

BACHELORARBEIT

Herr
Fabian Beu

Automatisierte Validierung und Implementierung von Firewallregeln

Mittweida, März 2023



Fakultät **Angewandte Computer- und Biowissenschaften**

BACHELORARBEIT

Automatisierte Validierung und Implementierung von Firewallregeln

Autor:

Fabian Beu

Studiengang:

IT-Sicherheit

Seminargruppe:

IF19W11-B

Erstprüfer:

Prof. Ronny Bodach

Zweitprüfer:

Simon Reuß, M.Sc.

Einreichung:

Mittweida, 26.03.2023

Verteidigung/Bewertung:

Mittweida, 2023

Faculty of **Applied Computer Sciences and Biosciences**

BACHELOR THESIS

Automated Validation and Implementation of Firewall Rules

Author:

Fabian Beu

Course of Study:

Cyber Security

Seminar Group:

IF19WI1-B

First Examiner:

Prof. Ronny Bodach

Second Examiner:

Simon Reuß, M.Sc.

Submission:

Mittweida, 26.03.2023

Defense/Evaluation:

Mittweida, 2023

Bibliografische Beschreibung:

Beu, Fabian:

Automatisierte Validierung und Implementierung von Firewallregeln. – 2023. – 47 S.

Mittweida, Hochschule Mittweida – University of Applied Sciences, Fakultät Angewandte Computer- und Biowissenschaften, Bachelorarbeit, 2023.

Referat:

In dieser Arbeit soll sich mit der Möglichkeit beschäftigt werden, den Prozess zur Freischaltung von Firewallregeln zu vereinfachen. Dafür soll ein Programm entwickelt werden, welches diesen Prozess automatisiert und dadurch den schwierigen Prozess vereinfachen soll. Des Weiteren wurde recherchiert, welche Richtlinien zum Schutz des Netzwerkes erforderlich sind und mit Firewalls umgesetzt werden können. Die Implementierung der Richtlinien soll hier als Code erfolgen und es einfach machen, diese Richtlinien automatisch durchzusetzen. Mithilfe dieser als Code implementierten Richtlinien sollen Anträge zur Freischaltung von Firewallregeln geprüft und validiert werden. Wurde der Antrag zur Freischaltung einer Firewallregel erfolgreich validiert, soll die Firewallregel automatisch implementiert werden. Durch diese und weitere Automatisierungen soll die menschliche Interaktion möglichst reduziert werden. Dadurch soll Zeit gespart werden und der Prozess zur Freischaltung einer Firewallregel effizienter durchgeführt werden können. Dafür wurde ein Programm implementiert, welches als Studie dienen und ein Machbarkeitsbeweis darstellen soll.

Inhaltsverzeichnis

Inhaltsverzeichnis	I
Abbildungsverzeichnis	III
Tabellenverzeichnis	V
Abkürzungsverzeichnis	VII
1 Einleitung	1
1.1 Aufgabenstellung und Motivation	1
1.2 Problemanalyse und Zielstellung	3
2 Grundlagen	5
2.1 Firewall Lösungen und Ansätze	5
2.1.1 Geschichte	6
2.1.2 Paketfilter	6
2.1.3 Stateful Inspection	7
2.1.4 Identitätsbasierte Firewall	8
2.1.5 NAT	8
2.1.6 Firewall Typen	9
2.1.6.1 Personal Firewall	9
2.1.6.2 Application-Firewall	10
2.1.6.3 Next Generation Firewalls	11
2.1.6.4 Software-Firewalls	12
2.1.6.5 Netzwerk-Firewalls	13
2.1.6.6 Firewalls für virtuelle Infrastrukturen	13
2.1.6.7 Cloud Firewall	14
2.1.6.8 High Availability Firewall	15
2.2 Platzierung von Servern	16
2.3 Richtlinien	16
3 Lösungskonzeption	19
3.1 Firewallrichtlinie	21
3.1.1 Erstellung einer Richtlinie	23
3.1.1.1 Portscans	26
3.1.2 Zusätzliche Firewall Schutzmaßnahmen	28
3.1.2.1 Erhöhter Schutzbedarf	30
3.1.2.2 Protokollierung	30
4 Implementierung	31
4.1 Entwicklungsumgebung	34
4.2 Ticket System	36
4.3 Firewall	36
4.4 Policy as Code	37

4.5	Open Policy Agent	37
4.5.1	Erstellung der Richtlinie	38
5	Ergebnisse und Diskussion	39
6	Fazit und Ausblick	45
6.1	Zukunftsausblick	45
	Anhang	49
	A Zusatz	49
	Glossar	51
	Literaturverzeichnis	57
	Stichwortverzeichnis	63
	Eidesstattliche Erklärung	65

Abbildungsverzeichnis

1.1	Firewallfreischaltungsantrag (Erstellt aus Unternehmenstemplate)	2
2.1	Meldung von Cybervorfällen durch Kritische Infrastrukturen (KRITIS) Unternehmen[14]	5
2.2	Investition in IT-Sicherheit Deutscher Firmen[25]	10
2.3	Schadensverursachende Angriffe[13]	14
3.1	Optimierter Freischaltungsantrag (Erstellt aus Unternehmenstemplate)	19
3.2	Aufbau von Attribute Based Access Control (ABAC) (Mit Drawio erstelltes Diagramm)	20
3.3	Traceroute zu ubuntuusers.de(Bildschirmfoto von Traceroute)	25
4.1	Architekturaufbau des Proof of Concept (PoC) (Mit drawio erstelltes Diagramm)	31
4.2	Formular zur Freischaltung der Firewall(Bildschirmfoto)	32
4.3	Beispiel einer dokumentierten Methode	35
5.1	ChatGPT frage nach Standardrichtlinie(Bildschirmfoto aus ChatGPT)	40
5.2	ChatGPT spezifische Frage nach einer Richtlinie(Bildschirmfoto aus ChatGPT)	41

Tabellenverzeichnis

2.1	High Availability	15
3.1	Scanarten	27
3.1	Scanarten: Fortsetzung	28

Abkürzungsverzeichnis

ABAC	Attribute Based Access Control
ACL	Access Control List
ADC	Application Delivery Controller
API	Application Programming Interface
APT	Advanced Persistent Threat
BSI	Bundesamt für Sicherheit in der Informationstechnik
CDN	Content Delivery Network
DaC	Diagram as Code
DDoS	Distributed Denial of Service
DHCP	Dynamic Host Configuration Protocol
DMZ	Demilitarised Zone
DNS	Domain Name System
DoS	Denial of Service
DPI	Deep Packet Inspection
FTP	File Transfer Protocol
GDPR	General Data Protection Regulation
HA	High Availability
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IaC	Infrastructure as Code
IaC	Infrastructure as Code
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IGMP	Internet Group Management Protocol
IoT	Internet of Things
IPS	Intrusion Prevention System
KI	Künstliche Intelligenz
KRITIS	Kritische Infrastrukturen
MFA	Multi Factor Authentication

MTU	Maximum Transmission Unit
NAPT	Network Address and Port Translation
NAT	Network Address Translation
NFS	Network File System
NGFW	Next Generation Firewall
NIST	National Institute of Standards and Technology
NTP	Network Time Protocol
OPA	Open Policy Agent
PaC	Policy as Code
PAT	Port Address Translation
PMTU	Path Maximum Transmission Unit
PoC	Proof of Concept
RBAC	Role Based Access Control
SCTP	Stream Control Transmission Protocol
SIEM	Security Information and Event Management
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SOAR	Security Orchestration, Automation and Response
SOD	Separation of Duty
SSH	Secure Shell
TCP	Transmission Control Protocol
TTL	Time to Live
UDP	User Datagram Protocol
UPnP	Universal Plug and Play
URL	Uniform Resource Locator
URLF	UniDirectional Link Filtering
UUID	Unique User ID
VoIP	Voice over IP
VPN	Virtual Private Network
WAF	Web Application Firewall
Wasm	WebAssembly

XML Extensible Markup Language

1 Einleitung

Durch die Steigerung der Sicherheitsvorfälle, siehe [Abbildung 2.3](#) auf Seite 14, ist es nötig mehr Geld und Arbeit für den Schutz der IT-Infrastruktur auszugeben. Eine wichtige Maßnahme, um Sicherheitsvorfälle zu reduzieren, ist die Firewall.

Diese Arbeit beschäftigt sich mit der Frage: „Wie kann der Prozess der Firewallfreischaltung optimiert werden und wie ist es möglich, diese effektiv mit Unternehmensrichtlinien zu verknüpfen?“. Um einen neuen Dienst zu betreiben oder einen Bestehenden im Netzwerk umzustellen, kann es sein, dass Firewallregeln geändert werden müssen. Dieser Prozess ist zumeist aufwendig, weswegen auch der Anfragende lange auf eine Bearbeitung der entsprechenden Anfrage und damit des zugehörigen [Tickets](#) warten muss. Dies ist nicht nur zeitaufwendig, sondern auch kostenintensiv.

Zusätzlich soll im Idealfall alles dokumentiert werden. Das Dokumentieren wird allerdings in vielen Unternehmen nicht ordnungsgemäß oder gar nicht durchgesetzt. Das verkompliziert den Prozess, da zunächst die Regeln der Firewall betrachtet werden müssen. Es muss geprüft werden, ob diese Regeln miteinander vereinbar sind.

Diese Arbeit soll Möglichkeiten aufzeigen, den Prozess der Integration neuer Firewallregeln zu automatisieren und an einem praktischen Beispiel die Vorteile und Nachteile zu evaluieren. Dafür soll als Erstes die Aufgabenstellung und Motivation analysiert werden. Hierfür wird der Prozess zur Freischaltung von Firewallregeln ausgewertet. Danach soll untersucht werden, welche Probleme beim Implementieren des zu entwickelnden Programms auftreten können. Als Nächstes werden benötigte Firewall Grundlagen aufgezeigt, die für diese Arbeit benötigt werden. Danach wird eine Lösungskonzeption erstellt, die versucht, die behandelten Probleme zu lösen. Dabei werden auch Richtlinien aufgezeigt, die benutzt werden sollten, um eine Unternehmensrichtlinie umzusetzen, die einen Basisschutz für das Unternehmen umsetzen. Danach folgt die Implementierung, welche die Entwicklung aufzeigt, mit der Entwicklungsumgebung. Dabei werden auch die verschiedenen Komponenten aufgezeigt, die für diese Lösung benutzt wurden, sowie warum diese gewählt wurden. Als Nächstes werden die Ergebnisse diskutiert. Hierbei wird geschaut, wie die Arbeit verlief und welche Probleme aufgetreten sind. So wird hier unter anderem geschaut, ob es möglich ist eine Richtlinie mit einer [Künstliche Intelligenz \(KI\)](#) zu erstellen. Als Letztes folgt das Fazit und der Zukunftsausblick.

1.1 Aufgabenstellung und Motivation

Im Folgenden soll die Aufgabenstellung sowie die Gründe für die Arbeit aufgezeigt werden. Der Prozess der Firewall Freischaltung ist langwierig, da er durch viele Interaktionen zwischen Mensch und Maschine gekennzeichnet ist. Siehe [Abbildung 1.1](#) auf der nächsten Seite.

Der erste Schritt der Freischaltung einer Firewallregel ist der Antrag selbst. Dieser wird von einem Mitarbeiter gestellt, welcher eine Freischaltung benötigt. Der gestellte Antrag wird dann an einen Administrator versendet. Dieser muss den Antrag mit den vorhandenen Unternehmensrichtlinien abgleichen. Hat der Antragssteller Informationen bei seinem Antrag



Abbildung 1.1: Firewallfreischaltungsantrag (Erstellt aus Unternehmenstemplate)

vergessen oder muss der Grund oder die Berechtigung geklärt werden, müssen sich der Administrator und der Antragssteller zusammensetzen. Durch die menschliche Interaktion erhöht sich der Zeitaufwand des Prozesses. Stimmt die Regel mit der Unternehmensrichtlinie überein und hat der Nutzer die Berechtigungen für eine solche Regel, kann diese akzeptiert werden. Ist die Regel akzeptiert, muss sie von einem Administrator implementiert und angewandt werden. Bevor die Regel jedoch angewendet werden kann, muss diese getestet werden, um sicherzustellen, dass die Funktion ordnungsgemäß ist. Sollte es bei einem der Schritte zu Fehlverhalten kommen, ist es möglich, dass ein Schritt revidiert werden muss. Dabei kann es sein, dass ein Antrag vollständig neu gestellt werden muss.

Deswegen ist das Ziel dieser Arbeit, diesen Prozess zu automatisieren und damit zu verbessern. Durch die Automatisierung können die Administratoren sich auf andere Aufgaben konzentrieren, wie Sicherheitsüberprüfungen und Wartungsarbeiten.

Es kommt immer wieder vor, dass die Unternehmensrichtlinien für Firewallregeln nicht eindeutig formuliert oder von den Administratoren uneinheitlich umgesetzt werden. So können Fehler entstehen, die zusätzlich durch Unachtsamkeit verstärkt werden können. Die Firewallregeln sind aber häufig provisorisch, wodurch sicherheitstechnische Probleme entstehen können.

Es kann passieren, dass Nutzer beantragte Firewallregeln freigeschaltet bekommen, ohne die benötigten Berechtigungen zu besitzen. Zudem können Fehler vorkommen durch die Nutzer oder Administratoren. Ein weiterer Nachteil ist, dass es bis zur Freigabe für Testzwecke und Einführung in die Produktion langwierige Prozesse gibt. Durch den hohen Zeitaufwand entstehen dann zusätzlich Kosten. Eine dynamische Erstellung von Diensten ist durch diesen alten Prozess nicht möglich. Soll beispielsweise ein neuer Dienst erstellt werden oder soll dieser nur zu bestimmten Zeiten zur Verfügung stehen, ist dies mit einer langen Verzögerung verbunden, welches eine dynamische Erstellung von Diensten unmöglich macht. Die Erstellung einer Infrastruktur mit einer [Infrastructure as Code \(IaC\)](#) ist nur manuell möglich und muss vorher bereits freigeschaltet sein. Soll ein Produktivserver erstellt werden oder ein anderer

Server, welcher von außen erreichbar sein soll, können bei der Beantragung zusätzlich auch die Firewallregeln beantragt werden. Dies kann den Deploymentprozess vereinfachen und schneller machen, wenn hier die Firewallregeln automatisch zur Verfügung gestellt werden.

1.2 Problemanalyse und Zielstellung

Aus der Aufgabenstellung ergeben sich die folgenden Probleme des Prozesses der automatisierten Integration neuer Firewallregeln.

Das erste Problem des Prozesses ist die Beantragung der Firewallregel. Sie muss gezielt abgefragt werden, um alle benötigten Daten direkt zu erhalten, und zu verhindern, dass eine Rücksprache erforderlich ist. Ein System zur Beantragung muss dabei in der Lage sein, vom Benutzer alle Informationen zu erfassen, welche benötigt werden, um die Firewallregel anzunehmen und zu implementieren. Die Abfrage sollte demnach so erfolgen, dass sie weiterverarbeitet und ausgewertet werden kann. Zudem sollte diese Abfrage nutzerfreundlich gestaltet werden. Es sollte so gestaltet werden, dass jeder Nutzer ohne großen Aufwand diese Informationen zur Verfügung stellen kann.

Das zweite Problem ist die Bearbeitung der Anfrage. Ein System muss prüfen, ob der Nutzer autorisiert ist, um eine Regel zu beantragen. Das System muss anschließend überprüfen, inwieweit die beantragten Regeln der Richtlinie des Unternehmens entsprechen. Beim Erstellen der Richtlinien muss auch eine Lösung gefunden werden, wie eine Implementierung möglich ist, sodass die Richtlinien wartbar und leicht verständlich sind. Dadurch erhöht sich nicht nur die Wartbarkeit, auch kann besser analysiert werden, falls es Fehler in den Richtlinien gibt oder eine neue Richtlinie hinzugefügt werden muss. Der manuelle Prozess sollte genutzt werden, wenn Probleme auftreten. Des Weiteren sollte der manuelle Prozess genutzt werden, wenn es sich um einen exotischen Zweck handelt. Ein Administrator muss diese Anfrage dann manuell bearbeiten.

Nach der Prüfung muss die Regel implementiert werden. Hier soll ebenfalls eine Automatisierung erfolgen, um die Implementierungszeit zu reduzieren.

Nach dem Implementieren erfolgt die Testphase, in der überprüft wird, inwieweit die geforderten Funktionalitäten erfüllt sind. Durch die Automatisierung kann die Testphase weggelassen werden, da ein System sich um die Implementierung kümmert und die Regeln direkt korrekt implementiert werden können. Doch dieser Automatismus muss ausführlich getestet werden, um sicherzustellen, dass die gewünschten Regeln korrekt implementiert werden. Ist der Prozess der Freischaltung von Firewallregeln ausreichend getestet worden, etwa durch Unittests liegt diese Testphase nur in der Entwicklungsarbeit vor doch nicht mehr im späteren Prozess. Es soll gewährleistet sein, dass der gesamte Prozess zur Freischaltung der Firewallregeln so schnell verläuft, dass Anträge innerhalb kürzester Zeit genehmigt werden, sodass es möglich ist dynamisch Firewallregeln zu beantragen.

2 Grundlagen

Dieses Kapitel soll die nötigen Grundlagen für diese Arbeit aufzeigen. Dafür sind zwei Technologien relevant, welche für diese Arbeit nötig sind. Zum einen ist dies die Firewall selbst, siehe [Abschnitt 2.1](#). Hier gibt es verschiedene Firewalls mit unterschiedlichen Stärken und Anwendungsfällen.

Zudem ist es wichtig geeignete Richtlinien zu erstellen, welche möglichst restriktiv sind, siehe [Abschnitt 3.1](#) auf Seite 21. Dafür wird die zweite Komponente, ein Server zum Validieren der Richtlinien genutzt. Hier ist es wichtig eine geeignete Richtlinie zu erstellen, welche möglichst restriktiv ist, aber zulässt, dass die Mitarbeiter noch arbeiten können.

2.1 Firewall Lösungen und Ansätze

In diesem Abschnitt sollen die theoretischen Grundlagen von Firewalls behandelt werden. Die Firewall ist in dieser Arbeit die wichtigste Komponente, da das Ziel in dieser Arbeit ist, den Prozess zur Freischaltung zu vereinfachen. Die Firewall ist in der jetzigen Zeit von immer größerer Bedeutung. Anhand der Statistik siehe [Abbildung 2.1](#) ist zu sehen, dass die Angriffe auf [KRITIS](#) Unternehmen in den vergangenen Jahren stark zugenommen haben.

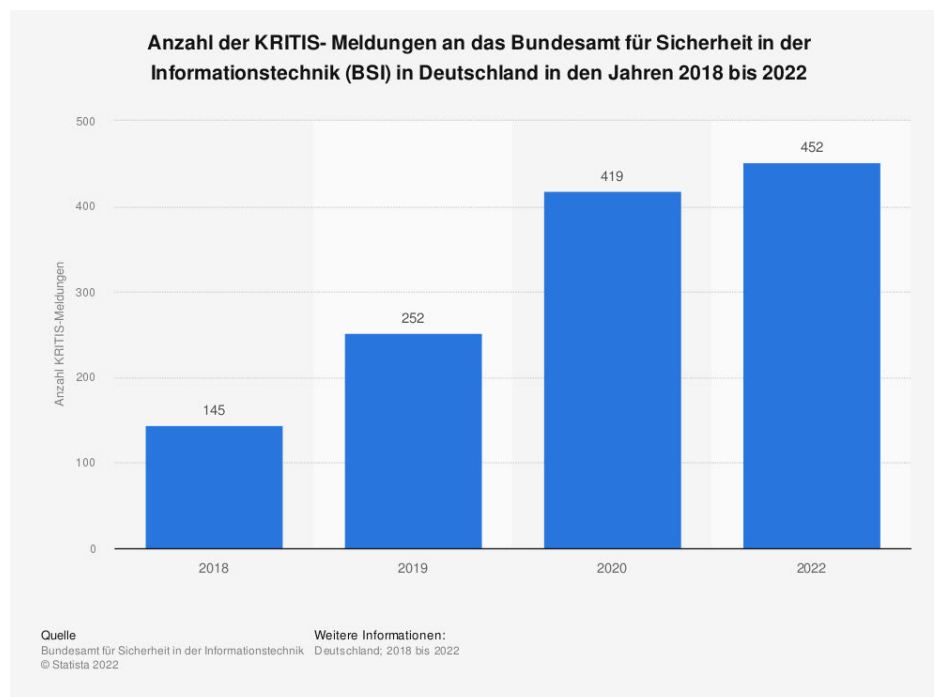


Abbildung 2.1: Meldung von Cybervorfällen durch [KRITIS](#) Unternehmen[14]

Die als [KRITIS](#) bezeichneten Unternehmen sind für die Bevölkerung überlebensnotwendig, da in diese Kategorie der Energiesektor und Wasserwerke fallen.[65] Auch zeigt diese Statistik den rasanten Anstieg an Angriffen. Dies betrifft nicht nur [KRITIS](#), immer mehr Unternehmen werden Opfer von Angriffen und müssen mehr in eine noch sichere Infrastruktur investieren. Selbst in der heutigen Zeit gibt es immer noch Unternehmen, welche sich nicht ausreichend schützen und in den Schutz vor Angriffen investieren.

2.1.1 Geschichte

Firewalls wurden in den späten 1980er-Jahren entwickelt. Durch das Aufkommen von Angriffen aus dem Internet wurden sie notwendig. Auch zuvor gab es Angriffe, welche zur Entwicklung der Antivirus Lösungen führte. Als in den Anfängen des neuen Jahrtausends Schwachstellen in Applikationen ausgenutzt wurden, entwickelten sich die [Intrusion Prevention System \(IPS\)](#). 2010 kamen, dann polymorphe, unbekannte und ausweichende Angriffe auf. Diese führten zur Entwicklung von Anti-Bot und Sandboxing Produkten. 2017 gab es dann großangelegte Angriffe, welche mit fortschrittlichen Technologien durchgeführt wurden und zur Entwicklung fortschrittlicher Bedrohungsabwehr führten.[71] Die Anfänge nahmen die Paketfilter-Firewalls, siehe [Abschnitt 2.1.2](#). Das größte Problem hierbei war, dass es möglich war Netzwerkpakete ohne jegliche Kontrolle über einen [Port](#) zuschicken, sofern dieser geöffnet war. Dies konnte bei den alten Betriebssystemen aufgrund des verwundbaren [Netzwerk Stack](#) zu einem Ausfall des Betriebssystems führen. Doch diese Firewalls ebneten den Weg für die nachfolgenden Firewalls.[59, S. 2] Der nächste Schritt waren die „Statefull Inspection“ Firewalls. Diese lösten das Problem der Paketfilter und fügten die Funktionalität der Verbindungsüberwachung ein, sodass nur legitim aufgebaute Verbindungen an der Firewall vorbeikamen.[59, S. 3] Siehe [Abschnitt 2.1.3](#) auf der nächsten Seite. Dabei benutzen viele Anwendungen den gleichen [Port](#) oder benutzen einen zufällig ausgewählten [Port](#) im sogenannten dynamischen [Port](#) Bereich. Der dynamische [Port](#) Bereich wird oft auch für ausgehende Verbindungen benutzt. Dafür wird ein zufälliger [Port](#) in diesem Bereich ausgewählt. Zu diesem Zweck wurde die „Statefull Inspection“ Firewall weiterentwickelt zur „Application-Firewall“. Diese wurde um das sogenannte [Deep Packet Inspection \(DPI\)](#) erweitert, welches die Erkennung der Anwendungen ermöglicht und dadurch eine bessere Kontrolle der Anwendungen unterstützt, sowie die Erkennung potenziell unerwünschter Netzwerkpakete.[59, S. 7, 8] Der nächste Schritt waren die „Next Generation Firewalls“ siehe [Abschnitt 2.1.6.3](#) auf Seite 11, die die Funktionen der vorherigen Firewall Versionen und neue Funktionen performant in eine performante Lösung bündelten. Siehe [Abschnitt 2.1.6.3](#) auf Seite 11.

2.1.2 Paketfilter

Der Paketfilter ist eine der grundlegenden Firewalltechniken. Alte Firewalls unterstützen nur diese Funktionalität. Sie routen dabei Netzwerkverkehr und besitzen eine Zugriffskontrollfunktionalität für die Schicht 3 im [ISO/OSI](#) Modell, die sogenannte Netzwerkschicht. Sie waren in dieser Schicht hauptsächlich auf Protokolle beschränkt. Zu den erkannten Informationen zählen [IP](#), [Port](#), [Interfaces](#) und das benutzte [Transportprotokoll Transmission Control Protocol \(TCP\)](#), [User Datagram Protocol \(UDP\)](#) oder [Internet Control Message Protocol \(ICMP\)](#). Diese Firewalls waren dabei auch bekannt als „stateless inspection Firewalls“. Hier wurde nicht beachtet, in welchem Zustand die sitzungsbasierte Kommunikation war. Bestehende Probleme der [TCP/IP](#) Spezifikation können von Angreifern ausgenutzt werden. Pakete bei denen [Spoofing](#) oder sonstige Änderungen angewendet wurden, können nicht erkannt werden. Die inkorrekten Header in den Netzwerkpaketen können für Angriffe ausgenutzt werden. Manche der Paketfilter-Firewalls unterstützen fragmentierte Netzwerkpakete. Diese Möglichkeit wird von der [TCP/IP](#) Spezifikation angeboten und sollte, auch wenn nötig benutzt werden. Die Fragmentierung der Netzwerkpakete erschwert die Erkennung von Angriffen. [47, S. 12, 13] Kommen diese Pakete auf unterschiedlichen Wegen in das Netzwerk, erschwert das die

Erkennung von Angriffen. Das nicht erkennen eines Angriffes kann passieren, wenn diese Pakete nicht oder falsch angeordnet analysiert werden. Zudem gibt es Techniken, um zu verhindern, dass die Pakete korrekt angeordnet werden. Mithilfe dieser Techniken fehlt bei der Analyse der schadhafte Teil, wodurch der Angriff nicht erkannt werden kann. Zudem wird die Zusammensetzung der fragmentierten Pakete von verschiedenen Betriebssystemen unterschiedlich vorgenommen, sollten sich die fragmentierten Pakete überlappen.[6][32] Heutzutage treten diese fragmentierten Netzwerkpakete größtenteils in [Virtual Private Network \(VPN\)s](#) auf. Da hier Netzwerkpakete in Netzwerkpakete gepackt werden, kann es schnell passieren, dass ein Paket die Mindestgröße überschreitet. Das Überschreiten der Mindestgröße erzwingt dann die Fragmentierung der Pakete. Das Blockieren der Fragmentierung ist eine häufige Fehlerquelle für nicht funktionierende [VPNs](#). Einige der Firewalls unterstützen die Möglichkeit, die fragmentierten Netzwerkpakete wieder zusammenzusetzen. Dazu muss die Firewall aber eine [VPN](#) Funktionalität haben. Doch es muss gut implementiert sein, andernfalls kann ein Angreifer dies für eine [Denial of Service \(DoS\)](#) Attacke ausnutzen.[47, S. 12, 13]

2.1.3 Stateful Inspection

Die „stateful inspection“ erweitert den Paketfilter, indem der Status der Verbindung überwacht wird und Netzwerkpakete blockiert werden, welche nicht den erwarteten Zustand haben. Dies wird dadurch erreicht, dass zusätzlich zu der vom Paketfilter überwachten [ISO/OSI](#) Schicht 3 nun auch Schicht 4, die sogenannte [Transportschicht](#) ausgewertet wird. Jede Verbindung wird dabei in einer Tabelle festgehalten. Diese enthalten die Quell [IP](#), Ziel [IP](#), [Port](#) Nummern und Verbindungsstatus. Die drei wichtigsten Verbindungszustände sind Verbindungsaufbau, Verbindungsabbau und bestehende Verbindung. Ein Paket muss entweder eine neue Verbindung aufbauen, sofern dies durch die Firewallregeln erlaubt ist oder Teil einer bestehenden Verbindung sein. Viele Firewalls überprüfen auch die [TCP](#)-Sequenznummer. Falls diese nicht korrekt ist, werden Netzwerkpakete abgelehnt. Diese „stateful inspection“ funktioniert allerdings nicht für [UDP](#)-Netzwerkpakete, da [UDP](#) verbindungslos ist. Das heißt, es kann nicht erkannt werden, in welchem Zustand diese Verbindung ist, da es kein Protokoll für den Verbindungsaufbau oder Abbau gibt. Allerdings haben einige Firewalls eine Möglichkeit für [Domain Name System \(DNS\)](#) und [Network Time Protocol \(NTP\)](#) implementiert. Diese Anfragen können nach einem empfangenen Paket geschlossen werden oder nach einem vorkonfigurierten Zeitlimit. Am häufigsten wird diese Verbindungsüberwachung bei [Hyper Text Transfer Protocol \(HTTP\)](#) benutzt. Ein Client innerhalb des Netzwerks darf eine Verbindung aufbauen und nutzen, von außerhalb ist es aber nur möglich mit einer bestehenden Verbindung auf das Netzwerk zuzugreifen. Das heißt, ein Server kann nur antworten, aber keine Anfragen stellen. Dies ist für [HTTP](#) wichtig, da [HTTP](#) Verbindungen von einem zufälligen unprivilegierten [Port](#) aus aufbaut. [47, S. 14] Diese Verbindungsüberwachung ist auch bei einigen [ICMP](#) Typen möglich:

- Echo Anfrage (ping, Typ 8) und echo Antwort (pong, Typ 0)
- Zeitstempel Anfrage (Typ 13) und Antwort (Typ 14)
- Informationsanfrage (Typ 15) und Antwort (Typ 16)
- Adressmasken Anfrage (Typ 17) und Antwort (Typ 18)

Die Verbindungsverfolgung ist hier möglich, da auf eine Anfrage immer eine Antwort erfolgen muss. Andere **UDP**-Verbindungen können die vorher bereits erwähnte Verbindungsverfolgung implementieren, indem auf ein **UDP** Paket eine Antwort erfolgen kann und falls dies nicht innerhalb einer bestimmten Zeit erfolgt, diese Verbindung geschlossen wird.

2.1.4 Identitätsbasierte Firewall

Die Administration von Firewalls wird komplexer, je größer ein Netzwerk ist. Um dies zu vereinfachen, kann ein nutzerbasiertes Design genutzt werden. Bei diesem wird der Datenverkehr anhand der Nutzereinstellungen zugelassen oder geblockt. Diese Firewalls sind meistens nicht nutzerfreundlich, da für jeden Nutzer die Rechte individuell eingestellt werden müssen. Dies sollen sogenannte identitätsbasierte Firewalls lösen. Sie sollen dazu führen, dass ein Firewall Administrator möglichst wenig eingreifen muss. Um Nutzer zu identifizieren, soll eine sogenannte **Unique User ID (UUID)** genutzt werden, welche vom Administrator den Benutzern zugewiesen wird. Dazu soll unter anderem der Netzwerkname genutzt werden. Durch die **UUID** kann geprüft werden, ob es sich bei dem Nutzer um einen lokalen Nutzer oder einen Domainnutzer handelt. Bei Windows werden alle Events aufgezeichnet, weshalb hier ein ereignisbasiertes System seine Stärken ausspielen kann. Treten ungewöhnliche Ereignisse auf, können zeitnah Gegenmaßnahmen ergriffen werden. Beispielsweise kann in Windows eine **IP** blockiert werden, welche drei fehlgeschlagene Anmeldeversuche hatte. Dazu muss in Windows auf das Ereignis mit der ID 529 gewartet werden, um festzustellen, wann eine **IP** gesperrt werden soll. Aufgrund der Vielzahl neuer Geräte, wie Smartphones und **Internet of Things (IoT)**-Geräte ist es wichtiger geworden, die einzelnen Geräte besser zu schützen. Um die zahlreichen Geräte zu koordinieren, werden die Domain Nutzernamen genutzt, welche Nutzer auf verschiedenen Geräten identifizieren und es so ermöglichen, auf verschiedenen Geräten auf die gleichen Informationen zuzugreifen. Außerdem arbeiten diese Firewalls auf der **ISO/OSI** Schicht 7 der höchsten, welche direkt die Daten verarbeitet, die der Nutzer zu sehen bekommt. Hier sind die wichtigen Protokolle wie **HTTPS**, **Secure Shell (SSH)** und weitere.[53]

2.1.5 NAT

Viele Firewalls können als **Network Address Translation (NAT)**, auch als **Port Address Translation (PAT)** bezeichnet, fungieren. Dies wird auch zu einem sogenannten **Network Address and Port Translation (NAPT)** vereinigt. Dabei dienen weder **NAT** noch **PAT** als Sicherheitsfeature. Die **NAT** wurde entwickelt, um dem Mangel an **IP**-Adressen entgegenzuwirken. Die Idee hinter **NAT** ist es, der Verbindung zwischen Internet und einem internen Netzwerk eine **IP**-Adresse zu geben und alle Geräte über diese **IP**-Adresse kommunizieren zu lassen, wenn sie mit externen Geräten kommunizieren wollen. Hierdurch benötigt nicht jedes Gerät eine eigene **IP**-Adresse, sondern kann die des **NAT** oder **NAPT** fähigen Gerätes nutzen. Diese Funktionen können grundsätzlich auch für jede Firewall eingestellt werden. Da dies jedoch komplizierter ist, hat sich **NAT** durchgesetzt. Bei **IPv6** ist dies aufgrund der zahlenmäßigen Adressen nicht nötig. Beabsichtigt ein Rechner innerhalb des Netzwerkes eine Verbindung zu einem Dienst außerhalb aufzubauen, wird die **NAT** kontaktiert. Diese ersetzt den Quell-**Port** der Anfrage durch einen freien **Port** auf der **NAT**. Nach diesem Austausch wird die Anfrage an den Zielhost außerhalb des Netzwerkes weitergeleitet. Der angefragte Dienst kann darauf antworten

und schickt die Nachricht zurück zur [NAT](#). Dann wird geschaut, welcher Computer diesem [Port](#) zugeteilt wurde und die Anfrage wird dann weiter an den entsprechenden Rechner im internen Netzwerk geschickt. Ein direkter Verbindungsaufbau von außen mit einem Client im internen Netzwerk ist nicht möglich, es sei denn, in der [NAT](#) existiert eine [Port-Weiterleitung](#), die alle Anfragen an einen bestimmten [Port](#) eines definierten Rechners weiterleitet. Diese Funktionalität wird als „pinholing“ bezeichnet. Ein klassischer Anwendungsfall hierfür ist ein Webserver, der hinter dieser [NAT](#) steht. [47, S. 25]

2.1.6 Firewall Typen

Es gibt verschiedenste Firewall Typen, mit unterschiedlichen Aufgaben und unterschiedlichen Einsatzorten. Diese Firewalls sind alle ähnlich aufgebaut, haben aber unterschiedliche Entwicklungsstufen oder sind für spezielle Einsatzzwecke konzipiert. Anhand des Diagramms [Abbildung 2.2](#) auf der nächsten Seite, ist zu sehen, dass die Firewall weiterhin einer der wichtigsten Schutzmechanismen ist, da die Firewall direkt an der Front arbeitet, um das Netzwerk zu schützen. Sie bestimmt, mit welchen Diensten kommuniziert werden kann und hat auch noch einige weitere Funktionen. Überdies ist sie standardmäßig auf den Betriebssystemen installiert und wird von den Benutzern normalerweise verwendet, siehe [Abschnitt 2.1.6.1](#). Die Netzwerkfirewalls arbeiten direkt an den Netzwerkgrenzen und sind damit der erste Verteidigungsmechanismus. Sie müssen dementsprechend auf leistungsstarker Hardware implementiert werden. In mittelgroßen Netzwerken sind diese Lösungen häufig in einen Router integriert. [OPNSENSE](#) und [PFSENSE](#) sind beispielsweise zwei leistungsfähige Router Lösungen, welche auch eine leistungsstarke Firewall Lösung beinhalten.

Neben diesen Firewalls gibt es einige andere Firewalls, wie Application-Firewalls und Web Application-Firewalls. Sie funktionieren ähnlich, blocken aber bestimmte Aktionen des Benutzers, sofern diese als schadhaft oder unerwünscht eingestuft werden.

2.1.6.1 Personal Firewall

Die Personal-Firewall ist eine Firewall, die direkt auf dem Computer installiert ist, der geschützt werden soll. Das heißt, dass sie in ihrem Leistungsumfang stark begrenzt ist gegenüber einer Firewall, die Netzwerke schützen muss. Dienste, die nicht benötigt werden oder ein mögliches Sicherheitsrisiko darstellen, können eingeschränkt oder blockiert werden. Ein Angriff innerhalb eines Netzwerkes kann von Firewalls nicht abgefangen werden. Die Personal-Firewall schützt jedoch vor solchen Angriffen, da sie direkt auf dem Computer installiert ist. Außerdem können sie den zulässigen Netzwerkverkehr für einen speziellen Server oder Computer noch weiter einschränken, um die Sicherheit für ein spezifisches Gerät noch weiter zu erhöhen.[47, S. 20]

Dabei haben diese meistens zwei Funktionalitäten, zum einen das Paket-Filtern, welches überprüft, ob die Netzwerkpakete erlaubt sind. Des Weiteren gibt es noch die [Sandbox](#) Funktionalität, bei welcher als gefährlich eingestufte Applikationen in einem Schutzbereich ausgeführt werden können. Sie sind von der restlichen Umgebung isoliert, sodass sie keinen Einfluss auf diese Systeme ausüben können. [4] Bei den Personal-Firewalls gibt es auch welche mit

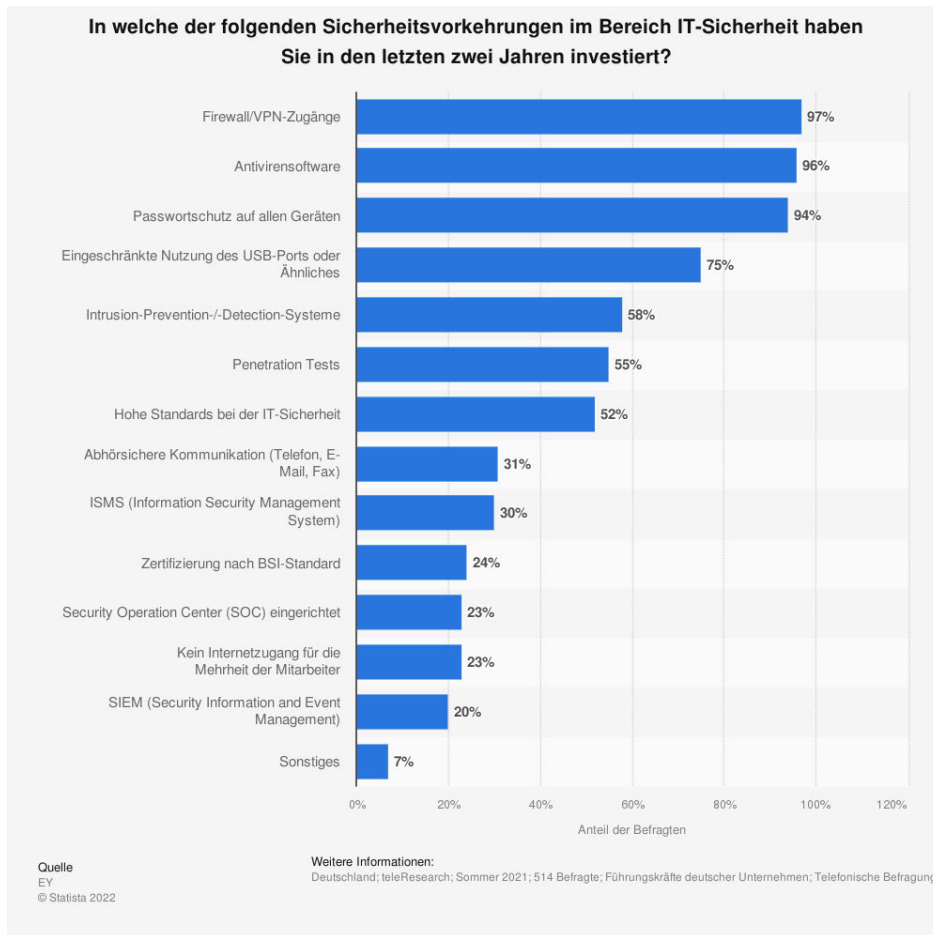


Abbildung 2.2: Investition in IT-Sicherheit Deutscher Firmen[25]

unterschiedlichen Profilen. Beispielsweise ein Profil für private Netzwerke und ein restriktiveres Profil für öffentliche Netzwerke. Diese Firewalls sollten zentralisiert verwaltet werden, um sicherzustellen, dass die Unternehmensrichtlinien durchgesetzt werden. Es spielt dabei keine Rolle, wie die Unternehmensrichtlinien aus technischer Sicht durchgesetzt sind, dem Nutzer sollte immer eine Mitteilung angezeigt werden, wenn die Firewall Netzwerkverkehr blockiert. So kann der Nutzer direkt sehen, wo das Problem liegt und der IT-Support muss nicht belastet werden.

Des Weiteren gibt es noch einen speziellen Anwendungsfall der Personal-Firewall. Dieser wird als „Host based“ Firewall bezeichnet und zeichnet sich durch ein nicht Vorhandensein einer Benutzeroberfläche aus. Diese dienen für Systeme, wie Server, die ohne grafische Oberfläche arbeiten. Dabei werden meistens applikationsbasierte und addressbasierte Kontrollen umgesetzt. Viele der „Host based“ Firewalls sind auch in der Lage IPS Funktionen umzusetzen. [47, S. 20–21]

2.1.6.2 Application-Firewall

Bei den Application-Firewalls wird die „stateful inspection“ mit der DPI Funktionalität zusammengefügt. Die DPI analysiert die Protokolle und fügt damit eine Einbruchserkennung hinzu. Dazu werden die Protokolle in der Schicht 7 des ISO/OSI Modells analysiert. Dabei werden

herstellerspezifische Protokollaktivitäten beobachtet und es wird analysiert, ob es Abweichungen vom üblichen Verhalten gibt. Eine Firewall kann den Netzwerkverkehr basierend auf dem Verhalten der Protokolle ablehnen oder erlauben. Beispielsweise, wenn innerhalb des E-Mail Protokolls eine ausführbare Datei verschickt wird. Dabei werden von den Firewalls vorwiegend die gängigen Protokolle wie [HTTP](#), Mail, [Voice over IP \(VoIP\)](#), [Extensible Markup Language \(XML\)](#) analysiert. [47, S. 15] Hiermit ist es auch möglich bestimmte Applikationen zu sperren, sollte die Unternehmensrichtlinie dies verlangen oder es keinen Grund gibt diese zu erlauben. Etwa wird [File Transfer Protocol \(FTP\)](#) nicht benötigt, da es veraltet und unverschlüsselt ist.

2.1.6.3 Next Generation Firewalls

Die [Next Generation Firewall \(NGFW\)](#) bieten auch die Möglichkeit der [DPI](#), mit der es machbar ist den Netzwerkverkehr zu analysieren. Dafür arbeitet sie auf dem [Application Layer](#) der obersten Schicht des Netzwerkmodells. Verdächtige Aktivitäten können erkannt und Gegenmaßnahmen getroffen werden. Dies ist ähnlich wie ein [Intrusion Detection System \(IDS\)/IPS](#) System, welches in einer Firewall integriert sein kann. Durch das [DPI](#) können Anwendungen und Nutzer kontrolliert werden. Um noch genauer zu sein, kann die [NGFW](#) an das [Active Directory](#) angeschlossen werden. Dadurch können Nutzeraktivitäten kontrolliert und aufgezeichnet werden.[34]

Dieser Ansatz ist höchst effektiv, allerdings funktioniert dies nur, wenn der gesamte Netzwerkverkehr mitgelesen wird. Da dies bei verschlüsselten Verbindungen nicht der Fall ist, müssen diese aufgebrochen werden. Um dies durchzuführen, wird auf der Firewall ein Zertifikat erstellt und auf den Rechnern innerhalb des Netzwerkes als vertrauenswürdig eingestuft. Die Rechner im Netzwerk benutzen dann dieses Zertifikat der Firewall. So kann die Firewall den Inhalt der Netzwerkpakete entschlüsseln. Nachdem alles geprüft wurde, verschlüsselt die Firewall den Datenverkehr wieder und schickt diesen zum Ziel.

Dies ist jedoch sicherheitstechnisch problematisch, da nun jeder mit Zugriff auf das entsprechende Zertifikat den gesamten Netzwerkverkehr aller Nutzer im Netzwerk mitlesen kann. Datenschutzrechtlich ist dies fragwürdig, da jeder mit Zugriff auf die Firewall die sensiblen Daten der Nutzer entschlüsseln kann.

Durch ihre [DPI](#) ist es den [NGFWs](#) so möglich verschiedenste Angriffe zu erkennen, welche die einfachen Firewalls nicht erkennen können. Dazu zählen die folgenden Angriffe:

- [Dynamic Host Configuration Protocol \(DHCP\) Starvation Attacke](#)
- [TCP SYN](#) Attacken
- [UDP Flood](#) Attacke
- [ICMP Smurf](#) Attacke

Die [NGFW](#) sucht mithilfe von Signaturen nach schadhaften Verbindungen und greift dabei auch auf Heuristiken und Verhaltensanalyse zurück, um weitere Angriffe und sogar [Zero Day](#) Attacken zu identifizieren. Dabei wird auch geprüft, ob ein Dienst ein abnormales Verhalten aufweist. Die [NGFW](#) filtert den Netzwerkverkehr sogar noch mehr, wenn ein [Active Directory](#) angeschlossen wird. Dabei wird jedem Nutzer die entsprechende [IP-Adresse](#) zugeordnet

und weitere Nutzerdaten werden gesammelt. So kann bestimmten Nutzern der Zugriff mit bestimmten Verbindungen untersagt und Nutzer spezifisch geloggt werden, welcher Nutzer was gemacht hat. [2]

Die [TCP-SYN](#) Attacke kann leicht abgewehrt werden, wenn der Server viele [SYN](#) Netzwerkpakete erkennt, welche sofort wieder verworfen werden. In einem Versuch mit einer [NGFW](#) Checkpoint Firewall, hatte ein dahinter liegender Server einen Leistungsanstieg von 10 bis 15 Prozent. Während im Versuch mit einer Statefull Firewall der IBM-ISS eine 90-prozentige höhere Auslastung zu erkennen war.[51]

Eine [NGFW](#) erlaubt auch, eine Upgrade-Funktion zur Integration künftiger Informationsquellen und Techniken zur Reaktion auf neue Bedrohungen.[65]

Um die Performance der [NGFW](#) zu verbessern, wird hier der sogenannte „Stream-based“ Scan angewandt. Das bedeutet, dass die Netzwerkpakete sofort beim Eintreffen gescannt werden. Andere Firewalls benutzen hier das „File-Based“ Verfahren, bei welchem darauf gewartet wird, dass die gesamte Datei vorliegt. Diese kann dann gescannt werden. Dies kostet allerdings Zeit und Speicher und verlangsamt die Firewall.

Die [NGFW](#) arbeitet mit mehreren Schichten. Dabei wird das Paket zunächst gescannt. Abhängig von dem Scanergebnis werden bestimmte Scans in der höheren Schicht ausgewählt. So werden unnötige Scans vermieden und die Performance verbessert. Dafür operiert die [NGFW](#) auch auf den [ISO/OSI](#) Schichten 3, 5 und 7. Auf Layer 3 wird die klassische Firewall benutzt, welche nach IP und [Port](#) filtert. Auf Layer 5 arbeitet die [NGFW](#) als Gateway und inspiziert jedes Datensegment, bevor es weitergeht. Auf Layer 7 werden High Level Scans durchgeführt. Hier gibt es zusätzlich zu den bereits genannten Techniken die Möglichkeit einer [Sandbox](#)-Erkennung für ausführbare Dateien.[29]

2.1.6.4 Software-Firewalls

Zu den Software-Firewalls zählen Firewalls, die lokal auf Rechnern oder Server laufen. Meistens ist eine Software-Firewall auf einem Rechner vorinstalliert. Die Software-Firewalls müssen besondere Anforderungen erfüllen. Es ist notwendig, dass die Firewall und die Anwendungen, die vom Nutzer ausgeführt werden, miteinander koexistieren können. Dazu zählt, dass sie nur wenig Rechnerleistung verbrauchen dürfen. Die Firewall muss auch sicher sein, sodass keine anderen Anwendungen unerlaubt auf die Firewall zugreifen dürfen. Das bedeutet, dass Konfigurationen der Firewall durch eine Zugriffskontrolle geschützt und protokolliert werden müssen. Dasselbe gilt für das Deinstallieren, Installieren oder Stoppen der Firewall. Es gibt auch Business-Firewalls, die Erweiterungen, wie eine zentralisierte Firewall-Verwaltung bieten. Zu den Software-Firewalls gehört etwa die Personal-Firewall, siehe [Abschnitt 2.1.6.1](#) auf Seite 9. Eine der Funktionen ist die granulare Kontrolle von Anwendungen. So ist es möglich, einzelne Anwendungen speziell im Zugriff einzuschränken, statt dies im Netzwerk zu realisieren. Software-Firewalls können auch mit anderen Lösungen integriert werden, wie einer Angriffserkennung, die automatisch Firewallregeln ändert. Ein Beispiel hierfür ist [CrowdSec](#). Ein weiterer Vorteil der Software-Firewalls ist, dass sie auch in anderen Netzwerken umgesetzt werden, da sie sich lokal auf dem Computer befinden. Dadurch können Unternehmensrichtlinien in Netzwerken außerhalb durchgesetzt werden.[72]

2.1.6.5 Netzwerk-Firewalls

Firewalls, die im Netzwerk platziert werden sind in der Regel Hardware-Firewalls. Diese besitzen neben der reinen Firewallfunktionalität zumeist weitere Ausstattungsmerkmale. Firewalls, welche im Netzwerk platziert werden, wie Hardware-Firewalls, besitzen meist neben den Firewall-Funktionalitäten noch andere Funktionalitäten. Dazu gehört Routing. Zu diesem Zweck agieren sie an der Grenze zwischen mehreren Netzwerken. Es ist also möglich, die Firewall als Router zu nutzen und Einfluss darauf zu nehmen, wohin Netzwerkpakete weitergeleitet werden. Dabei kann auch NAT zum Einsatz kommen. Dies erhöht die Sicherheit, da hier interne Adressen geschützt werden. Außerdem können sie auch genutzt werden, um Netzwerke zu verwalten. Alle Netzwerk-Firewalls haben eine hohe Netzwerklast zu bearbeiten und akzeptieren oder blockieren die jeweiligen Verbindungen in oder aus dem Netzwerk. Doch alle Netzwerk-Firewalls haben gemeinsam, dass sie immer an Netzwerkgrenzen stehen und ein Netz schützen. Aufgrund dieser hohen Last handelt es sich hierbei meistens um Hardware-Firewalls. Die beschriebenen Funktionen können auch in einem Gesamtpaket genutzt werden, welches alle diese Möglichkeiten vereint. Zwei Beispielprojekte sind etwa OPNSENSE und PFSENSE. Diese zwei Lösungen bieten neben der Firewall-Funktionalität noch eine Routing Möglichkeit, das Integrieren von IDS und IPS, sowie die Möglichkeit Erweiterungen zu nutzen.

Da diese Firewalls aufgrund ihres hohen Rechenaufwands und zum Schutz meistens Hardware-Firewalls sind, müssen besondere Maßnahmen beachtet werden. Eine dieser Maßnahmen ist schon umgesetzt, wenn es sich um eine Hardware-Firewall handelt. Die auf der Hardware laufenden Dienste wurden bereits auf ein Minimum beschränkt, um nur die Dienste zu haben, welche wirklich für den Einsatz auf einer Firewall benötigt werden. Dadurch wird die Gefahr von potenziellen Schwachstellen auf ein Minimum reduziert, da ein Dienst, der nicht installiert ist, keine Schwachstellen haben kann.

In einer unsicheren Umgebung sind Firewalls anfälliger für physischen Schaden sowie physische Angriffe. Beide Szenarien können zu einem vollständigen Versagen der Systeme führen oder Sicherheitsmechanismen außer Kraft setzen. Auch die technischen Anforderungen aller benötigten Geräte ist zu beachten. Dazu zählen eine qualitative Stromversorgung, eine Netzanbindung, sowie eine kontrollierte Umgebung. Diese Umgebung wird unter anderem temperaturüberwacht, besitzt eine Notfallstromversorgung und redundante Verbindungen zu externen Netzwerken. Überdies sollte es noch einen Katastrophenschutz geben, etwa bei Feuer oder Überschwemmungen. [63]

2.1.6.6 Firewalls für virtuelle Infrastrukturen

Virtuelle Infrastrukturen können auch mit einer Firewall abgesichert werden. Dabei werden auf einem Server mehrere Betriebssysteme virtualisiert. Jedes dieser Betriebssysteme agiert so, als wäre es das einzige Betriebssystem des Computers. Dadurch kann die Computerhardware besser ausgenutzt werden. Diese Systeme haben auch ein virtuelles Netzwerk, um miteinander zu kommunizieren. Dieser Netzwerkverkehr kann von einer externen Firewall nicht überwacht werden. Deshalb bieten manche Virtualisierungssysteme eine Erweiterung für eine Firewall an.[47, S. 19] Die Alternative zu einer Firewall-Erweiterung wäre die direkte Implementierung einer Firewall in der virtuellen Infrastruktur.

2.1.6.7 Cloud Firewall

Durch Cloud Firewalls werden Dienste angebunden, welche hochverfügbar sind. Die Dienste und Firewalls können auf mehrere Server und Standorte verteilt und dynamisch skaliert werden. Dies ist etwa nötig, wenn die Dienste zu wenig Ressourcen haben. Zudem hilft eine Verteilung auf mehrere Standorte dem Ausfall eines Standortes entgegenzuwirken und reduziert die Antwortzeit der Dienste bei einem standortnahen Zugriff. Auf diese Weise können selbst große **Distributed Denial of Service (DDoS)**-Angriffe abgewehrt werden. Der Nachteil ist, dass mit dem Netzwerkaufkommen auch die Kosten für die Cloud steigen. Server-Hoster bieten oftmals Firewall Lösungen an, um die bei ihnen gemieteten Server und Dienste zu schützen. Nachteilig ist, dass die Firewallschnittstelle im Internet steht und auch von potenziellen Angreifern erreichbar ist. Allerdings ist die Hochverfügbarkeit und der geringe Managementaufwand vorteilhaft. Der reduzierte Verwaltungsaufwand führt allerdings dazu, dass es weniger Konfigurationsmöglichkeiten gibt und einige Fälle nur schwer oder gar nicht konfiguriert werden können.

Es gibt auch Anbieter, welche sich auf **DDoS-Verteidigung** spezialisiert haben, wie **Cloudflare** oder der deutsche Anbieter **Link11**. Diese leiten den Datenverkehr über ihre eigenen Systeme weiter und versuchen Angriffe zu erkennen sowie diese abzuwehren. Dazu bieten diese Anbieter auch unter anderem **Content Delivery Network (CDN)** Dienste an, und verteilen so die Daten auch während eines Angriffes. **CDN-Dienste** allein sind mächtig genug, ein hohes Aufkommen an Anfragen zu bearbeiten. Die Analyse von Angriffen und die Prävention vermindern die Angriffe noch weiter. Für die Analyse werden dann unter anderem die Antwortzeit oder Anomalien wie häufiges Auftreten von Fehlermeldungen gemessen.[22][61]

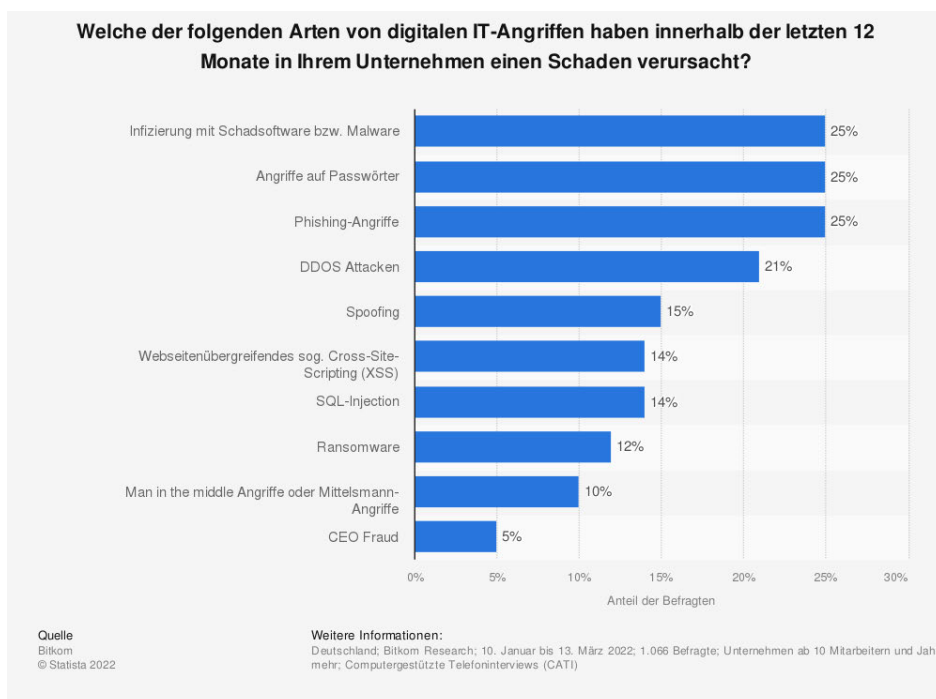


Abbildung 2.3: Schadensverursachende Angriffe[13]

2.1.6.8 High Availability Firewall

Die **High Availability (HA)** Firewall ist ein Cluster-Firewall Aufbau. Diese Konfiguration wird gewählt, um den Single Point of Failure zu eliminieren. Hierbei gibt es mehrere Firewalls die zusammengeschaltet werden. Um einen Ausfall zu vermeiden, schicken die Firewalls einander einen sogenannten Herzschlag. Fällt nun eine Firewall aus, kann eine andere übernehmen, ohne dass es zu einem Ausfall kommt. Die verschiedenen Verfahren für den Aufbau in der **Tabelle 2.1** zeigen, dass die Firewalls aktiv oder passiv sein können. Ist eine Firewall aktiv, läuft sie die ganze Zeit mit und ist sofort einsatzbereit, wenn eine andere ausfällt. Der Vorteil ist die sofortige Einsatzbereitschaft, allerdings muss die Firewall immer laufen. Dies führt zu hohen Strom- und Verschleißkosten für eine Firewall, die nicht aktiv genutzt wird. Die passive Firewall hat den Vorteil, dass sie kostengünstiger ist, benötigt dafür allerdings Zeit zum Starten. Dadurch ist die Verfügbarkeit der Infrastruktur beeinträchtigt. Die letzten Verfahren benutzen mehrere Systeme und sind für große Infrastrukturen gedacht. Die Funktionsweisen der einzelnen Firewalls unterscheiden sich nicht wesentlich von den vorher beschriebenen Firewalltypen. Allerdings funktionieren sie ähnlich wie die bereits erwähnten. Das letzte Verfahren, das sogenannte „N to N“, ist für den Einsatz in Clouds gedacht. Denn hier spielt lediglich die Lastverteilung eine Rolle. Ist diese zu groß, kann hochskaliert werden. Das bedeutet, die Systeme laufen für den Fall eines Ausfalls auf Hochlast, doch eine neue Firewall kann schnell erstellt werden. Viele Cloud Provider greifen für den Ernstfall noch auf externe **CDN** zurück, falls es ein zu hohes Aufkommen gibt. So ist es extrem unwahrscheinlich, die Grenzen zu erreichen.

Aufbau	Beschreibung
Active/Passive	Jeder aktive Knoten hat eine redundante Firewall, welche online geht, wenn die aktive Offline geht.
Active/Active	Geht der Knoten offline, übernimmt ein anderer.
N+1	Eine Gruppe von N Knoten hat einen Reserve-Knoten, welcher die Aufgaben des ausgefallenen übernimmt.
N+M	Eine Gruppe von N Knoten hat M Reserve Knoten, welche die Aufgaben ausgefallener übernehmen können.
N to N	In dieser Konfiguration wird Load-Balancing benutzt, um die Arbeit untereinander zu verteilen.

Tabelle 2.1: High Availability

Weitere Vorteile des Clusterings sind die hohe Verfügbarkeit, da die Ausfallzeit durch die angewandten Mechanismen minimal gehalten wird. Des Weiteren ermöglichen die gezeigten Verfahren eine Skalierung der Firewalls und der dahinter liegenden Systeme. Dies führt zu einer besseren Performanz aller Systeme. Als letztes gibt es noch die bessere Wartbarkeit der Systeme, da durch das Warten eines Systems die anderen noch zur Verfügung stehen und somit keinen Ausfall erzeugen.

Für dieses Clustering müssen zusätzlich noch zwei [Application Delivery Controller \(ADC\)](#) implementiert werden. Diese müssen vor und hinter das Firewall-Cluster gesetzt werden. Es handelt sich dabei um eine erweiterte Load-Balancing Lösung. Dabei muss diese Aufgaben wie asymmetrisches Routing, Verschlüsselung und die Skalierbarkeit des Clusters beherrschen.[7]

2.2 Platzierung von Servern

Die richtige Platzierung von Servern in einer Firewall Umgebung ist essenziell. Es gibt hierbei in einem großen Netzwerk sehr viele unterschiedliche Zonen mit unterschiedlichen Funktionen. Ein Verfahren, welches häufig in Netzwerken benutzt wird, ist das [Demilitarised Zone \(DMZ\)](#) Modell. Bei diesem Modell stehen zwei Firewalls an den Netzwerkgrenzen dreier Netzwerke. Die erste Firewall, auch als [Bastion](#) bezeichnet, steht an der Grenze zwischen dem Internet und der Firmeninfrastruktur. Hinter dieser Firewall ist das gleichnamige Netz, das [DMZ](#) Netz. In diesem stehen Server, die aus dem Internet erreichbar sein sollen, sowie von den Mitarbeitern. Beispiele für Server, die von außerhalb und innerhalb des Netzwerkes erreichbar sind, sind Web Server, E-Mail Server und Verzeichnisserver. Aus dem Internet erreichbare Server dürfen nicht in geschützten Netzwerken stehen, da sie am anfälligsten für Angriffe sind, sollten sie immer in der [DMZ](#) sein. Die zweite Firewall, auch als [Choke](#) bezeichnet, steht zwischen diesem Netzwerk und dem internen Netzwerk. In dem internen Netzwerk befinden sich die Rechner, die für die Arbeit benötigt werden. Dazu gehören unter anderem die Rechner der Mitarbeiter. Durch diesen Aufbau müssen für die zwei Firewalls unterschiedliche Richtlinien erstellt werden. Beispielsweise kann hier mit Reject-Regeln gearbeitet werden, da das interne Netzwerk vertrauenswürdiger ist als das Internet. Dadurch, dass die Reject-Regeln Fehlermeldungen zurückschicken, ist eine Diagnose innerhalb des Netzwerkes einfacher. Bei der Planung muss auch die Sensivität der Daten, sowie die Netzwerklast berücksichtigt werden. Interne Server sollten in geschützten Netzwerken stehen, welche von außerhalb nicht erreichbar sind höchstens für Updates. Außerdem müssen sowohl interne als auch öffentliche Server so eingerichtet werden, dass eine Kompromittierung möglichst keinen Einfluss auf andere Systeme haben kann. Ein wichtiger Fall sind [VPN](#)-Server. Diese werden in der [DMZ](#) aufgestellt, da sie zum einen von außerhalb erreichbar sein müssen, zum anderen damit sie Zugriff auf das interne Netzwerk haben. Zudem werden die Verbindungen zwischen dem internen Netzwerk und dem [VPN](#)-Server von der Firewall geprüft, bevor der [VPN](#) diese verschlüsselt. Wichtig ist auch der Umgang mit [IPv4](#) und [IPv6](#). Es muss überprüft werden, wie mit diesen Verfahren umgegangen werden soll. Beispielsweise kann die Verbindung von außerhalb [IPv6](#) nutzen, aber innerhalb gibt es der Einfachheit nur [IPv4](#). Dann ist es etwa sinnvoll, eine Umwandlung von [IPv6](#) zu [IPv4](#) vorzunehmen. Oder ob beide genutzt werden sollten. In diesem Fall müssen aber Regeln für beide Versionen erstellt werden. [26][63][24, S. 57]

2.3 Richtlinien

Um Firewallregeln umzusetzen, sind entsprechende Richtlinien notwendig. Diese können aus gesetzlichen Vorgaben oder Regularien entstehen und müssen für die Firmen angepasst werden. Anforderungen können nicht nur aus gesetzlichen Vorgaben hervorgehen, sondern

auch im Rahmen von Security-Audits oder Zertifizierungsprozessen von Unternehmen aufgenommen werden. Diese Anforderungen werden meist in schriftlicher Form bereitgestellt und lassen sich in organisatorisch und technisch umzusetzende Anforderungen unterteilen. Da sich diese Arbeit mit der Automatisierung von Firewall-Richtlinien beschäftigt, wird im Weiteren auf technisch umzusetzende Anforderungen eingegangen. Der Prozess des Umwandelns der Richtlinien in ein maschinenlesbares Format ist kein triviales Problem. Näheres hierzu ist in Kapitel [Abschnitt 3.1](#) auf Seite [21](#) nachzulesen.

3 Lösungskonzeption

Im Folgenden wird eine Möglichkeit für eine automatisierte Lösung des Prozesses für die Anpassung von Firewallregeln erläutert. Die Lösung der in der Problemanalyse genannten Probleme, siehe [Abschnitt 1.2](#) auf Seite 3, soll es sein, diesen langwierigen Prozess zu vereinfachen, siehe [Abbildung 3.1](#). Wie anhand des Schaubildes zu sehen, ist konnte der Prozess

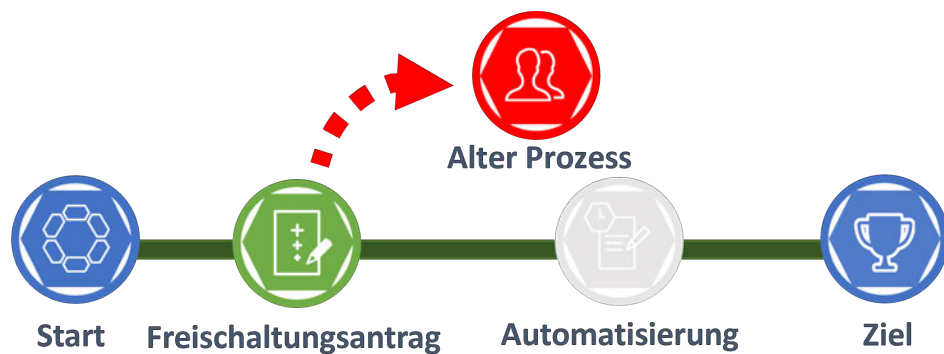


Abbildung 3.1: Optimierter Freischaltungsantrag (Erstellt aus Unternehmenstemplate)

vereinfacht und die menschliche Interaktion reduziert werden. Wie auch bei dem manuellen Prozess, siehe [Abbildung 1.1](#) auf Seite 2, muss der Nutzer einen Antrag stellen, doch wird dieser ohne weitere menschliche Interaktion automatisiert. Die Anfrage soll über ein Formular entgegengenommen werden, welches alle für die Freischaltung wichtigen Informationen abfragt. Dieses soll dann an die entwickelte Automatisierungslösung geschickt werden, die die Anfrage entgegennimmt. Dadurch soll dann ein neues [Ticket](#) in einem Ticketsystem mit der Anfrage erstellt werden. Der Automatisierungsserver wird nun durch einen [Webhook](#) kontaktiert und der Prozess zur Freischaltung wird gestartet.

Die benötigten Informationen werden extrahiert, um sie mit den Unternehmensrichtlinien zu vergleichen und die Anfrage zu validieren. Eine weitere menschliche Interaktion entsteht nur dann, wenn Fehler auftreten oder die Unternehmensrichtlinie verlangt, dass der Antrag von einem Mitarbeiter manuell validiert werden muss, beispielsweise um einen Antrag für ein sicherheitskritisches Netz zu validieren. In diesem Fall muss in den alten Prozess gewechselt werden. Hierfür muss es möglich sein, mit einer Ausnahmegenehmigung eine Firewallregel zu erstellen. Um dies durchzusetzen, muss der Administrator eine Ausnahmegenehmigung erstellen, welche vom System ausgewertet und anschließend angewendet werden kann. Um zu verhindern, dass diese Ausnahmegenehmigung manipuliert wird, soll diese signiert werden. Durch diese Signatur kann überprüft werden, ob die Ausnahmegenehmigung gefälscht wurde.

Die Berechtigungen werden automatisch geprüft und über ein Nutzermanagementsystem abgefragt. Dieses kann zudem einfach aktualisiert werden. Es gibt zwei interessante zur Verfügung stehende Verfahren. Das erste Verfahren ist das [Role Based Access Control \(RBAC\)](#) Verfahren. Bei diesem werden Rollen für das Berechtigungsmanagement erstellt. Zwei der zu erstellenden Rollen könnten Entwickler und IT-Support sein. Dem Entwickler kann erlaubt werden, ausschließlich Webserver zu erstellen. Der IT-Support kann etwa die Berechtigung

erhalten, auch Regeln für andere Protokolle oder Server zu erstellen, da sie diese Systeme warten und erstellen müssen. Zusätzlich gibt es das [Separation of Duty \(SOD\)](#) Verfahren. Dieses verhindert die Kombination von spezifizierten Rollen Ein Beispiel, welches sich nicht auf diese Arbeit bezieht, ist etwa die Berechtigung, eine Zahlung zu tätigen und zu autorisieren.

Ein zweites Verfahren, welches angewandt werden kann, ist [ABAC](#). Dieses Verfahren nutzt Attribute, siehe [Abbildung 3.2](#), um einen Vorgang zuzulassen oder abzulehnen. Auf eine Fire-



Abbildung 3.2: Aufbau von [ABAC](#) (Mit Drawio erstelltes Diagramm)

wallrichtlinie angewandt, könnte ein Anwendungsfall wie folgt aussehen: Es gibt einen Nutzer John, welcher die Rolle Administrator und das Sicherheitslevel vertraulich hat. Seine Umgebungsattribute sind 22 Uhr und Ort Zuhause. Die Dateiattribute werden der Einfachheit wegen gelassen, doch können Attribute der jeweiligen Regel sein, da es hier keine Dateien gibt. Nun kann geprüft werden, ob alle Attribute akkurat sind. Die Zugriffskontrolle wertet dies dann aus. Beispielsweise sind die Nutzerattribute Administrator und Sicherheitslevel vertraulich korrekt und werden von der Zugriffskontrolle akzeptiert, allerdings nicht die Nutzerattribute. Dem Nutzer sollte es nicht möglich sein, außerhalb des firmeninternen Netzwerkes und außerhalb der Geschäftszeiten auf die Firewallregeln zuzugreifen. Solche sensiblen Konfigurationen sollten nur innerhalb eines sicheren Netzwerkes und innerhalb der Geschäftszeiten erfolgen. Das Schaubild zeigt ein Beispiel mit Beispielattributen, siehe [Abbildung 3.2](#). Ein Vorteil beider Verfahren ist, dass es nicht nötig ist, jedem einzelnen Nutzer Berechtigungen erteilen zu müssen.[11]

Die zuvor beschriebene Abfrage der Daten für die Freischaltung soll anhand der Unternehmensrichtlinien abgefragt werden. Dafür sollen die Unternehmensrichtlinien mithilfe einer [Policy as Code \(PaC\)](#) Lösung implementiert und gewartet werden. Dadurch soll gewährleistet werden, dass die Unternehmensrichtlinien immer ordnungsgemäß und einheitlich angewandt werden. Der Nachteil ist, dass die Richtlinien eindeutig implementiert werden müssen. Jedoch ist diese Implementierung nur einmal pro Firma notwendig. Zudem muss sich nur ein Mitarbeiter oder ein kleines Team von Experten für Compliance Anforderungen mit diesem Thema beschäftigen. Basierend auf den Unternehmensanforderungen und anderen Anforderungen, wie denen der ISO 27001, werden die Regeln erstellt. Ein Administrator muss diese meist umfangreichen Anforderungen prüfen und in die Praxis umsetzen. Sind diese Anforderungen sauber implementiert, sind sie wartungsfreundlich und neue Anforderungen können schnell umgesetzt werden. Eine Wartung muss regelmäßig erfolgen, um die Richtlinien aktuell zu halten und möglicherweise neue Sicherheitslücken zu vermeiden.

Die Unternehmensrichtlinien sollen mit einem [PaC Server](#) validiert werden. Ein Ergebnis, ob die Freischaltung erfolgreich war, soll dann zurückgegeben werden. Basierend auf diesem Ergebnis soll umgehend die Regel auf der Firewall implementiert werden, entweder durch einen direkten Befehl an die Firewall oder durch eine [Application Programming Interface \(API\)](#) Schnittstelle. Zudem soll der Nutzer über das Ticketsystem eine Rückmeldung erhalten.

Allerdings muss bei der Freischaltung der Firewallregeln beachtet werden, dass viele Unternehmen aufgrund der [IP-Knappheit NAT](#) benutzen, siehe [Abschnitt 2.1.5](#) auf Seite 8. Das führt dazu, dass neben einer [Port-Freischaltung](#) eine [Port-Weiterleitung](#) nötig ist. Dazu soll es die Möglichkeit geben, die [Ports](#) an Server innerhalb des Unternehmensnetzwerkes weiterzuleiten. Nur wenn die [Port-Freischaltung](#) und die [Port-Weiterleitung](#) beide korrekt beantragt wurden, sind die Server zu erreichen. Dies beschränkt sich allerdings lediglich auf [IPv4](#), bei [IPv6](#) ist das Nutzen von [NAT](#) nicht nötig, da es genug [IP-Adressen](#) für alle im Moment existierenden Geräte gibt. Auch muss hier sichergestellt werden, dass es möglich ist aus dem Unternehmensnetzwerk Verbindungen von den dynamischen Ports zu festgelegten [Ports](#) im Internet zu unterstützen. Dies ist unter anderem wichtig, um Updates zu installieren.

Das Endergebnis soll ein Prozess sein, der die menschliche Interaktion so weit wie möglich reduziert, um dabei den Freischaltungsprozess effektiver umzusetzen. Doch nicht nur der Prozess wird bei diesem Konzept verbessert, auch die Wartung der Firewallregeln kann durch diesen Prozess verbessert werden. Auch das Verwalten der Firewallregeln kann mithilfe des entwickelten Programmes verbessert werden.

3.1 Firewallrichtlinie

Zur Inbetriebnahme dieses Programmes müssen Richtlinien erstellt werden. Allerdings ist es nicht möglich, Standardrichtlinien zu erstellen, da Unternehmen unterschiedliche Anforderungen und Vorgaben haben. Die Richtlinien müssen auf die verwendete Technik und auf länderspezifische Vorgaben abgestimmt sein. Um dies umzusetzen, sind im folgenden Minimalanforderungen an eine Richtlinie aufgezeigt, die von den Unternehmen umgesetzt

werden sollten, um eine sichere Infrastruktur aufbauen zu können. Diese Richtlinien sollen als Grundgerüst dienen und mit den Unternehmensanforderungen und den länderspezifischen Anforderungen erweitert werden.

Die Infrastruktur und das Unternehmen selbst müssen eine Reihe von Richtlinien einhalten, um einen sicheren Betrieb zu gewährleisten. Die meisten Anforderungen stammen dabei vom [Bundesamt für Sicherheit in der Informationstechnik \(BSI\)](#), da diese Vorgaben in Deutschland umgesetzt werden müssen. Die Anforderungen beschreiben theoretische Vorgaben und müssen entweder organisatorisch oder technisch umgesetzt werden. Die Unternehmen müssen selbst entscheiden, wie die Anforderungen für sie am besten umgesetzt werden können. Diese Vorgaben können mit einer Zertifizierung des Grundschutzes nachgewiesen werden. Dabei kann entweder die gesamte IT-Infrastruktur oder ein einzelnes Produkt zertifiziert werden. Ein vom [BSI](#) bevollmächtigter Prüfer bescheinigt diese Vorgaben anhand eines Anforderungskataloges. Diese Zertifikate sind vorrangig für Kunden, Geschäftspartner oder Versicherungen wichtig, um nachzuweisen, dass für eine ausreichende Sicherheit gesorgt wird. Einige Unternehmen verlangen sogar bestimmte Zertifizierungen, bevor sie einem anderen Unternehmen Zugriff auf Teile ihrer Infrastruktur geben.

Die Richtlinienerstellung muss für jeden Fall genau abgewägt werden. Doch die Richtlinien, die im Folgenden beschrieben werden, sind allgemeine Richtlinien, welche von allen Unternehmen genutzt werden können und sollten. Es gilt: „So wenig wie möglich, so viel wie nötig.“^[27] Das heißt ein Benutzer oder ein Programm soll nur so viele Berechtigungen bekommen, wie wirklich nötig sind. So benötigt ein Videoplayer nicht die Berechtigung, Daten zu ändern oder auf das Internet zuzugreifen. Dies ist nicht explizit erlaubt und sollte auch nicht möglich sein. Zudem sollten nicht benötigte Dienste abgeschaltet und von den Systemen entfernt werden. Des Weiteren sollte die Administration der Firewall nur aus einem sicheren Netzwerk erfolgen, beispielsweise aus einem Administrationsnetzwerk. Der Zugriff sollte zudem durch eine Authentifikation geschützt werden. Zudem sollte die Software der Firewalls regelmäßig aktualisiert werden. Sicherungskopien sollten regelmäßig angefertigt und auf externen Servern gespeichert werden. Es sollten redundante Sicherheitsstrukturen erstellt werden, um zu gewährleisten, dass ein Ausfall in vertretbarer Zeit kompensiert werden kann. Es muss dafür gesorgt werden, dass die Kompromittierung eines Systems nicht zur Kompromittierung der gesamten IT-Infrastruktur führt. Zudem darf keine schlecht geschützten Hintertüren geben, die die Sicherheitsmechanismen umgehen. Die Schutzmaßnahmen müssen ausgewogen und aufeinander abgestimmt sein. Ein ausgewogenes Verhältnis zwischen Sicherheit und Nutzbarkeit sollte jedoch angestrebt werden. Ein anschauliches Beispiel dafür ist das Blockieren von IP-Adressen. Je mehr IP-Adressen gesperrt werden, desto sicherer wird das Netzwerk. Allerdings schränkt das auch den Zugriff auf Informationen im Netzwerk ein. Wird etwa „<https://www.exploit-db.com/>“ blockiert, wird das Netzwerk vor dem Downloaden von Exploits geschützt. Allerdings macht dies unter anderem die Arbeit eines Pentesters schwieriger, der einen Pentest auf eine Anwendung oder das Netzwerk eines Kunden durchführen will. ^[27]

Die genutzten Richtlinien sind dabei entweder statisch oder dynamisch. Statische Richtlinien arbeiten feste Parameter ab, unter anderem die Sperrung von bestimmten Netzwerkressourcen oder die Verweigerung der Anmeldung in der Nacht. Dynamische Richtlinien passen sich der aktuellen Lage an. Dies betrifft unter anderem den Ausfall eines Routers oder das

Verlieren von Netzwerkpaketen. Wenn einer der Fälle eingetroffen ist, kann etwa das Video Streaming verboten werden, um für Stabilität im Netzwerk zu sorgen. Außerdem kann auch kategorisiert werden, unter anderem erhalten kritische Anwendungen eine höhere Priorität und die interne Unternehmenswebsite wird weniger hoch kategorisiert.[54]

3.1.1 Erstellung einer Richtlinie

Um eine Richtlinie zu definieren, müssen zunächst einige Anforderungen aufgestellt werden. Als Erstes müssen die Netzwerkgeräte identifiziert werden, um zu wissen, auf welche Geräte die Richtlinie angewandt werden soll. Als Nächstes müssen diese Geräte auf Applikations-schwachstellen analysiert werden. Nachfolgend muss der Kostenaufwand analysiert werden und der Nutzen des Sicherns dieser Schwachstellen. Außerdem muss eine Netzwerkanalyse durchgeführt werden, um zu analysieren, wie das Netzwerk aufgebaut ist und wie dieses selbst verbessert werden kann.

Unabhängig von der Art der Firewall-Richtlinie oder des Netzwerks sollte die Firewall immer standardmäßig eine Deny oder Block-Richtlinie haben. Es muss darauf geachtet werden, nur das zuzulassen, was notwendig ist und alles, was nicht benötigt wird, zu blockieren. Zudem ist es wichtig darauf zu achten, ob die Regeln für IPv4 oder IPv6 benötigt werden. Die Firewallrichtlinien sollten dafür die Quell und die Ziel IP-Adresse enthalten, den Typ des Netzwerkverkehrs, die genutzten Protokolle, die Ports. Zusätzlich müssen die Richtlinien die Interfaces und eine Aktion für die Richtlinien, wie erlauben, verweigern oder blockieren enthalten.

Eingehender Netzwerkverkehr auf der Firewall sollte immer unterbunden werden, sofern es sich nicht um ein autorisiertes Gerät handelt. Er sollte nur erlaubt sein, wenn das entsprechende Gerät autorisiert ist, Änderungen vorzunehmen. Eingehender Netzwerkverkehr, der von innerhalb des Netzwerks zu kommen scheint, das sogenannte **Spoofing** sollte verhindert werden. **ICMP** Netzwerkpakete aus nicht vertrauenswürdigen externen Netzwerken sollten nicht zugelassen werden, wenn dies nicht unbedingt nötig ist. Außerdem sollten nach dem **RFC 1918** als privat eingestufte Netzwerke eingehend und ausgehend blockiert werden. Da diese Netzwerke privat sind gibt es keine eingehenden oder ausgehenden Pakete aus oder in das Internet. Diese Netzwerke sind nur zur Kommunikation innerhalb eines Netzwerkes gedacht. Es gibt drei private Netzwerke, wobei das erste das Klasse A Netzwerk 10.0.0.0 ist. Das zweite Netzwerk der Klasse B ist 172.16.0.0 und das dritte ist das Klasse C Netzwerk 192.168.0.0. Auch sollten Netzwerkpakete von einem nicht autorisierten System, welches **Simple Network Management Protokoll (SNMP)**-Netzwerkpakete verschickt, geblockt werden. Zudem sollten eingehende Netzwerkpakete mit **IP Source Routing** Informationen blockiert werden. Diese haben die Möglichkeit, Pakete zu erstellen, welche die Firewall umgehen. Des Weiteren sollten eingehende oder ausgehende Netzwerkpakete aus dem lokalen Netzwerk blockiert werden. Zudem sollten eingehende oder ausgehende Netzwerkpakete mit Quell oder Ziel IP-Adresse 0.0.0.0 abgelehnt werden. Netzwerkpakete zu oder von Broadcast Adressen sollten ebenfalls blockiert werden. Mithilfe dieser Adressen ist es möglich **Broadcast Propagation**-Angriffe wie die **Smurf Attacke** auszuführen. Falls es möglich ist Dienste nutzerspezifisch zu erlauben, sollte dies ebenfalls eingerichtet werden. Dadurch ist es möglich, den Zugriff auf Ressourcen noch weiter einzuschränken. Das Erstellen von Alarmen und das

Protokollieren sollte ebenfalls erfolgen, sofern dies vorhanden ist. Außerdem muss geprüft werden, wie mit verschlüsseltem Datenverkehr umgegangen werden muss. Hier ist es möglich, die Verschlüsselung aufzubrechen, um den Datenverkehr zu analysieren. Das erhöht die Sicherheit des Netzwerkes, vermindert allerdings die Privatsphäre. Zudem bietet die Firewall Angreifern damit die Möglichkeit den Datenverkehr zu entschlüsseln, sollten Angreifer Zugriff auf die Firewall erhalten. Zudem ist es wichtig, die Firewallregeln regelmäßig auf ihre Aktualität zu prüfen und auf den neuesten Stand zu bringen. Alte, nicht mehr benötigte Regeln müssen gelöscht werden. [26] Bei der Erstellung der Richtlinien muss beachtet werden, dass alle Richtlinien auch in IPv6 umgesetzt werden sollten. Auch sollten Tunnelprotokolle wie „6-to-4“ und „4-to-6“ blockiert werden, wenn sie nicht benötigt werden. Die einzige Ausnahme ist, wenn IPv6 vom Unternehmen nicht benutzt wird, dann sollte sämtlicher Netzwerkverkehr blockiert werden.[47, S. 28, 30] Darüber hinaus sollte eine geeignete Teststrategie erstellt werden, die die erstellten Regeln prüft und evaluiert.[63] Die Logs sollten geprüft und an eine zentrale Instanz geschickt werden. Dies kann eine [Security Information and Event Management \(SIEM\)](#)-Lösung oder eine Log-Verwaltungslösung für Forensik und Berichterstellung sein.[1] Gleichzeitig muss darauf geachtet werden, dass die Firewall ständige Sicherheitsupdates bekommt.[1] Zugriffe auf Accounts der Firewall müssen gesichert werden, etwa durch den Einsatz von [Multi Factor Authentication \(MFA\)](#). [1] Anschließend sollten nur benötigte Protokolle von der Firewall erlaubt werden, wie [ICMP](#), [TCP](#) und [UDP](#). Andere Protokolle sollten so weit wie möglich eingeschränkt und nur bei Geräten zugelassen werden, die diese Protokolle zwingend benötigen. [Internet Group Management Protocol \(IGMP\)](#) ist ein Protokoll, das selten für die Kommunikation zwischen dem internen Netzwerk und dem Internet genutzt wird und deshalb blockiert werden kann. Beispielsweise könnte lediglich der Firewall erlaubt werden, das [VPN](#) Protokoll zu nutzen.[47, S. 28]

Neben diesen Richtlinien ist es auch wichtig, die unprivilegierten [Ports](#) nicht zu vernachlässigen. So gibt es etwa den [Port](#) 1241. Dies ist ein [Port](#) vom Nessus Schwachstellenscanner. Hat ein Angreifer Zugriff erhalten, kann er legitime Schwachstellenscans ausführen oder alte einsehen. Deshalb sollte eine ausgehende Verbindung zu diesem [Port](#) blockiert werden. Clients, welche den [Port](#) 1241 zum Verbindungsaufbau nutzen sollten nicht blockiert werden. Zum Beispiel wird beim [HTTP](#) Protokoll ein zufälliger unprivilegiertes [Port](#) genutzt, um zu dem [HTTP Port](#) zu verbinden. Ähnlich ist es bei [X11 Display Servern](#), allerdings müssen hier sowohl eingehende als auch ausgehende Verbindungen blockiert werden. Dies ist nötig, da [X11 Display Server](#) Server auch einen [SSH](#) Eingang bieten. Dafür werden die Ports 6000 bis 6063 benutzt.

Daneben gibt es auch [UDP](#) Dienste auf unprivilegierten [Ports](#). Hier kann allerdings nicht erkannt werden, ob das Paket eine Serverantwort oder eine Client-Anfrage ist. Deshalb sollten hier alle unprivilegierten [Ports](#) gesperrt und nur falls nötig bestimmte Services freigeschaltet werden wie [Network File System \(NFS\)](#), dieses benutzt den [Port](#) 2049. Da [NFS](#) üblicherweise nur lokal verwendet wird, kann es gesperrt werden.[26]

Außerdem zu beachten ist der richtige Umgang mit [ICMP](#). Das Blockieren von [ICMP](#) kann die Performanz sowie die Diagnostik einschränken. Es sollte darauf geachtet werden, alle Arten von [ICMP](#) zu blockieren, welche nicht durch die Firma explizit freigegeben wurden.[47, S. 31] Ein Freigeben von [ICMP](#) Typ 4 sollte auch vorgenommen werden, da dies signalisiert, dass der Server überlastet ist und dem Client signalisiert, dass er die Kommunikation reduzieren und

auf weitere Antworten warten soll. Diese Netzwerkpakete sind Beispiele für das Verbessern der Performanz, da der Server durch die Typ 4 Netzwerkpakete eine Überlastung vermeiden kann. Dieses Verfahren wird auch „Source-Quench“ genannt. [26] Des Weiteren gibt es noch die sogenannten zwei Pakettypen „Echo Request“ Typ 8 und den „Echo Reply“ vom Typ 0. Sie werden zur Diagnostik benutzt. Es ist ratsam diese nicht für aus dem Internet kommende Verbindungen zuzulassen, da diese sensible Daten über das Netzwerk preisgeben können. Innerhalb des Netzwerkes sollten sie allerdings erlaubt werden, um eine bessere Diagnostik zu gewährleisten. Eine Verbindung zu Diensten außerhalb sollte für ICMP Typ 8 ausgehend und Typ 0 eingehend erlaubt werden. Auf diese Weise kann geprüft werden, ob Dienste außerhalb erreichbar sind. [26] Ein weiteres wichtiges Paket ist das sogenannte „Time-Exceeded“ Paket vom Typ 11. Dieses weist darauf hin, dass die Lebensdauer des Pakets überschritten ist. Die sogenannte **Time to Live (TTL)** hat den Wert null erreicht. Das Programm Traceroute wurde gezielt entwickelt, um mit diesem Paket zu arbeiten. Der Benutzer muss dem Programm ein Ziel nennen. Nun schickt Traceroute ein Paket an das Ziel. Die **TTL** wird dabei auf eins gesetzt und mit jedem weiteren Paket erhöht. So ist es möglich, die Zwischenstationen eines Netzwerkpaketes zu ermitteln, da diese immer das ICMP Paket vom Typ 11 zurückschicken, sobald die **TTL** bei ihnen abgelaufen ist. In der Abbildung **Abbildung 3.3** ist ein Beispiel für einen Traceroute-Aufruf auf ubuntuusers.de zu sehen. Es ist zu sehen, dass als Erstes die

```
darks1de@bravo:~$ traceroute ubuntuusers.de
traceroute to ubuntuusers.de (87.79.26.37), 30 hops max, 60 byte packets
 1 192.168.178.1 (192.168.178.1) 0.490 ms 0.624 ms 0.771 ms
 2 192.0.0.1 (192.0.0.1) 8.432 ms 8.474 ms 8.508 ms
 3 62.214.32.83 (62.214.32.83) 11.226 ms 11.247 ms 11.273 ms
 4 * * *
 5 * * *
 6 * * ip-core-eup2-ae3.netcologne.de (195.14.195.53) 14.725 ms
 7 * * *
 8 * * *
 9 * * *
10 * * *
11 * * *
12 ha.ubuntu-de.org (87.79.26.37) 15.079 ms * *
```

Abbildung 3.3: Traceroute zu ubuntuusers.de(Bildschirmfoto von Traceroute)

Station mit der IP-Adresse 192.168.178.1 passiert wird. Bei dieser handelt es sich um den eigenen Router, welcher immer passiert werden muss, wenn in ein anderes Netzwerk verbunden wird. Die mit Sternchen bezeichneten Zwischenstationen senden kein Paket des Typs 11 zurück. Anhand der Bezeichnung der letzten Station lässt sich ableiten, dass es sich um einen **HAProxy** handelt. Von außen kommende Anfragen, sollten beschränkt werden, sodass diese keine sensiblen Informationen aus dem Netzwerk entdecken können. [26]

Die **ICMP** Netzwerkpakete vom Typ 3 sollten innerhalb des Netzwerkes und für öffentliche Server aktiviert werden. Sie signalisieren, dass das Ziel nicht erreichbar ist, auch mit „Destination-Unreachable“ bezeichnet. Durch dieses Netzwerkpaket muss der Client keine erneuten Verbindungsversuche unternehmen. Dadurch führt dieses Paket zur Vermeidung von Netzwerkverkehr und das Routing wird effizienter.[26]

Die **ICMP** Pakete mit dem Typ 3 Code 4 „Destination Unreachable Fragmentation Needed, DF Set“ sollten ebenfalls erlaubt werden. Denn diese geben an, dass die maximale Paketgröße, auch **Maximum Transmission Unit (MTU)** genannt, überschritten wurde. Ein Verfahren, welches diese Pakete nutzt, ist **Path Maximum Transmission Unit (PMTU)**. Dieses versucht

die größtmögliche **MTU** herauszufinden, die über eine Verbindung möglich ist. Als Erstes erzeugt das **PMTU** Verfahren ein Paket mit der maximalen Größe. Dieses wird dann an das Ziel geschickt. Überschreitet dieses Netzwerkpaket die maximal zulässige **MTU** einer der Zwischenstationen wird das **ICMP** Typ 3 Code 4 Paket und die maximale **MTU** zurückgeschickt. Dieser Vorgang wird so lange wiederholt, bis das Ziel erfolgreich erreicht wurde. Das bedeutet, die Station mit der kleinsten **MTU** gibt die Größe der Netzwerkpakete an.[38]

Die **ICMP** Pakete sind hilfreich, um Netzwerkprobleme zu diagnostizieren. Allerdings werden sie häufig blockiert, da sie für Angreifer ebenfalls sehr viele Informationen über den Aufbau des Netzwerkes bereitstellen. Einige Funktionen in **IPv6** benötigen gleichermaßen bestimmte **ICMP** Typen, um funktionieren zu können. Hierzu sollte die **RFC 4890** herangezogen werden, um zu entscheiden, welche **ICMP**-Typen sinnvoll sind zu erlauben. [47, S. 31]

Auch das **DNS** muss erlaubt werden, sofern es keinen **DNS** Server innerhalb des Netzwerkes gibt. Dieser Server benötigt dann auch die nötigen Rechte, **DNS** Anfragen öffentlicher Server zu spiegeln, um auf Dienste im Internet zuzugreifen. Besitzt das Unternehmen einen eigenen öffentlichen **DNS** Server, kann dieser auch direkt auf der Firewall, die an der Netzwerkgrenze zum Internet steht, integriert sein oder neben dieser stehen. Es ist auch möglich, einen internen **DNS** Server zu nutzen, der interne Dienste auflöst. Dieser kann dann in der **DMZ** stehen.[26]

3.1.1.1 Portscans

Eine Maßnahme muss ergriffen werden, um Portscans zu unterbinden. Dies kann einen Angriff erschweren. Es gibt verschiedenste Möglichkeiten Portscans durchzuführen. Wichtig anzumerken ist, es wird zwar von Scans gesprochen, doch die aufgeführten Methoden können auch für andere Angriffsarten benutzt werden. Eine weitere Methode, die genutzt werden kann, ist die Datenexfiltration, mit dieser Technik ist es möglich Daten in harmlos aussehenden Paketen zu verstecken. Die **ICMP** Netzwerkpakete haben die Möglichkeit Daten zu senden, allerdings wird dies nicht benötigt und eignet sich deshalb, um hier Daten aus oder einzuschleusen. Außerdem ist es möglich, mit den Netzwerkpaketen **DoS** und **DDoS** Angriffe auszuführen. Eine weitere Möglichkeit ist es, Schwachstellen direkt auf der Firewall auszunutzen. Einige der verwendeten Methoden können verwendet werden um zu prüfen, ob ein Rechner online ist, sowie das verwendete Betriebssystem erkennen.

Eine Chance solche Angriffe und Scans abzuwehren, besteht darin, einen Paketfilter zu verwenden, der alle Netzwerkpakete mit ungültigen Flags verwirft. Dies verhindert den Angriff oder das Scannen mit verschiedenen Kombinationen von Flags. Ein Limit für halb offene und offene Verbindungen sollte gesetzt werden, um Angriffen entgegenzuwirken.[5, S. 6] Es gibt viele verschiedene Arten, wie ein Ziel gescannt werden kann, die in **Tabelle 3.1** auf der nächsten Seite aufgezählt sind. Dazu kommen die **Ports** die gescannt werden können. Der bekannteste Portscanner ist „nmap“. Dieser unterstützt die aufgeführten Scanarten. Allerdings führt „nmap“ standardmäßig den sogenannten **TCP-SYN** Scan aus. Dieser wird auch als „stealth scan“ bezeichnet, da er durch das Beenden der Verbindung vor dem kompletten Aufbau schwieriger zu detektieren ist. Da es standardmäßig nicht vorgesehen ist, dass der Nutzer einen Verbindungsaufbau abbricht, müssen zur Durchführung dieses Scans die Netzwerkpakete, auch „Raw Pakete“ genannt, selbst erstellt werden. Dies ist nur möglich, wenn

der Benutzer diese erweiterte Berechtigung besitzt. Unter Linux wird dazu die sogenannte „CAP NET RAW“ capability benötigt. Liegt diese Berechtigung nicht vor, wird standardmäßig der TCP-Connect Scan ausgeführt. Dadurch, dass er eine vollständige Verbindung aufbaut, ist er allerdings leichter zu detektieren und langsamer. Diese Scans wurden standardmäßig gewählt, da diese meistens zum Erfolg führen, sollte der Port erreichbar sein. Zudem stellen die Entwickler „nmap“ eine Liste der meistgenutzten Ports zur Verfügung. „Nmap“ nutzt diese, um standardmäßig die 1000 Meistgenutzten zu scannen.

Tabelle 3.1: Scanarten

Scantyp	Beschreibung
SYN-Scan	Bittet um einen Verbindungsaufbau, indem das SYN Flag gesetzt ist. Sobald das Ziel signalisiert, dass der Verbindungsaufbau gestartet werden kann wird die Verbindung geschlossen.
TCP-Connect	Baut eine vollständige TCP Verbindung auf.
Xmas-Scan	„Setzt die FIN-, PSH- und URG-Flags und beleuchtet das Paket wie einen Weihnachtsbaum (engl. Xmas).„[41]
Null-Scan	„Setzt keinerlei Bits (der TCP-Flag-Header ist 0).„[41]
FIN-Scan	„Setzt nur das TCP „FIN-Bit.“„[41]
UDP Hosterkennung	Ein (leeres) UDP Paket wird an einen geschlossenen Port gesendet, antwortet dieser mit einer ICMP Nachricht, dass der Port nicht erreichbar ist, ist der Rechner online. Bei unerreichbaren Hosts/Netzwerken oder überschrittener TTL kann darauf geschlossen werden, dass der Rechner aus ist.
UDP Scan	Scannt UDP Ports und wertet anhand von verschiedenen ICMP-Typen den Status des Ports aus.
ICMP Netzwerkpakete	Die verschiedenen Arten von ICMP können genutzt werden zum Scannen und zur Datenexfiltration.
ACK Scan	„Beim Testpaket eines ACK-Scans wird nur das ACK-Flag gesetzt (es sei denn, Sie benutzen –scanflags). Beim Scannen ungefilterter Systeme werden sowohl offene als auch geschlossene Ports ein RST-Paket zurückgeben.“[41] Dies eignet sich gut, um von der Firewall nicht geschützte Ports zu finden.
Stream Control Transmission Protocol (SCTP) Init	TCP-SYN Alternative für SCTP
SCTP Cookie-echo	Netzwerkpakete werden bei offenen Ports nicht beantwortet, bei geöffneten mit einem abort beantwortet.
Scantyp	Beschreibung

Tabelle 3.1: Scanarten: Fortsetzung

Scantyp	Beschreibung
Window Scan	die Fenster-Größe wird analysiert, auf manchen Systemen ist diese bei offenen Ports positiv und negativ bei geschlossenen.
FTP bounce Scan	Angriff, bei dem FTP-Server als Proxy genommen wird.
Service Scan	Ein Scan der versucht den Service, der auf einem Port läuft, anhand unterschiedlicher Netzwerkpakete herauszufinden.
Betriebssystemerkennung	Analyse anhand betriebssystemspezifischer Parameter.
Fragmentation	Netzwerkpakete werden auf viele kleinere aufgeteilt, macht es schwieriger Angriffe zu erkennen, vor allem wenn Netzwerkpakete auf verschiedenen Wegen oder Firewalls ins Netz kommen.
TTL manipulation	Die Lebensdauer wird manipuliert und Netzwerkpakete werden zwischen schädliche, gepackt, um diese zu verschleiern. Aber die Firewall blockiert die zusätzlichen Netzwerkpakete und lässt nur die schädlichen durch.
UDP Paket Storm	„Am Paketfilter, der Server-Dienste schützt, die aus weniger oder nicht vertrauenswürdigen Netzen erreichbar sind, SOLLTEN die sogenannten Rate-Limits für UDP-Datenströme gesetzt werden.“[5, S. 6]
Sequence Number Guessing	„Am äußeren Paketfilter SOLLTE bei ausgehenden Verbindungen für TCP eine zufällige Generierung von Initial Sequence Numbers (ISN) aktiviert werden, sofern dieses nicht bereits durch Sicherheitsproxys realisiert wird.“[5, S. 6]
Scantyp	Beschreibung

[17]

3.1.2 Zusätzliche Firewall Schutzmaßnahmen

Sollte es zu einem Ausfall oder einer Überlastung der Firewall kommen, sollte es eine Failover-Richtlinie geben, welche regelt, was in einem solchen Fall passieren sollte. Meistens werden die Firewalls redundant platziert, sodass eine Firewall den Netzwerkverkehr übernimmt oder sie sich den Netzwerkverkehr teilen. Dazu können die Firewalls zusammengeschaltet und die Regeln auf die Failover-Firewall gespiegelt werden. Eine Vorgehensweise muss erstellt werden, die die Handlungsweise bestimmt, wenn es zu einer Überlastung des Systems kommt. Hier kann beispielsweise bestimmt werden, dass alle Pakete akzeptiert werden, um eine hohe Verfügbarkeit zu gewährleisten. Das Gegenteil wäre alles abzulehnen, um die Sicherheit aufrecht zu halten.[63] Die Schnittstellen zur Firewall müssen gut geschützt

werden. Eine sichere Authentifizierung muss den Zugriff beschränken. Zusätzlich darf die Firewall nur aus vertraulichen Netzen oder von vertraulichen IP-Adressen aus konfiguriert werden, welche sich in dem lokalen Netz befinden. Zugriffe von außen sollten vermieden werden. Des Weiteren sollte es eine Zeitbeschränkung für den Zugriff auf eine Firewall geben.[5, S. 4] „Es MUSS immer möglich sein, direkt auf die Firewall zugreifen zu können, sodass sie im Notfall auch dann, lokal administriert werden kann, wenn das gesamte Netz ausfällt.“[5, S. 4] „In den Einstellungen der Firewall MUSS das dynamische Routing deaktiviert sein, es sei denn, der Paketfilter wird entsprechend dem Baustein NET.3.1 Router und Switches als Perimeter-Router eingesetzt.“[5, S. 4]

„Am Paketfilter müssen Schutzmechanismen aktiviert sein, um IPv4- sowie IPv6 Fragmentierungsangriffe abzuwehren.“[5, S. 5] Die Aufgaben der Firewall, sowie alle Konfigurationsänderungen müssen nachvollziehbar dokumentiert werden. Dazu zählen insbesondere Änderungen an Systemdiensten und dem Regelwerk für die Firewalls. Diese Dokumentation sollte nur durch Mitarbeiter mit der nötigen Berechtigung eingesehen werden können. Sollte IPv4 oder IPv6 nicht verwendet werden, sollte das gesamte Netz oder das entsprechende Netzsegment deaktiviert werden.[5, S. 5] Die Administration der Firewall sollte über ein gesondertes Managementnetz erfolgen und über die entsprechenden Sicherheitskontrollen verfügen. Es sollte nur autorisierten Geräten in diesem Netzwerk möglich sein, mit Managementprotokollen zu arbeiten die Authentisierung, Integritätssicherung und Verschlüsselung durchsetzen. Die grundlegenden Internetprotokolle wie HTTP, Simple Mail Transfer Protocol (SMTP) und DNS sollten über Protokoll spezifische Sicherheitsproxys laufen, wie Apache oder Nginx, welche die Protokolle spezifisch weiterleiten und die HTTP Verbindung auf HTTPS hochstufen können. Außerdem sollten verschlüsselte Verbindungen in nicht vertrauenswürdige Netze temporär entschlüsselt werden, um das Protokoll zu verifizieren und um festzustellen, ob es sich um einen Angriff handelt. Es muss auf die rechtlichen Rahmenbedingungen geachtet werden, sowie veraltete und schlechte Verschlüsselungsverfahren abgelehnt werden. Zertifikate, die nicht vertrauenswürdig sind, sollten ebenfalls abgelehnt werden. Firmenzertifikate sollten nachgerüstet und vorkonfigurierte nicht benötigte Zertifikate entfernt werden. Eine sichere Zeitsynchronisation für die Firewall sollte eingerichtet werden, um die Protokollierung mit dem richtigen Zeitstempel zu erstellen.[5, S. 6] Dafür kann etwa ein sicherer Zeitserver des National Institute of Standards and Technology (NIST) benutzt werden, welcher mithilfe eines Challenge Response Verfahrens die Uhrzeit sicher überträgt.[35] Durch falsche Zeiten kann es zu Problemen bei der Analyse von Log-Daten kommen. Zudem können beispielsweise abgelaufene Kerberos Tickets nicht erkannt werden. Es ist auch möglich, eine sichere Zeitquelle außerhalb des Netzwerkes zu benutzen. Sollte ein eigener NTP Server genutzt werden, ist es wichtig Amplification Attacks zu unterbinden.[24, S. 140]

Eine regelmäßige Prüfung der Firewall sollte durchgeführt werden, um Sicherheitsprobleme und Fehlkonfigurationen zu finden. Eine Diagnose sowie Fehlerbehebung sollte bereits im Vorhinein geplant sein. „Für typische Ausfallszenarien SOLLTEN entsprechende Handlungsanweisungen definiert und in regelmäßigen Abständen aktualisiert werden“[5, S. 7] Die Notfallplanung sollte dabei mit der Störungs- und Notfallvorsorge abgestimmt sein und in Papierform vorliegen. Diese Notfälle müssen regelmäßig erprobt werden.[5, S. 7]

3.1.2.1 Erhöhter Schutzbedarf

Ein erhöhter Schutz besteht dann, wenn es sich um hochsensible Systeme, wie KRITIS handelt. Bei einem erhöhten Schutzbedarf sollte sichergestellt werden, dass die Firewall bei einem Absturz oder Neustart stets die neuesten Konfigurationen lädt. Funktionale Erweiterungen sollten auf dedizierte Hardware oder Software ausgelagert werden. Eine mehrstufige Firewall Architektur sollte aufgebaut werden, bei welcher verschiedene Firewall Produkte, sowie verschiedene Betriebssysteme zum Einsatz kommen. Diese mehrstufige Architektur sollte sowohl für die inneren als auch für die äußeren Firewalls erstellt werden.

Die Inhalte sollten zentral gefiltert werden und auch die Verschlüsselung aufbrechen um die Inhalte zu analysieren. Soziale Netzwerke, Exploit Datenbanken und andere Schädliche Seiten sollten blockiert werden. Ein Bandbreitenmanagement sollte etabliert sein, um zu gewährleisten, dass kritische Anwendungen immer genug Bandbreite zur Verfügung gestellt wird.[5, S. 7, 8] „Firewalls mit einer Sicherheitsevaluierung nach Common Criteria SOLLTEN eingesetzt werden, mindestens mit der Stufe EAL4.“[5, S. 7]

3.1.2.2 Protokollierung

Neben den Schutzmaßnahmen muss eine Protokollierung implementiert sein. „abgewiesene Netzverbindungen (Quell- und Ziel-IP-Adressen, Quell- und Zielport oder ICMP/ICMPv6-Typ, Datum, Uhrzeit)“[5, S. 4] müssen protokolliert werden. Anhand dieser lassen sich Angriffe und Portscans erkennen. Wird von einer IP-Adresse eine Verbindung zu mehreren Ports aufgebaut, kann dies ein Zeichen für einen Portscan sein. Zudem müssen „fehlgeschlagene Zugriffe auf System-Ressourcen aufgrund fehlerhafter Authentisierungen, mangelnder Berechtigung oder nicht vorhandener Ressourcen“[5, S. 4] aufgezeichnet werden. Mit diesen Logs ist es dann beispielsweise möglich IP-Adressen dynamisch zu sperren, wenn sie innerhalb einer vordefinierten Zeitspanne zu oft negativ auffallen. „Fehlermeldungen der Firewall-Dienste“[5, S. 4] sollten ebenfalls aufgezeichnet werden, anhand dieser ist es möglich Angriffe auf die Firewall selbst oder andere verdächtige Aktivitäten nachzuverfolgen. Auch sollten „allgemeine Systemfehlermeldungen“[5, S. 4] aufgezeichnet werden. Diese helfen zwar nicht unbedingt dabei Angriffe zu erkennen, können aber bei einer späteren Auswertung hilfreich sein die Schritte von Angreifern nachzuverfolgen. Besonders wichtig ist es „Konfigurationsänderungen (möglichst automatisch)“[5, S. 4] zu protokollieren. Denn die Änderung von Konfigurationen kann auf einen Angriff auf eine Firewall selbst oder auf Fehler hinweisen, die durch eine Änderung der Firewallregeln entstanden sind. Außerdem müssen Verstöße gegen Access Control List (ACLs) protokolliert werden, sofern diese umgesetzt werden.[5, S. 4] Die aufgeführten Maßnahmen beschreiben das Minimum der aufzuzeichnenden Logs. In jedem Netzwerk sollte abgewägt werden, was zusätzlich aufgezeichnet werden kann und muss. Neben dieser Protokollierung sollte die Firewall aber auch in ein geeignetes Systemüberwachungs- bzw. Monitorringkonzept eingebunden werden. Bei Fehlern oder der Grenzwertüberschreitung sollten die zuständigen Mitarbeiter alarmiert werden. Eine Möglichkeit ist es, die Logs an eine zentrale Log-Verwaltung oder ein SIEM zu schicken. Allerdings muss darauf geachtet werden, dass die Protokolldaten und Statusmeldungen nur über sichere Kommunikationswege erfolgen, da die Logs sensible Informationen über das Netzwerk oder die Nutzer offenbaren können.[5, S. 6]

4 Implementierung

Ein Programm, welches als **PoC** dient, wurde implementiert. Dieses dient dazu, die in [Kapitel 3](#) auf Seite 19 aufgezeigte Lösungskonzeption umzusetzen. Dazu sollen die in diesem Kapitel umgesetzten Schritte aufgezeigt und erklärt werden. Zudem sollen die Gründe für die Umsetzung bestimmter Implementierungsentscheidungen erläutert werden. Das entwickelte Programm ist dabei aufgeteilt in eine **API** Komponente, eine Firewall-Komponente und eine Komponente, welche ein Formular zur Verfügung stellt. Mithilfe dieser Komponenten werden mehrere Programme angebunden, siehe [Abbildung 4.1](#).

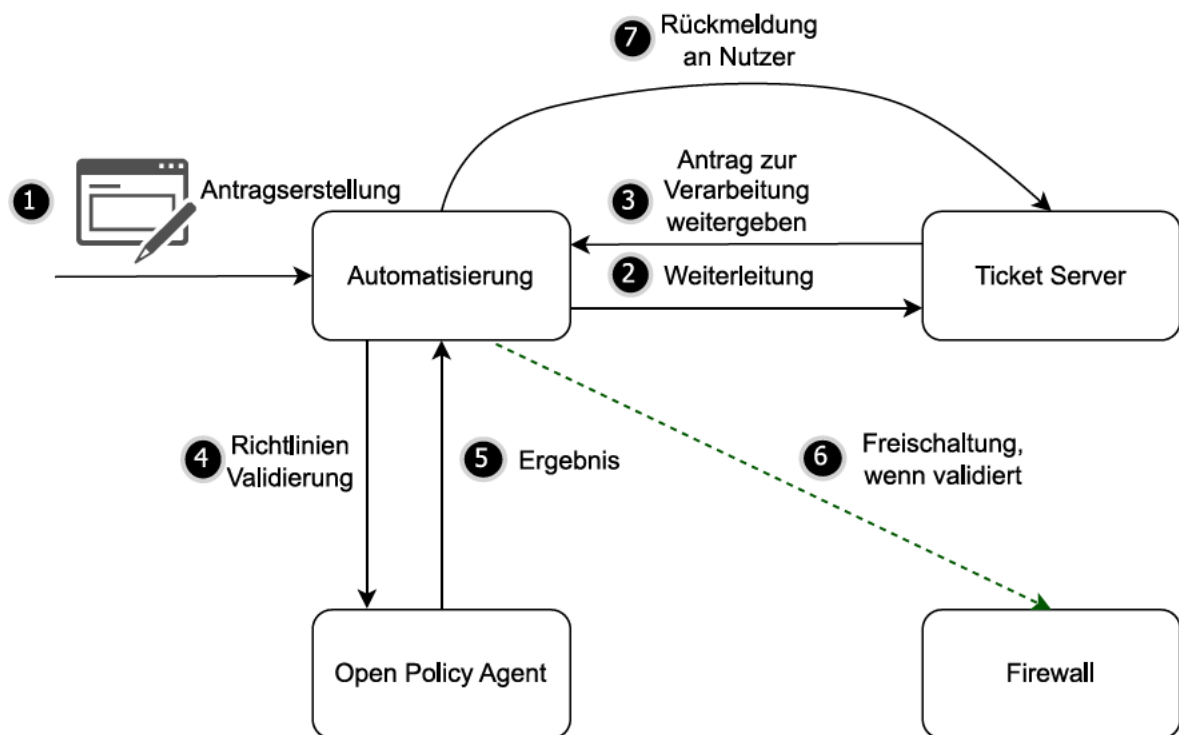


Abbildung 4.1: Architekturaufbau des PoC (Mit drawio erstelltes Diagramm)

Wie im Architekturdiagramm [Abbildung 4.1](#) dargestellt, steht die Automatisierungslösung im Mittelpunkt. Dabei stellt die Automatisierungslösung einen Flask-Server zur Verfügung. Dieser hat die Aufgabe, **Webhooks** entgegenzunehmen, welche durch Ereignisse auf dem **Ticket Server** ausgelöst werden. Das Entgegennehmen von **Webhooks** war die ursprüngliche Aufgabe des Flask-Servers. Im Laufe der Arbeit wurde ersichtlich, dass es noch weitere Möglichkeiten für den Einsatz eines Flask-Servers gibt, welche in den folgenden Absätzen weiter erläutert wird. Durch die **Webhooks** ist es ressourcenschonend möglich, neue **Tickets** zu erhalten, denn eine Netzwerkoperation wird nur ausgelöst, wenn es ein neues **Ticket** gibt. So entfällt ein Abfragen in Intervallen nach neuen **Tickets**. Allerdings muss der Trigger so eingestellt werden, dass er auf ein vordefiniertes Ereignis reagiert. Dafür eignet sich die Erstellung eines neuen **Tickets**, welches eine bestimmte Markierung trägt. Diese Markierung sorgt dafür, dass Anträge zur Freischaltung der Firewall erkannt werden. Der Vorteil ist, dass es möglich ist, dieses Ticketsystem nicht nur für das Beantragen von Firewallregeln zu verwenden, sondern auch für andere Aufgaben. Der **Webhook** hingegen muss so konfiguriert werden, dass

die Mitteilungen an die erstellte Automatisierungslösung geschickt werden. Der Flask-Server nimmt dann über diesen Webhook die Daten entgegen, um einen Freischaltungsantrag zu verarbeiten.

Der Flask-Server stellt zudem das [Hyper Text Markup Language \(HTML\)](#) Formular bereit. Dieses ist in [Abbildung 4.1](#) auf der vorherigen Seite der erste Schritt. Das Formular fragt vom Nutzer alle benötigten Informationen ab, die zur Freischaltung der Firewall benötigt werden. Dazu gibt es mehrere Eingabefelder, die unter anderem die E-Mail-Adresse abfragen, sowie eine Auswahl der definierten Firewallregeln. Dieses Formular ist in der [Abbildung 4.2](#) zu sehen. Basierend auf den ausgewählten Regeln werden mithilfe von [Pyscript](#) die notwendigen

E-Mail Adresse

Please add your email for status updates

Choose a firewall rule:

Please choose a rule to request a Firewall activation

Service

Please choose a service

Zone

The Zone to operate in

Allow

Allow or deny a service

Abbildung 4.2: Formular zur Freischaltung der Firewall(Bildschirmfoto)

Abfragen zum Formular hinzugefügt.

[Pyscript](#) ist eine Alternative zu Javascript. Es handelt sich hierbei um eine Bibliothek, die dafür sorgt, dass alles möglichst einheitlich in Python erstellt werden kann. Diese Lösung wurde auch genutzt, um ihre Möglichkeiten zu testen. Es wurde geschaut, wie gut es möglich ist, diese Lösung zum Erstellen des Formulars zu benutzen. Mit [Pyscript](#) ist es möglich, innerhalb des Browsers dynamischen Code zu erstellen. Die Idee von [Pyscript](#) ist hierbei, dass Python-Code im Browser ausgeführt wird. Dafür wird die sogenannte [WebAssembly \(Wasm\)](#) benutzt. Hierbei handelt es sich um ein Werkzeug, welches eine Art virtuelle Maschine im Browser erzeugt. Diese virtuelle Maschine ist stark minimiert und dient lediglich dazu, Programmiersprachen auszuführen. Die [Wasm](#) stellt einen C-Compiler zur Verfügung. Da die bekannteste und meistgenutzte Implementierung [CPython](#) C benutzt, kann hier Python dank der [Pyscript](#) Library, sowie der darunterliegenden Komponente [Pyodide](#), benutzt werden.[66][21]

Hierbei werden die Nutzereingaben dynamisch abgefragt. Basierend auf den Firewall-Optionen [Port öffnen](#), [Service erlauben](#), [ICMP-Typ blocken](#), [Port weiterleiten](#) und [IP blocken](#) werden dynamisch Elemente erstellt und zum Formular hinzugefügt. Einige der Felder ermöglichen dem Nutzer auch aus einer Auswahl von vorgefertigten Eingabemöglichkeiten zu wählen. Diese werden dazu bei der [API](#) Komponente abgefragt und dann dynamisch zur Verfügung gestellt. Es ist möglich, von Flask die vorkonfigurierten Dienste, die [ICMP](#)-Typen und Zonen abzufragen. Alle Elemente bekommen ein Label und einen Hilfetext zugeteilt und können dann ebenfalls befüllt werden. Anschließend werden die vom Nutzer eingegebenen Informationen wieder der [API](#) Komponente innerhalb des Automatisierungstools zur Verfügung gestellt. Durch diese Mechanismen soll die Nutzerfreundlichkeit erhöht und die Fehlerrate reduziert werden. Dadurch, dass das Eingabefeld nur vordefinierte und geprüfte Elemente enthält, können hier keine Fehler auftreten

Das Formular kann in andere Webseiten eingebunden werden, und so Kunden oder Mitarbeitern zur Verfügung gestellt zu werden. Der Vorteil an einer externen Komponente ist, dass sie veränderbar und anpassbar ist. Außerdem lässt sich diese Komponente bei Bedarf leicht austauschen.

Die Anfrage wird in Schritt zwei nach der Konvertierung in das [YAML](#)-Format an das Ticketsystem weitergeleitet. Dieses erstellt mit der Anfrage ein [Ticket](#). Ein Antrag zur Freischaltung einer Firewallregel ist durch die einfache Syntax, siehe [Listing 1](#) direkt im Ticketsystem möglich.

```
allow_service:  
  service: <service name>  
  zone:  
  allow: <true|false>
```

Listing 1: Beispiel der Freischaltung eines Dienstes in [YAML](#)

Dazu muss der Nutzer den Aufbau des Antrags im erstellten [YAML](#)-Format korrekt implementieren. Dadurch wird vom Ticketserver ein Trigger ausgelöst und durch einen Webhook wird das [Ticket](#) in Schritt drei wieder an die Automatisierungslösung zurückgegeben. Der [Open](#)

Policy Agent (OPA)-Server wertet anschließend die Anfrage der Richtlinie in Schritt vier aus. Die Richtlinien-Überprüfungskomponente im Programm leitet lediglich die Anfrage in Schritt vier an den Richtlinien-Server(**OPA**) weiter und gibt in Schritt fünf einen Wahrheitswert zurück, je nachdem, ob die Anfrage angenommen oder abgelehnt wurde.

Schritt sechs ist die Firewall-Komponente, welche auf einer abstrakten Firewall-Komponente basiert. Um dem Unternehmen in Zukunft Auswahlmöglichkeiten über das verwendete Ticketsystem zu geben, werden hier abstrakte Methoden definiert. Das bedeutet, dass der Methoden-Kopf erstellt wird, aber die Methoden erhalten keinen Code zum Ausführen. Eine spezifische Firewall kann nun von dieser Klasse eine spezifische Kind-Klasse erstellen. Die abstrakten Methoden müssen in dieser Kind-Klasse dann implementiert werden. Die Firewall-Komponente leitet nun diese Klasse ab und muss alle Methoden, welche als abstrakt definiert sind, sowie den Methodenkörper implementieren. Die Firewall benutzt hierbei die Bibliothek **Paramiko**, bei der es sich um eine **SSH** Komponente handelt, welche sich auf dem Server anmelden soll, um so die Firewall anzusteuern, da **Firewalld** keine **API** hat. Die anderen Methoden sind Wrapper, um **Firewalld**-Befehle. Auf der Firewall wird sich mit einem privaten **SSH** Schlüssel verifiziert. Eine Verifizierung mit Passwort funktionierte nicht ordnungsgemäß, wobei dies nicht nachteilig ist, da die Identifizierung mit **SSH**-Schlüsseln sicherer ist. Neben den Regeln, die angewendet werden, gibt es hier noch Methoden, welche weitere Informationen bereitstellen. Diese werden, wie bereits erwähnt, vom Flask-Server genutzt, um dem Nutzer Informationen bereitzustellen. Dazu zählt die Abfrage der Zonen, Dienste und **ICMP** Typen. Dadurch ist es möglich, Änderungen an diesen Informationen vorzunehmen und die Nutzer erhalten immer den aktuellen Stand. Die Dienste haben den Vorteil, dass sie übersichtlicher sind, vor allem wenn ein Dienst wie **FTP** mehrere **Ports** benutzt. Dann ist einwandfrei zu sehen, welche **Ports** zusammengehören und auch warum ein **Port** geöffnet wurde. Daneben können die nutzbaren Zonen sowie die zur Verfügung stehenden **ICMP**-Typen abgefragt werden, die blockiert werden sollen. Die Funktionen sollen die gewählte Firewallregel implementieren oder löschen.

Überdies gibt es noch eine Komponente, die ausschließlich die Anfragen an den Richtlinien-Validierungsserver schickt. Die letzte Komponente besteht aus Unittests, die sicherstellen sollen, dass die Automatisierungslösung ordnungsgemäß funktioniert.

4.1 Entwicklungsumgebung

Für die Entwicklung der Automatisierung wurde zur Versionsverwaltung **Gitlab** gewählt. Der Vorteil von **Gitlab** ist, dass viele Werkzeuge wie eine **CI/CD** Pipeline bereits enthalten sind. Die **CI/CD** kann genutzt werden, um automatisch Tests auszuführen. Außerdem wurde die **CI/CD** genutzt um das Python-Programm zu einem Python-Paket zu packen und in der **Gitlab**-Paket-Registry zur Verfügung zu stellen. Des Weiteren gibt es die sogenannte Funktion „**Gitlab** Pages“. Mit dieser ist es möglich, Webseiten mithilfe der **CI/CD** automatisch auf **Gitlab** bereitzustellen. Bei dieser Arbeit wurde diese Funktion dafür genutzt um die mit „**Sphinx**“ erstellte Dokumentation des Programmcodes zu hosten. Durch die Verwendung von „**Docstrings**“ werden Klassen und Funktionen beschrieben. Die „**Docstrings**“ werden in den Quellcode geschrieben und beschreiben dort die Funktionalität der einzelnen Codeteile, siehe **Abbildung 4.3** auf der nächsten Seite.

```
def allow_service(self, service: str, zone: str, allow: bool = True):
    """Allow or deny a service

    Args:
        service (str): The service to allow
        zone (str): In which zone the service should be allowed
        allow (bool, optional): Allows or denies a service. Defaults to True.
    """
    logger.debug(f"Creating Firewallrule for service {service}")
    if service in self.exec("--get-services").split():
        logger.debug("Now executing the firewall rule in real")
        if allow:
            self.exec(f"--add-service={service} --zone={zone} ")
        else:
            self.exec(f"--remove-service={service} --zone={zone} ")
```

Abbildung 4.3: Beispiel einer dokumentierten Methode

Ein von adesso gestellter Code wurde genutzt, der eine Infrastruktur aufbaut, welche als Test- und Entwicklungsumgebung dient. Dafür wurde die IaC Lösung [Terraform](#) benutzt. Diese beauftragt einen Server-Hoster, in diesem Fall Hetzner, damit Server zur Verfügung zu stellen. Mithilfe von [Terraform](#) wird neben den Servern auch ein lokales Netzwerk erstellt. Außerdem wird hierbei auch die sogenannte „cloud-init.yaml“ ausgeliefert, welche einen Nutzer und [SSH](#) konfiguriert. Diese Datei dient der initialen Konfiguration eines Servers und wird auf alle von [Terraform](#) erstellten Server angewandt. Damit auch die Server selbst konfiguriert werden, wurde die Automatisierungslösung [Ansible](#) genutzt. Diese nutzt die konfigurierte [SSH](#)-Schnittstelle, um auf die Server zuzugreifen. Dadurch ist es möglich, die Server ohne einen Agenten auf dem Computer zu konfigurieren. [Ansible](#) arbeitet dabei die in einer Liste konfigurierten Aufgaben ab. Eine der durch [Ansible](#) zu installierenden Abhängigkeiten ist Docker. Dafür wurde ein eigenes [Ansible](#)-Skript erstellt, welches von den Skripten für den Ticketserver und dem Server für die Automatisierungslösung genutzt wird. Das Skript für die Firewall installiert lediglich [Firewalld](#), bringt die Pakete auf den neuesten Stand und installiert [Cockpit](#). [Cockpit](#) ist eine grafische Administrationsoberfläche, die eine Webseite zur Verfügung stellt. Diese wurde getestet, um dem Kunden grafisch das Ändern der Firewallregeln aufzuzeigen, da geplant ist diese Arbeit in Zukunft weiter einzusetzen. Das Skript für den Ticketserver downloadet die Version von [Zammad](#), die für den Einsatz mit Docker Compose erstellt wurde und trifft hierfür einige Einstellungen. Als Nächstes werden die für [Zammad](#) benötigten Container gestartet, sowie im Anschluss ein Konfigurationsskript geladen. Das Skript für das entwickelte Werkzeug hat ebenfalls nur wenige Aufgaben. Als Erstes werden hierbei einige Pakete installiert, die zur Entwicklung benötigt werden. Der nächste Schritt ist die Konfiguration von Git und das Herunterladen des Automatisierungswerkzeugs. Außerdem wird noch „[Pyenv](#)“ genutzt, um unterschiedliche Python-Versionen zu installieren. Dies soll später dabei helfen, zu testen, welche Python-Versionen mit dem entwickelten Programm kompatibel sind. Dazu wurde „[Nox](#)“ benutzt, mit diesem ist es möglich die Tests auf den unterschiedlichen Python-Versionen auszuführen. Als Letztes wurde noch mit „[Playwright](#)“ versucht ein Skript zu erstellen, welches das Erstellen des Webhooks und des Triggers automatisiert und das Token in die „config.ini“ schreibt. Dies wird optimiert, indem die [Gitlab CI/CD](#) benutzt wird. Diese erstellt automatisch die Infrastruktur, wenn der Code für die Infrastruktur geändert wurde. Auch ist es möglich, diese Infrastruktur wieder abzureißen, wenn diese nicht mehr benötigt wird.

4.2 Ticket System

Als Ticketsystem wurde die in Deutschland entwickelte Open-Source-Lösung [Zammad](#) gewählt. Durch die Entwicklung dieser Lösung in Deutschland ist es leichter, mit dem Support zu kommunizieren. Das Ticketsystem ist gut strukturiert aufgebaut und hat eine benutzerfreundliche und übersichtliche Oberfläche. Außerdem werden viele Kommunikationskanäle zur Verfügung gestellt, um ein [Ticket](#) zu erstellen. Unter anderem ist dies eine Anbindung an Telegram. Das Wichtigste für diese Lösung ist die dokumentierte [API](#), die alle benötigten Funktionen besitzt und zufriedenstellend funktioniert. Zusätzlich besitzt [Zammad](#) die Funktion mithilfe eines Triggers einen Webhook bei bestimmten Ereignissen zu benachrichtigen, wie wenn ein [Ticket](#) erstellt wurde, das eine Firewall Anfrage erhält. Dieser Trigger sorgt dafür, dass das Programm nicht durchgehend die [Tickets](#) des Servers nach neuen, zu bearbeitenden, [Tickets](#) durchsuchen muss, sondern direkt benachrichtigt wird. Dadurch wird der Netzwerkverkehr verringert, was zu einer besseren Performanz führt. [Zammad](#) enthält auch die Möglichkeit, ein Formular auf der eigenen Website einzubinden, über das [Tickets](#) erstellt werden können. Eine Erstellung von selbst erstellten Feldern ist mit diesem Formular nicht möglich. Deshalb wurde ein eigenes Formular erstellt, mit dessen Hilfe es möglich ist, eine Firewall Freischaltung zu beantragen.

4.3 Firewallld

„[Firewallld](#) ist eine für viele Linux-Distributionen verfügbare Firewall-Verwaltungssoftware, die als Frontend für die kernelinternen nftables- oder iptables-Paketfiltersysteme von Linux dient.“[\[36\]](#)

Das Design-Ziel war es dabei, die Firewallregeln übersichtlicher darzustellen. Die Entwickler erkannten, dass es oft nur eine geringe Anzahl an [IP-Adressen](#) gibt, die für bestimmte Dienste verwendet werden. Dafür gibt es verschiedene Zonen. Der Netzwerkverkehr wird dafür anhand der Quelle und des Interfaces in die entsprechende Zone eingeteilt. Jede Zone hat ihre eigene Konfiguration mit Regeln. Zusätzlich wurde die Syntax vereinfacht, um die Bedienung zu erleichtern. Eine weitere Besonderheit ist die Benennung von Diensten, wodurch exakt geprüft werden kann, warum eine bestimmte Firewallregel angewandt wurde. Hier gibt es die Möglichkeit, Firewallregeln direkt zu speichern.[\[73\]](#)

Für [Firewallld](#) wurde sich entschieden, da dies eine Personal-Firewall, siehe [Abschnitt 2.1.6.1](#) auf Seite 9 ist. Es eignet sich nicht für große Netzwerke, da es nicht die nötigen Funktionen enthält, ist für diesen [PoC](#) jedoch völlig ausreichend. Da es Open-Source ist und einen überschaubaren Funktionsumfang bietet, welcher allerdings alle benötigten Funktionen enthält. So muss beispielsweise für das Freigeben von [SSH](#) nicht der [Port 22](#) geöffnet werden, sondern [SSH](#) freigegeben werden. Eine Kontrolle, weswegen der [Port](#) geöffnet ist, entfällt und es ist besser erkennbar, wenn ein Dienst mehrere [Ports](#) verwendet, da diese dann gruppiert werden können. Es ist möglich [IP-Adressen](#) freizugeben und zu blockieren, [Ports](#) zu öffnen und zu schließen, sowie [ICMP](#) Netzwerkpakete zu erlauben und zu verweigern. Eine besondere Funktionalität ist zudem die Einteilung in verschiedene Zonen. Dies erlaubt zum einen eine einfachere Dokumentation, zum anderen können mehrere Interfaces unterschiedlich

genutzt werden, beispielsweise kann die Firewall zwischen Internet und das lokale Netz gesetzt werden. Hierfür ist ein Interface mit dem Internet und ein weiteres mit dem lokalen Netzwerk verbunden. Dafür können die vordefinierten Zonen „internal“ und „external“ benutzt werden. Es kann außerdem die NAT Funktionalität für das vorherige Beispiel genutzt werden. Für komplexere Anforderungen unterstützt **Firewalld** eine einfache Sprache, die mit den sogenannten „rich rules“ erstellt werden können. Außerdem gibt es eine einfache Logging Möglichkeit.

4.4 Policy as Code

Bei dem **PaC** Verfahren werden Richtlinien als Code dargestellt, damit es möglich ist, diese automatisch zu prüfen. Dank dieses Verfahrens werden menschliche Fehler reduziert, da Regeln nicht mehr manuell überprüft werden müssen. Eine Richtlinie ist jede Regel, Bedingung oder Anweisung, die IT-Prozesse oder IT-Vorgänge regelt. Der erste Vorteil der **PaC** ist die Effizienz, denn die Richtlinie kann geteilt und automatisch auf eine beliebige Größe angewandt werden. Zudem können die Richtlinien in Echtzeit angewandt werden. Ebenso ist es leicht die Richtlinien einzusehen, was es erlaubt die praktische Anwendung zu sehen. Die Richtlinien können auf diese Weise auch kollaborativ angewandt werden, da es leicht möglich ist diese mit mehreren Leuten zu bearbeiten. Sind die Richtlinien korrekt implementiert, sind sie sehr genau und setzen die Unternehmensrichtlinien wie gefordert um. Durch die Implementierung als Code ist zudem eine Versionskontrolle mit git und das Nachvollziehen der Änderungshistorie möglich. Es ist möglich automatische Tests zu schreiben, die gewährleisten, dass die Unternehmensrichtlinien funktionieren. Des Weiteren ist es möglich, verschiedenste Standards anzuwenden und durchzusetzen. Als Letztes ist es auch möglich, andere Automatisierungslösungen zu überwachen. [28] [33]

4.5 Open Policy Agent

Der **OPA**-Server ist eine Open-Source-Lösung mit eigener Sprache. Desweiteren wird **OPA** verwendet, da dieser bereits in der Firma verwendet wird und eine Verwendung innerhalb des Teams geplant ist. Die von **OPA** genutzte Sprache zur Regelerstellung heißt Rego und ist an die Programmiersprache GO angelehnt, da **OPA** in GO geschrieben ist. Für die Regelerstellung stellt die Sprache bereits 150 vorkonfigurierte Funktionen bereit, welche unter anderem bei der String-Manipulation oder der Decodierung von **JWT** helfen. Es ist einfach, kontextabhängige Regeln zu erstellen, welche leicht verständlich sind und auch angepasst werden können. Überdies gibt es viele Integrationen zur Anbindung unterschiedlicher Dienste, wodurch der Einsatz vereinfacht wird. Python gehört leider nicht zu diesen Erweiterungen, allerdings gibt es hier die Möglichkeit, die **API** direkt anzusteuern. Die **API** gibt dann ein **JSON** zurück, dass eine Rückgabe erhält. Die Rückgabe von **JSON** gibt hier auch die Möglichkeit komplexere Antworten als *akzeptiert* und *verweigert* zurückzugeben. Eine Richtlinie kann direkt beim Start des Servers übergeben werden, sodass sie nicht manipuliert werden kann.

Bei Problemen mit der Regelerstellung kann der neue KI Chatbot **ChatGPT** genutzt werden. Dieser ist in der Lage auch Regeln, in Rego zu erstellen und diese zu verbessern. Diese Regeln sollten allerdings noch manuell überprüft werden. Eine Evaluierung von **ChatGPT** ist in [Kapitel 5](#) auf Seite 39 zu finden.

4.5.1 Erstellung der Richtlinie

Um zu überprüfen, ob ein Antrag valide ist, muss zunächst eine Richtlinie in Rego erstellt werden, die dann dem OPA-Server übergeben werden kann. Zum Testen des Programms wurde eine beispielhafte Richtlinie erstellt, siehe Listing 2. Bei dieser wird in Variablen definiert, was erlaubt ist. Beispielsweise werden die erlaubten Ports in Variablen gespeichert. Dann wird eine Überprüfung durchgeführt. Dazu dient das Codestück siehe Listing 2. Dieses zeigt bei der Überprüfung, ob es erlaubt ist, die IP-Adresse zu blockieren.

```
allow if {  
    input.type == "block ip"  
    net.cidr contains(nets[ ], input.ip address) # ip  
    input.zone == zones[ ] # zone  
}
```

Listing 2: Ausschnitt aus einer mit Rego Beispielhaft erstellten Richtlinie

Eine Prüfung zeigt, ob die richtige Firewall-Option ausgewählt wurde. Im Beispiel ist dies die Option „block ip“. Daneben wird in Zeile 3 des Ausschnitts mithilfe einer internen Funktion geprüft, ob die vom Nutzer angefragte Adresse in einem erlaubten Netzwerk liegt. Außerdem wird noch geprüft, ob die Regel in der angefragten Zone erlaubt ist. Sind alle diese Bedingungen gültig, wird die Variable „allow“ auf „True“ gesetzt. Je mehr Richtlinien und je granularer die Regeln erstellt werden, desto unübersichtlicher kann die Richtlinie werden. Außerdem gibt es die Autorisierung der Benutzer, welche je nach Umfang sehr komplex werden kann. Die Wartbarkeit der Regeln wird durch die hohe Komplexität verschlechtert, wodurch Fehler auftreten. Doch es ist möglich, die Richtlinien auf verschiedene Daten aufzuteilen. Dazu können zwei verschiedene Arten von Dateien erstellt werden. Zum einen eine Datei für verschiedenste Variablen, hier können dann die zugelassenen Dienste für verschiedene Server erstellt werden. Daneben ist es auch möglich, andere Regeldateien zu importieren. Dadurch ist es leicht möglich, die Logik effizient aufzuteilen, um die Regeln zu warten und um die Übersichtlichkeit zu erhöhen.

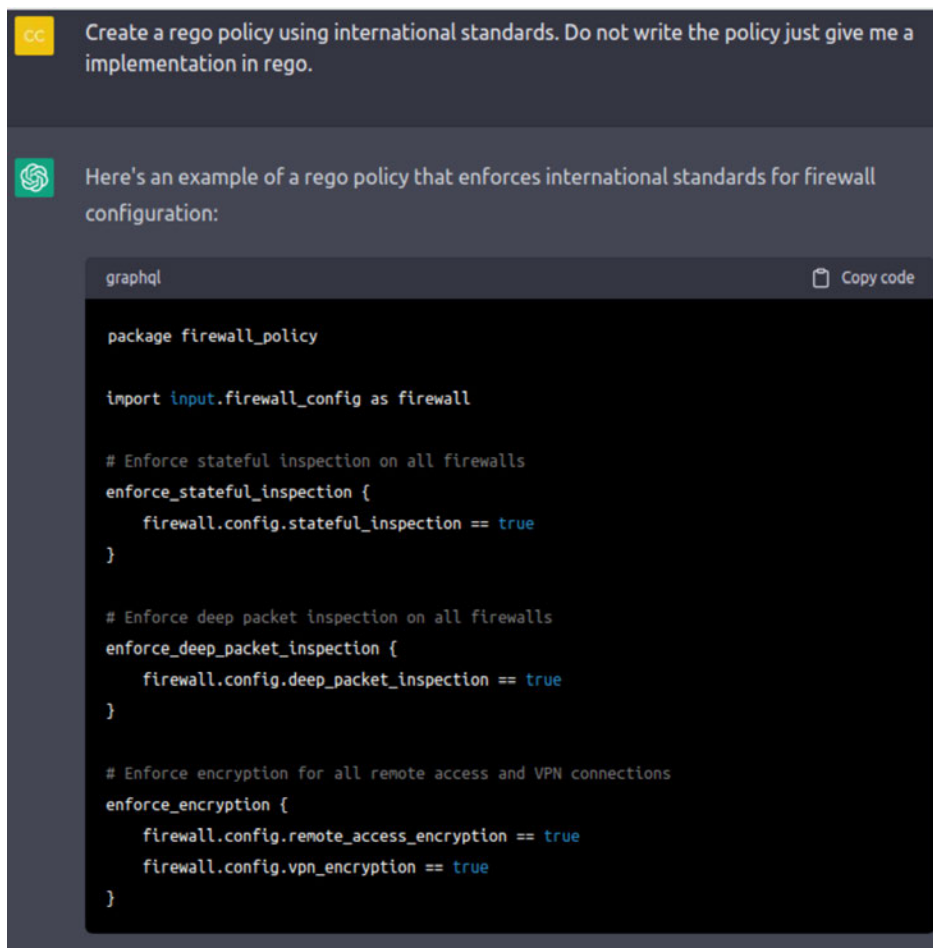
5 Ergebnisse und Diskussion

Es hat sich gezeigt, dass es leicht möglich ist, die Regeln der Firewall automatisch freizugeben, nachdem eine Richtlinie definiert wurde. Damit zeigt das entwickelte Werkzeug auf, dass die Freischaltung einer Firewallregel ohne kommerzielle Produkte mit Open-Source-Werkzeugen einfach umzusetzen ist. Die Integration weiterer Programme sollte in Zukunft mit einigen Optimierungen möglich sein, da hier lediglich die bereits vorhandenen Schnittstellen angepasst werden müssen.

Das entwickelte Werkzeug zur Prüfung und Anwendung von neuen Firewallregeln erleichtert die Beantragung überaus, da hier die Informationen abgefragt werden können, welche benötigt werden. Das automatische Validieren der Nutzerautorisierung und der Unternehmensrichtlinie funktionieren effizient. Das entwickelte [HTML](#) Formular sollte ausgetauscht oder verbessert werden, da es funktionell ist, aber im Moment noch nicht für den praktischen Betrieb fertig ist. Dieses Formular ist umständlich und Änderungen sind nicht gut umzusetzen. Daher sollte geprüft werden, wie es optimiert werden kann, sodass es leichter anpassbarer ist. Eine Alternative wäre hier das low-code Framework „[Pywebio](#)“. Dieses ermöglicht es, mit wenig Code Nutzerabfragen zu erzeugen. Durch diesen low code-Ansatz ist eine anpassbare Form einfach realisierbar. Dieser Ansatz könnte auch die Arbeit der Add-on Entwickler erleichtern, falls die Nutzerabfrage angepasst werden muss. Des Weiteren ist eine Erweiterungsfunktionalität wichtig, da diese Arbeit aufgezeigt hat, welche verschiedenen Systeme es gibt, wie [SIEM](#), [IDS](#) und [IPS](#). Außerdem müssen Firewalls sich in kurzen Zeiträumen neuen Anforderungen und Bedrohungen anpassen, wodurch eine gute Anbindung sinnvoll ist. Dieses Erweiterungssystem wäre zudem von Vorteil, wenn weitere Systeme angeboten werden sollen. Ein Beispiel wäre hier das bereits vorgestellte Werkzeug [CrowdSec](#).

Ein Problem bei der Automatisierung des Prozesses zur Freischaltung von Firewallregeln bleibt die Richtlinie. Es ist notwendig, diese regelmäßig zu aktualisieren, wenn es Anpassungen durch neue Vorgaben wie vom [BSI](#) gibt. Eine Aktualisierung kann zwar von einem Experten angefertigt werden, allerdings beschäftigt nicht jede Firma einen qualifizierten Experten. Deshalb wäre ein Werkzeug, welches bei der Einrichtung der Richtlinien hilft, von Vorteil. Dieses Werkzeug kann dann übersichtlich die verschiedenen Richtlinien aufzeigen und es vereinfachen, diese zu erstellen oder anzupassen. Eine andere Möglichkeit besteht darin, dies als Supportdienstleistung zu verkaufen. Durch die Notwendigkeit, diese Richtlinien regelmäßig zu warten, ist es sinnvoll einen entsprechenden Support für mehrere Jahre anzubieten. Ein weiterer möglicher Ansatz wäre die Umsetzung mit einer [KI](#), welche die Regeln erstellt oder hilft diese zu erstellen. Für diese Arbeit wurde die [KI ChatGPT](#) untersucht, da sie sogar außerhalb der IT-Szene Schlagzeilen macht. Überdies wurde in dieser Arbeit getestet, ob es möglich ist, Rego-Regeln zu erstellen. So wäre es insbesondere möglich, schriftlich formulierte Regeln mit einer [KI](#) automatisch umsetzen zu lassen und die Richtlinien in Echtzeit zu aktualisieren, sollte dies nötig sein. Es sollte allerdings beachtet werden, dass [ChatGPT](#) nicht spezifisch dazu entwickelt wurde, um Rego Regeln zu erstellen. [ChatGPT](#) wurde entwickelt, um Fragen zu allen Themengebieten beantworten zu können. Also ist das Wissen breit gestreut und ist nicht tiefgreifend. Im Test zeigte es sich zwar, dass [ChatGPT](#) in der Lage ist, Richtlinien zu erstellen, doch es müssen die richtigen Anfragen an [ChatGPT](#) gestellt

werden, um eine Richtlinie zu erhalten. Bei der Anfrage: „Erstelle eine Rego Regel, welche eine Richtlinie für einen Firewall DMZ Aufbau realisiert. In diese Richtlinie sollen Vorgaben des BSI angewendet werden.“ erstellte ChatGPT eine Antwort in Textform. Diese Richtlinien waren zwar gut, doch war auch eine Regel, dass die Firewall entsprechend dem Grundschutz erstellt werden soll. Dies ist zwar richtig, doch es war nach einer Regel gefragt und nicht nach einem Verweis auf den BSI Grundschutz. Doch das größte Problem war hier, dass trotz der expliziten Anfrage einer Rego Regel lediglich Richtlinien in Textform geliefert wurden. Bei der Bitte dies in Rego zu verfassen, übersetzte ChatGPT dies dann in Rego, doch wurde für die Richtlinie die Firewallregelmäßig zu warten eine Funktion aufgerufen mit dem Namen „perform regular maintenance()“. Ein weiterer Versuch war eine Anfrage auf Englisch, in der die Erstellung einer Richtlinie nach internationalen Standards angefragt wurde. Hier wurde in einem zweiten Satz extra nach der Erstellung einer Richtlinie in Rego gefragt, doch auch hier war das Ergebnis ein Fließtext. Ein dritter Versuch, bei dem explizit gefragt wurde, dass kein Text zurückgegeben werden soll, sondern lediglich die Richtlinie in Rego implementiert sein soll, brachte dann schlussendlich eine Implementierung in Rego. Allerdings war auch hier das Problem, dass die Richtlinien als Funktionsnamen genommen wurden, ohne diese sinnvoll umzusetzen. Siehe [Abbildung 5.1](#). Die Frage nach einer standardmäßigen Richtlinie



The screenshot shows a chat interface with a dark background. At the top, a yellow icon with 'cc' is followed by the text: "Create a rego policy using international standards. Do not write the policy just give me a implementation in rego." Below this, a green icon with a brain is followed by the text: "Here's an example of a rego policy that enforces international standards for firewall configuration:". Underneath, there is a code editor window with a "graphql" label and a "Copy code" button. The code is as follows:

```
package firewall_policy

import input.firewall_config as firewall

# Enforce stateful inspection on all firewalls
enforce_stateful_inspection {
    firewall.config.stateful_inspection == true
}

# Enforce deep packet inspection on all firewalls
enforce_deep_packet_inspection {
    firewall.config.deep_packet_inspection == true
}

# Enforce encryption for all remote access and VPN connections
enforce_encryption {
    firewall.config.remote_access_encryption == true
    firewall.config.vpn_encryption == true
}
```

Abbildung 5.1: ChatGPT frage nach Standardrichtlinie(Bildschirmfoto aus ChatGPT)

für Firewalls in einer DMZ funktionierte ebenfalls nicht besonders gut. Es funktionierte allerdings besser, als nach einer Richtlinie gefragt wurde, die für eine DMZ erstellt wurde und Standarddienste anbietet, siehe [Abbildung 5.2](#) auf der nächsten Seite. Hier wurden allerdings

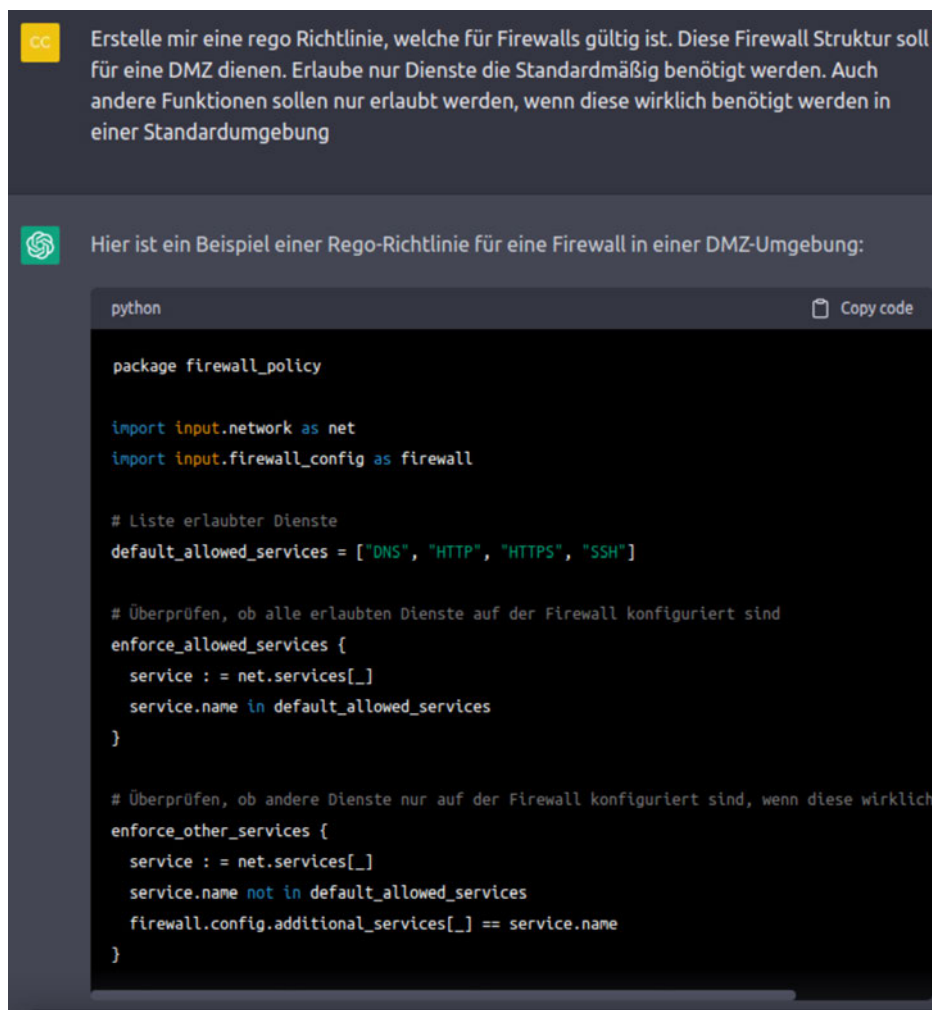


Abbildung 5.2: ChatGPT spezifische Frage nach einer Richtlinie(Bildschirmfoto aus ChatGPT)

nur Standarddienste erlaubt, da auch nach diesen explizit gefragt wurde, allerdings mehr nicht. So werden nur die Namen der Dienste abgefragt, allerdings keine Ports. An diesen Beispielen ist zu sehen, dass ChatGPT bei der Erstellung einer Richtlinie unterstützen kann, doch ist dies mit Aufwand verbunden, da ChatGPT genaue Anweisungen benötigt, um das gewünschte Ergebnis zu erhalten. Zudem muss diese Richtlinie erneut von Hand geprüft werden, aufgrund der spezifischen Anweisungen und der Umsetzung von ChatGPT. Das heißt, es ist nur möglich mit mehreren Nachfragen diese Firewallrichtlinie mit ChatGPT aufzubauen.

Die von der Richtlinie genehmigten Regeln werden direkt implementiert. Ein Test der Funktionalität der Firewallregeln wurde noch nicht erstellt. Eine automatische Testfunktion könnte implementiert werden, welche auf die Firewalls zugreift und prüft, ob die Regeln ordnungsgemäß implementiert sind. Ein Code zum Testen des entwickelten Programms wurde teilweise implementiert. Diese Tests sollen sicherstellen, dass das Programm und die implementierten Funktionen ordnungsgemäß funktionieren. Es muss sichergestellt werden, dass die Software auch weiterhin funktionsfähig ist, wenn sie weiterentwickelt oder gewartet wird. Durch eine regelmäßige Ausführung der CI/CD kann direkt erkannt werden, wenn Fehler auftreten. Für das Testen wurde die Testbibliothek „Pytest“ genutzt. Dies ist ein weitverbreitetes Testframework für Python mit vielen Erweiterungen, die beim Testen helfen. Um dies zu verbessern, wurde die Testautomatisierung „Nox“ benutzt. Mit dieser ist es möglich, den Code mit unter-

schiedlichen Python-Versionen zu testen, sogar unterschiedliche Python Implementierungen sind möglich. So ist unter anderem ein Test mit „Pyston“ oder „Pypy“ möglich. Die Nutzung von „Pyston“ funktionierte allerdings nicht, da einige der Abhängigkeiten nicht installiert werden konnten. Diese zwei Implementierungen sind Alternativen zu dem standardmäßig eingesetzten CPython, allerdings sind diese auf Leistung optimiert.

Dass die Richtlinien sehr komplex werden können, ist auch an den allgemeinen Richtlinien zu sehen, siehe [Abschnitt 3.1](#) auf Seite 21. Allgemeine Anforderungen, wie die von Behörden wie dem [BSI](#), welche umgesetzt werden müssen, sind meist lang und erklären nur was gemacht werden muss und nicht wie es gemacht werden muss.

Dort werden allgemeine Anforderungen dargestellt, die umgesetzt werden sollten. Zu diesen komplexen Standardregeln kommen noch zusätzliche, durch die Firma festgelegte Regeln für verschiedenste Dienste. Dies sind unter anderem spezielle Dienste, welche außergewöhnliche Regeln benötigen. Die allgemeinen Richtlinien allein bieten bereits einen guten Schutz des Netzwerks. Meine Forschung hat ergeben, dass die Aspekte der Erstellung einer geeigneten Richtlinie sehr komplex sind. Durch diese komplexen Richtlinien wird die Inbetriebnahme eines neuen Dienstes anspruchsvoller, da alle benötigten Freigaben eines neuen Dienstes geprüft werden müssen. Dafür ist es sicherer, da standardmäßig alles blockiert wird und nur die wirklich benötigten Freigaben erstellt werden. Es ist möglich, die meisten Richtlinien mit der erstellten Lösung durchzusetzen, allerdings gibt es auch einige, mit denen es nicht möglich ist, wie der Durchführung einer Wartung genauso wie der physischen Sicherheit, siehe [Abschnitt 2.1.6.5](#) auf Seite 13. Außerdem ist es auch sinnvoll, bestimmte Regeln direkt auf der Firewall zu implementieren, sodass es den Nutzern nicht möglich ist, diese zu ändern oder einen Antrag für diese Regeln zu erstellen. Beispielsweise könnte ein Unternehmen [FTP](#) blockieren, da dies ein veraltetes und unsicheres Programm ist. Bei der Implementierung direkt auf der Firewall würde dieses Protokoll dann generell für alle Nutzer deaktiviert werden und wäre durch einen Antrag auch nicht nutzbar.

Eine Überprüfung der Funktionalität in einer produktiv Umgebung konnte bei dieser Arbeit nicht durchgeführt werden, aufgrund der zeitlichen Begrenzung. Vor allem der typische Aufbau des Netzwerkes in einer [DMZ](#) Struktur konnte somit nicht getestet werden. Dies behindert eine Analyse der Eigenschaften dieser Forschung in einer großen Netzwerkkumgebung und vor allem auch im Zusammenspiel mit mehreren Firewalls. Die Auswirkung hat zwar den Nachteil, dass ein realer Anwendungsfall nicht aufgezeigt und untersucht werden konnte. Gleichwohl wird ein erster Schritt zum automatisierten Firewallrichtlinien-Management aufgezeigt, welcher bei einem Unternehmen umgesetzt werden kann. Jedoch kann diese Arbeit aufzeigen, wie das Programm in einer kleinen Testumgebung funktioniert. Diese Arbeit kann als Basis genommen werden zum Einstieg in dieses Thema und um erste Ansätze zu bekommen. Außerdem konnte leider nicht mit den Firewalls gearbeitet werden, die standardmäßig von Unternehmen genutzt werden. Die hier verwendete Lösung [FirewallD](#) eignet sich nicht für das Unternehmensumfeld, führt aber trotzdem den Einsatz und die Implementierung vor und zeigt auf, wie diese Arbeit dabei helfen kann, das Richtlinienmanagement der Firewall besser umzusetzen. Auch einige der verschiedenen Firewall-Funktionen wurden aufgezeigt, siehe [Abschnitt 2.1](#) auf Seite 5. Es ist zu sehen, dass Firewalls tendenziell immer mehr Funktionen bekommen. Dies könnte genutzt werden, um auch einige dieser Funktionen für Nutzer verwaltbar zu machen und ebenfalls durch die Richtlinien geprüft wird, ob es zulässig ist, diese

Funktionen zu ändern. Die Nachteile dieser Arbeit sind gering. Lediglich die Zeit, die für diese Arbeit und die Entwicklung nötig ist, kann nachteilig angesehen werden. Zudem wird noch Zeit für die Inbetriebnahme beim Kunden benötigt, sowie die Erstellung einer geeigneten Richtlinie in Rego. Zudem gibt es noch die Möglichkeit, dass es Programmierfehler gibt oder dieses von Angreifern ausgenutzt werden. Dadurch könnte die Firewallkonfiguration geändert oder deren Funktionalität darunter leiden. Zudem könnte auf die [API](#) des [Ticket](#)-Servers zugegriffen werden. Es ist aber möglich, den Zugriff auf die [API](#) einzuschränken. Bei dem im [PoC](#) genutzten [Ticket](#) Server ist dies möglich.

6 Fazit und Ausblick

Diese Arbeit hat gezeigt, dass das Themengebiet der Firewalls groß ist und obwohl die Firewalls eines der ersten Sicherheitssysteme waren, immer noch im Einsatz sind. Auch wenn die Firewalls bereits einige Evolutionsschritte weiter sind und es Technologien gibt, die auf ihnen aufbauen, wie [IPSS](#). Die vielfältigen Funktionen, die Firewalls über die Zeit gewonnen haben, machen sie komplexer und verbessern sie. Auch die bessere Überwachung und tiefere Analysen des Datenverkehrs innerhalb des [ISO/OSI](#) Netzwerkmodells verbessert sich stetig. Es wurde auch gezeigt, dass es nicht nur Funktionen zum Schutz gibt. Die modernen Firewalls besitzen unter anderem die Funktionen des [UniDirectional Link Filtering \(URLF\)](#) oder der [NAT](#). Zudem hat diese Arbeit die Komplexität der Richtlinien gezeigt, um eine Firewall sicher zu implementieren. Dieses Werkzeug führt zu einer leichteren Umsetzung der Richtlinien und einer Entlastung der Mitarbeiter. Dies ist möglich, da die Regel innerhalb kürzester Zeit kontrolliert und angewendet werden kann. Das führt auch dazu, dass dynamisch Regeln angelegt werden können, etwa wenn ein Dienst nur während der Geschäftszeiten benutzt werden soll. Danach kann diese Regel deaktiviert werden, um zu verhindern, dass diese Dienste ausgeutzt werden können, beispielsweise wenn ohnehin kein Mitarbeiter mehr anwesend ist. Weiterhin konnte hier aufgezeigt werden, wie Open-Source-Produkte sinnvoll miteinander verbunden werden können, um die Arbeit der Nutzer und Administratoren zu vereinfachen. Durch dieses Konzept kann auch leicht die Sicherheit des Netzwerkes erhöht werden. Ein Problem an dieser Implementierung ist die unübersichtliche Erstellung der Regeln, da hierfür keine Dokumentation existiert. Die automatische Anwendung der Richtlinien ist ebenfalls ein Vorteil. Diese werden genau beachtet und sind dadurch minimal.

6.1 Zukunftsausblick

Bei dem Firewallautomatisierungstool handelt es sich bei dieser Arbeit zwar nur um eine Machbarkeitsstudie, doch diese kann durch Unternehmen genutzt werden, wenn sie weiterentwickelt wird. Dazu müssen noch weitere Funktionen implementiert werden. Dazu gehört unter anderem die Dokumentation der Firewallregeln. Die Regeln sollten einheitlich und übersichtlich dargestellt werden. Dadurch ist der Aufbau des Netzwerkes, klar ersichtlich. Durch diese Übersicht ist es einfach neue Dienste und Regeln zu erstellen. Zudem können im Netzwerk auftretende Probleme besser diagnostiziert werden. Hierfür wäre es gut eine dynamische Karte zu erstellen, auf dem die Geräte und gegebenenfalls Regeln angezeigt werden. Die Karte kann außerdem genutzt werden, um zu überprüfen, ob im Netzwerk die Sicherheit der Firewall Konfiguration gewährleistet werden kann. Daneben ist es auch möglich hier eine Netzwerküberwachung einzubauen, welche festhält, wie der Netzwerkfluss ist und welche Firewallregeln benutzt werden. Es kann auch eine Überprüfung erstellt werden, die überprüft, ob eine Regel mit den bestehenden Regeln kompatibel ist, um zu vermeiden, dass andere Nutzer dadurch beeinträchtigt werden. Dazu existieren zwei verschiedene Lösungsansätze. Die eine ist „[Pyvis](#)“, bei dieser kann ein dynamischer Graph erstellt werden, mit Kreisen und Verbindungen. Hierbei können die Punkte mit der Maus beliebig verschoben und in das Diagramm kann hinein und herausgezoomt werden. Dies ist ausgezeichnet für die Übersichtlichkeit, da hier die Sicht gut geändert werden kann, um sich die relevanten

Bereiche anzuschauen. Allerdings hat dies den Nachteil, dass es nur Punkte gibt, eine Unterscheidung zwischen Firewall, Servern und anderen Rechnern ist schwierig und nur an den Labeln erkennbar. Ein anderes Tool ist „[Diagrams](#)“, dieses bezeichnet sich selbst als [Diagram as Code \(DaC\)](#) Lösung. Mit dieser Lösung ist es möglich, komplexe Sachverhalte klar aufzuzeigen und vor allem die einzelnen Netzwerkkomponenten aufzuzeigen, da es viele Elemente und auch Cluster gibt. Mit den Clustern ist es möglich, Elemente zu gruppieren, insbesondere Elemente in einem Netzwerk. Die Elemente sind Abbildungen von verschiedenen Themen, wie [Terraform](#) und [Ansible](#).

Es ist außerdem wichtig, Erweiterungen nutzen zu können. Dies ist primär bedeutsam, um die Firewall und die [Ticket-Server](#) Anbindung, mit unterschiedlichen Systemen zu nutzen. [Firewall](#) ist zwar eine gute Wahl für diese Machbarkeitsstudie, aber Firmen nutzen üblicherweise leistungsfähigere Systeme wie [OPNSENSE](#). Genauso werden verschiedene [Ticket-Server](#) Lösungen von unterschiedlichen Unternehmen genutzt. Es wäre auch möglich eine dynamische [IP-Sperrung](#) einzubauen. Dazu wäre eine Lösung wie [CrowdSec](#) ideal. Über die Oberfläche könnten gesperrte [IP-Adressen](#) freigegeben werden, die irrtümlich als schädlich eingestuft wurden oder um selbst welche zu sperren. Weiterhin ist es möglich, die Infrastruktur automatisch erstellen zu lassen und dabei direkt die benötigten Regeln zu erstellen. Hierbei kann dem Nutzer ermöglicht werden, einen Server zu beantragen. Dieser kann dann mithilfe von [IaC](#) automatisch eingerichtet und konfiguriert werden. Für den Server wird entweder eine virtuelle Maschine oder ein Container erstellt. Dann kann durch das Automatisierungstool geprüft werden, ob die Richtlinien dies erlauben und dann passende Firewallregeln erstellen, sodass der Server direkt genutzt werden kann. Vor allem das automatische Erstellen von Diensten ist einfach, da diese bereits vorkonfiguriert und leicht anpassbar sind. Auch können Server direkt in der Cloud mit passenden Firewallregeln erstellt werden. Wie dies bereits an der automatischen Erstellung der Entwicklungsumgebung, siehe [Abschnitt 4.1](#) auf Seite 35 gezeigt wurde.

Eine weitere Möglichkeit besteht darin, nach einer vordefinierten Zeit zu überprüfen, ob eine Regel wirklich benötigt wird. Gibt es keine Antwort oder wird die Regel als unnötig erachtet, wird sie entfernt. Es besteht die Möglichkeit, nutzerspezifische Richtlinien zu erstellen oder noch besser gruppenbasierte Richtlinien, die die Rollen der Mitarbeiter in einem Unternehmen steuern. Zudem sollte eine Löschung der Regeln nur von einem Administrator oder dem Ersteller durchführbar sein.

Für die Zukunft wurde geplant, die Komponenten für die Firewall und das Ticketsystem austauschen zu können. Dadurch kann das entwickelte Werkzeug an die Kundenwünsche angepasst werden. Zusätzlich soll es möglich sein, dass Kunden mehrere Firewalls anschließen können, da im Standard-Firewall-Konzept eine [DMZ](#) vorgesehen ist. Allein die [DMZ](#) benötigt zwei Firewalls, je größer der Kunde ist desto mehr Firewalls besitzt er für unterschiedliche Netzwerke und Standorte. Um dies zu ermöglichen, wurde das Werkzeug „[Pluggy](#)“ begutachtet. Mit diesem ist es möglich einfach Plug-ins zu entwickeln, umso dynamisch mehrere Abhängigkeiten wie mehrere und unterschiedliche Firewalls zu nutzen. Es bietet viele Funktionen, um den Plug-in-Entwicklern die Arbeit zu erleichtern und die Plug-ins gut in das Programm zu integrieren und einheitlich auszuführen. Zudem soll das Werkzeug „fastapi“ in

der Zukunft genutzt werden, da dieses zur performanten Entwicklung von APIs entwickelt wurde. Eine der vielen Funktionen ist unter anderem die automatische Dokumentation und die einfache Erweiterbarkeit der API.

Abschließend konnte aufgezeigt werden, dass eine Automatisierung des Prozesses zur Freischaltung der Firewallregel erhebliche Vorteile gegenüber dem manuellen Prozess bietet. Der Machbarkeitsbeweis konnte somit in dieser Arbeit erbracht werden und verschiedene Verbesserungsmöglichkeiten sind denkbar um die erstellte Lösung auch im Unternehmensumfeld einsetzen zu können.

Anhang A: Zusatz

Dieses Projekt soll in Zukunft unter einer Open-Source-Lizenz veröffentlicht werden und wird dann unter dem Link: <https://gitlab.com/projects/38691043> erreichbar sein. Eine kleine Anleitung zum Nutzen des Programms wird im Projekt-Wiki bereitgestellt, beziehungsweise liegt dem Programm bei. Diese ist in Englisch geschrieben, da es sich hier um ein Open-Source-Projekt handelt und es somit von jedem verwendet werden kann.

Glossar

ACK Das sogenannte „acknowledge“ Flag gibt an, dass Pakete empfangen wurden.

ACL Eine Liste, welche die Zugriffsrechte für jeden Benutzer speichert.

Active Directory Ein Nutzer Verzeichnisdienst unter Windows. Hier werden die Nutzer sowie ihre Zugriffsberechtigungen gespeichert. Dadurch ist es möglich mit einem Account auf unterschiedlichen Diensten anzumelden.

Amplification Attacken Bei einer Amplification Attacke wird die Quelladresse mit der des Opfers ersetzt und an einen Server geschickt. Dieser antwortet auf diese Attacke und verschleiert somit die IP Adresse des Angreifers. Außerdem soll eine kleine Anfrage eine möglichst große Anfrage zurückgeben, um das Opfer zu überlasten(DoS).

Ansible Eine Automatisierungslösung um Aufgaben auf entfernten Computern auszuführen[19].

API Eine API ist eine Schnittstelle eines Systems, die dazu gedacht ist, dass sie von anderen Systemen abgefragt werden kann.

Application Layer Die oberste Schicht im ISO/OSI Netzwerkmodell. In dieser Schicht werden die Protokolle genutzt, welche der User direkt nutzt, wie beispielsweise SSH, FTP oder HTTP.

Bastion Eine Firewall, die direkt an der Netzwerkgrenze steht und das Internet und die Firma trennt. Sie beschützt dabei das Netz in dem DMZ Aufbau.

Broadcast Propagation Starke Anhäufung von Broadcast Anfragen. Am Ende eines Angriffes können keine neuen Verbindungen aufgebaut und bestehende können unterbrochen werden. Der Netzwerkverkehr kann sich durch Broadcast Angriffe dramatisch erhöhen. Ein beispiel sind Smurf Attacke Angriffe.

CDN Verteilung von Daten auf mehreren Servern um die ganze Welt, um einen möglichst schnellen und hochverfügbaren Dienst anbieten zu können. Unter anderem Websites, welche viel aus verschiedenen Ländern benutzt werden, nutzen diesen Dienst.

Challenge Response Ein Verfahren, um nachzuweisen, dass man im Besitz eines Geheimnisses ist.

ChatGPT Eine neue KI die als Chatbot fungiert und im Chat Fragen beantworten kann[23].

Choke Die Firewall die zwischen dem lokalen und dem DMZ Netz liegt und das interne Netz beschützt. Es handelt sich hier um eine DMZ Architektur.

CI/CD Techniken zum Entwickeln, Testen und bereitstellen von Software. „Ziel ist es, die frühzeitige Entdeckung von Fehlern zu erhöhen, die Produktivität zu steigern und schnellere Veröffentlichungszyklen zu erreichen.“[8].

Cloudflare Ein Anbieter von CDN, sowie der DDoS Abwehr. Hierbei werden die Angriffe analysiert um sie abwehren zu können. Zudem können sie über das CDN geleitet werden um Ausfälle zu vermeiden[9].

Cockpit Eine grafische Oberfläche zur Verwaltung eines Linux-Systems[10].

Common Criteria Bezeichnet einen internationalen Standard zur Bewertung der IT-Sicherheit von Produkten. Das grundlegende Modell unterscheidet hier zwischen Funktionsumfang und der Qualität(Vertrauenswürdigkeit)[48].

CPython Die Standard Python Implementierung die genutzt wird. Da sie in C geschrieben wurde, wird sie CPython genannt[58].

- CrowdSec** CrowdSec analysiert die Logs mit dem sogenannten „Agent“. Erkennt CrowdSec Angriffe wird ein „Bouncer“ beauftragt die entsprechende IP-Adresse zu sperren. Es gibt „Agents“ und „Bouncer“ in unterschiedlichen Ausführungen für unterschiedliche Einsatzzwecke. Zusätzlich wird die IP-Adresse an das CrowdSec Netzwerk geschickt. Dieses sammelt und analysiert die Meldungen und prüft diese mit eigenen Verfahren. Die von CrowdSec erstellte Liste wird dann an die CrowdSec Agenten geschickt, welche diese IP-Adressen dann blockieren können, um global erkannte Angriffe schnellstmöglich abwehren zu können[12].
- DDoS** Eine gleichzeitige Attacke von mehreren Systemen, bei welchem so viele Anfragen geschickt werden, dass der Server nicht ordnungsgemäß laufen kann.
- dediziert** Dediziert bedeutet in der IT so viel wie eigenständig. Ein Server bei einem Hosting Provider ist beispielsweise dediziert, wenn nur ein Kunde Zugriff auf ihn hat und somit nicht mit anderen Kunden teilen muss[31].
- Deep Packet Inspection** Bei der Deep Packet Inspection wird das gesamte Packet gescannt, auch der entschlüsselte Payload, der die Daten enthält.
- DHCP** Protokoll zur Verteilung von IP Adressen in einem Netzwerk.
- Diagrams** Ermöglicht die Erstellung eines Diagramms mit Python. Bezeichnet sich dabei selbst als DaC-Lösung[15].
- DMZ** Zone innerhalb eines Netzwerkes, welche sich zwischen zwei Firewalls befindet und Schutz bieten soll. Die Firewalls stehen dabei zwischen dem Internet und der Zone sowie zwischen dem Internen Netzwerk und der Zone.
- DNS** Server zum Auflösen von Domains zu IP-Adressen.
- Domain** Weist einer IP-Adresse einen Namen zu. Die Domain ist für Nutzer besser zu merken, zudem können dadurch die IPs für die Domains geändert werden.
- DoS** Ein Angriff, bei der viele Anfragen geschickt werden, sodass der Server diese nicht mehr alle bearbeiten kann.
- EAL** „In den Common Criteria sind verschiedene Stufen der Vertrauenswürdigkeit definiert. Diese nennen sich Evaluation Assurance Level (EAL) und reichen von EAL1 bis EAL7. Während Stufe EAL1 lediglich eine funktionelle Prüfung bescheinigt, bedeutet die höchste Evaluierungsstufe EAL7 "formal verifizierter Entwurf und getestet.“[48].
- FIN** Mit diesem Flag innerhalb des TCP Pakets wird der Wunsch, die Verbindung zu beenden, signalisiert.
- Firewalld** Eine Software-Firewall für den Einsatz auf Computern gedacht[73].
- FTP** Protokoll zum Abrufen von Daten von einem Server. Da das Protokoll alt und außerdem unverschlüsselt ist, sollte es nicht mehr benutzt werden.
- Gitlab** Eine Versionsverwaltung die zahlreiche Funktionen bietet, um es Entwicklern leichter macht Programme zu entwickeln[57].
- HAProxy** Hochverfügbares Load Balancing und reverse Proxy Funktionalität. Anfragen werden an verschiedenste Server im Netzwerk verteilt, welche unterschiedliche Dienste bereitstellen können[18].
- HTML** Beschreibungssprache, um Webseiten darzustellen.
- HTTP** Protokoll zur Übermittlung von HTML Dokumenten.
- HTTPS** Eine verschlüsselte Erweiterung des HTTP Protokolls. Das angehängte „s“ steht dabei für secure.

- ICMP** Protokoll zur Übermittlung von Fehlern und Status Nachrichten.
- IGMP** Das Protokoll wird für multicast Verbindungen in einem [IP](#) Netzwerk benutzt[47, S. 28].
- Interface** Ein Interface ist eine Netzwerkschnittstelle, über die mit einem Netzwerk kommuniziert wird. Dabei agiert sie zwischen einem Computer und einem Netzwerk. Eine klassische Software Schnittstelle ist die sogenannte „lo“ Schnittstelle, die die Kommunikation zwischen Diensten auf dem Rechner ermöglicht. Meistens werden diese Schnittstellen durch Netzwerkgeräte abgebildet. Die Netzwerkkarte mit dem LAN Anschluss erhält klassischerweise das Interface „eth0“.
- IoT** Auch Internet der Dinge genannt. Bezeichnet kleinste Computer mit ganz speziellen Aufgaben, welche meistens zu größeren Systemen gehören. Dazu gehören das Smart home, aber auch smartwatches sowie einplatinen Computer wie der Raspberry Pi.
- IP** Das IP Protokoll ist ein Netzwerkprotokoll im [ISO/OSI](#) Modell. Diese wird auch als Vermittlungsschicht bezeichnet. Es ist ein verbindungsloses Protokoll. Dieses Protokoll stellt den angeschlossenen Geräten zur Kommunikation die [IP](#)-Adressen zur Verfügung.
- IP Source Routing** „Source Routing (Abk.: SR) ist eine Wegwahltechnik, mit der der Senderknoten eine vollständige Wegsequenz zum Zielknoten bestimmen kann.“[52] Damit ist es möglich, den Netzwerkverkehr auch über einen legitimen Rechner zu leiten, um Zugriff zu bekommen. Allerdings wurde das [IP](#) Source Routing in [IPv6](#) entfernt[24, S. 126].
- IPS** System welches verhaltens- oder signaturbasiert Angriffe erkennt, wie das [IDS](#). Doch zusätzlich noch mit der Möglichkeit Gegenmaßnahmen zu ergreifen.
- ISO/OSI** Das [ISO/OSI](#) Netzwerkmodell, soll die zur Übertragung benutzten Protokolle darstellen, welche zur Übertragung nötig sind. Diese Architektur ist in sieben Schichten gegliedert, wobei jede Schicht eine andere Funktion übernimmt. Wird eine bestimmte Schicht angewandt, müssen auch alle darunter liegenden Schichten eingesetzt werden, die darüber liegenden werden jedoch nicht verwendet.
- JSON** Ein Format um Datenstrukturen in Textform zu speichern.
- JWT** JSON Web Tokens sind ein offener RFC Standard. Diese Tokens enthalten Daten um beispielsweise einen Nutzer zu authentifizieren[3].
- Kerberos** Protokoll zum einfachen Einloggen bei mehreren Diensten auch „Single Sign On“ genannt. Nach erfolgreichem Einloggen wird ein Ticket erstellt mit dem es möglich ist sich bei konfigurierten Systemen anzumelden.
- KRITIS** Das [BSI](#) beschreibt Kritische Infrastrukturen wie folgt: „[...] Organisationen und Einrichtungen mit wichtiger Bedeutung für das staatliche Gemeinwesen, bei deren Ausfall oder Beeinträchtigung nachhaltig wirkende Versorgungsengpässe, erhebliche Störungen der öffentlichen Sicherheit oder andere dramatische Folgen eintreten würden.“ Dazu zählen unter anderem Krankenhäuser und Energieversorger[65].
- Link11** Ein Anbieter für den Schutz vor [DDoS](#) Angriffen. Eine selbstlernende [KI](#) analysiert Angriffe um zukünftige Angriffe abzuwehren[30].
- MFA** Mehrere Faktoren werden zur Authentisierung und Authentifizierung genutzt, etwa Passwort und Fingerabdruck.
- Netzwerk Stack** Beschreibt den Aufbau, der für die Übertragung erstellten Aufbau der Netzwerkfunktionalität, welche in einem Schichtenmodell aufgebaut ist, siehe [ISO/OSI](#).
- NFS** Linux Datei Server.

- Nox** Ein Werkzeug um Tests automatisiert in unterschiedlichen Python Umgebungen auszuführen[69].
- NTP** Das „Network Time Protocol“ wird verwendet, um die Zeit bei einem Server abzufragen. Dabei bezieht ein Zeitserver diese Zeit entweder direkt von einer Atomuhr oder durch einen Dritten, der diese Zeit weiterleitet wie den Internetprovidern oder große Firmen. Ein Beispiel ist die [Physikalische-Technische Bundesanstalt](#), welche eine Atomuhr bereitstellt und NTP Server, welche diese Zeit zur Verfügung stellen.
- OPNSENSE** Eine Open-Source Sicherheitslösung die verschiedene Lösungen wie Firewall, Routing und [IDS/IPS](#) Funktionen vereint. Dabei basiert sie auf [PFSENSE](#) und ist für den Einsatz in großen Netzwerken gedacht. Dazu existieren auch Funktionen zum sicherstellen einer hohen Verfügbarkeit. Es werden die meisten Funktionen kommerzieller Firewalls angeboten und weitere[37].
- Paramiko** Eine [SSH](#) Implementierung in Python[70].
- PFSENSE** Eine Open-Source Firewall, Routing und [IDS/IPS](#) Lösung, die viele Funktionen von Business Firewalls anbietet. Im Gegensatz zu [OPNSENSE](#) konzentriert sich diese Lösung mehr auf die Anwendung im Unternehmensbereich[39].
- Playwright** Ein Werkzeug mit dem es möglich ist Web-Anwendungen zu testen oder automatisiert Aufgaben auf Web-Anwendungen auszuführen[16].
- Pluggy** Ein Werkzeug, welches in Python programme integriert werden kann, um Erweiterungen einzubinden und zu verwalten[40].
- Port** Der Port bezeichnet einen Eingang am Computer. Auf einem Port kann ein Dienst laufen, also Anfragen annehmen. Außerdem kann eine Anwendung, welche eine Anfrage an einen Dienst gestellt hat, über einen Port mit diesem kommunizieren.
- Pyenv** Verwaltet mehrere Python Versionen und installiert diese[49].
- Pyodide** Eine Python Distribution für den die es mit hilfe von [Wasm](#) ermöglicht Python im Browser auszuführen[42].
- Pypy** Eine Python Implementierung die performanter sein soll und sich an den Python Standard halten soll[55].
- Pyscript** Ermöglicht die einfache Ausführung von Python Code im Browser[21].
- Pyston** Eine Python Implementierung die eine bessere Performanz haben soll[43].
- Pytest** Ein Werkzeug, dass Tests ausführt, um die Funktionalität eines Programmes zu testen[44].
- Pyvis** Ein Werkzeug mit dem es möglich ist einen dynamischen Netzwerk Graph in Python zu erstellen[60].
- Pywebio** Ein Framework, mit dem es möglich ist mit wenig Code eine webseite zu erstellen[45].
- RFC** Technische und organisatorische Dokumente zum Internet, die Spezifikationen beschreiben. Einige sind sogar Internetstandards[46].
- Sandbox** Eine vom restlichen System abgeschottete Umgebung. In dieser Umgebung können potenziell gefährliche Anwendungen ausgeführt werden. Dadurch ist es möglich das Verhalten dieser Anwendungen zu aktualisieren .
- SCTP** Protokoll zur gleichzeitigen Übertragung mehrerer Datenströme zwischen zwei Endpunkten. Es ist so entworfen, dass sich eine Telefonverbindung leichter über das Internet nutzen lässt. Im Speziellen ist es für das SS7 Protokoll entwickelt worden, welches neben Sprache noch Signale übermittelt[64].

- SIEM** System zur Analyse und Auswertung von Log Daten aus verschiedensten Quellen.
- SMTP** Protokoll zum Senden und Weiterleiten von Mails über das Netzwerk.
- Smurf Attacke** Bei dieser Attacke wird ein mit einer gefälschten Quelladresse erstelltes **ICMP** Echo Paket an die Broadcast Adresse eines Netzes gesendet. Die Geräte im Netzwerk antworten und erzeugen ein solch hohes Datenvolumen und verschleiern außerdem die **IP** des Angreifers[24, S. 131].
- SNMP** „SNMP ist ein Protokoll zur Verwaltung und Steuerung von Netzwerken. Es ist für den Transport von Management-Informationen, Status- und Statistikdaten zwischen Netzwerkstationen und einem Management-System zuständig“[50].
- Sphinx** Ein Werkzeug um eine Dokumentation des Codes zu erstellen[68].
- Spoofing** Eine Technik, bei der die Quell Adresse gefälscht wird.
- SSH** Zugriff auf die Shell eines entfernten Computers über eine verschlüsselte Verbindung.
- Starvation Attacke** Ein Client sendet mehrere gültige Anfragen an einen **DHCP** Server, um alle **IP** Adressen zu belegen. Andere Geräte haben dann keine **IP** Adressen, welche sie nutzen können.
- SYN** Dieses Flag gibt an, dass ein Verbindungsaufbau erwünscht ist.
- TCP** Protokoll zur vollständigen Übertragung von Daten. Dazu wird eine Sitzung aufgebaut und die vollständige Übertragung der Daten wird gewährleistet.
- Terraform** Eine **IaC**-Lösung um automatisch Infrastruktur zu erstellen[56].
- Ticket** Eine Anfrage bezieht sich meistens auf ein Problem oder eine Verbesserung. Dies wird bearbeitet und die erbrachte Arbeit kann dokumentiert werden.
- Transportprotokoll** Transportprotokolle bezeichnen die in der **ISO/OSI** Schicht 4 der **Transportschicht** angesiedelten Protokolle. Es handelt sich hier um Ende zu Ende Protokolle, welche sicherstellen soll, dass die Pakete richtig zugestellt werden. Dabei gibt es in dieser Schicht das verbindungsorientierte Protokoll **TCP** und das verbindungslose Protokoll **UDP**.
- Transportschicht** Die Transportschicht bezeichnet eine Schicht des **ISO/OSI** Schichtenmodells. In dieser Schicht sind die **Transportprotokolle**, die sicherstellen sollen, dass die Pakete ankommen.
- TTL** Gibt die Zahl an, wie lange ein Paket gültig ist. Bei jeder Zwischenstation wie einem Router wird diese Zahl dekrementiert. Dies wird gemacht, um eine Endlosschleife innerhalb des Netzwerkes zu verhindern.
- UDP** Protokoll zur schnellen Übertragung von Daten, wobei auch Pakete verloren gehen dürfen, unter anderem für Livestreams und Videokonferenzen.
- VoIP** Das Protokoll wird verwendet, um Sprache über ein Netzwerk zu vermitteln. Es baut dabei auf dem **IP** Protokoll auf[62].
- VPN** Das VPN-Protokoll wird genutzt, um eine verschlüsselte Verbindung zu einem anderen Computer oder Netzwerk aufzubauen. So ist es möglich so zu agieren, als wäre man wirklich in diesem Netzwerk, mit all den Vorteilen und Nachteilen dieses Netzwerkes.
- Webhook** Ein Webhook wird von Webapplikationen ausgelöst um ein Event zu melden, beispielsweise wenn neue Daten vorliegen. Dadurch ist es nicht mehr nötig regelmäßig Anfragen an einen Webserver stellen um über neue Events benachrichtigt zu werden[67].
- X11 Display Server** Ein Server zur Darstellung einer grafischen Oberfläche.

XML Eine Beschreibungssprache, um Daten zu speichern und zu transportieren. Sie soll dabei selbstbeschreibend sein und ist ähnlich wie [HTML](#) aufgebaut[74].

YAML Einfache und leicht zu lesende Datenbeschreibungssprache.

Zammad Ein System zum erfassen und bearbeiten von [Tickets](#)[20].

Zero Day Eine vorher noch unbekannte Schwachstelle.

Literaturverzeichnis

- [1] *8 Firewall Best Practices for Securing the Network*. Check Point Software. URL: <https://www.checkpoint.com/cyber-hub/network-security/what-is-firewall/8-firewall-best-practices-for-securing-the-network/> (besucht am 24. 11. 2022).
- [2] Andi Jehan Alhasan und Nico Surantha. „Evaluation of Data Center Network Security based on Next-Generation Firewall“. In: *International Journal of Advanced Computer Science and Applications* 12.9 (2021). ISSN: 21565570, 2158107X. DOI: [10.14569/IJACSA.2021.0120958](https://doi.org/10.14569/IJACSA.2021.0120958). URL: <http://thesai.org/Publications/ViewPaper?Volume=12&Issue=9&Code=IJACSA&SerialNo=58> (besucht am 01. 08. 2022).
- [3] auth0.com. *JWT.IO*. URL: <http://jwt.io/> (besucht am 15. 03. 2023).
- [4] BSI. *Firewall - Schutz vor dem Angriff von außen*. URL: <https://www.bsi.bund.de/dok/6596882> (besucht am 10. 11. 2022).
- [5] BSI. *NET.3.2 Firewall*. Feb. 2021. URL: [https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/IT-GS-Kompendium Einzel PDFs 2022/09 NET Netze und Kommunikation/NET 3 2 Firewall Edition 2022.pdf? blob=publicationFile&v=3#download=1](https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/IT-GS-Kompendium/NET_3_2_Firewall_Edition_2022.pdf?blob=publicationFile&v=3#download=1) (besucht am 12. 12. 2022).
- [6] *Bypassing Firewall Rules | Nmap Network Scanning*. URL: <https://nmap.org/book/firewall-subversion.html> (besucht am 27. 03. 2023).
- [7] Check Point Software Technologies Ltd. *High Availability (HA) Firewall*. Check Point Software. URL: <https://www.checkpoint.com/cyber-hub/network-security/what-is-firewall/high-availability-ha-firewall/> (besucht am 05. 12. 2022).
- [8] *CI/CD*. In: *Wikipedia*. Page Version ID: 230037963. 20. Jan. 2023. URL: <https://de.wikipedia.org/w/index.php?title=CI/CD&oldid=230037963> (besucht am 31. 01. 2023).
- [9] *Cloudflare: sicheres Hosting in der Cloud*. Cloudflare. URL: <https://www.cloudflare.com/de-de/> (besucht am 26. 03. 2023).
- [10] *Cockpit Project — Cockpit Project*. URL: <https://cockpit-project.org/FIXME> (besucht am 26. 03. 2023).
- [11] *Comparison to Other Systems*. Open Policy Agent. URL: <https://openpolicyagent.org/docs/latest/comparison-to-other-systems/> (besucht am 25. 01. 2023).
- [12] *CrowdSec - The open-source & collaborative security IPS*. URL: <https://www.crowdsec.net/> (besucht am 16. 11. 2022).
- [13] *Cyberangriffe - betroffene Unternehmen nach Art des Angriffs 2022*. Statista. URL: <https://de.statista.com/statistik/daten/studie/928943/umfrage/von-digitalen-angriffen-betroffene-unternehmen-nach-art-des-angriffs/> (besucht am 18. 01. 2023).
- [14] *Cybersicherheit - KRITIS-Meldungen bis 2022*. Statista. URL: <https://de.statista.com/statistik/daten/studie/1230654/umfrage/anzahl-der-kritis-meldungen-an-das-bsi/> (besucht am 26. 01. 2023).

- [15] *Diagrams · Diagram as Code*. URL: <https://diagrams.mingrammer.com/> (besucht am 26.03.2023).
- [16] *Fast and reliable end-to-end testing for modern web apps | Playwright*. URL: <https://playwright.dev/> (besucht am 26.03.2023).
- [17] Gordon Fyodor Lyon. *Port Scanning Techniques | Nmap Network Scanning*. URL: <https://nmap.org/book/man-port-scanning-techniques.html> (besucht am 12.12.2022).
- [18] *HAProxy*. In: *Wikipedia*. Page Version ID: 1128265405. 19. Dez. 2022. URL: <https://en.wikipedia.org/w/index.php?title=HAProxy&oldid=1128265405> (besucht am 16.01.2023).
- [19] Ansible Hat Red. *Ansible is Simple IT Automation*. URL: <https://www.ansible.com> (besucht am 26.03.2023).
- [20] *Helpdesk and Ticketing Software for Your Business | Zammad*. 13. März 2023. URL: <https://zammad.com/en> (besucht am 26.03.2023).
- [21] Anaconda Inc. *Pyscript.net*. URL: <https://pyscript.net/> (besucht am 26.03.2023).
- [22] *Infrastructure-DDoS-Protection von Link11*. Link 11 - Next Generation Cyber Security. URL: <https://www.link11.com/de/infrastructure-ddos-protection/> (besucht am 18.01.2023).
- [23] *Introducing ChatGPT*. URL: <https://openai.com/blog/chatgpt> (besucht am 26.03.2023).
- [24] Isi-Projektgruppe. „ISI-S Sichere Anbindung von lokalen Netzen an das Internet“. In: (26. Aug. 2014). URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Internetsicherheit/isi_lana_studie_pdf.pdf?blob=publicationFile&v=1.
- [25] *IT-Sicherheit - Investitionen in Sicherheitsvorkehrungen in Unternehmen Deutschland 2021*. Statista. URL: <https://de.statista.com/statistik/daten/studie/186758/umfrage/sicherheitsvorkehrungen-gegen-spionage-und-datenklau-in-deutschen-unternehmen/> (besucht am 08.01.2023).
- [26] Florian Kalhammer. „Crashkurs Firewallaufbau unter Linux“. In: (), S. 96.
- [27] Martin Kappes. „Richtlinien für die Praxis“. In: *Netzwerk- und Datensicherheit: Eine praktische Einführung*. Hrsg. von Martin Kappes. Wiesbaden: Springer Fachmedien, 2013, S. 335–342. ISBN: 978-3-8348-8612-5. DOI: [10.1007/978-3-8348-8612-5_16](https://doi.org/10.1007/978-3-8348-8612-5_16). URL: https://doi.org/10.1007/978-3-8348-8612-5_16 (besucht am 13.01.2023).
- [28] Rajesh Kumar. *What is Policy as Code? A Complete Guide*. DevOpsSchool.com. 2. Apr. 2022. URL: <https://www.devopsschool.com/blog/what-is-policy-as-code-a-complete-guide/> (besucht am 15.08.2022).
- [29] Junyan Liang und Yoohwan Kim. „Evolution of Firewalls: Toward Securer Network Using Next Generation Firewall“. In: *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*. 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC). Jan. 2022, S. 0752–0759. DOI: [10.1109/CCWC54503.2022.9720435](https://doi.org/10.1109/CCWC54503.2022.9720435).
- [30] *Link11 - Next Generation Cyber Security*. Link 11 - Next Generation Cyber Security. URL: <https://www.link11.com/de/> (besucht am 26.03.2023).

- [31] Marco Kratzenberg. *Dediziert: Was bedeutet es denn wirklich?* GIGA. 10. Sep. 2014. URL: <https://www.giga.de/ratgeber/tipps/dediziert-was-bedeutet-es-denn-wirklich/> (besucht am 14. 12. 2022).
- [32] *Network IPS Evasion Techniques > CCNP Security IPS 642-627 Official Cert Guide: Network IPS Traffic Analysis Methods, Evasion Possibilities, and Anti-evasive Countermeasures* | Cisco Press. URL: <https://www.ciscopress.com/articles/article.asp?p=1728833&seqNum=3> (besucht am 27. 03. 2023).
- [33] Palo Alto Networks. *What is policy-as-code?* Palo Alto Networks. URL: <https://www.paloaltonetworks.com/cyberpedia/what-is-policy-as-code> (besucht am 15. 08. 2022).
- [34] Kishan Neupane, Rami Haddad und Lei Chen. „Next Generation Firewall for Network Security: A Survey“. In: *SoutheastCon 2018*. SoutheastCon 2018. ISSN: 1558-058X. Apr. 2018, S. 1–6. DOI: [10.1109/SECON.2018.8478973](https://doi.org/10.1109/SECON.2018.8478973).
- [35] „NIST Authenticated NTP Service“. In: *NIST* (10. Feb. 2010). Last Modified: 2021-01-15T10:40:05:00. URL: <https://www.nist.gov/pml/time-and-frequency-division/time-services/nist-authenticated-ntp-service> (besucht am 14. 12. 2022).
- [36] Digital Ocean. *Einrichten einer Firewall mit firewalld unter CentOS 8* | DigitalOcean. 5. Dez. 2020. URL: <https://www.digitalocean.com/community/tutorials/how-to-set-up-a-firewall-using-firewalld-on-centos-8-de> (besucht am 15. 08. 2022).
- [37] *OPNsense® a true open source security platform and more - OPNsense® is a true open source firewall and more*. URL: <https://opnsense.org/> (besucht am 26. 03. 2023).
- [38] *Path MTU Discovery* – Wikipedia. URL: https://wiki.less.org/wiki/Path_MTU_Discovery (besucht am 22. 12. 2022).
- [39] *pfSense® - World's Most Trusted Open Source Firewall*. URL: <https://www.pfsense.org/> (besucht am 26. 03. 2023).
- [40] *pluggy — pluggy 1.0.0 documentation*. URL: <https://pluggy.readthedocs.io/en/stable/index.html> (besucht am 26. 03. 2023).
- [41] *Port-Scanning-Methoden* | *Nmap-Referenz-Handbuch (Man Page)*. URL: <https://nmap.org/man/de/man-port-scanning-techniques.html> (besucht am 12. 12. 2022).
- [42] *Pyodide — Version 0.22.1*. URL: <https://pyodide.org/en/stable/> (besucht am 26. 03. 2023).
- [43] *Pyston | Python Performance*. Pyston. URL: <https://www.pyston.org> (besucht am 26. 03. 2023).
- [44] *pytest: helps you write better programs — pytest documentation*. URL: <https://docs.pytest.org/en/7.2.x/> (besucht am 26. 03. 2023).
- [45] *PyWebIO - Build full stack web app with Python*. URL: <https://www.pyweb.io/> (besucht am 26. 03. 2023).
- [46] *Request for Comments*. In: *Wikipedia*. Page Version ID: 230332561. 29. Jan. 2023. URL: https://de.wikipedia.org/w/index.php?title=Request_for_Comments&oldid=230332561 (besucht am 22. 03. 2023).

- [47] Karen Scarfone und Paul Hoffman. *Guidelines on Firewalls and Firewall Policy*. NIST Special Publication (SP) 800-41 Rev. 1. National Institute of Standards und Technology, 28. Sep. 2009. DOI: [10.6028/NIST.SP.800-41r1](https://doi.org/10.6028/NIST.SP.800-41r1). URL: <https://csrc.nist.gov/publications/detail/sp/800-41/rev-1/final> (besucht am 14. 12. 2022).
- [48] Peter Schmitz und Stefan Luber. *Was ist Common Criteria?* 19. Jan. 2018. URL: <https://www.security-insider.de/was-ist-common-criteria-a-677228/> (besucht am 26. 01. 2023).
- [49] *Simple Python Version Management: pyenv*. original-date: 2012-08-31T06:57:52Z. 26. März 2023. URL: <https://github.com/pyenv/pyenv> (besucht am 26. 03. 2023).
- [50] *SNMP - Simple Network Management Protocol*. URL: <https://www.elektronik-kompndium.de/sites/net/0902011.htm> (besucht am 11. 01. 2023).
- [51] Benfano Soewito und Charlie Erwin Andhika. „Next Generation Firewall for Improving Security in Company and IoT Network“. In: *2019 International Seminar on Intelligent Technology and Its Applications (ISITIA)*. 2019 International Seminar on Intelligent Technology and Its Applications (ISITIA). Aug. 2019, S. 205–209. DOI: [10.1109/ISITIA.2019.8937145](https://doi.org/10.1109/ISITIA.2019.8937145).
- [52] *Source Routing – Wikipedia*. URL: https://wikiless.org/wiki/Source_Routing (besucht am 24. 11. 2022).
- [53] Nenad Stojanovski und Marjan Gusev. „Architecture of a Identity Based Firewall System“. In: *arXiv preprint arXiv:1108.1344* (2011).
- [54] G.N. Stone, B. Lundy und G.G. Xie. „Network policy languages: a survey and a new approach“. In: *IEEE Network* 15.1 (Jan. 2001). Conference Name: IEEE Network, S. 10–21. ISSN: 1558-156X. DOI: [10.1109/65.898818](https://doi.org/10.1109/65.898818).
- [55] The PyPy Team. *PyPy*. PyPy. 28. Dez. 2019. URL: <https://www.pypy.org/> (besucht am 26. 03. 2023).
- [56] *Terraform by HashiCorp*. Terraform by HashiCorp. URL: <https://www.terraform.io/> (besucht am 26. 03. 2023).
- [57] *The DevSecOps Platform*. URL: <https://about.gitlab.com/> (besucht am 26. 03. 2023).
- [58] *This is Python version 3.12.0 alpha 6*. original-date: 2017-02-10T19:23:51Z. 26. März 2023. URL: <https://github.com/python/cpython> (besucht am 26. 03. 2023).
- [59] Judy Thompson-Melanson. „Learn About Firewall Evolution from Packet Filter to Next Generation“. In: 2014. URL: <https://www.semanticscholar.org/paper/Learn-About-Firewall-Evolution-from-Packet-Filter-Thompson-Melanson/bfc93729f7760e39e82ef98870ce0cf4d059ee64> (besucht am 07. 01. 2023).
- [60] *Tutorial — pyvis 0.1.3.1 documentation*. URL: <https://pyvis.readthedocs.io/en/latest/tutorial.html> (besucht am 26. 03. 2023).
- [61] *Umfangreicher DDoS-Schutz*. Cloudflare. URL: <https://www.cloudflare.com/de-de/ddos/> (besucht am 18. 01. 2023).
- [62] *VoIP - Voice over IP*. URL: <https://www.elektronik-kompndium.de/sites/net/0503131.htm> (besucht am 19. 12. 2022).
- [63] John Wack, Ken Cutler und Jamie Pole. *Guidelines on Firewalls and Firewall Policy*. National Institute of Standards und Technology, 1. Jan. 2002, S. 75. URL: <https://apps.dtic.mil/sti/pdfs/ADA399879.pdf>.

- [64] *Was ist SCTP (Stream Control Transmission Protocol)? - Definition von Whats.com.* ComputerWeekly.de. Sep. 2019. URL: <https://www.computerweekly.com/de/definition/SCTP-Stream-Control-Transmission-Protocol> (besucht am 12. 12. 2022).
- [65] *Was sind Kritische Infrastrukturen?* Bundesamt für Sicherheit in der Informationstechnik. URL: <https://www.bsi.bund.de/DE/Themen/KRITIS-und-regulierte-Unternehmen/Kritische-Infrastrukturen/Allgemeine-Infos-zu-KRITIS/allgemeine-infos-zu-kritis.html?nn=126640> (besucht am 26. 01. 2023).
- [66] *WebAssembly.* URL: <https://webassembly.org/> (besucht am 01. 02. 2023).
- [67] *WebHooks.* In: *Wikipedia.* Page Version ID: 206804838. 22. Dez. 2020. URL: <https://de.wikipedia.org/w/index.php?title=WebHooks&oldid=206804838> (besucht am 15. 08. 2022).
- [68] *Welcome — Sphinx documentation.* URL: <https://www.sphinx-doc.org/en/master/> (besucht am 26. 03. 2023).
- [69] *Welcome to Nox — Nox 2022.11.21 documentation.* URL: <https://nox.thea.codes/en/stable/index.html> (besucht am 26. 03. 2023).
- [70] *Welcome to Paramiko! — Paramiko documentation.* URL: <https://www.paramiko.org/> (besucht am 26. 03. 2023).
- [71] *What is a Firewall? The Different Types of Firewalls.* Check Point Software. URL: <https://www.checkpoint.com/cyber-hub/network-security/what-is-firewall/> (besucht am 07. 01. 2023).
- [72] *What is Firewall Software?* Check Point Software. URL: <https://www.checkpoint.com/cyber-hub/network-security/what-is-firewall/what-is-firewall-software/> (besucht am 02. 01. 2023).
- [73] Thomas Woerner. *Firewalld Project Website.* firewalld. URL: <https://firewalld.org/firewalld.github.io/> (besucht am 11. 01. 2023).
- [74] *XML Introduction.* URL: https://www.w3schools.com/xml/xml_what_is.asp (besucht am 19. 12. 2022).

Stichwortverzeichnis

Automatisierung, 1–3, 17, 19, 31, g, 33–35, 37, 39, 41, 45–47

BSI, 22, 39, 40, 42

DDoS, 14, 26

Deep Packet Inspection, 6, 10, 11

DHCP, 11

DMZ, 16, 26, 40, 42

DNS, 7, 26, 29

DoS, 7, 26

Fragmentierung, 6, 7

HTTP, 7, 11, 24, 29

HTTPS, 8, 29

IaC, 2

ICMP, 7, 11, 23–27, 30, 33, 34, 36

IDS, 11, 13, 39

IoT, 8

IPS, 6, 11, 13, 39, 45

ISO/OSI, 7, 8, 10, 12, 45

KI, 37, 39–41

Netzwerkpaket, 7, 9, 11, 13, 23, 25, 26, 36

Netzwerkschicht, 6

Next Generation Firewall, 6

NTP, 7, 29

Paketfilter, 6

Protokoll, 7

Prozess, 1–3

Sandbox, 9, 12

Spoofing, 6

SSH, 8, 34–36

Stateful inspection, 7

Stateless inspection Firewall, 6

SYN, 12, 27

TCP, 7, 11, 12, 27, 28

Transportschicht, 7

UDP, 7, 8, 11, 24, 27, 28

Unternehmensrichtlinie, 1–3, 10–12, 19, 21, 22, 37, 39

Virtualisierung, 13

VPN, 7, 16

Zero Day, 11

Zugriffskontrolle, 6, 12, 20

Eidesstattliche Erklärung

Hiermit versichere ich – Fabian Beu – an Eides statt, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Sämtliche Stellen der Arbeit, die im Wortlaut oder dem Sinn nach Publikationen oder Vorträgen anderer Autoren entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt oder anderweitig veröffentlicht.

Essen, 27. März 2023

Ort, Datum



Fabian Beu