Hochschule Wismar
University of Applied Sciences
Technology, Business and Design
Fakultät für Ingenieurwissenschaften, Bereich EuI

Master-Thesis

Unidirektionale Datenverbindungen zwischen virtualisierten Gastsystemen

Eingereicht am: 25. September 2013

von: Jan Klemkow
geboren am 09.01.1988
in Crivitz

Betreuer: Prof. Dr.-Ing. Ernst Jonas
Hochschule Wismar

Dr. Christian Ehrhardt
genua mbh

Inhaltsverzeichnis

1 Einleitung **1**

2 Problemstellung und Anforderungen **3**

2.1 Anwendungsfälle von Datendioden 3

2.2 Rot-Schwarz-Terminologie 5

2.3 Anforderungen an Datendioden 5

3 Grundlagen **7**

3.1 Physische und Logische Datendioden 7

3.2 Unidirektionale Protokoll-Relays 9

3.3 Analyse bekannter Angriffsvektoren auf unidirektionale Kommunikationskanäle 10

3.3.1 Implementierungsfehler 10

3.3.2 Außen-Szenario 11

3.3.3 Innen-Szenario 11

3.4 Bestehenden Datendioden 12

3.4.1 NRL-Network-Pump 12

3.4.2 genugate-Datendiode 14

3.4.3 genuscreen-Paketdiode 15

3.4.4 SINA-One-Way-Gateway 16

3.4.5 Fort Fox Data Diode 17

3.4.6 Waterfall-Diode 17

3.5 Mechanismus zur Integritätssicherung in Datendioden . . 17

3.5.1 Gehashte Rückmeldung 17

3.5.2 Einseitiges TCP 19

3.5.3 Redundanz und automatische Fehlerkorrektur . . . 20

3.5.4 Bewertung 21

3.6 L4-Mikrokernel 21

3.7 L4OpenBSD 22

3.8 L4-Interprozesskommunikation 22

3.8.1 Messages 23

3.8.2 Shared-Memory 23

3.8.3 Interrupts 23
3.8.4 Bewertung 23
3.9 Unidirektionale Kommunikation auf einem Host-System . 24
3.9.1 Speicherzugriffsrechte 24
3.9.2 Dioden-Prozess 25

4 Analyse bestehender unidirektionaler Kommunikationsmechanismen 27
4.1 Protokoll der NRL-Pump 27
4.2 Protokoll der genugate-Datendioden 28
4.3 Bewertung 30

5 L4-Pipe-Kommunikation 31
5.1 L4-Pipe 31
5.2 Paket-Protokoll der L4-Pipe 32
5.3 TCP-Multiplexer 33
5.4 L4-Pipe-Filter 34
5.5 Schwächen im Unidirektionalen-Einsatz 35

6 Analyse bekannter Rückkanalmöglichkeiten 37
6.1 Impliziter Rückkanal 37
6.2 Answer-Bit 40
6.3 Multi-Bit-Choosing 41
6.4 ACK-Ordering 42
6.5 ACK-Timing 43
6.6 Full-Buffer 44

7 Design eines unidirektionalen Kommunikationsmechanismus 45
7.1 Behandlung von Fehlern während der Übertragung 45
7.1.1 Direkte Rückmeldung 45
7.1.2 Black-Hole-Strategie 46
7.1.3 Response-Polling 46
7.2 NAK-Delay 47
7.3 EOF-Sperre 49
7.4 Paketprotokoll 50
7.5 Implementierung 52

8 Analyse des verbleibenden Rückkanals **56**
8.1 Impliziter Rückkanal 56
8.2 Vergleich zur genugate-Datendiode 58
8.2.1 Paketebene 58
8.2.2 Zeitebene 60

9 Fazit und Ausblick **63**
9.1 Ergebnisse 63
9.2 Ausblick 63

10 Thesen **65**

11 Literatur **66**
11.1 Literatur 66

12 Glossar **68**

Abstract

This work describes the development of a secure data-diode protocol. In this case secure means a strictly reduced feedback channel. A reduced feedback channel leads to a reduced covert channel thus increasing the security of a data-diode. An attacker can effectively not use such reduced feedback channel for data transfer. This aim was reached by reducing the red side's degree of freedom of communication with the black side. A proxy process enforces the protocol described in this paper. All communication components are on the same host system to ensure reliability. The L4 microkernel is used to separate these components from each other. The techniques have been developed in this thesis can be used to design a data diode with a very limited feedback channel. Thus, data throughput and the number of parallel connections are not reduced in any way.

Diese Arbeit beschreibt die Entwicklung eines sicheren Datendioden-Protokolls. Sicher bedeutet in diesem Fall, dass der Rückkanal enorm reduziert wird. Dadurch wird ebenfalls der Covert-Channel reduziert und führt damit zur Erhöhung der Sicherheit einer Datendiode. Angreifer können dadurch den Rückkanal nicht mehr effektiv für die Übermittlung von Daten verwenden. Erreicht wurde dieses Ziel durch eine Reduktion der Freiheitsgrade, die der roten Seite in ihrer Kommunikation mit der schwarzen Seite zur Verfügung stehen. Die Durchsetzung des entwickelten Protokolls erfolgt durch einen sicheren Proxy-Prozess, welcher die Kommunikation zwischen schwarzer und roter Seite kontrolliert. Um nicht nur Sicherheit sondern auch Zuverlässigkeit zu gewährleisten, befinden sich alle Kommunikationskomponenten auf dem selben Host-System und werden durch einen L4-Mikrokernel voneinander separiert. Durch den Einsatz der in dieser Arbeit beschriebenen Techniken lässt sich eine Datendiode mit einem zu vernachlässigendem Rückkanal betreiben, ohne dabei den Durchsatz oder die Anzahl der parallelen Verbindungen in irgendeiner Weise einschränken zu müssen.

I dedicate this thesis to Private First Class Bradley Edward Manning to remind the reader of this thesis and the user of this technology of the necessity of such deeds. Security technology should not be used to suppress information in ways which compromise our freedom and democracy. I make this claim in the full knowledge that my work may hinder or prevent future attempts to expose such suppression.

1 Einleitung

Das Geheimhalten sensibler Informationen stellt sich dieser Tage als eine besondere Herausforderung dar. Privatpersonen und Unternehmen stehen seit Jahren immer professioneller werdenden Kriminellen gegenüber, die es auf persönliche Geheimnisse wie PIN-Nummern und Kennwörter abgesehen haben, um an Geld oder moderne Technologie zu gelangen.

Neben einfacher oder auch organisierter Kriminalität gibt es auch Staaten, die sich mit einem wesentlich höheren Einsatz von Ressourcen an der Beschaffung von vertraulichen Informationen beteiligen. Staaten sind dabei aber auch oft das Ziel. Durch den hohen Einsatz von Ressourcen auf Seiten der Angreifer haben es Staaten ebenfalls immer schwerer ihre Geheimnisse und sensiblen Informationen zu sichern. Seien es Positionen und Verhandlungspunkte beim Aushandeln von bilateralen Abkommen oder um taktische und militärische Informationen, die es als Staat zu wahren gilt. In jedem Fall müssen diese Informationen durch Computer verwaltet und verarbeitet werden um diese handhaben zu können. Dieser Umstand macht es für einen Angreifer, dem viele Ressourcen zur Verfügung stehen, sehr attraktiv diese Informationen durch Computer-Spionage zu erlangen.

Um solche sensible Informationen vor Angreifern zu schützen, denen viele Ressourcen zur Verfügung stehen, bedarf es eines besonderen Sicherheitsniveaus. Datendioden bieten hier eine Möglichkeit, um eine sichere Abschottung von Informationen unter solchen Bedingungen zu gewähren. Datendioden schotten Computernetze derart von einander ab, dass Ihr Einsatz mit hohen Einschränkungen in der Benutzbarkeit einhergeht. Sie sind dafür gebaut, um Daten zwischen zwei Netzen nur in eine Richtung zu übertragen. Diese Trennung ist teilweise so elementar, dass selbst Informationen wie Empfangsbestätigungen oder Steuerdaten zur Flusskontrolle nicht auf die entgegengesetzte Seite gelangen. Diese Eigenschaft macht den Einsatz von Datendioden zu einem sehr wirkungsvollem Mittel, um die Vertraulichkeit von sensiblen Informationen in separaten Computernetzen zu wahren, aber gefährdet dabei auch andere Ziele. Informationen die von außen über Datendioden in diese abgeschotteten Netze übertragen werden können ohne Empfangsbestätigung oder Flusskontrolle durch Übertragungs-

fehler ihre Integrität verlieren.

Diese Arbeit geht der Fragestellung nach, wie man eine hohe Abschottung und gleichzeitig einen adäquaten Rückkanal für Empfangsbestätigungen und Steuerdaten miteinander vereinen kann. Rückkanäle sind bei Datendioden ein großes Problem. Auch ein sehr streng kontrollierter Rückkanal kann von einem Angreifer, welcher auch ein Innentäter sein kann, potentiell immer als Covert-Channel missbraucht werden. Covert-Channel sind verdeckte Kanäle, die entgegen ihrer ursprünglichen Aufgabe von Angreifern zweckentfremdet werden, um Informationen an Sicherungsmaßnahmen vorbei zu schleusen. Bei Datendioden müssen diese Rückkanäle daher so gering wie möglich sein. Es gilt daher der Grundsatz: je kleiner ein Rückkanal einer Datendiode ist, desto größer deren Abschottung und Sicherheit.

Im Speziellen versucht diese Arbeit ein Kommunikationsprotokoll zu entwickeln, welches diese verschieden Eigenschaften miteinander verbindet. Dabei steht die Untersuchung und Reduktion des Rückkanals im Zentrum der Betrachtung. Eingebettet ist diese Arbeit in ein Projekt der Firma genua mbh[1], welches sich mit der Separierung von mehreren Software-Systemen auf einer Hardware auseinander setzt.

[1] http://www.genua.de/

2 Problemstellung und Anforderungen

Dieses Kapitel beschreibt die Einsatzbereiche und Aufgaben von Datendioden um von diesen auf die grundlegende Problemstellung und die daraus resultierenden Anforderungen zu schließen. Dabei werden der Einstatzzweck sowie die Bedingungen, unter denen Datendioden verwendet werden, näher erläutert.

2.1 Anwendungsfälle von Datendioden

Datendioden sind spezielle Netzwerkverbindungspunkte, an denen der Netzwerkverkehr ausschließlich in eine Richtung übertragen wird. Diese unidirektionale Übertragungseigenschaft wird benötigt, wenn zwei Netzwerke mit unterschiedlichen Sicherheitsniveaus verbunden werden. Im wesentlichen gibt es zwei Szenarien für die der Einsatz von Datendioden benötigt wird: Zum einen für den Schutz von geheimen Informationen im militärischen oder geheimdienstlichen Umfeld, zum anderen für den Schutz von Infrastrukturen im Bereich der Industrie.

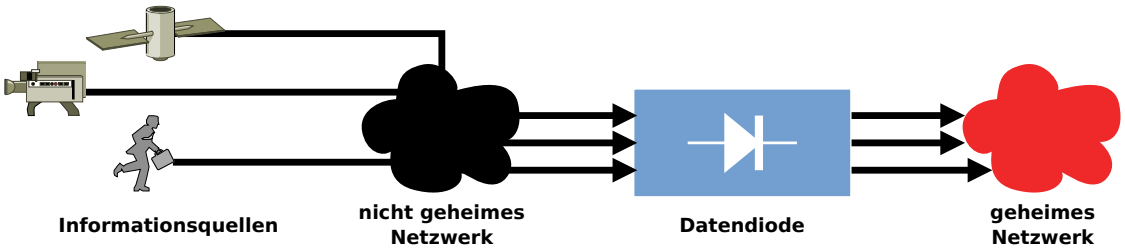

Abbildung 1: Anwendung von Datendioden zum Geheimnisschutz

Im Umfeld von Geheimdiensten und militärischer Aufklärung werden Informationen gesammelt und an höhere Stellen weitergeleitet. Diese fassen gesammelten Informationen zusammen und leiten diese Zusammenfassungen ebenfalls an höhere Stellen weiter. Jede dieser Hierarchiestufen versucht aus den gesammelten Informationen Wissen zu generieren. Dieses Wissen darf unter keinen Umständen an die unteren Hierarchiestufen zurückfließen. Durch dieses Vorgehen soll sichergestellt werden, dass die gewonnen Erkenntnisse nur einem bestimmen Personenkreis bekannt sind. Abteilungen mit niedrigeren Sicherheitseinstufungen kennen so nur einen Teil der unverarbeiteten

Informationen. Allerdings müssen die unteren Abteilungen in der Lage sein, automatisiert Informationen an höhere Abteilungen zu übermitteln. Bei diesem Übergang von Netzwerken mit niedriger und hoher Einstufung kommen Datendioden zum Einsatz. Sie sorgen dafür, dass Informationen jederzeit automatisiert und zuverlässig in nur eine Richtung übertragen werden und stellen gleichzeitig sicher, dass keine Informationen in die Gegenrichtung abfließen. In diesem Szenario ist die Vertraulichkeit von Informationen das wichtigste Schutzziel. Integrität und Verfügbarkeit werden diesem meist untergeordnet.

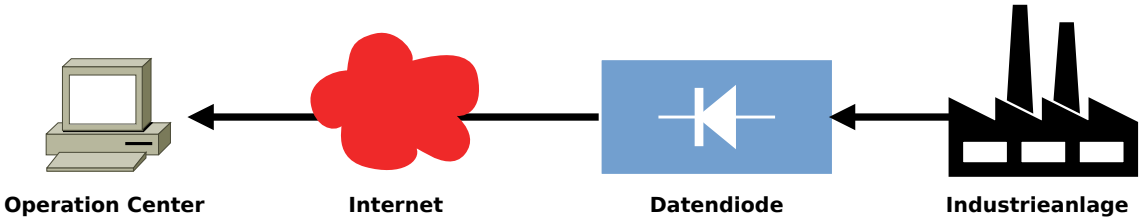

Abbildung 2: Anwendung von Datendioden zum Infrastrukturschutz

Im Gegensatz dazu ist der Anwendungsfall im Industrieumfeld ein vollkommen anderer. Hier werden große Industrieanlagen, wie etwa Fertigungsstraßen oder Kraftwerke, für jahrzehntelangen Betrieb gebaut. Die dabei verwendete IT-Technik ist daher in den meisten Fällen nicht auf dem aktuellstem Stand. Sicherheits-Updates werden oft wegen möglicher Ausfälle nicht eingespielt oder stehen nicht mehr zur Verfügung, da die verwendete Software vom Hersteller nicht mehr unterstützt wird. Dennoch müssen diese Anlagen weiter laufen und gewartet werden. Viele Automatisierungssysteme solcher Anlagen erstellen Log-Daten, welche in zentralen Leitstellen ausgewertet werden. Aus diesen Daten können dann Rückschlüsse auf den Zustand der Anlage sowie auf notwendige Wartungen gezogen werden. Um diese Anlagen nun sicher über ein eventuell auch unsicheres Netzwerk mit der Leitstelle verbinden zu können, kommen hier ebenfalls Datendioden zum Einsatz. Eine Datendiode stellt sicher, dass die Anlage ihre Daten an die Leitstelle übertragen kann ohne von außen kompromittiert zu werden. Die Integrität der Anlage ist für den Betreiber das oberste Schutzziel. Die Vertraulichkeit oder Verfügbarkeit der übermittelten Daten wird dem in der Regel untergeordnet.

2.2 Rot-Schwarz-Terminologie

Im Kontext von Datendioden wird oft von *rot* und *schwarz* gesprochen. Dieser Abschnitt soll die Herkunft und die Bedeutung dieser Begriffe für diese Arbeit erklären.

In dem Dokument „RED/BLACK Installation Guidance“ [McCo95] der NSA, welches sich mit der Abstrahlsicherheit von Elektronik auseinandersetzt, werden die Begriffe *rot* und *schwarz* mit dem Rot-Schwarz-Konzept wie folgt beschrieben:

> **RED/BLACK Concept**. Separation of … systems that handle national security information (RED), in electrical form, from those that handle non-national security information (BLACK) in the same form. …
>
> --- Section 2[McCo95]

Im selben Dokument [McCo95] werden die Begriffe *rot* und *schwarz* nicht nur für die Einstufung von Informationen verwendet, sondern auch für deren Zustand. So spricht man in diesem Dokument ebenfalls von verschlüsselten (schwarzen) und unverschlüsselten (roten) Informationen oder Signalen.

In dieser Arbeit werden die Begriffe ausschließlich für die Kennzeichnung einer unterschiedlichen Einstufung der zu verbindenden Netzwerke verwendet. Dabei wird nicht auf die Höhe der Einstufung selbst eingegangen, sondern auf den Umstand, dass sich diese unterscheiden. Die schwarze Seite ist dabei immer das Netz mit der niedrigeren Einstufung und rot das mit der höheren Einstufung. Gleichzeitig lässt sich an den Begriffen immer die Ausrichtung der Datendiode und damit die Flussrichtung der Informationen erkennen. Informationen werden ausschließlich von schwarz nach rot übertragen.

2.3 Anforderungen an Datendioden

An eine Datendiode werden verschiedene Anforderungen gestellt. Die Eigenschaft der Unidirektionalität ist dabei die wichtigste Anforderung. Eine Datendiode muss unter allen Umständen sicherstellen, dass keine Daten von rot nach schwarz übertragen werden.

Die Zuverlässigkeit einer Datendiode ist ein weiteres Ziel, welches generell für jede Datenübertragung gelten sollte. Bei der Übertragung sollte für den Sender ersichtlich sein, dass die gesendeten Informationen auf der roten Seite auch angekommen sind und korrekt interpretiert werden konnten. Wird diese Anforderung nicht erfüllt, kann es zu Datenverlusten kommen. Um sie umzusetzen, muss die Datendiode einen Rückkanal für eine Empfangsbestätigung bereitstellen. An dieser Stelle widersprechen sich die Anforderungen der Unidirektionalität mit der Zuverlässigkeit, denn die Zuverlässigkeit fordert einen Rückkanal, welchen die Unidirektionalität verbietet.

Um beide Anforderungen erfüllen zu können, muss beim Design des Datendioden-Protokolls auf einen sehr reglementierten Rückkanal geachtet werden. Dieser darf nicht von einem Angreifer für das Übermitteln von Informationen von rot nach schwarz missbraucht werden.

Die letzte Anforderung betrifft die Nachvollziehbarkeit der Implementierung. Weil Datendioden in den meisten Fällen in einem Umfeld eingesetzt werden, in denen solche sicherheitskritischen Kommunikationskomponenten zertifiziert oder zugelassen[2] werden müssen, sollte sich die Implementierung von Dritten einfach überprüfen lassen. Dafür sollte die Programmlogik sowie der Quelltext einen geringen Umfang sowie eine geringe Komplexität aufweisen. Diese Anforderung sollte bei der Implementierung bedacht werden.

[2] Zertifizierungen sowie Zulassungen im IT-Bereich werden in Deutschland meist vom Bundesamt für Sicherheit in der Informationstechnik durchgeführt. [BSIZ13]

3 Grundlagen

In diesem Kapitel werden dieser Arbeit zugrundeliegende Techniken und verwendete Fachbegriffe eingeführt. Zudem werden ähnliche Projekte und deren Unterschiede zu dieser Arbeit erläutert.

3.1 Physische und Logische Datendioden

Wenn man von Datendioden spricht, sollte man grundsätzlich zwei verschiedene Arten unterschieden. Im Folgenden werden die Begriffe *physische* sowie *logische* Datendiode zur Unterscheidung verwendet.

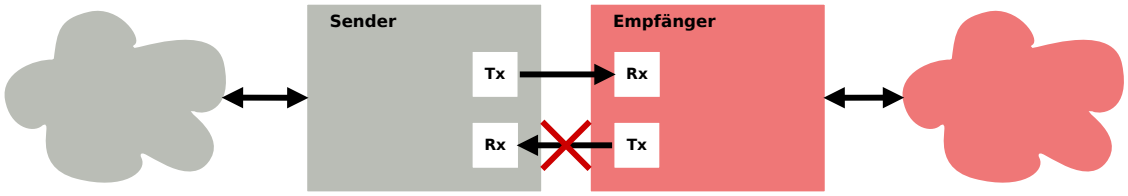

Abbildung 3: Aufbau einer physischen Datendiode

Bei physischen Datendioden wird die Unidirektionalität durch die Hardware sichergestellt, daher wird diese Art von Datendioden auch als Hardware-Dioden bezeichnet. Das bedeutet, dass zwei unabhängige Computer-Systeme mit einer Datenleitung verbunden werden, bei der ausschließlich Transmitter und Receiver einer Richtung miteinander verbunden sind. Der Transmitter und Receiver der Rückrichtung werden nicht miteinander verbunden.

Eine einfache Art einer unidirektionalen Datenverbindung auf physischer Ebene wird in [CaSi09] beschrieben. Eine Netzwerkverbindung über den AUI[3]-Standard lässt sich in eine unidirektional Verbindung umbauen, indem man die Pins 3 und 10 auf der roten Seite isoliert oder physisch durchtrennt. Dadurch wird die rote Seite gehindert, Daten gemäß des Standards zu übertragen. Diese Methode stellt nicht sicher, dass nicht über eine modifizierte Software dennoch Daten über

[3] AUI (Attachment Unit Interface) ist eine Netzwerkschnittstelle welche im IEEE 802.3 Standard definiert wird.

die Steuer- und Kontrollpins übertragen werden können und ist daher nicht als absolut sicher einzustufen. Sie ist aber ein Ansatz, welcher versucht das Problem der unidirektionalen Kommunikation auf Hardware-Ebene zu lösen.

Im Jahr 2006 wurde an der University of Iowa eine physische Datendiode für die RS-232-Schnittstelle entwickelt [Jone06]. Bei dieser Datendiode wurden Transmitter und Receiver auch elektrisch entkoppelt und über eine optische Datenverbindung mittels einer LED und eines Phototransistor, verbunden. Dieses Beispiel zeigt sehr deutlich, dass bei solch einer Datendiode die Unidirektionalität und damit die Sicherheit, auf physischer Ebene sichergestellt wird. Einem Angreifer ist es ohne Zugriff auf die Hardware nicht möglich, diese Übertragungsstrecke umzudrehen oder in eine bidirektionale zu überführen, um Daten in die Gegenrichtung zu übertragen.

Bei logischen Datendioden hingegen sind grundsätzlich alle Komponenten bidirektional miteinander verbunden. Die Unidirektionalität wird ausschließlich durch die implementierte Software-Logik erreicht. Dadurch sind logische Datendioden potentiell immer anfällig für Fehler in der Konfiguration sowie Angriffe auf Schwachstellen in der Implementierung.

Diese Diodenart bietet den Vorteil, durch die Übertragung von Empfangsbestätigungen dem Sender mitteilen zu können, dass seine Nachricht auf der roten Seite angekommen ist. Dieser sehr limitierte Rückkanal ermöglicht eine zuverlässigere Kommunikation als bei physischen Datendioden. Nachteile dieser Diodenart sind zum einen, dass ein Software-Fehler es einem Angreifer ermöglichen könnte bidirektional zu kommunizieren. Zum anderen kann auch ein stark limitierter Rückkanal potentiell immer zum Übertragen von Informationen als *Covert-Channel* ausgenutzt werden.

Aus den Eigenschaften der physischen und logischen Datendioden ergeben sich verschiedene Vor- und Nachteile, welche vor einer Auswahl und dem Einsatz bedacht werden sollten. Physische Datendioden sind immer auf mindestens zwei getrennte Hardware-Einheiten angewiesen. Dem gegenüber bieten logische Datendioden die Möglichkeit sich auf einer Hardware zu beschränken. Dieses kann in einer Einsatzumgebung mit beschränkten Raum ein entscheidendes Kriterium sein. Da sich bei physische Datendioden keinerlei Möglichkeit zur Flusskontrol-

le bietet, müssen diese ihre Übertragungsgeschwindigkeit soweit einschränken, dass die Empfängerseite die Daten in jedem fall verarbeiten kann. Das führt zu dem Umstand, dass physische Diodenart abhängig von der eingesetzten Übertragungstechnik meist unterhalb der Möglichkeiten der eingesetzten Hardware bleiben. Logische Datendioden haben diese Einschränkung nicht und erlauben es, prinzipiell ohne eine künstliche Drosselung der Geschwindigkeit, Daten zu übertragen. Die Tabelle 1 fasst den Vergleich von physischen und logischen Datendioden noch einmal zusammen.

Tabelle 1: Vergleich von phys. und log. Datendioden

Eigenschaft	**phy. Datendiode**	**log. Datendiode**
Sicherheit	Hardware-Ebene	Software-Ebene
Zuverlässigkeit	nicht sicher	sicher
Rückkanal	keiner	Flusskontrolle und Empfangsbestätigungen
OSI-Ebene	Layer 1 (Hardware)	Layer 2 bis 7 (Software)
Geschwindigkeit	begrenzt	unbegrenzt
Hardware-Einheiten	mindestens 2	mindestens 1

3.2 Unidirektionale Protokoll-Relays

Um über eine Datendiode Informationen verschicken zu können, bedarf es eines Anwendungsprotokolls. In der Regel werden dafür reguläre Anwendungsprotokolle wie etwa SMTP, FTP oder ähnliche verwendet. Die meisten dieser Protokolle setzen einen bidirektionalen Kommunikationskanal voraus. Der Anwendungsfall einer unidirektionalen Kommunikationsgrundlage mit einem stark reduziertem oder gar keinem Rückkanal ist in den meisten Anwendungsprotokollen nicht bedacht. Daher bedarf es spezieller Application-Level-Gateways, auch Proxys oder Relays genannt, welche die Umsetzung eines Anwendungs-

protokolls auf einen unidirektionalen Kommunikationskanal vornehmen.

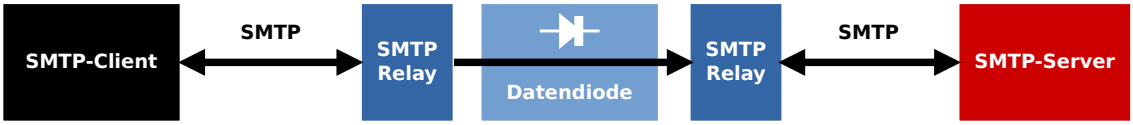

Abbildung 4: Einsatz von SMTP-Relays mit einer Datendiode

Diese Relays vermitteln zwischen einem Client oder Server auf der einen Seite und der Datendiode auf der anderen. Nach außen müssen sie dabei mit dem jeweils verwendeten Anwendungsprotokoll kommunizieren. Auf der Innenseite kann dann ohne weitere Kompatibilitätsprobleme mit einem beliebigen Protokoll kommuniziert werden.

3.3 Analyse bekannter Angriffsvektoren auf unidirektionale Kommunikationskanäle

Dieser Abschnitt erläutert verschiedene Angriffsszenarien auf logische Datendioden. Diese Angriffe werden dem Datendioden-System dieser Arbeit entgegen gestellt, um zu verdeutlichen, gegen welche Arten von Angriffen man sich beim Design eines Datendioden-Protokoll schützen muss.

3.3.1 Implementierungsfehler

Beim Angriff auf die Software einer logischen Datendiode geht man davon aus, dass der Angreifer aus dem schwarzen oder roten Netz heraus versucht, die Datendiode als ganzes zu kompromittieren, um dann ungehindert Daten in beide Richtungen übertragen zu können.

Abbildung 5: Angriff über Kompromittierung der Software

Implementierungsschwachstellen in Software lassen sich leider nie ausschließen. Durch die Methoden der defensiven Software-Entwicklung [Davi05] und dem externen Verifizieren von Software in einem Zertifizierungsprozess [BSIZ13] lässt sich die Gefahr dieses Angriffsvektors aber reduzieren.

3.3.2 Außen-Szenario

Beim Außen-Szenario wird angenommen, dass der Angreifer im schwarzen und roten Netz sitzt, um Daten aus dem roten Netz ins schwarze zu übertragen. In diesem Szenario kommuniziert der Angreifer mit den Schnittstellen, welche die Datendiode nach außen zur Verfügung stellt um über diese Daten ins schwarze Netz zu übertragen. Diese Schnittstellen sind meist die im Abschnitt 3.2 beschriebenen unidirektionalen Protokoll-Relays.

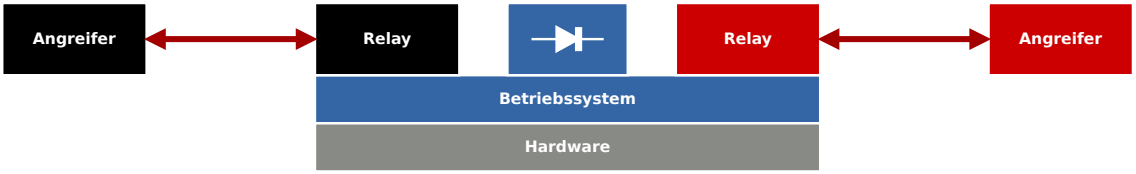

Abbildung 6: Außen-Angriff

Konkret lässt sich das an einem praktischem Szenario verdeutlichen, bei dem über die Datendiode E-Mails unter Verwendung des SMTP-Protokolls verschickt werden. Erhält der Angreifer Kontrolle über den roten sowie schwarzen SMTP-Server kann er deren Kommunikation zu seinen Gunsten beeinflussen. Die Datendiode selbst leitet die E-Mails von der schwarzen an die rote Seite weiter und übertragt dabei ausschließlich die Rückmeldung, ob der Transfer fehlerfrei abgelaufen ist oder nicht. Der Angreifer kann nun beliebige E-Mails von schwarz nach rot übertragen und auf der roten Seite die E-Mails je nach Bedarf annehmen oder mit einem Fehler ablehnen. Der Angreifer kann über diese Methode beliebige Nachrichten in das schwarze Netz übermitteln.

3.3.3 Innen-Szenario

Im Innen-Szenario wird dem Angreifer Kontrolle über Teile der Datendiode unterstellt. Bei einer physischen Datendioden (siehe Abschnitt 3.1)

ist dieses Szenario ungefährlich, da die Software-Logik oder die Kontrolle über die Software keine Auswirkung auf die Unidirektionalität haben. Bei einer logischen Datendiode, ist dieses grundsätzlich anders. Hat ein Angreifer Kontrolle über die Software kann er die unidirektionale Kommunikation in eine bidirektionale verändern und somit Daten aus dem roten in das schwarze Netz übertragen.

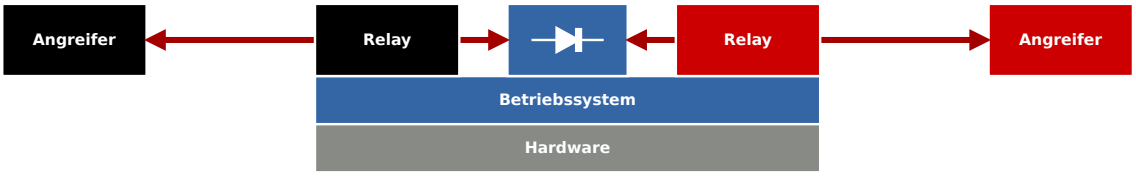

Abbildung 7: Innenangriff

Wird beim Außen-Szenario noch die Sicherheit der Relays betrachtet, so brachtet man hier den unidirektionalen Kommunikationsweg zwischen den Relays. Dieses Angriffsszenario wird im weiteren Verlauf dieser Arbeit sowie beim Design des unidirektionalen Protkolls als vorhanden angenommen. Es wird also davon ausgegangen, dass der Angreifer direkten Zugriff auf diesen interen Kommunikationsweg hat.

3.4 Bestehenden Datendioden

Dieser Abschnitt beschäftigt sich mit bereits entwickelten Datendioden, deren Funktionsweise, sowie deren Stärken und Schwächen. Es wurden dafür einige exemplarisch herausgesucht, da die meisten Datendioden reine physische Datendioden auf Basis von einseitigen Glasfaserverbindungen sind. Sie unterscheiden sich zumeist nur in der Anzahl der Anwendungsprotokolle die sie unterstützen.

3.4.1 NRL-Network-Pump

Die NRL-Pump ist eine logische Datendiode, welche in einer Reihe von Forschungsprojekten am „Naval Research Laboratory“ entwickelt wurde. Die Forschungsprojekte liefen in der Zeit von 1993 bis 2003. In der Veröffentlichung „The Pump: A Decade of Covert Fun“ [Kang05] wird die gesamte Entwicklung dieser Datendiode zusammenfassend erläutert. Im Gegensatz zu anderen Datendioden wurde im Pump-Projekt

nicht nur die Software entwickelt, sonder auch eine spezielle Hardware.

Abbildung 8: NRL-Pump-Produkt [Kang05]

Im Dokument [Kang05] wird nicht von rot und schwarz gesprochen, sonder von „Low“ und „High“. „Low“ meint dabei die schwarze und „High“ die rote Seite. In der Abbildung 9 ist der Kommunikationskanal durch die Pump dargestellt. Die Boxen „Low/High Enclave“ stellen dabei Hosts im schwarzen sowie roten Netz dar, welche über die Pump kommunizieren.

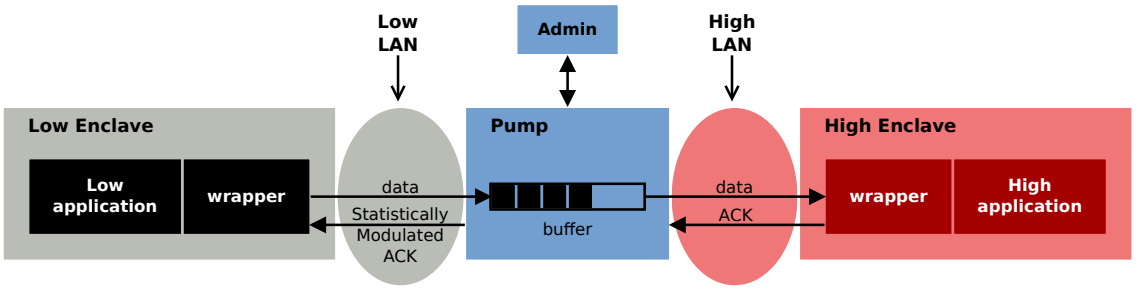

Abbildung 9: The Pump, wrappers, and applications, (nachgezeichnet und eingefärbt) [Kang05]

Die Pump selbst stellt keine Schnittstelle für reguläre Anwendungsprotokolle bereit. Das Design der Pump sieht vor, dass auf dem Host im schwarzen sowie roten Netz einen Anwendung direkt gegen einen Bibliothek programmiert wird, welche eine Kommunikationsschnittstelle

zur Pump bietet. In Abbildung 9 wird diese Schnittstelle als „Wrapper“ bezeichnet. „Low application“ und „High application“ sind die Anwendungen, die den „Wrapper“ benutzen um über die Pump zu kommunizieren.

3.4.2 genugate-Datendiode

Die genugate-Datendiode ist eine logische Datendiode, welche auf dem TCP/IP-Protokoll basiert. Sie wurde von genua entwickelt, um mit regulären Anwendungsprotokollen unidirektional von schwarz nach rot kommunizieren zu können. Diese Datendiode besteht aus drei von einander unabhängigen Hardware-Einheiten: Zwei Application-Level-Gateways (ALG) sowie einem Paketfilter. Diese getrennten Rechner sind ausschließlich über von einandere separierte Netzwerk miteinander verbunden.

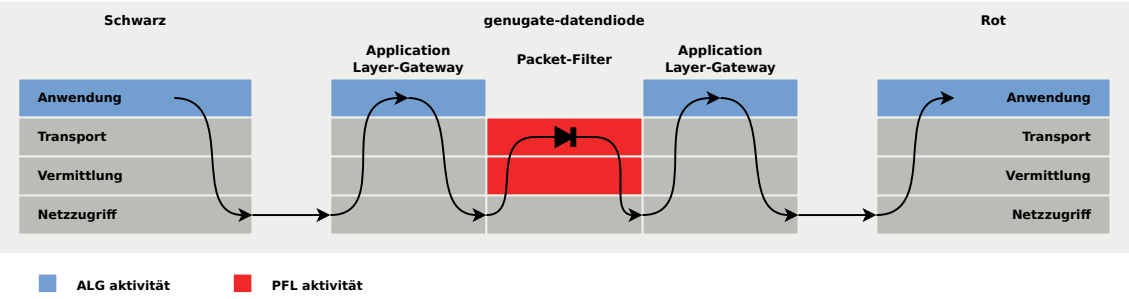

Abbildung 10: DoD-Schichtenmodell der genugate-Datendiode

Anwendungsprotokolle werden an den ALGs terminiert. Die beiden ALGs untereinander kommunizieren über einseitige TCP-Verbindungen, welche vom Paketfilter überprüft werden. Der Paketfilter erzwingt, dass ausschließlich die schwarze Seite TCP-Verbindungen aufbauen kann. Von der roten Seite werden nur Acknowledgement-Pakete zugelassen, welche den Datenfluss regulieren und gleichzeitig der schwarzen Seite signalisieren, dass die übertragenen Daten angekommen sind.

Abbildung 11: Hardware der genugate-Datendiode

Der größte Vorteil dieser Datendiode besteht in der Zuverlässigkeit. Die schwarze Seite bekommt eine Rückmeldung, ob die Daten angekommen sind oder nicht und kann im Fehlerfall entsprechend reagieren. Zudem kann man zur Kommunikation von schwarz nach rot mit regulären Anwendungsprotokollen arbeiten, ohne spezielle Programme dafür entwickeln zu müssen.

Nachteil dieser Variante ist, dass die Unidirektionalität von der Software-Logik des Paketfilters abhängt. Ist ein Angreifer in der Lage die Kontrolle über die Software zu erlangen, kann er ebenfalls bidirektional über die Datendiode kommunizieren und somit Daten von rot nach schwarz übertragen.

3.4.3 genuscreen-Paketdiode

Die genuscreen-Paketdiode ist eine physische Datendiode, welche von genua entwickelt wurde. Sie ist dafür entwickelt worden, UDP-Pakete [RFC768] unidirektional zwischen zwei Netzen zu übertragen. Im Gegensatz zur genugate-Datendiode gibt es hier keinen Proxy, welcher Anwendungsprotokolle transparent über die unidirektionale Verbindung überträgt. Die unidirektionale Übertragungsstrecke zwischen den beiden Hardware-Einheiten wird über eine einseitige Netzwerkverbindung auf Glasfaserbasis realisiert. Dabei wird nur der Transmitter der schwarzen Seite mit dem Receiver der roten Seite verbunden.

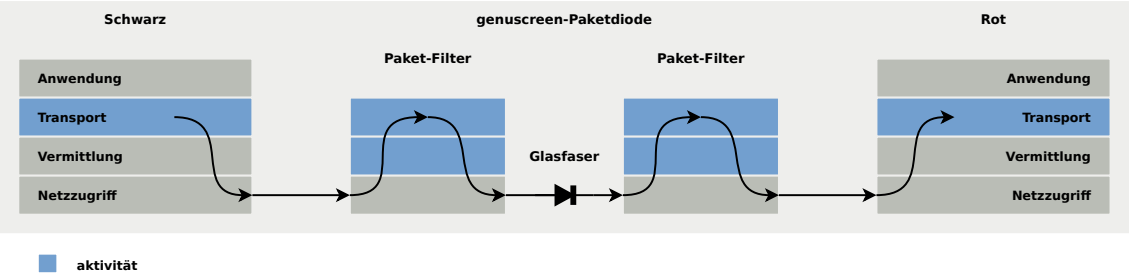

Abbildung 12: DoD-Schichtenmodell der genuscreen-Paketdiode

Daher stehen den Anwendungen im schwarzen und roten Netz nur die Möglichkeit von UDP-Paketen zur Kommunikation bereit. Für diese Datendiode müssen daher spezielle Netzwerkprotokolle entwickelt werden. Die Anwendung auf der schwarzen Seite kann beliebige UDP-Pakete an eine Anwendung auf der roten Seite verschicken. Die rote Seite kann allerdings nicht auf diese antworten. Der schwarzen Seite kann auch keine Empfangsbestätigung — weder von der rote Anwendung noch von der Datendiode selbst — mitgeteilt werden. Dieser Umstand macht die Kommunikation über die genuscreen-Datendiode unzuverlässig. Sie ist daher eher für die Übertragung von nicht zusammenhängenden Informationen geeignet, wie etwa kontinuierliche Zeitstempel, Audio- oder Video-Informationen. Bei diesen Informationen ist der Verlust von einzelnen Paketen nicht relevant, solange die nachfolgenden Pakete korrekt empfangen werden.

Vorteil dieser Datendiode ist, dass die Unidirektionalität, wie im Abschnitt 3.1 beschrieben, auf Hardware-Ebene gegeben ist. Somit kann ein Software-Fehler oder das erfolgreiche Eindringen eines Angreifers nie zum Abfliessen von Informationen aus dem roten Netz führen.

3.4.4 SINA-One-Way-Gateway

Das „SINA-One-Way-Gateway“ ist eine vom Hersteller Secunet[4] entwickelte physische Datendiode. Sie biete unidirektionale Protokoll-Relays für die Protokolle SMTP, SMB und FTP.

[4] http://www.secunet.com/

3.4.5 Fort Fox Data Diode

Die „Fort Fox Data Diode“ [Fox07] ist eine physische Datendiode, welche von Fox-IT[5] hergestellt wird. Die Unidirektionalität dieser Datendiode beruht auf einer einseitigen Glasfaser-Verbindung. Sie besteht aus zwei unabhängigen Hardware-Komponenten und bietet unidirektionale Protokoll-Relays für die Protokoll SMTP sowie FTP.

3.4.6 Waterfall-Diode

Von Waterfall® Security Solutions [6] wird eine ganze Reihe von Datendioden entwickelt. Alle Produkte basieren dabei auf einer physischen Datendiode, welche die Unidirektionalität wie bei anderen physischen Datendioden auch durch eine einseitige Glasfaser-Verbindung sicherstellt. Die Waterfall-Datendioden unterscheiden sich in der Vielzahl von unidirektionalen Protokoll-Relays, die es ermöglichen eine große Anzahl von Anwendungsprotokollen von schwarz nach rot zu übertragen.

3.5 Mechanismus zur Integritätssicherung in Datendioden

Ein Problem von Datendioden ist die Konsistenz der übertragenen Daten. Logische Datendioden haben an dieser Stelle den Vorteil über einen Rückkanal einen positiven oder negativen Status an den Sender übermitteln zu können. Diese Statusmeldung kann zur Integritätssicherung verwendet werden. Im folgenden werden bekannte Möglichkeiten zur Integritätssicherung von Datendioden erläutert und deren Vor- und Nachteile diskutiert.

3.5.1 Gehashte Rückmeldung

Im US-Patent „one-way data transfer system with built-in data verification mechanism“ [Hope10] wird ein Mechanismus beschrieben, der die Datenintegrität bei Einwegübertragungen sicherstellt. Benutzt wird diese Verfahrensweise bei der Owl-Datendiode[7].

[5] http://www.fox-it.com/en/
[6] http://www.waterfall-security.com/
[7] http://www.owlcti.com/

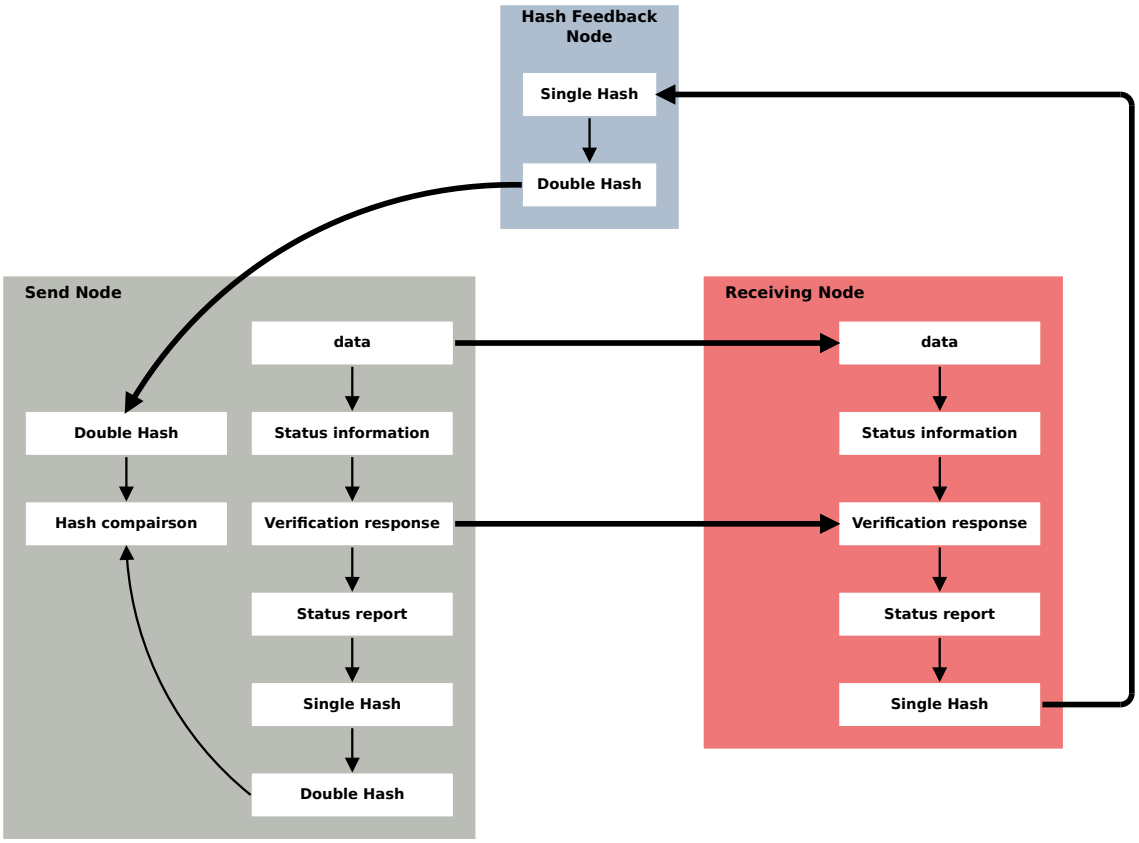

Abbildung 13: Datenverifikation in Datendioden mittels Hash-Übertragung, nachgezeichnet aus [Hope10]

Die Unidirektionalität wird dabei durch die Feedback-Node sicher gestellt. Diese als sicher angenommene Node empfängt einen Hash der vom Receiver empfangenden Daten, erzeugt einen neuen Hash aus dem empfangenden Hash und sendet den entstandenen Hash an den Sender weiter. Der Sender erzeugt nun von den zuvor gesendeten Daten einen Hash, von welchem wiederum erneut ein Hash erzeugt wird. Der sogenannte *double-hash* kann dann mit dem von der Feedback-Node empfangenen *double-hash* verglichen werden. So kann überprüft werden ob die Daten korrekt übertragen wurden.

Dadurch, dass die Feedback-Node alle Daten einer Hash-Operation unterzieht, bevor diese weiter versandt werden, ist sichergestellt, dass der Receiver direkt keine beliebigen Daten an den Sender übermitteln kann.

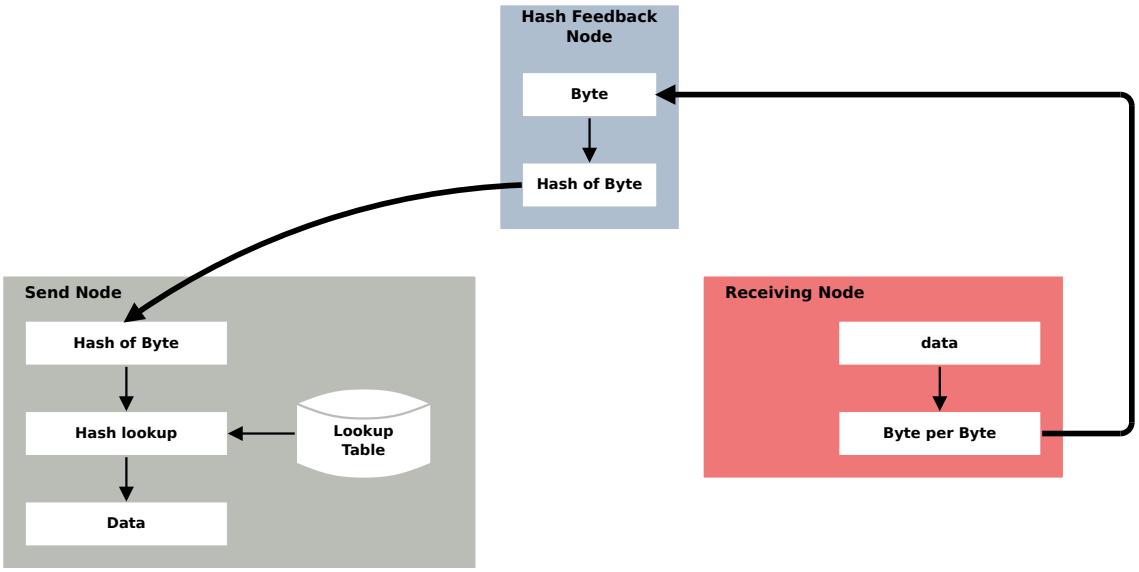

Abbildung 14: Rückkanal mit Hash-Lookup-Table

Der Sender kann aus den empfangenen Hash-Werten die Urdaten nicht direkt ermitteln, aber bei einem mit dem Receiver abgesprochenen Alphabet, den Wertebereich der Urdaten kennen. Mit dem Wissen dieses Wertebereichs, lässt sich eine Lookup-Table mit allen möglichen Hash-Werten zu ihren Urdaten erstellen.

Eine kleine Optimierung der Feedback-Node, wäre denkbar, indem diese einen sogenannten Salted-Hash-Wert aus dem vom Receiver empfangenden Daten erstellen würde. In diesem Fall müsste, die Send-Node für jeden empfangenden Hash-Wert erneut alle 256 Möglichkeiten eines Hashes in Verbindung mit dem Salt-Wert berechnen, um die Urdaten zu ermitteln. Da die Berechnung von 256 Hash-Werten pro Byte, mit aktueller Rechenleistung keinen großen Aufwand darstellt, ist auch diese kleine Optimierung kein ausreichendes Hindernis, welches die Ausnutzung des Rückkanals verhindern würde.

3.5.2 Einseitiges TCP

Das im Abschnitt 4.2 beschrieben Protokoll der genugate-Datendiode setzt zur Integritätssicherung der Übertragung auf Mechanismen des TCP-Protokolls [RFC793]. Im TCP-Header ist ein *Checksum*-Feld definiert, welches eine Prüfsumme über Paket-Header und -Daten enthält. Der Empfänger kann anhand dieser Prüfsumme Veränderungen des Pa-

kets durch den Übertragungsweg im Netzwerk erkennen. Zudem stellt die *Sequence-Number* im TCP-Header sicher, dass keinen Pakete unbemerkt verlorengehen können. Erkennt der Empfänger anhand dieser Mechanismen Fehler in der Übertragung, bestätigt er die fehlerhaften Teile nicht und bekommt diese Daten erneut übertragen.

Vorteil dieser Methode ist die Nutzung eines erprobten Verfahrens, welches zuverlässig im täglichen Einsatz funktioniert. Der Nachteil, insbesondere bei der Verwendung von TCP, sind die Freiheitsgrade, welche der roten Seite geboten werden. Im Dokument [Meis07] wird die Analyse des Rückkanal der genugate-Datendiode detailierter dargestellt. Die Freiheitsgrade in der Kommunikation der roten Seite sind die Grundlage für Angriffe auf den Rückkanal. Die Bedeutung der Freiheitsgrade für den Rückkanal wird im Abschnitt 6.1 genauer erläutert.

3.5.3 Redundanz und automatische Fehlerkorrektur

Bei physischen-Datendioden werden meist Fehlerkorrekturverfahren eingesetzt. So zum Beispiel auch bei der Fox-IT Datendiode. Die genauen Verfahren die zur Fehlerkorrektur eingesetzt werden, sind nicht bekannt. Es wird im Allgemeinen nur von Verfahren zur Fehlerkorrektur gesprochen. Dabei ist die Verwendung von sogenannten Vorwärtsfehlerkorrekturverfahren[FEC], wie zum Beispiel der Reed-Solomon-Algorithmus, denkbar. Bei diesen Verfahren werden Daten vom Sender so mit Redundanz kodiert, dass der Empfänger nach erkennen eines Übertragungsfehlers selbstständig die fehlerhaften Teile aus den restlichen Daten rekonstruieren kann.

Der Vorteil dieser Verfahren liegt auf dem Rückkanal, da auf diesen verzichtet wird. Diesem Vorteil stehen aber einige schwere Nachteile entgegen. Die künstliche Redundanz der übertragenen Informationen verlangsamt den Datenverkehr. Zudem sind diese Fehlerkorrekturverfahren nicht unfehlbar. Voraussetzung für die Wiederherstellung von korrupten Informationen ist, dass ein bestimmter Anteil der Informationen korrekt übertragen wurde. Wie groß dieser Anteil ist, kann über die Größe der zuvor gewählt Redundanz festgelegt werden. Dabei gilt: je größer die Redundanz, desto kleiner der Anteil an Daten, welche zur Rekonstruktion korrekt empfangen werden muss. Dabei gilt auch, je größer die Redundanz, desto kleiner der Durchsatz der Nutzdaten.

3.5.4 Bewertung

Alle zuvor beschriebenen Lösungen haben eines gemeinsam. Sie bestehen aus mehreren Hardware-Systemen und versuchen über ein unzuverlässiges Medium unidirektional zu kommunizieren. Dabei setzen einige Systeme, wie die genugate-Datendiode, auf einen Rückkanal um das Problem der Flusssteuerung sowie die Unzuverlässigkeit des Übertragungsmediums zu lösen. Andere Lösungen, wie etwa die meisten physischen Datendioden, versuchen mit geringer Bandbreite und viel Redundanz ohne Rückkanal diesem Problem entgegen zu treten.

Diese Arbeit löst dieses Problem, indem auf ein unzuverlässiges Kommunikationsmedium zwischen den Instanzen verzichtet wird. Die unidirektionale Verbindung zwischen der schwarze und roten Seite soll innerhalb eines Hosts-Systems stattfinden. Es muss dazu geklärt werden, wie eine sichere unidirektionale Kommunikation zwischen zwei von einer separierten Instanzen auf einem Host-System realisiert werden kann. Zur Separierung und Virtualisierung der schwarzen und roten Instanzen wird die Technik des L4-Mikrokernel genutzt. Die entwickelten Verfahren sind aber prinzipiell auch auf andere Virtualisierungstechniken anwendbar. Um die hier entwickelte Datendiode zur Außenwelt zugänglich zu machen, werden die unidirektionalen Relays der genugate-Datendiode verwendet. In den folgenden Abschnitten werden diese grundlegenden Techniken erläutert.

3.6 L4-Mikrokernel

L4 ist eine Mikrokernel-Familie, benannte nach dem Entwickler Jochen Liedtke. Diese Mikrokernel-Plattform ist in der Lage mehrere mittels Paravirtualisierung speziell angepasste Betriebssysteme isoliert von einander auf einer gemeinsamen Hardware zu betreiben. Zudem lassen sich auch native Anwendungen direkt auf der L4-Plattform ausführen. Diese Arbeit setzt auf einer L4-Implementierung names *Fiasco.OC* auf. *Fiasco.OC* ist eine in C++ implementierte Variante des L4-Mikrokernels. Die L4-Plattform entspricht nicht einem POSIX-artigen Design und ist daher nicht mit einem klassischem UNIX-Betriebssystem vergleichbar. Interprozesskommunikation und Programmierschnittstellen für native L4-Anwendungen (im weiteren L4-Tasks) haben daher starke Abweichungen zu POSIX-artigen Systemen.

3.7 L4OpenBSD

Die Portierung des OpenBSD-Betriebssystems auf die L4-Platform wird als *L4-OpenBSD* bezeichnet. Christian Ludwig portierte OpenBSD auf die L4-Plattform und beschrieb dieses in seiner Bachelor-Arbeit [Ludw11]. Diese Portierung wird von der Firma genua mbh weiterentwickelt und gepflegt. Sie wird benötigt um die, auf OpenBSD basierenden, Firewall- und VPN-Lösungen von genua virtualisiert betreiben zu können. Die in dieser Arbeit entwickelte Datendiode, soll später zwei L4-OpenBSD-Instanzen mit einander verbinden, um zwischen diesen beiden Instanzen einen sicheren unidirektionalen Datenverkehr zu ermöglichen. Die beiden L4-OpenBSD-Instanzen selbst sind mit der Außenwelt über reguläre Netzwerkschnittstellen verbunden.

3.8 L4-Interprozesskommunikation

Um basierend auf der L4-Plattform eine Datendioden zu implementieren, braucht man Möglichkeiten um Prozesse miteinander kommunizieren zu lassen. Dieser Abschnitt gibt eine Einführung in die verschiedenen Kommunikationsmittel der L4-Plattform sowie deren Vor- und Nachteile im Kontext der Entwicklung einer Datendiode.

Die L4-Platform stellt drei verschiedene Möglichkeiten zur Interprozesskommunikation (im weitern *IPC* abgekürzt) bereit. Jeder L4-Task hat eine Grundausstattung von sogenannten *Capabilities*, welche mit Teilen des L4-Kernels verbunden sind. Über diese kann ein Prozess mit dem Betriebssystem kommunizieren oder Ressourcen wie etwa Speicher oder weitere IPC-Medien anfordern. Diese *Capabilities* können ebenfalls zum direkten Austausch von Ressourcen zwischen verschiedenen L4-Tasks verwendet werden. Ob ein L4-Task mit einem anderen über ein IPC-Medium kommunizieren kann, wird vor dem Booten des Systems durch dessen Konfiguration festgelegt. Prozessen ist es zur Laufzeit nicht möglich, die in diesem Abschnitt beschrieben IPC-Medien zu nutzen, wenn diese vorher nicht explizit konfiguriert wurden.

3.8.1 Messages

Ein IPC-Mechanismus sind sogenannte *Messages*. Diese ermöglichen es Nachrichten zwischen zwei Prozessen per System-Call über den L4-Kernel auszutauschen. Größter Nachteil dieser Form der Kommunikation sind die blockierenden System-Calls. Der sendende Prozess wird solange blockiert, bis der empfangende Prozess die Nachricht komplett gelesen hat. Diese Eigenschaft sowie der Kontextwechsel in den Kernel-Kontext und zurück machen diese IPC-Art sehr langsam für größere Datenmengen. Dieses würde sich negativ auf den Durchsatz einer Datendiode auf Basis von *Messages* auswirken.

3.8.2 Shared-Memory

Mehrere L4-Tasks können auch ohne Interaktion des Kernels mittels *Shared-Memory* miteinander kommunizieren. Hierbei bekommen beide Prozesse einen gemeinsamen Speicherbereich zugewiesen. Die Synchronisation des Zugriffs übernehmen in diesem Fall die Prozesse selbst. Im Gegensatz zu den zuvor beschriebenen *Messages* ist diese Art der Kommunikation sehr schnell und kann ohne Kontextwechsel in den Kernel-Kontext ablaufen.

3.8.3 Interrupts

Software-*Interrupts* ist die letzte Art der Interprozesskommunikation zwischen L4-Tasks. Dabei wird für mehre L4-Tasks ein gemeinsamer IPC-Kanal definiert. Jeder beteiligte Prozess kann über diesen IPC-Kanal einen Interrupt bekommen. Anders als bei POSIX-artigen Systemen, muss ein L4-Prozess passiv mit einem blockierenden System-Call auf den Interrupt warten. Daher wird diese Art der Kommunikation meist in einen Thread ausgelagert.

3.8.4 Bewertung

Prinzipiell wäre alle IPC-Medien zur Kommunikation geeignet. Allerdings lassen sich mit *Messages* und *Interrupts* nur sehr langsam größere Datenmengen austauschen. Um eine praktisch benutzbare Datendiode zu entwickeln bedarf es einer schnellen Kommunikationsgrundla-

ge. Dafür ist das IPC-Medium *Shared-Memory* ideal geeignet. Im Bezug auf die Sicherheit einer Datendiode hat keines der benannten IPC-Medien einen entscheidenden Vorteil. In jedem Fall müsste zumindest die rückläufige Kommunikationsverbindung separat von den Kommunikationsparteien kontrolliert werden.

3.9 Unidirektionale Kommunikation auf einem Host-System

Dieses Kapitel beleuchtet die Möglichkeiten, mit denen eine unidirektionale Kommunikation auf einem Host-System realisiert werden kann. Dabei werden die Vor- und Nachteile der verschiedenen Aufbauten abgewogen, um eine sichere und gleichzeitig performante Lösung zu finden. Die vorgestellten Varianten sind dabei nicht exklusiv auf die L4-Plattform beschränkt. Alle Varianten sind auch auf andere Plattformen anwendbar.

3.9.1 Speicherzugriffsrechte

Eine Variante zwei Prozesse auf einem System unidirektional zu verbinden, könnte wie folgt über eine Shared-Memory-Seite realisiert werden. Dabei wird dem schwarzen Prozess der Speicher mit Lese-Schreib-Berechtigung in seinen virtuellen Speicher gemappt, dem roten Prozess hingegen ausschließlich mit Lese-Berechtigung.

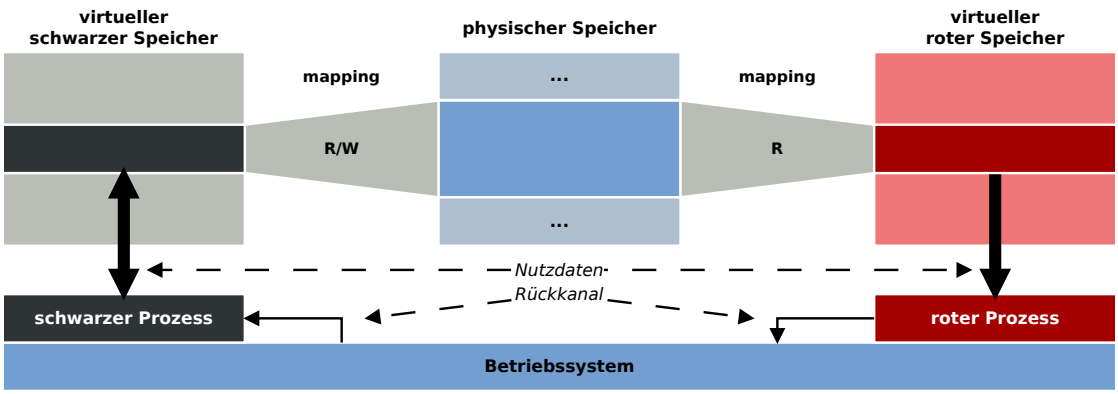

Abbildung 15: Schwarz-SHM-Rot

Vorteil dieser Methode ist die Übertragungsgeschwindigkeit. Da beim

Transfer der Informationen von einer zum anderen Instanz nur eine Kopier-Operation ausgeführt werden muss.

Diese Methode hat den Nachteil, dass die Synchronisation der Kommunikation nicht über das Shared-Memory erfolgen kann. Hierfür müsste ein weiter IPC-Kanal benutzt werden. Etwa *Interrupts* oder *Messages*, die einander über den Lese- oder Schreib-Status informieren.

Ein weiterer und schwerwiegenderer Nachteil ist der Umstand, dass nicht kontrolliert werden kann, ob die Statusberichte der roten Seite nicht zur ungewollten Kommunikation missbraucht werden. Da eine Kontrollinstanz seitens des Betriebssystems keinen Einblick in den Ablauf der Shared-Memory-Kommunikation hat.

3.9.2 Dioden-Prozess

Um die Gesamtkommunikation besser kontrollieren zu können, bietet sich eine Variante an, in der der Kopiervorgang durch einen Prozess auf Betriebssystemseite geschieht.

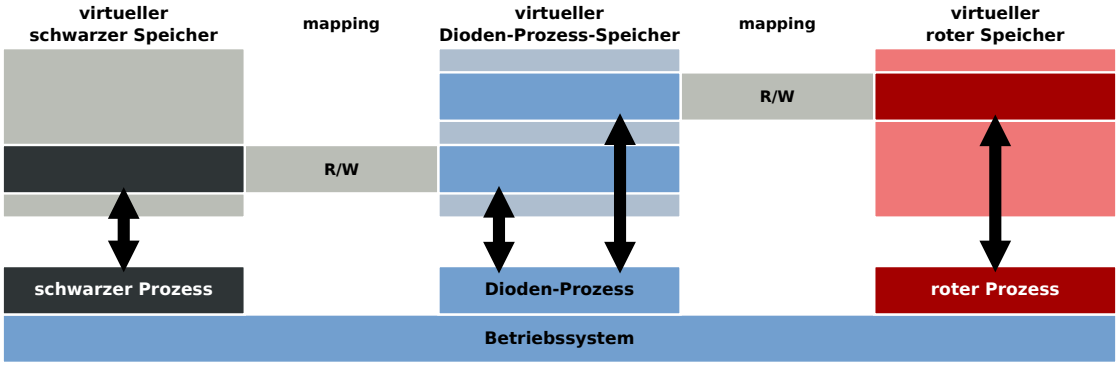

Abbildung 16: Kommunikation über einen Dioden-Prozess

In dieser Variante gibt es zur Kommunikation zwischen schwarz und rot zwei getrennte Speicherbereiche. Diese beiden Speicherbereiche werden mit Lese-Scheib-Berechtigung in den virtuellen Speicher eines neutralen Dioden-Prozesses gemappt. Dieser Dioden-Prozess kann nun die gesamte Kommunikation zwischen schwarz und rot kontrollieren und die zu übertragenen Informationen zwischen den beiden Speicherbereichen kopieren. Somit lässt sich der Rückkanal besser mit den übertragenen Nutzdaten koordinieren. Zudem muss die Flusssteuerung der Datenübertragung nicht mehr zwischen der schwarzen

und roten Seite direkt geschehen, sondern nur noch mit dem Dioden-Prozess. Damit kann dieser Teil des Rückkanals ebenfalls vom Dioden-Prozess kontrolliert werden.

4 Analyse bestehender unidirektionaler Kommunikationsmechanismen

Dieses Kapitel vergleicht die Kommunikationsmechanismen der NRL-Pump mit denen der genugate-Datendiode. Diese beiden bestehenden Lösungen sind zum Zeitpunkt dieser Arbeit die einzigen logischen Datendioden, zu denen Unterlagen über deren Funktionsweise zugänglich sind. Die Protokolle der Dioden werden im Hinblick auf die Rückmeldungen und deren Vor- und Nachteile beleuchtet.

4.1 Protokoll der NRL-Pump

Dieser Abschnitt erläutert die interen Funktionsweise der NRL-Pump. Der Protokollablauf ist hauptsächlich dem Dokument „A Data Pump for Communication“ [KM95] entnommen.

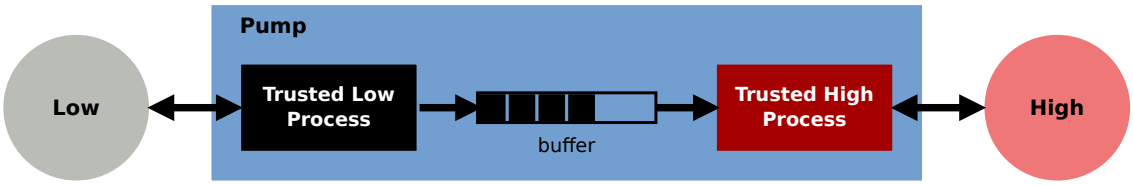

Abbildung 17: Interner Kommunikationsablauf der NRL-Pump

Innerhalb der NRL-Pump gibt es zwei Prozesse, welche als sicher angenommen werden, die den Zugriff auf den interen Puffer haben. Diese Prozesse sind Vermittler zwischen den externen Hosts im schwarzen sowie roten Netz und dem Puffer. Wie in Abbildung 17 dargestellt, kommuniziert der *Trusted-Low-Process* (TLP) mit den Hosts im schwarzen Netz und der *Trusted-High-Process* (THP) mit den Hosts im roten Netz.

Um nun Daten von *Low* nach *High* zu übertragen, verschickt *Low* einzelne Pakete an den *TLP*. Nach dem Verschicken eines jeden Pakets wartet *Low* auf eine positive Empfangsbestätigung bevor es das nächste Paket an den *TLP* verschickt. Bekommt *Low* eine negative Empfangsbestätigung, oder kommt bis zum Timeout gar keine Empfangsbestätigung, versucht es das entsprechende Paket erneut zu senden.

Der *TLP* nimmt Pakete von *Low* entgegen und versucht diese in den internen Puffer zu schreiben. Sobald der *TLP* das Paket in den Puffer schreiben konnte, sendet es eine positive Empfangsbestätigung an *Low* zurück. Ist es dem *TLP* nicht möglich das Paket in dem internen Puffer abzulegen, versendet er eine negative Empfangsbestätigung an *Low*, um zu signalisieren, dass dieses Paket *High* nicht erreichen wird.

Der *THP* entnimmt die Pakete, welche zuvor vom *TLP* hinterlegt worden sind, aus dem internen Puffer und sendet diese zu *High*. Ebenso wie *Low*, wartet er nun auf eine positive Empfangsbestätigung für das versendete Paket. Wird der Empfang von *High* positiv bestätigt, versendet der *THP* das nächste Paket aus dem internen Puffer. Bekommt der *THP* eine native Bestätigung von *High* oder läuft der Timeout ab, wird der *THP* versuchen, das Paket erneut zu versenden.

High empfängt ausschließlich Pakete vom *THP* und bestätigt deren Empfang.

Dieses Verfahren hat den Vorteil, dass die rote Seite keinen Einfluss auf die Empfangsbestätigungen nehmen kann, welche an die schwarze Seite verschickt werden, da die Empfangsbestätigungen nicht direkt an den Empfang der roten Seite gebunden sind, sondern nur an den Empfang an der Diode selbst. Die Empfangsbestätigungen von *High* an die NRL-Pump sind somit von denen, welche die NRL-Pump an *Low* sendet entkoppelt.

4.2 Protokoll der genugate-Datendioden

Im Gegensatz zur NRL-Pump sind die unidirektionalen Protokoll-Relays bei der genugate-Datendiode Teil der Diode selbst und nicht auf den Hosts des schwarzen und roten Netzes.

Diese Relays verhalten sich nach außen wie ein normaler Client oder Server, kommunizieren aber untereinander mit einer einseitigen TCP-Verbindung. Für die beiden Anwendungsprotokolle gibt es jeweils ein Relay, welches als Server agiert und Protokolldaten in eine TCP-Verbindung einpackt. Zudem gibt es eine Gegenseite, welche die Protokolldaten aus der TCP-Verbindung entnimmt und zum Ziel-Server als Client agiert. Für SMTP sind es das SMTP2TCP- und TCP2SMTP-Relay, für FTP dementsprechend das FTP2TCP- und TCP2FTP-Relay.

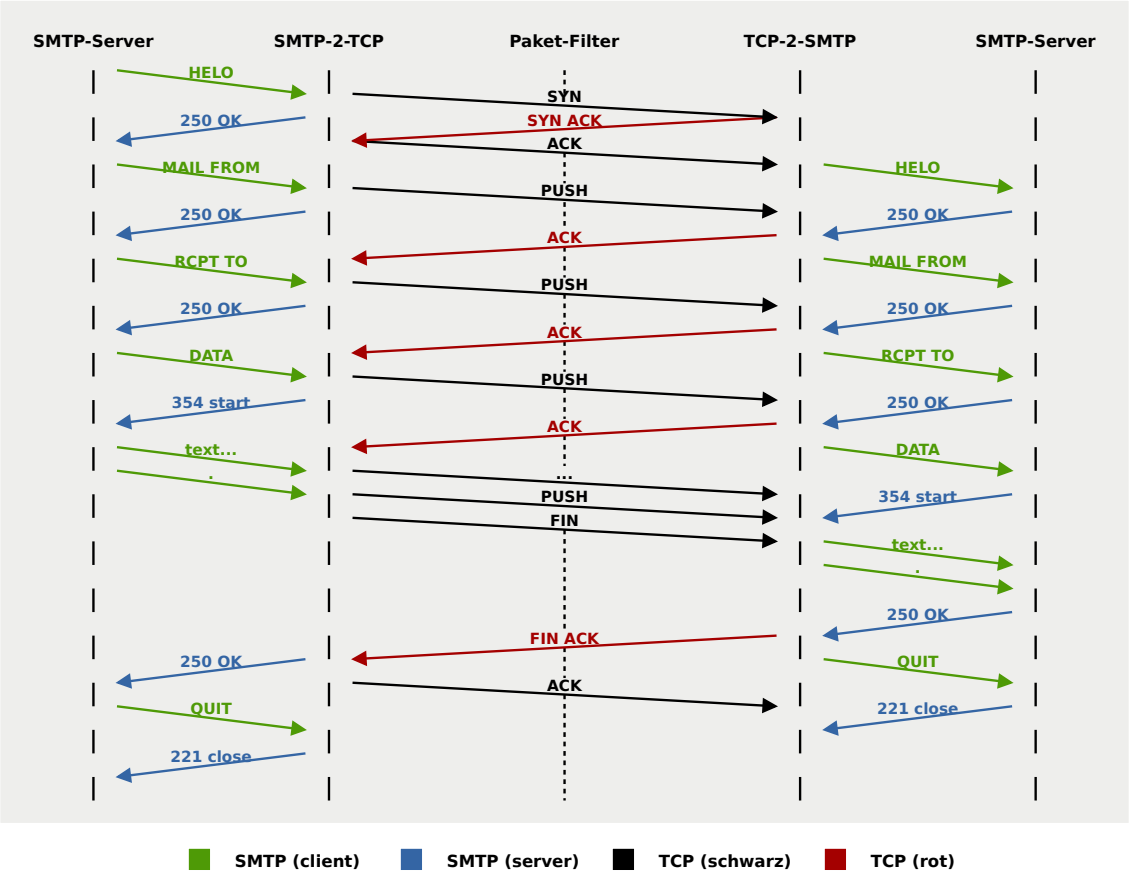

Abbildung 18: SMTP-Dioden-Relay

Abbildung 18 zeigt in einem Sequenzdiagramm den gesamten Protokollablauf beim der Übertragung einer E-Mail von schwarz nach rot.

Die Unidirektionalität wird bei diesem Konzept vom Paketfilter sichergestellt. Der Paketfilter ist eine speziell angepasste OpenBSD-Firewall. Dieser hat eine Konfiguration, welche von der roten Seite ausschließlich Acknowledgment-Pakte weiterleitet. Andere Pakete von der roten Seite werden verworfen. Zudem werden die Felder des TCP-Header geprüft und bei Unregelmäßigkeiten die Pakete verworfen. Dadurch werden die Freiheitsgrade der roten Seite in ihrer Kommunikation und der daraus resultierende Rückkanal soweit wie möglich eingeschränkt. Die genaue Funktionsweise des Paketfilters kann dem Manual `pf_oneway(9)` [Bluhm07] entnommen werden.

4.3 Bewertung

Die NRL-Pump hat den Vorteil der Entkopplung der Empfangsbestätigung von schwarzer und roter Seite. Daher hat die rote Seite keinen Einfluss auf die Empfangsbestätigungen, welche an die schwarze Seite verschickt werden. Da die NRL-Pump der schwarzen Seite nur den Empfang der Daten auf der Datendiode selbst bestätigt, ist die Datendiode selbst auch dafür verantwortlich das diese Daten in jedem Fall bei roten Seite ankommen. Die NRL-Pump stellt somit nur sicher, dass die Übertragenen Daten auf der anderen roten Seite ankommen.

Die genugate-Datendiode verfolgt an dieser Stelle einen anderen Ansatz. Bei dieser Diode bedeuten die Empfangsbestätigungen nicht nur, dass Daten angekommen sind, sonder auch, dass diese Verstanden wurden. Die unidirektionalen Protokoll-Relays der genugate-Datendiode geben nur dann eine positive Rückmeldung, wenn beispielsweise die versendete E-Mail auch zugestellt werden konnte. Sollte die Authentifizierung im SMTP-Protokoll fehlschlagen oder der Zielserver nicht erreichbar sein, wird dieses mit einer negativen Rückmeldung quittiert.

Dadurch entsteht der Nachteil, dass die rote Seite einen zeitlichen Einfluss auf die Empfangsbestätigungen nehmen kann. Auf der anderen Seite ist der gesamte Ablauf der Kommunikation zuverlässiger im Vergleich zur NRL-Pump, da die schwarze Seite nicht nur erfährt ob ihre Nachricht angekommen ist, sonder auch, ob diese auf Anwendungsebene verstanden wurde.

5 L4-Pipe-Kommunikation

Im Abschnitt 3.9 wurden zuvor mehre Möglichkeiten erläutert unidirektional auf einem Host-System zu kommunizieren. Nach Abwägung der Vor- und Nachteile ist beste Variante eine Shared-Memory-Kommunikation über einen sicheren Dioden-Task, welche die Kommunikation überwachen kann. Dieser Abschnitt beschreibt einen von genua entwickelten Kommunikationskanal für die L4-Plattform welcher das Shared-Memory-Medium nutzt. Aufbauend auf dieser Vorarbeit wird dann der Dioden-Task entwickelt.

5.1 L4-Pipe

Für eine einfache Kommunikation zwischen verschiedenen *L4OpenBSD*-Instanzen wurde von genua ein bidirektionaler Datenkanal namens *L4-Pipe* geschaffen. Dieser verbindet die *L4OpenBSD*-Instanzen über Shared-Memory miteinander.

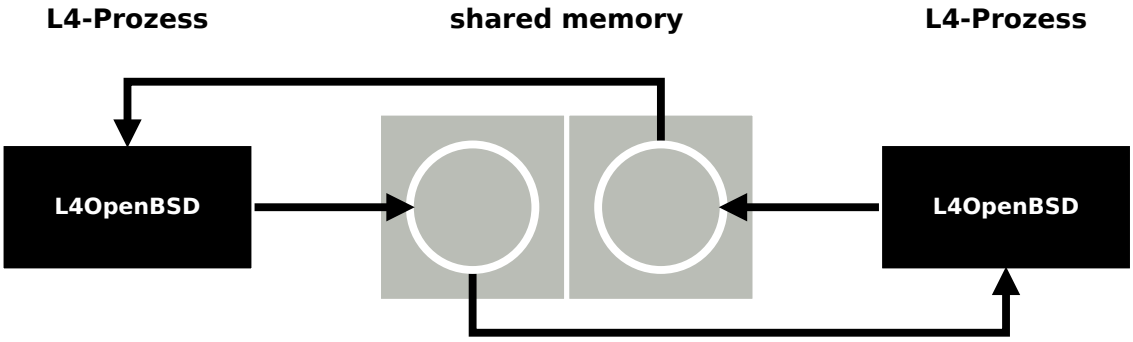

Abbildung 19: Kommunikation über Ringpuffer innerhalb der *L4-Pipe*

Kernstück dieses Kommunikationskanals sind zwei Ringpuffer, über die jeweils die Hin- sowie Rückrichtung ablaufen. Der gemeinsame Shared-Memory-Bereich wird dafür zwischen den beiden Ringpuffern aufgeteilt. Verwaltet wird der Shared-Memory-Bereich innerhalb des L4OpenBSD-Kernels. Dieser Kernel-Treiber stellt dem Userland eine Schnittstelle in Form einer `/dev/l4pipe0` Gerätedatei bereit. Über diese Gerätedatei können Prozesse zwischen zwei *L4OpenBSD*-Instanzen über die *L4-Pipe* miteinander kommunizieren. Dafür stehen ihnen die System-Calls `open(2)`, `read(2)`, `write(2)` und `close(2)` zur Verfügung.

5.2 Paket-Protokoll der L4-Pipe

Um über die *L4-Pipe* strukturiert kommunizieren zu können, wurde ein paketbasiertes Protokoll von genua entwickelt. Dieses erlaubt es Datenströme strukturiert, mit einem definierten Beginn und Ende, sowie einer Empfangsbestätigung ablaufen zu lassen. Damit ist es den Prozessen im Userland des L4OpenBSD ebenfalls möglich mehrere Datenströme parallel zu übertragen. Die Struktur der Pakete sowie der Ablauf der Kommunikation soll in diesem Abschnitt verdeutlicht werden, um das Protokoll später auf seine Tauglichkeit für den unidirektionalen Einsatz untersuchen zu können.

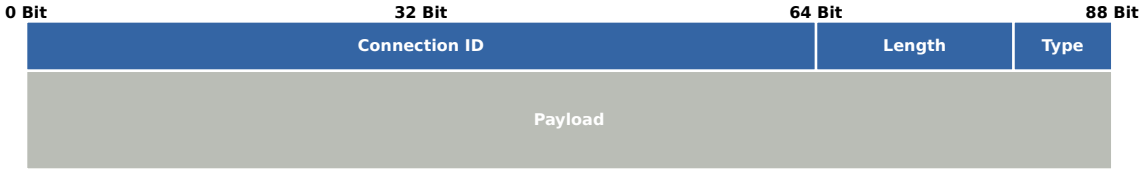

Abbildung 20: Paketaufbau der L4-Pipe

Jedes Paket beginnt mit einem 88 Bit langen Header. Dieser beinhaltet die drei Felder *ID*, *Length* und *Type*.

Mit dem 64 Bit langen *ID*-Feld, werden verschiedener parallele Verbindungen unterschieden. Da das Protokoll vorsieht, das *ID*-Feld für jede neue Verbindung zu inkrementieren und keine *ID* wiederzuverwenden wurde das Feld auf die Länge von 64 Bit festgelegt. Es wird dabei davon ausgegangen, dass die Obergrenze von 2^{64} Verbindungen im realen Einsatz nie erreicht wird.

Das *Length*-Feld gibt die Länge der Payload in Bytes an. Da dieses Feld auf 16 Bit begrenzt ist, ist auch die Payload auf maximal 64 KiB begrenzt.

Mit dem *Type*-Feld werden verschiedene Arten von Paketen unterschieden. Es wurden insgesamt fünf Pakettypen definiert: *SYN* (synchronization), *DATA*, *EOF* (end of file), *FIN* (finish) und *RST* (reset).

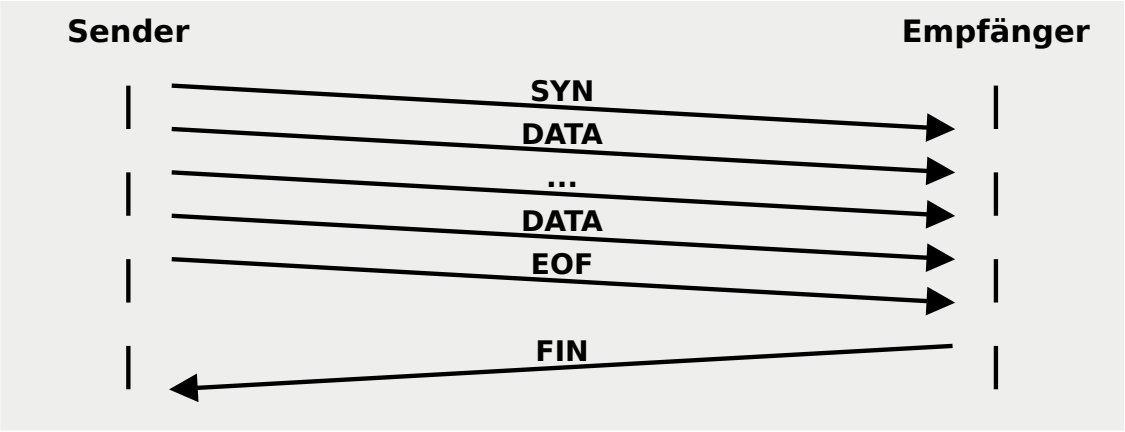

Abbildung 21: Sequenzdiagramm zum Protokoll

Der Sender beginnt eine Übertragung mit einem *Syn*-Paket. Dieses Paket legt die *ID* für diese Verbindung fest, welche in allen weiteren Paketen verwendet wird. Auf das *Syn*-Paket folgend, verschickt der Sender beliebig viele *DATA*-Pakete mit den zu versendenden Daten. Diese *DATA*-Pakete sind die einzigen Pakete die eine Payload enthalten. In allen anderen Pakettypen ist die Payload leer und das *Length*-Feld auf Null gesetzt. Um eine Übertragung abzuschließen, verschickt der Sender ein *EOF*-Paket und wartet auf die Bestätigung des Empfängers in Form eines *FIN*- oder *RST*-Paketes. Mit dem *FIN*-Paket signalisiert der Empfänger, dass alle Daten empfangen wurden. Tritt ein Fehler während der Übertragung auf, kann der Sender sowie der Empfänger die Verbindung jederzeit mit einem *RST*-Paket abbrechen.

5.3 TCP-Multiplexer

Um diesen Kommunikationskanal für bestehende Anwendungen nutzbar zu machen, wurde das Programm `l4pipemux` von genua entwickelt. Dieses Programm multiplext verschiedene parallele TCP-Verbindungen über die *L4-Pipe*. Die Schnittstelle zum Userland findet dabei über einen TCP/IP- oder UNIX-Domain-Socket statt.

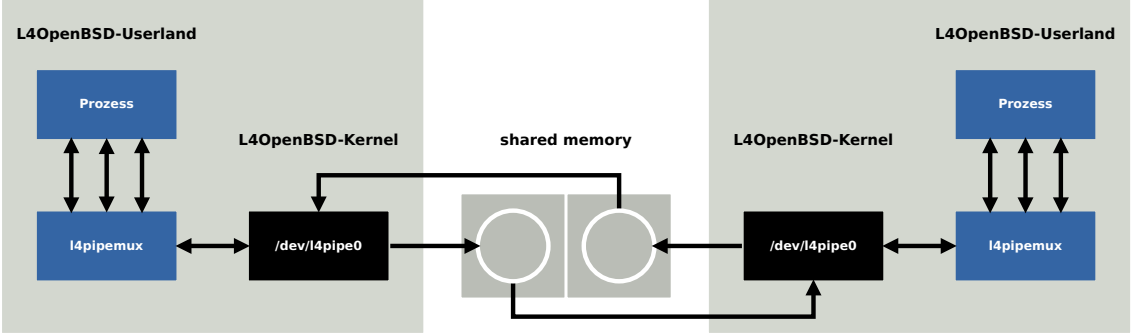

Abbildung 22: Kommunikation über `l4pipemux`

5.4 L4-Pipe-Filter

Um den Datenverkehr zwischen zwei *L4OpenBSD*-Instanzen kontrollieren zu können, wurde von genua ein Filterprogramm namens *SHM-Proxy* entwickelt. Dieser Filter ist ein nativer *L4-Task* und läuft direkt auf der L4-Plattform ohne eine weitere Umgebung wie etwa ein *L4-OpenBSD* oder ein *L4Linux*. Der *SHM-Proxy* bietet die Möglichkeit den Ablauf dem im Abschnitt 5.2 beschriebenen *L4-Pipe*-Protokolls zu kontrollieren und durchzusetzen. Dabei kann er einzelne Pakete gezielt aus dem Paketstrom herausfiltern oder auch ganze Verbindungen terminieren.

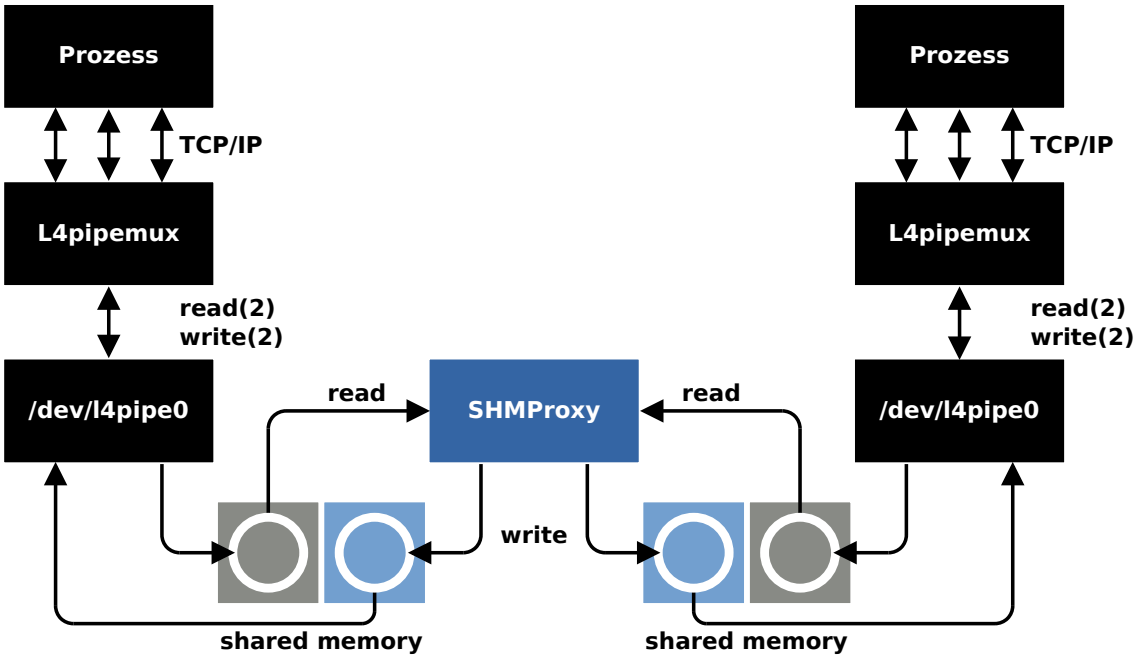

Abbildung 23: Kommunikationskanal mit *SHM-Proxy*

Damit der *SHM-Proxy* den Datenverkehr kontrollieren kann, wird er über Shared-Memory mit den beiden *L4-OpenBSD*-Instanzen verbunden.

5.5 Schwächen im Unidirektionalen-Einsatz

Will man diese bidirektionale Kommunikationsstrecke in eine eine unidirektionale umformen, scheint dies zunächst recht einfach. Der SHM-Proxy filtert einseitig alle Pakete zum Aufbau sowie zur Übertragung von Daten heraus und lässt ausschließlich *FIN*- und *RST*-Pakete zur Bestätigung zu.

Bei genauerer Betrachtung fällt auf, dass dieses Verhalten nicht ausreicht. Im Abschnitt 6 werden Möglichkeiten beschrieben, diese einfache Filterung zu umgehen, um dennoch Daten von rot nach schwarz zu übertragen.

Konzeptionell hat das Protokoll dabei eine weitere Schwäche. Da der Paket-Header aus Abschnitt 5.2 keine Adressinformation enthält, kann über eine *L4-Pipe* nur ein TCP-Service erreicht werden. Will man mehrere Services über eine *L4-Pipe* erreiche, müsste man mit diesem Ansatz mehre *L4-Pipes* parallel betreiben.

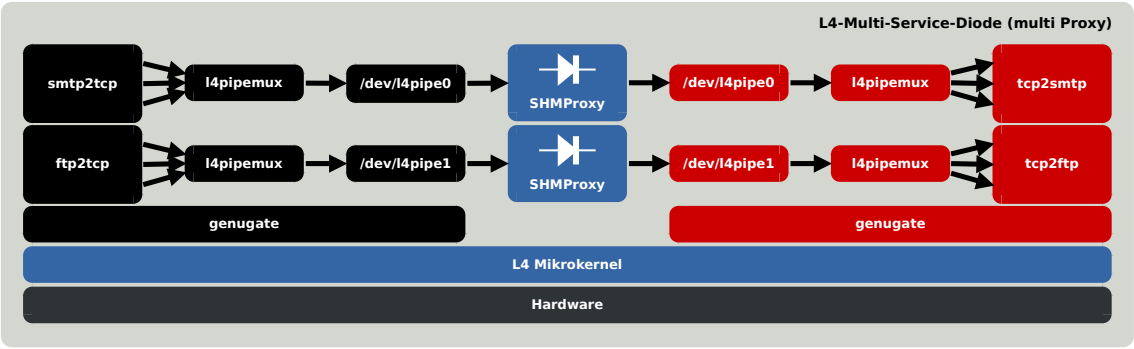

Abbildung 24: Multi-Service Kommunikationskanal mit mehreren *SHM-Proxy*

Für jeden Service, der von schwarz aus auf der roten Seite erreicht werden soll, müsste eine eigene Pipe, sowie ein eigener Dioden-Task gestartet werden. Der Rückkanal, den eine einzelne Verbindung mit sich bringt, würde sich in diesem Szenario mit der Anzahl der Verbindungen mindestens multiplizieren. Daher muss das Protokoll vor der Verwendung in einer Datendiode um eine Möglichkeit zur Adressierung erweitert werden. Damit ein einzelner Proxy die Verbindungen mehrerer Dienste parallel verwalten und kontrollieren kann.

6 Analyse bekannter Rückkanalmöglichkeiten

In diesem Kapitel werden die verschiedenen Möglichkeiten eines Rückkanals und ihre Größe analysiert. Zunächst soll dabei eine theoretische Analyse Aufschluss über das maximale Ausmaß des Rückkanals geben. Dabei wird davon ausgegangen, dass der Angreifer die absolute Kontrolle über die L4OpenBSD-Instanzen der Datendiode hat (siehe Abschnitt 3.3.3). Die Sicherheit des in dieser Arbeit entwickelten Datendioden-Protokolls stützt sich auf die Annahme, dass die L4-Plattform sowie der Dioden-Prozess nicht kompromittiert werden. Daran anschließend werden Verfahren erläutert, welche die theoretischen Möglichkeiten praktisch umsetzten. Diese Demonstration soll verdeutlichen, welche Ausmaße die theoretischen Möglichkeiten praktisch haben. Die praktischen Kodierungsmöglichkeiten werden nur einzeln und nicht kombiniert erläutert. Eine Analyse der Kombinationsmöglichkeiten finden nur im theoretischen Teil statt, da eine praktische Umsetzung den zeitlichen Rahmen dieser Arbeit überschreiten würde.

6.1 Impliziter Rückkanal

Will man eine grobe obere Abschätzung der Rückkanalmöglichkeiten treffen, müssen zunächst die verschiedenen Freiheitsgrade der Kommunikation analysiert werden.

Der erste und offensichtlichste Freiheitsgrad ist die Empfangsbestätigung einer Verbindung, die rot an schwarz zurücksendet. Direkt kann die rote Seite auf eine Verbindung mit einer positiven oder negativen Rückmeldung reagieren. Indirekt besteht die Möglichkeit gar nicht zu reagieren und die Verbindung in einen Timeout laufen zu lassen. Somit bleiben der roten Seite drei Arten auf eine Verbindung mittels Empfangsbestätigungen zu reagieren, welche von der schwarzen Seite erkannt werden kann. Die Reaktion der rote Seite mittels Timeout klingt zunächst zwar nur theoretisch, da ein Timeout im Verhältnis zur Übertragung einer Nachricht wesentlich mehr Zeit benötigt und dessen Einsatz als ineffizient erscheinen lässt. Bei genauer Betrachtung stellt sich heraus, dass sobald die schwarze Seite die übliche Reaktionszeit von

rot kennt, kann die schwarze Seite bereits auf einen Timeout schließen, sobald der Zeitraum einer direkten Reaktion überschritten wurde. Somit lässt sich ein Timeout effizient nutzen, ohne an dieser Stelle ineffizient zu werden.

Um die Rate von Bits pro Verbindung für einen Freiheitsgrad zu berechnen, bildet man den Logarithmus zur Basis zwei von den zur Verfügung stehenden Reaktionsmöglichkeiten. Somit erhält man beispielsweise für 256 Möglichkeiten 8 Bit, denn $log_2(256) = 8Bit$.

Für den Freiheitsgrad der Empfangsbestätigung ergibt sich somit eine Übertragungsrate von $log_2(3) = 1.58Bit/Verbindung$ die der roten Seite zur Verfügung steht, um Daten an die schwarze Seite zu übertragen.

Ein anderer Freiheitsgrad ist die Zeit zwischen dem Beenden einer Verbindung der schwarzen Seite und der Reaktion darauf. Der roten Seite steht es frei zu einem beliebigen Zeitpunkt innerhalb der Timeout-Zeitspanne eine Empfangsbestätigung zu verschicken. Die Variable T sei an dieser Stelle die Zeitspanne des Timeout einer Verbindung und die Variable t das kleinste Zeitintervall, welches von der schwarzen Seite noch sicher unterschieden werden kann. Dann ergibt sich aus T/t die Anzahl der Reaktionsmöglichkeiten und aus $log_2(T/t)$ die Übertragungsrate von Bits pro Verbindung, welche durch den Freiheitsgrad der Reaktionszeit entsteht.

Durch dem Umstand der parallelen Kommunikation kommt ein Freiheitsgrad hinzu, welcher es der roten Seite gestattet, die Reihenfolge in der sie auf die Verbindungen reagiert selbst zu bestimmen. Die Reaktionsmöglichkeiten der roten Seite bildet sich aus der Fakultät der parallel möglichen Verbindungen. Aus der Kombinatorik ist bekannt, dass die Fakultät einer Anzahl von Dingen, die Möglichkeiten der Anordnung dieser Menge von Dingen angibt. Somit ergibt sich die Formel $log_2(v!)$, mit der sich die Übertragungsrate dieses Freiheitsgrades berechnen lässt. Die Variable v entspricht dabei der Anzahl der Verbindungen.

Durch eine Kombination der verschiedenen Freiheitsgrade lässt sich die Bitrate um ein vielfaches steigern. Praktisch lassen sich nicht alle Freiheitsgrade direkt miteinander kombinieren und gleichzeitig die Übertragungsrate steigern, da sie sich gegenseitig beeinflussen. So lässt sich beispielsweise der Status einer Empfangsbestätigung in Form eines Timeouts nicht mit dem Zeitpunkt der Reaktionszeit vereinbaren.

Somit lassen sich in diesem Beispiel nur zwei der drei Zustände der Empfangsbestätigung mit einer beliebigen Reaktionszeit kombinieren.

$log_2(3) * v + log_2(T/t) * v + log_2(v!) = log_2((3T/t)^v * v!)$

Der folgende Graph zeichnet die Bitrate über die Anzahl der Verbindungen einzeln und in Summe ab.

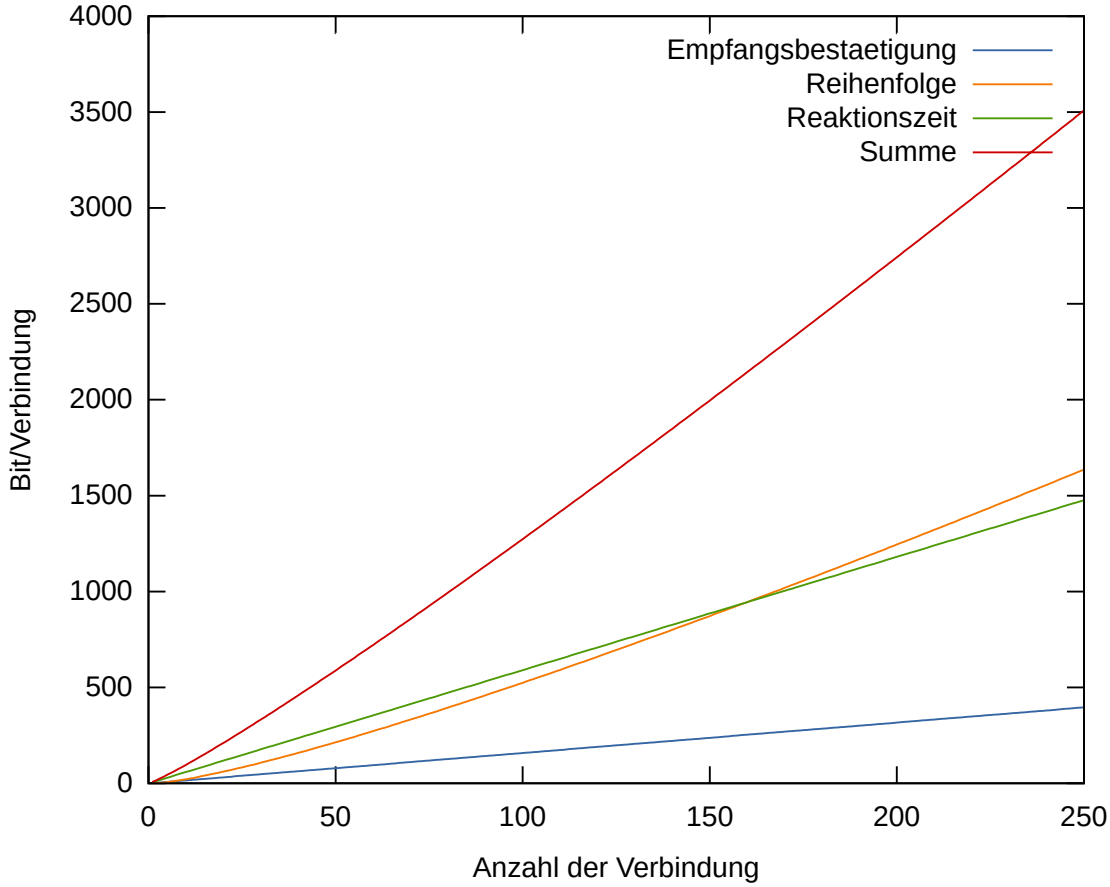

Abbildung 25: Theoretischer Rückkanal im Verhältnis zu Parallelen Verbindungen

Um die theoretischen Möglichkeiten aus dem Abschnitt 6.1 auch praktisch nachvollziehen zu können, sollen in den folgenden Abschnitten konkrete Angriffe auf die Freiheitsgrade aufgezeigt werden. Die gezeigten Verfahren sind dabei nicht auf möglichst hohe Übertragungsraten optimiert, sondern prinzipieller Natur.

6.2 Answer-Bit

Das *Answer-Bit*-Verfahren nutzt den Freiheitsgrad der Empfangsbestätigung aus. Dieses ist ein sehr prinzipielles Verfahren, welches sich bei logischen Datendioden mit einem Rückkanal für Empfangsbestätigungen immer anwenden lässt. Eine Ausnahme stellt hier die NRL-Pump dar, da diese einen entkoppelten Rückkanal hat. Daher lässt sich dieses Verfahren bei dieser nicht anwenden.

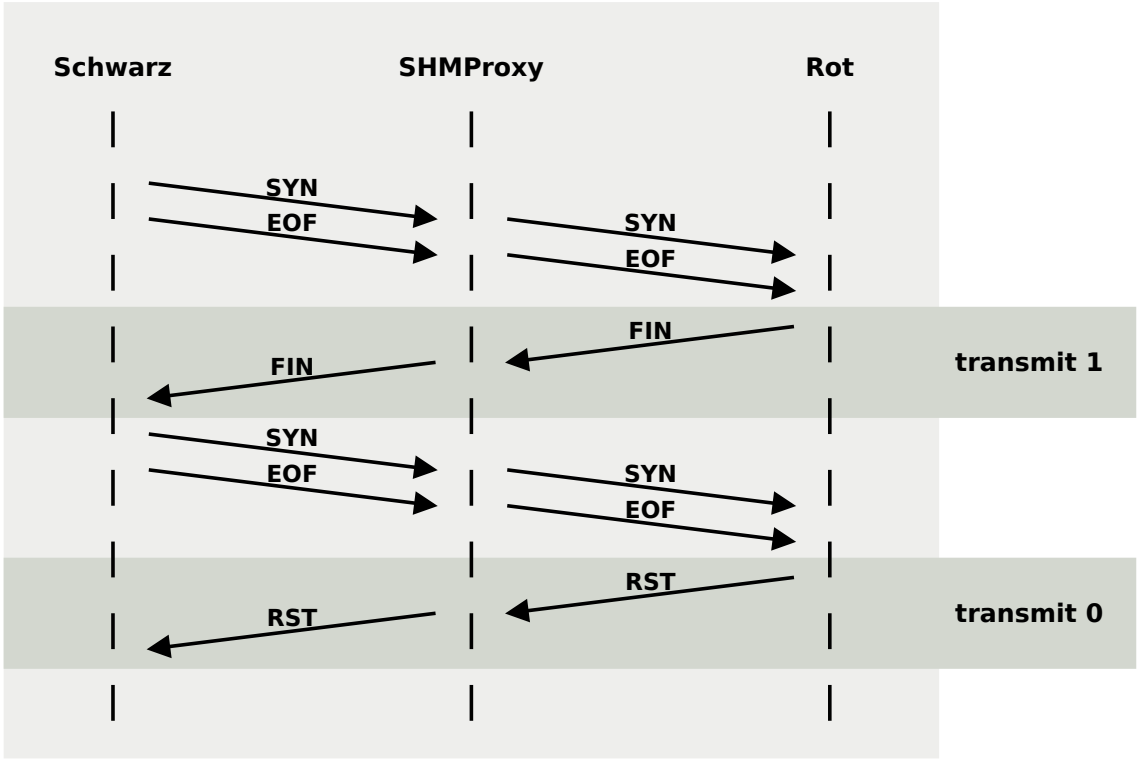

Abbildung 26: Sequenzdiagramm der Answer-Bit-Methode

Bei diesem Verfahren sendet die rote Seite immer wieder ein *SYN*- und ein *FIN*-Paket und wartet auf das Antwortpaket. Die rote Seite beantwortet diese Verbindungen nun mit einem *ACK*- oder *NAC*-Paket, abhängig davon ob es eine Eins oder eine Null übertragen möchte. Auf der schwarzen Seite werden die Antwortpakete wieder zu einer Nachricht zusammengesetzt. Um dieses Verfahren auch praktisch zu demonstrieren wurde das Programm `rpipe_ab` im Rahmen dieser Arbeit entwickelt. Dieses Programm implementiert das *Answer-Bit*-Verfahren und wurde erfolgreich eingesetzt.

6.3 Multi-Bit-Choosing

Beim *Multi-Bit-Choosing*-Verfahren nutzt die rote Seite den Freiheitsgrad der Parallelität aus. Das bedeute im Allgemeinen, dass die schwarze Seite eine Wahlmöglichkeit für die rote Seite schaft. Immer wenn dieser Zustand eintritt, kann die rote Seite über ihr Auswahl der schwarzen Seite eine Information mitteilen.

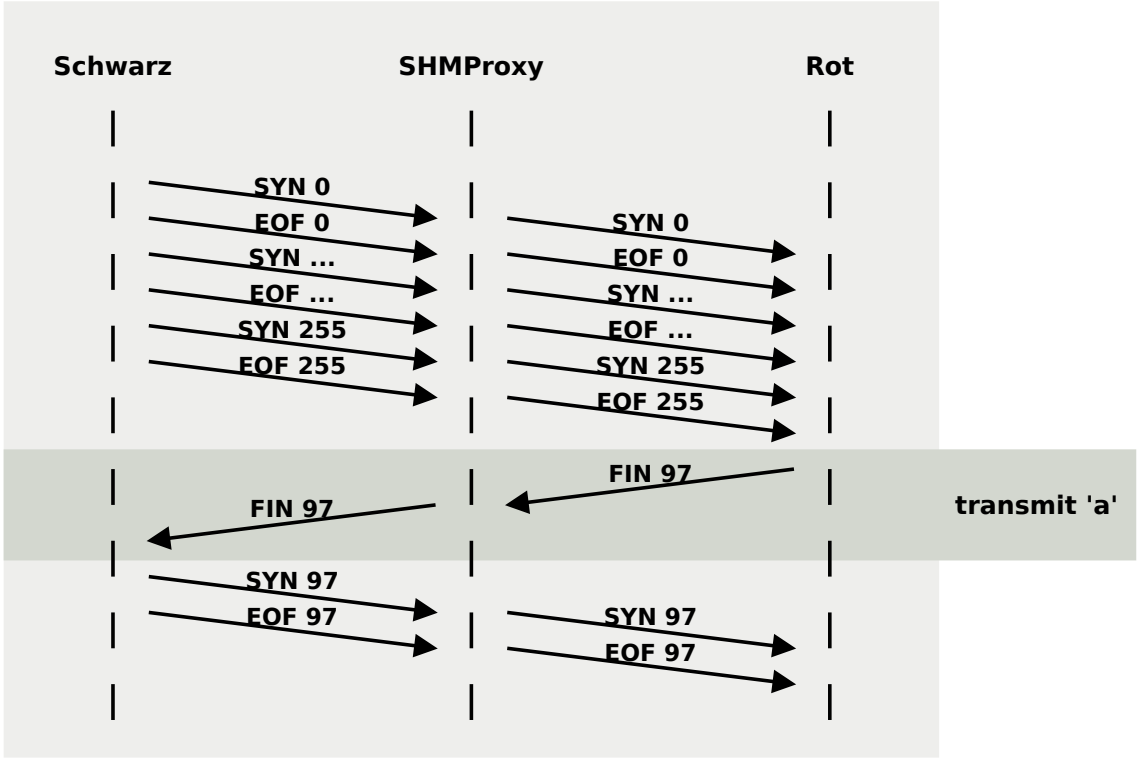

Abbildung 27: Sequenzdiagramm der Multi-Bit-Choosing-Methode

Abbildung 27 demonstriert dieses Verfahren an einem konkreten Beispiel. Die schwarze Seite öffnet und schließt mehrere Verbindungen parallel. Bestätigt die rote Seite nun eine Verbindung wird diese sofort wieder geöffnet und geschlossen, sodass der roten Seite immer eine feste Anzahl von Verbindungen zur Verfügung steht. Die rote Seite bestätigt nun die Verbindungen, die eine bestimmte Bit-Folge repräsentieren.

Im Beispiel hält die schwarze Seite immer 256 parallele Verbindungen offen. Die *Connection-ID* repräsentiert dabei ein Byte, welches die rote Seite wählen kann. Die rote Seite bestätigt nun die Verbindungen,

welche den Bytes entspricht, die zu übertragen sind. Auch dieses Verfahren wurde im Rahmen dieser Arbeit in dem Programm `rpipe_mbc` implementiert und erfolgreich getestet.

6.4 ACK-Ordering

Beim *ACK-Ordering* nutzt der Angreifer den Freiheitsgrad aus, dass er aus verschiedenen parallel Verbindungen wählen kann, in welcher Reihenfolge er die Verbindungen bestätigt.

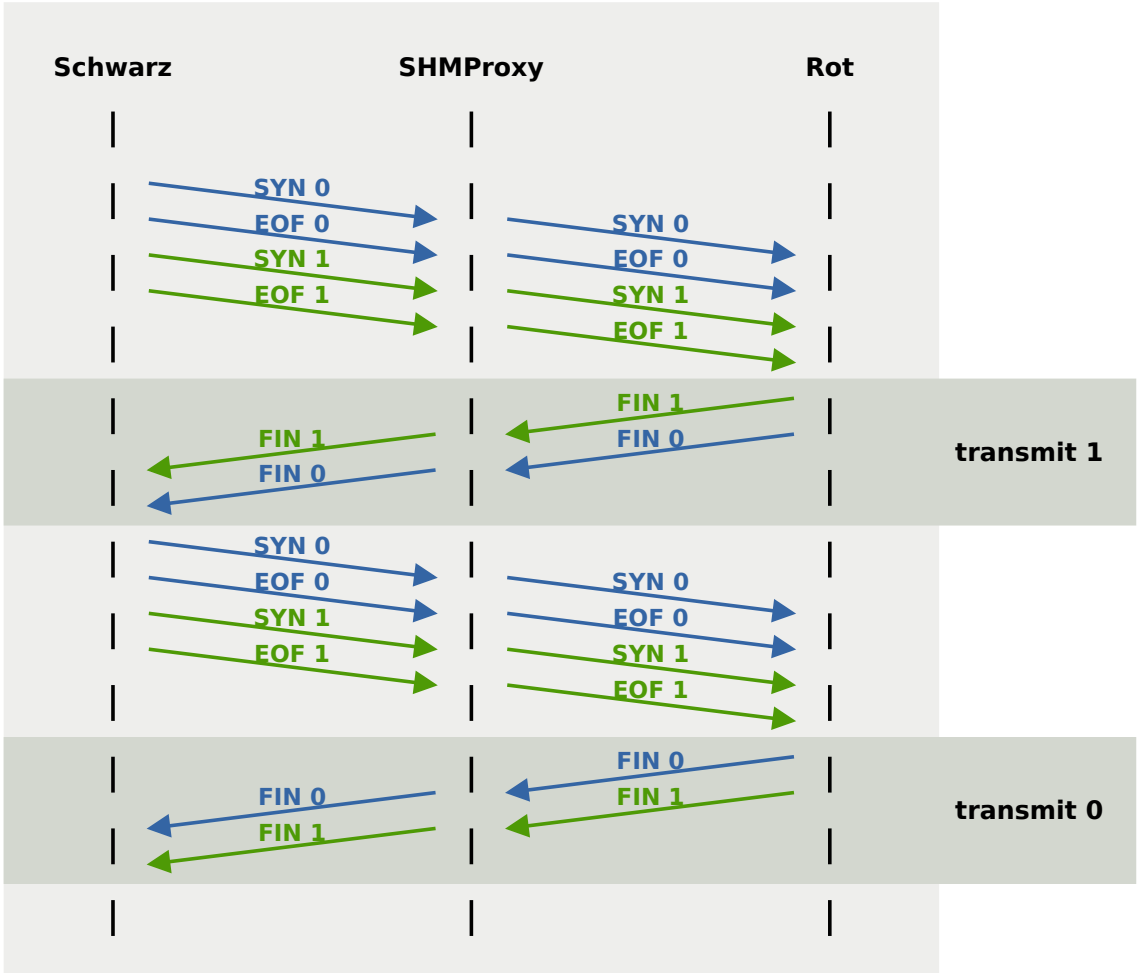

Abbildung 28: Sequenzdiagramm des Acknowledgement-Ordering

Im einfachsten Fall, öffnet die schwarze Seite zwei Kanäle und wartet auf deren Bestätigung durch die rote Seite. Behält die rote Seite beim Bestätigen der Verbindung die Reihenfolge des Öffnens bei, kann

sie damit der schwarzen Seite beispielsweise ein Bit in Form einer logischen Eins signalisieren. Kehrt die rote Seite die Reihenfolge um, kann die damit eine logische Null übertragen.

6.5 ACK-Timing

Das *ACK-Timing* nutzt den Freiheitsgrad der Reaktionszeit auf einer Verbindung aus. Die rote Seite kann zu einem beliebigen Zeitpunkt vom Empfang eines *EOF*-Pakets bis zum Timeout der entsprechenden Verbindung, die Empfangsbestätigung verschicken. Mit den Zeitintervallen, die zwischen dem Empfang des *EOF*-Pakets und dem Versenden einer Empfangsbestätigung liegen, kann die rote Seite Informationen kodieren, welche von der schwarzen Seite interpretiert werden.

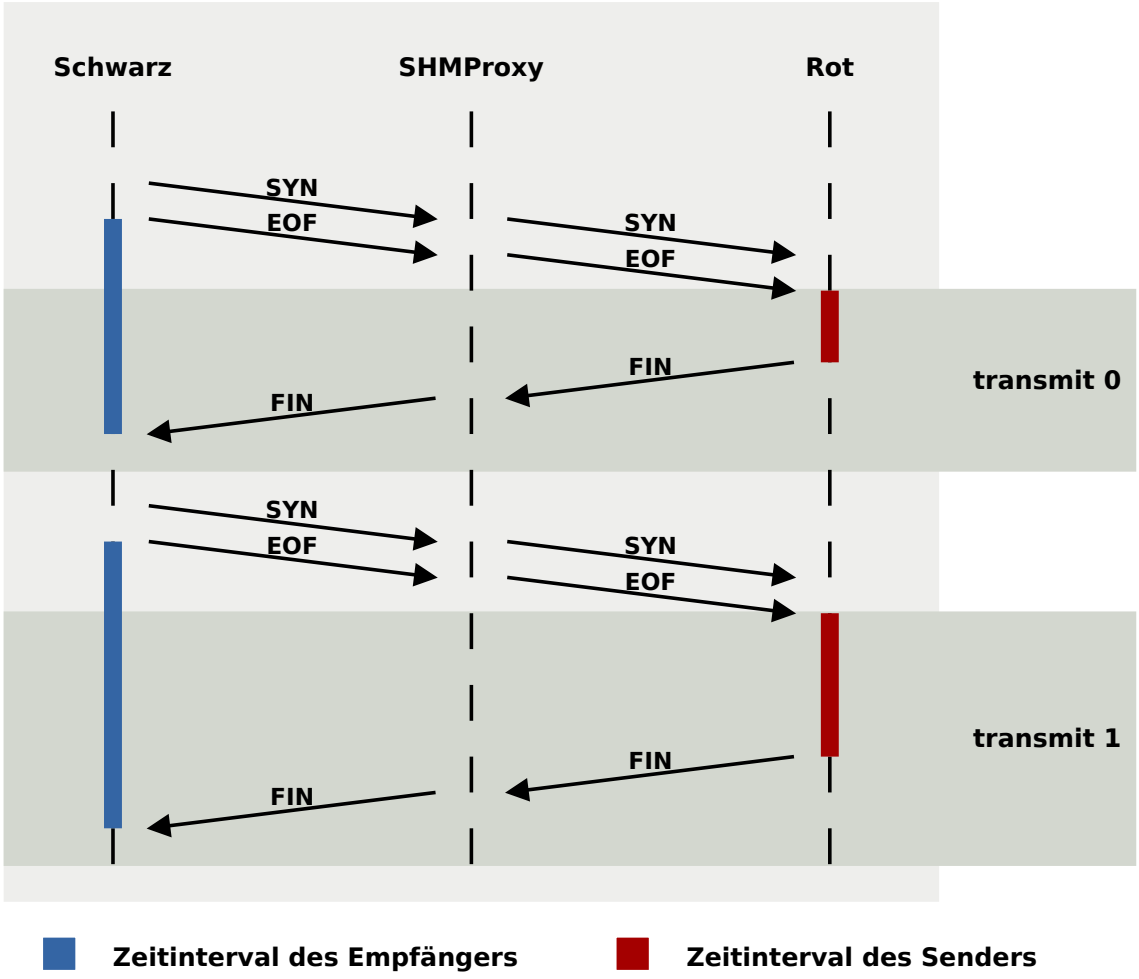

Abbildung 29: Sequenzdiagramm des Acknowledgement-Timing

Die Größe der Zeitintervalle, welche von der schwarzen Seite sicher unterschieden werden können, hängt stark von der Leistungsfähigkeit der eingesetzten Hardware sowie von der Belastung durch die laufende Software ab. Ebenso spielt der Umstand eine große Rolle, ob der Angreifer den Kommunikationskanal der Diode exklusiv oder parallel zum regulärem Datenverkehr nutzt.

6.6 Full-Buffer

Der letzte Freiheitsgrad in der Kommunikation ist die Entscheidung der roten Seite, Daten aus der Diode zu lesen oder nicht zu lesen. Dieser Freiheitsgrad ermöglicht einen weiteren Timing-Channel zur Kommunikation von rot nach schwarz. Bei dieser Methode sendet die schwarze Seite in kurzer Zeit viele Daten. Die rote Seite liest diese Daten nur sehr langsam von der Diode, sodass der intere Puffer voll läuft. Die schwarze Seite kommt dann durch die Flusskontrolle in einen blockierenden Zustand, da sie keine weiteren Daten mehr übermitteln kann. Die rote Seite liest nun in kleine Mengen Daten und kodiert mit den unterschiedlichen Zeitintervallen die zwischen den Zyklen des Lesens liege die zu Übermittelnden Informationen. Die schwarze Seite kann aufgrund der Dauer der blockierenden Zustände diese Zyklen wiedererkennen und damit auch die darin kodierten Informationen.

7 Design eines unidirektionalen Kommunikationsmechanismus

In diesem Kapitel wird das Protokoll beschrieben, mit dem die schwarze mit der roten Seite kommuniziert. Dabei werden zunächst einige Problemstellungen erläutert und dessen Lösungsvarianten diskutiert. Danach werden die in dieser Arbeit entwickelten Mechanismen des Dioden-Prozesses erklärt, welche die Angriffe auf den Rückkanal aus Abschnitt 6 verhindern oder abschwächen.

7.1 Behandlung von Fehlern während der Übertragung

Das fehlerfreie Übertragen großer Datenmengen ist ein generelles Problem in jeder Form eines Datentransfers. Bei einer komplexen Übertragungsstrecke wie etwa einer Datendiode mit vielen beteiligen Komponenten, muss mit dem Auftreten eines Fehlers zu jeder Zeit gerechnet werden. Um mit diesem Problem umzugehen, gibt es mehre Strategien.

7.1.1 Direkte Rückmeldung

In regulären Netzwerkprotokollen wie TCP [RFC793] werden bei der Übertragung regelmäßig Zwischenbestätigungen in Form von sogenannten ACK-Paketen versandt. Diese signalisieren dem Sender, dass alle Daten, welche bisher versandt wurden, korrekt empfangen worden sind. Gehen Daten verloren oder bleibt eine Bestätigung mittels ACK-Paket aus, müssen die Daten vom Sender erneut übertragen werden. Sollte bei einem der beiden Kommunikationspartner ein größeres Problem auftreten, kann jeder die gesamte Kommunikation zu jeder Zeit beenden.

Bei einer Datendiode ist eine solche Fehlerbehandlung nicht denkbar, da diese für einen Angreifer als Rückkanal genutzt werden kann. Denkbar wäre hier ein Angriff ähnlich dem im Abschnitt 6.3 beschriebenen *Multi-Bit-Choosing*, bei dem die Fehlermeldungen gezielt auf den Kanälen verschickt werden, welche man auswählen möchte, um eine

bestimmte Information damit zu kodieren. Bei einem Diodenprotokoll welches auf einen geringen Rückkanal angewiesen ist, ist dieses Verhalten daher nicht direkt anwendbar.

7.1.2 Black-Hole-Strategie

Zum anderen gibt es ein, in diesem Dokument als *Black-Hole-Strategie* bezeichnetes, Verfahren, bei dem die Daten von der schwarzen Seite aus solange ohne jede Rückmeldung an die rote Seite verschickt werden, bis die Übertragung von der schwarzen Seite beendet wurde. Erst dann übermittelt die rote Seite ihre Antwort und signalisiert einen positiven oder negativen Empfang.

Dieses Verfahren hat den Vorteil eines minimalen Rückkanals. Bei größeren Übertragungen, wird es allerdings zum Problem. Wenn bei einer Übertragung von etwa einem GByte während der ersten Bits bereits ein Fehler auftritt, muss die rote Seite zunächst die restliche Übertragung abwarten, bis sie den Fehler signalisieren kann. Im Fehlerfall kostet die *Black-Hole-Strategie* also Zeit sowie Übertragungskapazität, welche bei einem frühzeitigem Abbruch gespart werden könnten.

7.1.3 Response-Polling

Für die Lösung dieses Konflikts bietet sich an dieser Stelle ein Kompromiss zwischen einer direkten Rückmeldung sowie dem Vermeiden zusätzlicher Freiheitsgrade in der Kommunikation an. Der Kompromiss sieht vor, dass die rote Seite den Kanal — zudem es eine Statusbericht signalisieren will — nicht frei wählen kann. Beim in dieser Arbeit als *Response-Polling* beschriebenen Verfahren erfragt die schwarze Seite in regelmäßigen Abständen den Status einer Verbindung mittels spezieller Request-Pakete. Die rote Seite darf auf diese Pakete ausschließlich mittel *Response*- oder *Reset*-Paketen antworten, um einen positiven oder negativen Status der Verbindung zu übermitteln. Die schwarze Seite muss nach Erhalt des *Reset*-Paketes die Übertragung mit einem Fehler abbrechen.

Damit dieses Verfahren nicht als weitere *Multi-Bit-Choosing-Methode* ausgenutzt werden kann, muss der kontrollierende Dioden-Task, dafür sorgen, dass die *Request*-Pakete der schwarzen Seite nur einzeln an

die rote Seite weitergeleitet werden. Somit müssen die *Request*-Pakete ebenso behandelt werden, wie *EOF*-Paket mit der im Abschnitt 7.3 beschrieben EOF-Sperre. Damit kann die rote Seite ausschließlich auf eine Verbindung zur Zeit reagieren und verfügt über keinerlei Wahlfreiheit in ihrer Kommunikation.

7.2 NAK-Delay

Für eine gesicherte Kommunikation ist das Übertragen von Empfangsbestätigungen unumgänglich. Das im Abschnitt 6.2 beschriebene *Answer-Bit*-Verfahren, ermöglicht es somit grundsätzlich immer Daten von rot nach schwarz zu übertragen. Eine Möglichkeit darauf zu reagieren ist es, die gesamte Kommunikation auf eine vorbestimmte Anzahl von Verbindungen pro Zeit einschränken. Damit würde man auch den Rückkanal auf eine bestimmte Größe einschränken. Leider führt diese Vorgehensweise auch dazu, dass die reguläre Kommunikation enorm eingeschränkt wird. Dabei bietet sich hier ein Alternative an.

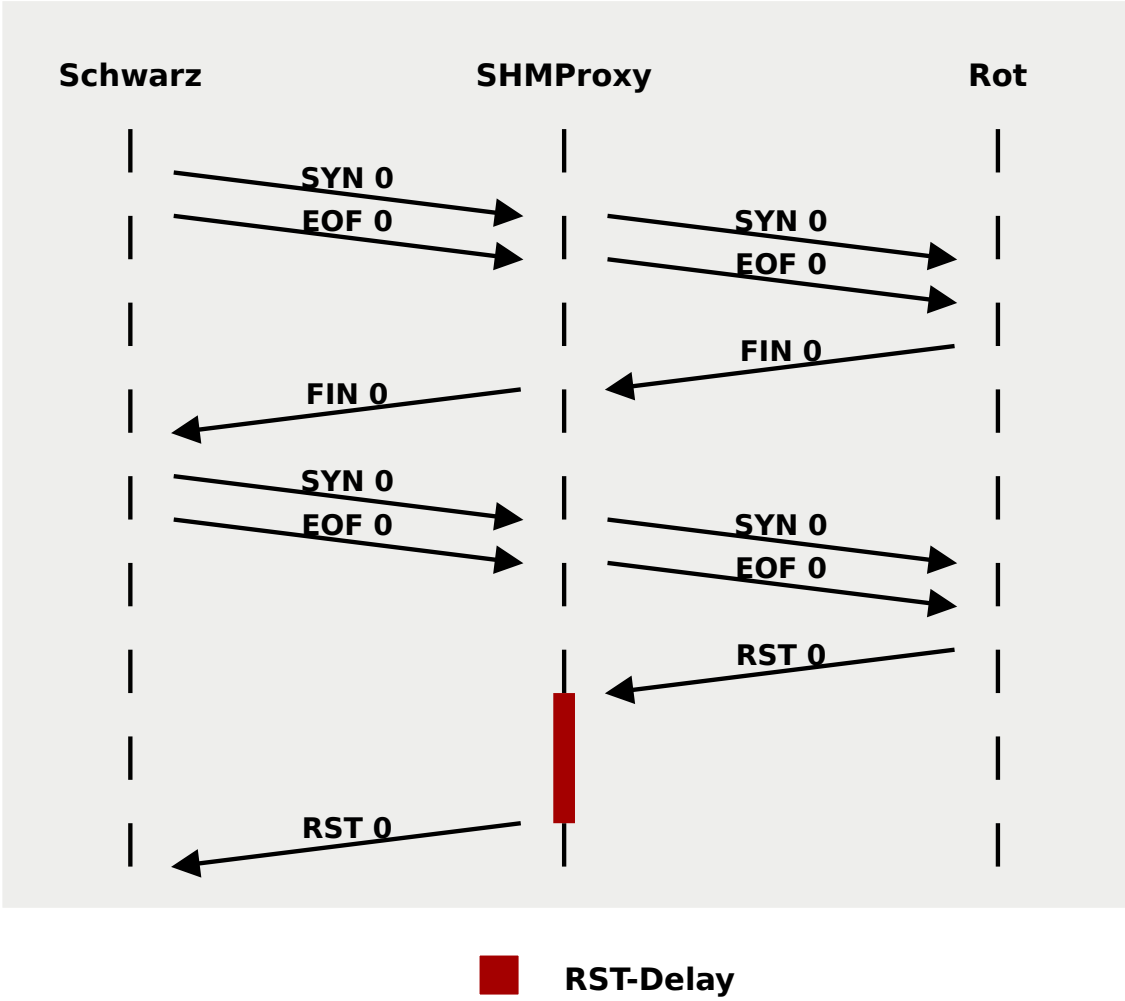

Abbildung 30: Sequenzdiagramm des NAK-Delay

Im regulären Fall einer Kommunikation von schwarz nach rot, werden überwiegend positive Empfangsbestätigungen (Acknowledgements) übertragen. Nur im Fehlerfall müsste die rote Seite eine negative Empfangsbestätigung (No Acknowledgement) übertragen. Für eine versteckte Kommunikation von rot nach schwarz würden aber gleichermaßen *ACK*- und *NAK*-Pakete verwendet. Anstatt nun die gesamte Kommunikation zu drosseln, reicht es aus ausschließlich die *NAK*-Pakte künstlich zu verlangsamen. Beim Empfang eines *NAK*-Paketes kann der Dioden-Prozess diese Pakete für eine vordefinierte Zeitspanne zwischenspeichern und danach weiterleiten. Wichtig ist dabei, dass sich die Delay-Zeiten von mehreren *NAK*-Paketen addieren und nicht parallel ablaufen. Da es sonst nicht zu einer Drosselung des Rückkanals kommt, sondern nur zu einer höheren Latenz.

7.3 EOF-Sperre

Die Parallelität von Verbindungen und der daraus resultierende Freiheitsgrad der roten Seite führt zu einem sehr großen Rückkanal. Eine Einschränkung auf eine rein serielle Kommunikation würde die Kommunikationsmöglichkeiten dabei aber enorm einschränken.

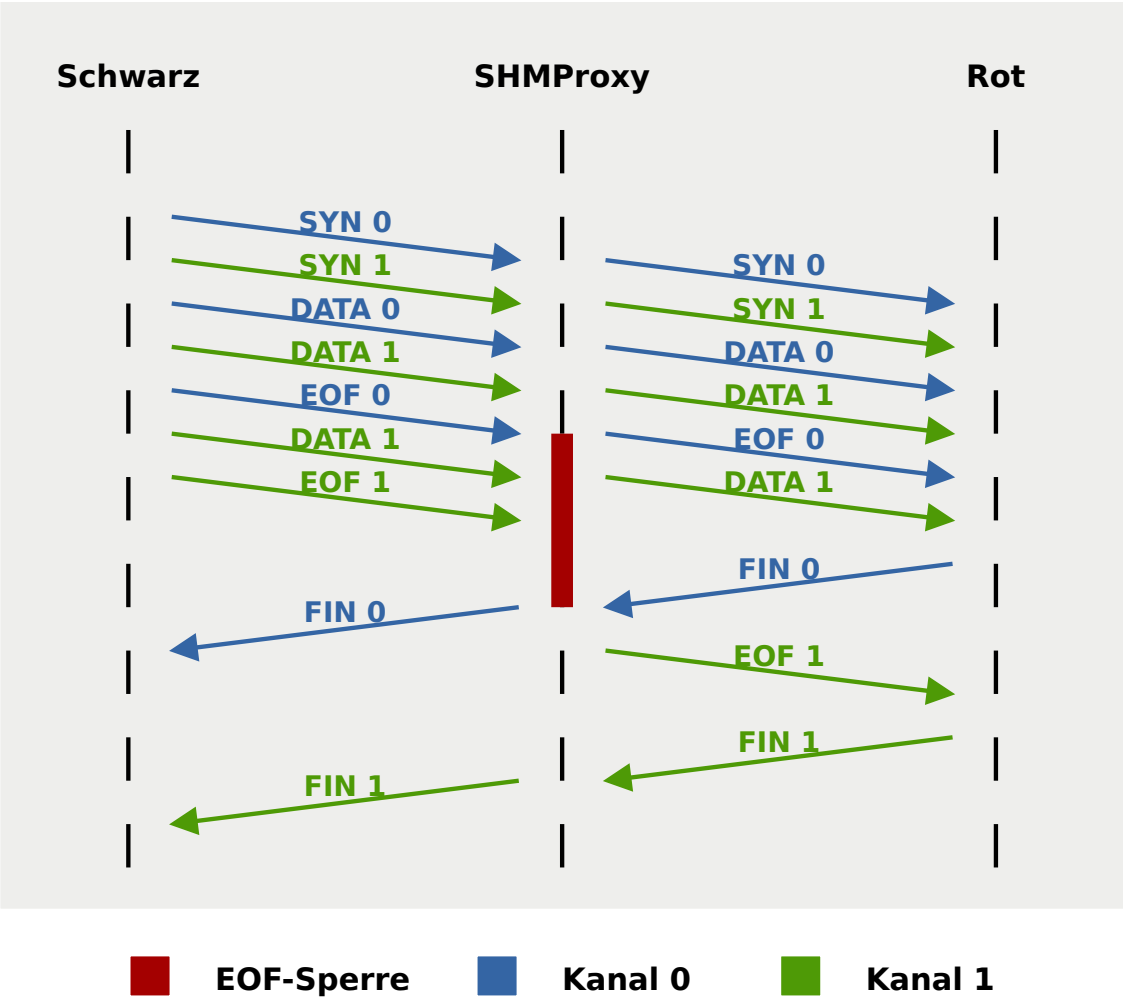

Abbildung 31: Sequenzdiagramm der EOF-Sperre

Ein möglicher Kompromiss ist, nur die Empfangsbestätigung der roten Seite zu serialisieren. Bei der *EOF-Sperre*, wird immer nur ein *EOF*-Paket an die rote Seite verschickt. Kommen weitere *EOF*-Pakete hinzu, während die rote Seite das erste noch bearbeitet, werden diese im *SHMProxy* zwischengespeichert. Erst wenn eine positive oder negative Bestätigung für das *EOF* empfangen wurde, überträgt der *SHMProxy* das nächste *EOF*-Paket zur roten Seite. *SYN*- sowie Datenpake-

te können hingegen die gesamte Zeit über ungehindert von schwarz nach rot übertragen werden. Diese Verfahren muss ebenfalls mit dem Abschnitt 7.1.3 beschrieben Response-Polling kombiniert werden. Das bedeutet, dass die Requests und Responses genauso behandelt werden müssen wie *EOF*-Pakete und Empfangsbestätigungen.

7.4 Paketprotokoll

Angelehnt an das zuvor beschriebene L4-Pipe-Paketprotokoll (siehe Abschnitt 5.2) wurde ein neues Paketprotokoll entworfen, welches den Anforderungen für den Einsatz in einer Datendiode gerecht wird.

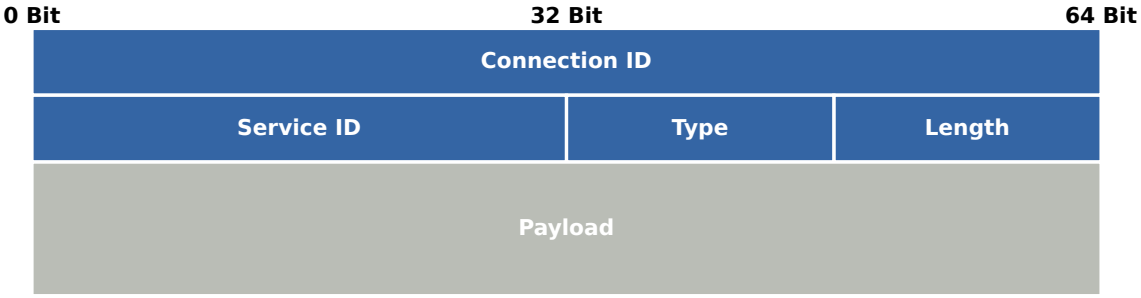

Abbildung 32: Paket-Header des L4-Pipe-Protokoll

Der Paket-Header wurde um das Feld *Service-ID* erweitert. Dieses Feld dient zur Adressierung von verschiedenen Diensten, welche auf der roten Seite erreicht werden sollen. Somit können nun mehre Dienste wie in Abbildung 33 über einen einzelnen Dioden-Prozess erreicht werden und die Erweiterung des Rückkanals aus Abschnitt 5.5 wird verhindert. Die Längen der Felder *Service-ID* sowie *Type* wurde so gewählt, dass der gesamte Header eine korrekte Speicherausrichtung [Rent05] erhält.

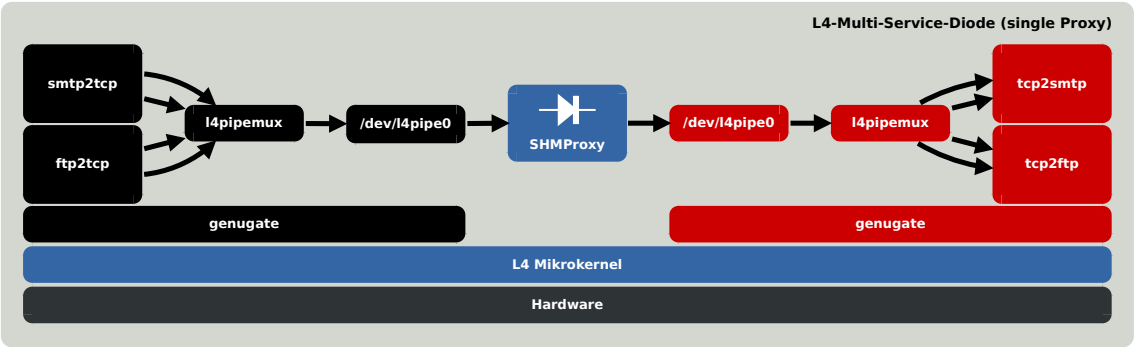

Abbildung 33: Multi-Service Kommunikationskanal mit einem *SHM-Proxy*

Die Pakettypen wurden um die Typen *REQ* und *RSP* erweitert. Dieser Erweiterung ermöglicht das im Abschnitt 7.1.3 beschriebene Response-Polling-Verfahren und erlaubt es bei großen Datenübertragungen den Status einer Verbindung zu erfragen.

Tabelle 2: Beschreibung der Pakettypen

Type	**Beschreibung**
SYN	Öffnet eine Verbindung
DAT	(data) Verschickt Daten über eine offene Verbindung
REQ	(request) Erfragt Statusbericht über Verbindung
EOF	(end of file) Beendet eine Verbindung
RST	(reset) Terminiert eine Verbindung im Fehlerfall
FIN	(finish) Positive Empfangsbestätigung
RSP	(response) Positive Reaktion auf einen Request von schwarz

Ein kompletter Protokollablauf sieht wie folgt aus.

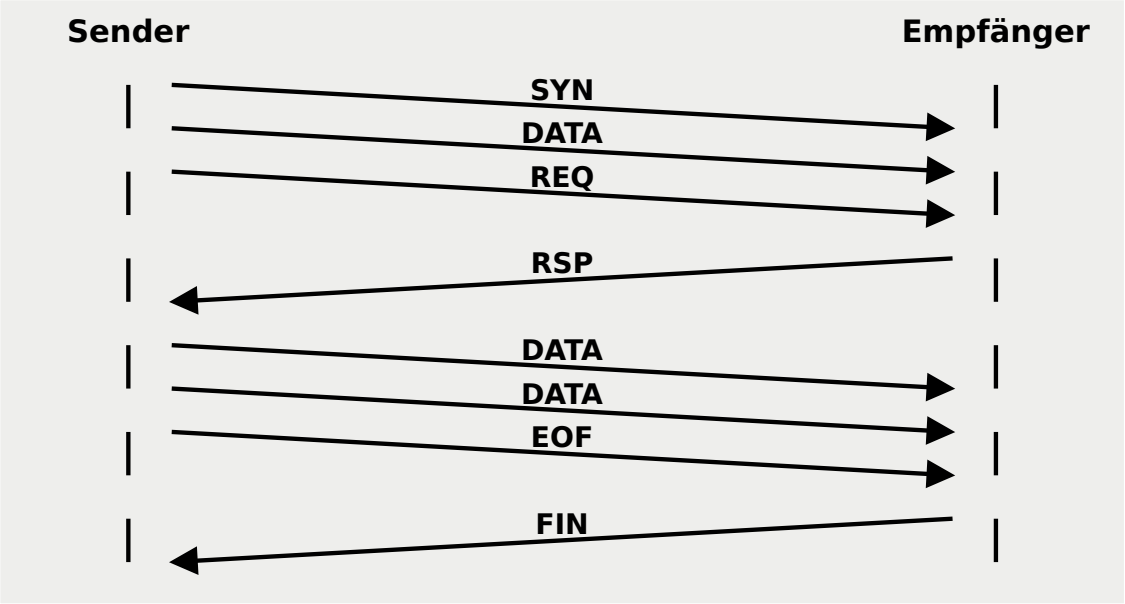

Abbildung 34: Sequenzdiagramm zum Protokoll

Der Sender öffnet einen Kanal mit einem *SYN*-Paket. Dazu wählt er eine eindeutige *Connection-ID* sowie die *Service-ID* des Dienstes, welchen er beim Empfänger erreichen möchte. Darauf folgen Datenpakete mit den zu übertragenen Informationen. Bei großen Übertragungen kann der Sender optional *REQ*-Pakete verschicken, um den Status der Verbindung zu erfragen. Sollte in der Zwischenzeit bei der Übertragung oder auf der Empfängerseite ein Fehler aufgetreten sein, kann diese die Verbindung mit einem *RST*-Paket abbrechen. Anderenfalls wird die Verbindung mit einem *FIN*-Paket weiter fortgesetzt. Das Ende der Übertragung signalisiert der Sender mit einem *EOF*-Paket. Erst wenn dieses durch den Empfänger mit einem *FIN*-Paket quittiert wurde, hat der Sender die Bestätigung, dass die Übertragung beim Empfänger komplett angekommen ist und positiv interpretiert werden konnte. Im Falle eines *RST*-Paket ist ein Fehler aufgetreten und die Übertragung damit fehlgeschlagen.

7.5 Implementierung

In diesem Abschnitt wird die Implementierung des Dioden-Prozesses erläutert. Es wird zunächst der grundlegende Mechanismus der unidirektionalen paketbasierten Kommunikation erläutert, um dann später

auf die Implementierung des EOF-Blockings sowie des NAK-Delays einzugehen.

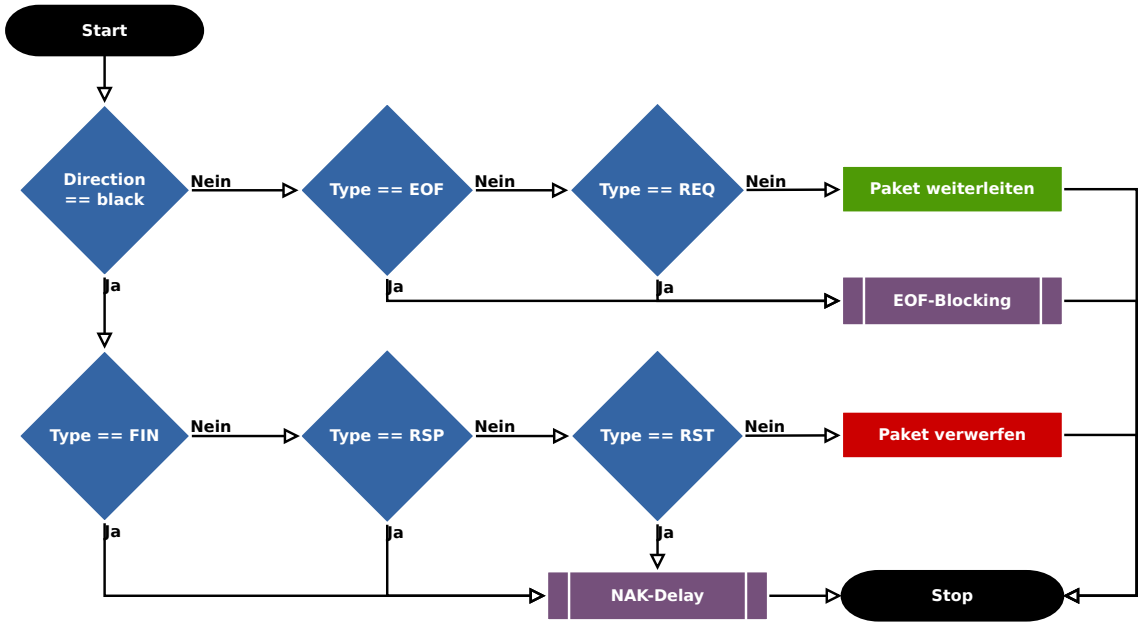

Abbildung 35: Programmablaufplan(PAP) des Dioden-Prozess

Abbildung 35 zeigt den grundlegenden Programmablaufplan des Dioden-Prozesses. Dabei werden zunächst die Richtung und der Typ der eingehenden Pakete geprüft. Von schwarz nach rot sind grundsätzlich alle Pakettypen erlaubt. Die Typen *EOF* und *RSP* werden allerdings speziell in der Routine *EOF-Blocking* behandelt.

Bei der Übertragungsrichtung von rot nach schwarz werden alle Pakete verworfen, welche nicht vom Typ *FIN*, *RSP* oder *RST* sind. Die erlaubten Pakete werden aber nicht direkt weitergeleitet sondern durch laufen zunächst noch das Unterprogramm *NAK-Delay*.

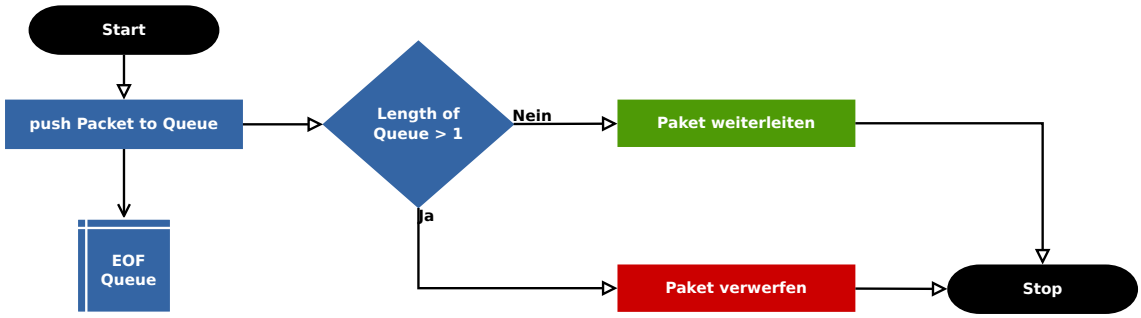

Abbildung 36: PAP des EOF-Blocking-Unterprogramms

In der Unterprogramm *EOF-Blocking* (siehe Abbildung 36) wird ein Teil des EOF-Blocking-Verfahrens implementiert. Vorausgesetzt ist hier der Umstand, dass das aktuelle Paket von schwarz nach rot übertragen wird und dem Typ *EOF* oder *REQ* entspricht. Der zentrale Bestandteil dieses Verfahrens ist die *EOF-Queue*. Dieses ist eine FIFO-Liste, welche alle unbeantworteten EOF-Pakete zwischenspeichert. Ist ein eintreffendes *EOF*- oder *REQ*-Paket das erste in der Liste, wird es direkt weitergeleitet. Sind vor diesem allerdings noch anderen Pakete in der *EOF-Queue* wird es in der Queue gespeichert und verworfen. Im Unterprogramm *NAK-Delay* werden die Antwortpakete der roten Seite verarbeitet. Dort befindet sich auch der zweite Teil des *EOF-Blocking*-Verfahrens.

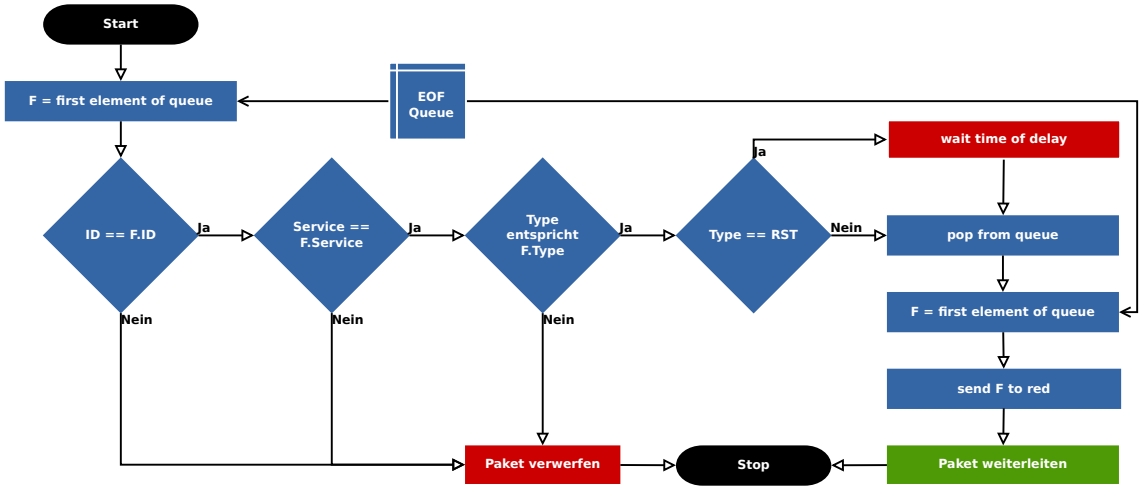

Abbildung 37: PAP des NAK-Delay-Unterprogramms

Das Unterprogramm *NAK-Delay* (siehe Abbildung 37) implementiert das *NAK-Delay*-Verfahren und den zweiten Teil des *EOF-Blocking*. Zunächst wird hier die *Connection-ID* und die *Service-ID* sowie das *Type*-Feld des aktuellen Pakets mit dem ersten Element in der *EOF-Queue* verglichen. Damit soll sichergestellt werden, dass die rote Seite auch nur auf das *EOF*- oder *REQ*-Paket antwortet, welches es zuvor bekommen hat. Handelt es sich um ein *RST*-Paket, also eine negative Empfangsbestätigung, wird nur eine vordefinierte Zeit gewartet, bevor mit der restlichen Verarbeitung fortgefahren wird. Dieses ist das eigentlich *NAK-Delay*, wie es bereits im Abschnitt 7.2 beschrieben wurde. Im Falle einer positiven Empfangsbestätigung mittels eines *FIN*- oder *RSP*-Pakets, wird unmittelbar mit Weiterverarbeitung fortgefahren.

Der nachfolgende Teils ist der Rest des *EOF-Blocking*. Nachdem von der roten Seite eine Antwort auf ein zuvor versendetes *EOF*-Paket empfangen wurde, muss das aktuelle, und nun bestätigte Paket aus der *EOF-Queue* entnommen und verworfen werden. Das darauffolgende Paket in der *EOF-Queue* muss dann an die rote Seite versandt werden. Somit wird sichergestellt, dass die rote Seite ausschließlich auf ein *EOF*-Paket zur Zeit reagieren kann. Das vorhalten der anderen *EOF*-Pakete in der *EOF-Queue* nimmt der roten Seite den Freiheitsgrad der Wahlmöglichkeit auf einen beliebige Verbindung zu reagieren und schränkt somit ebenfalls den potentiellen Rückkanal ein.

8 Analyse des verbleibenden Rückkanals

In den vorherigen Kapiteln wurde Mechanismen und Verfahren erläutert und entwickelt, welche zu einer sicherere logische Datendiode führen. Um diese Sicherheit auch zu prüfen, wird in diesem Kapitel der verbleibende Rückkanal untersucht und mit vorhergehenden Lösungen verglichen.

8.1 Impliziter Rückkanal

Dieser Abschnitt leitet den Rückkanal von den verbleibenden Freiheitsgraden her. Im Abschnitt 6.1 wurden vier Freiheitsgrade ermittelt, welcher der roten Seite in ihrer Kommunikation mit der schwarzen Seite zur Verfügung stehen.

Die Art der Empfangsbestätigung, die Reaktionszeit auf eine Verbindung, die Reihenfolge der Bestätigung von parallelen Verbindungen sowie Entscheidung Daten zu lesen oder nicht zu lesen.

Durch die in dieser Arbeit entwickelten Verfahren des EOF-Blocking sowie das NAK-Delay konnten die Freiheitsgrade der Art der Empfangsbestätigung sowie der Reihenfolge der Reaktion enorm eingeschränkt werden. Die die Entscheidung in welcher Reihenfolge die rote Seite auf parallele Verbindungen reagiert, konnte durch das EOF-Blocking auf die schwarze Seite verlagert werden. Mit diesem Verfahren stellt der in dieser Arbeit entwickelte Dioden-Prozess sicher, dass die Empfangsbestätigungen in der selber Reihenfolge an die schwarze Seite versendet werden, wie diese zuvor die Verbindungen geschlossen hat. Somit kann der Freiheitsgrad der Reihenfolge von der rote Seite nicht mehr genutzt werden.

Die Entwicklung des NAK-Delay, konnte den Freiheitsgrad der Empfangsbestätigung zwar nicht verhindern, aber deren Ausnutzung enorm erschweren. Die im Abschnitt 6.1 ermittelten Reaktionsmöglichkeiten einer positiven, negativen oder keiner Empfangsbestätigung auf eine Verbindung sind geblieben. Der Durchsatz, der durch die Nutzung der Empfangsbestätigungen zur Kommunikation einem potentiellen Angreifer zur Verfügung steht, konnte jedoch stark reduziert werden. Das Nicht-Reagieren auf eine Verbindung, also ein Timeout, ist für den

Durchsatz belastend, da die schwarze Seite zum Erkennen eines Timeouts zumindest eine überdurchschnittlich lange Zeit abwarten muss. Da mit dem in dieser Arbeit entwickelten NAK-Delay auch alle negativen Empfangsbestätigungen vom Dioden-Prozess mit einer Zwangswartezeit versehen werden, kann die schwarze Seite noch schlechter zwischen negativen oder ausbleibenden Empfangsbestätigungen unterscheiden. Die schwarze Seite muss zumindest eine überdurchschnittliche Zeit des NAK-Delay abwarten um zwischen Timeout und negativer Empfangsbestätigung zu unterscheiden. In Verbindung mit der EOF-Sperre und der damit einhergehenden Serialisierung der Empfangsbestätigungen, muss die schwarze Seite zum Empfangen der negativen und der darauffolgenden eventuell positiven Empfangsbestätigungen die gesamten vom Dioden-Prozess erzwungenen Wartezeiten verstreichen lassen. Bei zu vielen negativen Empfangsbestätigungen, die hintereinander in zu kurzer Zeit kommen, können diese Zeiten dann auch einen Timeout erreichen und die zuvor von der roten Seite kodierten positiven und negativen Bestätigungen als Timeout verschwinden. Durch die Wahl der Wartezeiten für negative Empfangsbestätigungen sowie der Wahl des Timeout, kann der Durchsatz dieses Kommunikationsverfahrens beliebig gedrosselt werden, ohne dabei die reguläre und fehlerfreie Kommunikation beeinträchtigt wird.

Die restlichen Freiheitsgrade konnten nicht weiter eingeschränkt werden. Die Reaktionszeit auf Verbindungen der schwarzen Seite lässt sich weiterhin von einem potentiellen Angreifer ausnutzen. Ebenso bleibt die Möglichkeit bestehen, über die Entscheidung der roten Seite Daten zu lesen oder nichts zu lesen mit der schwarzen Seite zu kommunizieren. Da diese beiden Verfahrensweisen nicht miteinander kombinierbar sind, können sie jeweils nur einzeln betrachtet werden.

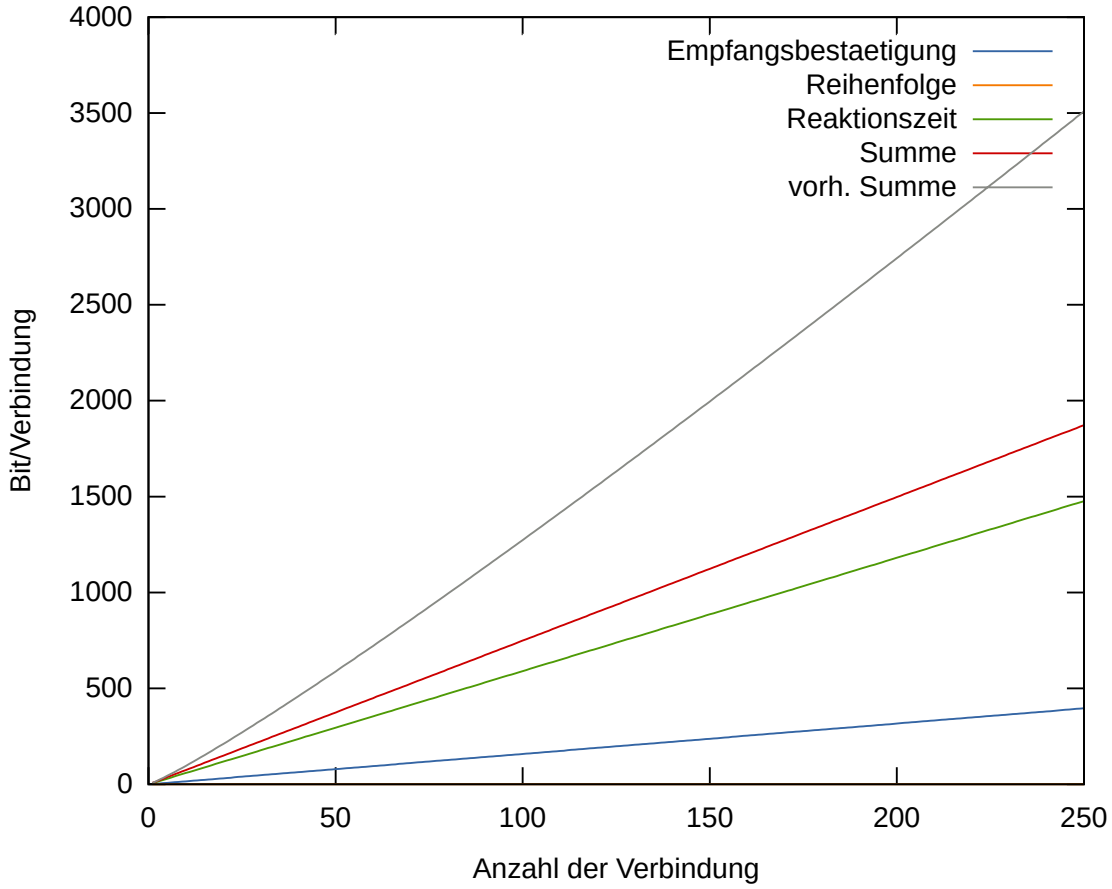

Abbildung 38: Verbleibender theoretischer Rückkanal im Verhältnis zu Parallelen Verbindungen

8.2 Vergleich zur genugate-Datendiode

Da die in dieser Arbeit entwickelte logische Datendiode mit Teilen der genugate-Datendiode kombiniert werden kann, werden die Rückkanäle beider Dioden in diesem Kapitel miteinander vergleichen. Der Vergleich bezieht sich auf den Paketfilter der genugate-Datendiode sowie den in dieser Arbeit entwickelten Dioden-Prozess, ohne die unidirektionalen Relays.

8.2.1 Paketebene

Im Dokument [Meis07] werden die Rückkanalmöglichkeiten der genugate-Datendiode analysiert. In der Analyse wird die genugate-Datendiode als ganzes mit den Relays betrachtet sowie auch nur der Paket-

filter. Dabei wurde eine Bitrate von 14 Bit pro Paket für den Rückkanal ermittelt. Diese Rate bezieht sich auf Pakete welche von rot nach schwarz als Antwort übertragen werden.

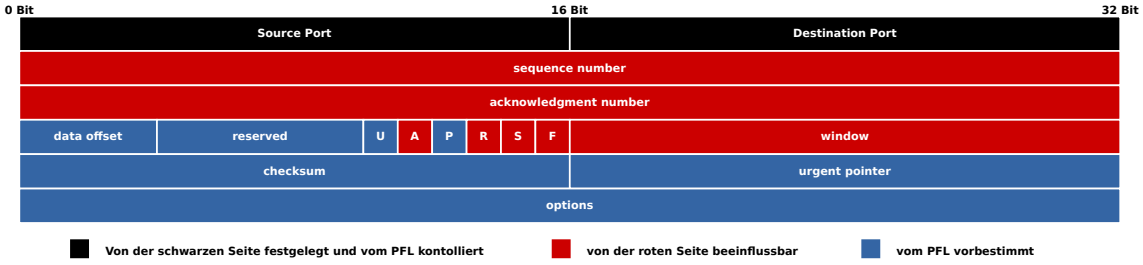

Abbildung 39: Einfluss auf den TCP-Header

Die in [Meis07] ermittelte Bitrate der Antwortpakete ist in den Freiheitsgraden des TCP-Protokolls begründet. Der roten Seite stehen einige Felder im TCP-Header zur Verfügung in denen sie Informationen kodieren kann. Diese Felder sind in Abbildung 39 rot gekennzeichnet. Die schwarzen Felder wurden initial beim Verbindungsaufbau von der schwarzen Seite festgelegt, werden durch den Paketfilter überprüft und sind somit unveränderlich. Die blauen Felder werden vom Paketfilter durch die Erweiterung `pf_oneway(9)` [Bluhm07] auf vordefinierte Werte festgelegt oder normalisiert. Einige der roten Felder werden ebenfalls vom Paketfilter soweit wie möglich normalisiert. Sie bieten aber weiterhin die Freiheitsgrade zur beliebigen Kodierung von Informationen die zur Summe von 14 Bit pro Paket führen.

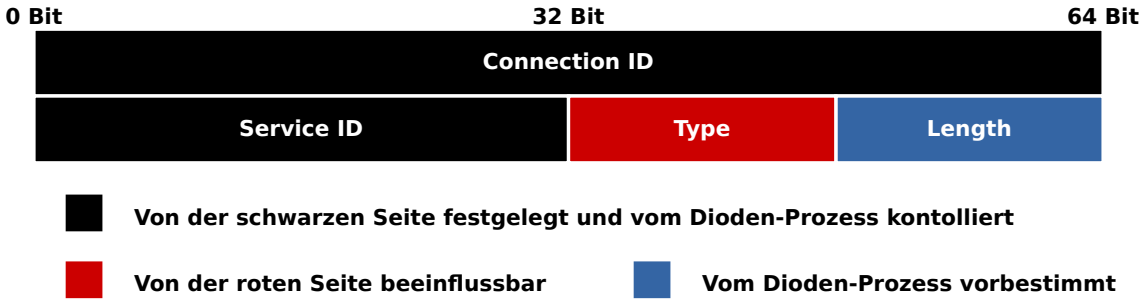

Abbildung 40: Einfluss auf den L4-Pipe-Header

Im direkten Vergleich dazu zeigt Abbildung 40 die gleichen Einflussmöglichkeit auf den L4-Pipe-Header. Durch die Reduktion der Komplexität des Protokolls im Vergleich zu TCP, ergeben sich hier wesentlich weniger Angriffsmöglichkeiten auf den Rückkanal. Der roten Seite

bleibt auf Paketebene nur das *Type*-Feld, welches sie auf die Werte *FIN* oder *RST* beziehungsweise *RSP* oder *RST* setzen kann. Je nach Kontext der Verbindung bleiben also nur zwei Antwortmöglichkeiten zwischen denen die rote Seite wählen kann. Somit ergibt sich auf Paketebene ein Rückkanal von einem Bit pro Paket.

8.2.2 Zeitebene

Im Dokument [Meis07] wird ebenfalls eine Abschätzung für einen Timing-Channel der genugate-Datendiode gegeben. Bei einem Timing-Channel werden Informationen wie im Abschnitt 6.5 erläutert mittelt Zeitverzögerungen kodiert. Für die genugate-Datendiode wurde ein Wert von 219,72 KiB pro Sekunde ermittelt.

Die Berechnung des Timing-Channels aus [Meis07] geht aus folgender Formel hervor.

$$K = \frac{1Bit}{Paket} \cdot \frac{Paket}{69Byte} \cdot \frac{Byte}{8Bit} \cdot \frac{10^9 Bit}{s} \approx \frac{1.8 \cdot 10^6 Bit}{s}$$

Diese Formel geht von zwei Symbolen auf dem Timing-Channel aus. Dieses bedeutet ein Symbol in Form eines schnellen Paketes sowie ein Symbol in Form eines Paketes welches mit einer zeitlichen Verzögerung verschickt wurde. Somit ist der Empfänger in der Lage die beiden Symbole von einer zu unterscheiden. Für die beiden Symbole die ein Paket repräsentieren kann, ergibt sich der Faktor $\frac{1Bit}{Paket}$.

Da für diese Abschätzung das kleinste Zeitintervall, welches der Empfänger noch sicher unterscheiden kann, nicht sicher ermittelbar ist, wird dieses vernachlässigt. In der Formel aus [Meis07] wird zur Berechnung des Timing-Channels die durchschnittlichen Paketrate daher auf den Wert der maximal möglichen Paketrate gesetzt. Die maximale Paketrate berechnet sich aus der Größe des TCP-Pakets (TCP-Header + IP-Header + Ethernet-Frame) und der größtmöglichen Geschwindigkeit des Übertragungskanals. Somit ergibt sich für ein TCP-Paket von mindestens 69 Byte und einem Netzwerkverbindung von einem GBit pro Sekunde der Faktor $\frac{Paket}{69Byte} \cdot \frac{Byte}{8Bit} \cdot \frac{10^9 Bit}{s}$. In einem realen Szenario wird dieser Wert nie erreicht, da ein Teil der Pakete mit einer künstlichen Verzögerung verschickt wird und damit die durchschnittliche Paketrate immer unter der Maximalrate liegt.

Für eine vergleichende Abschätzung des Timing-Channels für das in dieser Arbeit entwickelte Datendioden-Protokoll kann diese Formel eben-

falls verwendet werden. Dafür müssen die Werte der Paketgröße sowie der Geschwindigkeit des Übertragungskanals angepasst werden.

Der Paket-Header des Dioden-Protokolls (siehe Abschnitt 7.4) hat eine Größe von 16 Bytes.

Die Benennung einer maximalen Übertragungskapazität gestaltet sich an dieser Stelle etwas schwieriger. Im Gegensatz zu eine standardisierten GBit-Ethernet-Verbindung kann man für Shared-Memory keinen definierten Übertragungsraten angeben. Diese sind immer abhängig von der verwendeten Hardware sowie Software. Eine veränderte neuere Hardware mit schnelleren Bus-Takten oder ein verbesserter Scheduling-Algorithmus der Betriebssystems kann die Geschwindigkeit einer Shared-Memory-Kommunikation beeinflussen. Daher wird an dieser Stelle mit einem Messwert der L4-Pipe weiter gerechnet.

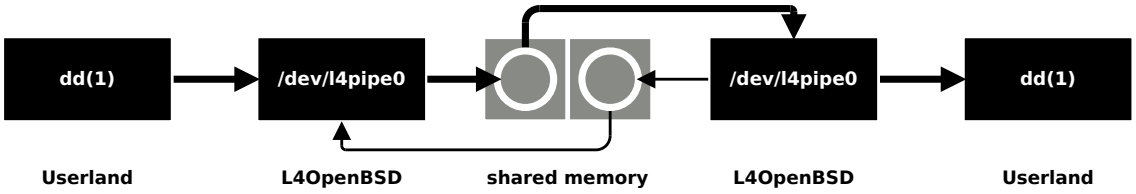

Abbildung 41: Messstrecke zur L4-Pipe

Ermittelt wurde dieser Messwert mir einer einfachen L4-Pipe-Verbindung zwischen zwei L4OpenBSD-Instanzen (siehe Abbildung 41). Das UNIX-Programm `dd(1)` wird dazu benutzt die Daten vom virtuellen Gerät `/dev/zero` über die L4-Gerätedatei `/dev/l4pipe0` in die L4-Pipe zu kopieren. Auf der anderen Seite werden die empfangenden Daten ebenfalls mit dem Programm `dd(1)` aus der L4-Pipe herauskopiert und über das virtuelle Device `/dev/null` verworfen. Nach dem manuellen Abbruch des sendenden `dd(1)`-Programms, zeigt dieses eine durchschnittliche Übertragungsrate von 83.294.347 Bytes pro Sekunde an. Durchgeführt wurde diese Messung auf einer genugate-800 Standard-Hardware[8].

Setzt man die Paketgröße sowie die gemessenen Übertragungsgeschwindigkeit in die Formel ein, ergbit sich für die in dieser Arbeit entwickelte

[8]Die Hardware mit der Bezeichnung genugate-800 Rev. 6 ist die leistungsfähigste Hardware mit der Produkte von genua ausgeliefert werden. Diese Hardware-Klasse entspricht folgender Leistungsdaten: 2x Intel Xeon X5672, 4 Cores, 3.2GHz per core, 6*4GB DDR3-RAM

Datendioden ein Timing-Channel von ca. 622 KiB pro Sekunde.

$K = \frac{1Bit}{Paket} \cdot \frac{Paket}{16Byte} \cdot \frac{Byte}{8Bit} \cdot \frac{6.6 \cdot 10^8 Bit}{s} \approx \frac{5.1 \cdot 10^6 Bit}{s}$

Durch den kleineren Header ist der Wert an dieser Stelle um den Faktor 2,8 größer. Dieser Wert lässt sich durch eine künstliche Reduktion der Übertragungsgeschwindigkeit ebenfalls auf einen beliebigen Wert reduzieren. Der Anwender der Datendiode muss hier ein für ihn akzeptables Verhältnis zwischen Durchsatz und Rückkanal finden.

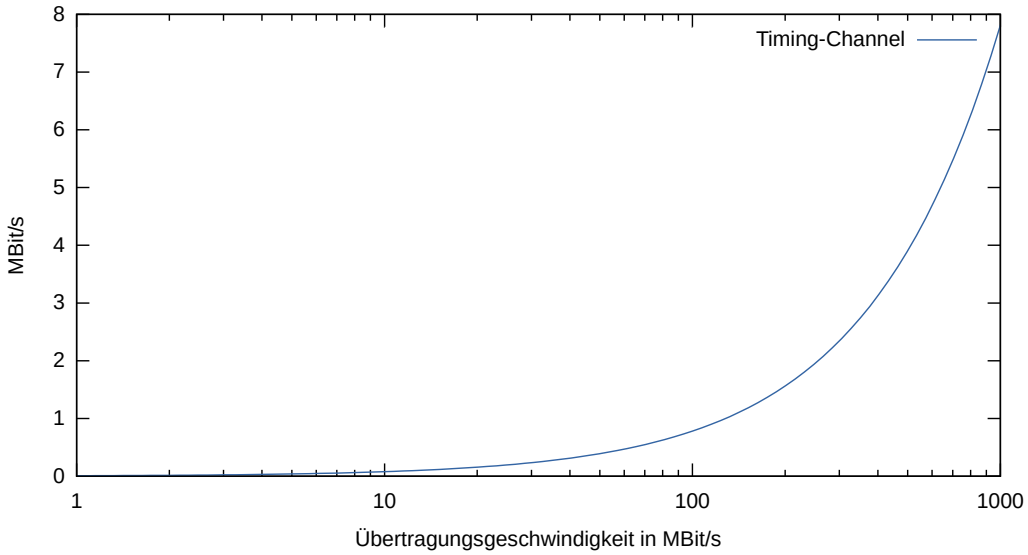

Abbildung 42: Verhältnis von Übertragungskanal und Timing-Channel

9 Fazit und Ausblick

Dieses Kapitel fasst die Ergebnisse dieser Arbeit noch einmal zusammen und gibt einen Ausblick auf noch offene Fragen, welche in zukünftigen Arbeiten geklärt werden sollten.

9.1 Ergebnisse

Durch die Entwicklung der Mechanismen EOF-Blocking sowie NAK-Delay wurde ein Dioden-Protokoll geschaffen, welches diesen Bereich der Rückkanäle für einen Angreifer unbenutzbar macht. Diese Verfahren erhöhen die Sicherheit des Datendioden-Protokolls ohne dabei direkt auf den Datendurchsatz des Übertragungskanals negative einzuwirken. Mit dem entwickelten Response-Polling-Verfahren konnte eine schnelle Rückmeldung von Übertragungsproblemen an die schwarze Seite erreicht werden, ohne dadurch einen großeren Rückkanal für den Angreifer zu schaffen. Zudem hat die Erweiterung des L4-Pipe-Protokolls um das *Service-ID*-Feld dazu geführt, dass mehre Dienste parallel über das Dioden-Protokoll angesprochen werden können. Dieses verhinderte eine unnötige Vergrößerung des Rückkanals, welcher durch den Einsatz von mehren parallelen Datendioden entstanden wäre.

Das in dieser Arbeit entwickelte Datendioden-Protokoll führt zu einer logischen Datendiode, welche die genugate-Datendioden hervorragend ergänzt. Das im Umfeld dieser entstehende Produkt, die sogenannte L4-Diode, erlaubt eine Kombination mit der genugate-Datendiode, indem es den Paketfilter zwischen den unidirektionalen Relays ersetzt. Dadurch werden die TCP-Verbindungen zwischen den Relays an der L4-Diode terminiert und der in [Meis07] ermittelte Rückkanal auf den verbleibenden Rückkanal des in dieser Arbeit entwickelten Protokolls reduziert (siehe Abschnitt 8).

9.2 Ausblick

Auch wenn diese Arbeit Antworten auf einige Rückkanalprobleme finden konnte, bleiben andere Problemstellungen weiter unbeantwortet. Zum einen konnte die Analyse des Timing-Channel aus Abschnitt 8.2.2

zeigen, dass das hier noch ein großes Potential für einen Covert-Channel verbleibt. Momentan kann man darauf nur mit einer Drosslung der gesamten Übertragungsgeschwindigkeit reagieren. Zukünftige Arbeiten in diesem Feld könnten diese Art des Rückkanals genauer betrachten um eventuell Alternativen zur Drosselung der gesamt Geschwindigkeit zu finden. So wäre es denkbar vor dem Versenden von Empfangsbestätigungen eine zusätzliche und zufällige Wartezeit zerstreichen zu lassen um den Timing-Channel zu stören. Diese würde zu einer Vergrößerung der kleinsten sicher zu erkennenden Zeitintervalle führen und damit zu einer Einschränkung des Rückkanal. Alternativ würden feste Zeitintervalle, zu denen Empfangsbestätigungen versendt werden, dazu führen, dass eine künstliche Untergrenze für die kleinsten sicher zu unterscheidenen Zeitintervalle festlegt wird. Diese und andere Massnahmen müssten zukünftigen Arbeiten weiter untersucht werden.

Ein anderer Bereich, welcher in dieser Arbeit unbetrachtet blieb, sind sogenannte Seitenkanäle. Da sich die beiden voneinander separierten L4OpenBSD-Instanzen auf einer physischen Hardware befinden muss untersucht werden, in wieweit dieses ohne direkte Interaktion via IPCs kommunizieren könnten.

10 Thesen

- Die Reduktion der Freiheitsgrade in der Kommunikation der roten Seite einer Datendiode, ist eine wirksame Vorgehensweise, um den Rückkanal für Angreifer zu reduzieren.
- Ohne einen Rückkanal kann es keine zuverlässige Kommunikation geben.
- Sobald die rote Seite keinen Einfluss auf die Empfangsbestätigungen der schwarzen Seite hat, kann einen zuverlässige Kommunikation nicht gewährleistet werden.
- Forward-Error-Correction bietet keine ausreichende Zuverlässigkeit in einer unidirektionalen Kommunikation.

11 Literatur

11.1 Literatur

[1] [Bluhm07] Alexander Bluhm, „One Way TCP State in pf“, Interne Entwicklungsdokumentation, genua mbh, 24. August 2007

[2] [BSIZ13] Zertifizierung des Bundesamtes für Sicherheit in der Informationstechnik, https://www.bsi.bund.de/DE/-Themen/ZertifizierungundAnerkennung/-zertifizierungundanerkennung_node.html, Abgerufen: 25. Juli 2013

[3] [CaSi09] Ludovic Piètre-Cambacédès und Pascal Sitbon, „An Analysis of Two New Directions in Control System Perimeter Security“, Electricité de France, Telecom ParisTech – LTCI CNRS France, 10. Juni 2009

[4] [Davi05] Noopur Davis, „Secure Software Development Life Cycle Processes“, Carnegie Mellon University, 5. Juli 2006, https://buildsecurityin.us-cert.gov/articles/-knowledge/sdlc-process/secure-software-development-life-cycle-processes, Abgerufen: 31. Juli 2013

[5] [FEC] Wikipedia, „Vorwärtsfehlerkorrektur“, 3. April 2013 20:08 Uhr, http://de.wikipedia.org/wiki/-Vorwärtsfehlerkorrektur, Abgerufen: 22. August 2013

[6] [Fox07] Fox-IT, „Fort Fox Data Diode: A Preferred Solution For High-Security Real-time Electronic Data Transfer Between Networks“, 25. Juni 2007

[7] [Hope10] James Hope, Jeffrey Menoher, Dennis P. Mirante und Ronald Mraz, „one-way data transfer system with built-in data verification mechanism“, US-Patent-Nr.: 7675867 B1, 9. März 2010

[8] [Jone06] D.W. Jones, „RS-232 - Data Diode Tutorial and Reference Manual“ - Draft, Univ. of Iowa, 28. Ju-

li 2006 http://www.cs.uiowa.edu/~jones/voting/diode/-RS232tech.pdf, Abgerufen: 09. August 2013

[9] [Kang05] Myong H. Kang, Ira S. Moskowitz und Stanley Chincheck, „The Pump: A Decade of Covert Fun“, 21st Annual Computer Security Applications Conference, Dezember 2005

[10] [KM95] M.H. Kang and I.S. Moskowitz, „A Data Pump for Communication“ NRL Memo Report 5540-95-7771, 29. September 1995

[11] [Ludw11] Christian Ludwig, Bachelorarbeit, „Porting OpenBSD to Fiasco“, 22. März 2011

[12] [McCo95] John Michael McConnell, „RED/BLACK Installation Guidance“, 12 December 1995, http://cryptome.org/tempest-2-95.htm, Abgerufen: 25. Juli 2013

[13] [Meis07] Roland Meister, „GeNUGate APA One Way Gateway Covert Channel Analysis“, internes Dokument, genua mbh, CC−Revision: 1.3, 27. November 2007

[14] [Rent05] Jonathan Rentzsch, „Data alignment: Straighten up and fly right“ 8. Februar 2005, http://www.ibm.com/developerworks/library/pa-dalign/, Abgerufen: 20. August 2013

[15] [RFC768] Jonathan Bruce Postel, „User Datagram Protocol“, USC/Information Sciences Institute, 28. August 1980

[16] [RFC793] „Transmission Control Protocol“, DARPA Internet Program, Protocol Specification, September 1981

12 Glossar

DoD

Das *Department of Defense* ist das Verteidigungsministerium der Vereinigten Staaten.

Fiasco.OC

Implementation eines L4-Mikrokernels welcher an der Universität Dresden entwickelt wird.

FIFO

Als *First-In-First-Out* werden Speicherkonstruktionen bezeichnet bei denen immer das älteste Element zuerst aus einer Liste entnommen wird. Die neusten Elemente werden dem entsprechend zuletzt entnommen.

FTP

Das *File-Transfer-Protocol* ist ein Netzwerkprotokoll für den Austausch von Dateien zwischen Hosts.

genua

Die Gesellschaft für Netzwerk- und UNIX-Administration mbH ist ein Unternehmen in der nähe von München, welches Firewall- und VPN-Lösungen erstellt.

Hash

Ein *Hash* (auch Hash-Wert genannt) ist das Ergebnis einer Hash- oder Streuwertfunktion. Dieses sind Einwegfunktionen, die leicht zu berechnen aber nur schwer umzukehren sind.

IPC

Als *Inter-Process-Communication* werden alle Kommunikationsmedien bezeichnet, welche den Austausch von Informationen zwischen zwei Prozessen ermöglicht.

KiB

IEC-Symbol für Kibibyte (entspricht 1024 Byte)

NRL

Das *Naval-Research-Laboratory* ist eine Forschungseinrichtung der U.S.-Marine.

NSA

Die *National-Security-Agency* ist eine Nachrichtendienst der Vereinigten Staten, welcher sich vorrangig mit elektronischer Kommunikation beschäftigt.

PAP

Ein *Programmablaufplan* (auch Flussdiagramm genannt) ist ein Graph, welcher den Ablauf einen Computer-Programms darstellt. Diese Darstellungsform wird zur Visualisierung von Algorithmen verwendet und ist im Standard `DIN 66001` definiert.

Paravirtualisierung

Die Paravirtualisierung simuliert kein Komplettes Rechnersystem, sondern bietet den Gastsystemen Schnittstellen bereit. Daher müssen Gastsysteme immer speziell für das Hosts-System angepasst werden.

PF

Der *Packet-Filter* ist die OpenBSD Standard-Firewall. Sie ist Teil des OpenBSD-Kernels und filtert Netzwerkverkehr auf den OSI-Schichten drei bis vier.

POSIX

Das *Portable-Operating-System-Interface* ist ein Standard der die Betriebssystemschnittstelle von Anwendungen vereinheitlicht.

Salted-Hash

Ein Salted-Hash ist ein Hash, bei dem vor der Hash-Operation dem Klartext ein Zufallswert angehängt wird. Dem darauf Entstehenden Hash wird dieser Zufallswert ebenfalls angehängt, damit er später mit dem selben Klartext wieder verglichen werden kann. Dieses Verfahren wird angewandt um das Vorberechnen von Klartexten in Hash-Tabellen zu verhindern.

SMB

Das *Server-Message-Block*-Protokoll ist ein Netzwerkprotokoll welches unter anderem für die Übertragung von Dateien verwendet wird.

SMTP

Das Simple Mail Transfer Protocol ist ein Netzwerkprotokoll für das Versenden von Nachrichten zwischen verschiedenen Hosts.

TCP

Das *Transmission-Control-Protocol* ist ein verbindungsorientiertes Netzwerkprotokoll der OSI-Schicht vier.

THP

Der Trusted-High-Process ist ein Prozess welcher zwischen der roten Seite und dem internen Puffer in der NRL-Pump vermittelt. Dieser Prozess liest ausschließlich Daten aus dem Puffen sendet sie ein Ziel im roten Netz.

TLP

Der Trusted-Low-Process ist ein Prozess welcher zwischen der schwarze Seite und dem internen Puffer in der NRL-Pimp vermittelt. Dieser Prozess schreibt nur in den internen Puffer.

UDP

Das *User-Datagram-Protocol* ist ein verbindungsloses Netzwerkprotokoll der OSI-Schicht vier.

VPN

Virtual-Private-Networks sind geschlossene Netzwerkverbindungen welche durch öffentliche oder unsichere Netzwerkverbindungen verschlüsselt übertragen werden.