTE aktuell über keine eigenen Maßnahmen zur Sicherung des Netzwerkverkehrs verfügt, ist es notwendig das TTE um diese zu erweitern, wenn es als Bordnetz eingesetzt werden soll. Die wenigen veröffentlichten Arbeiten schlugen die Verwendung von Signaturen zum Schutz der Integrität und Authentizität vor (vgl. Isakovic, 2011b; Steiner, 2013; Wasicek u. a., 2011). Dabei muss das TTE-Protokoll in jedem Fall erweitert werden. In den Arbeiten wird darauf hingewiesen, dass vor allem die TT-Kommunikation den Einsatz von Verschlüsselungsmechaniken erschwert. Daher wurde unter anderem auch eine nachträgliche Authentifikation und Prüfung der Integrität vorgeschlagen (vgl. Wasicek u. a., 2011).

In dieser Arbeit wurde mit dem IDS eine weitere Herangehensweise zum Schutz des Nachrichtenverkehrs vorgestellt. Die Überwachung durch ein IDS ist aber nur in der Lage ausgeführte Angriffe zu erkennen, nicht aber sie zu verhindern. Erst wenn ein Angriff diagnostiziert worden ist, können Maßnahmen eingeleitet werden, welche weiteren Schaden durch Angriffe verhindern. Steiner (2013) hat in seiner Veröffentlichung die Schwachstellen des TTEs aufgezeigt. Im ersten Schritt wurde aufgelistet, wie ein Angreifer die Schwachstellen ausnutzen kann, welche Ziele er damit erreichen kann und wie hoch das Risiko für das TTE, das Fahrzeug oder die Insassen ist. Anschließend wurden die Schritte zum Ausnutzen der Schwachstellen, mit Hilfe von Signaturnetzen, beschrieben. Dabei lässt sich erkennen, dass die Schwachstellen nur wenig Ereignisse brauchen, um ausgenutzt zu werden.

Überwacht werden dabei nur der Header der Nachrichten und die Events, die durch den Controller des TTEs ausgelöst werden. Es ist auch möglich den Payload der Nachrichten zu überwachen und, z.B. mit Hilfe von Plausibilitätstests, auszuwerten. Dafür muss aber der Wertebereich der Daten bekannt sein. Dies stand für diese Arbeit nicht ausreichend zur Verfügung. Andere Arbeiten haben gezeigt, dass somit eine Verbesserung der Erkennung von Angriffen erzielt werden kann (vgl. Roesch, 1999; Wang und Stolfo, 2004).

Damit die Befunde über erkannte Angriffe, auch mit Hinblick auf eine spätere Verwendung einer AD, auf ihre Plausibilität und Echtheit überprüft werden können, gibt es ein zentrales IDS. Dieser müssen alle Befunde gemeldet werden. Es kann weitere Informationen von anderen

lokalen IDS einholen und mit externen Teilnehmern kommunizieren. Zudem sollte es die Befunde mit allen erhobenen Informationen für eine Offline-Analyse vorhalten.

Ausblick

Als nächsten Schritt kann das gesamte IDS mit einem Versuchsaufbau untersucht werden. Dies kann z.B. mit einer Simulation umgesetzt werden, um Kosten und Zeit zu sparen. So kann man die Funktionalität des Konzepts testen und erste Erkenntnisse über den Ressourcenverbrauch erlangen. Wichtig ist vor allem, wie groß der Speicherbedarf für die Signaturen ist. Nach Feststellung der Funktionsfähigkeit kann das IDS in einen Versuchsaufbau mit realer Hardware umgesetzt werden, um reale Werte ermitteln zu können.

Wie angesprochen, kann die Kombination einer MD mit einer AD das Ergebnis der Erkennung verbessern, auch wenn die AD den Nachteil von falsch oder gar nicht erkannten Angriffen mit sich bringt. So ist man aber in der Lage auch neue Angriffe erkennen zu können. Daher sollte auch die AD mit dem Fokus auf das Bordnetz betrachtete werden.

Zudem ist wichtig, wie neue Schwachstellen in den Erkennungsprozess integriert werden können. Die Möglichkeit der Aktualisierung von Sicherheitsfunktionen ist aber nicht nur für das IDS wichtig, sondern auch für die restlichen Komponenten. Daher sollte dies auf einer höheren Ebene untersucht werden.

Abkürzungsverzeichnis

AD Anomaly Detection

BAG Bandwitdh Allocation Gap

BE Best Effort

BSI Bundesamt für Sicherheit in der Informationstechnik

C2C Car-to-Car

C2E Car-to-Environment

CAN Controller Area Network

CM Compression Master

COM/MON Commander/Monitor

CPA Coloured Petri Net Automaton

CPN Coloured Petri Net

DoS Denial of Service

DDoS Distributed Denial of Service

ECU Electronic Control Unit

EE Echtzeit Ethernet

GTM Global Time Message

IETF Internet Engineering Task Force

IFG Inter Frame Gap

IoT Internet of Things

IP Internet Protocol

ID Intrusion Detection

IDMEF Intrusion Detection Message Exchange Format

IDS Intrusion Detection System

MAC Message Authentication Code

MD Misuse Detection

MITM Man-In-The-Middle

MuSigs Misuse Signatures

MVM Membership Vector Message

NID Network Intrusion Detection

NIDS Network Intrusion Detection System

OS Operating System

PCF Protocol Control Frame

PTP Precision Time Protocol

RC Rate-Constrained

SC Synchronisation Client

SCSA Secure Clock Synchronization Algorithm

SM Synchronisation Master

STATL State Transition Analysis Technique Language

TAA Trusted Authentication Authority

TC Transparent Clock

TDMA Time Division Multiple Access

TSN Time Sensitive Network

TT Time-Triggered

TTE Time-Triggered-Ethernet

VID Virtual Link ID

VL ID Virtual Link Identifier

XML Extensible Markup Language

A. Anhang

A.1. Synchronisation

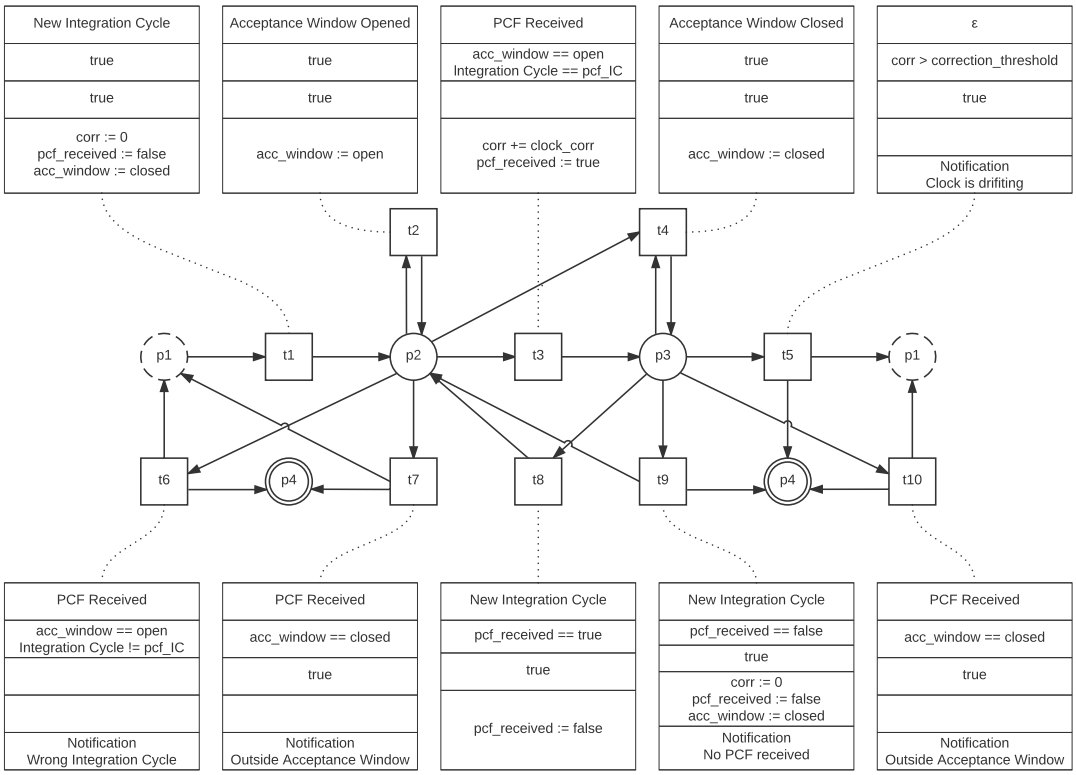

Abbildung A.1.: Signaturnetz zur Erkennung von Angriffen auf die Synchronisation

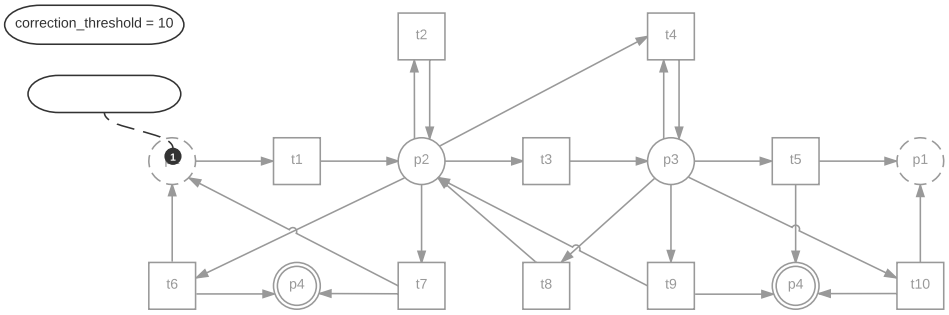

Abbildung A.2.: Signaturnetz Synchronisation mit Token

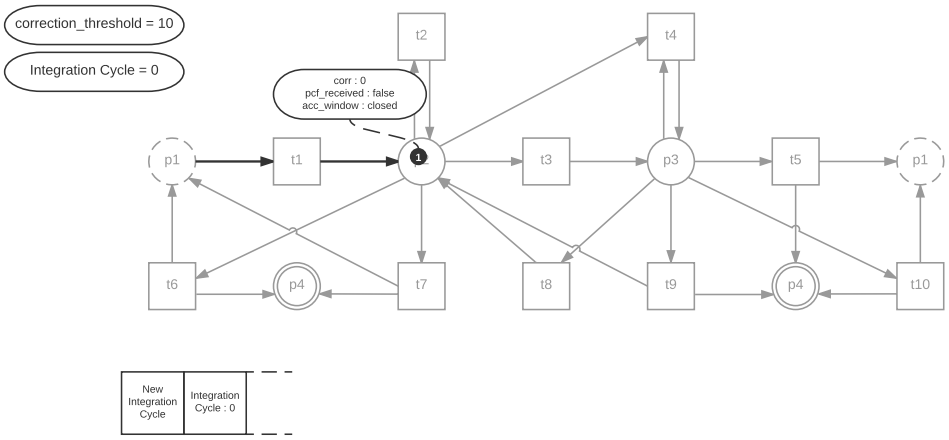

Abbildung A.3.: Signaturnetz Synchronisation mit Token - Neuer Integration Cycle

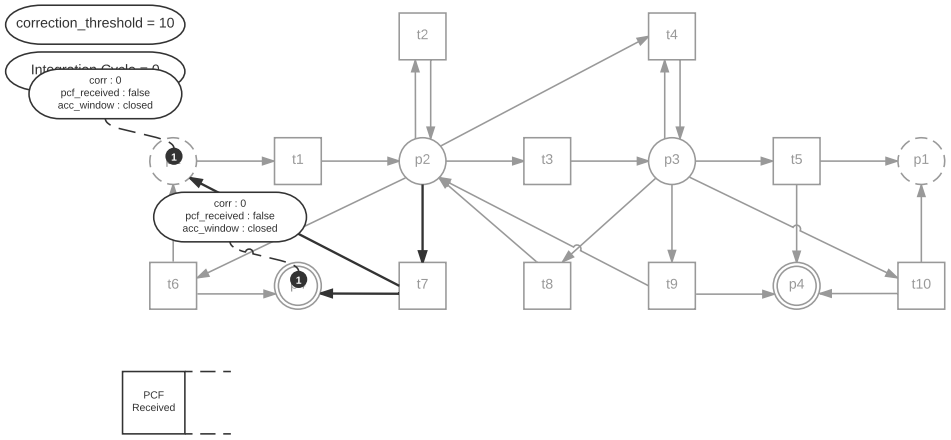

Abbildung A.4.: Signaturnetz Synchronisation mit Token - PCF außerhalb des Acceptance Window empfangen

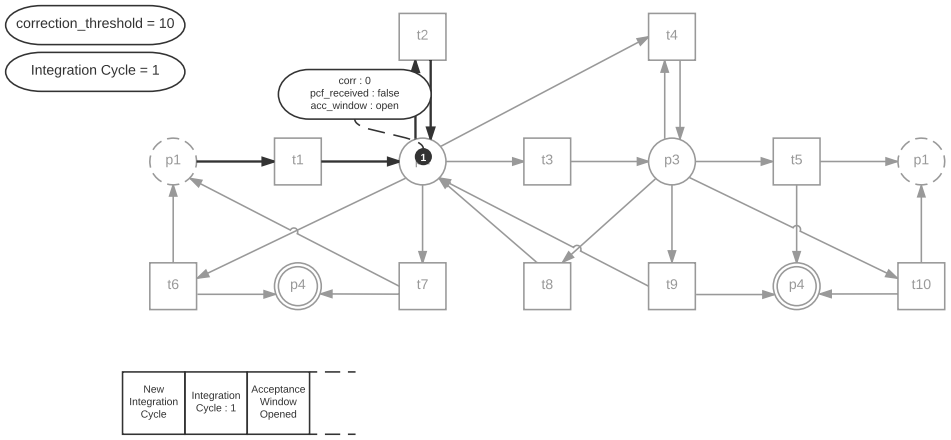

Abbildung A.5.: Signaturnetz Synchronisation mit Token - Acceptance Window geöffnet

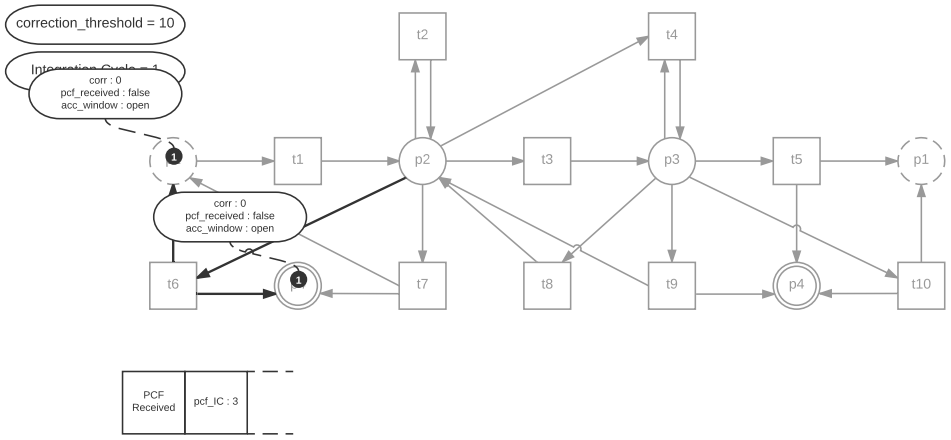

Abbildung A.6.: Signaturnetz Synchronisation mit Token - PCF mit falschem Integration Cycle empfangen

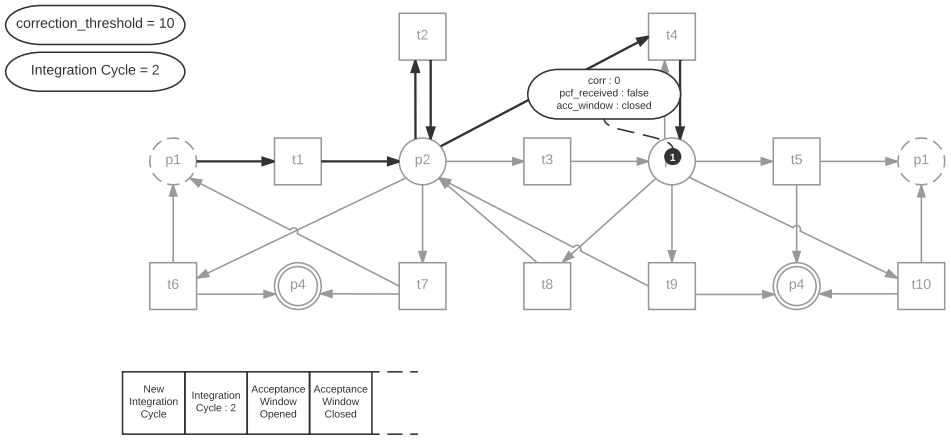

Abbildung A.7.: Signaturnetz Synchronisation mit Token - Kein PCF innerhalb des Acceptance Window empfangen

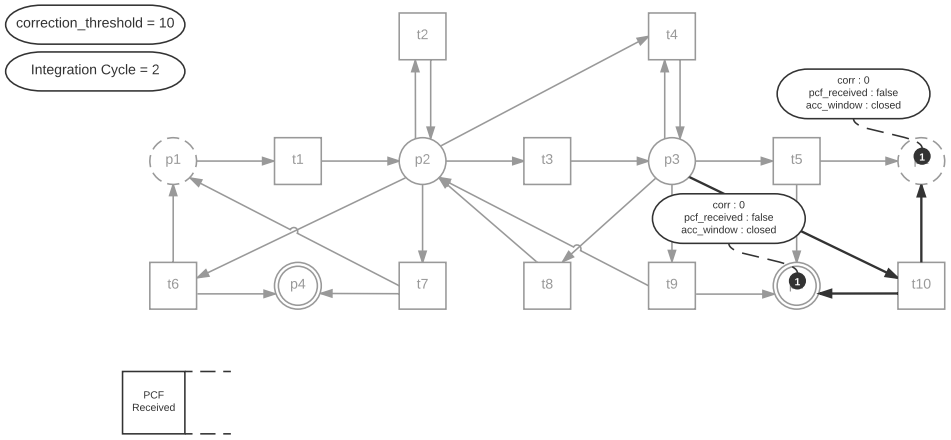

Abbildung A.8.: Signaturnetz Synchronisation mit Token - PCF außerhalb des Acceptance Window empfangen

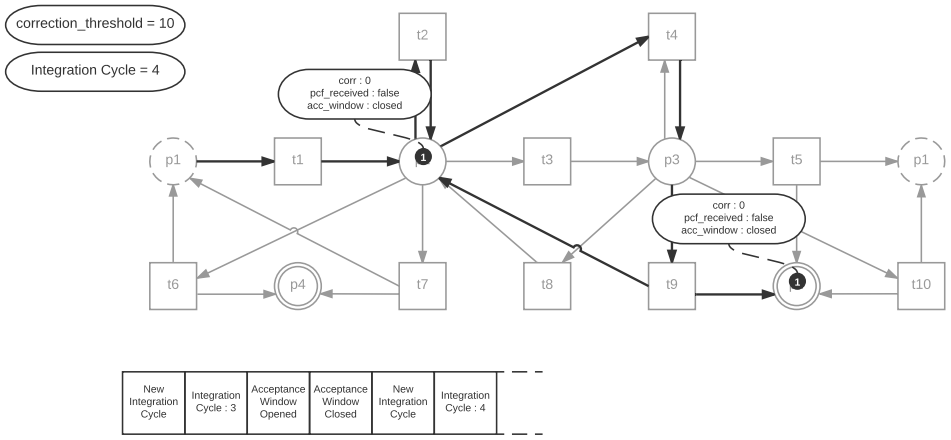

Abbildung A.9.: Signaturnetz Synchronisation mit Token - Kein PCF innerhalb des Acceptance Window empfangen

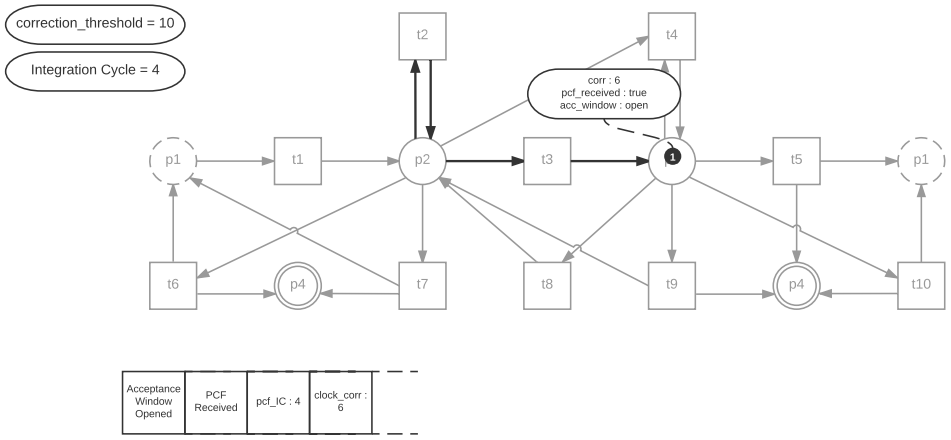

Abbildung A.10.: Signaturnetz Synchronisation mit Token - PCF innerhalb des Acceptance Window empfangen

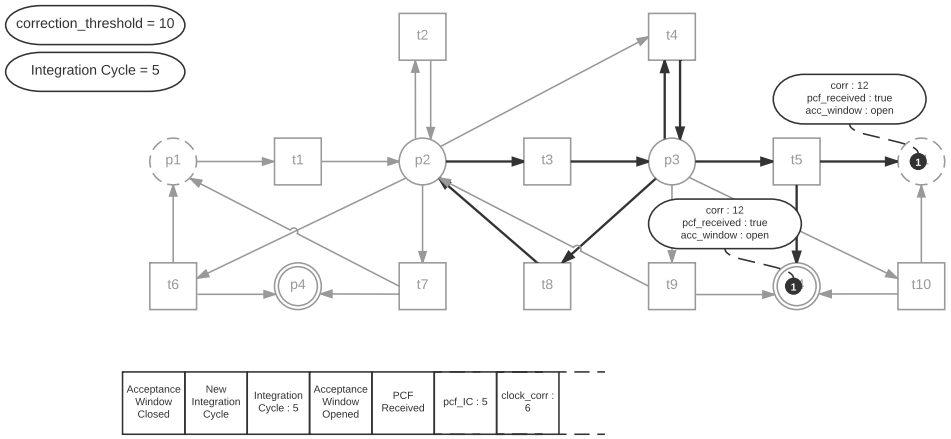

Abbildung A.11.: Signaturnetz Synchronisation mit Token - Correction Threshold erreicht

A.2. Time-Triggered-Nachrichten

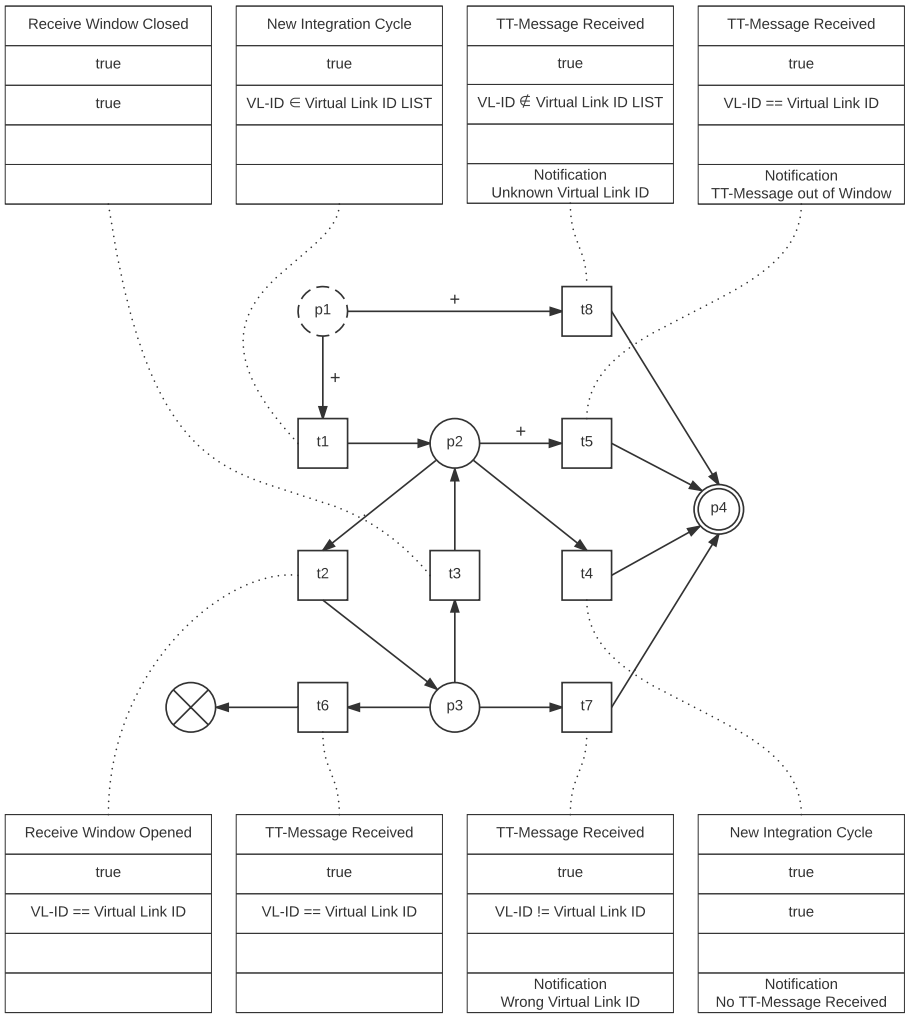

Abbildung A.12.: Signaturnetz zur Erkennung von Angriffen auf TT-Nachrichten

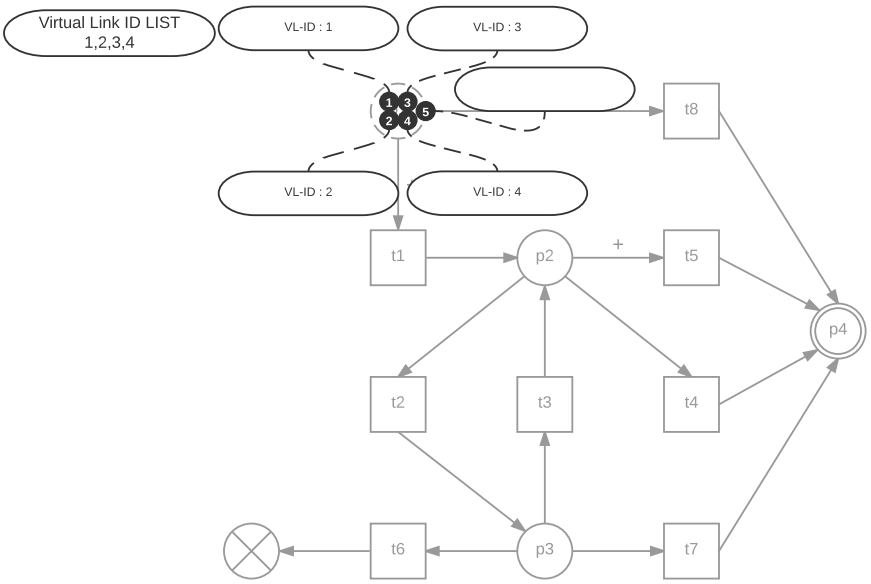

Abbildung A.13.: Signaturnetz zur Erkennung von Angriffen auf TT-Nachrichten mit Token

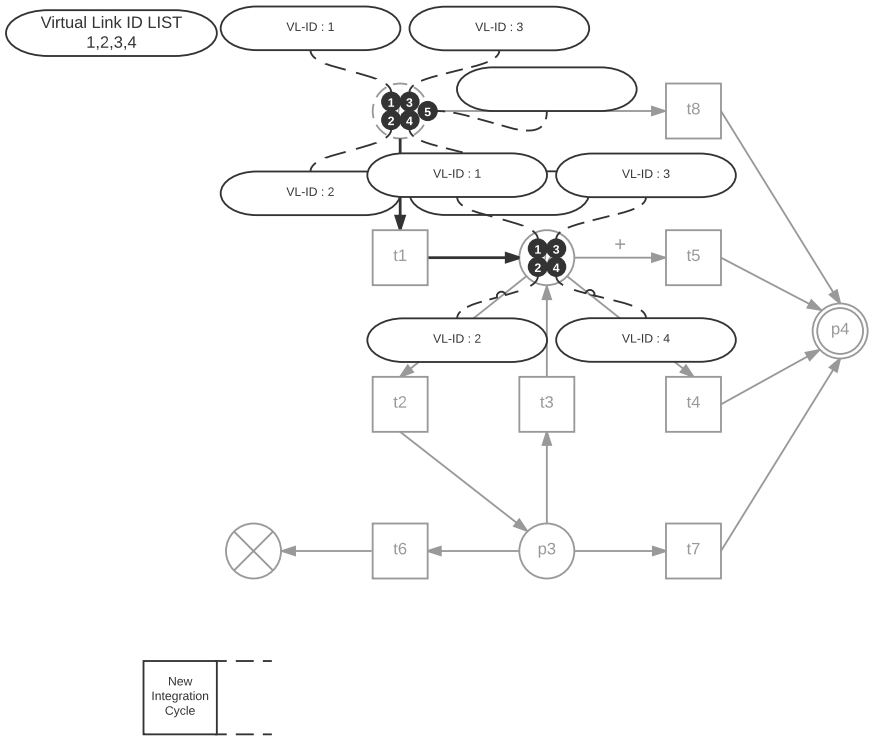

Abbildung A.14.: Signaturnetz zur Erkennung von Angriffen auf TT-Nachrichten mit Token - Neuer Integration Cycle

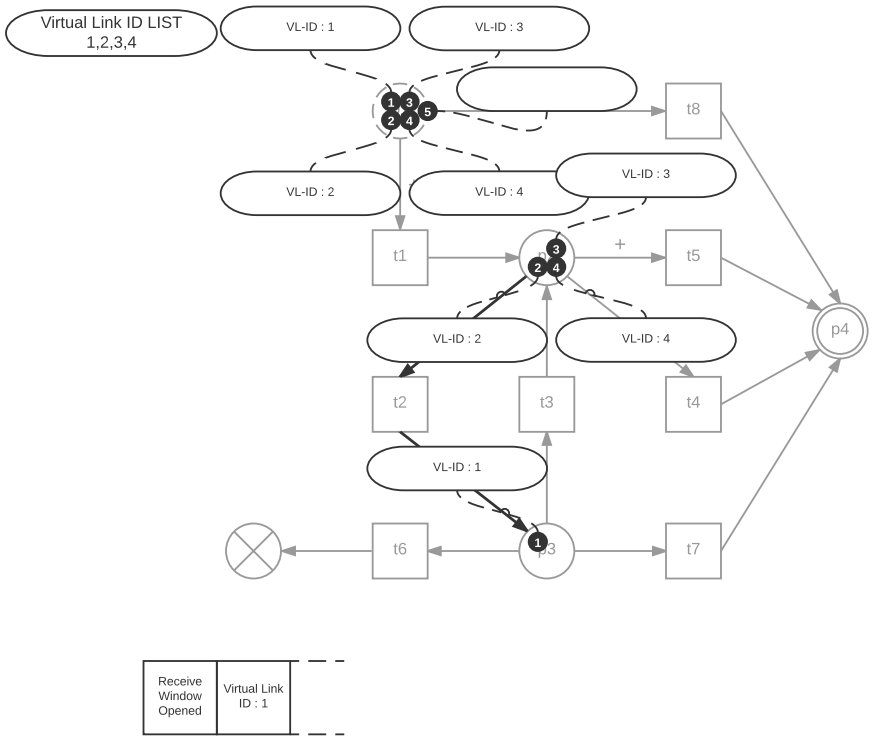

Abbildung A.15.: Signaturnetz zur Erkennung von Angriffen auf TT-Nachrichten mit Token - Receive Window geöffnet (VL-ID 1)

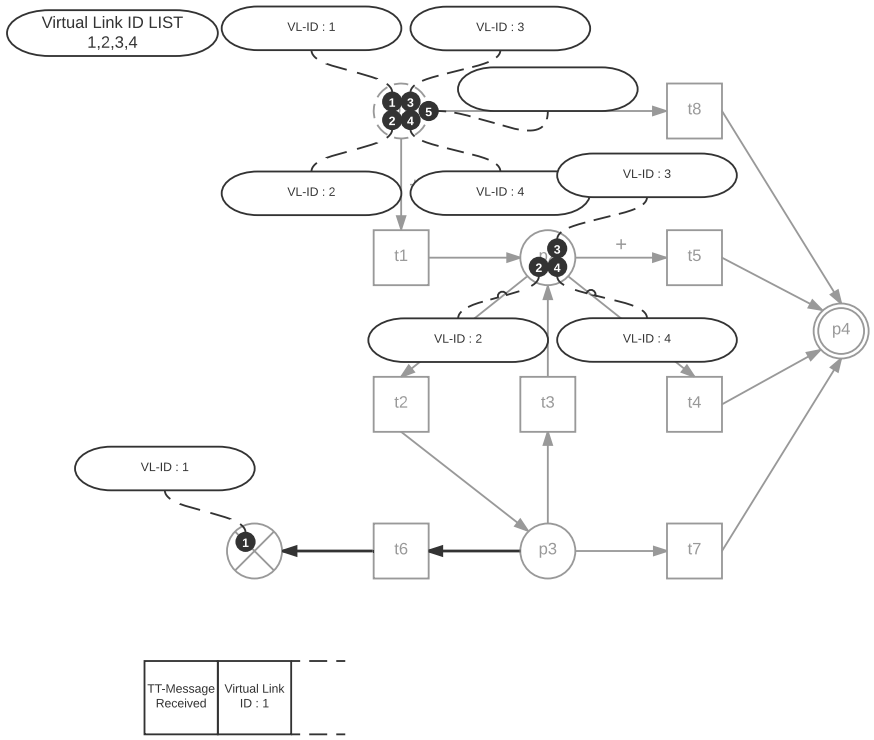

Abbildung A.16.: Signaturnetz zur Erkennung von Angriffen auf TT-Nachrichten mit Token - TT-Nachricht empfangen

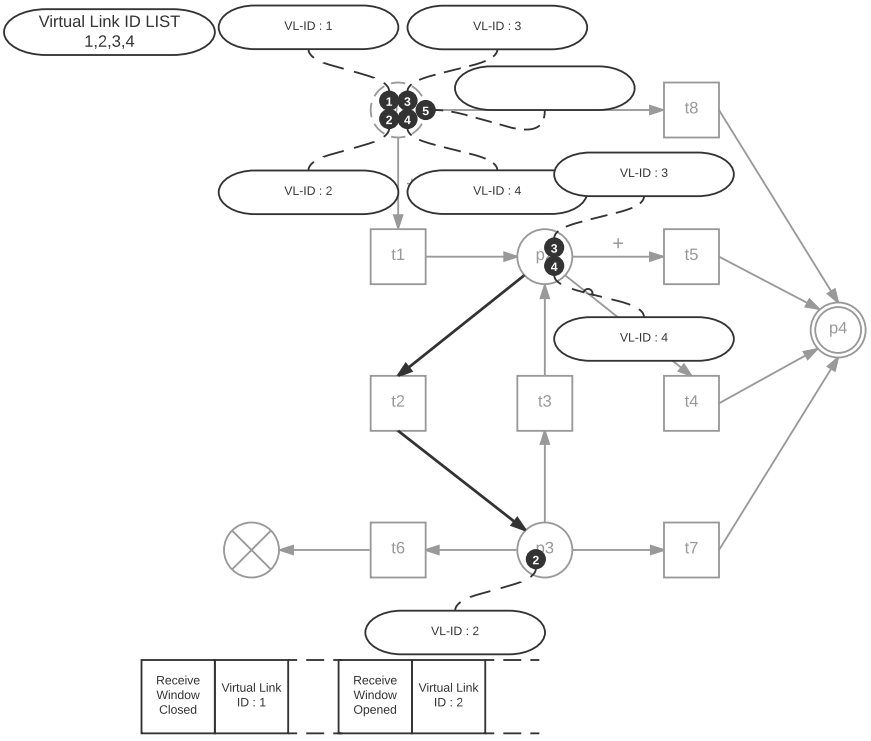

Abbildung A.17.: Signaturnetz zur Erkennung von Angriffen auf TT-Nachrichten mit Token - Receive Window geöffnet (VL-ID 2)

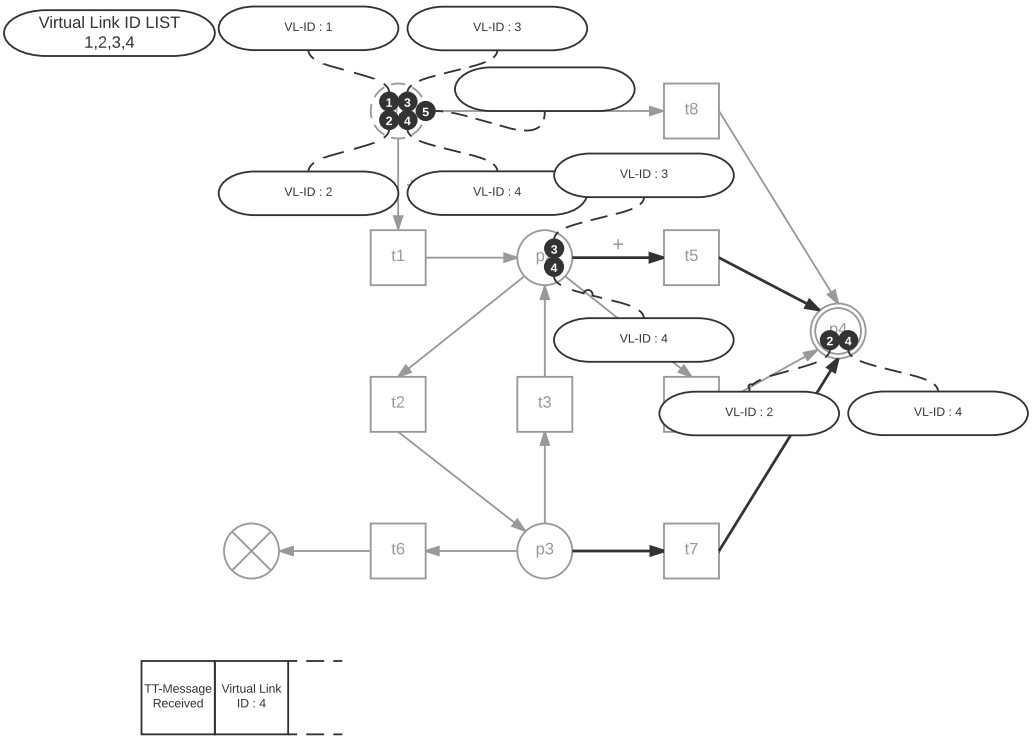

Abbildung A.18.: Signaturnetz zur Erkennung von Angriffen auf TT-Nachrichten mit Token - TT-Nachricht mit falscher VL-ID empfangen

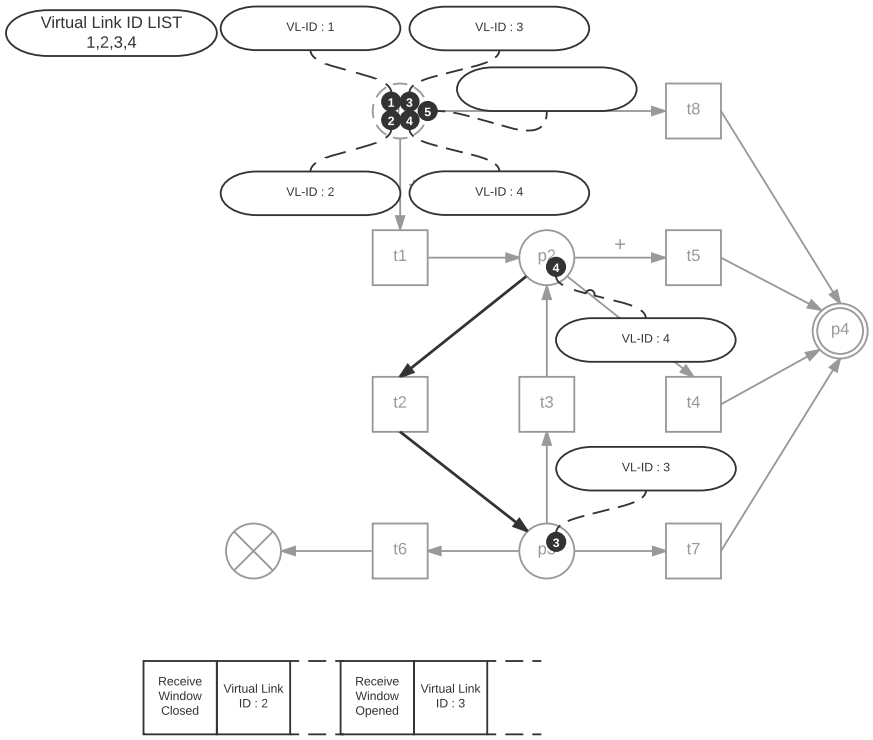

Abbildung A.19.: Signaturnetz zur Erkennung von Angriffen auf TT-Nachrichten mit Token - Receive Window geöffnet (VL-ID 3)

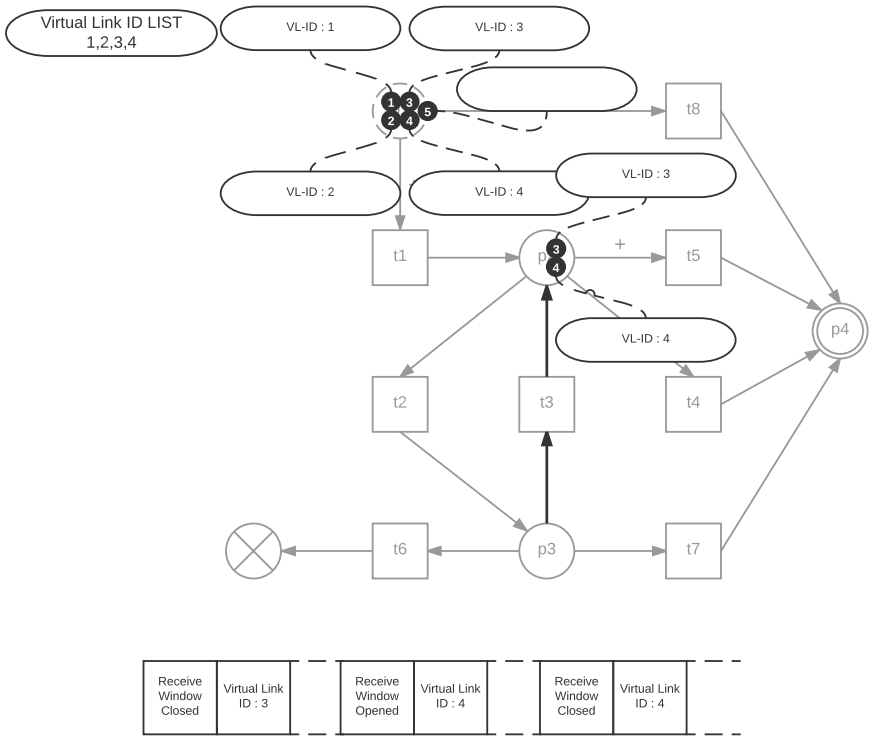

Abbildung A.20.: Signaturnetz zur Erkennung von Angriffen auf TT-Nachrichten mit Token - Keine Nachricht im Receive Window empfangen

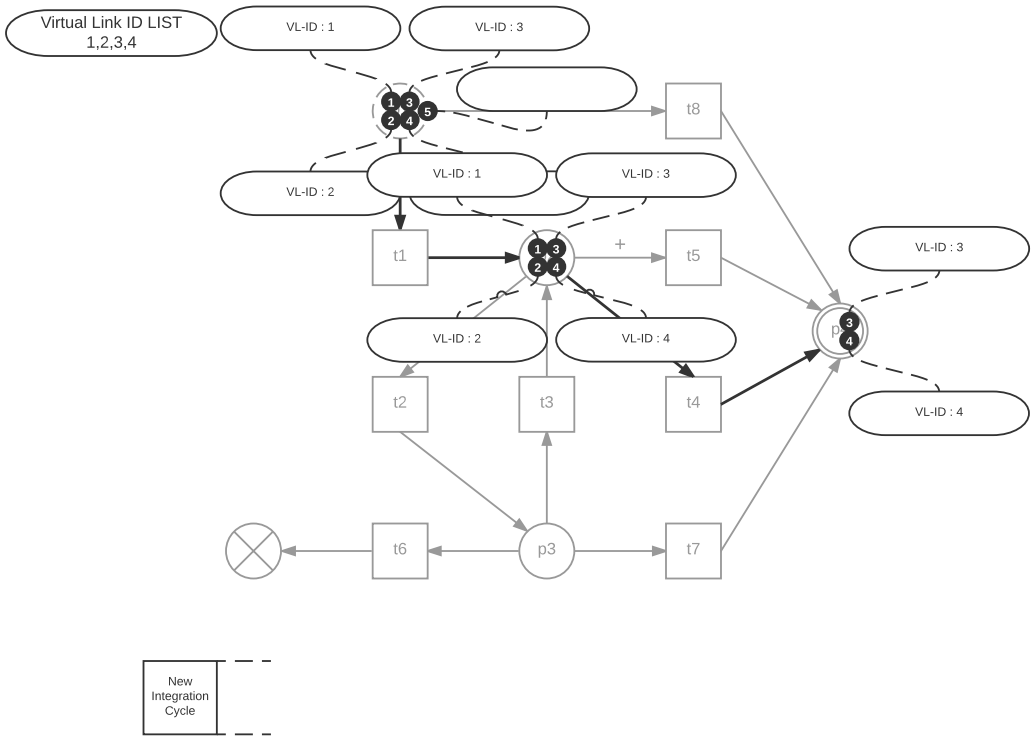

Abbildung A.21.: Signaturnetz zur Erkennung von Angriffen auf TT-Nachrichten mit Token - Neuer Integration Cycle

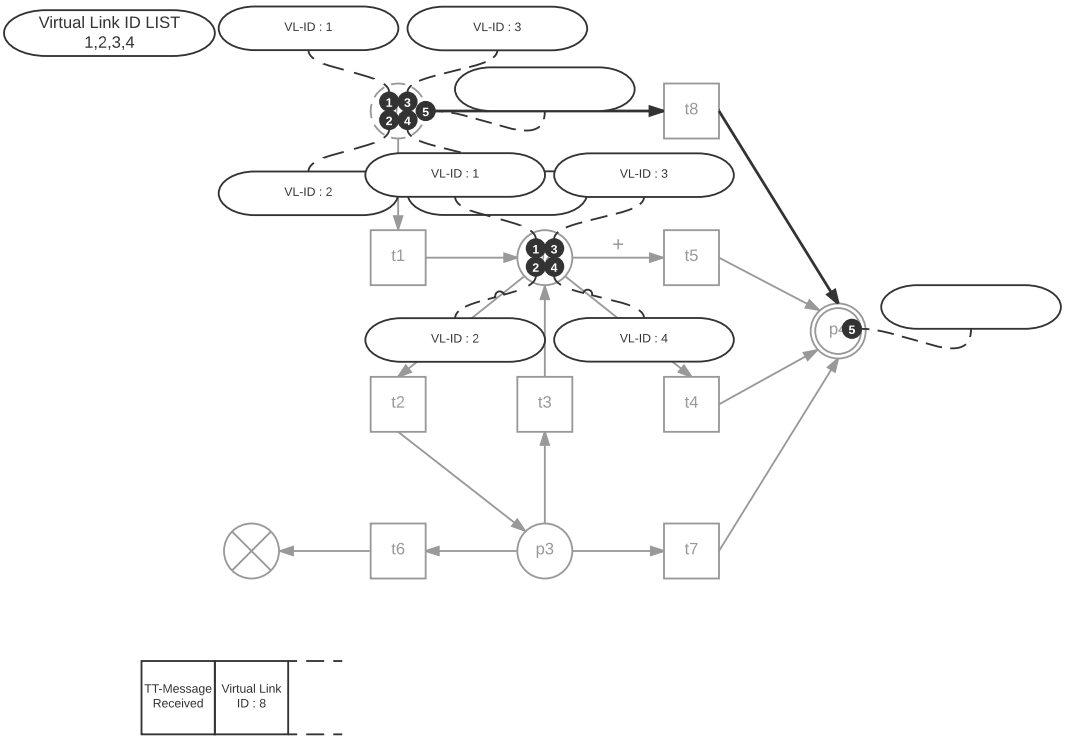

Abbildung A.22.: Signaturnetz zur Erkennung von Angriffen auf TT-Nachrichten mit Token - Unbekannte TT-Nachricht empfangen

A.3. Rate-Constrained-Nachrichten

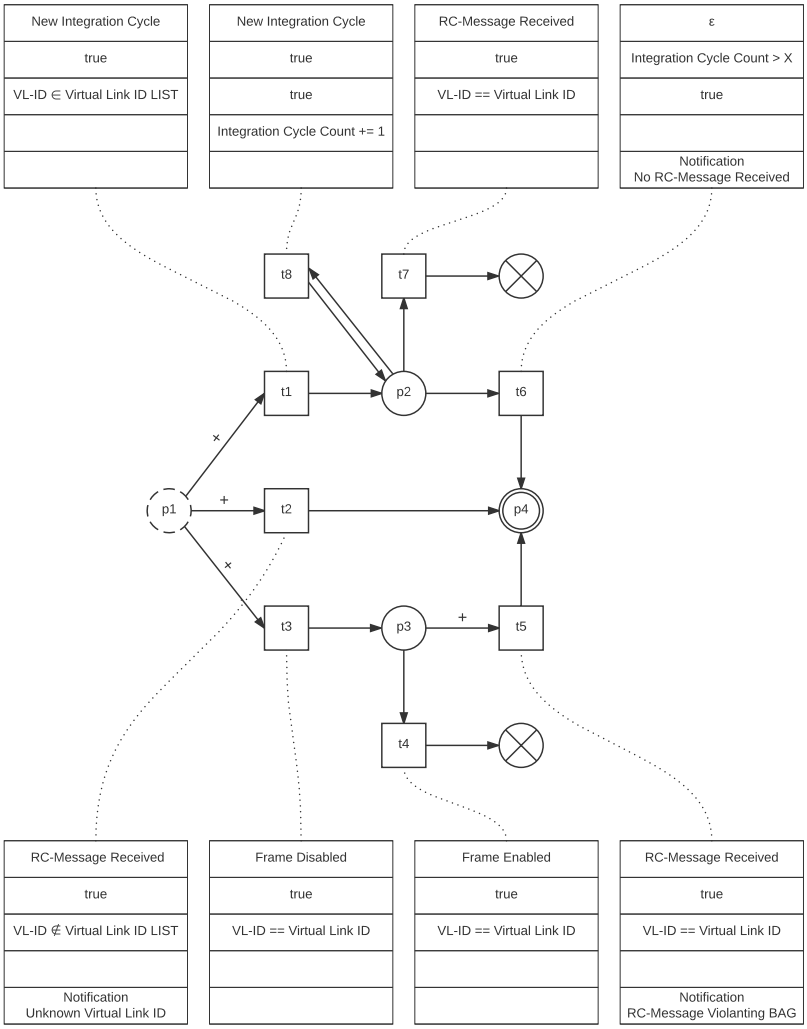

Abbildung A.23.: Signaturnetz zur Erkennung von Angriffen auf RC-Nachrichten

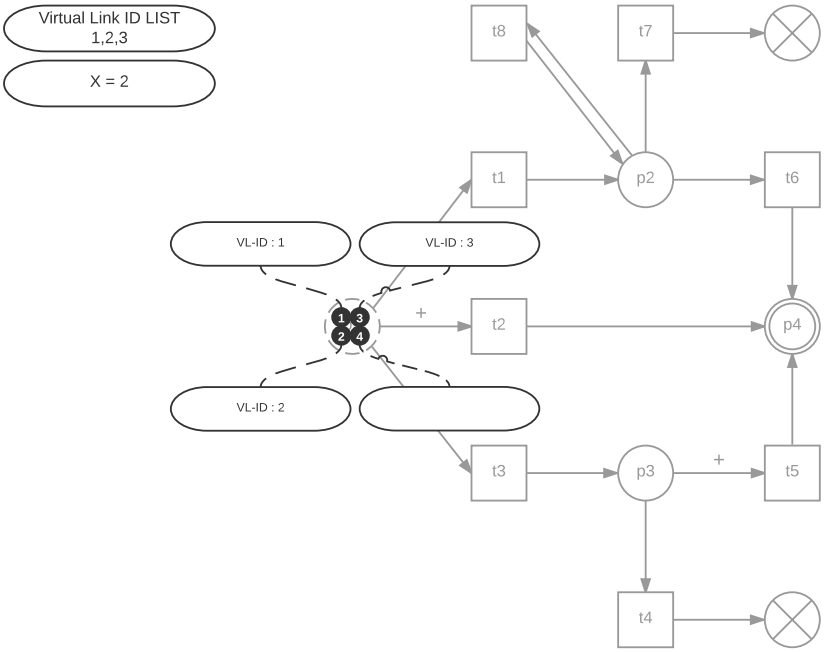

Abbildung A.24.: Signaturnetz zur Erkennung von Angriffen auf RC-Nachrichten mit Token

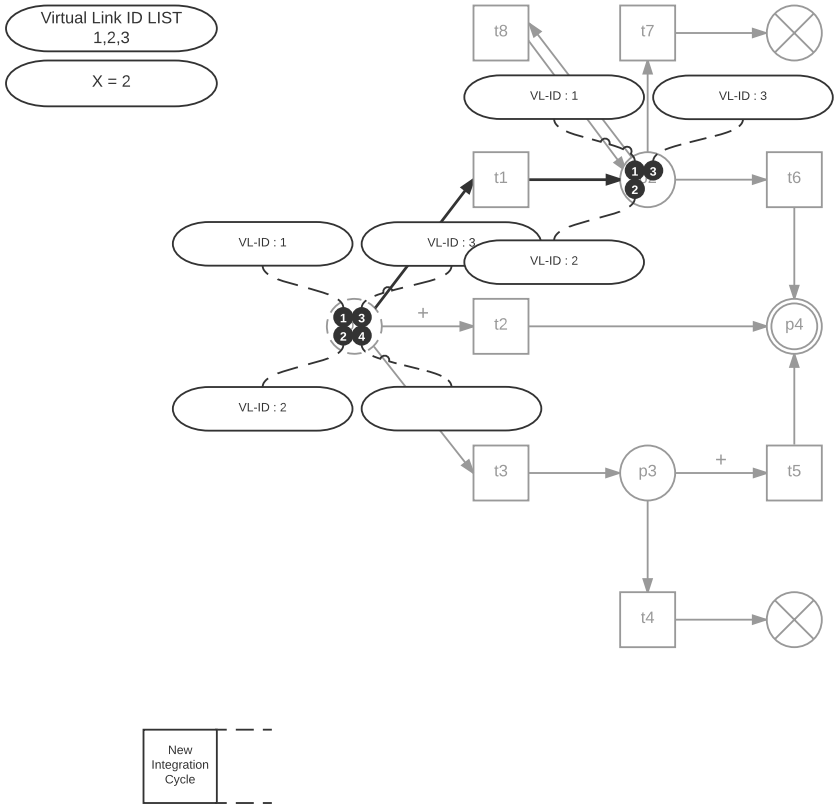

Abbildung A.25.: Signaturnetz zur Erkennung von Angriffen auf RC-Nachrichten mit Token - Neuer Integration Cycle

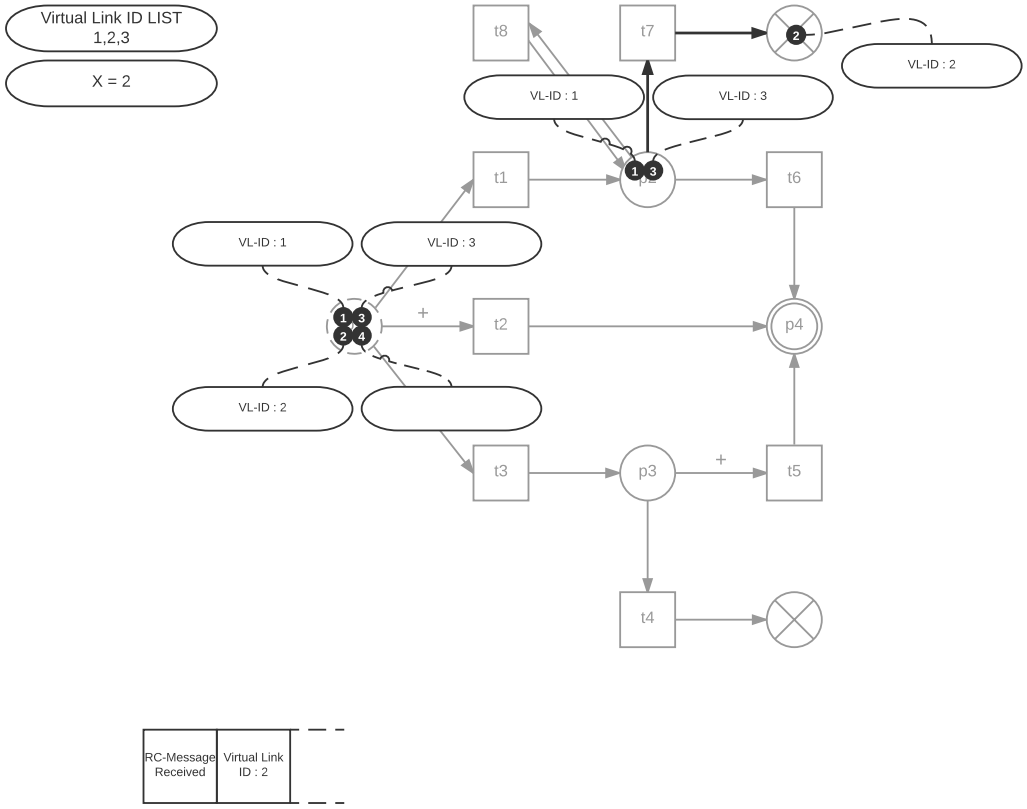

Abbildung A.26.: Signaturnetz zur Erkennung von Angriffen auf RC-Nachrichten mit Token - RC-Nachricht empfangen

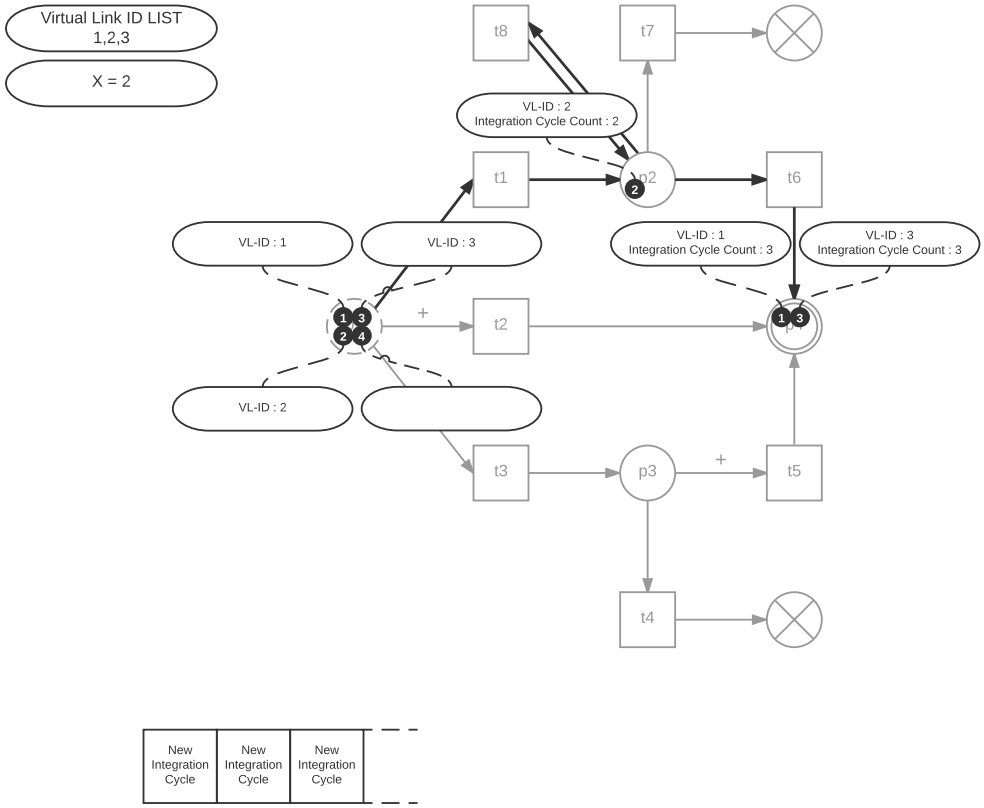

Abbildung A.27.: Signaturnetz zur Erkennung von Angriffen auf RC-Nachrichten mit Token - Keine RC-Nachrichten in einem Intervall (3 Integration Cycles) empfangen

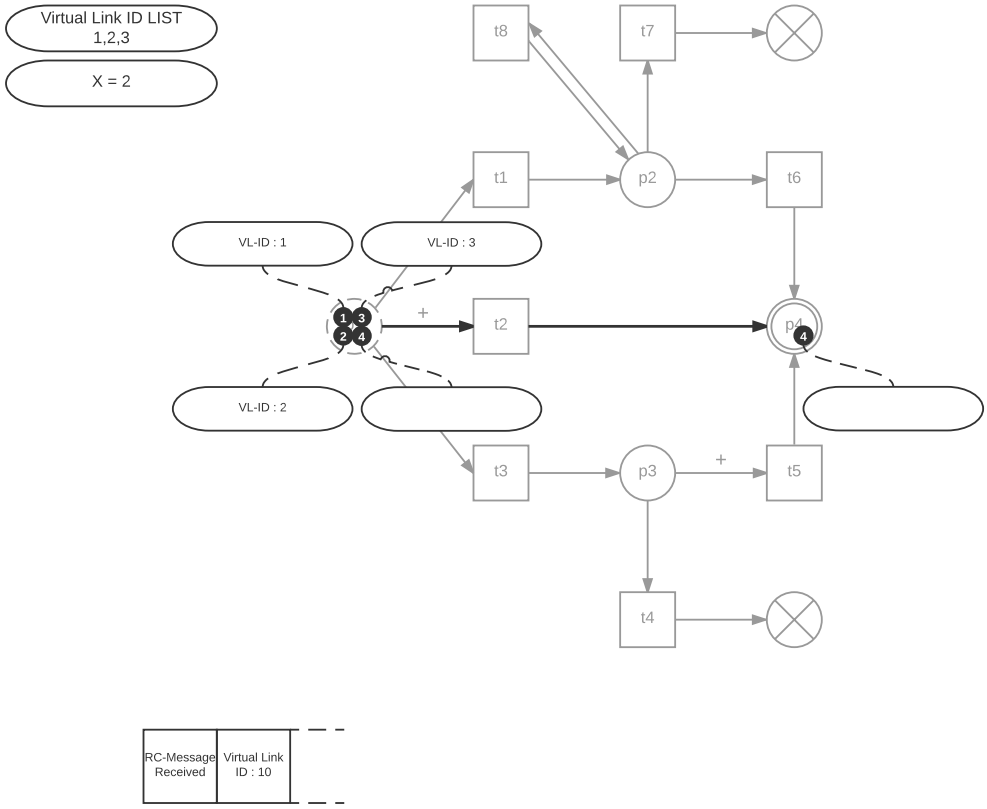

Abbildung A.28.: Signaturnetz zur Erkennung von Angriffen auf RC-Nachrichten mit Token - Unbekannte RC-Nachricht empfangen

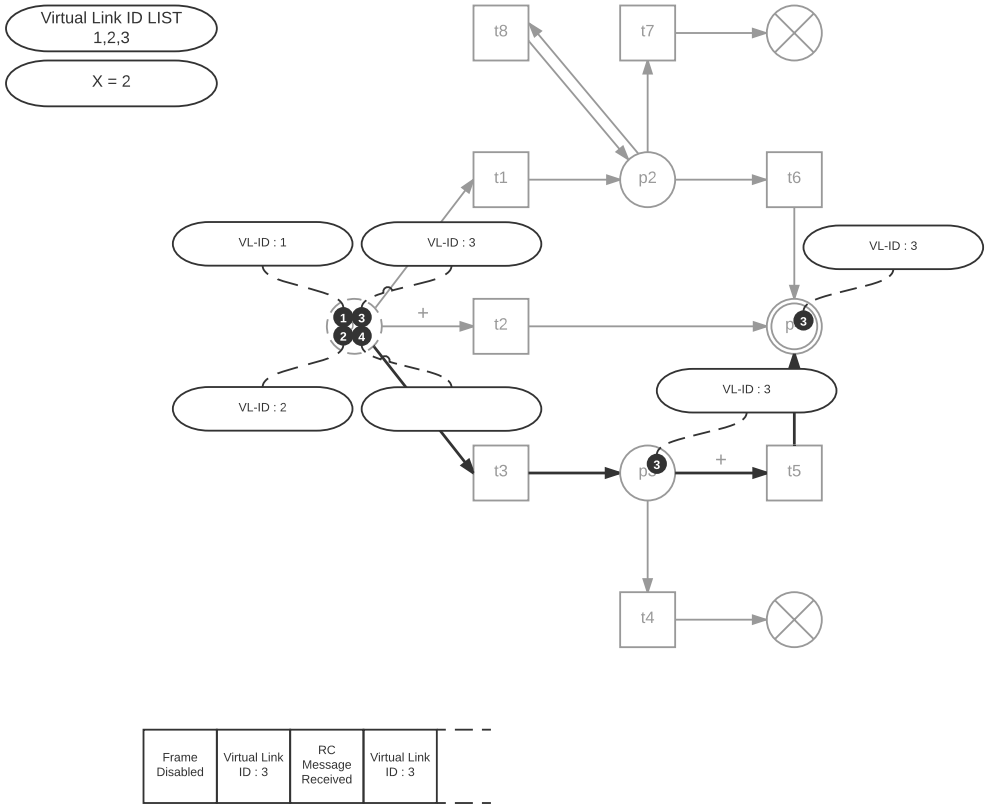

Abbildung A.29.: Signaturnetz zur Erkennung von Angriffen auf RC-Nachrichten mit Token - Gesperrte RC-Nachricht empfangen

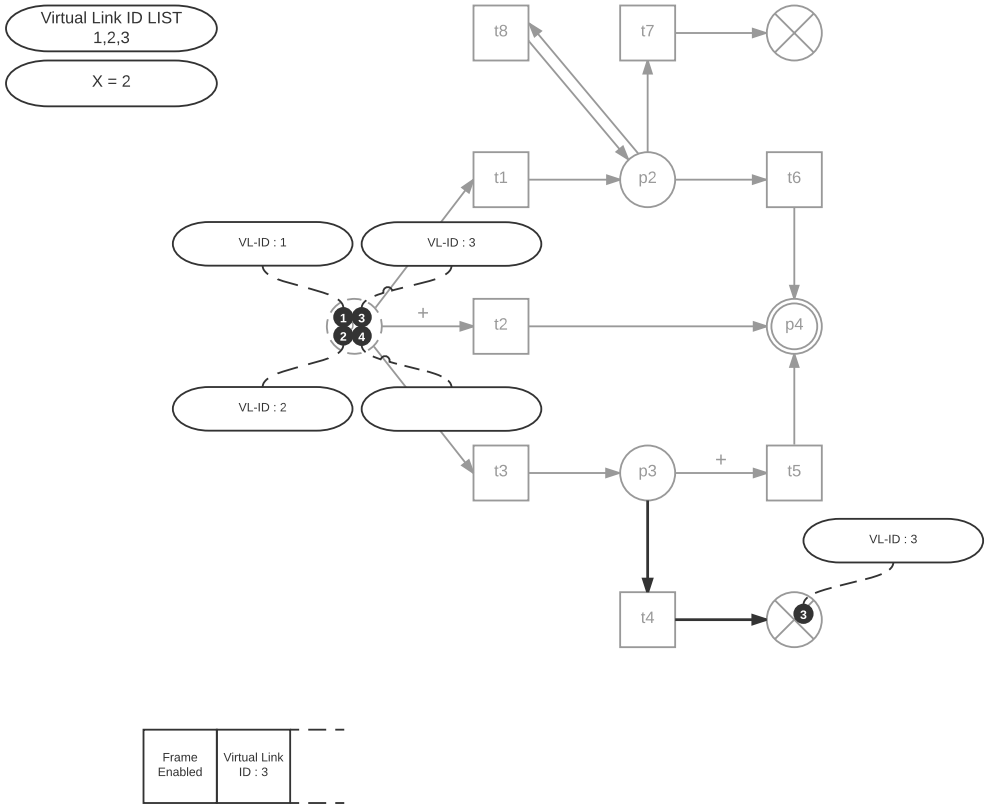

Abbildung A.30.: Signaturnetz zur Erkennung von Angriffen auf RC-Nachrichten mit Token - RC-Nachricht wieder freigegeben

Literaturverzeichnis

[TSNTaskGroup]

[Bace und Mell 2001] Bace, Rebecca ; Mell, Peter: NIST special publication on intrusion detection systems. In: *Nist Special Publication* (2001), S. 1–51. – URL http://oai.dtic.mil/oai/oai?verb=getRecord{&}metadataPrefix=html{&}identifier=ADA393326

[Bishop 1995] Bishop, Matt: A Standard Audit Trail Format. In: *18th National Information Systems Security Conference* (1995), S. 136–145. – URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.8480

[Böck 2017] Böck, Hanno: *Hacker zerstören das Internet of Insecure Things*. 2017. – URL https://www.golem.de/news/brickerbot-hacker-zerstoeren-das-internet-of-insecure-things-1704-127198.html. – Zugriffsdatum: 2017-04-07

[Bundesamt für Sicherheit in der Informationstechnik 2016] Bundesamt für Sicherheit in der Informationstechnik: BSI - Technische Richtlinie / Bundesamt für Sicherheit in Der Informationstechnik. URL https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf, 2016. – Forschungsbericht. – 1–78 S

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

[Cheng u. a. 2012] Cheng, Tsung H. ; Lin, Ying D. ; Lai, Yuan C. ; Lin, Po C.: Evasion techniques: Sneaking through your intrusion detection/prevention systems. In: *IEEE Communications Surveys and Tutorials* 14 (2012), Nr. 4, S. 1011–1020. – ISBN 1553-877X

[Corchado und Herrero 2011] Corchado, Emilio ; Herrero, Álvaro: Neural visualization of network traffic data for intrusion detection. In: *Applied Soft Computing Journal* 11 (2011), Nr. 2, S. 2042–2056. – URL http://dx.doi.org/10.1016/j.asoc.2010.07.002. – ISBN 1568-4946

[Debar und Morin 2002] Debar, Hervé ; Morin, Benjamin: *Evaluation of the Diagnostic Capabilities of Commercial Intrusion Detection Systems*. S. 177–198. In: Wespi, Andreas (Hrsg.) ; Vigna, Giovanni (Hrsg.) ; Deri, Luca (Hrsg.): *Recent Advances in Intrusion Detection: 5th International Symposium, RAID 2002 Zurich, Switzerland, October 16–18, 2002 Proceedings*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2002. – URL http://dx.doi.org/10.1007/3-540-36084-0{_}10. – ISBN 978-3-540-36084-1

[dragTimes 2014] dragTimes: *Tesla Model S Ethernet Network Explored, Possible Jailbreak in the Future?* 2014. – URL http://www.dragtimes.com/blog/tesla-model-s-ethernet-network-explored-possible-jailbreak-in-the-future. – Zugriffsdatum: 2016-11-24

[Feinstein u. a. 2007] Feinstein, Benjamin ; Curry, David ; Debar, Herve: *The Intrusion Detection Message Exchange Format (IDMEF)*. 2007. – URL https://rfc-editor.org/rfc/rfc4765.txt

[Gan u. a. 2011] Gan, Gang ; Lu, Zeyong ; Jiang, Jun: Internet of Things Security Analysis. In: *2011 International Conference on Internet Technology and Applications* (2011), S. 1–4. ISBN 978-1-4244-7254-3

[Henniger u. a. 2009] Henniger, Olaf ; Apvrille, Ludovic ; Fuchs, Andreas ; Roudier, Yves ; Ruddle, Alastair ; Weyl, Benjamin ; Paristech, Telecom ; Ltci, Cnrs ; Antipolis, Sophia: Security requirements for automotive on-board networks. (2009), S. 641–646. ISBN 9781424453474

[Hirschler und Treytl 2011] Hirschler, Bernd ; Treytl, Albert: Validation and verification of IEEE 1588 Annex K. In: *... (ISPCS), 2011 International IEEE ...* (2011), Nr. Icv. – URL http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=6070156. ISBN 9781612848938

[Isakovic 2011a] Isakovic, Haris: *A secure global time base for time triggered systems*, Technische Universität Wien, Institut für Technische Informatik, Dissertation, 2011

[Isakovic 2011b] Isakovic, Haris: *A secure global time base for time triggered systems*, Technische Universität Wien, Institut für Technische Informatik, Dissertation, 2011

[Koscher u. a. 2010] Koscher, Karl ; Czeskis, Alexei ; Roesner, Franziska ; Patel, Shwetak ; Kohno, Tadayoshi ; Checkoway, Stephen ; McCoy, Damon ; Kantor, Brian ; Anderson, Danny ; Shacham, Hovav ; Savage, Stefan: Experimental Security Analysis of a Modern Automobile. In: *2010 IEEE Symposium on Security and Privacy* (2010), S. 447–462. – URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5504804. ISBN 978-1-4244-6894-2

[Kumar und Spafford 1994] Kumar, Sandeep ; Spafford, Eugene H E.: An application of pattern matching in intrusion detection. URL http://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=2115{&}context=cstech, 1994. – Forschungsbericht. – 1–55 S

[Liao u. a. 2013] Liao, Hung-Jen ; Richard Lin, Chun-Hung ; Lin, Ying-Chih ; Tung, Kuang-Yuan: Intrusion detection system: A comprehensive review. In: *Journal of Network and Computer Applications* 36 (2013), Nr. 1, S. 16–24. – URL http://www.sciencedirect.com/science/article/pii/S1084804512001944. – ISBN 1084-8045

[McCarthy und Dayal 1989] McCarthy, Dennis ; Dayal, Umeshwar: The architecture of an active database management system. In: *Proceedings of the 1989 ACM SIGMOD international conference on Management of data - SIGMOD '89* (1989), S. 215–224. – URL http://doi.acm.org/10.1145/67544.66946{%}5Cnhttp://portal.acm.org/citation.cfm?doid=67544.66946. – ISBN 0897913175

[Meier 2008] Meier, M: *Intrusion Detection effektiv - Modellierung und Analyse von Angriffsmustern.* 2008. – ISBN 9783540482512

[Ning u. a. 2001] Ning, Peng ; Jajodia, Sushil ; Wang, Xiaoyang S.: Abstraction-based intrusion detection in distributed environments. In: *ACM Transactions on Information and System Security* 4 (2001), Nr. 4, S. 407–452. – URL http://portal.acm.org/citation.cfm?doid=503339.503342. – ISSN 10949224

[Önal und Kirrmann 2012] Önal, Cagri ; Kirrmann, Hubert: Security improvements for IEEE 1588 Annex K. (2012), S. 1–6. ISBN 9781457717161

[Parekh u. a. 2006] Parekh, Janak J. ; Wang, Ke ; Stolfo, Salvatore J.: Privacy-preserving payload-based correlation for accurate malicious traffic detection. In: *Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense - LSAD '06* (2006), S. 99–106. – URL http://portal.acm.org/citation.cfm?doid=1162666.1162667. ISBN 1595935711

[Phieler 2016] Phieler, Stephan: Ein Konzept für eine leichtgewichtige lokale Public Key Infrastruktur für den Einsatz im Bordnetz. 2016. – Forschungsbericht

[Rittinghouse und Hancock 2003] Rittinghouse, John W. ; Hancock, Bill: *Cybersecurity operations handbook*. Elsevier Digital Press, 2003. – 1287 S. – ISBN 9781555583064

[Roesch 1999] Roesch, M: Snort: Lightweight Intrusion Detection for Networks. In: *LISA '99: 13th Systems Administration Conference* (1999), S. 229–238. – URL http://static.usenix.org/publications/library/proceedings/lisa99/full{_}papers/roesch/roesch.pdf. ISBN 1-880446-25-1

[SAE - AS-2D Time Triggered Systems and Architecture Committee 2009] SAE - AS-2D Time Triggered Systems and Architecture Committee: *Time-Triggered Ethernet (AS 6802)*. 2009. – URL http://standards.sae.org/as6802/

[Sanok 2005] Sanok, Daniel J.: An analysis of how antivirus methodologies are utilized in protecting computers from malicious code. In: *Proceedings of the 2nd annual conference on Information security curriculum development - InfoSecCD '05* (2005), S. 142. – URL http://portal.acm.org/citation.cfm?doid=1107622.1107655. ISBN 1595932615

[Stavroulakis und Stamp 2010] Stavroulakis, Peter ; Stamp, Mark ; Stavroulakis, Peter (Hrsg.) ; Stamp, Mark (Hrsg.): *Handbook of Information and Communication Security*. 1. Berlin : Springer-Verlag Berlin Heidelberg, 2010. – 867 S. – ISBN 978-3-642-04116-7

[Steiner 2013] Steiner, Wilfried: Candidate security solutions for TTEthernet. In: *Digital Avionics Systems Conference (DASC), 2013 ...* (2013), S. 1–10. – URL http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=6712579. ISBN 9781479915385

[Studnia u. a. 2013a] Studnia, Ivan ; Nicomette, Vincent ; Alata, Eric ; Deswarte, Yves: Security of embedded automotive networks : state of the art and a research proposal. (2013)

[Studnia u. a. 2013b] Studnia, Ivan ; Nicomette, Vincent ; Alata, Eric ; Deswarte, Yves ; Kaâniche, Mohamed ; Laarouchi, Youssef ; Toulouse, F: Survey on Security Threats and Protection Mechanisms in Embedded Automotive Networks. (2013)

[Tillich und Wójcik 2012] Tillich, Stefan ; Wójcik, M: Security analysis of an open car immobilizer protocol stack. In: *Trusted Systems* 3 (2012). – URL http://link.springer.com/chapter/10.1007/978-3-642-35371-0{_}8

[Treytl und Hirschler 2009] Treytl, Albert ; Hirschler, Bernd: Security flaws and workarounds for IEEE 1588 (transparent) clocks. In: *2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication* 1588 (2009), oct, S. 1–6. – URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5340204. ISBN 978-1-4244-4391-8

[Vasiliadis u. a. 2008] Vasiliadis, Giorgos ; Antonatos, Spiros ; Polychronakis, Michalis ; Markatos, E. ; Ioannidis, S.: Gnort: High performance network intrusion detection using graphics processors. In: *Recent Advances in Intrusion Detection* (2008), S. 116–134. – URL http://www.springerlink.com/index/g2w54q11130r7126.pdf. ISBN 9783540874027

[Vigna u. a. 2000a] Vigna, G. ; Eckmann, S. T. ; Kemmerer, R. A.: The STAT tool suite. In: *Proceedings - DARPA Information Survivability Conference and Exposition, DISCEX 2000* 2 (2000), S. 46–55. ISBN 0769504906

[Vigna u. a. 2000b] Vigna, Giovanni ; Ekmann, Steven ; Kemmerer, Richard: Attack Languags. In: *In Proceedings of the IEEE Information Survivability Workshop* (2000)

[Wang u. a. 2006] Wang, Ke ; Cretu, Gabriela ; Stolfo, Salvatore J.: Anomalous Payload-Based Worm Detection and Signature Generation. In: *Recent Advances in Intrusion Detection* 3858 (2006), S. 227–246. – URL http://link.springer.com/10.1007/11663812{_}12. – ISBN 3540317783

[Wang und Stolfo 2004] Wang, Ke ; Stolfo, Salvatore J.: Anomalous Payload-Based Network Intrusion Detection. In: *Recent Advances in Intrusion Detection (RAID)* (2004), S. 203–222. – URL http://link.springer.com/chapter/10.1007/978-3-540-30143-1{_}11{%}0Ahttp://link.springer.com/10.1007/978-3-540-30143-1{_}11. – ISBN 978-3-540-23123-3

[Wasicek 2011] Wasicek, A: Security in time-triggered systems. 2011 (2011). – URL http://www.informatik.tuwien.ac.at/dekanat/Kurzfassung-Wasicek.pdf

[Wasicek u. a. 2011] Wasicek, Armin ; El-Salloum, Christian ; Kopetz, Hermann: Authentication in Time-Triggered Systems Using Time-Delayed Release of Keys. In: *2011 14th IEEE*

International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (2011), mar, S. 31–39. – URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5753589. ISBN 978-1-61284-433-6

[Wolf 2009] Wolf, Marko: Mehr Sicherheit auf unseren Straßen. (2009), S. 1–8

[Wu und Banzhaf 2010] Wu, Shelly X. ; Banzhaf, Wolfgang: The use of computational intelligence in intrusion detection systems: A review. In: *Applied Soft Computing* 10 (2010), Nr. 1, S. 1–35. – URL http://www.sciencedirect.com/science/article/pii/S1568494609000908. – ISBN 1568-4946

[Zhang und Lee 2000] Zhang, Yongguang ; Lee, Wenke: Intrusion detection in wireless ad-hoc networks. In: *Proceedings of the 6th annual international conference on Mobile computing and networking - MobiCom '00* (2000), S. 275–283. – URL http://portal.acm.org/citation.cfm?doid=345910.345958. – ISBN 1581131976

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 5. Mai 2017 Stephan Phieler