Hardware-basierte Sicherheitskonzepte für Teilnehmerzugangsnetzwerke

Dissertation
zur
Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)
der Fakultät für Informatik und Elektrotechnik
der Universität Rostock

vorgelegt von
Jens Rohrbeck (geb. Schulz), geb. am 21.02.1975 in Lübz,
aus Meinersen

Meinersen, den 08.05.2015

https://doi.org/10.18453/rosdok_id00002797

Gutachter der Promotionsschrift: Hardware-basierte Sicherheitskonzepte für Teilnehmerzugangsnetzwerke

Titel, akad. Grad	Vorname	Name	Postanschrift
Prof. Dr.-Ing.	Dirk	Timmermann	Universität Rostock Institut für Angewandte Mikroelektronik und Datentechnik 18051 Rostock
Prof. Dr.-Ing.	Christian	Hochberger	Technische Universität Darmstadt Elektrotechnik und Informationstechnik Merckstraße 25 64283 Darmstadt
Prof. Dr. phil. nat.	Bernd	Klauer	Helmut-Schmidt-Universität Fakultät für Elektrotechnik Technische Informatik Postfach 70 08 22 22008 Hamburg

Jahr der Einreichung: 2015
Jahr der Verteidigung: 2015

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

A

ACL ..... Access Control List
ADSL ..... Asymmetric Digital Subscriber Line
ALF ..... Application Layer Firewall
ALG ..... Application Level Gateway
ARP ..... Address Resolution Protocol
ARPA ..... Advanced Research Projects Agency
ARPANET .... Advanced Research Projects Agency Network
ASCII ..... American Standard Code for Information Interchange
ASIC ..... Application-Specific Integrated Circuit
ATM ..... Asynchronous Transfer Mode

B

BAN ..... Body Area Network
BF ..... Bloom-Filter
BRAS ..... Broadband Remote Access Server
BRAM ..... Block Random Access Memory
BSI ..... Bundesamt für Sicherheit in der Informationstechnik
BT ..... British Telecom

C

CAM ..... Content-addressable memory
CLG ..... Circuit Level Gateway
CRC ..... Cyclic Redundancy Check
CRLF ..... Carriage Return Line Feed
CPU ..... Central Processing Unit

D

DARPA ..... Defense Advanced Research Projects Agency

DCM ..... Digital Clock Manager
DDoS ..... Distributed Denial of Service
DDR ..... Double Data Rate
DFSM ..... Deterministic Finite State Machine
DNS ..... Domain Name System
DoS ..... Denial of Service
DPI ..... Deep-Packet-Inspection
dpt ..... data_path_type
DRAM ..... Dynamic Random Access Memory
DSL ..... Digital Subscriber Line
DSLAM ..... Digital Subscriber Line Access Multiplexer
DSP ..... Digital Signalprocessor

E

ECC ..... Error Correction Code
eof ..... End of Frame
ESP ..... Encapsulating Security Payload

F

FB ..... Framebuffer
FCS ..... Frame Check Sequence
FDM ..... Framedemultiplexer
FIFO ..... First In - First Out
FF ..... Flipflop
FPR ..... False Positive Rate
FPGA ..... Field Programmable Gate Array
FTP ..... File Transfer Protocol
FW ..... Firewall

G

GAN ..... Global Area Network
GbE ..... Gigabit Ethernet
GPL ..... GNU General Public License
GEMAC ..... Gigabit Ethernet Media Access Controller
GRE ..... General Routing Encapsulation
GUI ..... Graphical User Interface

H

HIDS ..... Host-basierte IDS
HTTP ..... Hypertext Transfer Protocol
HTTPS ..... Hypertext Transfer Protocol Secure

I

IANA ..... Internet Assigned Numbers Authority
ICMP ..... Internet Control Message Protocol
ID ..... Intrusion Detection
ID/IP ..... Intrusion Detection-/Intrusion Prevention
IDNA ..... Internationalizing Domain Names in Applications
IDS ..... Intrusion-Detection-System
IEC ..... International Electrotechnical Commission
IEEE ..... Institute of Electrical and Electronics Engineers
IFG ..... Inter Frame Gap
IP ..... Internet-Protokoll
IPS ..... Intrusion-Prevention-System
ISO ..... International Organization for Standardization
ISP ..... Internet Service Provider
IPv4 ..... Internet-Protokoll-Version 4
IPv6 ..... Internet-Protokoll-Version 6
IWF ..... Internet Watch Foundation

K

kbE ..... Kilobit Ethernet

L

LAN ..... Local Area Network
LCD ..... Liquid Crystal Display
LSB ..... Least Significant Bit
LUT ..... Lookup-Table

M

MAC ..... Media Access Control
MAN ..... Metropolitan Area Network
MAT ..... MAC-Address-Translation
MbE ..... Megabit Ethernet
MHz ..... Megahertz
MIG ..... Memory Interface Generator

MSB ..... Most Significant Bit
MSLW ..... Metadata Sliding Window

N

NAPT ..... Network Address Port Translation
NAT ..... Network Address Translation
NIDS ..... Netzwerk-basierte IDS
NSA ..... National Security Agency
NSLW ..... Normalized Sliding Window

O

OSI ..... Open Systems Interconnection

P

P2P ..... Peer-To-Peer
PAN ..... Personal Area Network
PC ..... Personal Computer
PCE ..... Packet Classification Engine
PCRE ..... Perl Compatible Regular Expressions
PDU ..... Protocol Data Unit
PLB ..... Processor Local Bus
PoP ..... Point-of-Presence
PPE ..... Packet Processing Engine

Q

QoS ..... Quality of Service
QoE ..... Quality of Experience

R

RAM ..... Random Access Memory
RIPE NCC .... Réseaux IP Européens Network Coordination Centre
RR ..... Resource Record
RSE ..... Rule Search Engine
RSS ..... Really Simple Syndication

S

SAN ..... Storage Area Network bzw. System Area Network
SDN ..... Software-Defined Networking
SDS ..... Software-Defined Security

SecAN ..... Secure Access Node
SFD ..... Start Frame Delimiter
SIP ..... Session Initiation Protocol
SLW ..... Sliding Window
sof ..... Start of Frame
SPI ..... Stateful Packet Inspection
SRAM ..... Static Random Access Memory
SSH ..... Secure Shell

T

TAE ..... Telekommunikationsanschlusseinheit
TCAM ..... Ternary Content-addressable memory
TCP ..... Transmission Control Protocol
TCP/IP ..... Transmission Control Protocol / Internet Protocol
TDP ..... True-Dual-Port
TLV ..... Type Length Value
TTL ..... Time to Live
TZN ..... Teilnehmerzugangsnetzwerk

U

UDP ..... User Datagram Protocol
URI ..... Uniform Resource Identifier
URL ..... Uniform Resource Locator
URN ..... Uniform Resource Name

V

VLAN ..... Virtual Local Area Network
VoIP ..... Voice over IP

W

WAN ..... Wide Area Networks
WWW ..... World Wide Web

X

XOR ..... Exklusiv-Oder

Z

ZBT ..... Zero-Bus-Turnaround

Danksagung

An dieser Stelle möchte ich die Gelegenheit ergreifen, mich bei meiner gesamten Familie zu bedanken, die mich stets aufopferungsvoll auf meinem Bildungsweg unterstützt hat.

Besonderer Dank gilt meiner Frau Diana. Sie motivierte mich stets und gab mit den nötigen Rückhalt, um in Ruhe die vorliegende Arbeit zu beenden. Meine Kinder, Annalena und Johanna, sorgten unausweichlich für die notwendigen Denkpausen.

Diese wissenschaftliche Arbeit entstand größtenteils während meiner Tätigkeit am Institut für Angewandte Mikroelektronik und Datentechnik der Universität Rostock. Die Grundlage für die Erstellung dieser Forschungsarbeit schuf mein Doktorvater Professor Timmermann durch meine Anstellung am Institut. Er gab mir stets den notwendigen Freiraum um diese Forschungsarbeit zu verwirklichen und hatte darüber hinaus stets ein offenes Ohr. Vielen Dank Professor Timmermann!

Weiterhin möchte ich die Gelegenheit ergreifen, meinen Kollegen Peter Danielis, Jan Skodzik und Vlado Altmann zu danken. Gemeinsam haben wir kreativ und innovativ gearbeitet. Ferner unterstützten sie mich maßgeblich durch Korrekturlesen der vorliegenden Forschungsarbeit.

Professor Timmermann, Peter Danielis, Jan Skodzik, Vlado Altmann, Dr. Frank Klinkenberg und Stephan Ritter möchte ich für die vielen „Motivationsschübe“ danken.

Überdies möchte ich mich beim gesamten Team der Nokia Siemens Networks GmbH & Co. KG für die harmonische und fruchtbare Kooperation der vergangenen Jahre bedanken, allen voran Herrn Dipl.-Inform. Thomas Bahls.

Außerdem möchte ich allen Studenten danken, die durch ihre wissenschaftlichen Arbeiten ihren Beitrag zum Gelingen diese Forschungsarbeit schufen.

Schlussendlich richte ich ein großes Dankeschön an alle Mitarbeiter des Instituts für Angewandte Mikroelektronik und Datentechnik, die mir ein Arbeiten in freundlicher und netter Atmosphäre ermöglichten.

Ohne die Unterstützung aller erwähnten Personen wäre die Verfassung dieser Forschungsarbeit nicht möglich gewesen. DANKE!

Kapitel 1

Einleitung

Am 18. Januar 1903 empfing der britische König Edward VII. via Funktelegramm eine Grußbotschaft vom US-Präsident Theodore Roosevelt. Mit dieser damals bahnbrechenden modernen Technologie wurde vor gut 100 Jahren die globale Kommunikation eingeläutet. Seither befindet sich der elektrotechnische Fortschritt auf der Überholspur.

Eine Vielzahl von technischen Entwicklungen führte 1962 zur Geburtsstunde des Advanced Research Projects Agency Network (ARPANET) [Sal08] - dem Vorläufer des heutigen Internet. Aus dem einst rein wissenschaftlichen, militärischen Projekt ist längst ein ubiquitäres Informationsmedium mit ständig steigenden Datenmengen geworden, die dem Moore'schen Gesetz [CO00] von 1965 folgen. Dies bestätigen die Verkehrsstatistiken z. B. des deutschen Internet-Zentralknotens in Frankfurt/Main [Dec13] und des Amsterdamer Internet-Zentralknoten [AMS11].

In der modernen Informationsgesellschaft nutzen die Anwender das Internet mit Personalcomputern, Notebooks und zunehmend auch mit anderen mobilen Endgeräten wie Smartphones und Tablets. Selbst Haushaltsgeräte kommunizieren über das Internet. Außerdem vollzieht sich ein Wandel in den Diensten wie bspw. Telefonie, Fernsehen und Radio, aber auch die klassischen Printmedien drängen zunehmend ins Internet.

Durch den beschriebenen Paradigmenwechsel steigt ständig der Bedarf an Datenrate, wie die Abbildungen 2.5 und 2.6 beweisen. Internet Service Provider (ISP) müssen mit diesem Bedarf Schritt halten, stehen jedoch gleichzeitig vor der Herausforderung, dass durch konkurrierende Flatrate-Angebote die Einnahmen stagnieren. Nach Aussage von Peter Cochrane, dem ehemaligen British Telecom (BT)-Forschungschef, fällt es den Netzbetreibern zunehmend schwerer, allein mit dem Bit-Transport Geld zu verdienen. Stattdessen müssen sie - neben der flächendeckenden Umrüstung auf Glasfaserkabel - in den Service-Markt drängen, um die permanent steigenden Kosten zu kompensieren [Sie12].

Netzwerktechnische Sicherheitslösungen können dieses Dilemma ausgleichen, wenn sie anwenderfreundlich sind und die Quality of Experience (QoE) der Internet-Nutzer nicht negativ beeinflussen.

1.1 Zielsetzung dieser Arbeit

Größere Unternehmen und Behörden besitzen in der Regel IT-Abteilungen mit entsprechenden Fachkräften. Einige dieser Fachkräfte sind Sicherheitsexperten und schützen Unternehmensdaten unter Verwendung von kostenintensivem Software- und/oder Hardware-Equipment. Diese finanziellen Aufwendungen können durchschnittliche Internet-Nutzer und kleine Unternehmen sich nicht leisten. Sie schützen ihre Netze und Daten mit dem sich angeeigneten Wissen bzw. setzen Software-Lösungen ein, die ihnen die Arbeiten abnehmen sollen. Aufgrund von Fehl- und/oder unterlassenen Konfigurationen können dann Löcher im Sicherheitsnetz entstehen, wodurch ihre Endsysteme ein besonders leichtes Zeil für Angreifer darstellen. Die polizeiliche Kriminalstatistik [hS11] bestätigt diese Aussage.

Primäres Ziel der vorliegenden Arbeit ist es, ein Sicherheitssystem zu konzipieren und prototypisch zu entwickeln, welches durchschnittliche Internet-Nutzer netzwerkseitig schützt, ohne dass deren Eingreifen erforderlich ist. Dabei muss das Sicherheitssystem eine niedrige Latenz aufweisen, damit die QoE bzgl. des Internet-Verhaltens für die Nutzer hoch bleibt. Dieser Fakt wiegt um so schwerer, weil auch in Zukunft die Datenraten weiter steigen werden. Weiterführende Anforderungen sind in Abschnitt 3.1.1 dargestellt.

Aus den genannten Gründen und unter den beschriebenen Voraussetzungen wird in der vorliegenden Forschungsarbeit ein modulares, Hardware-basiertes Sicherheitssystem für den Einsatz im Teilnehmerzugangsnetzwerk konzipiert und realisiert. Das Equipment des Internet-Nutzers ist nicht in das neue Sicherheitssystem involviert. Ziel ist es, dass Bedrohungen bereits im Teilnehmerzugangsnetzwerk (TZN) erkannt und eliminiert werden, bevor sie das Zielsystem des Internet-Nutzers erreichen. Das Sicherheitssystem wird aus den Subsystemen *Firewall*, *Web-Filter* und *Intrusion-Detection-System* bestehen. Neben dem Schutz des nutzerseitigen Netzwerks wird das Sicherheitssystem in der Lage sein, das Equipment des ISP zu schützen. Ferner wird das zu entwickelnde System sicherstellen, dass Angriffe in Richtung Internet erkannt und unterbunden werden.

Anmerkung:
Die in der vorliegenden Forschungsarbeit entwickelten Systeme erhöhen unumstritten die Netzwerksicherheit für alle angeschlossenen Endkunden. Dennoch sei darauf hingewiesen, dass die vorgestellten innovativen Technologien äußerst mächtig sind und auch missbräuchlich verwendet werden können. Prinzipiell ist es möglich, mit den beschriebenen Technologien die Privatsphäre der Internet-Nutzer zu verletzten sowie die Netzneutralität außer Kraft zu setzen. Prominente Beispiele für den missbräuchlichen Einsatz sind weltweit zu finden [Pou, Gua, CMW06]. Aus diesem Grund widmet sich die Forschungsarbeit der Thematik der Netzwerksicherheit ausschließlich vom technischen Standpunkt. Es ist nicht Ziel dieser Arbeit politische, ethische, gesellschaftliche bzw. (datenschutz)rechliche Aspekte zu betrachten.

1.2 Wesentliche wissenschaftliche Beiträge

Erstmal wird eine gesamte Security-Architektur für den Einsatz in Teilnehmerzugangsnetzwerken vorgestellt. Sie ermöglicht es dem ISP, seinen angeschlossenen Internet-Nutzern einen deutlichen Mehrwert in Richtung Security zu bieten, ohne dass die Internet-Nutzer mitwirken müssen.

Die neuartige Security-Architektur trägt den Namen **Secure Access Node** (SecAN). Ihre Leistungsfähigkeit wird durch drei prototypische Teilsysteme — den *Core-Modulen*: Firewall, Web-Filter und Intrusion-Detection-System — nachgewiesen. Vor der Entwicklung des SecAN wurden bekannte Systeme, wie bspw. das britische Web-Filtersystem „Cleanfeed“ und die *Chinesische Firewall* auf deren Schwächen untersucht. Die Ergebnisse der Untersuchungen flossen in die Konzeptphase ein, sodass der SecAN Datenströme aus Up- und Downstream-Richtung mit sehr niedrigen Latenzen verarbeiten kann und weiterhin einen bedeutend höheren Durchsatz als die bisher bekannten Systeme erzielt. Durch die Wahl bzw. den Entwurf spezieller Hardware-geeigneter Algorithmen konnte die Sicherheitsarchitektur ressourcenschonend umgesetzt werden. Ferner ist sie modular aufgebaut, wodurch sie wartungsfreundlich ist und leicht erweitert werden kann. Der modulare Charakter macht sie gleichzeitig sehr flexibel.

Neben den erwähnten *Core-Modulen*, die den Kern der Arbeit darstellen, wurden *Framework-Module*, welche den technischen Rahmen bilden, entwickelt. Die *Framework-Module* führen bspw. die Paketklassifizierung der Datenströme durch, die sonst von jedem Core-Modul separat durchgeführt werden müsste. Zusätzlich zu den Hardware-

Komponenten wurde je eine Konfigurations-Software für ISP-Administratoren und für angeschlossene Internet-Nutzer entwickelt. Die Software für den Internet-Nutzer ist optional und dient der Erweiterung der Security-Regeln des ISP.

Bereits im Vorfeld der vorliegenden Forschungsarbeit beschäftigte sich der Autor wissenschaftlich mit dem Thema *Sicherheitsmechanismen für TZN*. In diesem Zusammenhang entstanden wissenschaftliche Veröffentlichungen, welche für die Arbeit von Relevanz sind: [WKD+08, KWD+08c, KWD+08b, KWD+08a, DKW+08b, DKW+08c, DKW+09]. Ferner konnten im Rahmen der Forschungsarbeit folgende wissenschaftliche Ergebnisse erfolgreich veröffentlicht werden: [RAP+11a, RAP+11b, ARS+13]. Eine Auflistung aller wissenschaftlichen Veröffentlichungen sowie die durch den Autor betreuten studentischen Arbeiten sind in den Abschnitten: *Liste der Veröffentlichungen und Fachvorträge auf Tagungen* sowie *betreute studentische Arbeiten* zu finden.

1.3 Aufbau dieser Arbeit

Diese Arbeit gliedert sich in fünf Kapitel. Nach der Einleitung in Abschnitt 1 folgen in Abschnitt 2 zunächst die netzwerktechnischen Grundlagen sowie die notwendigen Grundlagen zur Thematik Netzwerksicherheit. Anschließend werden in Abschnitt 3 Anforderungen an das Sicherheitssystem sowie dessen Design-Konzept vorgestellt. Der Forschungsschwerpunkt dieser Arbeit liegt in Abschnitt 4. Hier werden 3 prototypische Sicherheitslösungen konzipiert, entwickelt und getestet. Besonderes Augenmerk liegt auf dem *Intrusion-Detection-System*. Abschnitt 5 fasst die Arbeit zusammen und legt die erreichten Ergebnissen dar. Abbildung 1.1 zeigt schematisch den Aufbau der vorliegenden Forschungsarbeit. Wesentliche eigene Beiträge sind in den Hauptkapiteln fett hervorgehoben.

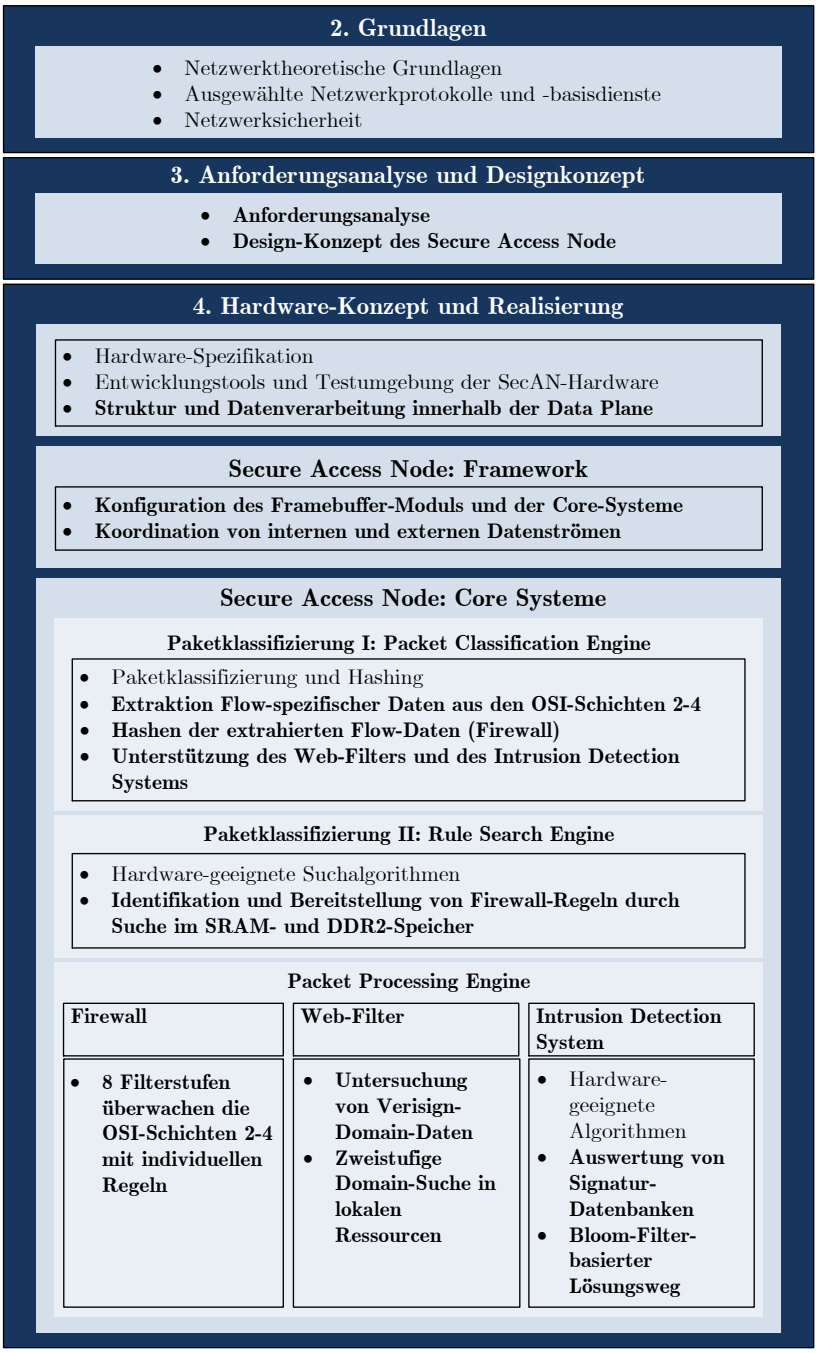


Abbildung 1.1: Struktur der Hauptkapitel der Forschungsarbeit

Kapitel 2

Grundlagen

Übergriffe auf private und unternehmerische Netzwerke gehören längst zu alltäglichen Meldungen in den Medien. Dies gilt selbst für Weltunternehmen wie Sony und Google, die ausreichend finanzielle Möglichkeiten besitzen, um sich gegen Hacker-Angriffen zu wehren. Weniger versierte und finanziell schwächere Internet-Teilnehmer möchten sich jedoch ebenfalls schützen, insbesondere wenn es sich um sensible Daten wie z. B. Bank-Logins handelt. Selbst auf dem Endverbrauchermarkt existiert eine Vielzahl von Security-Software. Jedoch bietet diese mit den Standardkonfigurationen oft nur unzureichenden Schutz. Die so erzielte „Sicherheit“ ist also äußerst trügerisch. Wenn die erforderlichen Fachkenntnisse nicht ausreichend ausgeprägt sind, können Fehlkonfigurationen zu Schlupflöchern in Netzwerken führen, welche von Angreifern ausgenutzt werden können.

Nachfolgend aufgeführte Grundlagen werden den Leser einerseits für die Problematik Netzwerksicherheit sensibilisieren und andererseits helfen, ein fundamentales Verständnis über die Funktionsweise von Computernetzwerken zu erlangen. In diesem Kapitel werden die netzwerktechnischen Grundlagen erörtert. Darüber hinaus wird ein Überblick über die Thematik Netzwerksicherheit sowie derzeit eingesetzte Sicherheitsmaßnahmen gegeben.

2.1 Netzwerktechnische Grundlagen

2.1.1 Referenzmodelle

Damit die Interoperabilität über unterschiedlichste Netzwerkformen hinaus nicht beeinträchtigt wird, wurden grundlegende Richtlinien und Spezifikationen entwickelt. Diese

regeln den Kommunikationsablauf sowohl in homogenen als auch in heterogenen Netzwerken. Die International Organization for Standardization (ISO) war federführend bei der Konzipierung des OSI-Referenzmodells [BW02]. Es dient als Design-Grundlage für Kommunikationsprotokolle und besteht, wie in Abbildung 2.1 dargestellt, aus sieben aufeinanderfolgenden Schichten (Layer).

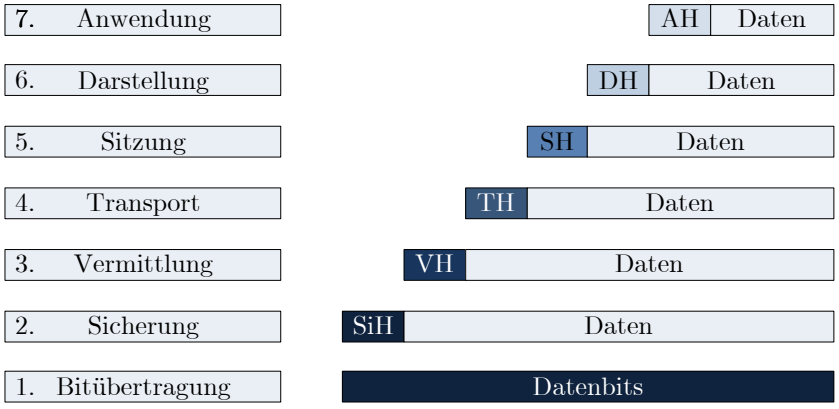


Abbildung 2.1: ISO-OSI-Referenzmodell [BW02]

Durch das Schichtenmodell entstehen notwendige Abstraktionsebenen mit definierten Schnittstellen zu den benachbarten Layern. In jeder Schicht existieren Protokolle mit verschiedenen Aufgaben. Dabei nimmt ein Protokoll Dienstleistungen der darunterliegenden Schicht in Anspruch und bietet seinerseits einen Dienst für die darüberliegende Schicht an.

In jeder Schicht werden abgeschlossene Datenpakete, sog. Protocol Data Units (PDUs), verarbeitet. PDUs bestehen immer aus einem Kopf (Header) und einer Nutzlast (Payload) (siehe Abbildung 2.1). Während die Header eine definierte Struktur mit Steuerinformationen der aktuellen Schicht beinhalten, besitzt die Payload einen variablen, schichtunabhängigen Datenteil. Einige PDUs besitzen darüber hinaus einen sog. Trailer. Dieser wird für die Datenprüfung der jeweiligen Schicht verwendet. Da die Nutzdaten vom Header und einem ggf. vorhandenen Trailer gekapselt werden, können Header und Trailer als Overhead bezeichnet werden. Weiterführende Informationen zum OSI-Referenzmodell sind neben den aufgeführten Quellen in [Int84, FK95, Lea02] zu finden.

Da die OSI-Schichten ein rein konzeptionelles Model mit definierten Aufgaben für jeden Layer darstellen, regeln Protokolle den Kommunikationsablauf im Speziellen [Tec07b]. Als wesentlichen Kritikpunkt am OSI-Referenzmodell stellt Tanenbaum [Tan03] dessen

enorme Komplexität fest. Inkompatibilitäten der verschiedenen Subnetze störten bereits die Zusammenarbeit im ARPANET. Eine neue Netzwerkarchitektur, welche die primär verwendeten Protokolle beinhaltet, sollte diese Störungen verhindern. Die Transmission Control Protocol / Internet Protocol (TCP/IP)-Protokollfamilie diente als Basis der gleichnamigen Architektur. Sie wurde Anfang der 1970er Jahre von der Advanced Research Projects Agency (ARPA) entwickelt. Es handelt sich bei der ARPA um eine US-amerikanische Forschungsbehörde, die 1996 in Defense Advanced Research Projects Agency (DARPA) umbenannt [Lei99] wurde. Das TCP/IP-Modell stellt eine robuste und flexible Architektur dar. Darüber hinaus können das TCP/IP-Modell und das OSI-Referenzmodell, wie in Abbildung 2.2 gezeigt, gegenübergestellt werden.

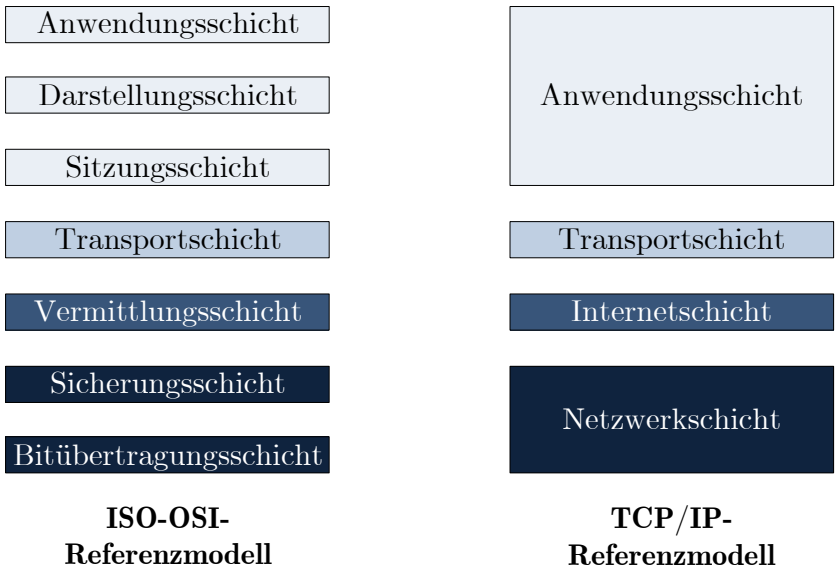


Abbildung 2.2: Vergleich zwischen dem OSI- und dem TCP/IP-Referenzmodell

Die oberen drei Schichten der TCP/IP-Architektur werden zu einer Anwendungsschicht vereint. In den unteren Schichten befindet sich die:

- Transportschicht: Sie entspricht im OSI-Modell ebenfalls der Transportschicht
- Internetschicht: Sie entspricht im OSI-Modell der Vermittlungsschicht
- Netzwerkschicht: Sie entspricht im OSI-Modell der Bitübertragungs- und Sicherungsschicht

Detaillierte Informationen zum TCP/IP-Modell sind in [Tec07b, Lei99] zusammengestellt. Da für die späteren Ausführungen ein Grundverständnis für bestimmte Netz-

werkprotokolle vorausgesetzt wird, müssen die für diese Arbeit wichtigen Protokolle im Folgenden näher betrachtet.

2.1.2 Computernetzwerke

Ein Computernetzwerk besteht aus einem Verbund von mindestens zwei autonomen Endgeräten. Deren Ziel ist es, zum Zweck des Datenaustausches eine Kommunikationsbeziehung aufzubauen. Aus diesem Grunde werden Computernetze auch als Kommunikationsnetze bezeichnet. Kommunikationsnetze können nach Netzwerktopologie, Übertragungstechnik und Übertragungsreichweite klassifiziert werden. Diese Klassifizierung erfolgt im Anhang A.

2.1.2.1 Teilnehmerzugangsnetzwerke

ISP-Endkunden werden über aufwendige und intelligente Teilnehmerzugangsnetzwerke mit den hochgradig leistungsfähigen und schnellen Kernnetzen verbunden. Während die Netzwerkarchitekturen auf Teilnehmerseite sehr heterogen ausfallen können, ist das Kernnetz eher ein uniformes Netz. Das TZN verbindet beide Netzstrukturen und stellt dabei verschiedenste Services zur Verfügung. Noch vor einigen Jahren herrschten im TZN hauptsächlich Asynchronous Transfer Mode (ATM)-basierte Formen vor. Aufgrund der Einfachheit und Robustheit sowie der geringeren Kosten verdrängten ethernetbasierte Formen die ATM-basierten Formen [REA05].

Durch die unmittelbare Nähe zum Endkunden bildeten sich zwei wichtige Begriffe für ein TZN. Abhängig von der Transferrichtung wird es in Upstream-Richtung (Daten in Richtung Kernnetz) als „First Mile“ und in Downstream-Richtung (Daten aus dem Kernnetz) als „Last Mile“ bezeichnet. Als Verallgemeinerung für beide Begriffe und weil Teilnehmern der Zugriff zum Internet ermöglicht wird, herrschen weiterhin die Bezeichnungen „Access-Bereich“ und Zugangsbereich vor. Letztere Bezeichnungen werden in dieser Arbeit völlig gleichberechtigt benutzt.

Gewöhnlich weisen TZN eine Baumstruktur mit verschiedenen Aggregationsstufen auf. Abbildung 2.3 zeigt am Beispiel eines typischen Asymmetric Digital Subscriber Line (ADSL)-Anschlusses der Deutschen Telekom AG die vorherrschenden Hierarchiestufen innerhalb eines TZN. Die Abkürzungen Kilobit Ethernet (kbE), Megabit Ethernet (MbE) und Gigabit Ethernet (GbE) in Abbildung 2.3 beschreiben die vorherrschenden Datenraten in den jeweiligen Abschnitten.

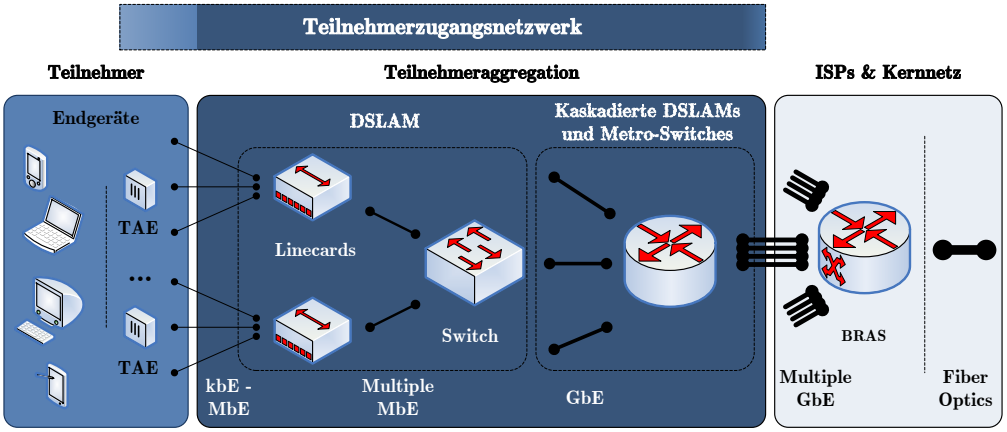


Abbildung 2.3: Architektur von Teilnehmerzugangsnetzwerken

Auf Teilnehmerseite werden die Endgeräte meist mittels Switches, Router oder eines Digital Subscriber Line (DSL)-Modems mit einer Telekommunikationsanschlusseinheit (TAE) verbunden. Sie stellt auf Teilnehmerseite den Ein- bzw. Austrittspunkt vom bzw. zum TZN dar.

Digital Subscriber Line Access Multiplexers (DSLAMs) existieren in verschieden großen Ausprägungen und bilden die erste Aggregationsstufe. Sie können mehrere Linecards aufnehmen und konzentrieren bis zu hundert Teilnehmeranschlüsse je Linecard. Die folgende Aggregationsstufe besteht aus Switches, welche die zu transportierenden Signale zu hochvolumigen Datenströmen kumulieren. Letztendlich terminieren die hoch aufkonzentrierten Anschlüsse außerhalb des TZN an einem Broadband Remote Access Server (BRAS). Der BRAS ist die erste Instanz eines ISP und bildet somit den sogenannten Point-of-Presence (PoP). Über ihn lassen sich Verbindungen durch den Kernbereich zu anderen ISP-Netzen etablieren. Während die Datenweiterleitung im TZN über Adressen der Sicherungsschicht erfolgt, wird im Kernnetz weitestgehend auf der Vermittlungsschicht weitergeleitet (geroutet).

2.1.2.2 Internet

Das Internet ist ein weltumspannendes Rechnernetzwerk (Global Area Network (GAN)). Es vereint alle Rechner aus Teilnetzen zu einem globalen Netzwerk (vgl. Abbildung 2.4). Die Kommunikation innerhalb der Teilnetze (häufig Local Area Networks (LANs)) erfolgt zumeist über reservierte IPv4-Adressen nach [The10], die im Internet nicht ge-

routet werden. Ab dem TZN, wie auch im Internet selbst, müssen öffentliche Internet-Protokoll (IP)-Adressen verwendet werden. Dazu vergibt die Internet Assigned Numbers Authority (IANA) öffentliche IP-Bereiche an die Regional Internet Registries (RIR).

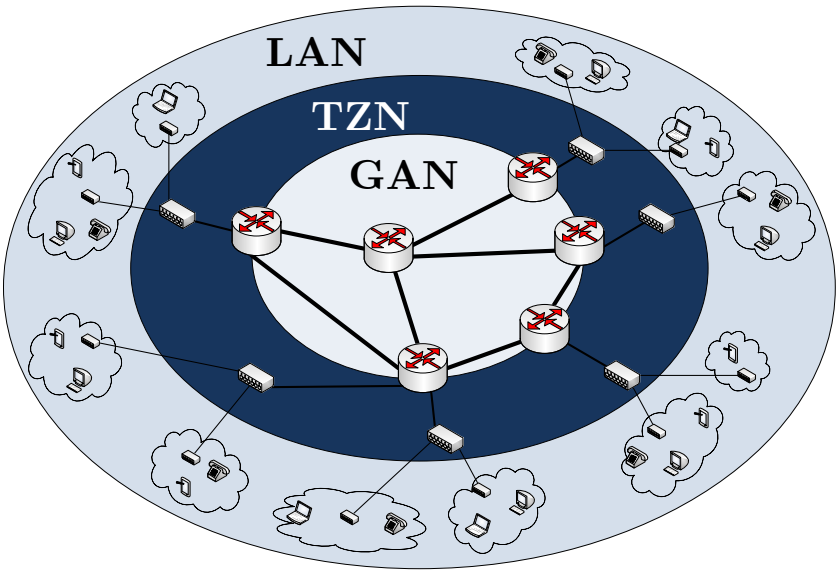


Abbildung 2.4: Netzwerkarchitektur des Internet

Für den europäischen und zentralasiatischen Raum sowie den Nahen Osten ist die Réseaux IP Européens Network Coordination Centre (RIPE NCC) die entsprechende regionale Organisation. Sie bedient neben Hochschulen und Großunternehmen vor allem auch ISP mit IP-Adressen. ISP weisen angeschlossenen Kunden eine IP-Adresse aus ihrem zugeordneten Pool zu. Da die weitverbreiteten IPv4-Adressen bereits aufgebraucht sind, werden diese Adressen meist vorübergehend zugewiesen. Erst wenn ein ISP-Kunde eine öffentliche IP-Adresse erhalten hat, kann er wie in Abbildung 2.4 dargestellt, Daten über das Internet austauschen.

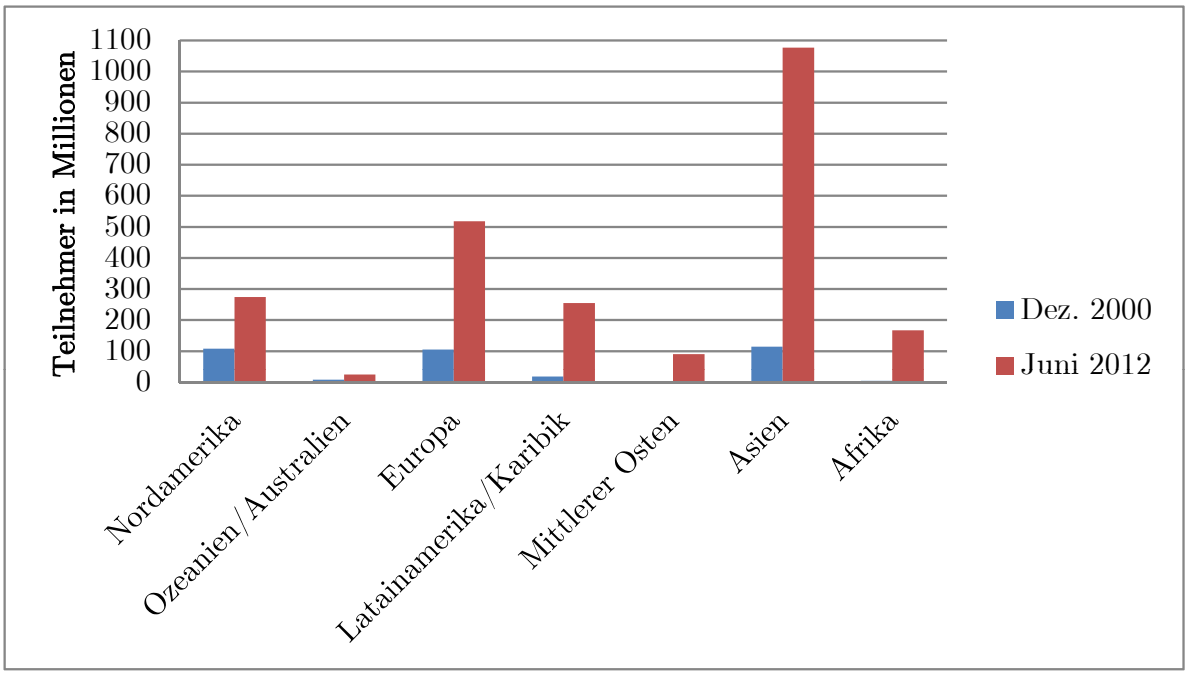


Abbildung 2.5: Wachstum des Internet zwischen den Jahren 2000 und 2012 (Quelle: [Intb])

Der ständig steigende Datenaustausch und damit die wachsende Bedeutung des Internet lässt sich an den Statistikdaten der „Internet World Stats“ aus Abbildung 2.5 ablesen. Seit der Jahrtausendwende bis zum Sommer 2012 wuchs das Internet weltweit durchschnittlich um mehr als 566 % von ca. 361 Millionen im Dezember 2000 auf ca. 2,4 Milliarden Teilnehmer im Dezember 2012 (vgl. Abbildung 2.5). Besonders stark wirken die Regionen Afrika, Mittlerer Osten sowie Lateinamerika und die Karibik am Wachstum mit. Auch in Europa ist das Internet im selben Zeitraum annähernd um den Faktor 5 gewachsen [Intb].

Mit dem Anstieg der Nutzerzahlen und aufgrund der Entwicklung des Internet wächst permanent der Bandbreitenbedarf. Dies spiegeln Messungen des größten deutschen Internet-Knoten DE-CIX in Frankfurt/Main [Dec13] wieder. Abbildung 2.6 stellt lediglich einen Auszug von 5 Jahren dar, verdeutlicht jedoch das jährliche Wachstum des Verkehrsaufkommens um den Faktor 2. Dieser Trend hält bereits seit mehreren Jahren an.

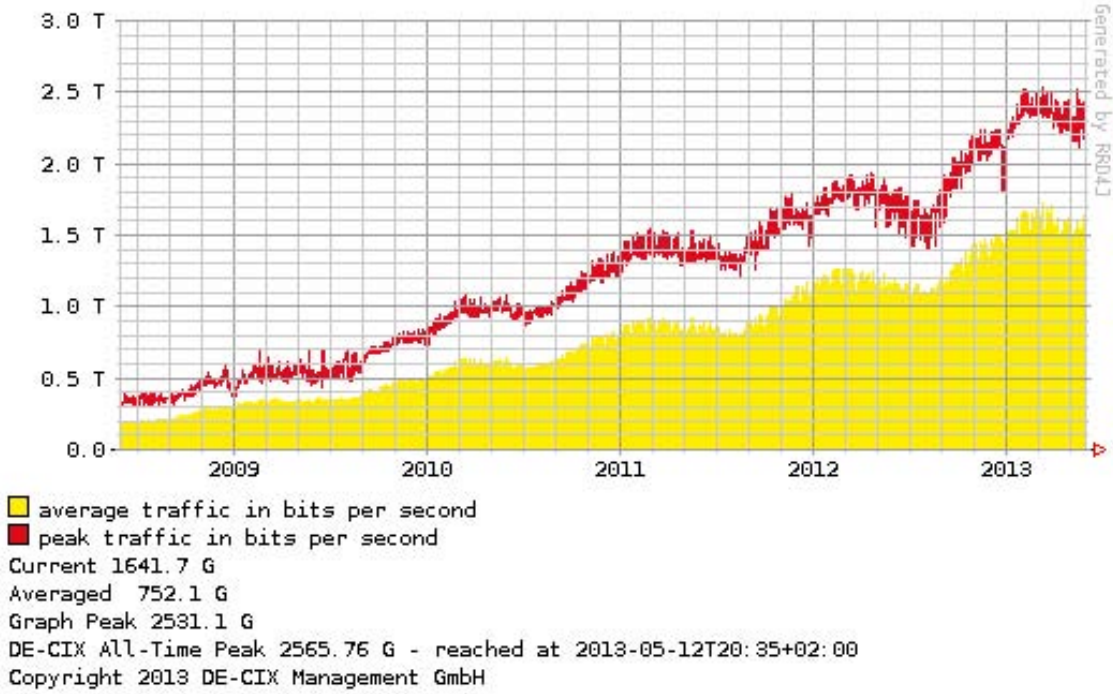


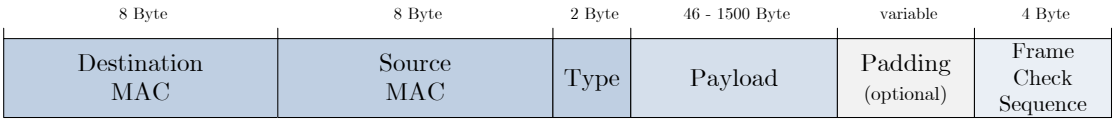
Abbildung 2.6: Verkehrsaufkommen am größten europäischen Internetknotenpunkt DE-CIX in Frankfurt/Main (Quelle: [Dec13])

2.2 Ausgewählte Netzwerkprotokolle

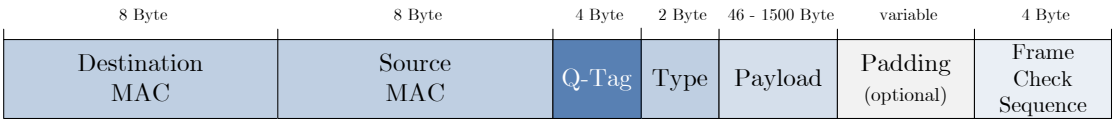
2.2.1 Ethernet

Das Ethernet-Protokoll wurde in der Institute of Electrical and Electronics Engineers (IEEE)-Norm 802.3 standardisiert. Seit Start des Projektes IEEE 802 hat es sich als das Standardprotokoll in lokalen und globalen Netzen durchgesetzt [IEE01]. Im Jahre 2008 erschien die vorerst letzte Aktualisierung des Standards IEEE 802.3 [IEE08]. In Bezug auf das OSI-Referenzmodell ist Ethernet auf den Layer 1 und 2 definiert.

PDUs, die auf Schicht 2 im OSI-Modell transportiert werden, werden als Frames bezeichnet. Im Falle von Ethernet besitzen diese PDUs, neben dem Header und der Payload, einen Trailer. Die Gesamtlänge eines Ethernet-Frames ist zwischen 64 und 1522 Byte definiert. Wie in Abbildung 2.7 gezeigt, folgt der standardisierte Ethernet-Frame zwei Versionen.



(a) Standard Ethernet-Frame nach IEEE Std 802.3 [IEEE08]



(b) Tagged Ethernet-Frame nach IEEE Std 802.1Q [IEEE06a]

Abbildung 2.7: Aufbau von Ethernet-Frames nach IEEE Standard

Zusätzlich zum Ethernet-Frame wird eine 7 Byte lange Präambel und ein 1 Byte langer Start Frame Delimiter (SFD) übertragen. Präamble und SFD sind abhängig vom Übertragungsmedium und befinden sich auf OSI-Layer 1. Beide werden heute nicht mehr benötigt, sind jedoch weiterhin aus Kompatibilitätsgründen vorhanden.

Ziel- und Start-Adresse: Der Header eines Ethernet-Frames beginnt mit der Zieladresse (Destination Media Access Control (MAC)) gefolgt von der Quelladresse (Source MAC). Beide Adressen sind 6 Byte lang.

Typ-Feld und Q-Tags: Im Folgenden Aufbau wurden 2 Varianten spezifiziert:

1. Der Basic-Ethernet-Frame: Auf die MAC-Adressen folgt das *Typ-Feld*. Es ist 2 Byte lang. Ist der codierte Wert kleiner 0x0600 (hexadezimal), so gibt der Wert Auskunft über die Länge des Ethernet-Frames. Werte größer 0x0600 (hexadezimal) geben Auskunft über das Protokoll der nächsthöheren Schicht. Ein Vertreter ist das Internet-Protokoll in den Versionen 4 und 6 [IEE08].

2. Der Tagged-Ethernet-Frame: Es handelt sich dabei um einen besonderen Ethernet-Frame. In den Frame wurde ein zusätzliches Feld, das sog. Q-Tag, eingefügt. Q-Tags wurden standardisiert [IEE06b], um logische Netzwerke innerhalb eines physischen Netzwerkes zu erzeugen. An der Position des Typ-Feldes ist für sie die hexadezimale Marke 0x8100 reserviert. Für die genaue Differenzierung der virtuellen Netze - der VLANs - stehen im Q-Tag 12 Bit zur Verfügung. Somit können $2^{12} = 4096$ Virtual Local Area Networks (VLANs) adressiert werden. Um die Anzahl der VLANs zu erhöhen, lassen sich Q-Tags kaskadieren. Diese Technik wird vor allem im TZN eingesetzt [IEE06a]. Im Anschluss an die Q-Tags folgt das Typ-Feld wie in Punkt

1 beschrieben.

Nutzlast und Padding Bytes: Ein Nutzdatenblock darf per Definition die Länge 1500 Byte nicht überschreiten. Durch das vorangestellte Typ-Feld entsteht eine eindeutige Interpretation der in der Nutzlast mitgeführten Daten. Unterschreiten die Nutzdaten den Minimalwert von 46 Byte, so müssen die Nutzdaten um Padding Bytes bis zum Minimalwert erweitert werden. *Padding Bytes* werden nicht interpretiert, sie dienen lediglich der Erkennung von Kollisionen in der jeweiligen *Collision Domain*.

Frame Check Sequence: Im Anschluss an den Nutzdatenblock folgt die Frame Check Sequence (FCS). Es handelt sich dabei um eine CRC32-Prüfsumme. Sie wird vom Sender erstellt und erstreckt sich über den Ethernet-Header und die Ethernet-Payload. Präambel, SFD und FCS werden bei der Berechnung nicht berücksichtigt. Die berechnete Prüfsumme wird an den Nutz- bzw. Padding-Datenblock angehängt. Der Empfänger führt dieselbe Berechnung wie der Sender durch. Stimmt der errechnete CRC32-Wert mit der Prüfsumme des Ethernet-Frames überein, geht der Empfänger von einer fehlerfrei empfangenen Nachricht aus. Weiterführende Informationen zur Struktur und zum Aufbau der Datenfelder von Ethernet-Frames können dem Standard [IEE01] sowie [Spr00, Hel03, Axe03] entnommen werden.

2.2.2 Internet-Protokoll

Während das Ethernet-Protokoll das Standardprotokoll in den OSI-Layern 1 und 2 ist, hat sich das Internet-Protokoll als zentrales Protokoll auf der Vermittlungsschicht im OSI-Modell, bzw. der Internet-Schicht im TCP/IP-Modell, durchgesetzt. Das Internet Protokoll wird derzeit in den beiden Versionen 4 und 6 verwendet. Die Version 4 des Internet Protokolls folgt dem Standard [INF81] von 1981 und die Version 6 dem Standard [The98] von 1998. Auf dem Weg zur Version 6 wurde das Internet Protokoll Version 5 unter dem Namen Internet Stream Protocol Version 2 in [INF95] spezifiziert. Die 5. Version des Protokolls konnte sich jedoch aufgrund des geringen Nutzens nicht durchsetzen und wird deshalb nicht verwendet.

Allgemein verbindet das Internet-Protokoll unabhängige Netzwerke über logische Adressen, sog. IP-Adressen, miteinander. Es ist zustandslos, arbeitet nach dem „best effort“-Ansatz, kapselt die verschiedensten Protokolle, leitet Datagramme innerhalb desselben Netzes bzw. zu anderen Netzen weiter und spiegelt somit grundlegende Charakteristika des Internet [Car96] wieder.

Datagramme im IP werden typischerweise als Pakete bezeichnet. In Abbildung 2.8 ist der grundlegende Aufbau eines IPv4-Paketes dargestellt [Tec07b].

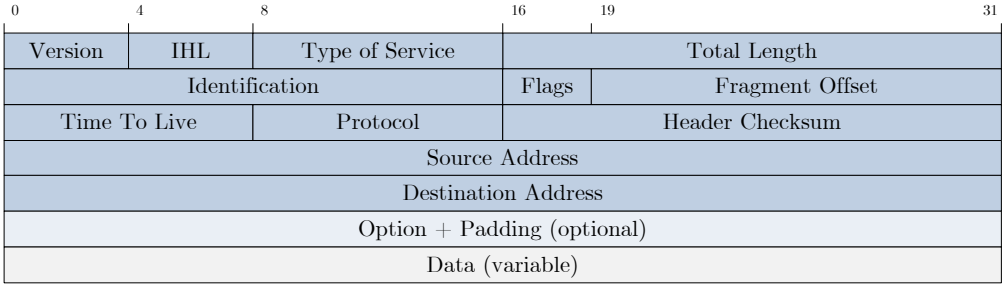


Abbildung 2.8: Aufbau eines IPv4-Pakets

Die für diese Arbeit wichtigsten Felder eines IPv4-Paketes sind:
Ver: Versionsangabe zum verwendeten Protokoll (IPv4, IPv6)
Protocol Number: Nummer des gekapselten Protokolls
Source- und Destination: Adresse des Absenders bzw. Empfängers
Data: Daten des gekapselten Protokolls

Der Wechsel zur Internet-Protokoll-Version 6 (IPv6) wurde aus mehreren Gründen notwendig. Zum einen spielt die zunehmende Adressverknappung eine wichtige Rolle. Die IANA - als verantwortlicher Koordinator des IP-Adresssystems [Inta] - bestätigte, dass alle IPv4-Adressen seit dem 31. Januar 2011 ausgeschöpft seien [HEIa].

IPv6 wurde als Nachfolger des Protokolls IPv4 konzipiert. Dieses wurde notwendig, da das Internet gerade in der jüngeren Vergangenheit ein enormes Wachstum erlebte (vgl. Abbildung 2.5) und sich darüber hinaus seit seiner Geburtsstunde grundlegend geändert hat. Durch die zunehmende Beliebtheit des Mediums wurden einerseits die adressierbaren Endgeräte knapp, andererseits kamen zu den bestehenden Diensten neue, z. B. echtzeitfähige Dienste, hinzu. Diese Ansprüche und weitere wichtige Randbedingungen [BM95] sollte das neue Protokoll berücksichtigen. Zu den wichtigsten Änderungen zählen:

- Der Adressraum wurde von 32 Bit auf 128 Bit erweitert. Dadurch lassen sich statt wie bisher $2^{32} \approx 4,3 \cdot 10^9$ nun $2^{128} \approx 3,4 \cdot 10^{38}$ Endgeräte adressieren. Darüber hinaus wurde die Hierarchie der Adresssierungsebenen vergrößert.

- Um eine beschleunigte Paketverarbeitung zu ermöglichen, sind einige Frame-Parameter des alten IPv4-Headers in der neuen Version als optional gekennzeichnet worden bzw. wurden komplett verworfen.
- In der Version 4 des Internet-Protokolls gab es bereits ein „Options“-Feld. Dieses Feld war auf eine Länge von 40 Byte limitiert. Um die Flexibilität in Bezug auf Erweiterungen und Optionen zu erhöhen, wurde die „Header“-Struktur optimiert.
- Ein neues 20 Bit langes „Flow Label“-Feld ermöglicht die Beeinflussung des Handlings, während das Paket transportiert wird. Auf diese Weise findet eine Verbesserung in Bezug auf Quality of Service (QoS) und Echtzeitfähigkeit des neuen Protokolls statt.

Die Anforderungen an das neue Protokoll wurden in [The98] umgesetzt und resultieren in dem angepassten IPv6-Paket, wie es in Abbildung 2.9 dargestellt ist.

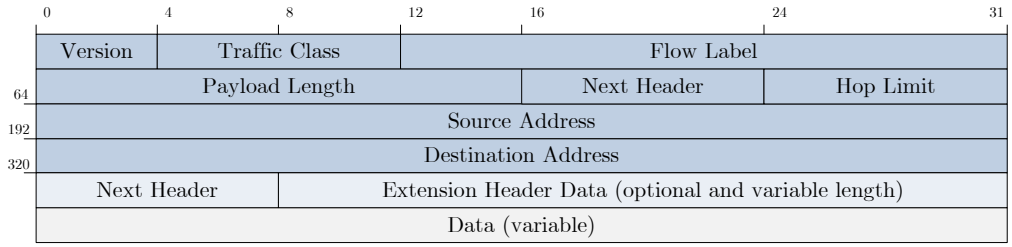


Abbildung 2.9: Aufbau eines IPv6-Pakets

2.2.3 User Datagram Protocol

Das User Datagram Protocol ist ein Protokoll der Transportschicht (vgl. Abbildung 2.1). Es handelt sich um ein „verbindungsloses“ Protokoll, da weder ein Verbindungsaufbau noch ein Verbindungsabbau erfolgt. Die Kommunikation über User Datagram Protocol (UDP) ist unzuverlässig, da keine Fehlerbehebungs- und keine Maßnahmen zur Flusskontrolle über empfangene Netzwerkpakete existieren. Der Datenaustausch erfolgt mittels einfacher Datagramme.

Eine Vielzahl von Netzwerkdiensten können auf demselben Computer koexistieren und dabei UDP als Kommunikationsprotokoll nutzen. Damit es nicht zu Engpässen kommt, werden den Diensten Protokoll-Ports zur Verfügung gestellt. Dabei nutzt die anfragende

Anwendung den „Source-Port“. Die Zielanwendung wird über den „Destination-Port“ erreicht. Der Aufbau eines UDP-Pakets nach [Pos80] ist in Abbildung 2.10 dargestellt.

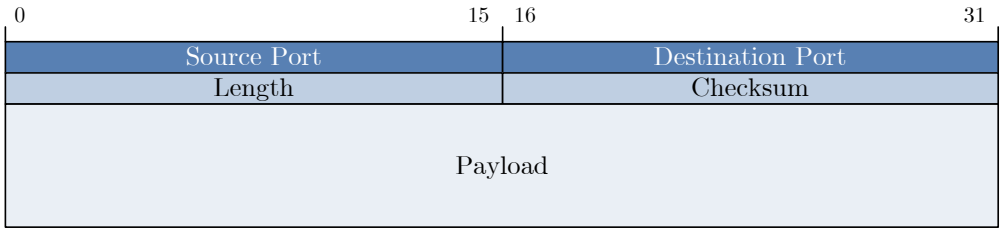


Abbildung 2.10: Aufbau eines UDP-Pakets

Dieses Protokoll kann eingesetzt werden, wenn höhere Protokolle die Fehlerbehebung und Flusskontrolle übernehmen. Ferner ist UDP für zeitkritische Dienste wie z. B. Voice over IP (VoIP) bzw. Netzwerkdienste geeignet, die mit wenigen Paketen auskommen wie z. B. DNS. UDP besticht durch seine Einfachheit und damit durch weniger „Overhead“ als bspw. Transmission Control Protocol (TCP).

2.2.4 Transmission Control Protocol

Genau wie UDP ist das Transmission Control Protocol (TCP) ein Protokoll der Transportschicht (vgl. Abbildung 2.1). Im Gegensatz zu UDP bietet TCP jedoch zuverlässige Datenübertragungen. Dazu wird im Vorfeld eine Verbindung etabliert. Anschließend werden die Daten in Form von TCP-Segmenten übertragen. Jedes Segment besitzt eine Sequenznummer, die vom Empfänger bestätigt werden muss. Fehlt eine erwartete Sequenznummer, wird dieses TCP-Segment erneut geordert. Über die Reihenfolge von Sequenznummern lässt sich ein Datum in der richtigen Reihenfolge „reassemblieren“.

Genau wie beim UDP kommunizieren die Netzwerkanwendungen bei TCP über Protokoll-Ports („Source-Port“ und „Destination-Port“, vgl. 2.11). Ein Tupel, bestehend aus IP-Adresse und Port-Nummer, wird allgemein als „Socket“ bezeichnet. Die Verwendung von Sockets ermöglicht es TCP, eine zuverlässige Kommunikation zu gewährleisten. Abschließend erfolgt der Abbau einer Verbindung.

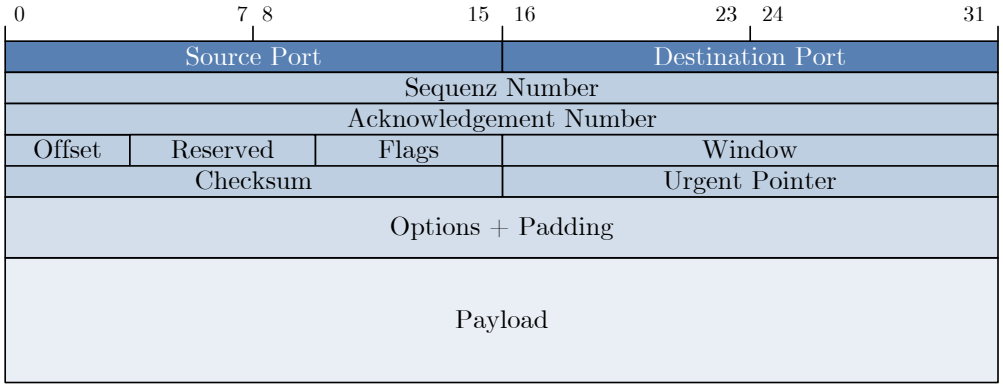


Abbildung 2.11: Aufbau eines TCP-Pakets

2.3 Ausgewählte Netzwerk-Basisdienste

2.3.1 HTTP

Das Hypertext Transfer Protocol ist ein Protokoll der Anwendungsschicht (Schicht 7 im ISO/OSI-Referenzmodell; vgl. Abbildung 2.1)[Tec07b]. Es wird seit 1990 im World Wide Web (WWW) eingesetzt. 1999 wurde die heute verwendete Version HTTP/1.1 in [The99] spezifiziert.

Die Kommunikation zwischen Client und Server kann über Zwischenstationen wie Proxies und Gateways, aber auch durch Tunnel geleitet werden. Während Proxies als Speicher- und Weiterleitungsknoten dienen und Teile der zu übertragenden Nachricht überschreiben können, dient ein Gateway als Empfangsknoten, welcher eine Anfrage in ein anderes Protokoll umwandelt und anschließend weiterleitet. Demgegenüber wirkt ein Tunnel als Übermittlungsstelle zwischen zwei Netzwerkknoten. Beim sendenden Netzwerkknoten werden komplette IP-Pakete verschlüsselt und anschließend die verschlüsselten Daten in ein oder mehrere neue IP-Pakete gekapselt. Der empfangende Knoten decodiert und reassembliert die Datenpakete und erhält so die Originalnachricht.

HTTP ist ein Anfrage-Antwort-Protokoll. Über 8 HTTP-Request-Methoden können Webbrowser Informationen von Webservern abfragen. Für die vorliegende Arbeit ist ausschließlich die GET-Methode von Bedeutung. Sie fordert unter Angabe eines Uniform Resource Identifier (URI) Ressourcen vom Server an. Ein URI kann ein Name - Uniform Resource Name (URN) - oder ein Ort (Location) - Uniform Resource Locator (URL) - sein und bestimmt eine Ressource, auf die eine Methode angewendet werden soll.

2.3.1.1 Aufbau und Codierung von Uniform Resource Locators

URLs werden im HTTP verwendet, um Netzwerkressourcen zu adressieren. Der Aufbau einer HTTP-URL ist in 4 Sektionen aufgeteilt und folgt dabei dem nachfolgenden Schema:

http://*hostname*[:*portnummer*][*absolute_server_path*[?*query*]]

Im ersten Bereich befindet sich das Übertragungsprotokoll „http://“. Darauf folgend wird der anzusprechende „Ziel-Hostname“ angegeben. Die Sektionen 3 und 4 können optional verwendet werden. Typischerweise wird die in Sektion 3 angegebene Port-Nummer mit dem Standardwert 80 belegt. In dem Fall braucht die Port-Nummer nicht angegeben zu werden. Soll über einen anderen als den Standard-Port kommuniziert werden, muss der Kommunikationsport angegeben werden. Zusätzlich können der absolute Pfad der Ressource auf dem Ziel-Host (absolute_server_path) sowie weitere Parameter übergeben werden. Die Angabe des absoluten Pfades wird notwendig, wenn sich die angesprochene Ressource bspw. im Unterverzeichnis „pictures“ befindet. Dann lautet die Pfadangabe: „/pictures/foto.jpg“. Des Weiteren können der Ressource, abgetrennt durch ein „?“ Anfrageparameter übergeben werden. Diese folgen auf das „?“ und werden als „Variablenname=Wert“ übergeben. Die einzelnen Ressourcen werden mit einem „&“ verknüpft.

Leider existiert keine international einheitliche Codierung für URLs. Besondere Aufmerksamkeit muss deshalb auf Zeichen gelegt werden, die als Separatoren innerhalb der URL verwendet werden und sich außerhalb des zulässigen Zeichensatzes befinden. Diese Zeichen müssen umcodiert werden. Typischerweise wird dafür das sogenannte „Percent-Encoding“ [For05] eingesetzt. Das Percent-Encoding encodiert jedes Byte durch drei Zeichen. Das erste Zeichen entspricht dem „%“-Zeichen gefolgt von zwei hexadezimalen Zahlen, die den numerischen Wert des Zeichens repräsentieren. Ein Leerzeichen wird in Percent-Encoding z. B. als „%20“ (American Standard Code for Information Interchange (ASCII)-Wert 0x20) dargestellt.

2.3.1.2 Aufbau von HTTP-Nachrichten

HTTP-Nachrichten werden zwischen Client und Server ausgetauscht. Abbildung 2.12 zeigt den nach [The99] definierten generischen Aufbau einer HTTP-Nachricht.

Sie beginnt mit einer Startzeile (Start Line) gefolgt von optional vorhandenen Headern. Die Startzeile nimmt Anfrage- oder Statusinformationen auf und darf anders als die Hea-

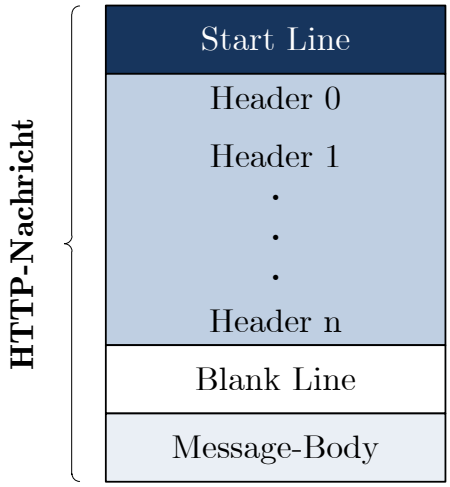


Abbildung 2.12: Aufbau einer HTTP-Nachricht

der nicht entfallen. Sowohl Startzeile als auch Header werden durch eine Endekennung (Carriage Return Line Feed (CRLF)) terminiert. Bevor die HTTP-Nachricht mit einem optionalen „Message Body“ endet, wird diesem eine zusätzliche Endekennung (Leerzeile) vorangestellt. Der Message Body ist bei Anfragen nicht vorhanden, trägt jedoch bei Antworten den darzustellenden Inhalt.

2.3.1.3 Aufbau von HTTP-Anfragen

Für die vorliegende Arbeit ist die Startzeile von HTTP-Nachrichten von elementarer Bedeutung. Ihr Aufbau ist standardisiert und in Abbildung 2.13 dargestellt.

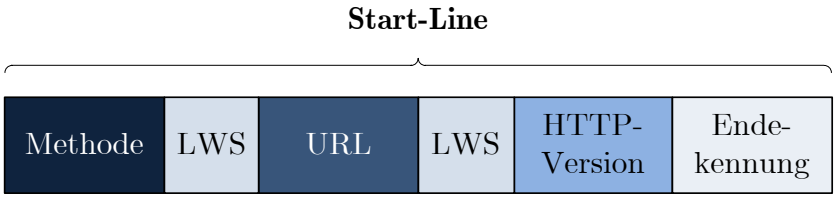


Abbildung 2.13: Aufbau einer HTTP-Anfragezeile

Eine HTTP-Anfragezeile beginnt mit der Anfrage-Methode, gefolgt von der angefragten URL sowie der verwendeten HTTP-Version. Separiert werden die drei Tupel durch Leerzeichen bzw. Tabulatoren - sogenannte „Linear White Spaces“ (LWS). Die Anfragezeile

schließt mit einer Endekennung.

Es existiert eine Reihe von Anfrage-Methoden. Die Methoden **HEAD** und **GET** müssen von allen Servern unterstützt werden. Mit Hilfe der HEAD-Methode werden Meta-Informationen über eine Ressource vom Server angefragt. Dabei wird die Ressource selbst nicht geladen, d. h. es existiert kein Nachrichten-Body. Soll hingegen die Ressource geladen werden, wird die GET-Methode verwendet. Für die vorliegende Arbeit besitzt die Methode GET größte Bedeutung. Alle weiteren Anfrage-Methoden sind optional. Ausführliche Informationen zu allen Anfrage-Methoden sind in [The99] nachzulesen.

Im Anschluss an die Anfrage-Methode folgt die URL. Sie beschreibt die Quelle der angeforderten Ressource und kann sowohl in absoluter als auch in relativer Form angegeben werden. Absolute URL-Angaben werden z. B. bei Anfragen über Proxy-Server verwendet und können wie folgt aussehen:

GET http://www.w3.org/pub/WWW/TheProject.html HTTP/1.1

Häufiger genutzt ist hingegen die relative Form. Diese wird vom Client benutzt, um ohne Proxy die Ressource direkt vom Server anzufordern. In diesem Fall befindet sich die absolute Pfadangabe der Ressource im URL-Bereich der Anfrage. Darüber hinaus wird der Host-Name in einem separaten Header - dem Host Header - übertragen. Eine URL in relativer Form würde bspw. wie folgt aussehen:

GET /pub/WWW/TheProject.html HTTP/1.1

Folgender Ablauf gilt bei der Ermittlung einer Ressource:

1. Wird nach einer URL in absoluter Form gefragt, so ist der Host-Name ein Teil der Anfrage-URL. Sollten Host-Header vorhanden sein, werden diese ignoriert.

2. Bei relativer URL-Anfrage wird der Host-Name anhand des Host-Headers bestimmt.

3. Ist der Host-Name kein gültiger Host-Name auf dem Server, antwortet der Server mit einer Fehlernachricht (Bad Request).

Weiterführende Informationen zu Fehlernachrichten sowie Informationen zu HTTP können [The99] entnommen werden.

2.3.2 DNS

Eine Vielzahl von Internet-Anwendungen nutzt den DNS-Dienst zur Namensauflösung — der Zuordnung eines Host-Namens zu einer IP-Adresse und umgekehrt. Ursprünglich, zu Zeiten des ARPANET, erfolgte die Namensauflösung ausschließlich über die lokal verfügbare Datei „Hosts.txt“. Die neue Verwaltung der Namensauflösung - das DNS - löste die Verwaltung der lokalen Host-Dateien ab, weil diese aufgrund des starken Wachstums des globalen Netzes unpraktikabel wurde.

Das heutige DNS ist als eine globale, verteilte, hierarchische Datenbank von Nameservern zu verstehen [Gib07]. Weil keiner der existierenden Nameserver alle DNS-Einträge besitzt, müssen oft mehrere Nameserver kontaktiert werden, um eine Anfrage zu beantworten. Um miteinander kommunizieren zu können, sind sie über Verknüpfungen (Delegationen) verbunden. Jeder Nameserver ist für bestimmte Domains verantwortlich. Zu jeder Domain gehört eine „Zonendatei“, die wiederum Resource Records (RRs) der verwalteten Domains enthält. Für die vorliegende Arbeit ist das A-RR von Bedeutung. In einem A-RR wird einer Domain eine IPv4-Adresse zugeordnet. Darüber hinaus enthält das A-RR einen 32 Bit langen Time to Live (TTL)-Wert. Dieser drückt die Gültigkeitsdauer der Zuordnung Domainname-IP-Adresse in Sekunden aus und kann aufgrund der Länge über 100 Jahre umfassen. Zur Erhöhung der Verfügbarkeit des DNS existieren neben dem „Primary Nameserver“ ein oder mehrere „Secondary Nameserver“. Bei Änderungen der RRs werden diese vom „Primary Nameserver“ an den oder die „Secondary Nameserver“ übertragen.

2.3.2.1 Definition der DNS-Komponenten

Das heutige DNS wurde in [Net87a, Net87b] spezifiziert und lässt sich in drei Hauptkomponenten unterteilen. Diese sind:

- Domain-Namensraum
- Nameserver
- Resolver

Auf sie wird anschließend näher eingegangen.

Domain-Namensraum: In Abbildung 2.14 ist die hierarchische Baumstruktur des DNS-Namensraumes am Beispiel der Domain „www.example.com“ dargestellt.

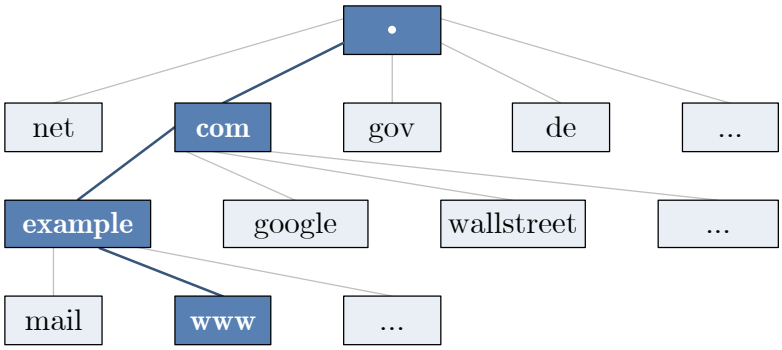


Abbildung 2.14: Schematische Darstellung der Baumstruktur des Domain Name Systems

Die gesamte Domain darf die Länge von 255 Byte nicht überschreiten. Ferner wird sie von rechts nach links interpretiert und in Subdomains unterteilt. Als Separator zwischen den Subdomains wird der „.“ verwendet. Weil ein „.“ an der Wurzel (root) der Baumstruktur keinen Informationsgehalt besitzt, wird dieser in der Regel nicht mitgeschrieben. In Interpretationsrichtung schließt sich die Top-Level-Domain („com“) an. Es folgt die Second-Level-Domain („example“) sowie die Third-Level-Domain („www“).

Nameserver: Es handelt sich dabei um Software- bzw. Hardware-Komponenten, die in der Lage sind, DNS-Anfragen zu beantworten. Nameserver werden in „autorative“ und „nicht-autorative“ Nameserver unterschieden. Während erstere gesicherte Informationen zu bestimmten Zonen besitzen, beziehen nicht-autorative Nameserver ihre Informationen von anderen Nameservern. Nicht-autorative Nameserver können z. B. Anfragen zum zuständigen Nameserver delegieren, oder wenn eine Antwort ausbleibt, einen der Root-Server kontaktieren. Derzeit existieren 253 DNS-Root-Server [roo11]. Sie stellen die höchste Instanz im DNS dar und besitzen Delegierungen zu den zuständigen Nameservern.

Resolver: Ein „Resolver“ ist ein Software-Modul in einem lokalen System. In der Kette zur Namensauflösung befindet er sich zwischen der Anwendung (z. B. Browser) und dem ersten Nameserver. Wird der Anwendung eine Domain oder URL übergeben, reicht die Anwendung diese an den lokalen Resolver weiter. Daraufhin kontaktiert der Resolver den konfigurierten Nameserver. Die Antwort des Nameservers leitet der Resolver wieder an die anfragende Anwendung.

DNS-Protokoll: DNS ist ein Internet-Basisdienst der Anwendungsschicht im ISO/OSI-Referenzmodell. Es kann die Transportschicht-Protokolle UDP und TCP nutzen. Anfragen zur Namensauflösung werden dabei in UDP-Paketen gekapselt. TCP wird genutzt, wenn sich bspw. RRs ändern und der „Primary Nameserver“ die „Secondary Nameserver“ davon unterrichtet. Die Übertragung der Daten erfolgt stets auf Port 53.

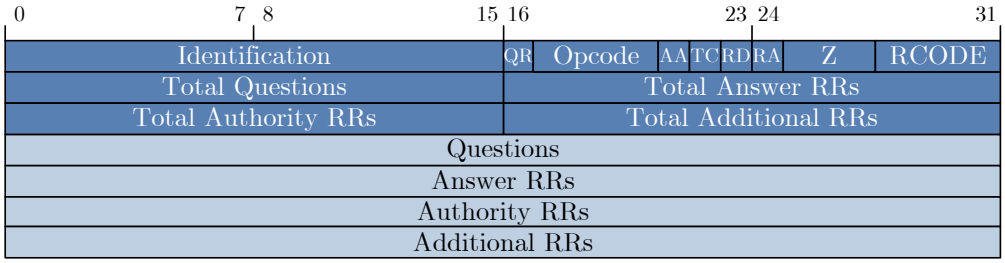


Abbildung 2.15: Aufbau eines DNS-Paketes

Ein DNS Paket ist in die 5 Abschnitte „Header“, „Question“, „Answer“, „Authority“ und „Additional“ unterteilt (vgl. Abbildung 2.15) [Net87b]. Weiterführende Informationen zum DNS sind in [Net87a, Net87b, Gib07] zu finden.

2.4 Netzwerksicherheit

2.4.1 Klassifizierung von Netzwerkbedrohungen

Bei der Nutzung des Internet existieren eine Reihe von Bedrohungen. Diese werden nach den Gesichtspunkten Ursache und Wirkung klassifiziert [Fuh00]. Die auftretenden Ursachen lassen sich wie folgt unterteilen:

- Konzeptionsfehler bei der Planung von Programmen und Protokollen
- Entwicklungsfehler bei der Programmierung von Programmen
- Konfigurationsfehler bei der Erstellung von Software-Parametern und Zugriffsrechten

Die Wirkungen, die auf die aufgeführten Ursachen folgen, lassen sich wie folgt unterteilen [Fuh00]:

- Verlust der Vertraulichkeit

- Verlust der Integrität
- Verlust der Verfügbarkeit (Denial of Service (DoS))

Ein Angreifer verfolgt dabei das Ziel, an sensible Informationen zu gelangen. Dazu fängt er Nachrichten ab, manipuliert sie und täuscht gefälschte Identitäten vor. Das Vortäuschen gefälschter Identitäten wird auch als „Spoofing“ bezeichnet. Es existieren verschiedene Spoofing Techniken, die im Anhang A.4.1 beschrieben werden. Ferner werden im Anhang A.4.2 Schadprogramme („Malware“ [Fuh00]) und Bedrohungsmuster beschrieben.

Weitere Gefahren drohen den Nutzern z. B. durch das Sniffing („Schnüffeln“), also das direkte Mitlesen von Netzwerkverkehrsdaten. Darüber hinaus erfassen Keylogger sämtliche Eingaben direkt von der Tastatur. Dadurch können Passwörter und andere sensible Informationen abgefangen und anschließend durch das Netz an den Angreifer weitergeleitet werden.

Zusätzlich können Protokolleigenschaften und -fehler für Angriffe auf Netzwerke ausgenutzt werden. Als Beispiele seien hier Tiny-Fragment-Attacken [Net95], TCP-Hijacking, TCP-SYN-Flooding oder File Transfer Protocol (FTP)-Bounce-Attacken genannt [Fuh00].

2.4.2 Methoden zur Erkennung von Netzwerkbedrohungen

Das Unternehmen **G Data** bringt halbjährlich einen Report zu den neuesten Bedrohungen im Internet heraus. In ihrem zweiten Report aus dem Jahr 2013 [G D13] befindet sich eine Verteilung der jährlich neu erscheinenden Bedrohungsmuster. Abbildung 2.16 zeigt den Trend zwischen den Jahren 2006 und 2013.

Die signifikante Erhöhung nach dem Jahr 2007 ist aufgrund der zunehmend professionellen Entwicklung hauptsächlich von Trojanern zurückzuführen [Sym08]. Dabei erleichtern Werkzeuge wie das „Zeus-Toolkit“ den Malware-Entwicklern die Arbeit deutlich.

2.4.2.1 Klassifizierung der Mustererkennungsverfahren

Eine Vielzahl von Werkzeugen kann eingesetzt werden, um Netzwerkbedrohungen zu eliminieren. Ihnen allen ist gemein, dass sie die Gefahren zunächst erkennen müssen, bevor sie aktiv etwas unternehmen können. Dabei verwenden sie einen oder mehrere Wege

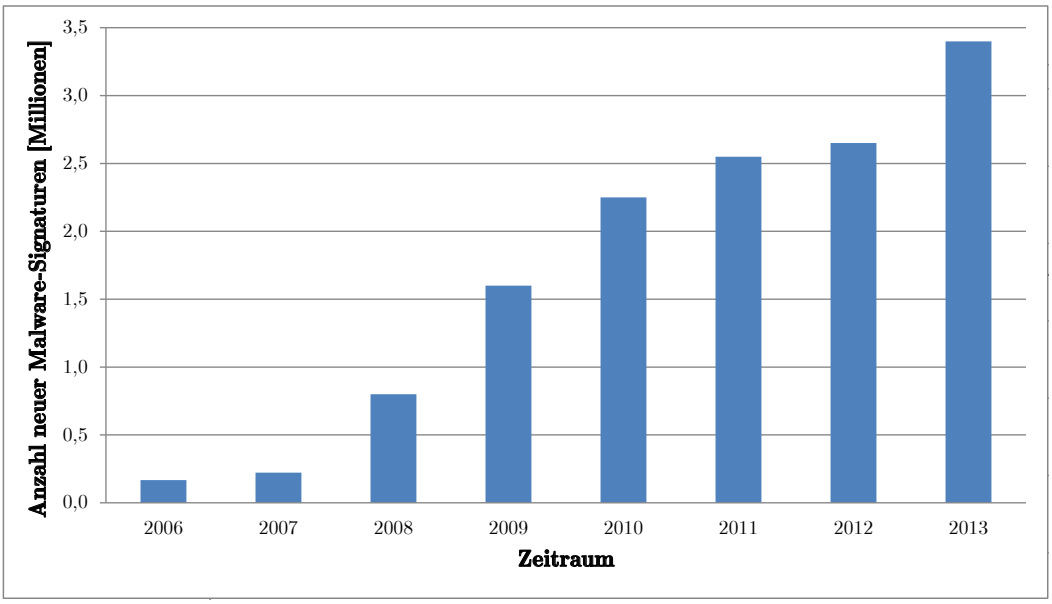


Abbildung 2.16: Anzahl neuer Malware-Signaturen nach Jahren (Quelle: [G D13])

zur Erkennung von Bedrohungsmustern. Abbildung 2.17 zeigt eine Klassifizierung von Muster

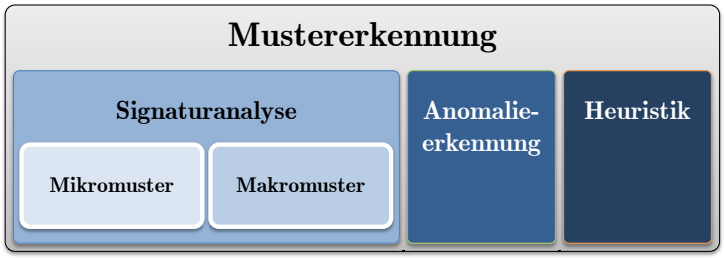


Abbildung 2.17: Klassifizierung von Mustererkennungsverfahren

Prinzipiell kann die Mustererkennung in drei Kategorien unterteilt werden (vgl. Abbildung 2.17). Die am weitesten verbreitete Methode, Netzwerkgefahren zu erkennen, ist die Signaturanalyse. Sie reagiert auf bereits bekannte Bedrohungen und erkennt diese anhand signifikanter Muster wieder. Die bekannten Signaturen lassen sich weiter in statische Muster (Mikromuster) und dynamische Muster (Makromuster) unterteilen. Bei Mikromustern handelt es sich um Protokolldaten wie z. B. IP-Adressen. Ferner werden einfache Such-Strings wie z. B. „root“, aber auch z. B. Web-Domains in diese Klasse eingeordnet. Deutlich komplexer sind Makromuster, bei denen protokollabhängige Abläufe

oder über mehrere Frames fragmentierte Informationen mögliche Bedrohungen enthalten und deshalb untersucht werden müssen.

Neben der Signaturanalyse kommen anomaliebasierte Suchalgorithmen zum Einsatz. Anomaliebasierte Ansätze untersuchen den Datenstrom nach außergewöhnlichem Verhalten. Dabei werden wie bei der Signaturanalyse Vergleiche benutzt. Die größte Herausforderung bei der Anomalieerkennung ist die Erfassung des „Normalzustandes“, um abweichendes Verhalten zu erkennen. Der „Normalzustand“ stellt die Basis für die Vergleiche dar. Überschreitet bspw. das Netzwerkverhalten einen definierten Schwellwert gegenüber dem „Normalzustand“, kann von einer Anomalie ausgegangen werden.

Die dritte Gruppe in der Mustererkennung stellt die heuristische Analyse dar. Sie wird häufig ergänzend eingesetzt, um mit begrenztem Wissen und in kurzer Zeit zutreffende Aussagen über ein System zu tätigen. Bei heuristischen Verfahren existiert kein „Normalzustand“. Weiterhin besitzen sie nicht den Anspruch, die optimale Lösung zu finden. Ihr Vorgehen basiert bspw. auf Annahmen, Erfahrungen und Schätzungen. Sie sind dennoch in der Lage, bisher nicht definierte Muster zu erkennen. Am Beispiel von Antiviren-Programmen durchsuchen sie den Code von Objekten nach Merkmalen und erhöhen einen „Verdächtigkeitszähler“, wenn z. B. verdächtige Befehle erkannt werden. Überschreitet der Zählerwert nach Sichtung des gesamten Codes einen Schwellwert (Erfahrung), so wird das Objekt als verdächtig markiert [Kas09, Kas10].

2.4.2.2 Methoden der Mustererkennung

Abgesehen von der heuristischen Analyse vergleichen Mustererkennungsverfahren Bit-Muster in einem Datenstrom mit bekannten Bedrohungsmustern. Erfolgt die Mustersuche lediglich in den ISO/OSI-Headern bis zur Schicht 4, handelt es sich um eine „flache“ Paketfilterung. Dabei kann die Suche zustandslos (*stateless*) oder zustandsbehaftet (stateful) erfolgen. Die Erweiterung der flachen Paketfilterung ist die „tiefe“ Paketfilterung oder auch die Deep-Packet-Inspection. Im Folgenden werden alle drei Methoden genauer vorgestellt und miteinander verglichen.

Stateless Packet Inspection: Eine *Stateless Packet Inspection* (zustandslose oder auch statische Paketfilterung) analysiert und kontrolliert einzelne Felder der ISO/OSI-Header der Schichten 2 bis 4 ohne Berücksichtigung des Verbindungsstatus und somit ohne

Berücksichtigung früherer Pakete. Die Kontrolle der Netzwerkdaten erfolgt durch sequenzielle Abarbeitung einer Zugriffssteuerungsliste (Access Control List (ACL)). ACLs steuern den Datenverkehr auf einzelnen Netzwerk-Interfaces, überprüfen Quell- und Zieladressen sowie Protokolldaten der ISO/OSI-Schicht 4. Ferner wird der Datentransfer je nach gesetzter Berechtigung in der ACL gewährt oder verweigert.

Stateful Packet Inspection: Eine *Stateful Packet Inspection (SPI)* (zustandsbehaftete oder auch dynamische Paketfilterung) arbeitet wie eine statische Paketfilterung, führt jedoch zusätzlich eine Verbindungsüberprüfung durch, d. h. der Status der Verbindung wird bei der Entscheidung, ob das Paket akzeptiert oder verworfen wird, einbezogen. Jedes Datenpaket kann dabei einer Session zugeordnet werden. Eine zustandsgesteuerte Paketfilterung kann bspw. verwendet werden, um eine FTP-Verbindung zu überwachen. Die besondere Herausforderung beim FTP ist die gleichzeitige Überwachung der Ports 20 (Daten-Port) und 21 (Steuer- bzw. Control-Port) der Transportschicht.

Deep-Packet-Inspection: Die dritte Methode der Mustererkennung ist die tiefe bzw. komplette Paketfilterung (DPI). Sie ist wie folgt definiert:

Jede Art der Datenprüfung, die über die der SPI hinausgeht, kann als „deep“ bezeichnet werden [Bed09]. Deep-Packet-Inspection ist eine erweiterte Methode der Paket Filterung und kann über alle Schichten des OSI-Referenzmodels angewendet werden. Meist wird sie jedoch für Daten der Anwendungsschicht des TCP/IP-Referenzmodells eingesetzt (vgl. Abbildung 2.2). DPI ermöglicht die Identifikation und Klassifizierung sowie das Umleiten und die Blockade von Paketen mit spezifischen Daten oder Code innerhalb der Nutzdaten. Spezifische Daten oder Code entsprechen den sogenannten Signaturen, welche mit genau definierten Mustern (Exact Pattern) bzw. mit regulären Ausdrücken (Regular Expression) abgeglichen werden. Zur Qualitätserhöhung werden in der DPI weiterhin statistische, heuristische und anomaliebasierte Algorithmen eingesetzt [Por05].

Anwendungen der Deep-Packet-Inspection Die DPI ist eine Methode mit äußerst breit gefächerten Anwendungsgebieten. Sie kann z. B. zur Priorisierung des Datenverkehrs verwendet werden. Denkbar ist, dass VoIP-Netzwerkverkehr mit höherer Priorität behandelt werden soll als bspw. Peer-To-Peer (P2P)-Netzwerkverkehr. Neben einer Priorisierung ist sie natürlich auch zur Filterung und damit zur Blockade von Inhalten

einsetzbar. Eine Auswahl weiterer Anwendungsgebiete für den Einsatz einer DPI ist in Tabelle 2.1 dargestellt.

Anwendungsgebiete	Beschreibung und Beispiele
Netzwerksicherheit	Erkennung von Angriffen (z. B. DDoS) auf Nutzer, Netzwerk Services und das globale Netzwerk
Netzwerk Monitoring	Erstellung von Nutzer- und Gruppenprofilen (z. B. für die Anomalieerkennung)
Staatliche Regulierungen	Anordnung einer Datenvorratshaltung oder der Filterung von gesetzeswidrigen Internet-Inhalten
Netzwerk Management	1. Sicherstellung, dass alle Nutzer Zugriff auf angebotene Services haben 2. Klassifizierung des Netzwerkverkehrs

Tabelle 2.1: DPI-Anwendungsszenarien [Wag09]

Grenzen der Deep-Packet-Inspection: DPI-basierte Verfahren sind sehr wirkungsvoll, wenn die zu untersuchenden Daten im Klartext vorliegen. Bei verschlüsselt übertragenen Daten hingegen versagen klassische DPI-Systeme, da die Vergleichsmuster im Klartext vorliegen. Demnach führt eine Verschlüsselung die einfache Mustererkennung als intrinsische Methode durch eine DPI ad absurdum. Obwohl die eigentlichen Nutzdaten verschlüsselt übertragen werden, so ist dennoch eine Analyse im Hinblick auf das Nutzerverhalten möglich. Bspw. können Video- und VoIP-Übertragungen anhand Paket-Header-Kenngrößen identifiziert und charakterisiert werden [Inf09]. Weiterhin ist die Mustererkennung mittels DPI komplexer als bspw. die Paketfilterung, wodurch die vorhandenen Rechenressourcen stärker belastet werden. Selbst auf lokalen Systemen, wo die Mustererkennung mittels DPI als Software-Lösung arbeitet, kann sich die Rechenleistung spürbar reduzieren.

Vergleich vorgestellter Mustererkennungsverfahren: Tabelle 2.2 liefert einen Vergleich der vorgestellten Mustererkennungsverfahren im Hinblick auf deren Anwendungsbereiche, Systemkomplexität, Erweiterbarkeit sowie deren Performance und Umsetzbarkeit in Hardware. Die wesentlichsten Unterschiede zeigen sich bei der Überwachung komplexer Protokollabläufe, der Systemperformance und der Umsetzung dieser Verfahren in

Hardware. Aufgrund der Tatsache, dass jedes Paket zustandslos - ohne Berücksichtigung des Kontextes zu anderen Paketen - betrachtet wird, sind statische Filter für komplexe Protokollabläufe vollends ungeeignet. Auf diese dynamischen Aufgaben sind zustandsbehaftete Filter optimiert. Da DPI-Filter den gesamten Datenframe überwachen, können sie prinzipiell auch zur Überwachung von Protokollabläufen der Transportschicht eingesetzt werden. Diese Art der Überwachung wäre jedoch nur äußerst selektiv und geht an der Idee eines DPI-Filters vorbei, der ja den gesamten Datenframe überwachen soll und daher eine Vielzahl weitere Vergleichsmuster beachten muss.

Weiterhin steht die Komplexität eines Systems dessen Performance gegenüber. Somit versprechen statische Filter eine Kontrolle von Netzwerkdaten mit maximaler Verarbeitungsgeschwindigkeit. Filter mit einem komplexeren Regelwerk lasten prinzipiell die vorhandenen Ressourcen stärker aus. Zusammengehörige Pakete treffen oft mit großen zeitlichen Abständen auf das Filtermodul. Da jedoch Statusinformationen zu jeder Verbindung zwischengespeichert werden müssen, steigt der Ressourcenbedarf des Systems. Je mehr Teilnehmer das Übertragungsmedium nutzen, um so mehr Verbindungen müssen individuell überwacht werden. In der Folge erhöht sich der Suchaufwand.

2.5 Zusammenfassung des Kapitels

Eine Vielzahl von technischen Geräten ist heute in der Lage, über Netzwerke zu kommunizieren. Die etablierten Standardprotokolle sind teilweise sehr alt und weisen zudem Schwächen oder sogar Fehler auf. Angreifer können dies ausnutzen, um Netzwerkendgeräte zu kompromittieren. Gegen Netzwerkbedrohungen auf Protokollebene werden seit Jahren lokale und globale Firewalls eingesetzt. Sie überprüfen den Netzwerkverkehr zustandslos oder zustandsbehaftet. Die zustandslose Paketüberprüfung benötigt aufgrund des weniger umfangreichen Regelwerks geringere Rechenressourcen, bietet jedoch auch minderen Schutz als die zustandsbehaftete Paketprüfung. Allerdings können DoS-Attacken zustandsbehaftete Paketfilter schneller in Überlast versetzen.

Die Einstellung der „policy“ bei Überlast entscheidet, ob Pakete ungeprüft durch den Filter geleitet oder ein Netzsegment netzwerkseitig isoliert wird. Das Vorhandensein einer vollständig richtig konfigurierten Firewall, die eine „flache“ Paketfilterung durchführt, ist jedoch nicht ausreichend, um vor allen Netzwerkbedrohungen zu schützen.

Vielmehr müssen Informationen, die sich tiefer als ein Transportschicht-Header des OSI-

Eigenschaften	Stateless Packet Inspection	Stateful Packet Inspection	Deep Packet Inspection
Eignung für komplexe Protokollabläufe (z. B. FTP und SIP)	- -	++	0
Datenkontrolle oberhalb der Transportschicht	- -	+	++
Fragmentierte Datenkontrolle	- -	+	+
Schutz vor Schadprogrammen, die eine Verbindung zum öffentlichen Netz aufbauen	- -	- -	+
Schutz vor (D)DoS-Attacken	-	0	+
Kontrolle verschlüsselter Verbindungen	-	-	-
Komplexität des Systems	niedrig bis mittel	mittel	mittel bis hoch
Komplexität der Regeln	niedrig bis mittel	mittel bis hoch	mittel bis hoch
Erweiterbar für neue Protokolle und Dienste	++	++	++
Systemperformance und Ressourcenbedarf	++	0	0
Umsetzbarkeit in Hardware	++	+	0

Tabelle 2.2: Leistungsvergleich von Mustererkennungsverfahren

Modells im Datenframe befinden, herangezogen werden. Dafür wurden Verfahren und Lösungen für die Deep-Packet-Inspection entwickelt. Web-Filter nutzen bereits DPI-Verfahren, um definierte Muster, die sich auch hinter dem Transportschicht-Header befinden, aufzuspüren.

In der letzten Kategorie befinden sich DPI-Systeme. Sie sind in der Lage den gesamten Datenstrom zu untersuchen und verwenden, ähnlich wie Web-Filter, Referenzmuster. Damit können sie z. B. Signaturen von Schadprogrammen und das unautorisierte Eindringen in Computersysteme erkennen. Eine Priorisierung des Datenverkehrs ist ebenfalls möglich. Prinzipiell können DPI-Systeme auch zum Detektieren von IP-Adressen oder Transportschicht-Protokollen benutzt werden. Meist ist ihr Regelwerk jedoch umfangreich, wodurch die Vergleiche sehr aufwendig werden. Deshalb ist es besser sie für Mustervergleiche einzusetzen, die andere Netzwerksicherheitskomponenten nicht leisten können.

Zusammenfassend lässt sich feststellen, dass keine der Lösungen für sich allein einen hundertprozentigen Schutz bieten kann, sie jedoch im Verbund einen deutlichen Mehrwert in Fragen der Netzwerksicherheit bieten können, solange es sich nicht um verschlüsselte Daten handelt. Eine Grundvoraussetzung, um Netze optimal zu schützen, ist, dass kombinierte Sicherheitslösungen vorhanden sind. Ferner müssen diese Lösungen vollständig richtig konfiguriert und regelmäßig gewartet werden. Zur Gewährleistung sicherer Systeme ist die Wartung von zunehmender Bedeutung. Die Existenz von Malware-Toolkits wie dem Zeus-Toolkit ist ursächlich für die exponentielle Zunahme neuer Malware-Signaturen (vgl. Abbildung 2.16). Aus der steigende Anzahl neuer Bedrohungsmuster resultiert, dass fortwährend höhere Rechenleistungen aufgewendet werden müssen. Tendenziell muss die Rechenleistung um so höher sein, je tiefer das Erkennungssystem in den Frame schauen muss. Hinzu kommt die parallel mit den Bedrohungsmustern steigende Datenrate der Netze. Unter diesen Bedingungen geraten besonders Software-Lösungen zusehends an ihre Grenzen. Auf Host-Systemen nutzen sie lokale Ressourcen und belasten die Central Processing Unit (CPU) und den Arbeitsspeicher sehr stark. In der Folge wird das gewohnte Arbeitsverhalten negativ beeinflusst. Aufgrund der genannten Tatsachen können Software-Lösungen in größeren Netzwerken nur sehr bedingt eingesetzt werden. Hier muss auf spezialisierte Hardware-Lösungen zurückgegriffen werden, die aufgrund ihres dedizierten Charakters finanziell sehr aufwendig sind.

Kapitel 3

Anforderungsanalyse und Designkonzept

In diesem Kapitel finden die notwendigen Vorüberlegungen statt, um die Netzwerksicherheit für Internet-Teilnehmer zu erhöhen. Die Analyse beginnt mit der Definition der primären und sekundären Ziele für das zu etablierende Sicherheitskonzept. Darauf aufbauend, werden geeignete Lokalitäten für das Sicherheitssystem untersucht. Im Anschluss an die Analysephase entsteht das Design-Konzept. Darin wird zunächst der Stand der Technik für den gewählten Einsatzort näher beschrieben. Danach werden die zu entwickelnden Sicherheitsmerkmale beleuchtet. Bevor eine Zusammenfassung das Kapitel abschließt, wird die Architektur für das Gesamtsystem beschrieben.

3.1 Anforderungsanalyse

3.1.1 Primäre und sekundäre Ziele

Die Anforderungsanalyse untersucht und berücksichtigt notwendige technische Voraussetzungen, Kundenbedürfnisse und vergleicht mit existierenden Lösungen.

3.1.1.1 Sicherheitsanforderungen

Im Hinblick auf Einsatz, Konfiguration und Wartung des Sicherheitskonzeptes wurden die Schutzziele für *informationsverarbeitende und -lagernde Systeme* identifiziert. Diese sind *Vertraulichkeit, Verfügbarkeit und Integrität* und lassen sich unter dem Oberbegriff *Informationssicherheit* zusammenfassen. Die Informationssicherheit bedient sich häufig an der internationalen ISO/International Electrotechnical Commission (IEC) 27.000-Reihe. Innerhalb der 27.000-Reihe werden IT-Sicherheitsstandards für die *Definition der*

Anforderungen (27.001), *Management von Sicherheitssystemen* (27.002), *Implementierungsrichtlinien* (27.003), *Messbarkeit von IT-Sicherheitslösungen* (27.004), *Risikoanalyse* (27.005) etc. definiert. Weiterführende Informationen zur ISO/IEC 27.000-Reihe sind ab [ISO15] zu finden.

Darüber hinaus greift der Autor auf den *Leitfaden Informationssicherheit* vom Bundesamt für Sicherheit in der Informationstechnik (BSI) [BSI15] zurück. Er beschreibt unter anderem die häufigsten Versäumnisse sowie wichtige Sicherheitsmaßnahmen in IT-Sicherheitssystemen und interpretiert dabei die Anforderungen der internationalen ISO/IEC 27.000-Reihe.

3.1.1.2 Latenz und Durchsatz

Neben den Sicherheitsanforderungen existiert das Kundenbedürfnis nach einer hohen QoE. Das bedeutet, dass die Latenz, die das neue Sicherheitssystem verursacht, so gering ausfällt, dass sie für den Kunden nicht spürbar ist. Nach [FIS08] muss das Gesamtsystem eine maximale Verzögerungen von 100 ms unterschreiten.

Aus Performance-Sicht muss das neue Sicherheitssystem in der Lage sein, derzeitigen und zukünftigen Ansprüchen auf Durchsatz gerecht zu werden. Laut [QP15] entspricht das einem Durchsatz von mind. 0,5 Gbit/s. Im TZN kann von mind. 1 Gbit/s ausgegangen werden. Aus diesem Grund muss das neue Sicherheitssystem einen Durchsatz von 1 Gbit/s gewährleisten. Höhere Durchsätze müssen durch Skalierbarkeit des Sicherheitssystems erreicht werden können.

Eine weitere Herausforderung ist die steigende Anzahl von Regel für Sicherheitssysteme. Nach [VV11] müssen aktuelle Systeme mehrere 10.000 Regeln verwalten und anwenden können. Dementsprechend werden diese Anforderungen in die neue Architektur übernommen.

Da das neue Sicherheitssystem flexibel auf Änderungen reagieren soll, darf die Update-Fähigkeit nicht vernachlässigt werden. Die Relevanz dieses Aspektes bestätigt bereits [GM00].

3.1.1.3 National eingesetzte Sicherheitssysteme

Abbildung 4.12 zeigt, dass Filtertechniken auf nationaler Ebene bereits eingesetzt werden. Über diese Systeme ist in der Regel wenig bekannt, da sie politisch motiviert verwen-

det werden. Aufgrund von Claytons Arbeiten sind Details über das britische Cleanfeed [Cla05] und die chinesische Firewall [CMW06] bekannt geworden. Beide Systeme weisen aufgrund ihres strukturellen Aufbaus Schwächen auf, die zu einer schlechten QoE für den Nutzer führen. Die Schwächen werden nachfolgend in Abschnitt 4.6 genauer beschrieben. Aufgrund der Erkenntnisse aus Claytons Arbeiten werden die Ziele für das neue Sicherheitssystem so definiert, dass es die bekannten Schwächen nicht enthält und gleichzeitig eine hohe QoE für den Nutzer entsteht.

3.1.1.4 Ergebnisse der Systemanalyse

Die aufgeführten Anforderungen sind mit dem Team der Nokia Siemens Networks GmbH & Co. KG (heute Adtran GmbH) in Greifswald abgestimmt und führen zu folgenden primären Zielen:

1. Die Robustheit des Sicherheitssystems soll gewährleistet werden. Es muss sich in einer vertrauenswürdigen, kompromittierungsfreien Instanz befinden.
2. Um gegen Angriffe geschützt zu sein, muss ein Konfigurieren aller Komponenten ausschließlich von administrativer Stelle aus erfolgen.
3. Das Erstellen lokaler Sicherheitsrichtlinien durch Netzteilnehmer soll möglich sein. Jedoch sollen Fehlkonfigurationen der Internet-Nutzer eliminiert werden. Dazu muss das geplante Sicherheitssystem übergeordnete Priorität besitzen und mit einem hohen Maß an netzwerktechnischem Wissen installiert, konfiguriert und gewartet werden.
4. Der gesamte Netzwerkverkehr muss durch das Sicherheitssystem geleitet werden. Es darf nicht möglich sein, das System zu unterlaufen.
5. Das Gesamtsystem muss einen garantierten Durchsatz von 1 Gbit/s erreichen.
6. Treten höhere Datenraten als die garantierten Datenraten auf, erfordert das die Skalierbarkeit des Systems.
7. Die maximale Anzahl Netzteilnehmer wird auf 1.000 festgelegt, wobei bis zu 32 parallele OSI-layer-4-Verbindungen je Teilnehmer (32.000 Kommunikationsverbindungen) existieren können. Jeder Verbindung soll ein individueller Satz von Filterregeln zugewiesen werden.
8. Durch die neue Sicherheitsstufe werden Quality of Service und Quality of Experience beeinflusst. Die Beeinflussung soll für den Internet-Teilnehmer nicht negativ

spürbar sein. Der Service (die Sicherheit) muss sich hingegen spürbar verbessern.

9. Das System muss alle Schichten des ISO/OSI-Modells überwachen können, um einen umfassenden Schutz bieten zu können.

Folgende Ziele werden als sekundär eingestuft:

1. Das Sicherheitssystem soll modular aufgebaut werden. Modulare Systeme sind leichter zu warten und zu erweitern. Ferner erhöhen sie die Robustheit des gesamten Systems.
2. Aktualisierungen des Systems sollen partiell und im laufenden Betrieb möglich sein. Auf diese Weise können Reboot-Zeiten des Sicherheitssystems weitestgehend vermieden werden.
3. Ein optionales, eingeschränktes Mitwirken des Internet-Teilnehmers kann helfen, das System sicherer zu gestalten. Dieses darf jedoch nicht gegen bestehende globale Sicherheitsrichtlinien verstoßen.

3.1.2 Untersuchung eines geeigneten Einsatzortes für das neue Sicherheitssystem

Bei der Wahl eines geeigneten Standortes für das einzuführende Sicherheitssystem ist zwingend auf die Vertrauenswürdigkeit und Kompromittierungsfreiheit dieser zu achten. Wie Abbildung 2.3 bereits verdeutlicht, existieren grundsätzlich drei relevante Areale: Equipment des Internet-Teilnehmers, Teilnehmerzugangsnetzwerk und Kernnetz. Als Grundlage für die Wahl eines geeigneten Standortes klassifiziert Tabelle 3.1 diese Areale in Abhängigkeit der vorherrschenden Eigenschaften sowie der unumgänglichen Anforderungen aus Sicht des Internet-Providers. Dabei ist ein „- -“ als ungenügend bzw. negativ und ein „+ +“ als sehr gut geeignet bzw. positiv zu werten.

3.1.2.1 Bereich des Internet-Teilnehmers

Aufgrund der relativ niedrigen Datenraten beim Internet-Teilnehmer können auch hochkomplexe Sicherheitslösungen, die hohe Rechenleistungen erfordern, problemlos eingesetzt werden. Jedoch sind die oft technisch wenig versierten Teilnehmer selten in der Lage, diese fehlerfrei zu konfigurieren, sodass die optimale Filterwirkung nicht erreicht wird. Eine Konfiguration und Wartung des Equipments durch den ISP kann nur sichergestellt werden, wenn alle Internet-Teilnehmer die Hardware des ISP verwenden würden.

Entscheidungs-kriterium	Equipment des Internet-teilnehmers	Teilnehmer-zugangs-netzwerk	Kernnetz
Vertrauenswürdigkeit des Standortes	- -	+ +	+ +
Fachlich kompetente Installation und Wartung	0	+ +	+ +
Sicherstellung der Schutzwirkung	-	+ +	+ +
Verarbeitung auftretender Datenraten	+ +	+	- -
Ausfallsicherheit und Robustheit	0	+	+
Verwendung hochkomplexer Sicherheitsmodule	+ +	0	- -

Tabelle 3.1: Standortklassifizierung für die Einführung einer kompromittierungsfreien Sicherheitslösung aus Internet-Providersicht

Dies kann jedoch nicht gewährleistet werden. In der Schlussfolgerung muss die Vertrauenswürdigkeit dieses Standortes als äußerst niedrig eingestuft werden.

3.1.2.2 Teilnehmerzugangsnetzwerk

Das Zugangsnetzwerk, als Bindeglied zwischen Netzwerk des Internet-Teilnehmers und Kernnetz, ist in jeglicher Hinsicht als geeigneter Standort einzustufen. Weil sich dieser Bereich allein in der Verwaltungshoheit des ISP befindet und somit frei von externen Manipulationen ist, kann er als vertrauenswürdig angesehen werden. Um die im Access-Bereich auftretenden Datenraten in angemessener Zeit untersuchen zu können, müssen Software-Hardware-Co-Designs bzw. reine Hardware-Lösungen entwickelt werden. Qualifizierte ISP-Mitarbeiter übernehmen die Wartung, wodurch die Gefahr von Fehlkonfigurationen auf ein Minimum reduziert wird. Hinzu kommt, dass auftretende Bedrohungen das Zielsystem/-netz nicht mehr erreichen, da sie bereits im Access-Netzwerk eliminiert werden. Last, but not least ist der Angriff auf Sicherheitssysteme im Access-Bereich, mit dem Ziel diese zu überlasten und auszuschalten bzw. sie zu umgehen, nur bedingt möglich. Diese Systeme befinden sich im Datenpfad und können nicht wie Endgeräte und

Services über eine Adresse oder Portnummern erreicht werden. Obwohl sie ihre Aufgaben aktiv ausführen, sind sie für Dritte in Bezug auf ihr Kommunikationsverhalten nur passive Komponenten und somit auf diese Weise nicht zu überlasten. Um ein solches System zu kompromittieren, müsste ein potenzieller Angreifer explizites Wissen über den Aufbau und die Filtermechanismen besitzen.

3.1.2.3 Kernnetz

Durch die kaskadierte Bündelung der Datenleitungen im Zugangsnetz werden im Kernnetz extrem hochbitratige Datenströme weitergeleitet (vgl. Abbildung 2.6). Bereits die Erfassung von Logdaten für spätere statistische Auswertungen stellt eine Herausforderung dar. Die Daten in angemessener Weise zu untersuchen, ohne dass unvertretbar hohe Latenzen entstehen, ist selbst mit dem Einsatz angepasster Hardware derzeit nicht möglich. Obwohl in diesem Areal kompromittierungsfreie Hardware-Lösungen eingesetzt werden könnten, muss dieser Standort wegen der zu hohen Datenraten ebenfalls ausgeschlossen werden.

3.1.2.4 Fazit der Wahl des Einsatzortes

Unter Beachtung der in Tabelle 3.1 aufgeführten Entscheidungskriterien muss festgestellt werden, dass das Teilnehmerzugangsnetzwerk sowohl der sinnvollste als auch der wirkungsvollste Standort für die Etablierung neuer Sicherheitssysteme ist. Auf diese Weise ist es möglich, alle angeschlossenen Internet-Teilnehmer vor Netzwerkbedrohungen zu schützen, ohne dass diese aktiv mitwirken müssen. Lokal vorherrschende Sicherheitslösungen bleiben von einer im Access-Bereich vorgeschalteten Sicherheitsinstanz unberührt. Darüber hinaus lassen sich Fehlkonfigurationen von lokalen Lösungen beheben. Der maximale Mehrwert wird erzielt, wenn das Sicherheitssystem auf einer Linecard in einem DSLAM eingeführt wird (vgl. Abbildung 2.3). DSLAMs bündeln Datenraten im Bereich von 51,84 Mbit/s (OC1) bis 2,48832 Gbit/s (OC48) [PKAC08, LXLS09]. Sie entsprechen somit der Größenordnung der zuvor definierten Vorgaben bezüglich der Datenraten.

3.2 Design-Konzept

Die erste Aggregationsstufe im TZN aus Nutzersicht stellen Ethernet-basierte DSLAMs dar. Diese sind gleichzeitig die am besten geeigneten Plattformen für die Etablierung des neuen Sicherheitssystems, da sie die geforderten Bedingungen aus Abschnitt 3.1.1 vollständig erfüllen. Im Rahmen dieser Arbeit wird somit eine Hardware-basierte Sicherheitsarchitektur für TZN entwickelt. Dazu wird zunächst der Stand der Technik in TZN skizziert. Anschließend werden die zu entwickelnden Sicherheitsfunktionalitäten erörtert.

3.2.1 Stand der Technik in Teilnehmerzugangsnetzwerken

Obwohl das Zugangsnetzwerk für den Datenverkehr vom und zum Nutzer transparent und somit schwer angreifbar ist, existieren bereits heute Sicherheitsmaßnahmen, die Services und Nutzer schützen. In deutschen Teilnehmerzugangsnetzwerken arbeiten die heute existenten Schutzmaßnahmen ausschließlich auf den OSI-Schichten 2 und 3. Ihre Schutzwirkung ist lediglich passiver Natur. Um den Schutz auszuweiten, werden physikalisch vorhandene, nicht manipulierbare Port-Informationen (vgl. [GPP03, Net10b]) einbezogen. Folgende Sicherheitsmerkmale sind bereits realisiert und werden angewendet:

- **Port-Isolation:** Jede Linecard innerhalb eines Access Nodes besitzt mehrere physikalische Ports. An jedem Port ist genau ein ISP-Endkunde angeschlossen. Port-Isolation bedeutet, dass Endkunden nicht direkt kommunizieren können. Wollen zwei ISP-Kunden, deren Ports sich auf derselben Linecard befinden, miteinander kommunizieren, wird ihr Datenverkehr zunächst in Richtung Kernnetz geleitet, bevor der zweite Teilnehmer diesen über seinen Linecard-Port empfängt. Durch Port-Isolation wird vermieden, dass ISP-Endkunden ihren Datenverkehr gegenseitig „sniffen“ können.
- **MAC-Address-Limitation:** An jedem Port einer Linecard darf nur genau eine Source-MAC-Adresse erkannt werden. Diese ist dem ISP bekannt, da ein Kunde mit seiner Source-MAC-Adresse bereits eine IP-Adresse angefragt und erhalten hat. Durch MAC-Address-Limitation wird vermieden, dass ISP-Endkunden Spoofing- und Flooding-Angriffe durchführen können.
- **MAC-Address-Translation (MAT):** Diese Technik wird eingesetzt, um MAC-Adressen in andere MAC-Adressen zu übersetzen. Wird die MAC-Adresse des ISP-

Kunden in eine Provider-MAC-Adresse übersetzt, handelt es sich um eine 1:1 MAT. Demgegenüber steht die n:1 MAT. Hierbei werden mehrere Kunden-MACs in eine ISP-MAC-Adresse übersetzt.

- **VLAN-Tags:** Physische Netzwerke werden durch VLANs in logische Netzwerke separiert. Die Unterteilung erfolgt auf OSI-Ebene 2, indem VLAN-Tags direkt hinter den MAC-Adressen in den Datenstrom eingefügt werden. Aufgrund der Positionierung der Tags sind VLANs robust gegen Angriffe auf OSI-Ebene 2. Zusätzlich werden VLANs zur Priorisierung von Datenströmen (VoIP, P2P) und Internet-Teilnehmern verwendet.

3.2.2 Komposition geeigneter Sicherheitsfunktionalitäten

Heute existiert eine große Vielfalt von Sicherheitspaketen für den durchschnittlichen Internet-Nutzer. Sie sind an die Ressourcen lokaler Systeme angepasst. Je nach Detailstufe der Filterleistung werden die vorhandenen Ressourcen unterschiedlich stark belastet, sodass das gewünschte Arbeitsverhalten möglicherweise sogar negativ beeinflusst wird. Meist weisen diese Pakete umfangreiche Software-Tools auf und umfassen mindestens einen Antiviren- und Firewall-Schutz. Weiterhin können Web-Filter und Filter gegen Malware und Spam im Umfang enthalten sein.

Die zu entwickelnde Sicherheitslösung wird den Namen Secure Access Node tragen, da sie speziell für den Einsatzort im TZN konzipiert wird. Sie wird den Access-Bereich um drei wesentliche Sicherheitsmerkmale erweitern, die in ähnlicher Form auch in handelsüblichen Sicherheitslösungen für Endkunden umgesetzt werden. Jedoch muss sie an die Gegebenheiten im Access-Bereich angepasst werden. Dies betrifft im Wesentlichen die deutlich höheren Datenraten im TZN gegenüber dem lokalen Netzwerk und die Ressourcenlimitierung der notwendigen Hardware (Field Programmable Gate Array (FPGA)). Aufgrund der Voraussetzungen im TZN und im Hinblick auf die Anwenderfreundlichkeit einiger Software-Lösungen werden nicht alle Sicherheitsmerkmale umgesetzt. Beispielsweise betrifft dies den Antiviren-Filter. Diese oft kostenlosen Programme sind einfach zu installieren. Programm-Updates werden ohne Eingreifen des Internet-Teilnehmers durchgeführt. Zudem wäre der Hardware-Aufwand, um alle heute bekannten Viren-Signaturen zu speichern und zu überwachen, extrem hoch.

Das Unternehmen „PANDA“ erfasste bereits im Sommer 2009 täglich 37.000 neue Viren-Signaturen. Seine Datenbank umfasste zum selben Zeitpunkt 30 Mio. Viren-Signaturen

[PAN09]. Bei dieser Flut an Viren-Signaturen rufen selbst lokale Software-Lösungen signifikante Verzögerungen hervor und erfordern hohe Rechenleistungen. Die Problematik verschärft sich im TZN weiter durch die höhere Datendichte je Zeiteinheit. Aufgrund der Abwägung zwischen dem beschriebenen Mehrwert und den entstehenden Kosten sei darauf hingewiesen, dass zu den Sicherheitsmodulen kein Antiviren-Modul zählen wird. Vielmehr wird das modular aufgebaute Sicherheitssystem aus einer Firewall, einem Web-Filter und einem Intrusion-Detection-System bestehen. Da diese Module im Verbund ein sehr mächtiges Mittel gegen Angriffe darstellen, versprechen sie, einen signifikanten Mehrwert zu erzielen.

3.2.2.1 Wahl der Firewall-Strategie

Im Abschnitt 4.5.2 werden unterschiedliche Firewall-Typen vorgestellt. Unter anderem werden die Aufgaben eines „Stateful Paketfilters“ beschrieben. Im Rahmen einer Diplomarbeit aus dem Jahr 2007 [Sie07], welche am Institut für Angewandte Mikroelektronik und Datentechnik der Universität Rostock geschrieben wurde, entstand ein Hardware-basierter zustandsgesteuerter Paketfilter. Alle Verbindungsdaten werden in einer sortierten Liste in einem DDR-Speicher abgelegt. Der größte Aufwand besteht in der Wartung - dem Löschen und Einfügen von Verbindungsdaten - der sortierten Liste. Da eines der primären Ziele des Sicherheitssystems die Überwachung von 32.000 parallelen Verbindungen ist, würde dies im *Worst Case* bedeuten, dass bis zu 31.999 Verbindungsdaten verschoben werden müssten. In der Folge entstehen Wartezeiten von bis zu 3,6 ms, was bei der angestrebten Zielgeschwindigkeit von 1 Gbit/s zu Verwürfen von ca. 297 maximal langen Datenframes je Sekunde führt. Solche hohen Verwurfraten sind jedoch nicht akzeptabel. Hinzu kommt, dass neben den Verschiebeoperationen relativ häufig Wartungszyklen im Speicher durchgeführt werden müssen. Damit wird dieser zustandsgesteuerte Paketfilter extrem anfällig für alle Arten von DoS-Attacken. Um sowohl die Wartung als auch die Suche zu beschleunigen, kann ein Assoziativspeicher (Content-addressable memory (CAM)) eingesetzt werden.

Assoziativspeicher: Assoziativspeicher sind Speicher, die für schnelle Suchen verwendet werden [Bre00]. Sie verhalten sich beim Schreiben wie Speicher mit Direktzugriff (RAM-Speicher). Durch Anlegen einer Adresse werden Inhalte hinzugefügt. Besonders ist jedoch das Lesen in Assoziativspeichern. Bei diesen Speichern werden keine Adressen, sondern Daten (Assoziationen) an den Leseeingang gelegt. Die interne Struktur des

Assoziativspeichers vergleicht das angelegte Datum mit allen Speichereinträgen parallel, wodurch das Lesen stark beschleunigt wird. Bis zu drei Ergebniswerte kann der Assoziativspeicher ausliefern (Match/Mismatch/Multimatch). Ein Ternary Content-addressable memory (TCAM) stellt eine spezielle Erweiterung der Assoziativspeicher dar und erhöht deren Flexibilität. Er besitzt neben den Zuständen 0 und 1 einen „Don't Care"-Zustand für jede Speicherstelle. So werden die als „Don't Care" markierten Bit für den Vergleich ignoriert, wodurch Vergleiche über Ähnlichkeiten zwischen Suchmuster und Speichereintrag realisierbar sind. Das in Abbildung 3.1 dargestellte Schema zeigt sowohl den Aufbau als auch die Suche in einem TCAM.

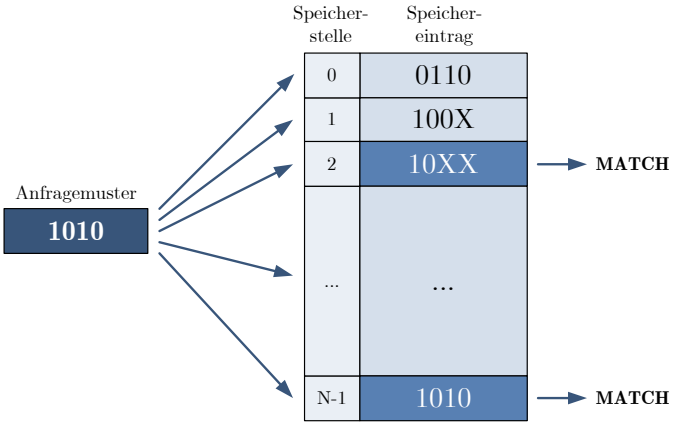


Abbildung 3.1: Schematische Darstellung und Suche in einem TCAM

Der hohen Arbeitsgeschwindigkeit von Assoziativspeicher steht ein hoher Stromverbrauch, geringe Speicherkapazität und eine relativ große Fläche gegenüber [PS06]. Somit sind die nicht für den Einsatz in DSLAMs geeignet.

Infolgedessen wird die Firewall als *stateless* arbeitender Hardware-Paketfilter ausgelegt. Ein *stateless* arbeitender Paketfilter ist in der Lage, die angestrebten Ziele zu gewährleisten. Er kann bspw. Schutz gegen Port-Scans, IP-Bereichsverletzungen und IP-Spoofing bieten. Port-Sperren, z. B. der TCP- und UDP-Ports 337-339 und 447, die das Auslesen von Netzwerkgerätenamen und freigegebenen Dateien verhindern, sind ebenfalls möglich. Allein durch den Einsatz eines zustandslosen Paketfilters im TZN wird bereits bei den beschriebenen Szenarien ein deutlicher Mehrwert entstehen.

3.2.2.2 Web-Filterung — Schutz vor bedenklichen Web-Inhalten

Viren, Malware und SPAM-E-Mails mit Phishing-Inhalten stellen real existierende Gefahren dar. Die daraus resultierenden Folgen erstrecken sich von Belästigungen bis hin zu finanziellen Verlusten. Um dieser Problematik zu begegnen, existiert eine Flut an Software-Lösungen, die anhand auftretender Muster die verschiedenen Bedrohungen erkennen können. Allesamt müssen installiert, konfiguriert und vor allem gewartet werden. Die dabei entstehenden Herausforderungen wurden bereits angesprochen. Darüber hinaus sind im Internet z. B. kinderpornografische, gewaltverherrlichende, verfassungswidrige und weitere bedenkliche Inhalte zu finden. Insbesondere für Kinder und Jugendliche stellt dies eine große Gefahr dar.

Ein Web-Filtermodul kann Abhilfe für die beschriebenen Bedrohungen schaffen. Einschlägige Webseiten, die Viren, Trojaner und Malware verbreiten, sind genauso bekannt wie Webseiten mit Phishing-Inhalten. Gleiches gilt für Webseiten der zweiten beschriebenen Kategorie.

Der Problematik, dass ständig neue Webseiten hinzukommen und andere wiederum entfallen, haben sich Unternehmen im Bereich Netzwerk- und Computer-Sicherheit sowie Bundesbehörden [hS11] bereits gestellt. Somit ist es nicht zwingend notwendig, dass die ISP diese Aufgabe ebenfalls erfüllen müssen. Denkbar ist, dass sie die Vorleistungen in Bezug auf die Erfassung dieser Web-Inhalte über Drittanbieter organisieren. Die Leistung der ISP wird dann ausschließlich in der Erkennung und Unterdrückung dieser Inhalte liegen. Ein Hardware-basiertes Web-Filtermodul im Access-Bereich, welches mit kurzfristigen Updates aktualisiert wird, kann sicherstellen, dass als bedrohlich eingestufte Web-Inhalte blockiert werden. Ob die blockierten Web-Inhalte zu gegebener Zeit auf den betreffenden Servern gelöscht werden, ist für die vorgeschlagene Maßnahme nicht von Bedeutung.

3.2.2.3 Intrusion-Detection-System — Erkennung von Eindringlingen

Die Auswertung der Header-Daten der OSI-Layer 2-4 wird durch das beschriebene Firewall-Modul übernommen. Allein durch den Einsatz einer Firewall werden längst nicht alle bekannten Angriffsmuster erkannt. Wird bspw. bei einem Remote-Angriff ein bestimmter Login-Name wie „root“ oder „admin“ verwendet, haben klassische Firewall-Verfahren keine Erkennungsmöglichkeiten, da diese Informationen tiefer im Datenframe vorkommen. Eine umfassende Untersuchung des gesamten Datenframes ist notwendig, um einen

Eindringling zu erkennen und wirkungsvolle Gegenmaßnahmen einzuleiten. Aus diesem Grund wird ein Hardware-basiertes Deep-Packet-Inspection-System konzipiert, welches in der Lage ist, den gesamten Datenstrom zu untersuchen. Dabei wird nach zuvor gespeicherten Mustern wie z. B. „root“ oder „admin“ innerhalb des Datenstroms gesucht. Wird mindestens ein bekanntes Muster erkannt, löst dieses einen internen Alarm aus, worauf die anzuwendende Regel bestimmt, wie mit dem Frame weiter verfahren wird.

An der Weiterentwicklung der Firewall-Technologien ist bereits ein deutlicher Trend in Richtung DPI sichtbar. Wurden Frames zunächst allein an Adressen und später an Address-Port-Tupel bewertet, so reicht der Blick heutzutage über das Address-Port-Tupel und den Status der Verbindung bis in das letzte Byte eines Frames hinein. Die Herausforderung liegt in der latenzarmen Überwachung von immer mehr Frame-Informationen bei gleichzeitig permanent steigenden Datenraten. Besonders im stark frequentierten Access-Bereich müssen leistungsstarke und Hardware-geeignete Algorithmen eingesetzt werden.

3.2.2.4 Zwischenfazit zu Sicherheitsfunktionalitäten

Das Projekt „Secure Access Node“ (SecAN) wird als Sicherheitsfunktionalitäten eine *stateless* arbeitende Firewall, einen Web-Filter und ein IDS besitzen. Alle Sicherheitsfunktionalitäten werden prototypisch separat entwickelt. Das Ziel wird es jedoch sein, dass jede Sicherheitsfunktionalität strukturell mit den anderen Sicherheitsfunktionalitäten kooperieren bzw. bei ausreichenden Ressourcen in ein FPGA integriert werden kann.

Aufgrund der hohen Datenraten im TZN werden bereits längere Zeit FPGAs im Datenpfad verwendet [DKW+08c, WKD+08, DKW+09]. Software-Lösungen, wie sie in [KL12] vorgestellt wurden, erreichen im *Worst Case* nur einen Durchsatz von unter $\approx 13,3\ Mbit/s$. Deshalb wird die Umsetzung der Filtermodule in FPGAs präferiert. Eine weitere Steigerung der Performance ließe sich mit dedizierten Application-Specific Integrated Circuit (ASIC)-Lösungen erzielen. Wenn auch FPGAs nicht als Optimum bezüglich der erreichbaren Verarbeitungsgeschwindigkeiten angesehen werden können, so bieten sie einen wesentlichen Vorteil, z. B. gegenüber ASIC-Lösungen — sie sind in ihrer Hardware flexibel. Es ist relativ problemlos möglich, ein FPGA Hardware-seitig im Betrieb zu aktualisieren, was bei einer dedizierten ASIC-Lösung nicht möglich ist. Aus den vorgestellten Gründen werden die zu entwickelnden Sicherheitsmodule unter

Verwendung von Hardware-geeigneten Algorithmen ihre Aufgabe im Wesentlichen durch FPGA-Hardware erfüllen. Insbesondere die Zwischenspeicherung von Informationen wird soweit möglich, in kostengünstigere externe flüchtige Speicher ausgelagert.

Abgesehen von den Hardware-technischen Voraussetzungen wird jedem Nutzer eine Art „Mitsprache“ bei der Filterung seiner persönlichen Daten eingeräumt werden. Sowohl ISP- als auch nutzerseitige Filterregeln werden richtungsspezifisch agieren. Gemeinsam stellen sie einen Pool von Filterregeln dar, den die benannten Filtermodule verarbeiten werden. Die Verarbeitung beginnt mit einer richtungsabhängigen Klassifizierung des Datenstroms. Anschließend erfolgt eine Bewertung mit richtungsspezifischen Firewall-Regeln, gegebenenfalls wird eine Filterung von unerwünschten Web-Inhalten sowie eine tiefe Paketfilterung zur Erkennung von Eindringlingen durchgeführt. Da das System mit einem flexiblen Satz von Filterregeln arbeiten soll, sind (Re-)Konfigurationen unvermeidlich. Zu diesem Zweck werden Software-Komponenten entwickelt. Beginnend mit der Systemarchitektur des SecAN-Prototypen werden im Folgenden die Entwicklungstools, die Testumgebungen sowie die Zielplattform vorgestellt.

3.2.3 Systemarchitektur

3.2.3.1 Software-Defined Networking

Eine zusätzliche Sicherheitsstufe, die im TZN den bidirektionalen Netzwerkverkehr von Internet-Teilnehmern überwacht, ist ein hochkomplexes System, das sorgfältig konzipiert werden muss. Um die auftretende Komplexität zu bewältigen, lehnt sich der Autor der vorliegenden Forschungsarbeit an den Ansatz des Software-Defined Networking (SDN) an.

Martin Casado und Nick McKeown von der Stanford Universität entwickelten bereits 2005 erste Konzepte zum Thema SDN [CM05]. In ihrer Veröffentlichung "The Virtual Network System"[CM05] entwickelten sie ein Lehr-Tool, um Studenten praktische Erfahrungen beim Entwerfen, Umsetzen und Debuggen von Netzwerkgeräten, wie Routern und Switches, zu vermitteln. Mit Hilfe des Tools sollte jeder Student beliebige paketverarbeitende Netzwerkgeräte entwerfen und sie in verschiedenen Topologien nutzen können. Mittels SDN ist es leicht möglich, Datenpfade und Netzwerkressourcen neu zu konfigurieren, wobei standardisierte Programmierschnittstellen genutzt werden [Kir13].

Bei SecAN-Gesamtsystem handelt es sich um eine generische modulare Sicherheitsar-

chitektur. Diese wird als zentrale (perspektivisch dezentrale) Kontrollinstanz eingesetzt, um maximale Netzwerksicherheit zu gewährleisten. In Annäherung zum SDN, bei dem Controller den Datenfluss administrieren und Koppelelemente Routing-Entscheidungen ausführen, lassen sich folgende Analogien zum SecAN-Gesamtsystem aufstellen:

1. Der Controller im SDN entspricht beim SecAN-Gesamtsystem den drei Software-Ebenen (vgl. Anhang B).
2. Als Koppelelemente werden im SecAN-Gesamtsystem die Filterkerne der Packet Processing Engine (PPE) — Firewall, Web-Filter und IDS — innerhalb der Data Plane umgesetzt.
3. Statt einer Routing-Entscheidung wird im SecAN-Gesamtsystem eine Security-Entscheidung getroffen.
4. Im SDN wird *OpenFlow* als Kommunikationsmodel zwischen Controller und Koppelelement eingesetzt. Dabei entspricht der *OpenFlow Header* der Flow ID im SecAN-Gesamtsystem.

Beim SecAN-Gesamtsystem liegen sehr ähnliche Strukturen wie beim SDN vor. Im Gegensatz zum SDN werden Netzwerkentscheidungen ausschließlich auf Security-Ebene getroffen. Aus diesem Grund überführt der Autor den Begriff der *Software-Defined Networking* in die für das SecAN-Gesamtsystem treffendere Bezeichnung *Software-Defined Security (SDS)*.

3.2.3.2 Architektur des SecAN-Gesamtsystems

Grundsätzlich lässt sich das SecAN-Gesamtsystem in zeitkritische und zeitunkritische Bestandteile gliedern. Dabei werden zeitunkritische Aufgaben von Software-Blöcken und zeitkritische Aufgaben von angepasster, monolithischer Hardware übernommen. Zu den nicht zeitkritischen Bestandteilen zählen Preprozesse, wie das Erstellen der Konfigurationsdaten für die Hardware sowie die Konfiguration dieser. Als zeitkritisch ist alles das einzustufen, was mit der latenzarmen Überwachung des Netzwerkverkehrs in Zusammenhang gebracht werden kann. In diesem Bereich werden zu entwickelnde Hardware-Komponenten sicherstellen, dass Datenstaus und darauffolgende Frame-Verwürfe auf ein Minimum reduziert werden.

Im Einklang mit den Systemanforderungen aus Abschnitt 3.1.1 lässt sich so ein sinnvoller Funktionssplit aus vier Abstraktionsebenen erstellen. Diese sind in Abbildung 3.2

dargestellt.

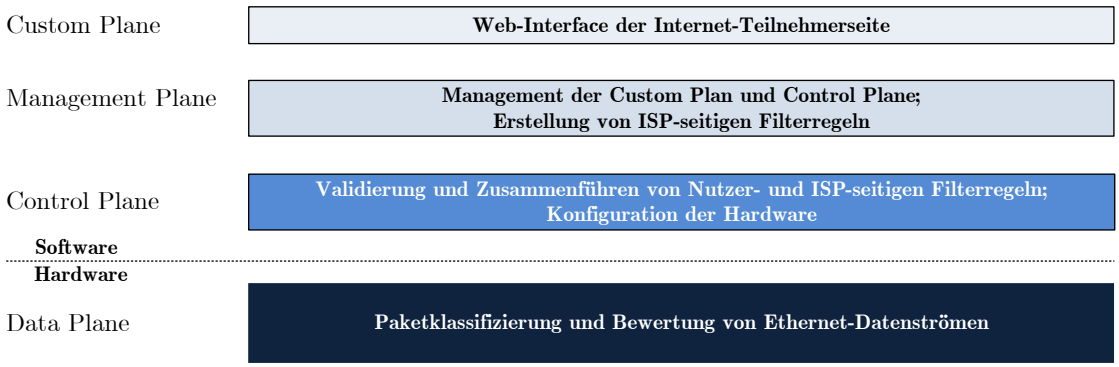


Abbildung 3.2: Struktureller Aufbau der Secure Access Node Architektur

Bei den oberen drei Schichten handelt es sich um Software-Schichten, die von der Custom Plane über die Management Plane bis zur Control Plane vom Abstraktionsniveau abnehmen. Die Komplexität wird in den oberen beiden Schichten, die für das Erfassen und Verwalten von Konfigurationsdaten verantwortlich sind, hinter Graphical User Interfaces (GUIs) verborgen. Ohne grafische Oberfläche bietet die Control Plane die Möglichkeit der kompromittierungsfreien, korrekten Erstellung von Hardware-geeigneten Regeln und letztendlich die Konfiguration der Hardware selbst. Im Detail werden Aufbau, Funktionalität und Umsetzung im Kapitel B beschrieben.

Das Hauptaugenmerk der vorliegenden Arbeit liegt auf der Data Plane. Es handelt sich um einen hochkomplexen Hardware-basierten Verarbeitungspfad für Ethernet-Daten. Dieser ist in der Lage, richtungsabhängig empfangene Ethernet-Daten zwischenzuspeichern und an das Klassifizierungsmodul (Packet Classification Engine (PCE)) (vgl. Abschnitt 4.3) weiterzuleiten. Die Basis für eine richtungsspezifische Klassifizierung stellen die sogenannten *Flow ID Masks* dar. Sie legen fest, welche Parameter in den Ethernet-Frames zur Klassifizierung herangezogen werden. An die Klassifizierung anschließend wird ein Satz von individuellen Filterregeln (Ruleset) durch die Rule Search Engine identifiziert und ausgeliefert (vgl. Abschnitt 4.4). Letztendlich wird der Eingangsdatenstrom mit Hilfe der Filterregeln durch die Filtermodule bewertet und gegebenenfalls verworfen.

Durch den vollständig modularen Aufbau der Hardware ist diese in Subsysteme zerlegbar und kann so leicht gewartet und erweitert werden. Die Aufgaben, Funktionalitäten, Umsetzungen und Ergebnisse werden im Kapitel 4 detailliert vorgestellt.

3.3 Zusammenfassung

Um einen umfassenden Schutz gegen Bedrohungen aus dem Internet zu gewährleisten, sind mehrere Schritte notwendig. Zunächst einmal muss ausreichendes Wissen über bestehende Gefahren vorhanden sein. Weiterhin müssen effektive Gegen- bzw. Präventivmaßnahmen bekannt sein, welche häufig kostspielig sind und zudem umfangreiches Wissen bezüglich deren Installation, Konfiguration und Wartung verlangen. In den meisten Fällen ist der durchschnittliche Internet-Nutzer nicht in der Lage, die finanziellen Mittel aufzuwenden. Zudem besitzt er nur selten das notwendige technische Know-how, um sein Netzwerk ausreichend vor Bedrohungen aus dem Internet zu schützen.

Um diesen Internet-Nutzern dennoch umfassenden Schutz vor Internet-Bedrohungen zu offerieren, wurden zu Beginn des Kapitels die primären und sekundären Ziele definiert (vgl. Abschnitt 3.1.1). Es wurden Rahmenbedingungen festgelegt, welche die Robustheit, Sicherheit und Effizienz des zu entwickelnden Sicherheitssystems gewährleisten. Ferner soll das System skalierbar in Bezug auf die Anzahl der Teilnehmer und steigende Datenraten sein, die lokalen Sicherheitsrichtlinien von Internet-Nutzern unterstützen und nutzerseitige Fehlkonfigurationen eliminieren. Gleichzeitig sollen sich Filterstufen nicht negativ auf das Internet-Verhalten auswirken und dennoch einen umfassenden Schutz bieten.

Drei unterschiedliche Netzwerkbereiche wurden näher miteinander verglichen, da sie potentielle Relevanz in Bezug auf den Einsatzort des Sicherheitssystems aufweisen. Die detaillierte Klassifizierung befindet sich in Tabelle 3.1. Dabei stellte sich heraus, dass der größte Mehrwert bei der Einführung von Sicherheitsstufen im TZN erzielt wird. Da keine Sicherheitslösung für sich allein einen umfassenden Schutz bieten kann, soll im Zugangsnetzwerk ein modulares, mehrstufiges, Hardware-basiertes Sicherheitssystem errichtet werden, welches allein durch die ISP-Administratoren gewartet werden kann.

Aufgrund der hochbitratigen Datenraten im TZN werden die Sicherheitsfunktionalitäten weitestgehend in frei konfigurierbarer FPGA-Hardware entstehen. Damit wird sichergestellt, dass der Datenstrom mit minimaler Verzögerung überwacht werden kann. Software-Komponenten werden die FPGA-Hardware bei zeitunkritschen Aufgaben unterstützen. Das einzuführende Sicherheitssystem wird über einen Basisschutz hinausreichen und neben einer Firewall und einem Web-Filter ein System zur Erkennung von Eindringlingen umfassen. Ein Ziel ist es, bis zu 32.000 Verbindungen gleichzeitig zu überwachen. Die angestrebte FPGA-Lösung kann jedoch die Menge an Verbindungsda-

ten nur mit einer hohen Anzahl an Paketverwürfen verwalten. Aus diesem Grund wird das Firewall-Modul *stateless* arbeiten. Ein Web-Filter-Modul wird Internet-Nutzern vor als bedrohlich eingestuften Web-Inhalten schützen. Da Web-Inhalte in der Regel über mehrere Instanzen angefragt werden, ist darauf zu achten, dass eine Erkennungsmethode gewählt wird, die nicht umgangen werden kann. Gelingt dies, so muss die zu wählende Filterstrategie in der Lage sein, die als bedrohlich erkannten Web-Inhalte unmittelbar zu blockieren, damit alle angeschlossenen Teilnehmer unverzüglich geschützt werden. Die dritte einzuführende Stufe stellt ein System zur Erkennung von Eindringlingen dar. Mit Hilfe leistungsstarker Algorithmen muss der gesamte Datenstrom nach zuvor bekannten Bedrohungsmustern untersucht werden. Im Falle einer positiven Erkennung wird dann entschieden, wie mit dem Ethernet-Frame verfahren werden soll.

Durch die Überwachung und Filterung des Datenstromes in Downstream-Richtung ist es möglich, lokale Misskonfigurationen zu eliminieren. Die gleichzeitige Überwachung und Filterung des Upstreams stellt sicher, dass die angestrebten Ziele, das Netzwerk des Internet-Teilnehmers, sowohl das Zugangs- als auch das Kernnetz zu schützen, erreicht wird.

Kapitel 4

Hardware-Konzept und Realisierung

In diesem Kapitel wird die Hardware des SecAN-Gesamtsystems - die *Data Plane* - beschrieben. Neben ihr existieren die Software-Module der *Custom Plane*, *Management Plane* und *Control Plane* (vgl. Abschnitt B). Alle Module der *Data Plane* werden als Hardware-Module entwickelt. Die Hardware unterteilt sich, wie in Abbildung 4.1 dargestellt, in die beiden Gruppen *SecAN-Framework-Module* (hellblau) und *SecAN-Core-Module* (dunkelblau).

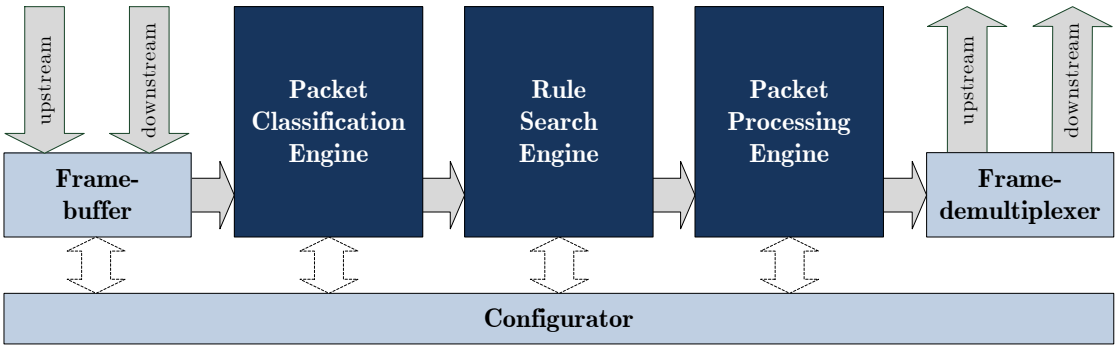


Abbildung 4.1: Blockschaltbild des SecAN-Gesamtsystems

Die in Abbildung 4.1 dargestellten *SecAN-Framework-Module* erheben nicht den Anspruch, einen wissenschaftlichen Mehrwert zu liefern. Vielmehr dienen sie der Steuerung der Konfigurationsdaten und koordinieren die Datenströme aus Up- und Downstream. Sie bestehen aus den Hardware-Modulen *Configurator*, *Framebuffer (FB)* und *Framedemultiplexer (FDM)*.

Neben redundanten Aufgaben (s. Anhang C.1.3) übernehmen die *SecAN-Core-Module*

die Klassifizierung und die sicherheitsrelevante Überwachung des Datenstromes. Ihre Aufgaben lassen sich wie folgt beschreiben:

- Richtungsabhängige Klassifizierung des Datenverkehrs (*Packet Classification Engine (PCE)*)
- Abbildung und Suche der klassifizierten Frame-Daten auf ein flow-spezifisches Regelwerk (*Rule Search Engine (RSE)*)
- Untersuchung, Bewertung und Löschung des Datenstroms bei Auffälligkeiten durch die drei Filterkerne - Firewall, Web-Filter, Intrusion-Detection-System - (*Packet Processing Engine (PPE)*)

Letztendlich sind in Abbildung 4.1 der Datenstrom (grau eingefärbte Pfeile) sowie die Konfigurationswege (weiß eingefärbte Pfeile) veranschaulicht.

In den Abschnitten 4.2 bis 4.7 werden alle Komponenten zunächst konzipiert, anschließend realisiert und letztendlich getestet. Durch das Zusammenspiel beider Modulgruppen entstehen drei voll funktionsfähige Prototypen für den industriellen Einsatz im Bereich Netzwerksicherheit. Jeder Prototyp ist in der Lage, eine definierte Sicherheitsaufgabe zu verrichten. Aufgrund der modularen Struktur des SecAN-Gesamtsystems sind sie weiterhin in der Lage, im Verbund zu arbeiten, wodurch der größte Sicherheitsmehrwert für die Endkunden von ISPs, das Netz des ISP sowie das Kernnetz entsteht.

Alle Filtermodule der PPE weisen identische Ein- und Ausgangsvektoren auf. Diese uniformen Modulstrukturen ermöglichen eine flexible Reorganisation der Filtermodule sowie eine leichte Erweiterung des SecAN-Gesamtsystems um weitere Filterfunktionalitäten. Eine durchdachte Datenführung und -verarbeitung innerhalb des Systems verhindert Deadlocks und ungewollten Datenverwurf (packet drop) (s. Anhang C.1) und sorgt auf diese Weise für eine hohe QoE beim Nutzer.

4.1 Hardware-Spezifikation

Das ML507-Evaluation Board[1] des Unternehmens XILINX[2] dient als Zielplattform für das SecAN-Gesamtsystem. Die Hardware der Zielplattform ist an den heutigen Anforderungen von Ethernet-basierten Zugangsnetzwerken ausgerichtet und so dimensioniert,

[1] Xilinx Development Board ML507 - http://www.xilinx.com/products/devkits/HW-V5-ML507-UNI-G.htm

[2] Xilinx - http://www.xilinx.com

dass jeweils ein Prototyp voll funktionsfähig entwickelt werden kann. Durch das Zusammenführen mehrerer ML507-Evaluation Boards [Xil09] lässt sich die gesamte Funktionalität des SecAN-Gesamtsystems abbilden.

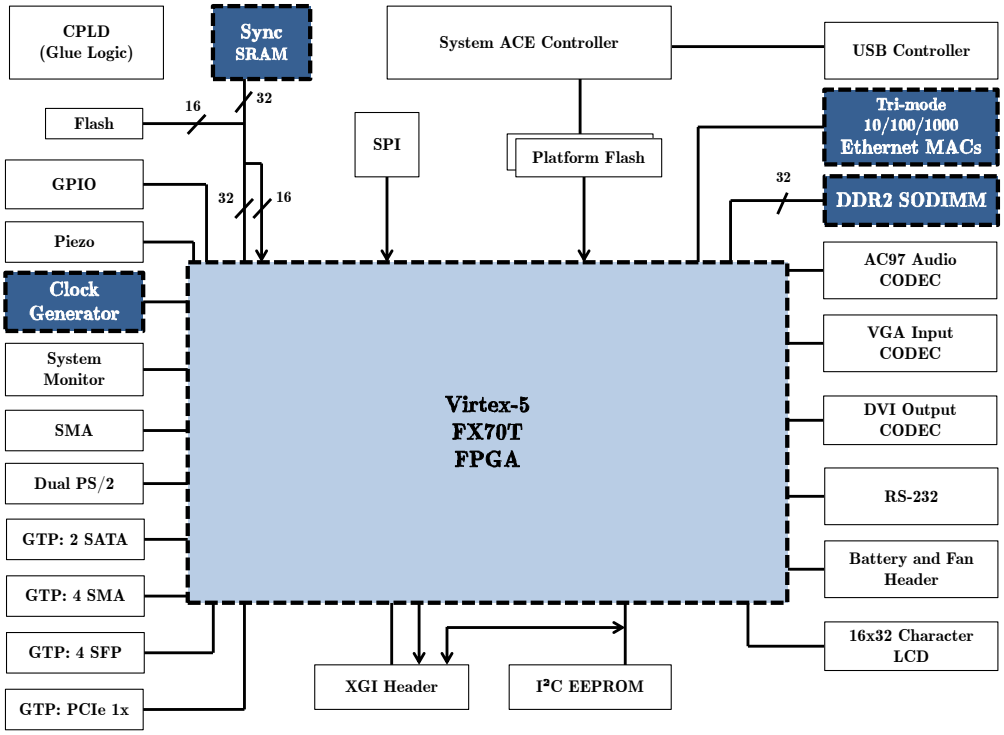


Abbildung 4.2: Schematische Darstellung des ML507-Entwicklungsboard nach Xilinx [Xil09]

Im Access-Bereich werden DSLAMs wie der SURPASS hiX 5622 [Net10b] eingesetzt. Sie sind auf die individuellen Bedürfnisse von ISP angepasst und enthalten dieselben Hardware-Komponenten wie die vom Autor gewählte Zielplattform. Abbildung 4.2 zeigt die verfügbaren Komponenten auf der Zielplattform und hebt die für das Projekt maßgeblichen Komponenten farblich hervor. Als zentrale Komponente sticht der FX70T-FPGA hervor. Um die gewünschten SecAN-Funktionalitäten zu erreichen, werden weiterhin zwei Ethernet-MACs, der synchrone SRAM, der DDR2-Speicher sowie der Clock-Generator verwendet.

Weiterführende Informationen zur Familie der verwendeten Zielplattform sind im Anhang D.1 zu finden. Anschließend werden die Entwicklungstools und die Testumgebungen

der SecAN-Hardware im Anhang D.4 vorgestellt. Letztendlich wird in Anhang C.1 die „Datenverarbeitung innerhalb der Data Plane“ und die „Bearbeitungsstrategie redundanter Aufgaben“ präsentiert.

4.2 Framework-Module für das SecAN-Gesamtsystem

In diesem Abschnitt wird das Framework der SecAN-Hardware konzipiert, entwickelt und getestet. Es handelt sich dabei um sekundäre Hardware-Module, die für das SecAN-Core-System Konfigurations- und Steuerungsaufgaben übernehmen. Dieses Framework besteht aus den Modulen *Configurator*, *Framebuffer* und *Framedemultiplexer*.

4.2.1 Konzipierung und Realisierung des Configurator-Moduls

Wie in Abschnitt B.3 beschrieben, übernimmt die *Control Plane* den Versand der Software-seitig erstellten Konfigurationsparameter. Alle Konfigurationsdaten werden durch die *Control Plane* in Ethernet-Frames mit definiertem Aufbau gekapselt (vgl. Abbildung B.11) und an die Hardware versandt.

Das Hardware-Modul *Configurator* übernimmt die Konfigurationsdaten und verteilt sie an die adressierten Hardware-Komponenten. Dabei kann eine Konfiguration sowohl offline (vor dem ersten Datenverkehr) als auch online (Unterbrechung des Datenverkehrs) erfolgen. Sowohl Online- als auch Offline-Konfiguration folgen dem Schema, das in Abbildung 4.3 dargestellt ist.

Ein einleitendes und abschließendes Handshake mit dem *FB-Modul* stellt sicher, dass die zu konfigurierende Hardware von dem anstehenden Konfigurationsprozess informiert wird. Dabei wird die Konfiguration durch eine Anfrage (*Start of Configuration (req)*) von der Software eingeleitet. Diese Anfrage wird vom *Configurator* an das *FB-Modul* weitergeleitet. Erhält das *Configurator*-Modul eine Bestätigung (*Start of Configuration (ack)*) vom *FB-Modul*, leitet es diese an die Software weiter, woraufhin die Konfigurationsdaten für die Hardware-Module von der Software versandt werden. Da jedes Konfigurationsdatum von der Hardware bestätigt werden muss, wird sichergestellt, dass kein Konfigurationsmuster verloren geht.

Der *Configurator* erkennt anhand eines Identifiers (vgl. Abbildung B.11) das zu konfigurierende Hardware-Modul und leitet die Daten entsprechend weiter. Als Modul-Identifier

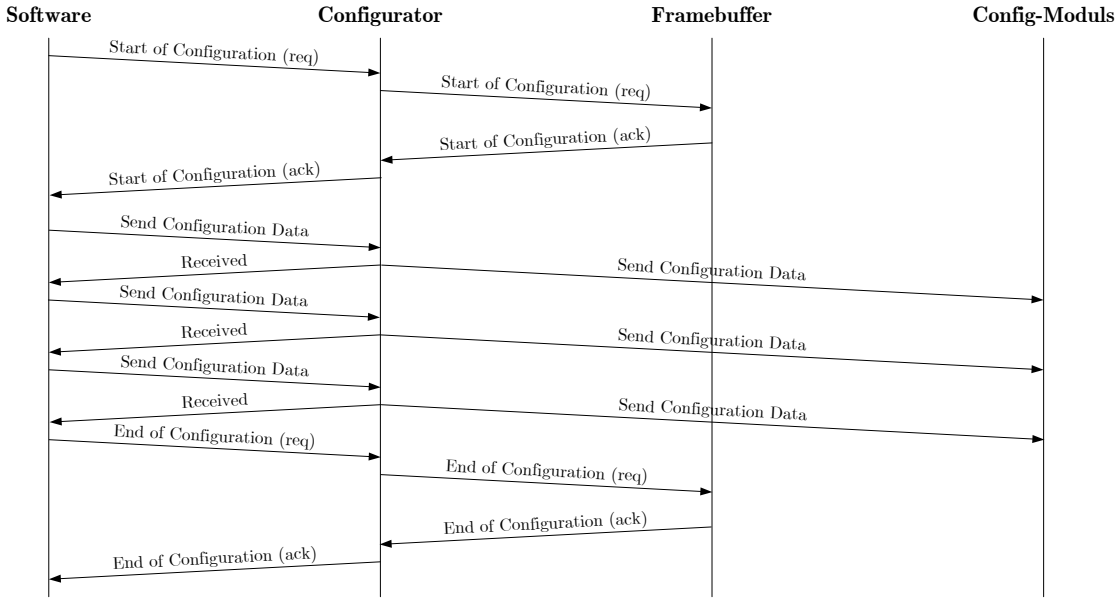


Abbildung 4.3: Konfiguration der SecAN-Hardware

wurden die in Tabelle 4.1 dargestellten Kombinationen verwendet. Die gewählten Abstände zwischen den Modul-Identifiern sorgen für eine ressourcenschonende Implementierung der Hardware und garantieren gleichzeitig die Erweiterbarkeit des Systems durch zusätzliche Filtermodule.

Neben dem Modul-Identifier zählt die Längenangabe (Config-Length (vgl. Abbildung B.11)) zum Header der Konfigurationsdaten. Die Längenangabe ist insbesondere für den verwendeten DDR2-Speicher wichtig, da dieser „burstartig“ beschrieben und gelesen wird und die Längeninformation zu Beginn einer Aktion bekannt sein muss. Der binäre Wert der Längenangabe gibt die Anzahl der Byte an.

Auf die Längeninformation folgt die Payload. Sie umfasst immer ein ganzzahliges Vielfaches von 4 Byte, sodass durch den data_path_type (dpt) (vgl. Anhang C.1) eine Hardware-schonende Implementierung gewährleistet wird. Zusammenfassend stellt das Blockschaltbild des *Configurator*-Moduls die Ein- und Ausgangsparameter in Abbildung 4.4 dar.

Modul-Identifier [hexadezimal]	Bedeutung
0000	Start der Konfiguration
0001	SRAM-Daten schreiben
0002	SRAM-Daten lesen
0003	DDR-Daten schreiben
0004	DDR-Daten lesen
0005	PCE-Daten schreiben
0006	PCE-Daten lesen
0032	Web-Filter-Daten schreiben
0033	Web-Filter-Daten lesen
0064	IDS-Daten schreiben
0065	IDS-Daten lesen
FFFF	Ende der Konfiguration

Tabelle 4.1: Identifier zur Konfiguration der SecAN-Hardware-Module

Ressourcenbedarf	Genutzt	Verfügbar	Auslastung
Anzahl Slice Register	835	44.800	1,9 %
Anzahl Slice LUTs	1292	44.800	2,9 %
BlockRam/FIFO	2	296	0,7 %

Tabelle 4.2: Ressourcenbedarf des Configurator-Moduls

4.2.2 Ressourcenbedarf des Configurator-Moduls

Nachdem alle Testmuster erfolgreich verarbeitet werden konnten, weist das auf Geschwindigkeit optimierte *Configurator*-Modul den in Tabelle 4.2 dargestellten Ressourcenbedarf auf.

Mit einer Zielfrequenz von $246,914$ $Megahertz$(MHz) erzielt das *Configurator*-Modul einen backannotierten Durchsatz von $\approx 7,901$ $Gbit/s$. Somit wird der geforderte minimale Durchsatz von 1 Gbit/s eingehalten. Die Latenz des *Configurators* beträgt 3 Systemtakte.

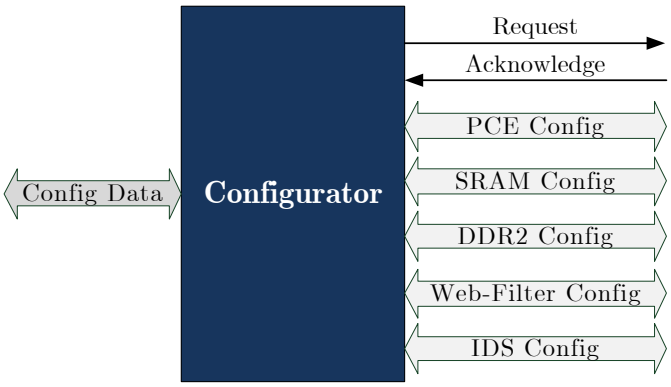


Abbildung 4.4: Blockschaltbild des Configurator-Moduls in der SecAN-Hardware

4.2.3 Zusammenfassung des Configurator-Moduls

Für die Hardware notwendige Konfigurationsinformationen werden von der Software über den *Configurator* an die jeweiligen Hardware-Komponenten verteilt. Ein Handshake zwischen Soft- und Hardware stellt sicher, dass keine Konfigurationsinformationen verloren gehen. Die Ressourcenauslastung des *Configurators* auf dem Zielsystem ist minimal (vgl. Tabelle 4.2). Gleichzeitig ist die Frequenz des Moduls nahezu 8-fach höher als gefordert. Diese Implementierung ist besonders Hardware-schonend und begünstigt einen hohen Durchsatz. Jedoch hat diese Implementierung den Nachteil, dass beim Hinzufügen weiterer Hardware-Module, die ebenfalls konfiguriert werden müssen, zusätzlich das *Configurator*-Modul angepasst werden muss. Da sich nur wenige Komponenten im System befinden, würde ein arbitriertes Bussystem deutlich mehr Ressourcen benötigen. Der zeitliche Aufwand der Anpassungen am *Configurator* kann dagegen als minimal eingestuft werden. Aus diesem Grund ist diese Implementierung optimal für den Anwendungsfall im SecAN.

4.2.4 Konzipierung und Realisierung des Framebuffer- und Framedemultiplexer-Moduls

Abbildung 4.5 zeigt das *Framebuffer*- und das *Framedemultiplexer-Modul* sowie deren Zusammenwirken innerhalb des SecAN-Gesamtsystems. Durch die Funktionalität beider Module wird es möglich, den Datenverkehr effizient durch die *Core*-Bereiche der SecAN-Hardware zu leiten.

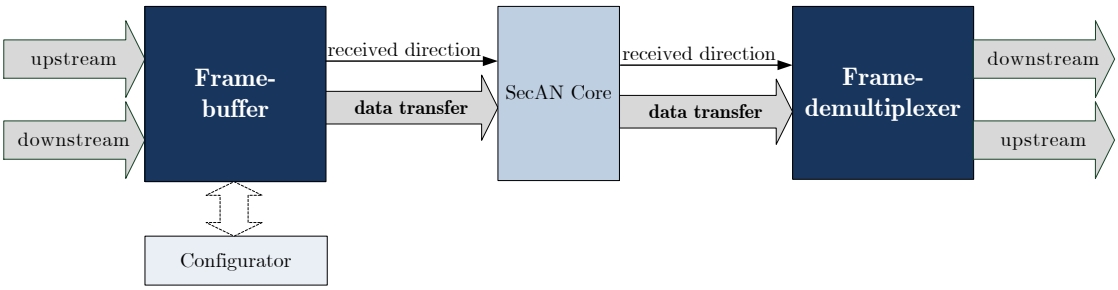


Abbildung 4.5: Blockschaltbild des Framebuffers und Framedemultiplexers in der SecAN-Hardware

Framebuffer-Modul:
Das *FB*-Modul besitzt zwei grundsätzliche Aufgaben. Zum einen ist es für die Datenunterbrechung während des Konfigurationsprozesses verantwortlich. Zum anderen koordiniert es den Empfang und die Zwischenspeicherung von Ethernet-Frames aus dem Up- und Downstream sowie deren sequenzielle Weiterleitung an die *PCE* (vgl. Abbildung 4.1).

Die Datenunterbrechung während des Konfigurationsprozesses wurde bereits in Abschnitt 4.2.1 angesprochen. Über Steuerinformationen schaltet das *FB-Modul* zwischen Datenverarbeitungs- bzw. Konfigurationsmodus um.

Ethernet-Frames werden in Abhängigkeit ihrer Empfangsrichtung in das *FB-Modul* geschrieben. Dabei hängt das Schreiben vom Füllstand der jeweiligen Puffer-*First In - First Out (FIFO)* ab. Ein Ethernet-Frame wird nur aufgenommen, wenn von der Puffer-FIFO signalisiert wird, dass sie einen maximal langen Ethernet-Frame aufnehmen kann.

Das Lesen aus den Zwischenspeichern beginnt, wenn sie *nicht leer* sind und der Backpressure-Mechanismus (vgl. Anhang C.1.1) der *PCE* dies zulässt. Kann die *PCE* einen Frame verarbeiten, liest das *FB*-Modul einen kompletten Frame aus einem der Zwischenspeicher aus. Befinden sich Daten in beiden Puffer-FIFOs, wird die Downstream-FIFO priorisiert, da die meisten Endkunden über einen asymetrischen Internet-Anschluss verfügen, wobei deren Downstream um mehrere Größenordnungen höher als ihr Upstream ist. Gleichzeitig mit dem ersten Datenwort des Frames signalisiert das *FB*-Modul der *PCE*, ob der Frame aus Up- oder Downstream-Richtung empfangen wurde (*receive direction*-Information). Diese Information ist für die Datenverarbeitung bis hin zum richtungsabhängigen Versand der Ethernet-Frames durch das *FDM*-Modul von Relevanz.

Framedemultiplexer-Modul:
Ethernet-Frames und Zusatzinformationen (z. B. Frame-Parameter und Regelsatz (vgl. Abschnitte 3.2.3 und 4.3.3)), die durch die Filterstufen nicht verworfen wurden, erreichen das *FDM*-Modul. Aufgabe dieses Moduls ist es, die eingehenden Daten auf den richtigen Systemausgang weiterzuleiten sowie die spezifischen Zusatzinformationen zu verwerfen. Für die Wahl des richtigen Systemausgangs wertet das *FDM*-Modul die *receive direction*-Information vom *FB*-Modul (vgl. Abbildung 4.5) aus.

4.2.5 Ressourcenbedarf des Framebuffer- und des Framedemultiplexer-Moduls

Nachdem alle Testmuster erfolgreich verarbeitet werden konnten, weisen die auf Geschwindigkeit optimierten Module *Framebuffer* und *Framedemultiplexer* den in den Tabellen 4.3 und 4.4 dargestellten Ressourcenbedarf auf:

Ressourcenbedarf	Genutzt	Verfügbar	Auslastung
Anzahl Slice Register	235	44.800	0,5 %
Anzahl Slice LUTs	250	44.800	0,6 %
BlockRam/FIFO	4	296	1,4 %

Tabelle 4.3: Ressourcenbedarf des Framebuffer-Moduls

Ressourcenbedarf	Genutzt	Verfügbar	Auslastung
Anzahl Slice Register	75	44.800	0,2 %
Anzahl Slice LUTs	117	44.800	0,3 %
BlockRam/FIFO	0	296	0,0 %

Tabelle 4.4: Ressourcenbedarf des Framedemultiplexer-Moduls

Mit einer Zielfrequenz von 268,294 MHz erzielt das Teilsystem, bestehend aus *Framebuffer*- und *Frame-Demultiplexer*-Modul, einen backannotierten Durchsatz von $\approx 8,585\ Gbit/s$. Somit wird der geforderte minimale Durchsatz von 1 Gbit/s eingehalten.

Die Latenzen des beschriebenen Systems teilen sich wie folgt auf:

- 379 Takte am Eingang des *Framebuffers*, wenn gerade begonnen wurde, einen maximal langen Frame an den SecAN-Core auszuliefern
- 2 Takte Verarbeitungszeit im *Framebuffer*
- 2 Takte Verarbeitungszeit im *Framedemultiplexer*

Somit entsteht eine maximale Verzögerung von 383 Takten im System. Im *Best Case* entstehen 4 Takte Verzögerung. Diese Werte werden anschließend benötigt, um mit Hilfe der Zielfrequenz die Latenz des Gesamtsystems zu bestimmen.

4.2.6 Zusammenfassung des Framebuffer- und Framedemultiplexer-Moduls

Die Module *Framebuffer* und *Framedemultiplexer* steuern den eingehenden und ausgehenden Datenverkehr der SecAN-Core-Module. Dabei bezieht sich die Steuerung sowohl auf den Up- als auch auf den Downstream. Ferner unterbrechen sie den Datenverkehr, wenn die Hardware in den Konfigurationsmodus versetzt wird.

Aufgrund einer effizienten Implementierung benötigen beide Module minimale Ressourcen (vgl. Tabellen 4.3 und 4.4). Gleichzeitig erzielen sie einen Durchsatz von mehr als 8 $Gbit/s$. Damit übererfüllen sie die gesteckten Ziele bezüglich Durchsatz. Durch den hohen Durchsatz wird das Verhalten der Core-Module nicht negativ beeinflusst.

4.3 Paketklassifizierung I: Packet Classification Engine

4.3.1 Motivation

Die Paketklassifizierung ist integraler Bestandteil paketverarbeitender Systeme. Als Paketklassifizierungsproblem wird die Aufgabe verstanden, eine Identifikation des Datenstromes durchzuführen und Regeln zu finden, die auf den Datenstrom anzuwenden sind. Zur Identifikation wird ein eindeutiger Schlüssel aus dem Datenstrom gebildet. Mit Hilfe des Schlüssels wird anschließend die Regel/der Regelsatz gesucht.

Damit nicht jede Filterstufe das Paketklassifizierungsproblem selbstständig lösen muss, wurde in Abschnitt C.1.3 hergeleitet, dass es günstiger ist, diese Aufgabe zentral zu lösen. Dafür werden für die beiden Teilaufgaben des Paketklassifizierungsproblems zwei kooperierende Module - die *Packet Classification Engine* und die *Rule Search Engine*

- entwickelt. Dabei übernimmt die *PCE* das Separieren und Vorbereiten von Header-Informationen für eine optimal schnelle Suche von Regelsätzen in der *RSE*. Weiterhin profitieren die nachfolgenden Filterstufen in der *Packet Processing Engine* von zusätzlich erfassten Metadaten des aktuellen Datenstromes. Da das Hauptaugenmerk der vorliegenden Forschungsarbeit auf den Filtermodulen der *Packet Processing Engine* liegt, wird eine für das SecAN-System angepasste optimale Lösung auf Basis des Standes der Technik erarbeitet.

4.3.2 Stand der Technik

Sowohl die Schlüsselbildung als auch das Auffinden des Regelsatzes sind Teilaufgaben des Paketklassifizierungsproblems. Sie bauen direkt aufeinander auf und werden typischerweise zusammen behandelt. Daher besitzen die Ausarbeitungen dieses Abschnittes Gültigkeit für die Unterkapitel 4.3 und 4.4. Die Aufteilung in die beiden Unterkapitel ist der Übersichtlichkeit sowie den besonderen Anforderungen an das SecAN-Gesamtsystem geschuldet. Nachfolgend werden verschiedene Ansätze zur Paketklassifizierung vorgestellt.

Grundsätzlich können eine oder mehrere Header-Informationen zur Klassifizierung von Datenströmen herangezogen werden. Vor wenigen Jahren wurden Tupel aus bis zu 5 Header-Werten gebildet (vgl. [Tay05, JRV08, JP09a, DBB09]). Durch die Umstellung auf SDN-Architekturen werden heute bis zu 15 Header-Felder erfasst (vgl. [QP15]). Aufgrund der Ähnlichkeit zum SDN werden beim SecAN 10 Flow-Parameter (vgl. Tabelle B.1) und als Meta-Daten ein *HTTP-Get*-Flag sowie die *Startposition der OSI-Layer 4 Payload* aus dem Datenstrom extrahiert.

Dabei stellen die extrahierten Flow-Parameter die Basis für die Zuordnung eines Regelsatzes dar. Der Regelsatz wird anschließend von der Firewall-Filterstufe zur Bewertung benötigt. Weiterhin unterstützen die Frame-Parameter und die Meta-Daten alle Filterstufen der PPE bei der Bewertung des Datenstromes und sorgen so für einen optimalen Durchsatz.

4.3.2.1 Paketklassifizierungstechniken

Die meisten Paketklassifizierungsalgorithmen können in zwei Hauptkategorien unterteilt werden: *Entscheidungsbaum-basierte Algorithmen* und *zerlegungsbasierte Algorithmen*

Entscheidungsbaum-basierte Algorithmen Grundlage dieser Algorithmen ist eine geometrische Betrachtung des Paketklassifizierungsproblems. Hier stehen die anzuwendenden Regeln im Vordergrund. Sie beschreiben jeweils einen multidimensionalen Raum. Die genaue Dimension des Raumes hängt von der Anzahl der Header-Felder, die zu der Regel führen, ab. Zwei Header-Felder bilden bspw. einen rechteckigen Raum, drei Header-Felder einen Quader u. s. w.

Werden für mehrere Regeln dieselben Header-Felder benötigt, so lässt sich dieser Zusammenhang in einer Baustruktur abbilden. Dabei bilden gemeinsam genutzte Header-Felder einen *Unterraum* im selben Zweig des Baumes. Der Entscheidungsbaum ist um so komplexer, je mehr Regeln gebildet werden und je mehr Header-Felder involviert sind. Im *Worst Case* beträgt der Speicherbedarf für derartige Baumstrukturen $O(N^D)$, wobei N die Anzahl der Regeln und D die Anzahl der Header-Felder darstellen. Beispiele für Entscheidungsbäume in der Paketklassifizierung sind in [PLW+03] und [SBVW03] zu finden.

Zerlegungsbasierte Algorithmen Bei diesen Algorithmen werden Teilergebnisse aus individuellen Header-Feldern gebildet, die für die Klassifizierung herangezogen werden. Anschließend wird aus allen Teilergebnissen ein finales Ergebnis gebildet.

Als Beispiel sei hier der *BV-Algorithmus* [JP09b] erwähnt. Nach jeder Lookup-Phase wird ein N-Bit-breiter Vektor zurückgeliefert, wobei jedes Bit innerhalb des Vektors einer Regel entspricht.

Fazit zu Paketklassifizierungstechniken Der *Entscheidungsbaum basierte Algorithmus* kann besonders gut für Präfix Machting verwendet werden. Durch die Verwendung weniger Bit je Header-Feld (*Präfix Matching*) lässt sich der Speicherbedarf reduzieren. Da beim SecAN *Perfect Matching* anvisiert wird, ist dieser Algorithmus nicht optimal.

Zerlegungsbasierte Algorithmen verwenden vollständige individuelle Header-Felder. Sie sind daher besser für das SecAN-System geeignet, da eine Überprüfung auf ein *Perfect Match* möglich ist. Allerdings ist die Anzahl an Speicheranfragen von der Anzahl der zu prüfenden Header-Felder abhängig.

Grundsätzlich benötigen beide Paketklassifizierungstechniken eine Vielzahl von initialen Speicheranfragen. Kann der Speicher nicht aus FPGA-Ressourcen gebildet werden, muss auf einen externen Speicher zurückgegriffen werden. Je nach Wahl des externen Speichers

entstehen unterschiedliche Latenzen. Bei der Verwendung externer Speicher muss darauf geachtet werden, die Anzahl der initialen Lesezugriffe zu minimieren.

Da beim SecAN bis zu 32 individuelle Regelsätze je Internet-Nutzer, bestehend aus unterschiedlich vielen Regeln, verwendet werden sollen, muss von der Verwendung externer Speicher ausgegangen werden. Daher sind beide Paketklassifizierungstechniken nicht optimal, können jedoch in Teilen nachgenutzt werden.

4.3.2.2 Hardware-geeignete Ansätze

Sehr schnelle Suchmethoden verwenden TCAMs [STT03, QP15]. Äußerst positiv ist deren zeitliche Komplexität von $O(1)$. Jedoch weisen alle Arten von CAMs hohe Anschaffungskosten, einen hohen Energiebedarf und geringe Speicherkapazitäten auf. Diese Eigenschaften führen dazu, dass sie für die Paketklassifizierung im SecAN-Projekt als ungeeignet eingestuft werden müssen. Weiterführende Informationen zu CAMs wurden in Abschnitt 3.2.2.1 beschrieben.

Einen sehr interessanten zweistufigen Ansatz stellen Gupta et. al. in [GLM98] vor. Bei dem Ansatz wird bspw. eine IP-Adresse in zwei Segmente zerlegt: 24 Bit und 8 Bit. Jedes Segment benötigt einen eigenen Speicher. Alle Regeln bis 24 Bit sind direkt in einem Speicher mit 2^{24} Adressen abgelegt. Für den Fall, dass weitere Regeln existieren, ist an der Speicherstelle ein Index für einen zweiten Speicher hinterlegt. Dieser Index wird mit den verbleibenden 8 Bit verkettet und beschreibt so die Speicheradresse für die höher priore Regel. Aufgrund der Priorisierung der Regeln, deren Aufteilung auf mehrere Speicher und aufgrund der Möglichkeit einer mehrstufigen Suche ist dieser Ansatz für den SecAN sehr interessant. Jedoch ist aus Sicht der verfügbaren Hardware der Speicherbedarf äußerst ungünstig. Ausgehend von 2 Byte pro Speichereintrag in den 2^{24}-Adressen-fassenden Speicher, werden 32 MByte benötigt. Diese Speichergrößen lassen sich derzeit nur mit Dynamic Random Access Memorys (DRAMs) kostengünstig abbilden. Für eine schnelle Suche sind SRAMs besser geeignet.

Ausgehend von dem zweistufigen Ansatz ist eine Verwendung des SRAMs und des DDR2-Speichers denkbar. Während der SRAM eine schnelle Suche ermöglicht, wenn die Position des Speichereintrages nicht genau bekannt ist, bietet der DDR2-Speicher ausreichend viel Kapazität, um die individuellen Regelsätze vorzuhalten.

Im SRAM können die eingangs erwähnten 10 Flow-Parameter (126 Bit), gefolgt von einer exakten Speicheradresse im DDR2-Speicher, hinterlegt werden. Ist die Speicheradresse

im SRAM bekannt, wird die Suchzeit, um den individuellen Regelsatz zu lesen, minimal sein. Dementsprechend muss es das Ziel sein, die Position des Speichereintrages im SRAM möglichst genau zu bestimmen. Hash-Verfahren bieten eine gute Möglichkeit, die 126-Bit-breiten Flow-Parameter zu einer Leseadresse für den SRAM-Speicher zu verkürzen.

4.3.2.3 Hashing

Dieser Abschnitt stellt Hashing-Methoden vor, die für die vorliegende Arbeit von spezieller Bedeutung sind. Verschiedene Methoden werden vom Web-Filter sowie vom *IDS* nachgenutzt.

Der Begriff „Hashing“ bedeutet „zerhacken“ oder auch „zerlegen“ und wird häufig verwendet, um ungeordnete Mengen zu verwalten. Das Besondere beim Hashing ist, dass große Mengen an Eingangswerten (z. B. Ethernet-Header) auf eine kleine Menge (Hash-Werte) abgebildet werden. Diese Hash-Werte (oder auch nur Fragmente davon) lassen sich bspw. als Speicheradressen verwenden. Weiterhin weisen Hash-Verfahren ein konstantes Zeitverhalten und lineare Speicherkomplexität auf. Somit tragen Hash-Verfahren dazu bei, Suchzeiten in Speichern deutlich zu verkürzen. Zum Hashen wird benötigt:

1. Eine *Hash-Tafel* $T[0, m-1]$ mit m Speicherplätzen
2. Eine Abbildungsfunktion $h : U \rightarrow [0, m-1]$

Die Abbildungsfunktion h ermöglicht die Abbildung des Universums U auf die Hash-Tafel T [Blu04]. An einem Beispiel sollen die Grundlagen des Hashings erläutert werden.

S ist eine Teilmenge des Universums U ($S \subseteq U$) und besteht aus den Elementen x ($x \in S$). Ziel ist es nun, $x \in S$ in $T[h(x)]$ zu speichern. Als Hash-Tafel T wird ein Speicher mit m Speicherplätzen angenommen.

Existieren keine zwei Elemente $x, y \in S, x \neq y$, die durch die Abbildungsfunktion h dieselbe Speicheradresse m (Hash-Wert) erzeugen, ist von einer *perfekten* Hash-Funktion auszugehen. Anderenfalls treten Kollisionen auf (vgl. Formel 4.1). Bei einer äußerst schlechten Hash-Funktion würden bspw. alle Elemente auf dieselbe Speicheradresse abgebildet. Demnach gilt: Hash-Funktionen sind qualitativ besser, je weniger Kollisionen sie verursachen.

$$h(x) = h(y) : x \neq y \tag{4.1}$$

Am beschriebenen Beispiel bedeutet Kollision, dass auf der Speicherstelle bereits ein Datum (Element) gespeichert wurde und die Speicherstelle somit nicht mehr zur Verfü-

gung steht. Da für zufällige Daten, wie sie bei paketverarbeitenden Systemen auftreten, keine perfekten Hash-Funktionen existieren, müssen Kollisionen mit geeigneten Strategien aufgelöst werden. Sowohl die Qualität von Hash-Funktionen als auch verschiedene Hash-Verfahren werden im Anhang A.5 vorgestellt.

Alle im Anhang A.5 vorgestellten Hashing-Methoden sind geeignet, um in Hardware realisiert zu werden, bieten jedoch unterschiedlich gute Ergebnisse. Am ressourcenschonendsten kann die XOR-Methode umgesetzt werden. Bessere Hash-Ergebnisse lassen sich mit den Hash-Funktionen der Klasse H_3 erzielen. Allerdings steigt der Ressourcenbedarf bei diesem Hashing-Verfahren. Zusätzlich hängt die Qualität von den Werten in der Bit-Matrix ab (vgl. Abbildung A.4). Hier sollten echte Zufallszahlen gewählt werden. Die CRC-Methode bietet sehr gute Hash-Eigenschaften. Werden jedoch eine Vielzahl unabhängiger CRC-Hash-Funktionen mit annähernd gleichen Eigenschaften wie bspw. einer Bit-Breite von m-Bit im Hash-Vektor benötigt, scheitert dieses Verfahren, da nur wenige als geeignet eingestufte Generatorpolynome existieren [KC04]. Hinzu kommt, dass von der IEEE lediglich 1 Polynom für den Ethernet-Bereich verabschiedet wurde, welches den hohen Qualitätsanforderungen entspricht [Ass02].

Fazit: Aufgrund der Erkenntnisse diesen Abschnittes wird eine Zweiteilung der Paketklassifizierung umgesetzt. Das wesentliche Zusammenwirken der PCE und RSE ist in Abbildung 4.6 dargestellt.

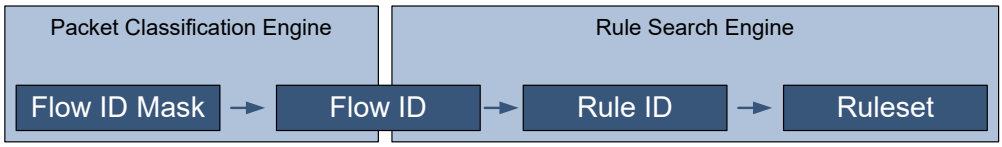


Abbildung 4.6: Funktionale Umsetzung der Packet Classification Engine

Auf Basis der *Flow ID Mask* wird die PCE die *Flow ID* erstellen. Der Hash-Wert der *Flow ID* dient als initiale Speicheradresse im SRAM. Die RSE vergleicht, ob an der Speicherstelle des Hash-Wertes die *Flow ID* zu finden ist. Wird die *Flow ID* gefunden, liest die RSE die Speicheradresse im DDR2-Speicher sowie die Länge des Regelsatzes. Speicheradresse und Länge des Regelsatzes werden als *Rule ID* bezeichnet. Anschließend fordert die RSE das *Ruleset* im DDR2-Speicher an und koordiniert dessen Auslieferung an die PCE. Letztendlich sendet die PCE den Daten-Frame, das Frame-Parameterset und den Regelsatz an die erste Filterstufe der PPE.

4.3.3 Konzipierung der Packet Classification Engine

Als erstes Core-Modul im SecAN-Gesamtsystem muss die *Packet Classification Engine* eine Vielzahl von Aufgaben erfüllen:

1. Es muss in der Lage sein, sowohl online als auch offline konfiguriert zu werden. Konfiguriert werden die beiden *Flow ID Masks* (vgl. Abschnitt 3.2.3.2).
2. Weiterhin soll die *PCE* aus dem aktuellen Daten-Frame ein Frame-Parameterset zusammenstellen. Das Frame-Parameterset wird neben den Parametern, die für die Frame-Klassifizierung notwendig sind, zusätzliche Metadaten enthalten. Diese erhöhen den Durchsatz im Web-Filter und in der IDS-Filterstufe (vgl. Abschnitte 4.5 bis 4.7).
3. Die dritte Aufgabe besteht im Zusammenfügen der *Flow ID* auf Basis der entsprechenden *Flow ID Mask*.
4. Letztendlich muss aus der *Flow ID* ein Hash-Wert gebildet werden. Dieser dient als initiale Speicheradresse für einen individuellen Regelsatz des Firewall-Systems.

4.3.3.1 Konfigurationsphase

Dieser Prozess wurde bereits in Abschnitt 4.2.1 vorgestellt.

4.3.3.2 Bildung des Frame-Parametersets

Erreicht ein neuer Datenstrom die *PCE*, beginnt diese mit der Zwischenspeicherung des Frames. Gleichzeitig werden für die Filterstufen signifikante Parameter im Datenstrom gesucht, extrahiert und zu einem Set – dem Frame-Parameterset – zusammengefasst. Neben den in Tabelle B.1 dargestellten Frame-Parametern werden die Informationen **HTTP-Get-Flag** und **Startposition der OSI-Layer 4 Payload** in das Frame-Parameterset integriert. Sie stellen wichtige Informationen für den Web-Filter und das IDS dar. Dieses Parameterset wird vollständig mit dem ersten Datum des Frames an die Filterstufen übertragen. Die Filterstufen „erfahren“ somit frühestmöglich, ob für sie relevante Parameter vorhanden sind und welchen Wert sie besitzen. Somit wird der Datenverarbeitungsprozess innerhalb der Filterstufen maximal beschleunigt.

4.3.3.3 Bildung der Flow ID

Für das Firewall-System muss ein individueller Regelsatz geladen werden. Abbildung 4.6 zeigt das dafür notwendige Vorgehen.

Die *Flow ID* wird auf Basis der *Flow ID Masks* gebildet, welche während der Konfigurationsphase für den Up- und den Downstream gespeichert wurden. In den *Flow ID Masks* wurden die Frame-Parameter festgelegt, die in den Klassifizierungs-Flow eingehen (vgl. Abschnitt B.2.2.1).

Mit dem Eintreffen des ersten Datums in der PCE wird untersucht, welche der beiden *Flow ID Masks* für den aktuellen Datenstrom von Relevanz ist. Die Relevanz der *Flow ID Mask* hängt von der Empfangsrichtung des Frames ab (Up- oder Downstream). Diese Information wird vom *FB*-Modul an die *PCE* geliefert.

Die entsprechenden Frame-Parameter werden aus dem Datenstrom extrahiert und zu einem Vektor mit einer Länge von bis zu 126 Bit (vgl. Abbildung C.5) zusammengesetzt. Sie bilden die *Flow ID*. Zusätzlich wird der Hash-Wert der *Flow ID* ermittelt. Er dient als SRAM-Speicheradresse innerhalb der *Rule Search Engine*. Als geeignetes Hash-Verfahren wurde die CRC16-Methode ermittelt (vgl. Abschnitt C.2).

Konnten nicht alle Frame-Parameter für die *Flow ID* aus dem Datenstrom extrahiert werden, wird dies vermerkt, da dann sowohl *Flow ID* als auch deren Hash-Wert ungültig sind. *Flow ID*, ihr Hash-Wert sowie die Information über ihre Gültigkeit werden an die *RSE* übergeben. Im Falle von ungültigen Informationen ergreift die *RSE* geeignete Maßnahmen. In jedem Fall wird von der *RSE* ein Regelsatz gefunden und an die *PCE* zurückgeliefert.

4.3.3.4 Datenversand an die erste Filterstufe

Nachdem das Frame-Parameterset komplettiert und ein Ruleset geladen wurde, werden beide gemeinsam mit dem zwischengespeicherten Daten-Frame an die erste Filterstufe der *PPE* geleitet. Signalisiert das System die Bereitschaft, die Daten zu übernehmen (vgl. Abschnitt C.1.1), kehrt die *PCE* in einen Zustand zurück, in dem sie neue Ethernet-Daten empfangen und verarbeiten kann.

Weiterführende Details zur Realisierung der *PCE* sind dem Abschnitt C.2 zu entnehmen.

4.3.4 Ressourcenbedarf

Die vorgestellte *Packet Classification Engine* wurde umfangreichen Tests unterzogen. Nachdem alle Testmuster erfolgreich durch die *PCE* verarbeitet werden konnten, wies das auf Geschwindigkeit optimierte Teilsystem den in Tabelle 4.5 dargestellten Ressourcenbedarf auf.

Ressourcenbedarf	Genutzt	Verfügbar	Auslastung
Anzahl Slice Register	802	44.800	1,8 %
Anzahl Slice LUTs	5382	44.800	12,0 %
BlockRam/FIFO	10	296	3,4 %

Tabelle 4.5: Ressourcenbedarf der Packet Classification Engine

Mit einer Zielfrequenz von $216,887$ MHz erzielt die *PCE* einen backannotierten Durchsatz von $\approx 6,612$ Gbit/s. Die Verzögerung des beschriebenen Systems beläuft sich auf maximal 22 Takte. Im *Best Case* entstehen 10 Takte Verzögerung. Der Wert für den *Worst Case* wird benötigt, um die Latenz des Teilsystems (PCE und RSE) zu bestimmen.

4.3.5 Zusammenfassung

Der erste Teil der Paketklassifizierung - die Schlüsselbildung - wird in der SecAN-Hardware durch die *Packet Classification Engine* übernommen. Als zentrale Komponente extrahiert sie relevante Informationen aus dem aktuellen Datenstrom und bündelt sie in einem Frame-Parameterset. Einige der Informationen aus dem Frame-Parameterset werden zur eindeutigen Identifikation des aktuellen Datenstroms herangezogen und zu einer *Flow ID* kombiniert. Abhängig von der Empfangsrichtung des Ethernet-Frames können unterschiedliche Frame-Parameter in die *Flow ID* eingehen. Aufgrund der *Flow ID* wird ein individueller Regelsatz für den Datenstrom gewählt. Frame-Parameterset und Regelsatz werden von der Firewall für eine beschleunigte Datenverarbeitung genutzt. Das Frame-Parameterset enthält zusätzlich relevante Informationen für die Web-Filter- und die *IDS*-Filterstufe.

Als zentrales Modul zur Paketklassifizierung reduziert die *PCE* den Hardware-Aufwand der einzelnen Filterstufen signifikant, da sie diese von Klassifizierungsaufgaben befreit.

Dadurch verringert sie gleichzeitig die Latenz des gesamten Systems. Letztendlich erzielt sie einen Durchsatz von mehr als 6,6 Gbit/s und übertrifft damit deutlich die Anforderungen.

4.4 Paketklassifizierung II: Rule Search Engine

4.4.1 Motivation

Das Paketklassifizierungsproblem besteht, wie in Abschnitt 4.3 beschrieben, aus den Teilproblemen Schlüsselbildung und der Zuordnung von Regeln zu einem Schlüssel. Den ersten Teil des Problems übernimmt die *Packet Classification Engine*. Für den zweiten Teil des Paketklassifizierungsproblems wird die *Rule Search Engine* benötigt. Ihre Aufgabe ist es, anhand der *Flow ID* individuelle Regelsätze zu finden und für den Firewall-Prototyp bereitzustellen. Zwei wesentliche Herausforderungen müssen erfüllt werden. Zum einen gilt es, eine optimale Speicherstrategie für die zu speichernden Daten zu entwerfen. Zum anderen muss ein latenzarmer Zugriff auf die Speicher und ein optimal schnelles Auffinden der benötigten Daten realisiert werden.

4.4.2 Konzipierung

Das Hauptaugenmerk der *RSE* liegt auf dem Speichern von Daten und dem Wiederfinden von Speichereinträgen. Diese Herausforderung beginnt mit der Wahl geeigneter Speicher, wobei die FPGA-internen BRAM-Zellen nicht zur Verfügung stehen, da sie eine stark limitierte Ressource darstellen. Alle weiteren zur Verfügung stehenden Speicher sind extern und weisen eine höhere Latenz als die BRAM-Zellen auf, weshalb ihr Einsatz sorgfältig geplant sein muss.

4.4.2.1 Wahl geeigneter Speicher

Auf der Zielplattform stehen ein ZBT-SRAM und ein DDR2-Speicher zur Verfügung. Prinzipiell können beide für die Speicherung der Regelsätze verwendet werden. Zu beachten ist, dass durch das Hashen der *Flow IDs* Kollisionen entstehen können. Kollisionen lassen sich erkennen, wenn ein anderes Datum als das der *Flow ID* im Speicher hinterlegt wurde. Dementsprechend müssen neben den Regelsätzen auch die jeweiligen *Flow IDs* gespeichert werden. Somit setzt sich der Speicherbedarf wie folgt zusammen:

$$MEM_{Mapping} \;=\; 32.000\ (\textit{Flow IDs}) \cdot 126\ \textit{Bit}\ (\textit{max.Länge}) \;\approx\; 493\ \textit{KByte}$$

Die maximale Länge eines Regelsatzes wird auf 1.024 Byte definiert. Diese Definition stellt sicher, dass die Verarbeitungslatenz eines Regelsatzes innerhalb der Vorgaben aus der Anforderungsanalyse (vgl. Abschnitt 3.1) bleibt. Weiterhin benötigen 1.000 Regelsätze (vgl. Abschnitt 3.1) einen Speicher der Größe:

$$MEM_{Ruleset} \;=\; 1.000 \cdot 1.024\ \textit{Byte} \;=\; 1.000\ \textit{KByte}$$

In der Summe werden ca. 1,5 MByte Speicher belegt. Es ist somit nicht möglich, alle Informationen im 1 MByte großen ZBT-SRAM zu speichern. Um nicht alle Informationen in dem latenzreichen DDR2-Speicher hinterlegen zu müssen und gleichzeitig einen optimierten Zugriff auf die Daten sicherzustellen, wird ein kaskadierter Ansatz präferiert.

Dabei dient der Hash-Wert der *Flow ID* als Speicheradresse im ZBT-SRAM. In diesem Speicher wird neben der *Flow ID* eine *Rule ID* abgespeichert. Die *Rule ID* umfasst die Startadresse des Regelsatzes sowie dessen Länge im DDR2-Speicher. Dieser kaskadierte Ansatz ist in den Abbildungen 4.6 und 4.7 schematisch dargestellt.

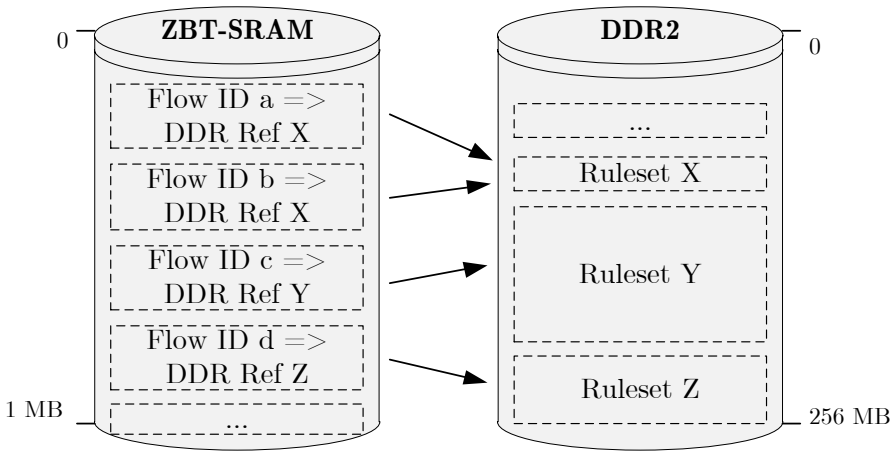


Abbildung 4.7: Kaskadierter Speicheransatz innerhalb der Rule Search Engine

Strategie für die Kollisionsauflösung im SRAM: Aufgrund des gewählten Hash-Verfahrens und in Abhängigkeit vom Belegungsfaktor des Speichers muss eine geeignete Strategie zur Auflösung von Kollisionen gewählt werden. Bei 32.000 Speichereinträgen, die dem Aufbau aus Abbildung C.5 folgen, ist der 1 MB große ZBT-SRAM mit 750 KByte Daten gefüllt. Eine lineare Kollisionsauflösung ist für den Anwendungsfall äußerst un-

Flow IDs gesamt: 32000

günstig. Wie in Abbildung 4.8 dargestellt, können ≈ 60 % der Einträge ohne Kollision

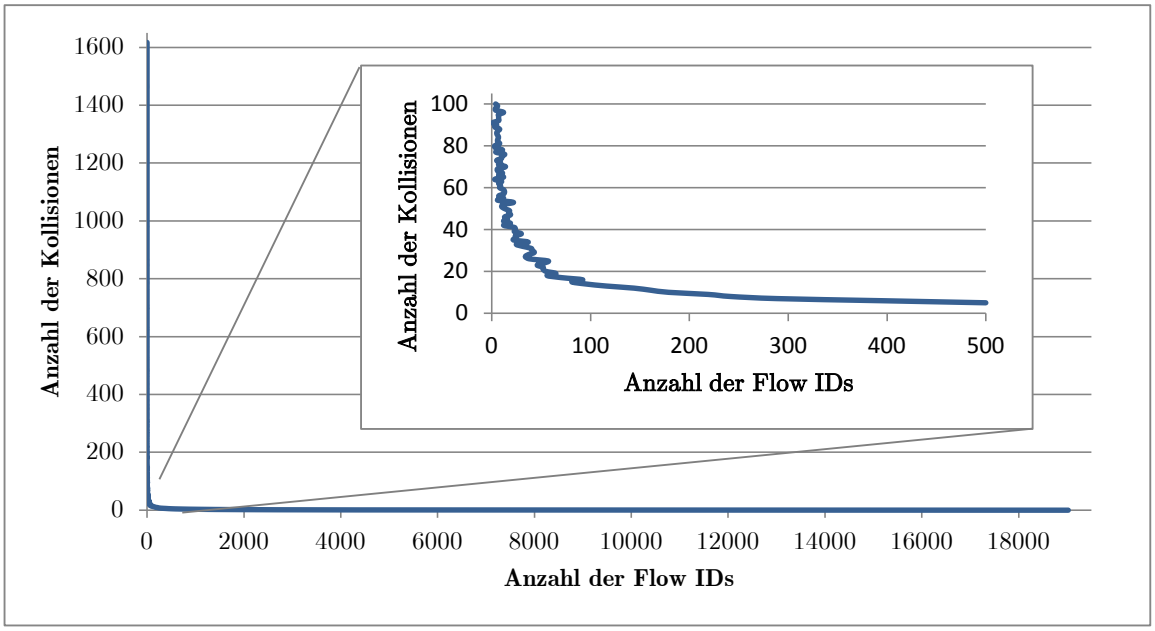


Abbildung 4.8: Lineare Kollisionsauflösung bei 75 % Speicherbelegung

angesprochen werden. Obwohl die Anzahl der *Flow IDs*, die viele Kollisionen verursachen, rapide abnimmt, entstehen bei einem Belegungsfaktor von 75 % bis zu 1.617 Kollisionen im *Worst Case* (vgl. Abbildung 4.8). Aufgrund der hohen Anzahl möglicher Kollisionen entstehen unvertretbare Wartezeiten, welche die QoE der Internet-Nutzer negativ beeinflussen.

Da alle Flow-Daten und deren Zuordnungen bereits zur Konfigurationszeit bekannt sind, wird offline eine optimale Kollisionsauflösung mit Verkettung durch die Konfigurations-Software errechnet. Eine Kollisionsauflösung mit Verkettung, basierend auf denselben Daten, erzeugt maximal sieben Kollisionen. Dabei entstehen genau für eine Speicheradresse diese sieben Kollisionen. Für alle weiteren Adressen werden weniger oder gar keine Kollisionen erzeugt (vgl. Abbildung 4.9).

Da sich die Kollisionsauflösung durch Verkettung deutlich besser für den Anwendungsfall eignet, wird sie der linearen Kollisionsauflösung vorgezogen. Detaillierte Informationen zum Mapping von *Flow ID* zu *Rule ID* sind in Abschnitt C.3 zu finden.

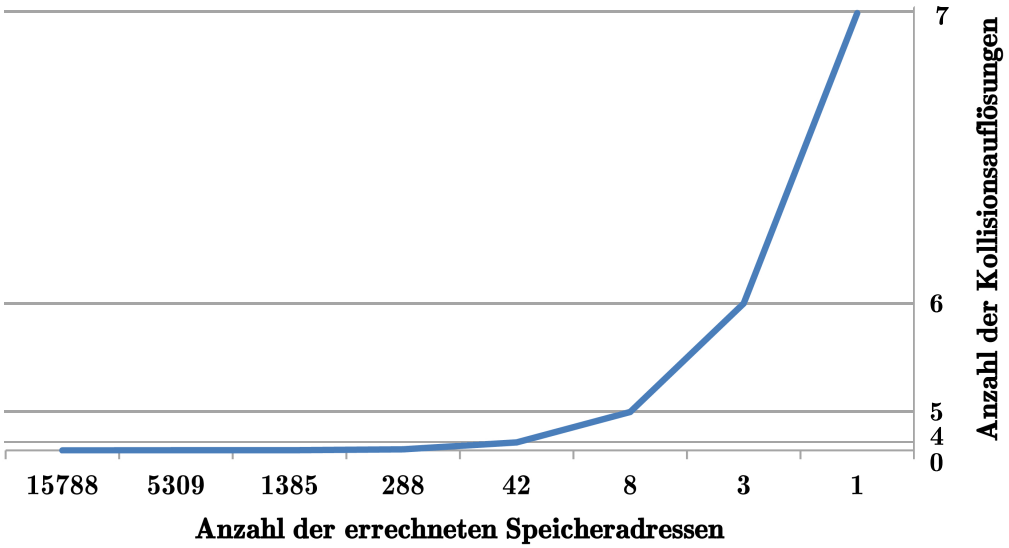


Abbildung 4.9: Kollisionsauflösung durch Verkettung bei 75 % Speicherbelegung

4.4.2.2 Speicherort von Regelsätzen

Aus Sicht des Speichers ist ein Regelsatz eine Bit-Kombination ohne bestimmte Struktur. Erst durch die Interpretation der Firewall-Filterstufen wird er strukturiert. Aufgrund der maximalen Größe der Regelsätze (1.000 Regelsätze x 1024 Byte) wird der verfügbare DDR2-Speicher ausgewählt. Er transferiert 4x mehr Daten je Systemtakt als der ZBT-SRAM (16 Byte statt 4 Byte) und bietet gleichzeitig genügend Reserven für weitere Ergänzungen der Regelsätze. Eine Suche im DDR2-Speicher ist nicht notwendig, da sowohl Startadresse also auch Länge des Regelsatzes aus der *Rule ID* bekannt sind.

4.4.2.3 Zugriff auf externe Speicher

Um die Zugriffszeiten auf externe Speicher zu reduzieren, wurden sowohl für den ZBT-SRAM als auch für den DDR2-Speicher separate Controller entwickelt. Sie ermöglichen den busfreien Zugriff auf den jeweiligen Speicher. Jeder Speicher-Controller sorgt für eine Zeitersparnis von ca. 90 %. Der Einsatz dieser Speicher-Controller wird ermöglicht, da andere Hardware-Module innerhalb desselben Prototyps nicht um diese Ressourcen konkurrieren. Detaillierte Informationen zur Realisierung der RSE bestehen im Wesentlichen aus den beiden Speicher-Controllern und sind in den Abschnitten C.3,C.3.2 und C.3.2.1 zu finden.

Ressourcenbedarf	Genutzt	Verfügbar	Auslastung
Anzahl Slice Register	470	44.800	1,0 %
Anzahl Slice LUTs	1.009	44.800	2,25 %
BlockRam/FIFO	12	296	4,1 %

Tabelle 4.6: Ressourcenbedarf des SRAM-Controllers in der RSE

Ressourcenbedarf	Genutzt	Verfügbar	Auslastung
Anzahl Slice Register	2.806	44.800	6,26 %
Anzahl Slice LUTs	1.009	44.800	4,16 %
BlockRam/FIFO	12	296	4,1 %

Tabelle 4.7: Ressourcenbedarf des modifizierten DDR2-Controllers in der RSE

4.4.3 Ressourcenbedarf

Das gesamte Subsystem wurde umfangreichen Tests unterzogen, bis alle Testmuster erfolgreich verarbeitet werden konnten. Anschließend wurde es auf Geschwindigkeit optimiert. Der Ressourcenbedarf der *Rule Search Engine* teilt sich im Wesentlichen auf die beiden Module, den ZBT-SRAM-Controller und den modifizierten DDR2-Controller, auf. Der ZBT-SRAM-Controller benötigt die in Tabelle 4.6 dargestellten Ressourcen. Mit einer Zielfrequenz von $223,611$ MHz erzielt der SRAM-Controller in der *RSE* einen backannotierten Durchsatz von $\approx 7,155$ $Gbit/s$.

In Tabelle 4.7 ist der Ressourcenbedarf des modifizierten DDR2-Controllers dargestellt. Mit einer Zielfrequenz von $220,836$ MHz erzielt der DDR2-Controller in der *RSE* einen backannotierten Durchsatz von $\approx 7,066$ $Gbit/s$.

Die maximalen Verzögerungen für die Rule Search Engine teilen sich wie folgt auf:

- 42 Takte für die Erfassung einer gültigen *Rule ID* aus dem SRAM und Übergabe dieser an den DDR2-Speicher
- 20 Takte Zugriff auf den DDR2-Speicher
- 32 Takte Lesen aus dem DDR2-Speicher
- 26 Takte Refresh-Zeit (optional)

- 1 Takt für den Zugriff auf die Puffer FIFO im DDR2-Controller

Somit entsteht eine maximale Verzögerung von 121 Takten im System.

4.4.4 Zusammenfassung

Die *Rule Search Engine* löst den zweiten Teil des Paketklassifizierungsproblems und findet zu einem gegebenen Schlüssel - der *Flow ID* - einen entsprechenden Regelsatz für den Firewall-Prototypen.

Das vorgestellte Verfahren benötigt zwei externe Speicher. Im relativ kleinen ZBT-SRAM-Speicher werden Zuordnungsinformationen von *Flow ID* zu *Rule ID* hinterlegt. Dabei dient der Hash-Wert der *Flow ID* als initiale Speicheradresse. Als Referenzwert für eine *Rule ID* wird jeweils die entsprechende *Flow ID* in einem Speichereintrag hinterlegt. Da alle *Flow IDs* bereits zum Konfigurationszeitpunkt bekannt sind, wird für den Fall einer Kollision eine Kollisionsauflösung durch Verkettung durchgeführt. Die notwendigen Informationen sind ebenfalls in jedem Speichereintrag hinterlegt. Zu jeder gültigen *Flow ID* existiert ein Speichereintrag. Waren die *Flow ID*-Daten ungültig, wird die *Default Rule ID* ausgewählt und an den DDR2-Controller übergeben.

Da *Rule ID*-Daten aus Startadresse und Länge des Regelsatzes zusammengesetzt sind, kann eine definierte Speicherstelle im DDR2-Speicher angesprochen werden. Somit entfällt eine zeitaufwendige Suche, wodurch die Performance des Gesamtsystems positiv beeinflusst wird. Ebenfalls performancesteigernd wirken sich die beiden Speicher-Controller aus, da beide ohne den FPGA-intern verfügbaren Processor Local Bus auskommen. Beide Speicher-Controller sind individuell entwickelt und auf die Bedürfnisse des SecAN-Gesamtsystems angepasst worden. Durch die Umgehung des Processor Local Bus (PLB) können die Controller um bis zu 93 % schneller auf den Speicher zugreifen. Mit einem Durchsatz von mehr als 7 Gbit/s übererfüllt die *RSE* die Anforderungen deutlich.

4.5 Firewall innerhalb der Packet Processing Engine

4.5.1 Motivation

Firewalls (FWs) bestehen aus einer oder mehreren Komponenten und bilden die Schnittstelle zwischen privaten und öffentlichen, unsicheren Netzen. Ihre Aufgaben bestehen in

der Überwachung von Kommunikationsregeln für ein Netzwerk und in der Beschränkung auf Ressourcen. Dazu besitzen sie einen definierten Satz an Regeln, der es ihnen ermöglicht, festzustellen, ob der Netzwerkverkehr die Firewall passieren darf oder nicht.

Die gewissenhafte und vollständige Konfiguration setzt ein umfangreiches Fachwissen im Bereich Netzwerke und Netzwerkkommunikation voraus. Da der durchschnittliche Internet-Nutzer dieses Wissen nicht besitzt, wird er nicht in der Lage sein, sein Netzwerk optimal zu schützen. Durch das umfassende Know-how der ISP-Administratoren wird das Firewall-System im TZN in vollem Umfang fehlerfrei konfiguriert. Dadurch entsteht ein Basisschutz für jeden ISP-Kunden, ohne dass dessen Mitwirken erforderlich ist.

Der Einsatz einer Hardware-basierten Firewall stellt einen Forschungsschwerpunkt innerhalb dieser Arbeit dar. An diesem Teilsystem wirkte die Broadband Access Division von Nokia Siemens Networks GmbH & Co. KG in Greifswald durch Inspiration und fortwährende Unterstützung mit. Für diesen Forschungsschwerpunkt läuft derzeit ein Patentanmeldeverfahren [Sch10].

Anmerkung 1: Grundlegende Informationen zu Firewalls befinden sich im Anhang C.4.1. Neben verschiedenen Firewall-Konzepten wird die Linux-Firewall *iptables* und deren Administration näher beschrieben. Bei der Konzipierung der SecAN-Firewall hat sich der Autor der vorliegenden Forschungsarbeit am erfolgreichen Konzept der Linux-Firewall orientiert, sodass die Hardware-Lösung der SecAN-Firewall der Software-Lösung von *iptables* in Bezug auf die Datenverarbeitung ähnelt.

Anmerkung 2: In Abschnitt 4.5.3 wird der Aufbau der Firewall-Komponenten beschrieben. Die weiteren Filterstufen der PPE besitzen denselben strukturellen Aufbau, weshalb in den nachfolgenden Kapiteln nicht weiter darauf eingegangen wird.

4.5.2 Stand der Technik

Auch wenn bereits eine Vielzahl an Firewall-Lösungen existieren, handelt es sich dennoch um ein hoch aktuelles Thema. So existieren Software-Lösungen von namhaften Herstellern. In [SK10] wurde das McAfee Security Center 9.3 und im Speziellen die Firewall-Lösung näher untersucht. McAfee verspricht mit seinem Security Center 9.3 größtmögliche Sicherheit für den Endkunden. Getestet wurden DDoS-Attacken (Ping) mit unterschiedlicher Netzwerklast. Erstaunlicherweise stürzte das Zielsystem abhängig von der DDoS-Last immer ab, wenn die Funktionalität zum Blockieren von Internet Control Message Protocol (ICMP)-Nachrichten aktiviert war. Bei Angriffen mit einer

Last von 100 Mbit/s betrug die Zeit bis zum Absturz ca. 2,5 Minuten, mit auf 1 Mbit/s reduzierter Last ca. 3 Stunden. Die Ursache war, dass sich der RAM-Speicher sukzessive bis zum Maximum füllte und das, obwohl keine weitere Programme auf dem Zielsystem liefen. Zusätzlich erzeugten die Attacken eine CPU-Last von 34% bei einem Core2 Quad mit 2.4 GHz. In [SK11] wurden diese Ergebnisse bestätigt.

Dieses Szenario zeigt, dass eine Software-Firewall auf einem lokalen System dieses bis hin zum Absturz stark auslasten kann. Die SecAN-Firewall befindet sich hingegen im TZN. Sie schont damit die lokalen Ressourcen und kann z. B. ICMP-Angriffen aus dem Internet, wie sie in [SK10] beschrieben wurden, abfangen, bevor diese das Zielsystem erreichen.

In [YY10] werden die Nachteile einer reinen Software-Firewall beschrieben. Da besonders die Paketklassifizierung viel CPU-Zeit in Anspruch nimmt, wurde in [YY10] eine hybride Lösung, bestehend aus einem Linux PC mit Netfilter-Funktionalität und einem FPGA als Co-Prozessor für die Paketklassifizierung und das Auffinden der Regeln vorgestellt. Aufgrund der begrenzten Ressourcen des FPGA (Cyclone II EP2C20F484C8) sind maximal 500 Regeln möglich, wobei das Gesamtsystem mit nur 3 Regeln (alles Port-Sperren) erfolgreich getestet wurde. Weiterhin ist das beschriebene System, gleich der SecAN-Firewall, durch den Endkunden konfigurierbar. Leider werden keine Aussagen in Richtung Systemperformance dieser hybriden Lösung getätigt. Nachteilig dürfte sich die aufwendige Buskommunikation zwischen CPU, FPGA und externen Speichern auswirken, die pro Paket mehrfach erfolgt.

In [AD11] ist eine Netzwerk-Firewall als Paket Filter für den durchschnittlichen Internet-Nutzer vorgestellt. Ein Nios II 32-Bit Soft Core Microprocessor mit einem Echtzeitbetriebssystem vergleicht die Header-Felder des eingehenden Datenstromes mit den Speichereinträgen des generierten CAMs. Gegenüber der SecAN-Firewall findet die Bewertung auf sehr simple Weise statt. Werden Tupel aus IP-Adressen und Port-Nummern im CAM gefunden, darf der Datenstrom die Firewall passieren. Der Vergleich mit der Linux Firewall *iptables* zeigt, dass die vorgestellte Lösung nicht sensitiv auf unterschiedliche Paketgrößen reagiert.

Dennoch weist die Lösung der Autoren Ajami Raouf und Dinh Anh einige Unterschiede bzw. Schwächen gegenüber der SecAN-Firewall auf. Um den theoretischen Durchsatz von 2,9 Gbit/s zu erzielen, ist der Einsatz von kostspieligen CAM-Modulen notwendig. Im Vergleich dazu erreicht die SecAN-Firewall ohne CAM-Modul einen theoretischen Durchsatz von mehr als 6,3 Gbit/s. Während von der SecAN-Firewall 32.000 Regeln

unterstützt werden, können aufgrund der begrenzten CAM-Ressourcen in [AD11] lediglich 16 Ports und 16 IP-Adressen/Bereiche überprüft werden. Damit ist die in [AD11] vorgestellte Lösung für kleine Netze durchaus geeignet. Sie lässt sich jedoch aufgrund der Architektur und des Durchsatzes nicht im TZN einsetzen.

Khakpour und Liu stellen in ihrem Paper „First Step Toward Cloud-Based Firewalling“ [KL12] eine „Paket Filter“-Lösung für den Einsatz im TZN vor. Ihr Bloom-Filter-basierter Ansatz richtet sich an Unternehmen. Die Implementierung erfolgte in C++ und lief auf einem UNIX-System mit einem Intel(R) Xeon(R) Quad-cores CPU, welcher mit 2.66 GHz getaktet wurde und über 16 GB RAM verfügte. Ihre Tests weisen nach, dass der von ihnen gewählte Ansatz einen Durchsatz von 13.346 Mbit/s bis 1,143 Gbit/s erzielt. Zudem weisen sie darauf hin, dass ihre Testumgebung mit einem reduzierten Regelwerk arbeitet, weil dies die verfügbaren Ressourcen auf der Zielplattform erforderten. Reduziert bedeutet in diesem Fall, dass sich der IP-und der Port-Bereich nicht über den vollen Umfang erstrecken.

Obwohl der in [KL12] beschriebene Lösungsansatz ebenfalls für TZN umgesetzt wurde, weist er gegenüber der SecAN-Firewall einige Schwächen auf. So müssen in [KL12] performante CPUs mit 16 GB RAM allein für die Firewall-Lösung eingesetzt werden. Gleichzeitig muss das System mit einem reduzierten Regelwerk auskommen. Einerseits entspricht der maximal erreichte Durchsatz den Anforderungen, die an die SecAN-Firewall gestellt werden. Andererseits befindet sich der minimale Durchsatz bei dem für ein TZN nicht akzeptablen Wert von ca. 13,5 Mbit/s. Die SecAN-Firewall hingegen erreicht selbst bei minimalen Frame-Längen einen Durchsatz von $\approx 6,337\ Gbit/s$.

Unternehmen wie Netgear, Cisco und Zyxel bieten bereits für unter 1.000,-€ Hardware-Firewall-Lösungen an (z. B. Netgear ProSafe FVS318N, Cisco Systems ASA 5505-BUN-K9, Zyxel ZyWALL USG-60). Sie erreichen Durchsätze zwischen 95 Mbit/s und 1 Gbit/s, müssen jedoch vom Endkunden konfiguriert werden. Höherwertige Produkte, z. B. Cisco (K/ASA 5510 Appl +AIP-SSM-20 5FE 3DES-AES) und WatchGuard (WatchGuard XTM 330 Network Security/Firewall), erreichen leicht Preise bis in den mittleren fünfstelligen Bereich. Beide Gruppen sind für den durchschnittlichen Internet-Nutzer unattraktiv. Einerseits kommen hohe Investitionen auf den Internet-Nutzer zu und zum anderen müssen die Produkte mit hohem technischen Verständnis eingerichtet und gewartet werden. Die SecAN-Firewall hingegen wird professionell eingerichtet und gewartet und skaliert mit steigenden Datenraten. Weiterhin können ISP alle anfallenden Kosten über eine größere Kundengruppe teilen, wodurch ein Mehrwert für alle angeschlossenen

Internet-Nutzer entsteht.

4.5.3 Konzipierung

Im Laufe der Arbeit wurden mehrere Punkte, die die Konzipierung des Firewall-Systems beeinflussen, bereits geklärt. So wurde in Abschnitt 3.2.2.1 festgestellt, dass bereits der Einsatz einer *stateless* arbeitenden Firewall im TZN einen deutlichen Mehrwert für die Internet-Nutzer erbringt. Hinzu kommt, dass die im TZN herrschenden hohen Anforderungen bezüglich der Verarbeitungsgeschwindigkeit durch eine *stateful* arbeitende Firewall derzeit nicht realisierbar sind (vgl. [Sie07]).

Um den Vergleich von Frame-Daten und Filterregeln zu beschleunigen, wurde von der *PCE* ein Frame-Parameterset erstellt (vgl. Abschnitt 4.3.3). Ferner befinden sich die Filterregeln in einem Regelsatz, welcher im DDR2-Speicher der *RSE* hinterlegt und über die *PCE* an das Firewall-Modul ausgeliefert wird. Der Aufbau der Regelsätze sowie die darin enthaltenen Regeln sind im Anhang C.4.2 näher beschrieben. Die gleichzeitige Auslieferung des Daten-Frames, des Frame-Parametersatzes und des Regelsatzes bilden die Rahmenbedingungen für eine hochperformante Datenverarbeitung im Firewall-Modul. Nun bleibt zu klären, ob es vorteilhafter ist, die Firewall-Komponenten parallel oder seriell anzuordnen.

4.5.3.1 Paralleles vs. serielles Arrangement der Firewall-Komponenten

Die Firewall besteht aus acht Hardware-Modulen. Zum einen bilden sie die Sicherheitsmerkmale, die bereits heute im TZN umgesetzt sind, ab (vgl. Abschnitt 3.2.1). Zum anderen werden klassische Filterungen nach MAC- und IP-Adressen realisiert. In der Konfigurationsoberfläche in Abschnitt B.2 sind die acht Firewall-Filterstufen der Management Plane grafisch dargestellt (vgl. Abbildung B.6). Dieser Abschnitt vergleicht die Vor- und Nachteile einer parallelen bzw. seriellen Anordnung der Firewall-Module:

- Latenz des Firewallsystems: In einem parallel ausgeführten Ansatz wird die Latenz durch die entscheidungsaufwendigsten Firewall-Komponenten sowie einer nachgeschalteten Entscheidungslogik bestimmt. Dagegen muss in einem seriellen Ansatz die Latenz über alle Firewall-Komponenten bestimmt werden. Durch die Einführung des dpt (vgl. Abbildung C.2) verdoppelt sich der interne Datendurchsatz von

16 bit/s auf 32 bit/s. Da weiterhin in jedem Takt eine Bewertung des eingehenden Datenstromes durch die Firewall-Komponenten erfolgt, ist eine Verzögerung ausgeschlossen. Zusätzlich würden die Backpressure-Signale (vgl. Abschnitt C.1.1) dafür Sorge tragen, dass es nicht zu internen Stauungen kommen kann. Somit sind sowohl der serielle als auch parallele Ansatz denkbar.

- Priorisierung von Filterregeln: Jede Firewall-Komponente übernimmt genau eine Firewall-Regel. In einem seriellen Ansatz wird durch die Variation der Filterstufen eine inhärente Priorisierung ermöglicht. Um eine Priorisierung bei einer parallelen Anordnung zu realisieren, ist eine zusätzliche Auswertelogik erforderlich. Letztere erfordert extra Hardware-Ressourcen. Somit ist in Hinsicht der Periodisierung ein serieller Ansatz der Firewall-Komponenten zu bevorzugen.
- Erweiterbarkeit des Firewall-Systems: Durch die Architektur der Firewall-Komponenten (vgl. Abbildung 4.10) wird eine leichte Erweiterbarkeit der SecAN-Firewall sichergestellt. Der zusätzliche Aufwand für eine serielle Anordnung beschränkt sich auf die Verdrahtung der neuen Filterstufen. Währenddessen muss es in einem parallel angeordneten Firewall-System eine vorgeschaltete Verteillogik für die seriell eintreffenden Firewall-Regeln geben. Diese muss für jede nachträglich hinzugefügte Firewall-Komponente erweitert werden.

Da praktisch keine Nachteile bei einem seriellen Ansatz vorhanden sind und die Vorteile für eine serielle Anordnung der Firewall-Komponenten sprechen, entscheidet sich der Autor für eine seriell ausgeführte SecAN-Firewall. Folgende Maßnahmen werden sich ebenfalls positiv auf die Firewall-Performance auswirken und die Systemlatenz verkürzen:

1. Jede Filterstufe entfernt die von ihr verwendete Regel aus dem Regelsatz. Dadurch verkürzt sich der Regelsatz sukzessive, wodurch eine konstante Suchzeit entsteht.
2. Jede Firewall-Komponente besitzt eine eindeutige ID. Untersucht eine Firewall-Komponente das erste *Type*-Feld des Regelsatzes und stellt fest, dass ihre eigene ID mit der ID im *Type*-Feld übereinstimmt, so wird sie die nachfolgende Regel verarbeiten. Anderenfalls wird der Datenstrom durch die Komponente nicht berücksichtigt.
3. Die Anordnung der Regeln im Regelsatz entspricht der Anordnung der Firewall-Komponenten. Unter Berücksichtigung der beiden zuvor genannten Maßnahmen bedeutet das, dass jede Filterstufe nur das erste *Type*-Feld überprüfen muss. Stellt

sie keine Übereinstimmung fest, wird sich keine Filterregel für sie im Regelsatz befinden.

4.5.3.2 Konzept der Firewall-Komponenten

Trotz der komplexen Struktur sind hohe Flexibilität, leichte Erweiterbarkeit und Wartung äußerst wichtig für alle SecAN-Prototypen. Dies gilt für die Firewall-Komponenten ebenfalls. Da alle Firewall-Komponenten dieselben Eingangsparameter benötigen, werden alle Filterstufen einen identischen äußeren Aufbau aufweisen (vgl. Abbildung 4.10).

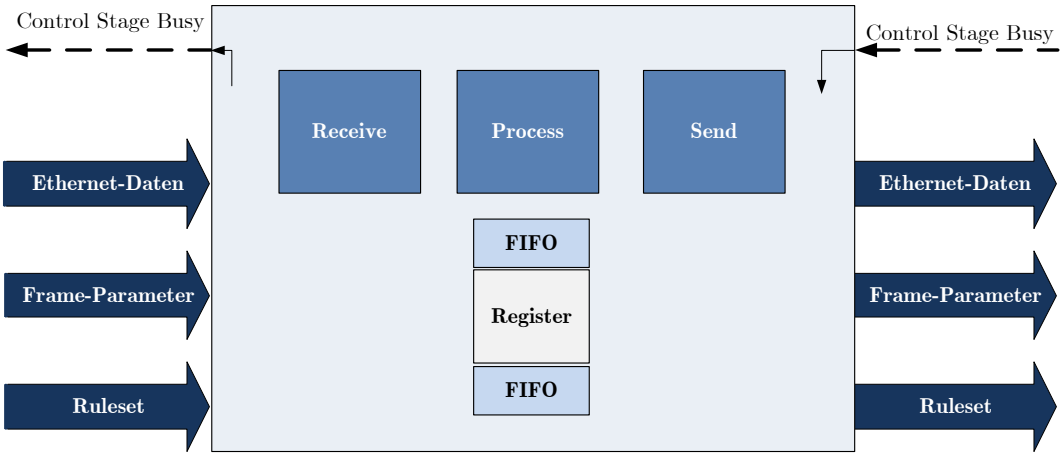


Abbildung 4.10: Aufbau einer Firewall-Filterstufe

Ein identischer Aufbau erleichtert zudem die Neuanordnung von Filterkomponenten, um bspw. eine globale Priorisierung der Filterregeln zu verwirklichen. Darüber hinaus wird jede Filterkomponente intern immer aus drei funktionalen Modulen sowie Pufferspeichern für die Eingangswerte in Form von FIFOs und Registern bestehen (vgl. Abbildung 4.10). Das *Receive*-Modul steuert den Empfang der Eingangsdaten, nachdem das Backpressure-Signal *Control Stage Busy* die Filterkomponente freigibt. Anschließend erfolgt durch die Logik des *Process*-Moduls die Auswertung des ersten Type-Feldes und ggf. die Extraktion und Verarbeitung der Filterregel. Wenn die Bewertung der Eingangsdaten nicht zum Verwurf dieser führt, müssen alle Eingangsdaten an die nächste Filterkomponente bzw. an den Ausgang der Firewall weitergeleitet werden. Diese Aufgabe übernimmt das *Send*-Modul, wenn das Backpressure-Signal *Control Stage Busy* dies zulässt. Da die Realisierung der verschiedenen Filterstufen nahezu identisch abläuft,

hat sich der Autor für die exemplarische Realisierung der *Destination Port*-Filterstufe entschieden und wird diese in Abschnitt 4.5.4 detaillierter vorstellen.

4.5.4 Realisierung

In diesem Abschnitt wird exemplarisch die Realisierung der *Destination Port*-Filterstufe beschrieben. Sie soll untersuchen, ob der *Destination Port* im Datenstrom einem bestimmten Port entspricht bzw. Teil eines Port-Bereichs ist. In Abhängigkeit von der gewünschten Aktion wird der Datenstrom verworfen oder weitergeleitet werden. Die Umsetzungen der weiteren Filterkomponenten sind grundsätzlich ähnlich und unterscheiden sich in Punkten wie Latenz, Komplexität der Filterregel und Größe der Zwischenspeicher.

Jede Filterkomponente zeigt über das Ausgangssignal *Control Stage Busy* an, ob sie bereit ist, neue Eingangsdaten aufzunehmen. Ist dies der Fall, werden Ruleset, Daten-Frame und Frame-Parameterset an je einen internen Zwischenspeicher weitergeleitet. Die Zwischenspeicher sind in der Lage, jeweils einen maximal langen Frame, ein maximal langes Ruleset sowie den gesamten Frame-Parametersatz aufzunehmen.

Parallel mit dem Einlesen der Eingangsdaten wertet das *Process*-Modul die Regel ID innerhalb des *Type*-Feldes der ersten Regel aus. Wird keine Übereinstimmung festgestellt, können die gesamten Eingangsdaten durch das *Send*-Modul weitergeleitet werden, wenn das eingehende *Control Stage Busy*-Signal dies erlaubt.

Für den Fall das Filterstufen ID und Regel ID übereinstimmen, wird die Aktion (Drop/ Accept) und das *Match*-Bit gespeichert. Am Ende der Kontrolle des Datenstromes wird die Aktion in Abhängigkeit von dem *Match*-Bit ausgeführt. Da die gegenwärtige Regel für zukünftige Zwecke nicht mehr benötigt wird, wird sie durch eine Schreibunterbrechung in die Ruleset-FIFO aus dem Regelsatz entfernt. Die Schreibunterbrechung wird erst aufgehoben, wenn das Ende der aktuellen Regel erreicht wurde.

Type-Feld und *Length*-Feld folgen dem Aufbau, wie in Abschnitt C.4.2 beschrieben. Dabei werden als gängige Protokolle TCP, UDP, Encapsulating Security Payload (ESP), General Routing Encapsulation (GRE) und ICMP unterstützt. Das Auftreten anderer Protokolle als der im *Type*-Feld unterstützten führt zu einem Verwurf aller Eingangsdaten. Im ersten Takt wird die ID der Filterstufe überprüft. Ab dem zweiten Takt wird das *Value*-Feld der aktuellen Regel abgearbeitet. Im Beispiel der *Destination Port*-Filterstufe enthält es Port-Angaben nach dem in Abbildung 4.11 vorgestellten Schema.

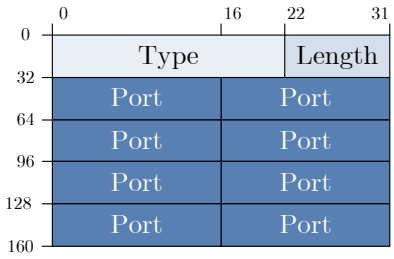


Abbildung 4.11: Aufbau der Regel für eine OSI-Layer 4-Filterstufe

Jede Port-Angabe wird mit 16 Bit kodiert. Das bedeutet, dass immer zwei Port-Angaben je Systemtakt ausgewertet werden können. Bei einem speziellen Port sind beide eingelesenen Port-Angaben identisch. Unterscheiden sich die beiden Port-Angaben, handelt es sich um einen Port Range.

Abhängig von den Anforderungen (Match/No Match sowie Drop/Accept) wird ein spezieller Logikpfad in der Filterstufe gewählt. Die Logik des *Process*-Moduls vergleicht sukzessive den im Frame-Parameterset übertragenen Destination Port mit den Port-Angaben aus der Destination Port-Filterregel und gleicht das Ergebnis mit den Anforderungen ab.

Wird aufgrund der Anforderungen der Daten-Frame verworfen, werden Ruleset, Daten-Frame und Frame-Parameterset bis zum jeweiligen Ende aus den Zwischenspeichern ausgelesen und nicht weitergeleitet. Auf diese Weise werden die Daten innerhalb der Filterstufe vernichtet. Ist dies geschehen, zeigt das ausgehende *Control Stage Busy*-Signal an, dass die Filterstufe neue Eingangsdaten aufnehmen kann. Wurden die Eingangsdaten als unbedenklich eingestuft, muss das eingehende *Control Stage Busy*-Signal ausgewertet werden, um festzustellen, ob die Eingangsdaten an die nächste Filterkomponente bzw. den Firewall-Ausgang weitergeleitet werden können.

4.5.5 Test und Ergebnisse

4.5.5.1 Simulativer Test der Firewall-Filterstufen

In Abschnitt D.4 wurde bereits das funktionale Testen mittels Modelsim beschrieben. Die SecAN-Firewall umfasst acht separate Firewall-Filterstufen. Diese wurden ausführlich getestet. Strukturen und Funktionalitäten jeder Filterstufe wurden in der Konzeptphase

definiert und als Entscheidungsdiagramm mit sämtlichen Entscheidungen und Kommunikationspfaden hinterlegt. Auf dieser Basis konnte für jede Filterstufe ein individueller Testkatalog erstellt werden. Die Testfälle sorgen für eine hundertprozentige Pfadabdeckung.

Tabelle 4.8 stellt je Filterstufe die Anzahl aller Entscheidungen und Kommunikationspfade dar. Teilweise entstehen Zyklen in den Kommunikationspfaden, die erst nach mehrmaligem Durchlaufen beendet werden. Jeder dieser Kommunikationspfade stellt einen funktionalen Testfall dar.

Filterstufe	Entscheidungen	Kommunikationspfade
Destination MAC	56	82
Source MAC	56	82
MAC Address Translation	46	52
Source IP	66	86
Destination IP	66	86
IP Antispoofing	74	102
Source Port	132	156
Destination Port	132	156

Tabelle 4.8: Firewall-Filterstufen und funktionale Testfälle

Für jede Entscheidung wurde je ein *Good Case*- und ein *False Case*-Testfall entwickelt. Diese belaufen sich auf 2 x 640 = 1280 Testfälle. Zusätzlich wurden für eine vollständige Pfadabdeckung 802 weitere Testfälle entwickelt. Um eine vollständig korrekte Funktionalität aller Firewall-Filterkomponenten nachzuweisen, wurden 2.082 Testfälle erstellt und erfolgreich getestet. Die Frame-Länge betrug jeweils fix 590 Byte. Abschließende Tests konnten simulativ die vollständige Funktionalität nachweisen.

4.5.5.2 Berechnung der maximalen Last

Das SecAN-Firewall-System wird ohne *packet loss* arbeiten, wenn die interne Verzögerung (T_{int}) $\leq$ der Verarbeitungszeit der externen Daten (T_{ext}) ist. Der größte Stress für das System entsteht, wenn folgende Bedingungen gleichzeitig eintreten:

- Datenblöcke mit minimal langen Ethernet-Frames treffen auf das System

- Jede Filterstufe muss eine Regel abarbeiten

Jeder Datenblock besitzt folgenden Aufbau:

- 12 Byte Inter Frame Gap
- 8 Byte Präambel & Start Frame Delimiter
- 60 Byte minimaler Ethernet-Frame
- 4 Byte Frame Check Sequence

Somit besitzt ein minimaler Datenblock eine Länge von 84 Byte. Bei einem Ethernet-Gbit-Betrieb ist die Dauer eines Taktes 8 ns. Dementsprechend ist T_{ext} = 84 Byte * 8 ns = 672 ns.

Nach *Post Place & Route* erreicht das SecAN-Firewall-System eine Frequenz von 198,036 MHz. Ein interner Systemtakt (CT_{int}) dauert somit $\frac{1}{198{,}036\ \text{MHz}} \approx 5{,}05\ ns$. Folglich kann das interne Firewall-System mit: $T_{int} = \frac{T_{ext}}{CT_{int}} \approx 133{,}08\ Takte = 133\ Takte$ verzögern, ohne *packet loss* zu riskieren. Damit besitzt das interne System eine Latenz von $133\ Takte * 5{,}05\ ns \approx 671\ ns$. Diese Latenz liegt $\approx 0{,}5\ ns$ unter der des externen Systems.

Im SecAN-Firewall-System besteht der kritische Pfad in der Bewertung des Datenstromes durch die Filterstufen. Für die abschließenden Lasttests wird ein Testregelsatz mit je einer Regel pro Filterstufe erzeugt. Dieser besitzt den Aufbau und erzeugt Verzögerungen, die in Tabelle 4.9 dargestellt sind.

Filterstufe	**Bit/Regel**	**Verzögerungen [Takte]**
Destination MAC	80	3
Source MAC	80	3
MAT	80	3
Source IP	64	2
Destination IP	64	2
IP-Antispoof	112	4
Source Port	1856	58
Destination Port	1856	58

Tabelle 4.9: Aufbau des Testfiltersatzes für die SecAN-Firewall

Die hohe Anzahl der Bit der Port-Filterstufen ergibt sich aus den 114 Port-Vergleichen. Sie werden auf beide Port-Filterstufen aufgeteilt. Der Testregelsatz erzeugt eine Verzögerung von 133 Takten und besitzt eine Länge von 4192 $Bit = 524$ $Byte$.

4.5.5.3 Hardware-Test der Firewall-Filterstufen

Für den Lasttest auf der Zielplattform wurden folgende Veränderungen gegenüber dem simulativen Test durchgeführt:

- Die Frame-Länge wurde auf die minimale Ethernet-Frame-Länge von 64 Byte (+ Inter Frame Gap (IFG) + Präambel + SFD = 84 Byte) angepasst.
- Die Länge des Regelsatzes wurde so optimiert, dass die Firewall-Filterstufen unter maximaler Last stehen.

Für diesen Test wurde der Traffic-Generator [TDT11] verwendet. Er ist in der Lage, unterbrechungsfrei eine maximale Datenrate von 1 Gbit/s zu erzeugen. Die Filterstufen innerhalb der SecAN-Firewall wurden im Verbund getestet. Wie aus den Simulationsergebnissen und den Lastberechnungen bereits zu erwarten war, wurden alle Testmuster erfolgreich verarbeitet. Damit konnte nachgewiesen werden, dass der SecAN-Firewall-Prototyp bei einer externen Datenrate von 1 Gbit/s kein *packet loss* erzeugt.

4.5.5.4 Ressourcenbedarf und Durchsatz

Die Syntheseergebnisse für das SecAN-Firewall-System sind in Tabelle 4.10 dargestellt. Sie zeigen, dass der Firewall-Prototyp mit ca. 4/5 der verfügbaren Ressourcen auf der Zielplattform umgesetzt werden kann.

Ressourcenbedarf	Genutzt	Verfügbar	Auslastung
Anzahl Slice Register	39.066	44.800	87.2 %
Anzahl Slice LUTs	38.260	44.800	85,4 %
BlockRam/FIFO	236	296	79,7 %

Tabelle 4.10: Ressourcenbedarf des Firewall-Prototyps

Nach *Post Place & Route* beläuft sich die Zielfrequenz des SecAN-Firewall-Systems auf $198,036 MHz$. Bei einer internen Datenverarbeitungsbreite von $32 Bit/Takt$ entspricht

das einem Durchsatz von $\approx 6,337\ Gbit/s$.

4.5.6 Zusammenfassung

Im Abschnitt 4.5 wurde das SecAN-Firewall-System zunächst konzipiert, anschließend entwickelt und letztendlich erfolgreich getestet.

Das System ähnelt von der Datenverarbeitung her der effizienten Linux-Firewall *iptables*, wurde jedoch in Hardware auf einem XILINX-Entwicklungsboard mit einem Virtex 5 FPGA umgesetzt [Xil09]. Es weist allein durch die Datenverarbeitung auf dem FPGA eine deutliche Performance-Steigerung gegenüber dem Autor bekannten Software-Lösungen auf.

Beim SecAN-Firewall-System handelt es sich — gleich dem Gesamtsystem SecAN — um ein modulares System. Demnach ist es leicht zu warten und zu erweitern. Durch die Erhöhung des internen Datenpfades weist das Firewall-System kein *packet loss* auf. Ferner existieren weder *False Positives* noch *False Negatives*, da die Aktionen der Filterregeln nur dann ausgeführt werden, wenn das entsprechende Suchkriterium eindeutig im Datenstrom identifiziert wurde.

Die Funktionalität einer statischen Firewall wird durch acht Filterstufen, welche auf den OSI-Schichten der Sicherungs- bis Transportschicht arbeiten, gewährleistet. Gegenüber lokalen Lösungen bietet das im TZN arbeitende SecAN-Firewall-System Netzwerkschutz, bevor ein Angreifer das Zielsystem erreicht. Durch die Ansiedlung des Firewall-Systems im TZN kann es ausschließlich durch Administratoren des ISP konfiguriert werden. Darüber hinaus ist es netzwerktechnisch nicht adressierbar, wodurch es einem Angreifer deutlich erschwert wird, das Firewall-System zu kompromittieren. Das System weist den in Tabelle 4.10 dargestellten Ressourcenverbrauch auf und erfüllt die geforderten Zielvorgaben hinsichtlich des Durchsatzes um mehr als den Faktor 6.

4.6 Web-Filter innerhalb der Packet Processing Engine

4.6.1 Motivation

Im privaten wie geschäftlichen Umfeld, unter Datenschützern und auf übergeordneten Ebenen wie bspw. Regierungen wird die Thematik Web-Filterung intensiv diskutiert und teilweise aktiv umgesetzt. Argumente wie die Sicherung der Rechte an geistigem

Eigentum, der Schutz der nationalen Sicherheit, die Einhaltung kultureller Normen und religiöser Werte sowie die Bewahrung von Kindern vor Pornografie und Ausbeutung können den Einsatz von Web-Filtertechniken rechtfertigen (vgl. [CW10]). Losgelöst von diesen Diskussionen soll der Fokus auf den Mehrwert für die definierte Zielgruppe - den durchschnittlichen Internet-Nutzer - gerichtet werden. Diese Zielgruppe enthält neben unbedarften Erwachsenen zunehmende auch Kinder und Jugendliche. Um die gesamte Zielgruppe vor den erwähnten Inhalten zu schützen, bedarf es einer Filterung der verfügbaren Web-Inhalte. Hinzu kommt, dass auch im geschäftlichen Umfeld Web-Filter-Lösungen eingesetzt werden können, um bspw. die Produktivität zu steigern und dabei gleichzeitig Haftungsrisiken zu senken. Aus den beschriebenen Gründen soll das SecAN-Gesamtsystem um eine Web-Filter-Lösung erweitert werden.

4.6.2 Stand der Technik

4.6.2.1 Filtertechniken

Es existiert eine Reihe von Strategien, um Internet-Nutzer zu „motivieren“, als sensibel eingestufte Inhalte nicht zu erreichen. Diese sind (vgl. [CW10, Var10]):

1. Manipulation von Suchmaschinenergebnissen
2. Blockade von IP-Adressen
3. Manipulation von DNS-Daten (DNS-Poisoning)
4. Inhaltliche Filterung auf höheren Protokollschichten (z. B. URL-Blockade)
5. Löschung zensierter Web-Inhalte

Diese Punkte werden nachfolgend näher betrachtet.

Manipulation von Suchmaschinenergebnissen: Unter der Manipulation von Suchmaschinenergebnissen ist zu verstehen, dass die von der Suchmaschine ausgelieferten Ergebnisse nicht denen entsprechen, die ein Nutzer erzielen wollte. Einerseits kann diese Technik benutzt werden, um Malware zu verbreiten[3], andererseits lässt sie sich auch für die Blockade von Web-Inhalten einsetzen. Durch Letzteres kann für kontrollierende Organisationen sowie für die gewählte Zielgruppe ein Mehrwert entstehen. Dieser stellt

[3] http://www.ip-insider.de/themenbereiche/sicherheit/bedrohungen-attacken/articles/412632

sich jedoch nur ein, wenn alle potentiell nutzbaren Suchmaschinen dieselbe Filterliste nutzen und entsprechende Inhalte nicht ausliefern.

Dass die Manipulation von Suchmaschinenergebnissen eine gängige Praxis ist, beweist z. B. der Konflikt zwischen der Regierung der Volksrepublik China und dem Suchmaschinenbetreiber Google[4].

Blockade von IP-Adressen: Durch die Blockade von IP-Adressen wird die vollständige Kommunikation mit den entsprechenden Servern unterdrückt. Da lediglich der IP-Header untersucht werden muss, lässt sich diese Art der Filterung bei niedrigem Ressourcenaufwand sehr gut in Hardware durchführen. Nachteilig bei diesem Verfahren ist jedoch, dass alle Web-Seiten, die unter derselben IP „gehostet“ werden, durch diese Filterung nicht mehr erreichbar sind — es entsteht ein sogenanntes „Overblocking“.

Manipulation von DNS-Daten: Grundsätzlich können DNS-Daten auf drei unterschiedliche Arten manipuliert werden [Var10, CW10].

1. **Name hijacking:** Es wird nicht die IP-Adresse der angefragten Domain zurückgesandt, sondern eine beliebige andere IP-Adresse.
2. **Name subversion:** Der DNS-Server antwortet mit einer ungültigen IP-Adresse. Diese führt in der Folge zu einer „Could not connect“-Fehlermeldung.
3. **Silence:** Der DNS-Server verweigert das Ausliefern einer Antwort. Auf diese Weise entsteht ein „time out“.

Auch bei dieser Technik entsteht ein „Overblocking“. So ist es bspw. nicht angemessen, Domains wie *myspace.com* zu sperren, da gleichzeitig eine Vielzahl von legitimen Webseiten gesperrt würde. DNS-“Overblocking“ unterscheidet sich jedoch vom IP-Overblocking insofern, dass keine Web-Seite mit anderen Domain-Namen auf demselben Server geblockt werden. Hinzu kommt, dass das DNS-System komplett umgangen werden kann, wenn die IP-Adresse des Zielsystems bereits bekannt ist.

Inhaltliche Filterung auf höheren Protokollschichten: URL-Blockaden arbeiten äußerst präzise, da sie nicht nur ganze Web-Seiten filtern, sondern es auch ermöglichen, spezifische Inhalte einer Web-Seite zu sperren. Über zwei Geräte wird diese Funktionalität bereitgestellt: *Web-Proxies* und *Deep Packet Level Firewalls*. Dabei erkennen

[4] http://www.zeit.de/digital/internet/2010-03/google-china-zensur

Web-Proxies das stets verwendete Protokoll HTTP, führen einen Abgleich der transportierten URL mit der Filterliste durch und greifen gegebenenfalls in die Kommunikation ein. DPI-basierte Verfahren können noch einen Schritt weitergehen und untersuchen neben Paket-Headern und Protokollen die Nutzlast von Datenpaketen. Aufgrund definierter Signaturen können so aktive Inhalte wie JavaScript und ActiveX erkannt und analysiert werden [Var10].

Löschung zensierter Web-Inhalte Letztendlich setzen die bis dato beschriebenen Techniken auf die Einsicht der Internet-Nutzer bzw. auf Sperrung der zensierten Web-Inhalte. Diese Techniken sind nur dann erfolgreich, wenn sie in einer geschlossenen Infrastruktur eingesetzt werden. Entstehen Lücken in diesem geschlossenen Netz, können zensierte Web-Inhalte wieder erreicht werden. Somit bleibt dann nur das Löschen dieser Inhalte auf den Web-Servern. Diese Überlegungen wurden auch in Deutschland getätigt. Mit dem „Zugangserschwerungsgesetz“ aus dem Jahre 2009 wurde auf Blockade gesetzt. Das Gesetz wurde im Dezember 2012 endgültig aufgehoben. Die Bundesregierung ist der Ansicht, dass die „Verbannung“ von strafbaren Inhalten umgangen werden kann und somit nicht im Interesse der Opfer sei[5]. Hohe Erfolgsquoten bei der Löschung der Web-Angebote bestätigen die Vorhaben der Bundesregierung[6][7].

4.6.2.2 Web-Filterung auf nationaler Ebene

Da neben dem Interesse durch Nutzer und Unternehmen zusätzlich staatliche Organe ein übergeordnetes Interesse an Web-Filtertechniken besitzen, werden die beschriebenen Techniken auch auf nationaler Ebenen eingesetzt (z. B. in Indien[8]). Abhängig von der jeweiligen Regierung werden Web-Filtertechniken in den Bereichen Politik, Sicherheit, Internet-Tools (z. B. E-Mail, Web-Suche, VoIP) und Soziales (z. B. Sexualität, Glücksspiel)[Ini11] praktiziert. Abbildung 4.12 stellt die Verteilung der Länder mit Web-Filtern auf sozialer Ebene dar.

Demnach werden diese Filtertechniken im Wesentlichen in Asien und im Nahen Osten

[5] http://www.heise.de/newsticker/meldung/Bundestag-beerdigt-Websperren-Gesetz-1388728.html
[6] http://www.heise.de/newsticker/meldung/eco-Kampf-gegen-Kinderpornografie-im-Web-erfolgreich-1171092.html
[7] http://blog.die-linke.de/digitalelinke/bka-zahlen-bestatigen-erfolg-von-loschen-statt-sperren/
[8] http://www.heise.de/newsticker/meldung/Indien-installiert-System-zur-umfangreichen-Onlineueberwachung-1858738.html

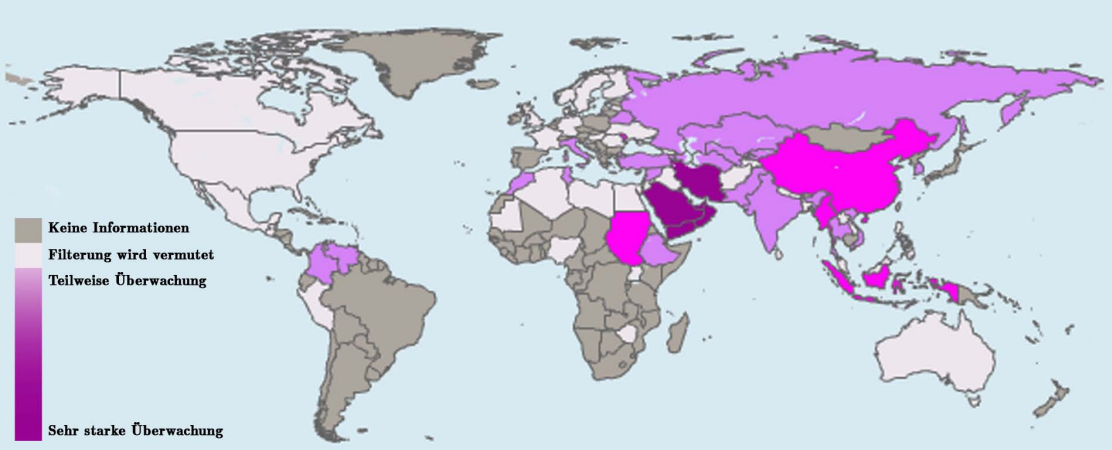


Abbildung 4.12: Länder, die Web-Filter-Techniken auf sozialer Ebene einsetzen [Ini11]

eingesetzt. Nachdem Anfang Juni 2013 die National Security Agency (NSA)-Filtertechniken durch Edward Snowden veröffentlicht wurden, dürfte die USA ebenfalls zu Ländern mit „Sehr starker Überwachung“ zählen. Weitere Übersichten zu Ländern, die Web-Filtertechniken auf anderer als auf sozialer Ebene einsetzen, sind unter [Ini11] zu finden.

4.6.2.3 Software-Lösungen

Rein Software-basierte Web-Filter-Lösungen eignen sich besonders gut im privaten Umfeld, sind aber auch für kleinere Unternehmen geeignet. Häufig bieten ISP wie die Deutsche Telekom und Kabel Deutschland Sicherheitspakete mit Web-Filter-Lösungen bereits bei Abschluss eines Internet-Vertrages an. Weiterhin herrscht auf dem Konsumermarkt eine sehr große Produktvielfalt.

In beiden Fällen ist der Nutzer in der Pflicht und muss die Software selbständig installieren, konfigurieren und warten. Nicht selten sind besonders durchschnittliche Internet-Nutzer mit diesen Aufgaben überfordert. Hinzu kommt, dass sich umfangreiche Filterlisten bei den relativ geringen Internet-Datenraten für Nutzer und Kleinunternehmen negative auf die QoE auswirken.

Gegenüber diesen Lösungen unterscheidet sich der SecAN-Web-Filter bereits durch seinen Einsatzort im TZN. Weiterhin sind im Hinblick auf hochbitratige Datenströme, wie sie in Teilnehmerzugangsnetzwerken vorherrschen, Software-Lösungen nicht praktikabel. Ferner ist der SecAN-Web-Filter für den Endanwender vollkommen wartungsfrei und er-

fordert nicht wie die Lösung [WWLZ13] zusätzliche Browser-Anpassungen. Nachteilig bei [WWLZ13] ist, dass Anfragen nach Web-Ressourcen durch das TZN geroutet und von Serverfarmen abgeglichen werden müssen.

4.6.2.4 Soft- und Hardware-Lösungen

Neben reinen Software-Produkten bieten Unternehmen wie Websense [Web11], Blue Coat [Blu11] und Barracuda Networks[Net] hybride Web-Filter-Lösungen an. Diese Produkte besitzen eine umfangreiche Leistungsvielfalt wie bspw. Spam- und Viren-Erkennung sowie Cloud-Unterstützung während der Filterung. Aufgrund der hohen Anschaffungskosten von mehreren 10.000,00 € (hinzu kommen Wartungskosten in gleicher Höhe) [Net12] eignen sich diese Produkte nicht für die vom Autor gewählte Zielgruppe.

Angesichts der Datenraten von bis zu 300 Mbit/s [AG12] und der versprochenen Funktionalität sind die beschriebenen Web-Filter [Web11, Blu11, Net] für den Einsatz in größeren Unternehmen und Konzernen durchaus geeignet. Sie sind jedoch nicht ausreichend, um den Anforderungen für den Einsatz im TZN gerecht zu werden. Da aufgrund der vorherrschenden Datenraten im TZN mehrere kommerzielle Web-Filter am DSLAM eingesetzt werden müssten, ist der Kostenfaktor das ausschlaggebende Kriterium. Dennoch ist bekannt, dass auch ISP in ihren TZN Filtermechanismen verwenden (vgl. Abbildung 4.12). Obwohl aus sicherheitstechnischen Gründen grundsätzlich sehr wenig Informationen in Bezug auf interne Struktur und Wirkungsweise der Web-Filtersysteme bekanntgegeben werden, konnten einige Eigenschaften durch Pressemitteilungen und Experimente in Erfahrung gebracht werden.

Beispielsweise hat BT – als einer der größten britischen Provider – im Jahre 2003 ein Sicherheitssystem namens „Cleanfeed“[9] entworfen. Ziel von Cleanfeed ist es, den Zugriff auf kinderpornografische Bilder, welche in der Internet Watch Foundation (IWF)-Datenbank gelistet sind, zu verhindern [Cla05]. Die IWF ist eine gemeinnützige Gesellschaft, die sich das Ziel gesetzt hat, die verfügbaren illegalen Inhalte zu minimieren. Im Speziellen sind strafrechtlich relevante Bilder – wie Kindesmissbrauch und andere obszöne Inhalte – gemeint.

Nach [Cla05] wird Cleanfeed seit 2004 von BT eingesetzt. Clayton beschreibt weiter, dass Cleanfeed zweistufig ausgelegt worden sei. In der ersten Instanz werden die Portnummer und Ziel-IP des Pakets untersucht. Erachtet die erste Stufe die Daten als unbedenklich,

[9] Der offizielle Name des Projektes lautetet „BT Anti-Child-Abuse Initiative“.

werden sie an das Zielsystem weitergeleitet. Anderenfalls erfolgt die Weiterleitung in die zweite Stufe - zu einem Web-Proxy. Er unterstützt HTTP-Anfragen und sendet die IP-Adresse und URL an die externe IWF-Datenbank. Wird ein Eintrag in der IWF-Datenbank ermittelt, erhält der anfragende Client eine Fehlernachricht. Anderenfalls wird die HTTP-Anfrage an das Ziel weitergeleitet. Die Struktur des Systems wirkt sich negativ auf dessen Performance aus. Insbesondere durch die zweite Stufe leidet es unter hohen Latenzen, sodass die QoE des Nutzers stark eingeschränkt wird. Der SecAN-Web-Filter besitzt ebenfalls eine mehrstufige *search engine*, verhindert jedoch durch die ausschließliche Nutzung lokaler Ressourcen die bei Cleanfeed auftretenden hohen Latenzen.

Unter dem Begriff „Great Firewall of China“ (vgl. [CW10]) ist ein weiteres Filtersystem, welches einen Web-Filter verwendet, bekannt geworden. Durch Experimente erlangte [CMW06] die Erkenntnis, dass das chinesische Internet-Filtersystem vermutlich eine DPI zur Auffindung spezifischer Schlüsselwörter nutzt. Die Erkennung der Schlüsselwörter erfolgt bidirektional, während die Ethernet-Frames das System passieren.

Wird eines der Schlüsselwörter erkannt, wird die Kommunikationsverbindung zurückgesetzt. Dies geschieht durch jeweils 3 Ethernet-Frames, die an beide Kommunikationspartner gesendet werden und bei denen das „Reset“-Flag im TCP-Header gesetzt wurde (vgl. [CW10]). Jeder der 3 Ethernet-Frame besitzt dabei eine andere Sequenznummer. Ausgehend von der initialen Sequenznummer sind die weiteren Sequenznummern um Vielfache der maximalen Paketgröße erweitert. Durch diesen Mechanismus soll sichergestellt werden, dass beide Kommunikationspartner das Reset akzeptieren. Wie Watson [Wat04] nachweisen konnte, muss die Sequenznummer des Reset-Pakets nicht exakt mit der erwarteten Sequenznummer übereinstimmen, sondern nur innerhalb eines Bereichs liegen, damit sie als gültige Sequenznummer vom Endsystem akzeptiert wird. Weiterhin zeigten die Experimente, dass trotz Abbruch der Verbindung einige Fragmente der angeforderten Web-Seite geladen werden konnten.

Warum das detektierende System gleich 3 Reset-Pakete sendet anstatt nur ein relevantes, konnte ebenfalls experimentell ermittelt werden. Alle Reset-Pakete besaßen einen TTL-Wert von 47, wohingegen die vom Web-Server gesandten Fragmente der Web-Seite den Kommunikationspartner mit einem TTL-Wert von 39 erreichten. Dies weist darauf hin, dass das Gerät, welches die Reset-Pakete versendet, und der Web-Server unterschiedliche Maschinen sind.

Damit weist das chinesische Filtersystem gleich zwei Schwächen auf. Zum einen liegt die

Kontrolle über den Verbindungsabbruch bei den Kommunikationspartnern. Ignorieren beide die Reset-Pakete, bleibt die Verbindung aufrechterhalten. Zum anderen muss das System, das Reset-Pakete versendet, den richtigen Sequenzbereich treffen. Geschieht dies nicht, bleibt die Verbindung ebenfalls etabliert. Der SecAN-Web-Filter weist keine dieser Schwächen auf, da die Kontrolle unabhängig von weiteren Komponenten ist (vgl. 4.6.3).

4.6.2.5 Fazit

Internet-Kriminalität besitzt viele Facetten, wie bspw. die Veröffentlichung kinderpornografischer und gewaltverherrlichender Inhalte. Ein wirkungsvolles Vorgehen gegen solche Web-Inhalte wird durch Blockaden und Löschungen erreicht. Effektiv sind Blockaden, die auf IP-Ebene (IP-Blockade und DNS-Poisoning) arbeiten. Sind über die blockierte IP-Adresse mehrere Domains erreichbar, führen sie zu einem „Overblocking“, sodass nichtkompromittierte Web-Inhalte ebenfalls nicht mehr erreichbar sind. Beim DNS-Poisoning kommt hinzu, dass es einfach umgangen werden kann, wenn die IP-Adresse des Ziels bereits bekannt ist.

Leistungsfähiger als IP-Blockaden ist die inhaltliche Filterung auf höheren Protokollschichten. Hier werden angefragte URLs mit Filterlisten verglichen und bei einer Übereinstimmung blockiert. Die URL-Filterung kann äußerst präzise eingesetzt werden, benötigt aber deutlich mehr Ressourcen als die IP-basierte Filterung. Ein deutlicher Vorteil gegenüber IP-Blockaden ist, dass „Overblocking“ entfällt. Allerdings kann die URL-Filterung zu „Underblocking“ führen. Das bedeutet, dass ähnliche Inhalte innerhalb derselben Domain, die nicht bekannt sind, nicht blockiert werden. Im Hinblick auf dynamisch generierte Web-Inhalte mit ständig aktualisierten Ressourcennamen wird die URL-Filterung sogar ad absurdum geführt.

Somit bieten die dem Autor bekannten und aufgeführten Verfahren keinen hundertprozentigen Schutz. Aufgrund systembedingter Schwächen lassen sich einige Verfahren sogar umgehen. Obwohl dies für die gewählte Zielgruppe nicht von Relevanz ist, sollte ein Sicherheitssystem nicht umgangen werden können.

Anmerkung: Nach Meinung des Autors muss das Ziel die Löschung von kinderpornografischen und gewaltverherrlichenden Web-Inhalte sein. Bis zur endgültigen Löschung müssen diese Web-Inhalte wirkungsvoll blockiert werden. Demnach sind Verfahren, die zu „Underblocking“ führen, für den SecAN-Web-Filter nicht akzeptabel.

4.6.3 Konzipierung

Der Vollständigkeit halber wurden in Abschnitt 4.6.2.1 die Manipulation von Suchmaschinenergebnissen und das Löschen zensierter Web-Inhalte aufgeführt. Beide Techniken lassen sich im TZN nicht umsetzen, da kein Zugriff auf die Endkomponenten besteht. Eine reine Filterung auf IP-Basis wird bereits durch eine Filterstufe der Firewall etabliert und braucht dementsprechend auch nicht näher betrachtet werden. Die verbleibenden Verfahren, die für den SecAN-Web-Filter als geeignet eingestuft werden, sollen im Hinblick auf folgende Punkte näher untersucht werden:

- Das Verfahren darf nicht unter „Underblocking“ leiden.
- Die Filterwirkung muss unmittelbar eintreten.
- Das einzusetzende Verfahren muss mit den Anforderungen aus Abschnitt 3.1.1 konform gehen. Im Mittelpunkt soll die ressourcenschonende Umsetzung bei hohen Verarbeitungsgeschwindigkeiten stehen.

4.6.3.1 DNS vs. URL

Typischerweise wird ein Nutzer seinen Browser verwenden, um eine Web-Seite zu laden. Vor dem Nutzer verborgen werden im Hintergrund zwei Schritte abgearbeitet, die im Folgenden analysiert werden.

In der ersten Phase muss zu der eingegebenen URL die IP-Adresse ermittelt werden. Dazu stellt der Browser eine Anfrage an den lokalen „Resolver“. Der „Resolver“ seinerseits schickt eine DNS-Anfrage an einen bekannten DNS-Server. Bei erfolgreicher Namensauflösung erhält der „Resolver“ die dazugehörende IP-Adresse zurück – im Fehlerfall eine entsprechende Fehlermeldung.

In der zweiten Phase versucht der Browser eine Verbindung mit dem Server aufzubauen. Konnte eine TCP-Verbindung hergestellt werden, schickt der Browser eine HTTP-GET-Anfrage an den Web-Server. Dieser gleicht die angefragte URL mit seinen Ressourcen ab. Existiert eine Übereinstimmung, hüllt der Web-Server die Ressource in ein HTTP-Entity ein und sendet dieses an den anfragenden Client. Anschließend übernimmt der Browser die Darstellung der empfangenen Ressource.

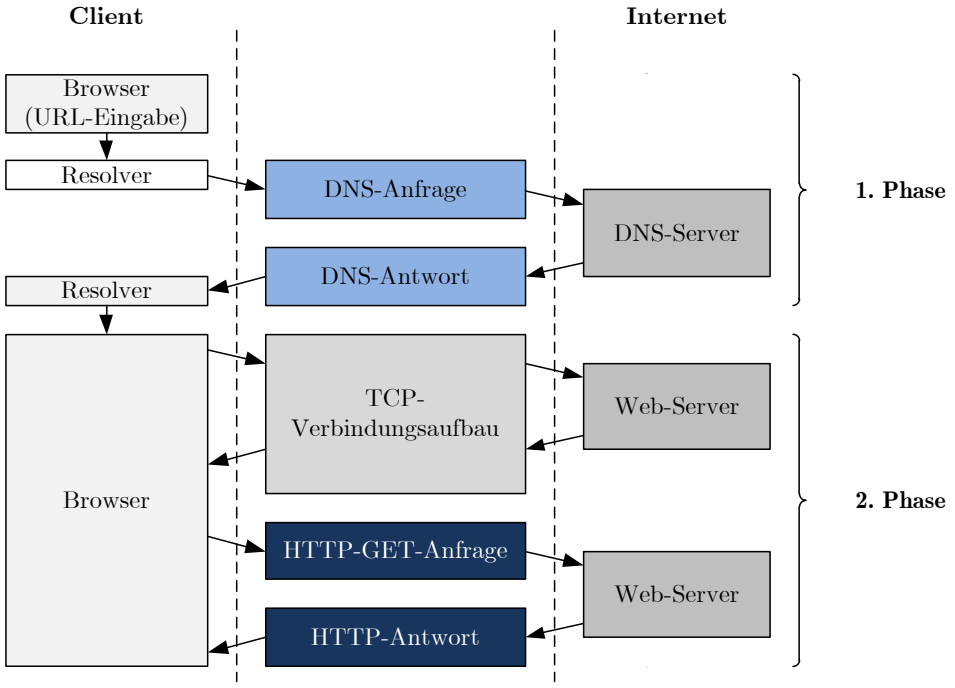


Abbildung 4.13: Laden einer Web-Ressource

Die beschriebene Ereigniskette ist schematisch in Abbildung 4.13 dargestellt. Sollte die Kette unterbrochen werden, kann die angefragte Ressource vom Browser nicht dargestellt werden. Somit kann ein Web-Filter bei der Namensauflösung (Phase 1) bzw. bei der HTTP-Anfrage oder HTTP-Antwort (Phase 2) eingreifen. Beide Unterbrechungsstellen haben ihre Vor- und Nachteile. Diese werden nachfolgend gegenübergestellt.

Vorteile der Filterung auf DNS-Ebene

- **Geordnete Protokoll-Strukturen:** Hardware-basierte Web-Filter können ressourcenschonend umgesetzt werden.
- **Multi-Protokoll-Sperrung:** Alle Protokolle, denen eine Namensauflösung vorausgeht, lassen sich sperren. Dazu zählen z. B. HTTP, FTP und Secure Shell (SSH).

Nachteile der Filterung auf DNS-Ebene

- **Umgehung des DNS:** Ist die IP-Adresse des Zielsystems z. B. durch die lokale

HOSTS-Datei bereits bekannt, wird das DNS nicht angefragt, da die Domain-Auflösung bereits lokal erfolgte.

- **Keine sofortige Sperrung:** Wie in Abschnitt 2.3.2 beschrieben, besitzt jedes A-RR einen TTL-Wert, der mehr als 100 Jahre umfassen kann. Da während dieses Zeitraums die Zuordnung Domainname-IP-Adresse gültig ist, wird keine neue DNS-Anfrage durchgeführt. Somit bleibt die angefragte Ressource erreichbar und wird erst nach Ablauf des TTL-Wertes gesperrt, wenn dieselbe Ressource erneut angefragt wird.
- **Keine DNS-Nutzung bei PROXY-Verbindungen:** Ein auf DNS-Basis arbeitender Web-Filter, der sich zwischen Client und HTTP-Proxy befindet, ist wirkungslos, weil der Client lediglich die IP-Adresse seines Proxy-Servers kennen muss. Der Client sendet keine DNS-Anfragen, sondern nur HTTP-GET-Anfragen. Die Aufgabe der Namensauflösung übernimmt sein HTTP-Proxy-Server, der sich, vom Client aus gesehen, hinter dem Web-Filter befindet.
- **Overblocking:** Typischerweise nutzen Web-Hoster IP-Sharing, um mehrere Domains auf einem Web-Server unterzubringen. Ist die IP-Adresse eines Web-Servers gesperrt, findet in der Regel „Overblocking“ statt, weil alle Domains des Web-Servers blockiert werden.

Vorteile der Filterung auf HTTP-Ebene

- **Keine Web-Ressource ohne HTTP-Anfrage:** Um eine Web-Ressource über HTTP zu laden, muss eine HTTP-GET-Anfrage gestellt werden.
- **Sofortige Sperrung:** Erkennt der Web-Filter, dass die angefragte Domain als bedrohlich eingestuft wurde, wird die Anfrage nicht an den Web-Server weitergeleitet. Somit wird dem Client der gesamte Inhalt der angefragten Web-Seite verborgen bleiben.
- **Keine Umgehung des Web-Filters durch HTTP-Proxy-Nutzung:** Sendet ein Nutzer eine HTTP-Anfrage über einen HTTP-Proxy, dann ist die angeforderte Domain sowohl zwischen Nutzer und HTTP-Proxy als auch zwischen HTTP-Proxy und Web-Server überprüfbar. Eine Umgehung des HTTP-Web-Filters wie bei einem DNS-Web-Filter ist somit nicht möglich.

Nachteile der Filterung auf HTTP-Ebene

- **Ungleichmäßig geordnete Strukturen:** Da die meisten HTTP-GET-Anfragen in relativer Form erfolgen, muss nach dem HOST-Header gesucht werden, um die Domain zu finden. Der HOST-Header befindet sich hinter der „Start-Line“, welche die relative URL beinhaltet. An welcher Stelle das ist, kann aufgrund fehlender Längenangaben nicht errechnet werden. Hinzu kommt, dass mehrere HOST-Header in einer Anfrage vorhanden sein können (vgl. Abbildung 2.12). Aus diesem Grund muss nach dem passenden HOST-Header „geparst“ werden.
- **Sperrung ausschließlich vom HTTP-Netzwerkverkehr:** Anders als bei einer DNS-Filterung wird bei einer HTTP-Sperre lediglich der HTTP-Netzwerkverkehr gefiltert und gegebenenfalls blockiert.

Auswertung der Möglichkeiten der Web-Filterung

Um die Erreichbarkeit von Web-Inhalten zu unterdrücken, existieren im Wesentlichen zwei Möglichkeiten: Erstens, die Domain-Namensauflösung über DNS-Anfragen wird unterbunden bzw. zweitens HTTP-GET-Anfragen/-Antworten werden blockiert. Obwohl durch eine Blockade auf DNS-Basis der nachfolgende Datenverkehr entfällt und somit Netzwerkverkehr eingespart werden würde, wiegen die Nachteile einer DNS-Filterung sehr schwer. Aufgrund des TTL-Wertes innerhalb der DNS-Antwort arbeiten DNS-Filter mit einer Verzögerung. Genauso schwerwiegend ist zum anderen, dass die Verwendung von HTTP-Proxy-Servern eine DNS-Filterung ad absurdum führt. Weiterhin kann es sein, dass gar keine DNS-Anfragen gestellt werden, wenn die IP-Adresse bereits bekannt ist (z. B. per E-Mail zugesandt wurde).

Demgegenüber steht die Blockade der HTTP-GET-Anfragen/-Antworten. Die unregelmäßig geordneten Strukturen einer HTTP-Nachricht erschweren das Auffinden der gesuchten Informationen gegenüber einer DNS-Nachricht. Damit ist dieser Ansatz für Hardware-basierte Web-Filter weniger gut geeignet. Jedoch sind HTTP-Nachrichten zwingend notwendig, um an die Inhalte einer Web-Seite zu gelangen. Da diese Anfragen nicht umgangen werden können, wird die HTTP-Filterung als die geeignete Art der Web-Filterung eingestuft. Ein weiterer positiver Nebeneffekt ist die unmittelbar einsetzende Filterwirkung. Damit verspricht HTTP-Filterung einen deutlich höheren Wirkungsgrad als eine DNS-Filterung. Hinzu kommt, dass bei einem HTTP-basierten Web-Filter die gesamte URL (Domain und relativer Pfad der Ressource) sowie die per DNS angeforderte IP-Adresse des Web-Servers zur Filterung herangezogen werden können. Somit überwiegen die Vorteile der HTTP-Filterung gegenüber ihren Nachteilen und einer Filterung auf DNS-Basis.

Nachfolgend wird ein Web-Filter entwickelt, der auf HTTP-Ebene fungiert. Eine hohe Dynamik der URLs sowie die Tatsache, dass sie keine definierte Länge besitzen, führen aus Hardware-technischer Sicht zu einem sehr aufwendigen Parsen. Dadurch können hohe Latenzen in der Verarbeitung entstehen. Aus Gründen der Hardware-freundlichen Verarbeitung wird deshalb die Domain, die eine maximale Länge von 255 Byte besitzen kann (vgl. Abschnitt 2.3.2), „geparst“. Durch diese Methode der Filterung entsteht ein „Overblocking“ bezüglich aller Ressourcen hinter einer Domain – nicht hinter einer IP. Somit wird inhärent ein „Underblocking“ vermieden.

4.6.3.2 Systembeschreibung

Der Web-Filter wird im Upstream aus Nutzersicht etabliert werden. Dort wird der Datenstrom auf HTTP-GET-Anfragen untersucht. HTTP-GET-Anfragen enthalten als Fragment der URL die Domain (vgl. Abbildung 2.12), die es zu erkennen gilt. Bei einer Blockade der HTTP-GET-Anfrage setzt die Filterwirkung unmittelbar ein, da bereits die Anfrage nach einer entsprechenden Ressource unterbunden wird. Abhängig von der Häufigkeit auftretender HTTP-GET-Anfragen und der Menge der Speichereinträge kann der Domain-Abgleich mit den Blacklist-Einträgen zur zeitkritischen Komponente werden.

Wie die Praxis aus Großbritannien zeigt, enthält die IWF-Filterliste zwischen 500 und 800 Einträgen[10]. Auch die Erfahrungen des Bundeskriminalamtes weisen keinen signifikanten Unterschied auf und fallen mit 143 Einträgen sogar geringer aus[11]. Auf dieser Basis aufbauend wird der Web-Filter mit 4096 Einträgen deutlich größer dimensioniert und damit zukunftssicher ausgelegt. Ferner erlaubt die Größe der Blacklist, dass neben den Domains auch die zugehörigen IP-Adressen abgelegt werden und so eine Umgehung des Systems durch Nutzung der IP-Adresse statt des Domain-Namens nicht möglich ist.

4.6.4 Realisierung

Wie in Abbildung 4.14 dargestellt, besteht das Web-Filtersystem aus den 3 Modulen *Packet Classification Engine*, *Web-Filter* und dem *Configurator*. Der Web-Filter seinerseits ist ebenfalls in drei Module unterteilt. Im Web-Filterkern befinden sich der HTTP-

[10] http://www.iwf.org.uk/services/blocking/blocking-faqs
[11] http://blog.die-linke.de/digitalelinke/bka-zahlen-bestatigen-erfolg-von-loschen-statt-sperren/

Parser und der „Hash-Lookup-Speicher“. Hinzu kommt der „Domain-Blacklist-Speicher“, der sich ebenfalls im Web-Filter befindet.

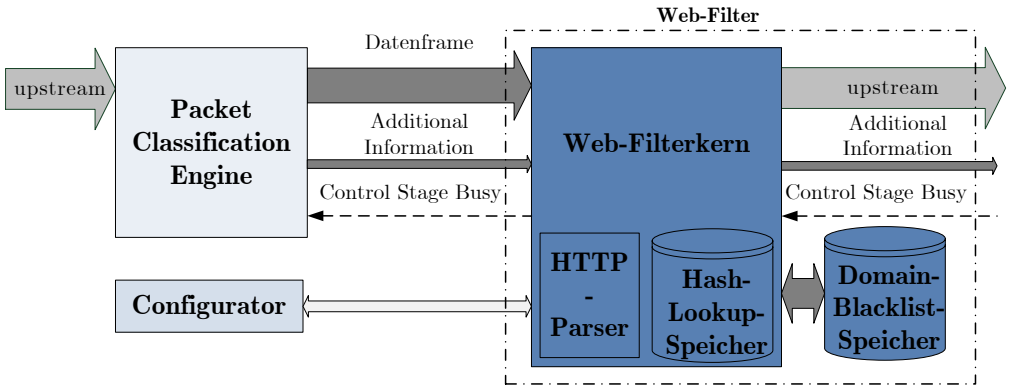


Abbildung 4.14: Blockschaltbild des Web-Filtersystems

Über verschiedene Interfaces kann der Web-Filter mit den umliegenden Modulen kommunizieren. Der Kommunikationsablauf gestaltet sich wie folgt: Nachdem die beiden internen Speicher des Web-Filters mit unkomprimierten Domains bzw. deren Hash-Werten konfiguriert wurden, beginnt die Datenkommunikation im Web-Filtersystem. Die *PCE* des Web-Filtersystems liefert im Frame-Parameterset zwei Zusatzinformationen zum aktuellen Frame aus, welche helfen, das Web-Filtersystem ressourcenschonend zu entwickeln. Dabei handelt es sich um folgende Werte:

- HTTP-GET-Flag: Zeigt an, ob im aktuellen Daten-Frame eine HTTP-GET-Anfrage enthalten ist.
- TCP-Payload-Start: Zeigt an, ab welcher Position im aktuellen Daten-Frame die TCP-Payload beginnt.

Daten-Frame und Zusatzinformationen erreichen gleichzeitig den Web-Filter und werden vorübergehend in separate FIFO-Puffer im Web-Filterkern gespeichert. Die Zwischenspeicherung ist notwendig, da die Überprüfung einer Domain nicht „on the fly“, sondern mit einer geringen Verzögerung erfolgt.

Der Verarbeitungsflow innerhalb des Web-Filterkerns ist in Abbildung 4.15 verdeutlicht. Bereits beim Eintreffen erkennt der HTTP-Parser durch das HTTP-GET-Flag, ob es sich um eine HTTP-GET-Anfrage handelt. Ist dies nicht der Fall, wird der Daten-Frame unverzüglich weitergeleitet. Anderenfalls beginnt das Parsen und Hashen der Domain aus dem aktuellen Datenstrom. Diese Aufgabe übernimmt ebenfalls der HTTP-Parser.

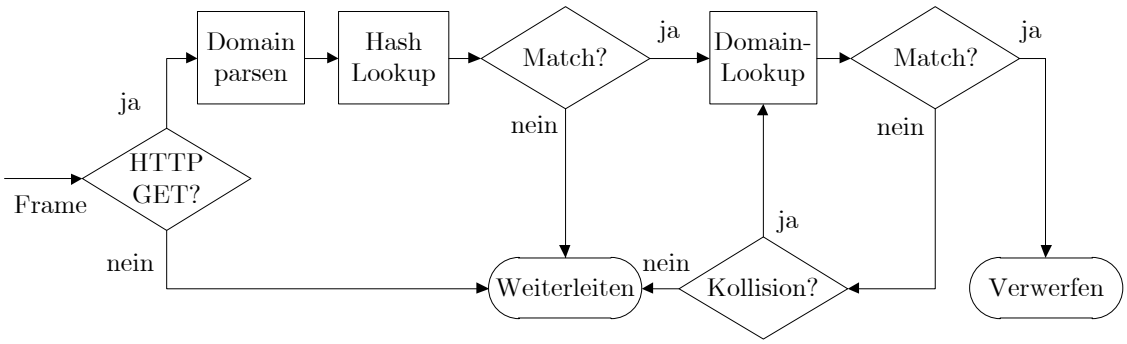


Abbildung 4.15: Datenfluss innerhalb des Web-Filters

Die Suche in den Speichern erfolgt ähnlich dem britischen „Cleanfeed“ in 2 Stufen. Zunächst wird im „Hash-Lookup-Speicher“ gesucht und anschließend im „Domain-Blacklist-Speicher“. Um die bei „Cleanfeed“ anfallenden hohen Latenzen zu kompensieren, erfolgt die Suche ausschließlich in lokalen Ressourcen.

In der 1. Stufe wird der aktuell errechnete Hash-Wert der Domain mit zuvor konfigurierten Hash-Werten von zu blockierenden Domains im „Hash-Lookup-Speicher“ verglichen. Bei einem Treffer im „Hash-Lookup-Speicher“ muss eine Verifikation der Domain aus dem aktuellen Datenstrom mit den unkomprimierten Domains im „Domain-Blacklist-Speicher“ durchgeführt werden. In der 1. Stufe können 3 Fälle eintreten:

1. Wird der errechnete Hash-Wert nicht im „Hash-Lookup-Speicher“ gefunden, handelt es sich um eine als „unbedenklich“ eingestufte Domain und der Datenstrom wird weitergeleitet.

2. Wird der errechnete Hash-Wert im „Hash-Lookup-Speicher“ gefunden, kann es sich um eine zu blockierende Domain handeln. Diese muss im „Domain-Blacklist-Speicher“ verifiziert werden.

3. Haben zwei zu blockierende Domains denselben Hash-Wert, muss eine Kollisionsauflösung im „Domain-Blacklist-Speicher“ erfolgen.

Im Anhang C.5 werden weiterführende Details zu den Funktionalitäten des HTTP-Parsers, der „Hash-Lookup-Phase“ und der „Domain-Lookup-Phase“ sowie deren Umsetzungen vorgestellt.

4.6.5 Test und Ergebnisse

Bereits vor dem Test der Hardware existieren aufgrund der Vorüberlegungen und Untersuchungen zum Web-Filter wichtige Ergebnisse. So wurden die Domain-Längen von 100.821.035 VeriSign-registrierten Domains untersucht. Sie wiesen eine durchschnittliche Länge von 18 Byte auf (vgl. Abbildung C.11).

Aufgrund der guten Kompressionseigenschaften und einer Hash-Wert-Länge kleiner 18 Byte erwies sich das CRC64-Verfahren für die erste Suchphase als optimal geeignet. Die zweite Suchphase ist optional und wird lediglich zur Verifikation gestartet, wenn die erste Suchphase einen Treffer ergab. Ähnlich dem Verfahren „Hashing durch Verkettung“ (vgl. Abschnitt A.5.3) wird für die 2. Suchphase eine indizierte Suche durchgeführt, wenn mehrere Domains auf denselben Hash-Wert abgebildet wurden.

CRC64-Tests mit 23 Millionen VeriSign-Domains führten zu insgesamt 159 Kollisionen, wobei lediglich jeweils zwei Domains miteinander kollidierten. Das entspricht einer Kollisionswahrscheinlichkeit in der ersten Suchphase $P = \frac{159}{23.000.000} = 6,91 \cdot 10^{-6}$. Aufgrund dieser Eigenschaft ist die Suche in den meisten Fällen nach einer Bucket-Anfrage, aber spätestens nach zwei Bucket-Anfragen beendet.

4.6.5.1 Simulativer Test der Web-Filterstufe

Der funktionale Nachweis für das Web-Filtersystem erfolgte nach demselben Herangehen wie bereits in Abschnitt 4.5.5 beschrieben. Alle Subsysteme wurden ausführlich getestet. Aus der Konzeptphase konnte ein Entscheidungsdiagramm mit sämtlichen Entscheidungen und Kommunikationspfaden erzeugt werden. Auf dieser Basis wurde ein Testkatalog erstellt. Die Testfälle sorgen für eine hundertprozentige Pfadabdeckung.

Tabelle 4.11 stellt für die Web-Filterstufe die Anzahlen aller Entscheidungen und Kommunikationspfade dar. Teilweise entstehen Zyklen in den Kommunikationspfaden, die erst nach mehrmaligem Durchlaufen beendet werden. Jeder dieser Kommunikationspfade stellt einen funktionalen Testfall dar.

Filterstufe	Entscheidungen	Kommunikationspfade
Web-Filter	202	312

Tabelle 4.11: Web-Filterstufe und funktionale Testfälle

Für jede Entscheidung wurde je ein *Good Case*- und ein *False Case*-Testfall entwickelt. Diese belaufen sich auf 2 x 202 = 404 Testfälle. Zusätzlich wurden für eine vollständige Pfadabdeckung 312 Testfälle entwickelt. Um eine vollständig korrekte Funktionalität des SecAN-Web-Filters nachzuweisen, wurden somit 716 Testfälle mit einer Frame-Länge von 590 Byte entwickelt und getestet.

4.6.5.2 Berechnung der maximalen Last

Das SecAN-Web-Filtersystem arbeitet ohne *packet loss*, wenn die interne Verzögerung (T_{int}) $\leq$ der Verarbeitungszeit der externen Daten (T_{ext}) ist. Dabei entsteht der größte Stress für das System, wenn Ethernet-Frames mit minimaler Länge auf das System treffen. Sie besitzen folgenden Aufbau:

- 12 Byte Inter Frame Gap
- 8 Byte Präambel & Start Frame Delimiter
- 14 Byte Ethernet-Header
- 20 Byte IP-Header
- 20 Byte TCP-Header
- 28 Byte HTTP-Anfragezeile & Host Header
- 4 Byte FCS

Somit hat ein minimal langer Ethernet-Frame für das Web-Filtersystem eine Länge von 106 Byte. Die Dauer eines externen Taktes errechnet sich aus der externen Datenrate (1 Gbit/s) und der externen Frequenz (125 MHz) und beträgt demnach $\frac{1}{125\ MHz} = 8\ ns$. Dementsprechend ist T_{ext} = 106 Byte * 8 ns = 848 ns.

Systemintern existieren *Interframe Gap*, *Präambel*, *Start Frame Delimiter* und *FCS* nicht. Deshalb setzt sich ein Datenblock im FPGA aus den restlichen 82 Byte zusammen. Aufgrund der internen Verarbeitungsbreite (dpt) wird der Datenblock zu einem ganzzahligen Vielfachen von 4 Byte aufgefüllt und beträgt somit 84 Byte. Die 84 Byte werden in 21 Systemtakten verarbeitet.

Die Verzögerungen durch beide Lookup-Phasen belaufen sich auf 82 Takte, sodass sich die interne Verarbeitungszeit wie folgt zusammensetzt:

- 21 Takte, innerhalb dieser Zeit werden die 84 Byte interne Daten verarbeitet

- 12 Takte durch die Hash-Lookup-Phase
- 93 Takte durch den DDR2-Controller, die vollständige zweistufige Suche im Speicher und einer anteiligen Refresh-Zeit für den DDR2-Speicher

Zu bestimmen ist nun die interne *cycletime* CT_{int}, durch die auf die gesamte interne Verarbeitungszeit und die minimal einzuhaltende Frequenz $f_{min_{int}}$ zurückgeschlossen werden kann. In Formel 4.2 werden beide Werte berechnet.

$$\begin{aligned} 126\ Takte \cdot CT_{int} &\leq 848\ ns \\ CT_{int} &\leq \frac{848\ ns}{126\ Takte} \\ CT_{int} &\cong 6,73\ ns \end{aligned} \tag{4.2}$$

Die interne *cycletime* CT_{int} entspricht einer minimal einzuhaltenden internen Frequenz von $f_{min_{int}} = \frac{1}{8,23\ ns} = 148,58$ MHz.

Anmerkung: Wenn auch der *Worst Case*-Fall, so wie er beschrieben und berechnet wurde, theoretisch denkbar ist, so ist er dennoch äußerst unwahrscheinlich. So müssten zwei nahezu identische Domains, die denselben CRC64 Hash-Wert aufweisen, sich lediglich in den letzten Byte unterscheiden.

4.6.5.3 Hardware-Test des Web-Filters

Das Web-Filtersystem wurde zunächst simulativ in Modelsim mit den definierten Testfällen getestet. Anschließend erfolgten weitere Tests auf dem Entwicklungsboard (vgl. Abschnitt D.1). Um die korrekte Funktionalität des SecAN-Web-Filters nachzuweisen, wurden dieselben Testmuster wie für die Simulation verwendet. Für diese Tests wurde erneut der Traffic-Generator [TDT11] verwendet. Wie aus den Simulationsergebnissen bereits zu erwarten war, wurden alle Testmuster erfolgreich verarbeitet.

4.6.5.4 Ressourcenbedarf und Durchsatz

Nachdem alle Testmuster erfolgreich verarbeitet werden konnten, wies das auf Geschwindigkeit optimierte Web-Filtersystem den in Tabelle 4.12 dargestellten Ressourcenbedarf auf.

Ressourcenbedarf	Genutzt	Verfügbar	Auslastung
Anzahl Slice Register	5003	44.800	11,2 %
Anzahl Slice LUTs	5957	44.800	13,3 %
BlockRam/FIFO	56	296	18,9 %

Tabelle 4.12: Ressourcenbedarf des Web-Filters

Nach *Post Place & Route* beläuft sich die Zielfrequenz des SecAN-Web-Filters auf $125,44$ MHz . Bei einer internen Datenverarbeitungsbreite von $32Bit/Takt$ entspricht das einem Durchsatz im *Worst Case* von $\approx 1,03$ $Gbit/s$. Somit wird der geforderte minimale Durchsatz von 1 Gbit/s eingehalten.

4.6.6 Zusammenfassung

Es existieren eine Reihe bekannter Web-Filtertechniken, wie [Ini11, CW10, Var10] beweisen. Um in Deutschland rechtswidrige Web-Inhalte einzudämmen, setzt die Bundesregierung auf Löschung dieser. Die Zahlen des Bundeskriminalamtes[12] beweisen, dass die meisten zensierten Inhalte nach einigen Wochen gelöscht sind. Um neben allen unbedarften Internet-Nutzern auch den Opfern gerecht zu werden, ist es aus Sicht des Autors zwingend notwendig, die erwähnten Inhalte unmittelbar nach ihrem Erkennen bis zur endgültigen Löschung so zu filtern, dass sie nicht mehr erreichbar sind.

Zu diesem Zweck wurde innerhalb des SecAN-Projektes ein Hardware-Web-Filter entwickelt, der nach aktuellem Wissensstand des Autors nicht umgangen werden kann. Zum einen müssen alle Anfrage, die aus dem Internet Daten anfordern, durch den Access-Bereich geleitet werden. Zum anderen sind HTTP-GET-Anfragen für das Erreichen von Web-Ressourcen wie Bildern und Videos unerlässlich.

Selbst wenn entsprechende Inhalte über Really Simple Syndication (RSS) Feed angefragt werden, wird dies erkannt. In diesem Fall werden lediglich Texte, jedoch keine Bilder, Videos oder dergleichen geladen. Derartige Inhalte müssen per HTTP-GET angefragt werden und werden dementsprechend detektiert. Lediglich verschlüsselte und getunnelte Verbindungen stellen eine Hürde dar, die das System nicht überwinden kann. Grundsätzlich haben alle Systeme, die eine tiefere Datenanalyse als eine Firewall durchführen, dieses Problem.

[12] http://www.spiegel.de/netzwelt/netzpolitik/0,1518,751857,00.html

Das System weist den in Tabelle 4.12 dargestellten Ressourcenverbrauch auf und erfüllt die geforderten Zielvorgaben hinsichtlich des Durchsatzes. Als limitierende Komponente tritt der modifizierte DDR2-Controller, ohne den die erreichte Zielfrequenz nicht möglich gewesen wäre, in Erscheinung. Sind die Speicher bzw. deren Controller nicht länger die limitierenden Komponenten, lässt sich laut *Post Place & Route*-Report die Frequenz bis auf 186, 74 MHz erhöhen. Bei einer Verarbeitungsbreite von 32 Bit/Takt, wie sie derzeit implementiert ist, entspricht dies einem Durchsatz von $\approx 5,975$ $Gbit/s$.

Durch die hohe Performance des Systems ändert sich die QoE für den Nutzer nicht. Hohe Verzögerungen, wie sie bspw. bei *Cleanfeed* [Cla05] der Fall sind, treten nicht auf, da nur lokale Ressourcen und keine externen Datenbanken verwendet werden. Gleichzeitig leidet das System nicht unter *False Positives*, da jede erkannte Domain in der Domain-Lookup-Phase verifiziert und der Datenstrom anschließend verworfen wird. Das bedeutet, dass alle zu sperrenden Web-Inhalte auch tatsächlich nicht mehr erreichbar sind. Die Filterwirkung tritt unmittelbar ein, da bereits die Anfragen nach Web-Ressourcen analysiert und gegebenenfalls unterbunden werden. Dementsprechend werden auch keine Fragmente an den Nutzer ausgeliefert, wie es bspw. beim chinesischen Web-Filtersystem der Fall ist [CMW06].

Mit anderen kommerziellen Produkten wie bspw. von Barracuda [Net], Blue Coat [Blu11], Websense [Web11] und Huawei [Hua] lässt sich der entwickelte Prototyp nur bedingt vergleichen, da diese Lösungen häufig auf eine URL-basierte Filterung setzen. Aus Sicht des Autors wird die URL-Filterung ad absurdum geführt, wenn es sich um dynamisch generierte Web-Inhalte mit ständig aktualisierten Ressourcennamen handelt. Deshalb filtert der SecAN-Web-Filter Domain-Namen, die deutlich länger verfügbar sind als URLs.

Wird der Domain-Name als Teil der URL betrachtet, so können auch kommerzielle Produkte Domain-basiert filtern. Jedoch liegt ihre Filterleistung mit wenigen Mbit/s (Huawei [Hua]) bis maximal 300 Mbit/s („Modell 910“ von Barracuda [AG12]) deutlich unter der Filterleistung des SecAN-Web-Filters mit $\approx 1,03$ $Gbit/s$. Werden zusätzlich die Anschaffungskosten von kommerziellen Produkten wie dem Barracuda Web-Filter „Modell 610“ (9.699,00 €)[Net12] herangezogen, so wird deutlich, dass diese Produkte für die gewählte Zielgruppe ungeeignet sind. Währenddessen können die Kosten für den SecAN-Web-Filter durch bis zu 1000 angeschlossene Endnutzer geteilt werden, wodurch sich das Produkt in kürzester Zeit für den Betreiber amortisiert.

Obwohl als zu blockierende Web-Seiten kinderpornografische und gewaltverherrlichende

Inhalte definiert wurden, muss fairerweise gesagt werden, dass selbstverständlich auch Inhalte außerhalb dieser Zielgruppe geblockt werden können. Wie bereits beschrieben, wird diese Praxis bereits in China angewandt, sodass ein solches System z. B. auch die Pressefreiheit einschränken kann. Jedoch kann den Opfern von im Internet veröffentlichtem Kindesmissbrauch nur mit dem entwickelten Web-Filter geholfen werden. Für den Fall, dass unkooperative Provider der Aufforderung, die Inhalte zu löschen, nicht folgen, bleibt die Domain-Sperrung im Web-Filter als dauerhafte Lösung.

4.7 Intrusion Detection System innerhalb der Packet Processing Engine

4.7.1 Motivation

Alle Netzwerksicherheitssysteme, die über den Header der Transportschicht hinaus den Datenstrom untersuchen, können als Deep-Packet-Inspection-Systeme bezeichnet werden. Somit zählen neben Web-Filtern auch Intrusion Detection-/Intrusion Prevention (ID/IP)-Systeme in diesen Bereich. Speziell wenn die Erkennung von Malware-Signaturen und Angriffsmustern eine wichtige Rolle spielen, zeigt sich die Daseinsberechtigung dieser Systeme. Prinzipiell kann ein ID/IP-System aber auch die Funktionalität einer Firewall sowie Web-Filteraufgaben übernehmen. Jedoch beweist sie ihre Leistungsfähigkeit erst in der unstrukturierten „Payload“ der Transportschicht. Um dort mit einem hohen Durchsatz betrieben werden zu können, benötigt sie häufig mehr oder spezielle Ressourcen als Firewall- und Web-Filterlösungen. Aus den genannten Gründen haben Firewall und Web-Filter ebenfalls ihre Existenzberechtigung. Somit ist es sinnvoll, die SecAN-Sicherheitsarchitektur um ein System zur Erkennung von Eindringlingen, von Malware und dergleichen zu erweitern.

4.7.2 Stand der Technik

Nach [Kap13] wird ein IDS in Funktion und Aufgabe wie folgt definiert:

> Eine *Intrusion* ist ein unerwünschter und/oder nicht-autorisierter Vorgang. Unter dem Begriff *Intrusion Detection* ist die Überwachung von Systemen und /oder Netzwerken zu verstehen, welche das Ziel hat, unerwünschte und/oder nicht-autorisierte Vorgänge zu erkennen, zu protokollieren und zu

melden. Ein *Intrusion-Detection-System* überwacht ein System und/oder Netzwerk und analysiert, ob das beobachtete Verhalten des Systems/Netzwerks vom erwarteten Normalzustand abweicht. Eine solche Abweichung wird als Anomalie bezeichnet.

4.7.2.1 Hardware-basierte Deep-Packet-Inspection

Im Jahr 2008 veröffentlichte Proudfoot et al. [PKAC08] ein flexibles „Software-Hardware Network Intrusion Detection System“. Das System verwendet den „Wu-Manber“-Algorithmus. Die Effizienz des Algorithmus ist abhängig von der Länge der zu suchenden Muster und ist um so geeigneter, je länger das kürzeste Muster ist. Getestet wurden Muster mit insgesamt 93284 Zeichen. Dabei erzielte das in [PKAC08] vorgestellte System einen Durchsatz von 0,139 Gbit/s auf einem Virtex-II FPGA. Der Durchsatz entspricht nicht den Anforderungen des SecAN-Systems, auch ein Musterlängen-abhängiger Algorithmus ist im Hinblick auf Robustheit des IDS ungeeignet.

In [AGGR09] wird ein FPGA-basierter „Simple SNORT Prozessor“ vorgestellt. Der Prozessor ist in der Lage *Byte_test* und *Byte_jump*-Instruktionen auszuführen. SNORT-Regeln erfordern häufig das Anwenden beider Instruktionen (39 % bzw. 45 % aller SNORT Regeln [AGGR09]). Auf einem Virtex-5 FPGA erzielte das System eine Frequenz von 35,448 MHz. Dies entspricht bei der verwendeten internen Datenrate von 8 Bit/Takt einem Durchsatz von 0,28 Gbit/s. Wenn auch aufgrund von Hardware-Limitierungen die SNORT Instruktionen/ Regeln nur teilweise unterstützt werden können, so entspricht das in [AGGR09] erreichte Ergebnis ebenfalls nicht dem angestrebten Durchsatz des ID-System des SecAN.

Liu et. al. veröffentlichte 2009 ein Paper mit dem Titel „A Fast and Configurable Pattern Matching Hardware Architecture for Intrusion Detection“ [LXLS09]. Die Architektur erzielt im *Worst Case* einen Durchsatz von 3,2 Gbit/s. Obwohl der erzielte Durchsatz den Anforderungen des SecAN-ID-Systems entspricht, wird die Funktionalität mit Unterstützung eines zusätzlichen Netzwerkprozessors sowie zweier SRAMs und eines DRAMs realisiert. Sowohl der Virtex-4 FPGA als auch der Netzwerkprozessor müssen auf die externen Speicher zurückgreifen, da die Software-Teile und Vergleichsmuster in ihnen hinterlegt sind. Das IDS des SecAN wird weder den Netzwerkprozessor noch zusätzlich Speicher benötigen, um auftretende Angriffsmuster zu erkennen. Für diese Funktionalität werden ausschließlich FPGA-Ressourcen benötigt.

In [ZYGX10] entwickeln Zhikai et. al. eine Bloom-Filter-basierte Architektur für DPI-Systeme. Als Datenbasis für die Signaturen wird eine SNORT-Datenbank verwendet. Bekannt ist, dass Bloom-Filter *False Positives* erzeugen können. Die Autoren geben an, dass die False Positive Rate (FPR) ihres Systems bei 10 % liegt. Für eine positive QoE muss die FPR im SecAN-IDS jedoch deutlich geringer ausfallen. Aus diesem Grund wurde für das IDS des SecAN eine FPR von 0,001 % gefordert und eingehalten. Weiterhin wird der Durchsatz des Systems ([ZYGX10]) mit bis zu 10 Gbit/s angegeben. Aus Sicht des Autors ist dieses Ergebnis nicht schlüssig nachvollziehbar, da die in [ZYGX10] beschriebene Architektur nicht implementiert wurde und somit die finale Frequenz nicht bestimmt werden konnte.

Fazit: Die in [PKAC08, AGGR09, LXLS09, ZYGX10] vorgestellten Systeme weisen eine Reihe von Schwächen auf, die für das SecAN-IDS nicht akzeptabel sind. Deshalb wird in der nachfolgenden Konzeptphase (vgl. 4.7.3) ein IDS entwickelt, das keinen Musterlängen-abhängigen Algorithmus verwendet und in Fragen des Durchsatzes den gestellten Anforderungen entspricht. Weiterhin werden keine zusätzlichen Netzwerkprozessoren und externe Speicher verwendet wie in [LXLS09]. Es wird ausschließlich auf lokale FPGA-Ressourcen zurückgegriffen, um die Latenz möglichst gering zu halten. Ferner wird im SecAN-IDS die FPR nicht 10 % sondern 0,001 % betragen, sodass es zehntausendmal seltener zu möglichen Treffern kommt.

4.7.2.2 Klassifizierung von Intrusion Detection-Systemen

Die Klassifikation von IDS kann aufgrund unterschiedlicher Gesichtspunkte erfolgen. Häufig werden diese Systeme in Host-basierte IDS (HIDS), Netzwerk-basierte IDS (NIDS) und hybride Intrusion-Detection-System unterteilt. Dabei stellen die hybriden IDS eine Kombination aus HIDS und NIDS dar. Durch die Kombination der Vorteile von HIDS und NIDS entsteht ein sehr robustes System zur Erkennung von Eindringlingen.

Host-basierte Intrusion Detection-Systeme HIDS haben die Aufgabe, einzelne Rechner (Hosts) vor Netzwerkschäden zu bewahren. Dabei können sie unmittelbar vor dem zu sichernden Host, aber auch direkt im Host eingesetzt werden. Mittels „Sensoren“ wird bei externen HIDS der bidirektionale Netzwerkverkehr geprüft. Interne HIDS sind darüber hinaus in der Lage, systeminterne Datenpfade mit zu prüfen. Im letzten Fall können so unautorisierte Zugriffe und Trojaner erkannt werden.

Da HIDS lokal eingesetzt werden, ist der Managementaufwand besonders bei einer großen Anzahl Hosts relativ hoch. Darüber hinaus verwenden sie lokale Ressourcen wie z. B. Prozessor und Arbeitsspeicher, wodurch die Systemleistung reduziert wird. Ferner können HIDS nur die Daten nutzen, die sie selbst empfangen haben. Sicherheitsrelevante Daten, die im Netzwerk erkannt wurden, stehen nicht zur Verfügung und können somit nicht zum Schutz des Hosts beitragen. Letztendlich können bestimmte DoS-Attacken ein HIDS ausschalten [BM01]. Durch den Einsatz des SecAN-IDS im TZN werden diese Nachteile behoben.

Netzwerk-basierte Intrusion Detection-Systeme Einige der Nachteile von HIDS können die NIDS kompensieren. NIDS werden oft im industriellen Bereich eingesetzt und überwachen den gesamten Netzwerkverkehr. Durch geschickte Verteilung der Sensoren, die teilweise im „stealth"-Modus arbeiten, entsteht ein robustes System, welches Statusmeldungen über Angriffe an eine zentrale Instanz leitet. Auf erkannte Gefahren kann zentral reagiert und so eine Vielzahl von Hosts umfassend geschützt werden.

Aufgrund des hohen Datenverkehrs im Netzwerk werden die Sensoren stark gestresst. Deshalb verwenden einige Hersteller Hardware- statt Software-Lösungen für Sensoren. Damit ein NIDS den gesamten Datenverkehr überwachen kann, müssen die eingesetzten Switches Monitor-Ports besitzen. Anderenfalls können durch den Switch geschaltete exklusive Kommunikationswege nicht überwacht werden. Eine besondere Herausforderung stellen verschlüsselte Informationen dar. Der Inhalt dieser Nachrichten kann nicht durch ein NIDS überprüft werden. Außerdem können Beeinträchtigungen und Abstürze von NIDS aufgrund fragmentierter oder fehlerhafter Pakete hervorgerufen werden [BM01].

4.7.2.3 Angriffe auf Intrusion Detection-Systeme

Viele Parameter haben Einfluss auf die Qualität von Intrusion Detection-Systemen. Prinzipiell existieren zwei wesentliche Problemgruppen, die Einfluss auf die Erkennungsrate eines IDS haben:

1. Gruppe: Ein IDS sollte eine allgegenwärtige Sicht über das gesamte Netzwerkverhalten aller Hosts besitzen. In der Realität ist diese Anforderung nicht realisierbar. So besitzen IDS in der Regel nur eingeschränktes Wissen über den Datenverkehr der Knoten eines Netzwerkes. Hinzu kommt, dass Angriffe aufgrund unvollständiger Beschreibungen

von Signaturen oft nur unzureichend erkannt werden. Ferner müsste ein IDS das Verhalten eines Hosts auf ein manipuliertes Netzwerkpaket vorherbestimmen können. Nur wenn alle Eckpunkte eingehalten werden, kann eine fehlerfreie Entscheidung bezüglich eines Alarms getroffen werden.

2. Gruppe: Hinzu kommt, dass IDS anfällig für verschiedene Angriffe sind. Die Ursachen sind in der begrenzten Rechenkapazität sowie der Netzlastverträglichkeit zu finden. Mit ständig steigenden Datenraten werden besonders Software-basierte IDS schnell an ihre Grenzen gelangen. Überschreiten die zu analysierenden Daten die Ressourcen des IDS, können grundsätzlich zwei architekturabhängige Situationen eintreten:

1. Das IDS arbeitet in Volllast. Zusätzlich eintreffende Datenpakete werden unkontrolliert am IDS vorbei geleitet. Diese Art der Reaktion auf einen Volllastbetrieb wird als „fail open“-Situation beschrieben. In diesem Fall sind Angriffe auf Zielsysteme durchführbar.
2. Das Netzsegment, welches durch ein IDS überwacht wird, wird isoliert. Die eingetretene Situation wird mit „fail close“-Situation beschrieben. Eine Kommunikation der Recheneinheiten ist lediglich im isolierten Netz möglich. Ein- und ausgehender Datenverkehr wird vom überlasteten IDS blockiert. Somit sind Angriffe von außerhalb des isolierten Netzes nicht erfolgreich.

Im Wesentlichen lassen sich die Angriffe auf IDS in die drei Gruppen „Insertion“, „Evasion“ und „DoS“ kategorisieren.

Insertion: Es werden zusätzliche Datenpakete in den Datenstrom injiziert. Diese weisen keine Bedrohungsmuster auf und werden vom IDS akzeptiert. Sie gelangen somit auf den Ziel-Host. Der Ziel-Host seinerseits verwirft diese Pakete, da sie Anomalien wie bspw. ungültige Prüfsummen aufweisen. Ein erfolgreicher Insertion-Angriff hat zum Ziel, fragmentierte Angriffsmuster durch das IDS zu leiten und den Ziel-Host die „überflüssigen“ Daten verwerfen zu lassen. Ferner wird der Ziel-Host das fragmentierte Angriffsmuster vollständig zusammensetzen und ist somit kompromittiert [PN98].

Evasion: Ähnlich wie bei der Insertion-Strategie werden zusätzliche Datenpakete in den Datenstrom eingefügt. Jedoch erkennt bei einem Evasion-Angriff das IDS die zusätzlich eingebrachten Pakete als Anomalie und verwirft sie. Das eigentliche Bedrohungsmuster ist über mehrere Datenpakete verteilt und wird, wenn alle „überflüssigen“ Datenpakete vom IDS verworfen wurden, vom Ziel-Host vollständig re-assembliert. Der Grund des Verwurfs von Paketen durch das IDS kann in angepassten TTL-Werten innerhalb des

IP-Headers liegen. Läuft der TTL-Wert direkt am IDS ab, wird das Paket vom IDS verworfen. Die Umgehung einer Angriffserkennung erfolgt also mit Hilfe des IDS. Das IDS erkennt und löscht alle anomaliebasierten Pakete, besitzt jedoch keinen globalen Überblick und erkennt somit nicht, dass das Bedrohungsmuster durch die restlichen Pakete etabliert wird [PN98].

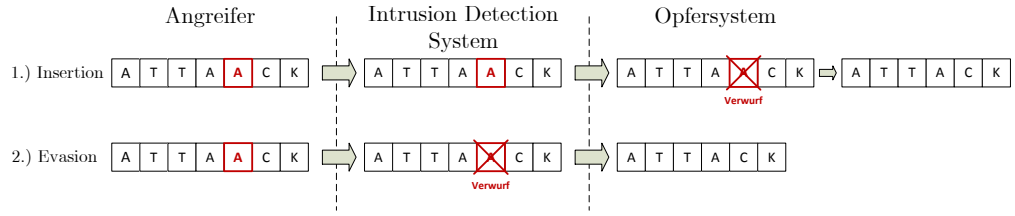


Abbildung 4.16: Insertion- und Evasion-Angriffe auf Intrusion-Detection-Systeme

Abbildung 4.16 verdeutlicht die Unterschiede der beschriebenen Methoden Insertion und Evasion. Für eine erfolgreiche Durchführung dieser Angriffe müssen auf Seiten des Angreifers äußerst gute netzwerktheoretische Kenntnisse sowie ausgeprägte Kenntnisse im Bereich Betriebssysteme vorhanden sein. Ferner können Fragmentierungen von Netzwerkpaketen eingesetzt werden, um die Funktionalität eines IDS ad absurdum zu führen.

DoS: Intrusion Detection-Systeme sind je nach Architektur extrem anfällig für Denial of Service-Attacken [PN98]. Bei einer DoS-Attacke werden die Ressourcen eines IDS bis zum Limit belastet. In der Folge kann das IDS in den „fail open“- bzw. „fail close“-Modus schalten. Aus Sicht der Netzwerksicherheit ist die „fail close“-Strategie die bessere Wahl, da sie selbst dann Schutz bietet, wenn das IDS bis zum Volllastbetrieb angegriffen bzw. betrieben wird. Für eine detaillierte Analyse der Angriffsmethoden auf Intrusion Detection-Systeme mit Fallbeispielen sei auf [PN98] verwiesen.

4.7.2.4 Erkennung von Eindringlingen mit SNORT

Die freie und quelloffene Software-Lösung „SNORT“[13] stellt den de facto Standard für Intrusion-Detection- und Intrusion-Prevention-System dar. 1998 von Martin Roesch programmiert, wird die Software heute von Sourcefire Inc.[14] weiterentwickelt.

SNORT ist ein auf der libpcap[15]-Bibliothek basierender Paket-Sniffer, welcher um Me-

[13] SNORT - http://www.snort.org/
[14] Sourcefire Inc. - http://www.sourcefire.com/
[15] Libpcap - http://sourceforge.net/projects/libpcap/

thoden zur Signatur- und Anomalieanalyse erweitert wurde und somit als IDS und Intrusion-Prevention-System (IPS) eingesetzt werden kann. Somit können durch SNORT Buffer Overflow-Attacken, Stealth Port Scans, CGI-Angriffe und SMB-Probes erkannt werden. Um die beschriebene Aufgaben zu bewältigen, greift SNORT auf eine Signaturdatenbank zurück. Diese wird aufgrund von Regeln, welche einer einfachen definierten Syntax folgen, erstellt.

Intern besteht SNORT aus den Komponenten „Packet Decoder“, „Detection Engine“ und dem „Logging/Alerting Subsystem“. Der „Packet Decoder“ ist ein Präprozessor. Er untersucht den Rohdatenstrom über das gesamte ISO/OSI-Schichtenmodell. In den Rohdaten werden Zeiger auf entsprechende Stellen in den Protokoll-Headern gesetzt.

Im Anschluss werden die klassifizierten Rohdaten der „Detection Engine“ zur Verfügung gestellt. Diese wendet, abhängig von den Header-Daten (IP-Adressen, Protokoll-Nummern und Port-Nummern), Regeln aus dem SNORT-Regelwerk auf den Daten-Frame an. Dabei löst der erste erfolgreich gefundene Eintrag eine Aktion aus („First Match Counts“-Prinzip). Um die Funktionalität der „Detection Engine“ zu erweitern, ist es möglich, „Plugins“ für diese zu entwickeln. Da das SNORT-Regelwerk ein wesentlicher Bestandteil des entwickelten SecAN-IDS ist, sind weiterführende Informationen im Anhang C.6.1 zu finden.

4.7.2.5 Bloom-Filter

Für die vorliegende Forschungsarbeit ist der Bloom-Filter-Ansatz von essentieller Bedeutung. Deshalb wird nachfolgend darauf eingegangen.

In den 1970er Jahren entwickelte Burton H. Bloom einen neuartigen Ansatz, um Mengenzugehörigkeiten nachzuweisen [Blo70]. Die nach ihm benannten Filter wurden zunächst für die Rechtschreibprüfung und Wortsilbentrennung und später für Datenbankanwendungen eingesetzt. Darüber hinaus sind sie ebenfalls zur Signaturanalyse in der Netzwerktechnik geeignet. Besonders wenn Speichereffizienz und deterministisches Zeitverhalten von grundlegender Bedeutung sind, zählen Hardware-basierte Bloom-Filter-Ansätze zu den vielversprechensten Lösungswegen.

Funktionsweise des Standard-Bloom-Filters: Neben dem von Burton H. Bloom entwickelten Ansatz existiert heute eine Vielzahl von Variationen des Bloom-Filter (BF)s

[BM04]. Sie alle besitzen grundlegende Eigenschaften und Funktionsweisen, die im Folgenden näher erläutert werden.

$S = x_1, x_2, x_3, ..., x_n$ ist eine Menge von Elementen und B ein m Bit breites Array, welches für alle Elemente den Initialwert logisch 0 enthält. Weiterhin nutzt ein BF k unabhängige Hash-Funktionen $H = h_1, h_2, ..., h_k$ über den gesamten Bereich des Arrays B $1, ..., m$. Jedes Element $x \in S$ wird von $1 \leq i \leq k$ Hash-Funktionen verarbeitet $(h_i(x))$. Die errechneten k Hash-Werte dienen als Indizes des Arrays B. Die Werte der Indizes wechseln zum logischen Wert 1.

Wenn bspw. zwei unabhängige Hash-Funktionen bei gleichem Eingangswert denselben Hash-Wert ermitteln, wird zwei Mal eine 1 auf denselben Index im Array B geschrieben. Lediglich der erste errechnete Hash-Wert verursacht einen Effekt im Array B. Sind alle Elemente von S „gehasht“ und alle errechneten Indizes des Arrays B besitzen den Wert 1, ist das Programmieren des BFs abgeschlossen. Somit stellt ein BF ein Bool'sches Array bezüglich der „gehashten“ Eingangsmuster dar. Abbildung 4.17 zeigt die beschriebenen ersten zwei Phasen der Initialisierung und Programmierung noch einmal grafisch.

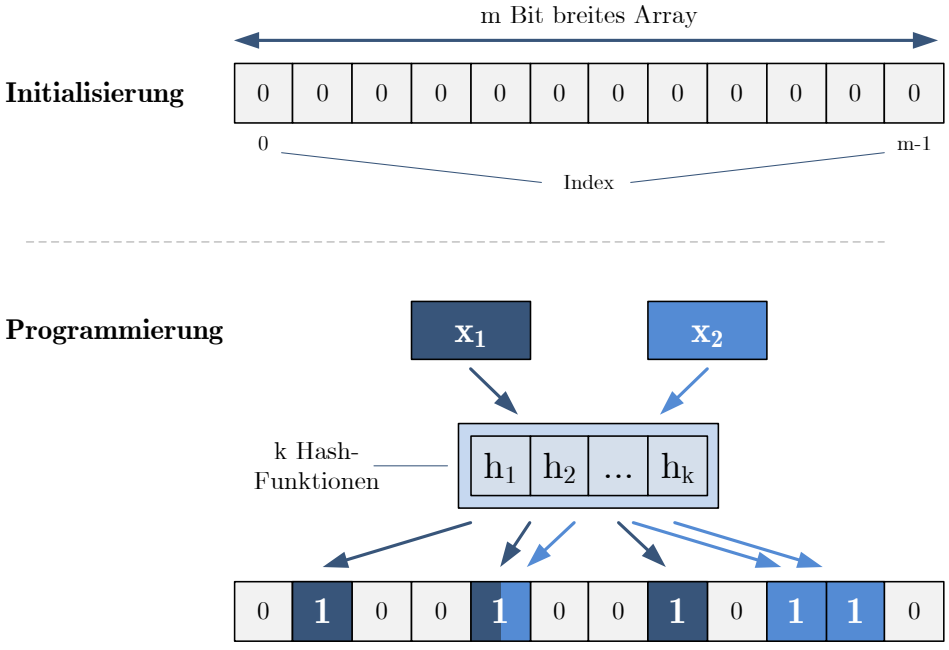


Abbildung 4.17: Initialisierung und Programmierung eines Standard-Bloom-Filters

Im oberen Bereich von Abbildung 4.17 ist das mit 0 vorinitialisierte Bool'sche-Array

veranschaulicht. Die Programmierung der Elemente x_1 und x_2 erfolgt, wie im unteren Teil der Abbildung 4.17 zu sehen, über die Indizes der k Hash-Funktionen. Bei den durch die Hash-Funktionen errechneten Indizes wechselt der Initialwert zum logischen Wert 1 bzw. bleibt der Wert 1, falls dieser bereits vorhanden ist. Die Programmierung eines BFs ist abgeschlossen, wenn alle Elemente der Menge S „gehasht“ und an den errechneten Indizes im Array B eine logische „1“ gespeichert wurde.

Die Suche in einem BF (Lookup-Phase) wird auf sehr ähnliche Weise wie die Programmierung realisiert. Für die Suche eines Elementes werden dieselben Hash-Funktionen wie für die Speicherung von Elementen verwendet. Die Hash-Werte dienen erneut als Indizes im Array B. Aufgrund dieser Eigenschaft lassen sich alle zuvor gespeicherten Elemente wiederfinden. Das Vorkommen von *False Negatives* - ein zuvor gespeichertes Element wird in einem BF nicht wiedergefunden - ist also ausgeschlossen.

Beispiel: Soll überprüft werden, ob die Elemente y_1, y_2 und y_3 in der Menge S enthalten sind ($y_1, y_2, y_3 \in S$), werden die Elemente mit den für die Programmierung verwendeten Hash-Funktionen verarbeitet. Ist unter einem beliebigen errechneten Index der logische Wert 0 zu finden, führt dies bereits zum Ausschluss des Elementes aus der Menge ($y_x \notin S$). Lediglich wenn alle bestimmten Indizes eines Elementes im Array B den logischen Wert 1 besitzen, handelt es sich mit einer gewissen Wahrscheinlichkeit um einen Treffer (Match). Die Wahrscheinlichkeit eines gültigen Treffers ist dabei $\leq 100\%$, weil durch eine Rekombination von 1en verschiedener Elemente der Menge S Treffer angezeigt werden, die im Grunde keine gültigen Treffer darstellen. Das Anzeigen eines solchen Elementes als Treffer wird als *False Positive* bezeichnet. Um eine endgültige Aussage treffen zu können, gilt es, *False Positives* aufzulösen. In Abbildung 4.18 ist die beschriebene Suchphase noch einmal grafisch dargestellt.

Die Suche beginnt mit dem Element y_1. Da nicht an allen errechneten Indizes in Array B der Wert 1 zu finden ist, gehört das Element y_1 definitiv nicht in die Menge S. Dahingegen erzeugen die Elemente y_2 und y_3 jeweils einen Treffer, weil alle errechneten Indizes auf eine 1 im Array B verweisen. Diese Elemente „können“ also Elemente der Menge S sein. Für das Element y_2 stimmt diese Annahme, für y_3 jedoch nicht. Wie Abbildung 4.17 zu entnehmen ist, setzt sich das Element y_3 aus den Indizes der Elemente x_1 und x_2 zusammen. Es handelt sich also um ein *False Positive*, welches an dieser Stelle jedoch nicht erkannt werden kann.

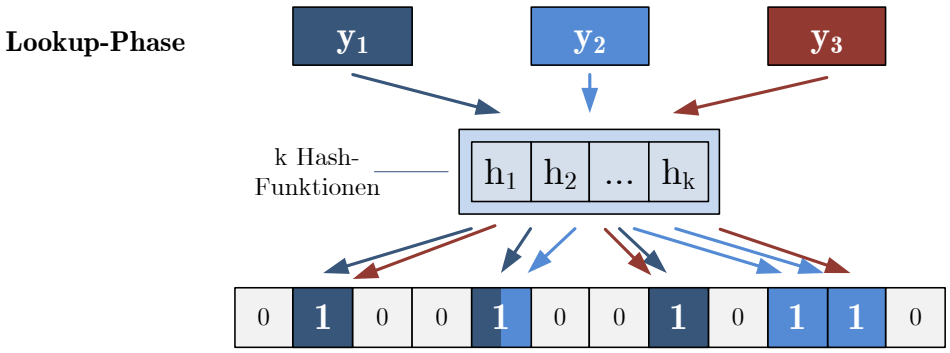


Abbildung 4.18: Suche im Standard-Bloom-Filter

False Positive Rate: Sind alle gewählten Hash-Funktionen untereinander unabhängig und erzeugen diese gleich verteilt Indizes, so lässt sich die FPR und damit die Qualität des BFs berechnen. Formel 4.3 [BM04] trifft eine Aussage über die Wahrscheinlichkeit, dass, nachdem alle n Elemente im m Bit breiten Array gehascht abgelegt wurden, ein spezielles Bit weiterhin den logischen Wert 0 besitzt.

$$\acute{p}(b_i = 0) = \left(1 - \frac{1}{m}\right)^{kn} \approx e^{-\frac{kn}{m}} = p \tag{4.3}$$

Damit die Abschätzung scharf ist, müssen kn und m die gleiche Größenordnung besitzen und m hinreichend groß sein ($kn < m$). p ist dabei die Verteilungsfunktion aller Bit mit dem logischen Wert 0 im vollständig programmierten BF und gleichzeitig eine asymptotische Approximation von $\acute{p}$ im Bereich $O(\frac{1}{m})$ ist. Der Erwartungswert von ρ ist nach [BM04]:

$$E(\rho) = \acute{p}(b_i = 0) \tag{4.4}$$

Abhängig von ρ ist die Wahrscheinlichkeit eines *False Positive*:

$$(1 - \rho)^k \approx (1 - E(\rho))^k \approx (1 - \acute{p}(b_i = 0))^k \approx (1 - p)^k. \tag{4.5}$$

Damit ergibt sich für die *False Positive*-Rate:

$$\acute{f} = (1 - (1 - \frac{1}{m})^{kn})^k = (1 - \acute{p}(b_i = 0))^k \tag{4.6}$$

Für die zugehörige asymptotische Approximation gilt:

$$f = (1 - e^{-\frac{kn}{m}})^k = (1 - p)^k \tag{4.7}$$

Weil sich mit BF *False Positives* nicht vermeiden lassen, muss das Ziel sein, die FPR zu optimieren und auf ein Minimum zu reduzieren. Nach [BM04] ist es grundsätzlich einfacher und vollkommen ausreichend, für die FPR p und f anstatt $\acute{p}$ und $\acute{f}$ zu verwenden. Um eine optimale Lösung zu finden, stehen dabei drei Freiheitsgrade (k,m,n) zur Verfügung, von denen möglichst alle genutzt werden sollten.

Beispiel: Sind für einen BF die Bit-Breite m und die Anzahl der Elemente n gegeben, so gilt es die Anzahl der Hash-Funktionen zu errechnen. [Sch11] weist für das angegebene Beispiel nach, dass die optimale Anzahl Hash-Funktionen für eine minimale FPR $k = ln(2)\frac{m}{n}$ ist.

Mit den beschriebenen Eigenschaften sind BF für die Verwendung von Mengenoperationen geeignet. Repräsentieren die Mengen S_1 und S_2 den Inhalt zweier gleich langer BF, so kann ein bitweises „ODER“ beider Filter durchgeführt werden. Auf diese Weise entsteht ein neuer BF mit der Menge S, welcher die Teilmengen S_1 und S_2 repräsentiert. Darüber hinaus kann die Länge von Bloom-Filtern dynamisch angepasst werden, wenn es sich dabei um Zweierpotenzen handelt. Wird bei einer Binärzahl das Most Significant Bit (MSB) verworfen, entspricht dies einer Halbierung des Wertebereichs. Im Falle von Bloom-Filtern wird das MSB der Ergebnisse der Hash-Funktionen in der Lookup-Phase ignoriert.

Nachdem in den 1970er Jahren der Standard BF entwickelt wurde, fanden in den Folgejahren eine Vielzahl an Modifikationen statt. So wurden unter anderem „Counting Bloom-Filter“ [FCAB00], „Dynamic Bloom-Filter“ [Hau07, GWCL06], „Scalable Bloom-Filter“ [ABP07], „Compressed Bloom-Filter“ [BM04], „Weighted Bloom-Filter“ [BGJ06], „Spectral Bloom-Filter“ [CM03], „Adaptive Bloom-Filter“ [MHK07] sowie „Distance-Sensitive Bloom-Filter“ [KM06] entwickelt.

4.7.3 Konzipierung

In diesem Abschnitt der Arbeit werden relevante Konzeptentscheidungen und Designalternativen diskutiert. Hinsichtlich der Ressourcenbeschränkungen ist es nicht möglich,

alle Konzepte zu implementieren. Aufgrund ihres innovativen Charakters müssen die Ideen und Ansätze dennoch erläutert werden.

Ferner werden verschiedene Mustererkennungsalgorithmen unter Berücksichtigung der verfügbaren Ressourcen und der Anforderungen an das System untersucht, um einen optimalen Algorithmus zu identifizieren. Abschließend findet eine Diskussion der wichtigsten Designkonzepte und -entscheidungen statt.

4.7.3.1 Auswahl eines geeigneten Mustererkennungsalgorithmus

Nachfolgend werden drei Kategorien von Algorithmen für die IDS-Filterstufe im SecAN untersucht und verglichen. Neben Implementierungen als Deterministic Finite State Machine (DFSM) werden ein TCAM-basierter Ansatz sowie ein BF-basierter Ansatz untersucht. Dazu existiert ein Kriterienkatalog, bestehend aus sieben Unterpunkten. Folgende Kritikpunkte sind Bestandteil des Kataloges:

1. **Flexibilität:** Wie flexibel ist das System, wenn sich das Regelwerk ändert?
2. **Online-Update:** Können neue DPI-Regeln im laufenden Betrieb in das DPI-System übernommen werden?
3. **Skalierbarkeit:** Wie stark werden Hardware-Ressourcen beansprucht, wenn sich das Regelwerk vergrößert bzw. verkleinert?
4. **Parallelität:** Können mehrere bzw. alle Signaturen parallel durchsucht werden?
5. **Zuverlässigkeit:** Findet das System alle „suspekten“ Signaturen wieder? Werden nicht alle Signaturen gefunden (*False Negatives*) bzw. werden Signaturen fälschlicherweise gefunden (*False Positives*)?
6. **Kosten:** Wie hoch ist der Hardware-Aufwand für das System? Kann es in die Zielplattform integriert werden?
7. **Performance:** Wie schnell erfolgt eine vollständige Signaturanalyse? Handelt es sich um ein deterministisches Verfahren bezüglich des Zeitverhaltens?

DFSM-basierter Ansatz: Beispielhaft kann der „Aho-Corasick“-Algorithmus für diese Kategorie gewählt werden. Einerseits können die Signaturen in einem externen Speicher (wie z. B. in [Dan06, Nin06] realisiert) und andererseits als DFSM direkt in FPGA-Ressourcen implementiert werden. Bei einer Random Access Memory (RAM)-basierten

Umsetzung ist mit unvertretbaren Latenzen zu rechnen (vgl. [Dan06, Nin06]). Eine DFSM als dedizierter, monolithischer Hardware-Block ist deutlich schneller, besitzt jedoch andere ungünstige Eigenschaften.

Bei Änderung des Regelwerkes muss die Hardware neu synthetisiert werden. Somit werden Online-Updates sehr zeitaufwändig und benötigen zudem Hardware-Entwicklungstools und spezielles Hardware-Fachwissen. Ein DFSM-basierter Ansatz skaliert nur sehr schlecht, wenn neue Signaturen als Infix-, Suffix- oder Präfixmuster in bestehenden Signaturen auftreten. In diesem Fall steigt die Anzahl der Zustands- und Fehlerübergänge überproportional. Obwohl die Implementierung lediglich FPGA-Logik benötigt, schlägt sich diese Eigenschaft negativ in den Hardware-Kosten nieder. Im Hinblick auf Zuverlässigkeit kann dieser Ansatz jedoch punkten, denn bei einer fehlerfreien Implementierung treten weder *False Positives* noch *False Negatives* auf. Gleichzeitig ist der „Aho-Corasick“ ein Multiple-Pattern-Matching Algorithmus der stets nach allen programmierten Signaturen parallel sucht. Positiv ist auch die Performance zu bewerten. Bei optimaler Umsetzung entsteht eine Latenz von nur einem Takt.

Assoziativspeicher-basierter Ansatz: Bei diesem Ansatz werden in einem Präprozess alle Signaturen des Regelwerkes in den Assoziativspeicher übertragen. In der anschließenden Suchphase wird das zu suchende Muster mit allen Einträgen des Speichers parallel verglichen. Als Speicher bieten sich hier TCAMs an, da diese neben den Zuständen „0“ und „1“ den „Don’t Care“-Zustand kennen, wodurch es möglich wird, unterschiedlich lange Muster wiederzufinden.

Im Hinblick auf das Auffinden von Signaturen sind inhaltsadressierbare Speicher in der Lage, eine First-Match/Last-Match/Multi-Match Entscheidung auszugeben. Das Entfernen, Hinzufügen und Manipulieren von Speicherelementen kann problemlos zwischen den Suchphasen erfolgen. Somit sind inhaltsadressierbare Speicher bestens für Online-Updates geeignet. Vorausgesetzt ein ausreichend großer TCAM ist verfügbar, so skalieren die Gesamtlänge aller Signaturen und die Größe des TCAM linear, da für jedes Byte der Signatur genau ein Byte im TCAM bereitgestellt werden muss. Die Zuverlässigkeit der TCAM-Ergebnisse wird durch Multi-Match-Ergebnisse sichergestellt.

Auf der Zielplattform ließe sich mittels Core-Generator (vgl. Abschnitt D.4) ein TCAM aus Slice-Registern generieren. Dieser hätte eine Leselatenz von einem Takt und eine Schreiblatenz von 16 Takten. Da das Schreiben in einem Präprozess abläuft und somit zeitlich unkritisch ist, kann die Geschwindigkeit des generierten TCAM als sehr hoch

TCAM-Tiefe	SLR16E (absolut)	SLR16E (in %)	MAP-Prozess
20	2.400	18%	erfolgreich
100	12.000	91%	nicht erfolgreich
1500	180.000	nicht realisierbar	nicht erfolgreich
4000	480.000	nicht realisierbar	nicht erfolgreich

Tabelle 4.13: TCAM-Machbarkeitsstudie auf einem Virtex-5 FX70T

eingestuft werden. Neben den vielen Vorteilen von CAM-Speichern fallen besonders deren Hardware-Kosten negativ ins Gewicht. Jedoch wird der TCAM den Ansprüchen in Bezug der zu speichernden Signaturen nicht gerecht. Tabelle 4.13 stellt die Tiefe des TCAM den benötigten Ressourcen gegenüber. Da sich ein TCAM auf der Zielplattform nicht aus BRAMs generieren lässt, sind als Ressourcen Slice-Register (SLR16E) aufgeführt. Bei einer maximalen Signaturlänge von 30 Byte war der MAP-Prozess nicht mehr in der Lage, einen TCAM der Tiefe 100 auf der Zielplattform abzubilden.

Bloom-Filter-basierter Ansatz: Als drittes wird der BF-basierte Ansatz untersucht. Um die Qualität des Filters zu verbessern, werden in einem BF nur Signaturen einer Länge abgespeichert. Gleichzeitig wird die maximale Länge der unterstützten Signaturen auf 30 Byte begrenzt. In der Konsequenz müssen mindestens 30 BF in die IDS-Filterstufe integriert werden.

Die Flexibilität dieses Ansatzes ist vom BF-Typ abhängig. Während das Hinzufügen neuer Signaturen von jedem Bloom-Filter problemlos realisiert werden kann, stellt das Entfernen eine Herausforderung dar. Aus diesem Grund wurden Counting-Bloom-Filter entwickelt. Sie beherrschen auch das fehlerfreie Löschen von Signaturen. Ist das Löschen von Signaturen von niedriger Relevanz bzw. nicht notwendig, können durch vollständige Neukonfiguration der BF problemlos Online-Updates durchgeführt werden. Die Skalierbarkeit von Bloom-Filtern ist von verschiedenen Parametern abhängig. So stehen sich die Länge des Filters, die Anzahl der Hash-Funktionen und die Anzahl zu speichernder Signaturen gegenüber und beeinflussen die FPR. Sind die Länge des Filters, die Anzahl der Hash-Funktionen sowie die FPR fix, ist es hinsichtlich des Hardware-Aufwandes unerheblich, ob die Obergrenze an zulässigen Signaturen erreicht wird. Wird die Obergrenze überschritten, müssen weitere BF zum System hinzugefügt werden, um die FPR konstant zu halten. Sind alle BF von ihren Eigenschaften identisch, verdoppelt sich bei

Kriterium	DFSM	TCAM	Bloom-Filter
Flexibilität	- -	++	+ bis ++
Online-Update	-	++	++
Skalierbarkeit	-	++	+
Parallelität	++	++	++
Zuverlässigkeit	++	++	- bis +
Kosten	- - bis ++	-	+
Performance	- bis ++	++	++

Tabelle 4.14: Machbarkeitsstudie zur Umsetzung auf einem Virtex-5 FX70T

Überschreitung der Obergrenze der Hardware-Aufwand. In diesem Fall skalieren die Ressourcen linear mit der Zahl der Signaturen. Alle in einen BF programmierten Elemente gehören in dieselbe Menge von Elementen (z. B. Elemente derselben Länge). Da BF für die Prüfung von Mengenzugehörigkeiten entworfen wurden, überprüfen sie stets alle gespeicherten Elemente parallel. Eine positive Eigenschaft von Bloom-Filtern ist, dass keine *False Negatives* auftreten. Allerdings können *False Positives* sehr wohl entstehen. Durch entsprechende Dimensionierung des Filters lässt sich die Wahrscheinlichkeit reduzieren, jedoch nicht aufheben. Insofern ist die Zuverlässigkeit des Filters stark von dessen Dimensionierung abhängig. Angesichts der Tatsachen, dass sich BF aus BRAM-Slices erstellen lassen, beträgt die Lese- und Schreiblatenz lediglich einen Takt. Somit kann eine hohe Performance für die BF-Struktur vorhergesagt werden.

Auswertung: In Tabelle 4.14 sind in komprimierter Form die Ergebnisse der Machbarkeitsstudie zur IDS-Filterstufe dargestellt. Die Ergebnisse reichen von einem doppelten Minus (sehr schlecht) bis zu einem doppelten Plus (sehr gut).

Eine reine DFSM-Implementierung ist zu unflexibel, um auf sich verändernde Regeln ausreichend schnell reagieren zu können. Auch eine zusätzliche Unterstützung durch einen externen Speicher kann nicht präferiert werden, da diese Lösung zu hohen Verzögerungen und erheblichen ungewollten Datenverwürfen führen würde. Obwohl ein TCAM die optimale Wahl bezüglich Performance wäre, kann diese Lösung ebenfalls nicht verfolgt werden, da die Anzahl der zu speichernden Signaturen auf unter 100 begrenzt wäre. Dahingegen konnte der BF-Ansatz in den meisten Punkten überzeugen. Lediglich das Auftreten von *False Positives* stellt eine Herausforderung dar. Sie müssen entweder tole-

riert oder mittels eines zusätzlichen Analyzer-Moduls eliminiert werden. Für den Autor der vorliegenden Arbeit können *False Positives* toleriert werden, wenn die IDS-Filterstufe mit einer Genauigkeit von 99,999% arbeitet.

4.7.3.2 SNORT-Hardware-Mapping

SNORT ist ein robustes und ausreichend erprobtes System mit einer langen Entwicklungsgeschichte. Es stellt den de facto Standard für Echtzeitanalyse für IP-basierte Netze dar. Daher ist es sinnvoll, die SNORT-Regeln für ein Hardware-IDS nachzunutzen.

Da der Aufbau von SNORT-Regeln beliebig komplex ausfallen kann (vgl. Abschnitt 4.7.2.4), muss die Regelvielfalt auf ein Hardware-verträgliches Niveau reduziert werden. Folglich müssen zunächst Hardware-kompatible Regeln identifiziert werden. Dazu wird in diesem Abschnitt eine Analyse des SNORT-Regelwerkes aus vier verschiedenen SNORT-Datenbanken durchgeführt. So können entsprechende Regeln identifiziert und Variationen im Regelwerk ermittelt werden. Die Analyseergebnisse werden in eine vereinfachte SNORT-Syntax überführt. Auf diese Weise können ISP-Administratoren leicht eigene Regeln erstellen und in das SecAN-IDS aufnehmen. Abschließend wird das Prinzip vorgestellt, mit dem die Abbildung der Signaturen und der Zusatzparameter auf die BF erfolgt.

Analyse des SNORT-Regelwerkes Der Aufbau der SNORT-Syntax wurde bereits in Abschnitt 4.7.2.4 vorgestellt. Im Folgenden werden die vier SNORT-Datenbanken 2.8.5.3, DB 100530, 2.8.6.0 sowie 2.8.6.1 auf ihre Schlüsselwörter und deren Hardware-Eignung analysiert.

Aktion:
Der Bereich *Aktion* besteht aus acht Schlüsselwörtern (vgl. Tabelle C.4). Von diesen werden die `activate` und `dynamic` nicht weiter verfolgt, da sie zu einem hochkomplexen stateful-IDS führen, welche sich mittels Bloom-Filtern nicht realisieren lässt.

Protokoll:
Über das Schlüsselwort *Protokoll* wird das zu analysierende Protokoll angegeben. Im SecAN-IDS werden die häufig verwendeten Protokolle TCP, UDP, ICMP und IP in den Versionen IPv4 und IPv6 unterstützt. Alle Protokolle lassen sich in Form von Protokollnummern in der SNORT-Syntax wiederfinden. In der vereinfachten SNORT-Syntax

werden sie jedoch im Klartext angegeben und erst durch die Konfigurations-Software wieder in Protokollnummern übersetzt.

Richtung:
Der Richtungsoperator befindet sich zwischen Quelle und Ziel. Es existieren zwei Operatoren:

$\rightarrow$: unidirektional von IP1/Port1 nach IP2/Port2

$\Leftrightarrow$: bidirektional

Zu beachten ist, dass bidirektionale Regeln in zwei unidirektionale Regeln überführt werden.

IP-Adressen:
Zu den Adressangaben einer SNORT-Regel gehören zwei IP-Adressen (Quelle und Ziel). In den untersuchten SNORT-Regelwerken befinden sich einzelnen IP-sensitive Regeln. Sie münden in einem hohen Hardware-Aufwand und werden deshalb nicht unterstützt. Stattdessen werden die zahlenmäßig deutlich häufiger vorkommenden Regeln mit den Platzhaltern `$EXTERNAL_NET` und `$HOME_NET` in die vereinfachte SNORT-Syntax integriert.

Port-Nummern:
Bei den *Port-Nummern* handelt es sich um die Port-Angaben der Header der Transportschicht des OSI-Modells. Eine spezielle Mikrosyntax ermöglicht unterschiedlichste Port-Angaben.

1. Single-Port (23): einzelner Port
2. Port-Range (23:43): alle Ports von 23 bis 43
3. Port-Liste ([23,33,43]): alle einzelnen Ports innerhalb der Liste
4. Kombinierte Liste ([23,33,43:48]): betrifft die Ports 23 und 33 sowie alle Ports von 43 bis 48
5. Full-Range (any): ein beliebiger Port zwischen 1 und 65636
6. Port-Variablen: unterstützt werden `$HTTP_PORTS` = [80,8080], `$SSH_PORTS` = [22]

Mit Ausnahmen des Ports `any` werden genau wie beim bidirektionalen Richtungsoperator alle Port-Angaben, die mehr als einen Port enthalten, in einzelne Port-Regeln aufgeschlüsselt. Für den Beispiel-Range [23:43] werden also 21 separate Regeln definiert.

Die Obergrenze für Port-Ranges wurde auf 100 Werte fixiert. Negierungen (!) und offene Port-Angaben (z. B. 23:) werden nicht unterstützt. Die dargestellten Port-Variablen werden wie einzelne Ports bzw. Ranges behandelt.

Regel:
Der letzte Teil innerhalb der SNORT-Syntax entspricht der Regel. Jede Regel kann aus zwei Teilen bestehen: einer Beschreibung und dem zu filternden Inhalt. Beide Teile werden von runden Klammern umschlossen. Beschreibungen werden durch das Schlüsselwort `msg` eingeleitet. Bei den zu filternden Inhalten werden die Schlüsselwörter `content` und `uricontent` unterstützt. In der vereinfachten SNORT-Syntax ist eine Regel somit wie folgt aufgebaut:

(`msg:"Beschreibung";[uri]content:"Signatur"`)

Einzelne Operanden der Regel werden mit einem Semikolon voneinander getrennt. Jeder Operand wird mit einem der beschriebenen Schlüsselwörter eingeleitet. Gefolgt von einem Doppelpunkt befindet sich in Anführungsstrichen die Beschreibung bzw. die zu filternde Signatur.

Gleich der SNORT-Syntax können Signaturen im Klartext und/oder im Byte Code bzw. als Kombination beider angegeben werden. Die drei folgenden Beispiele verdeutlichen die Darstellungsarten:

- `content:"hello world"`
- `content:"|104 101 108 108 111 32 119 111 114 108 100|"`
- `content:"he|108 108 111 32 119 111|rl|100"`

Da in den Bloom-Filtern des DPI-Filters nach diskreten Signaturen gesucht wird, werden ausschließlich einzelne Signaturen, sogenannte *Simple Pattern*, unterstützt. Hinzu kommt, dass die Schlüsselworte `content` und `uricontent` nur einmal je Regel und nicht in Kombination erscheinen dürfen. Obwohl die Original-SNORT-Syntax auch sogenannte *Nested Rules* unterstützt, können aufgrund von Hardware-Beschränkungen nur Regeln mit *Simple Pattern* kreiert werden. Regeln mit „Regulären Ausdrücken“ (Perl Compatible Regular Expressions (PCRE)) und komplexe Regeln werden demnach nicht unterstützt.

Beispiele: Nachdem herausgearbeitet wurde, welchen Aufbau und welche Eigenschaften die angepasste SNORT-Syntax im SecAN-Projekt aufweist, folgen zwei Beispiele.

Beide Beispiele können in dieser Form auch in das reguläre SNORT-System übernommen werden, da die modifizierte zur originalen SNORT-Syntax kompatibel ist.

Beispiel 1: Die folgende Regel besagt, dass alle TCP-Verbindungen aus dem externen Netz in Richtung internes Netz auf Port 22 überprüft werden. Befindet sich die Signatur „USER:root“ in der Payload, wird ein Alarm ausgelöst. Mit Hilfe dieser Regel lassen sich Authentifizierungsversuche als Administrator auf einem SSH-Server registrieren:

```
alert tcp $EXTERNAL_NET any → $HOME_NET 22 (msg:"SSH ROOT ACCESS";
content:"USER:root")
```

Beispiel 2: Die zweite Regel besagt Folgendes: Alle Ethernet-Frames, die aus dem externen Netz an das interne Netz gerichtet sind, die als Source-Port die Nummer 14554 aufweisen und als Destination-Port einen beliebigen HTTP-Port (80, 8080) besitzen, müssen untersucht werden. Sie werden verworfen, wenn in ihrer Payload die folgende Signatur enthalten ist: `|116 114 111 106 97 110 101 114|.exe`. Diese Regel unterbindet den Download von bestimmten Dateien:

```
drop tcp $EXTERNAL_NET 14554 → $HOME_NET $HTTP_PORTS
(msg:"BLOCK TROJAN ACCESS";content:"|116 114 111 106 97 110 101 114|.exe")
```

Analyse des SNORT-Regelsatzes: Im diesem Abschnitt werden vier SNORT-Regelwerke untersucht. Die gesamten Regelwerke werden nach den folgenden Punkten kategorisiert:

1. Alle auskommentierten (deaktivierten) Regeln werden in die Klasse „Commented“ extrahiert.
2. Alle Regeln, die das Schlüsselwort `pcre` enthalten, werden in die Klasse „PCRE“ extrahiert.
3. Die Regeln, die mehrere `uricontent`-Schlüsselwörter enthalten, werden in die Klasse „Complex_URI“ extrahiert.
4. Regeln, die mehrere `content`-Schlüsselwörter enthalten, werden in die Klasse „Complex“ extrahiert.
5. Alle Regeln, mit genau einem `uricontent`-Schlüsselwort, werden in die Klasse „Simple_URI“ extrahiert.
6. Die Regeln, mit genau einem `content`-Schlüsselwort, werden in die Klasse „Simple“ extrahiert.

Klasse	2.8.5.3	DB 100530	2.8.6.0	2.8.6.1
Gesamt	15.711	15.225	15.806	15.806
Commented	11.709	9.709	11.759	11.759
PCRE	1.863	2.248	1.871	1871
Complex_URI	551	138	0	0
Complex	825	616	994	994
Simple_URI	777	0	0	0
Simple	1.069	1.211	1.075	1.075
Others	107	113	107	107
HW-Compliant	**1.347**	**2.886**	**1.354**	**1.354**

Tabelle 4.15: Klassifizierung verschiedener SNORT-Datenbanken

7. Alle verbleibenden Regeln werden in der Restklasse „Others“ erfasst.

Die Ergebnisse der Klassifizierung verschiedener SNORT-Datenbanken ist in Tabelle 4.15 dargestellt. Alle Regeln, in denen die Schlüsselwörter `content` und `uricontent` ohne Konkatenation auftreten, sind für das SecAN-IDS geeignet. Sie werden in der Klasse HW_Compliant zusammengefasst. Zu beachten ist, dass Port-Ranges in separate Regeln aufgeschlüsselt werden. Somit existieren mehr HW_Compliant Regeln als die Summe der Regeln der Klassen „Simple_URI“ und „Simple“.

Nachdem die HW_Compliant-Regeln identifiziert wurden, werden sie für den Einsatz in einem erweiterten BF-IDS in vier Klassen aufgeteilt. Die Klassen (Any→Any, Port→Any, Any→Port, Port→Port) umfassen die Port-Abhängigkeiten der zu filternden Signaturen, ohne die sich die Qualität des IDS unnötig verschlechtern würde. Diese Metadaten werden gemeinsam mit den Signaturen in separaten Bloom-Filtern gespeichert. Durch diese Erweiterungen entstehen erweiterte BF. Tabelle 4.16 stellt die Verteilung HW_Compliant-Regeln innerhalb der vier Klassen dar.

Längenverteilung der SNORT-Datenbank *DB 100530*: Für die spätere Implementierung wurde die SNORT-Datenbank *DB 100530* ausgewählt, da sie die meisten Hardware-geeigneten Signaturen aufweist. Alle Port-Ranges bis 100 Ports wurden berücksichtigt. Ferner wurde die maximale Signaturlänge auf 30 Byte begrenzt. Zum einen stellt Abbildung 4.19 nach IDS-Klassen längenseparierte alle Signaturen bis 30 Byte dar. Zum

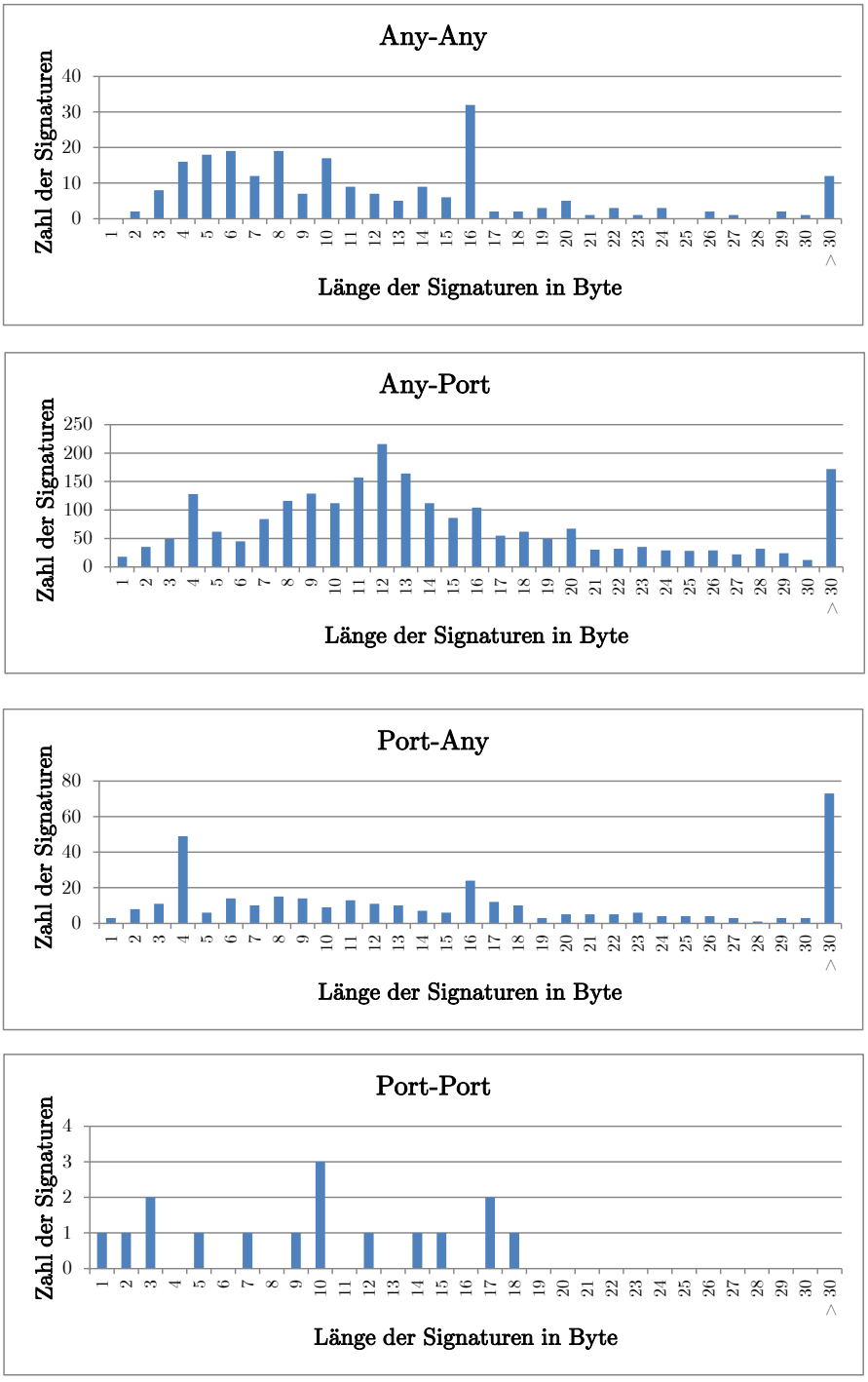


Abbildung 4.19: Längenverteilung von SNORT-Signaturen nach IDS-Klassen

IDS-Klasse	2.8.5.3	DB 100530	2.8.6.0	2.8.6.1
Any→Any (AA)	212	224	212	212
Port→Any (PA)	325	351	332	332
Any→Port (AP)	794	2.295	794	794
Port→Port (PP)	16	16	16	16

Tabelle 4.16: Verteilung der Hardware-kompatiblen Regeln in vier IDS-Klassen

anderen sind in der Kategorie *>30* alle Signaturen größer 30 Byte zusammengefasst und ebenfalls aufgeführt. Von allen 2.886 Hardware-geeigneten Regeln können aufgrund Hardware-Beschränkungen auf der Zielplattform ca. 91,1 % unterstützt werden.

Ausgenommen die Kategorie *>30* stellt Abbildung 4.19 den Füllstand aller 120 Bloom-Filter dar. Dabei entfallen ca. 80 % aller Hardware-geeigneten Regeln auf die IDS-Klasse *AP*. Damit ist die IDS-Klasse *AP* die Klasse, in der alle BF am stärksten ausgelastet sind. Gleichzeitig besitzt sie den BF (AP-L:12) mit der höchsten Anzahl zugeordneter Signaturen (216).

Alle weiteren IDS-Klassen mit insgesamt 90 Bloom-Filtern teilen sich die verbleibenden 20 % Hardware-geeigneter Signaturen. Besonders auffällig ist die IDS-Klasse *PP*. Ihre 30 BF werden nur ca. 0,6 % aller Hardware-geeigneten Signaturen beinhalten.

Mapping der SNORT-Regeln in die Bloom-Filter: Port-Zugehörigkeiten und Protokollinformationen sind wichtige Metadaten, die der Signatur zugeordnet werden müssen, um ein BF-basiertes IDS von höchster Qualität zu gewährleisten.

Die zusätzlichen Metadaten müssen gemeinsam mit der entsprechenden Signatur in das BF-System integriert werden. Dazu wird jeder Parameter und jede Signatur mit k-Hash-Funktionen zu k-Indizes komprimiert. Anschließend werden korrespondierende Indizes via XOR-Funktion zu einem neuen Index kombiniert. Prinzipiell lässt sich dieses Verfahren mit beliebig vielen Parametern realisieren. Die Qualität des erweiterten BFs in Bezug auf die False Positive Rate gegenüber dem Standard-Bloom-Filter-Verfahren kann nur gewahrt werden, wenn folgende Eigenschaft gilt:

Die errechneten Indizes sind ideal statistisch gleich verteilt, wenn bei p Parametern je Signatur und k-Hash-Funktionen $(p + 1) \cdot k$ unabhängige Hash-Funktionen definiert werden können.

Werden die errechneten Indizes anschließend XOR-verknüpft, bleibt die ideale, zufällige Gleichverteilung ebenfalls erhalten. Abbildung 4.20 zeigt vereinfacht den Mechanismus, mit dem sowohl die Signatur als auch die Metadaten mit je zwei Hash-Funktionen in den erweiterten BF integriert werden. Nach der Vorschrift aus Abbildung 4.20 werden je

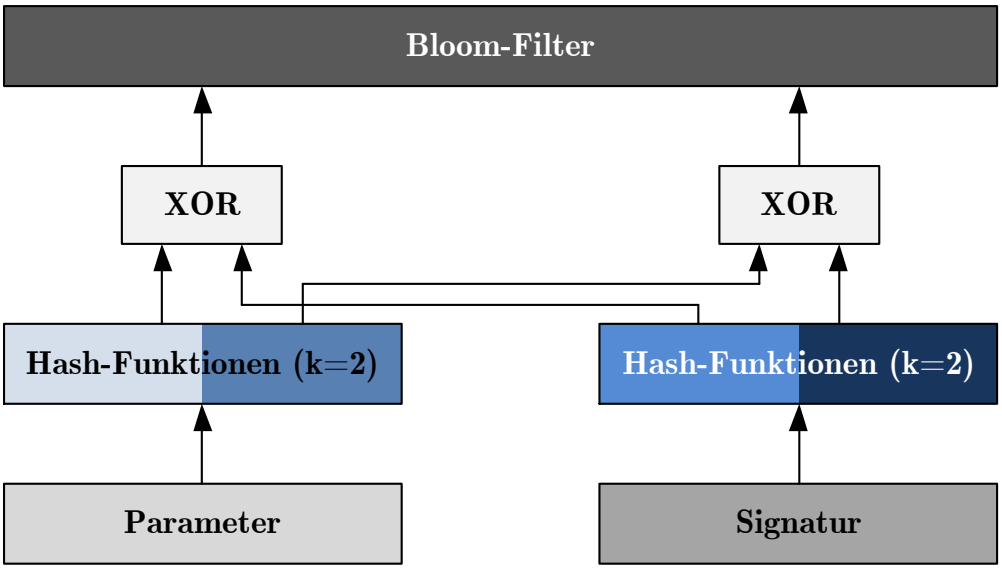


Abbildung 4.20: Integration von Signatur und Metadaten in den erweiterten Bloom-Filter

nach IDS-Klasse unterschiedlich viele Parameter in die Berechnung der BF-Indizes mit einbezogen. Da die unspezifische Port-Angabe `Any` keinen Informationsgehalt besitzt, wird für die IDS-Klasse neben der Signatur nur das Protokoll integriert. Für die beiden IDS-Klassen `PA` und `AP` wird jeweils die Port-Angabe mit dem Protokoll und der Signatur kombiniert. Lediglich die IDS-Klasse `PP` nutzt für die Berechnung der Indizes die volle Parameteranzahl (beide Port-Nummern, das Protokoll und die Signatur). Somit werden für jede Signatur-Länge vier BF bereitgestellt. Bei einer Begrenzung der unterstützten Signatur-Längen auf 30 Byte werden insgesamt 120 Bloom-Filter implementiert.

4.7.4 Realisierung

In diesem Abschnitt wird die Realisierung der IDS-Filterstufe beschrieben. Sie beginnt mit der qualitativen Bewertung und Dimensionierung des zu implementierenden erweiterten Bloom-Filter-Clusters. Anschließend werden alle verbleibenden Module der

IDS-Filterstufe in Abbildung 4.23 dargestellt. Diese sind: *Sliding-Window-Cluster*, *Meta-Kompressor*, *Kompressionsnetzwerk* und *Simple-Match-Analyzer*.

4.7.4.1 FPGA-basierte Bloom-Filter-Realisierung

Die BF bilden das Kernstück der IDS-Filterstufe. Dabei ist die Qualität der Filter - die False Positive Rate (f) - von drei verschiedenen Einflussfaktoren abhängig (vgl. Abschnitt 4.7.2.5):

$f = (1 - e^{-\frac{kn}{m}})^k$

Die False Positive Rate f: Da jeder Filter zu 99,999 % fehlerfrei arbeiten soll, beträgt die FPR 0,001.

Anzahl der Hash-Funktionen k: Die Anzahl der Hash-Funktionen k ist eine Hardware-abhängige Komponente. Implementiert werden die Bloom-Filter in BRAM-Blöcke. Jeder Block kann in einem True-Dual-Port-RAM-Modus betrieben werden. Der gesamte 14 Bit breite Adressraum lässt sich über zwei Adressleitungen je BRAM-Block adressieren. Um Inter-Index-Kollisionen der Hash-Funktionen im BF zu vermeiden, wird ein virtuelles Adressraum-Splitting durchgeführt. Das bedeutet:

- Das MSB der Adressleitung 1 erhält den logischen Wert „1“ und adressiert damit die obere Hälfte des BRAM-Blockes.
- Das MSB der Adressleitung 0 erhält den logischen Wert „0“ und adressiert damit die untere Hälfte des BRAM-Blockes.

Gleichzeitig werden die Hash-Ergebnisse auf 13 Bit reduziert. Abbildung 4.21 stellt das Prinzip grafisch dar.

Die Zielarchitektur verfügt über 296x18 Kbit BRAM-Blöcke. Unter Berücksichtigung der bereits zugeordneten Hardware-Ressourcen verbleiben für die 120 zu implementierenden BF ca. 250 BRAM-Blöcke. Wird eine zusätzliche Reserve von 10 BRAM-Blöcken für weitere IDS-Hardware-Module zurückgehalten, können jedem BF 2 BRAM-Blöcke zugeteilt werden. Dementsprechend wird jeder BF über vier Hash-Funktionen adressiert.

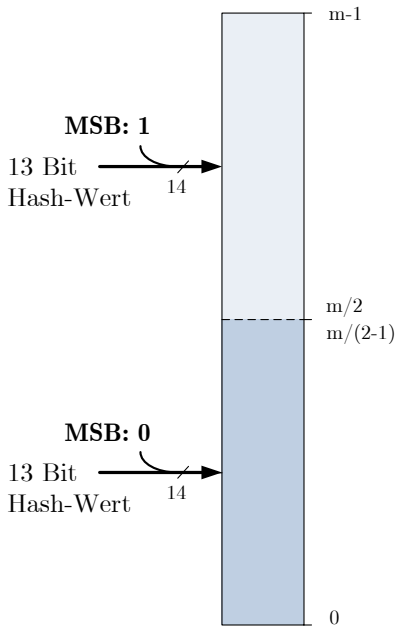


Abbildung 4.21: Virtuelles Adressraum-Splitting für True-Dual-Port-BRAM-Blöcke

Bit-Breite der Bloom-Filter m: Wie zuvor erläutert, führt das virtuelle Adressraum-Splitting zu einer Reduktion der Adressleitung um ein Bit. Somit stehen jedem virtuellen Adressraum nun 13 Bit (2^{13} Adressen) zur Verfügung. Aufgrund der Aufteilung eines BFs über zwei BRAM-Blöcke (b) besitzt jeder BF vier virtuelle Adressräume. Der resultierende finale Adressraum je BF besitzt somit: $m = 2 \cdot 2^{13} \cdot b = 2^{14} \cdot b = 32768\ Bit$.

Anzahl der Signaturen n: Wird die Formel für die Berechnung der FPR aus Abschnitt 4.7.2.5 zugrunde gelegt und nach n - der Anzahl der kodierbarer Signaturen - umgestellt, so ergibt sich die folgende Formel:

$n = -\frac{m}{k} \cdot ln(1 - \sqrt[k]{f})$

Mit $m = 2^{14} \cdot b$, $k = 2 \cdot b$ und $f = 0{,}001$ gilt:

$$n = -\frac{2^{14} \cdot b}{2 \cdot b} \cdot ln(1 - \sqrt[2 \cdot b]{0,001}) \tag{4.8}$$

Für das beschriebene Szenario bedeutet das, dass nach Formel 4.8 maximal 1.604 Signaturen in einem BF abgespeichert werden dürfen, ohne die FPR von 0,001 zu überschreiten. Werden weniger Signaturen in den BF kodiert, verbessert sich die FPR. Erst wenn

mehr als 1.604 Signaturen in den BF kodiert werden, verschlechtert sich die Qualität der Filter auf unter 99,999 %.

Jedoch ist aus Abbildung 4.19 über die Längenverteilung der Hardware-geeigneten Signaturen der SNORT-Datenbank *DB100530* klar geworden, dass diese Kapazität nicht erreicht wird. Der Bloom-Filter *AP-L:12* besitzt gerade einmal 216 gespeicherte Signaturen. Ausgehend von Formel 4.7 beträgt demnach seine korrespondierende FPR:

$$f_{AP-L:12} = (1 - \frac{1}{e^{\frac{864}{2^{15}}}})^4 \approx 4,58 \cdot 10^{-7}$$

Im Mittel entfallen auf jeden Bloom-Filter 24 Signaturen. Ausgehend von diesem Wert beträgt die mittlere FPR:

$$f_{ave} = (1 - \frac{1}{e^{\frac{96}{2^{15}}}})^4 \approx 7,32 \cdot 10^{-11}$$

Abschließend gibt Abbildung 4.22 einen Überblick über die Zahl der programmierbaren Signaturen und den Zuwachs an Signaturen, die in BF unterschiedlicher Länge programmiert werden dürfen, wenn eine FPR von 0,001 eingehalten werden soll. Die blaue Kurve zeigt die Zahl der programmierbaren Signaturen gegenüber den benötigten BRAM-Blöcken. Darüber hinaus sind auf der oberen Abszisse die exakte Anzahl von Signaturen bei gegebenen BRAM-Blöcken aufgetragen. Aufgrund von nicht verfügbaren Hardware-Ressourcen entscheidet sich der Autor für zwei BRAM-Blöcke je BF, obwohl die höchste Ausbeute bei drei BRAM-Blöcken liegen würde, wie die rote Kurve in Abbildung 4.22 zeigt. Über die Grenze von drei BRAM-Blöcken hinaus fällt der Zuwachs an Signaturen wieder. Der reduzierte Gewinn lässt sich mit dem Verhältnis aus der Anzahl der verwendeten Hash-Funktionen, der bereitgestellten Bit-Breiten sowie der Anzahl gespeicherter Signaturen erklären.

Durch das Speichern verschiedener Signaturen mit nur einer Hash-Funktion entsteht eine Vielzahl von 1-en im BF, wobei jede Signatur genau eine 1 erzeugt bzw. bestätigt, wie Abbildung 4.17 beschreibt. Innerhalb der Detektionsphase führt bereits das Auffinden einer 1 im BF zur Anzeige eines Matches, welches durch einen Match-Analyzer verifiziert werden muss. Entgegengesetzt dazu existiert die Möglichkeit der Verwendung einer Vielzahl von Hash-Funktionen. Im ungünstigsten Fall befindet sich dann an jeder Bit-Position eine 1 im BF, was ebenfalls ein ständiges Aufrufen des Match-Analyzers zur Folge hätte. In beiden Fällen wird das BF-Verfahren ad absurdum geführt. Es kommt also auf das richtige Verhältnis von Bit-Breite des BFs, Anzahl der Hash-Funktionen und Anzahl der gespeicherten Signaturen an.

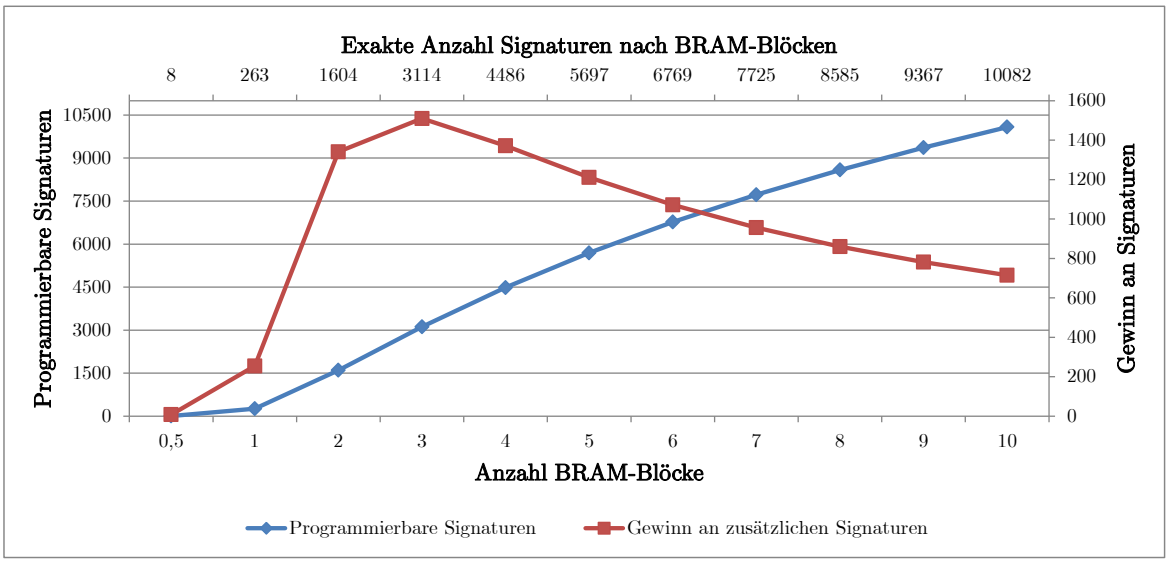


Abbildung 4.22: Anzahl maximal programmierbarer ID-Signaturen je Bloom-Filter gegenüber dem Gewinn an Signaturen mit steigender BRAM-Blöcken

H3-Hashing: Verschiedene Hash-Verfahren wurden in Abschnitt 4.3.2.3 vorgestellt und auf ihre Effizienz und Hardware-freundliche Umsetzung untersucht. Prinzipiell können für eine Bloom-Filter-Implementierung beliebige Hash-Funktionen zum Einsatz kommen. Für qualitativ hochwertige BF müssen die Anforderungen nach Unabhängigkeit und idealer Zufälligkeit erfüllt sein. Hinzu kommt, dass für die 120 BF ein Schema für eine ganze Familie von Hash-Funktionen abgeleitet werden muss. Ungeachtet der Hardware-ressourcenschonenden Implementierung von CRC kann dieses Verfahren nicht gewählt werden, da nicht genügend Generatorpolynome existieren, welche die geforderte Index-Breite von 13 Bit je BF erfüllen. Aus diesem Grund muss auf die aufwändigere Implementierung der von Wegman und Carter [WC81] vorgestellten H3-Klassen (vgl. Formel A.4) zurückgegriffen werden.

Bei der in Abschnitt 4.3.2.3 vorgestellten Abbildung $h_q(x) : X \longmapsto H$ handelt es sich um eine lineare Transformation. Dadurch können gleich verteilte Hash-Werte erzeugt werden, wenn echte Zufallszahlen $q(i)$ verwendet werden, die einer statistischen Gleichverteilung unterliegen. Ferner lassen sich beliebig viele Hash-Funktionen erzeugen, wenn die Konstanten q variiert werden, wodurch die Forderung für die erweiterten Bloom-Filter erfüllt wird. Statistisch gleich verteilte, echte Zufallszahlen werden auf der Web-Seite

www.random.org[16] bereitgestellt. Es handelt sich um Daten, die aus atmosphärischem Rauschen generiert werden.

4.7.4.2 Implementierung des Filterkerns

Der Filterkern ist als Makro-System implementiert und besteht aus einer Reihe von Modulen (vgl. Abbildung 4.23). Zunächst werden die Ethernet-Daten in ein Sliding-Window-Cluster geleitet. Dieses separiert unterschiedlich lange Muster aus dem Datenstrom und leitet sie an das *Kompressionsnetzwerk* weiter. Beim *Kompressionsnetzwerk* handelt es sich um die Hardware-Implementierung des H3-Hash-Netzwerkes. Synchron mit dem Eingangsdatenstrom erreichen die aktuellen Meta-Daten (Protokoll und Portnummern) den Filterkern. Sie bleiben über die gesamte Dauer des Ethernet-Frames konstant, werden über den Meta-Kompressor auf H3-Hash-Werte abgebildet und anschließend ebenfalls dem *Kompressionsnetzwerk* zur Verfügung gestellt.

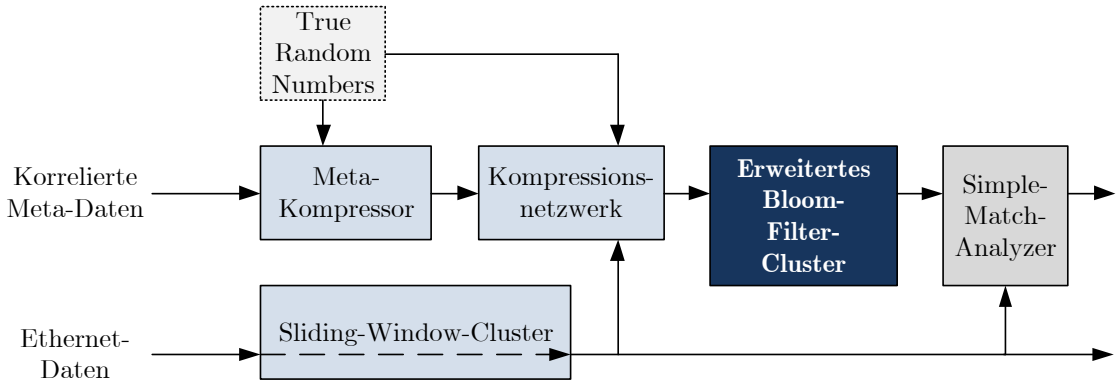


Abbildung 4.23: Blockschaltbild des IDS-Filterkerns

Ethernet- und Meta-Daten werden über das *Kompressionsnetzwerk* zu Index-Werten für das erweiterte Bloom-Filter-Cluster kombiniert. Das Bloom-Filter-Cluster überprüft daraufhin die Mengenzugehörigkeiten der komprimierten Eingangsmuster mit den Regeln des IDS-Regelwerkes.

Die Ergebnisse der Überprüfungen werden gemeinsam mit dem Eingangsdatenstrom an den *Simple-Match-Analyzer* übergeben. Der implementierte *Simple-Match-Analyzer* verwirft den Eingangsdatenstrom, sobald ein beliebiges Match angezeigt wird. Liefert

[16] True-Random-Number Generator: http://www.random.org/integers/

das erweiterte Bloom-Filter-Cluster hingegen keine Übereinstimmung mit den Vorgaben des Regelwerkes, wird der Datenstrom an den Systemausgang geleitet. Weiterführende Informationen zur Realisierung der Module *Sliding-Window-Cluster*, *Meta-Kompressor*, *Kompressionsnetzwerk* und *Simple-Match-Analyzer* sind im Anhang C.6.2 zu finden.

4.7.5 Clocking

Wie in Abschnitt C.1 beschrieben arbeitet das SecAN-Framework mit einer internen Datenbreite von 4 Byte/Takt. Die IDS-Filterstufe verarbeitet die Eingangsdaten mit einer Datenbreite von 1 Byte/Takt. Es existieren zwei Möglichkeiten, um diese Diskrepanz zu lösen (vgl. Abbildung 4.24) **Erste Möglichkeit:** In einem QuadCore-System untersu-

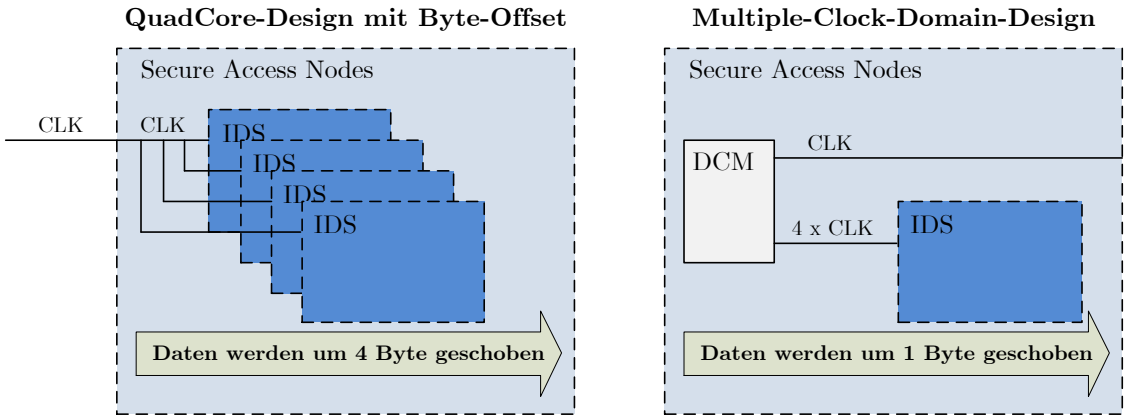


Abbildung 4.24: Vergleich QuadCore-Design vs. Multi-Clock-Domain-Design

chen vier Filterkerne die Eingangsdaten mit einem Offset von je 1 Byte. Ferner werden die Daten mit jedem Takt um vier Byte geschoben. Der Vorteil der Lösung liegt in einer gemeinsamen Clock-Domain mit dem SecAN-Framework. Eine zweite Clock-Domain kann, muss jedoch nicht, für die Ethernet-Controller existieren. Wird das Gesamtsystem mit 125 MHz betrieben, wird auch nur eine Clock-Domain benötigt. Diese Lösung fordert, dass der Hardware-Aufwand vervierfacht wird. Da jedoch die 120 Bloom-Filter eines Filterkerns bereits mehr als 81 % der BRAM der Zielplattform benötigen, lässt sich diese Clocking-Variante nicht realisieren.

Zweite Möglichkeit: Es existieren drei Clock-Domains. Die Ethernet-Controller werden mit einer 125 MHz Clock betrieben. Darüber hinaus werden für das SecAN-Framework und den Filterkern zwei Clock-Domains benötigt, wobei die Clock des Filter-

kerns mit einer vierfach höheren Geschwindigkeit gegenüber der SecAN-Clock betrieben werden muss.

Um den geforderten Durchsatz von 1 Gbit/s bei einer internen Verarbeitungsdatenrate zu gewährleisten, ist eine Eingangsfrequenz von 31,25 MHz notwendig. Die Zielplattform bietet die vier verschiedenen Taktfrequenzen 27 MHz, 33 MHz, 100 MHz und 200 MHz an. Aufgrund der geforderten Eingangsfrequenz von 31,25 MHz ist die einzige relevante Frequenz 33 MHz. Diese lässt sich leicht mit einem Digital Clock Manager (DCM) vervierfachen, sodass der Filterkern mit 132 MHz betrieben werden muss. Da beide Clock-Signale des zweiten DCM phasengleich sind, kann auf Synch-FIFOs zwischen SecAN-Framework und Filterkern verzichtet werden. Stattdessen wird ein Schieberegister verwendet, bei dem die Daten mit der niedrigeren Frequenz hineingeschrieben und mit der vierfach höheren Frequenz wieder herausgelesen. Anschließend werden sie in den Filterkern geleitet. Abbildung 4.24 zeigt das Clocking-Schema für das IDS-System.

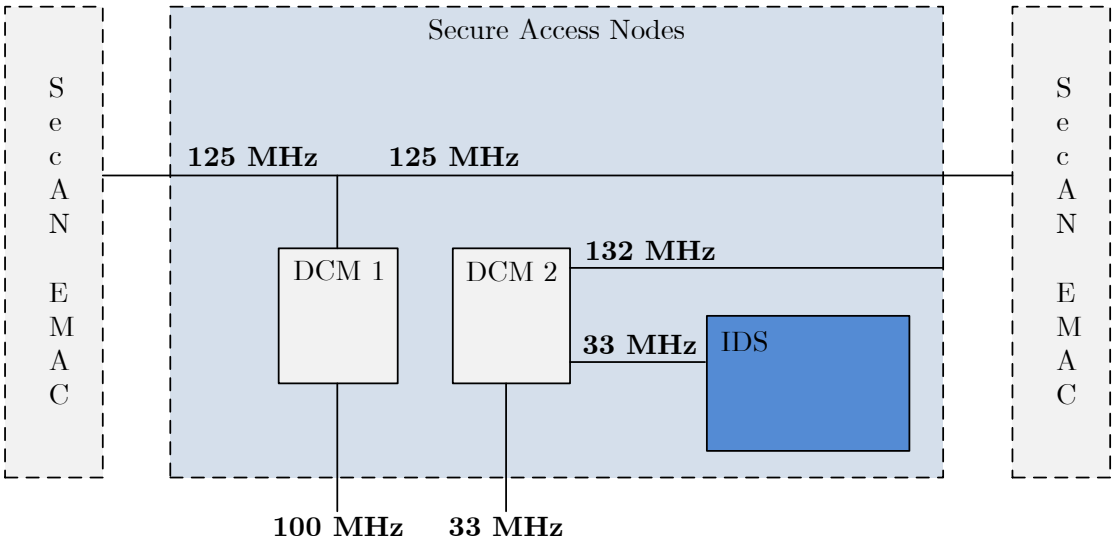


Abbildung 4.25: Triple-Clock-Design der IDS-Filterstufe

4.7.6 Test und Ergebnisse

In diesem Abschnitt wird das entwickelte Hardware-System einem ausführlichen Test unterzogen. Der Test erfolgt in der Simulationsumgebung ModelSim (vgl. Abschnitt D.4) sowie auf der Zielplattform (vgl. Abschnitt D.1) selbst.

Variation	Signatur	erkannt	fehlerfreie Funktion
Kleinschreibung	ringelblume	ja	ja
Großschreibung	RINGELBLUME	ja	ja
Normale Schreibweise	Ringelblume	ja	ja
Alternierend	RiNgElBlUmE	ja	ja
Infix-Position	Ringelblumenvase	ja	ja
Bindestrich	Ringel-Blume	nein	ja

Tabelle 4.17: Single-Signatur-Test der ID-Filterstufe mit Variation der Signatur

4.7.6.1 Simulativer Test der IDS-Filterstufe

Single-Signature-Test: Mittels der Simulationsumgebung ModelSim wird zunächst die Signatur „Ringelblume“ für alle vier IDS-Klassen konfiguriert. Beispielhaft für die Klasse *Any-Any* lautet die Regel:

```
alert tcp $EXTERNAL_NET any → $HOME_NET any
(msg:"Testsignatur";content:"Ringelblume")
```

Die vier Klassen werden in Byte-Code übersetzt und in das ID-System konfiguriert.

Anschließend werden 10.000 generierte Ethernet-Frames aus der Quelldatei gelesen und an die Simulation übergeben. Dabei weisen 10,02 % aller Frames die Test-Signatur in verschiedenen Variationen auf (vgl. Tabelle 4.17). Sie wurden zufällig über alle generierten Daten gestreut. Alle von der Simulation generierten Konsolen-Ausgaben werden in eine Datei umgeleitet. Nachdem das IDS alle 10.000 Ethernet-Frames verarbeitet hat, ergab die Analyse der Zieldatei die in Tabelle 4.17 dargestellten Ergebnisse.

Sowohl die Konfigurationsdaten der vier Bloom-Filter, als auch alle Frames konnten erfolgreich vom ID-System verarbeitet werden. Die ersten fünf Test-Signaturen aus Tabelle 4.17 wurden 835-mal erkannt. Lediglich die 6. Test-Signatur, die 167-mal vertreten war, wurde nicht als suspekt markiert. Insgesamt entsprechen die Ergebnisse der Tests den Erwartungen des Autors sowie einer fehlerfreien Funktionalität des ID-Systems.

Multiple-Signatur-Test: Um die Zusammenarbeit aller 120 Bloom-Filter nachzuweisen, wurde der *Single-Signature-Test* dahingehend modifiziert, dass alle Bloom-Filter

mit einer speziellen Signatur programmiert wurden. Dieselben Eingangsdaten wie beim *Single-Signature-Test* wurden in das System eingespielt. Das IDS zeigt keine Anomalien auf und bestand den Test ebenfalls erfolgreich.

Reverse-Brute-Force-Test: Die beiden Tests - Single-Signature und Multiple-Signature - weisen jedoch eine Schwäche auf. In beiden Fällen kann keine Unterscheidung zwischen einem tatsächlichen Match und einem *False Positiv* getroffen werden, sodass von einer fehlerfreien Funktionalität lediglich für die programmierten Signaturen ausgegangen werden kann. Mit der **Reverse-Brute-Force-Methode** lässt sich dieser Mangel beheben.

Die Herausforderung besteht im Auffinden von Signaturen, die beim H3-Hashen Indizes erzeugen, die mit Indizes von IDS-Regeln übereinstimmen, jedoch keine Indizes von IDS-Regeln darstellen. Lassen sich solche Signaturen provozieren, handelt es sich definitiv um *False Positives*. Das kontrollierte Testen aller Kombinationsmöglichkeiten von Signaturen und Indexkombinationen ist sehr komplex. Sollen bspw. alle möglichen Kombinationen von 30 Byte langen Signaturen getestet werden, so ergibt dies:

$30\ Byte = 240\ Bit \Longrightarrow 2^{240}\ Eingangskombinationen\ (Signaturen)$

Da der gesamte Adressraum eines BF über vier mal 13 Bit angesprochen wird, existieren:

$2^{52}\ Adresskombinationen.$

Somit wird mit 2^{240} verschiedenen Signaturen gegenüber 2^{52} unterschiedlichen Adresskombinationen eine kontrollierte Lösung des Problems extrem zeitaufwändig. Wird das Bloom-Filter-Prinzip hingegen umgekehrt, ist es möglich, dass anstehende Problem effizient zu lösen. Der folgende initialer Gedanke zeigt den Lösungsweg auf:

In einer Bibliothek, die alle deutschsprachigen Bücher beinhaltet, wird es mit sehr hoher Wahrscheinlichkeit die folgende Zeichenkombination `Di?uA()St#r°3*2N$§s4"JH$§(‘$=` in keinem Werk geben. Dieses Prinzip lässt sich auf Signaturen in Datenpaketen übertragen, wenn es sich nicht um verschlüsselte Verbindungen handelt. Lassen sich weitere Zeichenketten wie im Beispiel dargestellt bilden, können diese zur Konfiguration der Bloom-Filter benutzt werden. Alle im Test angezeigten Matches wären demnach echte *False Positives*. In fünf Schritten wird der „Reverse-Brute-Force-Test“ durchgeführt:

1. Zunächst wird unverschlüsselter Datenverkehr für die spätere Injektion des IDS aufgezeichnet.

2. Entsprechend der geforderten FPR werden mit Hilfe eines deutschsprachigen Wörterbuches ausreichend viele Signaturen erzeugt, die den Anforderungen des „Reverse-Brute-Force-Tests“ entsprechen. Dazu werden Wörter einerseits rekombiniert und andererseits mit Präfix-, Infix- und Suffix-Silben modifiziert.
3. Zur Kontrolle wird im aufgezeichneten Datenverkehr nach den erzeugten Signaturen gesucht. Gefundene Signaturen werden verworfen und neue Signaturen generiert. Dieser Vorgang wird solange wiederholt, bis keine generierte Signatur im aufgezeichneten Datenverkehr vorhanden ist.
4. Anschließend erfolgt die Konfiguration der Bloom-Filter.
5. In der Simulation wird der aufgezeichnete Datenverkehr in das Testsystem injiziert. Jeder Match entspricht einem echten *False Positive*. Anhand der programmierten und überprüften Signaturen lässt sich die Qualität der Bloom-Filter bestimmen.

Durch Variationen der Anzahl der Signaturen je Bloom-Filter lassen sich unterschiedliche False Positive Rates testen. Jeder Test wurde sechsmal mit unterschiedlichen Eingangsdaten wiederholt. Die Ergebnisse der Mittelwertbildungen, die praktische und theoretische False Positive Rates sowie die Anzahl der markierten Pakete sind für alle Tests in Tabelle 4.18 dargestellt.

Der Vergleich zwischen theoretisch berechneten und praktisch gemessenen *False Positives* bestätigt die Bloom-Filter-Theorie. Zusätzlich zeigt Abbildung 4.26 die Differenz zwischen beiden False Positive Rates bis zu einem maximalen Füllstand der Bloom-Filter von 1.604 Signaturen (FPR = 1 ‰). Marginale Abweichungen entstehen durch statistische Streuungen und können durch zusätzliche Messreihen weiter minimiert werden. Obwohl die Bloom-Filter des entwickelten IDS mit einer Genauigkeit von 99,999 % (bei 1.604 Signaturen) konzipiert wurden, werden durchschnittlich 33 % aller Pakete markiert (vgl. Tabelle 4.18). Wie kommt es zu derart vielen Markierungen?

Eine zu untersuchende Dateneinheit von 970 Byte besitzt insgesamt 1.000 Signaturen. Dabei entstehen die 30 zusätzlichen Signaturen durch das sukzessive Füllen des Sliding-Windows. Bei einer FPR von 1 ‰ wird also jede Dateneinheit mit einer Länge von 970 Byte durchschnittlich ein Mal markiert. Hinzu kommt, dass das Gesamtsystem aus 120 Bloom-Filtern besteht, sodass bei gleichmäßigem Füllstand der BF die FPR um den Faktor 120 höher liegt.

Die Lösung des Problems liegt in verschärften Anforderungen an das IDS. Bei einer FPR von $\frac{1}{120}$ ‰ dürfen maximal 452 Signaturen je BF programmiert werden. In Abschnitt

Signaturen pro Bloom-Filter	Untersuchte Muster (Mittelwert)	False Positives (Mittelwert)	FPR [‰] (praktisch)	FPR [‰] (theoretisch)	Markierte Pakete [%] (Mittelwert)
1	120.597	0,00	0,00	0,00	0
25	120.597	0,00	0,00	0,00	0
50	120.597	0,00	0,00	0,00	0
100	120.597	0,00	0,00	0,00	0
500	120.597	2,80	0,02	0,01	1
1.000	120.597	19,00	0,16	0,17	8
1.500	120.597	89,40	0,75	0,78	28
1.604	120.597	113,20	0,95	1,00	33
2.000	120.597	259,20	2,15	2,20	49
5.000	120.597	5.348,00	44,39	43,56	99
7.500	120.597	15.853,80	131,51	129,34	100
10.000	120.597	30.047,60	249,05	247,00	100
20.000	120.597	83.043,20	688,18	694,72	100
50.000	120.597	119.632,20	992,00	991,09	100

Tabelle 4.18: Ergebnisse der Bloom-Filter-Qualitätsanalyse

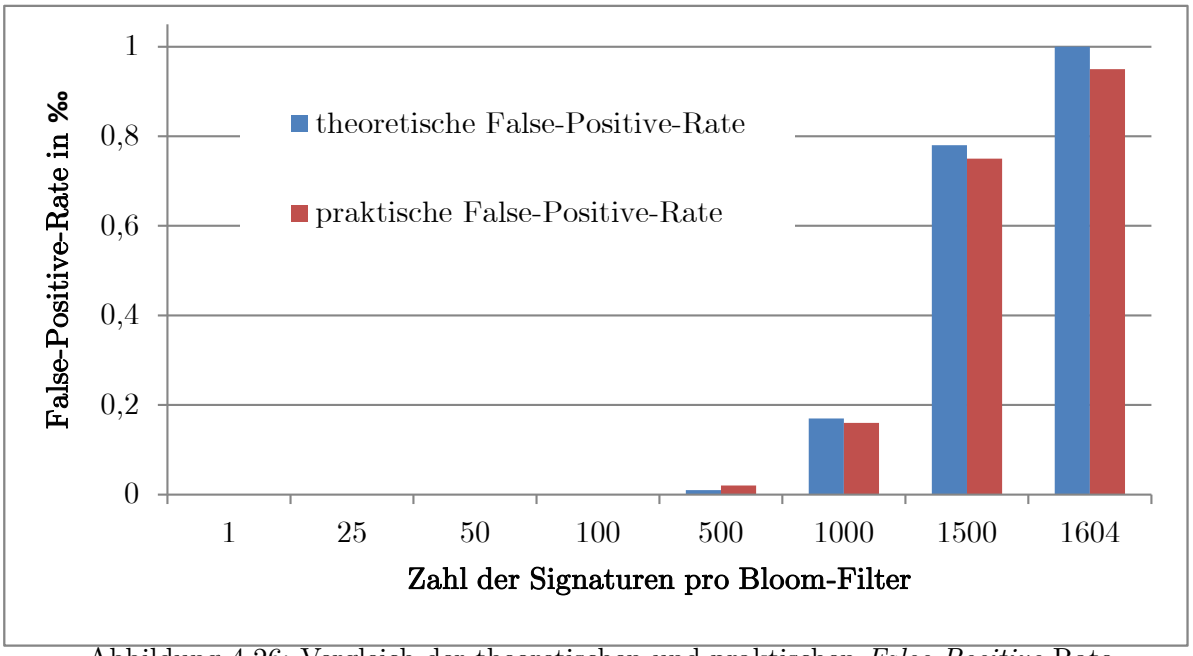


Abbildung 4.26: Vergleich der theoretischen und praktischen *False Positive*-Rate

4.7.3.2 wurden vier verschiedene SNORT-Datenbanken untersucht. Dabei wurde festgestellt, dass die Datenbank „DB 100530“ die meisten Hardware-geeigneten Signaturen aufweist und dass die Streuung der Signaturen zwischen 0 und 216 liegt. In der Konsequenz bedeutet das, dass die BF weniger als die Hälfte der maximal erlaubten Signaturen aufweisen und somit selbst die verschärfte Anforderung genügend Freiheitsgrade bietet, um eine systemweite FPR von unter 1 ‰ zu erreichen. Damit hat der Match-Analyzer, der *False Positives* zu 100 % eliminiert, mindestens 1000 Takte Zeit. Somit ist es möglich, dass der Match-Analyzer bei der gegenwärtigen Implementierung und verwendeter SNORT-Datenbanken auf einen externen Speicher (SRAM, DDR) zur Verifikation der Matches zurückgreifen kann.

4.7.6.2 Hardware-Test der IDS-Filterstufe

Der Test auf der Zielplattform beabsichtigt, den funktionalen Nachweis des implementierten IDS zu erbringen. Dazu werden dieselben aufgezeichneten bzw. generierten Verkehrsmuster, wie aus Abschnitt 4.7.6.1 bekannt, genutzt. Die Konfiguration der Bloom-Filter erfolgt in der prototypischen Implementierung ebenfalls mittels Ethernet-Frames. Alle Verkehrsmuster befinden sich in einer Wireshark-Dump-Datei und können mit dem

Ressourcenbedarf	Genutzt	Verfügbar	Auslastung
Anzahl Slice Register	15266	44.800	34,1 %
Anzahl Slice LUTs	16248	44.800	36,3 %
BlockRam/FIFO	268	296	90,5 %

Tabelle 4.19: Ressourcenbedarf des ID-Systems

Colasoft Packet Player[17] abgespielt werden.

Zur Anzeige erkannter Matches wird das Liquid Crystal Display (LCD)-Display genutzt. Der dort angezeigte Wert wird hexadezimal ausgegeben und besitzt einen Bereich von 0x0 bis 0xFFFFFF (0 bis $2^{24} - 1$). Jeder erkannte Treffer wird durch Erhöhung des Zählerwertes dargestellt. Zu Beginn eines jeden Tests beträgt der Wert des Zählers 0.

Auf der Zielplattform wurde der funktionale Nachweis durch Abspielen der beschriebenen Verkehrsmuster mittels *Colasoft Packet Player* realisiert. Der Link zwischen PC und Zielplattform betrug 1 Gbit/s und erbrachte dieselben Ergebnisse wie die in Abschnitt 4.7.6.1 beschriebenen. Während der Testphase zeigte das System keinerlei Instabilitäten auf.

Ein zusätzlicher Lasttest über 48 Stunden verfolgte das Ziel nachzuweisen, dass das ID-System auch unter voller Auslastung des Links ausfallsicher arbeitet. Mittels des bereits beschriebenen Traffic-Generators [TDT11] wurde der Test mit zufälligen Verkehrsmustern durchgeführt. Auch dieser Test verlief ohne Instabilitäten des Systems.

4.7.6.3 Ressourcenbedarf und Durchsatz

Nachdem alle Testmuster sowohl in der Simulation als auch im Test unter realen Bedingungen erfolgreich verarbeitet werden konnten und das System keine Instabilitäten aufwies, wurde die Hardware auf Geschwindigkeit optimiert und der Ressourcenverbrauch bestimmt. Das auf Geschwindigkeit optimierte IDS benötigt die in Tabelle 4.19 dargestellten Ressourcen.

Nach *Post Place & Route* beläuft sich die Zielfrequenz des IDS-Filterkerns auf 132,013 MHz. Da ein Byte je Takt verarbeitet wird, entspricht dies einem Durchsatz

[17] Colasoft Packet Player: http://www.colasoft.com/packet_player/

von $\approx 1,056\ Gbit/s$. Somit wird der geforderte minimale Durchsatz von 1 Gbit/s eingehalten.

4.7.7 Zusammenfassung

In Kapitel 4.7 wurde ein Hardware-System zur Erkennung von Eindringlingen konzipiert und umgesetzt. Dazu wurden im Vorfeld mehrere Hardware-basierte Ansätze untersucht. Dabei stellte sich heraus, dass Systeme, die für den Mustervergleich auf externe Speicher zurückgreifen, genauso ungeeignet sind wie Systeme, die ihre Vergleichsmuster in einer fixen DFSM-Struktur hinterlegen. Hinzu kommt, dass ausreichend große CAM-Speicher aus den verfügbaren Ressourcen nicht generiert werden können bzw. die Kosten für den IDS-Prototypen durch CAM-Speicher stark erhöht würden. Aus den genannten Gründen wurde die Bloom-Filter-Theorie aufgegriffen und ein System aus 120 Bloom-Filtern, welches ausschließlich lokale Ressourcen benötigt, umgesetzt.

In einer weiteren Untersuchung wurde das frei verfügbare ID/IP-System SNORT analysiert. Im Ergebnis konnten einfache Muster direkt nachgenutzt werden, ohne die Fehlerrate des Hardware-IDS unnötig zu erhöhen. Aufgrund von Hardware-Beschränkungen der Zielplattform wurde eine maximale Signaturlänge von 30 Byte definiert. 91,1 % der Hardware-geeigneten Signaturen besitzen ein Länge $\leq$ 30 Byte und können in das Hardware-IDS integriert werden. Wie in Tabelle D.1 dargestellt, besitzt der nächstgrößere FX100T-FPGA bereits 456 BRAM-Blöcke, wodurch die Integration von bis zu 50 Byte langen Signaturen problemlos realisiert werden kann. Damit können dann ca. 98,3 % der Hardware-geeigneten Signaturen aus der SNORT-Datenbank in das IDS integriert werden.

Für die Implementierung des IDS-Systems wurde weiterhin ein Verfahren entwickelt, das es erlaubt, den Bloom-Filter-Ansatz um zusätzliche Metainformationen zu erweitern. Metainformationen sind Protokolltypen und Portnummern. Diese Informationen werden gemeinsam mit den Signaturen in den 120 Filtern abgelegt. In jedem Bloom-Filter können bis zu 1604 Signaturen hinterlegt werden, wobei eine FPR von 1 ‰ eingehalten wird. Bei einer Gleichverteilung aller Signaturen kann das Regelwerk 192.480 Signaturen umfassen, die mit jedem Systemtakt parallel untersucht werden.

In der Qualitätsanalyse konnte mit Hilfe der „Reverse-Brute-Force“-Methode der praktische Funktionsnachweis erbracht werden. Da jedoch die FPR des Gesamtsystems um den Faktor 120 höher liegt als die FPR der einzelnen Filter, mussten die Bedingungen

nachgeschärft werden. Mit den geschärften Anforderungen sinkt die Anzahl der maximal programmierbaren Signaturen von 1.604 auf 452, um eine systemweite FPR FPR 1 ‰ einzuhalten. Die Analyse der 4 SNORT Datenbanken (vgl. Abbildung 4.16) beweist jedoch, dass der Bloom-Filter mit dem höchsten Belegungsfaktor 216 Signaturen aufnimmt und somit die erhöhten Anforderungen eingehalten werden können. Die systemweite FPR sinkt deutlich unter 1 ‰ und ermöglicht einem Match-Analyzer innerhalb von durchschnittlich 1000 Takten einen Match zu verifizieren.

Durch den Bloom-Filter-Ansatz leidet der SecAN-IDS-Prototyp nicht unter *False Negatives*. Das bedeutet, dass bei der Analyse alle im Datenstrom vorkommenden Signaturen 100%-ig detektiert werden.

Im Vergleich zum „Wu-Manber“-Algorithmus [PKAC08] ist die Effizienz des vorgestellten SecAN-Prototypen nicht von der Signaturlänge abhängig. Hinzu kommt, dass der Durchsatz mit 0,139 Gbit/s deutlich unter den gesteckten Anforderungen liegt. Gleiches gilt für den FPGA-basierter „Simple SNORT Prozessor“ aus [AGGR09]. Im Gegensatz zu [PKAC08] und [AGGR09] übertrifft die in [LXLS09] vorgestellte Lösung den SecAN-Prototypen in Hinsicht Durchsatz um den Faktor 3. Dieser, im Vergleich zum SecAN-Prototypen hohe Durchsatz, kann jedoch nur unter Zuhilfenahme eines zusätzlichen Netzwerkprozessors, zweier SRAMs und eines DRAMs generiert werden. Die beschriebenen Komponenten treiben die Kosten erheblich in die Höhe, sodass diese Lösung nicht für den Einsatz im TZN geeignet ist.

Mit dem in Tabelle 4.19 dargestellten Ressourcenbedarf und dem erreichten Durchsatz von $\approx 1,056$ *Gbit/s* wird das Einhalten der gesteckten Vorgaben aus Abschnitt 3.1 erreicht. In einem abschließenden Stresstest konnte ein stabiles Systemverhalten nachgewiesen werden. Optimierungen und Erweiterungen des vorgestellten SecAN-Prototypen sind im Anhang C.7 zu finden.

Kapitel 5

Zusammenfassung

Die vorliegende Forschungsarbeit widmet sich der Erhöhung der Netzwerksicherheit. Weil der durchschnittliche Internet-Nutzer meist nicht das notwendige netzwerktechnische Wissen besitzt, wird er häufig Opfer von Netzwerkbedrohungen. Somit ist es das Ziel dieser Arbeit, für die beschriebene Zielgruppe, einen deutlichen netzwerkseitigen Security-Mehrwert zu bieten, ohne dass eine aktive Beteiligung erforderlich wird. Zu diesem Zweck wurden drei Hardware-basierte Prototypen für den Einsatz in Teilnehmerzugangsnetzwerken konzipiert und umgesetzt. Diese Prototypen lassen sich einzeln und im Verbund betreiben, wobei der größtmögliche Mehrwert durch den kombinierten Einsatz entsteht. Ein gemeinsam genutztes Framework koordiniert den externen Datenverkehr, übernimmt redundante Aufgaben und versorgt alle Prototypen mit notwendigen Informationen, um mit minimaler Latenz die jeweilige Sicherheitsaufgabe zu bearbeiten. Das SecAN-Gesamtsystem basiert auf der Idee der SDN [Kir13]. Dadurch ist eine sinnvolle Entkopplung von Software und Hardware möglich. Weiterhin sei angemerkt, dass die ehemalige *Broadband Access Division* der Nokia Siemens Networks GmbH & Co. KG in Greifswald durch Inspiration und Unterstützung zum Gelingen der vorliegenden Forschungsarbeit beitrugen. Initiiert durch die *Broadband Access Division* läuft seit 2010 für den Forschungsschwerpunkt SecAN-Firewall ein europäisches Patentverfahren (vgl. [Sch10]).

5.1 Firewall im Secure Access Node

In Abschnitt 4.5 wurde die SecAN-Firewall-Lösung vorgestellt. Dazu wurden Software-Lösungen, Hardware-basierte Netzwerk-Firewalls ([AD11]) und eine Lösung für den Einsatz im TZN ([KL12]) untersucht. Dabei konnte festgestellt werden, dass Lösungen, die

auf *General Purpose* Prozessoren zurückgreifen, für den Einsatz im TZN ungeeignet sind.

Zum einen erzeugen sie hohe CPU-Last (vgl. [SK11, YY10]), was sich in einer negativen QoE für den Nutzer auswirkt. Zum anderen erzielen dieselbe TZN-Lösungen nur einen Durchsatz von ca. 13,5 Mbit/s (vgl. [KL12]). Teilweise kommen kostenintensive CAMs wie in [YY10] zum Einsatz. Besonders negativ sind Fehlimplementierungen, die bei DDoS-Attacken (Ping) in jedem Fall zum Systemabsturz führen und somit vollkommen ungeeignet sind (vgl. [SK11]).

Die Filterfunktionalität des SecAN-Firewall-Systems wird durch acht seriell angeordnete Hardware-Filterstufen realisiert. Die Filterstufen befinden sich in der *Data Plane* und gewährleisten Netzwerkschutz auf den OSI-Schichten 2-4. Durch ihre Anordnung, die sich mit geringem Aufwand ändern lässt, ermöglichen sie eine inhärente Priorisierung. Aufgrund der Erhöhung der internen Verarbeitungsdatenrate auf 32 $Bit/Takt$ erzielen die Firewall-Filterstufen mit einer Frequenz von 198,036 MHz einen Durchsatz von $\approx$ 6,337 Gbit/s. Die Latenz aller Firewall-Filterstufen beläuft sich auf $\approx 672ns$.

Das Firewall-System greift auf das iptables-Konzept der Linux-Firewall [Net11] zurück. Für personalisierte Einstellungen kann der Nutzer über die *Custom Plane* (vgl. Abschnitt B.1) das Regelwerk des ISP-Administrators erweitern. Die Einstellungen des Nutzers sind jedoch nicht zwingend erforderlich. In jedem Fall greifen die Firewall-Einstellungen des ISP-Administrators, welche über die *Management Plane* (vgl. Abschnitt B.2) getätigt werden. Die *Control Plane* (vgl. Abschnitt B.3) validiert nutzerspezifische und administrative Regeln und führt sie zusammen. Dabei haben administrativ erzeugte Regeln eine höhere Priorität und können ggf. nutzerspezifische Regeln außer Kraft setzen.

Somit liegen die erzielten Ergebnisse deutlich über den der Arbeit zugrundeliegenden Publikationen und erfüllen die selbstdefinierten Vorgaben.

5.2 Web-Filter im Secure Access Node

Abschnitt 4.6 präsentiert den SecAN-Web-Filter. Dazu wurden zunächst unterschiedliche Filtertechniken sowie die Filterung von Web-Inhalten auf nationalen Ebenen vorgestellt (vgl. Abschnitt 4.6.2.1). Eine anschließende Untersuchung von reinen Software- bzw. kombinierten Software-/Hardware-Lösungen deckte deren Schwächen auf.

Bei kostenlosen bzw. kostengünstigen Lösungen liegen diese z. B. in der Installation, Konfiguration und Wartung, welche durch den Endanwender durchgeführt werden müs-

sen. Neben dem erforderlichen Fachwissen um diese Tätigkeiten durchzuführen, können sich umfangreiche Filterlisten negativ auf die QoE auswirken, da sie die verfügbaren Rechenressourcen stark belasten. Demgegenüber stehen hybride Lösungen wie bspw. [Web11, Blu11, Net]. Sie greifen nicht auf die lokalen Ressourcen der Endsysteme des Internet-Nutzers zurück, sind jedoch aufgrund ihrer Anschaffungs- und Wartungskosten (vgl. [Net12]) für den Nutzer unattraktiv.

Eine Lösung für die dargestellten Probleme ist der Hardware-basierte SecAN-Web-Filter. Er befreit den Nutzer von zuvor genannten Aufgaben, da sich der SecAN-Web-Filter im TZN befindet und diese Aufgaben von fachkundigen Administratoren übernommen werden. Der Internet-Nutzer bekommt die gefilterten Inhalte, ohne selber eingreifen zu müssen. Dennoch ist er in der Lage, eigene Filterregeln aufzustellen. Diese werden direkt im TZN appliziert und belasten nicht die lokalen Ressourcen. Aus Sicht des ISP entstehen Kosten für Anschaffung und Wartung des Systems. Diese können durch eine Vielzahl angeschlossener Kunden geteilt werden.

Mit einem Durchsatz von $\approx 1,03\ Gbit/s$ erreicht der SecAN-Web-Filter eine mehr als dreifach höhere Filterleistung gegenüber dem „Modell 910“ von Barracuda (vgl. [AG12]). Ähnlich dem britischen „Cleanfeed“ führt der SecAN-Web-Filter eine zweistufige Suche durch. Da beim SecAN-Web-Filter ausschließlich lokale Ressourcen verwendet werden, entsteht im Filterkern eine maximale Latenz von $\approx 997\ ns$. Damit ist die QoE deutlich höher als beim britischen „Cleanfeed“, welches auf die externe IWF-Datenbank zurückgreifen muss.

Sinnvollerweise filtert der SecAN-Web-Filter bereits HTTP-GET-Anfragen nach Web-Inhalten. Dem Autor der vorliegenden Forschungsarbeit sind keine Wege bekannt, diese Anfrage zu umgehen. Aus derzeitiger Sicht ist sie zwingend notwendig, um Ressourcen wie Texte, Abbildungen, Videos etc. von beliebigen Quelle im Internet zu ordern (vgl. Abschnitt 4.6.2.1). Die beiden Suchphasen unterteilen sich in Hash-Lookup-Phase und Domain-Lookup-Phase. In der Hash-Lookup-Phase wird die Domain CRC64 „gehasht“. Mit Hilfe der Kompressionseigenschaften von CRC64 konnte die durchschnittliche Domainlänge von 18 auf 8 Byte reduziert werden. Durch die Untersuchungen von mehr als 23 Millionen VeriSign-registrierter Domains ergab sich mit dem CRC64-Verfahren eine Kollisionswahrscheinlichkeit von $6,91 \cdot 10^{-6}$. Dadurch werden nur wenige Anfragen über die schnelle Hash-Lookup-Phase hinaus in die Domain-Lookup-Phase geleitet. In der Domain-Lookup-Phase wird eine indizierte Suche in einem DDR2-Speicher durchgeführt.

Die beschriebene Vorgehensweise des SecAN-Web-Filters führt ggf. zu „Overblocking“ bzgl. nicht rechtswidriger Inhalte auf derselben Domain. Jedoch leitet das entwickelte Web-Filtersystem nicht unter „Underblocking“, da jede bekannte Domain in der Domain-Lookup-Phase verifiziert wird.

5.3 IDS im Secure Access Node

In Kapitel 4.7 wurde ein System zur Erkennung von Eindringlingen konzipiert und entwickelt. Vor der Umsetzung wurden mehrere Hardware-basierte Ansätze untersucht. Dabei stellte sich heraus, dass Systeme, die für den Mustervergleich auf externe Speicher zurückgreifen, genauso ungeeignet sind wie Systeme, die ihre Vergleichsmuster in einer fixen DFSM-Struktur hinterlegen (vgl. 4.7.2). Ferner wurden Assoziativspeicher-basierte Ansätze untersucht. Diese wurden für das SecAN-IDS außer acht gelassen. Zum einen sind sie aufgrund ihrer hohen Anschaffungskosten für den Einsatz im TZN unattraktiv. Zum anderen können ausreichend große CAMs nicht aus den verfügbaren Ressourcen generiert werden. Aufgrund der Auswertung der Machbarkeitsstudie (vgl. Tabelle 4.14) entschied sich der Autor für den Bloom-Filter-basierten Ansatz. Dieser besteht aus 120 Bloom-Filtern und benötigt ausschließlich lokale FPGA-Ressourcen.

In einer weiteren Untersuchung wurde das frei verfügbare ID/IP-System SNORT analysiert. Im Ergebnis konnten einfache Muster direkt nachgenutzt werden, ohne die Fehlerrate des Hardware-IDS unnötig zu erhöhen. Aufgrund von Hardware-Beschränkungen wurde eine maximale Signaturlänge von 30 Byte definiert. In der Folge werden ca. 91,1 % der Hardware-geeigneten Signaturen in das IDS integriert. Wie in Tabelle D.1 dargestellt, besitzt bereits der nächstgrößere FX100T-FPGA ausreichend Ressourcen, um bis zu 50 Byte lange Signaturen problemlos verarbeiten zu können. Dementsprechend werden dann 98,3 % der Hardware-geeigneten Signaturen in das IDS integriert.

Für die Implementierung des IDS wurde weiterhin ein Verfahren entwickelt, das es erlaubt, den Bloom-Filter-Ansatz um zusätzliche Meta-Informationen anzureichern. Zu den Meta-Informationen zählen Protokolltyp und Port-Nummern. Diese Informationen werden gemeinsam mit den Signaturen in den 120 Filtern abgelegt. In jedem Bloom-Filter können bis zu 1.604 Signaturen hinterlegt werden, um eine FPR von 1 ‰ einzuhalten. Bei einer Gleichverteilung aller Signaturen kann das Regelwerk 192.480 Signaturen umfassen, die mit jedem Systemtakt parallel untersucht werden.

In der Qualitätsanalyse konnte mit Hilfe der „Reverse-Brute-Force“-Methode der praktische Funktionsnachweis erbracht werden. Da jedoch die FPR des Gesamtsystems um den Faktor 120 höher liegt als die FPR der einzelnen Filter, mussten die Bedingungen für das Gesamtsystem verschärft werden. Mit den verschärften Anforderungen sinkt die Anzahl der maximal programmierbaren Signaturen von 1.604 auf 452, um eine systemweite FPR von 1 ‰ einzuhalten. Da die Filter maximal mit 216 Signaturen gefüllt sind, sinkt die systemweite FPR deutlich unter 1 ‰ und ermöglicht es einem Match-Analyzer, innerhalb von durchschnittlich 1000 Takten einen Match zu verifizieren. In einem abschließenden Stresstest konnte letztendlich ein stabiles Systemverhalten nachgewiesen werden. Die Latenz des IDS-Filters beläuft sich auf $\approx 440ns$, die des Gesamtsystems, bestehend aus den Subsystemen Paketklassifizierung (*PCE*) und Paketverarbeitung (*PPE*), auf $\approx 527ns$.

5.4 Fazit

Das entstandene Gesamtsystem erfüllt alle Anforderungen aus Abschnitt 3.1. Neben den bereits erwähnten Eigenschaften zählt dazu auch, dass ein Angriff auf das System äußerst schwierig ist, da es sich netzwerktechnisch nicht adressieren lässt. Ferner lässt sich das SecAN-System nicht umgehen, weil der gesamte Datenverkehr, bevor er das Kernnetz erreicht, durch das TZN und damit durch den SecAN geleitet wird. Voraussetzung ist jedoch, dass die Filterstufen des SecAN flächendeckend auf alle Linecards verbaut werden. Mit dem vorgestellten Gesamtsystem lassen sich nicht nur angeschlossene Nutzer, sondern auch das Kernnetz und die Hardware im TZN schützen.

Da die Latenz des Gesamtsystems mit $\approx 4,201\mu s$ deutlich unter 100 ms liegt, wird laut [FIS08] die QoE der angeschlossenen Nutzer nicht negativ beeinflusst. Weiterhin ist das SecAN-System zukunftssicher für weiter steigende Datenraten, da es frei skalierbar ist. Das bedeutet, dass ein linearer Zusammenhang zwischen dem Durchsatz und der Anzahl der FPGA-Ressourcen besteht.

Ferner ist das entwickelte Sicherheitssystem sowohl offline als auch online Update-fähig. Das bedeutet, das ein Herunterfahren des SecAN zur Konfiguration nicht zwingend erforderlich ist.

Letztendlich offeriert das SecAN-Gesamtsystem ein neues Geschäftsmodell für ISP. Bspw. könnte mit der nächsten netzwerktechnischen Umstrukturierung das in dieser Ar-

beit vorgestellte Gesamtsystem eingeführt werden. Die entstehenden Mehrkosten können sukzessive auf die angeschlossenen Internet-Nutzer umgelegt werden.

Einziger Schwachpunkt des entwickelten Systems ist der Umgang mit verschlüsselten und getunnelten Daten. Diese können nicht in Echtzeit dekodiert werden, wodurch der SecAN diese Daten ungefiltert passieren lässt. Grundsätzlich haben alle dem Autor bekannten Systeme dieses Problem.

5.5 Ausblick

Um eine weitere Performance-Steigerung für das SecAN-Gesamtsystem zu erzielen, ist der Einsatz von Caches denkbar. Bspw. könnten in der RSE häufig angefragte Regelsätze in einem Cache untergebracht werden, der parallel zum DDR2-Speicher durchsucht wird. Der Web-Filter würde von einem Cache profitieren, der die Domains beinhaltet, die rechtswidrige Inhalte besitzen und häufig angefragt werden.

Denkbar wäre auch, dass weitere Filterstufen, die z. B. eine Anomalieerkennung bzw. eine Suche nach Malware-Signaturen durchführen, umgesetzt und mit dem bestehenden Gesamtsystem vereint werden. Aufgrund der hohen Datenmengen, die im TZN anfallen, könnte eine anomalienerkennende Filterstufe relativ schnell den “Normalzustand“ und Abweichungen von diesem erkennen. Für die Filterstufe, die Malware-Signaturen detektieren soll, ist z. B. eine Zusammenarbeit zwischen ISP und einschlägigen Unternehmen wie bspw. Kaspersky denkbar. Damit ließen sich für beide Seiten neue Geschäftsfelder erschließen. ISP könnten ihren Kunden (z. B. auch Unternehmen) einen noch besser ausgebauten Schutz anbieten und Hersteller von Antiviren-Programme würden ein vollkommen neues Marktsegment erschließen.

Letztendlich ließen sich fixe Anteile des entwickelten Gesamtsystems vom FPGA in eine ASIC-Realisierungen umsetzen. Bisher werden auf Linecards häufig FPGAs eingesetzt, um spezifische Aufgaben mit möglichst niedrigen Latenzen zu bearbeiten. Das in dieser Forschungsarbeit vorgestellte Sicherheitssystem besteht in Teilen aus fixen Funktionalitäten, die in ASICs überführt werden könnten. Das Gesamtsystem würde alle Freiheitsgrade beibehalten und gleichzeitig die Produktionskosten für Linecards reduzieren. Zudem würden mit ASICs höhere Durchsätze erzielt werden können.

Anhang A

Weiterführende Informationen zu den Grundlagen

A.1 Netzwerktopologie

Unter dem Begriff Netzwerktopologie werden eine Reihe von Grundstereotypen zusammengefasst. So können teil- und vollvermaschte Netzwerke sowie Bus-, Baum-, Ring-, Stern- und Doppelstern-Topologien unterschieden werden. Ferner existieren vermaschte und Zell-Topologien. In Abbildung A.1 sind die bezeichneten Topologien dargestellt.

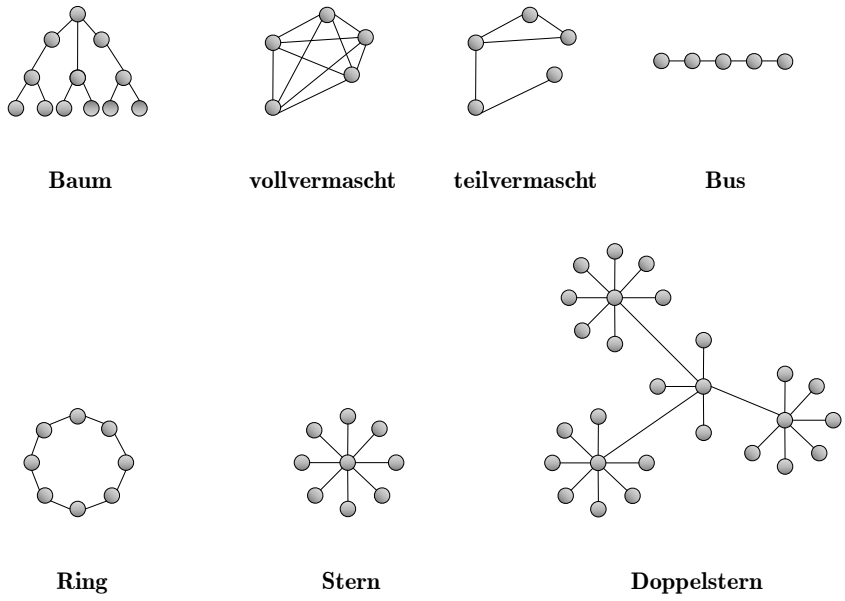


Abbildung A.1: Auswahl an Netztopologien zur Klassifikation von Netzwerken

Darüber hinaus können verschiedene Netztopologien miteinander verbunden werden. In diesem Fall wird von einem Netzverbund gesprochen. Treten unterschiedliche Netztopologien in einen Verbund, so wird dies als inhomogener Verbund bezeichnet. Der größte und bekannteste inhomogene Verbund von Netzwerken ist unter dem Begriff **Internet** bekannt.

A.2 Übertragungstechnik

Grundsätzlich lassen sich Kommunikationsnetzwerke in logische und physische Übertragungstechniken unterscheiden. Broadcast- und Punkt-zu-Punkt-Netzwerke gehören zu den logischen Übertragungstechniken. Während Broadcast-Netze über einen gemeinsam genutzten Übertragungskanal verbunden sind, bestehen Punkt-zu-Punkt-Netze aus einer Vielzahl von Verbindungen zwischen den Kommunikationspartnern [Tan03].

Die Art des Übertragungsmediums spielt bei den physischen Übertragungstechniken eine wichtige Rolle. Dort werden kupfer-, glasfaser- und funkbasierte Übertragungsmedien verwendet. Ihnen ist gemeinsam, dass sie elektromagnetische Wellen zur Informationsübertragung nutzen.

A.3 Übertragungsreichweite

Darüber hinaus lassen sich Kommunikationsnetze anhand ihrer lokalen Ausdehnung kategorisieren. Da On-Chip-Kommunikationsnetze für diese Arbeit keine Relevanz aufweisen, werden sie nicht in die Klassifizierung einbezogen. Darüber hinaus lassen sich Netzwerke in lokale Netze und Weitverkehrsnetze unterteilen. Abhängig von der Kommunikationsreichweite existieren innerhalb der beiden Klassen weitere Abstufungen. So werden das Body Area Network (BAN), (Wireless) Personal Area Network (PAN), Wireless Storage Area Network bzw. System Area Network (SAN) und LAN zu den lokalen Netzen gezählt. Sie besitzen Kommunikationsausdehnungen von wenigen Zentimetern im Falle von BANs bis hin zu wenigen Kilometern im Falle von LANs. Weitverkehrsnetze wie das Metropolitan Area Network (MAN), Wide Area Networks (WAN) und GAN hingegen weisen Ausdehnungen zwischen 100 km beim MAN bis hin zur weltweiten Entfaltung beim GAN auf.

Als Bindeglied zwischen lokalen Netzen und Weitverkehrsnetzen siedeln sich Teilnehmerzugangsnetzwerke (TZN) an. Sie ermöglichen den Kunden von ISP den Zugang zum Internet. TZN stehen unter Verwaltung von Netzbetreibern und besitzen eine räumliche Ausdehnung von bis zu zehn km. Die in der vorliegenden Arbeit entwickelten Sicherheitskonzepte wurden für TZNs konzipiert und umgesetzt.

A.4 Internet-Bedrohungen und deren Verbreitung

A.4.1 Spoofing-Angriffe

Spoofing ist eine Methode zur Verschleierung der Identität des Angreifers. Die Täuschungsversuche haben dabei das Ziel, Authentifizierungs- und Identifikationsverfahren zu untergraben [Fuh00]. Bedeutende Spoofing-Attacken sind:

A.4.1.1 ARP-Spoofing

Hierbei ist das Ziel, den Address Resolution Protocol (ARP)-Cache des Opfers zu manipulieren. Sind zwei Host-Rechner A und B in eine Kommunikationsbeziehung getreten, sendet der Angreifer beiden Kommunikationspartnern gefälschte ARP-Pakete. Der Angreifer trägt seine MAC-Adresse anstelle die der Gegenseite ein. Dabei erzeugt der Angreifer in den ARP-Caches beider Host-Rechner falsche IP-MAC-Zuordnungen. Er ist nun in der Lage, den Datenverkehr zwischen den Rechnern mitzuhören und zu manipulieren. Bei dieser Art des Angriffs spricht man von einer Man-In-The-Middle Attacke. Dieser Angriff kann z. B. durch den Einsatz eines IDS erkannt werden.

A.4.1.2 MAC-Spoofing:

Unter MAC-Spoofing wird die Änderung der eigenen MAC-Adresse in die eines bekannten Netzwerkgerätes verstanden. Mit der modifizierten MAC-Adresse gelingt es dem Angreifer, einen MAC-Filter zu überwinden und die im Netzwerk vorhandenen Ressourcen zu nutzen.

A.4.1.3 IP-Spoofing:

Wird die Quell-IP-Adresse absichtlich gefälscht, wird von IP-Spoofing gesprochen. Ein Angreifer setzt statt seiner IP-Adresse eine andere IP-Adresse- als Quell-Adresse ein. Auf diese Weise kann der Eindringling z. B. versuchen, eine IP-basierte Authentifizierung im Netzwerk zu überlisten.

A.4.1.4 DNS-Spoofing:

Beim DNS-Spoofing (auch DNS-Poisoning) gelingt es, die Zuordnung von einer Domain zur zugehörigen IP-Adresse zu ändern. In der Folge löst der DNS-Server die angefragte Domain zu einer falschen IP-Adresse auf. Der zum anfragenden Rechner gelieferte Inhalt kommt nun von der manipulierten IP-Adresse. DNS-Spoofing kann als zensierendes Mittel über Web-Inhalte verstanden werden und wird z. B. im Rahmen des Projektes „Goldener Schild“ in China eingesetzt.

A.4.1.5 URL-Spoofing:

Im Falle von URL-Spoofing bekommt der Anwender die Daten der Web-Seite nicht vom gewünschten Server, sondern vom Server des Angreifers. Beim Phishing ähneln die ausgelieferten Daten dabei stark den angeforderten Daten. Drei Terminologien werden beim URL-Spoofing verwendet:
Link-Spoofing - Die betrügerische URL ist direkt in der Adresszeile des Browsers sichtbar.
Frame-Spoofing - In die Web-Seite wird ein neuer Frame geschaltet. Auf diese Weise ist die Manipulation für den Anwender nicht mehr anhand der Adresszeile ersichtlich.
Website-Spoofing - Der Angreifer sorgt z. B. durch DNS-Spoofing dafür, dass Anfragen an einen bestimmten Web-Server zu ihm umgeleitet werden.

A.4.2 Schad-Software und deren Verbreitung

Eine Reihe von Software-basierten Lösungen ist in der Lage, Sicherheitslücken auszunutzen. Allgemein werden diese Schadprogramme als „Malware“ bezeichnet [Fuh00]. Tabelle A.1 gibt einen Überblick über die unterschiedlichen Malware-Klassen sowie deren Funktionen und Bedrohungsszenarien.

Schad-Software	Definition	Bedrohungsszenarien
Computervirus	Nicht selbstständige Programmroutine, die sich selbst reproduziert [Bun97]	• Manipulation im Systembereich • Manipulation an Programmen • Manipulation an Daten
Computerwurm	Selbstständiges, selbstreproduzierendes Programm, das zur Weiterleitung - meist in Netzwerken - Systemressourcen bindet und Schadroutinen enthalten kann	• Distributed Denial of Service • Verbreitung von Botnet-Software • Verbreitung von logischen Bomben • Verbreitung von Schad-Software
Trojanisches Pferd	Programme mit einer verdeckten, nicht dokumentierten Funktion oder Wirkung [Bun09]	• Einsatz von Rootkits • Einsatz als logische Bombe (führen Schadfunktion nach dem Auslösen eines externen Triggers aus) • Backdoor-Funktionalität • Spyware (Ausspionieren von sensiblen Daten)

Tabelle A.1: Definition und ausgewählte Bedrohungsszenarien von Malware nach [Fuh00]

Des Weiteren existieren bspw. „Hoaxes“ und „Scareware“[1].

A.4.2.1 Hoaxes und Scareware

Es handelt sich dabei um Falschmeldungen, die den Benutzer verunsichern und einschüchtern sollen. Ist ein Computer-Benutzer Opfer einer solchen Hoax- (Scherz) bzw. Schreckens-Nachricht geworden, wird er veranlasst, eine bestimmte Aktion auszuführen. Im Falle des Hoax „JDBGMGR.EXE“ - dem sogenannten *Teddybär-Virus* - wird der Anwender aufgefordert, die systeminterne Datei „JDBGMGR.exe“ zu löschen. Es handelt sich um eine Komponente des Java-Debugger (Java Debug Manager) von Windows. Weil die Datei einen Teddybären als Icon und einen beliebig erscheinenden Namen besitzt, wird sie von manchen Windows-Nutzern als verdächtig eingestuft.

Ist hingegen ein Computer von Scareware befallen (z. B. durch den Download eines kostenlosen Viren-Scanners), äußert sich dies bspw., indem dem Anwender ein Viren-Befall vorgetäuscht wird. Der Befall kann gegen Entrichtung einer Gebühr behoben werden. Wird die Zahlung getätigt, werden die Warnmeldungen schlicht unterdrückt.

A.4.2.2 Adware

Adware ist die Kurzform für Advertisement-Software (Werbe-Software). Sie verfolgt das Ziel, den Nutzer dazu zu bewegen, bestimmte Software- oder Hardware-Produkte käuflich zu erwerben. Bei der Software kann das Produkt wiederum Malware enthalten.

A.4.2.3 Backdoors und Botware

Einige Malware-Produkte bieten dem Initiator die Möglichkeit, das befallene System fernzusteuern. Dies wird durch Hintertüren in Programmen, den sogenannten Backdoors, realisiert. Ursprünglich wurden Backdoors von Programmentwicklern verwendet, um über ein Netzwerk oder andere Schnittstellen Einfluss auf den Programmablauf zu nehmen. Malware-Produzenten benutzen diese Methoden, um gekaperte Systeme, sogenannte Bots (kurz für Robot), fernzusteuern. Dabei wird eine Vielzahl von befallenen Systemen zu einem Botnetz zusammengefasst. Um weniger auffällig zu agieren, werden heutzutage große Botnetze in kleinere aufgeteilt. Gesteuert werden die Botnetze von

[1] Scareware ist ein englisches Kunstwort. Es besteht aus den Wörtern „Scare“ für Schrecken und „ware“ als Abkürzung für Software.

Botnetz-Betreibern durch „command-and-control server“ [Sym08]. Durch ein Botnetz lässt sich z. B. eine große Anzahl unerwünschter E-Mails (Spam) versenden. Ferner können mit Botnetzen verteilten Attacke (DDoS) durchgeführt werden, um die Ressourcen eines Systems zu überlastet. In beiden Fällen können beträchtliche finanzielle Schäden die Folge sein.

A.4.2.4 Spyware

Der Begriff Spyware ist ein englisches Kofferwort, bestehend aus den Wörtern „Spy“ für Spion und „ware“ für Software. Gelangt Spyware auf das Zielsystem, werden Tastatureingaben, Nutzer- und Surfverhalten und sensible Daten protokolliert. Sie können für eine spätere Verwendung entweder auf dem Zielsystem selbst abgelegt werden oder an den „Auftraggeber“ via Netzwerk versandt werden. Spyware wird z. B. durch Payload von Viren, Würmern und Trojanischen Pferden verbreitet.

A.4.2.5 Rootkits

Rootkits, ursprünglich aus der UNIX-Welt stammend, sind Programme, die sich tief in Betriebsystemen verankern. Sie manipulieren mit hohen Rechten Standardfunktionen des Systems. So tarnen sie sich z. B., indem sie Anfragen zu laufenden Prozessen fälschen. Ihr Ziel ist es, Schadfunktionen wie z. B. das Löschen von Antiviren-Programmen auszuführen. Wie Spyware können sich auch Rootkits als Payload von Viren und Würmern verbreiten. Darüber hinaus können Rootkits Bestandteil offizieller Software sein. Wie der SONY-Chef Howard Stringer bestätigt, habe das Musiklabel Sony BMG im Jahr 2005 einen Kopierschutz mit Rootkit-Funktionalität verwandet [HEI06].

A.4.2.6 Mischformen

Immer häufiger benutzt Malware mehrere Schadroutinen. So wird die Wahrscheinlichkeit der Verbreitung vergrößert und damit der Gewinn aus jeder Infizierung maximiert.

Den aufgeführten Varianten von Schad-Software ist gemein, dass sie durch ein eindeutiges Bit-Muster - einer Signatur - identifiziert werden können. Eine Ausnahme stellen die sogenannten metamorphen Computer-Viren dar. Sie besitzen die Eigenschaft, ihren Code selbstständig zu modifizieren, bevor sie sich weiter versenden. Da die Signatur aufgrund

der Code-Basis erstellt wird, habe sie keine eindeutige Signatur, wodurch ihr Auffinden erschwert wird.

A.4.2.7 Verbreitungsmechanismen

Wie in [Sym08] beschrieben, nimmt seit 2007 die professionelle Entwicklung von Schad-Software signifikant zu. Tools, wie das Zeus-Toolkit, beschleunigen die Erstellung von Malware um ein Vielfaches. Dennoch ist der Kostenfaktor für die organisierte Entwicklung von Malware nicht außer Acht zu lassen. Aus diesem Grund zielt ein Großteil der Bedrohungen auf finanzielle Gewinne ab. Eine Vielzahl von Verbreitungsvektoren einer Schad-Software sind also entscheidend für deren Erfolg.

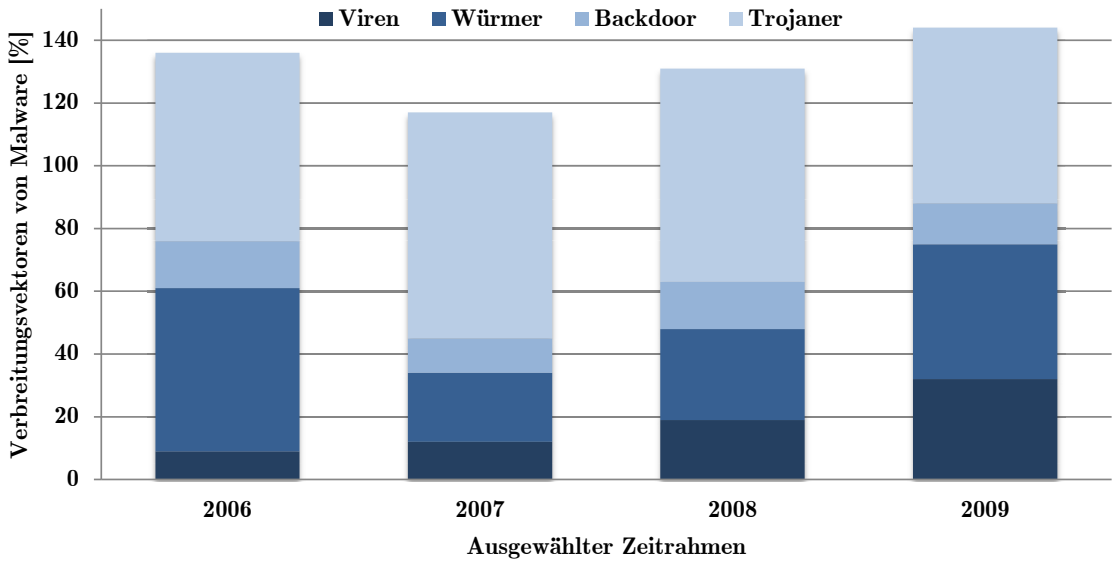


Abbildung A.2: Verbreitungsvektoren von Malware im Zeitraum 2006 - 2009

Abbildung A.2 zeigt die Verbreitungsvektoren von Malware im Zeitraum 2006 bis 2009. Dabei wird deutlich, dass die Viren-typische Verbreitung in den letzten Jahren ansteigt. Der signifikanteste Anteil entfällt jedoch auf die Verbreitung als Trojaner. Die Summe der jährlichen Verbreitungsvektoren überschreitet stets den Wert von 100% (vgl. Abbildung A.2). Dies weist nach, dass Entwickler von Malware multiple Verbreitungsvektoren verwenden.

A.5 Hashing

A.5.1 Qualität von Hash-Funktionen

Um die Anzahl der Kollisionen zu reduzieren, kann der Belegungsfaktor α (Quotient aus den Speichereinträgen S und der Speichergröße T) des Speichers reduziert werden (vgl. Formel A.1).

$$\alpha = \frac{S}{T} \tag{A.1}$$

Für die im SecAN-System angestrebte Menge von 32.000 Kommunikationsverbindungen wurden entsprechend viele Zufallswerte[2] erzeugt und CRC32 gehasht. Ausgehend von einem Speicher, der 40.000 Einträge aufnehmen kann, wurden unterschiedliche Belegungsfaktoren getestet. Die Testergebnisse sind in Abbildung A.3 dargestellt.

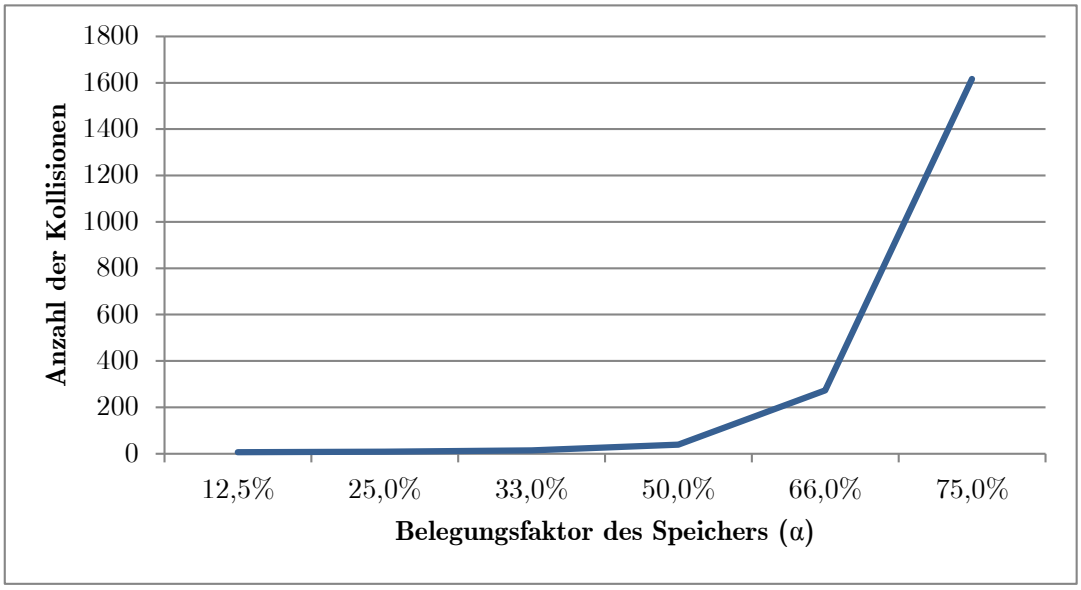


Abbildung A.3: Testergebnisse unterschiedlicher Speicherbelegungen

Demnach traten relativ wenige Kollisionen bis zu einem Belegungsfaktor $\alpha = 50$ % auf. Dieses Ergebnis deckt sich mit [Wid08], der einen Belegungsfaktor α von 1/2 für den beschriebenen Anwendungsfall als optimal ansieht. Muss ein Speicher dennoch über den

[2] True-Random-Number Generator: http://www.random.org/integers/

von [Wid08] angestrebten optimalen Belegungsfaktor von 50 % mittels Hashing adressiert werden, ist die Wahl des Verfahrens zur Kollisionsauflösung entscheidend. Kollisionen können durch „Hashing mit offener Adressierung“ und „Hashing durch Verkettung“ realisiert werden.

A.5.2 Hashing mit offener Adressierung

Diese Methode ist auch unter der Bezeichnung *offenes Hashing* bekannt und bezieht sich darauf, dass gleiche Hash-Werte unterschiedliche Speicherstellen zugewiesen bekommen. Beim Auftreten einer Kollision wird nach einer freien Speicherstelle gesucht, um ein Datum abzulegen. Dafür wird bei der Suche die initiale Speicheradresse erhöht. Ist diese Stelle unbelegt, ist die Suche beendet und das Datum kann gespeichert werden. Anderenfalls wird die Suche solange fortgeführt, bis eine freie Speicherstelle gefunden wurde bzw. der gesamte Speicherbereich einmal durchlaufen wurde. Die Suche im Speicher kann mittels **linearer Kollisionsauflösung** oder **Rehashing** durchgeführt werden.

A.5.2.1 Lineare Kollisionsauflösung

Die Suche beginnt an einer initialen Speicheradresse $h(x)$. Während der Suche wird die aktuelle Speicheradresse ständig um einen definierten Wert i erhöht. Um einen Überlauf der Speicheradressen zu vermeiden, wird mit jeder Inkrementierung der Speicheradresse eine Modulo-Operation (mod m) durch die Speichergröße berechnet. Im Beispiel wird die nächste Speicheradresse wie folgt bestimmt:

$$h(x,i) = (\acute{h}(x) + i) \ mod \ m \tag{A.2}$$

Die Suche ist beendet, wenn eine freie oder die initiale Speicherstelle gefunden wurde. Diese Art der Kollisionsauflösung ist leicht zu implementieren, allerdings neigt dieses Verfahren zu sogenanntem Clustering.

A.5.2.2 Rehashing

Das Rehashing nutzt zwei Hash-Funktionen. Führt das Hash-Ergebnis der ersten Hash-Funktion zu einer Kollision im Speicher, wird das Ergebnis der zweiten Hash-Funktion

zum Ergebnis der ersten Hash-Funktion addiert. Dieser Vorgang wird so lange wiederholt, bis eine freie oder die initiale Speicherstelle gefunden wurde. Um die Ergebnisse zu verbessern, sollten beide Hash-Funktionen voneinander unabhängig sein.

A.5.3 Hashing durch Verkettung (Chaining):

Beim Chaining werden kollidierende Elemente durch Verweise wie Perlen auf einer Kette aneinander gereiht. Typischerweise wird die verkettete Kollisionsauflösung durch eine Listen- oder Baumstruktur realisiert. Besitzt ein Element keinen Nachfolger, ist die Suche nach einem Element beendet. Bei der Speicherung eines Elementes wird dieses an die Liste bzw. Baumstruktur angefügt.

A.5.4 Hash-Verfahren zur Datenkompression

Hash-Verfahren können weiterhin zur Datenkompression eingesetzt werden. Obwohl die Länge der Eingangsvektoren variieren kann, ist die Länge des Hash-Vektors konstant. Wird der eingangs ermittelte Schlüssel mit Hilfe eines geeigneten Hash-Verfahrens komprimiert und als Speicheradresse verwendet, lässt sich eine im Mittel schnellere Suche im Speicher realisieren. Alle nachfolgend aufgeführten Hash-Funktionen zeichnen sich durch gute Hash-Eigenschaften aus und lassen sich zudem leicht in Hardware implementieren.

A.5.5 XOR-Methode

Der XOR-Hash-Wert entsteht, wenn einzelne oder alle Bit eines Elementes miteinander via XOR verknüpft werden. Abhängig von der Wahl der zu verknüpfenden Bit verändert sich die Qualität des Hash-Ergebnisses. Der XOR-Hash-Wert $xorh$, der alle Bit eines n Bit breiten Schlüssels k nutzt, berechnet sich wie folgt:

$$\begin{aligned} &for(i = 0; i < n; i++) \\ &\{ \\ &xorh = xorh \oplus k[i]; \\ &\} \end{aligned} \tag{A.3}$$

A.5.6 Hash-Funktionen der Klasse H_3

Bereits 1981 entwickelten Wegman und Carter [WC81] die Klasse der H_3-Hash-Funktionen. Sie definierten die Klasse H_3 wie folgt:

> Sei x der Bit-Vektor der Breite n, $x(k)$ das k-te Bit des Vektors x und X die Menge aller Bit-Vektoren x. Sei $h_q(x)$ ein Bit-Vektor der Breite m und H die Menge aller h_q. Sei weiterhin q eine Bit-Matrix mit m Zeilen und n Spalten. Sei $q(k)$ der Bit-Vektor der k-ten Spalte mit einer Länge von m-Bit und Q die Menge aller q.

Die Abbildung $h_q(x) : X \longmapsto H$ ist definiert durch:

$$h_q(x) = x(1) \cdot q(1) \otimes x(2) \cdot q(2) \otimes ... \otimes x(n) \cdot q(n) \qquad \forall q \in Q \tag{A.4}$$

Beispielhaft veranschaulicht Abbildung A.4 die Berechnung eines H_3-Hash-Wertes. Da es sich bei $h_q(x) : X \longmapsto H$ um eine lineare Transformation handelt, erzeugen echte Zufallszahlen q_i, die einer statistischen Gleichverteilung unterliegen, auch gleich verteilte Hash-Werte [RFB97].

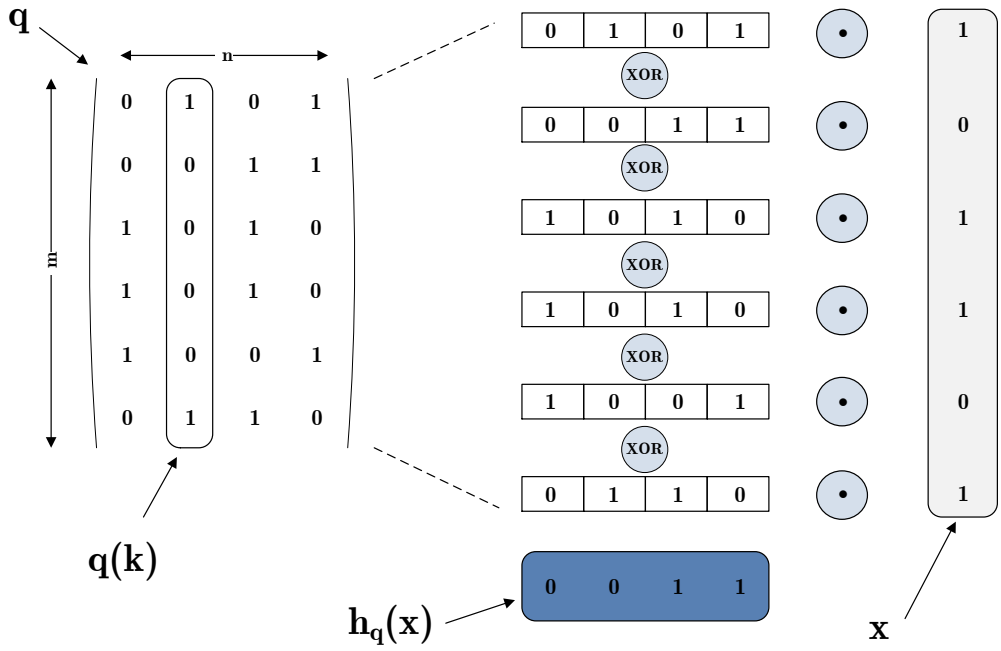


Abbildung A.4: Beispielhafte Hash-Funktion der Klasse H_3

A.5.7 Cyclic Redundancy Check

Eine sehr beliebte Hash-Methode ist die Berechnung von CRC-Werten. Diese Methode liefert sehr gute Ergebnisse, die äußerst nahe dem Maximalwert 1 liegen [Jai92]. Dazu wird eine Polynomdivision des Schlüssels durch ein Generatorpolynom durchgeführt. Der Rest der Rechenoperation bildet den Hash-Wert. Nicht alle Generatorpolynome sind gleich gut geeignet. Zwischen CRC1 und CRC64 existieren mehrere Generatorpolynome (zum Teil auch mehrere Polynome in einer Gruppe nebeneinander). Im Falle von Ethernet hat sich das CRC32-Polynom in der folgenden Binärdarstellung bewährt [DIX82]:

$$CRC32 = x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1 \quad \text{(A.5)}$$

Anhang B

Softwarekonzepte und Realisierungen

Kapitel B beschäftigt sich mit den drei Software-Ebenen des SecAN. Deren grobe Funktionalität wurde bereits in Abschnitt 3.2.3 beschrieben. In den folgenden Abschnitten werden absteigend im Abstraktionsniveau die *Custom Plane*, *Management Plane* und *Control Plane* vorgestellt.

B.1 Custom Plane

Wesentliches Ziel des SecAN-Projektes ist die Erhöhung der Netzwerksicherheit für Internet-Teilnehmer, ohne dass deren Mitwirken notwendig wird. Die optional zu bedienende Web-basierte *Custom Plane* soll es jedem ISP-Endkunden ermöglichen, seine individuellen Sicherheitseinstellungen zu tätigen, die über den ISP-seitigen Basisschutz hinausreichen. Hat ein Internet-Teilnehmer nicht das Bedürfnis, seine Sicherheitseinstellungen zu personalisieren, verbleiben ihm dennoch die Sicherheitseinstellungen des ISP-Administrators.

B.1.1 Konzipierung

Die Web-basierte *Custom Plane* muss eine Reihe von Aufgaben erfüllen. Diese sind:

1. Die Anmeldung muss mit individuellen *Login*-Daten möglich sein. Dabei erfolgt die Verifikation durch einen Abgleich mit den entsprechenden Datenbankeinträgen (vgl. Abschnitt B.2), damit jeder Internet-Nutzer ausschließlich auf seine Konfigurationsparameter Zugriff hat.
2. Die einstellbaren Konfigurationsparameter müssen aus einer Datenbank geladen werden. Diese befindet sich in der *Management Plane* (vgl. Abschnitt B.2).

3. Die Akzeptanz für den Service wird erhöht, wenn die Oberfläche sowohl für Leihen als auch für technisch versierte Endkunden intuitiv bedienbar ist. In der Folge müssen unterschiedliche Konfigurationslevel einstellbar sein.
4. Die nutzerspezifischen Einstellungen werden in einer Datenbank in der *Management Plane* vgl. Abschnitt B.2 abgespeichert.
5. Neben den authorisierten Nutzern dürfen ausschließlich ISP-Administratoren Zugriff auf die Tabellen der bezeichneten Datenbank haben. Die Sicherheitsrichtlinien müssen Modifikationen durch Dritte unterbinden.

B.1.2 Realisierung

Abbildung B.1 zeigt die eingangs beschriebene Web-Oberfläche nach erfolgreichem Login.

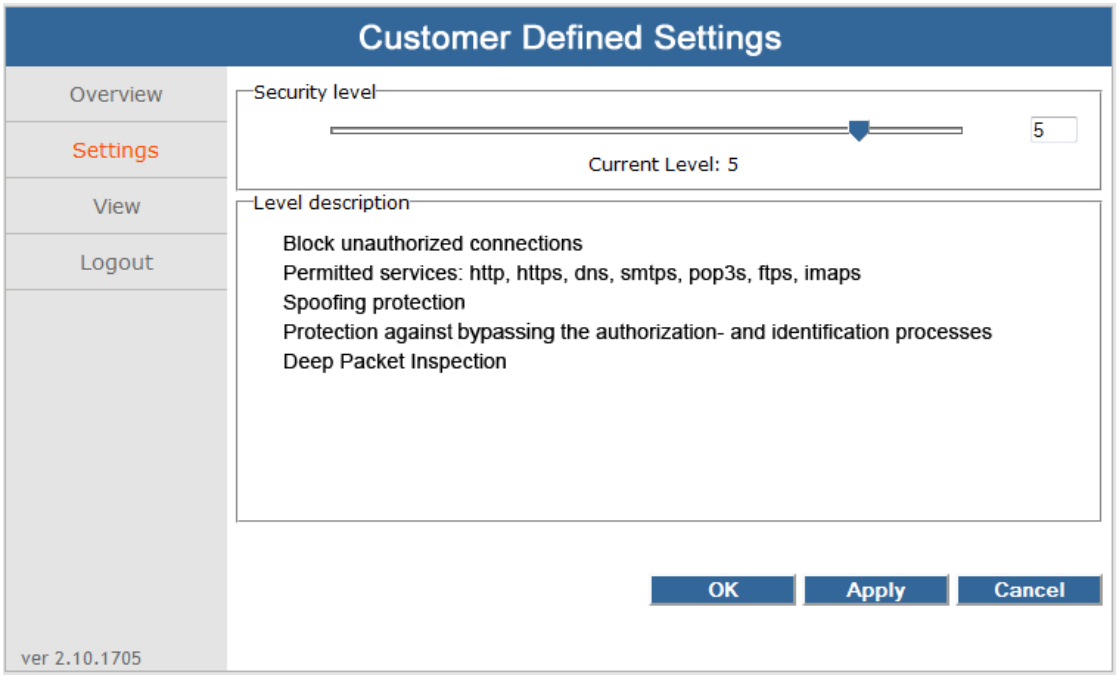


Abbildung B.1: Nutzerseitig definierte Sicherheitseinstellungen mit ISP-Vorgaben [Hilfswissenschaftliche Tätigkeit]

Das linksseitige Menü besitzt die folgenden vier Unterpunkte:

- **Overview:** Dem Nutzer wird sein Login-Name sowie sein aktuelles Konfigurationslevel angezeigt. Handelt es sich um den ersten Login, wird dem Nutzer eine Standardkonfiguration zugeteilt.
- **Settings:** In diesem Punkt lässt sich die aktuelle Konfiguration modifizieren. Es stehen ein „Simple Mode“ sowie ein „Custom Mode“ zur Auswahl.
- **View:** Dieser Punkt dient dem Wechsel zwischen „Simple Mode“ und „Custom Mode“. Die Einstellungen beider Modi werden separat gespeichert, wobei die Einstellungen des aktiven Modus für die Filterregeln verwendet werden.
- **Logout:** Hier ist eine ordnungsgemäße Abmeldung des Internet-Nutzers vorgesehen.

Technisch weniger versierte Nutzer werden auf den „Simple Mode“ zurückgreifen. Vom ISP freigegebene Dienste werden dort mit plakativen Begriffen wie „E-Mail“ und „Internet-Telefonie“ umschrieben. Über Checkboxen werden die verschiedenen Services an- oder abgewählt. Jedem Service entsprechen ein oder mehrere Kommunikations-Ports auf OSI-Schicht 4. Die Zuordnung der Bezeichnung zu den entsprechenden Ports erfolgt in der Datenbank. Weiterhin besteht die Möglichkeit der Aktivierung eines Standard-Web-Filters sowie eines „Musterfilters“ - des IDS.

Über 7 Sicherheitsstufen (0 bis 6) lassen sich im „Custom Mode“ komplexere Einstellungen vornehmen. Dabei gilt: Je höher die Nummer der Sicherheitsstufe, desto restriktiver ist die Netzwerkkommunikation (Sicherheitslevel 6 bildet eine Ausnahme). Die Sicherheitsstufen werden im oberen Teil der GUI eingeblendet und lassen sich per „Slider“ variieren (vgl. Abbildung B.1). Zur Information werden die aktuellen Filterregeln in der „Level Description“ dargestellt. In der niedrigsten Sicherheitsstufe (Stufe 0) sind alle Sicherheitsmechanismen deaktiviert. Es gelten nur noch die Vorgaben des ISP-Administrators. Dagegen wird in den Sicherheitsstufen 1 bis 5 eine differenzierte Nutzung einzelner Dienste ermöglicht (vgl. „Level Description“ in Abbildung B.1). In der höchsten Stufe des „Custom Mode“ (Stufe 6) besitzt der Internet-Nutzer die weitreichendste Kontrolle über sein persönliches Kommunikationsverhalten. Abbildung B.2 zeigt die Manipulationsmöglichkeiten für den Nutzer in dieser Stufe auf.

Zunächst legt der Nutzer die grundlegende Sicherheitsrichtlinie („Default policy“) - uneingeschränkte bzw. unterbrochene Kommunikation - fest. Bei der unterbrochenen Kommunikation („Block all Ports“) bleibt jedoch die Kommunikation zwischen Nutzer und ISP bestehen. Daran anschließend werden die Ausnahmen von der „Default Policy“ de-

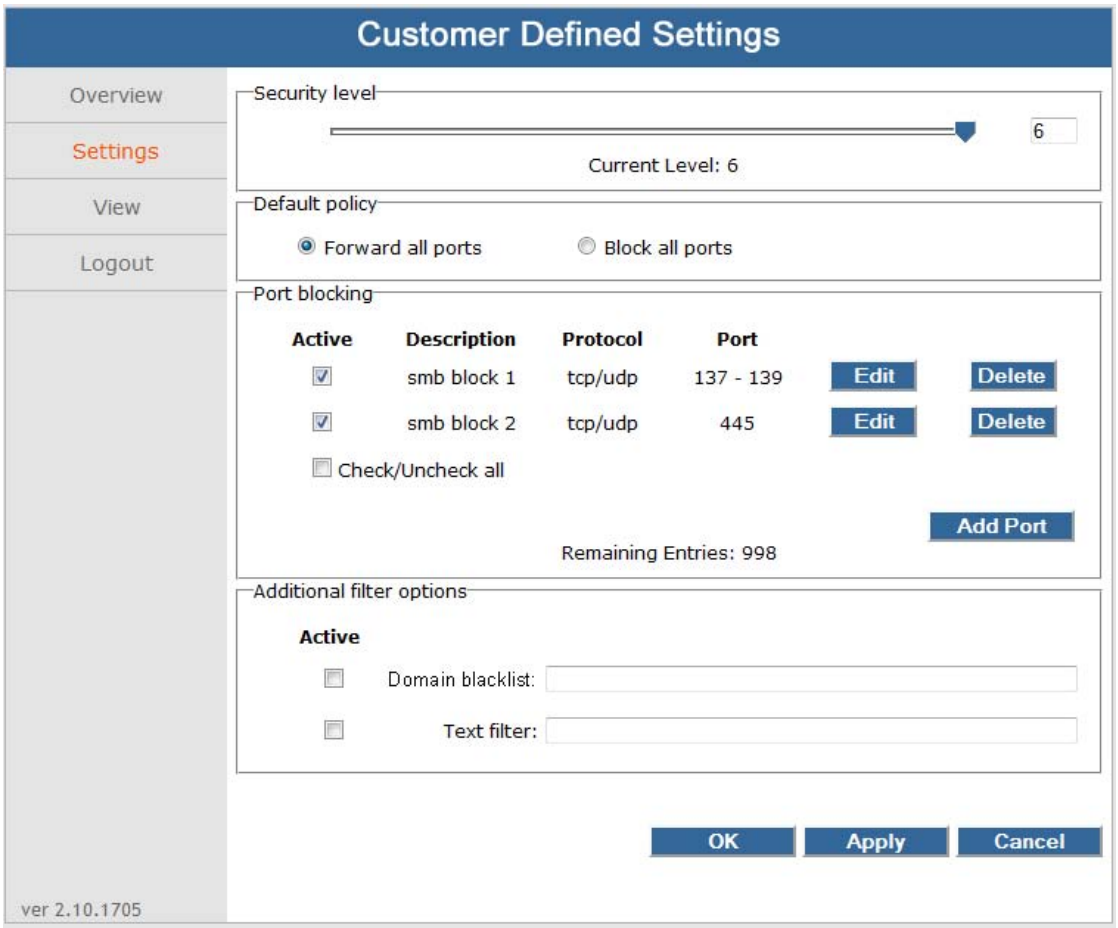


Abbildung B.2: Nutzerseitig definierte Sicherheitseinstellungen ohne ISP-Vorgaben [Hilfswissenschaftliche Tätigkeit]

finiert. Durch Festlegen einzelner Ports bzw. ganzer Port-Bereiche werden so Kommunikationswege gezielt geöffnet bzw. geschlossen. Im unteren Bereich lassen sich zusätzliche Filter aktivieren und um weitere Filtereinträge erweitern. Abschließend werden alle vom Nutzer getätigten Konfigurationen in die Datenbank der *Management Plane* übertragen.

B.1.3 Zusammenfassung Custom Plane

Durch die grafische Benutzeroberfläche in der *Custom Plane* wird den Internet-Nutzern die Möglichkeit gegeben, auf einfachste Weise durch persönliche Filterregeln den Basisschutz des ISP zu erweitern. Dieser Service ist optional, wird er nicht genutzt, gelten automatisch die Regeln des ISP. Anderenfalls ergänzen die nutzerspezifischen Regeln die des ISP. Die gesamte GUI ist intuitiv zu bedienen und so gestaltet, dass sie im Besonderen technisch versierten Nutzern einen Wiedererkennungseffekt mit bestehenden lokalen Router-Lösungen bietet. Für wenig routinierte Nutzer stellt bereits der „Simple Mode“ mittels plakativer Begriffe und Einstellmöglichkeiten einen Mehrwert dar.

B.2 Management Plane

Die *Management Plane* befindet sich ausschließlich unter Kontrolle des ISP und stellt das zentrale Sicherheitscenter des SecAN dar. Sie übernimmt Verwaltungsaufgaben und abstrahiert verschiedene Konfigurationsmöglichkeiten. Dazu besitzt sie Schnittstellen zu den benachbarten Architekturschichten (vgl. Abbildung 3.2).

B.2.1 Konzipierung

Die *Management Plane* ist zweigeteilt und besitzt für jeden Aufgabenbereich eine separate GUI. Genau wie in der *Custom Plane* handelt es sich bei der Schnittstelle zum Nutzer um eine Web-basierte GUI. Sie besitzt folgende Aufgaben:

Aufgabe 1: Bereitstellung und Verwaltung nutzerspezifischer Daten

1. Nutzerverwaltung: Erstellung, Rücksetzung, Sperren und Löschen von Nutzeranmeldungen sowie Überprüfung der Anmeldedaten des Nutzers
2. Erstellung und Speicherung der Konfigurationsvorgaben für den Nutzer

Aufgabe 2: Regelwerk des Basisschutzes

1. Erstellung und Verwaltung systemweiter Sicherheitsrichtlinien (Regeln für alle Filtermodule)
2. Zusammenfassen von administrativen und nutzerseitigen Filterregeln zu Konfigurationsparametern für die Hardware-Module (vgl. Kapitel 4)
3. Speicherung der Konfigurationsdaten in der Datenbank

B.2.2 Realisierung

Web-Oberfläche des ISP-Administrators Die Web-basierte grafische Benutzeroberfläche zur Verwaltung der Konfigurationsvorgaben für die Nutzer ähnelt vom Erscheinungsbild und Handling in weiten Teilen der in Abschnitt B.1 vorgestellten *Custom Plane*-GUI. Sie wird den ISP-Administratoren zur Verfügung gestellt und ist ausschließlich aus dem TZN erreichbar. Somit können nur autorisierte Personen auf sie zugreifen.

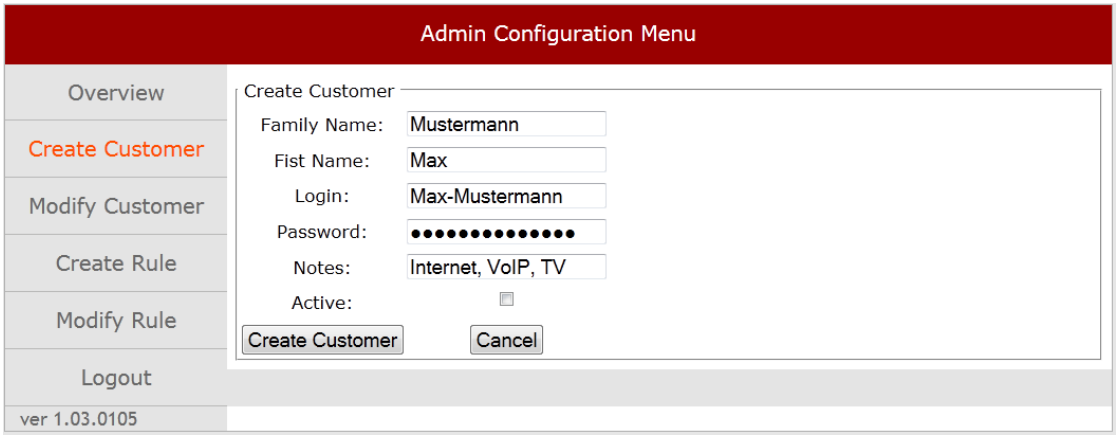


Abbildung B.3: Konfigurationsoberfläche des ISP-Administrators zur Verwaltung von Endkunden und Filterregeln

Nach erfolgreichem Login stehen dem Administrator die folgenden Menüpunkte zur Verfügung (vgl. Abbildung B.3):

- **Overview:** Dem Adminstrator wird sein Login-Name angezeigt.
- **Create Customer:** Neue Endkunden werden hier erstellt (vgl. Abbildung B.3). Ist der Zugang aktiv geschaltet, kann ein Endkunde sich anmelden.

- **Modify Customer:** Nutzer können gesucht werden. Anschließend ist das Manipulieren bzw. Löschen der nutzerspezifischen Daten möglich. Das Deaktivieren der Web-Logins für Endkunden ist ebenfalls möglich.
- **Create Rule:** Ähnlich wie bereits in Abbildung B.2 dargestellt, werden Service-Bezeichnungen erstellt und diesen Protokolle und Port-Nummern zugeordnet. Darüber hinaus werden die Services einem Sicherheitslevel zugewiesen und können aktiviert bzw. deaktiviert werden. Dabei gilt, dass Sicherheitslevel mit niedrigen Nummern in Sicherheitslevel mit hohen Nummern integriert werden. Beispielsweise sind alle Services der Sicherheitslevel 1, 2 und 3 im Sicherheitslevel 4 enthalten. Die so entstehende Hierarchie führt zu immer stärkeren Restriktionen bei der Kontrolle des Datenverkehrs. Weiterhin werden Filterregeln für den Web-Filter und das IDS erstellt. In der Darstellung des „Simple Mode“ werden statt der Protokolle und Portnummern nur die Service-Bezeichnungen angeboten.
- **Modify Rule:** Einzelne Regeln und Services können gesucht, modifiziert und gelöscht werden.
- **Logout:** Hier ist eine ordnungsgemäße Abmeldung des Administrators vorgesehen.

B.2.2.1 Erstellung und Verwaltung der ISP-seitigen Filterregeln

Die Erstellung und Verwaltung der systemweiten Sicherheitsrichtlinien ist von den nutzerseitigen Konfigurationsvorgaben vollkommen losgelöst und wird deshalb über eine zweite GUI realisiert. Sie trägt den Namen „GUItables“ und wurde im Rahmen einer *Hilfswissenschaftlichen Tätigkeit* plattformübergreifend mit der QT-Entwicklungsumgebung[1] entwickelt. „GUItables“ besteht aus einem Framework und drei Plugins (vgl. Abbildung B.4). Dieses Framework-Plugin-Konzept ermöglicht eine leichte Wartung und Erweiterung der Software. Jede Sicherheitsfunktionalität wird als eigenes Konfigurations-Plugin über einen Registerreiter in „GUItables“ bereitgestellt und besitzt somit eine eigene grafische Benutzeroberfläche (vgl. Abbildung B.4).

Während die Plugins die Funktionalität der *Management Plane* übernehmen, beinhaltet das Framework die Funktionalität der *Control Plane* (Aufgaben und Realisierung der *Control Plane* werden in Abschnitt B.3 beschrieben). Beide Schichten arbeiten sehr eng miteinander zusammen und wurden aus diesem Grund in einer Software vereint.

[1] QT: http://qt.nokia.com/

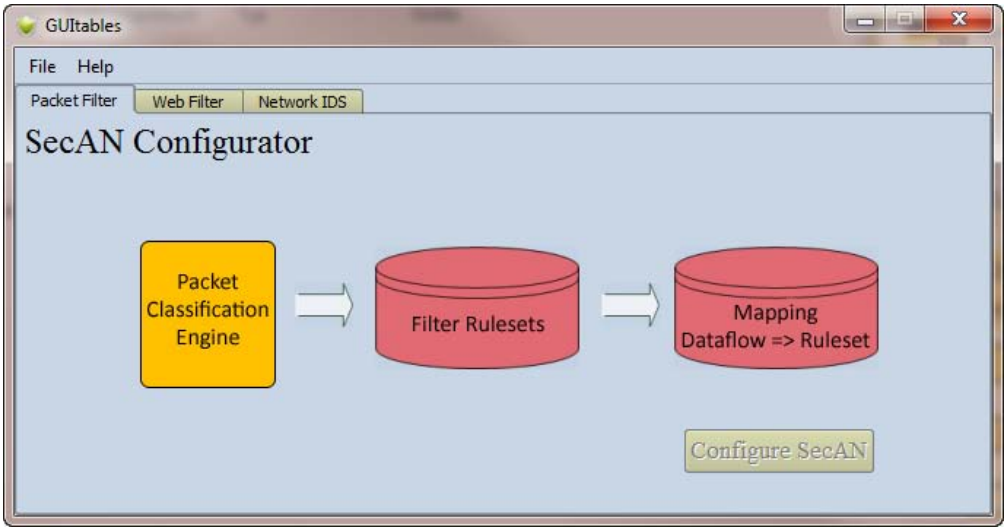


Abbildung B.4: Grafische Benutzeroberfläche zur Konfiguration der Hardware der *Data Plane* [Hilfswissenschaftliche Tätigkeit von Vlado Altmann]

Realisierung des *Packet Filter*-Plugins in *GUItables* Wie bereits in Kapitel 3.2 beschrieben, werden spezielle Filtermodule den Datenstrom untersuchen. Damit die Filtermodule ihre Aufgabe möglichst effizient erfüllen können, werden sie von zusätzlichen Hardware-Modulen unterstützt. Dazu zählen die PCE und die RSE, die im Kapitel 4 beschrieben werden.

Die grafische Benutzeroberfläche des *Packet Filter*-Plugins ist in Abbildung B.4 dargestellt und bietet folgende Interaktionsmöglichkeiten:

1. Konfiguration des Paketklassifizierungsmoduls: Festlegung, welche Frame-Parameter in die Paketklassifizierug aufgenommen werden (Erstellung der richtungsabhängigen „Flow ID Masks“ (oder auch „Flow ID Trigger“ vgl. Abbildung B.5))
2. Konfiguration der *Packet Filter*-Regeln: Erstellung von Filterregeln und Zusammenführung dieser zu Regelsätzen
3. Konfiguration des *Default Ruleset*: Dieses wird benutzt, wenn dem aktuellen Daten-Flow kein Regelsatz zugeordnet werden kann
4. Zuweisung eines Regelsatzes zu einem definierten Daten-Flow
5. Konfiguration der Hardware

Demnach erfolgt die Konfiguration des Paketfiltermoduls in drei Phasen. Wie in Abbildung B.4 angedeutet, wird zunächst die PCE konfiguriert, anschließend werden Filterregeln erstellt, zu Regelsätzen zusammengefasst und letztendlich auf dem klassifizierten Frame abgebildet.

Phase I: Festlegung der Klassifizierungsparameter

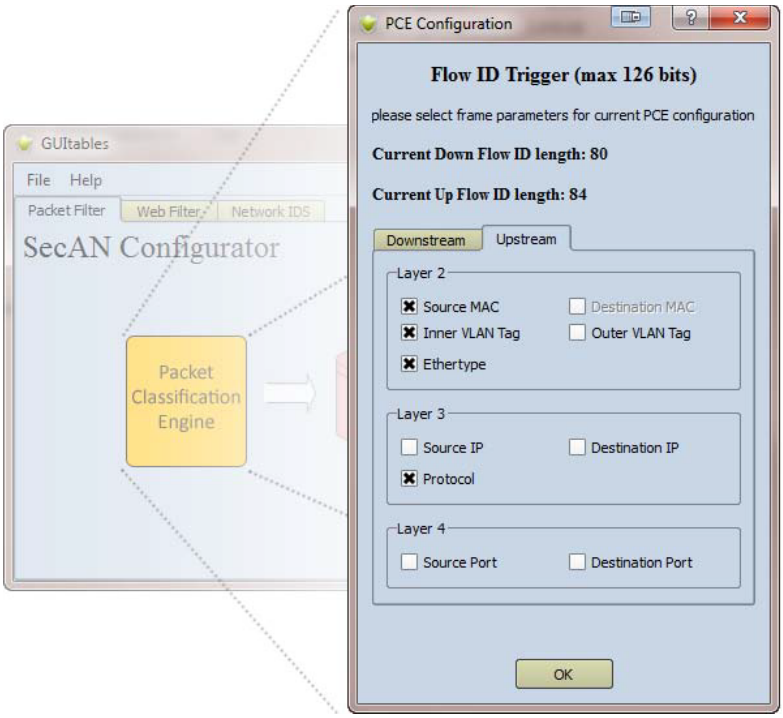


Abbildung B.5: Grafische Benutzeroberfläche zur Konfiguration des Paketklassifizierungsmoduls [Hilfswissenschaftliche Tätigkeit von Vlado Altmann]

Die richtungsabhängige Klassifizierung des Datenstromes erfolgt über richtungsabhängige „Flow ID Trigger“ (vgl. Abbildung B.5). Es handelt sich dabei um signifikante Bestandteile des Datenstroms (vgl. Tabelle B.1). Die Frame-Parameter werden zu einem Vektor - der *Flow ID* - zusammengefügt. Im nachfolgenden Schritt - dem Mapping von *Packet Filter*-Regeln - werden der *Flow ID* eindeutige Paketfilterregeln zugewiesen. Ferner dient die *Flow ID* zur Reduzierung der Suchzeit der Paketfilterregel (vgl. Abschnitt 4.4).

Typischerweise wurden bis zu fünf Frame-Parameter zur Klassifizierung herangezogen

Frame-Parameter	**Beschreibung**
L2-Adressen	Adressen der Schicht 2 im OSI-Modell (Source- und Destination MAC-Adressen (S-MAC und D-MAC))
VLANs	zwei VLANs (outer und inner VLAN)
L3-Protokoll	Protokoll der Schicht 3 im OSI-Modell (z. B. IP Version 4)
L3-Adresse	Adressen der Schicht 3 im OSI-Modell (IP-Adressen)
L4-Protokoll	Protokoll der Schicht 4 im OSI-Modell (z. B. TCP)
L4-Portnummern	Portnummern der Schicht 4 im OSI-Modell (Source- und Destination Port (S-PORT und D-PORT)

Tabelle B.1: Signifikante Bestandteile von Ethernet-Frames zur Bestimmung der Flow ID

[JRV08, JP09a, DBB09]. Durch die Umstellung auf SDN-Architekturen werden heute bis zu 15 Header-Felder erfasst (vgl. [QP15]). Die Erweiterung des Parametersatzes auf zehn Frame-Parameter (vgl. Tabelle B.1) erhöht die Freiheitsgrade für die Klassifizierung, wie sie im TZN gefordert wird. Beispielsweise können Datenströme anhand MAC-Adressen gesperrt oder anhand von VLAN-IDs priorisiert behandelt werden.

Die Klassifizierung des Datenstromes erfolgt richtungsabhängig d. h., es können unterschiedliche Frame-Parameter für den Up- und den Downstream gewählt werden. In der GUI werden die ausgewählten Frame-Parameter mit einem „x“ markiert und die Längenangabe der damit einhergehenden *Flow ID* aktualisiert. Die Begrenzung der Länge der *Flow ID* wird in Abschnitt 4.4 erläutert. Für die Konfigiuration über die GUItables-Oberfläche bedeutet das, dass Frame-Parameter nicht mehr aktiviert werden können, die die maximal gültige Länge der *Flow ID* überschreiten würden. Durch Betätigen des „OK“-Buttons werden die ausgewählten Frame-Parameter in der Datenbank gespeichert.

Phase II: Erstellung von *Packet Filter*-Regeln und Regelsätzen
In diesem Teil der Software werden die Filterregeln (Rules) für den Hardware-*Packet Filter* erstellt und zu sogenannten Regelsätzen (Rule Sets) gebündelt. Da die Länge des Regelsatzes auf die Verarbeitungszeit durch die Hardware Einfluss hat, wurde deren maximale Länge auf 1024 Byte begrenzt. Nach der Erstellung der Regelsätze werden diese zunächst in der Datenbank abgespeichert. Erst die Zuordnung von einer oder mehreren

Flow ID's zu einem Regelsatz führt dazu, dass diese später in die Hardware konfiguriert werden.

Für den ersten Regelsatz (Rule Set 0) besteht eine Besonderheit. Er ist der Standardregelsatz *Default Rule Set* und wird immer dann verwendet, wenn die *Packet Filter*-Hardware eine *Flow ID* errechnet, die keinem Regelsatz zugeordnet werden kann.

Gleich der Beschreibung aus Abschnitt B.1 werden bei der Erstellung eines Regelsatzes zunächst die *Default Policy* und anschließend die Ausnahmen von der *Default Policy* definiert. Acht Hardware-Paketfilterstufen stehen zur Verfügung und können, wie in Abbildung B.6 dargestellt, separat über je einen Button konfiguriert werden.

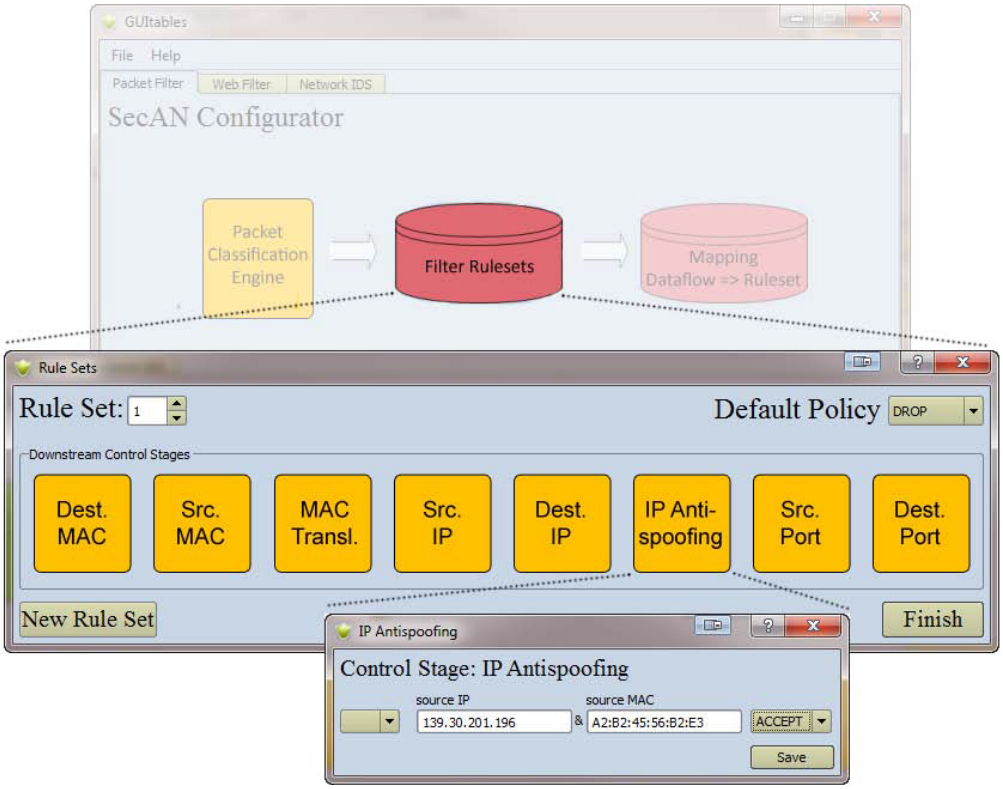


Abbildung B.6: Grafische Benutzeroberfläche zur Konfiguration von *Packet Filter*-Regeln [Hilfswissenschaftliche Tätigkeit von Vlado Altmann]

Mit den derzeit existenten Filterstufen können separate Regeln für beide Hardware-Adressen, beide Netzwerkadressen sowie beide Ports der 4. OSI-Schicht eines Ethernet-Frames definiert werden. Ferner können Filterregeln für eine MAC-Address-Translation

und Regeln gegen IP-Spoofing erstellt werden (vgl. Abbildung B.6). Anschließend wird jeder Regel eine Aktion („Drop“ oder „Accept“) zugewiesen. Darüber hinaus lassen sich alle Regeln invertieren. So können bspw. ganze IP-Bereiche blockiert oder zugelassen werden.

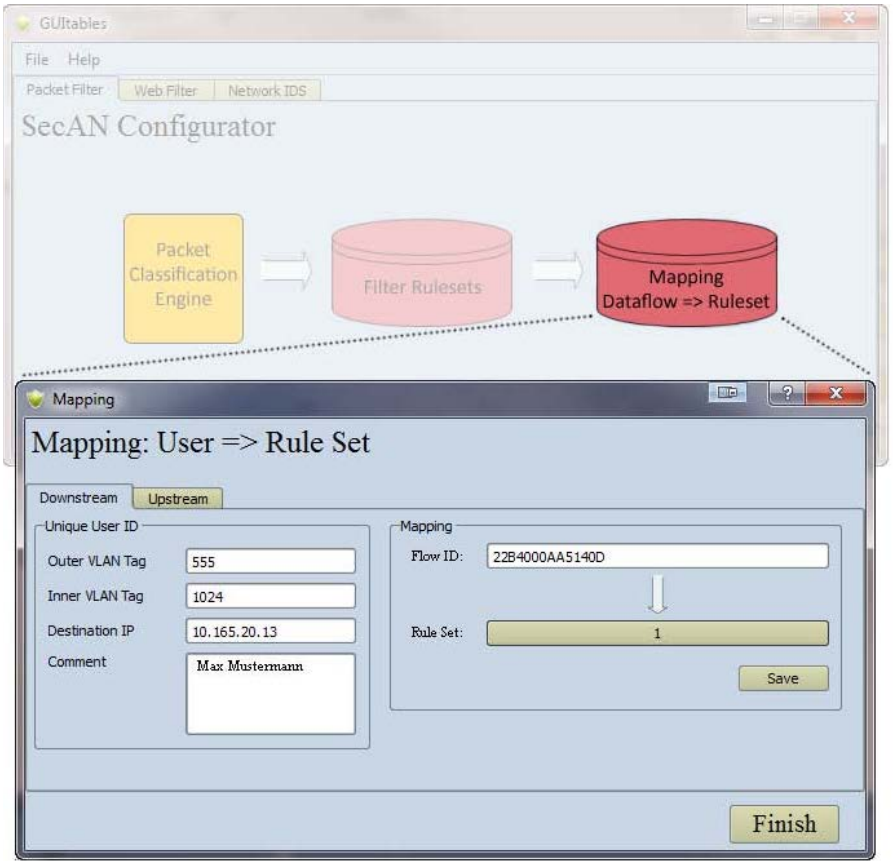


Abbildung B.7: Grafische Benutzeroberfläche zum Mapping von Nutzerkennungen und Regelsätzen [Hilfswissenschaftliche Tätigkeit von Vlado Altmann]

Phase III: Abbildung der *Flow ID's* auf *Rule Sets*

Der letzte Schritt bei der Konfiguration des *Packet Filters* ist die Erstellung gültiger *Flow IDs* und deren Zuweisung zu einem *Rule Set*. Im linken Bereich von Abbildung B.7 dienen die in Phase I festgelegten Frame-Parameter als „Label“ für die Eingabefelder. Ihnen werden nun real auftretende Netzwerkdaten sowie ein zusätzlicher Kommentar (im Beispiel der Name des angeschlossenen Teilnehmers) zugeordnet. Aus den eindeu-

tigen Werten der Frame-Parameter werden im rechten Bereich von Abbildung B.7 die eindeutigen *Flow ID's* erzeugt und letztendlich einem spezifischen *Rule Set* zugewiesen. Zu beachten ist, dass einer *Flow ID* genau ein *Rule Set* zugewiesen werden kann (1:1 Beziehung), jedoch einem *Rule Set* mehrere *Flow ID's* zugeordnet werden können (1:n Beziehung). Durch Betätigen des *Finish*-Buttons werden alle *Flow-ID's* und deren Zuordnungen in die Datenbank übertragen. Der Administrator befindet sich nun wieder in der Ausgangs-GUI (vgl. Abbildung B.4).

Realisierung des Web-Filter-Plugins in *GUItables* Die grafische Oberfläche aus Abbildung B.8 wird benutzt, um den Web-Filter zu konfigurieren. Dabei sind die vertikal angeordneten Zahlen 1 bis 5 in der GUI nicht vorhanden. Sie dienen lediglich der leichteren Beschreibung der entsprechenden Funktionen.

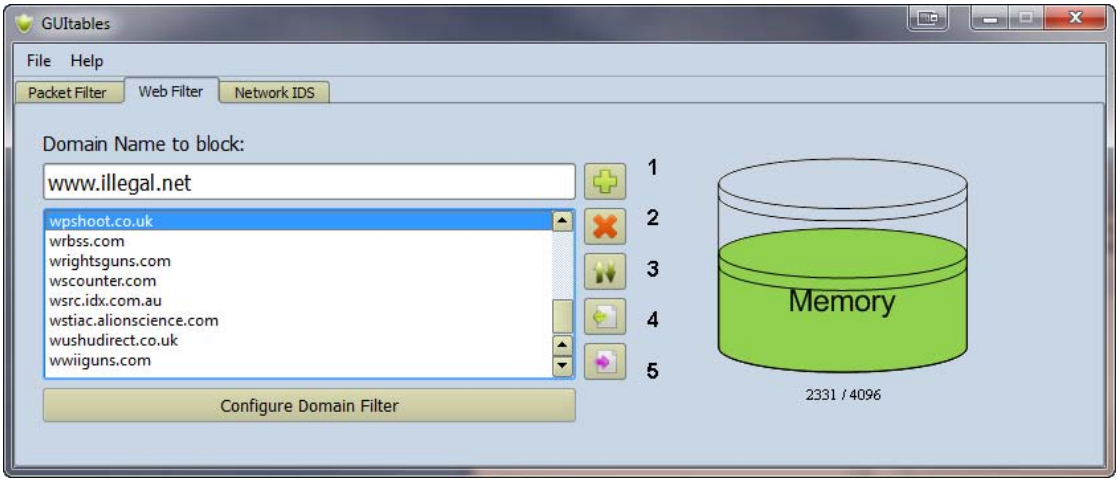


Abbildung B.8: Grafische Benutzeroberfläche zur Konfiguration der Web-Filterregeln [Alt10]

Die Filterliste ist parametrierbar. In der gegenwärtigen Version können bis zu 4.096 Einträge aufgenommen werden. Wie in Abschnitt 4.6.3.2 herausgearbeitet, ist dieser Wert für den Anwendungsfall ausreichend hoch. Neben der Eingabe einzelner Domains ist das Importieren ganzer Domain-Listen möglich. Solche Listen werden beispielsweise von *UrlBlackList*[2] und von der *Shalla Secure Services KG*[3] bereitgestellt. Sie liegen im

[2] urlblacklist.com
[3] www.shallalist.de

Klartext vor und können ohne Modifikationen importiert werden. Während die Aufnahme einzelner Domains über das Plus-Symbol neben der Eingabezeile (vgl. Punkt 1 in Abbildung B.8) erfolgt, werden die Einträge der Domain-Blacklisten durch das Import-Symbol (vgl. Punkt 4 in Abbildung B.8) hinzugefügt. Ungültige Domains, die bspw. die maximale Länge überschreiten bzw. ungültige Zeichen enthalten, werden durch den Eingabe-Validator identifiziert und nicht in die Liste aufgenommen. Gültige Domain-Namen, die nicht-ASCII-Zeichen enthalten wie z. B. deutsche Umlaute, werden ebenfalls erkannt, jedoch nicht ausgesondert. Stattdessen werden die Internationalizing Domain Names in Applications (IDNA)-Domains [P. 03] erkannt und mit dem Punycode [Ber03] in ASCII-Zeichen umkodiert, bevor die DNS-Anfrage an den Server versandt wird.

Wurde eine Domain z. B. fälschlich in die Blacklist aufgenommen, kann sie mit dem roten „X“-Icon (vgl. Punkt 2 in Abbildung B.8) gelöscht werden. Um Domains schnell zu finden, kann die gesamte Liste mit dem Doppelpfeil-Icon (vgl. Punkt 3 in Abbildung B.8) auf- bzw. absteigend sortiert werden. Letztendlich ermöglicht das Export-Icon (vgl. Punkt 5 in Abbildung B.8) den Export aller in die Filterliste aufgenommenen Domains.

Realisierung des Intrusion-Detection-System-Plugins in *GUItables* Die grafische Oberfläche des in Abbildung B.9 dargestellten Plugins dient der Konfiguration der IDS-Hardware Filterstufe. Durch das IDS-Plugin werden potentielle Bedrohungsmuster aus einer Datenbasis erfasst, separiert und so umkodiert, dass sie von der IDS-Hardware als Referenzmuster verwendet werden können. Als Datenbasis wird die weit verbreitete, freie und quelloffene Software-Lösung „Snort“ verwendet (vgl. Abschnitt 4.7.2.4). Eigene Datenbasen, die dem Aufbau von SNORT-Datenbanken folgen, können ebenfalls verwendet werden (vgl. Abschnitt 4.7.3.2).

Um die Konfiguration der IDS-Filterstufe zu ermöglichen, ist die grafische Benutzeroberfläche in drei Bereiche unterteilt.

1. Statusfenster zur Ausgabe von Zustandsmeldungen (roter Rahmen)
2. Auswertung der Datenbasis und Generierung der Konfigurationsmuster (blauer Rahmen)
3. Konfiguration der DPI-Filterstufe (gelber Rahmen)

Unter Berücksichtigung der aufgeführten drei Schritte lässt sich die grafische Benutzeroberfläche intuitiv bedienen. Da die Auswertung der Datenbasen und die Erstellung der

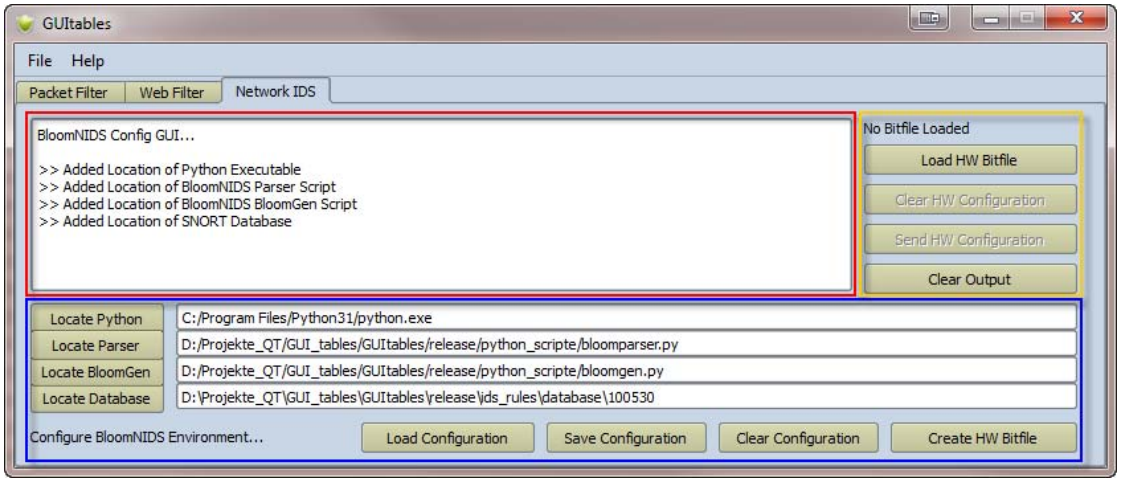


Abbildung B.9: Grafische Benutzeroberfläche zur Konfiguration der IDS-Regeln [Pfe10]

Konfigurationsmuster Skript-gesteuert abläuft, müssen zunächst die Orte des Interpreters der beiden Skipt-Dateien und der Datenbasis angegeben werden. Diese Einstellungen können zusätzlich gespeichert, geladen und gelöscht werden (vgl. Tabelle B.2).

Anschließend werden konfigurationsrelevante Informationen aus verschiedenen Datenbasen identifiziert und in einer gemeinsamen Datenbank zusammengeführt (vgl. Abschnitt 4.7.3.2). Diese Aufgabe übernimmt das Parser-Skript (bloomparser.py). Im dritten Schritt analysiert das „bloomgen.py“-Skript die gemeinsame Datenbank und generiert Hardware-geeignete Konfigurationsmuster. Durch Betätigen des Button „Create HW Bitfile“ (vgl. Tabelle B.2) werden diese ASCII-codiert in der Konfigurationsdatei „nids.bit“ gespeichert.

Über die Button innerhalb des gelben Rahmens ist es möglich, die Datei „nids.bit“ zu laden, an die Hardware zu versenden bzw. zu löschen. Darüber hinaus können die Angaben des Statusfensters gelöscht werden. Einen Überblick über die Funktionen der einzelnen Bedienelemente des *GUItable*-Plugins gibt Tabelle B.2.

B.2.3 Zusammenfassung Management Plane

Verwaltungsaufgaben für Endkunden und Filterregeln werden durch die *Management Plane* realisiert. Ihre grafischen Benutzeroberflächen sowie deren Funktionalitäten sind vollkommen voneinander separiert und werden ausschließlich durch ISP-Administratoren

Button	**Kategorie**	**Bedeutung**
Locate Python	Generierung der Konfigurationsmuster	Öffnet ein Dialogfenster zur Lokalisierung des Python-Interpreters
Locate Parser	Generierung der Konfigurationsmuster	Öffnet ein Dialogfenster zur Lokalisierung des Bloom-Parser-Scriptes *bloomparser.py*
Locate BloomGen	Generierung der Konfigurationsmuster	Öffnet ein Dialogfenster zur Lokalisierung des Bloom-Generator-Skriptes *bloomgen.py*
Locate Database	Generierung der Konfigurationsmuster	Öffnet ein Dialogfenster zur Lokalisierung des Ordners, welcher die IDS-Regelsätze enthält
Load Configuration	Generierung der Konfigurationsmuster	Lädt die Konfiguration der zuvor gespeicherten Parameter aus der Datenbank
Save Configuration	Generierung der Konfigurationsmuster	Speichert die Parameter der Python-Scripte in die Datenbank
Clear Configuration	Generierung der Konfigurationsmuster	Setzt die zur Generierung der Konfigurationsmuster notwendigen Parameter zurück
Create HW Bitfile	Generierung der Konfigurationsmuster	Ruft automatisiert den Parser und den Bloom-Generator mit den entsprechenden Parametern auf und erzeugt die Konfigurationsmuster
Load HW Bitfile	Versand der Konfigurationsmuster	Lädt die DPI-Filter-Konfigurationsmuster in GUItables und konvertiert die ASCII-codierte Binärdarstellung in Bytecode
Clear HW Configuration	Versand der Konfigurationsmuster	Setzt die intern geladene Konfigurationsmuster zurück
Send HW Configuration	Versand der Konfigurationsmuster	Versendet die konvertierte Konfigurationsmuster an die DPI-Filterstufe
Clear Output	Statusfenster	Löscht die Inhalte des Statusfensters

Tabelle B.2: Erläuterungen zu Elementen der DPI-GUI

verwaltet.

Dabei dient das Web-Interface der Verwaltung der Endkunden und deren Vorgaben für Filterregeln. Einzelne Regelvorgaben werden zu Regelsätzen zusammengefasst und in den Sicherheitsstufen 0 bis 5 angeboten. Dabei gilt, je höher die Sicherheitsstufe, um so komplexer ist das Regelwerk (niedere Sicherheitsstufen sind in höheren Sicherheitsstufen integriert).

Die GUItables-Oberfläche besteht aus einem Framework und verschiedenen Plugins. Durch den Plugin-basierten Aufbau ist die Software leicht zu warten und zu erweitern. Für die drei Sicherheitsfunktionalitäten des SecAN (Firewall, Web-Filter und IDS) existiert je ein Plugin. Jedes Plugin bietet weiter eine eigene GUI zur Erstellung und Verwaltung der entsprechenden Sicherheitsfunktionalität. Die Überprüfung der Konfigurationsdaten sowie deren Versand zur Hardware übernimmt die *Control Plane*.

B.3 Control Plane

Die *Control Plane* ist die unterste Ebene des Software-Stacks des SecAN (vgl. Abbildung 3.2). Als Framework der GUItables-Software besitzt sie selbst keine GUI. Vielmehr übernimmt sie Kontroll- und Steuerungsaufgaben.

B.3.1 Konzipierung

Wie bereits in Abschnitt B.2.2.1 beschrieben, besteht die GUItables-Software aus einem Framework - der *Control Plane* - und mehreren Plugins. Die *Control Plane* selbst besteht aus voneinander unabhängigen Komponenten, welche erst durch die darüber liegende *Management Plane* in einen gemeinsamen Kontext gebracht werden. Sie übernimmt grundlegende Aufgaben wie das Laden und Darstellen der Plugins, aber auch die Kommunikation mit den Konfigurationsdaten sowie die Binärkodierung und den Versand dieser an die Hardware. Abbildung B.10 stellt die funktionalen Module innerhalb der *Control Plane* dar.

Während der *Core* die Hauptkomponente der *Control Plane* darstellt und Kommunikationaufgaben mit den Plugins sowie dem Konfigurator übernimmt, wird der Konfigurator Filterregeln erfassen, umkodieren und an die *Data Plane* versenden.

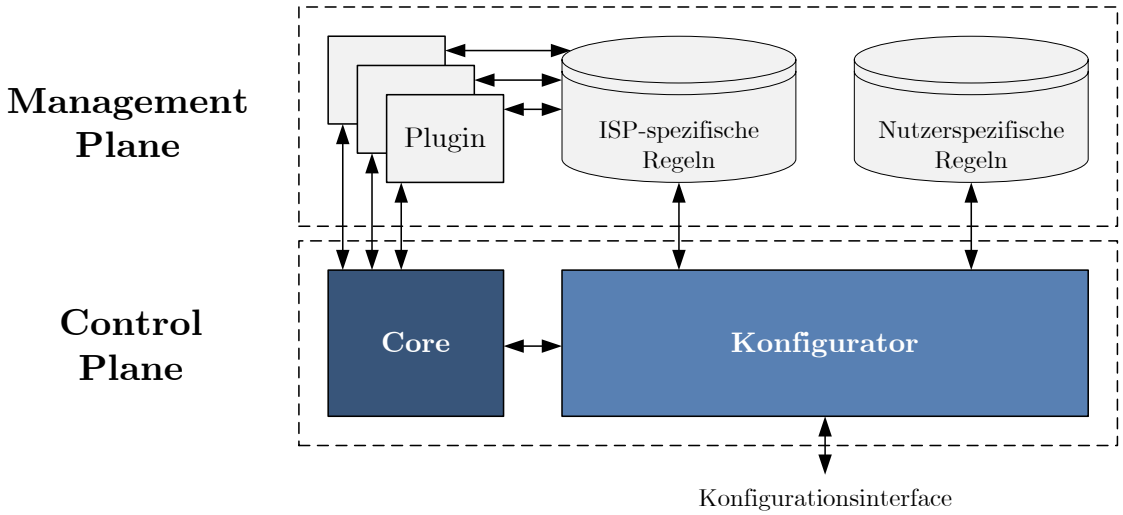


Abbildung B.10: Funktionale Module innerhalb der Control Plane

B.3.2 Realisierung

Mit dem Starten der Software steuert der *Core* das Laden der Plugins. Ferner überprüft er die Plugins und stellt fest, wann sie bereit sind, ihre Konfigurationsdaten an die *Data Plane* zu versenden. Der Trigger für diese Entscheidung wird durch die *Configure*-Button der Plugins gesetzt.

Anschließend kontaktiert das *Core*-Modul das *Konfigurator*-Modul, welche Konfigurationsdaten an die Hardware übergeben werden sollen. Dieses liest zunächst die ISP-spezifischen Konfigurationsdaten, bevor die nutzerspezifischen Konfigurationsdaten geladen werden. Bevor beide Regelwerke vereint werden, erfolgt eine Plausibilitätsprüfung. Bei Widersprüchlichkeiten setzen sich die ISP-spezifischen Konfigurationsdaten gegenüber den nutzerspezifischen Konfigurationsdaten durch. Durch die Plausibilitätsprüfung können auch nutzerseitige Fehlkonfigurationen, wie bspw. das Sperren des HTTP-Ports erkannt und behoben werden.

Letztendlich werden die Konfigurationsdaten in Bytecode konvertiert, anschließend in Ethernet-Frames gekapselt und an die Hardware versandt. Dabei besitzen die Konfigurations-Ethernet-Frames einen speziellen Aufbau, welcher von der Hardware als Konfigurations-Frame erkannt und entsprechend weiter verarbeitet wird (vgl. Abbildung B.11).

Konfigurations-Frames besitzen als Destination-MAC-Adressen das hexadezimale Mus-

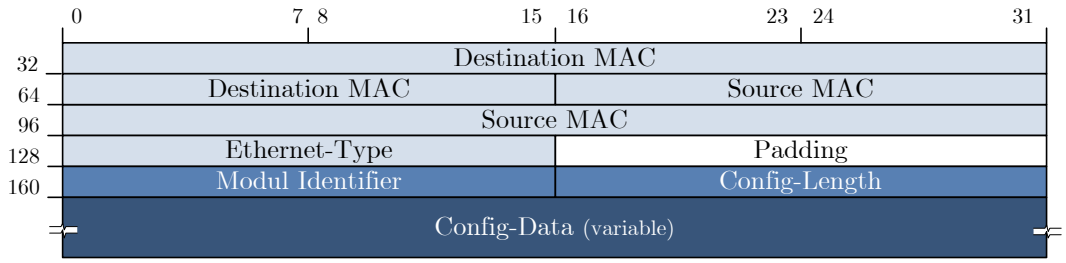


Abbildung B.11: Struktur eines Konfigurations-Frames

ter „01:01:01:01:01:01“ sowie den hexadezimalen Ethernet-Type „FFFF“. Bei einer internen Datenverarbeitungsrate von 4 Byte/Takt wird durch das Einfügen von 2 „Padding“-Byte eine Hardware-schonende Umsetzung sichergestellt. Anschließend folgen 2 Byte *Config-Type* sowie 2 Byte *Config-Length*. Dabei dient der *Modul Identifier* zur Klassifizierung. Ferner gibt die *Config-Length* die Länge der gesamten Konfigurationsdaten an.

Die hexadezimalen Werte des *Modul Identifiers* „0000“ und „FFFF“ wurden reserviert und zeigen den Anfang bzw. das Ende des Konfigurationsvorganges an. Sie können als Request-Frames verstanden werden und sind notwendig, um die Hardware in den Konfigurationsmodus zu versetzen bzw. diesen für die Hardware zu beenden. Kann der Konfigurationsvorgang beginnen, sendet die Hardware den Request-Frame zurück und teilt der *Control Plane* mit, dass sie sich im Konfigurationsmodus befindet. Fortan können die Konfigurationsdaten an die Hardware versandt werden. Jedes Datum wird zunächst von der Hardware bestätigt, bevor das nächste Konfigurationsdatum versandt wird. Sendet die Hardware den Konfigurations-Frame mit dem *Config-Type* „FFFF“ zurück, ist der Konfigurationsvorgang sowohl für die Hard- als auch für die Software beendet.

B.3.3 Zusammenfassung Control Plane

Die *Control Plane* arbeitet ohne Eingreifen des ISP-Administrators vollkommen autonom. Als Bindeglied zwischen der *Management Plane* und der *Data Plane* übernimmt sie Kommunikations- und Steuerungsaufgaben. Darüber hinaus ist sie für das Laden der Plugins, über welche in der *Management Plane* die ISP-seitigen Filterregeln erstellt werden, die Plausibilitätsprüfung der Konfigurationsdaten und den Konfigurationsvorgang verantwortlich.

Anhang C

Hardware-Konzepte und Realisierungen

C.1 Datenverarbeitung innerhalb der Data Plane

Aus Sicht der Datenverarbeitung sollen drei wesentliche Ziele erreicht werden:

1. Ethernet-Frames sollen aufgrund von Filterbewertungen verworfen werden. Verzögerungen, die durch den Verarbeitungspfad entstehen, sollen diesen Effekt nicht auslösen.
2. Wenn es zu Stauungen im System kommt, muss ein Mechanismus dafür sorgen, dass Ethernet-Frames nur am Systemeingang verworfen werden. Auf diese Weise wird der datenverarbeitende Pfad nicht unnötig belastet.
3. Das System muss Ethernet-Frames mit einer niedrigen Latenz verarbeiten. Internet-Teilnehmer werden den in dieser Arbeit vorgestellten Sicherheits-Service nur dann akzeptieren, wenn ihre QoE nicht negativ beeinflusst wird.

C.1.1 Backpressure-Signale der Data Plane

Um unterschiedliche Latenzen der internen Module zu kompensieren, führt der Autor *Backpressure*-Signale ein. Backpressure-Signale werden zur Intermodulkommunkation eingesetzt. Sie zeigen an, ob ein Modul bereit ist, Daten zur Verarbeitung aufzunehmen. Dementsprechend sind sie entgegengesetzt zur Datenverarbeitung gerichtet und verlaufen folglich vom Systemausgang in Richtung Systemeingang. Ist eine Weiterleitung der Daten nicht möglich, verbleiben diese so lange im aktuellen Modul, bis das nachfolgende Modul die neuen Daten aufnehmen kann. Somit tragen sie dazu bei, dass *Deadlocks*

[Tan09] vermieden und die Robustheit des Gesamtsystems erhöht wird. Ferner erzeugen die beschriebenen Backpressure-Signale den gewünschten Effekt, dass bei hohem Verkehrsaufkommen Verwürfe (packet drop) lediglich am Systemeingang entstehen.

C.1.2 Datenverarbeitungsbreite der Data Plane

Damit dieser Effekt weiter reduziert wird, muss die interne Datenverarbeitungsrate erhöht werden. Eine Anforderung aus Abschnitt 3.1.1 betrifft das Einhalten einer minimalen Zielgeschwindigkeit von 1 Gbit/s. Gbit-Ethernet-Frames werden mit 8 Bit/Takt ($\hat{=}$ 8 Bit/8 ns) übertragen. Eine Erhöhung der internen Datenverarbeitungsbreite auf 32 Bit/Takt reduziert die interne Latenz und sorgt für ausreichenden Puffer, um interne Verzögerungen zu kompensieren, wie sie bspw. bei dynamischen Speichern auftreten.

Um die Effizienz des SecAN-Gesamtsystems weiter zu optimieren, werden Synchronisations-FIFOs (Synch-FIFOs) am Systemeingang und -ausgang verwendet. Sie besitzen zwei Clock-Eingänge und ermöglichen, dass intern und extern mit unterschiedlichen Clock-Domains gearbeitet werden kann. Abbildung C.1 zeigt neben den unterschiedlichen Clock-Domains die beschriebenen Sync-FIFOs direkt hinter den Ethernet-Interfaces.

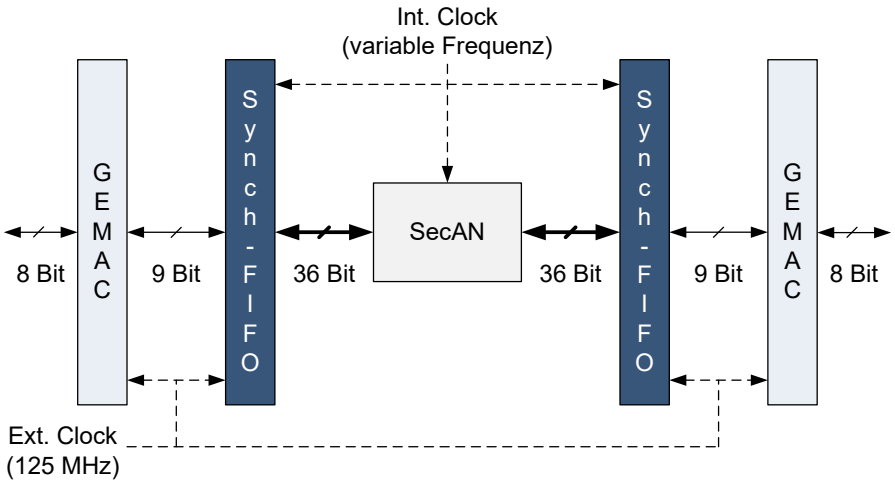


Abbildung C.1: Erweiterung des SecAN-Datenpfades

Ethernet-Daten werden über die Gigabit Ethernet Media Access Controller (GEMAC) empfangen. Dieser steuert das Schreiben in die Synch-FIFOs. Da entgegengesetzt der

extern empfangenen Daten die internen Datenblöcke mit je 4 Byte/Takt gelesen werden, ermittelt ein Counter die Anzahl der empfangenen Byte. Der gezählte Wert wird modulo der internen Verarbeitungsbreite gerechnet und zeigt an, wie viele Byte zusätzlich in die FIFO geschrieben werden müssen, damit wieder ein voller 4 Byte Block entsteht.

Nachdem der Frame vollständig empfangen wurde, wird das Ende des Frames mit einem separaten Flag markiert. Gleichzeitig signalisiert der GEMAC, ob es sich um einen gültigen Datenstrom handelt. Ist dies nicht der Fall, wird der empfangene Datenstrom direkt aus der Sync-FIFO entfernt. Anderenfalls wird er bis zur Endekennung aus der Sync-FIFO gelesen und in den *dpt* umgewandelt. Es handelt sich dabei um eine systeminterne Datenstruktur, wie sie in Abbildung C.2 dargestellt ist.

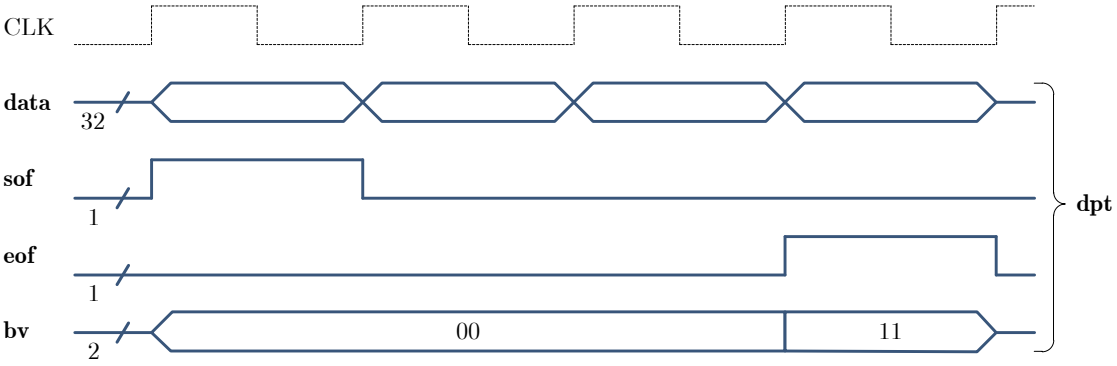


Abbildung C.2: SecAN internes Signaldiagramm zur Verarbeitung von Ethernet-Daten

Der „data“-Teil des dpt entspricht den empfangenen Ethernet-Daten. Er umfasst je Takt 4 Byte. Beim Lesen aus der Synch-FIFO wird für das erste Datum die Startkennung (Start of Frame (sof)) = „1“ gesetzt, für alle weiteren nicht. Weiterhin wird die Synch-FIFO solange ausgelesen, bis der Block mit der Endekennung (End of Frame (eof)) = „1“ erkannt wurde. Die beiden „bv“-Bit stehen für „byte valid“ und geben an, wie viele der 4 Byte eines Blocks zum Frame gehören - also gültig sind. Der Wert für die „bv“-Bit wird mittels Modulo 4-Kodierung bestimmt.

C.1.3 Bearbeitungsstrategie redundanter Aufgaben

Die Bearbeitung redundanter Aufgaben führt zu einem erhöhten Hardware-Bedarf und muss deshalb auf ein Minimum reduziert werden. Redundante Aufgaben entstehen vor der Bewertung der Eingangsdaten durch die Filterstufen bspw. während der Erkennung

von Frame-Parametern als Teil der Paketklassifizierung. Die Erkennung von Frame-Parametern und deren Zuordnung zu einem eindeutigen Datenstrom wird als Paketklassifizierungsproblem bezeichnet und besteht aus den Teilaufgaben:

1. Erkennung von Frame-Parametern zur Bildung eines eindeutigen Schlüssels (*Flow ID*)
2. Zuordnen einer Regel/eines Regelsatzes zu einer *Flow ID*

Alle Systeme im SecAN-Gesamtsystem orientieren sich an unterschiedlichen Frame-Parametern und müssen mindestens die erste Teilaufgabe des Paketklassifizierungsproblems lösen. Das Firewall-System muss auch die zweite Teilaufgabe des Paketklassifizierungsproblems durchlaufen und anhand der *Flow ID* entsprechende Filterregeln laden und anwenden.

Im Hinblick auf eine Hardware-schonende und latenzarme Implementierung wird nicht jedes System separat eine Paketklassifizierung durchführen. Vielmehr wird diese Aufgabe auf zwei kooperierende Module innerhalb eines Paketklassifizierungssystems aufgeteilt. Um die Suche nach Frame-Parametern, deren Extraktion, Bildung des Frame-Parametersets und der *Flow ID* kümmert sich die *Packet Classification Engine*. Für den Firewall-Prototyp wird, basierend auf der *Flow ID*, die *Rule Search Engine* einen speziellen Firewall-Regelsatz (Ruleset) suchen und bereitstellen. Da die *RSE* ausschließlich für die Firewall-Regeln verantwortlich ist, wird sie weder vom Web-Filter noch vom Intrusion-Detection-System benötigt.

C.2 Realisierung der Packet Classification Engine

Aufgrund der zu erfüllenden Aufgaben der *PCE* entsteht das in Abbildung C.3 dargestellte Blockschaltbild. Danach werden vier verschiedene funktionale Module mit unterschiedlicher Komplexität benötigt. Neben dem Konfigurationsmodul (Config), existiert ein Modul zur Datenerfassung (Data In & Hash). Vor der Untersuchung des Datenstromes mittels Firewall-Modul muss der individuelle Regelsatz aus der *RSE* geladen werden. Die Kommunikation mit der *RSE* erfolgt durch das *Packet Classification Engine*-Modul. Letztendlich wird der Datenversand durch das Modul *Data Out* an die nachfolgende Filterstufe übergeben. Die „Busy“-Signale dienen als Backpressure-Signale. Sollte ein nachfolgendes Modul „beschäftigt“ sein, muss das gegenwärtige Verarbeitungsmodul den

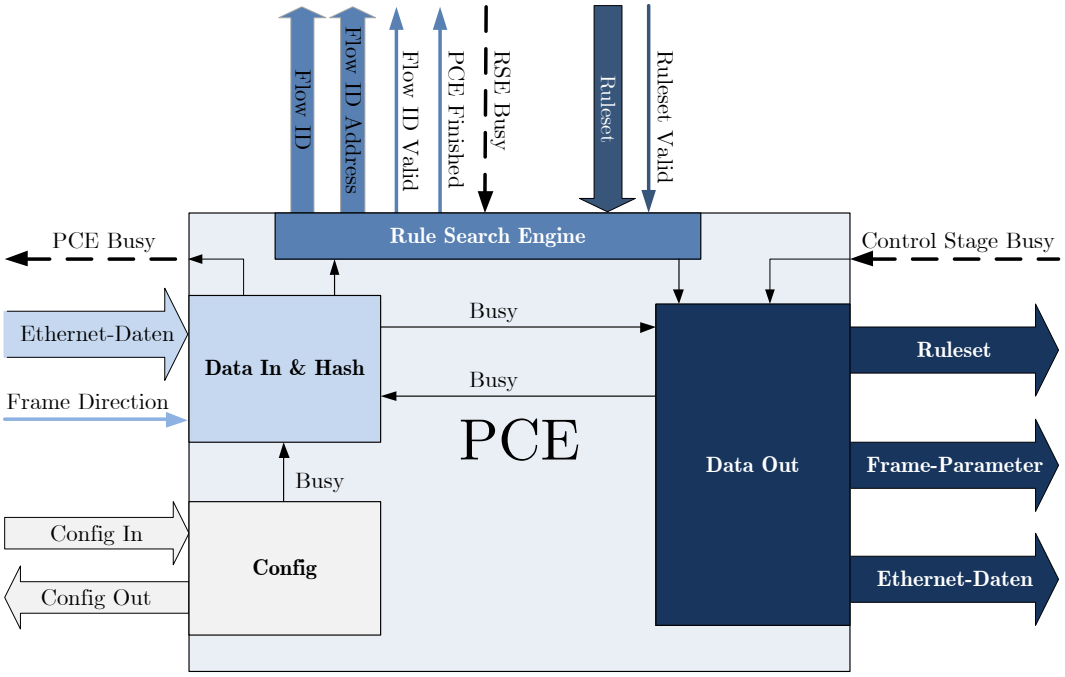


Abbildung C.3: Blockschaltbild der Packet Classification Engine

Datenaustausch so lange unterbinden, bis das Folgemodul durch das „Busy“-Signal anzeigt, dass es Daten empfangen kann.

Konfiguration der PCE:
Über den „Config In“-Port (vgl. Abbildung C.3) werden (neue) Konfigurationsdaten an die *PCE* übergeben. Es handelt sich dabei um die beiden *Flow ID Masks* für Up- und Downstreams.

Konfigurationsdaten liegen genau für einen Takt am Eingang des *Config*-Moduls an. Über ein *Valid*-Signal wird angezeigt, in welchem Takt die Konfigurationsdaten übernommen werden müssen. Die zu konfigurierenden *Flow ID Masks* werden in separate Register gespeichert. Von dort werden sie durch das *Data In*-Modul für die Frame-Verarbeitung ausgewählt. Lesende Anfragen von Konfigurationsdaten erkennt das *Config*-Modul ebenfalls. In diesem Fall muss das Bit-Muster der beiden *Flow ID Masks* aus den Registern gelesen und über den *Config Out*-Port versandt werden.

Vorverarbeitung von Ethernet-Daten:
Nach Initialisierung der Hardware sowie nach jeder Verarbeitung eines Frames signalisiert das *Data In & Hash*-Modul zum *Framebuffer*-Modul, dass es bereit ist, neue Daten zu erfassen. Mit dem ersten Datenwort zeigt die *PCE* an, dass sie ab sofort beschäftigt ist (*PCE Busy*) und dementsprechend keine neuen Daten aufnehmen kann. Jedoch wird der aktuelle Frame bis zum *eof*-Signal vollständig erfasst, zwischengespeichert und verarbeitet.

Abhängig von der Empfangsrichtung des Frames (*Frame Direction*) wird die relevante *Flow ID Mask* gewählt. Anschließend beginnt die Extraktion der in Tabelle B.1 vorgestellten Frame-Parameter. Alle in der *Flow ID Mask* aktivierten Frame-Parameter werden zu einer für den Datenstrom individuellen 126 Bit langen *Flow ID* zusammengeführt. Nachdem alle in der *Flow ID Mask* aktivierten Frame-Parameter in die *Flow ID* integriert wurden, werden ungenutzte Bit-Positionen mit Nullen aufgefüllt. Flow-Parameter, die nicht im aktuellen Frame enthalten sind, werden übersprungen. Jedoch entsteht durch das Fehlen von Parametern eine ungültige *Flow ID*. Die Gültigkeit einer *Flow ID* wird mittels *Flow ID Valid*-Port gegenüber der *RSE* signalisiert. Wurden alle notwendigen Parameter zu einer gültigen *Flow ID* erfasst, muss die *Flow ID* „gehasht“ werden. Der Hash-Wert bildet die initiale Startadresse für den Speicher in der *RSE*.

Hashen der Flow ID:
Im Abschnitt 4.3.2.3 wurden verschiedene Hashing-Methoden vorgestellt, die sich gut in

Hardware umsetzen lassen. Variationen in der Qualität sowie im Ressourcenverbrauch unterscheiden die Hash-Funktionen. Ferner wurde im Abschnitt 3.1.1 festgelegt, dass 32.000 unterschiedliche Verbindungen (und damit Flows) unterschieden werden müssen. Da das Hash-Ergebnis als initiale Speicheradresse dienen soll, müssen somit mindestens 16 Bit Hash-Werte erzeugt werden. Alle vier in Abschnitt 4.3.2.3 vorgestellten Verfahren sind dazu in der Lage.

Gegen das H_3-Verfahren spricht der relativ hohe Ressourcenaufwand im Verhältnis zu den übrigen drei Verfahren bei einer kleinen Liste von 32.000 Einträgen. Ressourcengünstiger lässt sich die XOR-Methode umsetzen. Im direkten Vergleich zwischen der XOR- und der CRC-Methode (vgl. Tabellen C.2 und C.3) erzielt die CRC-Methode bessere Hash-Ergebnisse. Aus diesem Grund wird auch die XOR-Methode nicht weiter berücksichtigt, sondern die CRC-Methode ausgewählt. Für die 32.000 Speichereinträge ist die CRC16-Methode ausreichend. Sie ist durch Tauschen des Generatorpolynoms leicht auf CRC32 erweiterbar. Somit stellt sie die geeignete Methode dar, um zukünftig auch Speicheradressen für mehr als 2^{16} Verbindungen zu generieren. Dementsprechend wird für die Berechnung der initialen Speicheradressen die CRC16-Methode mit dem Generatorpolynom $x^{16} + x^{15} + x^2 + 1$ verwendet. Die Berechnung (CRC16(Flow ID)) erfolgt unmittelbar, wenn alle Frame-Parameter in die *Flow ID* aufgenommen wurden bzw. wenn die Payload der 4. OSI-Schicht beginnt.

Ist die Berechnung beendet, erfolgt die Übergabe der *Flow ID*, deren Adresse (CRC16 (Flow ID)) und die Information über die Gültigkeit der Daten an die *RSE*. Ein Handshake durch die Ports „*PCE* Finished“ und „*RSE* Busy“ garantiert, dass durch die *PCE* ermittelten Daten ausreichend lange anliegen, bevor sie innerhalb eines Taktes übernommen werden.

Datenversand in Richtung der Filterstufen:
Der Versand der Daten in Richtung *PPE* wird initialisiert, wenn das Frame-Parameterset vervollständigt wurde bzw. der Regelsatz von der *RSE* eintrifft. Das Eintreffen des Regelsatzes wird durch den „Ruleset Valid“ Port angezeigt (vgl. Abbildung C.3). Durch das „Ruleset Valid“-Signal gesteuert, wird das *Ruleset* vom *Data Out*-Modul empfangen. Das Modul ist in der Lage, das gesamte *Ruleset* zwischenzuspeichern. Dies wird realisiert, wenn das *Control Stage Busy*-Signal anzeigt, dass die folgende Filterstufe nicht bereit ist, Daten von der *PCE* zu empfangen.

Empfängt die *PCE* das *Ruleset*, sendet sie es bei Freigabe des *Control Stage Busy*-Signals parallel mit den extrahierten Frame-Parametern und dem Frame an die erste

Filterstufe. Mit der Übertragung des ersten Datenwortes wird auch das „*PCE* Busy“-Signal anzeigen (vgl. Abbildung C.3), dass die *PCE* einen neuen Daten-Frame aufnehmen und klassifizieren kann.

C.3 Realisierung der Rule Search Engine

In Abbildung C.4 sind die Interfaces, die funktionalen Module und Speicher sowie die wesentlichen Kommunikationswege der *Rule Search Engine* dargestellt.

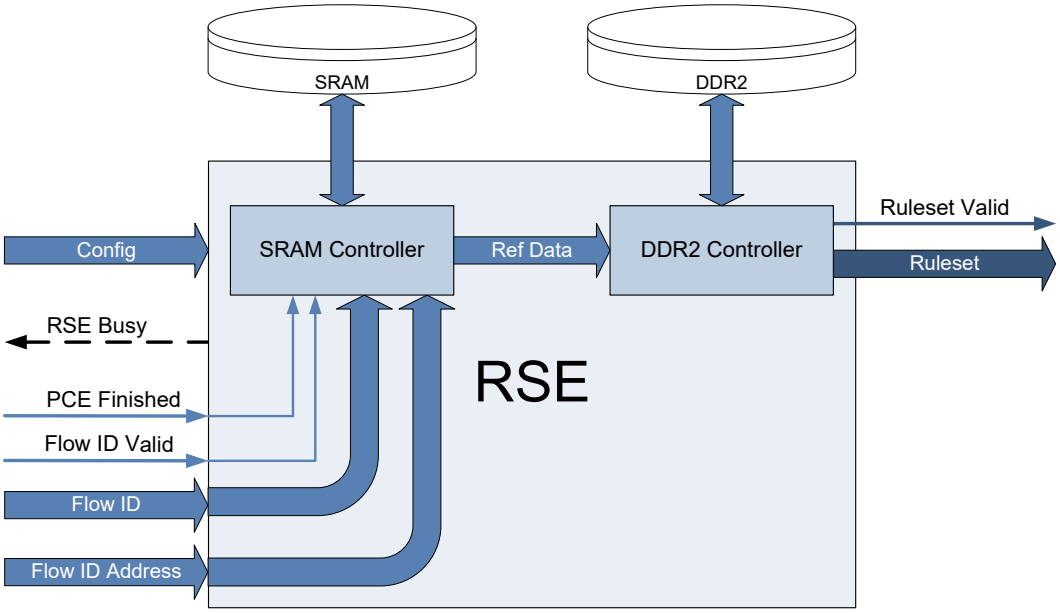


Abbildung C.4: Blockschaltbild der Rule Search Engine

Auf die *RSE* wird sowohl schreibend als auch lesend zugegriffen. Schreibende Zugriffe erfolgen ausschließlich während der Konfigurationsphase. Dabei wird anhand der Type-Felder der Konfigurationsdaten unterschieden, für welchen Speichercontroller die gegenwärtigen Daten bestimmt sind. Für lesende Anfragen durch die *PCE* muss der Modulausgang *RSE Busy* ausgewertet werden. Dieser ist mit der *PCE* verbunden und wird beim Übernehmen von Anfragedaten auf „Busy“ bzw. beim Versenden des Regelsatzes auf „Not Busy“ gesetzt.

C.3.1 Mapping von Flow ID zu Rule ID im SRAM:

Das Mapping der *Flow IDs* zu Regelsätzen erfolgt im ZBT-SRAM. Dazu wird, wie in Abbildung C.5 dargestellt, der gesamte Speicher in logische Blöcke unterteilt. Einem logischen Block werden sechs physikalische Speicheradressen zugeordnet, wobei jede davon 32 Bit Daten aufnehmen kann.

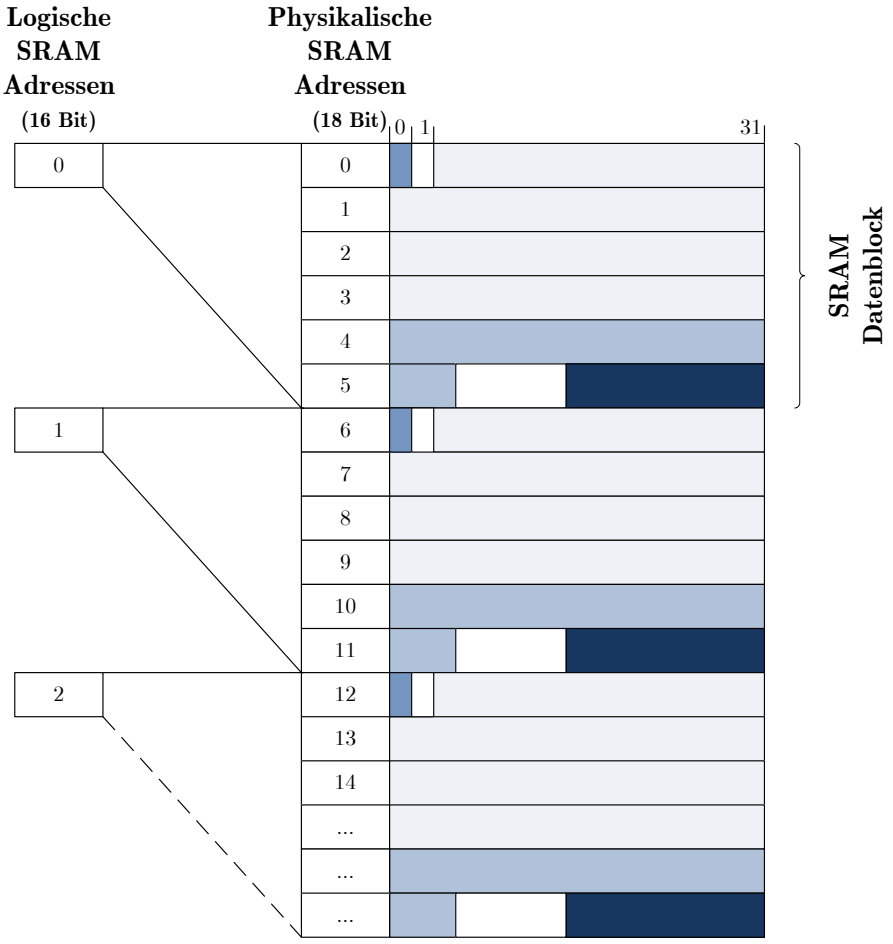


Abbildung C.5: Aufbau von Speicherblöcken im ZBT-SRAM

Ferner enthält jeder Speichereintrag vier Informationsgruppen:

1. Eine Information, ob die Speicherstelle belegt ist (Bit-Position 0)
2. Die *Flow ID* als Referenzwert, da durch das Hashen der *Flow IDs* in der *PCE* Kollisionen auftreten können (Bit-Positionen 2 - 128). Bei der folgenden Suche

wird eine Kollision erkannt, wenn sich die von der *PCE* übergebenen von der Referenz-*Flow ID* unterscheidet.

3. Referenzinformationen zum Regelsatz (*Rule ID*): Es handelt sich dabei um die Startadresse sowie die Länge des Regelsatzes im DDR2-Speicher (Bit-Positionen 129 - 163)
4. Kollisionsauflösung: Die Bit-Position 176 zeigt, ob bei einer Kollision ein weiterer SRAM-Eintrag vorhanden ist und an welcher SRAM-Adresse sich dieser befindet (Bit-Positionen 177 - 192).

Alle beschriebenen Informationsgruppen sind in Abbildung C.5 farblich markiert. Ungenutzte Bit-Positionen eines Speicherblocks sind für zukünftige Erweiterungen reserviert.

C.3.2 Realisierung des ZBT-SRAM-Controllers

Beim ZBT-SRAM-Controller handelt es sich um einen für das SecAN-Projekt entwickelten Speicher-Controller, der ohne ein Bussystem direkt auf den Speicher zugreifen kann. Durch den Wegfall des aufwendigen Prozesses der Busarbitrierung wird die Zeit, um einen vollständigen Speichereintrag auszulesen, von über 100 Takten auf sieben Systemtakte deutlich reduziert. Der ZBT-SRAM-Controller besteht im Wesentlichen aus den vier funktionalen Modulen: *SRAM Config*, *Search PCE*, *Arbiter* und *Driver* (vgl. Abbildung C.6).

Der Arbiter koordiniert den Zugriff auf den ZBT-SRAM. Er überführt logische in physikalische Adressen und übergibt bzw. übernimmt Speicherdaten vom Driver. Im Driver-Modul wird mittels Tri-State Buffer zwischen lesenden und schreibenden Zugriffen umgeschaltet.

Die Konfigurationsphase beginnt mit einem Nullen aller Speicherstellen des ZBT-SRAM. Durch die Initialisierung werden falsch zu interpretierende Speichereinträge vermieden. Anschließend werden die Konfigurationsdaten an das *SRAM Config* Modul und an das *Search PCE* Modul übergeben. Auch hier findet wieder eine Unterscheidung anhand des Type-Feldes statt (vgl. Abschnitt 4.2.1). Einerseits werden die Zuordnungsdaten von *Flow IDs* zu *Rule IDs* und andererseits die *Default Rule ID* im ZBT-SRAM abgelegt.

Darauf folgend können an die *RSE* lesende Speicheranfragen gestellt werden, wenn das *RSE Busy*-Signal dies erlaubt. Lesende Anfragen werden von der *RSE* anhand des Zustands des *PCE Finished*-Signals erkannt. Gleichzeitig wird über den *Flow ID Valid-*

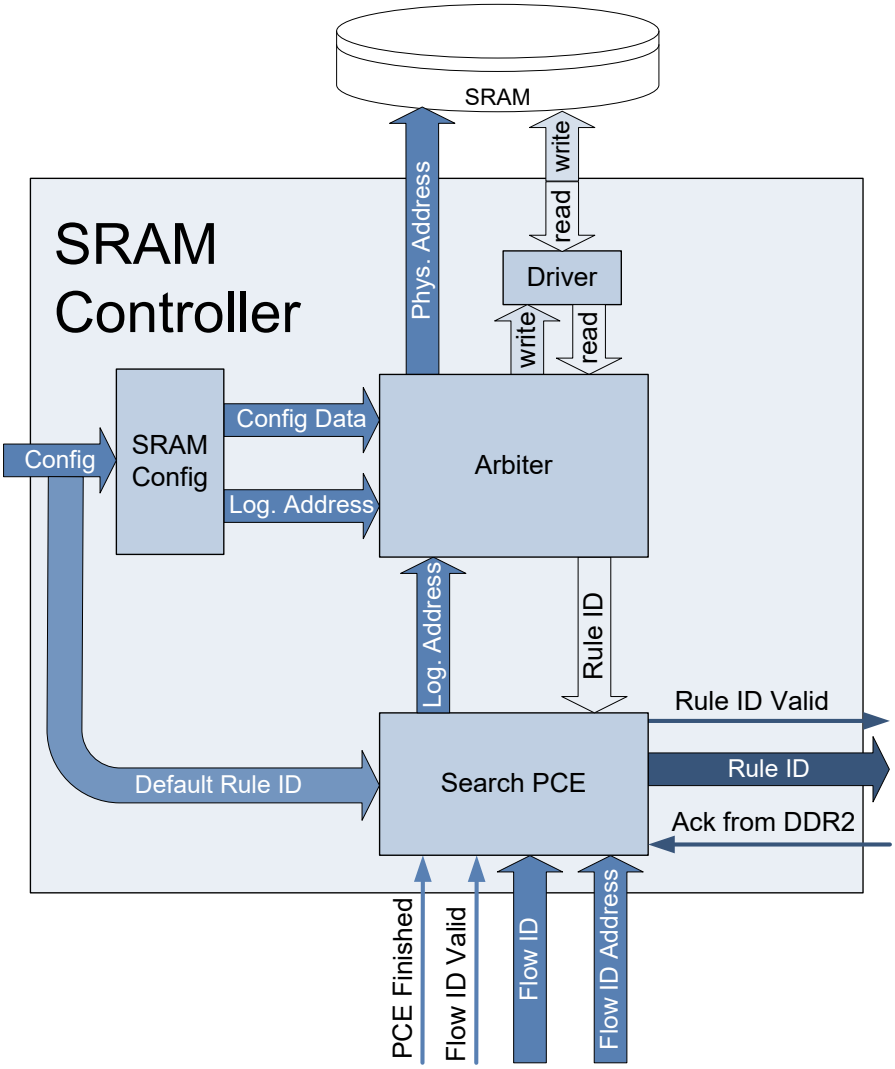


Abbildung C.6: Aufbau des modifizierten ZBT-SRAM Controllers

Eingang angezeigt, ob alle Parameter, die für eine gültige *Flow ID* notwendig sind, aus dem aktuellen Datenstrom extrahiert werden konnten. Ist dies nicht der Fall, wird die *Flow ID* durch die *PCE* als ungültig eingestuft. In der Folge wird die *RSE* die *Default Rule ID* an den DDR2-Speicher-Controller ausliefern. Die extrahierte *Flow ID* sowie deren Hash-Wert werden über die Eingänge *Flow ID* und *Flow ID Address* an den ZBT-SRAM der *RSE* übergeben.

Sind alle Daten vom *PCE Search* Modul ausgewertet, beginnt die Suche im ZBT-SRAM. Dabei dient der 16 Bit breite Hash-Wert der *Flow ID* als initiale logische Speicheradresse. Zunächst wird überprüft, ob an der jeweiligen Speicherstelle ein gültiger Eintrag vorhanden ist (vgl. Konzipierung im Abschnitt C.3.1). Ist dies nicht der Fall, wird die *Default Rule ID* ausgewählt und an den DDR2-Controller übertragen. Wurde hingegen eine gültige Speicherstelle identifiziert, wird der gesamte Speichereintrag gelesen. Währenddessen wird die gespeicherte *Flow ID* mit der übergebenen *Flow ID* verglichen. Sind beide identisch, werden die ausgelesenen *Rule ID*-Daten an den DDR2-Controller übergeben. Anderenfalls wird die ebenfalls gespeicherte Adresse zur Kollisionsauflösung als nächste Speicheradresse ausgewählt. Jede Suche endet, wenn die entsprechende *Rule ID* gelesen bzw. keine Folgeadresse verfügbar ist. Im letzten Fall wird die *Default Rule ID* ausgewählt.

C.3.2.1 Realisierung des DDR2-Controllers

Wie bereits in Abschnitt D.3 beschrieben, wird der verwendete DDR2-Speicher [Tec13] vom Memory Interface Generator (MIG)v3.3 unterstützt. Somit ist es möglich, einen Speicher-Controller zu generieren, der das interne Bussystem (PLB) umgeht und direkt auf den DDR2-Speicher zugreift. Während eine Speicheranfrage über den PLB ca. 200 Systemtakte benötigt, greift der generierte DDR2-Controller innerhalb von 20 Systemtakten auf den Speicher zu.

Die vom MIG generierten vhd-Dateien bieten Interfaces zum Schreiben und Lesen des Speichers. Da der Speicher mit einer Frequenz von 200 MHz betrieben wird und die systeminterne Frequenz davon abweichen kann, wurde zusätzlich ein DCM generiert. Dieser gestattet es, die verschiedenen Frequenzen beider Clock Domains aufeinander abzustimmen. Somit sind die Voraussetzungen geschaffen, eine Hardware (im Weiteren *User Logic* genannt) zu entwickeln. Sie wird vor den generierten Speicher-Controller geschaltet und nimmt Anfragen vom System entgegen. Ferner puffert sie Antworten vom

Speicher zwischen, bevor sie an das Zielmodul ausgeliefert werden. In Abbildung C.7 ist

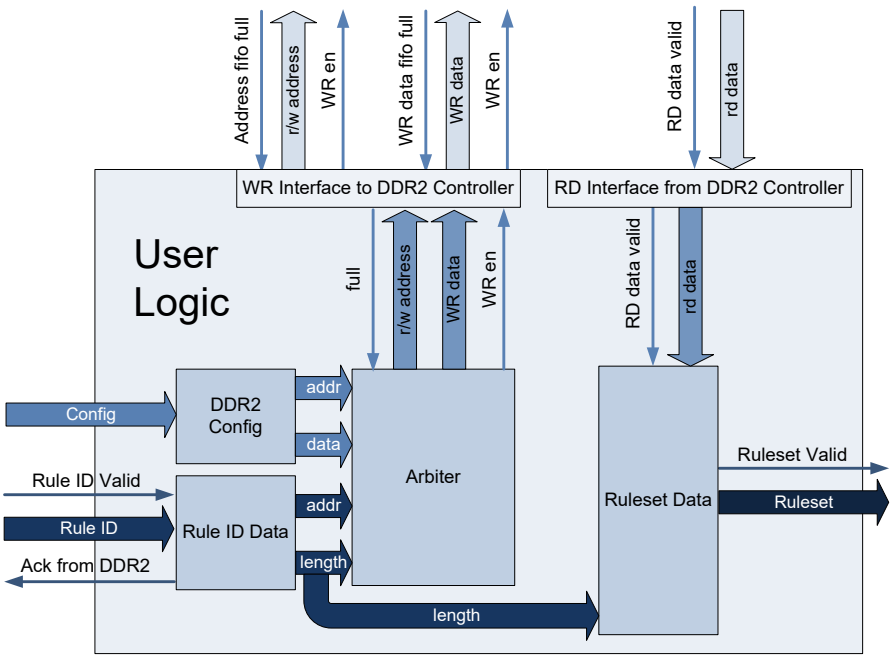


Abbildung C.7: Aufbau des modifizierten DDR2 Controllers

das Blockschaltbild der *User Logic* dargestellt. Zum einen verfügt sie über Schnittstellen zur Konfiguration des DDR2-Speichers und zum Empfang der *Rule ID*-Daten. Zum anderen besitzt sie getrennte Schnittstellen, um Anfragen an den Speicher zu stellen bzw. dessen Antworten zu empfangen. Darüber hinaus bietet sie eine Schnittstelle, um die gelesenen Daten an das nächste Modul - die *PCE* - zu versenden.

Während der Konfigurationsphase empfängt die *User Logic* die Regelsatzdaten. Das *DDR2-Config*-Modul innerhalb der *User Logic* nimmt die Konfigurationsdaten in Empfang, dekodiert sie und leitet sie an das *Arbiter*-Modul des DDR2-Controller weiter. Dieses koordiniert den Zugriff auf die Schnittstelle zum Speicher. Ein *Driver*-Modul wie beim ZBT-SRAM-Controller wird nicht benötigt, da bereits separate Schnittstellen für das Schreiben zum und das Lesen aus dem Speicher existieren. Die Aufgabe des *Driver*-Moduls übernimmt somit der vom MIG erzeugte Speicher-Controller selbstständig.

Lesende Anfragen beziehen sich auf das Auslesen der Regelsätze. Dementsprechend muss eine *Rule ID* vom ZBT-SRAM-Controller empfangen und durch das *Rule ID Data*-Modul in Startadresse und Längeninformation zerlegt werden. Das *Rule ID Data*-Modul hat

ferner die Aufgabe, Folgeadressen zu berechnen. Alle Adressinformationen sowie der Befehl, von diesen Adressen zu lesen, werden über das *Arbiter*-Modul an das Interface zum Speicher weitergeleitet.

Ab dem 21. Takt wird das Ruleset vom Speicher ausgeliefert. Dabei wird die Gültigkeit der Daten durch das *RD data valid*-Signal angezeigt. Wechselt das *RD data valid*-Signal zu „ungültig“, muss dies nicht zwangsläufig bedeuten, dass der Regelsatz vollständig aus dem Speicher gelesen wurde. Vielmehr kann es zu einer Unterbrechung durch einen Refreshzyklus des DDR2-Speichers (alle 64 ms [Tec13]) zu diesem Effekt kommen. Da jedoch die Länge des erwarteten Regelsatzes im *Ruleset Data*-Modul bekannt ist, kann leicht festgestellt werden, ob der Regelsatz vollständig ist. Mit dem Weiterleiten wird das Ruleset in den dpt konvertiert, sodass nachfolgende Module den Beginn und das Ende des Rulesets detektieren können.

C.4 Weiterführende Details zum Firewall-System

C.4.1 Grundlagen Firewalls

Eine FW ist eine Sicherheitsschleuse zwischen TCP/IP-Netzen zur Kontrolle des Datenverkehrs. Alle modernen Betriebssysteme besitzen eine lokale Software-Firewall. Darüber hinaus kann eine FW als dedizierte Hardware-Lösungen oder als Erweiterung von Netzwerkgeräten, wie z. B. Routern oder Proxy-Servern, vorhanden sein. Bei Firewalls wird grundsätzlich zwischen Filtern (Screens) und Gateways unterschieden [Fuh00, BW02]. Diese werden nachfolgend dargestellt:

Packet Filter: Sie arbeiten, wie in Abschnitt 2.4.2.2 beschrieben, nach den Methoden der statischen und der dynamischen Paketfilterung. Sie selektieren Parameter aus den Header-Feldern von der Netzwerk- bis zur Transportschicht des OSI-Modells und vergleichen diese Informationen mit definierten Regeln. Bei einem Treffer wird die Aktion der Regel ausgeführt, wodurch der Datenstrom verworfen oder weitergeleitet wird. Sollte keine der Regeln greifen, wurde eine „Standard-Aktion“ definiert, welche dann auf den Frame angewendet wird.

Gateways: Sie arbeiten als Relais für Dienste. Zwei bekannte Vertreter sind Application Level Gateway (ALG) und Verbindungs-Gateways (oder auch Circuit Level Gateway (CLG)).

Ein Application Level Gateway (auch Application Layer Firewall (ALF) genannt) ist ein System, welches Überwachungsfunktionen auf der Anwendungsebene durchführt. ALGs sind Proxy-Server und befinden sich zwischen den Kommunikationspartnern. Sie stellen stellvertretend eine Verbindung für diese her. Ferner sind sie in der Lage, Authentifizierungs-Verfahren zu berücksichtigen und eine inhaltliche Filterung durchzuführen. Um den Datenstrom einer Anwendung vollständig zu analysieren, müssen sie daher eine Vielzahl von Anwendungsprotokollen beherrschen. Über eine Analyse hinaus können ALGs Netzwerkdaten manipulieren, verwerfen und gegebenenfalls Einfluss auf die Verbindung nehmen.

CLGs werden eingesetzt, um Datagramme der Transportschicht, die unter Umständen fragmentiert wurden, zusammenzubauen. Weiterhin können CLGs eine Network Address Translation (NAT) durchzuführen. NAT ermöglicht das Verbergen von internen IP-Adressen vor dem Internet und trägt somit zur Erhöhung der Netzwerksicherheit bei. Darüber hinaus besitzen CLGs Regeln für eine Port-basierte Steuerung des Datenverkehrs.

Sowohl ALG als auch CLG besitzen Gemeinsamkeiten. In der Praxis werden ALG für den Datenverkehr nach intern (zum sicheren Netz) und CLG nach extern (zum unsicheren Netz) eingesetzt.

C.4.1.1 Grenzen des Firewall-Schutzes

Richtig konfigurierte Firewalls können Protokollverletzungen unterbinden und durch Address- und Port-Sperren Kommunikationswege blockieren. Gegen bestimmte Mechanismen sind sie jedoch machtlos. Existiert neben dem Datenweg durch die FW ein zweiter ungeschützter Verkehrsweg in das Netzwerk, so spricht man von einer „Backdoor“. In diesem Fall ist die FW machtlos, da sie lediglich den durch sie geleiteten Datenverkehr kontrollieren kann. Bei internen Angriffen, die direkt auf ein System und nicht durch die FW geleitet werden, ist diese ebenfalls wirkungslos.

Ferner ist eine FW bei einer Durchtunnelung unwirksam. Bei einer Durchtunnelung bettet eine Software die zu versendenden Daten in den Datenstrom ein. Dabei wird die FW nicht aktiv, solange die Protokollgrenzen von der Software eingehalten werden. Ein namhaftes und erfolgreiches Beispiel für eine Firewall-Umgehung ist die VoIP-Software „SKYPE“.

Des Weiteren können verschlüsselte Verbindungen wie Hypertext Transfer Protocol Secure (HTTPS) die inhaltliche Untersuchung des Datenpaketes durch eine FW vollständig unterbinden. Genauso machtlos ist eine FW, wenn es sich um sicherheitsrelevante Mängel innerhalb von Softwarelösungen handelt. Diese können durch eine FW nicht behoben werden. Sie kann lediglich auf bekannte Sicherheitslücken reagieren, jedoch keine unbekannten Sicherheitsmängel beseitigen.

Wie die aufgeführten Beispiele zeigen, kann aber auf keinen Fall von vollkommener Netzwerksicherheit ausgegangen werden. Aus diesem Grund ist es zwingend notwendig, weitere Netzwerksicherheitsmerkmale parallel zu nutzen. Eine sehr sinnvolle Ergänzung zu einer FW sind Systeme, die ähnlich wie ein Application Level Gateway eine DPI durchführen und so Bedrohungsmuster in den höheren OSI-Schichten eines Datenpaketes aufspüren. Solche Systeme sind bspw. Web-Filter (vgl. Abschnitt 4.6) und Intrusion Detection Systeme (vgl. Abschnitt 4.7). Weitere Informationen zum Themenbereich FW sind in [Wel00a, Wel00b, Net10a, Spe11] zu finden.

Aufbau, Funktionsweise und Administration der Linux-Firewall: Weil die „überwiegende Mehrheit aller Server im Internet und Intranet (...) auf UNIX-Systemen“ [HP03] basieren, sind die Prinzipien der Linux-FW von grundlegender Bedeutung für die vorliegende Arbeit. Netfilter, xtables und iptables sind drei wichtige Software-Komponenten dieser Firewall, auf die im Folgenden näher eingegangen wird. Als Bestandteil des Linux-Kernels ist die „Netfilter“-Software das Filter-Framework zur Untersuchung von Netzwerkpaketen. Es wurde unter der GNU General Public License (GPL) lizenziert und wird seit der Kernel-Version 2.4.x als Basiskomponente des Linux-Firewall-Systems eingesetzt. Die Software besitzt unter anderem die folgenden Eigenschaften [Net11]:

- Zustandslose IPv4- und IPv6-basierte Paketfilterung
- Zustandsbehaftete IPv4- und IPv6-basierte Paketfilterung
- NAT
- Network Address Port Translation (NAPT)
- Eine flexible und modulare Infrastruktur
- Eine große Anzahl von Plugins
- Eine umfangreiche API für Erweiterungen

Wenn ein Netzwerkpaket vom System empfangen, versandt oder weitergeleitet wird, ruft der Kernel die Netfilter-Software auf. Sie definiert sogenannte „Hooks“ im Linux-Netzwerkstack (vgl. Tabelle C.1). An den Hooks können sich weitere Kernelmodule anmelden, an die die Netzwerkpakete anschließend delegiert werden. Das Kernel-Modul „x_tables“ enthält den gemeinsamen Code der Kernel-Module „ip_tables“, „ip6_tables“, „arp_tables“ und „ebtables“ und nutzt das Hook-System vom Netfilter. Um den Datenstrom zu filtern und zu manipulieren, stehen in den „x_tables“-Modulen Tabellen (tables) mit Ketten (chains) von Filterregeln (rules) zur Verfügung. Abbildung C.8 zeigt den beschriebenen Aufbau einer Linux-FW und verdeutlicht somit den Zusammenhang zwischen Netfilter, x_tables sowie den Tabellen, Ketten und Regeln innerhalb der Kernel-Module.

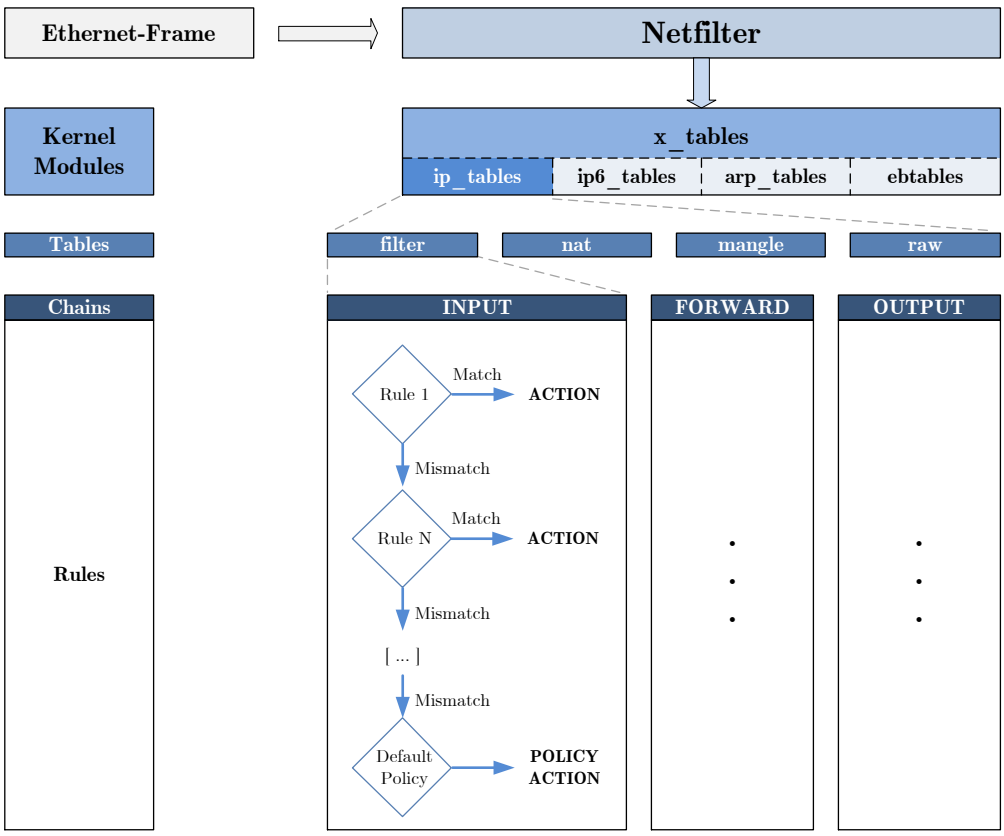


Abbildung C.8: Aufbau der Linux-Firewall

Die grundsätzlichen Filterregeln werden in die folgenden vier elementaren Tabellen (filter, nat, mangle und raw) gruppiert [Spe06].

- filter: Standardtabelle für Filterregeln
- nat: Regeltabelle für die Adressumsetzung (NAT) sowie spezielle Regeln für die Weiterleitung (Port Forwarding)
- mangle: Die Tabelle enthält Regeln zur Paketmanipulation
- raw: Stellt Ausnahmeregeln für das „Connection Tracking“ bereit

In jeder Tabelle sind einige der fünf vordefinierten Ketten hinterlegt (vgl. Tabelle C.1). Alle Netzwerkpakete, unabhängig ob eintreffend, ausgehend oder lokal, durchlaufen mindestens eine Kette. Über die Standardketten hinaus können Ketten auch frei definiert werden.

In den Ketten werden eine oder mehrere Regeln definiert. Die Regeln entsprechen dem Aufbau: „Wenn *Bedingung* dann *Aktion*“. Häufig werden die Aktionen „ACCEPT“, „DROP“, „REJECT“ und „LOG“ ausgeführt. Stimmt die Filterbedingung überein, können zusätzlich aber auch Sprung-Befehle ausgeführt werden. Dabei wird zu anderen Regeln (goto) bzw. zu alternativen Ketten gewechselt (jump). Wird das Paket während der Verarbeitung nicht verworfen, wird an die Quelle des ersten Sprung-Befehls zurück gewechselt. Generell erfolgt die Bearbeitung eines Pakets in sequenzieller Reihenfolge der Regeln der Kette. Auf diese Weise entsteht eine Priorisierung der Regeln. Wird eine anwendbare Regel gefunden („first match“), wird die Aktion ausgeführt und die weitere Filterung beendet.

Beispiel einer Regel: Falls beim erste Treffer alle ausgehenden FTP-Verbindungen aus einem dedizierten Subnetz erlaubt werden und die folgende Regel verbietet alle ausgehenden FTP-Verbindungen, so werden alle FTP-Verbindungen erlaubt. Ein Tausch der Regeln hätte zur Folge, dass keine ausgehenden FTP-Verbindungen erlaubt werden (wodurch die zweite Regel überflüssig würde). Durchläuft ein Paket eine Kette vollständig - es wurde also keine passende Regel für das Paket identifiziert - greift die „Default Policy“. Das Paket kann dann z. B. verworfen (Drop) oder weitergeleitet (Accept) werden.

Netfilter und „x_tables“ sind keine starren Konstrukte. Weiterentwicklungen werden durch ein „Core Team“ [Net11] vorangetrieben. Darüber hinaus ist jedem Anwender

Ketten	Zugehörige Tabellen	Beschreibung
PREROUTING	nat, mangle, raw	Gilt für alle Netzwerkpakete, bevor diese geroutet werden.
INPUT	filter, mangle	Gilt für alle eingehenden Netzwerkpakete, die einem lokalen Prozess zugeordnet werden können.
FORWARD	filter, mangle	Gilt für alle Netzwerkpakete, die weitergeleitet werden müssen.
OUTPUT	filter, nat, mangle, raw	Gilt für alle ausgehenden Netzwerkpakete, die vom System generiert wurden.
POSTROUTING	nat, mangle	Gilt für alle Netzwerkpakete, nachdem eine Routingentscheidung getroffen wurde.

Tabelle C.1: iptables-Ketten und deren Vorkommen in iptables Tabellen

freigestellt, eigene Module zu entwickeln. Abhängig vom jeweiligen „Release“ wird ein „Howto“ zur Entwicklung der Erweiterungen (Plugins) herausgegeben [EB11].

Administration der Linux-Firewall:
Um die verschiedenen Kernel-Module zu administrieren, wurde zu jedem Modul ein Userspace-Programm entwickelt. Linux-typisch handelt es sich dabei zunächst um Konsolen-Anwendungen. Das zum „x_tables“-Modul „ip_tables“ gehörige Userspace-Programm trägt den Namen „iptables“. Mit Hilfe von iptables werden nun die Tabellen des „ip_tables“-Moduls konfiguriert. Im Laufe der Zeit wurden zu den Userspace-Anwendungen grafische Benutzeroberflächen entwickelt. Die Verwaltung der Regeln wird dadurch wesentlich vereinfacht.

Iptables, wie es heute eingesetzt wird, hat bereits eine lange Entwicklungsphase hinter sich. Der ursprüngliche Paketfilter geht aus der UNIX-Betriebssystemwelt hervor. Die erste Erweiterung wurde in der Linux-Version 2.0 unter dem Namen „ipfwadm“ eingeführt. Daraus wurde in der Kernel-Version 2.2 „ipchains“. Seit der Kernel-Version 2.4 übernahm nun „iptables“ die Firewall-Funktionalität. Im Jahre 2008 wurde seitens des Netfilter Core Teams bereits über den Nachfolger „nftables“ nachgedacht. Die ersten positiven Ergebnisse wurden im Oktober 2010 auf dem „Netfilter Workshop“ in Sevil-

la, Spanien veröffentlicht [McH10]. Im folgenden Abschnitt werden die Strukturen der Regelsätze und der darin enthaltenen Regeln näher beschrieben.

C.4.2 Struktur von Regelsätzen und Regeln

Aus Sicht der *Rule Search Engine* handelt es sich bei allen Regelsätzen um beliebige Binärmuster. Die Firewall-Komponenten müssen die einzelnen Regeln eines Regelsatzes interpretieren können und erwarten einen strukturierten Aufbau. Dieser lehnt sich stark an das in Abschnitt C.4.1.1 beschriebene Linux-basierte x_tables an und ist in Abbildung C.9 dargestellt. Danach besteht ein Regelsatz aus einer Aneinanderreihung von N

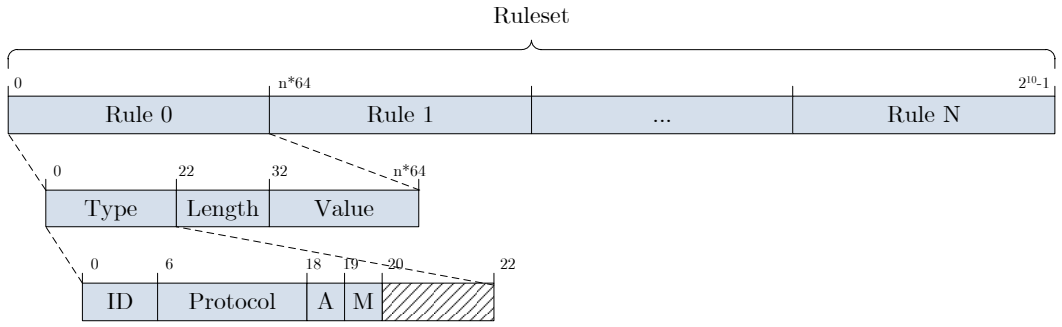


Abbildung C.9: Aufbau des Rulesets

Regeln.

C.4.2.1 Aufbau einer Regel im Regelsatz

Jede Regel besitzt einen fixen *Type Length Value (TLV)*-Aufbau. Das *Type*-Feld hat eine Länge von 20 Bit. Darin enthalten sind:

- eine systemweit eindeutige ID, die die Firewall-Filterstufe (z. B. Destination Port Filter) identifiziert (6 Bit)
- *one hot*-kodiert die OSI-Layer 4 Protokolle, für die die aktuelle Regel gilt (12 Bit)
- die Aktion (*ACCEPT* oder *DROP*), die ausgeführt wird (**A** in Abbildung C.9) (1 Bit)
- das Match-Bit (**M** in Abbildung C.9), das anzeigt, ob die Aktion bei einem Match bzw. Mismatch ausgeführt wird (1 Bit)

Daran anschließend folgen zwei Padding-Bit. Diese können für zukünftige Erweiterungen genutzt werden. Ein 10 Bit langes *Length*-Feld kodiert die Länge der gesamten Regel. Abschließend folgt das variabel lange *Value*-Feld. Dessen Daten vergleicht die Filterstufe mit den Referenzdaten aus dem Frame-Parameterset.

Da der Regelsatz im dpt (vgl. Abbildung C.2) übertragen wird, muss seine Länge ein ganzzahliges Vielfaches von 32 Bit ergeben. Daher ist es sinnvoll, dass auch die Länge jeder Regel ein ganzzahliges Vielfaches von 32 Bit aufweist. Für den Fall, dass nicht alle Bit-Stellen einer Regel benötigt werden, müssen diese mit Padding-Bits aufgefüllt werden. Mit diesen Maßgaben ergibt sich die maximale Anzahl an Regeln, die ein Regelsatz nach den Gesamtvorgaben besitzen kann:

- Minimale Länge einer Regel (s. TLV-Aufbau einer Regel) = 64 Bit (8 Byte)
- Maximale Länge eines Regelsatzes = 1024 Byte
- Maximale Anzahl der Regeln eines Regelsatzes: 1024 Byte/8 Byte = 128 Byte

Da bisher acht Filterstufen definiert sind und jede Filterstufe lediglich eine Regel bearbeitet, ist diese Anzahl zunächst ausreichend. Weiterhin handelt es sich nicht um starre Regeln, die für das gesamte Firewall-System gelten, sondern um individuell auf einen ISP-Endkunden abgestimmte Regeln. Darüber hinaus handelt es sich bei den aufgeführten Zahlenbeispielen um Empfehlungen des Autors, die nicht als fixe Werte zu verstehen sind. Selbstverständlich ist das Hardware-System in der Lage, auch deutlich längere (unbegrenzt lange) Regelsätze zu verarbeiten. Dies würde sich jedoch negativ auf die Gesamtlatenz des Systems und somit auf die QoE für den Internet-Nutzer auswirken.

C.5 Realisierung des SecAN-Web-Filters

C.5.1 Funktionalität des HTTP-Parsers im Web-Filterkern

Bei HTTP handelt es sich um ein sehr flexibles Protokoll (vgl. Abschnitt 2.3.1), sodass der HTTP-Parser relativ viele Bedingungen prüfen muss und der Hardware-Aufwand somit steigt. Zunächst wird in den Zusatzinformationen zum Frame überprüft, ob es sich um eine HTTP-GET-Anfrage handelt und ab welcher Position im Frame die Anfrage erscheint. Bei einer HTTP-GET-Anfrage, muss die darin enthaltene Domain gefunden, extrahiert und ge„hash“t werden. Bereits das Auffinden stellt eine Herausforderung für die Hardware dar, wird jedoch durch die vorgeschalteten *PCE* erleichtert. Abhängig

davon, ob es sich um eine relative oder absolute Anfrage handelt, befindet sich die Domain in der *Startzeile* oder in einem *Host Header*, dessen Position erst bestimmt werden muss (vgl. Abschnitt 2.3.1.3).

Das Parsen der Domain setzt ab der TCP-Payload-Start-Markierung, die aus den Zusatzinformationen bekannt ist, ein. Ab dieser Position sucht der HTTP-Parser die Zeichenkette „GET“ gefolgt von einem oder mehreren Leerzeichen. Anschließend ist entweder ein „http://“ (absolute Anfrageform) oder ein „/“ (relative Anfrageform) (vgl. Abschnitt 2.3.1.3) vorhanden. In beiden Fällen können wieder Leerzeichen folgen. Wenn keine der beiden möglichen Anfrageformen erkannt wird, handelt es sich um eine ungültige HTTP-GET-Anfrage. Solche Frames werden ohne weitere Untersuchung weitergeleitet. Bei einer gültigen HTTP-GET-Anfrage ist es die Aufgabe des HTTP-Parsers, die beschriebene Zeichenfolge zu erkennen und zu überspringen.

Bei der absoluten Anfrageform folgt meist der „www.“-String. Wenn vorhanden, wird dieser ebenfalls ignoriert. Anschließend befindet sich der HTTP-Parser direkt vor einer Domain, wie sie im „Hash-Lookup-Speicher“ hinterlegt ist.

Wurde hingegen die relative Anfrageform erkannt, muss der Parser den *Host Header* mit der gesuchten Domain finden. Er ist durch die Zeichenkette „\r\nHost:“ gekennzeichnet und kann an beliebiger Stelle vor dem HTTP-Body vorhanden sein (vgl. Abschnitt 2.3.1). Auch auf den „\r\nHOST:“-String können ein oder mehrere Leerzeichen sowie ein „www.“-String folgen, die wiederum übersprungen werden müssen, bevor der HTTP-Parser sich gleichermaßen vor der zu untersuchenden Domain befindet. In beiden beschriebenen Fällen beginnt der HTTP-Parser nun, die Domain zu erfassen und zwischenzuspeichern.

C.5.1.1 Normalisierung der Domain

Da innerhalb einer Domain sowohl die Groß- als auch die Kleinschreibung gestattet sind, entsteht das sogenannte *NoCase*-Problem. Es beschreibt die Datenmengen, die sich aus der Vielfalt von Groß- und Kleinschreibung ergeben und dementsprechend gespeichert werden müssen. Ist im Datenstrom Wort „ab“ enthalten, dann kann es in den 4 Kombinationen „ab“, „Ab“, „aB“ und „AB“ erscheinen. Bezogen auf das Hashen im HTTP-Parser entstehen dann 4 unterschiedliche Hash-Werte, die alle im „Hash-Lookup-Speicher“ hinterlegt werden müssten.

Da bis zu 255 Byte lange Domains möglich sind, ist dieses Verfahren unpraktikabel.

Um die verfügbaren Hardware-Ressourcen optimal zu nutzen, werden die Buchstaben aller Domains *Case-insensitve* behandelt. Das bedeutet, dass alle auftretenden Buchstaben in Kleinbuchstaben umgewandelt werden. Alle weiteren erlaubten Zeichen bleiben bestehen. Vor der Konfiguration des Web-Filtersystems wurden die Domains für den „Hash-Lookup-Speicher“ nach dem gleichen Verfahren behandelt.

Die Hardware-seitige Umcodierung der Buchstaben kann problemlos erfolgen, denn die mathematische Differenz zwischen einem Klein- und dem dazugehörigen Großbuchstaben entspricht in ASCII-Kodierung dem dezimalen Wert 32 - einer Zweierpotenz - wodurch nur ein Bit modifiziert werden muss. Wie Abbildung C.10 an drei Beispielen verdeutlicht, wird die Wandlung durch das Fixieren des 6. Bits aus Sicht des Least Significant Bit (LSB) erreicht.

Buchstabe	ASCII-Code dezimal	ASCII-Code binär
g	103	0110 0111
G	71	0100 0111
e	101	0110 0101
E	69	0100 0101
t	116	0111 0100
T	84	0101 0100
		MSB LSB

Abbildung C.10: Case-insensitive Umcodierung von Buchstaben

Hashen der Domain: Um ein geeignetes Hash-Verfahren zu bestimmen, wurden von VeriSign[1] - einem der weltgrößten Registrare für Web-Domains - dessen gesamte Verwaltungsliste angefordert. VeriSign verwaltet ca. die Hälfte aller weltweit registrierten Domains. Dazu zählen .com, .net, .tv, .name, .cc sowie .jobs. Eine erste Untersuchung zielte auf die Längenverteilung der Domains innerhalb der *.com und der *.net Zone-Files ab. Abbildung C.11 stellt die Ergebnisse dieser Untersuchung dar.

Untersucht wurden 100.821.035 Domains - die gesamten .com und .net Domains von VeriSign. Dabei wiesen die kürzesten Domains eine Länge von 5 Byte auf. Wie sich zeigte, wurde die obere Grenze von 255 Byte deutlich unterschritten, denn die längsten Domains besaßen eine Länge von 67 Byte. Aufgrund der Untersuchung konnte nachgewiesen werden, dass im Mittel die Domains eine Länge von $\approx 17,24$ Byte aufweisen.

[1] http://www.verisigninc.com

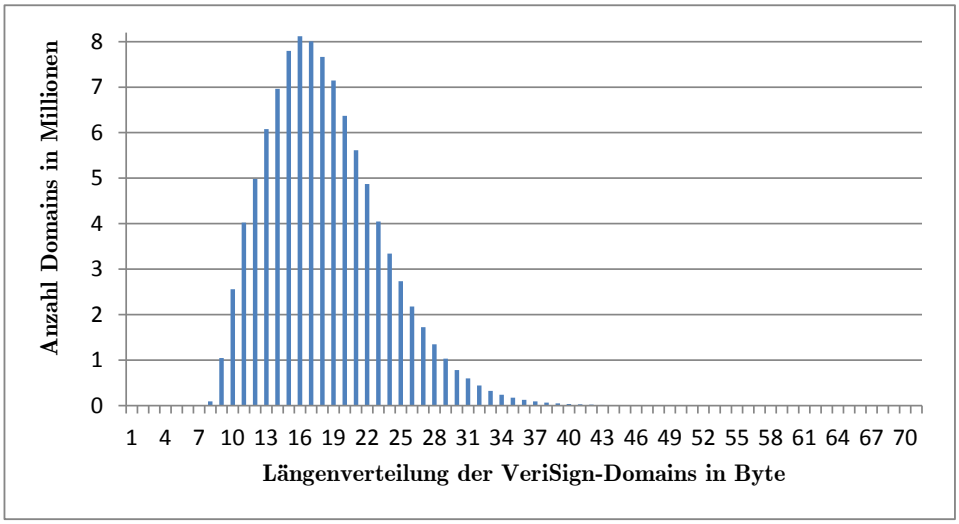


Abbildung C.11: Längenverteilung der VeriSign .com und .net Domains

Somit sind grundsätzlich alle Verfahren geeignet, deren Hash-Ergebnis eine Länge von 18 Byte unterschreitet.

In einer zweiten Untersuchung wurden verschiedene Hash-Verfahren auf 23 Millionen VeriSign-Domains angewandt. Dabei wurden nicht nur wie bei der ersten Untersuchung Zählerwerte bestimmt, sondern die Hash-Ergebnisse direkt miteinander verglichen.

Die H_3-Hash-Funktionen (vgl. Abschnitt 4.3.2.3) wurden für die Hash-Wert-Berechnung des Hash-Blacklist-Speichers nicht weiter verfolgt. Gegen H_3-hashing spricht in diesem Fall das Verhältnis aus einer relativ kleinen Filterliste (4.096 Einträge) und dem hohen Hardware-seitigen Ressourcenaufwand für die Implementierung zum gleichwertigen CRC-Verfahren. Aus diesen Gründen wurden die Untersuchungen mit der XOR-Methode und verschiedenen CRC-Verfahren durchgeführt (vgl. Abschnitt 4.3.2.3).

Hash-Lookup mittels XOR-Methode: Die XOR-Hash-Funktion hat die Laufzeit eines XOR-Gatters und ist somit sehr schnell. Der Ressourcenverbrauch beträgt ein XOR-Gatter je Hash-Wert-Bit. In Tabelle C.2 sind die Ergebnisse für drei verschiedene Hash-Wert-Längen aufgeführt.

Wie zu erwarten war, verbessert sich mit zunehmender Hash-Länge das Kollisionsverhalten. Bereits bei einer Länge von 64 Bit liegt dies unter 0,2 ‰. Für den Anwendungsfall bedeutet dies, dass erst bei ca. 10.000 Speichereinträgen mit einer Kollision gerechnet werden muss.

Hash-Lookup mittels CRC-Methoden: Für das CRC-Verfahren wurden zwei Gene-

Hash-Wert-Länge [Bit]	Anzahl Kollisionen	Kollisionswahrscheinlichkeit
32	198792	$\approx 0,864\ \%$
64	4528	$\approx 0,0197\ \%$
128	35	$\approx 1,5 \cdot 10^{-4}\ \%$

Tabelle C.2: Ergebnisse der XOR-Hash-Funktionen für 23 Mio. Domains

Hash-Wert-Länge [Bit]	Anzahl Kollisionen	Kollisionswahrscheinlichkeit
32	61485	$\approx 0,267\ \%$
64	159	$\approx 6,9 \cdot 10^{-4}\ \%$

Tabelle C.3: Ergebnisse der CRC-Hash-Funktionen für 23 Mio. Domains

ratorpolynome unterschiedlicher Länge verwendet. Dabei wurde für das CRC32-Hashing das Polynom aus Abschnitt 4.3.2.3 (nach [DIX82]) und für das CRC64-Hashing das Polynom $x^{64}+x^4+x^3+x+1$ (nach ISO 3309) verwendet. Für die Berechnung von CRC64 werden 224-XOR-Gatter sowie ein 64-Bit-Register benötigt. Die Berechnungszeit beträgt drei XOR-Gatter-Laufzeiten unter der Voraussetzung einer baumartigen Struktur. Tabelle C.3 zeigt die Ergebnisse für beide CRC-Verfahren auf.

Bei den eingangs erwähnten 23 Millionen Domains traten beim CRC64-Hashing lediglich 159 Kollisionen auf. In jedem Kollisionsfall existierten lediglich zwei Ergebnisse mit demselben Hash-Wert. Somit ist das CRC64-Verfahren selbst für sehr umfangreiche Filterlisten bestens geeignet.

Allgemein kann festgestellt werden, dass CRC-Verfahren bei gleichen Eingangsvektoren deutlich bessere Ergebnisse produzieren als XOR-Verfahren. Obwohl der Ressourcenverbrauch bei der XOR-Methode etwas günstiger ausfällt, ist ihr Kollisionsverhalten bei gleicher Hash-Länge (z. B. 64 Bit) deutlich schlechter. CRC64 bietet von den untersuchten Verfahren die beste Kombination aus Kollisionswahrscheinlichkeit, Berechnungszeit und Ressourcenverbrauch und wird daher als Verfahren für den „Hash-Lookup-Speicher“ ausgewählt.

C.5.2 Funktionalität in der Hash-Lookup-Phase

Um die Suchzeit im Hash-Lookup-Speicher weitgehend unabhängig von der Größe der Filterliste zu gestalten, wurde dieser als „Heap“ ausgelegt. Ein „Heap“ ist eine baumartige Struktur, bei der alle Schlüssel des einen Unterbaumes (z. B. linker Unterbaum) kleiner der Wurzel und alle Schlüssel des anderen Unterbaumes größer der Wurzel sind. Ferner besitzt jeder Knoten maximal zwei nachfolgende Knoten. Somit unterliegt er einer logarithmischen Suchkomplexität $O(log_2(n))$.

Durch die Vorverarbeitung der Domains mittels CRC64 entstehen Hash-Werte mit jeweils 8 Byte Länge. Es wird somit ein Speicher der Größe: $8\ Byte \cdot 4096\ Domains = 32768\ Byte = 32\ KByte$ benötigt. Der FPGA der Zielplattform - der FX70T - besitzt 148 BRAM-Zellen mit je 36 Kbit[2]. Somit werden für den Heap ≈ 8 BRAM-Zellen benötigt. Folglich lässt sich der „Hash-Lookup-Speicher“ Ressourcen-schonend aus BRAM-Zellen generieren.

Im Anwendungsszenario besitzt der „Heap“ den Adressraum 0 bis 4095. Die Suche beginnt in der Mitte des „Heap“. An jeder Adresse ist ein Hash-Wert hinterlegt, der mit dem Hash-Wert der aktuellen Domain verglichen wird. Ist der Wert des gegenwärtigen Schlüssels kleiner, wird nach links traversiert, anderenfalls nach rechts. Dabei wird jeweils an die Mitte der jeweiligen Subliste gesprungen. Die Suche kann auf verschiedene Weise beendet werden:

1. Der Heap wurde durchlaufen. Es existiert jedoch kein passender Schlüssel.
2. Der Hash-Wert der aktuellen Domain stimmt mit dem Hash-Wert der gegenwärtigen Adresse überein.

Während im 1. Fall der Datenframe bedenkenlos weitergeleitet werden kann, muss im 2. Fall eine Verifikation im „Domain-Blacklist-Speicher“ stattfinden. Dafür werden die Domain des Ethernet-Frames, deren Länge in Byte sowie die Speicheradresse des Matches im „Heap“ übergeben.

C.5.3 Funktionalität der Domain-Lookup-Phase

Die Suche im „Domain-Blacklist-Speicher“ erfolgt nur dann, wenn ein Match im „Heap“ des „Hash-Lookup-Speichers“ auftrat. Zu diesem Zweck müssen alle Domains der Filterliste in einem geeigneten Speicher hinterlegt werden.

[2] http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf

C.5.3.1 Wahl eines geeigneten Speichers

Wie in Abschnitt 4.1 beschrieben, stehen auf dem ML507 Entwicklungs-Board ein 1 MB großer ZBT-SRAM sowie ein 512 MB großer DDR2-Speicher zur Verfügung. Ferner wurde im Abschnitt 4.6.3.2 definiert, dass die Domain-Blacklist bis zu 4096 Einträge enthalten soll. Ausgehend von der maximalen Länge einer Domain (255 Byte) muss der Speicher eine Mindestgröße von 1 MB aufweisen (vgl. Formel C.1). Prinzipiell sind beide verfügbaren Speicher somit geeignet.

$$MEM_{Domain-Blacklist} \;=\; 255\ Byte\ \cdot 4096\ Domains\ \approx 0,996\ MB \tag{C.1}$$

Jedoch können, bedingt durch die Datenbusbreite beim ZBT-SRAM, lediglich 32 Bit Daten je Systemtakt transferiert werden. Beim Lesen müsste der Filterkern im *Worst Case* somit 64 Takte warten, um eine gesamte Domain zu lesen und zu vergleichen. Im Falle einer Kollision würde anschließend eine Kollisionsauflösung erfolgen. Da der Speicher keine weiteren Zusatzinformationen aufnehmen kann, muss diese linear erfolgen, wodurch im *Worst Case* eine Verzögerung von 262144 Takten entstehen kann (vgl. Formel C.2). Bei einer Datenverarbeitungsbreite im Web-Filtersystem von 32 Bit entspricht dies einer Verzögerung von ca. 692 maximal langen Ethernet-Frames.

$$Lesezeit_{SRAM} \;=\; 64\ Takte\ \cdot 4096\ Domains\ =\ 262144\ Takte \tag{C.2}$$

Aufgrund der möglichen Verzögerung erweist sich der ZBT-SRAM somit als ungeeignet, sodass auf den deutlich größeren DDR2-Speicher zurückgegriffen wird. Dieser wird zukünftig auch als „Verifikationsspeicher“ bezeichnet und ist aufgrund seiner Größe zukunftssicher, da er problemlos die mehrfache Menge der angestrebten Speichereinträge aufnehmen kann. Zudem transferiert er mit jedem Systemtakt 128 Bit Daten. Jedoch ist die einmalige Latenz, bis der Speicher Daten ausliefert, sehr hoch. Diese wird, wie bereits in Abschnitt C.3.2.1 vorgestellt, durch den modifizierten DDR2-Controller deutlich reduziert.

C.5.3.2 Suche im DDR2-Speicher

Für die Suche im Verifikationsspeicher wurden am Ende der „Hash-Lookup-Phase“ folgende Werte an die „Domain-Lookup-Phase“ übergeben:

- Domainname
- Länge der Domain
- Speicheradresse des Hash-Wertes (hash_address)

Eine gültige Speicheradresse im DDR2-Speicher wird mit Hilfe der Formel C.3 ermittelt.

$$Adresse_{\mathrm{DDR2}} = hash\_address \cdot Bucketgröße \; + \; OffsetAdresse \tag{C.3}$$

Der DDR2-Speicher ist ab der *OffsetAdresse* für den Web-Filter in Buckets aufgeteilt (vgl. Abbildung C.12). Dabei stellt die *OffsetAdresse* die Startadresse für das Web-Filter-Modul im DDR2-Speicher und die *Bucket-Größe* die Größe eines Speichereintrages im DDR2-Speicher dar. Weiterhin ist jedes Bucket ein Speichereintrag für genau eine Domain und besitzt den Aufbau wie im rechten Teil von Abbildung C.12 dargestellt. Neben der Länge der gespeicherten Domain (*Domain Length*) existiert eine Kollisionsanzeige (*COL*-Bit). Das *COL*-Bit zeigt an, ob für den zuvor ermittelten Hash-Wert mehrere Einträge im Verifikationsspeicher vorhanden sind. Sollte dies der Fall sein, befindet sich die nächste Domain im Bucket an der Adresse *Next DDR2 Address*. Letztendlich folgt nach 10 reservierten Byte die Referenzdomain.

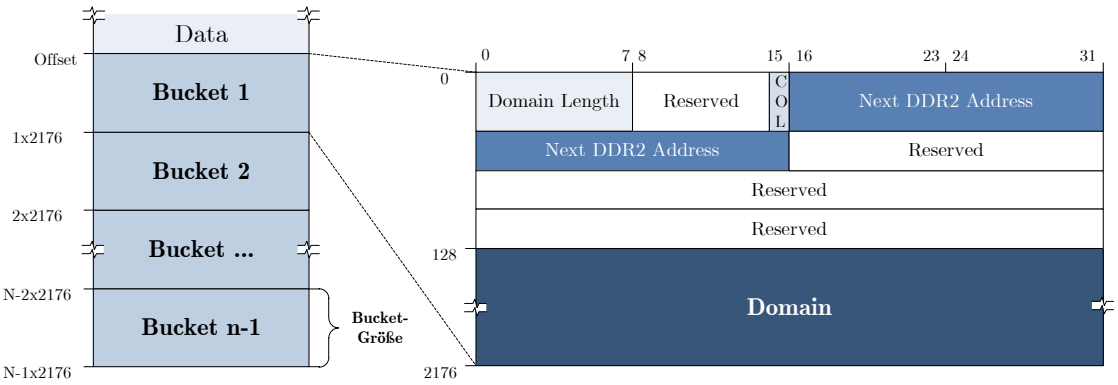


Abbildung C.12: Aufbau eines Buckets im DDR2-Speicher für das Web-Filter-Modul

Aufgrund der reservierten Bereiche sind die Buckets Hardware-Ressourcen-schonend implementiert und ermöglichen gleichzeitig ein optimal schnelles Auslesen der relevanten Informationen. Die im Bucket enthaltenen Zusatzinformationen werden innerhalb eines Systemtaktes ausgelesen. Ein Vergleich der Längenangaben der gespeicherten Domain und der Domain im Datenstrom ermöglicht eine Indikation bezüglich eines Treffers.

Aktion	Beschreibung
ALERT	Erzeugt einen Alarm und protokolliert das Paket
LOG	Protokolliert das Paket
PASS	System ignoriert das Paket
ACTIVATE	Erzeugt einen Alarm und leitet das Paket zu einer weiteren SNORT-Regel.
DYNAMIC	Versetzt das System in einen IDLE-Modus, bis es durch eine andere Regel wieder aktiviert wird. Im Anschluss wird das Paket protokolliert.
DROP	Das Paket wird blockiert und protokolliert.
REJECT	Das Paket wird blockiert und protokolliert. Im Anschluss wird ein Reset-Paket versandt (z. B. TCP Reset oder bei UDP-Paketen ein ICMP Port nicht erreichbar).
SDROP	Das Paket wird blockiert, aber nicht protokolliert.

Tabelle C.4: Regelaktionen in SNORT [RGI11]

Stimmen die Werte überein, wird anschließend die Domain zum Vergleich ausgelesen. Bei einem Mismatch der Längenangaben wird das *COL*-Bit ausgewertet. Ist es nicht gesetzt, endet die Suche - anderenfalls wird die Suche an der DDR2-Adresse (*Next DDR2 Address*) fortgesetzt. Schließlich endet die Suche, wenn das *COL*-Bit anzeigt, dass in keinem weiteren Bucket gesucht werden muss bzw. ein Match mit der Domain im Bucket stattgefunden hat. Ein Match führt in jedem Fall zum Verwurf aller empfangenen Eingangsdaten.

C.6 Weiterführende Details zum ID-System

C.6.1 Aufbau und Funktionsweise von SNORT

In der vom Autor verwendeten SNORT-Version 2.9.0 können vom „Logging/Alerting Subsystem“ die in Tabelle C.4 dargestellten acht unterschiedlichen Aktionen erfolgen [RGI11]:

C.6.1.1 SNORT-Regeln

Obwohl die Syntax von SNORT-Regeln recht einfach ist, bietet sie dennoch ausreichend Flexibilität, um eine Vielzahl von Angriffen, verdächtigen Paketinhalten oder Anomalien zu beschreiben. Grundsätzlich lassen sich alle SNORT-Regeln vereinfacht wie folgt darstellen:

```
Aktion Protokoll IP1 Port1 Richtung IP2 Port2 Regel
```

Eine sehr einfache SNORT-Regel lautet beispielsweise wie folgt:

```
log tcp 132.169.1.15 any → 141.13.1.77 22
```

Die Regel protokolliert jedes TCP-Datenpaket von der IP-Adresse 132.169.1.15 (Port beliebig = „any“) zum Host-Rechner 141.13.1.77 auf Port 22. Zusätzlich können jeder SNORT-Regel Optionen mitgegeben werden, die sich in runden Klammern hinter dem Basisaufbau der gezeigten Regel befinden. Optionen können Protokollnachrichten, Signaturen, Content-Modifier, Flags und PCRE enthalten. Diese können auf unterschiedliche Arten miteinander kombiniert werden, sodass beliebig komplexe SNORT-Regeln entstehen.

```
alert tcp any any → 141.13.1.77 80 (msg: "PHF Probe")
                    content: "/cgi-bin/phf")
```

Das aufgeführte Beispiel löst einen Alarm aus, wenn ein beliebiges TCP-Paket (IP-Adresse = „any“ und Port = „any“) zum Host mit der IP-Adresse 141.13.1.77 auf Port 80 geschickt wird und in dessen Nutzdaten der `/cgi-bin/phf` erscheint. Die protokollierte Nachricht enthält dabei als Identifikator die Angabe `PHF Probe`.

Weiterführende Informationen zu SNORT sind dem SNORT-Benutzerhandbuch [RGI11] sowie in [Sou11, Spe07] zu entnehmen. Eine Auswahl von ID/IP-Systemen, die neben SNORT existieren, ist in Tabelle C.5 dargestellt.

C.6.2 Realisierung des IDS-Filterkerns

In diesem Abschnitt wird auf die Implementierung der Module *Sliding-Window-Cluster*, *Meta-Kompressor*, *Kompressionsnetzwerk* und *Simple-Match-Analyzer* des IDS-Filterkerns eingegangen.

IDS/IPS	frei/kommerziell	Typ	Verweis
Bro IDS	frei	NIDS	www.bro-ids.org
CheckPoint IPS-1	kommerziell	Hybrid	www.checkpoint.com/products/ips-1
Cisco NetRanger	kommerziell	NIDS	www.cicso.com
ISS RealSecure	kommerziell	NIDS	www.iss.net
LIDS	frei	HIDS	www.lids.org
McAfee Entercept	kommerziell	HIDS	www.mcafee.com
Netscreen	kommerziell	NIDS	www.juniper.net
Prelude	frei	Hybrid	www.prelude-technologies.com
Samhain	frei	HIDS	www.la-samhna.de/samhain
SNORT	frei/kommerziell	NIDS	www.snort.org
Suricata	frei	NIDS	www.openinfosecfoundation.org
Symantec NetProwler	kommerziell	NIDS	www.symantec.com
Tripwire	frei/kommerziell	HIDS	www.tripwire.com

Tabelle C.5: Ausgewählte Intrusion Detection- und Intrusion Prevention-Systeme

C.6.2.1 Sliding-Window-Cluster

Die Sliding-Window-Hardware besitzt mehrere Aufgaben (Abbildung C.13):

- Normalisierung des eingehenden Datenstromes
- Bereitstellung des Byte-Musters zur Signaturanalyse
- Kompensation von Verzögerungen durch interne Prozesse (*Kompressionsnetzwerk* und *Simple-Match-Analyzer*)

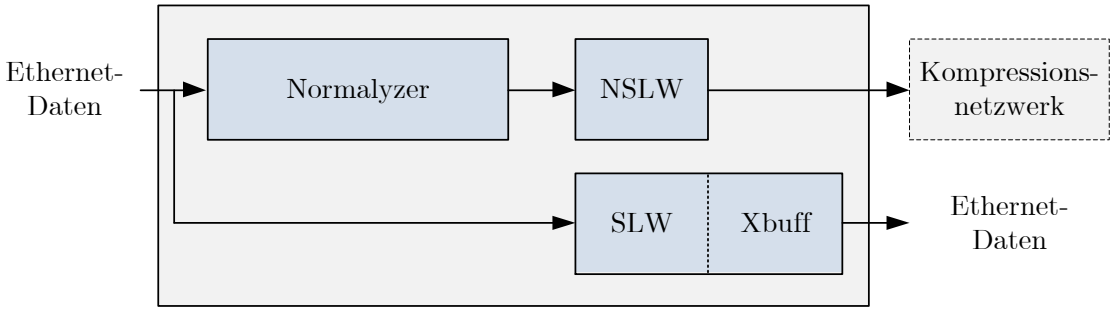


Abbildung C.13: Blockschaltbild des Sliding-Window-Clusters

Wie bereits in Abschnitt C.5.1 erläutert, ist eine Normalisierung der Eingangsdaten notwendig, um das *NoCase*-Problem zu lösen. Der Normalyzer übernimmt genau diese Aufgaben und überführt alle Buchstaben in Kleinbuchstaben. Anschließend werden die normalisierten Eingangsdaten dem Normalized Sliding Window (NSLW) übergeben. Dieses leitet mit jedem Takt 30 Byte an das Kompressionsnetzwerk. Da der Vorgang der Normalisierung irreversibel ist, der originale Datenstrom jedoch erhalten bleiben soll, werden die Eingangsdaten parallel an das Sliding Window (SLW) übergeben. Das SLW nimmt die originalen Eingangsdaten auf und speichert sie zwischen. Die für das Kompressionsnetzwerk benötigte Verzögerung wird durch den *Xbuff* realisiert. Dieser besteht aus einer Reihe von Registern, in die nacheinander die Daten übergeben werden (Pipelining). So kann durch die *Xbuff*-Erweiterung des SLWs sichergestellt werden, dass die Ergebnisse des Bloom-Filter-Clusters synchron mit den Eingangsdaten den *Simple-Match-Analyzer* erreichen.

C.6.2.2 Meta-Kompressor

Die H3-Hash-Werte für die Meta-Daten (Protokoll und Portnummern) des aktuellen Frames erzeugt der *Meta-Kompressor*. Dessen Flow ist in Abbildung C.14 dargestellt.

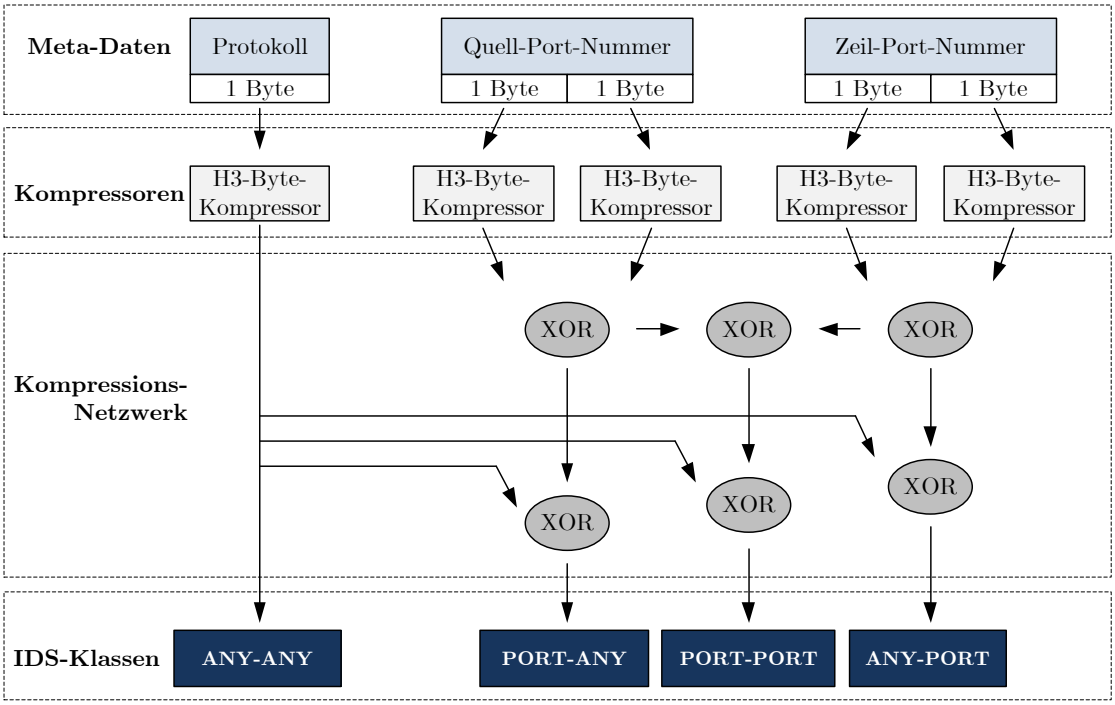


Abbildung C.14: H3-Kompression der IDS-Meta-Daten

Die Meta-Daten werden Byte-weise zerlegt und je einem H3-Byte-Kompressor zugeführt (vgl. Abbildung C.15). Ein H3-Byte-Kompressor führt eine logische UND-Verknüpfung jedes Bit der Meta-Daten mit der Binär-Darstellung einer Zufallszahl durch. Anschließend werden die Teilergebnisse über eine Baumstruktur XOR-verknüpft und ergeben einen H3-Hash-Wert. Der Aufbau eines H3-Byte-Kompressors ist in Abbildung C.15 dargestellt. Die H3-Hash-Werte werden, wie in Abbildung C.14 gezeigt, erneut XOR verknüpft und ergeben so die H3-Indizes der vier IDS-Klassen.

Da aufgrund der systeminternen Struktur ein Verlust der IFG möglich ist, können sich zwei verschiedene Frames mit unterschiedlichen Meta-Daten im Filterkern befinden. Um Überschneidungen und somit fehlerhafte Ergebnisse zu vermeiden, werden Meta-Daten für jedes Byte erzeugt und dem Metadata Sliding Window (MSLW) übergeben. Dieses

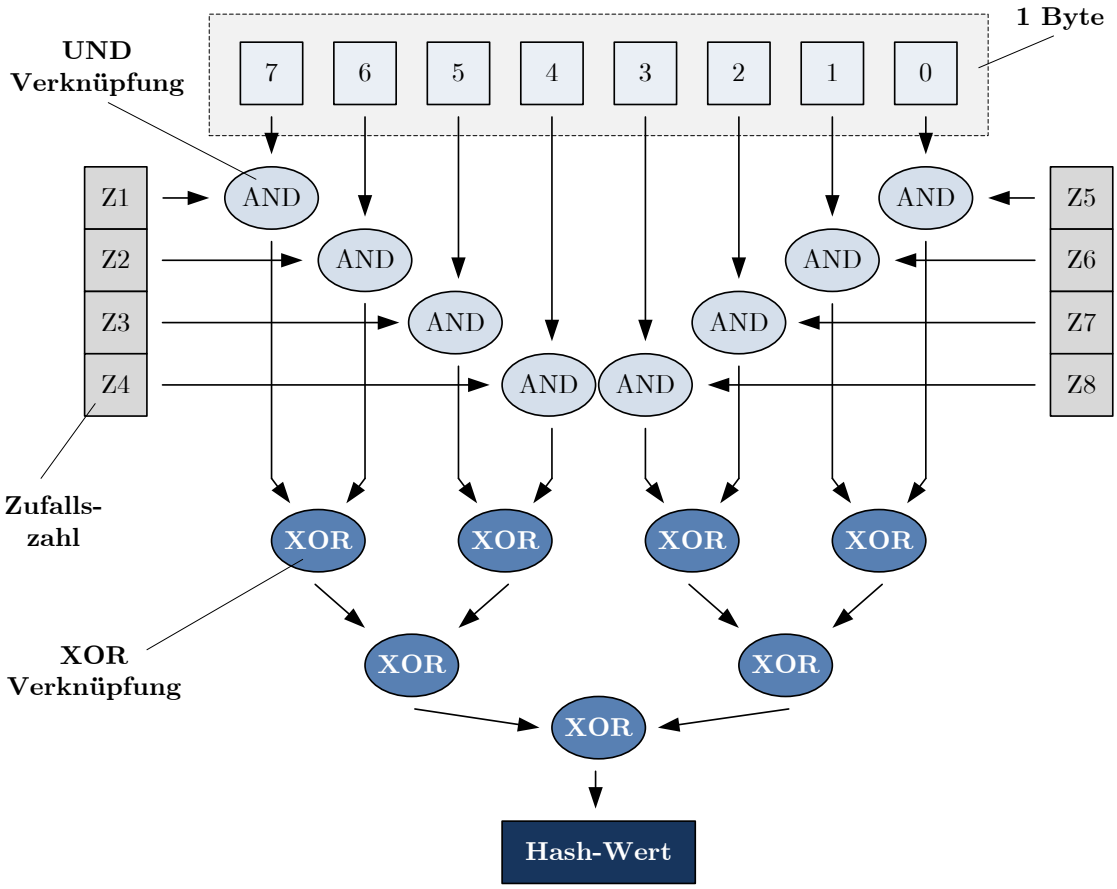


Abbildung C.15: Aufbau eines H3-Byte-Kompressors

ebenfalls 30-stufige Sliding Window befindet sich am Ausgang des *Meta-Kompressors*. Schließlich werden die Index-Werte des MSLW parallel mit den normalisierten Eingangsdaten des NSLW dem Kompressionsnetzwerk zugeführt.

C.6.2.3 Kompressionsnetzwerk

Das Kompressionsnetzwerk ist das Hashing-Schema zur iterativen H3-Kompression für Multi-Byte-Pattern. Als Basiskomponenten werden - genau wie im Meta-Kompressor - Byte-Kompressoren eingesetzt. Deren Aufbau wurde bereits in Abbildung C.15 gezeigt.

Beispielhaft soll die Bildungsvorschrift für zwei Bloom-Filter-Indizes an den Mustern $M1$ (1 Byte) und $M2$ (2 Byte) hergeleitet werden. Das Muster $M2$ enthält das Muster $M1$ als Präfix. Als zweites Byte ist das Muster $M2'$ im Muster $M2$ enthalten ($M2 = M1|M2'$). Die Kompression der einzelnen Byte erfolgt über Byte-Kompressoren (vgl. Abbildung C.15). Durch die Assoziativität von XOR lässt sich folgender iterativer Zusammenhang für beliebige H3-Hash-Funktionen darstellen:

$h3(M1) = H1$

$h3(M2) = h3(M1|M2') = h3(M1)xorh3(M2') = H1xorh3(M2')$

Für beliebige n Byte lange Muster gilt deshalb der Zusammenhang:

$$h3(M_n) = h3(M1)xorh3(M2)...xorh3(M_{n-1})xorh3(M_n) \tag{C.4}$$

Da jedes Byte im Sliding-Window einen eigenen Byte-Kompressor besitzt, müssen 30 disjunkte Teilmengen echter Zufallszahlen existieren, um 30 unabhängige und echt zufällig, gleichverteilte Indizes zu erzeugen. Das Kompressionsnetzwerk wurde nach der Vorschrift aus Formel C.4 aufgebaut und ist in Abbildung C.16 dargestellt. Es besitzt eine Tiefe von $\lceil log2(n) \rceil$, wobei n der maximalen Signaturlänge in Byte entspricht. Jeder Index steht repräsentativ für eine diskrete Fensterbreite und somit für eine Gruppe von IDS-Regeln gleicher Signaturlänge.

Um auf jeden BF mit vier Hash-Funktionen zugreifen zu können, muss der beschriebene Hardware-Aufwand vervierfacht werden. Infolgedessen werden vier unabhängige Matrizen mit echten Zufallszahlen benötigt, sodass schließlich 120 H3-Hash-Funktionen parallel existieren. Ferner müssen im Rahmen des erweiterten Bloom-Filter-Ansatzes die 120 H3-Hash-Ergebnisse der normalisierten Eingangsdaten mit den Index-Werten der

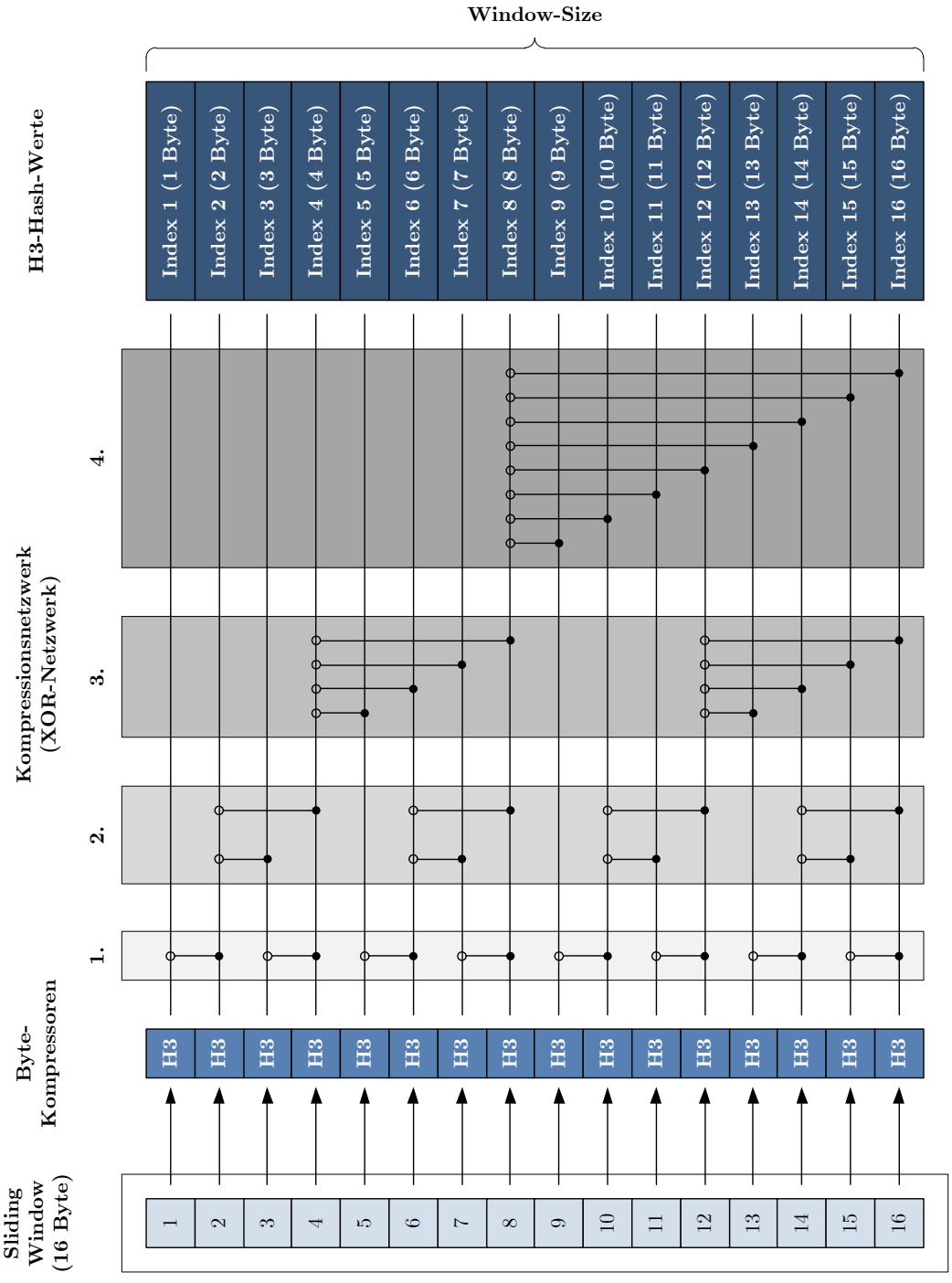


Abbildung C.16: H3-Kompressionsnetzwerk der Tiefe $log_2(n)$

IDS-Klassen (AA, AP, PA, PP) kombiniert werden. Das Zusammenführen erfolgt ebenfalls per XOR-Verknüpfung, sodass final 480 unterschiedliche Index-Werte entstehen. Jeweils vier Index-Werte werden demselben Bloom-Filter zugeordnet.

Somit werden je Systemtakt 480 Index-Werte erzeugt und mit den Einträgen in den 120 Bloom-Filtern verglichen. Nur wenn alle vier Indizes eines Bloom-Filters auf eine 1 verweisen, deutet dies auf ein Match hin und wird mit einer 1 im Match-Vektor markiert. Der Match-Vektor ist somit ein 120 Bit breiter Vektor, der in vier gleich große Sektionen aufgeteilt ist (4x30 Bit). Jede Sektion entspricht den Ergebnissen einer IDS-Klasse. Ferner zeigt die Bit-Position innerhalb einer Sektion die Länge der als Match erkannten Signatur an. Mit Hilfe dieser Klassifizierung des Match-Vektors wird der Match-Analyzer in die Lage versetzt, gezielte Vergleiche des Eingangsdatenstromes verknüpft mit Meta-Daten und den Regeln des Regelwerkes durchzuführen. Schematisch ist der Match-Vektor in Abbildung C.17 dargestellt. Mit jedem Systemtakt wird ein Match-Vektor an den

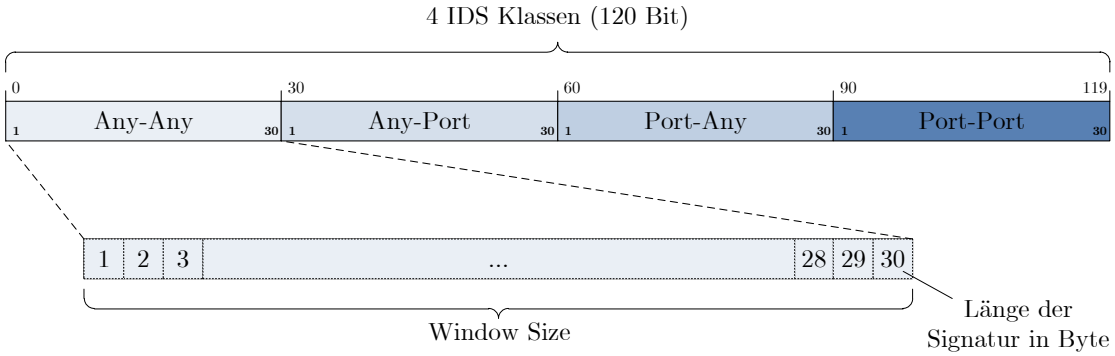


Abbildung C.17: Aufbau und Kodierung des Bloom-Filter Match-Vektors

Match-Analyzer übertragen. Dieser wertet den Vektor aus und entscheidet, wie mit dem gegenwärtigen Ethernet-Frame verfahren werden soll.

C.6.2.4 Simple-Match-Analyzer

Aufgabe des Match-Analyzers ist die Verifikation der als suspekt eingestuften Signaturen gegenüber dem Regelwerk. Durch die Verifikation kann eine False Positive Rate von 0 % erreicht werden. In der implementierten Version handelt es sich um einen *Simple-Match-Analyzer*. Er verfügt über die folgenden Informationen:

- Der 120 Bit breite Match-Vektor enthält indirekt alle relevanten Informationen der

suspekten Signaturen (IDS-Klasse, Länge der Signatur, Multiple-Matches, Präfix-Matches).

- Der Inhalt des Sliding-Window mit den unmodifizierten Daten, welcher über den *Xbuff* synchron mit dem Match-Vektor ausgeliefert wird.

Der *Simple-Match-Analyzer* reduziert durch eine logische OR-Verknüpfung den Inhalt des Match-Verktors auf ein Bit. Das Ergebnis der Kompression zeigt, dass eine suspekte Signatur im aktuellen Sliding-Window gefunden wurde. Ferner wird die Ausgabe auf dem LCD-Display der Zielplattform aktualisiert. Am Systemausgang wird überprüft, ob die Markierung für den gegenwärtigen Frame vorhanden ist. Wenn dies der Fall ist, wird der Frame verworfen, anderenfalls wird er zum vorgesehenen Ziel weitergeleitet. In einer optimierten Version des Match-Analyzers können zusätzlich die zum Frame gehörigen Meta-Daten zur Analyse herangezogen werden, wodurch der Verifikationsprozess beschleunigt würde.

C.7 Optimierungsmöglichkeiten des vorgestellten Intrusion Detection Systems

Die einfachste Lösung, das entwickelte ID-System zu optimieren, liegt in der Verwendung eines größeren FPGAs mit mehr freien Ressourcen wie bspw. dem FX100T. Dadurch können entweder eine größere Anzahl an Signaturen und Metadaten verarbeitet oder die FPR des Systems weiter gesenkt werden.

Natürlich ist es auch möglich, das gesamte System zu vervielfältigen und parallel zu betreiben. Dabei besteht ein linearer Zusammenhang zwischen der Erhöhung der Hardware-Ressourcen und den zu verarbeitenden Datenraten. Die Anzahl speicherbarer Signaturen bleibt dabei konstant, wie Abbildung 4.22 nachweist.

Neben der Verwendung eines größeren FPGAs und dem parallelen Einsatz der Hardware ist eine Faltung von Signaturen möglich. Dabei werden Signaturen bspw. in 30 Byte große Segmente zerlegt und in demselben Bloom-Filter hinterlegt. Der Rest der Signatur (<30 Byte) wird im dementsprechenden Bloom-Filter programmiert. Durch diesen Ansatz können Signaturen von Viren, Trojanern und anderer Malware in das System integriert werden, wodurch es zu einem Anti-Malware System erweitert würde. Unter gleichen Bedingungen verschlechtert dieses Verfahren jedoch die FPR.

Da die Verteilung der Signaturen der ausgewählten SNORT-Datenbank sehr unregelmäßig verläuft, ist ein Ansatz mit dynamisch allokierten Bloom-Filter-Slices möglich. Kirsch und Mitzenmacher [KM05] beweisen, dass lediglich zwei unabhängige Hash-Funktionen notwendig sind, wenn sie nach dem folgenden Schema hergeleitet werden:

$g_i(x) = h_1(x) + i \cdot h_2(x) mod p \forall i \in I = 0, 1, ..., k-1$

Dabei sind h_1 und h_2 die unabhängigen Hash-Funktionen, x das zu komprimierende Datum, i eine natürliche Zahl zwischen 0 und $k-1$ und p die Breite des Hash-Wertes. Nach diesem Prinzip würden für jede Signaturlänge neben einer statischen BRAM-Zelle eine oder mehrere BRAM-Zellen dynamisch hinzufügt werden. Jede dynamische BRAM-Zelle hätte zwei extra Digital Signalprocessor (DSP)-Blöcke, um mit den beiden zusätzlichen Parametern i und j beliebige Hash-Funktionen zu generieren. Auf diese Weise können der Signaturverteilung folgend angepasste Bloom-Filter erzeugt werden. Das Prinzip ist in Abbildung C.18 dargestellt.

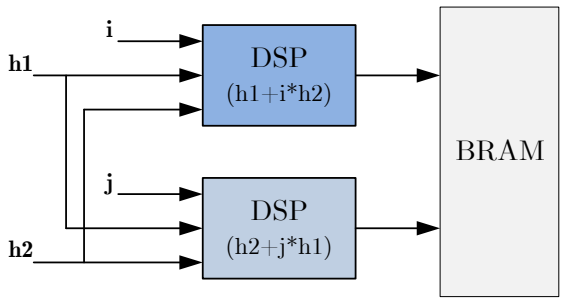


Abbildung C.18: Konzeptionelle Darstellung von dynamischen Bloom-Filter-Slices

Die Allokierung dynamischer Bloom-Filter-Slices erfolgt durch das Weiterleiten der Hash-Funktionen vom statischen Bloom-Filter-Slice. Aufgrund des Präprozesses, in dem die Signaturverteilung ermittelt wird, ist bereits im Vorfeld bekannt, wie viele dynamische Slices benötigt werden (I und J). Diese Werte lassen sich in Software-programmierbaren Registern in der Hardware hinterlegen.

Obwohl die beiden neuen Hash-Funktionen nicht mehr unabhängig von den statischen Hash-Funktionen sind, wird die Qualität der Filter nicht negative beeinflusst, da durch die virtuelle Adressraumaufteilung Inter-Hash-Kollisionen vermieden werden.

Sicherlich benötigt jeder dynamische Bloom-Filter-Slice mehr Ressourcen als der statische Ansatz, jedoch können durch dieses Verfahren auch ganze Bloom-Filter-Slices eingespart werden. Unter Annahme einer FPR von $\frac{0{,}001}{120}$ ‰ je Bloom-Filter können in

jedem Bloom-Filter-Slice 23 Signaturen programmiert werden. Wird weiter die SNORT Datenbank DB 100530 zugrunde gelegt, benötigen 90 der 120 Bloom-Filter keinen dynamischen Bloom-Filter-Slice.

Anhang D

Weiterführende Informationen zur Zielplattform und zu Entwicklungstools

D.1 Virtex-5-FPGA (XC5VFX70T)

Die grundlegenden Logikstrukturen der Virtex-5-Familie sind Flipflops (FFs), Lookup-Tables (LUTs) und BRAM-Module mit integrierter Error Correction Code (ECC)- und FIFO-Logik. Ihre Kombination ergibt konfigurierbare Logikblöcke. Ein Logikblock eines Virtex-5 besteht aus zwei Slices. Jedes Slice besitzt vier 6-Eingangs-LUTs, vier FFs, einen Wide-Function-Multiplexer und dedizierte Logikfunktionen. Im Virtex-5 werden diese Basis-Slices „SLICEL“ genannt. Einige Slices verfügen zusätzlich über integrierte RAM und 32-Bit-Kippstufen. Sie werden „SLICEM“ genannt [Xil09].

Neben den Basis-Slices besitzen die Virtex-5 FPGAs DSP-Slices. Sie werden für komplexe mathematische Berechnungen verwendet. Jedes DSP-Slice besitzt eine 25 x 18 Bit Zweierkomplement-Multiplikationslogik sowie komplexe Additionsschaltungen [Xil09]. Tabelle D.1 bietet einen Überblick über die verfügbaren Ressourcen des FX70T und vergleicht mit FPGAs der gleichen Familie.

D.2 Ethernet-MACs

Die Ethernet-MACs des FX70T wurden nach dem IEEE 802.3-2002-Standard entworfen. Sie bieten bspw. VLAN-Unterstützung an. Um die Funktionalität eines TZN zu ermöglichen, wurde das ML507 um einen zweiten PHY-Transceiver ergänzt. Dieser befindet

Feature	FX30T	FX70T	FX100T	FX130T	FX200T
Array (Row x Col)	80 x 38	160 x 38	160 x 56	200 x 56	240 x 68
Slices	5.120	11.200	16.000	20.480	30.720
Max Distributed RAM	380	820	1.240	1.580	2.280
DSP-Slices	64	128	256	320	384
Block-RAM (18k)	136	296	456	596	912
PowerPC-Blocks	1	1	2	2	2
Ethernet MACs	4	4	4	6	8
Max User-I/O	360	640	680	840	960

Tabelle D.1: Übersicht über die Mitglieder der Virtex-5-FXT-Familie nach Xilinx [Xil09]

sich auf einer PHY-Daughter-Card (HW-AFX-BERG-EPHY[1]). Die Daughter-Card ist über den Standard-BERG-Header-Adapter mit dem ML507 verbunden. Beide PHY-Transceiver arbeiten mit Übertragungsraten von 10 Mbit/s, 100 Mbit/s und 1 Gbit/s und entsprechen somit den Anforderungen aus Abschnitt 3.1.1.

D.3 Onboard-Speicher (DDR2-SDRAM/SRAM)

Das ML507-Entwicklungsboard besitzt zwei unterschiedliche Speicher - einen DDR2- und einen SRAM-Speicher. Beim DDR2-Speicherriegel (MT4HTF3264HY-53ED3) handelt es sich um einen Speicher der Firma Micron [Tec13]. Der Speicher kann mit bis zu 400 MHz getaktet werden [Xil11]. Mit Hilfe des MIG v3.3 lässt sich ein kompatibler DDR2-Controller erstellen, der über ein benutzerdefiniertes Interface verfügt und somit durch individuell entwickelte Hardware-Komponenten angesprochen werden kann. Darüber hinaus können die individuellen Hardware-Komponenten des MIG-Speicher-Controllers das FPGA-interne Bussystem umgehen, wodurch sich die Zugriffszeiten auf den Speicher deutlich reduzieren lassen.

Neben dem externen DDR2-Speicher besitzt das Entwicklungsboard einen 1 MByte großen ZBT synchronen SRAM-Speicher mit einem 32-Bit-Datenbus (vgl. Abbildung 4.2). Der SRAM eignet sich besonders für Systeme mit hohen Anforderungen an nied-

[1] Xilinx PHY-Daughter-Card - http://www.xilinx.com/products/devkits/HW-AFX-BERG-EPHY.htm

rige Speicherlatenz und hohe Zugriffsgeschwindigkeiten. Beide Speicher können für die Entwicklung des SecAN-Gesamtsystems eingeplant werden. Aufgrund der niedrigeren Speicherlatenz ist der SRAM dem DDR2-Speicher vorzuziehen, soweit dies möglich ist.

D.4 Entwicklungstools und Testumgebungen der SecAN-Hardware

Alle Hardware-Komponenten des SecAN-Gesamtsystems werden mit den Entwicklungstools der *Xilinx ISE Design Suite* entwickelt. Als zentrale Komponenten werden der *Project Navigator*, der *CORE Generator* und das Synthese-Tool *XST*[2] in der Version 12.3 verwendet.

Nach Fertigstellung der Komponenten werden diese einem funktionalen Test unterzogen. Neben den Single-Modul-Tests werden erfolgreich getestete Komponenten zu einem Teilsystem vereint und erneut getestet. Anschließend erfolgen Tests mit dem Gesamtsystem.

Zu diesem Zweck wird die Simulationsumgebung *ModelSim* von Mentor Graphics[3] in der Version SE 6.6 verwendet. Zum Testen werden spezielle Testbenches entwickelt. Sie ermöglichen den Aufbau spezifischer Datenmuster. Abhängig vom verwendeten Testszenario können von den Testbenches auch mit dem Netzwerkanalyseprogramm Wireshark[4] aufgezeichnete Verkehrsmuster eingelesen und abgespielt werden.

Das grundsätzliche Vorgehen ist in Abbildung D.1 dargestellt. In der *Testbench* werden die Simulationsmuster durch den *Stimulus-Prozess* aus einer Konfiguration bzw. aus aufgezeichneten Datenströmen erzeugt. Bevor sie an die *Testharness* übergeben werden, werden sie in einem *Puffer FIFO* gespeichert. Die Testharness vollzieht eine Schnittstellenumsetzung von der Testbench zum *Device Under Test*. Durch sie ist es möglich, dieselben Eingangsmuster für Komponenten, Teilsysteme und das Gesamtsystem zu verwenden. Der *Monitor-Prozess* lädt die zuvor gespeicherten Simulationsmuster und vergleicht sie mit den empfangenen Daten aus dem *Device Under Test*.

Finale Tests des Gesamtsystems werden in der Entwicklungsumgebung durchgeführt (vgl. Abschnitt 4.1). Sie überprüfen das Einhalten der Mindestanforderungen in Bezug auf Geschwindigkeit sowie die Stabilität des Systems. Dabei wird der Höchstdurchsatz

[2] Xilinx ISE Design Suite: http://www.xilinx.com/university/index.htm
[3] ModelSim: http://www.mentor.com/products/fv/modelsim/
[4] http://www.wireshark.org/

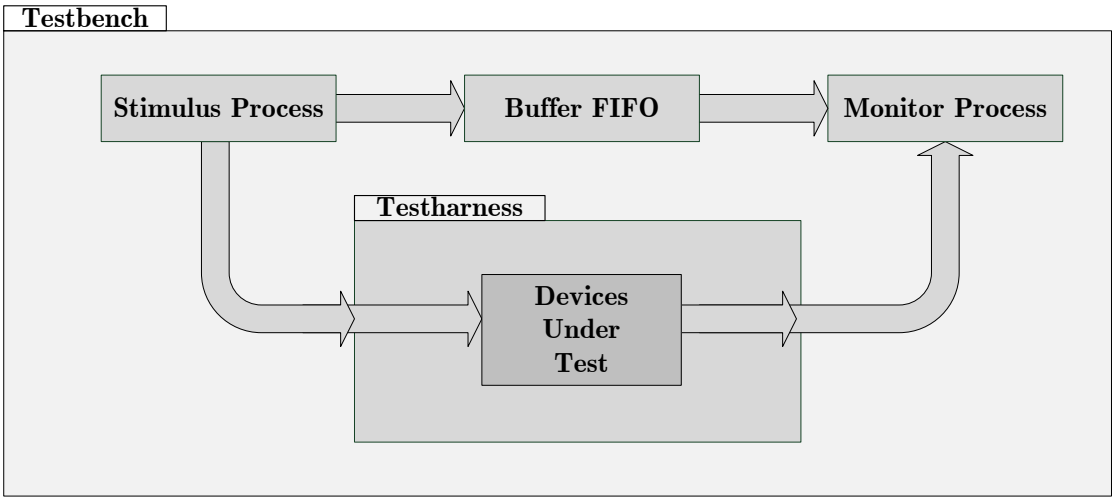


Abbildung D.1: Testaufbau für simulative Tests

mit einem FPGA-basierten Traffic-Generator [TDT11] erzeugt. Der Traffic-Generator kann so konfiguriert werden, dass er dieselben Daten erzeugt, wie sie für die Simulation mittels ModelSim verwendet wurden. Bei häufig wechselnden Verkehrsmustern kann der Traffic-Generator so konfiguriert werden, dass er ein bereits erzeugtes Muster x-mal wiederholt, bevor ein neues Verkehrsmuster erzeugt sein muss. Läuft der Traffic-Generator leer, wird dies über eine dauerhaft leuchtende LED angezeigt. So wird sichergestellt, dass der Traffic-Generator stets maximal viele Datenmuster erzeugt und versendet. Darüber hinaus werden die Werte für erzeugte Daten-Frames und erfolgreich versandte Daten-Frames über ein Display angegeben. Die Systemstabilitäten für die SecAN-Systeme Firewall, Web-Filter und IDS wurden durch mehrmonatige Langzeittests an einem Arbeitsplatz-Personal Computer (PC) nachgewiesen, bevor sie einer Prüfung nach industriellen Standards unterzogen wurden.

Literaturverzeichnis

[ABP07] Paulo Sérgio Almeida, Carlos Baquero, and Nuno Preguica. Scalable bloom filters. Information Processing Letters, Volume 101, Issue 6, 2007.

[AC75] Alfred. V. Aho and Margaret J. Corasick. Efficient string matching: An aid to bibliographic search. Murray Hill, N.J., USA, Association for Computing Machinery, Inc., 1975.

[AD11] R. Ajami and Anh Dinh. Design a hardware network firewall on fpga. In *Electrical and Computer Engineering (CCECE), 2011 24th Canadian Conference on*, pages 000674–000678, May 2011.

[AG12] Barracuda Networks AG. Barracuda web filter schützt internetbenutzer vor schädlichem und unpassendem inhalt `http://www.barracudanetworks.com/ns/downloads/Datasheets/Barracuda_Web_Filter_DS_DE.pdf` (12.1.2012), January 2012.

[AGGR09] E. Azimi, M.B. Ghaznavi-Ghoushchi, and A.M. Rahmani. Implementation of simple snort processor for efficient intrusion detection systems. In *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*, volume 3, pages 533 –537, nov. 2009.

[AMS11] AMSIX. AMS-IX - Amsterdam Internet Exchange, `http://www.ams-ix.net/`, 2011.

[Ass02] IEEE Standards Association. 802.3 ieee standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements part 3: Carrier sense multiple access with collision detection (csma/cd) access method and physical layer specifications `http://standards.ieee.org/getieee802/download/802.3-2002.pdf`. Technical report, IEEE Computer Society, March 2002.

[Axe03] Jan Axelson. *Embedded Ethernet and Internet Complete: Designing and Programming Small Devices for Networking.* Lakeview Research LLC, 2003.

[Bed09] Mark Bedner. Rechtmäßigkeit der "Deep Packet Inspection", Projektgruppe verfassungsverträgliche Technikgestaltung (provet), Universität Kassel, 2009.

[Ber03] A. Costello (UC Berkeley). Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA) `http://tools.ietf.org/html/rfc3492`, March 2003.

[BGJ06] J. Bruck, Jie Gao, and Anxiao Jiang. Weighted bloom filter. International Symposium on Information Theory, IEEE, 2006.

[Blo70] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. Communication of the ACM, Vol. 13, No. 7„ 1970.

[Blu04] Norbert Blum. *Algorithmen und Datenstrukturen: eine anwendungsorientierte Einführung.* Oldenbourg Wissenschaftsverlag, 2004.

[Blu11] Blue Coat. Blue coat webfilter `http://www.bluecoat.de/products/webfilter/index.php` (17.01.2012), 2011.

[BM95] Scott Bradner and Allison Mankin. The Recommendation for the IP Next Generation Protocol, `http://tools.ietf.org/html/rfc1752`, Januar 1995.

[BM01] Rebecca Bace and Peter Mell. NIST Special Publication on Intrusion Detection Systems, `http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA393326`, 2001.

[BM04] A. Border and M. Mitzenmacher. Network applications of bloom filters: A survey. In *Internet Mathematics*, volume 1, No. 4, 2004.

[Bre00] Jean-Louis Brelet. Using block ram for high performance read/write cams `http://www.xilinx.com/support/documentation/application_notes/xapp204.pdf` (02.05.2000), May 2000.

[BSI15] BSI. Leitfaden informationssicherheit it-grundschutz kompakt `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/Leitfaden/GS-Leitfaden_pdf.pdf?__blob=publicationFile` (29.03.2015), March 2015.

[Bun97] Bundesamt für Sicherheit in der Informationstechnik (BSI). *Informationen zu Computer-Viren, Schriftenreihe zur IT-Sicherheit*. Bundesanzeiger-Verlag, 1997.

[Bun09] Bundesamt für Sicherheit in der Informationstechnik (BSI). Trojanische pferde , `https://www.bsi.bund.de/ContentBSI/grundschutz/kataloge/g/g05/g05021.html(16.03.2011)`, 2009.

[Bun11] Bundeskriminalamt. BKA: Internet-Kriminalität nimmt weiter zu, `http://www.presseportal.de/polizeipresse/pm/7/2071082`, 2011.

[BW02] Christoph Busch and Stephen D. Wolthusen. *Netzwerksicherheit*. Spektrum Akademischer Verlag, 2002.

[Car96] B. Carpenter. Architectural Principles of the Internet, `http://tools.ietf.org/html/rfc1958`, Juni 1996.

[Cla05] Richard Clayton. *Anonymity and Traceability in Cyberspace*. PhD thesis, University of Cambridge, Computer Laboratory, Darwin College, August 2005.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. McGraw-Hill Higher Education 3rd edition, 2009.

[CM03] Saar Cohen and Yossi Matias. Spectral bloom filters. SIGMOD 2003, San Diego, CA, USA, ACM, 2003.

[CM05] Martin Casado and Nick McKeown. The virtual network system. St. Louis, Missouri, USA, SIGCSE05, 2005.

[CMW06] R. Clayton, S. J. Murdoch, and R. N. M. Watson. Ignoring the great firewall of china. *6th Workshop on Privacy Enhancing Technologies*, page 16, June 2006.

[CO00] K.G. Coffman and Andrew Odlyzko. Internet Growth: Is There a "Moore's Law"for Data Traffic?, `http://www.ssrn.com/abstract=236108`, 2000.

[CW10] T.M. Chen and V. Wang. Web filtering and censoring. *Computer*, 43(3):94–97, March 2010.

[Dan06] Peter Danielis. Implementation of an algorithm for realtime pattern recognition within an ethernet data stream. Master's thesis, Universität Rostock, 2006.

[DBB09] Mrudul Dixit, B. V. Barbadekar, and Ashwinee B. Barbadekar. Packet classification algorithms. *IEEE International Symposium on Industrial Electronics*, (ISIE):1407–1412, July 2009.

[Dec13] Decix. Traffic Statistics, `http://www.de-cix.net/content/network/statistics.html(29.05.2013)`, 2013.

[DIX82] Digital, Intel, and Xerox. The ethernet - a local area network data link layer and physikal layer, version 2.0. 1982.

[EB11] Jan Engelhardt and Nicolas Bouliane. Writing Netfilter modules, rev. February 07, 2011, `http://jengelh.medozas.de/documents/Netfilter_Modules.pdf`, February 2011.

[FCAB00] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder. Summary cache: A scalable wide-area web cache sharing protocol. IEEE/ACM Transactions on Networking, 2000.

[FIS08] Mario FISCHER. *Website Boosting 2.0*. 2008.

[FK95] Fritz E. Froehlich and Allen Kent. *The Froehlich/Kent encyclopedia of telecommunications Volume 9*. New York, Basel, Hong Kong, 1995.

[FLE82] JOHN G. FLETCHER. An arithmetic checksum for serial transmissions. IEEE Transactions on Communications, vol. 30, 1982.

[FMC11] Nicolas Falliere, Liam O Murchu, and Eric Chien. Symantec Security Response W32.Stuxnet Dossier Version 1.4, `http://www.ams-ix.net/`. 2011.

[For05] Internet Engineering Task Force. Uniform resource identifier (uri): Generic syntax, Januar 2005.

[Fuh00] Kai Fuhrberg. *Internet-Sicherheit*. Hanser Verlag, 2000.

[G D13] G DATA. G data securitylabs malware report halbjahresbericht juli - dezember 2013, `https://public.gdatasoftware.com/Presse/Publikationen/Malware_Reports/GData_PCMWR_H2_2013_DE.pdf` online 27.09.2014, 2013.

[Gib07] Steve Gibbard. Geographic implications of dns infrastructure distribution. The Internet Protocol Journal - Volume 10, No. 1, March 2007.

[GLM98] P. Gupta, S. Lin, and N. McKeown. Routing lookups in hardware at memory access speeds. In *INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1240–1247 vol.3, Mar 1998.

[GM00] Pankaj Gupta and Nick McKeown. Dynamic algorithms with worst-case performance for packet classification. In Guy Pujolle, Harry G. Perros, Serge Fdida, Ulf Körner, and Ioannis Stavrakakis, editors, *NETWORKING*, volume 1815 of *Lecture Notes in Computer Science*, pages 528–539. Springer, 2000.

[GM01] P. Gupta and N. McKeown. Algorithms for packet classification. *Network, IEEE*, 15(2):24–32, Mar 2001.

[GPP03] A. Guruprasad, P. Pandey, and B. Prashant. Security features in ethernet switches for access networks, TENCON 2003, Conference on Convergent Technologies for Asia-Pacific Region . pages 1211–1214, 2003.

[Gua] The Guardian. Edward snowden, nsa files source: 'if they want to get you, in time they will' `http://www.theguardian.com/world/2013/jun/09/nsa-whistleblower-edward-snowden-why` online 10.06.2013.

[GWCL06] Deke Guo, Jie Wu, Honghui Chen, and Xueshan Luo. Theory and network applications of dynamic bloom filters. Proceedings of the 25th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM, 2006.

[Hau07] Gardar Hauksson. Dynamic bloom filters: Analysis and usability. SC700A2: Randomized Network Algorithms, May 2007.

[HEIa] HEISE. Ipv4-adressen: Abschiedsgrüße, mahnungen und pappschilder.

[HEIb] HEISE. Stuxnet-wurm kann industrieanlagen steuern, `http://www.heise.de/security/meldung/Stuxnet-Wurm-kann-Industrieanlagen-steuern-1080584.html(23.02.2011)`.

[HEI06] HEISE. Sony-chef entschuldigt sich für kopierschutz per rootkit, `http://www.heise.de/newsticker/meldung/Sony-Chef-entschuldigt-sich-fuer-Kopierschutz-per-Rootkit-163399.html(17.03.2011)`, 2006.

[Hel03] Gilbert Held. *Ethernet Networks: Design, Implementation, Operation, Management*. John Wiley & Sons, 2003.

[HP03] Jörg Holzmann and Jürgen Plate. *Linux-Server für Intranet und Internet Den Server einrichten und administrieren 3. Auflage*. Hanser Verlag, 2003.

[hS11] heise Security. BKA und Bitkom kooperieren stärker bei Bekämpfung der Internetkriminalität, `http://www.heise.de/security/meldung/BKA-und-Bitkom-kooperieren-staerker-bei-Bekaempfung-der-Internetkrimina` `html`, 2011.

[Hua] Huawei. Greennet solution brochure `http://www.huawei.com/products/datacomm/catalog.do?id=3596` online 17.01.2012.

[IEE01] IEEE COMPUTER SOCIETY. EEE Standard for Local and Metropolitan Area Networks: Overview and Architecture. IEEE Std 802. New York, New York, USA : Institute of Electrical and Electronics Engineers, `http://standards.ieee.org/about/get/802/802.html`, 2001.

[IEE06a] IEEE COMPUTER SOCIETY. IEEE Standards for Local and metropolitan area networks-Virtual Bridged Local Area Networks-Amendment 4: Prvider Bridges. IEEE Std 802.1ad. New York, New York, USA : Institute of Electrical and Electronics Engineers, `http://standards.ieee.org/about/get/802/802.1.html`, 2006.

[IEE06b] IEEE COMPUTER SOCIETY. IEEE Standards for Local and metropolitan area networks-Virtual Bridged Local Area Networks. IEEE Std 802.1Q. New York, New York, USA : Institute of Electrical and Electronics Engineers, `http://standards.ieee.org/about/get/802/802.1.html`, 2006.

[IEE08] IEEE COMPUTER SOCIETY. IEEE Standard for Information Technology-Telecommunications and Informations Exchange between Systems-Local and Metropolitan Area Networks-Specific Requirements-Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications. IEEE Std 802.3. New York, New York, USA : Institute of Electrical and Electronics Engineers, `http://standards.ieee.org/getieee802/802.3.html`, 2008.

[INF81] INFORMATION SCIENCES INSTITUTE, UNIVERSITY OF SOUTHERN CALIFORNIA. Internet Protocol - DARPA In-

ternet Program Protocol Specification. RFC 791 (Standard), http://tools.ietf.org/html/rfc791, September 1981.

[INF95] INFORMATION SCIENCES INSTITUTE, UNIVERSITY OF SOUTHERN CALIFORNIA. Internet Protocol - DARPA Internet Program Protocol Specification. RFC 1819 (Standard), http://tools.ietf.org/html/rfc1819, September 1995.

[Inf09] Information & Communication Technology Authority. Pulblic Consultation on A Policy for Deep Packet Inspection and Similar Technologies, http://www.icta.ky/docs/DPI/CD%202009-4%20Deep%20Packet%20Inspection.pdf, 2009.

[Ini11] OpenNet Initiative. Global internet filtering map http://map.opennet.net/filtering-consec.html, December 2011.

[Inta] Internet Assigned Numbers Authority. Number Resources, http://www.iana.org/numbers/.

[Intb] Internet World Stats. INTERNET USAGE STATISTICS - The Internet Big Picture - World Internet Users and Population Stats, url = http://internetworldstats.com/stats.htm.

[Int84] Internationl Organization for Standardization and International Electrotechnical Committee. *Information Processing Systems - Open Systems Interconnection - Basic Reference Model International Standard 7498-1.* ISO, 1984.

[ISO15] ISO/IEC. Iso/iec 27001:2013 information technology – security techniques – information security management systems – requirements http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=54534 (19.08.2014), March 2015.

[Jai92] Raj Jain. A comparison of hashing schemes for address lookup in computer networks. IEEE Transactions on Communications, vol. 40, no. 10, 1992.

[JP09a] Weirong Jiang and Viktor K. Prasanna. A FPGA-based Parallel Architecture for Scalable High-Speed Packet Classification. *20th IEEE International Conference on Application-specific Systems, Architectures and Processors*, pages 24–31, July 2009.

[JP09b] Weirong Jiang and Viktor K. Prasanna. Field-split parallel architecture for high performance multi-match packet classification using fpgas. In *Proceedings of the Twenty-first Annual Symposium on Parallelism in Algorithms and Architectures*, SPAA '09, pages 188–196, New York, NY, USA, 2009. ACM.

[JRV08] Gajanan S. Jedhe, Arun Ramamoorthy, and Kuruvilla Varghese. A Scalable High Throughput Firewall in FPGA. *16th International Symposium on Field-Programmable Custom Computing Machines*, pages 43–52, April 2008.

[Kap13] Martin Kappes. Netzwerk- und datensicherheit: Eine praktische einführung, 2. auflage, 2013.

[Kas09] Kaspersky. Kaspersky Internet Security 2010, Der Heuristische Analyser in den Produkten von Kaspersky Lab der Version 2010, `http://support.kaspersky.com/de/kis2010/tech?qid=207619912`, 2009.

[Kas10] Kaspersky. Kaspersky Internet Security 2010, Verwendung der heuristischen Analyse bei der Untersuchung mit Datei-Anti-Virus in Kaspersky Internet Security 2010, `http://support.kaspersky.com/de/kis2010/fileav?qid=207620008`, 2010.

[KC04] P. Koopman and T. Chakravarty. Cyclic redundancy code (crc) polynomial selection for embedded networks. pages 145–154. 2004 International Conference on Dependable Systems and Networks (DSN'04), Florence, Italy, 2004.

[Kir13] Keith Kirkpatrick. Software-defined networking. *Commun. ACM*, 56(9):16–19, September 2013.

[KL12] AR. Khakpour and AX. Liu. First step toward cloud-based firewalling. In *Reliable Distributed Systems (SRDS), 2012 IEEE 31st Symposium on*, pages 41–50, Oct 2012.

[KM05] Adam Kirsch and Michael Mitzenmacher. Biuilding a better bloom filter. Technical Report TR-02-05. Computer Science Group, Havard University, Cambridge, 2005.

[KM06] Adam Kirsch and Michael Mitzenmacher. Distance-sensitive bloom-filter. Proceedings of the Eighth Workshop on Algorithm Engineering and Experiments, SIAM, 2006.

[Lea02] Delmar Learning. *Networks Fundamental: The Open System Interconnect Osi Reference Model: Open System Interconnect (OSI) Reference Model*. Delmar Cengage Learning, 2002.

[Lei99] Dr. Peter Leibner. *TCP/IP-Netze Grundlagen, Anwendungen, Sicherheit; Reihe Lehrstuhl Informatik, Band 3*. Krehl Verlag, 1999.

[LXLS09] Yizhen Liu, Daxiong Xu, Dong Liu, and Lingge Sun. A fast and configurable pattern matching hardware architecture for intrusion detection. In *Knowledge Discovery and Data Mining, 2009. WKDD 2009. Second International Workshop on*, pages 614 –618, jan. 2009.

[McH10] Patrick McHardy. nftables status update, `http://nfws.edenwall.com/en/?p=249`, 2010.

[MHK07] Yoshihide Matsumoto, Hiroaki Hazeyama, and Youki Kadobayashi. Adaptive bloom filter: A space-efficient counting algorithm for unpredictable network traffic. IEICE Transactions on Information and Systems, Volume E91.D, Issue 5, 2007.

[MS09] Klaus Mochalski and Hendrik Schulze. Deep Packet Inspection, Technology, Applications & Net Neutrality, `http://www.ipoque.com/resources/white-papers`. page 12, 2009.

[Net] Barracuda Networks. Barracuda web filter, `http://www.barracudanetworks.com/ns/downloads/Datasheets/Barracuda_Web_Filter_DS_DE.pdf` (17.01.2012).

[Net87a] Network Working Group. Domain names - concepts and facilities, November 1987.

[Net87b] Network Working Group. Domain names - implementation and specification, November 1987.

[Net95] Network Working Group. Security Considerations for IP Fragment Filtering, `http://www.faqs.org/rfcs/rfc1858.html`, October 1995.

[Net10a] Netfilter. Netfilter firewalling, NAT and packet mangling for linux, The netfilter.org iptables project, `http://www.iptables.org/projects/iptables/index.html`, 2010.

[Net10b] Nokia Siemens Networks. Surpass hix 5622/25/30/35 r3.7m, system description, ip-dslam, issue 3, 2010.

[Net11] Netfilter. Netfilter firewalling, NAT and packet mangling for linux, The netfilter.org project, `http://netfilter.org/`, March 2011.

[Net12] Barracuda Networks. Purchase online `https://www.barracudanetworks.com/ns/purchase/purchase.php` (12.1.2012), january 2012.

[Nin06] Matthias Ninnemann. Analyse und implementierung von fpga basierten, kooperierenden funktionseinheiten innerhalb einer kommunikationstechnischen anwendung, masterarbeit. Master's thesis, 2006.

[P. 03] P. Faltstrom (Cisco), P. Hoffman (IMC & VPNC), A. Costello (UC Berkeley). Internationalizing Domain Names in Applications (IDNA) `http://tools.ietf.org/html/rfc3490`, March 2003.

[PAN09] PANDA. 52% aller viren existieren nur 24 stunden `http://pandanews.de/?s=virensignaturen`, 2009.

[PC12] Karthi Palanisamy and Rich Chiu. High-Performance DDR2 SDRAM Interface in Virtex-5 Devices, rev. 2.2, January 2012.

[Pei10] Ali Peiravi. Application of string matching in internet security and reliability. Ferdowsi University of Mashhad, RAN, Journal of American Science 2010, 2010.

[PKAC08] R. Proudfoot, K. Kent, E. Aubanel, and Nan Chen. Flexible software-hardware network intrusion detection system. In *Rapid System Prototyping, 2008. RSP '08. The 19th IEEE/IFIP International Symposium on*, pages 182 –188, june 2008.

[PLW+03] D. Pao, C. Liu, A. Wu, L. Yeung, and K.S. Chan. Efficient hardware architecture for fast ip address lookup. *Computers and Digital Techniques, IEE Proceedings* -, 150(1):43–52, Jan 2003.

[PN98] Thomas H. Ptacek and Timothy M. Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. Calgary, Alberta, Canada, Secure Networks, Inc.„ 1998.

[Por05] Thomas Porter. The Perils of Deep Packet Inspection, `http://www.symantec.com/connect/de/articles/perils-deep-packet-inspection`, 2005.

[Pos80] J. Postel. User datagram protocol, August 1980.

[Pou] Kevin Poulsen. What's in the rest of the top-secret nsa powerpoint deck? `http://www.wired.com/threatlevel/2013/06/snowden-powerpoint/#slideid-57994` online 10.06.2013.

[PS06] Kostas Pagiamtzis and Ali Sheikholeslami. Content-addressable memory (CAM) circuits and architectures: A tutorial and survey. *IEEE Journal of Solid-State Circuits*, 41(3):712–727, March 2006.

[QP15] Y. Qu and V. Prasanna. High-performance and dynamically updatable packet classification engine on fpga. *Parallel and Distributed Systems, IEEE Transactions on*, PP(99):1–1, 2015.

[REA05] HEAVY READING. Beyond HSI: Reinventing the B-RAS, `http://www.d-cell.com/setyobudianto/resources/wired/beyond_high_speed_internet_bras.pdf`. White Paper, Dezember 2005.

[RFB97] M.V. Ramakrishna, E. Fu, and E. Bahcekapili. Efficient hardware hashing functions for high performance computers. *Computers, IEEE Transactions on*, 46(12):1378 –1381, dec 1997.

[RGI11] Martin Roesch, Chris Green, and Sourcefire Inc. SNORT Users Manual 2.9.0, The Snort Project, `http://www.snort.org/assets/166/snort_manual.pdf`, 2011.

[roo11] root-servers.org, July 2011.

[Ros08] K. Rosenthal. Deep Packet Inspection: 2009 Market Forecast, `https://www.dpacket.org/articles/deep-packet-inspection-2009-market-forecast`, 2008.

[Sal08] Peter H. Salus. *The ARPAnet Sourcebook: The Unpublished Foundations of the Internet (Computer Classics Revisited)*. Peer to Peer Communications, 2008.

[SBVW03] Sumeet Singh, Florin Baboescu, George Varghese, and Jia Wang. Packet classification using multidimensional cutting. In *In Proceedings of ACM SIGCOMM*, pages 213–224, 2003.

[Sch10] Uwe Schmidtke. Method for protection a network through port blocking, `http://www.freepatentsonline.com/y2010/0180341.html`, July 2010.

[Sch11] Prof. Dr. Georg Schnitger. Vorlesung internet algorithmen ss 2011,

http://www.thi.informatik.uni-frankfurt.de/Internet11/skript2011.pdf, 2011.

[Sie07] Martin Siemroth. Entwicklung einer struktur zur zustandsüberwachung verbindungsorientierter kommunikationsprotkolle, Mai 2007.

[Sie12] Richard Sietmann. ëinfältig und dumm". c't - Magazin für Computertechnik, February 2012.

[SK10] S. Surisetty and S. Kumar. Is mcafee securitycenter/firewall software providing complete security for your computer? In *Digital Society, 2010. ICDS '10. Fourth International Conference on*, pages 178–181, Feb 2010.

[SK11] S. Surisetty and S. Kumar. Mcafee securitycenter evaluation under ddos attack traffic. In *Journal of Information Security 2011*, pages 113–121, Feb 2011.

[Sou11] Sourcefire. SNORT, http://www.snort.org/, 2011.

[Spe06] Ralf Spenneberg. *Linux Firewalls mit iptables & Co, Sicherheit im Kernel 2.4 und 2.6 für Linux-Server und -Netzwerk*. ADDISON-WESLEY Verlag, 2006.

[Spe07] Ralf Spenneberg. *Intrusion Detection und Prevention mit Snort 2.x & Co: Einbrüche auf Linux-Servern erkennen und verhindern*. ADDISON-WESLEY Verlag, 2007.

[Spe11] Ralf Spenneberg. *Linux-Firewalls: Sicherheit für Linux-Server und -Netzwerke mit IPv4 und IPv6*. ADDISON-WESLEY Verlag, 2011.

[Spr00] Charles E. Sprugeon. *Ethernet: The Definitive Guide*. O'Reilly & Associates, 2000.

[STT03] E. Spitznagel, D. Taylor, and J. Turner. Packet classification using extended tcams. In *Network Protocols, 2003. Proceedings. 11th IEEE International Conference on*, pages 120–131, Nov 2003.

[Sym08] Symantec Corporation. Symantec global internet security threat report trends for july-december 07 volume xiii, published april 2008, http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_internet_security_threat_report_xiii_04-2008.en-us.pdf(18.03.2011), 2008.

[Tan03] Andres S. Tanenbaum. *Computer Networks*. New Jersey, USA, Pearson Studium, 4 edition, 2003.

[Tan09] Andres S. Tanenbaum. *Modern Operating Systems*. it informatik, 2009.

[Tay05] David E. Taylor. Survey and taxonomy of packet classification techniques. *ACM Comput. Surv.*, 37(3):238–275, September 2005.

[TDT11] Andreas Tockhorn, Peter Danielis, and Dirk Timmermann. A configurable fpga-based traffic generator for high-performance tests of packet processing systems. page 6. 6th International Conference on Internet Monitoring and Protection, ICIMP 2011, March 2011.

[Tec07a] Agilent Technologies. The journal of internet test methodologies edition 3.1 `http://www.ixiacom.com/pdfs/test_plans/agilent_journal_of_internet_test_methodologies.pdf`, May 2007.

[Tec07b] Javvin Technologies. *Network Protocols Handbook Vierte Auflage*. Javvin Technologies, 2007.

[Tec13] Inc. Micron Technology. Micron ddr2 sdram sodimm `http://download.micron.com/pdf/datasheets/modules/ddr2/HTF4C16_32_64x64H.pdf`, May 2013.

[The98] The Internet Engineering Task Force (IETF). Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Standard), `http://tools.ietf.org/html/rfc2460`, December 1998.

[The99] The Internet Engineering Task Force (IETF). Hypertext transfer protocol – http/1.1, June 1999.

[The10] The Internet Engineering Task Force (IETF). Special Use IPv4 Addresses, `http://tools.ietf.org/html/rfc5735`, Januar 2010.

[Var10] V. Varadharajan. Internet filtering issues and challenges. *Security & Privacy, Building Confidence in a Networked World*, 8:4, July-Aug. 2010.

[VV11] Balajee Vamanan and T. N. Vijaykumar. Treecam: Decoupling updates and lookups in packet classification. In *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies*, CoNEXT '11, pages 27:1–27:12, New York, NY, USA, 2011. ACM.

[Wag09] Ben Wagner. Deep Packet Inspection and Internet Censorship: International Convergence on an 'Integrated Technology of Control,

http://advocacy.globalvoicesonline.org/wp-content/uploads/2009/06/deeppacketinspectionandinternet-censorship2.pdf, 2009.

[Wat04] Paul Watson. Slipping in the window: Tcp reset attacks. *CanSecWest/core04 http://www.osvdb.org/reference/SlippingInTheWindow_v1.0.doc*, 2004.

[WC81] Mark Wegman and Larry Carter. New hash functions and their use in authentication and set equality. *Computer and Systems Sience*, pages 265–279, 1981.

[Web11] Websense. Websense; essential information protection, http://www.websense.com, http://www.e92plus.com/vendors/product-detail268898.aspx (17.01.2012), 2011.

[Wel00a] Harald Welte. The journey of a packet through the linux 2.4 network stack, Version 1.2, http://ftp.gnumonks.org/pub/doc/conntrack+nat.html, October 2000.

[Wel00b] Harald Welte. The journey of a packet through the linux 2.4 network stack, Version 1.4, http://ftp.gnumonks.org/pub/doc/packet-journey-2.4.html, October 2000.

[Wid08] Harald Widiger. *PAKETVERARBEITENDE SYSTEME - ALGORITHMEN UND ARCHITEKTUREN FÜR HOHE VERARBEITUNGSGESCHWINDIGKEITEN*. PhD thesis, Universität Rostock, 2008.

[WTDDD07] Yaron Weinsberg, Shimrit Tzur-David, Danny Dolev, and Shimrit Tzur-David Danny Dolev. One algorithm to match them all: On a generic nips pattern matching algorithm. High Performance Switching and Routing Conference, New York, USA, 2007.

[WWLZ13] Aili Wang, Chao Wang, Xi Li, and Xuehai Zhou. Soba: A services-oriented browser architecture with distributed url-filtering mechanisms for teenagers. In *Services (SERVICES), 2013 IEEE Ninth World Congress on*, pages 67–74, June 2013.

[Xil09] Inc. Xilinx. "virtex-5 family overview."technical manual, http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf, 2009.

[Xil11] Xilinx. Ml505, ml506, ml507 evaluation platform, user guide, ug3347

, `http://www.xilinx.com/support/documentation/boards_and_kits/ug347.pdf`, May 2011.

[YY10] Yang Yang and W. Yonggang. A software implementation for a hybrid firewall using linux netfilter. In *Software Engineering (WCSE), 2010 Second World Congress on*, volume 1, pages 18–21, Dec 2010.

[ZYGX10] Zhang Zhikai, Zhao Youjian, Yang Guanghui, and Zhang Xiaoping. Fast string matching with overlapped substring classifier in deep packet inspection systems. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1 –6, dec. 2010.

Liste der Veröffentlichungen und Fachvorträge auf Tagungen

[ARS+13] Altmann, Vlado ; Rohrbeck, Jens ; Skodzik jan ; Danielis, Peter ; Timmermann, Dirk ; Rönnau, Maik ; Ninnemann, Matthias: SWIFT: A Secure Web Domain Filter in Hardware. In: *The 7th International Symposium on Security and Multimodality in Pervasive Environment (SMPE-2013)* SMPE, 2013, S. 678–683

[DKW+08a] Danielis, Peter ; Kubisch, Stephan ; Widiger, Harald ; Schulz, Jens ; Duchow, Daniel ; Bahls, Thomas ; Timmermann, Dirk: A Conceptual Framework for Increasing Physical Proximity in Unstructured Peer-To-Peer Networks. In: *IEEE Sarnoff Symposium 2008*, 2008. – ISBN 978–1–4244–1843–5, S. 1–5

[DKW+08b] Danielis, Peter ; Kubisch, Stephan ; Widiger, Harald ; Schulz, Jens ; Duchow, Daniel ; Bahls, Thomas ; Timmermann, Dirk ; Lange, Christian: Trust-by-Wire in Packet-switched IP Networks: Calling Line Identification Presentation for IP (Hardware Prototype Demonstration). In: *Design, Automation and Test in Europe Conference and Exhibition (DATE 2008), University Booth* DATE, 2008, S. 375–382

[DKW+08c] Danielis, Peter ; Kubisch, Stephan ; Widiger, Harald ; Schulz, Jens ; Timmermann, Dirk ; Bahls, Thomas ; Duchow, Daniel: IPclip - An Innovative Mechanism to Reestablish Trust-by-Wire in Packet-switched IP Networks. In: *3. Essener Workshop "Neue Herausforderungen in der Netzsicherheit (Prospective Challenges in Network Security)"*, 2008, S. 1–2

[DKW+09] Danielis, Peter ; Kubisch, Stephan ; Widiger, Harald ; Rohrbeck, Jens ; Altmann, Vladyslav ; Skodzik, Jan ; Timmermann, Dirk ; Bahls, Thomas ; Duchow, Daniel: Trust-by-Wire in Packet-Switched IPv6 Networks: Tools and FPGA Prototype for the IPclip System. In: *6th IEEE Consumer*

Communications and Networking Conference, 2009. – ISBN 978–1–4244–2309–5, S. 1–2

[DSR+11] Danielis, Peter ; Skodzik, Jan ; Rohrbeck, Jens ; Altmann, Vlado ; Timmermann, Dirk ; Bahls, Thomas ; Duchow, Daniel: Using Proximity Information between BitTorrent Peers: An Extensive Study of Effects on Internet Traffic Distribution. In: *International Journal on Advances in Systems and Measurements* 4 (2011), Nr. 3&4, S. 212–221. – ISSN 1942–261x

[KWD+08a] Kubisch, Stephan ; Widiger, Harald ; Danielis, Peter ; Schulz, Jens ; Timmermann, Dirk ; Bahls, Thomas ; Duchow, Daniel: Complementing E-Mails with Distinct, Geographic Location Information in Packet-switched IP Networks. In: *MIT 2008 Spam Conference*, 2008, S. 1–25

[KWD+08b] Kubisch, Stephan ; Widiger, Harald ; Danielis, Peter ; Schulz, Jens ; Timmermann, Dirk ; Bahls, Thomas ; Duchow, Daniel: Countering Phishing Threats with Trust-by-Wire in Packet-switched IP Networks - A Conceptual Framework. In: *22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS), 4th International Workshop on Security in Systems and Networks (SSN 2008)*, 2008. – ISBN 978–1–4244–1694–3, S. 1–8

[KWD+08c] Kubisch, Stephan ; Widiger, Harald ; Danielis, Peter ; Schulz, Jens ; Timmermann, Dirk ; Bahls, Thomas ; Duchow, Daniel: Trust-by-Wire in Packet-switched IP Networks: Calling Line Identification Presentation for IP. In: *1st ITU-T Kaleidoscope Conference: Innovations in Next Generation Networks - Future Network and Services*, 2008. – ISBN 92–61–12441–0, S. 375–382

[RAP+11a] Rohrbeck, Jens ; Altmann, Vlado ; Pfeiffer, Stefan ; Danielis, Peter ; Skodzik, Jan ; Timmermann, Dirk ; Ninnemann, Matthias ; Rönnau, Maik: The Secure Access Node Project: A Hardware-Based Large-Scale Security Solution for Access Networks. In: *International Journal On Advances in Security* 4 (2011), Nr. 3&4, S. 234–244. – ISSN 1942–2636

[RAP+11b] Rohrbeck, Jens ; Altmann, Vlado ; Pfeiffer, Stefan ; Timmermann, Dirk ; Ninnemann, Matthias ; Rönnau, Maik: Secure Access Node: an FPGA-based Security Architecture for Access Networks. In: *International*

Conference on Internet Monitoring and Protection (ICIMP) ICIMP, 2011, S. 54–57

[SBS+07] Salzmann, Jakob ; Behnke, Ralf ; Schulz, Jens ; Danielis, Peter ; Lieckfeldt, Dominik ; Cornelius, Claas ; You, Jiaxi ; Timmermann, Dirk: *Mobile Generation: Unbegrenzte Möglichkeiten, unbegrenztes Risiko?* Lange Nacht der Wissenschaften, Rostock, April 2007

[SDA+11a] Skodzik, Jan ; Danielis, Peter ; Altmann, Vlado ; Rohrbeck, Jens ; Timmermann, Dirk ; Bahls, Thomas ; Duchow, Daniel: DuDE: A Distributed Computing System using a Decentralized P2P Environment. In: *36th IEEE LCN, 4th International Workshop on Architectures, Services and Applications for the Next Generation Internet*, 2011. – ISBN 978–1–61284–927–0, S. 1060–1067

[SDA+11b] Skodzik, Jan ; Danielis, Peter ; Altmann, Vlado ; Rohrbeck, Jens ; Timmermann, Dirk ; Bahls, Thomas ; Duchow, Daniel: DuDE: A Prototype for a P2P-based Distributed Computing System. In: *36th IEEE LCN, 4th International Workshop on Architectures, Services and Applications for the Next Generation Internet*, 2011

[SRBT08] Schulz, Jens ; Reichenbach, Frank ; Blumenthal, Jan ; Timmermann, Dirk: Low Cost System for Detecting Leakages along Artificial Dikes with Wireless Sensor Networks. In: *Workshop on Real-World Wireless Sensor Networks* EuroSys 2008, 2008, S. 66–71

[SSB+08a] Salzmann, Jakob ; Schulz, Jens ; Behnke, Ralf ; Danielis, Peter ; Timmermann, Dirk: *Entwicklung des Computers.* Lange Nacht der Wissenschaften, Rostock, April 2008

[SSB+08b] Salzmann, Jakob ; Schulz, Jens ; Behnke, Ralf ; Danielis, Peter ; Timmermann, Dirk: *Entwicklung des Internets.* Lange Nacht der Wissenschaften, Rostock, April 2008

[SST07] Schulz, Jens ; Salomon, Ralf ; Timmermann, Dirk: Positionsbestimmung mittels positionssensibler, zweidimensionaler Photodioden. In: *12. Symposium Maritime Elektrotechnik, Elektronik und Informationstechnik* IEF, 2007

[WKD+08] Widiger, Harald ; Kubisch, Stephan ; Danielis, Peter ; Schulz, Jens ; Timmermann, Dirk ; Bahls, Thomas ; Duchow, Daniel: IPclip: An

Architecture to restore Trust-by-Wire in Packet-switched Networks. In: *33rd Annual IEEE Conference on Local Computer Networks (LCN)*, 2008. – ISBN 978–1–4244–2413–9, S. 312–319

Betreute studentische Arbeiten

[Alt08] Altmann, Vladyslav: *Entwurf und Implementierung von Regelstufen für eine Hardwarefirewall*, Universität Rostock, Diplomarbeit, 2008

[Alt09] Altman, Vladyslav: *Erweiterung von iptables für den Einsatz in einer Hardware-Firewall auf einem Xilinx Evaluation Board*, Universität Rostock, Projektarbeit, 2009

[Alt10] Altmann, Vlado: *Konzipierung und Implementierung eines Hardware-DNS-Filters für hochbitratige Ethernet Datenströme*, Universität Rostock, Masterarbeit, 2010

[Fis11] Fischer, Hagen: *Anomalieerkennung in Zugangsnetzwerken*, Universität Rostock, Literaturarbeit, Januar 2011

[Gag09] Gag, Martin: *Implementierung und Optimierung eines Sicherheitskonzeptes zur Echtzeit-Mustererkennung in hochbitratigen Ethernet-Datenströmen mittels SystemC*, Universität Rostock, Masterarbeit, 2009

[Koc08] Koch, Ruben-Eric: *Entwurf und Implementierung eines DDR SDRAM-Controllers in Hardware*, Universität Rostock, Bachelorarbeit, 2008

[Koc10] Koch, Ruben-Eric: *Konzipierung und Implementierung eines ressourcenschonenden Hardware-Moduls zur Erfassung von Statistikdaten*, Universität Rostock, Projektarbeit, 2010

[Lip10] Lipowski, Tilo: *Inbetriebnahme eines Netzwerk-Tranceiver-Moduls für das Entwicklungsboard ML507 von XILINX*, Universität Rostock, Großer Beleg, Dezember 2010

[Pfe10] Pfeiffer, Stefan: *Konzipierung und Implementierung eines ressourcenschonenden Hardware-Filters zur Intrusion-Detection*, Universität Rostock, Masterarbeit, 2010

[Rhi08a] Rhinow, Grit: *Trust-by-Wire in paketvermittelnden IP-Netzen: Analyse und Konzept zur Migration des IPclip Mechanismus von IPv4 nach IPv6*, Universität Rostock, Kleiner Beleg, März 2008

[Rhi08b] Rhinow, Grit: *Trust-by-Wire in paketvermittelnden IP-Netzen: Migration des IPclip-Mechanismus von IPv4 nach IPv6*, Universität Rostock, Großer Beleg, Juli 2008

[Sam10a] Samara, Hani: *Grundlagen zur Netzwerksicherheit - Schutz des Nutzers und des Netzes*, Universität Rostock, Literaturarbeit, November 2010

[Sam10b] Samara, Hani: *Konzipierung und Implementierung eines Hardware-Caches für den beschleunigten Zugriff auf Speicherdaten*, Universität Rostock, Bachelorarbeit, 2010

[Str08] Strzeletz, Andy: *Konzipierung und Umsetzung einer Evaluierungsplattform zur Entwicklung von Algorithmen für PON-Systeme*, Universität Rostock, Masterarbeit, 2008

[Wes10] Westphal, Heiko: *Multicore-Systeme*, Universität Rostock, Vortragsseminar, Mai 2010

[Wol10] Wolff, Johann-Peter: *Erkennung von Virensignaturen*, Universität Rostock, Literaturarbeit, November 2010

[Wol11a] Wolff, Johann-Peter: *Entwurf und Implementierung einer Softwarelösung zur Erfassung und Analyse von Logdaten*, Universität Rostock, Bachelorarbeit, 2011

[Wol11b] Wolff, Johann-Peter: *Internetbedrohungen - Wie schütze ich mich richtig?*, Universität Rostock, Vortragsseminar, Dezember 2011

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die Dissertation zum Thema

Hardware-gestützte Sicherheitskonzepte für Teilnehmerzugangsnetzwerke

vollkommen selbst verfasst habe und zu ihrer Anfertigung keine anderen als die angegebenen Hilfsmittel und Quellen verwendet habe.

Meinersen, den 08.05.2015

Thesen

1. Im Massenmedium Internet, welches über viele Jahre stetig gewachsen ist, nimmt die Anzahl „Schwarzer Schafe“ permanent zu. Da die Mehrheit der Internet-Teilnehmern nicht in der Lage ist, sich ausreichend zu schützen, müssen grundlegende Sicherheits-Services für das Teilnehmerzugangsnetzwerk entwickelt werden.
2. Ein zusätzlich in den Datenpfad des TZN eingebrachtes Sicherheitssystem muss kompromittierungsfrei arbeiten. Dazu zählt, dass es netzwerktechnisch nicht adressierbar ist. Ferner darf es nur durch autorisierte Instanzen wie ISP-Administratoren beeinflussbar sein.
3. Der gesamte Netzwerkverkehr muss durch das Sicherheitssystem geleitet werden. Es darf nicht möglich sein, das System zu unterlaufen.
4. Die lokalen Sicherheitsrichtlinien beim Internet-Nutzer dürfen nicht negative beeinflusst werden. Nutzerseitige Fehlkonfigurationen müssen jedoch behoben werden, sofern die Filter des Sicherheitssystems dies ermöglichen.
5. Im privaten Umfeld werden häufig Software-Lösungen eingesetzt um einzelne Rechner und ganze Netzwerke zu schützen. Im TZN herrschen deutlich höhere Datenraten vor. Sie sind der Grund, weshalb Software-Lösungen im TZN ungeeignet sind.
6. Die minimal zu erzielende Datenrate ist 1 Gbit/s. Es handelt sich um eine typische Datenrate in der sogenannten „First Mile“.
7. Durch ein zusätzlich in den Datenpfad eingebrachtes Sicherheitssystem werden QoS und QoE beeinflusst. Die Beeinflussung darf jedoch für den Nutzer nicht spürbar sein, da er das System ansonsten nicht akzeptieren wird.
8. Um den Durchsatz eines paketverarbeitenden Systems zu maximieren, müssen Redundanzen vermieden werden. So ist es beim Einsatz mehrerer Bewertungsstufen sinnvoll, eine einmalige Paketklassifizierung durchzuführen.

9. Ein zusätzlich in den Datenpfad eingebrachtes Sicherheitssystem muss skalierbar sein, um zukunftssicher mit stetig steigenden Datenraten umgehen zu können. Abgeschlossene Teilsysteme, an die der zu bewertende Datenverkehr weitergeleitet wird, ermöglichen dies.
10. Die SecAN - Firewall, Web-Filter und Intrusion-Detection-System sind modular aufgebaut. Sie werden über eine zentrale Paketklassifizierungseinheit versorgt und stellen jeweils ein abgeschlossenes System dar, welches einen Durchsatz von mindestens einem Gbit/s erzielt.
11. Die Firewall im SecAN-Gesamtsystem besitzt acht Filterstufen, welche die OSI-Schichten 2-4 überwachen. So werden bspw. unautorisierte Zugriffe aus dem Internet auf Netzwerkfreigaben durch Sperren der Ports 137-139 (TCP und UDP) sowie 445 (TCP) auf allen Windows-Systemen verhindert.
12. Das Web-Filtersystem im SecAN schützt vor rechtswidrigen Inhalten durch eine zweistufige Suche in ausschließlich lokalen Ressourcen. Durch Tests unterschiedlicher Hashing-Algorithmen mit 23 Millionen VeriSign-Domains wurde das effiziente CRC64-Verfahren mit einer Kollisionswahrscheinlichkeit von lediglich $6,91 \cdot 10^{-6}$ als am besten geeignet identifiziert. Durch den Verifikationsspeicher leidet das Web-Filtersystem nicht an „Underblocking“.
13. Die freie und quelloffene Software-Lösung „SNORT“ stellt den de facto Standard für ID/IP-Systeme dar. Mit geringen Einschränkungen lassen sich SNORT-Regeln für ein Hardware-IDS verwenden.
14. Je größer das Regelwerk eines IDS, umso schlechter sind externe Speicher geeignet, da viele Vergleiche das Gesamtsystem ausbremsen.
15. Mit Hilfe des von Burton H. Bloom entwickelten Bloom-Filter-Ansatzes können Mengenzugehörigkeiten mit sehr niedrigen Latenzen nachgewiesen werden. Dies gilt auch für SNORT-Regeln in Bloom-Filtern.
16. Hash-Funktionen der Klasse H3 sind sehr gut für die Indexgenerierung von Bloom-Filtern geeignet, da sich ausreichend viele unabhängige Hash-Funktionen generieren lassen, wenn die H3-Matrizen auf echten Zufallszahlen basieren.
17. Jedes Element, das in einem Bloom-Filter hinterlegt wurde, wird bei dessen Suche wiedergefunden. Somit erzeugen Bloom-Filter keine „False Nagatives“.
18. Durch die eingesetzten Hash-Methoden können „False Positives“ entstehen. Sie müssen aufgelöst werden. Das Bloom-Filter-basierte IDS im SecAN weist eine False

Positive Rate von 1 ‰ auf, wodurch ein Verifikationsspeicher durchschnittlich 1000 Takte für die Überprüfung Zeit hat.

19. Die vorliegende Forschungsarbeit beweist die Machbarkeit von Hardware-basierten Sicherheitslösungen für das Teilnehmerzugangsnetzwerk. Dabei liegt die Latenz des Gesamtsystems mit $\approx 4,201\mu s$ deutlich unter der für Nutzer wahrnehmbaren Schwelle von 0,1 Sekunden.

20. Durch die Einführung weiterer Services könnten zukünftig Hersteller von Antimalware mit ISP zusammenarbeiten, um ihre Geschäftsportfolien gemeinsam zu erweitern. Somit ließen sich die Reaktionszeiten, um Malware wirksam entgegen zu treten, stark reduzieren.

21. Mit den erweiterten Portfolien gehören auch Unternehmen, die ihre Geschäftsgeheimnisse gewahrt wissen wollen, zu potentiellen Kunden.

Kurzreferat

Das Internet: Aus dem einst rein wissenschaftlichen Netz ist vor vielen Jahren ein Massenmedium geworden. Obwohl das Internet bereits einige Evolutionsstufen durchlaufen hat, beruht es im Wesentlichen immer noch auf den Protokollen aus dessen Entstehungszeit. Viele Schwachstellen wurden identifiziert und Workarounds geschaffen. Jedoch bestehen nach wie vor viele reale Gefahren für einzelne Rechner und ganze Rechnernetze. Besonders Internet-Nutzer mit geringem netzwerktechnischen Wissen sind diesen Bedrohungen ausgesetzt, da sie häufig auch mit Software-Lösungen im Bereich Netzwerksicherheit überfordert sind. Die vorliegende Forschungsarbeit zeigt einen Lösungsweg aus dem beschriebenen Dilemma, indem Hardware-basierte Sicherheitslösungen im TZN angesiedelt werden. Im Rahmen des SecAN-Projektes wurden Konzepte für eine Firewall, einen Web-Filter und ein Intrusion-Detection-System ausgearbeitet. Da im TZN deutlich höhere Datenraten als im privaten Bereich vorherrschen, wurden die konzipierten Lösungen auf FPGAs umgesetzt. Ganz bewusst wurde sparsam mit dem Einsatz externer Speichern umgegangen, da diese zu höheren Verzögerungen führen. Das Gesamtsystem ist weitestgehend frei von Manipulationen, da es außer vom ISP-Administrator nicht adressiert und konfiguriert werden kann. Zudem ist es frei skalierbar und somit sicher für zukünftig steigende Datenraten.

Im Laufe der Arbeit entstanden so drei voll funktionsfähige, modular aufgebaute Prototypen, die separat betrieben werden können. Der größte Mehrwert entsteht jedoch, wenn diese im Verbund arbeiten. Das Design-Konzept ist so gestaltet, dass das Gesamtsystem leicht gewartet und erweitert werden kann. Somit kann bspw. das IDS-System dahingehend erweitert werden, dass es Malware-Signaturen erkennt und darauf reagiert, da deren Anzahl stetig wächst. Ferner kann eine anomalieerkennende Filterstufen umgesetzt werden, die z. B. DDoS-Attacken unterbindet. Mit den erweiterten Security-Services des SecAN-Projektes können neben der derzeitigen Zielgruppe - dem durchschnittlichen Internet-Nutzer - auch Unternehmen als Kunden akquirieren werden.

Abstract

Since many years, the Internet has outgrown from its infancy. Originally developed as purely scientific network, it has become a mass medium. Although the Internet has been through some evolutionary stages, it still bases on old protocols from its time of origin. Many weaknesses were identified and workarounds were created. However, there are many threats for computers and computer networks. Especially, Internet users with low network technical knowledge are exposed to these threats. Often, the user uses software solutions, which overwhelm them. The presented PhD thesis offers a solution to this dilemma by installing a hardware-based security solution inside the access area of the Internet. In the course of the thesis, concepts for a firewall, a Web filter, and an intrusion detection system were developed. As in the access area considerably higher data rates than in the customers area are given, the presented solutions were developed as FPGA solutions. Deliberately, the use of external memory is restricted as the access to this memory leads to higher delays. The complete system is safe from manipulation as it is not addressable from the Internet. Only ISP administrators have the right to configure it. Besides, it is highly scalable and therefore future-proof for increasing data rates.

During the thesis, 3 fully functional, modular and separately working prototypes were developed. The system delivers the highest benefit if all prototypes work in combination. The design concept offers an easy way to maintain and extend the complete system. For example, it is possible to extend the intrusion detection system in such a way that it detects and reacts on malware signatures reliably even though the number of known signatures increases rapidly. Furthermore, it is reasonable to implement an anomaly detecting control stage that eliminates DDoS attacks. From the service view, not only private customers can use the new security service in the access area but also companies can benefit from these services.