FH Aachen
Fachbereich Elektrotechnik und Informationstechnik
Prof. Dr. Marko Schuba

FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Bachelorarbeit

Implementierung einer Zero Trust Netzwerk Architektur durch ein Service Mesh

Eingereicht von:
Calvin David Köcher
Matrikelnummer: 3231558

Studienrichtung: Informatik

17. August 2022

Betreuer: Prof. Dr. Marko Schuba
Korreferent: Dr. Tim Niggemann

Kurzfassung

IT-Systeme haben mittlerweile in allen Bereichen des beruflichen und privaten Lebens Einzug gehalten. Der Funktionsumfang und Einfluss dieser Systeme wächst stetig, was auch eine stärkere Vernetzung und Anbindung an das Internet bedeutet. Dabei schafft die steigende Komplexität und Konnektivität dieser Systeme eine kontinuierlich wachsende Angriffsfläche. Abhilfe verspricht das sogenannte "Zero Trust Modell", dieses bringt keinem System und keinen Verbindungen mehr Vertrauen entgegen, ohne dass deren Vertraulichkeit, Integrität und Authentizität gesichert ist.

In dieser Arbeit wird dieses Modell vorgestellt, implementiert und analysiert. Anhand einer Testumgebung wird der praktische Einsatz erprobt und Vor- und Nachteile im Bezug auf Sicherheit, Zuverlässigkeit und Wartbarkeit im Vergleich zur Perimeter basierten Netzwerkarchitektur aufgezeigt.

Schlagwörter: IT-Sicherheit, Cyberkriminalität, Lateral Movement, Zero Trust Model, Microservices, Mikrosegmentierung, Service Mesh, HashiCorp Consul

Abstract

IT systems have now found their way into all areas of professional and private life. The greater interoperability and connection to the Internet means that the range of functions and influence of these systems are continuously expanded. At the same time, the increasing complexity and connectivity of these systems creates a steadily growing attack surface. The "Zero Trust" promises to remedy this situation. It means that no system or connection can be trusted without ensuring its confidentiality, integrity and authenticity.

This model ist presented and analyzed in this work. Practical use is tested using a test environment, and advantages and disadvantages in terms of security, reliability and maintainability in comparison to the Perimeter Model are highlighted.

Keywords: IT security, Cybercrime, Lateral Movement, Zero Trust Model, Microservices, Microsegmentation, Service Mesh, HashiCorp Consul

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

In alphabetischer Reihenfolge:

AES	Advanced Encryption Standard
API	Application Programming Interface
ATTCK	Adversarial Tactics, Techniques,Common Knowledge
CNCF	Cliud Native Computing Foundation
CPU	Central Processing Unit
CRM	Customer Relationship Management
CVE	Common Vulnerabilities and Exposures
DC	Data Center
DMZ	Demilitarisierte Zone
DNS	Domain Name System
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IaaS	Infrastructure as aService
IaC	Infrastructure as Code
IoT	Internet of Things
IP	Internet Protokoll
IT	Informations Technologie
ITU-T	Telecommunication Standardization Sector
kB	Kilobyte
MB	Megabyte
MIT	Massachusetts Institute of Technology
ms	Millisekunden

mTLS	Mutal Transport Layer Security
NAT	Network Address Translation
OS	Operaring System
PaaS	Platform as a Service
PKI	Public Key Infrastructure
RSA	Rivest Shamir Adleman
SaaS	Software as a Service
SAN	Subject Alternative Name
SMB	Server Message Block
SSD	Solid State Drive
SSH	Secure Shell
TCI	Teledata Communications
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
VM	Virtuelle Maschine
VPN	Virtual Private Network
XML	Extensible Markup Language
YAML	Yet Another Markup Language
ZTN	Zero Trust Network
ZTNA	Zero Trust Network Architecture

1 Einleitung

1.1 Motivation

Informationen sind einer der wertvollsten Güter für Unternehmen, Staat und Gesellschaft. Dabei sind die informationsverarbeitenden Systeme ein zentraler Bestandteil und in nahezu allen Geschäftsprozesse involviert. Kommt es hier zu Beeinträchtigungen, kann dies schnell gesamte Unternehmen zum Stillstand bringen und einen hohen wirtschaftlichen Schaden verursachen. Dazu zählt vor allem der Diebstahl, die Manipulation oder der Verlust von Kunden- und Unternehmensdaten durch vorsätzliche Angriffe. Je nach Schwere und Dauer des Vorfalls, kann dies für eine Firma existenzbedrohend werden [LL22].

Schätzungen zufolge, entstand der deutschen Wirtschaft im Zeitraum 2020/2021 ein Schaden von ca. 223,5 Milliarden Euro pro Jahr durch Cyberangriffe [GA22]. Die Schadensumme hat sich hierbei im Gegensatz zum Zeitraum 2018/2019, wo diese noch bei geschätzten 103 Milliarden Euro pro Jahr lag, mehr als verdoppelt. Viele Angriffe zielen auf den Faktor Mensch ab. Hierbei werden beispielsweise durch Social Engineering, menschliches Fehlverhalten ausgenutzt um an sensible Daten wie Passwörter zu gelangen. Umfragen zur Folge trägt auch der vermehrte Einsatz von Homeoffice zur steigenden Zahl von Sicherheitsvorfällen bei. Als Gründe hierfür lassen sich unter anderem ungeschützte Geräte und Kommunikationskanäle anführen. Laut Umfragen haben 24 % der Unternehmen als Reaktion auf die verschärfte Bedrohungslage ihre Investitionen in IT-Sicherheit deutlich erhöht. Dabei bleiben die durchschnittlichen Aufwendungen für IT-Sicherheit verhältnismäßig jedoch weiter gering. Im Schnitt werden nur 7 % der verfügbaren IT-Mittel für IT-Sicherheit verwendet [GA22].
Angriffe gegen Unternehmen und deren IT-Systeme werden gezielter, ausgefallener und unauffälliger [Kin10b]. Dabei dringen Angreifer häufig tief in die Systeme ein und extrahieren Daten und vertrauliche Informationen oft über Wochen oder Monate hinweg [Kin10b]. Der Zugangspunkt liegt dabei häufig im Inneren des Unternehmens, in einer vertrauten Umgebung. Dieser kann ein Mitarbeiter mit böswilligen Absichten oder ein System sein, über welches der Angreifer mittels Schadcode die Kontrolle übernehmen konnte. Bisher gängige Schutzkonzepte sind oft machtlos gegen Angriffe, welche aus dem inneren des Netzwerks heraus stattfinden, da sich bei diesen die Sicherheitsmechanismen an den Grenzen verschiedener Netzwerkzonen befinden. Homeoffice, stärker vernetzte Anwendungen und der Einsatz der Cloud weichen die Grenzen zwischen “vertrauten“ und “nicht vertrauten“ Bereichen des Netzwerks zusätzlich auf und schwächen so die vorhandenen Sicherheitsmechanismen weiter [Kin10b].

Neue Gefahren erfordern neue Sicherheitskonzepte und ein überarbeitetes Verständnis von Vertrauen in die eigenen Netzwerke und Systeme. Jegliche Kommunikation ist als unsicher anzusehen bis ihr Ursprung, ihr Ziel und ihr Inhalt durch technische Mechanismen kontrolliert wurde. Dabei darf Vertrauen nicht länger durch eine bestimmte Zone oder einen bestimmten Netzwerkabschnitt impliziert werden [Kin10b].

1.2 Ziel dieser Arbeit

Das Ziel dieser Arbeit ist es, eine mögliche Lösung für die beschriebenen Sicherheitsprobleme zu untersuchen und deren Vor- und Nachteile im Vergleich zum etablierten Perimeter-Modell aufzuzeigen. Als Lösungsansatz wird in dieser Arbeit, das von John Kindervag entwickelte "Zero Trust-Modell" vorgestellt. Es wird gezeigt, wie sich dieses Modell technisch in einer IT-Umgebung realisieren lässt und welche Besonderheiten und Limitierungen zu beachten sind. Anhand dieser Testumgebung können Aussagen im Bezug auf Zuverlässigkeit, Performance und Administrierbarkeit getroffen werden. Es wird gezeigt, in welchem Rahmen das Modell einen Schutz gegen bekannte Angriffe bietet und wo neue Sicherheitslücken entstehen können. Abschließend wird bewertet, ob das Zero Trust Modell eine echte Alternative zum vorherrschenden Sicherheitsmodell darstellt und für welche Anwendungsfälle dies in Frage kommt.

2 Problemstellung

Im folgenden Kapitel wird die Problemstellung und deren Ursachen genauer betrachtet und durch Beispiele verdeutlicht.

2.1 Ursachen der Gefährdung

Um die veränderte Bedrohungslage besser verstehen zu können, hilft ein Blick in die Vergangenheit. Dazu betrachten wir exemplarisch das "Philip Cummings Problem" [Kin10b]. Dieses beschreibt einen schwerwiegenden Sicherheitsvorfall in der amerikanischen Firma Teledata Communications (TCI) aus dem Jahr 2000, in welcher Philip Cummings als HelpDesk Angestellter tätig war. TCI stellte Software für Kreditinstitute her, Philip Cummings war hier als Support-Mitarbeiter tätig und hatte dadurch Zugriff auf Kundendaten und Kundenpasswörter. Während seiner Anstellung wurde er von einer kriminellen nigerianischen Organisation kontaktiert, welche ihm 60 Dollar für jede Kreditauskunft bot. Philip Cummings ließ sich auf das Geschäft ein und begann die gewünschten Information zu stehlen und zu verkaufen. Auch wenn dieser Vorgang natürlich illegal war, stellt dieser dennoch einige wichtige Aspekte anschaulich dar, welche im Bezug auf IT-Sicherheit betrachtet werden müssen:

- Durch einen durch Cummings speziell programmierten Laptop im Unternehmensnetzwerk, waren seine kriminellen Partner nun eigenständig in der Lage, Informationen der Kreditinstitute abzugreifen. Bemerkenswert hierbei ist, dass die meisten kriminellen Aktivitäten im Zeitraum zwischen 2000 und 2002 stattfanden, obwohl Cummings das Unternehmen bereits im Jahr 2000 verlassen hatte. In diesen zwei Jahren, bemerkte weder TCI, noch die dazugehörigen Kreditinstitute das Datenleck, da die Daten langsam und unauffällig an Sicherheitseinrichtungen vorbei abflossen.
- TCI hat das Verbrechen nie entdeckt, tatsächlich war es ein Kunde eines Kreditbüros, der im Jahr 2002 auf den Betrug aufmerksam wurde. Diesem sind bestimmte Muster in den zahlreichen Beschwerden von Verbrauchern aufgefallen, welche Opfer von Identitätsdiebstahl und Betrug geworden waren.
- Es wird geschätzt, dass Informationen von circa 30.000 Personen und Organisationen gestohlen wurden. Der finanzielle Schaden belief sich dabei auf mindestens 2,7 Millionen Dollar. Cummings wurde zu 14 Jahren Haft und 1 Million Doller Schadensersatz verurteilt. Bis heute bleibt dieser Vorfall einer der größten Identitätsdiebstähle in der US-Geschichte [fbi04; Law15]. Nur durch Insiderinformationen, genaue Kenntnisse

über das betroffene IT-System und einem Angriff, welcher auf dem Inneren des Unternehmensnetzwerkes ausging, war diese Attacke so lange erfolgreich ohne dabei entdeckt zu werden.

[Kin10b; Sul04]

Auch wenn an dieser Stelle ein älteres Beispiel gewählt wurde, ist dessen Aktualität nach wie vor unverändert. Aktuelle Netzwerke und Systeme werden immer komplexer und abhängiger von externen Ressourcen und Dienstleistern. Das Netzwerk wird ein immer wichtigeres Bindeglied im Betrieb moderner Softwaresysteme, daher werden auch mehr Daten über das Netzwerk übertragen. Homeoffice und Cloudanwendungen erfordern oft weitreichenden Zugriff auf interne Ressourcen aus dem Internet. All diese Veränderungen schaffen ein komplexeres Netzwerk, Grenzen wie "sichere/unsichere" oder "interne/externe" Netzwerkzonen werden unschärfer und schwerer zu schützen [WO20].

Wie das vorangestellte Beispiel gut illustriert, ist Netzwerksicherheit eine vielschichtige Anforderung, bei welcher der alleinige Schutz vor Angriffen von außerhalb nicht ausreicht. Stattdessen müssen auch interne Aktivitäten und Datenströme genau so akribisch überwacht und beschränkt werden wie externe Verbindungen. Dieser Gedanke ist der erste Schritt in Richtung des "Zero Trust Modells", welches in Kapitel 3.1 genauer vorgestellt wird und diesen Grundgedanken weitertreibt.

2.1.1 Das Perimeter-Modell

John Kindervag, der Erfinder des Zero Trust Modells, beschrieb das Perimeter-Modell wie folgt: "Wir wollen dass, unsere Netzwerke sind wie ein M&M, mit einer harten und knusprigen Schale und einem weichem Inneren." [Kin10b]. Dies ist das Motto, mit dem eine ganze Generation von Sicherheitsexperten aufgewachsen ist. Dieses basiert auf der Annahme, dass böswillige Aktivitäten die "harte Schale" nicht durchdringen können [Kin10b]. Seit vielen Jahrzehnten stellt diese Annahme die Vertrauensbasis dar, auf der Personen oder Dienste auf Ressourcen zugreifen können [Ker+20].

Die herkömmliche Netzwerkarchitektur unterteilt das vorhandene Netzwerk in verschiedene Zonen, welche durch eine oder mehrere Firewalls getrennt werden. Für jede dieser Zonen wird ein bestimmtes Maß an Vertrauen festgelegt, dadurch wird bestimmt, welche Ressourcen in diesem Netzwerk erreichbar sind. Hierbei können die Netzwerke ineinander verschachtelt werden, was zu einer stärkeren Verteidigung in der Tiefe führt. Ein Angreifer muss theoretisch alle Zonen nacheinander durchdringen, um zur innersten Zone zu gelangen. Die genaue Aufteilung und Bezeichnung der Zonen, sowie deren Anzahl hängen vom jeweiligen Anwendungsfall ab, typische Zonen bei diesem Modell sind:

- **Internet**
  Dem Internet als externes Netzwerk wird keinerlei Vertrauen entgegen gebracht, ein Angriff aus diesem Bereich gilt als erwartbar. Ziel ist es, die sensiblen Netzwerke des Unternehmens weitestgehend mittels Firewalls abzuschotten.

- **Demilitarisierte Zone**
  Eine demilitarisierte Zone wird als Pufferzone zwischen einem externen und dem internen Netzwerk eingesetzt. Hier werden oft Dienste und Server platziert, welche aus dem Internet erreichbar sein müssen und damit einer höheren Gefährdung ausgesetzt sind. Dazu können beispielweise Web-, Mail- oder VPN-Server zählen. Die Besonderheit einer DMZ liegt in ihrer Position zwischen dem Internet und dem internen Netzwerk. Dadurch ist es möglich, Anwendern Zugriff auf interne Ressourcen zu gewähren, ohne diese direkt aus dem Internet erreichbar zu machen. Die Kommunikation wird hierbei von einem Server in der DMZ in das interne Netz weitergeleitet Diese Server werden auch als “Jump-Host“ oder “Relais Server“ bezeichnet.

- **Vertraute Zone**
  Das interne Unternehmensnetzwerk stellt die vertraute Zone dar. Hier befinden sich ein Großteil der Anwender, Clientrechner sowie die internen Server.

- **Privilegierte Zonen**
  Diese Zonen können eingerichtet werden, um gegebenenfalls sensible Daten noch besser zu schützen. Dazu können bestimmte Server oder ganze Abteilungen vom restlichen Unternehmensnetzwerk abgegrenzt werden. Dazu wird der Datenverkehr der vertrauten Zone durch zusätzliche Firewalls weiter beschränkt.

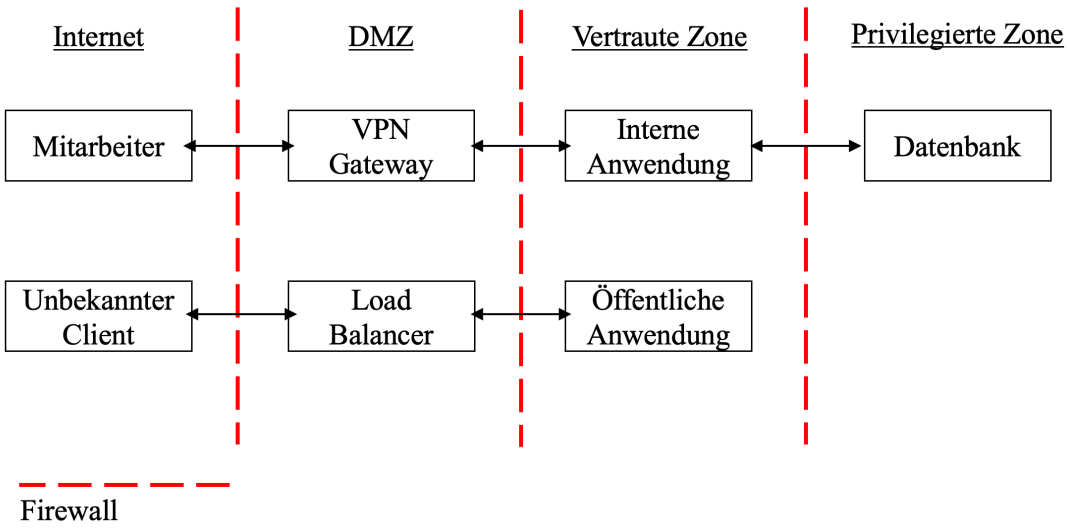

Abbildung 2.1: Herkömmliche Netzwerkarchitektur nach dem Perimeter Modell - [GB17]

Die Grafik zeigt eine für das Perimeter Modell typische Aufteilung des Netzwerks in die Zonen: Internet, Demilitarisierte Zone, Vertraute Zone und der Privilegierten Zone.

Die Platzierung mehrerer Hürden in der Tiefe des Netzwerks ist ein Fortschritt gegenüber den flachen Netzwerke aus den Anfängen des Internets [Kin10b]. Jedoch ist dieser Schutz unzureichend gegen moderne Cyberattacken und bietet einige Nachteile gegenüber einer Zero Trust Architektur:

- Es findet keine Überwachung des Datenverkehrs innerhalb der Zonen statt.
- Server können nicht flexibel zwischen den Zonen verschoben werden (sowohl physisch als auch logisch).
- Firewalls können einen "Single Point of Failure" darstellen.
- Ist es einem Angreifer gelungen die äußere Schale zu durchdringen, erhält dieser Zugriff auf den gesamten Netzwerkbereich und dessen Ressourcen.
- Gut organisierte Cyberkriminelle können durch das Rekrutieren von Insidern oder der Entwicklung neuartiger Angriffsmethoden, diese Sicherheitsmechanismen umgehen.

[Kin10b; GB17]

2.1.2 Trend zu Mikroservices

Unter Microservices versteht man in der Softwareentwicklung eine Architektur, bei welcher die Software in kleine, voneinander unabhängige Dienste, sogenannte Services, aufgeteilt wird. Jeder Services wird im Hinblick auf einzelne Unternehmensfunktionen entwickelt und erfüllt eine bestimmte Funktion. Die einzelnen Services kommunizierten untereinander über das Netzwerk mittels sorgfältig definierter APIs. Dadurch, dass jeder Service eine eigenständige Anwendung ist und diese unabhängig voneinander ausgeführt werden, können diese problemlos aktualisiert, bereitgestellt und skaliert werden. Auch eine vertikale Skalierung der Services ist durch den Betrieb mehrerer Instanzen des selben Services, verteilt in einem oder mehreren Rechenzentren, einfacher möglich. In der Regel sind für jeden Service kleine, voneinander unabhängige, Teams zuständig wodurch eine verringerte Entwicklungszeit erreicht werden kann.

Das Gegenstück zu Microservices bildet die weit verbreitete monolithische Softwarearchitektur. Bei dieser sind alle Prozesse eng miteinander verbunden und in einer Anwendung gebündelt [aws22a]. Diese Anwendung wird als einzelner Service betrieben. Eine flexible Skalierung einzelner Prozesse ist hierbei nicht möglich. Tritt in einem Prozess der Anwendung eine Spitze auf, muss die gesamte Anwendung skaliert werden. Aufgrund der häufig sehr umfangreichen Codebasis solcher Anwendungen, ist das Hinzufügen und Verbessern einzelner Bestandteile der Anwendung aufwändig [aws22a]. Auch das Experimentieren und die Umsetzung neuer Konzepte wird durch die Größe und Komplexität monolithischer Softwarearchitekturen erschwert. Viele miteinander verknüpfte und voneinander abhängig Prozesse erhöhen zudem die Auswirkungen eines einzelnen Prozessausfalls[aws22a].

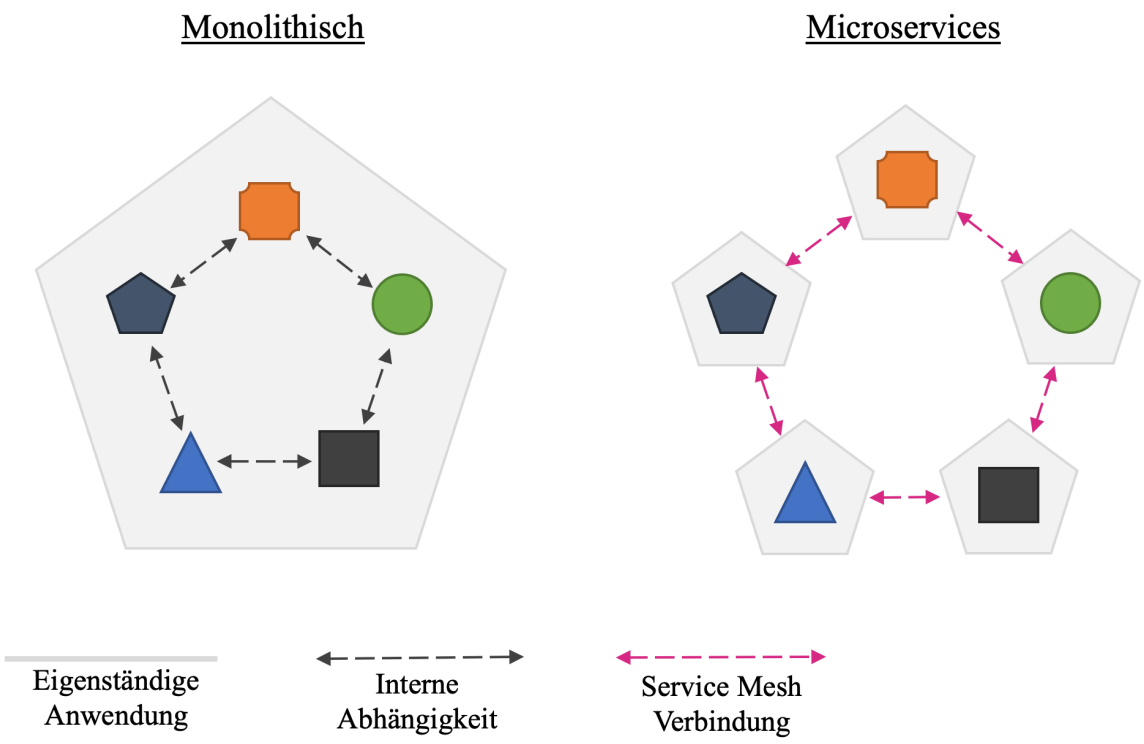

Abbildung 2.2: Schematischer Vergleich zwischen einer Monolithischen Softwarearchitektur und Mikroservices

Bei der auf der linken Seite dargestellten Monolithischen Softwarearchitektur ist die Programmlogik in einer einzelnen Anwendung gebündelt. Im Gegensatz dazu teilt das Konzept der Microservices auf der rechten Seite diese Logik auf mehrere voneinander unabhängige Services auf.

2.1.3 Trend zu Cloudanwendungen

Unternehmen verlagern ihre On-Premise-Infrastruktur und Workflows immer mehr in die Cloud. Dies ist schon lange kein vorübergehender Trend mehr, sondern mittlerweile gängige Unternehmenspraxis [Bre21]. Die weltweiten Ausgaben für Public-Cloud-Dienste im Jahr 2022 werden vom Marktforschungsinstitut Gartner auf insgesamt 494,7 Milliarden US-Dollar geschätzt. Das ist ein Plus von 20,4 % zum Vorjahr, das höchste Wachstum für 2022 wird mit 30,6 % im Bereich der System as a Service (SaaS) erwartet [gar22].
Der Wechsel zur Cloud bietet Unternehmen viele Vorteile. In der Cloud lassen sich IT-Ressourcen flexibel ändern, dadurch lassen sich Prozesse und Geschäftsmodelle schnell an sich verändernde Anforderungen anpassen. Unternehmen können direkt von Innovationen und Weiterentwicklungen seitens des Cloudanbieters in Form von neuen oder verbesserten Angeboten profitieren. Im Gegensatz zu On-Premise-Lösungen, verursachen Cloud-Services keine hohen Investitionen in die benötigte Hardware und deren Unterhalt. Ein auf das jeweilige Unternehmen zugeschnittener Leistungsumfang, kann vergleichen mit einer schwerer skalierbaren On-Premise-Lösung, zu einer Kostenreduktion führen [Bre21]. Es gibt folgende Stufen, in welchem Rahmen Cloudanbieter ihre Dienste anbieten:

- **Infrastructure as a Service (IaaS) z.B. ganze Systeme und Server**
  Hier wird dem Kunden Hardware zur Verfügung gestellt, über welche dieser frei verfügen kann. Bei diesem Modell hat der Kunde die größte Kontrolle und Verantwortung, der Service Provider ist lediglich für die Bereitstellung des Servers, des Netzwerks und des Speicher zuständig.
- **Platform as a Service (PaaS) z.B. Datenbanken und Entwicklungsumgebungen**
  Bei dieser Form des Cloud-Computings wird die Hardware zusammen mit einer Anwendungsplattform bereitgestellt. Der Kunde hat weiterhin volle Kontrolle und Verantwortung über sein System, kann jedoch die vom Service-Provider bereitgestellte Anwendungen nutzen.
- **Software as a Service (SaaS) z.B. E-Mail, Messaging und CRM**
  Der Kunde erhält Zugriff auf eine Anwendung und kann diese im Rahmen des geschlossenen Vertrages nutzen. Über die darunterliegende Infrastruktur zur Bereitstellung der Anwendung hat Kunde keine Kontrolle, diese liegt voll im Verantwortungsbereich des Service-Providers. Das Gegenstück dazu bildet die Private-Cloud, hier wird die Bereitstellung zwar weiterhin vollständig vom Service-Provider übernommen, jedoch stellt dieser dedizierte Hardware für den Kunden bereit.

Der Umstieg in die Cloud findet meist schrittweise statt, oft werden zunächst nur einzelne Prozesse oder Systeme übertragen. Dies führt zu einem hybriden Betrieb aus Cloud- und On-Premise-Diensten. Die sichere Vernetzung von Services in verschiedenen Rechenzentren über das Internet kann dabei zum Problem werden. Haben die Services viele Abhängigkeiten untereinander, führt dies zu einem hohen und schwer überschaubaren Datenverkehr zwischen den Standorten. Eine Reglementierung einzelner Service-zu-Service Verbindung ist daher meist nicht mehr praktikabel durchführbar. Als Konsequenz dessen, entscheiden sich Administratoren daher häufig dazu, diese Netzwerkbereiche beispielsweise durch VPNs zusammenzuschließen. Der Perimeter des Netzwerks wird dadurch immer unschärfer und bietet potentiellen Angreifern mehr Angriffsfläche und Möglichkeiten zur Ausbreitung.

2.1.4 Lateral Movement

Angriffe, welche einen Zugangspunkt innerhalb der eigenen Netze haben, sind auch ohne Insider wie Cummings mit böswilligen Absichten oder Personen mit grob fahrlässigen Handeln möglich. Heutige IT-Systeme werden immer komplexer und vielfältiger. Zudem erfolgt oft eine Erweiterung der Software durch Updates, Plugins und AddOns von Drittanbietern. Es reicht meist ein Design- oder Programmierfehler in der Anwendung, um eine Schwachstelle zu schaffen. Die Common Vulnerabilities and Exposures (CVE) Datenbank der Mitre Corporation umfasst aktuell ca. 180.000 Einträge (Stand Juli 2022) [mit22a]. Jeder Eintrag in dieser Datenbank repräsentiert eine bekannte sicherheitsrelevante Schwachstelle einer Software in einer bestimmten Version. Durch manche dieser Schwachstellen

ist es einem Akteur möglich, die Kontrolle über die Anwendung und den darunterliegen Server zu übernehmen. Diese verwundbaren Anwendungen oder Systeme sind dabei nicht zwingend das eigentliche Ziel des Angreifers, sondern oft nur Teil des Angriffs. Hat ein Angreifer die Kontrolle über einen Server des Unternehmens übernommen, baut dieser seinen Zugriff darauf aus und sichert diesen. Im Anschluss wird die vorteilhafte Position des Servers innerhalb des Unternehmensnetzwerkes ausgenutzt um weitere verwundbare Server ausfindig zu machen und zu übernehmen. Dabei kann sich ein Angreifer über mehrere Server bewegen um sein Hauptziel zu erreichen [GB17, S.13]. Dieses Vorgehen wird als "Lateral Movement" bezeichnet. Besonders in sehr flachen Netzwerkarchitekturen ist dies möglich, da keine Firewalls oder sonstige Sicherheitseinrichtungen auf Netzwerkebene überwunden werden müssen.

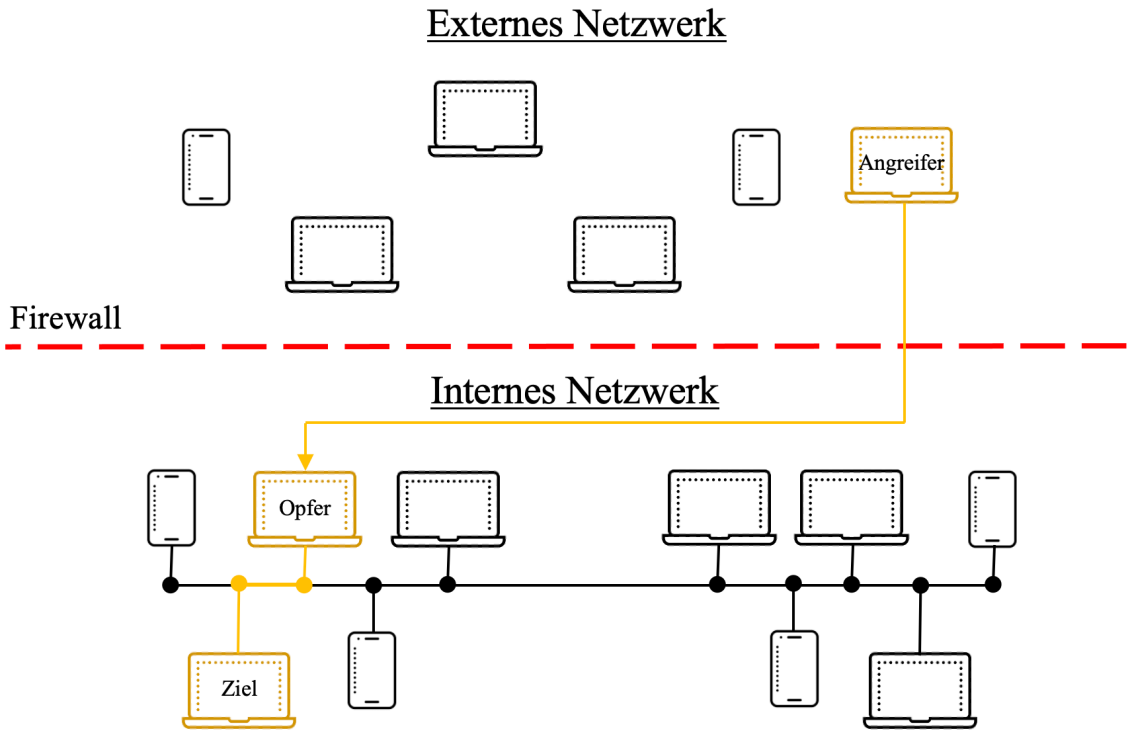

Abbildung 2.3: Schematische Darstellung der Wege eines Angreifers welcher sich durch Lateral Movement im Netzwerk ausbreitet

Die Abbildung zeigt ein externes- und ein internes Netzwerk, welche durch den Perimeter voneinander getrennt wird. Es ist dargestellt, wie ein Angreifer in das interne Netzwerk eindringt und sich vom ersten übernommenen System zum eigentlichen Ziel bewegt.

2.2 Handlungsmöglichkeiten

Systeme und Netzwerke, welche nicht vollständig von der Außenwelt abgeschottet sind, sind immer dem Risiko ausgesetzt, potentiellen Angreifern eine durch menschliche oder technische Fehler geschaffene Angriffsfläche zu bieten. Beispielweise können auch beobachtbare physikalische Effekten bei der Verarbeitung von sensitiven Daten dazu verwendet werden, um Erkenntnisse über diese zu gewinnen. Zu diesen Effekte zählen unter anderem Laufzeitverhalten, Energieverbrauch, elektromagnetische Abstrahlung und Cache-Verhalten eines Systems [bsi22]. Daher kann kein System seine absolute Sicherheit garantieren. Es ist lediglich möglich Hürden zu schaffen, welche es einem Angreifer erschweren, Sicherheitslücken auszunutzen und den potenziellen Schaden den diese anrichten können zu verringern. Auch Angriffe von Insidern, welche diese im Rahmen ihrer Befugnisse durchführen, können durch technische Lösungen nicht ausgeschlossen werden.

2.2.1 Ausbreitung verhindern

Bei der Prävention von weitreichenden Sicherheitsvorfällen wird oft von der "Begrenzung des Explosionsradius" gesprochen. Damit ist ein Sicherheitsmodell gemeint, das die Auswirkung eines einzelnen Sicherheitsvorfalls beschränkt und eine weitere Ausbreitung verhindern kann. Auf Netzwerkebene wird dies oft durch die Unterteilung des Netzwerks in individuellen Sicherheitssegmente oder Sicherheitszonen erreicht. Ziel ist es, diese möglichst klein zu fassen. Die Zonen können sogar bist zur Ebene einzelner Workloads reichen. Für jede Zone werden Sicherheitskontrollen definiert und der Netzwerkverkehr zwischen den Zonen durch Firewalls überwacht.

Eine Herausforderung kann die Einteilung der Dienste und Server in sinnvolle Gruppen sein. Wird beispielsweise ein Dienst von zwei ansonsten unabhängigen Anwendungen benötigt, kann das zu Segmente führen die größer sind als eigentliche notwendig. Zudem schafft jedes Segment zusätzlichen administrativen Aufwand in Form von Netzwerkkomponenten, Firewalls und deren Regelwerke. Hier muss oft ein Kompromiss zwischen Aufwand und Sicherheit gefunden werden.

2.2.2 Detektion ermöglichen

Da bestimmte Angriffe nie ganz ausgeschlossen werden können und eine absolute Sicherheit nicht garantiert werden kann, ist es umso wichtiger, Angriffe beziehungsweise deren Versuch zuverlässig zu detektieren und rekonstruieren zu können. Es ist eine flächendeckende Protokollierung der Aktivitäten im Netzwerk erforderlich. Im Falle eines Angriffs oder einem auffälligen Verhalten eines bestimmten Systems, muss automatisch eine Benachrichtigung der Administratoren erfolgen.

2.2.3 Kontrolle statt Vertrauen

Es darf keine Unterscheidung mehr zwischen “sicheren“ bzw. vertrauenswürdigen und “unsicheren“ bzw. nicht-vertrauenswürdigen Netzwerksegmenten stattfinden. Stattdessen ist jedes Netzwerk und jedes System als potenziell kompromittiert anzusehen. Dies gilt insbesondere für die internen Server und Netzwerke. Bei jeder Verbindung zu einem anderen Netzwerkteilnehmer muss dessen Identität sichergestellt sein, bevor Daten empfangen oder gesendet werden. Nur Verbindung welche explizit erlaubt sind werden zugelassen. Alle anderen werden blockiert und protokolliert. Jeder Netzwerkteilnehmer erhält nur Zugriff auf jene Netzwerkressourcen, welche für seine Funktion zwingend erforderlich sind. Dies ist der Grundgedanke des Zero Trust Modells, welches im folgenden Kapitel detaillierter vorgestellt wird.

3 Stand der Technik

Im folgenden Kapitel wird ein möglicher Lösungsansatz zu Umsetzung der beschrieben Handlungsmöglichkeiten vorgestellt. Dabei werden nötige technische Grundlagen und Konzepte geschaffen.

3.1 Das Zero Trust Modell

3.1.1 Historie

John Kindervag entwickelte das Zero Trust Modell im Jahr 2010. Als leitender Analyst bei Forrester-Research erkannte Kindervag, dass herkömmliche Zugriffsmodelle auf der überholten Annahme basierten, dass Unternehmen ihren Netzwerken vertrauen sollten. Kindervag erkannte früh die Gefahren dieser Modelle [Kin10b; Kin10a; Tur22].
Zeitgleich begann Google mit der Entwicklung seiner eigenen Zero Trust Systeme. Google schuf BeyondCorp für die Migration herkömmlicher VPN-Zugriffsrichtlinien (Virtual Private Network) auf eine neue Infrastruktur. Dieses basiert ebenfalls auf dem Grundgedanken, dass keine Systeme vertrauenswürdig. Google stellt eine umfangreiche Dokumentation zur Verfügung, die heute einen Industriestandard für Zero Trust setzt [Tur22].

3.1.2 Grundsätze

Das Zero Trust Modell lässt sich wie folgt zusammenfassen: “Kontrolle statt Vertrauen“. Zero Trust hilft dabei, Lateral-Movement und weitreichende Sicherheitsvorfälle zu verhindern, indem es das implizite Vertrauen gegenüber Systemen entfernt. Zero Trust fordert eine Verifizierung vor jedem Netzwerkzugriff. Im Wesentlichen erkennt Zero Trust nicht nur an, dass Bedrohungen innerhalb und außerhalb des Netzwerks existieren, sondern geht auch davon aus, dass ein Angriff unvermeidlich ist (oder wahrscheinlich bereits stattgefunden hat). Daher wird ständig nach bösartigen Aktivitäten gesucht und der Benutzerzugriff auf das beschränkt, was für die Ausführung der Aufgabe zwingend erforderlich ist. Auf diese Weise wird verhindert, dass sich ein böswilliger Akteure durch das Netzwerk bewegen und auf Daten zugreifen können, die nicht explizit eingeschränkt wurden. Das Zero Trust Modell erstreckt sich über alle Bereiche und Schnittstellen einer IT-Infrastruktur. Dazu gehören neben dem bereits angesprochenen Netzwerk auch: Anwendungen, Endgeräte, Identitäten, Infrastruktur und Daten [GB17; Tur22].

Aufbau
Das Modell lässt sich auf folgende acht Säulen reduzieren. Sie bilden eine defensive

Architektur, welche den Anforderungen komplexer Netze von heute gerecht wird.

- **Sichere Identitäten**
  Dieser Bereich wird auch als Mitarbeiter- oder Benutzersicherheit bezeichnet und konzentriert sich auf die Verwendung von Authentifizierungs- und Zugriffskontrollrichtlinien. Diese identifiziert und zu überprüft Benutzer, die versuchen eine Verbindung zum Netzwerk herzustellen. Die Identität eines Benutzers oder einer Einheit wird durch eine Reihe von Attributen eindeutig beschrieben. Durch eine dynamische und kontextbezogenen Datenanalyse wird sichergestellt, dass den richtigen Benutzern zum richtigen Zeitpunkt Zugriff gewährt wird [Tur22].

- **Sichere Endpunkte**
  Ähnlich wie die Identitätssicherheit führt auch die Endpunkt-Sicherheit eine Validierung der Geräte durch. Dazu zählen sowohl benutzergesteuerte als auch autonome Geräte, die versuchen sich mit dem Unternehmensnetzwerk zu verbinden. Diese Säule konzentriert sich auf die Überwachung und Aufrechterhaltung des Gerätezustands bei jedem Schritt. Unternehmen sollten alle Geräte des Unternehmens (einschließlich Mobiltelefone, Laptops, Server und IoT-Geräte) inventarisieren und sichern um zu verhindern, dass nicht autorisierte Geräte auf das Netzwerk zugreifen [Tur22].

- **Sichere Anwendungen**
  Die Anwendungs- und Workload-Sicherheit umfasst sowohl lokale als auch cloudbasierte Dienste und Systeme. Die Sicherung und Verwaltung der Anwendungsschicht ist ein wichtiger Bestandteil auf dem Weg zu einer Zero Trust Umgebung [Tur22].

- **Sichere Daten**
  Diese Datensäule konzentriert sich auf die Sicherung des Zugriffs auf Daten. Dazu werden die Daten kategorisiert und auf der Grundlage ihrer Kritikalität einem Speicherort zugeordnet. Zudem ist die Entwicklung einer entsprechenden Datenverwaltungsstrategie Teil eines robusten Zero Trust Ansatzes [Tur22].

- **Sichtbarkeit und Analysen**
  Der Einblick in alle Sicherheitsprozesse und die Kommunikation im Zusammenhang mit Zugriffskontrolle, Segmentierung, Verschlüsselung und anderen Zero Trust Komponenten bietet entscheidende Informationen über das Verhalten von Benutzern und Systemen. Dadurch kann die Erkennung und Analyse von Bedrohungen verbessert werden. Es wird dem Administrator ermöglicht, fundierte Sicherheitsentscheidungen zu treffen und sich an die sich ständig verändernde Sicherheitslandschaft anzupassen [Tur22].

- **Automatisierung**
  Automatisierung kann wesentlich dazu beitragen menschliche Fehler zu vermeiden und die Infrastruktur konsistent und effektiv zu administrieren. Sie ermöglicht es, Sicherheitskonzepte auch bei wachsenden IT-Infrastrukturen effektiv zu skalieren [Tur22].

- **Sichere Infrastruktur**
  Diese Säule stellt den Schutz von Systemen und Workloads vor unerlaubten Zugriffen und potentiellen Sicherheitsrisiken sicher [Tur22].

- **Sichere Netzwerke**
  Ein wesentlicher Punkt des Zero Trust Modells konzentriert sich auf die Isolierung sensibler Ressourcen vor unbefugtem Zugriff. Dazu gehört die Implementierung von Techniken zur Mikrosegmentierung, die Festlegung des Netzzugangs und die Ende-zu-Ende-Verschlüsselung des Datenverkehrs [Tur22].

Eine Implementierung des Zero Trust Modells bringt gegenüber einer klassischen Netzwerkarchitektur Vorteile, welche über die Sicherheitsaspekte hinaus gehen. Es vereinfacht zum einen den effektiven und sicheren Zugriff auf benötigte Ressourcen. Zudem kann das hybride Arbeiten mit On-Premise- und Cloud-Lösungen verbessert werden [Tur22]. Zero Trust kann auch eine Alternative zum Einsatz von virtuellen privaten Netzwerken (VPNs) zwischen verschiedenen Standorten und Rechenzentren darstellen. VPNs bieten eine Perimeter-basierte Sicherheit, wodurch meist ein Netzwerk-weiter Zugriff gewährt wird, wenn dies nicht durch spezielle Zonen und Filter verhindert wird. Im Gegensatz dazu gewährt Zero Trust nach einer Überprüfung und Authentifizierung nur Zugriff auf bestimmte Ressourcen, welche auch wirklich benötigt werden. Dies stärkt die Sicherheit von internen und externen Netzwerken, indem die Angriffsfläche reduziert und eine granularere Kontrolle implementiert wird [Tur22].

3.1.3 Vertrauen schaffen durch ITU-T X.509

Die Identifizierung seinen Kommunikationspartner anhand von Attributen wie dessen IP-Adresse, Passwörter oder sonstigen Geheimnissen ist unzureichend. Es muss davon ausgegangen werden, dass ein Angreifer bereits die Kontrolle über das Netzwerk übernommen hat und in der Lage ist, Daten von beliebigen IP-Adressen zu senden und den Datenverkehr mitzulesen oder zu manipulieren. Auch bei bisher unbekannten Kommunikationspartnern muss es jedem Teilnehmer möglich sein, verlässlich sicherzustellen, dass dieser auch derjenige ist, der er vorgibt zu sein.

X.509

Vertrauen in die Identität digitaler Systeme und Daten lässt sich nur durch den Einsatz von kryptographischen Verfahren gewinnen. Der am weitesten verbreitete Ansatz um diese Ziele zu erreichen, ist der ITU-T X.509 Standard [ITU19b] zur Erstellung einer Public-Key-Infrastruktur. X.509 wurde im Jahr 1998 erstmals veröffentlicht und hat seinen Ursprung in der Entwicklung des X.500 Standards [ITU19a]. Dieser ist heute ein De-facto-Standard zur Erstellung digitaler Zertifikate und findet beispielsweise Anwendung im ebenfalls weit verbreiteten TLS Verschlüsselungsprotokoll. Durch diese Zertifikate ist es möglich, die Identität eines Systems durch kryptographischen Verfahren zu verifizieren.

Die Integrität eines Zertifikats wird durch zwei Schlüssel sichergestellt: dem öffentlichen und dem privaten Schlüssel, welche als Paar durch ein mathematisches Verfahren generiert werden. Dabei ist es nicht möglich, von einem Schlüssel auf den jeweils anderen zu schließen. Ein Verfahren, das hierbei zur Erzeugung der Schlüssel zum Einsatz kommt, ist beispielsweise das Rivest-Shamir-Adleman (RSA) Kryptosystem, welches von den drei Mathematikern Rivest, Shamir und Adleman am MIT im Jahr 1977 entwickelt wurde [ele22a]. Der öffentliche Schlüssel kann frei und öffentlich zugänglich verbreitet werden, im Gegensatz dazu gilt es den privaten Schlüssel bestmöglich zu schützen, da dessen Verlust die Sicherheit des Systems aushebeln würde. Die Besonderheit liegt nun in der Fähigkeit, Daten so zu verschlüsseln, dass nur der jeweils andere Schlüssel in der Lage ist diesen wieder zu entschlüsseln, dies wird auch als asymmetrische Verschlüsselung bezeichnet. Dadurch ist es zwei Kommunikationspartnern möglich, eine verschlüsselte Kommunikation über ein unsicheres Übertragungsmedium zu etablieren, ohne vorher ein gemeinsames Geheimnis austauschen zu müssen. Sendet ein System eine verschlüsselte Nachricht an ein anderes System und dieses ist in der Lage diese Nachricht zu entschlüsseln und zurück zu senden, ist dadurch bewiesen, dass das andere System im Besitz des privaten-Schlüssels ist, ohne diesen jemals preisgeben zu müssen. Dieser Mechanismus bildet die Grundlage zur Verifizierung von digitalen Identitäten. Es ist zu beachten, dass dabei nicht die eigentliche Identität des Kommunikationspartners überprüft wird, sondern nur ob dieser im Besitzt eines Geheimnisses ist, welches seine Aussage belegt. Wird dieses Geheimnis von einem System gestohlen, wird damit auch dessen Identität gestohlen und ein potentieller Angreifer ist nun der Lage, sich diese selber anzueignen. Aus diesem Grund sind digitale Zertifikate meist mit einem Kennwort geschützt und mit einem Ablaufdatum versehen, um den beim Verlust entstehenden Schaden zu begrenzen.

RSA-Schlüssel sind Teil der eigentlichen X.509-Zertifikate und werden verwendet um deren Inhalt kryptographisch zu belegen. Die bisher beschriebene Public-Key-Kryptographie kann bestätigen, dass jemand im Besitz des privaten Schlüssels ist, welcher die Aussagen im öffentlichen Zertifikat belegt. Jedoch ist dies noch kein Nachweis über die tatsächliche Identität des Kommunikationspartners, da dieser die Zertifikate in der Praxis selber generiert haben kann. An dieser Stelle kommt die Public-Key-Infrastructure (PKI) ins Spiel, welche es ermöglicht, die Authentizität eines öffentlichen Zertifikates zu verifizieren. Dies wird erreicht, indem eine vertraute dritte Partei die Angaben im öffentlichen Schlüssel eines Systems mit ihrem privaten Schlüssel verschlüsselt und das Zertifikat somit signiert. Diese Partei wird auch als Zertifizierungsstelle bezeichnet. Ein Client kann nun mithilfe des öffentlichen Schlüssels der Zertifizierungsstelle einen Teil des öffentlichen Zertifikates des fragwürdigen Systems entschlüsseln und damit nachweisen, dass dieses Zertifikat von der angegeben Zertifizierungsstelle signiert wurde. Da die Zertifizierungsstelle von allen Teilnehmern der Kommunikation als vertrauenswürdig angesehen wird, wird nun dieses Vertrauen auch auf die von ihr signierten Zertifikate und damit auf die Systeme, welche diese Zertifikate vorweisen können, impliziert. In der Praxis wird dieser Prozess oft

durch mehrere Zwischenzertifizierungsstellen verkettet, um zum einen mehr Flexibilität zu schaffen und zum anderen die Ursprungs-Zertifizierungsstelle zu entlasten und zu schützen. Alle Zero Trust Netzwerke benötigen eine PKI, um Identitäten im Netzwerk überprüfen zu können, diese bildet somit die Kernkomponente für alle Authentifizierungen. Aufgrund der hohen Anzahl an Zertifikaten in einem ZTN, ist es um so wichtiger, deren Ausstellung, Signierung und Verteilung zu automatisieren, um menschliche Fehler auszuschließen und so die Gesamtsicherheit des Systems zu erhöhen.

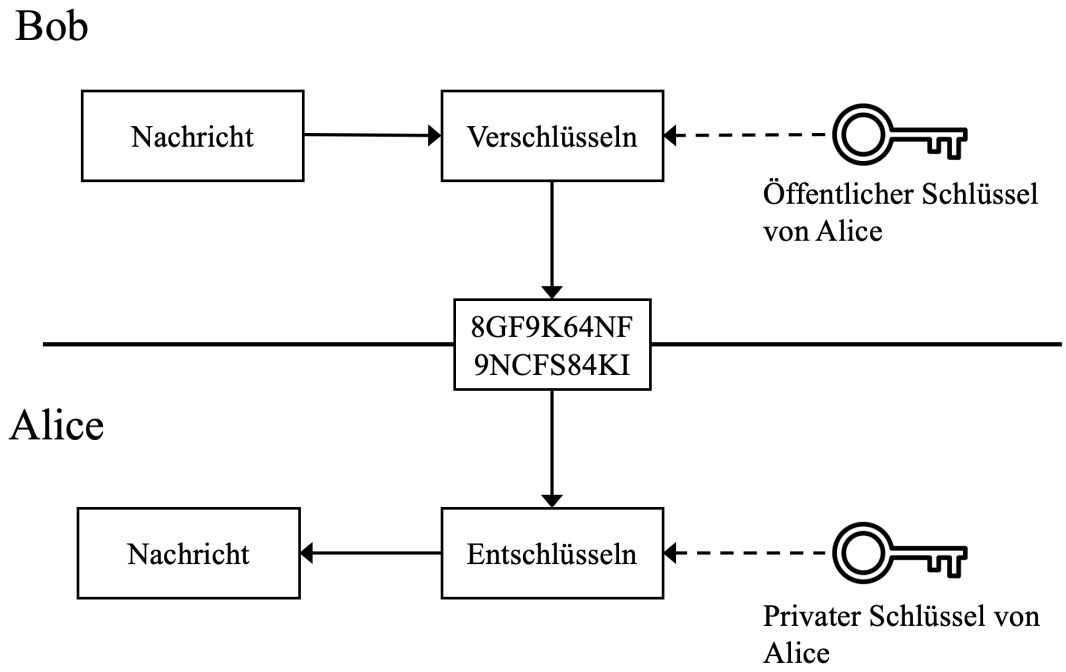

Abbildung 3.1: Verschlüsselte Kommunikation mit dem RSA-Verfahren

Die Grafik zeigt schematisch eine mit dem RSA-Verfahren verschlüsselte Datenübertragung zwischen zwei Parteien.

TLS

Konnte nun sichergestellt werden, dass der Kommunikationspartner auch jener ist, der er behautet zu sein, ist dennoch das Problem der vertraulichen und manipulationssicheren Kommunikation noch offen. Transport Layer Security (TLS) [E R18] ist ein Protokoll, das der Authentifizierung und Verschlüsselung von Netzwerkverbindungen dient. Dieses findet Anwendung als eigene Schicht zwischen TCP und dem eigentlichen Anwendungsprotokoll. TLS erfüllt zwei Hauptaufgaben, zum einen wird der Datenverkehr gesichert und gegen das Mitlesen oder die Manipulation durch Dritte geschützt [ele22b]. Zum anderen kann durch den Einsatz von X.509 Zertifikaten die Authentizität der Kommunikationspartner sichergestellt werden. Im öffentlichen Internet ist TLS meist so konfiguriert, dass dies nur in eine Richtung stattfindet, so überprüft zwar der Client die Authentizität des Servers, aber bleibt dabei selber dabei anonym. Technisch ist TLS jedoch in der Lage eine gegenseite Authentifizierung der Kommunikationspartner durchzuführen, dies wird als “Mutually Authenticated TLS“ (mTLS) bezeichnet und bildet die Grundlage zur

sicheren Kommunikation in einem ZTN. Hierbei wird der TLS-Verbindungsaufbau um die Übertragung und Verifizierung des Client-Zertifikates erweitert, dies wird in der Abbildung 3.2 durch die hinzugefügten Schritte 4 und 5 dargestellt.

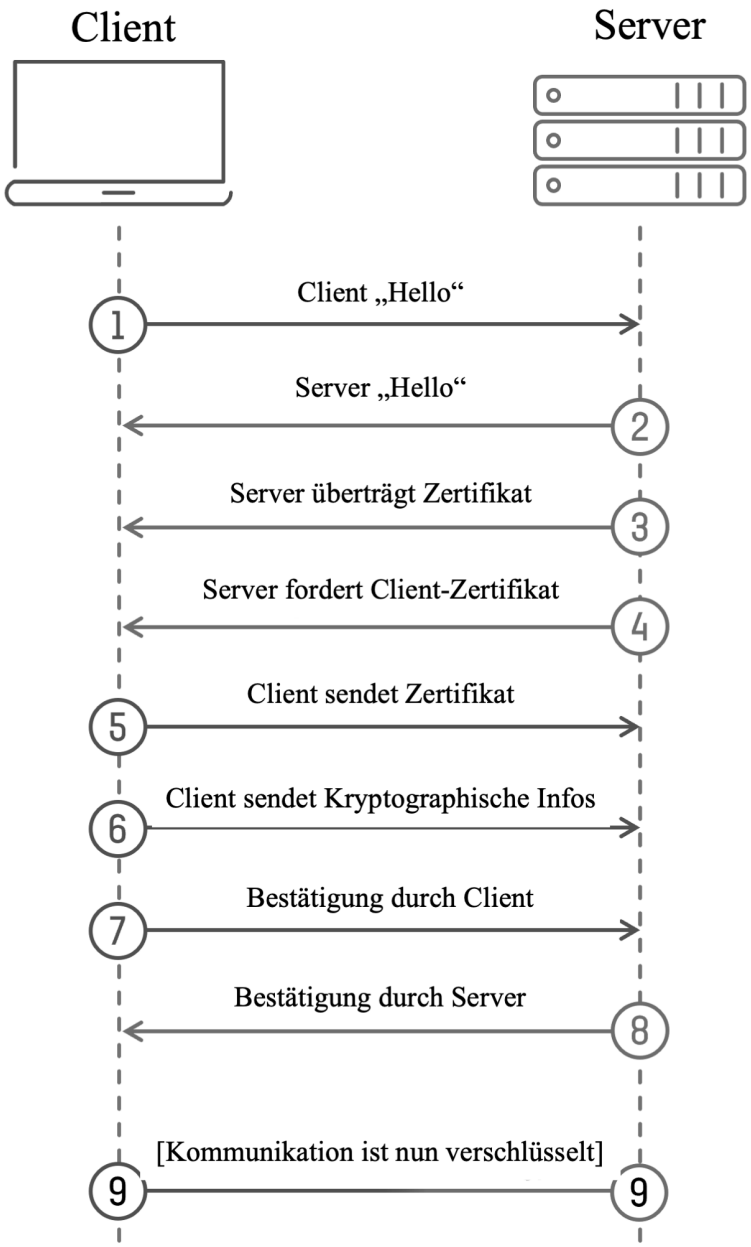

Abbildung 3.2: Erweiterter TLS 1.2 Handshake für die gegenseite Authentifizierung

Die Grafik zeigt TLS Handshake in der Version 1.2, welcher um die Schritte 4 und 5 erweitert wurde um eine gegenseite Authentifizierung der Kommunikationspartner durchzuführen.

Während des TLS-Verbindungsaufbaus, wird zwischen Server und Client ausgehandelt, welche Verschlüsselungsverfahren zum Einsatz kommen sollen. Ist dies geschehen, wird ein statischer Schlüssel ausgehandelt, welcher nun zur Ver- und Entschlüsselung der eigentlichen Nutzdaten verwendet wird. Ein elementares Verfahren zur Aushandlung des statischen Schlüssels, ist der Diffie-Hellman-Schlüsselaustausch [E R99]. Dieses Verfahren erlaubt es, ein gemeinsames Geheimnis über einen unsicheren Kommunikationskanal auszuhandeln. Im Anschluss kann mithilfe des statischen Schlüssels, welcher nun beiden Parteien bekannt ist, auf einfachere und effizientere symmetrische Verschlüsselungsverfahren, wie beispielsweise dem Advanced Encryption Standard (AES) [Blu04] zurückgegriffen werden. Durch den Einsatz von TLS wird eine vertrauliche, authentifizierte und manipulationssichere Kommunikation zwischen Systemen über ein unsicheres Übertragungsmedium gewährleistet und somit Vertrauen geschaffen [GB17].

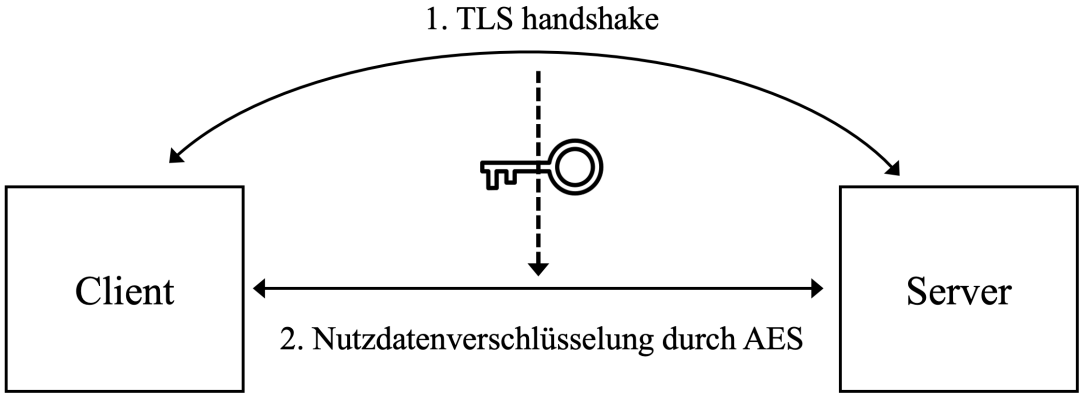

Abbildung 3.3: Wechsel auf ein symmetrisches Verschlüsselungsverfahren nach dem TLS Handshake

Die Grafik zeigt schematisch den Wechsel auf ein symmetrisches Verschlüsselungsverfahren nach dem TLS Handshake

Der Mindeststandard des BSI zur Verwendung von Transport Layer Security fordert zum Zeitpunkt der Arbeit, ausschließlich die Versionen 1.2 und 1.3 zu verwenden, ältere Versionen sind aus Sicherheitsgründen zu deaktivieren [bsi21].

3.1.4 Trennung zwischen Steuer- und Nutzdaten

Es ist gängige Praxis bei der Administration von Netzwerken zwischen Steuer- und Nutzdaten zu unterscheiden. Die Grundidee dabei ist, dass jedes Netzwerkgerät den Datenverkehr in zwei klar voneinander getrennte Schichten unterteilt. Zum einen gibt es die Schicht der Nutzdaten, welche den Großteil des Datenverkehrs im Netzwerk ausmacht. Da diese Schicht hohe Datenraten bereitstellen soll, ist deren Logik relativ einfach gehalten und wir oft durch spezialisierte Hardware unterstützt. Die Steuerschicht hingegen ist für die Verwaltung des Netzwerkes selber zuständig und kann daher eine höhere Komplexität aufweisen. Da diese nicht in der Lage ist, große Datenmengen zu verarbeiten, ist es wichtig

die Schnittstelle zwischen den Schichten so zu gestalten, dass fast alle benötigten Mechanismen zur Steuerung des Datenverkehrs auf der Nutzdaten-Schicht implementiert werden. So kann sichergestellt werden, dass nur sehr wenige Operation auf die Steuerschicht ausgelagert werden müssen. Das Konzept der Trennung zwischen Steuer- und Nutzdaten bezog sich Ursprünglich auch auf die physische Trennung der Netzwerke. Auch wenn keine physische Trennung der Netzwerke umgesetzt wird, bildet dieses Konzept eine Grundlage zur logischen Organisation der Netzwerkbestandteile.

Ein Zero Trust Netzwerk unterscheidet zwischen diesen Schichten. Die Sicherheitseinrichtungen in einem solchen Netzwerk müssen in der Lage sein, schnell entscheiden zu können, ob der Netzwerkverkehr zugelassen oder blockiert werden soll. Es muss verhindert werden, dass ein Angreifer welcher sich zunächst auf der Nutzdatenschicht befindet, die Kontrolle über die Steuerschicht übernehmen kann, um sich so im Netzwerk ausbreiten zu können. Jegliche Kommunikation auf der Steuerschicht muss durch Verschlüsselung und Authentifizierung vor Manipulationen geschützt sein. An dieser Stelle lässt sich die Isolierung mit der Trennung zwischen Benutzer- und Kernel-Bereichen in einem modernen Betriebssystem vergleichen [GB17].

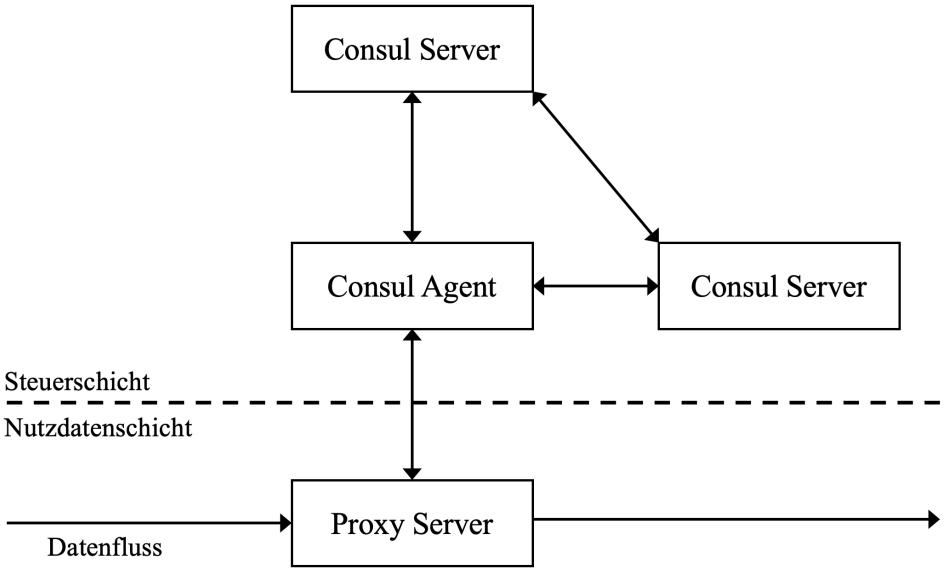

Abbildung 3.4: Trennung zwischen Steuer- und Nutzdatenschicht

Die Abbildung zeigt die Trennung zwischen Steuer- und Nutzdatenschicht in einem Zero Trust Netzwerk. Aufwändige Operationen befinden sich auf der Steuerschicht, die Nutzdatenschicht hingegen ist so einfach wie möglich gestaltet um eine hohe Effizienz bei der Datenverarbeitung zu erreichen

3.2 Das Service Mesh

Das folgende Kapitel beschreibt das Konzept eines Services Mesh und wie dieses zur Umsetzung einer Zero Trust Network Architecture, kurz ZTNA, verwendet werden kann. Ein Service Mesh ist eine Infrastrukturschicht, welche es einzelnen Services ermöglicht

effizient und sicher untereinander kommunizieren zu können.

3.2.1 Einsatzbereich

Das Konzept des Service Mesh entstand dabei im Zusammenhang mit dem Aufkommen von Microservices und wurde ursprünglich dazu konzipiert, um einigen Nachteilen dieser Architektur entgegenzuwirken [Mar20]. Wie im Kapitel 2.1.2 über Mikroservices bereits beschrieben wurde, teilen diese die Anwendungslogik in mehrere kleine Anwendungsteile auf, welche über das Netzwerk miteinander kommunizieren und zusammenarbeiten. Dabei muss ein Service jederzeit wissen, wo dieser einen anderen benötigten Service im Netzwerk erreichen kann. Es hat viele Nachteile dieses Problem durch statische Adressierungen und Konfigurationen zu lösen, da so die ursprünglich durch das Service Mesh geschaffene Flexibilität aufgehoben wird. Soll beispielsweise ein Mikroservices aus Last- oder Redundanz-Gründen in mehr als eine Instanz bereitgestellt werden, erfordert dies zur Adressierung den Einsatz eines Load-Balancers vor diesen Instanzen. Bei vielen verschiedenen Services führt dies zu mehr benötigter Infrastruktur und mehr Hops, welches ein Datenpaket zwischen den Services zurücklegen muss. Da der Einsatz eines einzelnen Loadbalancers für einen bestimmen Service einen Single-Point-of-Failure schafft, werden auch hier wiederum mindestens zwei benötigt, um eine Redundanz sicherzustellen zu können. Ein Service Mesh extrahiert die Logik für die Inter-Servicekommunikation aus den einzelnen Services und überträgt sie in eine Infrastrukturschicht. Durch ein Service Mesh ist eine Implementierung der Logik für die Kommunikation auf Anwendungsseite nicht länger notwendig, der Entwicklerfokus kann somit voll auf die geschäftlichen Ziele gerichtet werden [GB17].

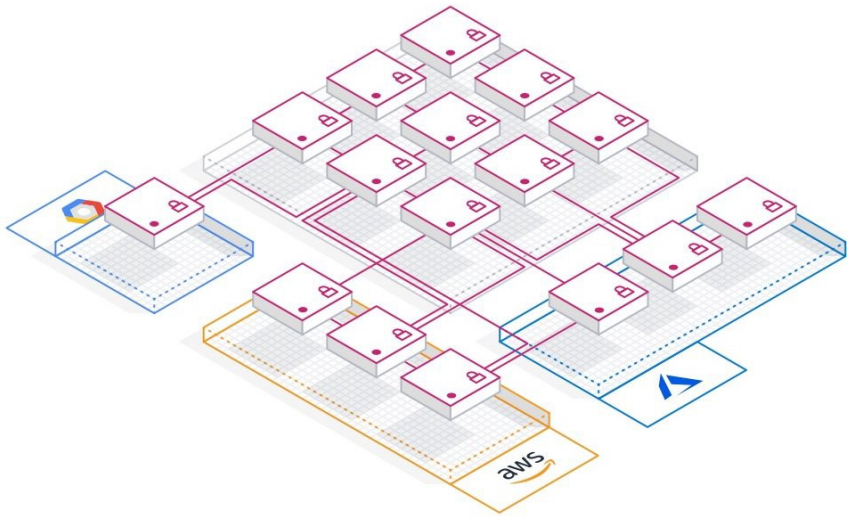

Abbildung 3.5: Schematische Darstellung eines Standortübergreifenden Service Mesh - [con22]

Die Grafik zeigt ein abstrakte Darstellung der von Serviceverbindungen welche durch ein Service Mesh gesteuert werden. Die Verbindungen sind dabei unabhängig vom physischen Standort des Services.

Die Hauptaufgaben eines Service Mesh lassen sich in folgende drei Bereiche unterteilen:

- **Erkennung von Diensten**
  Das Service Mesh agiert als Vermittlungsstelle zwischen den einzelnen Services um deren Zusammenarbeit zu ermöglichen. Dazu zählt die Registrierung, die Auskunft über die Adressierung von Services und deren Verbindung. Es ist in der Lage für eine gleichmäßige Verteilung ankommender Anfragen auf mehrere Instanzen des selben Service zu sorgen und damit eine Lastverteilung zu erreichen. Auch kann auf eine schwankende Anzahl verfügbarer Instanzen eines Service flexibel reagiert werden oder fehlerhafte Instanzen von neuen Verbindungen ausgeschlossen werden. Flexible Regelwerke ermöglichen dem Administrator den Datenstrom beliebig zu lenken. So können Instanzen einer neueren Version eines vorhandenen Services vorerst nur Anfragen ausgewählter Anwender erhalten und so in der Produktivumgebung getestet werden, ohne Auswirkungen auf andere Nutzer zu haben [Hub20].

- **Beschränkung des Netzwerkverkehrs**
  Dem Service Mesh stehen genaue Informationen über alle Services und deren Abhängigkeiten untereinander zur Verfügung. Dadurch ist dieses nicht nur in der Lage, Verbindungen zwischen Services zu unterstützen, sondern auch ungewollte Verbindungsversuche zu unterdrücken und zu melden [Hub20].

- **Dienst Konfiguration**
  Neben Funktionalität auf der Netzwerkschicht bietet das Service Mesh auch Unterstützung auf der Anwendungsebene. Dazu zählt meist der Betrieb einer X.509 Zertifizierungsstelle, sowie die automatische Ausstellung, Verteilung und Erneuerung dieser Zertifikate an die entsprechenden Dienste. Ein weiterer Mechanismus wie beispielsweise ein “Schlüssel-Wert-Speicher“ wird durch das Mesh bereitgestellt und steht den Anwendungsentwicklern für ihre Zwecke zur Verfügung [Hub20].

3.2.2 Realisierung einer ZTNA durch ein Service Mesh

Ein Service Mesh lässt sich so konfigurieren, dass dieses alle Anforderungen einer ZTNA erfüllt. Die Netzwerkrichtlinien werden so eingerichtet, dass standardmäßig alle Verbindungen unterdrückt werden. Durch die Konfiguration eines Service-Graphen, können legitime Verbindungen durch den Administrator konfiguriert werden. Das Netzwerk wird auf der Ebene einzelner Systeme segmentiert, dies kann durch den Einsatz der im Betriebssystem vorhandenen Firewall erreicht werden. Die Firewall blockiert alle eingehenden Verbindungen, welche nicht mittels mTLS verschlüsselt und signiert wurden. Ausgehende Verbindungen werden in keiner Weise beschränkt. Die Verifizierung, ob eine Verbindung legitim ist, wird dabei ausschließlich vom Zielsystem durchgeführt. Dadurch ist keine zusätzliche Infrastruktur auf Netzwerkebene notwendig und die Auswirkung auf die Netzwerkleistung wird minimiert.

Robuste Sicherheit
Eine Beschränkung und Überwachung des ausgehenden Netzwerkverkehrs, welche eigenverantwortlich von den Systemen durchgeführt wird, ist nicht zielführend, da jedes System als potentiell kompromittiert anzusehen ist. Übernimmt ein Angreifer beispielsweise durch eine Sicherheitslücke in einer Anwendung die vollständige Kontrolle über ein System, ist dieser in der Lage die lokalen Firewallregeln und sonstige Sicherheitseinrichtungen außer Kraft zu setzen. Daher ist das interne Netzwerk stets als unsicher zu betrachten. Jedes System ist für seinen Schutz eigenverantwortlich. Das Service Mesh bietet Systemen, welche nicht Teil des Meshs sind, keinerlei Schutz vor Angriffen welche aus dessen inneren heraus ausgehen. Die Sicherheit liegt darin, dass sich alle anderen Systeme vom Netzwerk abschotten müssen und nur durch mTLS verschlüsselte und signierte Verbindungen zulassen. Lateral Movement ist daher nur in den Grenzen der Berechtigungen des Systems möglich, da dem System nicht die nötigen Zertifikate zur Verbindung mit anderen Diensten zur Verfügung stehen. Werden auf einem System die Dienste des Service Mesh durch den Angreifer abgeschaltet, isoliert sich dieser dadurch im Mesh. Andere Systeme, welche Ziel unbefugter Verbindungsversuche werden, können diese an eine zentrale Instanz melden. Als Folge dessen kann das Kompromittierte System schnell ausfindig gemacht und abgestoßen werden.

3.2.3 Auswahl einer geeigneten Service Mesh Implementierung

Viele der gängigen Implementierungen sind für die Unterstützung von Mikroservices in der Container-Orchestrierungs Software Kubernetes [kub22] Umgebungen optimiert, oft besteht eine Abhängigkeit zu dessen Diensten und Features. Gesucht wird eine Software, welche sowohl in Kubernetes beziehungsweise Container-Umgebungen, als auch auf gewöhnlichen (virtuellen) Servern eingesetzt werden kann. Durch Recherchen konnte folgende Liste gängiger Service Mesh Lösungen erstellt werden, welche in die engere Auswahl kommen:

- Linkerd[lin22]
- Consul[con22]
- Grey Matter[gre22]
- Istio[ist22]
- Kuma[kum22]

Drei sehr populäre Systeme (GitHub-Bewertung, Google Trends, Erwähnungen etc.) sind hierbei Linkered, Consul und Istio. Alle drei bieten die Möglichkeit eines vollumfänglichen Service Mesh und Verschlüsselung des Datenverkehrs. Durch eine erste Einarbeitung und weitere Recherchen zu den Implementierungen, konnte die Auswahl weiter eingeschränkt werden. Ein detaillierter Vergleich kann folgender Tabelle entnommen werden:

Tabelle 3.1: Vergleich zwischen Istio, Linkered und Consul [Man22; Has22b]

	Istio	Linkerd	Consul
Unterstützte Einsatzgebiete	Wo kann das Service Mesh eingesetzt werden ?		
Einsatzgebiete	Kubernetes + VMs	Kubernetes only	Kubernetes + VMs
Architektur	Wie ist die Implementierung aufgebaut ?		
Single point of failure	Nein	Nein	Nein
Proxy Server	Ja (Envoy)	Ja	Ja (Envoy)
Agent auf jedem System	Nein	Nein	Ja
Externe Abhängigkeiten	Externer Service-Katalog	Kubernetes	Keine
Support	Wie verbreitet ist die Lösung ?		
Open-Source	Ja	Ja	Ja
Lizenz	Apache-2.0	Apache-2.0	MPL-2.0
Beliebtheit (GitHub-Sterne)	30.8k	8.6k	25.0k
Dokumentation	Hervorragend	Hervorragend	Hervorragend
Verschlüsselung	Welche Verschlüsselungsdienste werden geboten ?		
mTLS Tunnel	Ja	Ja	Ja
Zertifikatsmanagement	Ja	Ja	Ja
Authentifizierung/Autorisierung	Ja	Ja	Ja

Alle drei Lösungen bieten einen ähnlichen Funktionsumfang und Aufbau. Lediglich bei den externen Abhängigkeiten lassen sich klare Unterschiede ausmachen. So kann Linkered nur in einer Kubernetes Umgebung betrieben werden und kommt daher für diese Arbeit nicht in Frage. Istio kann sowohl in Kubernetes als auch auf herkömmlichen Servern betrieben werden, jedoch wird ein externer Service-Katalog benötigt, welcher zum Beispiel durch Kubernetes, Consul, Eureka etc. bereitgestellt werden muss. Die Lösung, welche im Rahmen dieser Arbeit erprobt werden wird, soll ein möglich breites Spektrum bestehender IT-Infrastrukturen abdecken. Daher wurde sich für die Service Mesh Implementierung Consul der Firma HashiCorp entschieden, da diese keine externen Abhängigkeiten aufweist und im vollständigen Stand-alone-Betrieb eingesetzt werden kann. Consul hat einen hohen Verbreitungsgrad und weist eine ausführliche und sorgfältige Dokumentation auf [Has22a].

3.3 HashiCorp Consul

Consul ist eine kostenlose Open-Source-Service-Networking-Plattform, welche federführend von der amerikanischen Firma HashiCorp entwickelt wurde. HashiCorp wurde im Jahr 2012 gegründet, hat seinen Hauptsitz in San Francisco und beschäftigt aktuell etwa 1850 Mitarbeiter [has22]. Kerngeschäft ist die Entwicklung und Vermarktung von Infrastrukturmanagement-Produkten, welche alle Bereiche eines modernen Rechenzentrums abdecken.

Die erste Version von Consul wurde im Jahr 2014 veröffentlicht, zu Beginn diente Consul lediglich als eine zentrale Registrierung für Services. Mittlerweile erfüllt die Software alle Funktionen eines vollwertigen Service Mesh und kommt sowohl in der Container-Orchestrierung, Cloud-Umgebungen als auch in On-Premise Lösungen zum Einsatz. Programmiert ist Consul in der von Google entwickelten Programmiersprache Golang [god22] und ist auf allen gängigen Betriebssystemen und Rechnerarchitekturen lauffähig. Die Software steht in ihrer Basisversion unter der "Mozilla Public License 2.0", welche den kostenlosen Einsatz für private und kommerzielle Zwecke, Veränderungen sowie die Verbreitung gestattet. Neben der Open-Source Version steht auch eine kostenpflichtige Enterprise Version mit einem leicht erweiterten Funktionsumfang Verfügung. Consul besteht aus einer einzelnen ausführbaren Datei und erfordert keinerlei Installation innerhalb des Betriebssystems oder sonstige Abhängigkeiten. Die Konfiguration erfolgt durch die an XML angelehnte Auszeichnungssprache YAML in einer oder mehreren Konfigurationsdateien. Abhängig von der Konfiguration kann Consul als Server oder Agent gestartet werden. Da die Software Teil der HashiCorp-Suite von Infrastrukturmanagement-Produkten ist, bringt diese eine breite Tool-Unterstützung und Integration in andere Systeme mit. So kann zum Beispiel die Infrastructure as Code (IaC)- Plattform HashiCorp-Terraform zur Administration von Consul verwendet werden.

3.3.1 Struktur und Funktionsweise

Consul besteht aus drei Einheiten:

- **Der Consul Server (Steuerschicht)**
  Die zentrale Steuereinheit eines unter Consul betrieben Service Mesh wird durch ein oder mehrere Consul Server dargestellt. Es wird der Einsatz von drei beziehungsweise fünf Servern empfohlen, um Redundanz und somit die Ausfallsicherheit sicherzustellen. Durch ein Wahlverfahren wird ein Server ausgewählt, welcher das Service Mesh führt. Dieser übernimmt zentrale Aufgaben, wie die Authentifizierung und Autorisierung der Agents, sowie den Betrieb der Public-Key Infrastruktur [Has22d].

- **Der Consul Agent (Steuerschicht)**
  Auf jedem Knoten, der Teil des Service Mesh ist, befindet sich ein als Daemon lokal laufender Consul Agent. Der Consul Agent für lokale Konfiguration am Proxy Server, die Bereitstellung eines DNS-Dienstes und das Monitoring der Services zuständig ist. Der Agent kommuniziert sowohl mit einem der Consul Server als auch untereinander mit anderen Agents, so können Änderungen an der Struktur des Service Mesh oder eine veränderte Verfügbarkeit eines Service schnell und effizient im Mesh bekannt gemacht werden. Durch diesen dezentralen Aufbau wird der Consul Server entlastet und die Fehleranfälligkeit verringert [Has22d].

- **Der Envoy Proxy Server (Nutzdatenschicht)**
  Die Consul Dienste sind ausschließlich für die Verwaltung und Überwachung des Service Mesh zuständig und stellen somit die Steuerschicht des Netzwerkes dar. Der Transport der eigentlichen Nutzdaten erfolgt über die Nutzdatenschicht und wird durch Proxy Server übernommen. Diese werden dazu dynamisch von den Consul Agent konfiguriert und überwacht. Jeder Dienst, welcher Teilnehmer des Service Mesh ist, betreibt seinen eigenen lokalen Proxy Server, welcher parallel zur eigentlichen Anwendung ausgeführt wird. Dieser wird daher auch häufig als “Sidecar Proxy“ bezeichnet [Has22d]. Der lokale Consul Agent verfügt über einen Zwischenspeicher und regelt den Großteil der Konfiguration der Nutzdatenschicht eigenständig. So kann die Zahl der Anfragen an einen der zentralen Server gering gehalten werden und die Funktion der Nutzdatenschicht bei einem Ausfall der Steuerschicht vorübergehend aufrecht erhalten werden.

[Hub20; Has22c]

3.3.2 Nahtlose Anwendungsintegration durch lokalen DNS Server

Als Schnittstelle zwischen den Anwendungen und dem Service Mesh wird das DNS-Protokoll verwendet. Dazu stellt jeder Consul Agent einen lokalen DNS-Server zur Verfügung, welcher es den lokalen Anwendungen erlaubt andere Services im Netzwerk ausfindig zu machen. Dabei beruft sich der DNS-Server auf die interne Servicedatenbank. Der Agent kann die Rückgabewerte der DNS Anfragen so wählen, dass Verbindungen zwischen den Services durch Richtlinien beeinflusst und gelenkt werden. So kann beispielsweise auf eine sich ändernde Anzahl von Instanzen eines Services reagiert werden oder Funktionalitäten eines Load-Balancers simuliert werden.
Consul verfolgt folgendes Schema bei der Namensauflösung von Services:

```
[tag.]<service>.service[.datacenter].<domain>
```

Wird nun beispielsweise eine Instanz eines Services Names “redis“ benötigt, kann diese durch folgende Abfrage ermittelt werden:

```
1  $ dig @127.0.0.1 -p 8600 redis.service.consul SRV
2
3  ; <<>> DiG 9.8.3-P1 <<>> @127.0.0.1 -p 8600 redis.service.consul ANY
4  ; (1 server found)
5  ;; global options: +cmd
6  ;; Got answer:
7  ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50483
8  ;; flags: qr aa rd; QUERY: 1, ANSWER: 3, AUTHORITY: 1, ADDITIONAL: 1
9  ;; WARNING: recursion requested but not available
10
11 ;; QUESTION SECTION:
12 ;redis.service.consul.        IN  SRV
13
14 ;; ANSWER SECTION:
15 redis.service.consul.     0   IN  SRV 1 1 8300 server4.node.dc1.consul.
16
17 ;; ADDITIONAL SECTION:
18 server24.node.dc1.consul. 0   IN  A   10.1.10.12
```

Aus der Ausgabe lässt sich in Zeile 15 ablesen, dass eine Instanz des Services Namens **redis.service.consul.** auf dem Host **server4.node.dc1.consul.** zur Verfügung steht.

3.3.3 Envoy Sidecar Proxys

Um Einfluss auf die Nutzdatenschicht gewinnen zu können, muss der Datenverkehr zusätzliche Kontroll- und Steuerinstanzen passieren. Wie auch viele andere Service Mesh Umgebungen, setzt Consul den weit verbreiteten Envoy Proxy auf der Nutzdatenschicht ein [env22]. Envoy ist ein Open-Source Edge und Service Proxy, welcher von der Cloud Native Computing Foundation (CNCF) betreut wird und speziell für Cloud-Native Anwendungen entwickelt wurde [cnc22]. Durch die entspreche Konfiguration von Envoy lässt sich nicht nur der Datenverkehr steuern und überwachen, sondern auch die gegenseite Authentifizierung und Verschlüsselung des Datenverkehrs durch den Einsatz eines mTLS-Tunnels erzwingen. Die Kommunikation des Services erfolgt hierbei ausschließlich mit dem lokalen Proxy, dies kann auf zwei verschiedene Arten erreicht werden. Im ersten Fall bindet der Proxy lokale Ports auf der Maschine, welche dann in den entsprechen Anwendungen als Kommunikationspartner hinterlegt werden. Alternativ kann der Proxy in einem transparenten Modus arbeiten. Dazu werden durch lokale Firewallregeln alle ausgehenden Verbindung auf den lokalen Proxy umgeleitet und von diesem weiter verarbeitet. In beiden Fällen handelt dieser anschließend mit der Gegenstelle die Verschlüsselung und die gegenseitige Authentifizierung aus. Danach werden die Nutzdaten über den geschaffenen mTLS-Tunnel zwischen den Services transportiert. Beide Ansätze werden im weiteren Verlauf dieser Arbeit praktisch vorgestellt und deren Vor- und Nachteile analysiert.

3.3.4 Verbindungsaufbau im Service Mesh

Am Verbindungsaufbau zu einem Service auf einem anderen System sind insgesamt fünf Softwarekomponenten beteiligt: zum einen der lokale Consul Agent und Proxy Server, zum anderen der Consul Agent und Proxy Server des Zielsystems. Wird die Verbindung erstmalig hergestellt und werden weitere Informationen benötigt, erfolgt eine Kontaktierung des zentralen Consul Servers. Der Verbindungsaufbau besteht dabei auf folgenden Schritten:

1. **Lokaler Proxy Server** → **Lokaler Consul Agent**
   Der lokale Proxy Server kontaktiert den lokalen Consul Agent um zu erfragen, unter welcher Netzwerkadresse der andere Service erreichbar ist. Sollte sich diese Information nicht im Cache des des Agents befinden oder müssen andere Regelwerke ausgeführt werden, kontaktiert dieser den zentralen Consul Server um diese Informationen zu erfragen.

2. **Lokaler Proxy Server** → **Remote Proxy Server**
   Der lokale Proxy Server stellt nun eine TLS-Verbindung zum Proxy Server des Remote System her. Dabei findet, wie bereits in Kapitel 2 beschrieben, ein Zertifikatsaustausch und Validierung statt, wodurch die Systeme ihre gegenseitige Identität verifizieren.

3. **Remote Proxy Server** → **Remote Consul Agent**
   Der Remote Proxy Server kontaktiert nun seinen lokalen Consul Agent um zu überprüfen, ob die Verbindung laut den Netzwerkrichtlinien zulässig ist. Sollte sich diese Information nicht im Cache des Agents befinden, wird auch hier der zentrale Consul Server befragt.

4. **Lokaler Proxy Server** → **Remote Consul Agent**
   Ist die Verbindung zulässig, ist eine Kommunikation der Services über den durch die Proxy Server eingerichteten mTLS Tunnel möglich.

4 Versuchsaufbau

Nachdem im vorangegangenen Kapitel das allgemeine Konzept eines Zero Trust Netzwerks erläutert wurde, folgt nun eine praxisnahe Anwendung der bisher vorgestellten Techniken und Konzepte.

4.1 Ziele

Ziel ist der Aufbau einer Serverumgebung zur Bereitstellung diverser, voneinander abhängiger Dienste. Die Kommunikation untereinander soll dabei durch den Einsatz eines Service Meshs koordiniert und gesichert werden. Die Server werden durch entsprechende Firewallrichtlinien vom gemeinsamen Netzwerk abgeschottet. Das Service Mesh unterbindet dabei alle Verbindungen, welche nicht explizit zugelassen sind. So wird eine praktische Umsetzung der Zero Trust Network Architecture erreicht. Neben den Erfahrungen, die bei der Installation und Konfiguration des Service Mesh gesammelt werden können, wird im Nachgang eine ausführliche Analyse der geschaffenen Infrastruktur durchgeführt. Dabei werden unter anderem die Auswirkung des Services Meshs auf Aspekte der Performance, Sicherheit und Wartbarkeit der Serverumgebung betrachtet.

Die Dienste und Anwendungen, die im Netzwerk zur Verfügung gestellt werden, sollen eine möglichst realistisches und breit aufgestelltes Anforderungsprofil an ein Service Mesh stellen. Einschränkung ist, dass nur Anwendungen verwendet werden, die frei zugänglich unter einer Open Source Lizenz bezogen werden können und auf UNIX Betriebssystemen lauffähig sind. Aus der nachfolgenden Tabelle 4.1 lässt sich entnehmen, welche Dienste angebotenen werden, welche Software dafür eingesetzt wird und über welche Protokolle und Ports diese erreichbar sind. Alle Services werden anhand einer eindeutigen Service-ID identifiziert, diese folgt dabei folgender Namenskonvention für Consul Services:

```
[tag.]<service>.service[.datacenter].<domain>
```

Tabelle 4.1: Angebotene Dienste im Service Mesh

Dienst	Software	Protokoll	Port	Service-ID
Web Dashboard das Daten von einem Backend Service erhält	Consul Test Counting Dashboard	TCP/ HTTP	80	dashboard.service.koecher.consul
Backend HTTP API	Consul Test Counting Service	TCP/ HTTP	80	counting.service.koecher.consul counting.service.merkur.consul counting.venus.consul.consul
Webhost mit Backend Datenbankanbindung	WordPress	TCP/ HTTP	80	wordpress.service.koecher.consul
Datenbank Instanzen im lokalen Rechenzentrum	MariaDB	TCP/ MariaDB	3306	database1.service.koecher.consul
Cloud Filestorage mit Backend Datenbankanbindung	Nextcloud	TCP/ HTTPS	443	nextcloud.service.koecher.consul
Datenbank Instanz im remote Rechenzentrum	MariaDB	TCP/ MariaDB	3306	database1.service.koecher.consul database2.service.venus.consul
Dateiserver (SMB) - Fileserver	SMB	TCP/ SMB	445	samba.service.koecher.consul.

4.2 Systemstruktur

Zum Aufbau der Testumgebung stehen insgesamt sieben virtuelle Maschinen zur Verfügung, von denen zwei von einem Cloud-Anbieter bereitgestellt werden. Die anderen fünf Maschinen werden im lokalen Rechenzentrum zur Verfügung gestellt. Diese Maschinen befinden sich in einem privaten Netzwerk, das durch eine Firewall vom Internet abgetrennt ist. Drei der Maschinen sind für die eigentliche Anwendungsbereitstellung verantwortlich. Von den übrigen zwei lokalen Maschinen dient eine als zentraler Server für Consul. Die anderen verwendet ein Desktop Betriebssystem und soll zum Testen und Analysieren des Servicer Meshs verwendet werden.

Als Hypervisor kommt hierzu "VMware Fusion Player" in der Version 12.1.2 zum Einsatz [vmw22]. Als Betriebssystem für die virtuellen Maschinen wurde sich für Debian GNU/Linux 11 mit dem Linux-Kernel in der Version 5.10.0-13-amd64 entschieden [deb22a]. Sowohl die Anwendungen als auch die für die ZTNA benötigte Infrastruktur wird auf den Servern mithilfe von Containern betrieben, dafür kommt Docker in der Version 20.10.14 zum Einsatz [doc22a]. Eine detailliertere Aufstellung der eingesetzten Server kann folgender Tabelle 4.2 und der Abbildung 4.1 entnommen werden:

Tabelle 4.2: Struktur des Versuchsaufbaus

Name	Verwendungszweck	OS	Ort	Server-ID
Server	Servicebereitstellung	Debian 11	merkur	server.node.merkur.consul
Server	Servicebereitstellung	Debian 11	venus	server.node.venus.consul
Desktop	Desktop Rechner zu Testzwecken	Linux Mint 20	lokal	desktop.node.koecher.consul
Server	Zentraler Consul Server	Debian 11	lokal	server.node.koecher.consul
Client 1	Servicebereitstellung	Debian 11	lokal	client1.node.koecher.consul
Client 2	Servicebereitstellung	Debian 11	lokal	client2.node.koecher.consul
Client 3	Servicebereitstellung	Debian 11	lokal	client3.node.koecher.consul

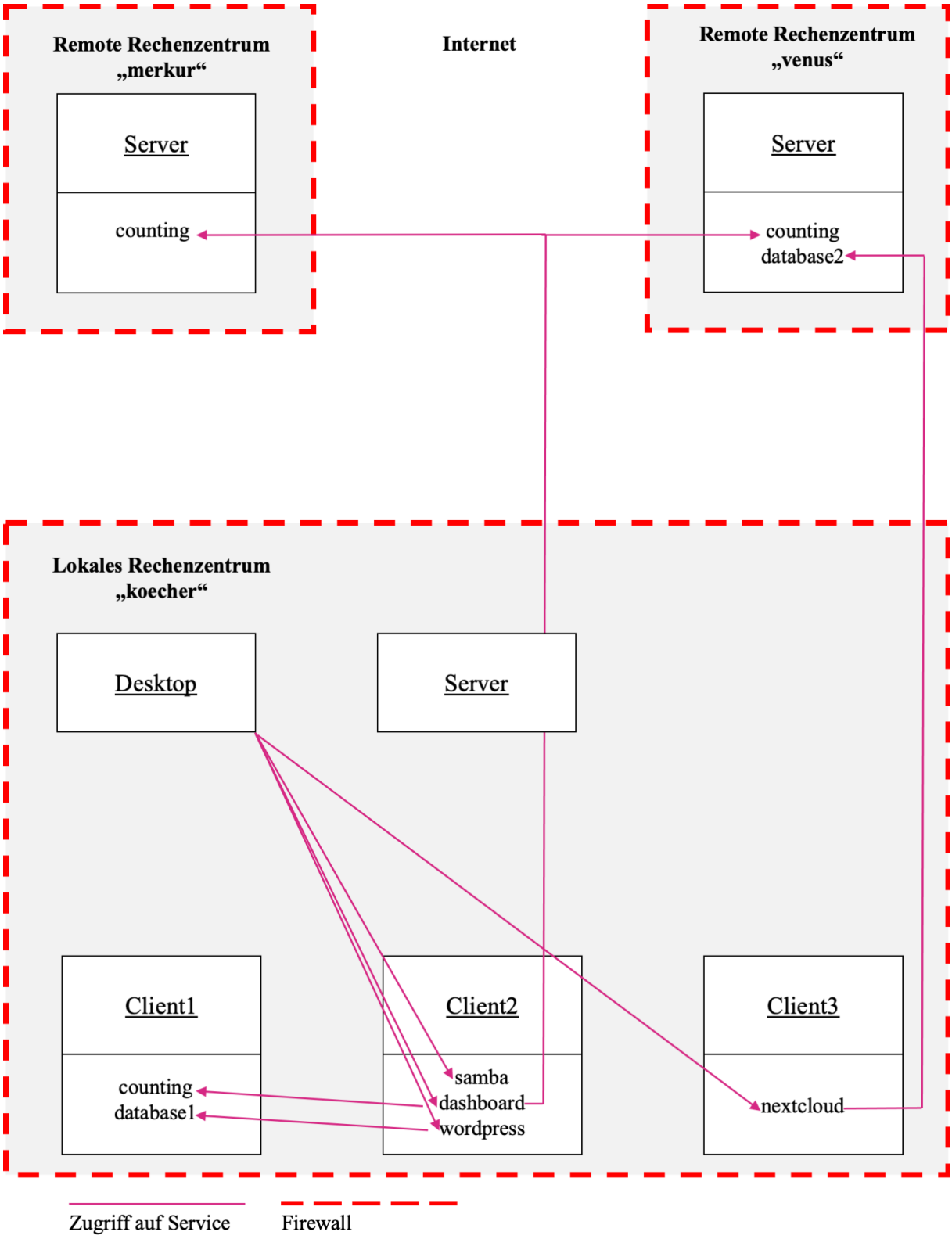

Abbildung 4.1: Servicebeziehungen des Versuchsaufbaus

Die Abbildung zeigt das Systemlayout des Versuchsaufbaus und dessen Servicebeziehungen. Zu sehen sind alle Systeme des Service Mesh und deren Aufteilung auf die drei Standorte. Es ist zu erkennen, welcher Service durch welches System bereitgestellt wird. Durch Pfeile sind die Kommunikations-Beziehungen zwischen den verscheiden Services dargestellt.

4.3 Installation der Dienste

Dieses Kapitel beschreibt, wie die in Kapitel 4.2 definierte Testumgebung realisiert wird.

4.3.1 Installation der virtuellen Maschinen

Die Installation von Debian erfolgt auf virtueller Hardware, bestehend aus jeweils 2 vCPUs, 2GB RAM und 500GB SSD Massenspeicher. Das Netzwerk der VMs wird im "Bridged-Modus"betrieben, so erscheinen diese als eigenständiger Host unter eigener IP im lokalen Netzwerk. Nach der erfolgreichen Installation von Debian erfolgt die Ersteinrichtung der Maschine. Dazu zählt die Konfiguration des Hostnamen und die Installation eigener SSH-Zertifikate zum Zugriff auf die Maschine. Die weitere Administration erfolgt nun ausschließlich über den geschaffenen SSH-Zugang. Als Vorbereitung zur Installation von Docker wird der Debian Paketmanager-APT aktualisiert und benötigte Abhängigkeiten installiert.

```
$ sudo apt-get update

$ sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

Im nächsten Schritt werden die Docker Paketquellen und die dafür benötigten Signaturen installiert:

```
$ sudo mkdir -p /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o
↪  /etc/apt/keyrings/docker.gpg

$ echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
↪  https://download.docker.com/linux/debian $(lsb_release -cs) stable" | sudo tee
↪  /etc/apt/sources.list.d/docker.list > /dev/null
```

Anschließend kann die eigentliche Installation von Docker gestartet werden:

```
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

[doc22b]
Mit der erfolgreichen Installation von Docker, ist die Einrichtung der Maschine weitestgehend abgeschlossen, dieses Verfahren wird für alle 7 VMs durchgeführt. Die weitere Einrichtung und Konfiguration des Service Mesh und der Anwendungen erfolgt ausschließlich innerhalb Dockers, ein weiterer Zugriff auf die virtuelle Maschine ist nicht weiter

nötig. Eine Ausnahme bildet die Anpassung der Firewall Richtlinien des Linux-Kernels im nachfolgenden Kapitel 4.5.

4.3.2 Provisionierung der Container mittels IaC

Das Hauptaugenmerk bei der Erstellung der Testumgebung und der Inbetriebnahme liegt auf der einheitlichen Bereitstellung und Konfiguration der Dienste. Eine klare Dokumentation und zentrale Konfiguration ermöglicht einen verbesserten Überblick über die geschaffene Serverumgebung. Die damit verbundene Fähigkeit, reproduzierbare Ergebnisse zu erzielen, ist besonders bei der Fehleranalyse von großem Nutzen.

Aus diesen Gründen kommt bei der Provisionierung aller Dienste im Service Mesh, Infrastruktur als Code (IaC) zum Einsatz. IaC ist eine Methode zur Verwaltung und Bereitstellung von IT-Ressourcen durch maschinenlesbare Definitionsdateien anstelle von interaktiven Konfigurationstools oder einer physischen Hardwarekonfiguration [red22]. Zum Einsatz kommt hier das Produkt Terraform der Firma HashiCorp. Dieses bietet eine hervorragende Unterstützung, sowohl für Docker als auch für Consul [ter22]. Dadurch ist es möglich, die gesamte Einrichtung und Konfiguration der Testumgebung von einem zentralen Verzeichnis aus mit den entsprechen Konfigurationsdaten durchzuführen. Um Fehlerzustände verlässlich reproduzieren und nachvollziehen zu können, wird das Versionskontrollsystem GIT eingesetzt [git22]. Dieses ermöglicht Änderungen an den Konfigurationsdaten nachzuvollziehen und zum letzten stabilen Zustand zurückkehren zu können [git22]. Das Konfigurationsverzeichnis wird aufgrund des großen Umfangs als digitale Anlage zu dieser Arbeit bereitgestellt, im weiteren Verlauf werden nur Auszüge dieser Konfiguration betrachtet.

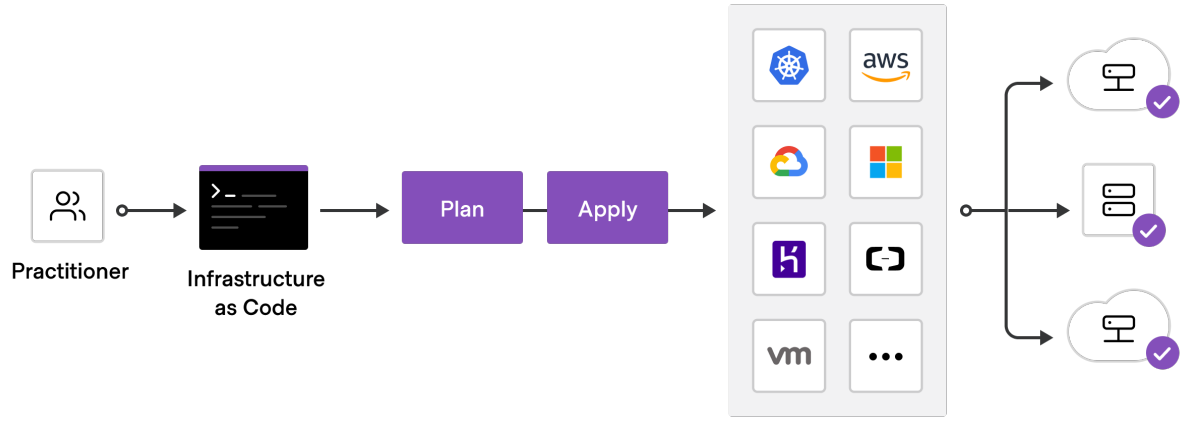

Abbildung 4.2: Konfiguration einer IT-Infrastruktur durch HashiCorp Terraform - [aws22b]

Die Abbildung zeigt die schematische Darstellung der Konfiguration einer IT-Infrastruktur durch die IaC Plattform HashiCorp Terraform.

4.4 Installation des Service Meshs

Die eigentliche Installation und Konfiguration des Consul Service Meshs wird an dieser Stelle verkürzt dargestellt, da dies seitens des Herstellers bereits aussreichend dokumentiert ist.

4.4.1 Installation und Konfiguration der Consul Server

X.509 bildet die zentrale Komponente zur Sicherstellung der Authentizität innerhalb eines Consul Service Mesh. Daher besteht der erste Schritt der Installation aus der Generierung dieser Zertifikate. Die Generierung erfolgt durch eine lokale Installation von Consul auf dem Rechner des Administrators. Im ersten Schritt wird das selbstsignierte Wurzelzertifikat des Clusters erstellt, mit welchem alle nachfolgenden Zertifikate signiert werden:

```
$ consul tls ca create -domain consul
==> Saved consul-agent-ca.pem
==> Saved consul-agent-ca-key.pem
```

Anschließend werden die Zertifikate der Consul Server an den einzelnen Standorten generiert. Es werden Zertifikate für folgende drei Standorte benötigt: koecher, merkur, venus. Es folgt exemplarisch die Generierung der Zertifikate für den Standort “koecher“:

```
$ consul tls cert create -server -dc koecher -domain consul -node server
==> WARNING: Server Certificates grants authority to become a
    server and access all state in the cluster including root keys
    and all ACL tokens. Do not distribute them to production hosts
    that are not server nodes. Store them as securely as CA keys.
==> Using consul-agent-ca.pem and consul-agent-ca-key.pem
==> Saved koecher-server-consul-0.pem
==> Saved koecher-server-consul-0-key.pem
```

Das generierte Zertifikat lässt sich mithilfe des Kommandozeilenwerkzeugs *openssl* wie folgt darstellen:

```
1 $ openssl x509 -text -noout -in koecher-server-consul-0.pem
2 Certificate:
3     Data:
4         Version: 3 (0x2)
5         Serial Number:
6             af:df:7b:d7:d8:e8:8f:e7:4f:be:e0:2a:34:11:94:a8
7     Signature Algorithm: ecdsa-with-SHA256
8         Issuer: C=US, ST=CA, L=San Francisco/street=101 Second Street/postalCode=94105,
          ↪  O=HashiCorp Inc., CN=Consul Agent CA
          ↪  339576761432331032200675563021990436351
```

```
9         Validity
10            Not Before: Mar 30 09:51:04 2022 GMT
11            Not After : Mar 30 09:51:04 2023 GMT
12        Subject: CN=server.koecher.consul
13        Subject Public Key Info:
14            Public Key Algorithm: id-ecPublicKey
15                Public-Key: (256 bit)
16                pub:
17                    04:5a:b7:54:2d:2f:de:5d:38:18:e8:0f:38:c5:f6:
18                    34:91:c4:f9:ae:6f:dc:b7:29:10:87:94:8f:69:ba:
19                    5b:d8:e3:0c:cd:8f:0e:16:a2:ad:f3:0d:7a:39:9f:
20                    6f:7d:08:b7:67:02:6b:7e:33:2c:fd:0a:bd:5c:e2:
21                    63:ed:6b:95:4d
22                ASN1 OID: prime256v1
23                NIST CURVE: P-256
24        X509v3 extensions:
25            X509v3 Key Usage: critical
26                Digital Signature, Key Encipherment
27            X509v3 Extended Key Usage:
28                TLS Web Server Authentication, TLS Web Client Authentication
29            X509v3 Basic Constraints: critical
30                CA:FALSE
31            X509v3 Subject Key Identifier:
32
                  ↪  D6:56:B7:C5:01:6B:DC:B7:66:9B:83:68:F2:90:EE:71:71:5E:1B:05:E2:F4:03:C7:F8:39:B8
33            X509v3 Authority Key Identifier:
34
                  ↪  keyid:0F:8A:C0:9D:19:CF:9C:64:E1:A0:52:05:F8:B1:8E:AF:50:2E:09:93:BB:FF:B7:09:5B
35            X509v3 Subject Alternative Name:
36                DNS:server.server.koecher.consul, DNS:server.koecher.consul,
                  ↪  DNS:localhost, IP Address:127.0.0.1
37    Signature Algorithm: ecdsa-with-SHA256
38         30:45:02:20:6f:62:27:6f:42:65:40:bf:b0:45:86:9d:56:1c:
39         1b:e7:ba:51:42:ef:30:81:0a:4e:eb:d2:0a:24:7d:db:30:e3:
40         02:21:00:e1:2c:2d:e9:75:fe:f4:8d:85:8d:1b:1f:1c:69:93:
41         6d:57:4a:2f:c2:dd:f7:89:2c:7e:85:30:ed:6a:71:02:30
```

Erwähnenswert ist der Parameter “X509v3 Subject Alternative Name“ (SAN) in Zeile 35. Dieser erlaubt es dem Zertifikat, seine Gültigkeit auch bei Verbindungen zum Consul Server zu behalten, die nicht den üblichen Subject Common Name *server.koecher.consul* verwenden. Dadurch sind auch lokale Verbindungen zum Consul Server und Consul Agent möglich, dies wird im weiteren Verlauf bei der Kommunikation mit den lokalen Proxy Servern relevant sein. Dies stellt in der Theorie ein Sicherheitsrisiko dar, da die Verifizierung des Zielservernamens über den uneindeutigen Hostnamen “localhost“ nicht verlässlich ist. Praktisch setzt die Manipulation der lokalen Namensauflösung “localhost“ jedoch vorraus, dass die Maschine bereits kompromittiert wurde. Sicherheitsbedenken hinsichtlich einer geschwächten X.509 Verifizierung sind daher hinfällig.

Im nächsten Schritt zum Service Mesh erfolgt die Bereitstellung der zentralen Consul Dienste auf der Maschine *server.koecher.local*. Seitens HashiCorp wird zu diesem Zweck ein Docker Image zur Verfügung gestellt, welches bereits eine Installation enthält und je nach Konfiguration als Consul Server oder Agent eingesetzt werden kann [doc22c]. Da in vielen Anwendungsfällen Consul eine lokale Installation des Envoy Proxy Servers benötigt, wird mithilfe von Terraform ein eigenes Image erstellt. Es bündelt die Bestandteile des Consul- und Envoy-Docker Images. Das Image kann durch folgende Terraform Konfigurationsdatei generiert werden, es wird von nun an zur Erstellung eigener Container eingesetzt.

```
1  #SYSTEM:    ADMIN
2  #CONTAINER: -
3  #PATH:      Terraform/Docker@server.koecher.local/consulenvoy.tf
4  #AUTHOR:    Calvin Köcher
5
6  resource "local_file" "dockerfile" {
7      filename = "consulenvoy/Dockerfile"
8      content     = <<-EOT
9      FROM consul:latest
10     FROM envoyproxy/envoy:v1.20.2
11     COPY --from=0 /bin/consul /bin/consul
12     ENTRYPOINT ["consul"]
13     EOT
14 }
15
16 resource "docker_image" "consulenvoy" {
17     name = "consulenvoy"
18     build {
19         path  = "consulenvoy"
20     }
21 }
```

In der Datei *consulenvoy.tf* werden zwei Ressourcen erzeugt, dabei beschreibt Terraform Ressourcen in folgender Struktur:

```
resource "[Typ]" "[Name]" {
    [Schlüssel] = [Wert]
}
```

Zunächst wird die Ressource *dockerfile* definiert. Diese veranlasst Terraform dazu, eine lokale Datei Namens “consulenvoy/Dockerfile“ auf der Maschine des Administrators zu erzeugen. Der Inhalt der Datei wird hierbei durch den Schlüssel *content* definiert. Die Datei stellt eine Konfigurationsdatei für den nachfolgenden Erstellungsprozess des neuen Docker Images dar, die zweite Ressource die eigentliche Generierung des Images. Es steht von nun an unter dem Namen *consulenvoy* zur Verfügung.

Nachdem das benötige Image für den Consul Container vorbereitet wurde, wird dieser nun durch folgende Ressourcendefinition von Terraform auf dem Zielserver erzeugt. Die Nachfolgende Konfigurationsdatei “server.tf“ fasst alle Schritte die zum Betrieb des Consul Servers nötig sind zusammen. Dazu zählt die Erstellung des Volumes “consul-data“, die Definition des Containers selber und die Bereitstellung der benötigten Konfigurationsdateien. Die Konfiguration des Consul Servers findet in der Datei “/consul/config/config.hcl“ statt, welche ab der Zeile 31 definiert wird. Die nachfolgende Datei beschränkt sich auf einige wichtige Konfigurationen, weitere Erläuterungen sind als Kommentar hinzugefügt.

```
1  #SYSTEM:    ADMIN
2  #CONTAINER: -
3  #PATH:      Terraform/Docker@server.koecher.local/server.tf
4  #AUTHOR:    Calvin Köcher
5
6  # Erstellung des Volumes "consul-data"
7  resource "docker_volume" "consul-data" {
8      name = "consul-data"
9  }
10
11 # Erstellung des Containers "server"
12 resource "docker_container" "server" {
13     # Selbst generiertes Image
14     image = docker_image.consulenvoy.latest
15
16     # Container Name
17     name  = "server"
18
19     ...
20     # Container nutzt Host-Netzwerk
21     network_mode = "host"
22
23     # Start von Consul innerhalb des Containers unter Verwendung des
   ↪  Konfigurationsverzeichnis: /consul/config
24     entrypoint = [
25         "consul",
26         "agent",
27         "-config-dir=/consul/config"
28     ]
29
30     # Upload folgender Konfigurationsdatei in den Container
31     upload {
32         file = "/consul/config/config.hcl"
33         content = <<-EOT
34
35         # NODE
36         node_name  = "server"
37         datacenter = "koecher"
```

```
38          data_dir   = "/consul/data"
39          ...
40
41          # NETWORK
42          client_addr = "0.0.0.0"
43          bind_addr = "192.168.1.176"
44          ports {
45              dns   = 53
46              http  = 8500
47              https = -1
48              grpc  = 8502
49          }
50
51          # ENCRYPTION
52          encrypt = "Luj2FZWwlt8475wD1WtwUQ=="
53          ca_file = "ca.pem"
54          cert_file = "cert.pem"
55          key_file = "key.pem"
56          ...
57
58          # SERVER MODE
59          server = true
60          ui_config {
61              enabled = true
62          }
63          connect {
64              enabled = true
65              enable_mesh_gateway_wan_federation = true
66          }
67
68
69          EOT
70      }
71      ...
72
73      # Upload der X.509 Zertifizierungsstelle und des öffentlichen- bzw. privaten
    ↪  Serverzertifikats
74      upload {
75          file = "ca.pem"
76          source = "consul-agent-ca.pem"
77          source_hash = filemd5("consul-agent-ca.pem")
78      }
79      upload {
80          file = "cert.pem"
81          source = "koecher-server-consul-0.pem"
82          source_hash = filemd5("koecher-server-consul-0.pem")
83      }
84      upload {
```

```
85          file = "key.pem"
86          source = "koecher-server-consul-0-key.pem"
87          source_hash = filemd5("koecher-server-consul-0-key.pem")
88      }
89
90  }
```

Durch das Kommando “$ terraform apply“ wird die getätigte Konfiguration auf dem Zielserver angewendet. Insgesamt wird dies auf den Servern in allen drei Rechenzentren angewendet. Diese bilden im Anschluss ein vereintes Service Mesh. Beim ersten Start des neu erstellten Consul Containers initialisiert Consul eine Datenbank und stellt kurz darauf ein Dashboard zur Systemübersicht auf dem Port 8500 zur Verfügung (siehe Abbildung 4.3). Sollten es zu diesem Zeitpunkt bereits andere aktive Server im Service Mesh geben, findet eine automatische Synchronisation des Datenbestandes zwischen den Servern statt. Die Server entscheiden durch ein Wahlverfahren, welcher Server die Leitung über das Mesh übernimmt, andere Server befinden sich im sogenannten “Follow-Modus“. Sie überwachen und synchronisieren die Aktivitäten des “Leaders“ und sind jederzeit in der Lage die Führung zu übernehmen sollte dieser ausfallen. Um ein “Split Brain“ Szenario zu verhindern, wird seitens des Herstellers eine ungerade Anzahl an Servern empfohlen. Als Split Brain Szenario bezeichnet man den Fall, in welchen ein Verbund mehrerer Server durch eine Störung in mindestens zwei Gruppen aufgeteilt wird. Da die Gruppen nicht in der Lage sind, miteinander zu kommunizieren, besteht die Gefahr, dass jede Gruppe eigenständig die Leitung über das Clusters übernimmt [dew22]. Bei der Konfiguration der Server besteht zudem die Möglichkeit, eine Mindestanzahl an Servern festzulegen, welche bei der Wahl beteiligt sein müssen. Besitzt ein Mesh beispielsweise fünf Server, bietet sich eine Konfiguration von mindestens drei Servern an.

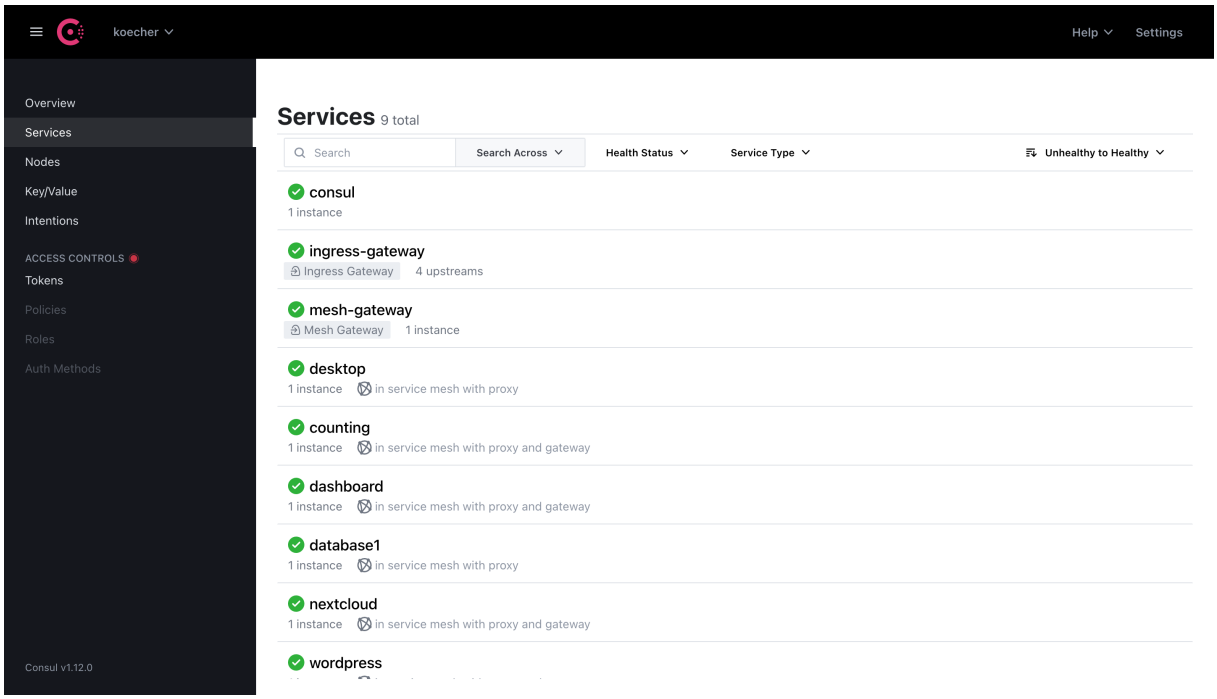

Abbildung 4.3: Consul Dashboard mit allen definierten Services

Die Abbildung zeigt eine Übersicht des gestarteten Consul Clusters in der bereits alle definierten Services registriert und online sind.

4.4.2 Installation und Konfiguration der Consul Agents

Die Installation und Konfiguration der Consul Agents auf den Anwendungsservern ist der des Servers sehr ähnlich. Die Bereitstellung des Images und des Containers sind dabei identisch, lediglich bei der Konfiguration des Agents gibt es Unterschiede. Damit ein Consul Agent beim Start eine Verbindung zum bestehenden Cluster aufbauen kann, müssen eine oder mehrerer Netzwerkadressen oder Hostnamen der Consul Server angeben werden. Konnte der Agent erfolgreich einen der Consul Server kontaktieren, erhält dieser alle Informationen über das Service Mesh, dessen Topologie und die Adressen aller weiteren verfügbaren Consul Server. Sollten alle initial konfigurierten Server nicht länger verfügbar sein, wird so sichergestellt, dass der Agent nicht vom Mesh abgeschnitten wird. Anschließend registriert der Agent alle konfigurierten Dienste und startet für jeden Service eine Envoy Proxy Instanz, welcher die eigentliche Kommunikation des Services mit dem Service Mesh auf der Datenschicht übernimmt.

4.4.3 Konfiguration der Services

Jeder Service der später im Service Mesh zur Verfügung stehen soll, muss dazu zunächst definiert und registriert werden. Die Registrierung erfolgt hierbei durch den Consul Agent, der sich auf dem selben System wie die Serviceanwendung befindet. Definiert werden die Services in lokalen Konfigurationsdateien, die durch den Consul Agent gelesen und verarbeitet werden. Bei der Konfiguration wird ein Service-Name sowie eine eindeutige

Service-ID definiert. Sollten im Service Mesh mehrere Services mit der selben Service-ID existieren, werden diese vom Service Mesh als weitere Instanzen des selben Services angesehen und ankommende Anfragen gleichmäßig auf diese verteilt. Das Attribut “port“ gibt den Port an, unter welchem die Service-Anwendung lokal erreichbar ist. Benötigt die lokale Anwendung Zugriff auf andere Services, kann auf diese über lokale Ports zugegriffen werden. Die Ports werden vom lokalen Envoy Proxy gebunden und durch das Service Mesh zum Ziel Service geroutet, diese Verbindungen werden im folgenden als “Upstreams“ bezeichnet. Der im nachfolgenden Beispiel definierte Service “dashboard“, benötigt zum Betrieb Zugriff auf den Service “counting“. Dazu wird im Segment “connect“ ein upstream definiert, der Konnektivität zum Service “counting“ im Rechenzentrum “venus“ über den lokalen Port 9003 ermöglicht.

```
#SYSTEM:    client2.koecher.local
#CONTAINER: agent
#PATH:      /consul/config/dashbaord.service.hcl
#AUTHOR:    Calvin Köcher

service {
    name = "dashboard"
    id = "dashboard"
    port = 9002
    connect {
        sidecar_service {
            proxy {
              transparent_proxy {
                  dialed_directly = true
              }
              upstreams = [
                {
                  destination_name = "counting",
                  datacenter = "venus",
                  local_bind_port  = 9003
                }
              ],
            }
        }
    }
    check {
        id       = "dashboard-check"
        http     = "http://127.0.0.1:9002/health"
        method   = "GET"
        interval = "1s"
        timeout  = "1s"
    }
}
```

Im letzten Segment der Service-Definition besteht die Möglichkeit, die Verfügbarkeit des Services zu überwachen. Das Monitoring wird dabei vom lokalen Consul Agent durchgeführt und eine Statusänderung im Cluster publiziert. Dadurch ist es dem Administrator möglich, die Verfügbarkeit einzelner Service-Instanzen zu überwachen und im Fehlerfall das Problem besser eingrenzen zu können. Auch das Service Mesh verwendet diese Informationen, um beispielsweise automatisch betroffene Instanzen aus der Verteilung des Load Balancers zu entfernen oder neue Verbindungen auf einen Service in einem anderen Rechenzentrum umzuleiten.

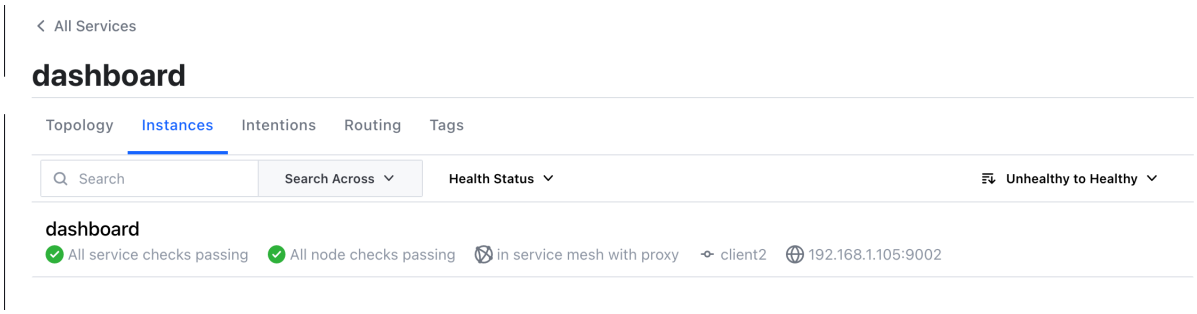

Abbildung 4.4: Übersicht über die Instanzen eines Services im Consul Dashboard

Die Abbildung zeigt eine Übersicht über die Instanzen eines Services im Consul Dashboard und dessen Monitoring.

4.5 Sichere Übertragung durch den Envoy Sidecar Proxy

Die Kommunikation der Services untereinander erfolgt ausschließlich über eine durch den Envoy Proxy Ende-zu-Ende verschlüsselte Verbindung. Im weiteren Verlauf wird Consul so konfiguriert, dass nur explizit erlaubte Verbindungen zugelassen werden, alle anderen werden blockiert. Auch wenn zum aktuellen Zeitpunkt alle Verbindungen der Services untereinander authentifiziert und verschlüsselt werden, fehlt zur vollwertigen Implementierung einer ZTNA die Umsetzung der Mikrosegmentierung. Dazu müssen alle Systeme durch Firewalls so vom Netzwerk abgeschottet werden, dass nur Verbindungen die über den Envoy Proxy Server geleitet werden möglich sind. Erreicht wird dies durch entsprechende Konfiguration der im Linux Kernel integrieren Firewall durch das Konfigurationsprogramm “IP-Tables“.

4.5.1 Funktionsweise und Betriebsarten

Envoy bildet die zentrale Anlaufstelle, für eingehende und ausgehende Verbindungen auf dem System. Verbindungen, die am Proxy Server vorbei direkt auf Netzwerkressourcen zugreifen, müssen daher unterbunden werden. Auf der eingehenden Seite, werden durch entsprechende Firewall Richtlinien alle Verbindungen blockiert, die nicht an den Port des lokalen Proxy Servers gerichtet sind. Als Ausnahme hierzu wird in diesem Versuchsaufbau eine Sonderstellung für den Port 22 konfiguriert, um jederzeit den Zugriff über das SSH Protokoll auf das System sicherzustellen. Wie in Kapitel 3 bereits beschrieben, ist

keine Beschränkung des ausgehenden Netzwerkverkehrs vorgesehen. Dennoch kann es in bestimmten Anwendungsfällen vorteilhaft sein, auch den ausgehenden Verkehr durch den Proxy zu leiten. Der Envoy Proxy kann daher in zwei verschiedenen Modi betrieben werden:

Einsatz als direkter Proxy

Dies ist der Standartfall, der auch im vorangegangenen Beispiel einer Service-Definition beschrieben wurde. Die Anwendung wird explizit so konfiguriert, dass Verbindungen zu einen anderen Service über den lokalen Envoy Proxy aufbaut werden. Der Proxy Server bindet einen lokalen Port über welchen den Datenverkehr zum Zielservice Weitergeleit wird. Dazu müssen vor dem Servicestart alle benötigten "Upstreams" in den Servicekonfiguration definiert sein. Anwendungen auf dem System können andere beziehungsweise externe Netzwerkressourcen direkt und ohne Einschränkungen kontaktieren. DNS-Anfragen werden vom Betriebssystem an den lokalen Consul Agent weitergeleitet. Handelt es sich bei dieser DNS-Anfrage um eine Consul Ressource wird die Anfrage durch den lokalen Consul Agent beantwortet, ansonsten wird die Anfrage rekursiv an einen definierten externen Namensserver weitergeleitet. Abbildung 4.5 stellt die Kommunikation zwischen den beschreiben Softwarekomponenten in diesem Modus detailliert dar. Die beschrieben Anforderungen werden durch folgende manuelle Konfiguration der Firewall umgesetzt:

```
1  #SYSTEM:    -
2  #CONTAINER: -
3  #PATH:      /etc/iptables/rules.v4
4  #AUTHOR:    Calvin Köcher
5
6  *filter
7  # Default Policy
8  :INPUT DROP [0:0]
9  :FORWARD DROP [0:0]
10 :OUTPUT ACCEPT [0:0]
11
12 # Consul Input Filter (Exception for SSH, Exception for Consul RPC, Exception for
   ↪  Envoy Sidecars)
13 -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
14 -A INPUT -i lo -j ACCEPT
15 -A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
16 -A INPUT -p tcp -m tcp --dport 8301 -j ACCEPT
17 -A INPUT -p udp -m udp --dport 8301 -j ACCEPT
18 -A INPUT -p tcp -m tcp --dport 21000:21255 -j ACCEPT
19 -A INPUT -j LOG --log-prefix "DROPPED INPUT: "
20 -A INPUT -j DROP
21
22 COMMIT
```

Erläuterung des Filterprozesses der Firewall für eingehende Verbindungen:

- Zeile 8-10 : Eingehende und weitergeleitete Verbindungen werden blockiert. Ausgehende Verbindungen bleiben unbeschränkt.
- Zeile 13 : Ausnahme für bereits bestehende eingehende Verbindungen.
- Zeile 14 : Ausnahme für Verbindungen zu lokalen Zielen (Loopback Interface).
- Zeile 15 : Ausnahme für Verbindungen auf Port 22/TCP, um den Zugriff über SSH sicherzustellen.
- Zeile 16-17 : Ausnahme für Verbindungen zum Consul Agent auf Port 8301/TCP und 8301/UDP.
- Zeile 18 : Ausnahme für Verbindungen zu lokalen Envoy Proxy Servern, diese binden den Portbereich 21000:21255/TCP.
- Zeile 19 : Verbindungen, die zu diesem Zeitpunkt auf keine der Ausnahmen zutrifft, werden im nächsten Schritt verworfen. Daher wird an dieser Stelle ein Protokolleintrag dieser unzulässigen Verbindung erstellt.
- Zeile 20 : Verwerfen der Verbindung. Der Empfänger erhält keine Information darüber, dass die Verbindung aktiv abgelehnt wurde.

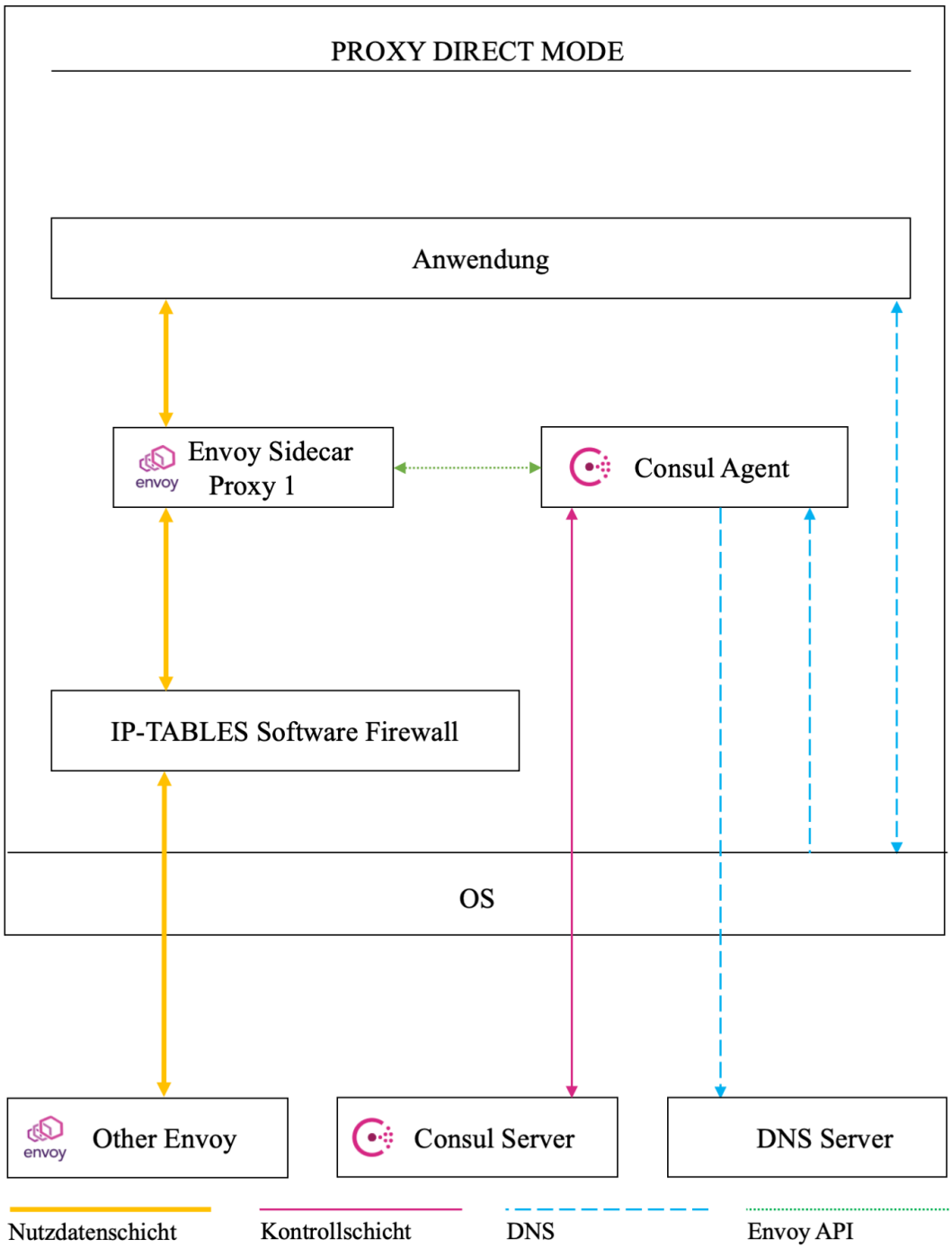

Abbildung 4.5: Betrieb des Envoy Proxy Servers im direkten Modus

Die Grafik zeigt den Betrieb des Envoy Proxy im direkten Modus. Es wird dargestellt, wie die Anwendung direkt mit dem Proxy Server kommuniziert um “Upstream“ Ressourcen zu erreichen. Eingehende Verbindungen werden durch die Firewall überwacht, die Kommunikation zum Consul- und DNS Server wird dadurch nicht beeinträchtigt.

Einsatz als transparenter Proxy

Neben dem direkten Modus kann der Envoy Proxy auch transparent betrieben werden. Benötigt die lokale Anwendung Zugriff auf eine große Anzahl oder wechselnde Services im Services Mesh, kann eine statische Konfiguration der dafür notwendigen "Upstreams" unverhältnismäßig aufwändig sein. Die Konfiguration der benötigten "Upstreams" und das Binden lokaler Ports ist beim transparenten Betriebsmodus nicht länger notwendig. Die Anwendung kann den externen Service direkt über seinen Servicenamen kontaktieren. Um dies zu ermöglichen, wird durch entsprechende Firewallrichtlinien der gesamte ausgehende Datenverkehr zurück zum internen Proxy Server geleitet. Dieser filtert Verbindungen, die sich an Services im Services Mesh richten heraus und baut für diese voll automatisch den benötigten Netzwerktunnel zum Zielservice auf. Alle anderen Verbindung werden vom Proxy unberührt an deren Zieladresse weitergeleitet. Verbindungen die durch den Proxy Server hergestellt werden, müssen die Firewall ohne weitere Umleitung passieren können. Hierzu wird der Envoy Proxy unter der Benutzerkennung "9999" betrieben und entsprechende Ausnahmen für diesen Benutzer in der Firewall hinterlegt. Der DNS Mechanismus ist identisch zum Betrieb eines Proxy Servers im direkten Modus. In Kombination mit diesem lassen sich im transparenten Modus Ressourcen des Services Meshs nahtlos über deren jeweiligen virtuellen Hostnamen aus allen Anwendungen des Systems aufrufen. Die nötigen Firewallrichtlinien sind dabei jedoch umfangreicher, da nicht nur eine Verarbeitung des eingehenden-, sondern auch des ausgehenden Datenverkehrs erfolgen muss. Die Regeln für den eingehenden Netzwerkverkehr sind dabei nahezu identisch zu denen im vorangegangenen Fall. Hier werden lediglich Verbindungen auf unbekannte Ports nicht länger blockiert, sondern zur weiteren Verarbeitung über den Port 21000 auf dem Proxy Server umgeleitet (Zeile 21-27 im nachfolgenden Code). Abbildung 4.6 stellt die Kommunikation zwischen den beschreiben Softwarekomponenten in diesem Modus detailliert dar. Die beschrieben Anforderungen werden durch folgende manuelle Konfiguration der Firewall umgesetzt:

```
1  #SYSTEM:     -
2  #CONTAINER: -
3  #PATH:       /etc/iptables/rules.v4
4  #AUTHOR:     HashiCorpConsul, Calvin Köcher
5
6  *nat
7  # Default Policy
8  :PREROUTING ACCEPT [0:0]
9  :INPUT ACCEPT [0:0]
10 :OUTPUT ACCEPT [0:0]
11 :POSTROUTING ACCEPT [0:0]
12
13 # Create Consul Chains
14 :CONSUL_DNS_REDIRECT - [0:0]
15 :CONSUL_PROXY_INBOUND - [0:0]
16 :CONSUL_PROXY_IN_REDIRECT - [0:0]
```

```
17 :CONSUL_PROXY_OUTPUT - [0:0]
18 :CONSUL_PROXY_REDIRECT - [0:0]
19
20 # Consul Input Redirect (Exception for SSH, Exception for Prometheus Monitoring,
   ↪  Exception for Consul RPC)
21 -A PREROUTING -p tcp -j CONSUL_PROXY_INBOUND
22 -A CONSUL_PROXY_INBOUND -p tcp -m tcp --dport 22 -j RETURN
23 -A CONSUL_PROXY_INBOUND -p tcp -m tcp --dport 8301 -j RETURN
24 -A CONSUL_PROXY_INBOUND -p udp -m udp --dport 8301 -j RETURN
25 -A CONSUL_PROXY_INBOUND -j LOG --log-prefix "REDIRECT INPUT: "
26 -A CONSUL_PROXY_INBOUND -p tcp -j CONSUL_PROXY_IN_REDIRECT
27 -A CONSUL_PROXY_IN_REDIRECT -p tcp -j REDIRECT --to-ports 21000
28
29 # Consul Output (Exception for localhost, Exception for User 9999, Rewrite DNS to
   ↪  localhost:8600,Rewrite DNS)
30 -A OUTPUT -p tcp -j CONSUL_PROXY_OUTPUT
31 -A CONSUL_PROXY_OUTPUT -d 127.0.0.1/32 -j RETURN
32 -A CONSUL_PROXY_OUTPUT -m owner --uid-owner 9999 -j RETURN
33 -A CONSUL_PROXY_OUTPUT -j LOG --log-prefix "REDIRECT OUTPUT: "
34 -A CONSUL_PROXY_OUTPUT -j CONSUL_PROXY_REDIRECT
35 -A CONSUL_PROXY_REDIRECT -p tcp -j REDIRECT --to-ports 15001
36
37 COMMIT
```

Erläuterung des Filterprozesses der Firewall für ausgehende Verbindungen:

- Zeile 30 : Weiterleitung auf eine eigene Regelkette zur weiteren Filterung.
- Zeile 31 : Ausnahme für Verbindungen zu lokalen Zielen (Loopback Interface).
- Zeile 32 : Ausnahme für Verbindungen von Anwendungen, die unter dem Benutzer mit der ID 9999 ausgeführt werden. So wird eine freie Kommunikation nach außen für den Proxy Server gewährleistet.
- Zeile 33 : Logging der umgeleiteten Verbindung zu Debug- und Überwachungszwecken.
- Zeile 34 : Weiterleitung an den lokalen Proxy Server über den Port 15001.

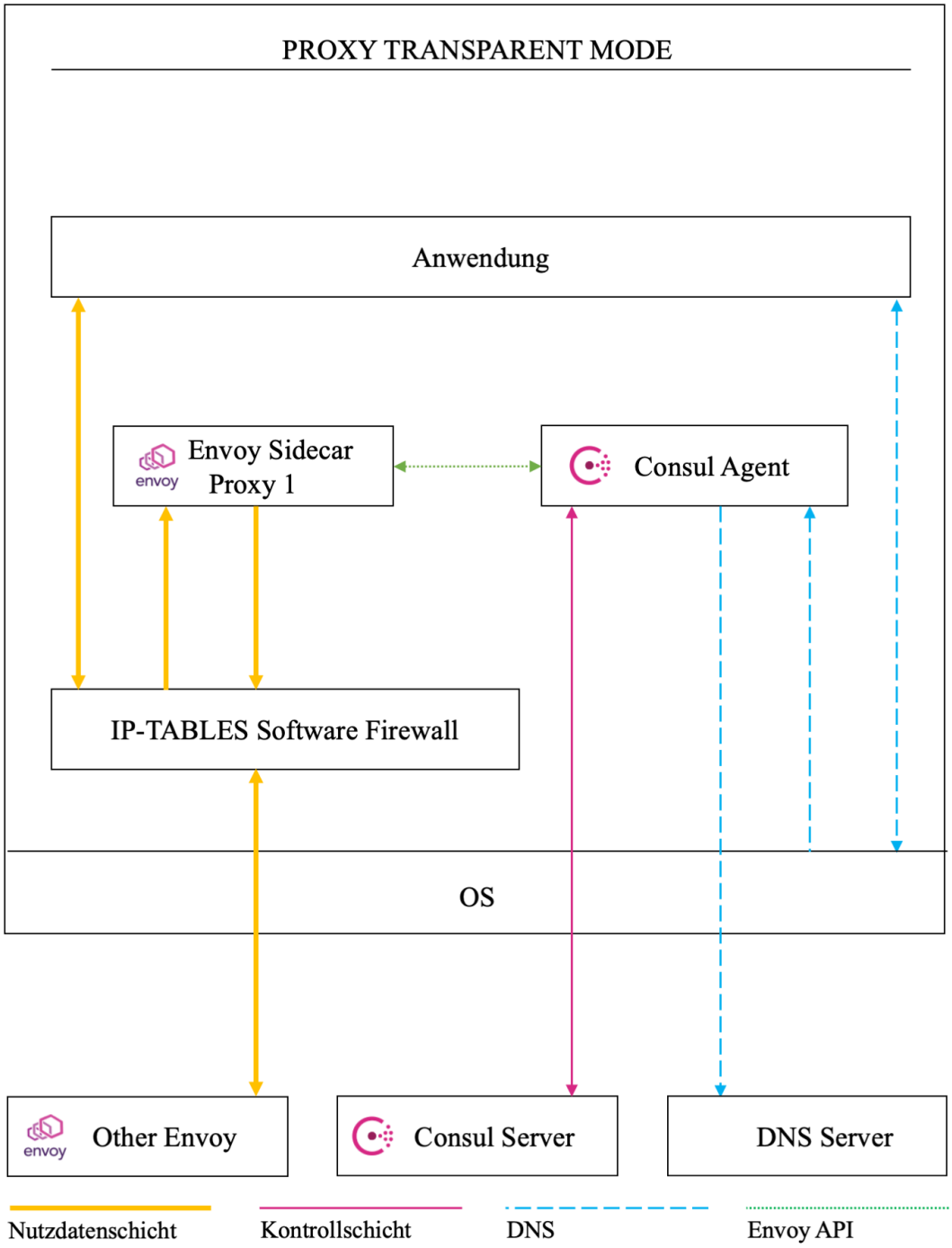

Abbildung 4.6: Betrieb des Envoy Proxy Servers im transparenten Modus

Die Grafik zeigt den Betrieb des Envoy Proxy Servers im transparenten Modus. Es ist dargestellt, wie der Datenverkehr der Anwendung durch die Firewall zum lokalen Proxy Server umgeleitet und verarbeitet wird. Eingehende Verbindungen werden durch die Firewall überwacht, die Kommunikation zum Consul- und DNS Server wird dadurch nicht beeinträchtigt.

4.6 Regelwerke zur Kommunikation der Dienste untereinander

Die Konfiguration der Regeln, inwieweit Services auf andere Services zugreifen dürfen, erfolgt über das zentrale Dashboard des Consul Servers (siehe Abbildung 4.7). Die Anwendung der Regeln erfolgt durch Consul abhängig von ihrer Spezifität in absteigender Reihenfolge. Um den Grundgedanken einer ZTNA gerecht zu werden, wird als Grundregel sämtliche Kommunikation unterbunden. Benötigt ein Service nun Zugriff auf einen anderen Service, kann dieser durch entsprechende Regeln gewährt werden.

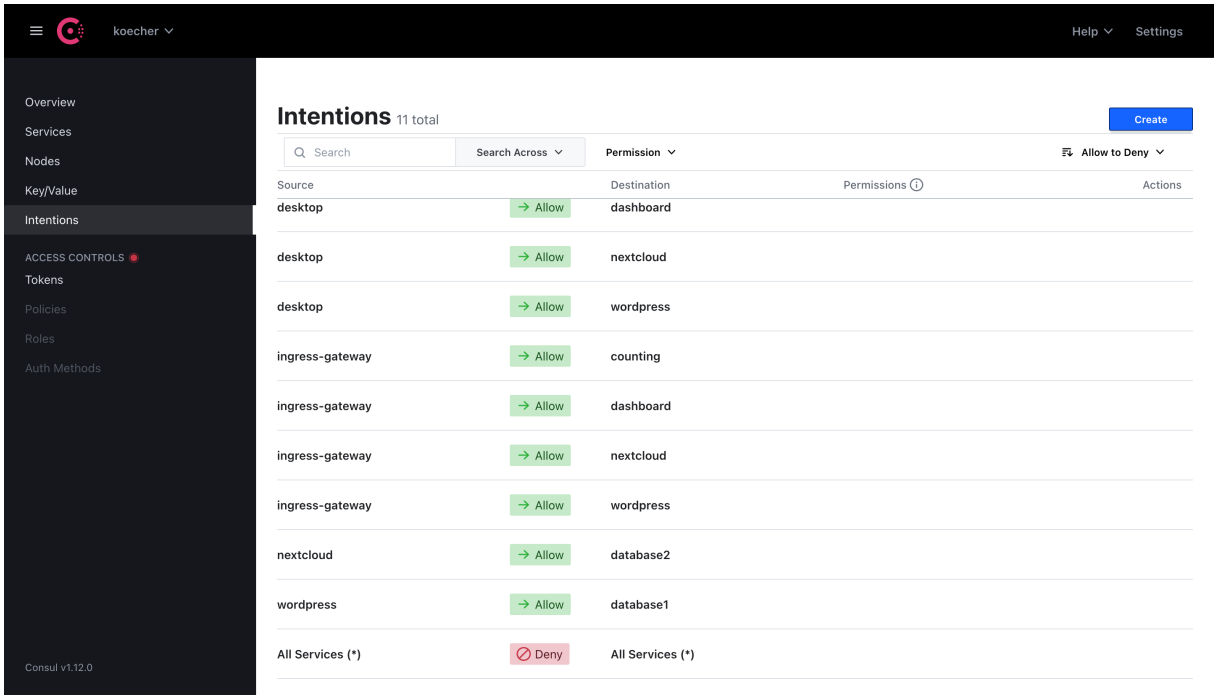

Abbildung 4.7: Regelwerke zur Kommunikation von Services unter Consul

Die Abbildung zeigt die Regeln, welche die Kommunikation im Service Mesh steuern. Durch eine allgemeine Regel wird zunächst sämtlicher Datenverkehr im Service Mesh unterbunden. Die Kommunikation zwischen zwei Services muss durch weitere Regeln explizit erlaubt werden.

4.7 Verbindung zwischen Service Mesh und der Außenwelt

Neben den Proxy Servern, die den Datenverkehr für die Anwendungen regeln, werden zum Betrieb des Service Meshs, weitere Proxys mit gesonderten Aufgaben benötigt.

4.7.1 Das Mesh Gateway

Das Mesh Gateway stellt einen zentralen Anlaufpunkt für Verbindungen dar, welche als Ziel einen Service in einem anderen Rechenzentrum haben (siehe Abbildung 4.8). Je nach Konfiguration erfolgt die Kommunikation dabei direkt mit dem entfernen Gateway oder wird erst über das eigene Gateway geleitet. So können auch Services auf Maschinen in anderen Rechenzentren erreicht werden, die selber durch Firewalls abgeschottet sind.

Auf technischer Ebene erfolgt die Adressierung des Ziel-Rechenzentrums und des Ziel-Services dabei durch optionale Header in den Servicezertifikaten. Dadurch kann der eigentliche Datenverkehr von den Gateways nicht gelesen werden, wodurch eine Ende-zu-Ende Verschlüsselung zwischen den Services gewahrt wird [Has22d, S.74]. Bei der Infrastruktur, die im Rahmen dieser Arbeit aufgebaut wird, wurde sich darüber hinaus dazu entschieden, auch die Kontrollschicht über die Gateways an andere Standorte zu leiten (siehe Abbildung 4.1). Dazu wird eine Funktion der Gateways aktiviert, welche das Bündeln von Daten der Kontrollschicht in Pakete der Nutzdatenschicht ermöglicht. Die Pakete werden anschließend über die bestehenden Mesh Gateways zwischen den Standorten transportiert. Dies erlaubt es, ein gesamtes Service Mesh über einen einzelnen Port aus dem Internet erreichbar zu machen.

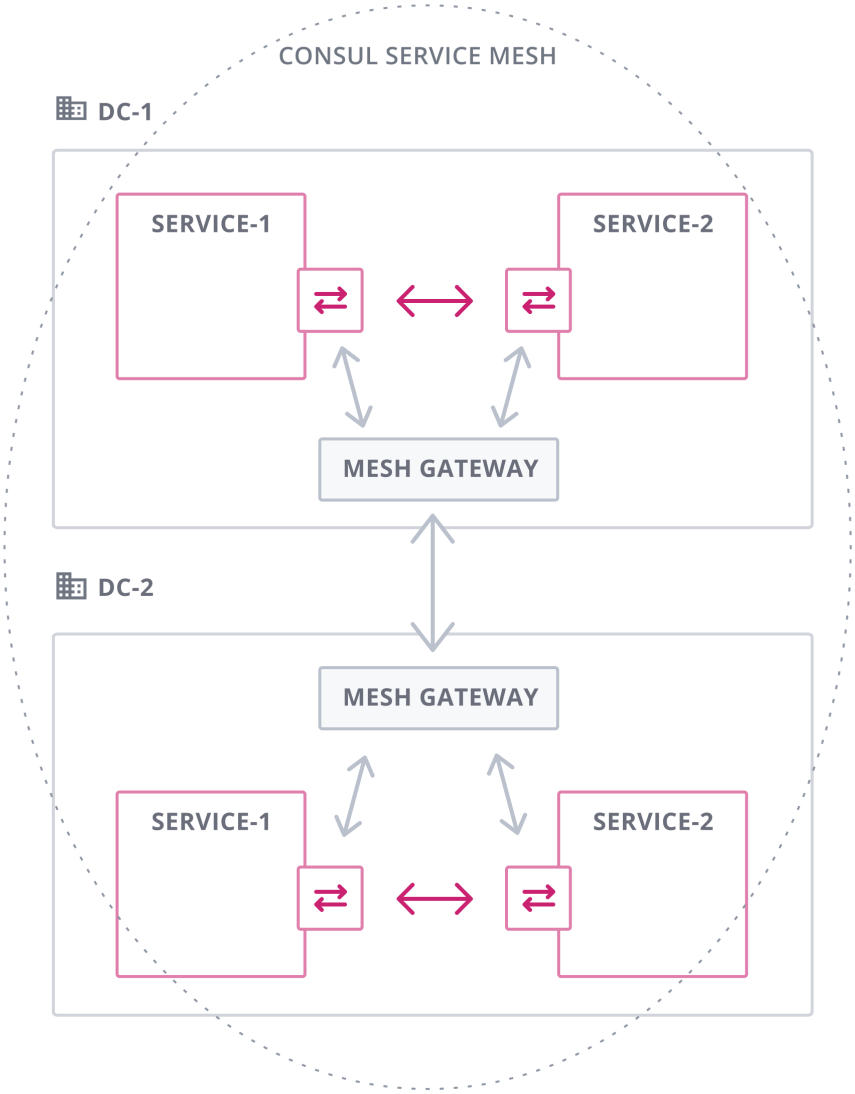

Abbildung 4.8: Funktionsweise eines Mesh Gateways in einem Consul Service Mesh - [boo22b]

Die Grafik zeigt die Funktionsweise eines Mesh Gateways in einem Consul Service Mesh. Es werden zwei Rechenzentren dargestellt, deren Services in der Lage sind, über das jeweils gegeben Mesh Gateway zu kommunizieren.

4.7.2 Der Ingress Controller

Der Ingress Controller bildet die Schnittstelle zwischen Services im Service Mesh und der Außenwelt. Wird im Service Mesh beispielsweise ein Webserver betrieben, ist dieser aufgrund der getroffenen Firewallrichtlinien nicht außerhalb des Meshs erreichbar. Soll dieser Server nun der Außenwelt zur Verfügung gestellt werden, erfolgt dies über den Ingress Controller. Dieser bindet einen öffentlich erreichbaren Port und leitet den Datenverkehr an den entsprechenden Service weiter (siehe Abbildung 4.9). Zu beachten ist hierbei, dass auch für den Ingress Controller dazu eine Zugriffsberechtigung auf den Service hinzugefügt werden muss. An dieser Stelle lassen sich ebenfalls Funktionalitäten wie ein Fail-Over oder ein Load-Balancer umsetzen.

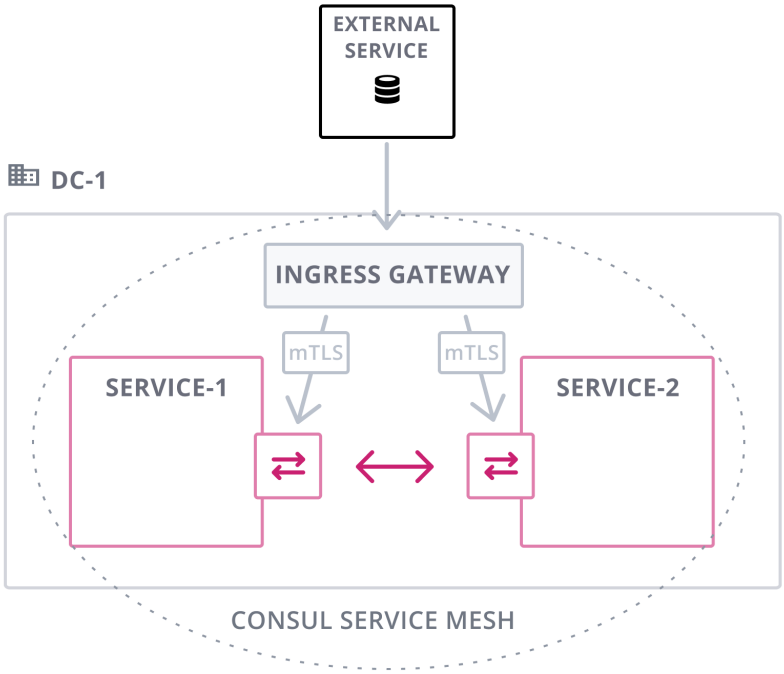

Abbildung 4.9: Funktionsweise eines Ingress Controllers in einem Consul Service Mesh - [boo22a]

Die Grafik zeigt die Funktionsweise eines Ingress Controllers in einem Consul Service Mesh. Es wird ein externer Service dargestellt, der mittels des Ingress Gateways auf interne Services zugreifen kann.

4.8 Übersicht über den Versuchsaufbau

An dieser Stelle verfügt der Versuchsaufbau über ein funktionsfähiges Service Mesh. Alle bei der Zielsetzung definierten Services sind betriebsbereit. Besonders hervorzuheben ist an dieser stelle die Rolle des Systems *desktop.node.koecher.consul*. Das System ist Teil des Meshs und verfügt über ein Linux-Desktop-Betriebssystem. Dieses wird im weiteren Verlauf als Ausgangspunk zur Durchführung von Tests verwendet. Das System betreibt den lokalen Envoy Proxy im transparenten Modus und hat daher Zugriff auf alle bereitgestellten Dienste. Eine Übersicht über das Service Mesh und deren Kommunikation untereinander kann folgender Abbildung 4.10 entnommen werden.

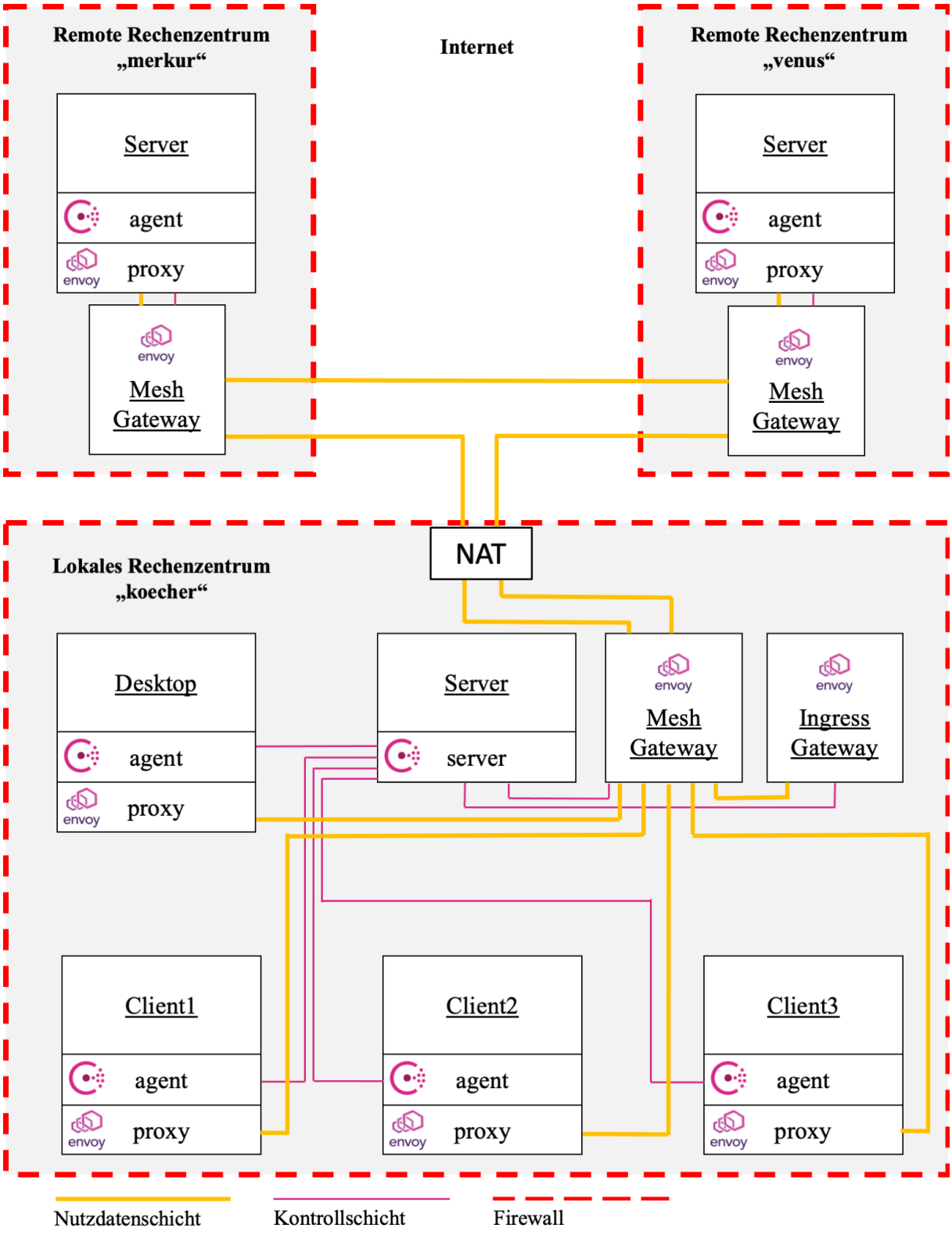

Abbildung 4.10: Vollständiges Systemlayout des Versuchsaufbaus

Die Abbildung zeigt das Systemlayout des Versuchsaufbaus. Zu sehen sind alle Systeme des Service Mesh und deren Aufteilung auf die drei Standorte. Zudem zeigt die Grafik die Softwarekomponenten von Consul auf den einzelnen Systemen und deren Interaktionen über die Nutzdaten- und Kontrollschicht.

5 Analyse und Sicherheit

Nachdem im vorangegangenen Kapitel die Installation und Konfiguration des Consul Service Meshs beschrieben wurde, folgt in diesem Kapitel eine genauere Betrachtung des Versuchsaufbaus. Dazu wird das Service Mesh hinsichtlich Sicherheit und Leistung analysiert.

5.1 Limitierungen einer ZTNA

In der Praxis kann die Verwendung eines Service Meshs zu Einschränkungen führen, was zu Problemen beim Einbinden eines Dienstes führt oder diesen ganz aus dem Service Mesh ausschließt.

5.1.1 Einschränkungen auf Netzwerkebene

Consul unterstützt ausschließlich einen einzelnen TCP Port pro Service. Daher ist der Betrieb von Anwendungen welche andere Protokolle, wie beispielsweise UDP verwenden nicht möglich. Dazu zählen unter anderem Protokolle wie SNMP, NTP oder RTP. Ein weiteres Problem stellen Dienste dar, die über mehrere TCP Ports kommunizieren müssen. Als Workaround können hier mehrere Services im Services Mesh registriert werden und damit verbunden auch mehrere Instanzen des Envoy Sidecar Proxy gestartet werden. Consul betrachtet diese Dienste dann als voneinander unabhängig. Knoten, die Zugriff auf diese Dienste benötigen, müssen nun für beide Ziele berechtigt werden. Dienste, die Server Ports dynamisch binden, wie es beispielsweise bei FTP [J P85] implementiert ist, können nicht mit ins Service Mesh aufgenommen werden, da alle Ports, Consul und dem lokalen Proxy vorher bekannt sein müssen.

5.1.2 Erschwerte Fehlersuche

Ein Service Mesh erhöht die Komplexität einer IT-Infrastruktur stark. Da mehrere Komponenten bei der Kommunikation zwischen zwei Anwendungen beteiligt sind, steigt auch die Anzahl möglicher Fehlerursachen. Consul unterstützt den Administrator hier, indem es neben den bei der Serviceregistrierung definierten Checks auch eigene Checks hinzufügt. Diese überwachen die Verfügbarkeit des für den Service zuständigen Agent und Proxy Servers. Dadurch kann besonders bei neuen Diensten eine grobe Fehlkonfiguration ausgeschlossen werden und die erfolgreiche Registrierung überprüft werden. Treten bis zu diesem Punkt Fehler auf, lassen sich diese schnell durch aussagekräftige Fehlermeldungen des Consul Server bzw. Clients eingrenzen. Ist dennoch keine Verbindung möglich, fokussiert sich die

Fehlersuche auf die zuständigen Instanzen des Envoy Proxy Servers der Nutzdatenschicht. Da diese standardmäßig keine Informationen über aktuelle Verbindungen und deren Status ausgeben, wurde bei allen Instanzen das erweiterte Logging durch das Hinzufügen des Parameters “-l trace“ aktiviert. Die nun erhaltenen Ausgaben sind in der Praxis jedoch nur bedingt hilfreich und nicht intuitiv verständlich.

5.2 Performance Analyse

Ein Datenpaket, das zwischen zwei Anwendungen im Service Mesh ausgetauscht wird, passiert im Gegensatz zu einem normalen Netzwerk zwei Proxy Server. Es soll festgestellt werden, welchen Einfluss der Einsatz eines Services Meshs auf die Netzwerkperformance hat. Dazu sollen verschiedene Messgrößen wie Durchsatz und Latenz zu verschiedenen Services analysiert werden. Die Tests werden vom System *desktop.node.koecher.consul* aus durchgeführt, die verwenden Befehle und deren Ausgaben befinden sich im Anhang. Um representative Ergebnisse zu erhalten, wird zwischen der Durchführung der Tests so wenig wie möglich an den Systemen verändert. Um Vergleichswerte der Performance einer direkten Netzwerkverbindung zu erhalten, werden im zweiten Schritt alle Firewallregeln zur Umleitung des Datenverkehrs auf den Servern deaktiviert. Die Adressierung der Zielservices ändert sich dadurch nicht, die Datenpakete nehmen nun einen direkten unverschlüsselten Pfad zum Zielsystem ohne dabei einen Proxy zu passieren.

5.2.1 HTTP Latenztest

Im ersten Test soll die Latenz zu einem HTTP Service betrachtet werden. Durchgeführt werden die Tests mit dem Kommandozeilen Werkzeug “httping“, welches es ermöglicht, flexibel konfigurierbare Latenztests zu HTTP-Ressourcen durchzuführen [deb22b]. Hierzu wird ein statisches Ziel auf einem HTTP Server ausgewählt, das selber keine weiteren Abhängigkeiten zu anderen Diensten oder Servern hat. Als Ziel wird hier der HTTP Dienst *wordpress.service.koecher.consul* auf dem Server *client2.node.koecher.consul* ausgewählt. Beide Server befinden sich als virtuelle Maschinen auf dem selben Host und sind über ein virtuelles Netzwerk miteinander verbunden. Die Kommunikation erfolgt dabei direkt über die jeweiligen Proxy Server der Systeme. Es ist also kein Mesh Gateway oder sonstige Komponenten involviert. Als Ziel auf dem HTTP Server wurde eine ca. 100 kB große, statische Grafik gewählt.

Um den Einfluss durch Latenzen bei der Namensauflösung des Hostnames auszuschließen, wird der Dienst direkt über seine IP adressiert. Im ersten Test wird die virtuelle Mesh-IP 240.0.0.55 verwendet, nach der Deaktivierung der Firewall-Richtlinien wir der Dienst über seine tatsächliche Netzwerkadresse *192.168.1.105* kontaktiert. Es wird ebenfalls der Einfluss von der Verwendung persistenter HTTP-Verbindungen betrachtet. Die Testergebnisse werden in folgenden Abbildungen 5.1.und 5.2 dargestellt.

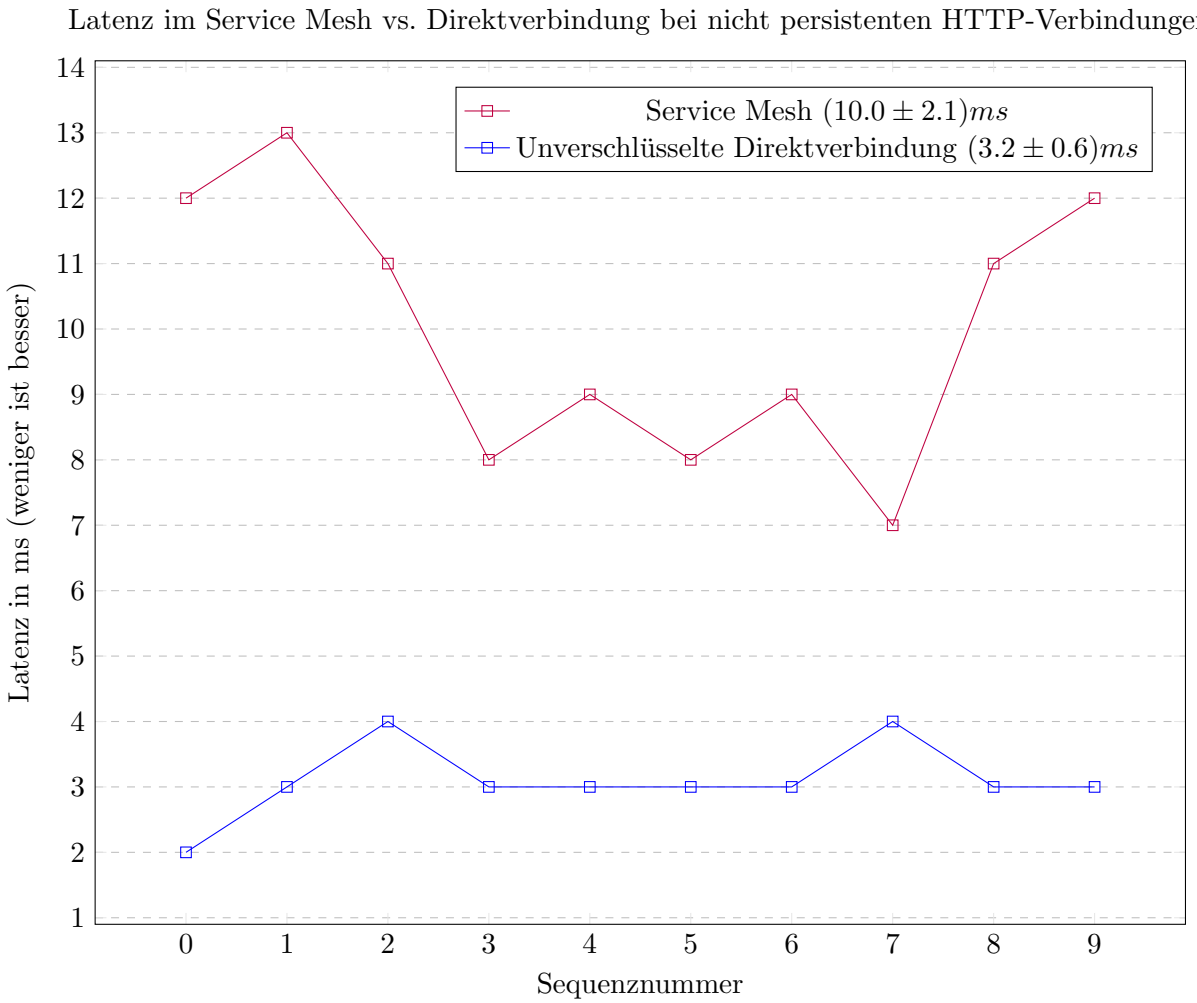

Abbildung 5.1: Latenz im Service Mesh vs. Direktverbindung bei nicht persistenten HTTP-Verbindungen

Die Abbildung zeigt die Latenz im Service Mesh vs. Direktverbindung bei nicht persistenten HTTP-Verbindungen. Verbindungen, die über das Service Mesh durchgeführt werden, zeigen eine mit circa 10 ms deutlich höhere Latenz im Vergleich zu Direktverbindungen.

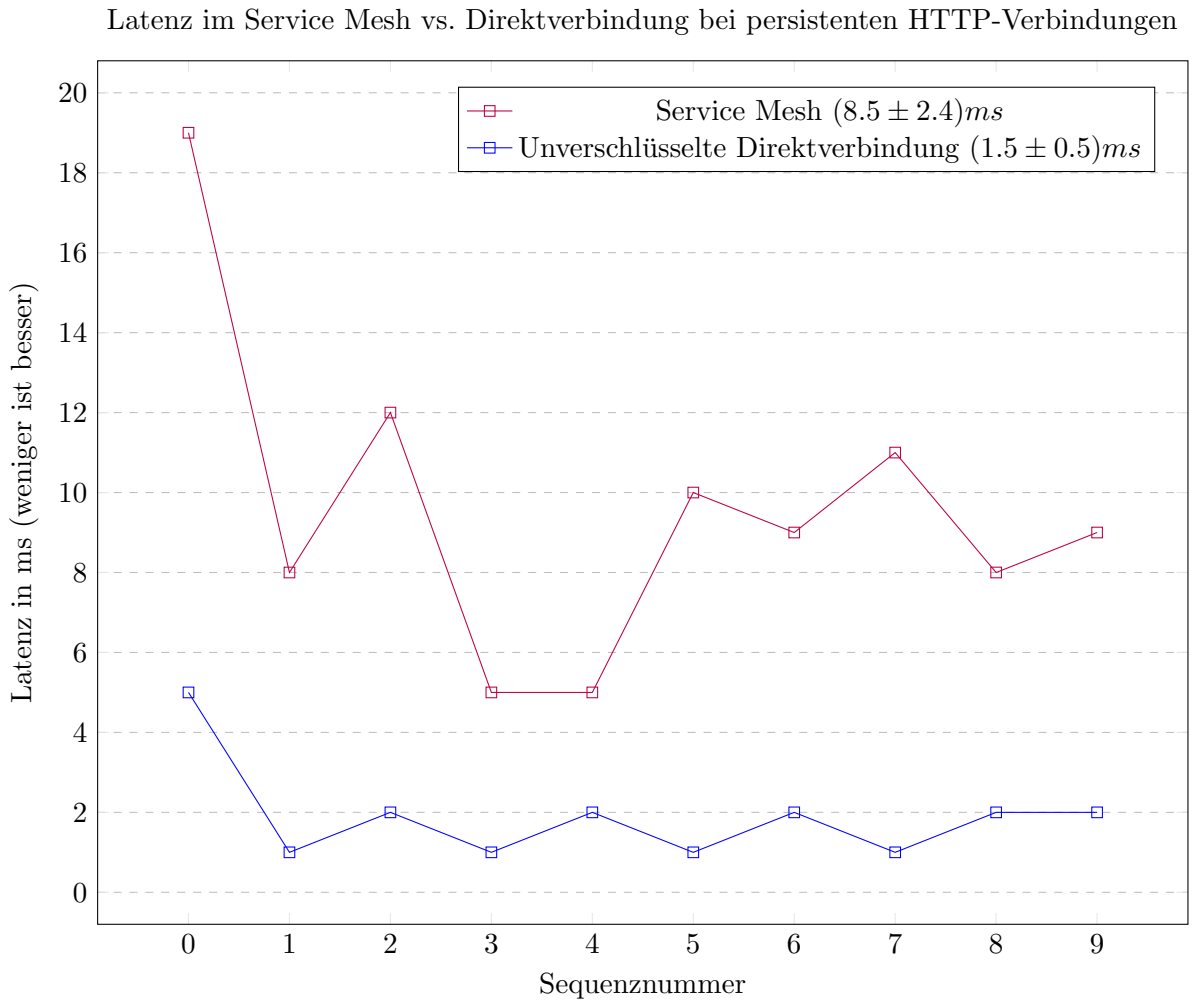

Abbildung 5.2: Latenz im Service Mesh vs. Direktverbindung bei persistenten HTTP-Verbindungen

Die Abbildung zeigt die Latenz im Service Mesh vs. Direktverbindung bei persistenten HTTP-Verbindungen. Auch bei der Verwendung von persistenten HTTP-Verbindungen weisen Verbindungen, die über das Service Mesh aufgebaut wurden, deutlich höhere Latenzen auf. Direktverbindungen profitieren am meisten von der persistenten Verbindung. Die Werte des initialen Verbindungsaufbaus wurden bei der Berechnung der Mittelwerte nicht berücksichtigt.

Die Testergebnisse zeigen eine drei mal höhere Latenz bei einer Verbindung durch das Service Mesh gegenüber einer unverschlüsselten Direktverbindung bei nicht persistenten HTTP Verbindungen. Die Latenz beschreibt den Zeitraum von Beginn der Verbindung, über den Transport der Nutzdaten bis zum Schließen der Verbindung. Persistente HTTP-Verbindungen liefern jeweils leicht bessere Ergebnisse. Bei beiden Methoden ist eine deutliche Spitze bei der ersten Messung zu erkennen. Diese Spitze wird durch den initialen Verbindungsaufbau verursacht, welcher bei der ersten Messung erfolgt. Im Gegensatz zum Verbindungsaufbau einer Direktverbindung, welche nur aus dem eigentlichen TCP-Verbindungsaufbau besteht, wird bei der Verbindung über das Service Mesh ein mTLS Tunnel aufgebaut. Der Tunnel wird solange aufrecht erhalten, wie auch die TCP-Verbindungen zwischen den Services und den zuständigen Proxy existiert.

5.2.2 SMB Durchsatz

Um den maximalen Netzwerkdurchsatz beim Dateitransfer zu ermitteln, soll mithilfe des SMB-Protokolls Dateien zwischen zwei Systemen kopiert werden. Als Ziel wird hier ein SMB-Dateiserver *samba.service.koecher.consul* auf dem Server *client2.node.koecher.consul* verwendet. Durchgeführt werden die Tests mit dem Kommandozeilen Werkzeug “dd“ [ubu22], welches es ermöglicht, flexibel maschinelle Kopiervorgänge durchzuführen. Der Test besteht aus der Übertragung von 100 MB Nutzdaten auf eine Freigabe des SMB-Dateiservers, zu diesem Zweck wird die Freigabe im Betriebssystemen eingebunden. Die 100 MB werden in Form unterschiedlich großer Dateien übertragen. Die Dateigrößen reichen dabei von 1 kB bis hin zu einer einzelnen 100 MB Datei. Im nachfolgenden Diagramm 5.3 sind die Datentransferraten, mit und ohne Service Mesh, abhängig von der Dateianzahl dargestellt.

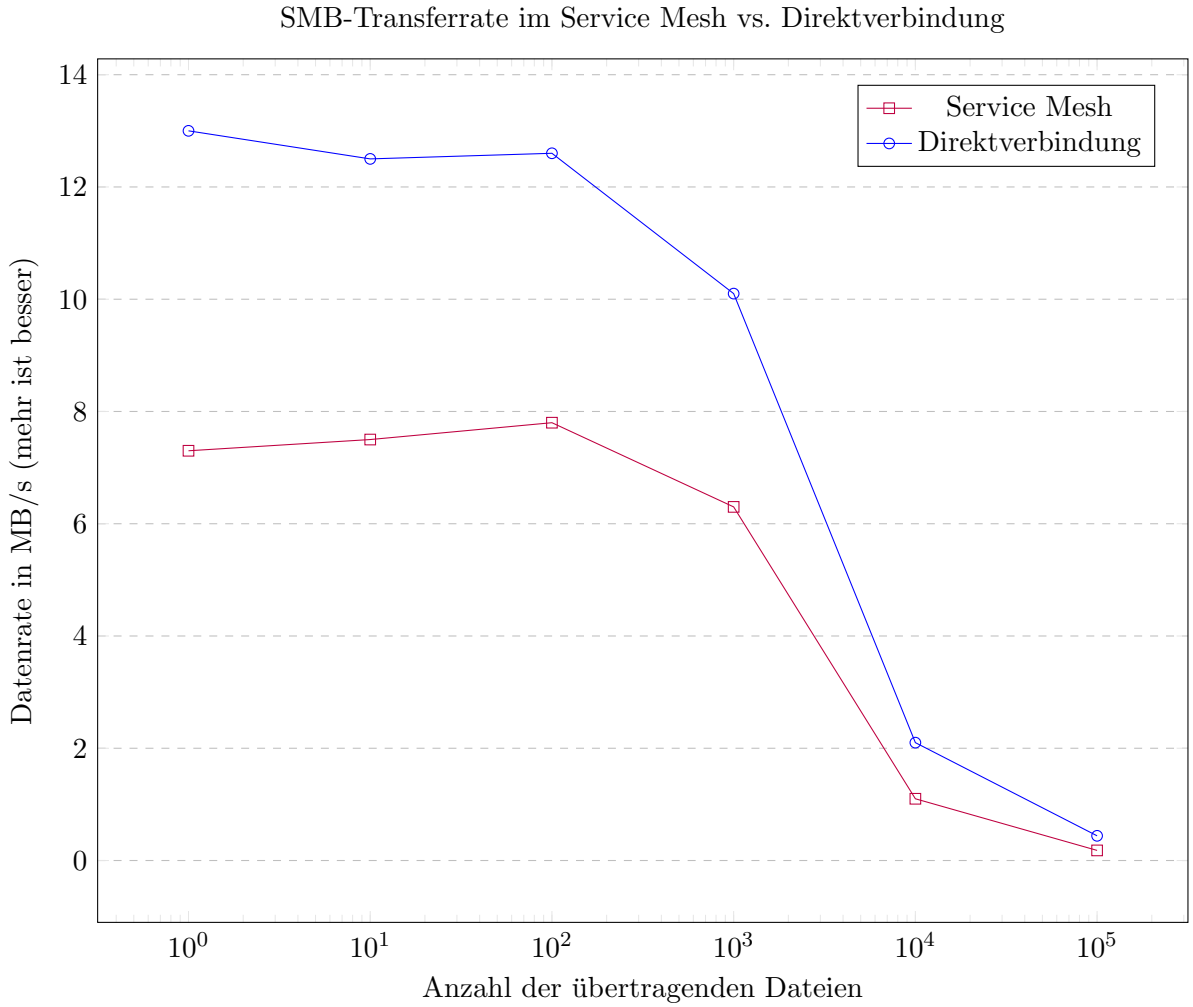

Abbildung 5.3: SMB Transferrate im Service Mesh vs. Direktverbindung

Die Abbildung zeigt die Transferrate eines Kopiervorgangs über das Netzwerk unter Verwendung des SMB Protokolls. Die effektive Übertragungsgeschwindigkeit ist abhängig von der Dateigröße und fällt bei beiden Versuchen mit zunehmender Dateianzahl. Zwei Grafen zeigen die Testergebnisse bei Verwendung des Service Meshs und einer Direktverbindung.

Die Testergebnisse zeigen einen durch das Service Mesh verursachen Performanceverlust von 43.8 % bei einer Dateigröße von 100 MB und 59.1 % bei einer Dateigröße von 1 KB. Die nachfolgende Grafik 5.4 zeigt das Leistungsverhältnis des Service Mesh gegenüber einer Direktverbindung abhängig von der Dateianzahl. Die Übertragungsrate der Direktverbindung wird hierzu als Referenzwert der schnellsten möglichen Übertragung angesehen.

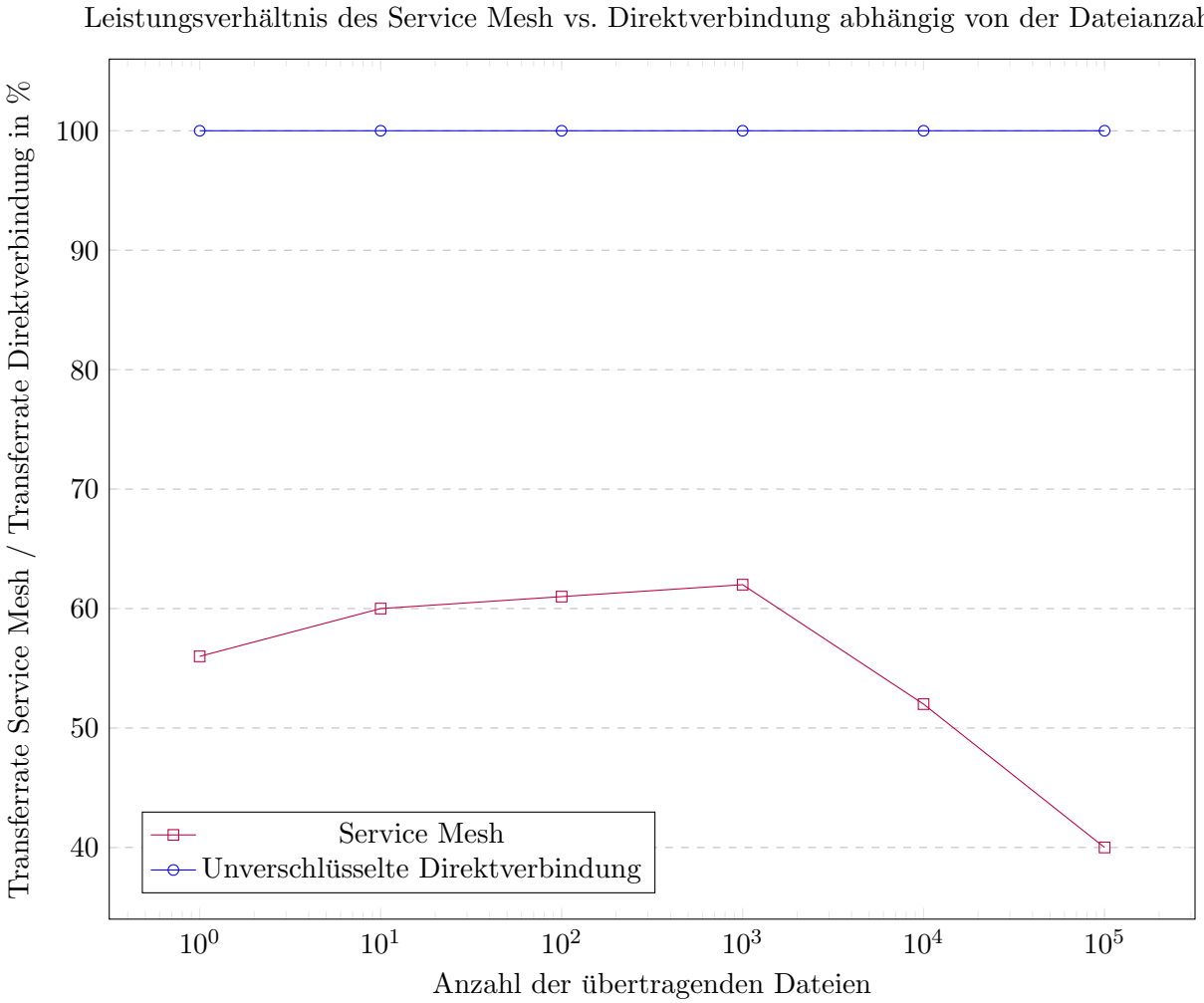

Abbildung 5.4: Leistungsverhältnis des Service Mesh gegenüber einer Direktverbindung abhängig von der Dateianzahl

Die Abbildung zeigt das Leistungsverhältnis des Service Mesh gegenüber einer Direktverbindung abhängig von der Dateianzahl.

Es ist zu erkennen, dass der Leistungsverlust durch das Service Mesh bis 10^2 übertragenen Daten konstant ist. Ab 10^3 übertragenen Daten ist eine Abnahme der Leistung zu beobachten.

Während der Tests bei der Übertragung über das Service Mesh, ließ sich eine hohe CPU-Auslastung des Client- und Server-Systems beobachten, die Last wird überwiegend von den Proxy Servern verursacht (siehe Abbildung 5.4). Es ist davon auszugehen, dass die maximale Übertragungsgeschwindigkeit beim Dateitransfer über das Service Mesh in

diesem Versuchsaufbau durch CPU-Ressourcen begrenzt wird. Eine direkte Übertragung ist deutlich effizienter als über das Service Mesh.

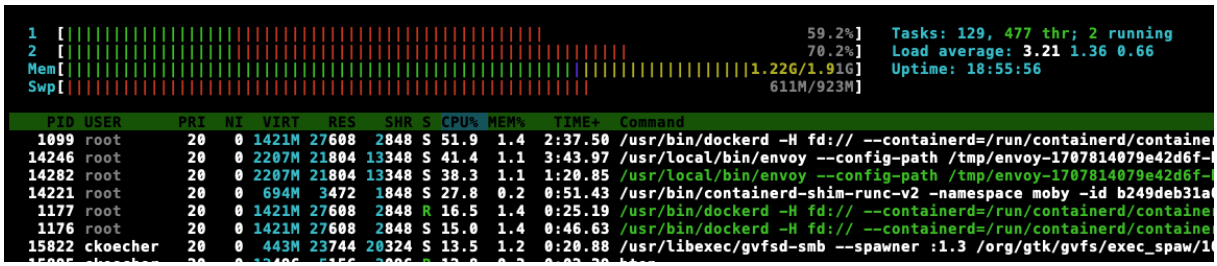

Abbildung 5.5: CPU-Auslastung bei einem SMB-Dateitransfer, verursacht durch den Envoy Proxy Server

Die Abbildung zeigt eine hohe CPU-Auslastung auf dem System *client2.node.koecher.consul*, welche bei einem SMB-Dateitransfer durch den Envoy Proxy Server verursacht wurde. Dargestellt werden die Informationen durch das Kommandozeilenwerkzeug “htop“.

5.3 Betrachtung von Angriffsvektoren mit der MITRE ATT&CK-Matrix

Um einordnen zu können, welchen Effekt der Einsatz einer ZTNA auf die verschiedenen Aspekte der IT-Sicherheit hat, wurde das MITRE ATT&CK Enterprise Framework [mit22b] verwendet. Das MITRE ATT&CK Enterprise Framework ist eine global zugängliche Datenbank für cyberkriminelle Taktiken und Techniken, die auf realen Beobachtungen der letzten Jahre basiert, ATT&CK steht als Akronym für Adversarial Tactics, Techniques (ATT), & Common Knowledge (CK). Die Datenbank untergliedert sich in drei Bereiche: Mobile Endgeräte (Mobile), Unternehmen (Enterprise) und industrielle Steuersysteme (ICS). Im folgenden wird die Enterprise Matrix näher betrachtet.
Die Matrix unterscheidet zwischen Taktiken und Techniken, wobei eine Taktik mehrerer Techniken zusammenfasst, mit der ein Angreifer ein bestimmtes Ziel erreichen kann. Im Folgendem werden aus der Enterprise Matrix insgesamt fünf Taktiken und deren Techniken betrachtet, die im Rahmen eines Angriffes zum Einsatz kommen. Bei der Auswahl der Taktiken und Techniken wurde darauf geachtet, das deren Angriffsvektor im Anwendungsbereich einer ZTNA liegt. Es soll gezeigt werden, gegen welche Angriffstechniken auf Netzwerkebene eine ZTNA Schutz bieten kann und gegen welche nicht. Dies wurde in Form eines Punktesystems mit eins bis fünf Punkten bewertet.

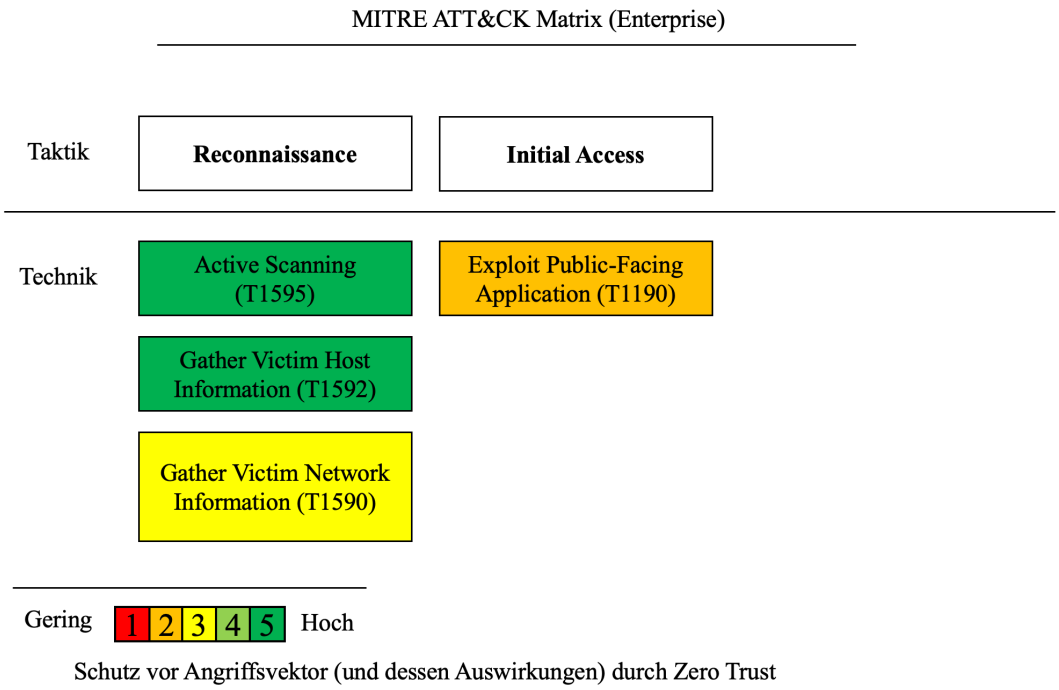

Abbildung 5.6: Auszug aus der MITRE ATT&CK Matrix mit den Taktiken Reconnaissance und Initial Access [mit22b]

Die Abbildung zeigt einen Auszug aus MITRE ATT&CK Matrix mit den Taktiken Reconnaissance und Initial Access. Die dazugehörigen Techniken wurde danach bewertet, wie gut der Einsatz einer ZTNA vor diesen schützt.

- **Aufklärung (Reconnaissance)**
  Die Taktik Aufklärung besteht aus Techniken, mit denen der Angreifer Informationen sammelt, die zur Unterstützung der Zielsetzung verwendet werden können. Das Sammeln der Informationen kann aktiv oder passiv erfolgen. Die Informationen können Einzelheiten über das Personal, die Organisation des Opfers oder dessen Infrastruktur beinhalten [mit22b]. Im Rahmen dieser Arbeit ist mit der Erkundung des Angriffsziels durch den Angreifer, die Erkundung auf technischer Ebene gemeint.

  Technik: Aktives Scannen (Active Scanning)
  Durch Aufklärungs-Scans sammelt ein Angreifer Informationen, die für ein gezieltes Vorgehen genutzt werden können. Bei aktiven Scans sondiert der Angreifer die Infrastruktur des Opfers über das Netzwerk durch direkte Interaktion [mit22b].

  Da durch entsprechende Firewallrichtlinien standardmäßig alle Ports der sich im Netzwerk befindlichen Systeme geschlossen sind, ist ein klassischer Portscan für den Angreifer nicht zielführend. Ein solcher Scan liefert lediglich Informationen darüber, wie viele Systeme es im Netzwerk gibt und das diese einen Proxy Server betreiben. Informationen über die Systeme selber oder die Services, die diese betreiben können, nicht erhoben werden. Die ZTNA wurde bei der Abwehr dieser Technik mit 5 von 5 Punkten bewertet.

Technik: Sammeln von Informationen über das Opfer (Gather Victim Host Information)
Der Angreifer kann Informationen über die Hosts des Opfers sammeln, die für gezielte Angriffe genutzt werden können. Die Informationen können eine Vielzahl von Details umfassen, dazu gehören unter anderem Angaben zum Betriebssystem, Name, IP-Adresse oder die Funktionalität [mit22b].

Der Einsatz einer ZTNA erschwert es einem Angreifer erheblich, Systeme innerhalb des Netzwerks zu suchen und zu analysieren. Da alle Ports des zu untersuchenden Systems geschlossen sind, ist es dem Angreifer nicht möglicher nähere Informationen über das eingesetzte Betriebssystem oder die darauf befindliche Software zu sammeln. Geräte, die nicht Teil des Services Mesh sind, können weiterhin gefunden und analysiert werden, dazu gehören beispielsweise Netzwerkkomponenten oder Hypervisor. Die ZTNA wurde bei der Abwehr dieser Technik mit 5 von 5 Punkten bewertet.

Technik: Sammeln von Informationen über das Opfernetzwerk (Gather Victim Network Information)
Der Angreifer kann Informationen über die Netze des Opfers sammeln, die für gezielte Angriffe genutzt werden können. Zu den Informationen gehören Angaben wie beispielsweise IP-Bereiche, Domänennamen oder die Netzwerk-Topologie [mit22b].

Das Mitscheiden und Analysieren des Netzwerkverkehrs liefert nur begrenzte Informationen über das Netzwerk, da sämtliche Daten ab Netzwerk-Layer-4 verschlüsselt sind. Informationen über die Topologie und die eingerichteten Netzwerk- bzw. IP-Bereiche können durch den Angreifer jedoch auch bei Einsatz einer ZTNA gesammelt werden. Da eine ZTNA diese Technik nur teilweise unterbinden kann, wurde deren Effektiv zur Prävention der vorgestellten Technik nur mit 3 von 5 Punkten bewertet.

- **Erster Zugriff (Initial Access)**
  Diese Taktik beschreibt Techniken mit der ein Angreifer versucht, durch die Nutzung verschiedener Zugangsvektoren in ein Netzwerk einzudringen [mit22b].

  Technik: Angriff öffentlich zugänglicher Anwendungen (Exploit Public-Facing Application)
  Der Angreifer versucht eine Schwachstelle in einem dem Internet zugewandten Computer oder Programm auszunutzen, um ein unbeabsichtigtes oder unvorhergesehenes Verhalten hervorzurufen. Bei der Schwachstelle im System kann es sich um einen Fehler, eine Störung oder eine Konstruktionsschwachstelle handeln [mit22b].

  Eine ZTNA kann keinen Schutz gegen Angriffe bieten, welche sich gegen öffentlich zugängliche Anwendungen richten. Weist eine dieser Anwendungen eine Schwachstelle auf, kann diese durch den Angreifer ausgenutzt werden, um die Kontrolle über diese oder das darunter liegende System zu erlangen. Eine ZTNA kann an dieser Stelle

dabei helfen, den Einbruch zu erkennen und die weitere Ausbreitung zu unterbinden. Durch detailliertes Logging der Verbindungsversuche auf anderen Maschinen, können Ausbreitungsversuche und Netzwerkanalysen durch den Angreifer, die von dem infizierten System ausgehen, erkannt werden. Auch wenn eine ZTNA keinen Schutz vor der vorgestellten Technik bietet, wurde diese mit 2 von 5 Punkten bewertet, da theoretisch eine Erkennung und Aufklärung des Angriffs möglich ist.

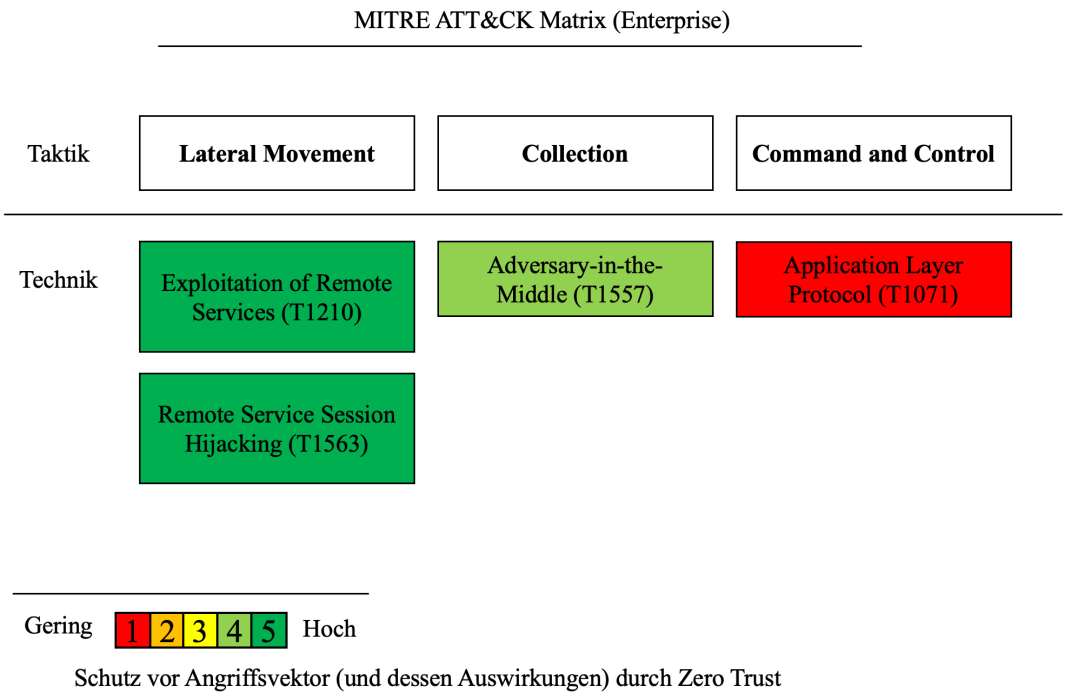

Abbildung 5.7: Auszug aus der MITRE ATT&CK Matrix mit den Taktiken Lateral Movement, Collection und Command and Control [mit22b]

Die Abbildung zeigt einen Auszug aus MITRE ATT&CK Matrix mit den Taktiken Lateral Movement, Collection und Command and Control. Die zugehörigen Techniken wurde danach bewertet, wie gut der Einsatz einer ZTNA vor diesen schützt.

- **Horizontale Ausbreitung (Lateral Movement)**
  Lateral Movement wurde in Kapitel 2.1.4 bereits ausgiebig beschrieben.

  Technik: Angriff auf Netzwerkdienste (Exploitation of Remote Services)
  Durch das Ausnutzen eines Remote-Dienstes, kann sich der Angreifer unerlaubten Zugang zu internen Systemen zu verschaffen, sobald sie sich in einem Netzwerk befinden. Der Angreifer nutzt einen Programmierfehler in einem Programm, einem Dienst oder in der Betriebssystemsoftware bzw. im Kernel selbst, um eigenen Code auszuführen [mit22b].

  Ein Hauptgrund zur Einführung einer ZTNA ist die Unterbindung von Lateral Movement. Durch die Abschottung aller Systeme vom eigentlichen Netzwerk und

die Verschlüsselung des gesamten Datenverkehrs, wird es dem Angreifer erschwert, andere Systeme in selben Netzwerk anzugreifen. Weißt beispielsweise ein älterer Dienst auf einem System bekannte Schwachstellen auf, stellt dies in der Regel ein großes Sicherheitsrisiko dar. Eine ZTNA verhindert den direkten Zugriff auf den Dienst und somit die Ausnutzung von dessen Schwachstellen durch den Angreifer. Die ZTNA wurde bei der Abwehr dieser Technik mit 5 von 5 Punkten bewertet.

Technik: Übernhame von Netzwerksitzungen (Remote Service Session Hijacking)
Um sich seitlich in einer Umgebung zu bewegen, kann der Angreifer die Kontrolle über bereits bestehende Sitzungen mit Remote-Diensten übernehmen. Zu diesen Diensten zählen Beispielsweise Anwendungen wie Telnet, SSH und RDP [mit22b].

Anwendungsprotokolle, welche über keine oder eine nur eine schwache Verschlüsselung verfügen, sind anfällig für diese Art von Angriffen. Der durch die Proxy Server aufgebaute mTLS-Tunnel verhindert das Mitlesen oder eine Manipulation des Anwendungsprotokolls durch den Angreifer. Die ZTNA wurde bei der Abwehr dieser Technik mit 5 von 5 Punkten bewertet.

- **Sammeln von Daten (Collection)**
  Diese Taktik besteht aus Techniken, die ein Angreifer nutzen kann, um für ihn relevante Informationen im Netzwerk des Opfers zu finden und zu stehlen [mit22b].

  Technik: Eingriff in den Netzwerkverkehr (Adversary-in-the-Middle)
  Bei der Adversary-in-the-Middle-Technik (AiTM) positioniert sich der Angreifer zwischen zwei oder mehreren vernetzten Geräten. Dies kann durch den Missbrauch von Funktionen gängiger Netzwerkprotokolle, die den Fluss des Netzwerkverkehrs bestimmen können wie beispielsweise ARP oder DNS, erreicht werden. Der Angreifer ist nun in der Lage den Netzwerkverkehr mitzulesen oder zu manipulieren [mit22b].

  Eine ZTNA schützt, ungeschützte Dienste und andere Datenquellen in einem Netzwerk gegen unbefugte Zugriffe. Durch die Verschlüsselung des Datenverkehrs ist es einem Angreifer nicht möglich, Datenverkehr mitzulesen oder zu manipulieren. Es ist zu beachten, dass eine ZTNA keine Sicherheit auf den Layern 1-3 liefern kann, Angriffe wie beispielsweise ARP- und IP-Spoofing erfordern weitere Sicherheitsmechanismen. Die ZTNA wurde bei der Abwehr dieser Technik mit 4 von 5 Punkten bewertet.

- **Steuerung und Kontrolle (Command and Control)**
  Command and Control besteht aus Techniken, mit denen der Angreifer mit einem durch ihn übernommenes System im Zielnetzwerk kommuniziert. Der Angreifer versucht in der Regel, den normalen, erwarteten Datenverkehr zu imitieren, um eine

Entdeckung zu vermeiden [mit22b].

Technik: Protokoll der Anwendungsschicht (Application Layer Protocol)
Befehle und deren Ergebnisse an das entfernte System werden in den Protokollverkehr zwischen Client und Server eingebettet, um die Erkennung/Netzwerkfilterung zu umgehen [mit22b].

Bei einer ZTNA, wie sie in dieser Arbeit umgesetzt wurde, wird keine Beschränkung des ausgehenden Datenverkehrs durchführt. Daher ist es für einen Angreifer ohne weiteres möglich, eine Verbindung zu einem externen Command and Control Server aufzubauen. Auch eine Kommunikation über legitime Anwendungsprotokolle zu Zeilen, die im Service Mesh freigeschaltet wurden, ist ohne weiteres möglich. Da eine ZTNA keinen Schutz im Bezug auf diese Technik bietet, wurden an dieser Stelle 0 von 5 Punkten vergeben.

5.4 Bekannte Schwachstelle in Consul

Auch wenn eine ZTNA und dessen Implementierung mittels Consul großes Augenmerk auf Sicherheit legt, kann auch diese durch fehlerhafte Programmierung oder Konfiguration eine Angriffsfläche bieten. Zur Analyse wurde das Penetrationstest-Tool Metasploit [met22] hinzugezogen. Dieses bietet Informationen zu Sicherheitslücken zahlreicher Systeme und wird zur Entwicklung und Ausführung von Angriffen verwendet. Consul weist hier eine bekannte Sicherheitslücken auf, welche durch eine unachtsame Konfiguration des Consul Servers verursacht wird. Beide Schwachstellen erlauben bei erfolgreicher Ausführung den Fernzugriff auf den Consul Server unter dem Benutzer, mit dem der Consul Server gestartet wurde [MAK]. Ein direkter Zugriff auf den Consul Server stellt einen schweren Bruch der Sicherheit des Service Mesh dar. Durch den Zugriff auf das X.509 Zertifikat und den privaten Schlüssel des Servers, kann sich der Angreifer Zugriff auf alle Dienste des Service Mesh verschaffen.

Voraussetzung für diesen Angriff ist, dass der Angreifer bereits einen Server im Rechenzentrum kompromittiert hat und in der Konfiguration des Consul Service Mesh die Option zur Ausführung externer Skripte aktiviert wurde. Eine Aufgabe des Service Mesh ist es, die Verfügbarkeit der Services zu überwachen, um im Fehlerfall den Service aus dem Mesh zu entfernen und Anfragen auf andere Instanzen des Services umzuleiten. Diese Service Checks werden bei der Konfiguration des Services durch den Administrator definiert. Hierbei stehen Funktionen zur Überprüfung der Verfügbarkeit von Netzwerk-Ports, Anwendungsprotokollen, dem Status von Diensten oder auch eigen definierbare Skripte zur Verfügung. Wird nun ein Anwendungsserver beispielsweise durch eine Schwachstelle in der Anwendung selbst kompromittiert, ist der Angreifer in der Lage einen der lokalen Dienste des Systems

zu bearbeiten und anschließend mit den vorhandenen Berechtigungen des Consul Agents im Service Mesh neu zu registrieren. Wird nun der Service Check des Services durch Schadcode ersetzt, wird dieser bei nächster Ausführung des Checks durch den Consul Server gestartet und kann zum Beispiel eine Hintertür zur ausführenden Instanz öffnen. Metasploit automatisiert diesen Angriff und fügt als Schadcode die Angriffs-Nutzlast "Meterpreter" ein, diese stellt eine TCP-Verbindung zum Angreifer her und schafft so Fernzugriff auf den Server mittels Rückkanal. Dieser Angriff konnte im aufgebauten Cluster erfolgreich demonstriert und anschließend durch Umkonfiguration des Servers abgewehrt werden. In neuen Versionen ist die Option für Script Checks aus diesen Gründen Standartmaßgig deaktiviert und es wird seitens des Herstellers eindringlich vor der Aktivierung gewarnt. Als Alternative bietet Consul die Möglichkeit, nur Skripte für Checks zu erlauben, welche sich lokal auf dem Server befinden.

Die Abbildung 5.8 zeigt den beschrieben Angriff gegen den Consul Server durch Metasploit. Nachdem innerhalb von Metasploit das entsprechende Modul ausgewählt wurde, wird dieses für den Angriff vorbereitet. Dazu wird der Parameter *RHOSTS* auf die IP-Adresse des anzugreifenden Consul Servers konfiguriert. Der Parameter *LHOSTS* definiert die IP-Adresse des angreifenden Systems, dies ist die Adresse zu welcher der Rückkanal aufgebaut wird. Anschließend wird die Software durch das Kommando *exploit* gestartet. Die beschrieben Schritte werden nach dem Start von Metasploit mithilfe folgender Befehle durchgeführt:

```
$ use multi/misc/consul_service_exec
$ set RHOSTS 192.168.1.176
$ set LHOSTS 192.168.1.165
$ exploit
```

Die Ausgabe zeigt wie ein Service mit dem zufälligen Namen "tIdIC" im Service Mesh registriert wird. Anschließend wird gewartet, bis der Service Check den eingefügten Code ausführt. Der Schadcode startet Meterpreter und öffnet somit den Rückkanal. Um den Angriff möglichst unauffällig zu gestalten, wird nach der Ausführung des Schadcodes der Service "tIdIC" automatisch aus dem Mesh entfernt. Dieser ist für ca. 3 Sekunden im Consul Service Dashboard sichtbar. Der Angreifer hat nun die Kontrolle über den Consul Server und ist beispielsweise in der Lage den aktuellen Verzeichnisinhalt auszulesen, zu kopieren oder zu manipulieren.

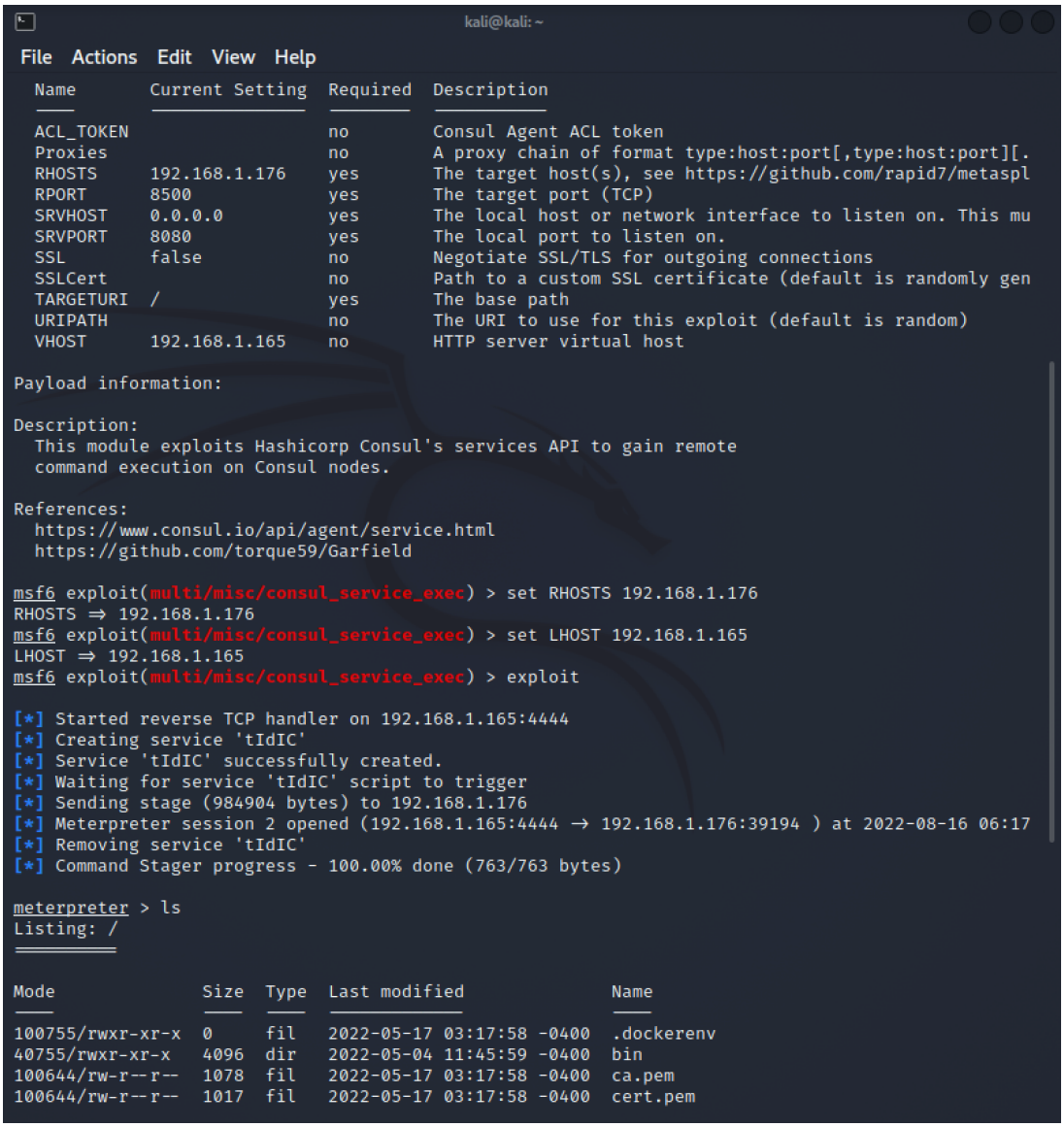

Abbildung 5.8: Durchführung eines Angriffes auf den Consul Server mit dem Penetrationstests Tool Metasploit

Die Abbildung zeigt die Durchführung eines Angriffes auf den Consul Server mit dem Penetrationstest Tool Metasploit. Neben der Vorbereitung und Konfiguration des Angriffes, wird auch dargestellt wie der Verzeichnisinhalt der übernommenen Maschine ausgelesen werden kann.

5.5 Überwachung und Logging

Durch die in den Firewallrichtlinien konfigurierte Logging Funktionalität, ist es möglich Verbindungsversuche, die von der Firewall geblockt wurden, zu protokollieren. Dies betrifft alle Verbindungsversuche, die sich nicht an Port 21 (SSH), Port 8301 (Consul Agent) oder Port 21000:21255 (Lokale Instanzen den Envoy Proxy Servers) richten. Dies kann nicht nur bei der Fehlersuche von großem Nutzen sein, sondern auch durch entsprechende Auswertung zur Erkennung von potentiellen Angriffen verwendet werden. Beispielsweise verursacht ein mit dem Kommandozeilenwerkzeug “nmap“ durchgeführter Portscan eine hohe Anzahl

abgelehnter Verbindungsversuche in kurzer Zeit. Die Protokolleinträge beinhalten unter anderem Informationen über die Quelladresse, den Quell- und Ziel-Port und das verwendete Protokoll. Diese Einträge können ein guter Indikator für auffällige Aktivität im Netzwerk sein, absolut verlässlich ist dieser jedoch nicht. So kann beispielsweise durch die Verwendung des Parameters "-sI" ein sogenannter "Idle-Scan" durchgeführt werden. Welche andere Systeme nutzt um zu überprüfen, ob ein bestimmen TCP-Port des Zielsystems offen ist [nma22]. Dadurch wird der eigentliche Ursprung des Scans im Netzwerk verschleiert. Gespeichert werden die Einträge im Systemprotokoll des Linux-Kernels, diese können über folgenden Befehl ausgegeben werden:

```
tail -f /var/log/messages
```

Nachfolgend ist ein Eintrag des Systems "client1.koecher.local" dargestellt, gegen welches ein Portscan von System "client2.koecher.local" aus durchgeführt wird.

```
Aug  3 10:13:10 client1 kernel: [ 4112.662591] DROPPED INPUT: IN=ens33 OUT= MAC=...
↪  SRC=192.168.1.105 DST=192.168.1.177 LEN=64 TOS=0x00 PREC=0x00 TTL=255 ID=0 DF
↪  PROTO=TCP SPT=58758 DPT=113 WINDOW=65535 RES=0x00 CWR ECE SYN URGP=0
```

5.6 ZTNA vs. Perimeter-Modell

Die Entscheidung, ob man sich für die Implementierung einer ZTNA mittels Consuls entscheiden sollte, hängt dabei stark vom individuellen Anwendungsfall ab. Im Folgenden werden die Vor- und Nachteile betrachtet, um im Anschluss eine eigene Einschätzung treffen zu können, für welche Anwendungsfälle der Einsatz einer ZTNA zielführend ist.

Nachteile gegenüber dem Perimeter Modell

- Der Betrieb einer ZTNA mittels Consul erhöht die Komplexität der IT-Infrastruktur. Neben den Anwendungen selber muss nun auch die Consul Kontrollschicht und die Instanzen der Envoy Proxy Server administriert werden.
- Die gesteigerte Komplexität der IT-Infrastruktur macht nicht nur deren Administrator sondern auch deren Betrieb selber ressourcenintensiver. Dies kann zu einer erhöhten Prozessorauslastung und zu einem Leistungsverlust der Netzwerk Übertragungsgeschwindigkeit von 50 % führen, wie der Test in Kapitel 5.2.2 zeigt.
- Der Einsatz von Consul stellt eine weitere mögliche Fehlerquelle dar. Durch teils unklare Fehlermeldungen, vor allem auf der Nutzdatenschicht, wird die Fehlersuche und Behebung erschwert.
- Consul ist nicht für alle Anwendungsfälle geeignet. Wie Kapitel 5.1.1 zeigt, sind nicht alle Anwendungen mit dem Service Mesh kompatibel. Auch der Einsatz auf Microsoft

Windows Plattformen ist nicht ohne weiteres möglich und seitens des Herstellers unzureichend dokumentiert.

Vorteile gegenüber dem Perimeter Modell

- Der Hauptgrund zum Einsatz von Consul ist die gesteigerte Sicherheit des Netzwerkes. Wie in Kapitel 5.3 betrachtet, bietet eine ZTNA effektiven Schutz gegen bestimmte Techniken.
- Die Funktionalitäten des Services Mesh können bei komplexeren Service Strukturen oder bei der Anbindung externer Rechenzentren und Services essenziell sein. Durch entsprechende Konfiguration lassen sich Funktionalitäten wie eine automatische Lastverteilung oder eine Ausfallsicherheit bestimmter Services realisieren.
- Eine Service Mesh kann dabei helfen, die sich veränderten Anforderungen an eine flexiblere und offenere IT-Infrastruktur zu realisieren.

Bei Infrastrukturen, die überwiegend auf Unix Systemen oder einer containerbasiserten Umgebungen besteht und standortübergreifende Konnektivität zwischen den Services benötigt, ist der Einsatz einer ZTNA empfehlenswert. Wird eine hohe Anzahl an verschiedenen Services zur Verfügung gestellt oder sind einige dieser Services öffentlich zugänglich, sollte ebenfalls eine ZTNA eingesetzt werden. Wie in Kapitel 5.2 gezeigt wird, bietet diese nur gegen bestimmte Taktiken eine effektiven Schutz, andere Taktiken erfordern weitere präventive Maßnahmen. Eine ZTNA ist daher nur als Teil des Schutzkonzeptes zu sehen und sollte nicht als Ersatz für andere Sicherheitsmaßnahmen angesehen werden. Auch der hybride Einsatz innerhalb einer bestehenden Infrastruktur für besonders schützenswerte Systeme unter dem Perimeter-Modell ist denkbar. Es ist ebenfalls zu beachten, dass eine Implementierung einer ZTNA sowie alle Sicherheitsmaßnahmen im Allgemeinen, keine absolute Sicherheit garantieren kann.

5.7 Ähnliche Anwendungsbereiche

Das in diesem Versuchsaufbau zu Testzwecken verwendete Desktop-System, bedarf einer aufwendigen Integration in das Consul Service Mesh, der Anpassung lokaler Firewall-Regeln und den Betrieb mehrere Dienste. An dieser Stelle sind vereinfachte und für den Desktop-Bereich optimierte Lösungen erwähnenswert. Durch diese sind ebenfalls gesicherte Punkt-zu-Punkt Verbindungen zwischen einem Anwenderrechner und den entsprechenden Diensten im Rechenzentrum möglich. Tools wie beispielsweise strongDM [str22], Teleport [got22] oder HashiCorp Boundary [bou22] stellen hier nennenswerte Implementierungen dar. Im Rahmen dieser Arbeit wurde zu Testzwecken das Produkt Boundary der Firma HashiCorp installiert und getestet. Das System besteht aus einer für alle gängigen Betriebssysteme verfügbaren Desktop-Anwendung, die auf dem Client-Rechner installiert wird und die Auswahl und Verbindung zu den Diensten ermöglicht. Technisch lässt sich Boundary als vereinfachte und spezialisierte Implementierung von Consul beschreiben, seitens des Servers wird hier ebenfalls zwischen Steuer- und Nutzdatenschicht unterschieden. Zur Kommunikation der Control Plane zwischen dem Desktop-Client und dem zuständigen Server kommt HTTPS zum Einsatz. Zur Übertragung der Nutzdaten wird, wie bei Consul, ein mTLS-Tunnel verwendet. Auf den Einsatz eines externen Envoy Sidecar Proxy wird hierbei verzichtet, dieser ist bereits in der Anwendung integriert.

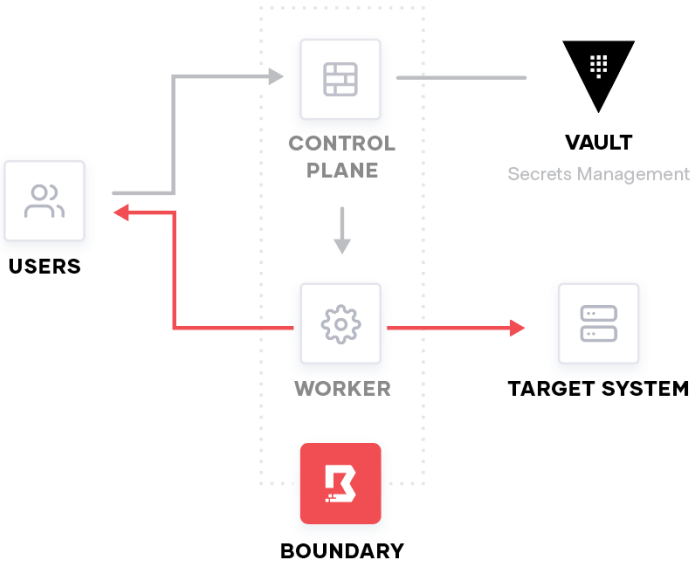

Abbildung 5.9: HashiCorp Boundary Systemstruktur - [bou22]

Die Abbildung zeigt die HashiCorp Boundary Systemstruktur bestehenden aus der Kontroll- und der Nutzdatenschicht.

Nach dem Starten und erfolgreicher Anmeldung der Client-Anwendung durch den Benutzer hat dieser die Möglichkeit, aus einem Katalog der für ihn freigeschalteten Dienste zu wählen. Soll eine Verbindung zu einem Dienst hergestellt werden, wird durch die Anwendung ein

lokaler Port auf dem Client Rechner gebunden und der mTLS-Tunnel zum Zielserver initialisiert. Die Kommunikation mit dem Dienst erfolgt dann über das lokale Loopback-Interface (127.0.0.1) und dem zugewiesenen Port, Boundary fungiert somit als Tunnel für den Datenverkehr um diesen zu verschlüsseln. Das praktische Einsatzgebiet für diese Lösung ist jedoch sehr beschränkt und ist überwiegend für Administratoren oder IT-Affine Personen interessant. Da die Verbindungen zu den Diensten manuell initiiert werden müssen und sich die lokalen Ports bei jeder Verbindung ändern, erfordert dies eine ständige Anpassung der Konfiguration lokaler Anwendungen/Dienste, welche über Boundary mit ihrem Server kommunizieren.

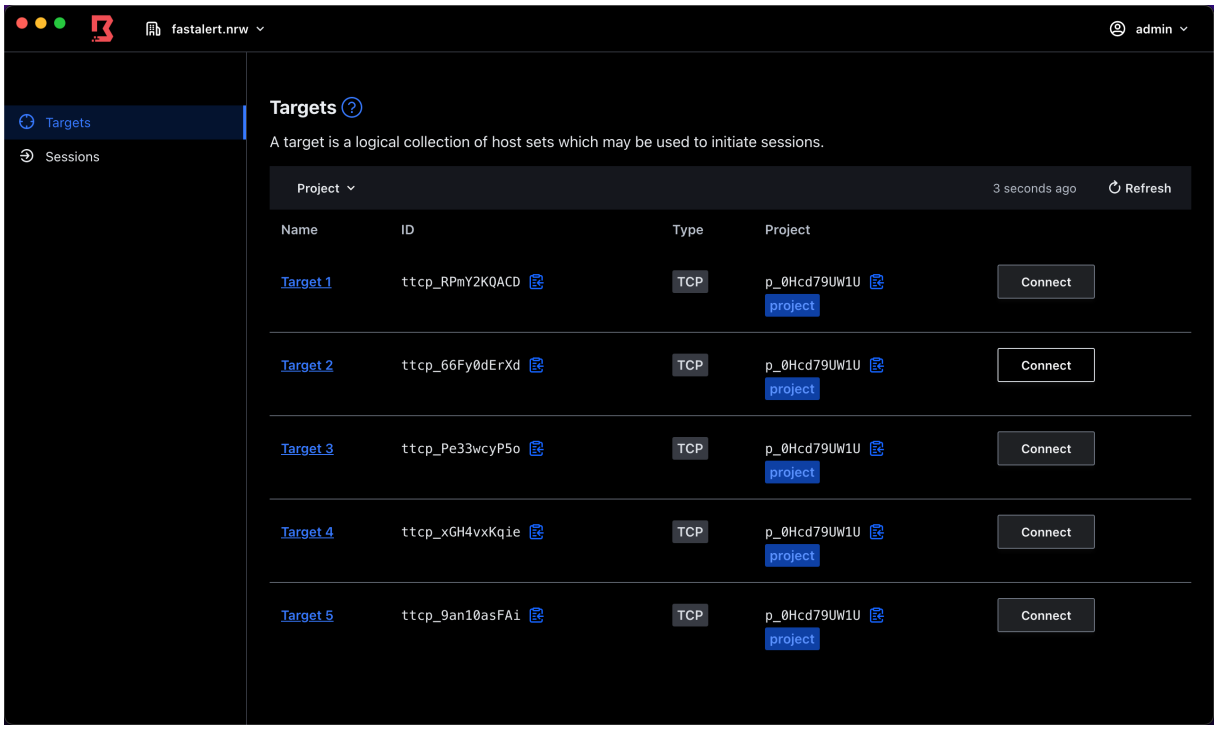

Abbildung 5.10: HashiCorp Boundary Anwendung - [bou22]

Die Abbildung zeigt die HashiCorp Boundary Anwendung, welche eine Auswahl an möglichen Zielen darstellt zu denen eine Verbindung aufgebaut werden kann.

6 Fazit und Ausblick

Die steigende Komplexität und Konnektivität von IT-Systemen schafft eine stetig wachsende Angriffsfläche. Durch gezieltere und aufwendigere Angriffstechniken ist das gängige Konzept zur Netzwerksicherheit zunehmend unzureichend. In dieser Arbeit wurden vorhandene Sicherheitsprobleme aufgezeigt und das Zero Trust Model als Lösungsansatz vorgestellt. Es wurde eine technische Implementierung des Modells beschrieben und in einer Testumgebung angewendet. Tests und Analysen haben administrative und sicherheitsrelevante Besonderheiten aufgezeigt, die es bei einer praktischen Anwendung zu beachten gilt. Durch die Betrachtung bekannter Angriffstechniken wurde dargestellt, in welchen Bereichen das Modell einen hohen Grad an Sicherheit bieten kann und wo andere Abwehrmechanismen eingesetzt werden müssen. Eine ZTNA erhöht die Komplexität des Netzwerkes und den damit verbunden administrativen Aufwand. Die Verarbeitung und Verschlüsselung des Datenverkehrs machen sich in einer geringeren Leistung des Netzwerks und einer gesteigerten Ressourcenauslastung der Systeme bemerkbar.

Anforderungen an offene Netzwerke und der Trend zu Cloud-Anwendungen schafft wachsende Anwendungsbereiche für eine ZTNA. Das Konzept wird in bestimmten Bereichen bestehende Perimeter basierte Netzwerkarchitekturen ersetzen können oder mit diesen gemeinsam existieren.
Wie in dieser Arbeit gezeigt wurde ist auch die Einbindung eines Desktop-Systems, welches transparent auf Ressourcen im Service Mesh zugreifen kann, realisierbar. Die Umsetzung gestaltet sich jedoch aufwendig und ist daher nicht im großen Maßstab realisierbar, auch eine Umsetzung auf Microsoft Windows Plattformen ist nicht ohne weiteres möglich. Hier wäre eine komfortable Desktopanwendung nach dem Vorbild von HashiCorp Boundary, die sich nahtlos in Consul integriert, wünschenswert.
Als Fortsetzung dieser Arbeit würde sich ein Projekt zur Übertragung und Auswertung der in Kapitel 5.5 vorgestellten Firewall Protokolldateien anbieten. Durch ein flächendeckendes Logging und eine automatisierte Echtzeitauswertung könnte ein Angriff schnell erkannt und das betroffene System bestimmt werden.

Anhang

1 Ausgabe HTTP Latenztests

```
TEST 1, MESH: ON, PERSIST: OFF

    $ httping http://240.0.0.55/wp-content/flight-path-on-transparent-d.png -Z

    connected to 240.0.0.55:80 (257 bytes), seq=0 time= 12.37 ms
    connected to 240.0.0.55:80 (257 bytes), seq=1 time= 13.72 ms
    connected to 240.0.0.55:80 (257 bytes), seq=2 time= 11.20 ms
    connected to 240.0.0.55:80 (257 bytes), seq=3 time=  8.08 ms
    connected to 240.0.0.55:80 (257 bytes), seq=4 time=  9.40 ms
    connected to 240.0.0.55:80 (257 bytes), seq=5 time=  8.67 ms
    connected to 240.0.0.55:80 (257 bytes), seq=6 time=  9.67 ms
    connected to 240.0.0.55:80 (257 bytes), seq=7 time=  7.56 ms
    connected to 240.0.0.55:80 (257 bytes), seq=8 time= 11.82 ms
    connected to 240.0.0.55:80 (257 bytes), seq=9 time= 12.34 ms

TEST 2, MESH: ON, PERSIST: ON

    $ httping http://240.0.0.55/wp-content/flight-path-on-transparent-d.png -Z -Q

    pinged host 240.0.0.55:80 (294 bytes), seq=0 time= 19.20 ms
    pinged host 240.0.0.55:80 (293 bytes), seq=1 time=  8.48 ms
    pinged host 240.0.0.55:80 (293 bytes), seq=2 time= 12.19 ms
    pinged host 240.0.0.55:80 (293 bytes), seq=3 time=  5.37 ms
    pinged host 240.0.0.55:80 (293 bytes), seq=4 time=  5.88 ms
    pinged host 240.0.0.55:80 (293 bytes), seq=5 time= 10.87 ms
    pinged host 240.0.0.55:80 (293 bytes), seq=6 time=  9.23 ms
    pinged host 240.0.0.55:80 (293 bytes), seq=7 time= 11.87 ms
    pinged host 240.0.0.55:80 (293 bytes), seq=8 time=  8.35 ms
    pinged host 240.0.0.55:80 (293 bytes), seq=9 time=  9.10 ms

TEST 3, MESH: OFF, PERSIST: OFF

    $ httping http://192.168.1.105/wp-content/flight-path-on-transparent-d.png -Z

    connected to 192.168.1.105:80 (257 bytes), seq=0 time=  2.32 ms
    connected to 192.168.1.105:80 (257 bytes), seq=1 time=  3.60 ms
    connected to 192.168.1.105:80 (257 bytes), seq=2 time=  4.93 ms
    connected to 192.168.1.105:80 (257 bytes), seq=3 time=  3.57 ms
```

```
    connected to 192.168.1.105:80 (257 bytes), seq=4 time=  3.78 ms
    connected to 192.168.1.105:80 (257 bytes), seq=5 time=  3.63 ms
    connected to 192.168.1.105:80 (257 bytes), seq=6 time=  3.49 ms
    connected to 192.168.1.105:80 (257 bytes), seq=7 time=  4.17 ms
    connected to 192.168.1.105:80 (257 bytes), seq=8 time=  3.48 ms
    connected to 192.168.1.105:80 (257 bytes), seq=9 time=  3.54 ms

TEST 4, MESH: OFF, PERSIST: ON

    $ httping http://192.168.1.105/wp-content/flight-path-on-transparent-d.png -Z -Q

    pinged host 192.168.1.105:80 (294 bytes), seq=0 time=  5.25 ms
    pinged host 192.168.1.105:80 (293 bytes), seq=1 time=  1.44 ms
    pinged host 192.168.1.105:80 (293 bytes), seq=2 time=  2.50 ms
    pinged host 192.168.1.105:80 (293 bytes), seq=3 time=  1.65 ms
    pinged host 192.168.1.105:80 (293 bytes), seq=4 time=  2.13 ms
    pinged host 192.168.1.105:80 (293 bytes), seq=5 time=  1.72 ms
    pinged host 192.168.1.105:80 (293 bytes), seq=6 time=  2.65 ms
    pinged host 192.168.1.105:80 (293 bytes), seq=7 time=  1.41 ms
    pinged host 192.168.1.105:80 (293 bytes), seq=8 time=  2.64 ms
    pinged host 192.168.1.105:80 (293 bytes), seq=9 time=  2.94 ms


    -Z   / --no-cache        ask any proxies on the way not to cache the requests
    -Q   / --persistent-connections use a persistent connection, i.e. reuse the same
    ↪  TCP connection for multiple HTTP requests. usually possible when 'Connection:
    ↪  Keep-Alive' is sent by server. adds a 'C' to the output if httping had to
    ↪  reconnect
```

2 Ausgabe SMB Durchsatz

```
TEST 1, MESH: ON

    dd if=/dev/zero of=testfile bs=100M count=1  status=progress
    104857600 bytes (105 MB, 100 MiB) copied, 14.3825 s, 7.3 MB/s

    dd if=/dev/zero of=testfile bs=10M  count=10 status=progress
    104857600 bytes (105 MB, 100 MiB) copied, 14.0312 s, 7.5 MB/s

    dd if=/dev/zero of=testfile bs=1M   count=100 status=progress
    104857600 bytes (105 MB, 100 MiB) copied, 13.4297 s, 7.8 MB/s

    dd if=/dev/zero of=testfile bs=100k count=1000 status=progress
    102400000 bytes (102 MB, 98 MiB) copied, 16.1485 s, 6.3 MB/s

    dd if=/dev/zero of=testfile bs=10k  count=10000 status=progress
    102400000 bytes (102 MB, 98 MiB) copied, 96.0154 s, 1.1 MB/s

    dd if=/dev/zero of=testfile bs=1k   count=100000 status=progress
    102400000 bytes (102 MB, 98 MiB) copied, 542.191 s, 189 kB/s


TEST 2, MESH: OFF

    dd if=/dev/zero of=testfile bs=100M count=1  status=progress
    104857600 bytes (105 MB, 100 MiB) copied, 8.0927 s, 13.0 MB/s

    dd if=/dev/zero of=testfile bs=10M  count=10 status=progress
    104857600 bytes (105 MB, 100 MiB) copied, 8.36165 s, 12.5 MB/s

    dd if=/dev/zero of=testfile bs=1M   count=100 status=progress
    104857600 bytes (105 MB, 100 MiB) copied, 8.3034 s, 12.6 MB/s

    dd if=/dev/zero of=testfile bs=100k count=1000 status=progress
    102400000 bytes (102 MB, 98 MiB) copied, 10.1628 s, 10.1 MB/s

    dd if=/dev/zero of=testfile bs=10k  count=10000 status=progress
    102400000 bytes (102 MB, 98 MiB) copied, 49.7848 s, 2.1 MB/s

    dd if=/dev/zero of=testfile bs=1k   count=100000 status=progress
    102400000 bytes (102 MB, 98 MiB) copied, 228.49 s, 448 kB/s
```

3 Dokumentation und Dateien

Dieser Bachelorarbeit liegt ein exFAT (MBR) formatierter USB-Stick bei. Der Inhalt ist folgender:

- **Ordner: Consul@server.koecher.local**
  Terraform Konfigurationsdateien für Consul auf der Maschine server.koecher.local
- **Ordner: Docker@server.node.koecher.consul**
  Terraform Konfigurationsdateien für Docker auf der Machine server.node.koecher.consul
- **Ordner: Docker@desktop.node.koecher.consul**
  Terraform Konfigurationsdateien für Docker auf der Machine desktop.node.koecher.consul
- **Ordner: Docker@client1.node.koecher.consul**
  Terraform Konfigurationsdateien für Docker auf der Machine client1.node.koecher.consul
- **Ordner: Docker@client2.node.koecher.consul**
  Terraform Konfigurationsdateien für Docker auf der Machine client2.node.koecher.consul
- **Ordner: Docker@client3.node.koecher.consul**
  Terraform Konfigurationsdateien für Docker auf der Machine client3.node.koecher.consul
- **Ordner: Docker@server.node.merkur.consul**
  Terraform Konfigurationsdateien für Docker auf der Machine server.node.merkur.consul
- **Ordner: Docker@server.node.venus.consul**
  Terraform Konfigurationsdateien für Docker auf der Machine server.node.venus.consul
- **Datei: IP-Tables-direct.txt**
  IP-Tables Konfigurationsdatei zum Betrieb des Envoy Proxy Servers im direkten Modus
- **Datei: IP-Tables-transparent.txt**
  IP-Tables Konfigurationsdatei zum Betrieb des Envoy Proxy Servers im transparenten Modus

Der Inhalt ist ebenfalls Online verfügbar:

https://github.com/calkoe/ZeroTrustConsul

Literaturverzeichnis

[aws22a] aws.amazon.com. „Was sind Microservices?“ In: (2022). URL: https://aws.amazon.com/de/microservices/ (besucht am 11. 08. 2022).

[aws22b] awsworkshop.io. „awsworkshop“. In: (2022). URL: https://hashicorp-terraform.awsworkshop.io/040_terraform_cloud_setup/1-infrastructure-as-code.html (besucht am 07. 08. 2022).

[Blu04] U. Blumenthal. „The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model“. In: (2004). URL: https://www.rfc-editor.org/rfc/rfc3826 (besucht am 17. 08. 2022).

[boo22a] bookstack.cn. „Ingress Gateways“. In: (2022). URL: https://www.bookstack.cn/read/consul-1.8.x-en/757c181e97d82c2b.md (besucht am 12. 08. 2022).

[boo22b] bookstack.cn. „Mesh Gateways“. In: (2022). URL: https://www.bookstack.cn/read/consul-1.8.x-en/47c6f90a6c2cdb78.md (besucht am 12. 08. 2022).

[bou22] boundaryproject.io. „Boundary“. In: (2022). URL: https://www.boundaryproject.io (besucht am 15. 07. 2022).

[Bre21] Fabian Brechlin. „Strukturierte Cloud Transformation in Unternehmen – Veränderungen durch Covid-19?“ In: (2021). URL: https://link.springer.com/article/10.1365/s40702-021-00742-y (besucht am 19. 07. 2022).

[bsi21] bsi.bund.de. „Mindeststandard des BSI zur Verwendung von Transport Layer Security“. In: (2021). URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Mindeststandards/Mindeststandard_BSI_TLS_Version_2_2.pdf?__blob=publicationFile&v=5 (besucht am 09. 08. 2022).

[bsi22] bsi.bund.de. „Seitenkanalresistenz“. In: (2022). URL: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Kryptografie/Seitenkanalresistenz/seitenkanalresistenz_node.html (besucht am 12. 08. 2022).

[cnc22] cncf.io. „CNCF“. In: (2022). URL: https://www.cncf.io (besucht am 12. 08. 2022).

[con22] consul.io. „Consul“. In: (2022). URL: https://www.consul.io (besucht am 02. 08. 2022).

[deb22a] debian.org. „Debian GNU/Linux“. In: (2022). URL: https://www.debian.org (besucht am 08. 08. 2022).

[deb22b] debian.org. „httping“. In: (2022). URL: https://manpages.debian.org/testing/httping/httping.1.en.html (besucht am 11. 08. 2022).

[dew22] dewiki.de. „Split Brain (Informatik)“. In: (2022). URL: https://dewiki.de/Lexikon/Split_Brain_(Informatik) (besucht am 15. 08. 2022).

[doc22a] docker.com. „Docker“. In: (2022). URL: https://www.docker.com (besucht am 03. 08. 2022).

[doc22b] docker.com. „Docker Setup“. In: (2022). URL: https://docs.docker.com/engine/install/debian/ (besucht am 14. 07. 2022).

[doc22c] docker.com. „DockerConsul“. In: (2022). URL: https://hub.docker.com/_/consul (besucht am 22. 07. 2022).

[E R18] Mozilla E. Rescorla. „The Transport Layer Security (TLS) Protocol Version 1.3“. In: (2018). URL: https://www.rfc-editor.org/rfc/rfc8446.html (besucht am 17. 08. 2022).

[E R99] RTFM Inc. E. Rescorla. „Diffie-Hellman Key Agreement Method“. In: (1999). URL: https://www.rfc-editor.org/rfc/rfc2631 (besucht am 17. 08. 2022).

[ele22a] elektronik-kompendium.de. „RSA - Rivest, Shamir und Adleman“. In: (2022). URL: https://www.elektronik-kompendium.de/sites/net/1910121.htm (besucht am 12. 08. 2022).

[ele22b] elektronik-kompendium.de. „TLS - Transport Layer Security“. In: (2022). URL: https://www.elektronik-kompendium.de/sites/net/1706131.htm (besucht am 12. 08. 2022).

[env22] envoyproxy.i. „Envoy“. In: (2022). URL: https://www.envoyproxy.io (besucht am 12. 08. 2022).

[fbi04] fbi.gov. „No Ordinary Case of Identity Theft - The Largest in U.S. History“. In: (2004). URL: https://archives.fbi.gov/archives/news/stories/2004/october/uncoveridt_101504 (besucht am 11. 08. 2022).

[GA22] Florian Gehm und Sebastian Artz. „Angriffsziel deutsche Wirtschaft: mehr als 220 Milliarden Euro Schaden pro Jahr“. In: (2022). URL: https://www.bitkom.org/Presse/Presseinformation/Angriffsziel-deutsche-Wirtschaft-mehr-als-220-Milliarden-Euro-Schaden-pro-Jahr (besucht am 11. 08. 2022).

[gar22] gartner.com. „Gartner Forecasts Worldwide Public Cloud End-User Spending to Reach Nearly 500 Billion in 2022“. In: (2022). URL: https://www.gartner.com/en/newsroom/press-releases/2022-04-19-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-reach-nearly-500-billion-in-2022 (besucht am 04. 08. 2022).

[GB17] Evan Gilman und Doug Barth. „Zero Trust Networks - Building Secure Systems in Untrusted Networks“. In: (2017).

[git22] git-scm.com. „Git“. In: (2022). URL: https://git-scm.com (besucht am 24. 07. 2022).

[god22] go.dev. „golang“. In: (2022). URL: https://go.dev (besucht am 16. 07. 2022).

[got22] goteleport.com. „Teleport“. In: (2022). URL: https://goteleport.com (besucht am 11. 08. 2022).

[gre22] greymatter.io. „Grey Matter“. In: (2022). URL: https://greymatter.io (besucht am 08. 08. 2022).

[Has22a] HashiCorp. „Consul Service Mesh“. In: (2022). URL: https://www.consul.io/docs/connect (besucht am 17. 07. 2022).

[Has22b] HashiCorp. „Consul vs. Istio“. In: (2022). URL: https://www.consul.io/docs/intro/vs/istio (besucht am 07. 07. 2022).

[Has22c] HashiCorp. „How Service Mesh Works“. In: (2022). URL: https://www.consul.io/docs/connect/connect-internals (besucht am 17. 07. 2022).

[Has22d] HashiCorp. „Life of a Packet Through Consul Service Mesh“. In: (2022). URL: https://www.datocms-assets.com/2885/1588705030-life-of-a-packet-consul-whitepaper-v10-digital.pdf (besucht am 07. 07. 2022).

[has22] hashicorp.com. „Hashicorp“. In: (2022). URL: https://www.hashicorp.com (besucht am 01. 08. 2022).

[Hub20] Nicole Hubbard. „HashiCorp Consul Introduction: What is a Service Mesh?“ In: (2020). URL: https://www.youtube.com/watch?v=UHLr8UsHuDA (besucht am 06. 08. 2022).

[ist22] istio.io. „Istio“. In: (2022). URL: https://istio.io (besucht am 08. 08. 2022).

[ITU19a] TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU. „The Directory: Overview of concepts, models and services“. In: (2019). URL: https://www.itu.int/rec/T-REC-X.500-201910-I/en (besucht am 17. 08. 2022).

[ITU19b] TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU. „The Directory: Public-key and attribute certificate frameworks“. In: (2019). URL: https://www.itu.int/rec/T-REC-X.509-201910-I/en (besucht am 17. 08. 2022).

[J P85] J. Reynolds J. Postel. „FILE TRANSFER PROTOCOL (FTP)“. In: (1985). URL: https://www.rfc-editor.org/rfc/rfc959 (besucht am 15. 08. 2022).

[Ker+20] Alper Kerman u. a. „IMPLEMENTING A ZERO TRUST ARCHITECTURE“. In: National Institute of Standards and Technology (Okt. 2020). URL: https://www.nccoe.nist.gov/sites/default/files/legacy-files/zta-project-description-final.pdf (besucht am 29. 07. 2022).

[Kin10a] John Kindervag. „Build Security Into Your Network's DNA: The Zero Trust Network Architecture“. In: Forrester Research Inc. (2010).

[Kin10b] John Kindervag. „No More Chewy Centers: Introducing The Zero Trust Model Of Information Security“. In: Forrester Research, Inc. (Sep. 2010).

[kub22] kubernetes.io. „Kubernetes“. In: (2022). URL: https://kubernetes.io/de/ (besucht am 26. 07. 2022).

[kum22] kuma.io. „Kuma“. In: (2022). URL: https://kuma.io (besucht am 08. 08. 2022).

[Law15] Michigan Consumer Credit Lawyers. „7 of the Largest Identity Theft Crimes“. In: (2015). URL: https://micreditlawyer.com/7-of-the-largest-identity-theft-crimes/ (besucht am 11. 08. 2022).

[lin22] linkerd.io. „Linkerd“. In: (2022). URL: https://linkerd.io (besucht am 09. 08. 2022).

[LL22] Michael Lang und Hans Löhr. „IT-Sicherheit: Technologien und Best Practices für die Umsetzung im Unternehmen“. In: (2022). URL: https://books.google.de/books?hl=de&lr=&id=6hJyEAAAQBAJ&oi=fnd&pg=PR13&dq=IT-Sicherheit&ots=1RtIuSToLR&sig=D4vQUp6woS0NLG-ws2epDO0ETZ8&redir_esc=y#v=onepage&q=IT-Sicherheit&f=false (besucht am 11. 08. 2022).

[MAK] Bharadwaj Machiraju, Francis Alexander und Quentin Kaiser. „Hashicorp Consul - Remote Command Execution via Services API (Metasploit)“. In: (). URL: https://www.exploit-db.com/exploits/46074 (besucht am 09. 08. 2022).

[Man22] Sachin Manpathak. „Kubernetes Service Mesh: A Comparison of Istio, Linkerd, and Consul“. In: (2022). URL: https://platform9.com/blog/kubernetes-service-mesh-a-comparison-of-istio-linkerd-and-consul/ (besucht am 08. 08. 2022).

[Mar20] Marcus Marohn. „Leitfaden zur Eignung von Service Meshes in cloudnativen Anwendungen“. In: (2020).

[met22] metasploit.com. „Metasploit“. In: (2022). URL: https://www.metasploit.com (besucht am 08. 07. 2022).

[mit22a] mitre.org. „Common Vulnerabilities and Exposures MITRE Database“. In: (2022). URL: https://cve.mitre.org/index.html (besucht am 09. 08. 2022).

[mit22b] mitre.org. „MITRE ATT&CK“. In: (2022). URL: https://attack.mitre.org/matrices/enterprise/ (besucht am 08. 08. 2022).

[nma22] nmap.org. „TCP Idle Scan (-sI)“. In: (2022). URL: https://nmap.org/book/idlescan.html (besucht am 15. 08. 2022).

[red22] redhat.com. „Was ist IaC (Infrastructure as Code)?“ In: (2022). URL: https://www.redhat.com/de/topics/automation/what-is-infrastructure-as-code-iac (besucht am 15. 08. 2022).

[str22] strongdm.com. „strongDM“. In: (2022). URL: https://www.strongdm.com (besucht am 26. 07. 2022).

[Sul04] Bob Sullivan. „Man pleads guilty in huge ID theft case“. In: (2004). URL: https://www.nbcnews.com/id/wbna6001526 (besucht am 11. 08. 2022).

[ter22] terraform.io. „Terraform“. In: (2022). URL: https://www.terraform.io (besucht am 22. 07. 2022).

[Tur22] John Turner. „Zero Trust Architecture Explained: The Ultimate Guide to Zero Trust Security“. In: (2022). URL: https://www.strongdm.com/zero-trust (besucht am 23. 07. 2022).

[ubu22] ubuntuusers.de. „dd“. In: (2022). URL: https://wiki.ubuntuusers.de/dd/ (besucht am 11. 07. 2022).

[vmw22] vmware.com. „VMware Fusion Player“. In: (2022). URL: https://www.vmware.com/de/products/fusion/fusion-evaluation.html (besucht am 23. 07. 2022).

[WO20] Denis Werner und Dominik Oepen. „Traue keinem!“ In: Heise Online (2020). URL: `https://www.heise.de/select/ix/2020/5/2007114411887053031` (besucht am 11. 07. 2022).

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Aachen, den 17. August 2022