Wilhelm Büchner Hochschule
Hilpertstr. 31
D-64295 Darmstadt

Masterarbeit

Fachbereich Informatik

Risk-Aware SASE: Dynamische Policy-Orchestrierung für hybride Multi-Cloud-Infrastrukturen

Autor:	Kevin Dallmann
Matrikelnummer:	863160
Studiengang:	Embedded Systems and Digital Technologies
Betreuer:	Prof. Dr. Andreas U. Schmidt
Abgabetermin:	01. Februar 2026

Zusammenfassung

Die Arbeit entwickelt ein konzeptionelles Framework für einen risikobasierten Orchestrator in Secure Access Service Edge (SASE)-Architekturen, der Risikoklassen nach Bundesamt für Sicherheit in der Informationstechnik (BSI)-Standard 200-3 in technisch wirksame, herstellerunabhängige Policy-Kategorien überführt. Die Ergebnisse zeigen: Risikobasierte Schutzmaßnahmen lassen sich konsistent modellieren und in eine Orchestrierungslogik überführen, eine vollständig technische Durchsetzung scheitert jedoch in der Praxis häufig an strukturellen Einschränkungen aktueller SASE-Application Programming Interfaces (APIs). Als Konsequenz wird ein hybrides Enforcement-Modell abgeleitet, das technische und organisatorische Maßnahmen kombiniert und damit eine durchgängige Umsetzung auch bei API-Gaps ermöglicht.

Methodisch wurden fünf Risikoklassen definiert und in ein generisches Metamodell überführt, das zentrale Kategorien technischer Steuerobjekte abbildet. Eine API-Analyse der Prisma-SASE-Plattform identifiziert, welche Maßnahmen automatisierbar sind und wo Steuerobjekte fehlen. Darauf aufbauend wurde ein Systemdesign nach dem 4+1-Modell erarbeitet, das den Prozess von der Risikoadaption über die Delta-Berechnung bis zur technischen bzw. organisatorischen Durchsetzung beschreibt. Ein analytisches Performance-Modell bewertet die Time-to-Policy und bestätigt die Einhaltung der geforderten Reaktionszeit von unter 30 Sekunden.

Das Framework dient als konzeptioneller Referenzrahmen für zukünftige Implementierungen und Forschung zur Kopplung von Risikomanagement und Policy-Orchestrierung in heterogen-hybriden Umgebungen.

Schlüsselwörter: SASE, Risikoklassen, Multi-Cloud-Infrastruktur, IT-Sicherheit, Policy-Orchestrierung, Automatisierung

Abstract

This thesis develops a conceptual framework for a risk-aware orchestrator in SASE architectures that translates risk classes based on the BSI Standard 200-3 into technically effective, vendor-agnostic policy categories. The results show that risk-based protection measures can be modelled consistently and integrated into an orchestration logic; however, fully technical enforcement often fails in practice due to structural limitations of current SASE APIs. As a consequence, a hybrid enforcement model is derived that combines technical and organisational measures, enabling end-to-end implementation even in the presence of API gaps.

Methodologically, five risk classes are defined and transferred into a generic metamodel that represents key categories of technical control objects. An API analysis of the Prisma SASE platform identifies which measures are automatable and where control objects are missing. Building on this, a system design based on the 4+1 model is developed, describing the process from risk adaptation and delta computation through to technical and/or organisational enforcement. An analytical performance model evaluates the time-to-policy and confirms compliance with the required response time of under 30 seconds.

The framework serves as a conceptual reference for future implementations and research on integrating risk management and policy orchestration in heterogeneous hybrid environments.

Keywords: SASE, risk classes, multi-cloud infrastructure, IT security,
policy orchestration, automation

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Problemstellung . . . 1
- 1.2 Forschungsfragen und Hypothesen . . . 3
- 1.3 Zielsetzung der Arbeit . . . 4
- 1.4 Beitrag der Arbeit . . . 4
- 1.5 Abgrenzung der Arbeit . . . 5
- 1.6 Struktureller Aufbau der Arbeit . . . 6

2 Theoretischer Rahmen **7**
- 2.1 Grundlagen . . . 7
  - 2.1.1 Secure Access Service Edge (SASE) . . . 8
  - 2.1.2 Zero Trust und risikobasierte Zugriffskontrolle . . . 13
  - 2.1.3 Risikobewertung nach BSI-Standard 200-3 . . . 13
- 2.2 Stand der Forschung . . . 16
  - 2.2.1 SASE-Automatisierung . . . 16
  - 2.2.2 Risikobasierte Policy-Orchestrierung . . . 17

3 Methodik und Vorgehensweise **18**
- 3.1 Darstellung der Vorgehensweise . . . 18
  - 3.1.1 Das Design-Science-Research-Prozessmodell . . . 18

4 Analyse und Design **21**
- 4.1 Komponentenanalyse . . . 21
  - 4.1.1 Auswahl der Untersuchungsobjekte . . . 21
  - 4.1.2 Analyserahmen und Vorgehen . . . 24
  - 4.1.3 Analyse der Untersuchungsobjekte . . . 25
- 4.2 Konzeption und Systemdesign . . . 46
  - 4.2.1 Definition der Risikoklassen . . . 46
  - 4.2.2 Risikoklassen und erwartete Policy-Kategorien . . . 49
  - 4.2.3 Mapping der Risikoklassen auf SASE-Steuerobjekte . . . 58

5 Systemdesign nach dem 4+1 Modell **64**
- 5.1 Methodische Begründung und Aufbau des Sichtenmodells . . . 65
- 5.2 Architekturspezifikation nach dem 4+1-Modell . . . 66
  - 5.2.1 Logische Sicht (Logical View) . . . 67
  - 5.2.2 Prozesssicht (Process View) . . . 71
  - 5.2.3 Entwicklungssicht (Development View – Design-Blueprint) . . . 75
  - 5.2.4 Physische Sicht (Physical View) . . . 77
  - 5.2.5 Szenarien (Scenarios) . . . 79
- 5.3 Architektonische Evaluation . . . 82
  - 5.3.1 Theoretische Fundierung der analytischen Modellierung . . . 82
  - 5.3.2 Performance-Modellierung (Time-to-Policy) . . . 83
  - 5.3.3 Berechnung am Referenzszenario . . . 85
  - 5.3.4 Diskussion der Ergebnisse und Validierung von H2 . . . 85

6 Reflexion und Fazit **87**

Tabellenverzeichnis **91**

Abbildungsverzeichnis **92**

Abkürzungsverzeichnis **93**

Literaturverzeichnis **95**

Anhang **A**

Eidesstattliche Erklärung

1 Einleitung

Cloud-First-Strategien und Remote-Work-Modelle verändern die IT-Landschaft. Workloads laufen zunehmend in hybriden und Multi-Cloud-Umgebungen. Kostendruck verstärkt diesen Trend, da Workloads regelmäßig zwischen unterschiedlichen Infrastrukturen verschoben werden, um Ressourcenkosten zu senken. Abstraktionstechnologien wie Terraform ermöglichen solche dynamische Workload-Migrationen. Damit verschieben sich auch die Anforderungen an die Zugriffsinfrastruktur.

1.1 Problemstellung

Diese Flexibilität erzeugt neue Herausforderungen an die Sicherheit. Die Zugriffsinfrastruktur muss bei jeder Migration angepasst werden. Dabei entsteht ein Überwachungsproblem: Es ist nachzuvollziehen, welche Workloads zwischen welchen Standorten wechseln und welche Sicherheitsanforderungen dabei gelten. Diese Anforderungen basieren oft auf einer Datenklassifikation – einer Einstufung der Daten nach ihrem Schutzbedarf (z.B. hinsichtlich Vertraulichkeit, Integrität oder rechtlicher Vorgaben wie der Datenschutz-Grundverordnung (DSGVO)). Bei Daten mit einer hohen Schutzklasse (hoch klassifizierten Daten) ist eine Migration zu einem kostengünstigen, aber möglicherweise unsicheren Anbieter unzulässig.

Statisch konfigurierte Netz- und Sicherheitsrichtlinien berücksichtigen diese dynamischen Risiken nicht ausreichend. Ein Beispiel sind kurzfristige Lastspitzen in Public Clouds, bei denen ungeprüfte Verbindungen zentrale Sicherheitskomponenten wie Firewalls oder VPN-Gateways umgehen.

SASE kombiniert Netzwerk- und Sicherheitsdienste an global verteilten Service-Edges. Es fehlen jedoch Mechanismen für eine feingranulare, risikobasierte Policy-Durchsetzung während des Betriebs.

Für die Umsetzung sind Schutzmaßnahmen erforderlich, die sich dynamisch zur Laufzeit erzeugen lassen.

Damit solche Entscheidungen automatisiert und nachvollziehbar umgesetzt werden können, müssen die daraus resultierenden Sicherheitsanforderungen in vordefinierte Risikoklassen überführt werden. Aus diesen lassen sich zur Laufzeit konkrete SASE-Policies ableiten. Diese ermöglichen eine automatisierte Policy-Durchsetzung und erfüllen die unterschiedlichen Anforderungen von Workloads und Datenklassifikationen.

Derzeit fehlt eine technische Lösung, die Zero-Trust-Prinzipien mit einer operativen Policy-Orchestrierung über verschiedene Cloud-Provider und On-Premises-Systeme hinweg verbindet und die dynamische Verteilung moderner Workloads in heterogen-hybriden Umgebungen berücksichtigt. Diese Lücke spiegelt sich auch im Stand der Forschung wider (vgl. Kapitel 2.2), in dem eine solche durchgängige Orchestrierung bislang kaum konzeptionell behandelt wird.

Da etablierte Frameworks für diese spezifische Problemstellung fehlen, verfolgt diese Arbeit einen explorativen Ansatz. Ziel ist es, das bestehende Wissen zu erweitern, indem ein konzeptioneller Lösungsrahmen (Solution Framework) entwickelt wird, der die identifizierten Anforderungen adressiert und als Blaupause für zukünftige Implementierungen dienen kann. Damit leistet die Arbeit einen Beitrag zur Schließung dieser wissenschaftlichen und technischen Lücke.

1.2 Forschungsfragen und Hypothesen

Zentrale Forschungsfrage

> *Können angemessene, zur Laufzeit dynamisch erzeugbare Schutzmaßnahmen in heterogen-hybriden Umgebungen über SASE so abgebildet werden, dass sie den vordefinierten Risikoklassen (1–5) entsprechen?*

Um diese Forschungsfrage strukturiert zu untersuchen, werden drei Teilfragen (TF1–TF3) formuliert, die jeweils durch eine prüfbare Hypothese ergänzt werden.

TF	Teilfrage	Hypothese
1	Wie muss ein Metamodell konzipiert sein, um die Risikoklassen auf generische SASE-Policy-Objekte abzubilden?	H1: Die für die fünf Risikoklassen erforderlichen Schutzmaßnahmen lassen sich auf generische SASE-Policy-Objekte abbilden
2	Wie schnell kann ein risikobasierter Orchestrator einen Wechsel der Risikoklasse (z.B. Kl2 -> Kl4) technisch wirksam abbilden?	H2: Die Zeitspanne zwischen Risikoerhöhung und Policy-Aktivierung liegt unter 30 Sekunden
3	Wie muss die Architektur eines risikobasierten SASE-Orchestrators gestaltet sein, um API-Heterogenität und Funktionslücken kompensieren zu können?	H3: Ein modularer Orchestrator mit abstrahierendem Mapping-Layer kann API-Heterogenität kompensieren und konsistente Policies erzeugen, selbst wenn einzelne Hersteller keine vollständigen Steuerobjekte bereitstellen

Tabelle 1.1: Teilfragen und Hypothesen

Die 30 Sekunden in H2 orientieren sich am 1-10-60-Benchmark: Wenn Eindämmung innerhalb einer Minute wirken muss, darf die technische Durchsetzung nur einen Bruchteil davon beanspruchen[1].

Die drei Hypothesen werden im Kapitel Architektonische Evaluation 5.3 reflektiert und dort systematisch gegen die definierten Gestaltungsprinzipien sowie die empirischen Befunde der Komponentenanalyse gespiegelt.

[1] vgl. Crowdstrike, *Global Security Attitude Survey*, S.1 - S.3.

1.3 Zielsetzung der Arbeit

Ziel dieser Arbeit ist die Entwicklung eines konzeptionellen Frameworks für einen *"Risk-Aware SASE Orchestrator"*, der abstrakte Risikoklassen in konsistente, technologieneutrale SASE-Policies überführt. Grundlage bildet die Risikoanalyse-Methodik des BSI-Standards 200-3, aus der fünf Risikoklassen definiert werden, die anschließend in ein generisches, plattformunabhängiges Policy-Modell überführt werden. Das resultierende Artefakt ist eine vollständig spezifizierte Systemarchitektur, die die risikobasierte Ableitung, Strukturierung und Abbildung von Maßnahmen beschreibt.

Die Zielsetzung der Arbeit gliedert sich in fünf Teilziele:

- Entwicklung eines Klassifikationsschemas für fünf Risikoklassen (1–5) auf Basis der Methodik des BSI-Standards 200-3.
- Analyse der relevanten SASE-Komponenten im Hinblick auf ihre Steuerungs- und Management-APIs sowie Identifikation der steuerbaren Policy-Objekte.
- Entwurf eines Metamodells, das die Risikoklassen auf generische, vendorunabhängige Policy-Kategorien abbildet.
- Ableitung einer vollständigen Systemarchitektur für einen Risk-Aware-SASE-Orchestrator, bestehend aus Komponentenmodell, Schnittstellen, Datenflüssen und Risiko-zu-Policy-Logik.
- Konzeptionelle Evaluation des Artefakts anhand der in Kapitel 1.2 formulierten Hypothesen sowie der in Kapitel 3 definierten Gestaltungsprinzipien.

1.4 Beitrag der Arbeit

Der zentrale wissenschaftliche Beitrag dieser Arbeit liegt in der Konzeption eines Metamodells, das eine formale Verbindung zwischen BSI-basiertem Risikomanagement und der technischen Policy-Steuerung in SASE-Architekturen herstellt. Während der Stand der Forschung (vgl. Abschnitt 2.2) die Notwendigkeit dynamisch adaptiver Sicherheitsmechanismen herausstellt, fehlt bislang ein konsistentes, herstellerunabhängiges Modell zur systematischen Ableitung technischer Maßnahmen aus abstrakten Risikoklassen. Die vorliegende Arbeit soll diese

Lücke schließen, indem sie fünf Risikoklassen (KL 1–5) definiert und diese auf generische, komponentenübergreifende SASE-Policy-Kategorien abbildet.

Darüber hinaus besteht ein wesentlicher Beitrag in der analytischen Evaluation dieses Metamodells durch eine detaillierte Untersuchung der API-Strukturen der Prisma-SASE-Plattform. Diese Analyse zeigt, welche Policy-Objekte zur Laufzeit tatsächlich steuerbar sind und wo strukturelle Grenzen bestehen. Auf dieser Grundlage entwickelt die Arbeit ein konsistentes Systemdesign eines „Risk-Aware SASE Orchestrators“, das als konzeptionelles DSR-Artefakt dient. Das Artefakt demonstriert, wie risikobasierte Entscheidungen technisch strukturiert, abgebildet und orchestriert werden können, selbst wenn eine vollständige Implementierung aufgrund externer Systemrestriktionen nicht realisierbar ist.

Damit leistet die Arbeit sowohl einen theoretischen Beitrag durch das Metamodell als auch einen praktischen Beitrag durch das darauf aufbauende Systemdesign. Dieses kann als Referenzrahmen für zukünftige Implementierungen oder weitergehende prototypische Entwicklungen dienen.

1.5 Abgrenzung der Arbeit

Die Arbeit konzentriert sich auf die konzeptionelle Entwicklung eines risikobasierten Orchestrierungsmodells für SASE-Architekturen. Im Mittelpunkt stehen die Definition der Risikoklassen, die Ableitung generischer Policy-Kategorien sowie das darauf aufbauende Metamodell und Systemdesign des "Risk-Aware-SASE-Orchestrators".

Nicht Gegenstand der Arbeit ist die vollständige technische Implementierung oder der Betrieb eines funktionsfähigen Orchestrators auf Basis produktiver SASE-Plattformen. Aufgrund fehlender Zugänge zu den relevanten produktiven API-Endpunkten erfolgt keine empirische Performancemessung und keine prototypische Laufzeitvalidierung der Hypothesen. Die Evaluation beschränkt sich daher auf eine analytische Beurteilung des Metamodells und der Systemarchitektur anhand der in Kapitel 3 definierten Gestaltungsprinzipien sowie der Ergebnisse der API-Analyse im Abschnitt 4.1.

Ebenso nicht im Fokus steht ein systematischer Vergleich verschiedener Anbieter. Die Analyse basiert ausschließlich auf der öffentlich dokumentierten API der Prisma-SASE-Plattform

von Palo Alto Networks. Aussagen zu anderen Herstellern sind daher nicht Gegenstand der Untersuchung und können nur indirekt über theoretische Überlegungen abgeleitet werden.

Die Arbeit verfolgt damit einen konzeptionellen Anspruch: Sie entwickelt ein konsistentes Modell und ein vollständig beschriebenes Systemdesign, ohne eine produktive Implementierung oder performanzbasierte Validierung bereitzustellen.

1.6 Struktureller Aufbau der Arbeit

Der Aufbau der Arbeit folgt der Logik des Design-Science-Research-Prozesses und führt schrittweise von der Problemstellung über die Entwicklung des Artefakts hin zu dessen analytischer Evaluation.

Die Arbeit ist wie folgt strukturiert:

- **Kapitel 2 – Theoretischer Rahmen:** Vermittlung des theoretischen Rahmens und Beleuchtung des aktuellen Stands der Forschung zu SASE-Automatisierung und risikobasierter Policy-Orchestrierung.
- **Kapitel 3 – Methodik und Vorgehensweise:** Begründung der wissenschaftlichen Vorgehensweise und detaillierte Vorstellung des Design Science Research (DSR)-Prozessmodells, das die strukturgebende Klammer für die gesamte Arbeit bildet.
- **Kapitel 4 – Analyse und Design:** Beginn der Artefakt-Entwicklung (DSR-Phase 3) mit der konzeptionellen Hauptarbeit. Dies umfasst eine detaillierte Komponentenanalyse von SASE-APIs und die Entwicklung des Systemdesigns, welches BSI-Risikoklassen auf technische Steuerobjekte abbildet.
- **Kapitel 5 – Systemdesign nach dem 4+1 Modell:** Weiterführung des Metamodells zu einer Systemarchitektur des "Risk-Aware-SASE-Orchestrators". Das Kapitel beschreibt Zielsetzung, Aufbau, Komponenten, Schnittstellen und Funktionsweise des Artefakts und enthält eine exemplarische Demonstration in Form eines modellbasierten Szenarios sowie eine analytische Evaluation des Designs.
- **Kapitel 6 – Reflexion und Fazit:** Kritische Diskussion der Gesamtergebnisse, Einordnung der Erkenntnisse in den Forschungskontext, Ableitung praktischer Empfehlungen und Identifikation künftiger Forschungsbedarfe.

2 Theoretischer Rahmen

Kapitel 2 beschreibt die theoretischen Grundlagen zu SASE. Der Abschnitt erläutert Aufbau und Funktionsweise der SASE-Architektur, beschreibt Zero Trust mit risikobasierter Zugriffskontrolle und führt in die dynamische Policy-Orchestrierung ein. Zwei Unterkapitel gliedern den Inhalt: Abschnitt 2.1 beschreibt SASE sowie die Komponenten im Detail. Abschnitt 2.2 stellt den Stand der Forschung zu SASE-Automatisierung und risikobasierter Policy-Orchestrierung dar.

2.1 Grundlagen

Zu Beginn des Kapitels stehen die technischen und konzeptionellen Grundlagen, die das Fundament dieser Arbeit bilden. Der Abschnitt beschreibt zentrale Eigenschaften moderner Netzwerk- und Sicherheitsarchitekturen im Kontext SASE und erläutert deren Bedeutung für das Forschungsvorhaben. Ein kurzer historischer Überblick ordnet die Begriffe SASE und Zero-Trust-Prinzip ein. Anschließend folgt die Darstellung der vom BSI vorgegebener Anweisung eine Risikobewertung durchzuführen. Die Kombination dieser Themen bildet einen Referenzrahmen, der die in den folgenden Kapiteln dargestellten Inhalte einordnet und die Argumentation strukturiert. Das Kapitel «Grundlagen» legt die Basis, um die später beschriebenen Lösungsansätze technisch und methodisch zu verorten.

2.1.1 Secure Access Service Edge (SASE)

SASE bezeichnet einen von Gartner 2019 eingeführten Paradigmenwechsel, bei dem Netzwerk- und Sicherheitsfunktionen zu einem cloudbasierten Dienst zusammengeführt werden[1]. Getrieben durch Mobilität, Cloud-Migration und den Bedarf an niedriger Latenz ersetzt SASE unteranderem traditionelle Campus-Firewalls und Multiprotocol Label Switching (MPLS)-Backbones durch ein identitäts- und risikobasiertes Policy-Framework. Tabelle 2.1 fasst die fünf zentralen Bausteine zusammen. Die anschließende Darstellung erläutert sie im Detail. Abbildung 2.1 zeigt ihre Einordnung im Gesamtkonzept.

Komponente	Hauptaufgabe im SASE-Kontext
Software-Defined Wide Area Network (SD-WAN)	Verkehrsleitung und Pfad optimierung
Secure Web Gateway (SWG)	Web-Content-Filter, Malware-Analyse
Cloud Access Security Broker (CASB)	Cloud-Transparenz, Richtliniendurchsetzung
Firewall as a Service (FWaaS)	Layer-3/4-Schutz als Cloud-Service
Zero Trust Network Access (ZTNA)	Identitäts- und Kontextbasierter Zugriff
Next Generation Firewall (NGFW)	Ergänzung um IDS / IPS Funktion

Tabelle 2.1: Kernkomponenten eines SASE-Stacks[2]

[1] vgl. Macdonald, *The Future of Network Security Is in the Cloud: Introducing the Secure Access Service Edge*.
[2] vgl. Macdonald, *The Future of Network Security Is in the Cloud: Introducing the Secure Access Service Edge*

Wie in Abbildung 2.1 dargestellt, steht im Zentrum der Grafik ein globales Netzwerk von Point of Presence (PoP)s. Dieses Netzwerk bindet alle Standorte und Cloud-Ressourcen an. Jeder PoP enthält die Kernkomponenten des SASE-Stacks. SD-WAN leitet den Datenverkehr optimiert und mit geringer Latenz in das PoP-Netz. Dort übernimmt der SWG Web-Content-Filterung und Malware-Analyse. Danach setzt der CASB Richtlinien bei Cloud-Applikationen durch und stellt Transparenz her. Anschließend führt FWaaS auf OSI-Layer 3/4 Paketinspektion durch, während NGFW auf OSI-Layer 7 Anwendungsinspektion ausführt. Zum Schluss prüft ZTNA die Identität, Gerätekontext und Risikoprofil und gewährt nur den minimal erforderlichen Anwendungszugang. Diese Abfolge bildet eine cloudnative und skalierbare Sicherheits- und Netzwerklösung mit Schutz gegen moderne Bedrohungen.

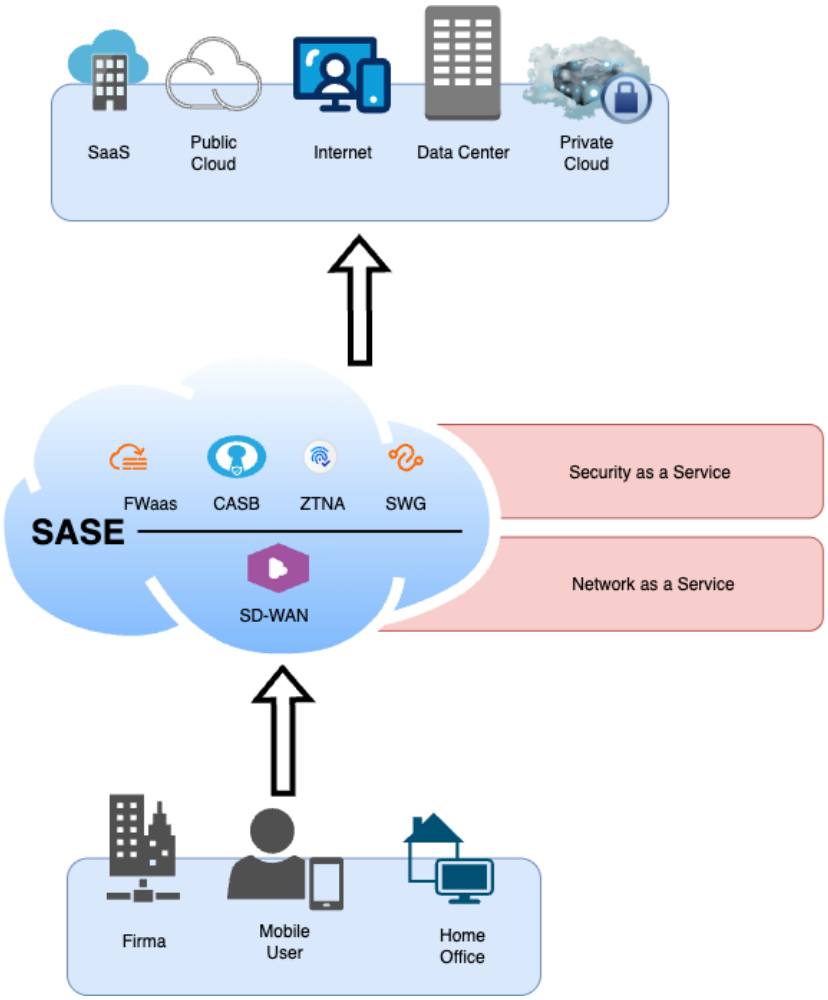

Abbildung 2.1: SASE-Architektur mit integrierten Sicherheits- und Netzwerkdiensten

Software-Defined Wide Area Network (SD-WAN)

Ein SD-WAN ist ein virtueller Netzwerküberbau, der mehrere Transportmedien wie MPLS, Breitband-Internet oder 5G logisch bündelt und zentral steuert[3]. Im Kontext von SASE bildet SD-WAN die Grundlage für die Datenanbindung. Es leitet Datenströme zu dem jeweils nächstgelegenen Security-PoP, wo darüberliegende Dienste wirken. Die Abstraktion der Underlay-Links ermöglicht die Festlegung von Regeln für Pfadauswahl, Quality of Service und Verschlüsselung unabhängig von der physikalischen Topologie. Ein Beispiel ist die Lenkung von Echtzeitverkehr über eine Verbindung mit niedriger Latenz, während weniger kritische Backups kostengünstigere Leitungen nutzen. Im Unterschied zu klassischen Wide Area Network (WAN)-Architekturen entfällt die manuelle Konfiguration proprietärer Router an jedem Standort. Tunnel und Regelwerke werden zentral bereitgestellt. Die Integration mit SASE ermöglicht die Anwendung bedingungsabhängiger Sicherheitskontrollen entlang desselben Datenpfads.

Secure Web Gateway (SWG)

Ein SWG kontrolliert Web-Verkehr und prüft HTTP(S)-Anfragen auf Schadsoftware, unerwünschte Inhalte und Verstöße gegen Richtlinien[4]. Innerhalb einer SASE-Plattform läuft der SWG als Cloud-nativer Microservice an jedem PoP. Dieser entschlüsselt eingehende Pakete, bewertet sie mithilfe von URL-Filter-Datenbanken, Anti-Malware-Scanning und Data-Loss-Prevention-Engines und blockiert oder verändert sie bei Bedarf[5]. Im Unterschied zu einem traditionellen Proxy kombiniert er diese Funktionen mit weiteren Kontrollen wie CASB oder ZTNA über ein zentrales Policy-Backend. Dadurch erhält dieselbe Identität restriktive Content-Filter für Social Media, während geschäftskritische SaaS-Anwendungen Vorrang erhalten. Dieser Ansatz reduziert die Angriffsfläche und ermöglicht konsistente Prüf- und Berichtsfunktionen über den gesamten Web-Verkehr. Separate Proxy-Chains entfallen.

[3] vgl. MEF Forum, *MEF Standard MEF 70.2 SD-WAN Service Attributes and Service Framework*.
[4] vgl. Lawrence Miller, *Secure Access Service Edge (SASE) For Dummies®, Palo Alto Networks 2nd Special Edition*, S.16, 38, 61.
[5] vgl. Sangfor Technologies, *What is a Secure Web Gateway (SWG)? A Comprehensive Guide to SWG*.

Cloud Access Security Broker (CASB)

Ein CASB vermittelt zwischen Endnutzern und Cloud-Diensten, um Transparenz, Compliance und Bedrohungsschutz in Software as a Service (SaaS)-, Platform as a Service (PaaS)- und Infrastructure as a Service (IaaS)-Umgebungen sicherzustellen[6]. Im SASE-Kontext ergänzt der CASB die Web-Sicherheit um APIs mit direktem Zugriff auf Anwendungsobjekte. Er erkennt, wenn sensible Kundeninformationen in eine öffentlich zugängliche Collaboration-Freigabe gelangen, und erzwingt Verschlüsselung oder Quarantäne. Durch fortlaufendes Shadow-IT-Scannen – (der Erkennung von Cloud-Diensten, die ohne Genehmigung der IT genutzt werden) – erstellt er einen Katalog genutzter Cloud-Dienste, bewertet deren Risiko und blockiert Anwendungen anhand festgelegter Schwellenwerte. Im Unterschied zu einem SWG arbeitet der CASB nicht nur inline (also direkt im aktiven Datenstrom), sondern nutzt auch API-basierte Post-Scan-Prozesse, um Verstöße in gespeicherten Daten zu erkennen. Die Integration in das zentrale SASE ermöglicht konsistente Identitäts- und Kontextprüfungen über alle Zugriffswege.

Firewall as a Service (FWaaS)

FWaaS verlagert die Paketprüfung und Richtlinienkontrolle von physischen Geräten in eine elastische Cloud-Infrastruktur mit Mandantenisolation – (also der strikten logischen Trennung der Daten und Konfigurationen verschiedener Kunden (Mandanten) auf derselben Infrastruktur) – und horizontaler Skalierung[7]. Innerhalb eines SASE-PoP dient FWaaS als Kontrollschicht für Layer-3/4-Traffic außerhalb von Web-Ports. Funktionen wie Stateful Inspection, Intrusion Prevention und Geo-IP-Filtering laufen auf verteilten Engines, um Durchsatzspitzen abzufangen, ohne Bandbreitenlimits lokaler Geräte zu erreichen. Gegenüber Next-Generation-Firewalls bietet FWaaS eine global einheitliche Richtlinienverwaltung, bei der Änderungen in Sekunden an alle Kontrollpunkte verteilt werden. Ein Beispiel ist die Segmentierung von Produktions-OT-Netzen, die zuvor nur in Werks-Firewalls möglich war. Die Kopplung an SD-WAN und ZTNA ermöglicht adaptive Regeln, die Verbindungsweg und Benutzer-Risiko berücksichtigen. Dies stellt eine einheitliche Sicherheitslage über Ländergrenzen hinweg sicher.

[6] vgl. Lawrence Miller, *Secure Access Service Edge (SASE) For Dummies®, Palo Alto Networks 2nd Special Edition*, S.40, 61, 67.

[7] vgl. Lawrence Miller, *Secure Access Service Edge (SASE) For Dummies®, Palo Alto Networks 2nd Special Edition*, S.44-45.

Zero Trust Network Access (ZTNA)

ZTNA ersetzt das klassische Virtual Private Network (VPN)-Konzept, indem es implizites Vertrauen innerhalb und außerhalb des Unternehmensnetzes entfernt[8]. Eine Verbindung entsteht erst nach positiver Identitätsprüfung, Gerätebewertung und Kontextanalyse. Jede Sitzung erhält nur die für den definierten Zweck erforderlichen Zugriffsrechte. In der SASE-Architektur arbeitet ZTNA als Policy-Engine auf Anwendungsebene. Nachdem SD-WAN den Traffic in den PoP leitet, prüft das System, ob die erforderlichen Rollen vorliegen, das Endgerät ein aktuelles Patch-Level besitzt und das Risikoniveau unter einem festgelegten Schwellenwert liegt. Im Unterschied zu Remote-Access-VPNs legt ZTNA keinen vollständigen Netzwerklayer offen, sondern nur einen einzelnen TCP-Port. Eine Kombination mit kontinuierlicher Authentifizierung ermöglicht die Anpassung der Zugriffsrechte während einer Sitzung, wenn sich Rahmenbedingungen ändern.

Next-Generation Firewall (NGFW)

NGFWs erweitern klassische Firewalls durch zusätzliche Inspektionsverfahren, die über die Paketfilterung auf OSI-Layer 3 und 4 hinausgehen[9]. Traditionelle Firewalls werten ausschließlich Header-Informationen wie IP-Adressen, Ports und Protokolle aus. NGFWs analysieren dagegen mit Deep Packet Inspection (DPI) den vollständigen Paketinhalt bis zur Anwendungsschicht. In der SASE-Architektur arbeitet die NGFW als anwendungsbewusster Sicherheitsfilter. Nach der Verkehrsweiterleitung durch SD-WAN kontrolliert sie den Datenverkehr granular auf Basis von Anwendungsidentität, Benutzerkontext und Inhaltsanalyse. Die Integration von Intrusion Prevention System (IPS) und Intrusion Detection System (IDS) ermöglicht die Analyse verdächtiger Aktivitäten in Echtzeit durch Signaturerkennung und verhaltensbasierte Anomalieerkennung. Threat-Intelligence-Feeds aktualisieren die NGFW fortlaufend und passen sie an neue Bedrohungsmuster an. Auf diese Weise verhindert sie Angriffe, die sich der port- und protokollbasierten Filterung entziehen.

[8] vgl. Lawrence Miller, *Secure Access Service Edge (SASE) For Dummies®, Palo Alto Networks 2nd Special Edition*, S.35-37.

[9] vgl. Lawrence. Miller, *Next-Generation Firewalls For Dummies®*.

2.1.2 Zero Trust und risikobasierte Zugriffskontrolle

Das Zero-Trust-Modell geht davon aus, dass kein Akteur, Gerät oder Netzwerksegment grundsätzlich vertrauenswürdig ist[10]. Jeder Zugriff wird einzeln geprüft und fortlaufend überwacht. Risikobasierte Zugriffskontrolle ergänzt dieses Prinzip um Kontextparameter wie Standort, Zeit, Gerätezustand oder Sensitivität der angeforderten Daten. Eine identische Rolle kann je nach Sicherheitslage unterschiedliche Berechtigungen erhalten. Ein Entwickler im Büro klont beispielsweise Quellcode-Repositories, während derselbe Benutzer über ein unbekanntes WLAN zunächst eine MFA-Prüfung durchläuft. Innerhalb von SASE erfolgt die Bewertung in Echtzeit durch die Zusammenführung von Telemetrie aus ZTNA, CASB und Endpunkt-Agenten. Die ermittelte Risikostufe legt fest, ob ein Zugriff erfolgt, eingeschränkt wird oder entfällt. Dadurch verbindet sich Identitäts- und Zugriffsmanagement mit fortlaufender Bedrohungsanalyse zu einem geschlossenen Regelkreis. Dies erhöht die Geschwindigkeit der Reaktion auf neue Angriffe und verringert mögliche Eskalationspfade.

2.1.3 Risikobewertung nach BSI-Standard 200-3

Eine zentrale Komponente des IT-Grundschutzes ist die systematische Analyse und Behandlung von Informationssicherheitsrisiken. Um die im BSI-Standard gewählte Methodik wissenschaftlich einzuordnen, muss zunächst der zugrunde liegende Risikobegriff geklärt werden. In der IT-Sicherheit hat sich eine spezifische Sichtweise etabliert, die sich vom allgemeinen, chancenorientierten Risikobegriff abgrenzt. Man spricht hierbei vom sogenannten „Downside-Risiko“, welches sich ausschließlich auf negative Zielabweichungen und deren Verlustpotenziale konzentriert[11].

Theoretisch lässt sich die Entstehung eines solchen Risikos durch ein Kausalmodell beschreiben: Ein Risiko manifestiert sich erst dann, wenn eine *Bedrohung* auf eine vorhandene *Schwachstelle* (Vulnerability) an einem schützenswerten *Risikoobjekt* (Asset) trifft. Fehlt die Schwachstelle, läuft die Bedrohung ins Leere; fehlt die Bedrohung, bleibt die Schwachstelle latent, verursacht aber keinen akuten Schaden[12]. Der BSI-Standard 200-3 operationalisiert dieses theoretische Wirkungsgefüge durch ein qualitatives Verfahren. Dieser Ansatz trägt dem Umstand

[10] vgl. Garbis und Chapman, *Zero Trust Sicherheit: Ein Leitfaden für Unternehmen*, S.207-214.
[11] vgl. Königs, *IT-Risikomanagement mit System*, S. 12.
[12] vgl. Königs, *IT-Risikomanagement mit System*, S. 14–15.

Rechnung, dass in der IT-Sicherheit oft keine verlässlichen statistischen Datenbasen existieren, um Eintrittswahrscheinlichkeiten exakt zu quantifizieren[13].

Schritt 1: Schutzziele und Schadenshöhe definieren

Der Ausgangspunkt der Bewertung ist die Analyse des potenziellen Schadensausmaßes (Impact). Dies erfolgt anhand der drei klassischen Schutzziele der Informationssicherheit, die als normative Vorgaben für den Soll-Zustand des Systems dienen[14]:

- **Verfügbarkeit:** Die Sicherstellung, dass Systeme und Informationen bei Bedarf nutzbar sind.
- **Integrität:** Die Gewährleistung der Unversehrtheit und Korrektheit von Daten.
- **Vertraulichkeit:** Die Beschränkung des Zugriffs auf autorisierte Instanzen.

Da sich Schäden in der IT – insbesondere Reputationsverluste oder Vertrauenseinbußen – oft schwer monetarisieren lassen, empfiehlt die Literatur den Einsatz von Ordinalskalen. Anstatt kardinaler Werte (z.B. Euro-Beträge) werden qualitative Rangfolgen gebildet[15]. Der BSI-Standard nutzt hierfür Kategorien wie „vernachlässigbar“ bis „existenzbedrohend“.

Schritt 2: Eintrittshäufigkeit kategorisieren

Analog zur Schadenshöhe wird auch die Eintrittswahrscheinlichkeit nicht stochastisch exakt, sondern über qualitative Kategorien (z.B. „selten“ bis „sehr häufig“) abgebildet. Dies dient der Reduktion von Komplexität und Scheingenauigkeiten in einem Umfeld, das von hoher Ungewissheit geprägt ist.

Schritt 3: Risiko in der Matrix ermitteln

Ein wesentliches Merkmal des BSI-Standards ist der Verzicht auf die klassische mathematische Risikoberechnung ($Risiko = Eintrittswahrscheinlichkeit \cdot Schadensausmas$). Königs weist darauf hin, dass diese multiplikative Verknüpfung bei sogenannten „Katastrophenrisiken“ zu

[13] vgl. Königs, *IT-Risikomanagement mit System*, S. 19.
[14] vgl. Königs, *IT-Risikomanagement mit System*, S. 13.
[15] vgl. Königs, *IT-Risikomanagement mit System*, S. 23.

Fehleinschätzungen führen kann[16]. Da solche Ereignisse extrem selten auftreten (Wahrscheinlichkeit ≈ 0), würde das rechnerische Produkt ein minimales Risiko ergeben, was das tatsächliche Existenzbedrohungspotenzial bei Eintritt verschleiern würde[17].

Um diese methodische Schwäche zu umgehen, nutzt der Standard eine Risikomatrix als heuristisches Instrument:

$$\text{Risiko} = f(\text{Schadenshöhe}, \text{Eintrittshäufigkeit})$$

Die Matrix fungiert hierbei als Abbildungsfunktion, die es erlaubt, die spezifische Risikowahrnehmung der Organisation direkt in das Bewertungsmodell zu integrieren. So lässt sich beispielsweise definieren, dass ein Ereignis mit „existenzbedrohendem“ Schaden unabhängig von seiner Eintrittswahrscheinlichkeit immer als „hohes Risiko“ eingestuft wird. Dies spiegelt eine risiko-aversive Haltung wider, die für Sicherheitskonzepte typisch ist[18].

Ergebnis: Ein klassifiziertes Risiko

Das Resultat ist eine priorisierte Risikoklassifizierung (z.B. „hoch“, „mittel“, „gering“). Diese dient als Steuerungsinstrument, um Sicherheitsmaßnahmen dort zu fokussieren, wo der Handlungsbedarf aus Sicht der Organisationsziele am dringendsten ist.

[16] vgl. Königs, *IT-Risikomanagement mit System*, S. 5–10.
[17] vgl. Königs, *IT-Risikomanagement mit System*, S. 19–20.
[18] vgl. Königs, *IT-Risikomanagement mit System*, S. 21.

2.2 Stand der Forschung

Der Abschnitt beschreibt den aktuellen Stand zu SASE-Automatisierung und risikobasierter Policy-Orchestrierung. Die Literatur dokumentiert ein zunehmendes Interesse an integrierten SASE-Frameworks. Eine durchgängige End-to-End-Orchestrierung über Anbieter- und Technologiegrenzen existiert nicht.

2.2.1 SASE-Automatisierung

Die Automatisierung von SASE-Architekturen bildet ein zentrales Forschungsfeld. Treiber sind die Anforderungen verteilter Multi-Cloud-Umgebungen und die Komplexität hybrider Arbeitsmodelle. Velayutham untersucht den Einsatz von SASE zur Absicherung von Remote-Arbeit und zeigt, dass perimeterbasierte Ansätze an Wirksamkeit verlieren, wenn hybride Arbeitsmodelle vorherrschen. Er beschreibt die flexible Kombination von Funktionen wie ZTNA, Secure Web Gateways und Firewall-as-a-Service[19].

Die Global TD-LTE Initiative (GTI) entwickelt ein Framework, das Netzwerk- und Sicherheitsfunktionen in einer Architektur vereint und Multi-Vendor-Umgebungen konsistent verwaltet[20]. Grand View Research prognostiziert ein Wachstum des globalen SASE-Markts von 3,82 Mrd. USD im Jahr 2024 auf 17,22 Mrd. USD im Jahr 2030.[21]

Hassan et al. beschreibt vier Implementierungsansätze: Abstraktionsschichten, Security Service Meshes, zentrale Kontrollsysteme und schrittweise Rollouts[22]. Das Metro Ethernet Forum (MEF) definiert Composite und Atomic Policies als Basiskomponenten für SASE-Services[23]. Die automatische Generierung und Anpassung von Richtlinien in hochdynamischen Umgebungen bleibt ungeklärt.

[19] vgl. Velayutham, „Secure Access Service Edge (SASE) Framework in Enhancing Security for Remote Workers and Its Adaptability to Hybrid Workforces in the Post-Pandemic Workplace Environment“, S. 1–3.
[20] vgl. GTI Group, *GTI Orchestration Framework for Secure Access Service Edge (SASE)*, S. 8–12.
[21] Grand View Research, *Secure Access Service Edge Market Size, Share Report 2030*, vgl.
[22] vgl. Hassan, „Enterprise Deployment Challenges of SASE: A Multi-Cloud Approach“, S. 2–4.
[23] vgl. MEF Forum, *MEF 117 - SASE Service Attributes and Service Framework*, S. 17–19.

2.2.2 Risikobasierte Policy-Orchestrierung

Die risikobasierte Policy-Orchestrierung erweitert Automatisierungsansätze um dynamische Sicherheitsrichtlinien während des Betriebs. Alsaadi et al. schlagen ein Neuro-Fuzzy-Modell vor, das Kontextdaten wie Nutzerverhalten und Ressourcensensitivität auswertet, um statische Richtlinien zu ersetzen[24]. In Internet of Things (IoT)-Umgebungen erweitern Atlam und Wills das Role-Based Access Control (RBAC)-Modell um kontinuierliche Risikobewertungen auf Basis von Nutzer- und Umgebungsdaten[25].

Rahmenwerke definieren grundlegende Orchestrierungsprinzipien. Die National Security Agency trennt zwischen Policy Decision Points und Enforcement Points[26]. Horcas et al. verbinden Autonomic Computing mit Aspect-Oriented Programming für Echtzeitanpassungen[27].

Fernández Saura nutzt Large Language Models zur Übersetzung von High-Level-Richtlinien in API-Aufrufe[28]. Blockchain-gestützte Compliance-Modelle[29] und Bayesian Networks für proaktive Risikobewertung zeigen die Spannbreite dynamischer Orchestrierungskonzepte[30]. Antonakakis et al. identifizieren, dass nur 8,9 Prozent der Studien dynamische Ansätze untersuchen, was die Anpassungsfähigkeit einschränkt[31].

Zusammengefasst kann gesagt werden: Es bestehen Lücken bei der Integration von SASE-Automatisierung und risikobasierten Policy-Frameworks, der standardisierten Einstufung von Risikoklassen in heterogenen Umgebungen sowie der End-to-End-Orchestrierung über Anbieter- und Technologiegrenzen. Diese Forschungslücken bilden die Grundlage für den in dieser Arbeit entwickelte Lösungsansatz.

[24] vgl. Alsaadi, Alasadi und Fadhil, „Efficient NFS Model for Risk Estimation in a Risk-Based Access Control Model for IoT Applications“, S. 2–5.
[25] vgl. Atlam u. a., „Developing an Adaptive Risk-Based Access Control Model for the Internet of Things“, S. 420–432.
[26] vgl. National Security Agency, *Advancing Zero Trust Maturity Throughout the Automation and Orchestration Pillar*, S. 5–7.
[27] vgl. Horcas, Pinto und Fuentes, „Runtime Enforcement of Dynamic Security Policies“, S. 1–3.
[28] vgl. Fernández Saura u. a., „On Automating Security Policies with Contemporary LLMs“, S. 2–4.
[29] vgl. Alevizos, „Automated cybersecurity compliance and threat response using AI, blockchain and smart contracts“, S. 3–6.
[30] vgl. Mahmoud, „A Novel Proactive and Dynamic Cyber Risk Assessment Methodology“, S. 1–3.
[31] vgl. Antonakakis u. a., „A comprehensive survey of manual and dynamic approaches for taxonomy generation in cybersecurity“, S. 2–4.

3 Methodik und Vorgehensweise

Das Kapitel legt das wissenschaftliche Fundament der Arbeit. Es beschreibt die methodische Vorgehensweise und begründet die Wahl des DSR-Paradigmas[1]. Es wird dargelegt, wie dieser Ansatz die systematische Konstruktion und Evaluation des Risk-Aware-SASE-Frameworks leitet. Die hier vorgestellten Phasen des DSR-Prozessmodells – von der Problemidentifikation bis zur Evaluation – bilden die strukturelle Klammer für alle folgenden Kapitel dieser Arbeit.

3.1 Darstellung der Vorgehensweise

Die Methoden richten sich am Ziel aus, ein praxisrelevantes Artefakt systematisch zu entwickeln und dessen Wirksamkeit empirisch zu prüfen. Das Paradigma des DSR[2] trägt die Konstruktion des Frameworks und die theoriegeleitete Auswertung der Ergebnisse.

3.1.1 Das Design-Science-Research-Prozessmodell

Das DSR ist ein Forschungsparadigma, das primär auf die Konstruktion und Evaluation von innovativen Artefakten abzielt, um spezifische Probleme der Praxis zu lösen[3]. Im Gegensatz zu rein erklärender Forschung, die 'das was ist' beschreibt, fokussiert DSR auf die Gestaltung und Untersuchung von 'Was könnte sein'. Das Ziel ist, durch die Entwicklung eines Lösungsartefakts – in dieser Arbeit ein konzeptionelles- Framework und ein-Proof of Concept (PoC) – einen validen Beitrag zur Wissensbasis zu leisten.

Dieses Modell[4] wurde gewählt, da es einen klaren, sechsstufigen und iterativen Leitfaden von der Problemidentifikation bis zur Kommunikation der Ergebnisse bietet. Es eignet sich daher ideal, um die Entwicklung des "Risk-Aware SASE Frameworks"(das Artefakt) systematisch und nachvollziehbar zu strukturieren.

[1] vgl. Peffers u. a., „A design science research methodology for information systems research“.
[2] vgl. Peffers u. a., „A design science research methodology for information systems research“.
[3] vgl. Hevner u. a., „Design Science in Information Systems Research“.
[4] vgl. Peffers u. a., „A design science research methodology for information systems research“.

Die konkrete Anwendung dieses Modells in der Arbeit gestaltet sich wie folgt:

1. **Problemidentifikation und Motivation:** Abschnitt 1.1 identifiziert die fehlende Möglichkeit, Schutzmaßnahmen in SASE-Architekturen dynamisch an veränderte Risikosituationen anzupassen. Diese Lücke bildet die Grundlage für die Entwicklung eines risikobasierten Orchestrierungsansatzes.
2. **Definition der Ziele für eine Lösung:** Abschnitt 1.3 leitet die Zielsetzungen ab und definiert das Artefakt als konzeptionelles Framework zur systematischen Ableitung und Strukturierung risikobasierter SASE-Policies.
3. **Design und Entwicklung:** Kapitel 4 entwickelt das Metamodell zur Abbildung der Risikoklassen auf generische Policy-Kategorien. Kapitel 5 führt dieses Metamodell zu einer vollständigen Systemarchitektur des Orchestrators weiter, einschließlich Komponenten, Schnittstellen und Datenflüssen.
4. **Demonstration:** Die Funktionsweise des Artefakts wird in Kapitel 5 anhand eines exemplarischen Szenarios demonstriert. Die Demonstration erfolgt modellbasiert im Rahmen der Architekturspezifikation und zeigt die theoretische Anwendung der Architektur bei einer Risikoeskalation.
5. **Evaluation:** Die analytische Evaluation in Kapitel 5 bewertet das Systemdesign anhand der in Kapitel 3 definierten Gestaltungsprinzipien und reflektiert die Hypothesen H1–H3 mittels eines Performance-Modells im Hinblick auf die konzeptionelle Tragfähigkeit des Artefakts.
6. **Kommunikation:** Die vorliegende Arbeit dokumentiert Problemstellung, methodisches Vorgehen, Artefaktentwicklung sowie die gewonnenen Erkenntnisse und ordnet diese in Kapitel 6 in den wissenschaftlichen Kontext ein.

Gestaltungsprinzipien (Design Rationale)

Die Entwicklung des Artefakts (DSR-Schritt 3) orientiert sich an fünf zentralen Gestaltungsprinzipien, die aus der Problemstellung (Abschnitt 1.1) und den Zielsetzungen (Abschnitt 1.3) abgeleitet wurden. Diese Prinzipien strukturieren sowohl das Metamodell als auch die daraus entwickelte Systemarchitektur:

- **Anbieterunabhängigkeit:** Das Artefakt muss in heterogenen Multi-Cloud- und Hybridumgebungen einsetzbar sein. Daher basiert das Metamodell auf generischen Policy-Kategorien, die unabhängig von proprietären SASE-Implementierungen formuliert sind.
- **Konsistenz und Rücksetzbarkeit:** Die Architektur sieht Mechanismen vor, die sicherstellen, dass risikobasierte Policy-Änderungen konsistent, atomar und jederzeit reversibel modelliert werden können, um instabile Zwischenzustände zu vermeiden.
- **Minimale *Time-to-Policy*:** Obwohl das Artefakt konzeptionell ist, wird das Systemdesign so ausgelegt, dass zwischen Risikoereignis und Policy-Ableitung möglichst geringe Latenzen entstehen. Dieses Prinzip adressiert die in Abschnitt 1.1 identifizierte Schwäche statischer Policies.
- **Nachvollziehbarkeit und Auditierbarkeit:** Alle modellierten Änderungen müssen eindeutig abbildbar und protokollierbar sein. Die Architektur umfasst daher definierte Logging- und Traceability-Punkte für die gesamte Orchestrierungskette.
- **Abbildung externer Vorgaben:** Das Systemdesign berücksichtigt regulatorische und organisatorische Anforderungen (z.B. BSI-Vorgaben, Schutzbedarfsdefinitionen), indem diese als Eingangsgrößen in die Policy-Ableitung integriert sind.

Die nachfolgenden Kapitel setzen diese Gestaltungsprinzipien systematisch um. Kapitel 4 entwickelt darauf basierend das Metamodell und die Mapping-Logik, während Kapitel 5 die daraus abgeleitete Systemarchitektur ausformuliert und ihre Funktionsweise anhand eines Szenarios demonstriert. Die anschließende analytische Evaluation bewertet das Artefakt abschließend im Hinblick auf die hier definierten Gestaltungsprinzipien.

4 Analyse und Design

Aufbauend auf dem in Kapitel 3 eingeführten DSR-Prozessmodell beschreibt dieses Kapitel die Phase „Design und Entwicklung“ (DSR-Schritt 3). Es bildet die konzeptionelle Kernphase der Arbeit und legt die Grundlage für das in Kapitel 5 entwickelte Artefakt.

Das Kapitel gliedert sich in zwei Hauptteile. Zunächst führt die Komponentenanalyse (Abschnitt 4.1) eine systematische Untersuchung existierender SASE-Lösungen durch, um relevante API-Schnittstellen und Steuerobjekte zu identifizieren. Darauf aufbauend entwickelt die Konzeption und das Systemdesign (Abschnitt 4.2) das Metamodell. Dies umfasst die Definition der Risikoklassen nach BSI-Standard und die Entwicklung der zentralen Mapping-Logik, welche die Risikoklassen auf die zuvor identifizierten SASE-Steuerobjekte abbildet.

4.1 Komponentenanalyse

Dieses Kapitel legt den Grundstein für die kommenden Kapitel und das Framework. Es bildet den ersten praktischen Schritt im Design-Science-Research-Prozess (vgl. Abschnitt 3.1), indem es die Datengrundlage für die Artefaktentwicklung (DSR-Schritt 3: Design und Entwicklung) schafft. Hier werden die Untersuchungsobjekte definiert und systematisch analysiert.

4.1.1 Auswahl der Untersuchungsobjekte

Die Auswahl der Untersuchungsobjekte zielt auf eine repräsentative und zugleich fokussierte Analyse. Da diese Arbeit dem DSR-Paradigma (vgl. Abschnitt 3.1) folgt und ein praxisrelevantes Artefakt das Ziel ist, orientiert sich die Auswahl primär an der Marktrelevanz der Lösungen. Wissenschaftliche Veröffentlichungen dienen der Fundierung des Stands der Forschung (vgl. Kapitel 2), für die Identifikation marktführender kommerzieller Anbieter wird hingegen der Gartner Magic Quadrant for SASE Platforms vom 9. Juli 2025 herangezogen[1], da dieser als Industriestandard für die Bewertung der Marktreife und Vision von Anbietern gilt.

[1] vgl. Forest, MacDonald und Koeppen, *2025 Gartner Magic Quadrant for SASE Platforms*, S. 3.

Die Untersuchung konzentriert sich exemplarisch auf einen kommerziellen Anbieter und ergänzt diesen um eine Lösung aus dem Free and Open Source Software (FOSS)-Umfeld. Die Wahl für den kommerziellen Anbieter fällt auf *Palo Alto Networks*. Der Magic Quadrant bewertet *Ability to Execute* und *Completeness of Vision*; Palo Alto Networks weist in beiden Dimensionen hohe Ausprägungen auf[2]. Die Analyse von Palo Alto Prisma SASE dient als Fallbeispiel für eine technologisch gereifte und am Markt etablierte Plattform.

Als zweites Objekt dient das Open-Source-Framework *OpenSASE*[3]. Es liefert eine technische Baseline und Referenz für einen anbieterunabhängigen, standardisierten Ansatz, relevant für ein generisches Metamodell.

Die Kombination aus kommerziellem Produkt und offenem Standard verbindet Praxisnähe mit konzeptioneller Fundierung. Voraussetzung für die Aufnahme ist eine öffentlich zugängliche Dokumentation; beide Objekte erfüllen diese Bedingung.

Abbildung 4.1 ordnet Palo Alto Networks im Gartner Magic Quadrant entlang der Dimensionen *Ability to Execute* und *Completeness of Vision* dem Quadranten der *Leaders* zu und dient als Kontext zur Begründung der Referenzplattformwahl.

[2] vgl. Forest, MacDonald und Koeppen, *2025 Gartner Magic Quadrant for SASE Platforms*, S. 3.
[3] vgl. Shain Singh und Sebastian Weitzel, *OpenSase Github Repo*, S. 3.

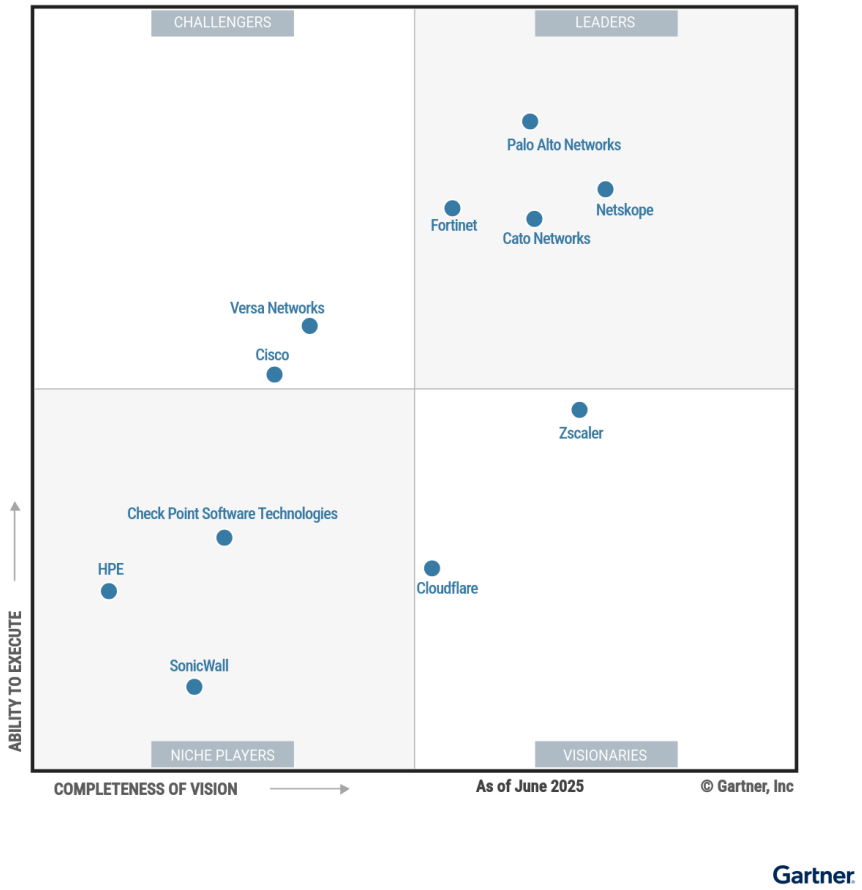

Abbildung 4.1: Gartner Magic Quadrant for SASE Platforms 2025
Quelle: Forest, MacDonald und Koeppen, *2025 Gartner Magic Quadrant for SASE Platforms*

Auswahlkriterien

Die im Gartner Magic Quadrant verwendeten Kriterien wie *Ability to Execute* (z.B. Produktqualität, Marktdurchdringung, Vertrieb) und *Completeness of Vision* (z.B. Marktverständnis, Innovationskraft, Zukunftsstrategie) dienen in dieser Arbeit als Makro-Filter zur Identifikation eines relevanten Marktführers. Diese strategischen Kriterien decken jedoch nicht die spezifischen technischen Anforderungen ab, die für die Entwicklung des Orchestrierungs-Artefakts dieser Arbeit notwendig sind.

Die nachfolgenden, technischen Kriterien (Mikro-Filter) wurden daher zusätzlich definiert. Sie fokussieren sich ausschließlich auf die Eignung der API-Schnittstellen für eine automatisierte, risikobasierte Steuerung und sind für die Zielerreichung dieser Arbeit entscheidend. Die wichtigsten Kriterien sind:

- **Qualität der API-Schnittstellen:** Dies umfasst die Existenz von Management- und Policy-APIs sowie deren Eignung für eine Automatisierung. Bewertet wurden Eigenschaften wie **Idempotenz** (wichtig für stabile, wiederholbare Automatisierungen), **Versionierung** (zur Vermeidung von Brüchen durch Anbieter-Updates) und dokumentierte **Rate-Limits** (als Obergrenze für die Orchestrierungsgeschwindigkeit).
- **Abdeckung der SASE-Kernkomponenten:** Das Untersuchungsobjekt muss APIs für die in Kapitel 2.1 definierten Kernkomponenten (insb. ZTNA, SWG, FWaaS und CASB) bereitstellen, um eine ganzheitliche Policy-Steuerung zu ermöglichen.
- **Authentifizierungsmechanismen:** Die APIs müssen moderne, aber automatisierbare Verfahren für Authentifizierung und Autorisierung (z.B. OAuth 2.0, Scoped API-Keys) bieten, damit das Framework sicher mit ihnen interagieren kann.
- **Qualität der Dokumentation:** Da die Analyse (siehe Unterabschnitt 4.1.2) auf Basis öffentlicher Dokumente erfolgt, ist deren Verfügbarkeit, Vollständigkeit und Verständlichkeit eine Grundvoraussetzung.

Die Untersuchung dokumentierter Einschränkungen ist ebenfalls Teil der Analyse, um die Grenzen der Lösungsentwicklung abzustecken.

4.1.2 Analyserahmen und Vorgehen

Die Analyse der Objekte (*Palo Alto* und *OpenSASE*) erfolgt in zwei Schritten:

1. eine API-Bestandsaufnahme zur Datenerhebung.
2. eine darauf aufbauende Komponentenprofilierung zur Einordnung der Stellgrößen im Framework.

Die API-Bestandsaufnahme identifiziert Management- und Policy-APIs zentraler
SASE- Komponenten. Grundlage bildet die Analyse der offiziellen Entwicklerdokumentation.

Die Komponentenprofilierung erfasst die Daten, stellt sie strukturiert dar und ordnet sie den jeweiligen Komponenten zu. Jedes Profil benennt zehn prägnante Stellgrößen je Komponente und erläutert deren Einordnung. Ergänzend dokumentiert die Profilierung nicht-funktionale Merkmale wie Leistungskennzahlen und bekannte Einschränkungen. Die Ergebnisse speisen das Metamodell, das Risikoklassen auf SASE-Steuerobjekte abbildet.

4.1.3 Analyse der Untersuchungsobjekte

Der Abschnitt untersucht die zuvor ausgewählten Anbieter (*Palo Alto* und *OpenSASE*) anhand des in Unterabschnitt 4.1.2 definierten Rahmens. Für jeden Anbieter entsteht ein Profil mit API-Architektur, verfügbaren Policy-Objekten und relevanten nicht-funktionalen Eigenschaften. Jedes Profil schließt mit einem kurzen Zwischenfazit zu den Implikationen für das Metamodell. Vollständige API-Inventare stehen im Kapitel Anhang zur verfügung.

Palo Alto Networks Prisma SASE

Palo Alto Networks bietet die SASE-Plattform *Prisma SASE* an. Die offizielle Dokumentation steht online frei zugänglich bereit.[4] Das offizielle Repository stellt Beispielprojekte und Referenzcode zur Nutzung der API-Schnittstellen bereit.[5]

Plattform- und API-Kontext Prisma SASE bietet Management- und Policy-Endpunkte im REST-Stil. Die Versionierung nutzt Pfadsegmente (z. B. `/sdwan/v2.x/`). Die Authentifizierung verwendet Token. Die Mandantentrennung erfolgt über einen Tenant-Kontext (z. B. HTTP-Header `x-ps-tenant`).

```
curl -X GET \
  -H 'Authorization: Bearer <ACCESS_TOKEN>' \
  -H 'x-ps-tenant: <tenant_id>' \
  https://api.sase.paloaltonetworks.com/sdwan/v2.0/api/securitypolicysets
```

Listing 1: Beispielhafter API-Aufruf zur Authentifizierung

Listing 1 zeigt exemplarisch die Authentifizierung gegenüber der Prisma-SASE-API und illustriert die Nutzung von Token und Tenant-Kontext.“

Datenquellen und Erhebung Die Analyse nutzt die Herstellerdokumentation[6] und die API Beispiele im GitHub Repoisitory[7]. Die Datensammlung umfasst insgesamt 3 028 Endpunkte

[4] vgl. Palo Alto, *Palo Alto Prisma SASE API Dokumentation*.
[5] vgl. Palo Alto, *Palo Alto Github Repository*.
[6] vgl. Palo Alto, *Palo Alto Prisma SASE API Dokumentation*.
[7] vgl. Palo Alto, *Palo Alto Github Repository*.

für Prisma SASE. Die Rohdaten finden sich im Kapitel Anhang unter -1-. Da nicht alle API-Steuerelemente von Relevanz sind, wird eine Reduktion durchgeführt.

Reduktionspfad Die Auswertung folgt vier Schritten:

1. Die Zuordnung der Endpunkte zu den Komponenten {SD-WAN, FWaaS, SWG, ZTNA, CASB}
2. Die Gruppierung zu Ressourcentypen (z. B. Set, Rule, Zone, Profile)
3. Die Auswahl policyrelevanter Stellgrößen über Schlüsselwörter und den Ausschluss reiner Abfragepfade
4. Die Normalisierung von Duplikaten und *legacy*/*unified*-Varianten

Abschließend erfolgt die Darstellung der Reduktion als Create, Read, Update, Delete (CRUD).

Ergebnisse der Reduktion Die Ergebnisse der Reduktion werden anhand von Kategorien dargestellt, die für das Design des Orchestrierungs-Frameworks entscheidend sind:

- **Komponente:** Dient der Zuordnung der API-Funktion zu den logischen SASE-Bausteinen (z.B. FWaaS).
- **Gruppen (Ressourcengruppen):** Zählt die Anzahl unterschiedlicher, steuerbarer Ressourcentypen (z.B. "Policy-Regel", SZone"). Diese Zahl indiziert die **Breite** der Steuerbarkeit.
- **Methoden (HTTP-Methoden):** Zählt die Gesamtanzahl aller Operationen. Die Aufteilung nach **CRUD** – hier als GET, POST, PUT, PATCH, DELETE – ist entscheidend. Sie zeigt das Verhältnis zwischen reinen Leseoperationen (GET) und **schreibenden Operationen** (POST, PUT etc.), die für eine aktive Orchestrierung (das Ändern von Regeln) benötigt werden.

Tabelle 4.1 fasst Ressourcengruppen und Methoden je Komponente zusammen. Abbildung 4.3 zeigt die Breite der Ressourcengruppen. Abbildung 4.2 visualisiert die Verteilung der HTTP-Methoden. Die vollständig reduzierte Liste befindet sich im Anhang unter -2-.

Komponente	Gruppen	Methoden	GET	POST	PUT	PATCH	DELETE
CASB	13	15	12	2	0	0	1
FWaaS	148	292	98	98	48	0	48
SD-WAN	1135	1918	772	614	286	4	242
SWG	2	2	1	1	0	0	0
ZTNA	4	6	2	2	2	0	0

Tabelle 4.1: Ressourcengruppen und Methoden je Komponente (Prisma SASE)

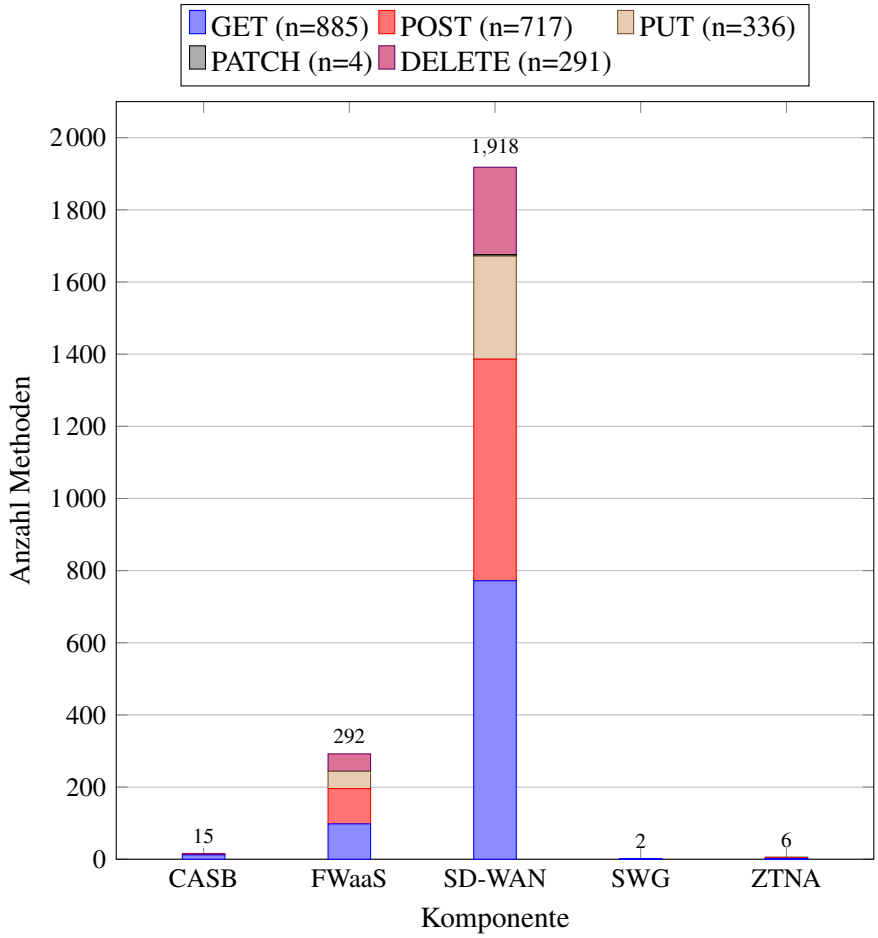

Abbildung 4.2: Verteilung der HTTP-Methoden je Komponente (CRUD)

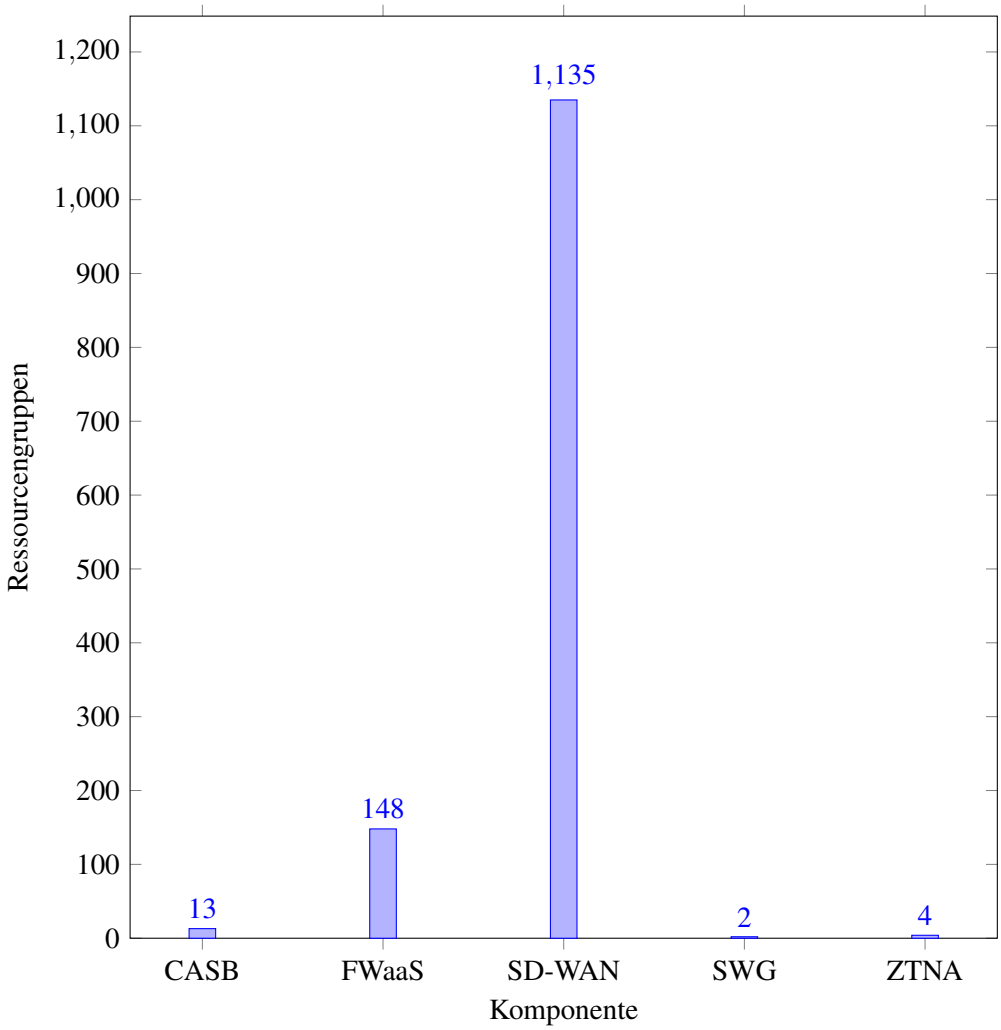

Abbildung 4.3: Ressourcengruppen je Komponente

Abbildung 4.3 zeigt: SD-WAN umfasst 1 135 , FWaaS 148, CASB 13, ZTNA vier und SWG zwei Stellmöglichkeiten. Und einen Schwerpunkt auf mengen- und regelbasierten Mechanismen in der API

Die Ergebnisse der Reduktion bilden die Grundlage für die im Folgenden dargestellten Komponentenprofile. Jedes Profil folgt einer einheitlichen Struktur, um die Vergleichbarkeit der SASE-Bausteine und ihrer Steuerflächen sicherzustellen. Zunächst erfolgt eine kontextuelle Einordnung der jeweiligen Komponente, die ihre funktionalen Schwerpunkte und ihren operativen Stellenwert innerhalb einer SASE-Architektur beschreibt. Anschließend werden spezifische Besonderheiten und strukturelle Grenzen der verfügbaren Management- und Steuerungsmechanismen herausgearbeitet. Darauf folgt eine analytische Einordnung der identifizierten

Steuerflächen. Abschließend werden zehn exemplarisch ausgewählte API-Operationen detailliert betrachtet.

Diese einheitliche Struktur dient dazu, die Heterogenität der Plattformmechanismen systematisch aufzubereiten und eine belastbare Grundlage für die in Kapitel 4.2 entwickelte Mapping-Logik zu schaffen.

Komponentenprofil FWaaS

Dieses Profil basiert auf der reduzierten FWaaS-CRUD-Liste (siehe im Kapitel Anhang unter -3-). Als Ergebnis der Reduktion verbleiben für dieses Profil 148 Ressourcengruppen mit 292 Operationen, die sich wie folgt auf die HTTP-Methoden aufteilen: 98 GET- (Lesen), 98 POST- (Erstellen), 48 PUT- (Aktualisieren) und 48 DELETE- (Löschen) Operationen.

Die Funktionsschwerpunkte liegen in zwei Feldern. Erstens steuern Zonen und NGFW-Policy-Stacks die Segmentierung und den Durchsatz; sie unterstützen Policy-Entwurf, Regelerstellung und Standortzuordnung. Zweitens erfolgt die Verkehrslenkung und Adressübersetzung über NAT-Sets und Pools, Statusabfragen liefern jeweils den aktuellen Zustand.

Besonderheiten und Grenzen Für das zu entwickelnde Orchestrierungs-Framework ist besonders relevant, dass ein expliziter Echtzeit-Commit- oder Telemetrie-Endpunkt in der Reduktion fehlt. Deploymentschritte liegen außerhalb des Scopes dieser API-Analyse. Die enge Kopplung an den SD-WAN-Kontext (Sites, Elements) verlangt konsistente Tenant-IDs über alle Aufrufe hinweg, was die Komplexität der Orchestrierungslogik erhöht. Mandantenfähigkeit ist nur implizit gegeben, und dedizierte Audit- oder Rollback-Routen fehlen. Dies bedeutet, dass eine Transaktionssicherheit (ein Rollback bei fehlerhaften Änderungen) im Framework selbst abgebildet werden muss.

Analytische Einordnung der FWaaS-Steuerflächen Die folgenden Annotationen interpretieren zehn exemplarische Operationen als verbleibende Steuerflächen des reduzierten FWaaS-Portfolios. Die Bewertung betrachtet Eingabestruktur, Steuerungstiefe, betriebliche Einsatzlogik und Konsequenzen für das Orchestrierungsmuster. Auffällig ist, dass die Methoden unter dem Pfad Prefix *sdwan* eingeordnet sind.

- **Pfad:** `/sdwan/v2.0/api/elementsecurityzones/query`
  Methode: POST
  Eingaben: JSON-Body nach `ElementSecurityZoneQuery` mit Paginierung (`limit`, `dest_page`), Aggregationen und Filtern (u. a. `site_id`, `zone_id`, `interface_ids`, `group_by`).
  Einordnung: Die Query-Funktion schafft Sichtbarkeit über mandanten- und standortübergreifende Segmentierungsobjekte. Sie bildet die Grundlage für konsistente Policy-Zonierung und erlaubt formale Prüfungen, etwa auf Zonenredundanz oder Interzonenflüsse im Zero-Trust-Ansatz.

- **Pfad:** `/sdwan/v2.0/api/ngfwsecuritypolicysetstacks`
  Methoden: GET, POST
  Eingaben: POST erwartet `SecurityPolicyV2SetStackScreen` mit Pflichtfeld `name` sowie optional `policyset_ids` (1–4 Sets), `defaultrule_policyset_id`, `tags`.
  Einordnung: Das Anlegen neuer Policy-Stacks schafft eine hierarchische Ebene für Set-Vererbungen. Die Trennung von Entwurf und Distribution erleichtert Vergleiche zwischen gestapelten Regelsätzen und Wirkungsanalysen entlang der Deployments.

- **Pfad:** `/sdwan/v2.0/api/ngfwsecuritypolicysets/{policy_set_id}/ngfwsecuritypolicyrules`
  Methoden: GET, POST
  Eingaben: Pfadparameter `policy_set_id`; POST-Body `SecurityPolicyV2N2RuleScreen` mit `name`, `action`, `enabled` sowie optionalen Sammlungen (`source_zone_ids`, `destination_zone_ids`, `prefix_ids`, `service_contexts`, `app_def_ids`, `user_or_group`, `tags`).
  Einordnung: Die Operation erzeugt die eigentlichen Regeln. Die Selektivität bildet granulare Kontrollflächen ab (Schicht 7, Benutzer- und Applikationskontexte). Analysen zur Regelqualität und -komplexität setzen hier an, etwa zu Überlappungen oder Minimalitätskriterien.

- **Pfad:** `/v2.2/api/tenants/{tenant_id}/ngfwsecuritypolicysets/{policy_set_id}/ngfwsecuritypolicyrules/{policy_rule_id}`
  Methode: PUT
  Eingaben: `tenant_id`, `policy_set_id`, `policy_rule_id`;
  Body wie `SecurityPolicyV2N2RuleScreen` mit aktualisierten Feldwerten.
  Einordnung: Regeländerungen markieren mögliche Abweichungen zum Sollmodell. Die

Operation liefert Daten für Drift-Analysen und für Korrelationen zwischen Änderungen und Incident-Raten.

- **Pfad:** `/v2.2/api/tenants/{tenant_id}/ngfwsecuritypolicyrules/query`
  Methode: POST
  Eingaben: Pfadparameter `tenant_id`; Body nach `SecurityPolicyV2N2RuleQueryFilter` mit optionalen Filtern (z. B. `action`, `enabled`, Zonensets, `prefix_ids`, Applikations-IDs) sowie Paginierung und Sortierung.
  Einordnung: Die Query-Schnittstelle macht das Policy-Set analytisch zugänglich. Daraus entstehen Regelmetriken (z. B. Anzahl aktiver Ausnahmen) und verifizierbare Regelmengen, etwa mit Constraint-Lösern.

- **Pfad:** `/sdwan/v2.0/api/ngfwsecuritypolicyglobalprefixes`
  Methoden: GET, POST
  Eingaben: POST nutzt `SecurityPolicyGlobalPrefixScreen` mit `name` sowie optional `ipv4_prefixes` (Listen) und `tags`; GET ohne Payload.
  Einordnung: Globale Präfixobjekte wirken als Normalisierungshebel. Eindeutige Präfixdefinitionen reduzieren Duplikate und senken Komplexität im Regelwerk.

- **Pfad:** `/v2.1/api/tenants/{tenant_id}/sites/{site_id}/ngfwsecuritypolicylocalprefixes`
  Methoden: GET, POST
  Eingaben: `tenant_id`, `site_id`; POST-Body `SiteSecurityPolicyV2N1PrefixAssociationScreen` mit `prefix_id` sowie optional `ipv4_prefixes`, `ipv6_prefixes`, `tags`.
  Einordnung: Lokale Präfixassoziationen koppeln Standorttopologien und Policyobjekte. Dadurch lassen sich globale und lokale Adresssteuerung vergleichen und standortspezifische Zuweisungen modellieren.

- **Pfad:** `/sdwan/v2.0/api/natpolicysets`
  Methoden: GET, POST
  Eingaben: POST nimmt `NATPolicySet` entgegen (Pflicht: `name`; optional u. a. `clone_from`, `policy_rules[]`, Aktivierungsflags, `destination_zone_policyrule_order`, `tags`).
  Einordnung: NAT-Policy-Sets strukturieren Quell- und Zielübersetzungen. Die Operation

unterstützt Analysen zur Kopplung von NAT- und Security-Policies, etwa zu gemeinsamen Rollout-Zyklen.

- **Pfad:** `/sdwan/v2.0/api/natpolicysets/{nat_policy_set_id}/natpolicyrules`
  Methoden: GET, POST, PUT, DELETE
  Eingaben: `nat_policy_set_id`; POST/PUT-Body `NATPolicyRule` mit `name`, `actions` (Übersetzungsanweisungen) sowie optionalen Eigenschaften für Zonen/Präfixe (`source_zone_id`, `destination_zone_id`, `prefix_ids`), Portbereiche, Protokoll, Poolreferenzen, `tags`.
  Einordnung: NAT-Regeln beschreiben konkrete Transformationslogiken. Analysen leiten daraus Strukturmerkmale (z. B. SNAT vs. DNAT) und Effekte auf Session-Metadaten ab.

- **Pfad:** `/sdwan/v2.0/api/natpolicysets/{nat_policy_set_id}/status`
  Methode: GET
  Eingaben: `nat_policy_set_id`; kein Body.
  Einordnung: Die Statusabfrage liefert Feedback zu Implementierungsfortschritt und Health-Indikatoren. Die Messpunkte stützen Governance über den Policy-Lifecycle und helfen bei der Erkennung von Rollout-Anomalien.

Komponentenprofil CASB

Dieses Profil basiert auf der reduzierten CASB-CRUD-Liste (siehe im Kapitel Anhang unter -4-). Diese umfasst 13 Ressourcengruppen mit 15 Operationen. Hiervon entfallen zwölf auf GET, zwei auf POST und eine auf DELETE. Die Funktionsschwerpunkte liegen in zwei Feldern: (1) SaaS-Anwendungs- und Plugin-Inventar (Registrierung, Integrationspfade, Zugriffe von Drittanbietern); (2) Identitäts- und Compliance-Workflows (MFA-Telemetrie, Account-Audits, Ticketing zur Behebung).

Besonderheiten und Grenzen Die Steuerflächen sind leselastig und setzen den Header `x-ps-tenant` voraus. Ändernde Operationen betreffen primär Report-Generierung und Ticket-Lifecycle. Direkte Policy- oder Konfigurationsänderungen an SaaS-Systemen gehören nicht zur reduzierten Menge.

Analytische Einordnung der CASB-Steuerflächen Die folgenden Annotationen interpretieren zehn exemplarische Operationen als verbleibende Steuerflächen des reduzierten CASB-Portfolios. Die Bewertung betrachtet Eingabestruktur, Steuerungstiefe sowie Nutzen für Überwachung und Governance.

- **Pfad:** `/sspm/api/v1/apps`
  Methode: GET
  Eingaben: optionale Query-Parameter `filter`, `order_by`, `page_token`.
  Einordnung: Liefert die Referenzliste eingebundener SaaS-Applikationen und bildet die Basis für Risikobewertung, Abdeckungsanalyse und gezielte Kontrollen.

- **Pfad:** `/sspm/api/v1/apps/{app_id}/settings`
  Methode: GET
  Eingaben: Pfadparameter `app_id`.
  Einordnung: Erschließt app-spezifische Konfigurationsparameter und ermöglicht Soll/Ist-Abgleiche auf Einstellungsebene, etwa zu SSO oder Audit-Logging.

- **Pfad:** `/sspm/api/v1/plugin-users`
  Methode: GET
  Eingaben: Header `x-ps-tenant`; optionale Query-Parameter `filter`, `order_by`, `page_token`, `limit`.
  Einordnung: Verknüpft Benutzer mit angebundenen Plugins und schafft Transparenz über externe Integrationen als Grundlage für Privileg-Reviews.

- **Pfad:** `/sspm/api/v1/plugin-users/{id}/plugins`
  Methode: GET
  Eingaben: Header `x-ps-tenant`; Pfadparameter `id`.
  Einordnung: Zeigt Integrationen eines Benutzers und unterstützt Least-Privilege-Strategien sowie forensische Analysen.

- **Pfad:** `/sspm/identity/v1/idps/{idpId}/mfa_activity/count_by_app_type`
  Methode: GET
  Eingaben: Pfadparameter `idpId`; optionale Query-Parameter `filter`, `limit`, `page`; Header `x-ps-tenant`.

Einordnung: Aggregiert MFA-Nutzungszahlen nach Anwendungstyp und zeigt den Reifegrad der Identity-Härtung für Compliance und Anomalieerkennung.

- **Pfad:** `/sspm/identity/v1/saas_instances`
  Methode: GET
  Eingaben: Header `x-ps-tenant`.
  Einordnung: Liefert das operative SaaS-Portfolio und verankert CASB-Sichtbarkeit im Mandantenkontext.

- **Pfad:** `/sspm/identity/v1/saas_instances/{saasInstanceId}/saas_accounts`
  Methode: GET
  Eingaben: Pfadparameter `saasInstanceId`; optionale Query-Parameter `filter`, `limit`, `page`, `sortBy`; Header `x-ps-tenant`.
  Einordnung: Liefert Account- und Rollendaten zur Instanz und unterstützt Entitlement-Reviews sowie Lizenzplanung.

- **Pfad:** `/sspm/identity/v1/saas_instances/{saasInstanceId}/saas_accounts/csv_report`
  Methode: POST
  Eingaben: Pfadparameter `saasInstanceId`; optionale Query-Parameter `filter`, `sortBy`; Header `x-ps-tenant`; Body `DownloadCsvRequest`.
  Einordnung: Erzeugt exportfähige Account-Reports und erleichtert Audit-Trails sowie Offline-Analysen.

- **Pfad:** `/sspm/identity/v1/saas_instances/{saasInstanceId}/saas_activity`
  Methode: GET
  Eingaben: Pfadparameter `saasInstanceId`; optionale Query-Parameter `filter`, `limit`, `page`, `sortBy`; Header `x-ps-tenant`.
  Einordnung: Liefert Aktivitätslogs (Aktion, Ressource, Standort) als Grundlage für Verhaltensanalysen und Incident Response.

- **Pfad:** `/sspm/identity/v1/{saasInstanceId}/tickets`
  Methoden: GET, POST, DELETE

Eingaben: Pfadparameter `saasInstanceId`; Header `x-ps-tenant`; Body `CreateTicketRequest` bzw. `UnlinkTicketRequest`.

Einordnung: Überführt Findings in Tickets, pflegt sie oder entfernt Verknüpfungen. Das unterstützt Nachweisführung und Service Level Agreement (SLA)-Steuerung.

Komponentenprofil SD-WAN

Dieses Profil basiert auf der reduzierten SD-WAN-CRUD-Liste (siehe im Kapitel Anhang unter -5-). Diese umfasst 1 135 Ressourcengruppen mit 1 918 Operationen, davon 772 GET-, 614 POST-, 286 PUT-, vier PATCH- und 242 DELETE-Operationen. Die Funktionsschwerpunkte liegen in zwei Feldern: (1) Konfigurations- und Automationsflächen für Sites, Service Connections, Cellular- und Autonomous Digital Experience Management (ADEM)-Profile; (2) Betriebs- und Observability-Funktionalitäten von Bulk-Abfragen bis zu Korrelationsevents.

Besonderheiten und Grenzen Die Analyse zeigt eine hohe Komplexität durch heterogene Request-Schemata und methodische Redundanzen die das Framework bei der Auswahl der Steuerflächen berücksichtigen muss. Besonders relevant ist die enge Verzahnung mit Identitäts- und Monitoring-APIss, die eine reine SD-WAN-Orchestrierung erschwert und eine zustandsbehaftete Logik erfordert, die auch diese externen Datenquellen einbezieht.

Analytische Einordnung der SD-WAN-Steuerflächen Die folgenden Annotationen interpretieren zehn exemplarische Operationen als verbleibende Steuerflächen des reduzierten SD-WAN-Portfolios. Die Bewertung betrachtet Eingabestruktur, Steuerungstiefe, betriebliche Einsatzlogik und Konsequenzen für das Orchestrierungsmuster.

- **Pfad:** `/mt/monitor/v1/agg/serviceConnectivity`

  Methode: POST

  Eingaben: Query-Parameter `agg_by`, Header `X-PANW-Region`; JSON-Body mit Aggregationsdefinition.

  Einordnung: Verdichtet Status von Remote-Networks und Service Connections mandantenübergreifend. Liefert Health-Telemetrie für Managed Service Provider (MSP)-Betrieb und SLA-Monitoring.

- **Pfad:** `/sdwan/v2.0/api/activeuserips/query`
  Methode: POST
  Eingaben: Body `ActiveUserIPQuery`.
  Einordnung: Verknüpft Identitätszuordnungen mit aktiven Sessions und unterstützt Policy-Zuweisungen sowie Troubleshooting im Zero-Trust-Kontext.

- **Pfad:** `/sdwan/v2.0/api/sites/bulk_config_state/query`
  Methode: POST
  Eingaben: Body `BaseQueryDTO`.
  Einordnung: Liefert gebündelte Config-/State-Deltas und ermöglicht Drift-Erkennung sowie Compliance-Checks in großen Flotten.

- **Pfad:** `/sdwan/v2.0/api/sites/bulkoperations`
  Methode: POST
  Eingaben: Body `BulkSiteUpdate`.
  Einordnung: Hebel für Massenänderungen. Unterstützt orchestrierte Rollouts und Infrastructure-as-Code.

- **Pfad:** `/sdwan/v2.0/api/sites/appacceleration/query`
  Methode: POST
  Eingaben: Body `AppAccelerationStatusQuery`.
  Einordnung: Bewertet App-Acceleration-Status je Site und liefert Steuerdaten für SaaS-Optimierung und Experience-Monitoring.

- **Pfad:** `/sdwan/v2.0/api/anynetlinks/correlationevents/query`
  Methode: POST
  Eingaben: Body `AnynetLinkQuery`.
  Einordnung: Konsolidiert linkbezogene Ereignisse, erleichtert Root-Cause-Analysen und automatisierte Incident-Workflows.

- **Pfad:** `/sdwan/v2.0/api/serviceconnections`
  Methode: GET
  Eingaben: keine.
  Einordnung: Liefert das Inventar aktiver Service Connections. Grundlage für Traffic Engineering in Hybrid-Cloud-Umgebungen und Kapazitätsplanung.

- **Pfad:** `/sdwan/v2.0/api/apnprofiles`
  Methoden: GET, POST
  Eingaben: POST-Body `APNProfileScreen`.
  Einordnung: Steuert Mobilfunkzugänge (APN). Wichtig für Edge-Redundanz und IoT-Anbindung; Profilvalidierung sichert Carrier-Compliance.

- **Pfad:** `/sdwan/v2.0/api/sites/{site_id}/demsiteconfigs`
  Methoden: GET, POST
  Eingaben: Pfadparameter `site_id`; POST-Body `DemSiteConfigScreen`.
  Einordnung: Verwaltung von ADEM-Site-Configs verankert Experience-Messpunkte an Standorten und koppelt SD-WAN mit Digital Experience Monitoring.

- **Pfad:** `/sdwan/v2.0/api/sites/{site_id}/demsiteconfigs/{config_id}`
  Methoden: GET, PUT, DELETE
  Eingaben: Pfadparameter `site_id`, `config_id`; PUT-Body `DemSiteConfigScreen`.
  Einordnung: Deckt den Lifecycle einzelner ADEM-Konfigurationen ab. Unterstützt gezielte Änderungen, Rollback und Quality Gates.

Komponentenprofil ZTNA

Dieses Profil basiert auf der reduzierten ZTNA-CRUD-Liste (siehe im Kapitel Anhang unter -6-). Diese umfasst vier Endpunkte mit sechs Operationen, hiervon entfallen zwei auf GET, zwei auf POST und zwei auf PUT. Der Schwerpunkt liegt auf Vertrauensbewertungen für IoT- und Geräteidentitäten, mandantenunabhängig (*unified*) und mandantenspezifisch (*legacy*).

Besonderheiten und Einschränkungen Die Reduktion zeigt eine starke Einschränkung. Alle Operationen nutzen das Schema DeviceIdConfidence direkte Policy- oder Zugriffsendpunkte (z.B. Erstelle ZTNA-Regel) verbleiben nicht. Für das Orchestrierungs-Framework bedeutet dies, dass eine risikobasierte ZTNA-Steuerung nicht direkt, sondern nur indirekt über die Anpassung von Vertrauensbewertungen (Confidence Scores) möglich ist.

Analytische Einordnung der ZTNA-Steuerflächen Da die Reduktion nur sechs Operationen ergeben hat, werden im Folgenden alle verbleibenden Steuerflächen des ZTNA-Portfolios interpretiert.

- **Pfad:** `/sdwan/v2.0/api/iotservices`
  Methode: GET
  Eingaben: keine.
  ***Einordnung**:* Liefert das globale Mapping der Confidence-Werte und schafft Transparenz über automatische Klassifizierungen.

- **Pfad:** `/sdwan/v2.0/api/iotservices`
  Methode: POST
  Eingaben: Body `DeviceIdConfidence`.
  ***Einordnung**:* Aktualisiert zentral Vertrauenswerttabellen und beeinflusst ZTNA-Zugriffsentscheidungen.

- **Pfad:** `/sdwan/v2.0/api/iotservices/{iot_service_id}`
  Methode: PUT
  Eingaben: Pfadparameter `iot_service_id`; Body `DeviceIdConfidence`.
  ***Einordnung**:* Korrigiert gezielt einzelne Mapping-Einträge, etwa nach manueller Verifikation.

- **Pfad:** `/v2.0/api/tenants/{tenant_id}/iotservices`
  Methode: GET
  Eingaben: Pfadparameter `tenant_id`.
  Einordnung: Spiegelt die Vertrauenswerte in der Mandantensicht und macht Abweichungen vom globalen Default sichtbar.

- **Pfad:** `/v2.0/api/tenants/{tenant_id}/iotservices`
  Methode: POST
  Eingaben: Pfadparameter `tenant_id`; Body `DeviceIdConfidence`.
  Einordnung: Erlaubt mandantenspezifische Overrides für differenzierte Sicherheitsanforderungen.

- **Pfad:** `/v2.0/api/tenants/{tenant_id}/iotservices/{iot_service_id}`
  Methode: PUT
  Eingaben: Pfadparameter `tenant_id`, `iot_service_id`; Body `DeviceIdConfidence`.
  Einordnung: Schärft tenant-spezifische Einträge nach, wenn lokale Telemetrie neue Evidenz liefert.

Komponentenprofil SWG Dieses Profil basiert auf der reduzierten SWG-CRUD-Liste (siehe im Kapitel Anhang unter -7-) (zwei Endpunkte, zwei Operationen; eine GET-, eine POST-Operation). Der Funktionsschwerpunkt liegt in der Verwaltung von Benachrichtigungsprofilen für Cloud-Sicherheitsereignisse.

Besonderheiten und Einschränkungen Das Profil ist extrem fokussiert. Es verbleiben keine Policy- oder Session-Endpunkte. Konfigurationsänderungen betreffen ausschließlich Notification-Flows. Dies ist ein Schlüsselergebnis der Analyse: Eine direkte, risikobasierte Steuerung von SWG-Regeln (z.B. "Blockiere Kategorie X für User Y") ist über die untersuchte API nicht vorgesehen. Das Framework kann die SWG-API daher primär als passive Alarmquelle (via Webhooks), aber nicht als aktiven Durchsetzungspunkt (Enforcement Point) nutzen.

Analytische Einordnung der SWG-Steuerflächen Die Reduktion für das SWG-Profil liefert lediglich zwei Operationen. Die folgenden Annotationen interpretieren diese beiden verbleibenden Steuerflächen.

- **Pfad:** `/api/cloud/2.0/agg/notifications/profiles`
  Methode: POST
  Eingaben: Body `NotifProfile` (Tenant-Liste, Notification-Typen, Kanaldefinitionen inkl. Webhook und E-Mail).
  Einordnung: Steuert, welche Ereignisse über welche Kanäle verteilt werden. Zentral für Betriebsalarme und Compliance-Alerts.

- **Pfad:** `/api/cloud/2.0/agg/notifications/profiles/types`
  Methode: GET
  Eingaben: keine.
  Einordnung: Liefert die Taxonomie validierter Notification-Typen, Kategorien und Subkategorien. Voraussetzung für konsistente Profilkonfigurationen.

OpenSASE

OpenSASE ist ein frei verfügbares Open-Source-Projekt, das von einer Community freiwilliger Entwickler initiiert und weitergeführt wird. Es integriert zentrale SASE-Funktionalitäten in einer containerisierten Architektur auf Basis von Docker. Die offizielle Projektdokumentation sowie sämtliche Artefakte sind im GitHub-Repository frei zugänglich[8].

Plattform- und API-Kontext OpenSASE bildet zentrale SASE-Funktionen mit OpenVPN (ZTNA), Squid (SWG), C/ICAP und ClamAV (Threat Prevention) sowie Dnsmasq (DNS) in separaten Containern ab. Die Lösung leitet Hypertext Transfer Protocol (HTTP)/Hypertext Transfer Protocol Secure (HTTPS)-Verkehr aus dem VPN an Squid weiter und prüft Inhalte per ICAP/ClamAV gegen Signaturen und Sperrlisten. Dnsmasq stellt konsistente Namensauflösung für Proxy und Clients bereit.

API-Architektur und Authentifizierung Eine zentrale Management-API fehlt. Die Steuerung erfolgt über Konfigurationsdateien und Container-Kommandos, nicht über Representational State Transfer (REST) oder GraphQL. OpenVPN nutzt Transport Layer Security (TLS)-Clientzertifikate und definiert Routen und Domain Name System (DNS). Squid setzt eine eigene Root-Certificate Authority (CA) für Secure Sockets Layer (SSL)-Inspection ein. Ein OAuth- oder Token-Flow liegt nicht vor.

Datenquellen und Erhebung Als Datenquelle dienen die Artefakte aus dem Projekt-Repository (z. B. `docker-compose.yml`, Kubernetes-Manifeste sowie die Konfigurationsdateien für Proxy und VPN). Da eine zentrale API fehlt, erfolgte die Erhebung durch eine manuelle, systematische Analyse dieser Quelltext-Artefakte. Die relevanten Schemata und Manifeste wurden gesichtet, um die policyrelevanten Schlüsselpfade (z.B. Pfade zu Sperrlisten) und Kommandos händisch zu extrahieren. Diese wurden anschließend in strukturierte Comma-Separated Values (CSV)/JavaScript Object Notation (JSON)-Artefakte (siehe Kapitel Anhang unter -8-) überführt.

[8] vgl. Shain Singh und Sebastian Weitzel, *OpenSase Github Repo*.

Reduktionspfad Die Verdichtung folgt **vier** Schritten:

1. **Einordnung** der Konfigurationsdateien und Container in die Funktionsblöcke {ZTNA, SWG, Threat Prevention, DNS}.
2. **Gruppierung** der Steuerobjekte nach Ressourcentypen {Compose-Service, Kubernetes-Manifest, Volume}.
3. **Auswahl** der policyrelevanten Stellgrößen (z. B. `acl`-Pfade, SSL-*bump*-Ausnahmen, ICAP-Profile).
4. **Normalisierung** von Aliassen und Dubletten.

Ergebnisse der Reduktion Abbildung 4.4 zeigt die Anzahl der Dienste je Funktionsblock. Abbildung 4.5 stellt die Artefakt-Typen dar.

Hinweis zur Metrik Eine zentrale Management-API liegt nicht vor. CRUD-Zählungen sind daher nicht anwendbar. Ersatzmetriken sind Container-, Manifest- und Volume-Zählungen sowie identifizierte Konfigurationsfragmente als policyrelevante Stellgrößen.

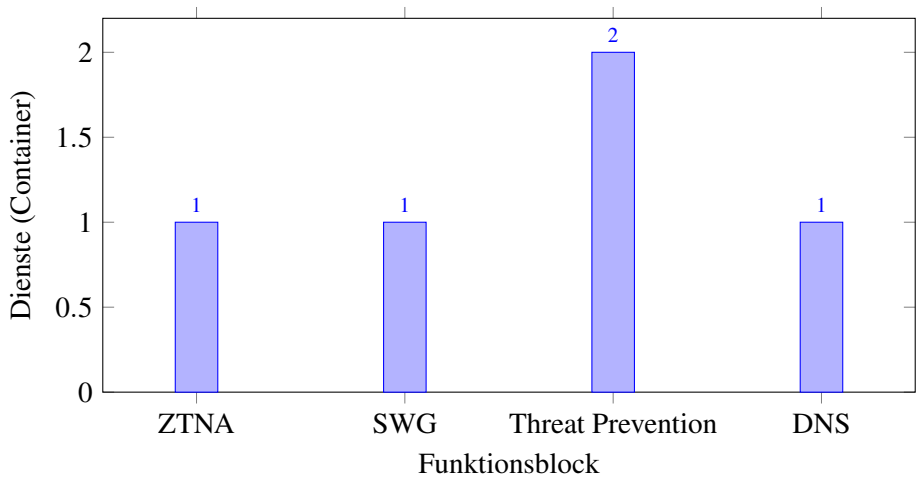

Abbildung 4.4: OpenSASE: Stellmöglichkeiten je Komponente

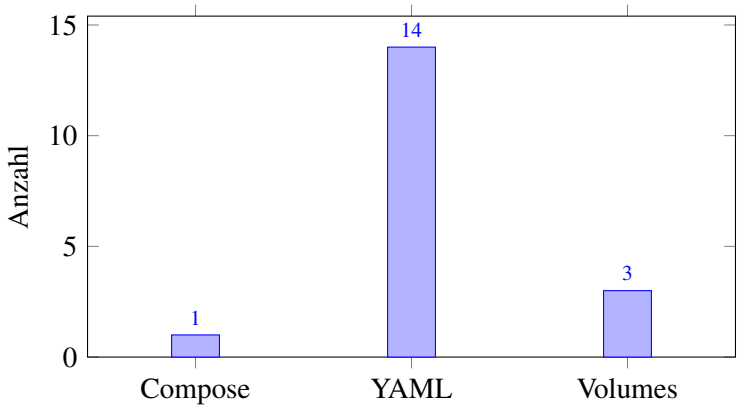

Abbildung 4.5: OpenSASE: Artefakte nach Typ

Komponentenprofile und analytische Einordnung

ZTNA (OpenVPN)

Konfiguration: openvpn/server.conf, Client-Zertifikate, Routen, DNS.

Steuerungstiefe: Tunnelparameter, Netzzuordnungen, Weiterleitung von HTTP/HTTPS an Squid.

Einordnung: Zuweisung von Zugriffspfaden und Adressräumen; keine API-Ebene.

SWG (Squid)

Konfiguration: squid/squid.conf, acl/*.lst, eigene Root-CA.

Steuerungstiefe: acl-Definitionen, Filterregeln, SSL-bump, Ausnahmelisten (nobump.txt).

Einordnung: Web-Filterung, HTTPS-Inspection, Sperrlisten-Integration; keine zentrale Policy-API.

Threat Prevention (C/Internet Content Adaptation Protocol (ICAP) + ClamAV)

Konfiguration: c-icap/c-icap.conf, clamav/clamd.conf.

Steuerungstiefe: ICAP-Services, Scan-Modi, Signaturquellen.

Einordnung: Inhaltsprüfung im Datenpfad über ICAP; Bindung an Squid.

DNS (Dnsmasq)

Konfiguration: dnsmasq/*.conf.
Steuerungstiefe: Namensauflösung für Proxy und Clients; Hostnamen im Docker-Netz.
Einordnung: Konsistente Namensauflösung als Voraussetzung für stabile HTTPS-Inspection.

Service-Bindings und Aktivierung

Artefakte: docker-compose.yml, Kubernetes-Manifeste, Volumes.
Ablauf: docker compose build/up, Container-reload für Konfigurationsänderungen.
Einordnung: Kein transaktionaler Commit; Aktivierung über Neustart/Reload; keine Rate-Limits oder dedizierte Telemetrie. Das Repository rät vom Produktionseinsatz ab.

Zwischenfazit und Anforderungen an das Metamodell

Das Metamodell bildet AccessPolicy, Condition, Zone, NatPolicy, ServiceBinding und AuditSink ab. Beziehungen verknüpfen Richtlinien, Bedingungen und Bindings. Die Struktur koppelt Richtlinien an Risikoklassen und berücksichtigt Container-Lebenszyklen bei der Aktivierung.

Komponentenabdeckung im Vergleich

Tabelle 4.2 vergleicht die Abdeckung zentraler SASE-Komponenten in Prisma SASE und OpenSASE.

Komponente	**Prisma SASE**	**OpenSASE**
FWaaS	vorhanden (NGFW-Policies, NAT)	nicht vorhanden
CASB	vorhanden	nicht vorhanden
SD-WAN	vorhanden	nicht vorhanden
ZTNA	vorhanden	vorhanden (OpenVPN)
SWG	vorhanden	vorhanden (Squid)
Threat Prevention	integriert	vorhanden (C/ICAP, ClamAV)
DNS	integriert	vorhanden (Dnsmasq)
Management-API	vorhanden (Rest)	Containersteuerung

Tabelle 4.2: Vergleich der Komponentenabdeckung in Prisma SASE und OpenSASE

Die Gegenüberstellung in Tabelle 4.2 verdeutlicht, dass Prisma SASE eine vollständige Abdeckung der zentralen Komponenten einschließlich einer versionierten Management-API bietet, während OpenSASE nur Teilbereiche implementiert und keine konsolidierte Steuerungsschicht bereitstellt. Daher bildet Prisma SASE die Grundlage für die Entwicklung des Metamodells, während OpenSASE nicht weiterführend analysiert wird.

4.2 Konzeption und Systemdesign

Auf Grundlage der Komponentenanalyse aus Kapitel 4.1 entwickelt dieses Kapitel das detaillierte Design des Risk-Aware SASE-Frameworks. Der Prozess gliedert sich in zwei Schritte: Zuerst definiert das Kapitel die Risikoklassen als Eingangsgrößen für das System. Anschließend entwirft die zentrale Mapping-Logik, welche diese Klassen auf konkrete SASE-Steuerobjekte abbildet.

4.2.1 Definition der Risikoklassen

Dieser Abschnitt entwickelt die Risikoklassen, die als Eingang für den Policy-Orchestrator dienen. Die Modellierung folgt der theoretischen Grundlage aus Kapitel 2.1.3 und überträgt das qualitative Modell des BSI auf den Kontext der SASE-Architektur.

Schritt 1: Operationalisierung der Bewertungsdimensionen

Die Risikomatrix nutzt die Dimensionen Schadenshöhe und Eintrittshäufigkeit. Die folgenden Kriterien konkretisieren beide Dimensionen für den betrachteten Anwendungsfall:

- **Schadenshöhe:** Die Skala beschreibt den Einfluss auf zentrale Geschäftsprozesse und mögliche Reputationsschäden. Die höchste Stufe *Existenzbedrohend* liegt vor, wenn kritische Dienste länger als 24 Stunden ausfallen oder sensible Daten unwiederbringlich verloren gehen[9].

- **Eintrittshäufigkeit:** Die Skala nutzt Erfahrungswerte und Threat-Intelligence-Daten. Die Stufe *Sehr häufig* beschreibt mehrere Vorkommnisse pro Woche. Die Stufe *Selten* beschreibt Ereignisse mit einem Intervall von mehr als zwölf Monaten.

[9] vgl. Königs, *IT-Risikomanagement mit System*, S. 313.

Schritt 2: Definition der Risikoklassen

Die Modellierung nutzt fünf Risikoklassen. Sie bilden den Risikoappetit ab, den das Framework verwendet. Die Klassen ordnen das Risikoniveau ein. Jede Klasse besitzt ein Handlungsziel. Diese Ziele bestimmen nicht das Risiko selbst. Sie beschreiben den Umgang mit dem Risiko im Rahmen des Frameworks.

Schritt 3: Aggregation in der Risikomatrix

Die Risikomatrix kombiniert Schadenshöhe und Eintrittshäufigkeit. Die Skala folgt einem moderat-konservativen Risikoappetit[10]. Schadensereignisse mit existenzbedrohendem Einfluss erhalten unabhängig von der Eintrittshäufigkeit ein hohes Risikoniveau. Ereignisse mit begrenztem Einfluss, aber hoher Frequenz, erreichen ein erhöhtes Risikoniveau, da eine hohe Wiederholungsrate eine dauerhafte Belastung erzeugt.

Die Matrix umfasst sechzehn Felder. Jedes Feld repräsentiert ein eindeutiges Risikoniveau. Die Klassifizierung fasst diese Felder zu fünf Risikoklassen zusammen. Die Aggregation erleichtert die Steuerung im Framework und verdichtet ähnliche Risikoniveaus zu gemeinsamen Klassen. Abbildung 4.6 zeigt die Zuordnung.

Diese definiert ausschließlich das Risikoniveau. Diese enthält keine Maßnahmen. Das Framework verknüpft die Klassen erst im Policy-Mapping mit organisatorischen oder technischen Reaktionen. Die Maßnahmen entstehen somit im Systemdesign. Sie sind nicht Bestandteil der Risikoklasse im Sinne der BSI-Definition.

[10] vgl. Königs, *IT-Risikomanagement mit System*, 33–44.

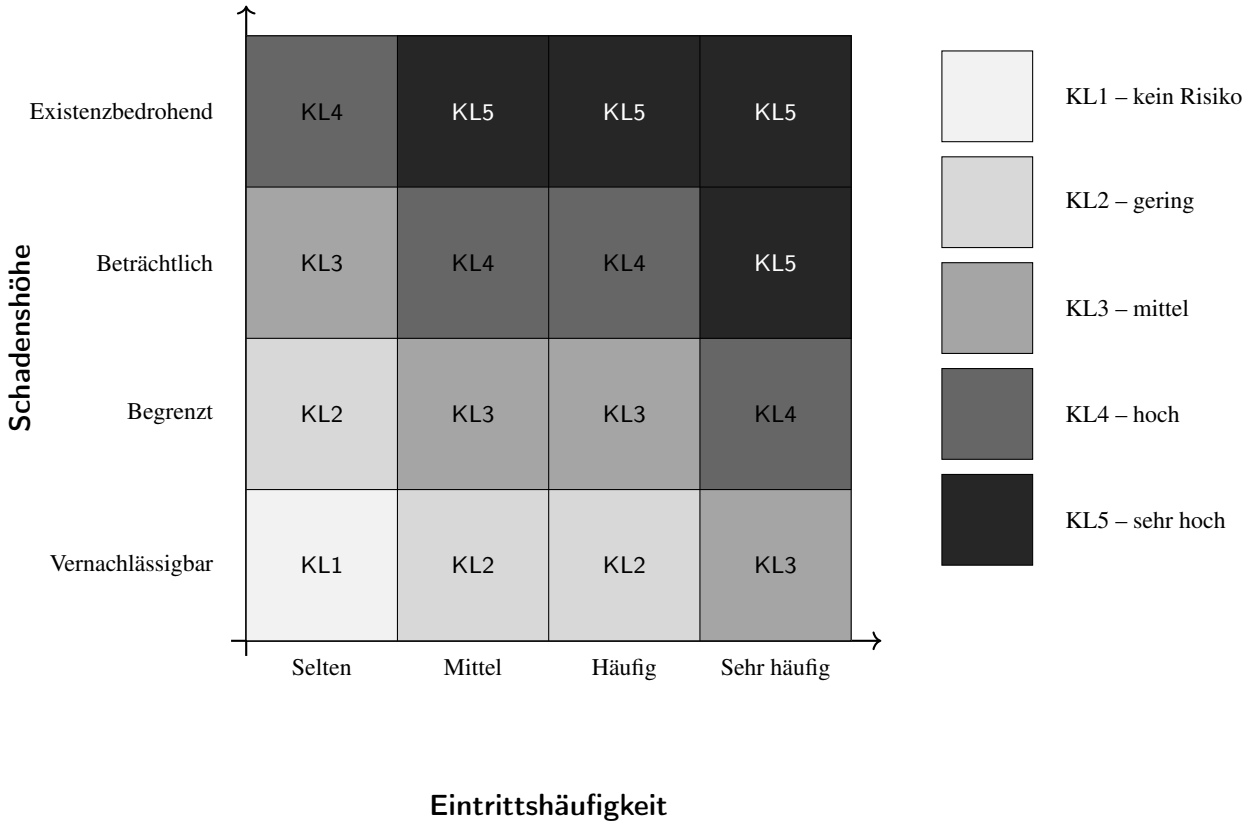

Abbildung 4.6: Matrix der Risikoklassen in Anlehnung an BSI-Standard 200-3

Ergebnis: Risikoklassenschema für das Framework

Die Risikomatrix in Abbildung 4.6 definiert die Risikoklassen als Kombination von Schadenshöhe und Eintrittshäufigkeit. Die Risikoklassen dienen im Framework als Steuergröße. Die folgenden Beschreibungen legen fest, welche Ziele das Framework pro Klasse verfolgt. Die Ziele entstehen aus der Matrix, nicht aus der BSI-Definition der Klassen.

Die Risikomatrix legt das Risikoniveau fest. Tabelle 4.3 beschreibt die Ziele, die das Framework den fünf Risikoklassen zuordnet. Die Tabelle entsteht aus der Matrixklassifikation und definiert den Umgang des Frameworks mit den einzelnen Risikostufen. Sie erweitert somit die reine Risikodefinition um eine systeminterne Steuerlogik.

Risikoklasse	Ziel im Framework
1 – Kein Risiko	Das Framework akzeptiert das Risiko und überwacht den Zustand.
2 – Gering	Das Framework ergänzt begrenzte organisatorische oder technische Schritte und aktiviert ein Monitoring.
3 – Mittel	Das Framework plant und priorisiert weitergehende Maßnahmen.
4 – Hoch	Das Framework senkt das Risiko zeitnah, zum Beispiel durch stärkere Zugangskontrollen oder Segmentierung.
5 – Sehr hoch	Das Framework setzt unverzügliche Maßnahmen wie Blockierungen oder Isolation um und informiert das Management.

Tabelle 4.3: Risikoklassen und zugeordnete Ziele im Framework

4.2.2 Risikoklassen und erwartete Policy-Kategorien

Die in Abschnitt 4.2.1 definierten Risikoklassen bilden gemeinsam mit dem in Abschnitt 4.2.3 eingeführten Metamodell der SASE-Steuerobjekte den Rahmen für einen vendorunabhängigen Policy-Layer. Dieses Unterkapitel schließt die Lücke zwischen abstrakter Risikodefinition und technischem Metamodell: Ausgehend von den Risikoklassen werden generische, vendorunabhängige Policy-Kategorien abgeleitet, die im folgenden Abschnitt auf konkrete Steuerobjekte des Metamodells abgebildet werden.

Die Ableitung orientiert sich primär an BSI IT-Grundschutz, International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 27002:2022 und den Center for Internet Security (CIS) Critical Security Controls.[11] Für den Zero-Trust-Kontext dient National Institute of Standards and Technology (NIST) SP 800-207 als Referenz.[12] ENISA-Guidelines ergänzen segmentierungs- und evidenzbasierte Betriebsmaßnahmen.[13]

[11] vgl. Bundesamt für Sicherheit in der Informationstechnik, *IT-Grundschutz-Kompendium*; International Organization for Standardization, *ISO/IEC 27002:2022 — Information security, cybersecurity and privacy protection — Information security controls*; Center for Internet Security, *CIS Critical Security Controls® v8 CIS Critical Security Controls*.

[12] vgl. Rose u. a., *Zero Trust Architecture*.

[13] vgl. European Union Agency for Cybersecurity (ENISA), *TECHNICAL IMPLEMENTATION GUIDANCE*; European Union Agency for Cybersecurity (ENISA), *ENISA Threat Landscape 2024*, S.21-S.28.

Die Übertragung der Risikoklassen auf generische Policy-Kategorien erfolgt in drei Schritten:

1. Zunächst wird je Risikoklasse ein dominantes Schutzziel definiert (z. B. KL 2: »Angriffsfläche reduzieren«).
2. Anschließend werden die genannten Standards (insb. BSI, NIST, CIS) systematisch auf konkrete technische Kontrollen hin analysiert, die auf dieses Schutzziel einzahlen. Die identifizierten Bausteine (z. B. BSI NET.3.2 »Firewall«), Controls (z. B. CIS Safeguard 13.x) und Prinzipien (z. B. NIST, Zero Trust Architecture (ZTA)) werden den relevanten SASE-Komponenten (FWaaS, SWG, ZTNA, SD-WAN, CASB/SaaS, Threat Prevention/DNS) zugeordnet.
3. Im letzten Schritt werden diese Kontrollen zu vendorunabhängigen Policy-Kategorien aggregiert, die eine konsistente Eskalationslogik über alle Risikoklassen hinweg abbilden. Die resultierenden Kategorien sind in Tabelle 4.2.2 zusammengefasst und bilden die Policy-Schicht im technischen Metamodell.

Mit steigender Risikoklasse erhöht sich sowohl die Tiefe der technischen Maßnahmen als auch der Grad ihrer Laufzeitdynamik: von statischen Grundschutz-Konfigurationen (KL 1) über kontextbasierte, applikations- und identitätsbewusste Kontrollen (KL 2–4) bis hin zur weitgehenden Isolation im Ereignisfall (KL 5).

Disclaimer

Die im Folgenden beschriebenen Policy-Kategorien stellen generische Referenzen dar. Die konkrete Ausgestaltung und Parametrisierung der Maßnahmen obliegt den einzelnen Organisationen und richtet sich nach deren Geschäftsprozessen, Risikobereitschaft und regulatorischen Anforderungen.

Risikoklasse 1 — Kein Risiko

Ziel: Grundschutz sicherstellen und beobachten.

FWaaS: Eingehende Verbindungen blockieren; ausgehende Standarddienste mit minimalem Port-/Protokollumfang erlauben; Network Time Protocol (NTP)-Synchronisation für Log-Kohärenz.[14]

SWG: HTTP/HTTPS-Proxy ohne SSL-Inspection aktivieren; Kategorien „Malware/Phishing“ blockieren; Baseline-DNS-Filtering umsetzen.[15]

ZTNA: Einfache Identitätsprüfung aktivieren; Zugriff auf nicht-kritische Ressourcen freigeben.[16]

SD-WAN: Statische Pfade nutzen; keine Priorisierung; Basis-Telemetrie erfassen.[17]

CASB/SaaS: App-Inventar erfassen; Audit-Logging aktivieren.[18]

Threat Prevention/DNS: Signaturbasierte Anti-Virus (AV)-Prüfung aktivieren; Sinkhole für bekannte Schad-Domains konfigurieren.[19]

Die Maßnahmen in Risikoklasse 1 unterstützen einen stabilen Grundschutz mit geringer Eingriffstiefe. Die Konfiguration orientiert sich an grundlegenden Anforderungen aus BSI IT-Grundschutz und ISO/IEC 27002, die Systeme ohne kontextabhängige Entscheidungen absichern. Die Stufe definiert den Ausgangszustand, auf dem alle weiteren Eskalationsstufen aufbauen.

[14] vgl. BSI, Baustein NET.3.2 „Firewall“ (Anf. A2–A4, A9, A20, A22) und OPS.1.2.6 „NTP-Zeitsynchronisation“ (Anf. A1–A4). Bundesamt für Sicherheit in der Informationstechnik, *IT-Grundschutz-Kompendium*.

[15] vgl. BSI, Baustein OPS.1.1.4 „Schutz vor Schadprogrammen“ und APP.3.6 „DNS-Server“ (Anf. A4, A7, A13–A16). Bundesamt für Sicherheit in der Informationstechnik, *IT-Grundschutz-Kompendium*.

[16] vgl. NIST SP 800-207, Kap. 2–3 (Grundprinzipien). Rose u. a., *Zero Trust Architecture*.

[17] vgl. BSI, Baustein NET.3.1 „Router und Switches“ (Anf. zu Netzarchitektur). Bundesamt für Sicherheit in der Informationstechnik, *IT-Grundschutz-Kompendium*.

[18] vgl. ISO/IEC 27002:2022, Control 8.15 (Logging). International Organization for Standardization, *ISO/IEC 27002:2022 — Information security, cybersecurity and privacy protection — Information security controls*.

[19] vgl. BSI, Bausteine OPS.1.1.4 und APP.3.6 (Anf. A4, A7, A15–A17). Bundesamt für Sicherheit in der Informationstechnik, *IT-Grundschutz-Kompendium*.

Risikoklasse 2 — Gering

Ziel: Angriffsfläche reduzieren und Prävention ergänzen.

FWaaS: Ausgehende Ports auf definierte Protokolle begrenzen; Geo-/Autonomous System Number (ASN)-Filter einsetzen; Egress-Kontrollen durchsetzen.[20]

SWG: Blacklists und Safe-Browsing-Feeds beziehen; risikoreiche Dateitypen blockieren.[21]

ZTNA: Anmeldung verpflichten; einfache Gerätekonformität (z. B. Operating System (OS)-Version) prüfen.[22]

SD-WAN: Grobe Segmentierung (Benutzer/Server) einführen; Quality of Service (QoS) für geschäftskritische SaaS anwenden.[23]

CASB/SaaS: Schatten-IT erkennen; riskante OAuth-Grants kennzeichnen.[24]

Threat Prevention/DNS: Web-/Mail-AV aktiv; DNS-Allow- und Blocklisten pflegen.[25]

Die Maßnahmen in Risikoklasse 2 erweitern den Grundschutz um eine moderate Reduktion der Angriffsfläche. Die Konfiguration nutzt vorliegende Informationen wie Ports, Protokolle, Kategorien und Listen, ohne tiefgreifende Inhaltsanalysen oder starke Nutzereinschränkungen einzuführen. Die Stufe erweitert die Prävention, ohne den operativen Aufwand deutlich zu erhöhen.

[20] vgl. BSI, NET.3.2 „Firewall“, Anf. A2–A4, A19–A20; CIS Controls v8, Safeguards 3, 4, 13. Bundesamt für Sicherheit in der Informationstechnik, *IT-Grundschutz-Kompendium*; Center for Internet Security, *CIS Critical Security Controls® v8 CIS Critical Security Controls*.

[21] vgl. BSI, OPS.1.1.4 „Schutz vor Schadprogrammen“, Anf. A3, A5–A7, A10–A14. Bundesamt für Sicherheit in der Informationstechnik, *IT-Grundschutz-Kompendium*.

[22] vgl. NIST SP 800-207, Kap. 3–4 (Policy Decision Points, subject & device attributes). Rose u. a., *Zero Trust Architecture*.

[23] vgl. BSI, NET.3.1 „Router und Switches“ und NET.3.2 „Firewall“ (Anf. NET.3.2.A16, A30 zu Segmentierung und Bandbreitenmanagement). Bundesamt für Sicherheit in der Informationstechnik, *IT-Grundschutz-Kompendium*.

[24] vgl. ISO/IEC 27002:2022, Controls zu Cloud-Nutzung; CSA, Zero Trust Guidance v1.1 (SaaS-/Identity-Sicht). International Organization for Standardization, *ISO/IEC 27002:2022 — Information security, cybersecurity and privacy protection — Information security controls*; Cloud Security Alliance, *Zero Trust Guiding Principles v1.1*.

[25] vgl. BSI, OPS.1.1.4 und APP.3.6 (Anf. A4, A7, A13–A17). Bundesamt für Sicherheit in der Informationstechnik, *IT-Grundschutz-Kompendium*.

Risikoklasse 3 — Mittel

Ziel: Missbrauch erschweren und Erkennung beschleunigen.

FWaaS: IDS/IPS aktivieren; applikationsbewusste Regeln einsetzen; Session-Timeouts für privilegierte Zonen konfigurieren.[26]

SWG: SSL-Inspection selektiv für risikoreiche Kategorien; Data Loss Prevention (DLP)-Basisregeln für ausgehenden Web-Traffic.[27]

ZTNA: Mehrfaktor-Authentisierung für sensible Anwendungen; Gerätezustand (*posture*) prüfen; Sitzungen an Identität und Gerät binden.[28]

SD-WAN: Feingranulare Segmentierung (z. B. Abteilungen); Pfadwahl nach L7-Anwendungsprofilen.[29]

CASB/SaaS: Konfigurations-Baselines (Single Sign-On (SSO), Logging, Aufbewahrung) prüfen; anomale Logins/Downloads melden.[30]

Threat Prevention/DNS: Sandboxing/Detonation für unbekannte Dateien; DNS-Tunneling-Indikatoren erkennen.[31]

Die Maßnahmen in Risikoklasse 3 verstärken den Schutz durch kontext- und identitätsbezogene Kontrollen. Die Konfiguration nutzt zusätzlich zu netzwerkbasierten Filtern verstärkt Sitzungs-, Identitäts- und Verhaltensmerkmale. Diese Stufe entspricht den erhöhten Anforderungen in NIST SP 800-207 und CIS Controls, die eine frühzeitige Erkennung und Erschwerung von Missbrauch unterstützen.

[26] vgl. BSI, NET.3.2 „Firewall“, H-Anforderungen (A19–A23, A27–A30); CIS Controls v8, Safeguards 13.x (Network Monitoring & Defense). Bundesamt für Sicherheit in der Informationstechnik, *IT-Grundschutz-Kompendium*; Center for Internet Security, *CIS Critical Security Controls® v8 CIS Critical Security Controls*.

[27] vgl. BSI, NET.3.2 „Firewall“, Anf. A21 (Entschlüsselung) und OPS.1.1.4 (Analyse). Bundesamt für Sicherheit in der Informationstechnik, *IT-Grundschutz-Kompendium*.

[28] vgl. NIST SP 800-207, Kap. 3–4 (Continuous Authentication); NCSC UK, Zero-Trust-Prinzipien. Rose u. a., *Zero Trust Architecture*; National Cyber Security Centre (UK), *Zero Trust Architecture: Design Principles*.

[29] vgl. BSI, NET.3.1 und NET.3.2 (mehrstufige Architekturen, Bandbreitenmanagement). Bundesamt für Sicherheit in der Informationstechnik, *IT-Grundschutz-Kompendium*.

[30] vgl. CIS Controls v8, Safeguards 5.x, 16.x (Account Management, Application Security Monitoring). Center for Internet Security, *CIS Critical Security Controls® v8 CIS Critical Security Controls*.

[31] vgl. BSI, OPS.1.1.4 (Anf. A10–A14) und APP.3.6 (Anf. A7, A15–A17). Bundesamt für Sicherheit in der Informationstechnik, *IT-Grundschutz-Kompendium*.

Risikoklasse 4 — Hoch

Ziel: Laterale Bewegung unterbinden und Datenabfluss verhindern.

FWaaS: Zonen strikt isolieren; Ost–West-Verkehr nur explizit freigeben; anomale Flüsse adaptiv blockieren; Outbound-Rate-Limits erzwingen.[32][33]

SWG: SSL-Inspection für definierte Zielgruppen vollständig; Content-Disarm-and-Reconstruct für Office/Portable Document Format (PDF); strikte Dateityp-Kontrollen.[34]

ZTNA: Kontext-Policies (Ort, Zeit, Risiko-Score); Just-in-Time-Rechte; kontinuierliche Verifikation.[35]

SD-WAN: Mikrosegmentierung standortübergreifend; Policy-Tags; Echtzeit-Telemetrie-Korrelation.[36][37]

CASB/SaaS: DLP-Regeln für Personally Identifiable Information (PII)/Payment Card Industry (PCI); Upload-/Share-Kontrollen; Token-Risiko-Bewertung.[38]

Threat Prevention/DNS: Resolver-Policy strikt (z. B. DNS over TLS (DoT)/DNS over HTTPS (DoH)); Command-and-Control-Indikatoren blockieren.[39]

Die Maßnahmen in Risikoklasse 4 fokussieren auf die Begrenzung lateraler Bewegungen und die Kontrolle sensibler Datenströme. Die Konfiguration stärkt Segmentierung, Inhaltsprüfung und Datenabflusskontrollen und folgt damit Empfehlungen von BSI, NIST und European Union Agency for Cybersecurity (ENISA) zu hochkritischen Lagen. Die Stufe ergänzt die Kontrollen der Klassen 1 bis 3 um deutlich restriktivere Zugriffs- und Verkehrsregeln.

[32] vgl. BSI, NET.3.2 „Firewall“, Anf. A16 (P-A-P-Struktur), A27–A30 (mehrstufige Architektur, Bandbreitenmanagement). Bundesamt für Sicherheit in der Informationstechnik, *IT-Grundschutz-Kompendium*.

[33] vgl. ENISA, Threat Landscape 2024, Kap. 8 (DDoS). European Union Agency for Cybersecurity (ENISA), *ENISA Threat Landscape 2024*.

[34] vgl. BSI, NET.3.2 (Anf. A21) und OPS.1.1.4 (Umgang mit nicht vertrauensw. Dateien). Bundesamt für Sicherheit in der Informationstechnik, *IT-Grundschutz-Kompendium*.

[35] vgl. NIST SP 800-207, Kap. 4 (Policy Engine, kontinuierliche Evaluation); NCSC UK, Zero-Trust-Prinzipien (JIT-Rechte). Rose u. a., *Zero Trust Architecture*; National Cyber Security Centre (UK), *Zero Trust Architecture: Design Principles*.

[36] vgl. BSI, NET.3.1 und NET.3.2 (VLAN-Segmentierung). Bundesamt für Sicherheit in der Informationstechnik, *IT-Grundschutz-Kompendium*.

[37] vgl. ENISA, Kap. 6.8 (network segmentation). European Union Agency for Cybersecurity (ENISA), *TECHNICAL IMPLEMENTATION GUIDANCE*.

[38] vgl. ISO/IEC 27002:2022 (Controls zu DLP, Cloud-Nutzung); CSA, Zero Trust Guidance v1.1 (Data/SaaS-Säule, API-/Token-Härtung). International Organization for Standardization, *ISO/IEC 27002:2022 — Information security, cybersecurity and privacy protection — Information security controls*; Cloud Security Alliance, *Zero Trust Guiding Principles v1.1*.

[39] vgl. BSI, APP.3.6 (Anf. A4, A7, A13–A18) und OPS.1.1.4 (C2-Kommunikation). Bundesamt für Sicherheit in der Informationstechnik, *IT-Grundschutz-Kompendium*.

Risikoklasse 5 — Sehr hoch

Ziel: Risiko technisch sofort senken und isolieren.

FWaaS: Quarantäne-Netze aktivieren; nicht freigegebene Verbindungen sofort blockieren; temporär „deny-all außer Whitelist“ umsetzen.[40]

SWG: Uploads zu Cloud-Speichern sperren; Nur-Lesen-Zugänge erzwingen; erweiterte DLP-Profile mit Verschlüsselung.[41]

ZTNA: Sitzungen re-autorisieren; Rechte schrittweise reduzieren; Geräte in Quarantäne versetzen.[42]

SD-WAN: Betroffene Segmente isolieren; Blackholing für durch Indikatoren erkannte Ziele.[43]

CASB/SaaS: Aktive Sitzungen widerrufen; OAuth-Token invalidieren; API-basierte Zugriffe temporär sperren.[44]

Threat Prevention/DNS: Sperrlisten vollständig anwenden; Indicator of Compromise (IOC)-basierte Sperren priorisieren.[45]

Die Maßnahmen in Risikoklasse 5 zielen auf eine unmittelbare Senkung des Risikos durch Isolation und Entzug von Rechten. Die Konfiguration orientiert sich an Notfall- und Incident-Response-Empfehlungen der herangezogenen Standards und setzt temporär maximal restriktive Regeln um. Die Stufe nutzt alle vorgelagerten Kontrollen und ergänzt diese um Quarantäne-, Sperr- und Abschaltmechanismen.

[40] vgl. BSI, NET.3.2 „Firewall“, Anf. A1–A4, A16, A30–A32 (restriktive Regeln, Notfallvorsorge). Bundesamt für Sicherheit in der Informationstechnik, *IT-Grundschutz-Kompendium*.

[41] vgl. ISO/IEC 27002:2022 (Controls zu DLP); CIS Controls v8, Safeguards 13.x, 15.x. International Organization for Standardization, *ISO/IEC 27002:2022 — Information security, cybersecurity and privacy protection — Information security controls*; Center for Internet Security, *CIS Critical Security Controls® v8 CIS Critical Security Controls*.

[42] vgl. NIST SP 800-207, Kap. 4–5 (Session-Termination); ENISA, Kap. 3.3–3.6 (Incident Response). Rose u. a., *Zero Trust Architecture*; European Union Agency for Cybersecurity (ENISA), *TECHNICAL IMPLEMENTATION GUIDANCE*.

[43] vgl. BSI, NET.3.2 (Notfallvorsorge); ENISA, Kap. 6.8 (Isolation) und Kap. 8 (Blackholing). Bundesamt für Sicherheit in der Informationstechnik, *IT-Grundschutz-Kompendium*; European Union Agency for Cybersecurity (ENISA), *TECHNICAL IMPLEMENTATION GUIDANCE*; European Union Agency for Cybersecurity (ENISA), *ENISA Threat Landscape 2024*.

[44] vgl. ISO/IEC 27002:2022 (Controls zu Access Management); CSA, Zero Trust Guidance v1.1 (SaaS/API-Sicherheit). International Organization for Standardization, *ISO/IEC 27002:2022 — Information security, cybersecurity and privacy protection — Information security controls*; Cloud Security Alliance, *Zero Trust Guiding Principles v1.1*.

[45] vgl. BSI, APP.3.6 (Anf. A7, A15–A20) und OPS.1.1.4 (IOC-Erkennung). Bundesamt für Sicherheit in der Informationstechnik, *IT-Grundschutz-Kompendium*.

Diskussion der generischen Eskalationslogik

Die detaillierte Auflistung der Maßnahmen zeigt ein kumulatives Eskalationsmuster, das für das spätere technische Mapping zentral ist:

- **Risikoklasse 1 (Kein Risiko):** Definiert einen Grundschutz- und Monitoring-Zustand. Die Maßnahmen sind weitgehend statisch (z. B. Standard-Firewall-Regeln, Basis-Logging).

- **Risikoklasse 2 (Gering):** Ergänzt eine proaktive, jedoch weiterhin überwiegend statische Prävention (z. B. Einschränkung der Angriffsfläche durch Ports/Protokolle, Blockieren bekannter schädlicher Inhalte).

- **Risikoklasse 3 (Mittel):** Führt erstmals dynamischere, applikations- und kontextbewusste Kontrollen ein (z. B. IDS/IPS, Multi-Faktor-Authentisierung (MFA), Posture-Checks). Der Fokus verschiebt sich von reinem Netzwerkfokus hin zu Identität und Kontext.

- **Risikoklasse 4 (Hoch):** Eskaliert zu strikter Segmentierung (Ost–West-Restriktionen) und Datenkontrolle (z. B. DLP), um laterale Bewegungen und Exfiltration aktiv zu unterbinden.

- **Risikoklasse 5 (Sehr hoch):** Stellt die höchste Stufe der automatisierten Reaktion dar, bei der es um sofortige Isolation und Entzug von Rechten geht (Quarantäne, Blackholing, Session-Termination).

Diese Stufenlogik bildet die Policy-Perspektive des in Unterabschnitt 4.2.3 beschriebenen technischen Metamodells: Jede höhere Risikoklasse ergänzt zusätzliche, restriktivere Kontrollen, ohne die Anforderungen der niedrigeren Klassen zu ersetzen.

Risikoklasse	Erwartete Policy-Kategorien (vendorunabhängig)
KL1 – Kein Risiko	Basis-Logging und Monitoring; Grundschutz-Konfigurationen ohne zusätzliche Filter.
KL2 – Gering	Einfache Firewall-Filter (Port/Protokoll); Blacklists für schädliche Domains; Baseline-DNS-Filtering.
KL3 – Mittel	Mehrfaktor-Authentisierung, Session-Timeouts, IDS/IPS, erweiterte Protokollierung.
KL4 – Hoch	Strikte Segmentierung (Zonenbildung), SSL-Inspection für definierte Zielgruppen, Just-in-Time-Access, Anomalieerkennung im Traffic, Outbound-Rate-Limits.
KL5 – Sehr hoch	Quarantäne, Portsperren, IOC-/Command and Control (C2)-basierte Sperren, Zero-Trust-Durchsetzung mit Echtzeit-Policy-Updates.

Tabelle 4.4: Risikoklassen und generische Policy-Erwartungen

Die in Tabelle 4.2.2 zusammengefassten Policy-Kategorien konkretisieren die Eskalationsstufen der Risikoklassen und bilden die Brücke zwischen abstrakter Risikodefinition und der Abbildung auf konkrete SASE-Steuerobjekte. Die Risikoklassen sind kumulativ zu verstehen: Eine Maßnahme der Klasse 5 impliziert die Umsetzung aller Anforderungen der Klassen 1 bis 4; mit zunehmender Risikoklasse steigt damit sowohl die Intensität als auch die Vollständigkeit der zu aktivierenden Kontrollen.

4.2.3 Mapping der Risikoklassen auf SASE-Steuerobjekte

Nachdem die Risikoklassen und die erwarteten technischen Maßnahmen als Eingangsgrößen definiert sind, entwirft der nächste Schritt die zentrale Mapping-Logik des Frameworks. Diese legt fest, wie eine Änderung der Risikoklasse in eine Anpassung von SASE-Policies übersetzt wird. Die Zuordnung ist das Kernstück des Orchestrators und berücksichtigt die Erkenntnisse aus der Komponentenanalyse (Abschnitt 4.1).

Das Mapping bildet die zentrale Logik des zu entwickelnden Artefakts: Es übersetzt die abstrakten, in Unterabschnitt 4.2.1 definierten Risikoklassen (KL 1–5) in konkrete, aufrufbare API-Operationen, die in der Komponentenanalyse (Abschnitt 4.1) als steuerbar identifiziert wurden.

Kumulative Eskalationslogik

Ein zentrales Gestaltungsprinzip (vgl. Abschnitt 3.1) ist die kumulative Natur der Risikoklassen. Ein Wechsel von Risikoklasse 2 (KL 2) auf 3 (KL 3) bedeutet nicht, dass KL 2-Maßnahmen ersetzt werden. Stattdessen behält das Framework alle aktiven KL 1- und KL 2-Kontrollen bei und fügt die spezifischen KL 3-Maßnahmen **zusätzlich** hinzu. Bei einer Deeskalation (z. B. KL 3 $\rightarrow$ KL 2) werden nur die spezifischen KL 3-Maßnahmen widerrufen. Tabelle 4.5 stellt dieses Metamodell dar.

Stufe 1: Konzeptionelles Metamodell (Mapping-Tabelle)

Die Tabelle 4.5 stellt das konzeptionelle Metamodell dar. Sie analysiert jede einzelne generische Maßnahme aus Unterabschnitt 4.2.2 und prüft sie gegen die identifizierten API-Steuerobjekte der Prisma SASE-Plattform. Die Spalte „Laufzeit-fähig?“ bewertet, ob die Maßnahme dynamisch zur Laufzeit – und damit z.B.: Automatisiert durch ein Orchestrierungs-Framework – umgesetzt werden kann. Dies dient der direkten Beantwortung der Forschungsfrage.

Anmerkung: **GAP** = Technisch nicht umsetzbar aufgrund fehlender API-Endpunkte (vgl. Analyse Abschnitt 4.1).

Klasse	Maßnahme (kumulativ)	Komponente	Steuerobjekt	Laufzeit-fähig?
KL 1	Grundschutz Firewall (Ingress deny, Egress Standard)	FWaaS	`ngfwsecuritypolicysets` (Basis-Regelwerk)	Ja (Initial)
	SWG: Proxy (ohne SSL-I), Malware-Kategorien blockieren	SWG	Kein API-Objekt zur Policy-Steuerung identifiziert	**GAP**
	ZTNA: Einfache Identitätsprüfung, Zugriff auf Unkritisches	ZTNA	`iotservices` (Default Confidence Score)	Ja (Initial)
	SD-WAN: Statische Pfade, keine Priorisierung, Basis-Telemetrie	SD-WAN	`staticroutes` `networkpolicysets` (Standard-Pfadwahl)	Ja (Initial)
	CASB/SaaS: App-Inventar erfassen, Audit-Logging aktivieren	CASB	`/sspm/api/v1/apps` (Monitoring via GET)	Nein (Monitoring)
	DNS: Signaturbasierte AV-Prüfung, Sinkhole für Schad-Domains	SD-WAN (DNS)	`dnsserviceprofiles` (Zuweisung eines Basis-Profils)	Ja (Initial)
KL 2	FWaaS: Ports begrenzen, Geo-/ASN-Filter, Egress-Kontrollen	FWaaS	`ngfwsecuritypolicyrules` (Erstellen/Ändern von Regeln mit Geo-/Prefix-Objekten)	Ja (POST/PUT)
	SWG: Blacklists beziehen, riskante Dateitypen blockieren	SWG	Kein API-Objekt zur Policy-Steuerung identifiziert	**GAP**
	ZTNA: Anmeldung verpflichten, Gerätekonformität prüfen	ZTNA	`iotservices` (Anpassen des Confidence Score)	Ja (POST/PUT)
	SD-WAN: Grobe Segmentierung, QoS für SaaS	SD-WAN	`networkcontexts` `prioritypolicysets` (QoS-Regeln erstellen/-zuweisen)	Ja (POST/PUT)

Klasse	Maßnahme (kumulativ)	Komponente	Steuerobjekt	Laufzeit-fähig?
	CASB/SaaS: Schatten-IT erkennen, riskante OAuth-Grants	CASB	`/sspm/api/v1/apps` `/sspm/api/v1/plugin-users`	Nein (Monitoring via GET)
	DNS: Web/Mail AV, DNS Allow-/Blocklisten	SD-WAN (DNS)	`dnsserviceprofiles` (Aktualisieren von Listen im Profil)	Ja (PUT)
KL 3	FWaaS: IDS/IPS aktivieren, App-Regeln, Session-Timeouts	FWaaS	`ngfwsecuritypolicyrules` (Modifizieren von Regeln, um Security-Profile zu laden)	Ja (PUT)
	SWG: Selektive SSL-Inspection, DLP-Basisregeln	SWG	Kein API-Objekt zur Policy-Steuerung identifiziert	**GAP**
	ZTNA: MFA erzwingen, Posture prüfen	ZTNA / CASB	`mfa_activity/count_by_app_type`	Indirekt (nur Monitoring via GET)
	SD-WAN: Feingranulare Segmentierung (Abteilungen)	SD-WAN	`networkcontexts` (Erstellen/Anpassen von Segmenten)	Ja (POST/PUT)
	CASB/SaaS: Konfigurations-Baselines prüfen, Anomale Logins melden	CASB	`/sspm/api/v1/apps/{id}/settings` `saas_instances/.../saas_activity`	Nein (Monitoring via GET)
	DNS: Sandboxing, DNS-Tunneling-Erkennung	SD-WAN	`dnsserviceprofiles` (Zuweisung von Advanced-Profilen)	Ja (PUT)
KL 4	FWaaS: Zonen strikt isolieren (Ost-West deny), Outbound-Rate-Limits	FWaaS	`ngfwsecuritypolicyrules` (Erstellen von ‘deny‘-Regeln zw. Zonen) `elementsecurityzones` (Referenz)	Ja (POST)
	SWG: SSL-Inspection (voll), CDR für Office/PDF	SWG	Kein API-Objekt zur Policy-Steuerung identifiziert	**GAP**

Klasse	Maßnahme (kumulativ)	Komponente	Steuerobjekt	Laufzeit-fähig?
	ZTNA: Kontext-Policies (Ort, Zeit, Score), JIT-Rechte	ZTNA	`iotservices` (Dynamisches Anpassen von Scores basierend auf Kontext)	Ja (POST/PUT)
	SD-WAN: Mikrosegmentierung (standortübergreifend)	SD-WAN	`networkcontexts`	Ja (POST/PUT)
	CASB/SaaS: DLP-Regeln (PII/PCI), Upload-/Share-Kontrollen	CASB	Kein API-Objekt zur Policy-Steuerung identifiziert	**GAP**
	DNS: Resolver-Policy strikt (DoT/DoH), C2-Block	SD-WAN	`dnsserviceprofiles` (Zuweisung von „Strict“-Profil)	Ja (PUT)
KL 5	FWaaS: Quarantäne-Netze, „deny-all außer Whitelist“	FWaaS	`ngfwsecuritypolicyrules` (Erstellen von „Block-Host“-Regel mit Top-Priorität)	Ja (POST)
	SWG: Uploads sperren, Nur-Lesen-Zugänge	SWG	Kein API-Objekt zur Policy-Steuerung identifiziert	**GAP**
	ZTNA: Sitzungen re-autorisieren, Geräte-Quarantäne	ZTNA	`iotservices` (Setzen des Confidence Score auf 0)	Ja (POST/PUT)
	SD-WAN: Segmente isolieren, Blackholing	SD-WAN	`staticroutes` (Erstellen von Blackhole-Route)	Ja (POST)
	CASB/SaaS: Sitzungen widerrufen, OAuth-Token invalidieren	CASB	Kein direkter API-Endpunkt. Indirekte Steuerung via: `saas_instances/.../tickets`	Indirekt (nur Ticket via POST)
	DNS: Sperrlisten (IOCs) priorisieren	SD-WAN	`dnsserviceprofiles` (Aktualisieren von Listen im Profil)	Ja (PUT)

Tabelle 4.5: Detailliertes Mapping-Metamodell der Risikoklassen auf SASE-Steuerobjekte

Analyse der Mapping-Ergebnisse

Die Tabelle in Tabelle 4.5 liefert drei zentrale Erkenntnisse für die Beantwortung der Forschungsfragen:

1. **Bewertung der Hypothese H1 und Einordnung der Forschungsfrage**
   H1: *„Die für die fünf Risikoklassen erforderlichen Schutzmaßnahmen lassen sich auf generische SASE-Policy-Objekte abbilden."*
   Die Ergebnisse in Tabelle 4.5 zeigen keine Bestätigung dieser Hypothese in der formulierten Allgemeinheit. Generische Maßnahmen für die Kernkomponenten FWaaS, SD-WAN und ZTNA lassen sich zwar auf zur Laufzeit steuerbare API-Objekte abbilden. Steuerungsmöglichkeiten in weiteren Domänen fehlen jedoch, vor allem in SWG und CASB. Diese Gegenbeispiele widerlegen H1.
   Da H1 die operative Form der Forschungsfrage darstellt, ist auch diese in ihrer allgemeinen Form zu verneinen. Eine vollständige Umsetzung der definierten risikobasierten Maßnahmen zur Laufzeit gelingt mit dem verfügbaren API-Umfang der untersuchten Plattform nicht.

2. **Grenzen einer risikobasierten Orchestrierung**
   Eine risikobasierte Orchestrierung aller SASE-Komponenten gelingt mit der analysierten API nicht. Die Analyse zeigt Lücken in der Laufzeit-Steuerbarkeit. Die SWG-Komponente bietet keine Endpunkte zur dynamischen Anpassung zentraler Policy-Regeln wie Sperrung von Dateitypen oder Aktivierung von SSL-Inspection. Die CASB-Komponente ermöglicht überwiegend Monitoring über GET-Operationen und bietet keine direkte Durchsetzung von Maßnahmen wie Sperrung von OAuth-Grants oder Erzwingung von MFA.
   Der vorhandene API-Umfang reicht für eine risikobasierte Steuerung aller SASE-Komponenten nicht aus.

3. **Implikationen für das Orchestrator-Design und für das Maßnahmenspektrum**
   Einzelne kritische Maßnahmen besitzen keinen direkten API-Befehl. Die `iotservices`-Objekte in der Prisma-SASE-API bilden einen Gerätekontext mit einem „Confidence Score“, der in ZTNA-Policies als Attribut dient. Eine gezielte Absenkung dieses Scores versetzt ein Gerät faktisch in einen Quarantäne-Zustand, obwohl ein gesonderter „Quarantine“-Endpunkt nicht existiert. Diese Möglichkeit zeigt die grundsätzliche Umsetzbarkeit risikobasierter Reaktionen, erfordert jedoch zusätzliche Abstraktionslogik und erhöht die Komplexität im Orchestrator.

Zu beachten ist, dass die Ergebnisse stark von der zugrunde liegenden Definition des technischen Maßnahmenspektrums abhängen. Eine veränderte Festlegung der erwarteten Maßnahmen pro Risikoklasse hätte zu einem abweichenden Befund geführt. Die Analyse bezieht sich ausschließlich auf die API der Prisma-SASE-Plattform von Palo Alto Networks; die identifizierten Steuerungslücken sind daher nicht zwingend allgemeingültig. Andere Anbieter können über abweichende oder weitergehende API-Funktionen verfügen. Die Untersuchung folgt einem komponentenübergreifenden Verständnis der Risikoklassen und ihrer technischen Ausprägungen und stützt sich auf die analysierte Plattform. Eine weiterführende Diskussion der methodischen Entscheidungen und ihrer Folgen erfolgt in Kapitel 6.

Im Ergebnis markieren die Mapping-Befunde zugleich den Übergang von der analytischen Betrachtung des Metamodells hin zur konzeptionellen Ausgestaltung eines Orchestrators. Zwar ist das Metamodell konsistent anwendbar, doch begrenzen die identifizierten API-Lücken dessen operative Umsetzung.

Das folgende Kapitel stell die Entwicklungsschritte für die Systemarchitektur des Risk-Aware-SASE-Orchestrators dar. Es beschreibt Zielsetzung, Aufbau und Einbettung des Artefakts und demonstriert dessen Funktionsweise anhand eines exemplarischen Szenarios als Teil der konzeptionellen Evaluation.

5 Systemdesign nach dem 4+1 Modell

Dieses Kapitel bildet den Kern der DSR-Phase 3 (*Design und Entwicklung*). Es überführt das in Kapitel 4 entwickelte Metamodell in eine konzeptionell-technische Ausarbeitung des Lösungsartefakts „Risk-Aware SASE Orchestrator“.

Ziel ist die Konstruktion eines Systemdesigns, das die in Kapitel 3 definierten Gestaltungsprinzipien – insbesondere *Anbieterunabhängigkeit*, *Konsistenz* und *minimale Time-to-Policy* – technisch operationalisiert. Da die in Kapitel 1.5 beschriebenen Einschränkungen eine empirische Messung der Laufzeit nicht zulassen, erfolgt die Validierung der Realisierbarkeit durch ein detailliertes Systemdesign. Dieses Design erlaubt zugleich die Ableitung und Parametrisierung eines analytischen Performance-Modells, mit dem die *Time-to-Policy* rechnerisch bestimmt und in Kapitel 5.3 bewertet wird. Im Rahmen dieser Arbeit wird das Artefakt nicht implementiert, sondern als Soll-Architektur auf konzeptioneller Ebene entworfen. Die nachfolgenden Sichten beschreiben daher ein Systemdesign als Design-Blueprint, das als Grundlage für eine mögliche spätere Realisierung dienen kann.

Um die funktionalen, technischen und organisatorischen Anforderungen verschiedener Stakeholder adäquat abzubilden, folgt der Entwurf dem *4+1 Sichtenmodell der Softwarearchitektur* nach Kruchten[1]. Das Modell wurde gewählt, weil es funktionale und nicht-funktionale Anforderungen getrennt darstellt und ein szenariengetriebenes Vorgehen unterstützt. Die Trennung ist in dieser Arbeit besonders hilfreich, da die funktionale Sicht die Orchestrierungslogik (Risikoklasse → Policy-Delta → Durchsetzung) strukturiert, während die nicht-funktionale Sicht die Bewertung zentraler Qualitätsziele wie *minimale Time-to-Policy*, Robustheit und Wartbarkeit ermöglicht. Damit lassen sich die Hypothesen sowohl auf Ebene der Funktionsfähigkeit (H1/H3) als auch auf Ebene der Performance (H2) systematisch einordnen.

[1] vgl. Kruchten, *Architectural Blueprints-The "4+1"View Model of Software Architecture*, S. 42–50.

5.1 Methodische Begründung und Aufbau des Sichtenmodells

Die Architektur des Artefakts lässt sich nicht durch eine einzige Abstraktionsebene hinreichend beschreiben, weil sie mehrere orthogonale Fragestellungen gleichzeitig abdecken muss: (1) fachliche Orchestrierungslogik und Mapping (Risikoklasse → Policy-Delta), (2) technische Schnittstellen und Adapter pro SASE-Domäne, (3) Laufzeitablauf vom Trigger bis zur Durchsetzung (*Time-to-Policy*) sowie (4) Deployment- und Betriebsaspekte (z. B. Token-Caching, Fehlerbehandlung, Monitoring). Eine einzelne Darstellung würde diese Perspektiven vermischen und damit entweder unpräzise oder überladen. Ein einzelnes Diagramm führt bei Systemen mit mehreren Domänen und Qualitätszielen oft dazu, dass unterschiedliche Aspekte – fachliche Struktur, Laufzeitverhalten, Modulorganisation oder Deployment – in einer überladenen Darstellung verschwimmen[2]. Das hier entwickelte Artefakt erfüllt diese Bedingungen, da es mehrere SASE-Komponenten (z. B. FWaaS, SD-WAN, ZTNA, CASB/SWG) über heterogene APIs adressiert, *hybrides Enforcement* (technisch + organisatorisch) vorsieht und mit der *Time-to-Policy* ein explizites Performanceziel bewertet.

Das 4+1-Modell adressiert dieses Problem, indem es die Architektur in fünf komplementäre Sichten zerlegt. Jede Sicht bildet einen eigenen Stakeholder-Concern ab und ermöglicht es, funktionale Anforderungen und die in Kapitel 3.1 definierten Gestaltungsprinzipien systematisch zu prüfen.

Die konkrete Anwendung des 4+1-Modells auf das zu entwickelnde Artefakt ist in Abbildung 5.1 dargestellt. Die Abbildung verdeutlicht, dass jede der folgenden Sichten einen spezifischen Aspekt der Soll-Architektur adressiert und erst ihr Zusammenspiel ein vollständiges Bild des Artefakts ergibt. Sie dient damit als Orientierungsrahmen für die Struktur dieses Kapitels und verankert die Ableitung der Hypothesenbewertung in den einzelnen Sichten.

[2] vgl. Kruchten, *Architectural Blueprints-The "4+1"View Model of Software Architecture*, S. 42.

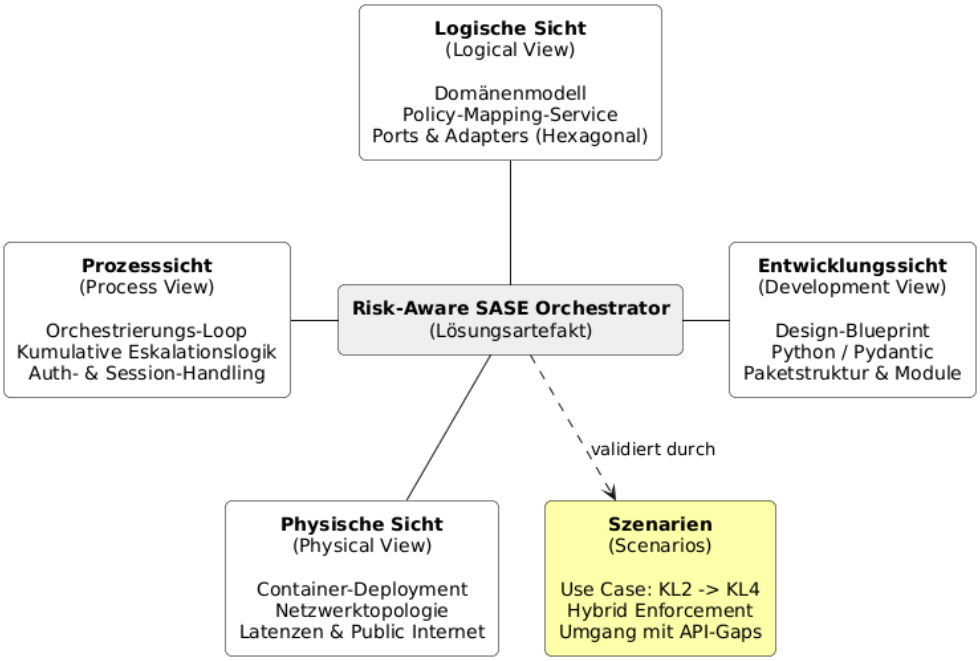

Abbildung 5.1: Strukturierung des Artefakts nach dem 4+1 Sichtenmodell

5.2 Architekturspezifikation nach dem 4+1-Modell

Die nachfolgende Spezifikation operationalisiert das Lösungsartefakt „Risk-Aware SASE Orchestrator“. Ziel des Entwurfs ist es, das in Kapitel 4.2 entwickelte Metamodell in eine robuste Softwarearchitektur zu überführen, die trotz der identifizierten technischen Limitationen (vgl. Unterabschnitt 4.2.3) funktionsfähig ist.

Die Zerlegung in fünf Sichten dient der Reduktion von Komplexität. Die in der Analyse festgestellte Heterogenität der SASE-APIs sowie fehlende Steuerungsoptionen (GAPs) erfordern eine strikte Trennung zwischen der fachlichen Entscheidungslogik (Risikoklassen-Mapping) und der technischen Ausführung (API-Calls). Die folgenden Unterkapitel adressieren diese Herausforderung aus unterschiedlichen Perspektiven:

- Die Logische Sicht (Unterabschnitt 5.2.1) definiert die statischen Bausteine und Schnittstellen. Sie legt den Fokus auf Modularität und die Kapselung der proprietären Anbieter-Logik.

- Die Prozesssicht (Unterabschnitt 5.2.2) beschreibt das dynamische Laufzeitverhalten, insbesondere den Orchestrierungs-Loop und das Handling von Latenzen, was für die spätere Performance-Evaluation (H2) relevant ist.

- Die Entwicklungs- und Physische Sicht (Unterabschnitt 5.2.3, Unterabschnitt 5.2.4) konkretisieren die Implementierungsdetails und die Deployment-Umgebung des Artefakts.

- Abschließend validieren die Szenarien (Unterabschnitt 5.2.5) das Zusammenspiel aller Sichten anhand eines konkreten Anwendungsfalls.

Diese multiperspektivische Darstellung ermöglicht es, sowohl die funktionalen Anforderungen der Risikosteuerung als auch die nicht-funktionalen Randbedingungen der API-Performance im Design zu berücksichtigen.

5.2.1 Logische Sicht (Logical View)

Die logische Sicht beschreibt die funktionale Zerlegung des Systems in diskrete Bausteine und deren statische Beziehungen. Um die in Kapitel 3 geforderten Prinzipien der *Anbieterunabhängigkeit* und *Erweiterbarkeit* technisch zu gewährleisten, folgt der Entwurf dem Muster der **Hexagonalen Architektur** (Ports and Adapters) nach Cockburn[3].

Der Stil wird gewählt, weil er eine stabile Domänenschicht gegenüber volatilen Integrationen (mehrere APIs, heterogene SASE-Domänen, Ticketing) priorisiert und damit die in dieser Arbeit geforderte Anbieterunabhängigkeit und Erweiterbarkeit architektonisch absichert.

Dieser Architekturstil isoliert die fachliche Domänenlogik (den Kern des Orchestrators) von externen Einflüssen wie Benutzerschnittstellen, Datenbanken oder spezifischen API-Implementierungen. Dies ist für das vorliegende Artefakt wesentlich, da die identifizierte Heterogenität der SASE-APIs eine strikte Kapselung der Kommunikationsschicht erfordert.

[3] vgl. Cockburn Alistair, „The Hexagonal (Ports & Adapters) Architecture “.

Das System gliedert sich in drei logische Bereiche:

1. **Domain Layer (Kern):** Enthält die geschäftslogischen Entscheidungsregeln und das in Kapitel 4.2.3 definierte Metamodell.
2. **Application Layer (Orchestrierung):** Steuert den Ablauf, verarbeitet Trigger und koordiniert die Transaktionen.
3. **Infrastructure Layer (Adapter):** Implementiert die technische Kommunikation zu externen Systemen (Security Information and Event Management (SIEM), Prisma SASE, Ticketing).

Architekturkomponenten und Zerlegung

Abbildung 5.2 zeigt die gewählte hexagonale Struktur und verdeutlicht insbesondere die Trennung von Domänenlogik und technischen Adaptern, die für die geforderte Anbieterunabhängigkeit entscheidend ist.

Im Zentrum steht der *Policy Mapping Service*, der als Single Source of Truth für die Zuordnung von Risikoklassen zu technischen Maßnahmen fungiert.

Trigger-Interface (Northbound Adapter) Der Einstiegspunkt in das System ist der `RiskWebhookListener`. In der Zielarchitektur fungiert er als *Driving Adapter* und nimmt standardisierte JSON-Payloads von externen Risiko-Quellen (z. B. SIEM, Security Orchestration, Automation, and Response (SOAR) oder manuellen Admin-Konsolen) entgegen. Er normalisiert eingehende Daten (z. B. „Critical“ $\rightarrow$ KL 5) und übergibt das gewünschte Ziel-Risikoniveau an den Application Layer.

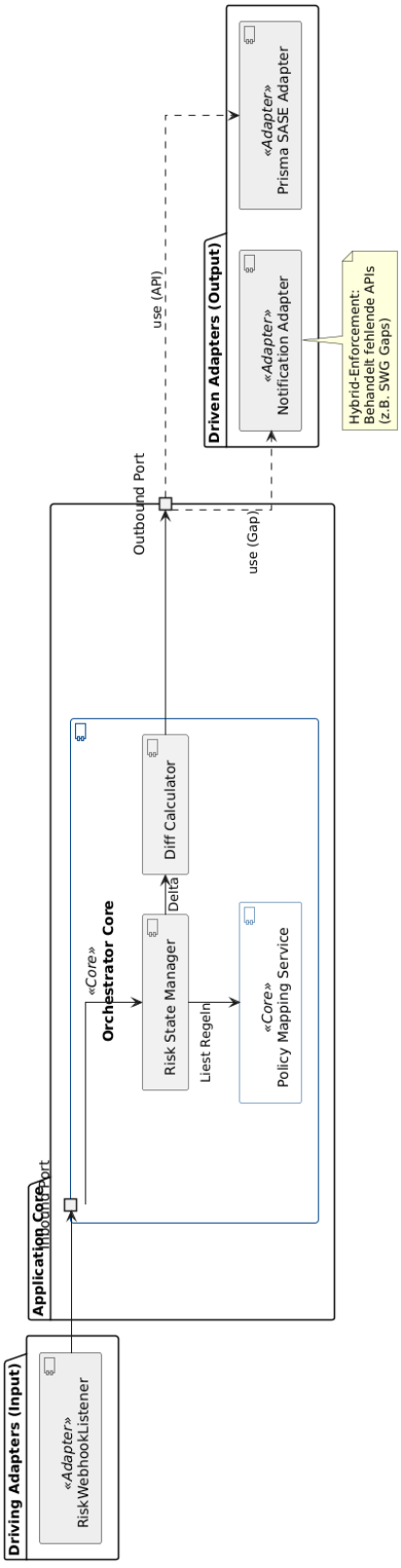

Abbildung 5.2: Logische Sicht: Hexagonale Architektur mit Trennung von Kernlogik und technischen Adaptern

Orchestrator Core (Application Layer) Diese Komponente bildet in der Soll-Architektur die Kontrollinstanz. Sie ist zuständig für:

- **State Management:** Ermittlung des aktuellen Systemzustands (Ist-Risikoklasse).
- **Diff-Calculation:** Berechnung der notwendigen Delta-Operationen zwischen Ist- und Soll-Zustand. Dies minimiert die Anzahl der API-Aufrufe, was angesichts der Performance -Anforderungen vorteilhaft ist.
- **Execution Strategy:** Entscheidung, ob Maßnahmen sequenziell oder parallel ausgeführt werden.

Policy Mapping Service (Domain Layer) Der Mapping Service kapselt das in Tabelle 4.5 entwickelte Regelwerk. Er hält die Definitionen, welche abstrakte Maßnahme (z. B. „Isolierung“) auf welches technische Steuerobjekt (z. B. `staticroutes`) abgebildet wird. Besonders relevant ist hierbei das **Gap-Handling**: Der Service markiert Maßnahmen, die im Metamodell als technisch nicht umsetzbar identifiziert wurden (vgl. SWG-Analyse), mit einem speziellen Flag (`EnforcementType.MANUAL`).

Prisma SASE Adapter (Southbound Adapter) In der Zielarchitektur bildet dieser Adapter die Schnittstelle zur proprietären SASE-Plattform von Palo Alto Networks. Seine Aufgabe besteht darin, die vom Kern erzeugten abstrakten, zielorientierten Anweisungen auf Policy-Ebene (z. B. „Isoliere Segment X“) in die spezifischen REST-Aufrufe der Ziel-API zu übersetzen.

Um die Komplexität der externen Kommunikation vom Kern fernzuhalten, kapselt der Adapter die folgenden technischen Aspekte:

- **Session- und Token-Management:** Wie in der Szenario-Analyse (vgl. Unterabschnitt 5.2.5) dargestellt, erfordert jeder API-Aufruf eine erfolgreiche Authentifizierung. Im Design ist der Adapter dafür vorgesehen, das initiale Abrufen des Access-Tokens, das sichere Caching

und die automatische Erneuerung (Refresh) zu übernehmen, um die Latenz durch unnötige Login-Calls zu minimieren.

- **Context-Injection:** Anreicherung der API-Calls mit notwendigen Metadaten, wie z. B. der korrekten Mandanten-ID im HTTP-Header.

- **Rate-Limiting und Retries:** Vorgesehene Implementierung von exponentiellem Backoff bei HTTP-429-Antworten, um die Stabilität der API-Anbindung zu gewährleisten.

Notification Adapter (Hybrid-Enforcement) Als Konsequenz aus der Widerlegung von Hypothese H1 (nicht alles ist automatisierbar) sieht das Design einen Notification Adapter vor. Erhält der Orchestrator eine Anweisung für eine Maßnahme, die als `MANUAL` geflaggt ist, wird statt eines API-Calls ein Ticket in einem externen IT Service Management (ITSM)-System generiert. Dies realisiert eine **hybride Sicherheitsarchitektur**, die Automatisierungslücken durch definierte Prozesse schließt.

Zusammenfassend etabliert die logische Sicht durch die Entkopplung von Kern und Adaptern die notwendige Struktur, um die identifizierten API-Lücken durch eine hybride Architektur zu schließen, ohne die Konsistenz des Gesamtsystems zu gefährden.

5.2.2 Prozesssicht (Process View)

Während die logische Sicht die statische Struktur der Komponenten definiert, beschreibt die Prozesssicht das dynamische Laufzeitverhalten des Systems. Der Fokus liegt auf der Steuerung von Nebenläufigkeit, der Synchronisation von Abläufen und insbesondere auf der Abbildung des in Kapitel 4.2.3 definierten Eskalationsmodells in einen deterministischen Algorithmus.

Zentrales Element dieser Sicht ist der *Orchestrierungs-Loop*. Abbildung 5.3 operationalisiert das Phasenmodell der Risikoadaption und zeigt explizit, wie technische Durchsetzung und organisatorischer Fallback im Algorithmus verankert sind. Sie bildet die Grundlage für die spätere Szenarioinstanz und die analytische Modellierung der Time-to-Policy.

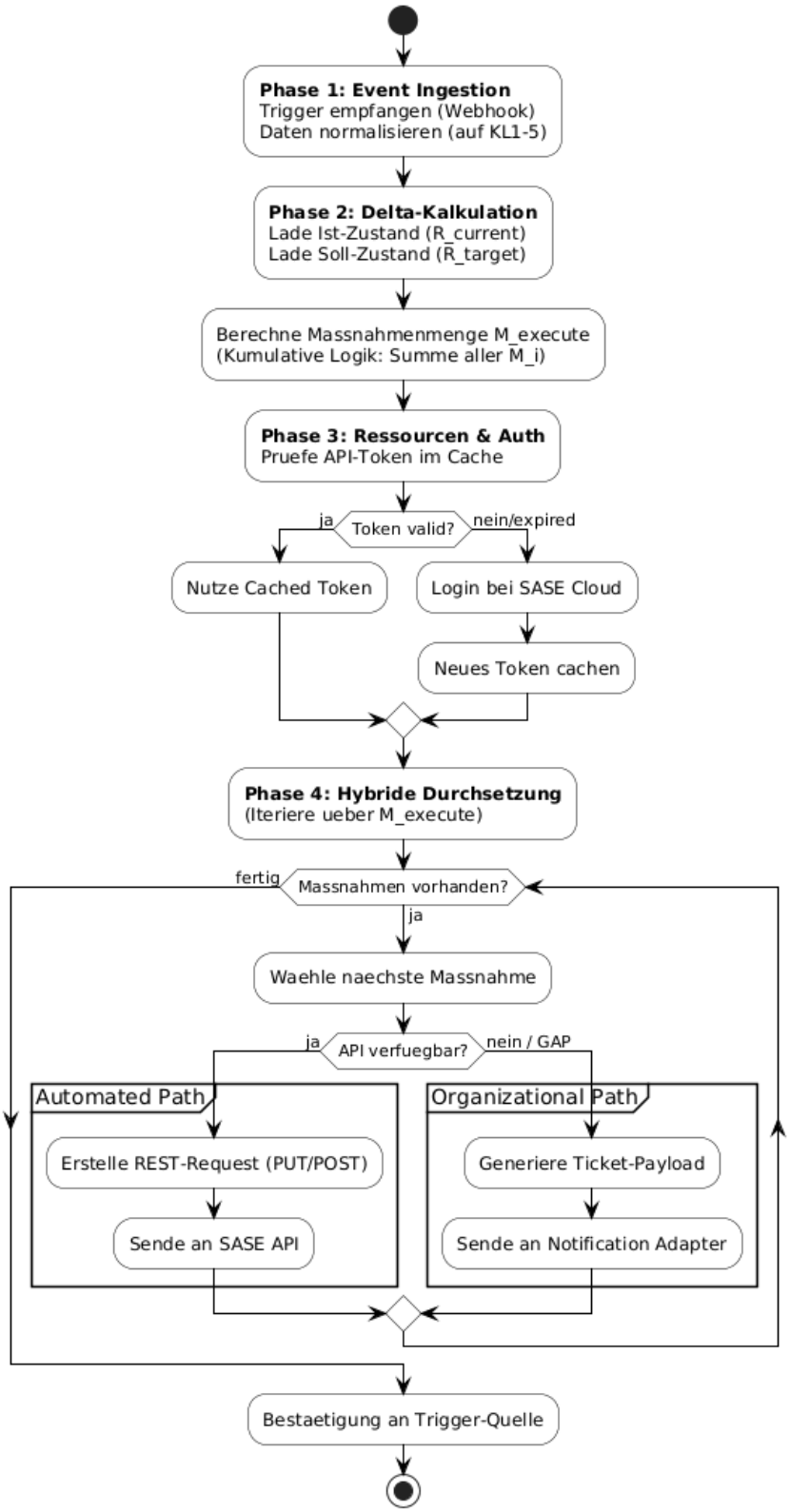

Abbildung 5.3: Der generische Orchestrierungs-Loop mit hybrider Entscheidungslogik

Dieser Prozess transformiert ein eingehendes Risiko-Ereignis (Trigger) unter Berücksichtigung der technischen Restriktionen (Authentifizierung, API-Gaps) in konkrete Systemänderungen und dient als Referenzmodell für die spätere Instanziierung im Szenario.

Der Orchestrierungs-Algorithmus

Der Ablauf einer Risikoadaption folgt einem streng sequenziellen Phasenmodell, um Race Conditions bei der Konfiguration der Cloud-Plattform zu vermeiden. Der Prozess lässt sich in vier Phasen unterteilen:

Phase 1: Event Ingestion und Normalisierung Der Prozess wird durch den Eingang eines JSON-Payloads am `WebhookController` initiiert. Da externe Quellen (z. B. SIEM, Admin-Konsole) unterschiedliche Formate liefern können, erfolgt zunächst eine Normalisierung auf das interne Datenmodell der Risikoklassen (KL 1 bis KL 5).

Phase 2: Delta-Kalkulation und Pfadermittlung Der Kern des Prozesses ist die Berechnung der notwendigen Zustandsänderungen. Der *Diff Calculator* lädt den aktuellen Systemzustand (R_{current}) und vergleicht ihn mit dem geforderten Zielzustand (R_{target}). Hierbei kommt die in Kapitel 4.2.2 hergeleitete **kumulative Sicherheitslogik** zum Tragen: Eine Eskalation darf nicht lediglich die Maßnahmen der Zielklasse aktivieren. Vielmehr muss der Algorithmus sicherstellen, dass auch alle dazwischenliegenden Sicherheitskontrollen wirksam werden. Die Menge der auszuführenden Maßnahmen M_{execute} berechnet sich formal als Vereinigung aller Maßnahmenmengen der überschrittenen Risikostufen:

$$M_{\text{execute}} = \bigcup_{i=R_{\text{current}}+1}^{R_{\text{target}}} M_i \tag{5.1}$$

Die kumulative Anwendung verhindert, dass Zwischenmaßnahmen – wie etwa die Aktivierung von IDS/IPS-Profilen in KL 3 bei einem direkten Sprung auf KL 4 – ausgelassen werden.

Phase 3: Ressourcen-Allokation und Authentifizierung Bevor schreibende Operationen durchgeführt werden, initiiert der *Prisma SASE Adapter* den Verbindungsaufbau. Der Prozess prüft die Gültigkeit des im lokalen Cache vorhandenen Tokens. Ist dieses abgelaufen oder nicht vorhanden, wird ein synchroner Login-Request gesendet. Erst nach erfolgreichem Erhalt des Tokens geht das System in die Durchsetzungsphase über. Dies vermeidet Latenzen durch fehlgeschlagene API-Calls im späteren Verlauf.

Phase 4: Hybride Durchsetzung (Enforcement) In dieser Phase werden die ermittelten Maßnahmen aus M_{execute} abgearbeitet. Aufgrund der in Kapitel 4.2.3 identifizierten API-Lücken unterscheidet der Prozess hier dynamisch zwischen zwei Pfaden:

1. **Automatisierter Pfad (API):** Für Maßnahmen, die mit einem konkreten Steuerobjekt verknüpft sind, generiert der Adapter den entsprechenden REST-Call (PUT/POST). Das gültige Auth-Token wird hierbei im Header injiziert.
2. **Organisatorischer Pfad (Fallback):** Stößt der Algorithmus auf eine Maßnahme, die als *GAP* klassifiziert ist, wird der Aufruf an den *Notification Adapter* umgeleitet. Dieser erzeugt statt eines API-Calls ein Ticket im ITSM-System.

Dieser hybride Ansatz stellt sicher, dass der Orchestrator auch in unvollständigen API-Umgebungen weiterarbeiten kann und den Prozessabschluss im Zweifel durch Delegation an organisatorische Maßnahmen erreicht.

Fehlerbehandlung und Transaktionssicherheit

Da die untersuchten SASE-APIs keine verteilten Transaktionen (Two-Phase-Commit) unterstützen, implementiert die Prozesssicht ein *Best-Effort*-Modell. Schlägt ein einzelner API-Aufruf fehl, wird dieser spezifische Schritt protokolliert. Der Gesamtprozess wird jedoch nicht abgebrochen, um zumindest den Teilschutz der erfolgreich umgesetzten Maßnahmen zu erhalten. Kritische Fehler lösen eine sofortige Alarmierung über den Notification Adapter aus, um manuelles Eingreifen zu ermöglichen.

Damit liefert die Prozesssicht den deterministischen Ablaufrahmen, der für die in Hypothese H2 geforderte Time-to-Policy essenziell ist, indem sie zeitkritische Pfade (Caching) von blockierenden Ausnahmen (Gaps) trennt.

5.2.3 Entwicklungssicht (Development View – Design-Blueprint)

Die Entwicklungssicht definiert auf konzeptioneller Ebene die Vorgaben für die softwaretechnische Realisierung des Artefakts. Um die in Kapitel 3 geforderten Designprinzipien der Wartbarkeit und Erweiterbarkeit zu erfüllen, legt dieser Entwurf verbindliche Standards und eine modulare Paketstruktur fest. Diese Spezifikation beschreibt eine mögliche Soll-Architektur und dient als Grundlage für eine spätere Implementierung sowie für die theoretische Evaluation der Systemperformance.

Technologie-Agnostik und Anforderungen

Die Soll-Architektur ist technologieunabhängig konzipiert. Eine spätere Implementierung kann jede Programmiersprache nutzen, die folgende Anforderungen erfüllt:

- **Asynchrone I/O-Verarbeitung:** Um die geforderten Antwortzeiten (vgl. Hypothese H2) theoretisch erreichen zu können, muss die Laufzeitumgebung effiziente Muster für parallele oder nicht-blockierende Netzwerkkommunikation unterstützen.

- **Strikte Datenvalidierung:** Da der Orchestrator sicherheitskritische Entscheidungen auf Basis externer Payloads trifft, ist eine robuste Validierung strukturierter Daten (z. B. Schema-Validierung für JSON) gegen das definierte Datenmodell erforderlich.

Vorgesehene Paketstruktur

Um die logische Trennung der hexagonalen Architektur (vgl. Unterabschnitt 5.2.1) auch auf Code-Ebene zu erzwingen, definiert der Entwurf die folgende Paketstruktur als Referenzim-

plementierung. Diese Struktur verhindert zirkuläre Abhängigkeiten, indem der fachliche Kern isoliert wird:

- `src.domain`**:** Soll ausschließlich die Datenmodelle (*Entities*) und das Metamodell enthalten (keine externen Abhängigkeiten).

- `src.application`**:** Beheimatet die Steuerungslogik, insbesondere den Algorithmus zur Delta-Berechnung gemäß Gleichung 5.1.

- `src.adapters`**:** Enthält die Implementierungen der Schnittstellen zu externen Systemen.

Abbildung 5.4 konkretisiert diese logische Trennung in Form einer möglichen Paketstruktur und illustriert, wie Domänenlogik, Orchestrierungslogik und Adapter in einer Referenzimplementierung voneinander entkoppelt werden könnten. Als Entwicklungssprache wird hier exemplarisch Python verwendet.

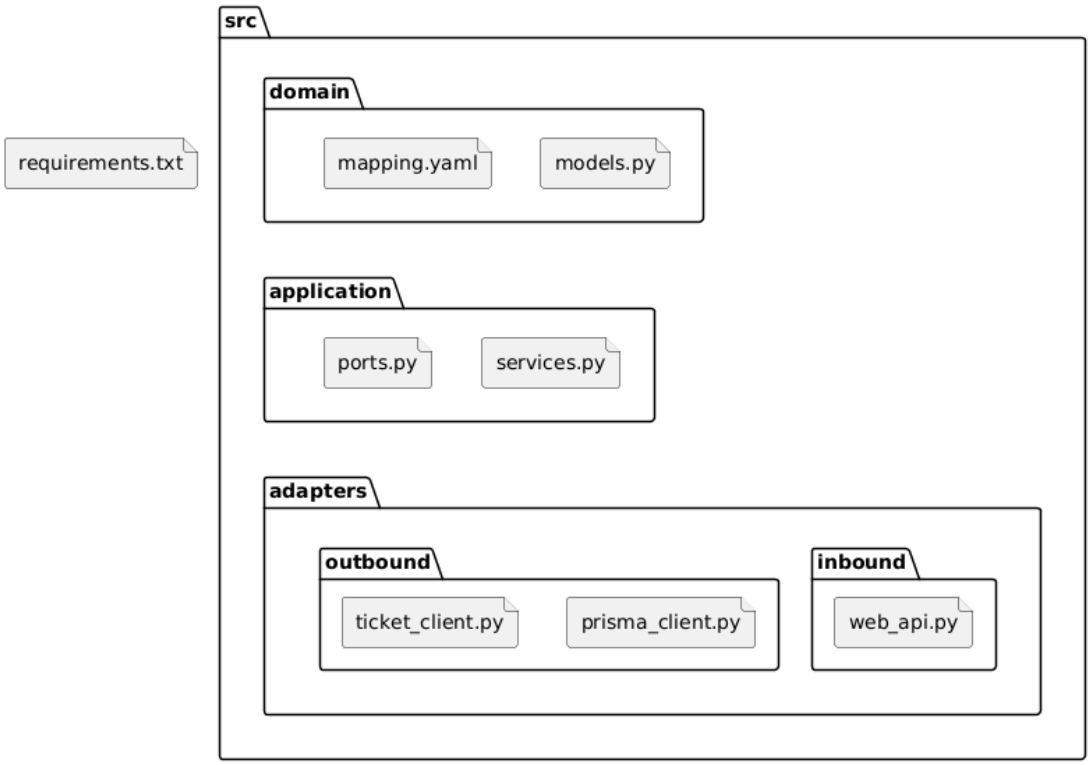

Abbildung 5.4: Design-Blueprint der Paketstruktur gemäß Hexagonal Architecture

Diese Sicht belegt, dass das entworfene Artefakt modular erweiterbar ist und Änderungen an den Adaptern (z. B. bei API-Updates) ohne Auswirkungen auf die Kernlogik vorgenommen werden können, was die Wartbarkeit im Sinne des DSR-Prozesses unterstützt.

5.2.4 Physische Sicht (Physical View)

Die physische Sicht beschreibt die Abbildung der Softwarekomponenten auf die Hardware- und Netzwerkinfrastruktur. Da die Leistungsfähigkeit des Orchestrators maßgeblich von den Latenzzeiten zu den externen Schnittstellen abhängt, definiert dieser Abschnitt die notwendigen Anforderungen an die Deployment-Umgebung.

Containerisierte Laufzeitumgebung

Das Systemdesign schlägt eine Bereitstellung als Container-Service vor. Dies entkoppelt das Artefakt von der zugrundeliegenden Server-Hardware und gewährleistet, dass die spezifischen Abhängigkeiten gekapselt bereitgestellt werden. Aus architektonischer Sicht wird für eine spätere Umsetzung eine *Single-Instance*-Bereitstellung präferiert, um State-Konflikte im *Diff Calculator* zu vermeiden, da das Systemdesign aktuell keine verteilte Zustandsdatenbank vorsieht.

Netzwerktopologie und Latenzanforderungen

Die Positionierung des Orchestrators im Netzwerk ist ein zentraler Faktor für die „Time-to-Policy“. Abbildung 5.5 zeigt die Zuordnung der Artefaktkomponenten zu einer exemplarischen Infrastruktur. Die Darstellung verdeutlicht insbesondere die Latenzpfade zu den externen Cloud-APIs und schafft damit die Grundlage für die spätere Abschätzung der Time-to-Policy in der Evaluation.

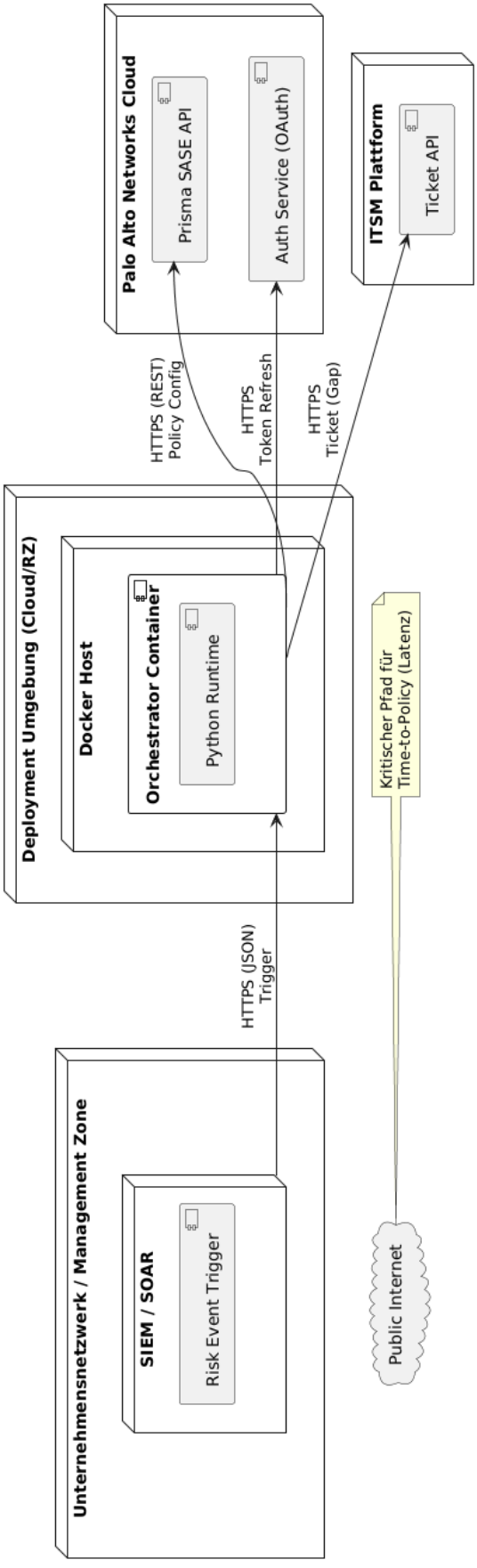

Abbildung 5.5: Physische Sicht: Deployment-Topologie und Kommunikationspfade über öffentliche Netze

Das Design stellt die folgenden Anforderungen an den Betriebsort:

1. **Proximity to Control Plane:** Der Orchestrator muss über einen leistungsfähigen Internet-Breakout verfügen, da die Kommunikation zur Palo Alto Cloud API den Hauptteil der Verarbeitungszeit ausmacht. Eine Platzierung in einer Public Cloud Region mit direktem Peering zum SASE-Anbieter ist vorteilhaft.
2. **SIEM-Konnektivität:** Der `WebhookController` muss für die ereignisgebenden Systeme erreichbar sein. Dies erfordert eine Ingress-Konfiguration, die eingehende HTTPS-Trigger aus dem Management-Netz zulässt.

Die Trennung in eine physische Sicht verdeutlicht, dass der Orchestrator selbst keine hohen Rechenanforderungen (Central Processing Unit (CPU)/Random Access Memory (RAM)) stellt, sondern primär Input/Output (I/O)-gebunden arbeitet. Die Performance wird nicht durch die Rechenleistung, sondern durch die Netzwerklatenz (Round Trip Time (RTT)) dominiert. Damit schafft die physische Sicht die notwendige Transparenz über die infrastrukturellen Rahmenbedingungen, die zur Validierung von Hypothese H2 erforderlich sind.

5.2.5 Szenarien (Scenarios)

Die Szenario-Sicht validiert die Architektur, indem sie das Zusammenspiel der definierten Komponenten anhand eines konkreten Anwendungsfalls demonstriert. Sie dient als konzeptioneller Proof-of-Concept und zeigt, dass das Systemdesign in der Lage ist, die funktionale Komplexität der Risikosteuerung sowie die technischen Limitationen (GAPs) koordiniert zu handhaben.

Als Referenzszenario wird die Eskalation von Risikoklasse 2 (Gering) auf Risikoklasse 4 (Hoch) gewählt. Dieses Szenario wurde selektiert, da es die höchsten Anforderungen an die Orchestrierungslogik stellt: Es erfordert die korrekte Anwendung der kumulativen Maßnahmenlogik, die Interaktion mit der API unter Authentifizierungszwang sowie den Umgang mit nicht automatisierbaren Maßnahmen (Hybrid-Enforcement).

Ablaufbeschreibung des Referenzszenarios

Das Sequenzdiagramm in Abbildung 5.6 instanziiert den zuvor definierten Orchestrierungs-Loop für das Referenzszenario KL 2→KL 4 und zeigt, wie kumulative Maßnahmenlogik, Authentifizierung und hybrid organisiertes Gap-Handling konkret zusammenwirken.

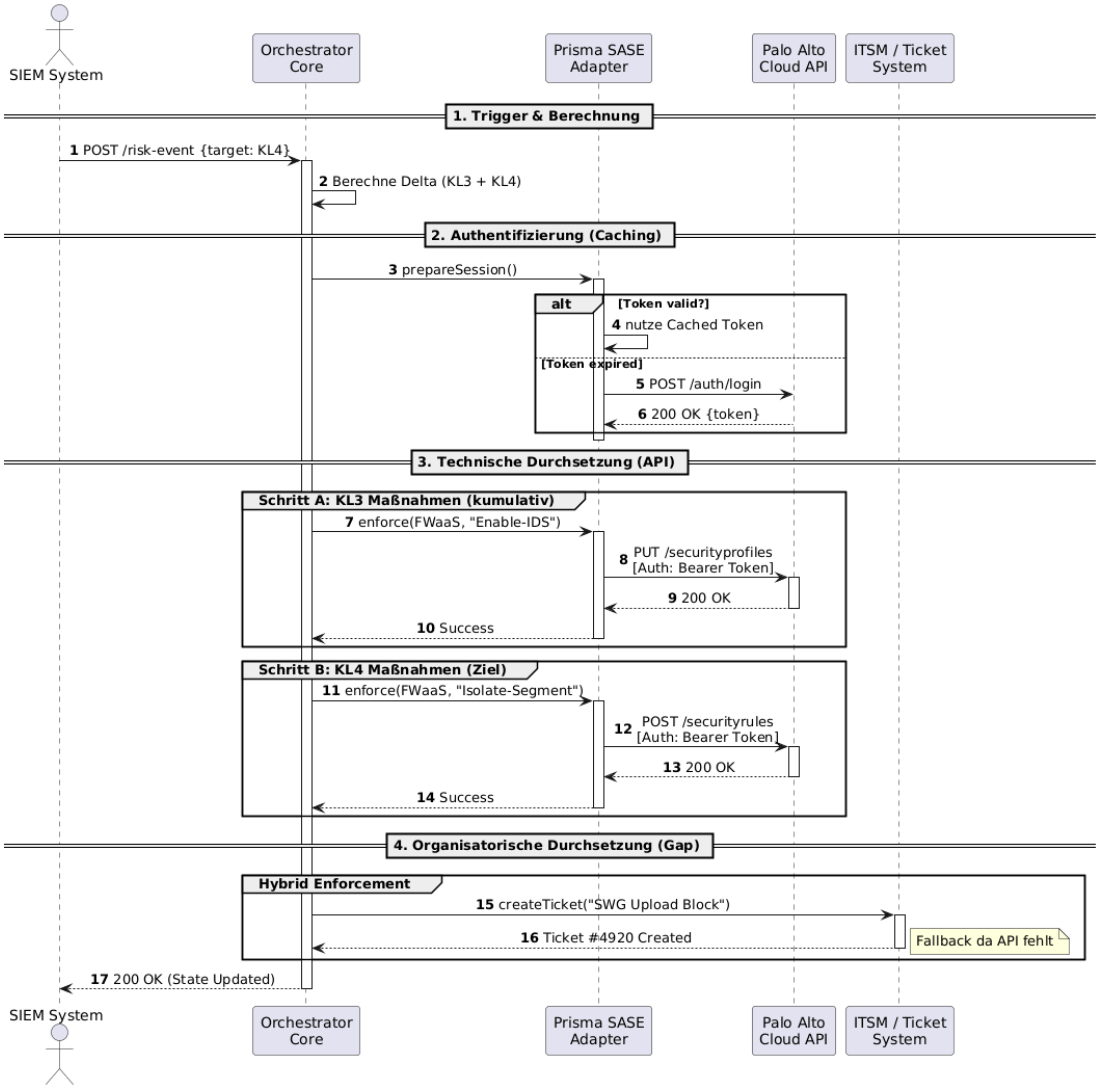

Abbildung 5.6: Sequenzdiagramm: Ablauf der kumulativen Eskalation mit Authentifizierung und Gap-Handling

Der Prozess gliedert sich in vier logische Abschnitte:

1. **Initialisierung und Delta-Berechnung:** Auslöser ist ein REST-Aufruf eines externen SIEM-Systems, der eine Erhöhung des Risikoniveaus auf *KL 4* fordert. Der *Orchestrator Core* lädt den Ist-Zustand und ermittelt das Delta. Gemäß der in Unterabschnitt 5.2.2 definierten kumulativen Logik identifiziert das System, dass nicht nur Maßnahmen der Zielklasse 4, sondern auch noch nicht aktive Maßnahmen der übersprungenen Klasse 3 (z. B. IDS-Aktivierung) umgesetzt werden müssen.
2. **Session-Management (Authentifizierung):** Bevor technische Änderungen erfolgen, prüft der *Prisma SASE Adapter* den Sicherheitskontext. Da im Szenario kein gültiges Token vorliegt, führt der Adapter zunächst einen Login durch. Das empfangene *Bearer Token* wird für alle nachfolgenden Aufrufe zwischengespeichert.
3. **Technische Durchsetzung (API-Pfad):** Der Adapter iteriert über die automatisierbaren Maßnahmen.
   - Zuerst wird die Maßnahme aus **KL 3** umgesetzt: Ein `PUT`-Request aktiviert das IDS-Profil.
   - Anschließend folgt die Maßnahme aus **KL 4**: Ein `POST`-Request erstellt Block-Regeln für den Ost-West-Verkehr (Isolierung).

   Beide Aufrufe erfolgen unter Verwendung des zuvor erlangten Authentifizierungs-Tokens.
4. **Organisatorische Durchsetzung (Gap Handling):** Das Delta beinhaltet auch die Anforderung „SWG: Block Uploads“ (Teil von KL 4). Da die Analyse in Kapitel 4.1 ergab, dass hierfür kein geeigneter API-Endpunkt existiert, wechselt der Orchestrator auf den Fallback-Pfad. Der *Notification Client* erzeugt ein Ticket im ITSM-System.

Bewertung des Szenarios

Das Szenario verdeutlicht die Funktionsfähigkeit des Designs im Rahmen der getroffenen Modellannahmen. Es zeigt, dass der Orchestrator trotz der unvollständigen API-Abdeckung der SASE-Plattform einen konsistenten Sicherheitszustand herstellen kann. Die strikte Trennung von Logik und Technik verhindert, dass technische Hürden den Sicherheitsprozess blockieren. Das Szenario dient somit als konzeptioneller Nachweis auf Architekturebene, dass die Kernanforderungen funktional erfüllt werden können.

5.3 Architektonische Evaluation

Abschließend erfolgt die Evaluation des Systemdesigns gegen die in der Einleitung definierte Performance-Hypothese **H2**. Da im Rahmen dieser Arbeit keine vollumfängliche Implementierung und Feldmessung erfolgt (vgl. Abgrenzung in Kapitel 1.5), wird die Hypothese mittels eines analytischen Performance-Modells bewertet.

5.3.1 Theoretische Fundierung der analytischen Modellierung

Die Bewertung von Systemarchitekturen durch analytische Modelle ist ein gängiges Vorgehen im Rahmen von Design Science Research, insbesondere wenn eine empirische Messung im Produktivbetrieb noch nicht möglich ist. Der hier gewählte Ansatz zerlegt die Gesamtreaktionszeit („Time-to-Policy“) in ihre konstituierenden Latenzkomponenten. Dieses Vorgehen folgt etablierten Frameworks zur Analyse vernetzter Systeme, bei denen Übertragungs- und Verarbeitungszeiten sowie Protokoll-Scheduling-Effekte als Parameter für die Systemdynamik modelliert werden[4].

Obwohl die genannten Arbeiten primär IoT-Protokolle oder Regelkreise adressieren, ist die methodische Zerlegung der Latenz in Verbindungsaufbau-, Übertragungs- und Verarbeitungskomponenten auf die hier betrachtete SASE-Orchestrierung übertragbar. Auch in vergleichbaren Untersuchungen zu Kommunikationsprotokollen hat sich die additive Betrachtung von Overhead und Nutzdatenübertragung als valides Mittel zur Performance-Abschätzung bewährt[5]. Das im Folgenden aufgestellte Modell adaptiert diese Prinzipien, indem es Netzwerk-Round-Trip-Times und API-Verarbeitungszeiten als diskrete Summanden betrachtet.

[4] vgl. Zou u. a., „Communication-protocol-based analysis and synthesis of networked systems: progress, prospects and challenges“, S. 3015.

[5] vgl. Bayılmış u. a., „A survey on communication protocols and performance evaluations for Internet of Things“, S. 1102.

5.3.2 Performance-Modellierung (Time-to-Policy)

Die zentrale Metrik ist die *Time-to-Policy* (T_{total}). Sie definiert die Zeitspanne zwischen dem Erkennen einer Risikoänderung im Quellsystem und der wirksamen Aktivierung der Gegenmaßnahme in der SASE-Plattform. Das Performance-Modell definiert diese Gesamtzeit wie folgt:

$$T_{\text{total}} = T_{\text{Trigger}} + T_{\text{Compute}} + T_{\text{Auth}} + \sum_{i=1}^{n} \left(T_{\text{Net},i} + T_{\text{API\_Process},i} \right) \tag{5.2}$$

Abbildung 5.7 visualisiert die in Gleichung (5.2) definierte Zerlegung der Time-to-Policy. Die Darstellung zeigt die Abhängigkeiten der einzelnen Latenzkomponenten entlang der Verarbeitungsstrecke und erleichtert damit die Überleitung von der formalen Modellierung zur konkreten Szenario-Berechnung.

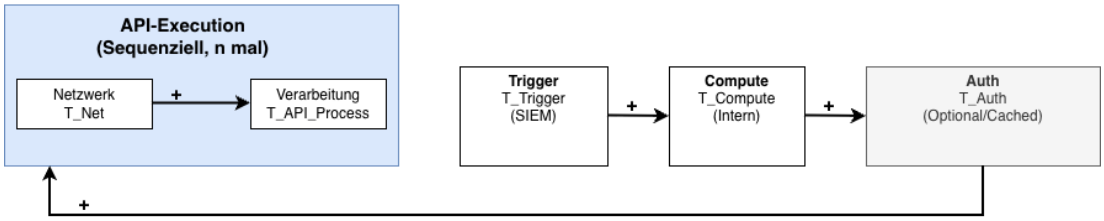

Abbildung 5.7: Visualisierung der Latenzkomponenten des analytischen Performance-Modells

Die Parameter sind wie folgt definiert: Um eine belastbare *Worst-Case-Betrachtung* zu ermöglichen, werden die Parameter als konservative Obergrenzen (Designannahmen) festgelegt. Öffentlich dokumentierte Latenzkennzahlen für die untersuchte SASE-API liegen nicht in belastbarer Form vor; daher dient das Modell einer Worst-Case-Abschätzung und Sensitivitätsbetrachtung. Die Zerlegung in Netzwerklatenz und serverseitige Verarbeitungszeit folgt dem Request/Response-Charakter REST-basierter Schnittstellen.

Für die Szenario-Rechnung werden konstante Obergrenzen angenommen, d. h. $T_{\text{Net},i} = T_{\text{Net}}$ und $T_{\text{API\_Process},i} = T_{\text{API\_Process}}$ für alle $i \in \{1, \ldots, n\}$.

Damit vereinfacht sich die Summe zu $T_{\text{total}} = T_{\text{Trigger}} + T_{\text{Compute}} + T_{\text{Auth}} + n \cdot (T_{\text{Net}} + T_{\text{API\_Process}})$.

- T_{Trigger}: Latenz vom SIEM-Event bis zum Empfang am Webhook. Dieser Wert wird als konservative Obergrenze für die Event-Weiterleitung in einer Enterprise-Pipeline angenommen. *Annahme:* $\leq 1,0\,s$.

- T_{Compute}: Interne Verarbeitungszeit des Orchestrators. Aufgrund der geringen Komplexität vernachlässigbar. *Annahme:* $\leq 0,1\,s$.

- T_{Auth}: Zeit für den initialen Login-Handshake (OAuth) gegen die Palo Alto Cloud. Dieser Wert fällt dank Token-Caching (vgl. Unterabschnitt 5.2.2) nur beim ersten Aufruf an. *Annahme (Worst-Case):* $1,0\,s$.

- n: Anzahl der sequenziell durchzuführenden Schreiboperationen (POST/PUT) gegen Ziel-APIs im Szenario KL2→KL4.
  Wert: 6 (2x FWaaS Policy-Änderungen: IDS/IPS-Profil-Update und Zonen-Isolierungsregel, 1x ZTNA Score-/Kontext-Update, 1x SD-WAN Segment-Update, 1x DNS-Profil-Update, 1x Ticket-Erstellung für nicht automatisierbare Maßnahmen).

- $T_{\text{Net},i}$: Netzwerk-RTT zur jeweils adressierten Ziel-API der i-ten Operation.
  Annahme: $\leq 0,1\,s$ (100 ms).

- $T_{\text{API\_Process},i}$: Serverseitige Verarbeitungszeit der jeweils angesprochenen Zielplattform für die i-te Schreiboperation. Der gewählte Wert ist als konservative Obergrenze für transaktionale Cloud-APIs zu verstehen. *Annahme:* $\leq 2,0\,s$.

5.3.3 Berechnung am Referenzszenario

Wendet man diese Parameter auf das in Unterabschnitt 5.2.5 beschriebene Szenario der Eskalation von KL 2 auf KL 4 an, ergibt sich folgende Kalkulation:

$$T_{\text{total}} \leq 1{,}0\,\text{s} + 0{,}1\,\text{s} + 1{,}0\,\text{s} + 6 \cdot (0{,}1\,\text{s} + 2{,}0\,\text{s})$$

$$T_{\text{total}} \leq 2{,}1\,\text{s} + 6 \cdot (2{,}1\,\text{s})$$

$$T_{\text{total}} \leq \mathbf{14{,}7}\,\text{s}$$

5.3.4 Diskussion der Ergebnisse und Validierung von H2

Das rechnerische Ergebnis von **14,7 Sekunden** liegt signifikant unter dem in Hypothese H2 geforderten Grenzwert von 30 Sekunden. Selbst bei einer Verdopplung der angenommenen API-Latenzen bliebe der Wert mit ca. 27,3 Sekunden innerhalb des Zielkorridors.

Bewertung der Hypothese H2:

> *H2: Die Zeitspanne zwischen Risikoerhöhung und Policy-Aktivierung liegt unter 30 s.*

Auf Basis der Architekturanalyse und der konservativen Modellrechnung kann diese Hypothese als **gestützt** angesehen werden. Das Modell zeigt, dass die strukturellen Latenzen der gewählten Architektur – insbesondere die begrenzte Anzahl externer API-Aufrufe und der Einsatz von Token-Caching – die in H2 definierten Zeitanforderungen nicht verletzen.

Einfluss der Architekturentscheidungen auf die Performance

Die positive Bewertung der Performance ist das direkte Resultat spezifischer Designentscheidungen in Kapitel 5:

- **Token-Caching:** Durch das Caching des OAuth-Tokens entfällt T_{Auth} für die Mehrheit der Operationen, was die Latenz im Durchschnitt weiter senkt.

- **Single-Instance-Deployment:** Der Verzicht auf komplexe verteilte Konsistenzmechanismen eliminiert Koordinations-Overhead.

- **Verwendung von Objektgruppen:** Das Design adressiert *Policy Sets* und *Address Groups* statt einzelner User-Objekte. Dadurch bleibt der Faktor n konstant klein ($O(1)$), unabhängig von der Anzahl der geschützten Endpunkte.

Grenzen des Modells und der Skalierbarkeit

Es ist anzumerken, dass das analytische Modell eine Idealisierung darstellt. Es abstrahiert von dynamischen Effekten wie serverseitigem Rate-Limiting oder internem Queuing in der Cloud-Plattform bei Lastspitzen. Diese Faktoren könnten in Extremfällen zu längeren Laufzeiten führen und wären im Rahmen einer zukünftigen empirischen Evaluation gesondert zu betrachten.

Zudem beruht die Skalierbarkeit maßgeblich auf der bewussten Designentscheidung für globale Steuerobjekte. Würde das Design hingegen individuelle Regeln pro Benutzer ändern (Komplexitätsklasse $O(u)$), würde die Laufzeit linear ansteigen. Das entwickelte Systemdesign vermeidet dieses Risiko explizit durch den Einsatz aggregierter Policy-Objekte.

6 Reflexion und Fazit

Ziel der vorliegenden Arbeit war die konzeptionelle Entwicklung eines risikobasierten Orchestrators für SASE-
Architekturen. Der Untersuchungsrahmen umfasste die Definition von Risikoklassen auf Grundlage des BSI-Standards 200-3, die Analyse der technischen Steuerungsobjekte moderner SASE-Plattformen und die Entwicklung eines Metamodells, das beide Ebenen systematisch verbindet. Auf dieser Grundlage entstand eine Soll-Architektur nach dem 4+1-Modell, welche die technische und organisatorische Umsetzung risikobasierter Maßnahmen beschreibt.

Metamodell, Systemarchitektur und Performance-Modell adressieren die zentralen Aspekte der Zielsetzung. Das entwickelte Metamodell ordnet den abstrakten Risikoklassen (KL 1–5) generische Policy-Kategorien zu. Die daraus abgeleitete Systemarchitektur beschreibt den Ablauf von der Risikoadaption bis zur technischen Durchsetzung. Das analytische Performance-Modell zeigt, dass die für Hypothese H2 definierte Time-to-Policy unterschritten bleibt.

Die Untersuchung zeigt deutliche Einschränkungen der derzeit verfügbaren SASE-APIs.
Die Analyse der Prisma-SASE-Plattform belegt, dass ein Teil der geforderten Maßnahmen keine Entsprechung in steuerbaren Objekten der dokumentierten API erhält (siehe Unterunterabschnitt 4.2.3, Absatz 4.1.3, Unterunterabschnitt 4.1.3). Das hybride Enforcement-Modell ergänzt deshalb die technische Durchsetzung durch organisatorische Schritte. Die Analyse zeigt damit, dass ein reiner Automatisierungsansatz im aktuellen Reifegrad der Plattformen keine vollständige Umsetzung ermöglicht. Ein tragfähiges Risikomodell benötigt neben technischen Steuerungswegen auch organisatorische Steuerungswege, etwa die Ticket-Erstellung in einem ITSM-System.

Die Ergebnisse verdeutlichen, dass das Mapping im Metamodell von der Definition der erwarteten Maßnahmen abhängt. Für das Mapping werden generische und risikoorientierte Maßnahmensets verwendet, die unabhängig von spezifischen Anbieterplattformen formuliert sind. Eine alternative Maßnahmendefinition mit engerem Bezug zu proprietären Funktionsumfängen führt zu einer anderen Bewertung der Automatisierungsfähigkeit. Das Mapping besitzt daher keinen absoluten Charakter, sondern hängt von der Risikodefinition ab.

Die Analyse der GAPs zeigt, dass diese Lücken aus strukturellen Eigenschaften der untersuchten API entstehen (siehe Unterunterabschnitt 4.2.3). GAPs treten dort auf, wo das Metamodell Maßnahmen vorsieht, die in der öffentlich dokumentierten API keine Abbildung finden. Die Analyse betrifft vor allem Funktionen, die die Plattform zwar besitzt, jedoch ohne zugehörige Steuerobjekte in der API bereitstellt. Die GAPs zeigen damit keine Funktionsdefizite der Plattform, sondern die Abhängigkeit der Automatisierung von der API-Exposition der jeweiligen Hersteller. Das Metamodell beschreibt dadurch die Grenzen eines vollständig automatisierten Orchestrators in heterogen-hybriden Umgebungen.

Die zentrale Forschungsfrage lautete:

> *Können angemessene, zur Laufzeit dynamisch erzeugbare Schutzmaßnahmen in heterogen-hybriden Umgebungen über SASE so abgebildet werden, dass sie den vordefinierten Risikoklassen (1–5) entsprechen?*

Auf Basis der Ergebnisse zeigte sich, dass sich risikobasierte Schutzmaßnahmen konzeptionell modellieren lassen. Eine vollständige technische Abbildung der Risikoklassen ausschließlich über steuerbare SASE-Policy-Objekte ist mit den untersuchten Plattformstrukturen aufgrund identifizierter API-Lücken nicht erreichbar. Für eine durchgängige Umsetzung sind daher ergänzende organisatorische Maßnahmen erforderlich.

Die Bewertung der drei Hypothesen ergänzt dieses Ergebnis:

- **H1:** Die für die fünf Risikoklassen erforderlichen Schutzmaßnahmen lassen sich auf generische SASE-Policy-Objekte abbilden.

- **H2:** Die Zeitspanne zwischen Risikoerhöhung und Policy-Aktivierung liegt unter 30 s.

- **H3:** Ein modularer Orchestrator mit abstrahierendem Mapping-Layer kann API-Heterogenität kompensieren und konsistente Policies erzeugen, selbst wenn einzelne Hersteller keine vollständigen Steuerobjekte bereitstellen.

H1 erhält nur teilweise Unterstützung, weil technische Lücken die vollständige Automatisierung verhindern. H2 erhält Bestätigung, da die modellierte Reaktionszeit unter dem Grenzwert von 30 s liegt. H3 erhält Unterstützung durch die modulare Architektur, welche den Umgang mit heterogenen Schnittstellen und fehlenden Steuerobjekten ermöglicht.

Die drei Teilfragen lauteten:

- **TF1:** Wie muss ein Metamodell konzipiert sein, um die Risikoklassen auf generische SASE-Policy-Objekte abzubilden?

- **TF2:** Wie schnell kann ein risikobasierter Orchestrator einen Wechsel der Risikoklasse (z. B. KL 2 $\rightarrow$ KL 4) technisch wirksam abbilden?

- **TF3:** Wie muss die Architektur eines risikobasierten SASE-Orchestrators gestaltet sein, um API-Heterogenität und Funktionslücken kompensieren zu können?

TF1 wird durch das entwickelte Mapping beantwortet. Die Abbildbarkeit der Risikoklassen auf generische Policy-Kategorien bleibt grundsätzlich möglich, jedoch durch nicht automatisierbare Maßnahmen begrenzt. TF2 wird durch die analytische Modellierung beantwortet. Die berechnete Time-to-Policy von 14,7 s liegt unter dem Grenzwert von 30 s (siehe Abschnitt 5.3). TF3 wird durch die modulare Architektur beantwortet. Die strukturierte Trennung von Domänenlogik, Orchestrierung und Adaptern ermöglicht einen kontrollierten Umgang mit API-Heterogenität und fehlenden Steuerobjekten.

Offen bleibt die empirische Überprüfung der Laufzeitannahmen im praktischen Einsatz.
Die Bewertung stützt sich auf analytische Modelle, da produktive Messdaten nicht vorliegen. Auch die Übertragbarkeit des Metamodells auf weitere Anbieter bleibt offen, da die Analyse ausschließlich die Prisma-SASE-Plattform betrachtet. Die Sicherheit des Orchestrators bleibt ebenfalls offen, da der Schutz vor manipulierten Triggern, unberechtigten API-Zugriffen und Missbrauch der Automatisierungslogik nicht vertieft wird.

Die Entwicklung eines Prototyps würde die empirische Überprüfung der Laufzeitannahmen ermöglichen und potenzielle Grenzen im praktischen Betrieb sichtbar machen. Dies war im Zuge der Arbeit nicht möglich, da keine geeignete SASE-API-Testumgebung zur Verfügung stand. Eine Übertragung des Metamodells auf weitere SASE-Plattformen kann klären, ob sich ein plattformübergreifendes Meta-API-Modell ableiten lässt. Eine weiterführende Analyse der Sicherheit des Orchestrators kann den Schutz vor Fehlsteuerungen und Manipulationen untersuchen. Die Integration alternativer Risikomodelle kann die Abhängigkeit vom BSI-Standard 200-3 reduzieren.

Abschließend zeigt sich, dass die Verbindung von Risikomanagement und technischer Policy-Orchestrierung ein relevantes und bisher wenig untersuchtes Thema bildet. Das entwickelte Metamodell und die daraus abgeleitete Systemarchitektur schaffen einen strukturierten Lösungsrahmen, der als Grundlage für weiterführende Forschung und zukünftige Implementierungen dient.

Tabellenverzeichnis

Tabelle 1.1 Teilfragen und Hypothesen 3

Tabelle 2.1 Kernkomponenten eines SASE-Stacks 8

Tabelle 4.1 Ressourcengruppen und Methoden je Komponente (Prisma SASE) 27
Tabelle 4.2 Vergleich der Komponentenabdeckung in Prisma SASE und OpenSASE . . 45
Tabelle 4.3 Risikoklassen und zugeordnete Ziele im Framework 49
Tabelle 4.4 Risikoklassen und generische Policy-Erwartungen 57
Tabelle 4.5 Detailliertes Mapping-Metamodell der Risikoklassen auf SASE-Steuerobjekte 61

Abbildungsverzeichnis

Abbildung 2.1 SASE-Architektur mit integrierten Sicherheits- und Netzwerkdiensten . . 9

Abbildung 4.1 Gartner Magic Quadrant for SASE Platforms 2025
Quelle: Forest, MacDonald und Koeppen, *2025 Gartner Magic Quadrant for SASE Platforms* 23
Abbildung 4.2 Verteilung der HTTP-Methoden je Komponente (CRUD) 27
Abbildung 4.3 Ressourcengruppen je Komponente 28
Abbildung 4.4 OpenSASE: Stellmöglichkeiten je Komponente 42
Abbildung 4.5 OpenSASE: Artefakte nach Typ 43
Abbildung 4.6 Matrix der Risikoklassen in Anlehnung an BSI-Standard 200-3 48

Abbildung 5.1 Strukturierung des Artefakts nach dem 4+1 Sichtenmodell 66
Abbildung 5.2 Logische Architektur des Risk-Aware SASE Orchestrators 69
Abbildung 5.3 Ablaufdiagramm des Orchestrierungs-Loops 72
Abbildung 5.4 Design-Blueprint der Paketstruktur gemäß Hexagonal Architecture . . . 76
Abbildung 5.5 Physisches Verteilungsdiagramm 78
Abbildung 5.6 Sequenzdiagramm der Risikoeskalation 80
Abbildung 5.7 Visualisierung der Latenzkomponenten des analytischen Performance-Modells 83

Abkürzungsverzeichnis

ADEM Autonomous Digital Experience Management
API Application Programming Interface
ASN Autonomous System Number
AV Anti-Virus
BSI Bundesamt für Sicherheit in der Informationstechnik
C2 Command and Control
CA Certificate Authority
CASB Cloud Access Security Broker
CIS Center for Internet Security
CPU Central Processing Unit
CRUD Create, Read, Update, Delete
CSV Comma-Separated Values
DNS Domain Name System
DLP Data Loss Prevention
DoH DNS over HTTPS
DoT DNS over TLS
DPI Deep Packet Inspection
DSR Design Science Research
DSGVO Datenschutz-Grundverordnung
ENISA European Union Agency for Cybersecurity
FOSS Free and Open Source Software
FWaaS Firewall as a Service
GTI Global TD-LTE Initiative
HTTP Hypertext Transfer Protocol
HTTPS Hypertext Transfer Protocol Secure
I/O Input/Output
IaaS Infrastructure as a Service
ICAP Internet Content Adaptation Protocol
IDS Intrusion Detection System
IEC International Electrotechnical Commission
IOC Indicator of Compromise
IoT Internet of Things
IPS Intrusion Prevention System
ISO International Organization for Standardization
ITSM IT Service Management
JSON JavaScript Object Notation
MEF Metro Ethernet Forum
MFA Multi-Faktor-Authentisierung
MPLS Multiprotocol Label Switching
MSP Managed Service Provider
NGFW Next Generation Firewall
NIST National Institute of Standards and Technology
NTP Network Time Protocol

OS Operating System
PaaS Platform as a Service
PCI Payment Card Industry
PDF Portable Document Format
PII Personally Identifiable Information
PoC Proof of Concept
PoP Point of Presence
QoS Quality of Service
RAM Random Access Memory
RBAC Role-Based Access Control
REST Representational State Transfer
RTT Round Trip Time
SaaS Software as a Service
SASE Secure Access Service Edge
SD-WAN Software-Defined Wide Area Network
SIEM Security Information and Event Management
SLA Service Level Agreement
SOAR Security Orchestration, Automation, and Response
SSL Secure Sockets Layer
SSO Single Sign-On
SWG Secure Web Gateway
TLS Transport Layer Security
VPN Virtual Private Network
WAN Wide Area Network
ZTA Zero Trust Architecture
ZTNA Zero Trust Network Access

Literaturverzeichnis

Alevizos, Lampis. „Automated cybersecurity compliance and threat response using AI, blockchain and smart contracts“. In: *International Journal of Information Technology* (2024). DOI: 10.1007/s41870-024-02324-9. URL: https://link.springer.com/article/10.1007/s41870-024-02324-9 (besucht am 29. Nov. 2025).

Alsaadi, Husam Fadhil, Mustafa Ali Alasadi und Nasser Fadhl Fadhil. „Efficient NFS Model for Risk Estimation in a Risk-Based Access Control Model for IoT Applications“. In: *PMC* 8914835 (2022). DOI: 10.3390/s22052005. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC8914835/ (besucht am 7. Nov. 2025).

Antonakakis, Manos u. a. „A comprehensive survey of manual and dynamic approaches for taxonomy generation in cybersecurity“. In: *Knowledge and Information Systems* (2025). DOI: 10.1007/s10115-025-02382-w. URL: https://link.springer.com/article/10.1007/s10115-025-02382-w (besucht am 21. Okt. 2025).

Atlam, Hany F u. a. „Developing an Adaptive Risk-Based Access Control Model for the Internet of Things“. In: *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. 2017, S. 655–661. DOI: 10.1109/iThings-GreenCom-CPSCom-SmartData.2017.103.

Bayılmış, Cüneyt u. a. „A survey on communication protocols and performance evaluations for Internet of Things“. In: *Digital Communications and Networks* 8.6 (Dez. 2022), S. 1094–1104. ISSN: 23528648. DOI: 10.1016/j.dcan.2022.03.013.

Bundesamt für Sicherheit in der Informationstechnik. *IT-Grundschutz-Kompendium*. 2023. ISBN: 978-3-8462-0906-6. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/IT-GS-Kompendium/IT_Grundschutz_Kompendium_Edition2023.pdf (besucht am 7. Nov. 2025).

Center for Internet Security. *CIS Critical Security Controls® v8 CIS Critical Security Controls*. Techn. Ber. 2021. URL: https://www.cisecurity.org/controls/.

Cloud Security Alliance. *Zero Trust Guiding Principles v1.1*. Techn. Ber. Cloud Security Alliance, März 2024. URL: https://cloudsecurityalliance.org/artifacts/zero-trust-principles-v-1-1 (besucht am 7. Nov. 2025).

Cockburn Alistair. „The Hexagonal (Ports & Adapters) Architecture “. In: *HaT Technical Report 2005.02* (Apr. 2005). URL: https://alistair.cockburn.us/hexagonal-architecture (besucht am 7. Nov. 2025).

Crowdstrike. *Global Security Attitude Survey*. Techn. Ber. Crowdstrike, 2019. URL: https://www.crowdstrike.com/en-us/blog/global-security-attitude-survey-takeaways-2019/ (besucht am 7. Nov. 2025).

European Union Agency for Cybersecurity (ENISA). *ENISA Threat Landscape 2024*. Techn. Ber. ENISA, 2024. DOI: 10.2824/0710888. URL: https://www.enisa.europa.eu/publications/ENISA-Threat-Landscape-2024 (besucht am 7. Nov. 2025).

– *TECHNICAL IMPLEMENTATION GUIDANCE*. Techn. Ber. ENISA, Juni 2025. DOI: 10.2824/2702548.

Fernández Saura, Pablo u. a. „On Automating Security Policies with Contemporary LLMs“. In: *arXiv preprint arXiv:2506.04838* (2025). DOI: 10.48550/arXiv.2506.04838. URL: https://arxiv.org/pdf/2506.04838.pdf (besucht am 7. Nov. 2025).

Forest, Jonathan, Neil MacDonald und Dale Koeppen. *2025 Gartner Magic Quadrant for SASE Platforms*. Juli 2025. URL: https://www.catonetworks.com/resources/gartner-magic-quadrant-for-sase-platforms-2025/ (besucht am 14. Sep. 2025).

Garbis, Jason und Jerry W. Chapman. *Zero Trust Sicherheit: Ein Leitfaden für Unternehmen*. Deutsch. Apress, 2024. ISBN: 979-8-8688-0105-1.

Grand View Research. *Secure Access Service Edge Market Size, Share Report 2030*. 2024. URL: https://www.grandviewresearch.com/industry-analysis/secure-access-service-edge-market-report (besucht am 7. Nov. 2025).

GTI Group. *GTI Orchestration Framework for Secure Access Service Edge (SASE)*. 2024. URL: https://www.gtigroup.org/Uploads/File/2024/09/30/u66fa0bdd041ac.pdf (besucht am 7. Nov. 2025).

Hassan, Ahmed. „Enterprise Deployment Challenges of SASE: A Multi-Cloud Approach“. In: *European Journal of Computer Science and Information Technology* 13.50 (Juli 2025). DOI: 10.37745/ejcsit.2013/vol13n50152161. URL: https://eajournals.org/ejcsit/vol13-issue50-2025/enterprise-deployment-challenges-of-sase-a-multi-cloud-approach/ (besucht am 7. Nov. 2025).

Hevner, Alan u. a. „Design Science in Information Systems Research“. In: *Management Information Systems Quarterly* 28 (Sep. 2004), S. 75. DOI: 10.2307/25148625.

Horcas, J M, M Pinto und L Fuentes. „Runtime Enforcement of Dynamic Security Policies“. In: *Software Architecture. ECSA 2014. Lecture Notes in Computer Science*. Bd. 8627. Springer, 2014. URL: https://link.springer.com/chapter/10.1007/978-3-319-09970-5_29 (besucht am 7. Nov. 2025).

International Organization for Standardization. *ISO/IEC 27002:2022 — Information security, cybersecurity and privacy protection — Information security controls*. 2022. URL: https://www.iso.org/standard/75652.html (besucht am 7. Nov. 2025).

Königs, Hans-Peter. *IT-Risikomanagement mit System*. Deutsch. Techn. Ber. 2017, S. 1–482. DOI: 10.1007/978-3-658-12004-7.

Kruchten, Philippe. *Architectural Blueprints-The "4+1"View Model of Software Architecture*. Techn. Ber. 6. 1995, S. 42–50.

Macdonald, Neil. *The Future of Network Security Is in the Cloud: Introducing the Secure Access Service Edge*. Techn. Ber. 2020. URL: https://assets-powerstores-com.s3.amazonaws.com/data/org/20033/media/doc/the_future_of_network_security_is_in_the_cloud_int_1599852842797001fko4_(1)-9fb8d55d823780b55988b056046fbc1b.pdf (besucht am 7. Nov. 2025).

Mahmoud, Ahmed. „A Novel Proactive and Dynamic Cyber Risk Assessment Methodology“. In: *SSRN Electronic Journal* (2024). DOI: 10.2139/ssrn.5009538. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5009538 (besucht am 4. Dez. 2025).

MEF Forum. *MEF 117 - SASE Service Attributes and Service Framework*. 2022. URL: https://www.mplify.net/wp-content/uploads/MEF-117.pdf (besucht am 7. Nov. 2025).

– *MEF Standard MEF 70.2 SD-WAN Service Attributes and Service Framework*. Techn. Ber. 2023. URL: https://www.mplify.net/resources/mef-70-2-sd-wan-service-attributes-and-service-framework/ (besucht am 7. Nov. 2025).

Miller, Lawrence. *Secure Access Service Edge (SASE) For Dummies®, Palo Alto Networks 2nd Special Edition*. Techn. Ber. 2022.

– *Next-Generation Firewalls For Dummies®*. Wiley, 2011, S. 68. ISBN: 978-0-470-93955-0.

National Cyber Security Centre (UK). *Zero Trust Architecture: Design Principles*. Techn. Ber. NCSC, 2021. URL: https://www.ncsc.gov.uk/collection/zero-trust-architecture (besucht am 7. Nov. 2025).

National Security Agency. *Advancing Zero Trust Maturity Throughout the Automation and Orchestration Pillar*. Techn. Ber. NSA Cybersecurity Information, 2024. URL: https://media.defense.gov/2024/Jul/10/2003500250/-1/-1/0/CSI-ZT-AUTOMATION-ORCHESTRATION-PILLAR.PDF (besucht am 3. Okt. 2025).

Palo Alto. *Palo Alto Github Repository*. 2025. URL: https://github.com/PaloAltoNetworks (besucht am 7. Nov. 2025).

– *Palo Alto Prisma SASE API Dokumentation*. 2025. URL: https://pan.dev/sase/docs/ (besucht am 7. Nov. 2025).

Peffers, Ken u. a. „A design science research methodology for information systems research“. In: *Journal of Management Information Systems* 24 (Sep. 2007), S. 45–77. DOI: 10.2753/MIS0742-1222240302.

Rose, Scott u. a. *Zero Trust Architecture*. Techn. Ber. Gaithersburg, MD: National Institute of Standards und Technology, Aug. 2020. DOI: 10.6028/NIST.SP.800-207. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf (besucht am 7. Nov. 2025).

Sangfor Technologies. *What is a Secure Web Gateway (SWG)? A Comprehensive Guide to SWG*. Okt. 2022. URL: https://www.sangfor.com/glossary/cybersecurity/what-is-secure-web-gateway-swg (besucht am 7. Nov. 2025).

Shain Singh und Sebastian Weitzel. *OpenSase Github Repo*. 2020. URL: https://github.com/shsingh/opensase (besucht am 7. Sep. 2025).

Velayutham, Arunkumar. „Secure Access Service Edge (SASE) Framework in Enhancing Security for Remote Workers and Its Adaptability to Hybrid Workforces in the Post-Pandemic Workplace Environment“. In: *International Journal of Scientific Advances* (2023). URL: https://norislab.com/index.php/ijsa/article/view/94 (besucht am 7. Nov. 2025).

Zou, Lei u. a. „Communication-protocol-based analysis and synthesis of networked systems: progress, prospects and challenges“. In: *International Journal of Systems Science* 52.14 (2021), S. 3013–3034. ISSN: 14645319. DOI: 10.1080/00207721.2021.1917721.

Anhang

Der beiliegende Datenträger enthält die folgenden Dokumente:

- **-1- Vollständiger API Export aller Palo Alto Endpunkte**
  - Anhang -1- PaloAlto API Komplett.json
- **-2- Reduzierte Liste (CRUD) Palo Alto API Endpunkte**
  - Anhang -2- PaloAlto API CRUD Liste.json
- **-3- Reduzierte Liste FWaaS Palo Alto API Endpunkte**
  - Anhang -3- PaloAlto API FWaaS.json
- **-4- Reduzierte Liste CASB Palo Alto API Endpunkte**
  - Anhang -4- PaloAlto API CASB.json
- **-5- Reduzierte Liste SD-WAN Palo Alto API Endpunkte**
  - Anhang -5- PaloAlto API SD-WAN.json
- **-6- Reduzierte Liste ZTNA Palo Alto API Endpunkte**
  - Anhang -6- PaloAlto API ZTNA.json
- **-7- Reduzierte Liste SWG Palo Alto API Endpunkte**
  - Anhang -7- PaloAlto API SWG.json
- **-8- OpenSASE Artefakte**
  - Anhang -8- OpenSASE Artefakte.json

Eidesstattliche Erklärung

Studierender: Kevin Dallmann
Matrikelnummer: 863160

Hiermit erkläre ich an Eides statt, dass ich diese Arbeit selbstständig abgefasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Sämtliche Stellen der Arbeit, die im Wortlaut oder dem Sinne nach Publikationen oder Vorträgen anderer Autoren entnommen sind, habe ich als solche kenntlich gemacht. Ich bin mit einer Plagiatsprüfung einverstanden.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Weingarten, 31.01.2026 — Kevin Dallmann

Ort, Abgabedatum — Unterschrift (Vor- und Zuname)

WILHELM BÜCHNER HOCHSCHULE