

Entwurf und Implementierung eines Gateway-Systems für die Netzwerksegmentierung nach dem Zero-Trust-Modell

Abschlussarbeit

für die Prüfung zum
Bachelor of Science

des Studiengangs Informatik
an der Dualen Hochschule Baden-Württemberg Stuttgart

von
Maximilian Hengl

09.09.2024

Bearbeitungszeitraum
Matrikelnummer, Kurs
Ausbildungsfirma
Betreuer
Gutachter

12 Wochen
4907952, TINF21E
telent GmbH, Backnang
Karsten Luchs
Andreas Buckenhofer

Erklärung

Ich versichere hiermit, dass ich meine Abschlussarbeit mit dem Thema: *Entwurf und Implementierung eines Gateway-Systems für die Netzwerksegmentierung nach dem Zero-Trust-Modell* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Stuttgart, 09.09.2024

Maximilian Hengl

Zusammenfassung

Diese Arbeit behandelt die Entwicklung und Implementierung eines Gateway-Systems zur Netzwerksegmentierung gemäß dem Zero-Trust-Modell. Sie fokussiert sich darauf, wie ein solches System die Netzwerkkommunikation präzise steuern und absichern kann. Methodisch wird ein mehrstufiger Ansatz verfolgt, der die Analyse von Kommunikationsprotokollen und Firewall-Typen, die Definition spezifischer Anforderungen sowie den Vergleich vorhandener Firewall-Lösungen umfasst. Die Arbeit stützt sich auf technische Literatur, Marktforschungsberichte und Produktdokumentationen. Das Ergebnis ist ein Konzept für eine Firewall-Erweiterung, die eine granulare Filterung von Datenpaketen ermöglicht. Aufgrund zeitlicher Einschränkungen konnte jedoch nur ein Teil der Benutzeroberfläche implementiert werden, was eine funktionale Testung verhinderte. Die Arbeit schließt mit einer kritischen Diskussion der Ergebnisse und identifiziert zukünftige Forschungs- und Entwicklungsperspektiven im Bereich der Zero-Trust-Netzwerksegmentierung.

Abstract

This thesis deals with the development and implementation of a gateway system for network segmentation according to the Zero Trust model. It focusses on how such a system can precisely control and secure network communication. Methodologically, a multi-stage approach is followed, which includes the analysis of communication protocols and firewall types, the definition of specific requirements and the comparison of existing firewall solutions. The work is based on technical literature, market research reports and product documentation. The result is a concept for a firewall extension that enables granular filtering of data packets. Due to time constraints, however, only part of the user interface could be implemented, which prevented functional testing. The thesis concludes with a critical discussion of the results and identifies future research and development perspectives in the area of zero trust network segmentation.

Inhaltsverzeichnis

Abkürzungsverzeichnis	VI
Abbildungsverzeichnis	VII
Listings	IX
1 Einleitung	1
2 Aufgabenstellung	2
3 Wirtschaftliche Betrachtung	3
4 Stand der Technik	4
4.1 Kommunikationsprotokolle	4
4.1.1 Vermittlung	5
4.1.2 Transport	6
4.1.3 Anwendung	8
4.2 Firewall	19
4.2.1 Paketfilter	19
4.2.2 Application-Gateway	20
4.2.3 Circuit-Gateway	21
4.2.4 Next Generation Firewall	21
4.3 Netzwerksicherheitszonen	23
4.4 Zero Trust	23
5 Anforderungen	26
5.1 Allgemein	26
5.2 Anwendungsszenarien	27
5.2.1 Prozessleitsysteme	27
5.2.2 Intelligente Stromnetze	29
5.2.3 Medizinische IoT-Geräte	30
6 Ansätze	32
6.1 Bestehende Firewall Produkte	32
6.1.1 pfSense	32
6.1.2 OPNsense	33
6.1.3 Quantum Force	34
6.1.4 FortiGate	35
6.1.5 Palo Alto Networks	36
6.1.6 Firepower	36

6.2	Bestehende OPNsense Erweiterungen	37
6.2.1	Zenarmor	37
6.2.2	Suricata	37
6.2.3	Maltrail	38
6.2.4	ClamAV	38
6.2.5	FreeRADIUS	38
6.2.6	Postfix	39
6.3	Eigene Erweiterung	39
6.3.1	Low-Fidelity Prototyp	39
6.3.2	Scapy	42
6.3.3	Reguläre Ausdrücke	43
6.3.4	Programmablauf	44
7	Implementierung einer Erweiterung	47
7.1	Erste Einrichtung von OPNsense	47
7.2	Entwicklung der Benutzerschnittstelle	49
8	Diskussion	56
9	Ausblick	58
	Literatur	60

Abkürzungsverzeichnis

OSI	Open Systems Interconnection
IP	Internet Protocol
IANA	Internet Assigned Numbers Authority
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
DNS	Domain Name System
NTP	Network Time Protocol
SMTP	Simple Mail Transfer Protocol
LDAP	Lightweight Directory Access Protocol
SNMP	Simple Network Management Protocol
DHCP	Dynamic Host Configuration Protocol
RADIUS	Remote Authentication Dial-In User Service
TLS	Transport Layer Security
URL	Uniform Resource Locator
CDM	Continuous Diagnostics and Mitigation
PKI	Public Key Infrastructure
PDU	Protocol Data Unit
MD5	Message-Digest Algorithm 5
IoT	Internet of Things
SCADA	Supervisory Control and Data Acquisition
GPS	Global Positioning System
DNP3	Distributed Network Protocol
HTML	Hypertext Markup Language
XML	Extensible Markup Language
PDF	Portable Document Format
OLE	Object Linking and Embedding

Abbildungsverzeichnis

3.1	Weltweiter Jahresumsatz im Netzwerksicherheit-Markt: reale Werte bis 2023 und prognostizierte Werte danach [35]	3
4.1	Vergleich des OSI-Modells mit dem TCP/IP-Modell	4
4.2	TCP-Verbindungsaufbau [13]	6
4.3	TCP-Header	7
4.4	UDP-Header	8
4.5	Diffie-Hellman-Schlüsselaustausch mit öffentlicher Primzahl p , öffentlicher natürlicher Zahl $g < p$, privaten Schlüsseln $a, b < p$ und Sitzungsschlüssel K [11]	9
4.6	DNS-Header	10
4.7	Schematische Darstellung der rekursiven und iterativen DNS-Abfrage [21]	10
4.8	RADIUS-Header	13
4.9	Das Prinzip der SNMP-Kommunikation [4]	14
4.10	SNMP-Paket	15
4.11	DHCP-Header	16
4.12	NTP-Paket ohne optionale Felder	17
4.13	DNP3-Antwortpaket	18
4.14	Modbus-Header	18
4.15	Logische Zero Trust Architektur [39]	24
5.1	Beispielhaftes Prozessleitsystem [55, S. 28]	28
5.2	Aufbau eines intelligenten Stromnetzes [33]	30
6.1	pfSense Firewall Richtlinien	33
6.2	Vergleich proprietärer Firewalls aus Marktforschungsbericht von Miercom [36]	35
6.3	OPNsense Firewall Filterregeln oben und Filterregeln der Erweiterung Suricata unten	40
6.4	Low-Fidelity Prototyp der Hauptseite	41
6.5	Low-Fidelity Prototyp der Einstellungsseite für Filterregeln	42
6.6	Programmablauf für die einfache Paketinspektion	44
6.7	Programmablauf für Filterung auf Basis der Paketinspektion	45
7.1	OPNsense Konsole in einer virtuellen Maschine	48

7.2	<i>Zenarmor</i> -Reiter auf der <i>OPNsense</i> -Konfigurationsseite	49
7.3	Ordnerstruktur der Erweiterung	50

Listings

7.1	<code>mvc/app/models/Telent/ZeroTrust/ZeroTrust.xml</code>	Grundstruktur . . .	51
7.2	<code>mvc/app/views/Telent/ZeroTrust/index.volt</code>	Grundstruktur	51
7.3	<code>mvc/app/controllers/Telent/ZeroTrust/IndexController.php</code>	Grundstruktur	52
7.4	<code>mvc/app/controllers/Telent/ZeroTrust/forms/general.xml</code>	Grundstruktur	52
7.5	<code>mvc/app/views/Telent/ZeroTrust/index.volt</code>	JavaScript	53
7.6	<code>mvc/app/controllers/Telent/ZeroTrust/Api/ServiceController.php</code>	Grundstruktur	53
7.7	<code>service/templates/Telent/ZeroTrust/zerotrust.conf</code>	Grundstruktur	54
7.8	<code>scripts/Telent/ZeroTrust/filter.py</code>	ConfigParser	54
7.9	<code>scripts/Telent/ZeroTrust/filter.py</code>	sniff	55

1 Einleitung

In der heutigen digitalen Ära hat die zunehmende Vernetzung von Systemen über das Internetprotokoll die IT-Landschaft grundlegend verändert. Diese Entwicklung bringt nicht nur Vorteile mit sich, sondern stellt auch erhöhte Anforderungen an die IT-Sicherheit, insbesondere wenn es darum geht, sensible Systeme oder Netzbereiche zu schützen. Traditionell wurden kritische Infrastrukturen wie Produktionsanlagen, Sicherheitssysteme sowie Entwicklungs- und Laborumgebungen oft vollständig isoliert betrieben, um maximale Sicherheit zu gewährleisten. Dieser Ansatz erwies sich jedoch als zunehmend unpraktisch, da er wesentliche Funktionen einschränkte und den Betrieb erheblich erschwerte.

Die Notwendigkeit von Fernwartung, effizientem Datentransfer und präziser Zeitsynchronisation hat dazu geführt, dass selbst hochsensible Bereiche über Firewalls mit anderen Netzen verbunden werden müssen. Obwohl herkömmliche Firewalls einen wichtigen Schutz bieten, arbeiten sie primär auf der Ebene von Adressen und Protokollen. Dies kann dazu führen, dass trotz ihrer Implementierung unerwünschter Datenverkehr zugelassen wird, was potenzielle Sicherheitsrisiken birgt. In diesem Kontext gewinnt das Zero-Trust-Modell zunehmend an Bedeutung. Es stellt einen Paradigmenwechsel in der Netzwerksicherheit dar, indem es das Prinzip *Vertraue niemandem, überprüfe alles* propagiert. Dieser Ansatz erfordert eine kontinuierliche Authentifizierung und Autorisierung jeder Interaktion innerhalb des Netzwerks, unabhängig von der Herkunft oder dem Ziel des Datenverkehrs.

Die vorliegende Arbeit widmet sich der Entwicklung und Implementierung eines Gateway-Systems, das die Netzwerksegmentierung gemäß dem Zero-Trust-Modell ermöglicht. Durch eine detaillierte Analyse der Kommunikationsprotokolle und der Funktionsweise verschiedener Firewall-Typen wird das Ziel verfolgt, die Kommunikation innerhalb eines Netzwerks präziser zu steuern und abzusichern. Dabei werden sowohl bestehende Technologien als auch innovative Erweiterungen betrachtet, um ein robustes Sicherheitskonzept zu entwickeln, das den Anforderungen moderner IT-Infrastrukturen gerecht wird. Im Rahmen dieser Untersuchung wird ein besonderes Augenmerk auf die Implementierung eines Prototyps gelegt, der eine granulare Filterung eingehender Datenpakete ermöglichen kann.

2 Aufgabenstellung

Das primäre Ziel besteht in der Erforschung und Entwicklung von Lösungsansätzen zur Implementierung einer Zero-Trust-Architektur in Netzwerken mittels eines Gateway-Systems. Zu diesem Zweck werden zunächst verschiedene Kommunikationsprotokolle und deren Funktionsweisen einer eingehenden Analyse unterzogen. Darauf aufbauend sollen Methoden zur Einschränkung der Kommunikation über diese Protokolle erarbeitet werden. Ein weiterer Schwerpunkt liegt auf der Erörterung verschiedener Firewall-Typen, des Konzepts der Netzwerksicherheitszonen sowie der Grundlagen der Zero-Trust-Architektur.

Im Anschluss daran werden sowohl allgemeine als auch spezifische Anforderungen an ein Gateway-System für bestimmte Anwendungsszenarien definiert. Es folgt ein Vergleich verschiedener Firewall-Lösungen, wobei eine für den weiteren Verlauf der Arbeit ausgewählt wird. Die bestehende Funktionalität dieser Firewall wird detailliert untersucht, gefolgt von ihrer Installation.

Ein zentraler Aspekt der Arbeit besteht in der Konzeption einer Erweiterung, die es der Firewall ermöglichen sollte, eingehende Datenpakete basierend auf ihrem Inhalt zu filtern. Ursprünglich war vorgesehen, diese Erweiterung zu implementieren und zu testen. Aufgrund zeitlicher Beschränkungen konnte jedoch lediglich ein Teil der Benutzeroberfläche realisiert werden, wodurch eine funktionale Testung nicht möglich war.

Die Arbeit schließt mit einer kritischen Diskussion der erzielten Ergebnisse sowie der Identifikation von Forschungslücken. Im Ausblick werden Perspektiven für zukünftige Forschungs- und Entwicklungsarbeiten aufgezeigt, einschließlich möglicher Testverfahren für eine vollständig entwickelte Erweiterung.

3 Wirtschaftliche Betrachtung

Netzwerksicherheit wird angesichts der fortgeschrittenen Digitalisierung und deren Rolle bei aktuellen kriegerischen Auseinandersetzungen immer wichtiger. Das macht sich auch am Umsatz der Branche bemerkbar. Nach Angaben der *Statista GmbH* steigt dieser stetig. In den nächsten fünf Jahren wird ein Umsatzwachstum um fast fünfundsiebzehn Prozent erwartet. [35]

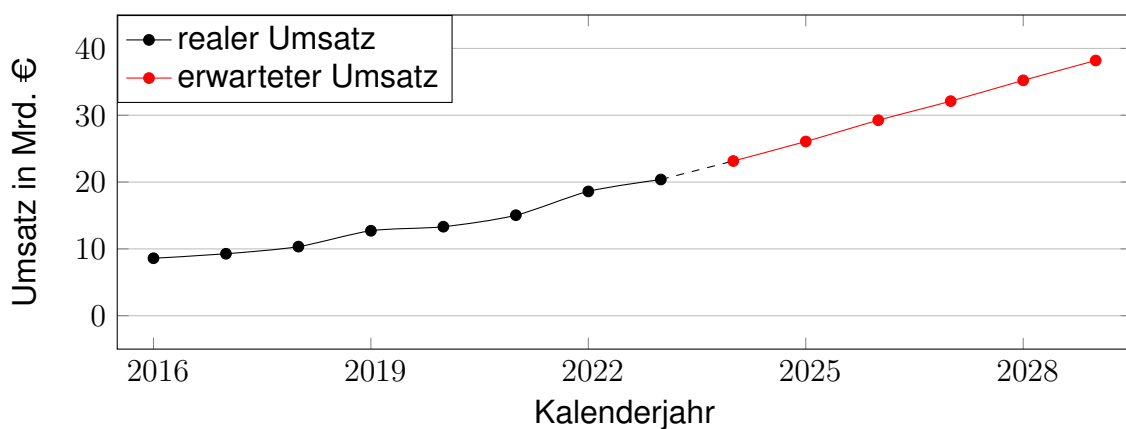


Abbildung 3.1: Weltweiter Jahresumsatz im Netzwerksicherheit-Markt: reale Werte bis 2023 und prognostizierte Werte danach [35]

Die *telent GmbH* hat sich auf die Planung, Installation und Wartung von Kommunikationsnetzwerken für kritische Infrastrukturen spezialisiert. Betreiber von kritischer Infrastruktur sind dazu verpflichtet, Netzwerksicherheit auf dem Stand der Technik umzusetzen und das alle zwei Jahre dem Bundesamt für Sicherheit in der Informationstechnik nachzuweisen [62]. Unter der Marke *KORAMIS* bietet die *telent GmbH* Sicherheitskonzepte für Industrienetze an.

Für die Implementierung eines Sicherheitskonzepts werden hauptsächlich Produkte von Drittanbietern verwendet. Ein eigenes Produkt, das die Kommunikation zwischen Netzwerkkomponenten auch auf Anwendungsebene nach dem Zero-Trust-Prinzip filtert, kann ein Alleinstellungsmerkmal für *KORAMIS* werden und einen neuen Stand der Technik schaffen, dessen Umsetzung verpflichtend wird.

4 Stand der Technik

Die Betrachtung des aktuellen Stands der Technik ist unerlässlich, wenn dieser weiterentwickelt werden soll. In diesem Abschnitt wird zunächst betrachtet, wie die Kommunikation zwischen verschiedenen Netzwerkkomponenten funktioniert. Danach werden verschiedene Firewall-Typen zur Filterung dieser Kommunikation betrachtet und zwei moderne Netzwerksicherheitskonzepte vorgestellt.

4.1 Kommunikationsprotokolle

Die Kommunikation zwischen zwei oder mehreren Diensten oder Geräten erfordert die Einhaltung gemeinsamer Regeln. Diese werden in sogenannten Kommunikationsprotokollen festgehalten. Um die Kommunikation eines Netzwerks einschränken zu können, ist es folglich von essenzieller Bedeutung, die entsprechenden Protokolle zu verstehen. Die Unterscheidung zwischen verschiedenen Aufgaben, die Protokolle erledigen können, erfolgt durch das [OSI](#)- und das [TCP/IP](#)-Referenzmodell.

OSI	TCP/IP
Anwendung	Anwendung
Darstellung	
Sitzung	
Transport	Transport
Vermittlung	Internet
Sicherung	Netzzugriff
Bitübertragung	

Abbildung 4.1: Vergleich des [OSI](#)-Modells mit dem [TCP/IP](#)-Modell

Innerhalb des **TCP/IP**-Modells werden Aufgabenbereiche des **OSI**-Modells, die in der Realität meistens von einem Protokoll erledigt werden, als übergeordneter Aufgabenbereich zusammengefasst. Die Aufgabenbereiche Bitübertragung und Sicherung werden unter Netzzugriff zusammengefasst. In beiden Modellen werden die Prozesse der Vermittlung und des Transports voneinander getrennt betrachtet. Das **TCP/IP**-Modell fasst die Aufgaben Sitzung, Darstellung und Anwendung unter der Aufgabe Anwendung zusammen. [61] [24] [3]

4.1.1 Vermittlung

Das Internet Protocol (**IP**) wird als grundlegendes Protokoll für die Vermittlung von Daten betrachtet, da es die Basis für die Kommunikation zwischen Endgeräten in einem modernen Netzwerk bildet. Die Funktion des Protokolls besteht in der logischen Adressierung von Geräten innerhalb von Netzwerken, welche eine grundlegende Voraussetzung für das Routing darstellt. Das Protokoll ist in der ursprünglichen Version **IPv4** und in der neuen Version **IPv6** verfügbar. Die Vergabe globaler **IP**-Adressen erfolgt durch die *Internet Assigned Numbers Authority (IANA)*, welche diese an regionale Vergabestellen delegiert. Ist ein Netzwerk jedoch nicht mit dem Internet verbunden, ist eine weltweit eindeutige **IP**-Adresse nicht erforderlich, sodass die Adressen selbst vergeben werden können.

Eine **IPv4**-Adresse setzt sich aus vier Bytes zusammen, die durch Punkte voneinander getrennt dargestellt werden. Die Aufteilung in einen Netzanteil und einen Geräteanteil erfolgt mittels sogenannter Netzmasken. Eine Netzmaske wird in binärer Darstellung durch eine Folge von Einsen und einer darauf folgenden Folge von Nullen definiert. Die Einsen repräsentieren dabei die Bit der **IPv4**-Adresse, welche den Netzanteil bilden, während die Nullen die Bit darstellen, die den Geräteanteil bilden. Die Aufteilung eines Netzes in mehrere Subnetze erlaubt eine logische Gruppierung von Geräten. [41]

Im Rahmen des **IPv6**-Protokolls haben Adressen eine Länge von sechzehn Bytes. Sie werden hexadezimal dargestellt, wobei nach vier Ziffern ein Doppelpunkt als Trennzeichen folgt. Es besteht die Möglichkeit, Blöcke, die lediglich aus Nullen bestehen, zu ignorieren. Ein solcher Block kann folglich durch zwei aufeinanderfolgende Doppelpunkte dargestellt werden. Falls ausschließlich solche Blöcke folgen, endet die Adresse dort. Die Darstellung von Netzen erfolgt in **IPv6** durch ein sogenanntes Präfix. Dieses besteht aus dem ausgeschriebenen Netzanteil, gefolgt von einem Schrägstrich und der dezimalen Angabe der Bitlänge des Netzanteils. [12]

4.1.2 Transport

Die bedeutendsten Protokolle der Transportschicht sind das Transmission Control Protocol (**TCP**) und das User Datagram Protocol (**UDP**). Die Funktion beider Protokolle besteht darin, die Datenübertragung zu kontrollieren. Während **TCP** verbindungsorientiert arbeitet, erfolgt die Datenübertragung mit **UDP** verbindungslos. Beide nutzen Ports um Prozesse zu adressieren.

Vor der Übertragung von Daten über **TCP** muss folglich eine Verbindung etabliert werden. Der Client übermittelt dem Server dafür ein sogenanntes **SYN**-Paket, welches den Verbindungswunsch des Clients signalisiert. Das betreffende Paket beinhaltet eine gesetzte **SYN**-Flag sowie eine initiale Sequenznummer, die als **SEQ** bezeichnet wird. Der Server bestätigt den Empfang und erwidert den Verbindungswunsch. Dazu sendet er ein Paket mit gesetzter **SYN**-Flag, eigener initialer Sequenznummer **SEQ** und der Empfangsbestätigung **ACK** für das eingetroffene **SYN**-Paket. Die Empfangsbestätigung **ACK** hat dabei den Wert der eingetroffenen Sequenznummer **SEQ** um die Zahl Eins inkrementiert. Der Verbindungsaufbau wird durch den Client abgeschlossen, indem der Empfang des **SYN/ACK**-Pakets über ein **ACK**-Paket bestätigt wird, welches demselben Schema ohne gesetzte **SYN**-Flag folgt.

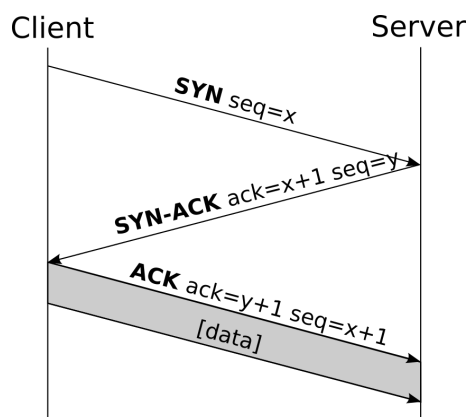


Abbildung 4.2: **TCP**-Verbindungsaufbau [13]

Der Verbindungsaufbau informiert beide Endpunkte über die verwendeten Ports und die IP-Adresse des Gegenübers. Außerdem einigen sich die Verbindungspartner über die Felder Receive Window und Maximum Segment Size auf eine maximale Paketgröße. Die Felder **SEQ** und **ACK** sorgen dafür, dass die Daten in der richtigen Reihenfolge ankommen. Wenn ein Paket nicht ankommt, kann der Server es noch mal senden. **TCP**

sorgt auch dafür, dass die Daten nicht verändert werden. Dazu wird eine Prüfsumme erstellt.

Ein Verbindungsende wird durch das Senden eines Pakets mit gesetzter FIN-Flag initialisiert. Der Verbindungspartner antwortet mit einem FIN/ACK-Paket, woraufhin ein letztes ACK-Paket folgt. Der Wert MSL gibt die maximale Lebenszeit eines Pakets an. Nach dem Senden des letzten ACKs wartet der Verbindungspartner für die Dauer von zwei maximalen Lebenszeiten auf verspätete Pakete, welche verworfen werden. Wenn das Netzwerk überlastet ist, kann das dem Verbindungspartner mitgeteilt werden. Dazu wird die Flag ECE gesetzt. Der Verbindungspartner kann darauf mit der Flag CWR antworten und bestätigen, dass er die Senderate reduziert hat. [16]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
IP-Absenderadresse																															
IP-Empfängeradresse																															
0								6								Länge															
Quellport																Zielport															
SEQ																															
ACK																															
Offset				0				Flags								Empfangsfenstergröße															
Prüfsumme																Urgent Pointer															
Optional																															

Abbildung 4.3: TCP-Header

UDP ist einfacher aufgebaut als TCP, wodurch die Datenübertragung schneller ist. Der Header eines UDP-Pakets enthält die Adresse des Senders und Empfängers sowie die Ports. Außerdem enthält er ein Feld, das die Länge des Pakets angibt. Eine Prüfsumme ist optional.

Datenpakete werden ohne Vorwarnung gesendet. UDP bestätigt nicht, ob Pakete angekommen sind. Wenn ein Paket fehlerhaft ist, kann der Empfänger dem Sender nur mitteilen, dass die Übertragung fehlerhaft war, aber nicht, welches Paket genau fehlerhaft war. UDP sollte nur verwendet werden, wenn die Geschwindigkeit wichtiger als die Zuverlässigkeit ist. Das kann zum Beispiel beim Streaming der Fall sein. [40]

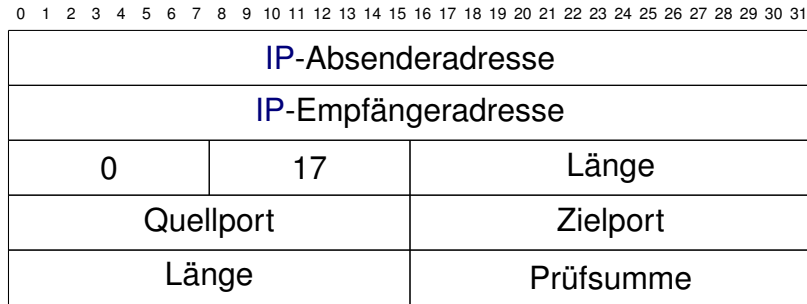


Abbildung 4.4: UDP-Header

4.1.3 Anwendung

In der Anwendungsschicht finden sich alle Kommunikationsprotokolle, die den Austausch anwendungsspezifischer Daten ermöglichen. Zu den essenziellen Protokollen zählen das Hypertext Transfer Protocol ([HTTP](#)), das Domain Name System ([DNS](#)), das Simple Mail Transfer Protocol ([SMTP](#)) und das Lightweight Directory Access Protocol ([LDAP](#)). Darüber hinaus werden der Remote Authentication Dial-In User Service ([RADIUS](#)), das Syslog-Protokoll, das Simple Network Management Protocol ([SNMP](#)), das Dynamic Host Configuration Protocol ([DHCP](#)), das Network Time Protocol ([NTP](#)), das Distributed Network Protocol ([DNP3](#)) und das Modbus-Protokoll behandelt.

Hypertext Transfer Protocol [HTTP](#) wird benutzt, um Webseiten aus dem Internet in einen Browser zu laden. Es kann aber auch für die Übertragung anderer Dateien verwendet werden. Es benutzt dabei standardmäßig den Port 80. Die verschlüsselte Version [HTTPS](#) benutzt den Port 443.

[HTTP](#) ist zustandslos und baut auf einem zuverlässigen Transportprotokoll auf. Hierfür wird in den Versionen [HTTP/1.0](#), [HTTP/1.1](#) und [HTTP/2](#) [TCP](#) verwendet. [HTTP/3](#) verwendet ein auf [UDP](#) basierendes Transportprotokoll. Diese Version ist allerdings bisher mit keinem gängigem Browser kompatibel.

[HTTP](#) bietet eine Vielzahl an Anfragemethoden, wobei GET und POST die gängigsten Methoden sind. Der Header einer Anfrage besteht aus dem Namen der Methode, dem [URL](#)-kodierte Inhalt der Anfrage, dem verwendeten Protokoll und der Domain des Hosts. Außerdem können weitere optionale Parameter verwendet werden. Im Gegensatz

zur Methode GET, die den gesamten Inhalt der Anfrage in einem Hyperlink speichert, speichert POST nur die angefragte Webseite im Hyperlink, während es Anfragen an die Webseite selbst versteckt. Daher eignet sich POST auch zur Übertragung größerer Datenmengen.

Neben GET und POST werden auch die Methoden PUT, PATCH und DELETE häufig genutzt, um Dokumente hochzuladen, bestehende Dokumente zu verändern oder sie zu löschen. HTTP bietet außerdem zwei Methoden zur Authentifizierung und eine Methode um Daten unter Verwendung eines beliebigen Kompressionsverfahrens zu komprimieren. Antworten auf Anfragen enthalten bei HTTP immer Statuscodes, die auf die Art der Antwort hinweisen. So gibt es verschiedene Fehlercodes und Codes, die auf Informationen, eine erfolgreiche Operation oder eine Umleitung hinweisen. [2]

Zur Gewährleistung einer sicheren Verbindung wird bei HTTPS das Verschlüsselungsprotokoll Transport Layer Security (TLS) eingesetzt. Dieses verwendet unter anderem das Diffie-Hellman-Verfahren, um einen Sitzungsschlüssel zwischen Client und Server zu generieren, über den dann die Daten vor dem Austausch über ein Strom- oder Blockchiffre verschlüsselt werden und danach wieder entschlüsselt werden können. [17] [45]

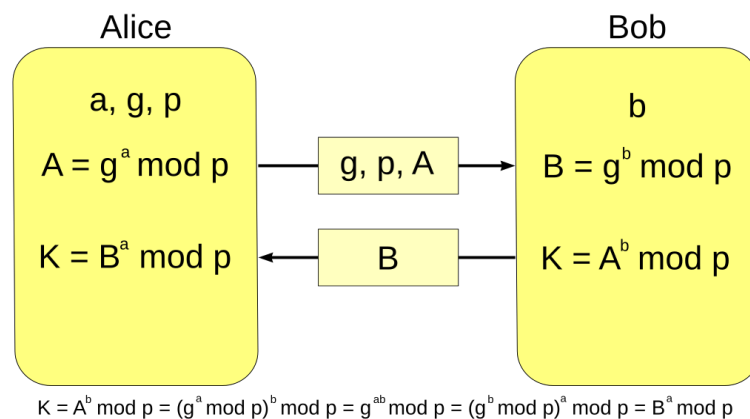


Abbildung 4.5: Diffie-Hellman-Schlüsselaustausch mit öffentlicher Primzahl p , öffentlicher natürlicher Zahl $g < p$, privaten Schlüsseln $a, b < p$ und Sitzungsschlüssel K [11]

Domain Name System Damit HTTP funktioniert, muss es Domains IP-Adressen zuordnen können. Diese Aufgabe übernimmt DNS. Das Protokoll besteht aus zwei Komponenten: Resolvern und Nameservern. Resolver bieten Schnittstellen, über die Anwendungen die Namensauflösung einer Domain anfragen können. Sie haben einen

zugeordneten Nameserver, an den sie die Anfragen weiterleiten können. Dies geschieht in der Regel über **UDP**.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ID																Flags															
Anzahl Fragen																Anzahl Antworten															
Anzahl Quellen																Anzahl zus. Informationen															

Abbildung 4.6: **DNS-Header**

Kennt der Nameserver die **IP**-Adresse zur angefragten Domain, sendet er diese zurück. Wenn er sie nicht kennt, aber über einen eigenen Resolver verfügt, leitet er die Anfrage an einen anderen Nameserver weiter und speichert das Ergebnis seiner Anfrage für eine begrenzte Zeit in Form eines sogenannten Records im Arbeitsspeicher. Dieser Prozess wird rekursive Namensauflösung genannt.

Bei der iterativen Namensauflösung verweist der Nameserver auf einen anderen Nameserver und der Resolver muss die Anfrage selbst an den nächsten Nameserver stellen. Domains werden von hinten nach vorne aufgelöst. Das heißt, wenn die Domain **www.example.edu** aufgelöst wird, wird erst nach einem Nameserver gesucht, der für die Top-Level-Domain **edu** verantwortlich ist. Über die 13 Root-Nameserver, die alle Top-Level-Domains kennen, ist nur iterative Namensauflösung möglich. [29] [30]

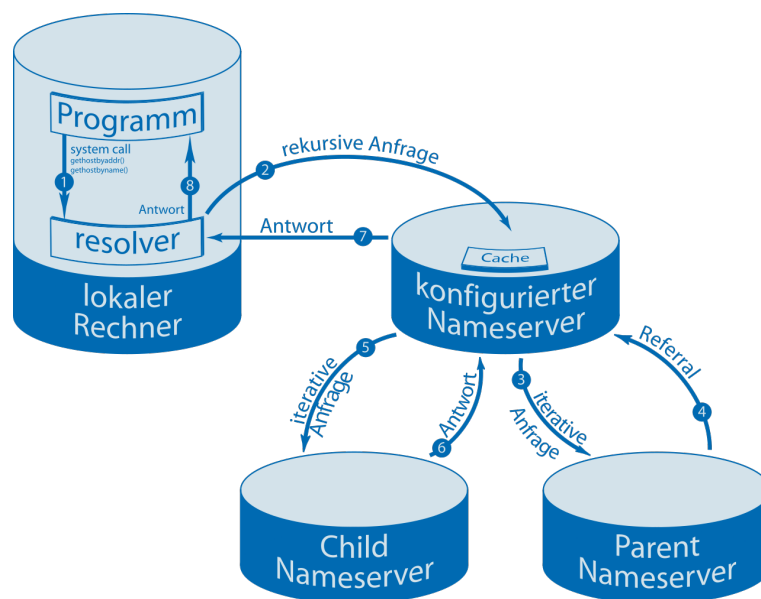


Abbildung 4.7: Schematische Darstellung der rekursiven und iterativen **DNS-Abfrage** [21]

Simple Mail Transfer Protocol [SMTP](#) ist ein Protokoll für den Austausch von E-Mails. Es benutzt standardmäßig den Port 25. Für [TLS](#)-verschlüsselte Kommunikation kann aber auch Port 465 oder Port 587 als Mail Submission Agent verwendet werden. [SMTP](#) verwendet ausschließlich [TCP](#) als Transportprotokoll.

Für das versenden einer E-Mail baut der Client zunächst eine [TCP](#)-Verbindung zum Mail-Server auf. Dann übermittelt er ihm seine Domain. Wenn der Server den Empfang bestätigt hat, nennt ihm der Client die Absenderadresse. Nach deren Bestätigung führt der Client mit der Empfängeradresse fort. Dann kündigt er den Inhalt der E-Mail an. Die E-Mail besteht aus einem Header und den Nutzdaten. Im Header stehen die Absenderadresse, die Empfängeradresse, der Titel und das Datum. Die beiden Adressen müssen dabei nicht mit den Adressen übereinstimmen, die dem Mailserver übermittelt wurden. Sie dienen nur der Darstellung im Mail User Agent. Der Beginn der Nutzdaten wird durch eine Leerzeile angekündigt, ihr Ende durch eine Zeile aus einem einzigen Punkt. Nach Absenden der E-Mail muss der Client noch die Verbindungstrennung anfordern.

Mail User Agents sind die Benutzerschnittstelle für E-Mails. Damit Mail Transfer Agents nicht mit Spam-E-Mails überlastet werden, wird ein Mail Submission Agent zwischen den Mail User Agent und den Mail Transfer Agent geschaltet. Der Mail Submission Agent akzeptiert nur E-Mails von berechtigten Nutzern. Ähnlich wie [HTTP](#) benutzt [SMTP](#) Statuscodes zur Kommunikation von Fehlern und Wartezeiten. [25]

Lightweight Directory Access Protocol Über [LDAP](#) kann auf verteilte Verzeichnisdienste zugegriffen werden. Es ist auf Authentifizierung, Autorisierung und Adressbuchsuchen optimiert und wird auch hauptsächlich dafür eingesetzt. [LDAP](#) benutzt [TCP](#) als Transportprotokoll und ist über den Port 389 erreichbar. Mit [TLS](#) gesicherte Verbindungen sind auf Port 636 erreichbar.

[LDAP](#)-Verzeichnisse sind hierarchisch organisiert und folgen einer Baumstruktur. Daten werden als Objekte gespeichert, deren Schemata durch Klassen definiert werden. Jedes Objekt gehört zu mindestens einer Klasse, die seine Attribute und deren erlaubte Werte definiert. Abstrakte Klassen definieren ausschließlich die Schemata untergeordneter Klassen, keiner Objekte. Objekte werden eindeutig durch ihren Distinguished Name definiert, der sich aus den Relative Distinguished Names, die seine Klassen und seine Objekt-ID definieren, zusammensetzt.

Eine **LDAP** Anfrage besteht aus dem Feld `bind`, über das der Distinguished Name des Benutzers vermittelt wird, der den Zugriff anfordert, und dem Feld `baseDN`, über das der Distinguished Name des Zweiges angegeben wird, in dem gesucht werden soll. Außerdem können Einträge durch reguläre Ausdrücke nach Attributen gefiltert werden und die Antwort kann auf bestimmte Attribute eines Objekts reduziert werden. [52]

Remote Authentication Dial-In User Service Ähnlich wie **LDAP** wird auch **RADIUS** zur Authentifizierung und Autorisierung verwendet. Es nutzt **UDP** und ist standardmäßig über den Port 1812 erreichbar. Es ist schneller als **LDAP**, hat jedoch eine geringere Verfügbarkeit, weil es nicht für verteilte Systeme geeignet ist.

Zu Beginn einer Session sendet ein Network Access Server eine `Access-Request` an den **RADIUS**-Server. Diese enthält typischerweise einen Benutzernamen und den **MD5**-Hashwert des zugehörigen Passworts, kann aber auch noch mehr Informationen über den sich einwählenden Benutzer enthalten. Der **RADIUS**-Server kann darauf mit den Paketen `Access-Accept`, `Access-Reject` und `Access-Challenge` antworten. `Access-Reject` bedeutet, dass dem Benutzer der Zugriff verweigert wird. In einem `Access-Challenge`-Paket fordert der **RADIUS**-Server zusätzliche Informationen an. Dieses Paket kann beispielsweise bei einer Zwei-Faktor-Authentifizierung zum Einsatz kommen. `Access-Accept` erlaubt dem Benutzer den Zugriff auf die angefragte Ressource. Der **RADIUS**-Server wird regelmäßig überprüfen, ob der Benutzer befugt ist, bestimmte Dienste zu nutzen. [47]

Nachdem der Zugriff erlaubt wurde, informiert der Network Access Server den **RADIUS**-Server durch eine `Accounting-Request` über den Beginn der Sitzung. Dieser antwortet mit einer `Accounting-Response`, um zu bestätigen, dass er die Nachricht erhalten hat. In weiteren `Accounting-Requests` informiert der Network Access Server den **RADIUS**-Server über die aktuelle Sitzungsdauer und die aktuelle Datennutzung. Um die Sitzung zu beenden sendet der Network Access Server eine letzte `Accounting-Request`. Diese Funktionen bietet **LDAP** nicht. [46]

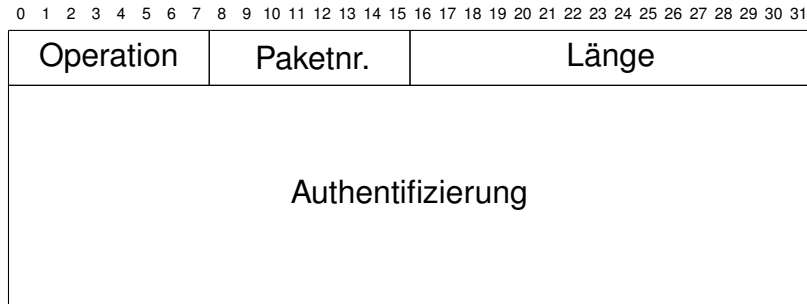


Abbildung 4.8: RADIUS-Header

Syslog Syslog ist für die Übermittlung von Systemmeldungen zuständig. Es benutzt dafür das Transportprotokoll **UDP** und den Port 514. Ein Syslog-Paket besteht aus den Feldern PRI, VERSION, TIMESTAMP, HOSTNAME, APP-NAME, PROCID, MSGID, STRUCTURED-DATA und MSG. Das Feld PRI gibt dabei die Priorität der Meldung und das System an, das sie erzeugt hat. PROCID gibt die Prozedur an, die die Meldung erzeugt hat, während MSGID die ID der Meldung selbst angibt. STRUCTURED-DATA ist ein Objekt, das zusätzliche Informationen über die Meldung enthält. MSG enthält die Meldung selbst. Diese darf nicht größer als ein Kibibyte sein. [20]

Simple Network Management Protocol **SNMP** ist ein Protokoll zur Überwachung und Fernsteuerung von Netzwerkkomponenten. Es benutzt **UDP** Port 161 und Port 162 für TRAP-Pakete. **SNMP** teilt Kommunikationspartner in Agenten und Manager ein. Ein Agent ist dabei ein Programm, das auf einer Netzwerkkomponente läuft, deren Zustand erfassen und Einstellungen vornehmen kann. Der Manager ist eine zentrale Station, die den Agenten Anweisungen gibt.

SNMP nutzt verschiedene Datenpakete zur Kommunikation. Die Pakete GET-REQUEST, GETNEXT-REQUEST und GETBULK können vom Manager als Anfragen nach Management-Datensätzen versendet werden. GETBULK ist dabei eine Anfrage nach einer bestimmten Anzahl an Datensätzen und hat den gleichen Effekt wie mehrere GETNEXT-REQUESTs. Agenten antworten auf diese Anfragen mit einer GET-RESPONSE. Sie können aber auch unaufgefordert Management-Datensätze als TRAP-Pakete senden. Über eine SET-REQUEST kann der Manager Datensätze verändern und dadurch Einstellungen an den Netzkomponenten vornehmen.

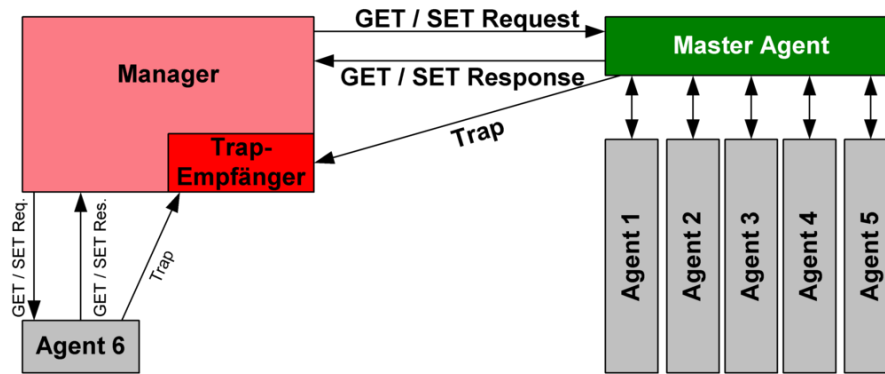


Abbildung 4.9: Das Prinzip der **SNMP**-Kommunikation [4]

Ein **SNMP**-Paket besteht aus dem **SNMP**-Header und mehreren sogenannten Protocol Data Units (**PDU**s), die jeweils einen Header und einen Body besitzen. Der **SNMP**-Header enthält die Versionsnummer und einen sogenannten Community-Namen, über den die Zugriffsrechte definiert sind. Der Community-Name `public` steht dabei für Lesezugriff während `private` für Schreib- und Lesezugriff steht. Der **PDU**-Header enthält den Pakettyp, die ID der Anfrage, einen Fehlerstatus und den Index des Datensatzes bei dem der Fehler auftrat. Standardmäßig haben die letzten beiden Fehler den Wert Null, was bedeutet, dass kein Fehler auftrat. Der **PDU**-Body besteht aus variablen Verbindungen einer Netzwerkkomponente, die durch eine ID und einen Wert dargestellt werden.

Bei TRAP-Paketen ist der **PDU**-Header anders aufgebaut. Er besteht aus dem Pakettyp TRAP, der ID und der **IP**-Adresse der Netzwerkkomponente, die das Paket erzeugt hat, einer allgemeinen und einer spezifischen TrapID und einem Zeitstempel. Die TrapID gibt den Grund für das Versenden des TRAP-Pakets an. Die Bedeutung allgemeiner TrapIDs ist dabei für jedes System gleich definiert, während die spezifischen TrapIDs konfigurierbar sind. [6] [5]

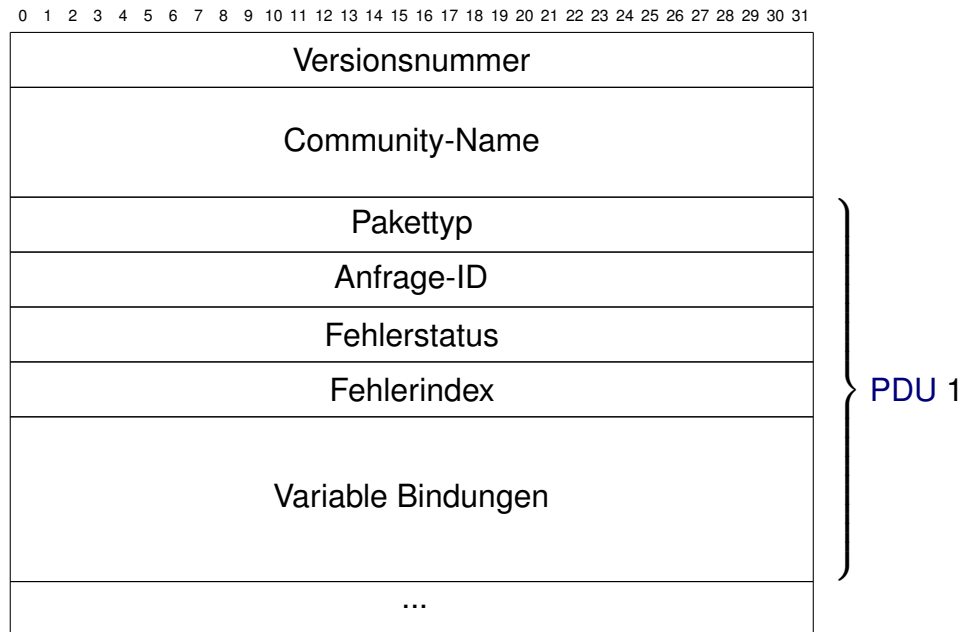


Abbildung 4.10: SNMP-Paket

Dynamic Host Configuration Protocol Neben **SNMP** wird auch **DHCP** für Einstellungen an Netzwerkkomponenten verwendet. Es benutzt ebenfalls **UDP** als Transportprotokoll, belegt aber insgesamt vier Ports: Port 67 für **IPv4** Server, Port 68 für **IPv4** Clients, Port 547 für **IPv6** Server und Port 546 für **IPv6** Clients. Über **DHCP** können dem Client seine **IP**-Adresse und Netzmaske, sein Standard-Gateway, sein **DNS**-Server und lokaler **DNS**-Baum, sein Zeitserver und seine Proxy-Einstellungen zugewiesen werden.

Wenn ein Client noch keine **IP**-Adresse hat, versendet er ein **DISCOVER**-Paket an alle **DHCP**-Server im lokalen Netz. Diese antworten mit einem **OFFER**-Paket, das freie **IP**-Adressen beinhaltet. Der Client kann dann über **REQUEST** eine freie **IP**-Adresse für sich beantragen und über **INFORM** nach weiteren Einstellungen fragen. **REQUEST** kann vom Client aber auch benutzt werden, um dem Server mitzuteilen, welche **IP**-Adresse er aktuell nutzt. Der Server kann mit **ACK** die **IP**-Adresse des Clients bestätigen und weitere Einstellungen übertragen. Er kann sie aber auch über **NAK** ablehnen. Über **DECLINE** kann ein Client dem Server mitteilen, dass eine Netzwerkadresse bereits benutzt wird. Über **RELEASE** kann er seine **IP**-Adresse freigeben.

Über **SNMP** können deutlich mehr Konfigurationen an Netzwerkkomponenten vorgenommen werden. **DHCP** kann aber im Gegensatz zu **SNMP** **IP**-Adressen an Geräte vergeben, die noch keine haben. Die Adressierung des Clients läuft dabei über dessen

Hardwareadresse. [14] [15]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Operation								Netztyp								Adresslänge								Relay-Agents							
Verbindungsnummer																															
Sekunden seit Start																Flags															
Client-IP-Adresse																															
eigene IP-Adresse																															
Server-IP-Adresse																															
Relay-Agent-IP-Adresse																															
Client-Hardwareadresse																															
DHCP-Servername																															

Abbildung 4.11: DHCP-Header

Network Time Protocol **NTP** ist ein Protokoll für die externe Zeitsynchronisierung. Es nutzt den Port 123 und funktioniert sowohl mit **UDP** als auch mit **TCP**. **NTP** nutzt sogenannte Zeitserver, die sich gegenseitig synchronisieren. Diese teilt es in sogenannte Strata ein, wobei Stratum 1 die Menge aller Zeitserver bildet, die sich direkt mit einer Atom- oder Funkuhr (Stratum 0) synchronisieren. Stratum-2-Server sind Zeitserver, die sich mit Stratum-1-Servern synchronisieren. Die ungenaueren Zeitserver sind Stratum-15-Server. Wenn ein Client seine Zeit synchronisieren möchte, wählt er dafür das niedrigste erreichbare Stratum. Weil jeder Zeitserver auch ein Client ist, wird so sichergestellt, dass die Zeit so genau wie möglich ist.

Der Client sendet seinen Zeitstempel t_1 in Sekunden seit dem Beginn einer Ära als Festkommazahl mit 32 Vorkommabit und 32 Nachkommabit an den Zeitserver. Ära 0 startete am 01.01.1900 um 00:00:00 Uhr. Dieser erstellt bei Eintreffen der Nachricht einen Zeitstempel t_2 und vor dem Absenden seiner Antwort einen Zeitstempel t_3 . Bei Erhalt der Antwort erstellt der Client einen Zeitstempel t_4 . Aus diesen vier Zeitstempeln berechnet der Client die Abweichung $\theta = \frac{(t_2 - t_1) + (t_3 - t_4)}{2}$ und die Paketumlaufzeit $\delta = (t_4 - t_1) - (t_3 - t_2)$. **NTP** nutzt numerische Verfahren, um die Uhrzeit und -frequenz des Clients graduell anzupassen, wodurch starke Zeitschwankungen verhindert werden. Ziel ist, die Abweichung θ zu minimieren.

Ein **NTP**-Paket besteht aus dem Stratum, den vier Zeitstempeln, der Referenz-ID und verschiedenen Genauigkeitsindikatoren. Optional kann auch ein Nachrichten-Authentifizierungscode verwendet werden. [28]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
LI	VN	Mod	Stratum												Abstimmung										Genauigkeit						
Verzögerung																															
Streuung																															
Referenz-ID																															
Referenzzeitstempel																															
Ursprünglicher Zeitstempel t_1																															
Zeitstempel bei Empfang t_2																															
Zeitstempel vor Absenden t_3																															
Zeitstempel bei Antwort t_4																															

Abbildung 4.12: **NTP**-Paket ohne optionale Felder

Distributed Network Protocol **DNP3** ist ein Protokoll, das speziell für die Fernüberwachung und -steuerung von Fernbedienungsterminals entwickelt wurde [23, S. iv–ix]. Es benutzt standardmäßig **TCP**-Port 20000, kann aber auch über **UDP** oder sein eigenes Transportprotokoll verwendet werden [23, S. 361–365].

DNP3-Pakete bestehen aus einem Paketheader und mehreren **DNP3**-Objekten, die jeweils einen Objektheader besitzen. Anfragen an Fernbedienungsterminals werden von einer Leitstelle gestellt. Ein Anfrageheader besteht aus den Feldern `Application Control` und `Function Code`. Ersteres bietet Informationen über den Platz des Pakets innerhalb einer Sequenz und gibt an, ob eine Empfangsbestätigung erwünscht ist. Zweiteres gibt den Zweck der Nachricht an. Es gibt insgesamt dreiunddreißig verschiedene Anfragen, vier verschiedene Antworten und die Empfangsbestätigung. In einem Antwortheader ist zusätzlich das Feld `Internal Indications` enthalten, das Systemmeldungen des Fernbedienungsterminals beinhaltet.

Ein Objektheader besteht aus den Feldern `Object Type`, `Qualifier` und `Range`. `Object Type` spezifiziert den verwendeten Datentyp und das Datenformat. Die anderen beiden Felder geben die Indexnummer des Objekts und seine Größe an. [23, S. 13–34]

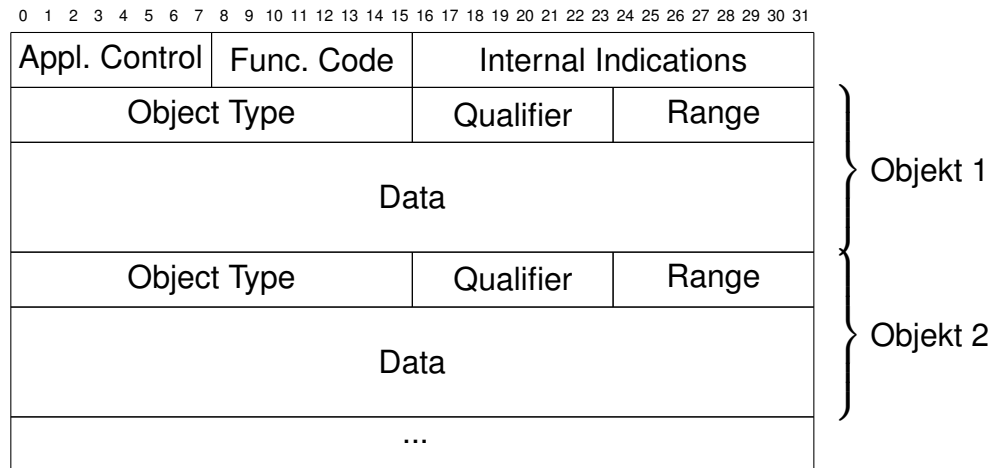


Abbildung 4.13: DNP3-Antwortpaket

Modbus Das Modbus-Protokoll wird verwendet, um Daten aus Registern zu lesen und in Register zu schreiben. Es kann mit [TCP](#) über den Port 502 verwendet werden, ist jedoch nicht grundsätzlich an den Internetprotokollstack gebunden. Der Header eines Modbus-Pakets über [TCP](#) besteht aus den Feldern `Transaction Identifier`, `Protocol Identifier`, `Length`, `Unit Identifier` und `Function Code`.

Der `Transaction Identifier` wird benutzt, damit Antworten Anfragen zugeordnet werden können. Der `Protocol Identifier` hat den Wert 0 und weist auf die Verwendung von [TCP](#) hin. Das `Length`-Feld gibt die Anzahl der nachfolgenden Bytes an. Der `Unit Identifier` weist auf den angesprochenen *Slave* hin, meist eine speicherprogrammierbare Steuerung, und dient dem Modbus internen Routing. Der `Function Code`, gibt an, ob Daten nur gelesen oder auch geschrieben werden sollen und ob es sich um analoge oder digitale Eingänge handelt. [31] [32]

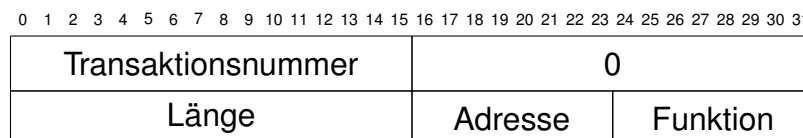


Abbildung 4.14: Modbus-Header

Die beschriebenen Protokolle sind nicht eindeutig durch ihre Ports identifizierbar, weil Ports mehrfach vergeben werden können und Protokolle gekapselt werden können.

4.2 Firewall

Eine Firewall ist eine Software, welche den Schutz eines Netzwerkes oder eines Computers vor einem unbefugten Zugriff gewährleistet. Die entsprechenden Richtlinien werden in einem Firewall-Regelwerk definiert. Die ersten Firewalls wurden Ende der Achtzigerjahre entwickelt, als Hacker im Internet aktiv wurden. In der Zwischenzeit wurden unterschiedliche Arten von Firewalls entwickelt, die jeweils vor spezifischen Angriffen schützen. Das Unified Threat Management sowie die Next-Generation-Firewall bieten einen umfassenden Schutz vor einer Vielzahl von Angriffen. [34]

4.2.1 Paketfilter

Der Paketfilter stellt die älteste Form einer Firewall dar. Seine Entwicklung erfolgte im Jahr 1988 und seine Funktionsweise basiert auf der Vermittlungsschicht des OSI-Referenzmodells. Der Paketfilter erlaubt lediglich den Austausch von Paketen mit bestimmten Quell- und Ziel-IP-Adressen sowie bestimmten Quell- und Ziel-Ports. Des Weiteren ist eine Filterung nach verwendeten Protokollen möglich.

Die Überprüfung der Identität des Absenders ist nicht Bestandteil dieser Firewall. Daher ist sie anfällig für Angriffe, bei denen gefälschte IP-Adressen verwendet werden. Des Weiteren erfolgt keine Überprüfung der Nutzdaten. Dadurch besteht die Möglichkeit, dass Spam oder Schadsoftware übertragen wird. [27] [7, S. 176–178]

Eine weiterentwickelte Variante ist der zustandsorientierte Paketfilter. Er ist auf der Transportschicht angesiedelt und überwacht aktive Verbindungen in beide Richtungen. Der Zustand dieser Verbindungen wird in einem Cache gespeichert. Im Falle der Initiierung einer neuen TCP-Verbindung erstellt die Firewall ein entsprechendes Profil zu dieser Verbindung. Das Profil besteht aus den IP-Adressen und Ports im Header des SYN-Pakets. Die Verbindung wird folglich wieder gelöscht, sobald das FIN-Paket eintrifft oder wenn sie zu lange inaktiv ist.

Eine Verbindung wird nach den Regeln für die Paketfilterung etabliert. [TCP](#)-Pakete ohne gesetzte SYN-Flag werden nicht blockiert, wenn sie einer bestehenden Verbindung zugeordnet werden können.

Der zustandsorientierte Paketfilter weist somit eine höhere Geschwindigkeit als der normale Paketfilter auf. Allerdings ist auch diese Lösung nicht frei von den Sicherheitsrisiken des zustandslosen Paketfilters. Des Weiteren birgt sie das Risiko eines Pufferüberlaufs bei einer Denial-of-Service-Attacke. Ein derartiger Fehler kann zum Absturz der Firewall führen. [\[27\]](#) [\[7, S. 188\]](#)

4.2.2 Application-Gateway

Im Gegensatz zum Paketfilter reicht das Application-Gateway eintreffende Pakete nicht bloß weiter, sondern analysiert deren Nutzlast und kann diese auch manipulieren. In diesem Kontext werden zwei Verbindungen aufgebaut, wobei eine zum Server und eine zum Client führt. Dies hat zur Konsequenz, dass dem Client der direkte Zugriff auf den Server verwehrt wird. Der Zugriff auf die Serverdaten ist beschränkt und erfolgt ausschließlich über einen von der Firewall bereitgestellten Proxy.

Aus diesem Grund wird das Application-Gateway auch als Proxy-Firewall bezeichnet. Da die verwendeten Anwendungsprotokolle bekannt sein müssen, ist die Verwendung mehrerer Proxys für verschiedene Protokolle erforderlich. Des Weiteren besteht die Möglichkeit, für ein Protokoll mehrere Proxys zu verwenden. Dies kann beispielsweise erforderlich sein, um [HTTP](#)-Anfragen an verschiedene Websites unterschiedlich zu behandeln.

Das Application-Gateway weist im Vergleich zum Paketfilter eine deutlich höhere Sicherheit auf, ist jedoch aufgrund des Proxys auch deutlich langsamer. Im Rahmen einer Anfrage eines Clients an einen Server ist das Application-Gateway dazu verpflichtet, Pakete beidseitig zu analysieren, zu manipulieren und bereitzustellen. [\[27\]](#) [\[7, S. 185–186\]](#)

4.2.3 Circuit-Gateway

Das Circuit-Gateway ist ein auf der Sitzungsschicht arbeitendes System, welches wie das Application-Gateway als eigenständiger Verbindungsteilnehmer zwischen Server und Client fungiert. Analog zum Application-Gateway ist auch das Circuit-Gateway für den Client ein Server und für den Server ein Client. Dem Client wird eine direkte Verbindung zum Server vorgetäuscht. Wie der zustandsorientierte Paketfilter ist auch das Circuit-Gateway für die Verwaltung von Verbindungen zuständig. Die Untersuchung geht jedoch über diese hinaus und umfasst die Prüfung der Legitimität der Verbindungen sowie die Evaluierung der Korrektheit der Antworten auf eine Anfrage.

Circuit-Gateways weisen im Vergleich zu zustandsorientierten Paketfiltern eine höhere Sicherheit auf, da sie eine Authentifizierung des Clients verlangen und unerwartete Antworten abfangen. Sie weisen eine höhere Geschwindigkeit als Application-Gateways auf und können, wie diese, die Kommunikation protokollieren. Allerdings weisen sie auch eine geringere Sicherheit als Application-Gateways auf, weil sie sich nicht für den Inhalt eines Pakets interessieren. [27] [7, S. 186–188]

4.2.4 Next Generation Firewall

Die Next-Generation-Firewall stellt ein Konzept dar, welches die Stärken der bislang vorgestellten Firewall-Lösungen in einer einzigen Firewall zu vereinen versucht. Das Ziel einer solchen Firewall besteht in der Erkennung von Anwendungen, wobei das genutzte Protokoll sowie der Port unerheblich sind. Des Weiteren ist das Ziel, Benutzer zu identifizieren, wobei die Identität des Nutzers bei der Analyse des Pakets Berücksichtigung finden soll. Außerdem soll die tatsächliche Absicht, die sich hinter der Nutzlast eines Pakets verbirgt, ermittelt werden.

Statt einer Kombination der bisherigen Firewalls, wie sie beim Unified Threat Management erfolgt, wird eine effizientere Lösung entwickelt. Im Gegensatz zu Paketfiltern und Circuit-Gateways, welche lediglich den Header eines Pakets betrachten, analysiert die Next-Generation-Firewall das gesamte Paket. Traditionelle Firewalls konnten durch Port-Hopping, TLS-Verschlüsselung und Tunnelprotokolle umgangen werden. Die Next-Generation-Firewall erhebt den Anspruch, auch gegen derartige Angriffe gesichert zu sein.

In den meisten Fällen lässt sich eine spezifische Charakteristik feststellen, anhand derer sich ohne Kenntnisse des Protokolls oder des Ports die verwendete Anwendung ableiten lässt. Die Next-Generation-Firewall blockiert alle Pakete, deren Nutzlast nicht zur Charakteristik des vorgegebenen Protokolls passt und deren Portnummer und Protokoll nicht übereinstimmen. Dies verhindert die Nutzung von Tunnelprotokollen und Port-Hopping durch Angreifer. Die Blockierung bekannter Angriffe, die Filterung von URLs und Dateien sowie die Erkennung verschleierte Anwendungen durch Heuristik stellen die Grundlagen für die Sicherheit auf Anwendungsebene dar.

Die Next-Generation Firewall erweitert die Funktionalität des zustandsorientierten Paketfilters, indem sie nicht nur Verbindungen überwacht, sondern auch eine Benutzeridentifikation ermöglicht. Wie der Paketfilter erhält sie IP-Adressen und Portnummern aus dem Header des Pakets. Im Gegensatz zum Paketfilter sind diese Informationen für sie jedoch nicht aussagekräftig genug. Zur Ermittlung der hinter den jeweiligen IP-Adressen stehenden Nutzer kommuniziert sie mit LDAP-Verzeichnissen. Des Weiteren ist sie in der Lage, Rollen- und Gruppenzuordnungen anzufordern, um auf dieser Grundlage den Kommunikationsfluss zu filtern.

Die Next-Generation-Firewall weist eine höhere Geschwindigkeit als das Unified Threat Management auf. Dies ist darauf zurückzuführen, dass sie mehrere Prozesse gleichzeitig ausführt und keine Prozesse mehrfach durchläuft. Im Rahmen des Unified Threat Management werden Dateien zunächst vollständig geladen, bevor sie einem Scan auf Schadsoftware unterzogen werden. Erst wenn die vollständige Datei gescannt wurde, erfolgt ihre Weiterleitung, sofern keine Gefahr besteht. Die Next-Generation-Firewall initiiert den Scan beim Eintreffen des ersten Pakets und sendet dieses umgehend weiter, sobald es gescannt wurde. Das Unified Threat Management verwendet mehrere Scanner, die jeweils auf einen Typ von Schadsoftware spezialisiert sind. Folglich müssen Dateien mehrfach gescannt werden. Die Next-Generation-Firewall verwendet einen Dateiscanner, der jede bekannte Art von Schadsoftware erkennen kann. Infolgedessen ist das Scannen von Dateien nach Schadsoftware in der Next-Generation-Firewall deutlich schneller als im Unified Threat Management.

Das Unified Threat Management besteht aus mehreren seriell miteinander verbundenen Firewalls. Jede der Firewalls muss die eintreffenden Pakete dekodieren, analysieren und filtern. Dies resultiert in einer hohen Redundanz an Prozessen. Des Weiteren beginnt jede Firewall erst mit ihrer Arbeit, wenn die Firewall davor ihre Funktion abgeschlossen hat. Der Ansatz der Next-Generation-Firewall besteht in der parallelen

Ausführung einer möglichst großen Anzahl von Untersuchungen. Im Falle der Feststellung eines Verstoßes werden sämtliche andere Untersuchungen abgebrochen und die Bearbeitung des nächsten Pakets initiiert. Beim Unified Threat Management kann es ebenfalls dazu kommen, dass ein Verstoß erst in der letzten Firewall identifiziert wird. Somit wäre die gesamte Bearbeitungszeit der anderen Firewalls vermeidbar gewesen. [27]

4.3 Netzwerksicherheitszonen

Um die Netzwerksicherheit zu erhöhen, wird ein Netzwerk in Sicherheitszonen und -kanäle aufgeteilt, die unterschiedliche Sicherungsstufen aufweisen. Die Zonen umfassen Netzwerkkomponenten, die die gleichen Sicherheitsanforderungen aufweisen. Kanäle stellen spezielle Zonen dar, welche mehrere Zonen miteinander verbinden. Die Norm IEC 62443 empfiehlt die Einteilung von Zonen und Kanälen in Sicherheitsstufen. Die Stufe 0 definiert keine Sicherheitsanforderungen. Die erste Stufe gewährleistet den Schutz vor ungewollten Sicherheitsverstößen. Die Stufen 2 bis 4 schützen vor Angreifern mit steigenden Fähigkeiten, Ressourcen und zunehmender Motivation.

Im Rahmen eines iterativen Prozesses sind Maßnahmen zu ergreifen, die dazu dienen, das Sicherheitsrisiko für jede Zone zu reduzieren und sie somit an die jeweilige Sicherheitsstufe anzupassen. Der Standard definiert dafür Anforderungen, die je nach Sicherheitsstufe in unterschiedlicher Ausprägung für eine Zone umgesetzt werden sollen: Benutzer müssen identifiziert und authentifiziert werden. Die Benutzung von sowie der Zugriff auf Ressourcen sind zu überwachen. Die Integrität des Netzwerks muss gewahrt werden. Die Vertraulichkeit und Verfügbarkeit von Daten muss gewährleistet werden. Der Datenverkehr muss eingeschränkt werden. Es muss zeitnah auf Ereignisse reagiert werden. [43]

4.4 Zero Trust

Das Sicherheitskonzept Zero Trust basiert auf der Prämisse, dass kein Benutzer, Gerät oder Netzwerk vertrauenswürdig ist. Die Identität aller Clients wird fortlaufend geprüft und es wird sichergestellt, dass sie nur den minimal notwendigen Zugang zu Ressourcen erhalten. Netzsegmente dürfen nur so groß wie nötig sein und der Netzwerkverkehr muss kontinuierlich überwacht werden. Damit das Zero-Trust-Modell

funktioniert, müssen Sicherheitsrichtlinien in regelmäßigen Abständen an neue Anforderungen angepasst werden. [48, S. 4–8]

Die wesentlichen Komponenten einer Zero-Trust-Architektur sind eine Policy Engine, ein Policy Administrator und Policy Enforcement Points. Die Policy Engine fungiert als Sicherheitskomponente, welche basierend auf vorgegebenen Kriterien darüber entscheidet, ob einem Client Zugriff auf eine Ressource gewährt wird. Zu diesem Zweck bedient sie sich eines sogenannten Vertrauensalgorithmus, der mit Daten wie der Zugriffsanfrage, Informationen über den Client, Informationen über die Ressource, Richtlinien für den Zugriff auf die Ressource sowie externen Risikobewertungen gespeist wird. [48, S. 17–19]

Der Policy Administrator führt die Entscheidungen der Policy Engine aus. Zu seinen Aufgaben gehört der Aufbau und die Schließung des Kommunikationspfads zwischen dem Client und der Ressource mittels entsprechender Befehle an die betreffenden Policy Enforcement Points. Zudem ist er für die sitzungsweise Authentifizierung und Anmeldung des Clients für den Zugriff auf die Ressource verantwortlich. Gemeinsam mit der Policy Engine wird er auch als Policy Decision Point bezeichnet.

Die Policy Enforcement Points können Komponenten wie eine Firewall oder einen Agenten auf dem Endgerät des Clients darstellen. Sie kommunizieren mit dem Policy Administrator um Zugriffsanfragen zu stellen und ihre Richtlinien zu aktualisieren. Policy Enforcement Points gehören somit zur Datenebene.

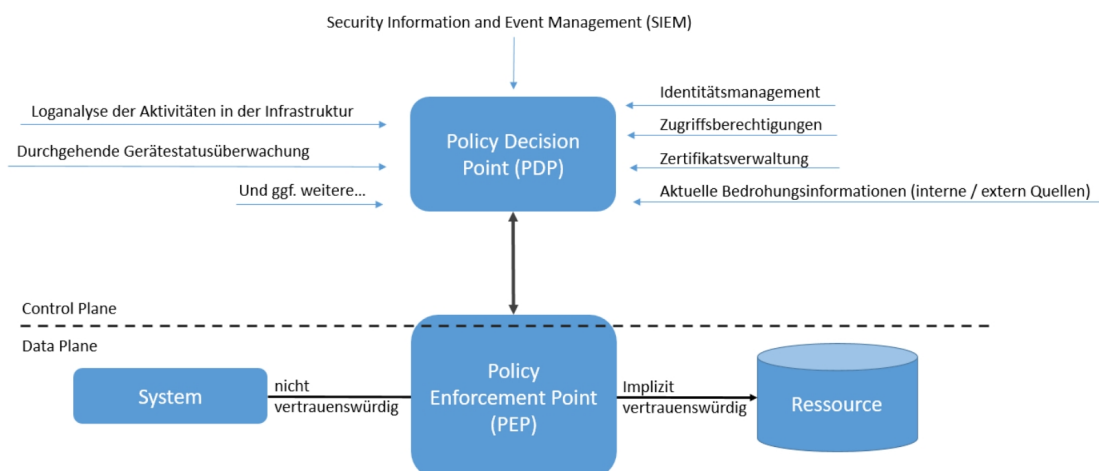


Abbildung 4.15: Logische Zero Trust Architektur [39]

Neben den genannten Hauptkomponenten können noch weitere Komponenten in einer Zero-Trust-Architektur eine Rolle spielen. Beispielsweise wird die Implementierung eines Continuous Diagnostics and Mitigation (CDM) Systems empfohlen. Dieses überprüft alle Ressourcen auf Schwachstellen, Integrität und Updates. Außerdem kann eine Komponente, die Angriffe und Schwachstellen protokolliert, von der Policy Engine als weitere Entscheidungsgrundlage für den Vertrauensalgorithmus benutzt werden. Über eine Public Key Infrastructure (PKI) kann der Policy Administrator die Authentifizierung auslagern. Des Weiteren müssen Sicherheitsrichtlinien und Informationen zu Clients und Ressourcen durch Komponenten bereitgestellt werden. [48, S. 9–11]

Weil es sehr aufwendig sein kann, ein ganzes Netzwerk nach der Zero-Trust-Architektur aufzubauen, wird diese häufig nur auf die kritischsten Sicherheitszonen angewandt. So kann ein Kompromiss zwischen Sicherheit und Rentabilität gefunden werden.

5 Anforderungen

Damit ein Gateway-System nach der Zero-Trust-Architektur arbeiten kann, ist es nötig, Anforderungen an dieses Gateway-System zu stellen. Die meisten Anforderungen resultieren aus der Definition der Zero-Trust-Architektur. Die Protokolle, mit denen das Gateway umgehen muss, sind ebenfalls von entscheidender Bedeutung.

5.1 Allgemein

Das Gateway besteht aus einer Firewall und einem Router oder Switch und kontrolliert in seiner Funktion als Policy Enforcement Point die Kommunikation zwischen einer inneren Sicherheitszone und einer äußeren Sicherheitszone, indem es verschlüsselte Verbindungen öffnet, überwacht und schließt. Es darf je Sitzung nur den minimal erforderlichen Zugriff auf die innere Sicherheitszone erlauben und muss regelmäßig die Identität und die Berechtigungen der Clients aus der äußeren Sicherheitszone überprüfen. Die Netzarchitektur ist so zu gestalten, dass kein direkter Zugriff von außen auf die innere Sicherheitszone möglich ist. Die logischen Komponenten einer Zero-Trust-Architektur müssen nicht technisch voneinander getrennt sein. In diesem Fall können Policy Engine und Policy Administrator zusammen als Verwaltungskomponente des Gateways dienen.

Die Integrität eines Clients kann nicht überprüft werden, wenn ein Gateway ohne weitere Komponenten verwendet wird. Um die Sicherheitsmaßnahmen zu verschärfen, besteht die Möglichkeit, den Zugriff auf Ressourcen lediglich isolierten Anwendungen des Clients zu gestatten. Obwohl die Implementierung eines Software-Agenten zur Überprüfung des Clients auf Schadsoftware in Betracht gezogen werden kann, bringt diese Lösung signifikante Nachteile mit sich. Die gesteigerte Verantwortung für die Verwaltung und Aktualisierung des Agenten stellt einen Mehraufwand dar. Darüber hinaus erhöht sich der Aufwand für die Erteilung von Zugriffsrechten an neue Clients, da diese zunächst den Software-Agenten installieren müssen. [48, S. 4–17]

Das Gateway soll als **NTP**-, **DNS**- und **DHCP**-Server fungieren, damit alle Netzwerkkomponenten in der inneren Sicherheitszone das Gateway als vertrauenswürdige Quelle für die Uhrzeit, Domains und **IP**-Adressen verwenden können. Außerdem soll es nur Datenpakete mit bestimmten Inhalten, also bestimmten Feldwerten, zulassen.

5.2 Anwendungsszenarien

Da sich die genauen Anforderungen an das Gateway-System je nach Anwendungsszenario unterscheiden, werden in den folgenden Abschnitten vier verschiedene Anwendungsszenarien aus der Industrie und der kritischen Infrastruktur vorgestellt und die jeweiligen Anforderungen evaluiert. Neben den vorgestellten Szenarien gibt es noch viele weitere Szenarien, für die ein Gateway-System nach dem Zero-Trust-Modell verwendet werden könnte.

5.2.1 Prozessleitsysteme

Prozessleitsysteme überwachen und kontrollieren lokale Produktionsanlagen. Sie bilden eine übergeordnete Struktur zu einzelnen Steuerungen, welche Sensorwerte mit Sollwerten vergleichen und elektrisch, mechanisch, hydraulisch oder pneumatisch betriebene Aktoren steuern. Ein zentraler Server kann Sensorwerte von den einzelnen Steuerungen anfordern und Sollwerte setzen. Zudem stellt er eine Benutzerschnittstelle bereit und kann Teil eines übergeordneten Netzes sein.

Prozessleitsysteme müssen eine hohe Verfügbarkeit aufweisen, um einen reibungslosen und effizienten Betrieb zu gewährleisten. Der Ausfall eines Prozesses kann andere, davon abhängige Prozesse beeinträchtigen. Des Weiteren können Produktionsarbeiter gefährdet und Produkte beschädigt werden. Um Ausfälle zu vermeiden, sind Prozessleitsysteme in der Regel redundant aufgebaut. Es werden häufig mehrere Server verwendet, die sich gegenseitig aktualisieren. Außerdem werden Benutzerschnittstellen auch für einzelne Steuerungen bereitgestellt.

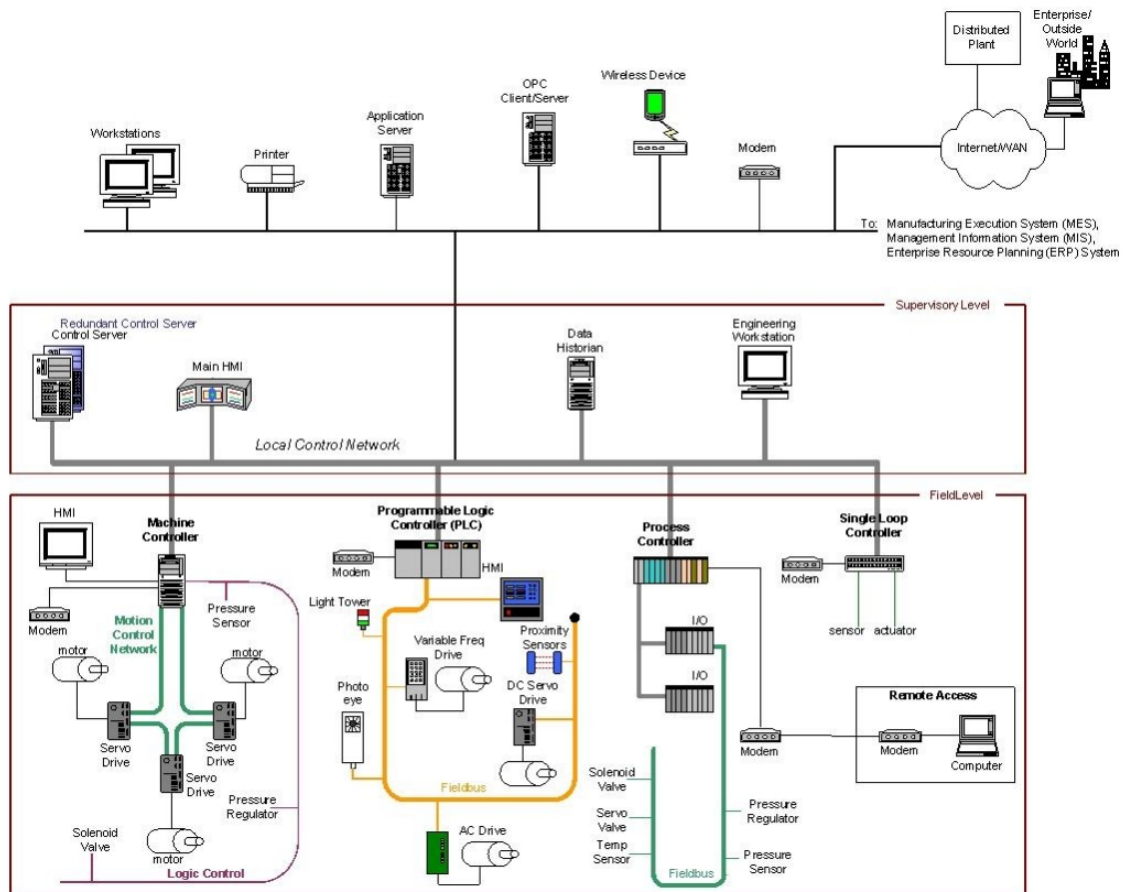


Abbildung 5.1: Beispielhaftes Prozessleitsystem [55, S. 28]

Meistens werden in Prozessleitsystemen proprietäre Kommunikationsprotokolle verwendet, die keine Sicherheitsmechanismen wie Authentifizierung oder Verschlüsselung enthalten. Außerdem muss eine Aktualisierung der Software gründlich geplant werden, weil es dabei zu Ausfällen der Produktionsanlage kommen kann. Somit haben potenzielle Angreifer ohne weitere Sicherheitsmaßnahmen nur wenige Hürden zu überwinden. [55, S. 26–33]

Um den potenziellen Schaden eines Angriffes auf ein Prozessleitsystem einzugrenzen, besteht die Möglichkeit, das Prozessleitsystem in mehrere kleine Einheiten aufzuteilen, die nicht unmittelbar voneinander abhängig sind und daher kaum miteinander kommunizieren. Prozessleitsysteme können sowohl physisch als auch logisch in Form von virtuellen Netzen voneinander getrennt werden. Da der Netzverkehr zwischen einzelnen Prozessleitsystemen statisch ist, können Firewall-Regeln ebenfalls statisch definiert werden. Dies bedeutet, dass festgelegt wird, welche Geräte welche Daten auf welche Art miteinander austauschen dürfen. Jede Kommunikation, die nicht explizit erlaubt ist, wird blockiert.

Der Zugriff aus der äußeren Sicherheitszone sollte ausschließlich über verschlüsselte Kommunikationsprotokolle erfolgen und stets aufgezeichnet werden. Des Weiteren wird die Einrichtung einer demilitarisierten Zone zwischen dem Prozessleitsystem und der äußeren Sicherheitszone empfohlen. Diese Zone kann für eine Prozessdatenbank genutzt werden. Remote Procedure Calls sind lediglich zwischen dem Prozessleitsystem und der demilitarisierten Zone zulässig. Die Nutzung von SMTP durch das Prozessleitsystem ist auf den Versand von Systembenachrichtigungen zu beschränken. [55, S. 50–74]

5.2.2 Intelligente Stromnetze

Traditionelle Stromnetze verteilen elektrische Energie von einem zentralen Kraftwerk an Gewerbe und Haushalte. Zu diesem Zweck wird zunächst die Spannung erhöht und anschließend stufenweise wieder gesenkt. Diese Architektur weist jedoch eine Reihe von Nachteilen auf, da sie nicht für dezentrale Energiequellen wie beispielsweise Windkraftanlagen geeignet ist, nur langsam auf Veränderungen des Energiebedarfs reagieren kann und ein signifikanter Anteil der elektrischen Energie auf dem Transportweg verloren geht. Intelligente Stromnetze versuchen Probleme, die bei traditionellen Stromnetzen auftreten, über Kommunikations- und Informationstechnik zu lösen. Die dafür benötigte Infrastruktur erstreckt sich über das gesamte Stromnetz inklusive der Erzeuger und Verbraucher.

Ein intelligentes Stromnetz besteht in der Regel aus intelligenten Messsystemen, automatisierten Umspannwerken und einem SCADA-System. Intelligente Messsysteme bestehen aus digitalen Stromzählern und einem zentralen Server, der die Messwerte der Stromzähler empfängt, aufzeichnet, analysiert und weiterreicht. Die Stromzähler kommunizieren mit dem Server über drahtlose Protokolle wie ZigBee. Umspannwerke regulieren Spannungsschwankungen und verteilen elektrische Energie an Verbraucher. Sie bestehen meistens aus einem Fernbedienungsterminal, einer Benutzerschnittstelle, einer GPS-Antenne zur Zeitsynchronisierung und mehreren elektronischen Geräten, wie Schaltern, Transformatoren und Kondensatoren. Ein SCADA-System überwacht und kontrolliert das gesamte Stromnetz, in dem es einzelne intelligente Messsysteme und Umspannwerke in einem Netz verbindet. Es ist in seiner Funktion einem Prozessleitsystem ähnlich, arbeitet aber mit deutlich größeren und weiter verteilten Netzen.

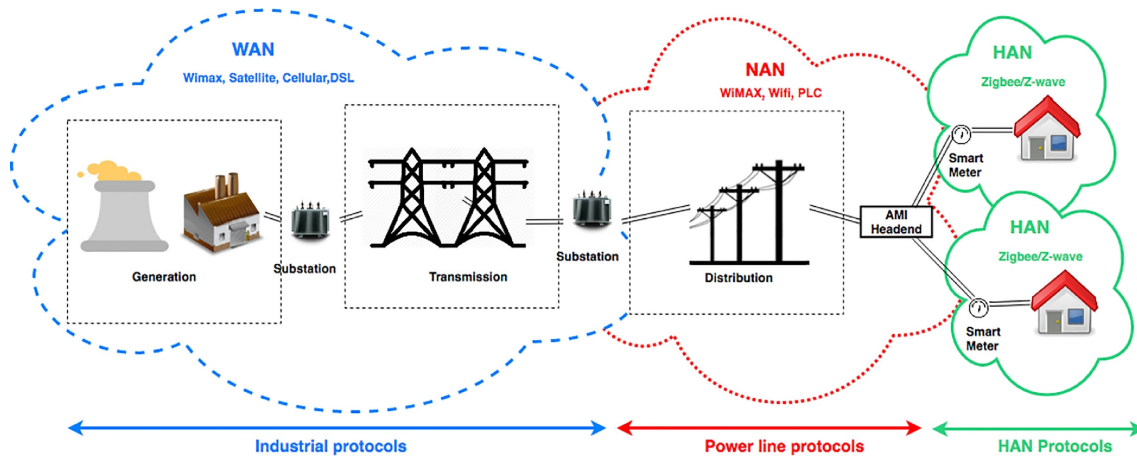


Abbildung 5.2: Aufbau eines intelligenten Stromnetzes [33]

Die Messwerte der Stromzähler sind als Kundendaten vor unerlaubtem Zugriff zu schützen. Außerdem sollten Umspannwerke nur von befugtem Personal gewartet werden können. Jede Änderung an der Funktionsweise eines Umspannwerkes muss auf die dafür verantwortliche Person zurückführbar sein. Um Stromausfälle zu vermeiden, muss das **SCADA**-System schnell auf Bedarfs- und Angebotsänderungen reagieren können. Zu den häufigsten Angriffen auf intelligente Stromnetze zählen die Versuche, das Kommunikationsnetz zu überlasten, sich im Kommunikationsnetz als anderer Teilnehmer auszugeben, Sensorwerte zu manipulieren oder zu lesen und das System mit Schadsoftware zu infizieren.

Um eine Überlastung des Kommunikationsnetzes zu verhindern, können Netzwerkkomponenten redundant aufgebaut werden. Außerdem sollten **TCP**-Verbindungen überwacht werden, weil häufig Netzwerkkomponenten mit **SYN**-Paketen überlastet werden. Wenn die Netzwerkauslastung überwacht wird, können auch weitere Attacken erkannt werden, die die Überlastung des Netzes zum Ziel haben. Um zu verhindern, dass Unbefugte Zugriff auf das System bekommen, kann eine Authentifizierung verlangt werden. Daten können geschützt werden, indem nur die **TLS**-verschlüsselten Versionen von **DNP3** und Modbus für **SCADA** zugelassen werden. Wenn die Nutzdaten der Pakete gescannt werden, kann Schadsoftware rechtzeitig gefunden werden. [33]

5.2.3 Medizinische IoT-Geräte

Das Internet of Things (**IoT**) gewinnt in einer Vielzahl von Lebensbereichen an Bedeutung. Auch im Gesundheitssystem lässt sich diese Entwicklung beobachten. Tragbare

Geräte, zu denen unter anderem Blutzuckersensoren und Smartwatches zählen, erfassen eine Vielzahl von Vitaldaten, wie beispielsweise die Herzfrequenz, die Atemfrequenz, die Sauerstoffsättigung sowie die Körpertemperatur. Auf diese Weise können sie bei der Diagnose sowie der Behandlung diverser Krankheiten assistieren. Des Weiteren sind sie imstande, im Falle eines Unfalls einen Notruf auszulösen.

Systeme wie das sogenannte Ambient Assisted Living unterstützen Senioren und Menschen mit Behinderung im Alltag, um ihnen ein selbstbestimmtes Leben zu ermöglichen, auch wenn sie auf Pflege angewiesen sind. Ein solches System stellt ein Netzwerk aus unterschiedlichen Geräten dar, welches nicht nur Vitalwerte, sondern ebenfalls Umweltfaktoren wie die Qualität der Luft und die Raumtemperatur misst. Zudem sind diese Geräte dazu in der Lage, Küchengeräte abzuschalten, Rollläden, Lichter und Heizungen zu steuern. Häufig sind solche Systeme mit telemedizinischen Einrichtungen verbunden, welche die Patienten bei gesundheitlichen Problemen betreuen.

Die Daten, die von medizinischen IoT-Geräten erfasst werden, sind besonders schutzbedürftig, da sie den Alltag und die Gesundheit von Patienten dokumentieren. IoT-Geräte kommunizieren über kabellose Übertragungsmedien mit geringer Reichweite. Protokolle wie Bluetooth oder ZigBee beinhalten bereits Funktionen zur Verschlüsselung und Authentifizierung. Insbesondere die Übertragung der erhobenen Daten über das Internet muss reguliert werden. Es ist sicherzustellen, dass ausschließlich der Patient selbst sowie autorisierte medizinische Fachkräfte auf die Daten zugreifen können. Dies impliziert, dass sowohl die übertragenen Daten als auch Rezepte und Krankenakten durch eine Verschlüsselung gesichert werden müssen. Eine hohe Verfügbarkeit ist lediglich für Notrufe erforderlich. [42]

6 Ansätze

In diesem Kapitel werden Ansätze zur Realisierung eines Zero-Trust-Gateways betrachtet. Dafür werden zunächst verschiedene bestehende Produkte betrachtet und im Anschluss eine eigene Lösung betrachtet.

6.1 Bestehende Firewall Produkte

Nach den Anforderungen beinhaltet das Gateway eine Next Generation Firewall. Um die Zeit und den Aufwand für die Implementierung des Prototypen zu reduzieren, wird ein bereits existierendes Produkt verwendet. In den folgenden Abschnitten werden deshalb zwei quelloffene und vier proprietäre Firewalls vorgestellt und miteinander verglichen.

6.1.1 pfSense

Die quelloffene Software *pfSense* vereint die Funktionen eines Routers mit denen einer Firewall. Sie baut dafür auf dem unixoiden Betriebssystem *FreeBSD* und der Paketfilter-Firewall *pf* auf [58, S. 4–10]. Sie kann entweder direkt auf einem Rechner oder auf einer virtuellen Maschine installiert werden und über eine webbasierte Schnittstelle konfiguriert werden [58, S. 359–495]. Die Community Edition von *pfSense* ist kostenlos erhältlich. Die Premiumversion *pfSense Plus*, die über mehr Funktionen verfügt und Kundenbetreuung zu jeder Zeit bietet, ist ab einem Preis von 129 US-Dollar pro Jahr verfügbar [38].

Die Hauptfunktionen von *pfSense* sind neben der Firewall [58, S. 596–655] und dem Router [58, S. 683–697] auch Virtual Local Area Networks [58, S. 710–717], Virtual Private Networks [58, S. 738–848] und ein Intrusion Prevention System, mit dem *pfSense* verhindert, dass Unbefugte in das System eindringen [58, S. 1221–1265]. Des Weiteren kann *pfSense* als DHCP- und DNS-Server benutzt werden [58, S. 906–914]. Es können Portale eingerichtet werden, über die Benutzer aufgefordert werden, sich für bestimmter Schnittstellen zu authentifizieren [58, S. 933–955]. Außerdem können die Netzauslastung, Benutzer von Portalen, Meldungen der Firewall und viele weitere

Informationen in einem konfigurierbaren Dashboard angezeigt werden [58, S. 970–1062].

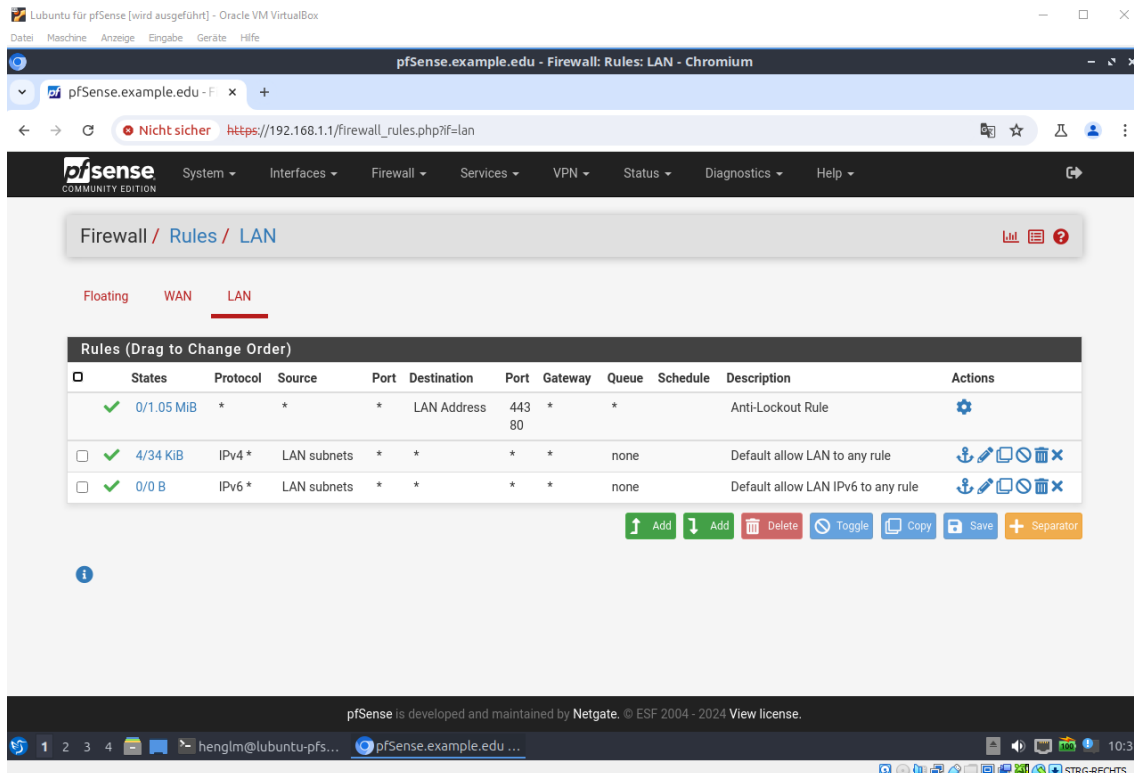


Abbildung 6.1: *pfSense* Firewall Richtlinien

Die Software wird auch für ihre Erweiterbarkeit geschätzt. Es können zusätzliche Pakete installiert oder Pakete, die nicht benötigt werden, deinstalliert werden [58, S. 1092–1102]. Beispielsweise kann die ebenfalls quelloffene Next Generation Firewall *Zenarmor* installiert werden [60]. Außerdem können eigene PHP- oder Shell-Skripte verwendet werden [58, S. 1796–1839]. [26]

6.1.2 OPNsense

OPNsense ist ebenfalls eine quelloffene Software, die auf *pfSense* aufbaut und die gleichen Funktionen besitzt *OPNsense* [1]. Im Gegensatz zu *pfSense* lässt sich *OPNsense* nicht über Pakete sondern über Plugins erweitern, die eine grafische Benutzeroberfläche bieten und dadurch einfacher zu verwalten sind. So gibt es Plugins für einen Virens Scanner, einen URL-Filter, ein E-Mail-Gateway und einen **RADIUS**-Server [10]. Ebenfalls gibt es ein Plugin für die Next Generation Firewall *Zenarmor* [60].

OPNsense ist kostenlos und es muss im Gegensatz zu *pfSense* kein Account angelegt werden, um die Software zu installieren. Es gibt eine Business Edition, die 149 Euro für ein Jahr oder 447 Euro für drei Jahre kostet. Diese verfügt über mehr Funktionen, enthält aber keine Kundenbetreuung [53]. Für 329 Euro pro Jahr können zwei Stunden Kundenberatung an Arbeitstagen zusätzlich gekauft werden. Wenn bereits die Business Edition gekauft wurde, kann der Preis auf 263 Euro gesenkt werden [56].

6.1.3 Quantum Force

Das Unternehmen *Miercom* hat im Auftrag der *Check Point Software Technologies Ltd.* fünf proprietäre Next Generation Firewalls aus dem Jahr 2024 getestet. Darunter war die Firewall *Quantum Force* des Auftraggebers, aber auch Produkte anderer Hersteller, unter anderem die Firewalls von *Fortinet*, *Palo Alto Networks* und *Cisco*. *Quantum Force* konnte dabei nach Angaben von *Miercom* 99,8% der eingespeisten Schadsoftware abwehren und war somit diesbezüglich den anderen getesteten Firewalls überlegen. Außerdem habe *Quantum Force* mit 0,13% die wenigsten Programme fälschlicherweise als Schadsoftware kategorisiert und mit 0,0% die wenigsten URLs zugelassen, die zu schädlichen Webseiten führen. [36]

Quantum Force bietet für die Basislizenz *NGFW* ein Intrusion Prevention System, ein Virtual Private Network und die Möglichkeit, den Netzverkehr auf Anwendungsebene zu filtern. Die Lizenz *NGTP* beinhaltet zusätzlich einen URL-Filter, Sicherheitsfunktionen für das DNS und Mechanismen gegen Bots, Viren und Spam. Bei dem Kauf der Hardware wird die Lizenz *SNBT* für ein Jahr vergeben. Sie bietet zusätzlich zu den Funktionen von *NGTP* Möglichkeiten, Anwendungen zu isolieren, unerlaubte Teile aus Dokumenten zu entfernen und Phishing-Attacken abzuwehren. Mechanismen, mit denen *Quantum Force* Angriffe erkennt, wurden nach eigenen Angaben durch maschinelles Lernen entwickelt. [44]

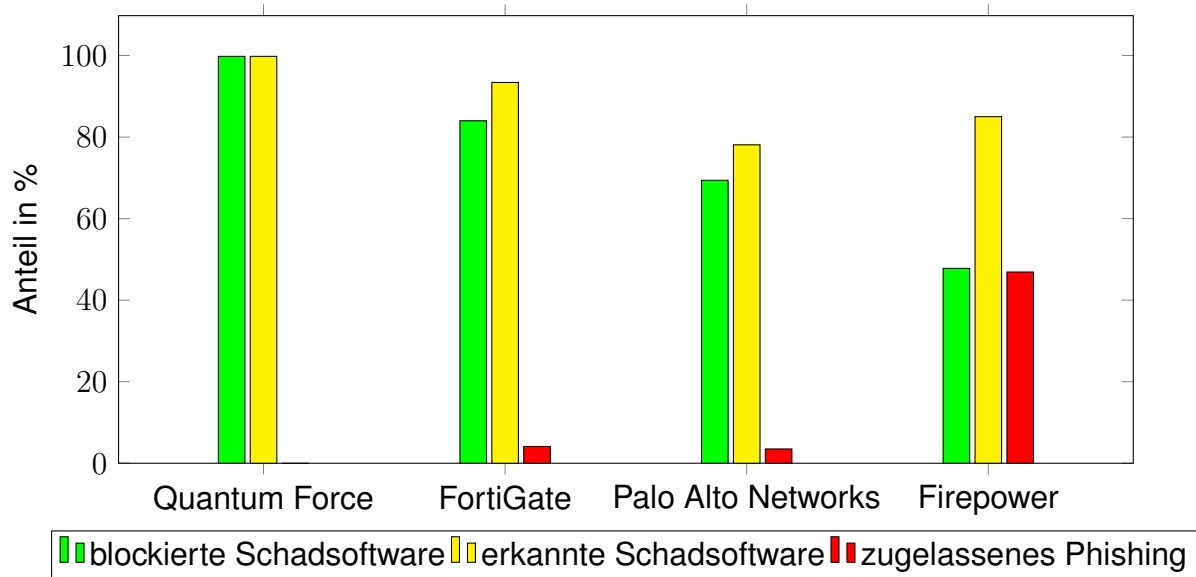


Abbildung 6.2: Vergleich proprietärer Firewalls aus Marktforschungsbericht von *Miercom* [36]

6.1.4 FortiGate

Die Next Generation Firewall *FortiGate* des Herstellers *Fortinet Inc.* ist in vier unterschiedlichen Lizenzen erhältlich. Die Basislizenz *Advanced Threat Protection* enthält ein Intrusion Prevention System, Kontrolle über den Datenverkehr auf Anwendungsebene und Mechanismen gegen Schadsoftware, Bots und Phishing. Dazu gehören auch die Möglichkeiten, Anwendungen isoliert auszuführen und Dokumente von unerlaubten Inhalten zu bereinigen.

Bei der Lizenz *Unified Threat Protection* kommen [URL-](#), [DNS-](#), Video- und Spamfilter zu den Basisfunktionen dazu. Die Lizenz *Enterprise Protection* erweitert die Firewall um eine schnelle präventive Erkennung von Schadsoftware und Mechanismen gegen unerwünschten Datenabfluss. Außerdem bietet sie spezielle Schutz- und Überwachungsfunktionen für [IoT](#)-Geräte. In der *A-la-carte*-Lizenz sind zusätzlich Schutzfunktionen speziell für industrielle Kontrollsysteme und Verwaltungsfunktionen für Software-defined Wide Area Networks enthalten. Außerdem übernimmt bei dieser Lizenz *Fortinet* die Verantwortung für das Netzwerk und Sicherheitsvorfälle, die dieses betreffen.

Alle Lizenzen beinhalten Kundenberatung zu jeder Zeit. Für die Lizenz *A-la-carte* gibt es zusätzlich ein Beratungsteam speziell für jeden Kunden inklusive Network und Security Operations Center. [18]

6.1.5 Palo Alto Networks

Die proprietäre Next Generation Firewall der *Palo Alto Networks Inc.* konnte im Marktforschungsbericht von *Miercom* weniger Schadsoftware als *FortiGate* abwehren, war aber in der Erkennung von Phishing *FortiGate* überlegen [36]. Sie ist nach Herstellerangaben die erste Firewall dieser Art.

Neben einer schnellen präventiven Erkennung von Schadsoftware, bietet das Produkt Funktionen gegen Phishing und Richtlinienempfehlungen für die Firewall durch Verhaltensanalyse der Netzkomponenten. Es kann den Netzverkehr auf Anwendungsebene kontrollieren und protokollieren. Außerdem fordert das Produkt für jede Anwendung eine Multi-Faktor-Authentifizierung und passt Sicherheitsmaßnahmen an das Verhalten der Clients an. Zusätzliche Funktionen können als Services gekauft werden. Darunter fallen URL-Filter, DNS-Sicherheitsfunktionen, Mechanismen gegen unerwünschten Datenabfluss und Sicherheitsfunktionen speziell für Cloud- und IoT-Systeme. [37]

6.1.6 Firepower

Die *Cisco Systems Inc.* hat mit *Firepower* nach Angaben von *Miercom* eine Next Generation Firewall, die zwar insgesamt mehr Schadsoftware als die Firewall der *Palo Alto Networks Inc.* erkennt und weniger Software falsch einordnet, aber von den getesteten Firewalls am wenigsten Schadsoftware tatsächlich abwehrt. Auch bei der Phishing-Bekämpfung hat *Firepower* am schlechtesten abgeschnitten, indem sie 47% der Phishing-URLs zugelassen hat. [36]

Firepower bietet Kontrolle über den Netzverkehr auf Anwendungsebene, ein Intrusion Prevention System und die Möglichkeit, ein Virtual Private Network einzurichten. Außerdem können Anwendungen isoliert werden, Bedrohungen automatisch aufgezeichnet werden und Zertifikate oder Schlüssel authentifiziert werden. *Firepower* kann entweder lokal über den *Device Manager* oder global über den *Threat Defense Manager* konfiguriert werden. [8]

6.2 Bestehende OPNsense Erweiterungen

Da *OPNsense* weit verbreitet ist, es erlaubt Erweiterungen mit grafischer Benutzeroberfläche zu entwickeln und quelloffen ist, wird *OPNsense* in dieser Arbeit als Gateway verwendet. *OPNsense* bietet eine Vielzahl an Erweiterungen, die zusätzlich zu den Grundfunktionen installiert werden können.

6.2.1 Zenarmor

Zenarmor ist eine Next Generation Firewall, die als Erweiterung in *OPNsense* installiert werden. Sie bietet ein Dashboard, das den Netzverkehr, die Beanspruchung des Prozessors, des Arbeitsspeichers und der Festplatte anzeigt. Außerdem können Anwendungen, Netzwerkkomponenten, Verbindungen und Schnittstellen überwacht werden. Systemmeldungen können in einer frei wählbaren Datenbank gespeichert werden.

Zenarmor beschränkt sich bei der Filterung des Netzverkehrs auf [DNS](#) und [HTTPS](#). Beispielsweise können Phishingseiten, unbekannte Seiten und Seiten ohne Inhalt blockiert werden. Außerdem ist es möglich, nach Kategorien zu Filtern. So können beispielsweise Werbeseiten, Messenger, Nachrichtenseiten oder Soziale Medien blockiert werden.

Dadurch dass sich *Zenarmor* auf Netzverkehr beschränkt, der bei Internetverbindungen entsteht, ist *Zenarmor* nicht für den Schutz eines industriellen Systems geeignet. Außerdem kann *Zenarmor* in der kostenfreien Version nur Netzwerkschnittstellen und keine einzelnen Netzwerkkomponenten schützen. Ein Intrusion Prevention System ist erst in der kostenpflichtigen Version enthalten. [60]

6.2.2 Suricata

Suricata ist ein kostenfreies Intrusion Prevention System. Es kann Netzverkehr anhand von [IP](#)-Adressen und [TLS](#)-Fingerabdrücken blockieren. Hierfür können manuell Regeln definiert werden, es kann aber auch eine vordefinierte Regelliste installiert werden. Hierfür stehen verschiedene Listen von *abuse.ch* und der *Proofpoint Inc.* zur Verfügung.

Die Listen beinhalten Botnetze, kompromittierte Rechner und Rechner, die nachweislich Schadsoftware verbreiten. Durch den [TLS](#)-Fingerabdruck können auch Geräte blockiert werden, die ihre [IP](#)-Adresse verschleiern. *Suricata* kann alle [TLS](#)-Zertifikate, [HTTP](#)-Verbindungen und [DNS](#)-Anfragen, die im Netzverkehr auftreten, für die spätere Analyse speichern. [57]

6.2.3 Maltrail

Maltrail ist ebenfalls ein Intrusion Prevention System, das Domains, [IP](#)-Adressen, [HTTP](#)-Header und heuristische Analysen nutzt, um Schadsoftware zu entdecken. *Maltrail* trennt logisch den Sensor, der die Schadsoftware erkennt, von dem Server, der den Vorfall dokumentiert. Dadurch sollen Unterbrechungen in der Erkennung von Schadsoftware verhindert werden. [54]

6.2.4 ClamAV

ClamAV ist ein Programm, das Dateien auf Schadsoftware überprüfen kann. In der Erweiterung kann eingestellt werden, welche Dateien überprüft werden sollen und wie viel Zeit, Speicherplatz und Leistung dafür maximal in Anspruch genommen werden sollen. Zu den möglichen Dateiformaten gehören ausführbare Dateien, [PDF](#)-, [XML](#)-, [HTML](#)- und [OLE](#)-Dateien, E-Mails und Archive. *ClamAV* ist mit anderen Erweiterungen kombinierbar. [9]

6.2.5 FreeRADIUS

FreeRADIUS ist eine kostenlose Erweiterung, die die Authentifizierung und Autorisierung von Nutzern und die Aufzeichnung von Nutzeraktivitäten mittels [RADIUS](#) erlaubt. In der Erweiterung können Benutzer mit Alias, Passwort und [IP](#)-Adresse angelegt werden. Außerdem können Dienste angelegt werden, für die sich Nutzer authentifizieren müssen. In der neuen Version kann die Erweiterung auch mit einem [LDAP](#)-Server verknüpft werden, sodass nicht alle Benutzerdaten in *OPNsense* gespeichert werden müssen. [19]

6.2.6 Postfix

Postfix kann als freier E-Mail-Server installiert werden. In der Erweiterung kann eingestellt werden, welche IP-Adressen und Domains als vertrauenswürdig betrachtet werden sollen. Außerdem können eigene Domains vergeben werden und es kann eingestellt werden, dass nur E-Mails von authentifizierten Sendern mit bekannter Domain entgegengenommen werden. Mit den Erweiterungen *ClamAV* und *Rspamd* können außerdem Spam-E-Mails blockiert werden. [22]

6.3 Eigene Erweiterung

Über eine *OPNsense*-Erweiterung könnten eintreffende Datenpakete überprüft und gefiltert werden. Bestimmte Bits innerhalb des Pakets könnten mit einem regulärem Ausdruck oder einer anderen Stelle im Datenpaket, wie zum Beispiel einer Prüfsumme oder einem Längenfeld, verglichen werden. Über eine grafische Benutzeroberfläche könnte ein Benutzer einstellen, für welchen Port die Regeln gelten, welche Paketfelder überprüft werden sollen und welche Art der Überprüfung stattfinden soll.

Der Benutzer kann Felder mit regulären Ausdrücken vergleichen. Außerdem kann der Benutzer einstellen, zwischen welchen Adressräumen die Regeln gelten sollen. Über diese Erweiterung könnten Regeln für verschiedene Anwendungsprotokolle erstellt werden, die sich an der Struktur eines Datenpakets des Protokolls orientieren. Beispielsweise könnten bei *DNP3*-Paketen nur bestimmte Befehle erlaubt werden.

6.3.1 Low-Fidelity Prototyp

Low-Fidelity Prototypen stellen wenige bis keine Interaktionsmöglichkeiten bereit. Sie sind im Gegensatz zu High-Fidelity Prototypen nicht dazu da, die Nutzerfreundlichkeit einer Anwendung zu testen. Sie stellen nur deren Konzept vor. Low-Fidelity Prototypen können schnell erstellt, verändert und verworfen werden. Sie sind für Anwendungsentwickler eine Möglichkeit, sich das Programm vorzustellen und es zu bewerten bevor es entwickelt wird. [49]

Für das Design des Low-Fidelity Prototyps wurden zunächst bereits bestehende *OPNsense*-Anwendungen betrachtet. Dabei ist aufgefallen, dass jede Erweiterung ein Markierungsfeld `Enabled` besitzt, über das sie aktiviert und deaktiviert werden kann. Bei der Paketfilter-Firewall und bei *Suricata* sind außerdem die bestehenden Filterregeln in einer Tabelle zu sehen.

The image shows two screenshots from the OPNsense web interface. The top screenshot displays the Firewall Filter rules configuration page. It shows a table with columns: Protocol, Source, Port, Destination, Port, Gateway, Schedule, and Description. There are three rules listed: 'Automatically generated rules', 'Default allow LAN to any rule', and 'Default allow LAN IPv6 to any rule'. Below these, there are two active rules: 'pass' and 'pass (disabled)'. The bottom screenshot shows the Suricata rules configuration page. It displays a table with columns: sid, Action, Source, ClassType, Message, and Info / Enabled. There are 16 rules listed, all with the action 'alert' and source 'compromised.rules'. The last rule has the source 'abuse.ch.sslblacklist.rules' and class type '#none#'. Below the table, there are checkboxes for 'Alert' and 'Drop'.

Protocol	Source	Port	Destination	Port	Gateway	Schedule	Description
							Automatically generated rules
IPv4 *	LAN net	*	*	*	*	*	Default allow LAN to any rule
IPv6 *	LAN net	*	*	*	*	*	Default allow LAN IPv6 to any rule
pass							log (disabled)
pass (disabled)							log (disabled)

sid	Action	Source	ClassType	Message	Info / Enabled
2500000	alert	compromised.rules	misc-attack	ET COMPROMISED Known Compromised or Hostile Host Traffi...	Info / Enabled
2500002	alert	compromised.rules	misc-attack	ET COMPROMISED Known Compromised or Hostile Host Traffi...	Info / Enabled
2500004	alert	compromised.rules	misc-attack	ET COMPROMISED Known Compromised or Hostile Host Traffi...	Info / Enabled
2500006	alert	compromised.rules	misc-attack	ET COMPROMISED Known Compromised or Hostile Host Traffi...	Info / Enabled
2500008	alert	compromised.rules	misc-attack	ET COMPROMISED Known Compromised or Hostile Host Traffi...	Info / Enabled
2500010	alert	compromised.rules	misc-attack	ET COMPROMISED Known Compromised or Hostile Host Traffi...	Info / Enabled
2500012	alert	compromised.rules	misc-attack	ET COMPROMISED Known Compromised or Hostile Host Traffi...	Info / Enabled
2500014	alert	compromised.rules	misc-attack	ET COMPROMISED Known Compromised or Hostile Host Traffi...	Info / Enabled
2500016	alert	compromised.rules	misc-attack	ET COMPROMISED Known Compromised or Hostile Host Traffi...	Info / Enabled
903200000	alert	abuse.ch.sslblacklist.rules	#none#	SSLBL: Malicious SSL certificate detected (Shylock C&C)	Info / Enabled

Abbildung 6.3: *OPNsense* Firewall Filterregeln oben und Filterregeln der Erweiterung *Suricata* unten

Rechts von der Tabelle gibt es die Optionen, Regeln zu bearbeiten. Bei der Firewall können die Regeln dort auch gelöscht werden. Bei *Suricata* müssen sie links ausgewählt werden und dann über einen zusätzlichen Knopf gelöscht unten links gelöscht werden. Bei der Firewall werden also für n Regeln n Klicks zum entfernen der Regeln gebraucht, während bei *Suricata* $n + 1$ Klicks benötigt werden. Deshalb hat die Erweiterung neben jeder Regel einen Knopf zum Entfernen dieser Regel. Außerdem können Regeln bei der Firewall links deaktiviert werden, während sie bei *Suricata* rechts deaktiviert werden. Weil der Knopf `Enabled` bei allen Erweiterungen ganz oben zu sehen ist und man das lateinische Alphabet von links nach rechts und von oben nach unten liest, ist der Eintrag zu dessen Status im Prototyp links neben jeder Regel zu finden. So ist er auch in der Tabelle die erste Information, die der Benutzer liest.

Enabled ☒

New rule

Rules

Enabled	Interface	Protocol	Port	Field name	RegEx	Description		
y	lan	TCP	25	MAIL FROM	^\w+@example\.edu	mail from		
y	lan	TCP	25	RCP TO	^\w+@example\.edu	mail to		
N	opt1	UDP	161	community	^public	SNMP read only		
y	opt2	TCP	20000	Function Code	^[1[34]]	DNP3 no reboot		

Abbildung 6.4: Low-Fidelity Prototyp der Hauptseite

Weil nach dem Zero-Trust-Ansatz standardmäßig alle Verbindungen blockiert werden, gibt es in der eigenen Erweiterung keine Auswahlmöglichkeit, ob eine Filterregel Pakete blockieren oder erlauben soll. Alle Filterregeln existieren, um bestimmte Pakete zu erlauben. Wenn ein Benutzer eine neue Regel erstellt oder eine alte Regel bearbeitet, soll sich ein neues Fenster öffnen.

Dort soll der Benutzer in zwei Klapplisten zwischen verschiedenen Schnittstellen und den beiden Transportprotokollen wählen. Außerdem soll er einen gültigen Port angeben, der überwacht werden soll. Die Angabe der IP-Adressen, zwischen denen die Kommunikation erlaubt sein soll, ist optional. Der Name des Paketfelds und der reguläre Ausdruck, mit dem dieses verglichen werden soll, sind erforderliche Benutzereingaben. Außerdem kann der Benutzer seine Filterregel beschreiben, damit er später noch weiß, wofür diese benutzt wird.

A low-fidelity prototype of a settings page for filter rules. The form is enclosed in a rectangular border and contains the following elements:

- Enabled:** A checkbox with a small square icon.
- Interface:** A dropdown menu with "LAN" selected and a downward arrow.
- Protocol:** A dropdown menu with "TCP" selected and a downward arrow.
- Port:** A text input field containing "0 - 65535".
- IP (from):** A text input field containing "0.0.0.0 - 255.255.255.255".
- IP (to):** A text input field containing "0.0.0.0 - 255.255.255.255".
- Field name:** A text input field containing "Name of the packet field".
- RegEx:** A text input field containing "Regular expression".
- Description:** A text input field containing "A few words about this filter rule".
- Buttons:** Two buttons at the bottom: "Save" on the left and "Cancel" on the right.

Abbildung 6.5: Low-Fidelity Prototyp der Einstellungsseite für Filterregeln

6.3.2 Scapy

Dienste für OPNsense werden in der Programmiersprache Python implementiert. Scapy ist ein interaktives Programm zur Paketmanipulation, das auch als Python-Bibliothek verwendet werden kann. Die Software ermöglicht es den Benutzern, Netzwerkpakete für eine Vielzahl von Protokollen zu erstellen, zu senden, zu inspizieren, zu zerlegen und zu fälschen. Das Tool ist in der Lage, eine Vielzahl gängiger Netzwerkoperationen durchzuführen, darunter die Rekonstruktion von Netzwerkruten, das Scannen nach Ports und die Netzwerkerkennung.

Die wesentliche Stärke von Scapy liegt in seiner Flexibilität, welche es dem Nutzer ermöglicht, Pakete Schicht für Schicht zu konstruieren, sie über das Netzwerk zu senden, Antworten zu erfassen und die Ergebnisse zu analysieren. Allerdings unterstützt Scapy bestimmte Transportprotokolle nur eingeschränkt und kann nicht mit vielen Paketen gleichzeitig umgehen. [50]

6.3.3 Reguläre Ausdrücke

Ein regulärer Ausdruck ist eine Zeichenkette, die eine Menge von Zeichenketten beschreibt. Er kann verwendet werden, um einzuschränken, welche Zeichenketten bei einem Paketfeld erlaubt sind. Außerdem kann er verwendet werden, um nach Zeichenketten zu suchen. Reguläre Ausdrücke können in endliche Automaten übersetzt werden und als solche ausgeführt werden. Werden in einem regulären Ausdruck viele Platzhalter, Gruppen möglicher Symbole oder Wiederholungen mit Längenrestriktion verwendet, kann der Automat sehr komplex und ineffizient werden. Yu et al. haben Regeln formuliert, mit denen reguläre Ausdrücke speziell zur Überprüfung von Datenpaketen effizienter gestaltet werden können.

Reguläre Ausdrücke, die dem Schema $AB[^C]\{10\}$ folgen, ergeben für die Suche nach Zeichenketten exponentiell wachsende Automaten. Sie prüfen am Anfang, ob eine Zeichenfolge vorkommt, und am Ende, ob ein Zeichen in einer bestimmten Anzahl folgender Zeichen nicht vorkommt. Der Automat muss dabei für jede neue Zeichenkette AB , die in $[^C]\{10\}$ vorkommt, überprüfen, ob diese selbst als neuer Treffer gültig ist, während die Gültigkeit der vorherigen Zeichenkette weiterhin bewiesen werden muss. Durch eine Erweiterung des regulären Ausdrucks um alle ungültigen Ausdrücke vor dem eigentlichen Ausdruck, kann das exponentielle Wachstum des suchenden Automaten verhindert werden: $([^A] | A[^B] | AB[^C]\{0,9\}) * AB[^C]\{10\}$.

Reguläre Ausdrücke, die dem Schema $^AB+[^C]\{10\}$ folgen, ergeben unnötig komplizierte Automaten, weil alle B , die nach dem ersten B kommen, als Teil von $B+$ sowie als Teil von $[^C]\{10\}$ interpretiert werden. Als Vereinfachung lassen sich die Ausdrücke $^AB+[^B][^C]\{9\}$ für den ersten Fall und $^AB[^C]\{10\}$ für den zweiten Fall implementieren.

Yu et al. haben außerdem ein Verfahren vorgestellt, mit dem deterministische endliche Automaten zusammengeführt werden, wenn die Größe des zusammengeführten Automaten nicht die Größe der einzelnen Automaten überschreitet. Deterministische endliche Automaten haben nur jeweils einen aktiven Zustand, weshalb sie eine geringe Zeitkomplexität aufweisen. Dieser Umstand kann jedoch auch zu einem exponentiellen Wachstum an Zuständen führen, was eine hohe Speicherkomplexität zur Folge hat. Das Verfahren von Yu et al. hat zum Ziel, einen Kompromiss zu finden, bei dem die Anzahl der gleichzeitig aktiven Zustände und die Anzahl der Zustände insgesamt möglichst gering gehalten werden soll. Durch die strikte Trennung der einzelnen deterministischen

endlichen Automaten ist die Zeitkomplexität geringer als bei einem nichtdeterministischen endlichen Automat, weil die parallel aktiven Zustände keinen Einfluss auf die Folgezustände des jeweils anderen haben können. [59]

6.3.4 Programmablauf

In diesem Abschnitt geht es darum, wie die geplante *OPNsense*-Erweiterung mit eintreffenden Paketen umgehen soll. Zunächst wird dafür die Anwendung einer Regel auf ein Paketfeld betrachtet. Danach wird die Anwendung mehrerer Regeln auf ein Paket betrachtet. Diese beinhaltet den ersten Teil.

Wenn ein Paket eintrifft, wird zunächst der Feldwert des Pakets, der laut Regel überprüft werden soll, geladen. Parallel dazu wird geprüft, ob der reguläre Ausdruck, der auf den Feldwert angewandt werden soll, bereits zu einem deterministischen endlichen Automat kompiliert wurde. Diese werden im Arbeitsspeicher zwischengespeichert, damit sie nicht für jedes Paket neu erzeugt werden müssen. Wenn noch kein Automat existiert, wird einer erzeugt. Dann durchläuft der Feldwert den Automaten. Wenn der Automat in einem akzeptierenden Zustand terminiert, stimmt der Feldwert mit dem regulären Ausdruck überein und das Paket ist nach der angewandten Regel gültig. Sonst wird das Paket verworfen.

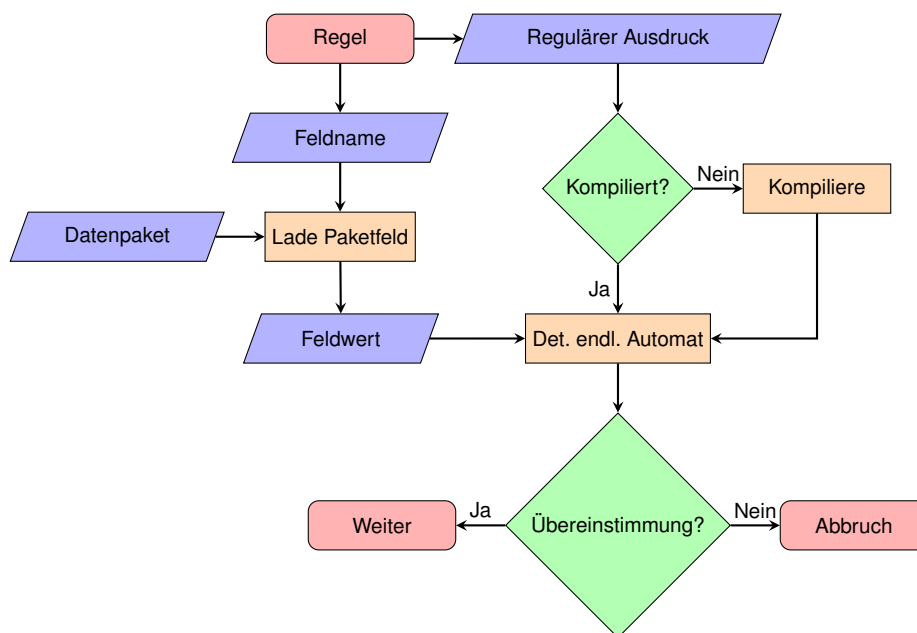


Abbildung 6.6: Programmablauf für die einfache Paketinspektion

Häufig sind für Pakete mehrere Regeln gültig. Wenn ein Paket eintrifft wird daher zunächst ein Regelset ausgewählt, das für die IP-Adressen und den Port des Pakets gültig ist. Das Paket wird zunächst auf die erste Regel aus dem Set getestet. Ist sie ungültig, wird das Paket verworfen. Sonst wird überprüft, ob das bereits die letzte Regel aus dem Regelset war. War sie es nicht, wird die nächste Regel über den oben beschriebenen Vorgang geprüft. Sonst wird das Paket versandt. Die Vergabe mehrerer Regeln für das gleiche Paketfeld wird unterbunden, weil mehrere Regeln auch in einem regulären Ausdruck zusammengefasst werden können.

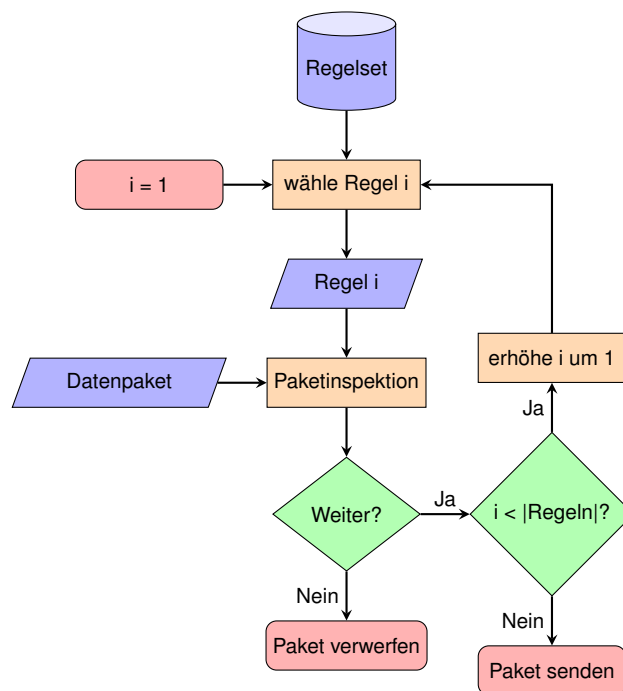


Abbildung 6.7: Programmablauf für Filterung auf Basis der Paketinspektion

Alternativ dazu wäre auch eine parallele Prüfung mehrerer Paketfelder möglich. In diesem Fall würden alle Paketfelder, wenn ein Paketfeld ungültig ist, trotzdem geprüft werden, da dies parallel geschieht. Die Laufzeit wäre zwar kürzer, aber die benötigte Rechenleistung höher. Da nicht beliebig viele Prozesse parallel laufen können, können Prozesse stattdessen sinnvoller an anderer Stelle parallelisiert werden. Wenn die Regeln nach Automatengröße und Strenge sortiert werden, können diejenigen, die eine kurze Laufzeit und eine hohe Rate an herausgefilterten Paketen haben, zuerst geprüft werden. So kann die Laufzeit optimiert werden.

Damit das passende Regelset schnell gewählt werden kann, sollten die Regeln in einer relationalen Datenstruktur gespeichert werden. So können sie schnell über den

Port und die IP-Adresse ausgewählt werden. Außerdem kann in einer Spalte die Anzahl $|Q|$ der Zustände des kompilierten Automaten gespeichert werden. In zwei weiteren Spalten können die Anzahl $|P|$ an Paketen, die durch diese Regel überprüft wurden, und die Anzahl $|P_{\text{drop}}|$ an Paketen, die durch diese Regel blockiert wurden, gespeichert werden. Aus diesen drei Werten lässt sich ein Indikator $x = \frac{|P| \cdot |Q|}{|P_{\text{drop}}|}$ berechnen, nach dem die Regeln priorisiert werden können. Je niedriger dieser Wert ist, desto früher soll die Regel ausgeführt werden. Regeln, die noch nicht zu einem Automat kompiliert wurden, werden zuerst ausgeführt, damit die Reihenfolge dynamisch bleibt.

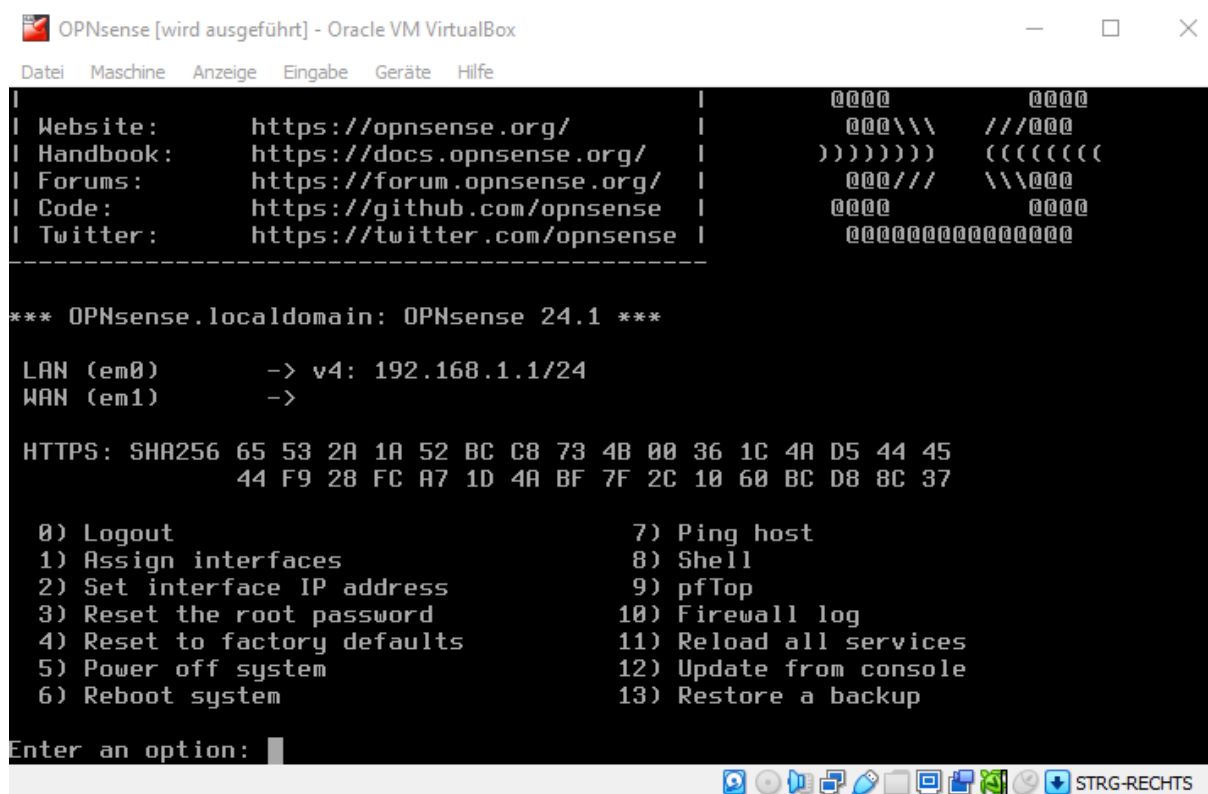
7 Implementierung einer Erweiterung

In diesem Kapitel wird beschrieben, wie die Firewall *OPNsense* in einer virtuellen Maschine der Marke *VirtualBox* installiert und eingerichtet werden kann. Außerdem wird beschrieben, wie die Benutzerschnittstelle einer eigenen *OPNsense*-Erweiterung implementiert werden kann. Der funktionale Teil der Erweiterung konnte wegen zeitlichem Mangel nicht implementiert werden. Darüber hinaus beschränkt sich die Benutzeroberfläche nur auf die Einstellungen für eine Regel.

7.1 Erste Einrichtung von OPNsense

Um *OPNsense* einzurichten, sind mehrere Schritte nötig. Zunächst muss eine virtuelle Maschine eingerichtet werden. Dafür wird im *VirtualBox Manager* eine neue virtuelle Maschine für das Betriebssystem *FreeBSD* in der 64-Bit-Architektur erstellt. Diese bekommt acht Gigabyte Hauptspeicherplatz, vierundsechzig Megabyte Grafikspeicherplatz und zwei Hauptprozessoren. Außerdem bekommt sie eine virtuelle Festplatte mit zehn Gigabyte Speicherplatz und ein optisches Laufwerk. In den Netzwerkeinstellung bekommt sie einen Netzwerkadapter des Typs NAT und zusätzlich einen Host-only Adapter.

Bevor die virtuelle Maschine gestartet wird, wird ein ISO-Abbild von *OPNsense* in das optische Laufwerk gelegt. Dann startet die virtuelle Maschine in der Konsole von *OPNsense*. Von dort aus wird über eine Anmeldung als `installer` mit dem Passwort `opnsense` das Installationsmenü gestartet. Nachdem die Tastaturbelegung und die Zeitzone eingestellt wurden, kann der NAT Adapter als Schnittstelle für das Wide Area Network und der Host-only Adapter als Schnittstelle für das Local Area Network eingestellt werden. Nachdem die virtuelle Festplatte nach dem *Unix File System* formatiert wurde und *OPNsense* installiert wurde, muss das ISO-Abbild aus dem optischen Laufwerk ausgeworfen werden, bevor die virtuelle Maschine neu startet.



```

OPNsense [wird ausgeführt] - Oracle VM VirtualBox
Datei Maschine Anzeige Eingabe Geräte Hilfe
| Website:      https://opnsense.org/ |
| Handbook:    https://docs.opnsense.org/ |
| Forums:      https://forum.opnsense.org/ |
| Code:        https://github.com/opnsense |
| Twitter:     https://twitter.com/opnsense |
|-----|
*** OPNsense.localdomain: OPNsense 24.1 ***

LAN (em0)      -> v4: 192.168.1.1/24
WAN (em1)      ->

HTTPS: SHA256 65 53 2A 1A 52 BC C8 73 4B 00 36 1C 4A D5 44 45
              44 F9 28 FC A7 1D 4A BF 7F 2C 10 60 BC D8 8C 37

0) Logout                      7) Ping host
1) Assign interfaces           8) Shell
2) Set interface IP address    9) pfTop
3) Reset the root password     10) Firewall log
4) Reset to factory defaults   11) Reload all services
5) Power off system            12) Update from console
6) Reboot system               13) Restore a backup

Enter an option:

```

Abbildung 7.1: *OPNsense* Konsole in einer virtuellen Maschine

Nach dem Neustart öffnet sich wieder die Konsole von *OPNsense*. Diesmal meldet sich der Benutzer als *root* mit dem Passwort *opnsense* an. Es öffnet sich ein Menü, in dem es die Option gibt, den Netzwerkschnittstellen *IP*-Adressen zuzuweisen. Über *DHCP* bekommen die Schnittstellen automatisch funktionierende *IP*-Adressen. Danach kann die Konfigurationsseite von *OPNsense* über die *IP*-Adresse des lokalen Netzwerks auf dem Gastgeber-System aufgerufen werden.

Nach der Anmeldung auf der Konfigurationsseite kann *OPNsense* unter dem Reiter System: Firmware aktualisiert werden. Dort können auch Erweiterungen installiert werden. Um *Zenarmor* zu installieren muss zunächst die Erweiterung *os-sunnyvalley* installiert werden. Danach kann die Erweiterung *os-sensei* installiert werden. Diese enthält die Next Generation Firewall. Nach einem Neustart des Systems erscheint ein neuer Reiter *Zenarmor*.

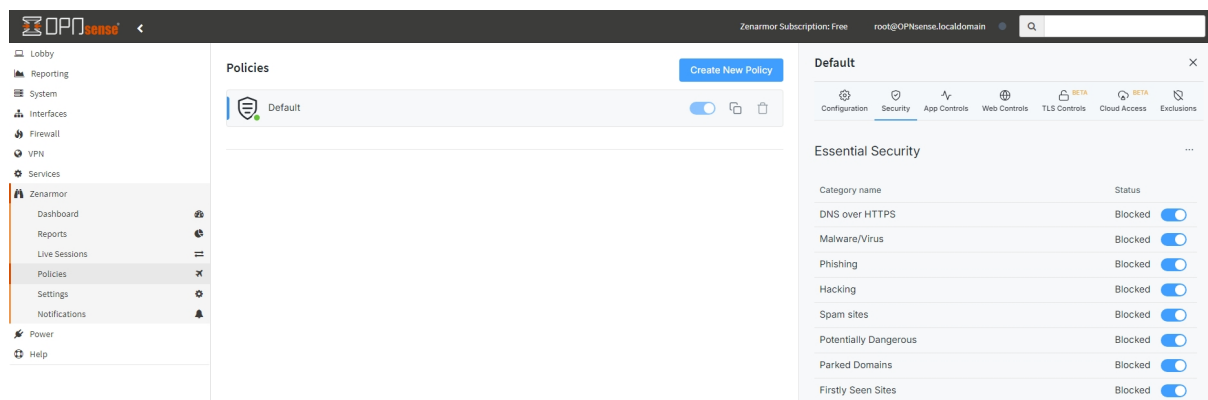


Abbildung 7.2: Zenarmor-Reiter auf der OPNsense-Konfigurationsseite

Damit das Gateway später getestet werden kann, wird die virtuelle Maschine um zwei Netzwerkadapter des Typs `Internes Netzwerk` erweitert. Diese Adapter können auf der Konfigurationsseite unter `Interfaces: Assignments` als Schnittstellen hinzugefügt werden. Beim Anklicken der neuen Schnittstellen können unter `Description` Namen für die Schnittstellen vergeben werden. Unter `IPv4 Configuration Type` wird `Static IPv4` ausgewählt und unter `IPv4 Address` werden die Adressen `192.168.1.1` und `192.168.2.1` jeweils mit der Netzmaske `/24` vergeben. Unter `Services: ISC DHCPv4` wird für die beiden Schnittstellen jeweils `Enable` ausgewählt und der volle Adressraum vergeben. Unter `Firewall: Rules` können Regeln vorgegeben werden, nach denen der Netzverkehr eingeschränkt wird. Pakete können nach `IP-Adressen` und `Ports` gefiltert werden.

7.2 Entwicklung der Benutzerschnittstelle

Um eine eigene Erweiterung zu entwickeln muss in der *OPNsense*-Konsole die Option `8, Shell`, gewählt werden. Über den Befehl `cd <Ordnername>` kann in einen anderen Ordner gewechselt werden, über `mkdir <Ordnername>` kann ein neuer Ordner erstellt werden und über `rmdir <Ordnername>` kann ein Ordner gelöscht werden. *OPNsense* folgt der Model-View-Controller-Architektur, bei der die Darstellungsebene von der Datenebene und der Logikebene getrennt ist. Erweiterungen gliedern sich in diese Struktur ein, wobei sie in der jeweiligen Ebene aus einem Ordner nach dem Schema `<Herausgeber>/<Projektname>` bestehen.

```
./user/local/opnsense/  
├── mvc/app/  
│   ├── controllers/Telent/ZeroTrust/  
│   │   ├── Api/  
│   │   │   ├── ServiceController.php  
│   │   │   └── SettingsController.php  
│   │   ├── forms/general.xml  
│   │   └── IndexController.php  
│   ├── models/Telent/ZeroTrust/  
│   │   ├── Menu/Menu.xml  
│   │   ├── ZeroTrust.php  
│   │   └── ZeroTrust.xml  
│   └── views/Telent/ZeroTrust/index.volt  
├── scripts/Telent/ZeroTrust/filter.py  
├── service/  
│   ├── conf/actions.d/actions_zerotrust.conf  
│   └── templates/Telent/ZeroTrust/  
│       ├── +TARGETS  
│       └── zerotrust.conf
```

Abbildung 7.3: Ordnerstruktur der Erweiterung

Um Dateien zu erstellen und zu Bearbeiten wurde der Editor *Vim* gewählt, weil dieser nicht auf eine grafische Benutzeroberfläche angewiesen ist. *Vim* kann über den Befehl `apt install vim` installiert werden. Über `vim <Dateiname>` können Dateien erstellt und bearbeitet werden, über `rm <Dateiname>` können sie entfernt werden. Auf dem Gastgeber-System können Programme in einer mächtigeren Entwicklungsumgebung, wie beispielsweise *Visual Studio Code*, entwickelt werden. Mit *PuTTY* kann der Programmcode dann über eine Secure Shell in Dateien der virtuellen Maschine kopiert werden.

Der erste Schritt bei der Entwicklung einer Erweiterung besteht darin, ein Datenmodell zu erstellen. Dafür wird die Klasse `ZeroTrust` erstellt, die nichts weiter macht, als von `BaseModel` zu erben. Sie ist in der Datei `ZeroTrust.php` definiert. Das tatsächliche Datenmodell wird in [XML](#) gebaut. Jede Erweiterung braucht die Möglichkeit, aktiviert und deaktiviert zu werden. Dafür wird ein `BooleanField` mit dem Namen `Enabled` erstellt. Weitere Felder folgen. Dazu gehören das `InterfaceField` namens `Interface`, das Feld `Protocol`, ein `OptionField` für das Transportprotokoll, ein `PortField` namens `Port` und zwei `NetworkFields`, `FromIP` und `ToIP`, für den Datenverkehr zwischen zwei spezifischen [IP](#)-Adressen. Außerdem gibt es das `TextField` namens `FieldName`, das angibt, welches Feld des Pakets überprüft werden soll. Die Felder mit den Namen `Regex` und `Description` sind einfache `TextFields`. Die verschiedenen Arten von Feldern sind

unter `models/OPNsense/Base/FieldTypes` vordefiniert.

```
1 <model>
2   <mount>//Telent/zerotrust</mount>
3   <description>
4     ZeroTrust application for bachelor thesis
5   </description>
6   <items>
7     <general>
8       <Enabled type = "BooleanField">
9         <default>0</default>
10        <Required>Y</Required>
11      </Enabled>
12      <!-- TODO -->
13    </general>
14  </items>
15</model>
```

Listing 7.1: `mvc/app/models/Telent/ZeroTrust/ZeroTrust.xml` Grundstruktur

Im nächsten Schritt wird die Darstellungsebene implementiert. Dafür wird die Datei `index.volt` erstellt. Sie lädt über die Funktion `partial` die Darstellungsvorlage `base_form`. Später werden in der Datei über die Skriptsprache JavaScript auch Getter und Setter implementiert. In [HTML](#) werden Knöpfe hinzugefügt.

```
1 {{ partial(
2   "layout_partials/base_form",
3   ['fields': generalForm, 'id': 'frm_GeneralSettings']
4 ) }}
5 <script type = "text/javascript"></script>
6 <div class = "col-md-12"></div>
```

Listing 7.2: `mvc/app/views/Telent/ZeroTrust/index.volt` Grundstruktur

Als nächstes wird die Logikebene implementiert. Dazu braucht es zunächst einen `IndexController`, der für das Laden der Startseite verantwortlich ist. Dafür verwendet er die Funktion `indexAction`. Den Umgang mit der Darstellungsebene erbt er von `Base`. Außerdem benutzt er das Formular `general`, das eine Schnittstelle zwischen dem Datenmodell und der Darstellungsebene bildet.

```
1 namespace Telent\ZeroTrust;
2 class IndexController extends \OPNsense\Base\IndexController {
3     public function indexAction() {
4         $this->view->pick("Telent/ZeroTrust/index");
5         $this->view->generalForm = $this->getForm("general");
6     }
7 }
```

Listing 7.3: `mvc/app/controllers/Telent/ZeroTrust/IndexController.php` Grundstruktur

Im Formular `general.xml` werden alle Datenstrukturen aus dem Datenmodell mit grafischen Elementen wie Eingabefeldern, Checkboxes oder Klapplisten verknüpft. Dabei muss sichergestellt sein, dass das Datenmodell mit dem grafischen Element kompatibel ist. Beispielsweise eignen sich Checkboxes nur für binäre Wahrheitswerte. Es können auch Elemente wie Titel und Infotexte hinzugefügt werden.

```
1 <form>
2     <field>
3         <id>zerotrust.general.Enabled</id>
4         <label>enabled</label>
5         <type>checkbox</type>
6         <help>Enable this feature</help>
7     </field>
8     <!-- TODO -->
9 </form>
```

Listing 7.4: `mvc/app/controllers/Telent/ZeroTrust/forms/general.xml` Grundstruktur

Nun kann die Benutzeroberfläche unter <https://192.168.56.101/ui/zerotrust> aufgerufen werden. Damit Benutzereingaben gespeichert werden können, wird ein `SettingsController` benötigt. Dieser erbt von `ApiMutableModelControllerBase`, bekommt als `internalModelName` den String `zerotrust` und als `internalModelClass` den Pfadnamen zu dem Datenmodell übergeben, das in `ZeroTrust.php` definiert ist. Auf der Darstellungsebene wird nun ein Skript hinzugefügt, dass die Einstellungen lädt und Änderungen speichern kann.

```

1 $(document).ready(function() {
2     var data_get_map = {
3         "frm_GeneralSettings": "/api/zerotrust/settings/get"
4     };
5     mapDataToFormUI(data_get_map);
6     $("#saveAct").click(function() {
7         saveFormToEndpoint(
8             url = "/api/zerotrust/settings/set",
9             formid = "frm_GeneralSettings",
10            callback_ok = function() { ajaxCall(
11                url = "/api/zerotrust/service/reload",
12                sendData = {},
13                callback = function(data, status) {}
14            ); }
15        );
16    });
17 });

```

Listing 7.5: mvc/app/views/Telent/ZeroTrust/index.volt JavaScript

In den div-Container unter dem Skript soll außerdem ein Knopf mit der ID `saveAct` hinzugefügt werden. Über diesen kann dann die Funktion `saveFormToEndpoint` aufgerufen werden. Gespeicherte Daten können jetzt unter <https://192.168.56.101/api/zerotrust/settings/get> abgerufen werden. Über die Funktion `ajaxCall` wird der `ServiceController` aufgerufen. Über diesen können Eingaben permanent gespeichert und für Dienste zugänglich gemacht werden.

```

1 class ServiceController extends ApiControllerBase {
2     public function reloadAction() {
3         $status = "failed";
4         if ($this->request->isPost()) {
5             $backend = new Backend();
6             $bckresult = trim($backend->configdRun("reload"));
7             if ($bckresult == "OK") {
8                 $status = "ok";
9             }
10        }
11        return array("status" => $status);
12    }
13 }

```

Listing 7.6: mvc/app/controllers/Telent/ZeroTrust/Api/ServiceController.php
Grundstruktur

Als letzter Schritt werden für die Speicherung noch die Dateien `zerotrust.conf` und `+TARGETS` benötigt. In `+TARGETS` wird der Speicherort der Konfigurationsdaten als Dateipfad angegeben. Die Form der gespeicherten Daten wird in `zerotrust.conf` angegeben.

```
1 {% if helpers.exists('Telent.zerotrust.general') and
2 Telent.zerotrust.general.Enabled|default("0") == "1" %}
3     [general]
4         Interface = {{ Telent.zerotrust.general.Interface|default("lan") }}
5         # TODO
6 {% endif %}
```

Listing 7.7: `service/templates/Telent/ZeroTrust/zerotrust.conf` Grundstruktur

Damit ein Dienst mit der Benutzerschnittstelle verknüpft werden kann, muss er unter `actions_zerotrust.conf` registriert werden. Dort bekommt der Dienst in eckigen Klammern einen Namen und darunter unter `command` einen Dateipfad zugewiesen. Da es sich um ein Python Skript handelt bekommt die Konfigurationsdatei zusätzlich die Zeile `type:script_output`.

Im Python Skript können Benutzereingaben über einen `ConfigParser` eingelesen werden. Dieser liest alle gespeicherten Werte als String ein. Wenn es sich um Zahlenwerte handelt, können diese über die Funktion `int(string, base)` in Ganzzahlen umgewandelt werden. Mit den Befehlen `try` und `except` können Ausnahmesituationen behandelt werden. Wenn alle Benutzereingabemöglichkeiten Pflichtfelder sind, sollten normal keine Ausnahmesituationen auftreten.

```
1 if os.path.exists(conf_path):
2     cnf = ConfigParser()
3     cnf.read(conf_path)
4     if cnf.has_section('general'):
5         try:
6             interface = cnf.get('general', 'Interface')
7             # TODO
8         except:
9             # TODO
```

Listing 7.8: `scripts/Telent/ZeroTrust/filter.py` `ConfigParser`

Über die *Scapy*-Funktion `sniff` können bestimmte Datenpakete, die eine Schnittstelle erreichen, analysiert werden. Über einen String können dabei alle Datenpakete gefiltert werden, so dass nur relevante Pakete, die die Firewall nicht schon blockiert, analysiert werden. Über den Parameter `prn` kann eine Funktion angegeben werden, die für jedes Paket, das auf den Filter zutrifft, ausgeführt werden soll.

```
1 strfilter = f"ip proto {protocol} && port {port} &&
2   src net {fromIP} && dst net {toIP}"
3 sniff(iface = interface, filter = strfilter,
4   prn = lambda pkt: insp_pkt(pkt, fieldName, regex, interface)
5 )
```

Listing 7.9: scripts/Telent/ZeroTrust/filter.py sniff

Aufgrund zeitlicher Mängel konnte die eigentliche Paketinspektion nicht mehr implementiert werden. Im Folgekapitel werden Probleme, die während der Arbeit aufgetreten sind, und Mängel dieses Berichts diskutiert. Danach wird vorgestellt, wie diese fortgesetzt werden könnte.

Damit die Erweiterung im *OPNsense*-Menu unter *Services* angezeigt wird, muss das in der Datei `Menu.xml` definiert werden. Diese besteht aus dem Element `ZeroTrust`, das in das Element `Services` eingebettet ist. Dieses ist wiederum in das Element `menu` eingebettet. Das Element `ZeroTrust` hat die Attribute `VisibleName`, den Anzeigenamen, und `url`, den Pfad zur Hauptseite.

8 Diskussion

Ziel dieser Arbeit war es, ein Gateway-System zu entwerfen und zu implementieren, das nur notwendige, authentifizierte Kommunikation zwischen Netzwerksegmenten erlaubt. Um dieses Ziel zu erreichen, wurden zunächst verschiedene Kommunikationsprotokolle betrachtet. Erst später wurden Anwendungsfälle eines solchen Systems herausgearbeitet. Um die Menge der Kommunikationsprotokolle einzuschränken, wäre es sinnvoller gewesen, erst Anwendungsfälle zu betrachten.

Insgesamt wurden drei Anwendungsfälle betrachtet, von denen sich zwei mit der Kontrolle und Überwachung physikalischer Größen beschäftigen. Der dritte Anwendungsfall befasst sich mit der Erfassung und Verbreitung hochsensibler personenbezogener Daten. Anwendungsfälle für klassische Büro-Umgebungen wurden nicht betrachtet. Beispielsweise hätten zusätzlich zu den drei betrachteten Anwendungsfällen noch die Anforderungen an die Kommunikation zwischen Behörden betrachtet werden können. Diese können sich in Teilen von den Anforderungen an den dritten Anwendungsfall unterscheiden.

Nach der Betrachtung der Anwendungsfälle wurden sechs Firewall-Produkte betrachtet. Zwei davon waren quelloffen und vier waren proprietär. Die beiden quelloffenen Firewalls wurden auf einer virtuellen Maschine installiert und anhand der Funktionen, die sie dort bieten und die in ihrem Benutzerhandbuch beschrieben sind, wurden sie beurteilt. Dabei fiel auf, dass *OPNsense* leichter zu installieren war und eine übersichtlichere Benutzeroberfläche bot. Diese Bewertung ist subjektiv. Objektiv ist *OPNsense* eine Weiterentwicklung von *pfSense* und in der Grundfunktionalität gleich, weshalb es keinen großen Unterschied gemacht hätte, wäre *pfSense* für den weiteren Verlauf der Arbeit ausgewählt worden. Die proprietären Firewalls wurden anhand von Herstellerangaben und einem Marktforschungsbericht bewertet. Es bestand also ein Unterschied in der Methodik bei der Bewertung der proprietären Firewalls im Vergleich zur Bewertung der quelloffenen Firewalls.

Nachdem *OPNsense* für den weiteren Verlauf der Arbeit ausgewählt wurde, wurden Erweiterungen betrachtet, die es für dieses Firewall-Produkt gibt. Die Erweiterungen wurden zwar installiert und ihre Benutzeroberflächen wurden analysiert, sie wurden

aber nicht getestet. Ebenso wurden die Firewall und der DHCP-Server zwar eingestellt, aber nicht weiter getestet. Der Router wurde gar nicht betrachtet, weil er im Kontext der Arbeit nicht relevant ist.

Ursprünglich war geplant, dass die geplante Erweiterung nach dem Programmentwurf in Teilen implementiert und anschließend getestet werden sollte. Dafür war es vorgesehen, zunächst eine Erweiterung zu entwickeln, die nur eine konfigurierbare Regel beherrscht und die Regel dann in einem Netz aus virtuellen Maschinen zu testen. Der Entwicklungsaufwand wurde unterschätzt und die Recherche priorisiert, weshalb nur die Benutzerschnittstelle dafür zustande kam und nichts getestet werden konnte.

Ziel der Erweiterung war es, den Benutzern die vollständige Kontrolle darüber zu geben, welche Datenpakete ausgetauscht werden dürfen. Dafür sollten sie über reguläre Ausdrücke nur bestimmte Feldwerte von Paketen zulassen. Die Zeit- und Speicherkomplexität für die Prüfung regulärer Ausdrücke wurden zwar betrachtet, beim Programmentwurf fanden die vorgeschlagenen Lösungen zur Minimierung der Komplexität jedoch kaum Beachtung. Es wurde davon ausgegangen, dass einzelne Paketfeldwerte wegen ihrer meistens geringen Größe keine komplexen regulären Ausdrücke zur Überprüfung brauchen.

Ein Nachteil der gewählten Lösung ist, dass sich Administratoren der Erweiterung mit den Kommunikationsprotokollen, die sie kontrollieren wollen, sehr gut auskennen müssen. Sie müssen ihre Ports, ihre Paketstrukturen und die Bedeutung der einzelnen Felder kennen. Daher ist die Einstiegshürde für die Benutzung der Erweiterung hoch.

Bei dem Programmentwurf wurde keine Entschlüsselung von Anwendungsprotokollen, die TLS nutzen, eingeplant. Hierfür muss die Firewall als Proxy dienen. Die verschlüsselte Kommunikation findet nicht von Ende zu Ende sondern jeweils von Ende zu Firewall statt. Die Entschlüsselung wäre zwar wichtig, weil nach dem Zero-Trust-Konzept nur verschlüsselte Kommunikation erlaubt ist, sie hätte aber den Programmentwurf komplizierter gemacht und so noch mehr Zeit beansprucht.

9 Ausblick

Nachdem der Prototyp in dieser Arbeit nur in Teilen implementiert wurde, kann er nachfolgend vervollständigt werden. Für eine effiziente Arbeitsweise wird in der Software-Entwicklung das Vorgehensmodell Scrum benutzt. Dieses beschreibt, wie die Arbeit an einem Software-Projekt geplant werden kann, um schnell auf sich ändernde Anforderungen reagieren zu können. Demnach soll die Arbeit an einem Projekt in verschiedene Schritte eingeteilt werden, die in einem sogenannten Product Backlog festgehalten werden. Dann werden sogenannte Sprints vereinbart. Das sind Zeitspannen, in denen ein Teil der Arbeit erledigt werden soll. Zu Beginn eines jeden Sprints wird ein Sprint Backlog aus Teilschritten festgelegt. Am Ende jedes Sprints wird betrachtet, wie viele Aufgaben aus dem Sprint Backlog erledigt werden konnten. So kann die Menge an Aufgaben für den nächsten Sprint besser eingeschätzt werden. [51]

Durch diese Arbeitsweise kann die Zeit, die für die Implementierung der fehlenden Funktionen benötigt wird, besser eingeschätzt werden. Zu den fehlenden Funktionen der *OPNsense*-Erweiterung zählen die Paketinspektion durch einen deterministischen endlichen Automaten, eine relationale Datenstruktur und eine Benutzeroberfläche zur Speicherung mehrerer Regeln und verschiedene Laufzeitoptimierungen. Ein Algorithmus, der Regeln priorisiert, und ein Algorithmus, der die Automaten minimiert, können implementiert werden, um die Laufzeit zu reduzieren. Außerdem kann die Erweiterung in einen Proxy eingebettet werden und verschlüsselten Verkehr zur Paketinspektion entschlüsseln.

Wenn die Erweiterung nach Plan implementiert wurde, kann sie getestet werden. Hierfür kann ein Netz aus virtuellen Maschinen aufgebaut werden. Dafür wird zunächst eine virtuelle Maschine erzeugt. Auf ihr kann ein Betriebssystem ohne grafische Benutzeroberfläche installiert werden. Hierfür eignet sich beispielsweise *Ubuntu Server*. Wenn das Betriebssystem installiert wurde, kann *Scapy* installiert werden, um selbst erstellte Pakete zu versenden. Die Protokolle [HTTP](#), [HTTPS](#), [LDAP](#) und [SMTP](#) werden von dem Programm *cURL* unterstützt, das bereits in den meisten Linux-Distributionen vorinstalliert ist.

Nachdem die virtuelle Maschine konfiguriert wurde, kann sie geklont werden. Dann gibt es mehrere virtuelle Maschinen mit den gleichen Einstellungen. Diese können im *VirtualBox Manager* zu einer Gruppe zusammengefasst werden und so später gleichzeitig gestartet werden. Als Netzwerkadapter kann den beiden virtuellen Maschinen jeweils eins der internen Netze zugeteilt werden, die das Gateway ebenfalls als Netzwerkadapter hat. So wird sichergestellt, dass sie nur über die Firewall miteinander kommunizieren können.

Über **DHCP** werden den beiden virtuellen Maschinen vom Gateway **IP**-Adressen zugewiesen. Danach kann mit einer Echo-Anfrage überprüft werden, ob das Gateway und die jeweils andere virtuelle Maschine im **IP**-Netz erreichbar sind. Sind sie erreichbar, kann mit *Scapy* getestet werden, ob explizit erlaubte Datenpakete übermittelt werden und ob alle anderen Datenpakete blockiert werden. Außerdem kann ein Skript geschrieben werden, das automatisch Pakete versendet. So kann getestet werden, wie leistungsfähig die Erweiterung ist. Gegebenenfalls kann sie danach weiter optimiert werden.

Auch die Benutzerfreundlichkeit der Erweiterung kann durch Umfragen getestet werden. Dafür müssen zunächst primäre Benutzergruppen definiert werden, denen die Erweiterung vorgestellt wird. Auf Basis von deren Meinungen können Konzepte zur Verbesserung des Benutzererlebnisses entwickelt werden. Diese können dann wieder den primären Benutzergruppen vorgestellt und anschließend verworfen oder weiterentwickelt und implementiert werden.

Neben der Paketinspektion ist für die Erfüllung der Anforderungen an eine Zero-Trust-Architektur auch wichtig, dass Geräte authentifiziert werden. Dafür kann die Erweiterung *Free**RADIUS*** verwendet werden. Sie wurde in dieser Arbeit zwar kurz vorgestellt, aber nicht konfiguriert und getestet. Diese Aufgaben können ebenfalls Teil zukünftiger Arbeiten sein.

Insgesamt bietet diese Arbeit die Grundlagen für die Implementierung und Weiterentwicklung einer dynamisch anpassbaren Gateway-Komponente nach dem Zero-Trust-Modell. Diese ermöglicht umfassende Kontrolle darüber, welche Inhalte zwischen Netzwerkkomponenten in Form von Datenpaketen ausgetauscht werden.

Literatur

- [1] *About OPNsense*. Webseite. Deciso B.V., 2024. URL: <https://opnsense.org/about/about-opnsense>.
- [2] Mike Bishop. *RFC 9114: HTTP/3*. Standard. Internet Engineering Task Force, 2022. DOI: [10.17487/RFC9114](https://doi.org/10.17487/RFC9114).
- [3] Robert Braden. *RFC 1122: Requirements for Internet Hosts - Communication Layers*. Standard. Internet Engineering Task Force, 1989. DOI: [10.17487/RFC1122](https://doi.org/10.17487/RFC1122).
- [4] Rene Bretz. *Das Prinzip der SNMP-Kommunikation*. Grafik. Wikimedia Commons, 2006. URL: <https://commons.wikimedia.org/wiki/File:Snmp.PNG>.
- [5] Jess Case u. a. *RFC 1157: A Simple Network Management Protocol (SNMP)*. Standard. Internet Engineering Task Force, 1990. DOI: [10.17487/RFC1157](https://doi.org/10.17487/RFC1157).
- [6] Jess Case u. a. *RFC 3410: Introduction and Applicability Statements for Internet Standard Management Framework*. Standard. Internet Engineering Task Force, 2002. DOI: [10.17487/RFC3410](https://doi.org/10.17487/RFC3410).
- [7] W.R. Cheswick, S.M. Bellovin und A.D. Rubin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison-Wesley, 2003. ISBN: 9780201634662. URL: <https://wilyhacker.com>.
- [8] *Cisco Firepower 1000 Series*. Datenblatt. Cisco Systems Inc., 2023. URL: <https://www.cisco.com/c/en/us/products/collateral/security/firepower-1000-series/datasheet-c78-742469.pdf>.
- [9] *ClamAV*. Produktdokumentation. Deciso B.V., 2024. URL: <https://docs.opnsense.org/manual/how-tos/clamav.html>.
- [10] *Community Plugins*. Produktdokumentation. Deciso B.V., 2024. URL: <https://docs.opnsense.org/plugins.html>.
- [11] DaMutz. *Diffie-Hellman-Schlüsselaustausch*. Grafik. Wikimedia Commons, 2008. URL: <https://commons.wikimedia.org/wiki/File:Diffie-Hellman-Schl%C3%BCsselaustausch.svg>.
- [12] Steve Deering und Robert Hinden. *RFC 8200: Internet Protocol, Version 6 (IPv6) Specification*. Standard. Internet Engineering Task Force, 2017. DOI: [10.17487/RFC8200](https://doi.org/10.17487/RFC8200).

- [13] Denniss. *Diagramm des Beginns einer TCP-Verbindung*. Grafik. Wikimedia Commons, 2005. URL: <https://commons.wikimedia.org/wiki/File:Tcp-handshake.png>.
- [14] Ralph Droms. *RFC 2131: Dynamic Host Configuration Protocol*. Standard. Internet Engineering Task Force, 1997. DOI: [10.17487/RFC2131](https://doi.org/10.17487/RFC2131).
- [15] Ralph Droms u. a. *RFC 3315: Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*. Standard. Internet Engineering Task Force, 2003. DOI: [10.17487/RFC3315](https://doi.org/10.17487/RFC3315).
- [16] Wesley Eddy. *RFC 9293: Transmission Control Protocol (TCP)*. Standard. Internet Engineering Task Force, 2022. DOI: [10.17487/RFC9293](https://doi.org/10.17487/RFC9293).
- [17] Roy Fielding, Mark Nottingham und Julian Reschke. *RFC 9110: HTTP Semantics*. Standard. Internet Engineering Task Force, 2022. DOI: [10.17487/RFC9110](https://doi.org/10.17487/RFC9110).
- [18] *FortiGate 400F Series*. Datenblatt. Fortinet Inc., 2024. URL: <https://www.fortinet.com/content/dam/fortinet/assets/data-sheets/fortigate-400f-series.pdf>.
- [19] *FreeRADIUS*. Produktdokumentation. Deciso B.V., 2024. URL: <https://docs.opnsense.org/manual/how-tos/freeradius.html>.
- [20] Rainer Gerhards. *RFC 5424: The Syslog Protocol*. Standard. Internet Engineering Task Force, 2009. DOI: [10.17487/RFC5424](https://doi.org/10.17487/RFC5424).
- [21] Hank van Helvete. *Schematische Darstellung der rekursiven und iterativen DNS-Abfrage*. Grafik. Wikimedia Commons, 2006. URL: <https://commons.wikimedia.org/wiki/File:Dns-abfrage.svg>.
- [22] *How To: Setting Up A Mail Gateway*. Produktdokumentation. Deciso B.V., 2024. URL: <https://docs.opnsense.org/manual/how-tos/mailgateway.html>.
- [23] *IEEE 1815-2010: Standard for Electric Power Systems Communications - Distributed Network Protocol (DNP3)*. Standard. Institute of Electrical und Electronics Engineers, 2010. DOI: [10.1109/IEEESTD.2010.5518537](https://doi.org/10.1109/IEEESTD.2010.5518537).
- [24] *ISO/IEC 7498-1:1994: Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model*. Standard. International Organization for Standardization und International Electrotechnical Commission, 1994. URL: [http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994\(E\).zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip).
- [25] John Klensin. *RFC 5321: Simple Mail Transfer Protocol*. Standard. Internet Engineering Task Force, 2008. DOI: [10.17487/RFC5321](https://doi.org/10.17487/RFC5321).

- [26] D. Kreculj u. a. „pfSense Router and Firewall Software“. In: *Sinteza 2023 - International Scientific Conference on Information Technology, Computer Science, and Data Science*. 2023, S. 132–137. DOI: [10.15308/Sinteza-2023-132-137](https://doi.org/10.15308/Sinteza-2023-132-137).
- [27] Junyan Liang und Yoohwan Kim. „Evolution of Firewalls: Toward Securer Network Using Next Generation Firewall“. In: *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*. 2022, S. 752–759. DOI: [10.1109/CCWC54503.2022.9720435](https://doi.org/10.1109/CCWC54503.2022.9720435).
- [28] David Mills u. a. *RFC 5905: Network Time Protocol Version 4: Protocol and Algorithms Specification*. Standard. Internet Engineering Task Force, 2010. DOI: [10.17487/RFC5905](https://doi.org/10.17487/RFC5905).
- [29] Paul Mockapetris. *RFC 1034: Domain Names - Concepts and Facilities*. Standard. Internet Engineering Task Force, 1987. DOI: [10.17487/RFC1034](https://doi.org/10.17487/RFC1034).
- [30] Paul Mockapetris. *RFC 1035: Domain Names - Implementation and Specification*. Standard. Internet Engineering Task Force, 1987. DOI: [10.17487/RFC1035](https://doi.org/10.17487/RFC1035).
- [31] *Modbus Application Protocol Specification V1.1b3*. Standard. Modbus Organization Inc., 2012. URL: https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf.
- [32] *Modbus Messaging on TCP/IP Implementation Guide V1.0b*. Standard. Modbus Organization Inc., 2006. URL: https://modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf.
- [33] Zakaria El Mrabet u. a. „Cyber-security in smart grid: Survey and challenges“. In: *Computers & Electrical Engineering* 67 (2018), S. 469–482. ISSN: 0045-7906. DOI: [10.1016/j.compeleceng.2018.01.015](https://doi.org/10.1016/j.compeleceng.2018.01.015).
- [34] *NET.3.2 Firewall*. Standard. Bundesamt für Sicherheit in der Informationstechnik, 2023. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/IT-GS-Kompodium_Einzel_PDFs_2023/09_NET_Netze_und_Kommunikation/NET_3_2_Firewall_Edition_2023.pdf.
- [35] *Netzwerksicherheit - Weltweit*. Marktanalyse. Statista GmbH, 2024. URL: <https://de.statista.com/outlook/tmo/cybersecurity/cyberloesungen/netzwerksicherheit/weltweit>.
- [36] *NGFW Firewall Security Benchmark 2024 (SR240117C)*. Marktforschungsbericht. Miercom, 2024. URL: <https://miercom.com/check-point-earns-ngfw-certified-secure>.
- [37] *PA-400 Series*. Datenblatt. Palo Alto Networks Inc., 2024. URL: <https://www.paloaltonetworks.de/resources/datasheets/pa-400-series>.

- [38] *pfSense Plus Subscriptions*. Preisliste. Rubicon Communications LLC, 2024. URL: <https://www.netgate.com/pfsense-plus-software/software-types>.
- [39] *Positionspapier Zero Trust 2023*. Positionspapier. Bundesamt für Sicherheit in der Informationstechnik, 2023. URL: <https://www.bsi.bund.de/dok/zero-trust>.
- [40] Jon Postel. *RFC 768: User Datagram Protocol*. Standard. Internet Engineering Task Force, 1980. DOI: [10.17487/RFC768](https://doi.org/10.17487/RFC768).
- [41] Jon Postel. *RFC 791: Internet Protocol - DARPA Internet Program - Protocol Specification*. Standard. Internet Engineering Task Force, 1981. DOI: [10.17487/RFC791](https://doi.org/10.17487/RFC791).
- [42] Bikash Pradhan, Saugat Bhattacharyya und Kunal Pal. „IoT-Based Applications in Healthcare Devices“. In: *Journal of Healthcare Engineering* (2021), S. 2040–2295. DOI: [10.1155/2021/6632599](https://doi.org/10.1155/2021/6632599).
- [43] Dominik Püllen u. a. „Safety Meets Security: Using IEC 62443 for a Highly Automated Road Vehicle“. In: *Computer Safety, Reliability, and Security*. Springer International Publishing, 2020, S. 325–340. ISBN: 978-3-030-54549-9. DOI: [10.1007/978-3-030-54549-9_22](https://doi.org/10.1007/978-3-030-54549-9_22).
- [44] *Quantum Force 9100*. Datenblatt. Check Point Software Technologies Ltd., 2024. URL: <https://www.checkpoint.com/de/downloads/products/quantum-force-9100-datasheet.pdf>.
- [45] Eric Rescorla. *RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3*. Standard. Internet Engineering Task Force, 2018. DOI: [10.17487/RFC8446](https://doi.org/10.17487/RFC8446).
- [46] Carl Rigney. *RFC 2866: RADIUS Accounting*. Standard. Internet Engineering Task Force, 2000. DOI: [10.17487/RFC2866](https://doi.org/10.17487/RFC2866).
- [47] Carl Rigney u. a. *RFC 2865: Remote Authentication Dial In User Service (RADIUS)*. Standard. Internet Engineering Task Force, 2000. DOI: [10.17487/RFC2865](https://doi.org/10.17487/RFC2865).
- [48] Scott Rose u. a. „Zero Trust Architecture“. In: *NIST Special Publication 800-207*. 2020. DOI: [10.6028/NIST.SP.800-207](https://doi.org/10.6028/NIST.SP.800-207).
- [49] Jim Rudd, Ken Stern und Scott Isensee. „Low vs. high-fidelity prototyping debate“. In: *Interactions* 3.1 (1996), 76–85. ISSN: 1072-5520. DOI: [10.1145/223500.223514](https://doi.org/10.1145/223500.223514).
- [50] Rohith Raj S u. a. „SCAPY- A powerful interactive packet manipulation program“. In: *2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS)*. 2018, S. 1–5. DOI: [10.1109/ICNEWS.2018.8903954](https://doi.org/10.1109/ICNEWS.2018.8903954).
- [51] Ken Schwaber und Jeff Sutherland. *The Definitive Guide to Scrum: The Rules of the Game*. Anleitung. The Scrum Guide, 2020. URL: <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf>.

- [52] Jim Sermersheim. *RFC 4511: Lightweight Directory Access Protocol (LDAP): The Protocol*. Standard. Internet Engineering Task Force, 2006. DOI: [10.17487/RFC4511](https://doi.org/10.17487/RFC4511).
- [53] *Software & Licenses*. Preisliste. Deciso B.V., 2024. URL: https://shop.opnsense.com/product-categorie/software_and_licenses.
- [54] Miroslav Stampar und Mikhail Kasimov. *Maltrail*. Produktdokumentation. GitHub Inc., 2024. URL: <https://github.com/stamparm/maltrail/blob/master/README.md>.
- [55] Keith Stouffer u. a. „Guide to Industrial Control Systems (ICS) Security“. In: *NIST Special Publication 800-82 Revision 2*. 2015. DOI: [10.6028/NIST.SP.800-82r3](https://doi.org/10.6028/NIST.SP.800-82r3).
- [56] *Support*. Preisliste. Deciso B.V., 2024. URL: <https://shop.opnsense.com/product-categorie/support>.
- [57] *Suricata Features*. Webseite. Open Information Security Foundation, 2024. URL: <https://suricata.io/features>.
- [58] *The pfSense Documentation*. Produktdokumentation. Electric Sheep Fencing LLC und Rubicon Communications LLC, 2024. URL: <https://docs.netgate.com/manuals/pfsense/en/latest/the-pfsense-documentation.pdf>.
- [59] Fang Yu u. a. „Fast and memory-efficient regular expression matching for deep packet inspection“. In: *Proceedings of the 2006 ACM/IEEE Symposium on Architecture for Networking and Communications Systems*. ANCS '06. 2006, 93–102. ISBN: 1595935800. DOI: [10.1145/1185347.1185360](https://doi.org/10.1145/1185347.1185360).
- [60] *Zenarmor for Open Source Firewalls*. Webseite. Sunny Valley Cyber Security Inc., 2024. URL: <https://www.zenarmor.com/zenarmor-next-generation-firewall>.
- [61] H. Zimmermann. „OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection“. In: *IEEE Transactions on Communications* 28.4 (1980), S. 425–432. DOI: [10.1109/TCOM.1980.1094702](https://doi.org/10.1109/TCOM.1980.1094702).
- [62] „§ 8a Sicherheit in der Informationstechnik Kritischer Infrastrukturen“. In: *Gesetz über das Bundesamt für Sicherheit in der Informationstechnik (BSI-Gesetz - BSIg)*. Bundesamt für Justiz, 2021. URL: https://www.gesetze-im-internet.de/bsig_2009/_8a.html.