

Duale Hochschule Sachsen
Staatliche Studienakademie Leipzig

Vergleich eines kommerziellen Intrusion-Prevention Systems und einer Open-Source-Lösung

Bachelorthesis

zur Erlangung der staatlichen Abschlussbezeichnung eines
Bachelor of Science (B. Sc.)

im Studiengang Informatik

Eingereicht von: Leon Baumgarten
Traupitzer Dorfstraße 23, 06729 Elsteraue
CS22-1
5002213

Erstgutachter: B.Sc. Oliver Hels
WBS IT-Service GmbH

Zweitgutachter: Diplom IT Carsten Nitsch
Duale Hochschule Sachsen

23. August 2025

Inhaltsverzeichnis

| | | |
|----------|---|-----------|
| 1 | Einleitung | 1 |
| 1.1 | Motivation und Problemstellung | 1 |
| 1.2 | Stand der Technik und Forschungslücke | 2 |
| 1.3 | Zielsetzung und Forschungsfragen | 3 |
| 1.4 | Aufbau der Arbeit | 4 |
| 2 | Theoretische Grundlagen | 6 |
| 2.1 | Bedrohungslage | 6 |
| 2.1.1 | Denial-of-Service-Angriff | 6 |
| 2.1.2 | Malware | 8 |
| 2.1.3 | Exploits | 10 |
| 2.2 | IDS / IPS Grundlagen | 12 |
| 2.2.1 | Vom passiven Detektieren zum aktiven Verhindern | 12 |
| 2.2.2 | Funktionsprinzip: Deep Packet Inspection (DPI) | 15 |
| 2.2.3 | Zentrale Erkennungsmethoden | 16 |
| 2.3 | Vorstellung der Systeme | 20 |
| 2.3.1 | Der kommerzielle Vertreter: FortiGate | 20 |
| 2.3.2 | Die Open-Source-Alternative: OPNsense | 22 |
| 2.3.3 | Synoptische Gegenüberstellung | 27 |
| 3 | Aufbau der Testumgebung | 31 |
| 3.1 | Physische und Virtuelle Komponenten | 31 |
| 3.1.1 | Test-Hardware | 31 |
| 3.1.2 | Virtualisierungssoftware | 33 |
| 3.1.3 | Virtuelle Maschinen (VMs) | 34 |
| 3.2 | Netzwerkarchitektur | 35 |
| 3.2.1 | Netzwerkarchitektur Testumgebung OPNsense | 35 |
| 3.3 | Konfiguration der Firewalls und IPS-Systeme | 36 |
| 3.3.1 | Konfiguration der FortiGate VM | 36 |
| 3.3.2 | Konfiguration von OPNsense | 39 |
| 4 | Testing | 43 |
| 4.1 | Definition der Testfälle | 43 |
| 4.1.1 | Netzwerk-Scans (Reconnaissance) | 43 |
| 4.1.2 | Denial-of-Service (DoS)-Angriffe | 45 |

| | | |
|----------|--|-----------|
| 4.1.3 | Exploits (Ausnutzung von Schwachstellen) | 45 |
| 4.1.4 | Passwortangriffe (Brute-Force) | 46 |
| 4.1.5 | Malware-Erkennung | 47 |
| 4.2 | Durchführen der Tests | 47 |
| 5 | Auswertung | 48 |
| 5.1 | Analyse der Testergebnisse | 48 |
| 5.2 | Vergleich der IPS-Systeme | 48 |
| 6 | Fazit und Ausblick | 49 |
| 7 | Abkürzungsverzeichnis | 50 |
| 8 | Selbstständigkeitserklärung | 51 |
| 9 | Abbildungsverzeichnis | 55 |

1 Einleitung

1.1 Motivation und Problemstellung

Die Digitalisierung durchdringt unaufhaltsam alle Bereiche von Wirtschaft und Gesellschaft. Während sie Effizienzgewinne, neue Geschäftsmodelle und eine globale Vernetzung ermöglicht, schafft sie gleichzeitig eine ebenso wachsende wie komplexe Angriffsfläche für Kriminalität. Die Bedrohungslage im Cyberraum hat sich in den letzten Jahren dramatisch verschärft und ist von einer Randnotiz für IT-Spezialisten zu einer strategischen Herausforderung für Unternehmen jeder Größenordnung geworden. Das Bundesamt für Sicherheit in der Informationstechnik (BSI) bewertet die Lage in seinem jährlichen Bericht als „angespannt bis besorgniserregend“ und verweist auf eine zunehmende Professionalisierung und Arbeitsteilung in der cyberkriminellen Schattenwirtschaft . [1]

Die Angriffsvektoren sind vielfältig und reichen von Denial-of-Service-Angriffen (DoS), die gezielt die Nichtverfügbarkeit von Diensten herbeiführen, über die Infiltration von Netzwerken mittels Schadsoftware (Malware) wie Viren und Trojanern, bis hin zur aktiven Ausnutzung von Software-Schwachstellen durch Exploits. Insbesondere Ransomware-Angriffe, bei denen Daten verschlüsselt und nur gegen Lösegeld wieder freigegeben werden, haben sich zu einer der größten Bedrohungen für die deutsche Wirtschaft entwickelt. Laut einer Studie des Digitalverbands Bitkom entstanden hierdurch allein im Jahr 2024 Schäden in einer Höhe von 266,6 Milliarden Euro, wobei ein Großteil der Angriffe auf kleine und mittelständische Unternehmen (KMU) zielte. [3]

Gerade KMU stehen vor einer besonderen Herausforderung: Sie verfügen oft nicht über die gleichen finanziellen und personellen Ressourcen wie Großkonzerne, um sich umfassend zu schützen. Gleichzeitig sind sie aufgrund ihrer Rolle in Lieferketten und als Träger der regionalen Wirtschaft ein attraktives Ziel. Der traditionelle Schutz des Unternehmensnetzwerks durch eine reine Paketfilter-Firewall, die den Verkehr nur anhand von Ports und IP-Adressen kontrolliert, ist angesichts der modernen Bedrohungslage nicht mehr ausreichend. Angriffe finden heute oft auf der Anwendungsebene statt und verstecken sich im legitimen Datenverkehr, beispielsweise über den standardmäßigen Web-Port 443.

An dieser Stelle setzen Intrusion Prevention Systeme (IPS) an. Als Weiterentwicklung von Intrusion Detection Systemen (IDS), die Angriffe nur erkennen und melden, sind IPS in der Lage, böartigen Datenverkehr aktiv zu analysieren, zu klassifizieren und in Echtzeit zu blockieren, bevor er Schaden im internen Netzwerk anrichten kann. Diese Systeme agieren als wachsame "Torwächter", die den Inhalt der Datenpakete tiefgehend inspizieren (Deep Packet Inspection) und mit bekannten Angriffsmustern (Signaturen) oder anormalem Verhalten abgleichen.

Für Unternehmen, die eine solche Schutztechnologie implementieren möchten, stellt sich eine grundlegende strategische Frage: Soll auf eine kommerzielle „All-in-One“-Lösung eines etablierten Herstellers gesetzt werden, oder kann eine flexiblere und potenziell kostengünstigere Open-Source-Lösung einen vergleichbaren Schutz bieten? Kommerzielle Produkte, oft als Next-Generation Firewalls (NGFW) vermarktet, versprechen eine hohe Integration, professionellen Support und eine einfache Verwaltung aus einer Hand. Demgegenüber stehen Open-Source-Alternativen, die durch Transparenz, Anpassbarkeit und den Wegfall von Lizenzgebühren überzeugen, jedoch oft ein höheres Maß an technischem Know-how in der Implementierung und Wartung erfordern.

Zusätzlich zu diesen externen Bedrohungen hat sich die Erkenntnis durchgesetzt, dass das interne Netzwerk (LAN) nicht per se als vertrauenswürdig gelten kann. Angriffe können ebenso von innen heraus erfolgen, sei es durch kompromittierte Endgeräte, die bereits Teil des Netzwerks sind, oder durch böswillige Insider. Dieses Paradigma, bekannt als Zero Trust, geht davon aus, dass Bedrohungen überall lauern können und somit auch der Datenverkehr innerhalb des Netzwerks einer Überprüfung bedarf. Eine zentrale Aufgabe moderner IPS ist es daher nicht mehr nur, die Grenze zum Internet zu schützen, sondern auch eine Segmentierung und Überwachung des internen Datenverkehrs zu ermöglichen.

1.2 Stand der Technik und Forschungslücke

Die vorliegende Arbeit positioniert sich exakt in diesem Spannungsfeld. Sie zielt darauf ab, den in der Praxis relevanten Entscheidungsprozess zwischen einem kommerziellen und einem Open-Source-System durch einen systematischen, technischen und praktischen Vergleich zu objektivieren.

Als Repräsentant für die kommerzielle Welt wird eine virtuelle Appliance (FortiGate VM) der Firma Fortinet herangezogen. Fortinet ist einer der weltweiten Marktführer

im Bereich der Netzwerksicherheit. Die FortiGate-Produktfamilie, die auf dem Betriebssystem FortiOS basiert, integriert Firewall-, VPN- und IPS-Funktionen in einer einzigen Lösung. Während physische FortiGate-Appliances ihre hohe Performance oft durch den Einsatz spezialisierter Security Processors (SPs) erreichen, wird in dieser Arbeit die Flexibilität und Skalierbarkeit der virtualisierten Variante untersucht.

Als Gegenstück wird OPNsense untersucht, eine weit verbreitete und aktiv weiterentwickelte Open-Source-Firewall-Distribution, die auf dem robusten Betriebssystem FreeBSD basiert. OPNsense kann auf Standard-Hardware betrieben werden und integriert für die IPS-Funktionalität die leistungsfähige Open-Source-Engine Suricata. Suricata ist ein hochperformantes IDS/IPS, das von der Open Information Security Foundation (OISF) entwickelt wird. Im Gegensatz zum kommerziellen Ansatz ist der Anwender hier selbst dafür verantwortlich, die Hardware zu dimensionieren und die Regelwerke (Signaturen) aus verschiedenen freien oder auch kostenpflichtigen Quellen zu beziehen und zu verwalten.

Während es zahlreiche Marketing-Vergleiche und allgemeine Gegenüberstellungen von Vor- und Nachteilen beider Ansätze gibt, fehlt es an detaillierten, praxisorientierten und reproduzierbaren Vergleichsstudien. Oft bleiben solche Analysen an der Oberfläche und vergleichen lediglich Feature-Listen, ohne die tatsächliche Schutzwirkung und die betrieblichen Aspekte in einer kontrollierten Umgebung zu testen. Diese Arbeit schließt diese Lücke, indem sie nicht nur die theoretischen Konzepte, sondern die konkrete Implementierung und Leistungsfähigkeit von FortiGate und OPNsense in einem eigens aufgebauten Testlabor direkt gegenüberstellt.

1.3 Zielsetzung und Forschungsfragen

Das primäre Ziel dieser Bachelorarbeit ist es, eine fundierte und objektive Entscheidungsgrundlage vor allem für KMU zu schaffen, die vor der Wahl eines geeigneten IPS stehen. Es soll ermittelt werden, inwieweit die Open-Source-Lösung OPNsense eine technisch ebenbürtige und praktikable Alternative zum kommerziellen Marktführer FortiGate darstellt.

Um dieses Ziel zu erreichen, wird die Untersuchung von folgender zentraler Forschungsfrage geleitet:

Inwieweit kann das Open-Source-System OPNsense in Bezug auf Schutzwirkung, Performance und Verwaltungsaufwand eine effektive Alternative zur kommerziellen Next-Generation-Firewall FortiGate für den Einsatz in kleinen und mittelständischen Unternehmen darstellen?

Zur systematischen Beantwortung dieser Hauptfrage werden die folgenden untergeordneten Forschungsfragen untersucht:

Worin liegen die konzeptionellen und architektonischen Unterschiede bei der Implementierung der IPS-Funktionalität in der FortiGate-Appliance und dem OPNsense-System mit Suricata?

Wie effektiv erkennen und blockieren beide Systeme eine Reihe definierter, praxisnaher Angriffe aus den Kategorien Malware, Exploits und Denial-of-Service in einer kontrollierten Testumgebung?

Welche qualitativen Unterschiede bestehen hinsichtlich des Konfigurationsaufwands, der Benutzerfreundlichkeit der Verwaltungsoberflächen und der Qualität der Protokollierungs- und Reporting-Funktionen?

1.4 Aufbau der Arbeit

Die Struktur der vorliegenden Arbeit orientiert sich konsequent an der Beantwortung der formulierten Forschungsfragen.

Das erste Kapitel legt die theoretischen Grundlagen. Es wird ein Überblick über die aktuelle Bedrohungslage gegeben und die Funktionsweise sowie die technologischen Konzepte von Intrusion Detection- und Prevention Systemen erläutert. Anschließend werden die beiden zu vergleichenden Systeme, FortiGate und OPNsense, mit ihren jeweiligen Architekturen und Besonderheiten vorgestellt.

Im zweiten Kapitel wird der Aufbau der Testumgebung detailliert beschrieben. Dies umfasst die Auswahl der Hardware- und Softwarekomponenten, die entworfene Netzwerkarchitektur sowie die grundlegende Konfiguration der beiden IPS-Systeme, um eine transparente und nachvollziehbare Testbasis zu schaffen.

Das dritte Kapitel widmet sich dem Testing. Hier werden die konkreten Testfälle, die zur Überprüfung der Schutzwirkung und Performance dienen, definiert und deren Durchführung systematisch protokolliert.

Die im Praxisteil gewonnenen Daten werden im vierten Kapitel ausgewertet. Zunächst werden die Testergebnisse analysiert und anschließend die beiden Systeme anhand der zuvor definierten Kriterien wie Erkennungsrate, Performance und Benutzerfreundlichkeit verglichen.

Das fünfte und letzte Kapitel fasst die Erkenntnisse in einem Fazit zusammen, beantwortet die Forschungsfragen und gibt eine abschließende Bewertung sowie eine Handlungsempfehlung ab.

2 Theoretische Grundlagen

2.1 Bedrohungslage

Die Kernaufgabe eines jeden technischen Sicherheitssystems, insbesondere eines Intrusion Prevention Systems (IPS), ist die Erkennung und Abwehr konkreter Angriffe. Eine systematische Analyse und ein aussagekräftiger Vergleich solcher Systeme, wie er in dieser Arbeit angestrebt wird, setzen daher zwingend ein klares Verständnis der zu bekämpfenden Bedrohungen voraus.

Bevor die Fähigkeiten der Testsysteme praktisch evaluiert werden können, muss dieses theoretische Fundament geschaffen werden. Aus diesem Grund konzentriert sich dieser Abschnitt auf drei fundamentale Angriffskategorien, die für die Prüfung eines IPS zentral sind: Denial-of-Service-Angriffe, die Infiltration durch Malware und die Ausnutzung von Exploits.

Die nachfolgenden Unterkapitel widmen sich jeweils einer dieser Kategorien. Es werden die technischen Grundlagen, gängige Varianten und die charakteristischen Merkmale erläutert, anhand derer ein Schutzsystem den jeweiligen Angriff erkennen kann. Dieses Wissen ist die Voraussetzung, um die Konfiguration der Systeme im Testaufbau nachzuvollziehen und die Ergebnisse der praktischen Tests zu analysieren. [4]

2.1.1 Denial-of-Service-Angriff

Ein Denial-of-Service-Angriff (DoS), oder in seiner heute weitaus verbreiteteren Form als Distributed-Denial-of-Service-Angriff (DDoS), zielt fundamental auf die Beeinträchtigung der Verfügbarkeit eines IT-Dienstes ab. Das Ziel ist die Sabotage des regulären Betriebs, indem die endlichen Ressourcen des Zielsystems gezielt erschöpft werden. Bei einem DDoS-Angriff wird diese Überlastung durch eine Vielzahl kompromittierter Systeme, einem sogenannten Botnetz, gleichzeitig herbeigeführt. Diese massive Parallelisierung macht eine simple Abwehr durch das Blockieren einzelner IP-Adressen praktisch unmöglich.

Technisch lassen sich diese Angriffe in drei Hauptkategorien unterteilen, die auf unterschiedliche Schichten des OSI-Modells abzielen:

Volumetrische Angriffe (Schicht 3/4): Diese Angriffe zielen darauf ab, die gesamte zur Verfügung stehende Netzwerkbandbreite der Internetanbindung des Ziels zu sättigen. Eine Flut von Paketen wird an ein Ziel gerichtet. Ein klassisches Beispiel ist der UDP-Flood, bei dem eine riesige Menge an UDP-Paketen an zufällige Ports des Zielsystems gesendet wird. Da UDP ein verbindungsloses Protokoll ist, muss der Server für jedes eingehende Paket prüfen, ob eine Anwendung auf dem Port lauscht, und eine ICMP-Antwort "Destination Unreachable" generieren, was seine Ressourcen bindet. Ein ICMP-Flood (oder Ping-Flood) funktioniert nach einem ähnlichen Prinzip, indem er das Ziel mit ICMP-Echo-Request-Paketen überflutet und es zwingt, eine ebenso große Anzahl von Echo-Reply-Paketen zu senden. Das primäre Ziel ist hier die reine Masse an Traffic.

Protokoll-Angriffe (Schicht 4): Diese Angriffe nutzen Schwachstellen in der Implementierung von Netzwerkprotokollen aus, um die Ressourcen von Netzwerkkomponenten wie Firewalls oder Load Balancern zu erschöpfen. Der bekannteste Vertreter ist der TCP-SYN-Flood. Er missbraucht den drei-Wege-Handshake von TCP (SYN, SYN-ACK, ACK). Der Angreifer sendet eine hohe Zahl von SYN-Paketen, oft von gefälschten Quell-IP-Adressen. Das Zielsystem antwortet mit SYN-ACK und reserviert Ressourcen in seiner Verbindungstabelle (Backlog Queue) für die erwartete ACK-Antwort. Da diese Antwort niemals eintrifft, bleiben die Verbindungen "halboffen", bis die Tabelle voll ist und keine neuen, legitimen Verbindungsanfragen mehr angenommen werden können und überlastet ist.

Angriffe auf der Anwendungsebene (Schicht 7): Diese Angriffe sind subtiler und oft schwerer zu erkennen, da sie scheinbar legitimen Traffic imitieren. Sie zielen nicht auf die Netzwerkbandbreite, sondern auf die CPU- und Speicherressourcen des Servers ab. Beispiele hierfür sind das wiederholte Anfordern sehr großer Dateien oder das Ausführen komplexer Suchanfragen über eine Website, die serverseitig aufwändige Datenbankoperationen auslösen. Auch Angriffe auf Login-Schnittstellen durch massenhafte POST-Requests oder das Ausnutzen rechenintensiver API-Endpunkte fallen in diese Kategorie. Da diese Angriffe oft mit einer relativ geringen Datenrate auskommen, können sie unter dem Radar traditioneller, rein volumetrisch arbeitender Schutzsysteme hindurchgehen.

[4, 5]

Für ein Intrusion Prevention System besteht die Herausforderung darin, die Muster dieser unterschiedlichen Angriffsarten von legitimen Lastspitzen zu unterscheiden und sie präzise zu blockieren, ohne den regulären Nutzerverkehr zu beeinträchtigen.

2.1.2 Malware

Der Begriff Malware, eine Kurzform für "malicious software", dient als Oberbegriff für jegliche Art von Software, die mit der Absicht entwickelt wurde, auf einem Computersystem unerwünschte oder schädliche Aktionen auszuführen, Daten zu entwenden oder unautorisierten Zugriff zu erlangen. Die Ziele sind dabei vielfältig und reichen vom Diebstahl sensibler persönlicher oder geschäftlicher Informationen über die Störung von Betriebsabläufen, bis hin zur vollständigen Übernahme der Kontrolle über ein System, um es beispielsweise als Teil eines Botnetzes für DDoS-Angriffe zu missbrauchen. Die Infektion eines Netzwerks erfolgt oft mit einem mehrstufigen Prozess.

Zuerst muss die Malware auf das Zielsystem gelangen (Zustellung), was meist durch das Öffnen von manipulierten E-Mail-Anhängen (Phishing), das Klicken auf Links zu kompromittierten Websites (Drive-by-Downloads) oder über infizierte externe Speichermedien geschieht. Anschließend wird die Malware durch eine Benutzerinteraktion oder eine Sicherheitslücke aktiviert (Ausführung) und installiert sich fest im System, um einen Neustart zu überdauern (Persistenz). In vielen Fällen kontaktiert die Schadsoftware danach einen externen Command-and-Control-Server des Angreifers, um Anweisungen zu empfangen oder gestohlene Daten zu übermitteln.

Je nach Vorgehensweise und primärer Funktion lässt sich Malware in verschiedene Haupttypen klassifizieren, deren Grenzen jedoch zunehmend verschwimmen. Eine grundlegende Unterscheidung ist für das Verständnis der Bedrohungslandschaft unerlässlich.

Viren:

Klassische Viren sind Programmsegmente, die sich nicht eigenständig ausführen können, sondern eine legitime, ausführbare Wirtsdatei benötigen, um sich zu verbreiten. Sobald der Nutzer diese infizierte Datei startet, wird auch der virale Code aktiviert, der sich dann in weitere Dateien auf dem System oder auf verbundenen Netzlaufwerken kopiert. Die Schadfunktion, die von der reinen Replikation bis zur Zerstörung von Daten reichen kann, wird erst durch diese Nutzerinteraktion ausgelöst.[1]

Würmer:

Im Gegensatz dazu sind Würmer eigenständige Schadprogramme, die sich aktiv und ohne Zutun des Nutzers über Netzwerke verbreiten. Sie nutzen gezielt Sicherheitslücken in Betriebssystemen oder Anwendungen aus, um sich von einem infizierten System auf andere, verwundbare Systeme zu replizieren. Dieser autonome Verbreitungsmechanismus kann zu einer extrem schnellen, kaskadenartigen Infekti-

onswelle führen, die ganze Unternehmensnetzwerke lahmlegt. Ein prominentes Beispiel hierfür ist der Wurm WannaCry, der 2017 die "EternalBlue"-Schwachstelle im SMB-Protokoll von Windows-Systemen ausnutzte, um sich weltweit zu verbreiten und auf den infizierten Systemen Ransomware zu installieren.[2]

Trojaner:

Trojaner, benannt in Anlehnung an das Trojanische Pferd der griechischen Mythologie, tarnen sich als nützliche oder legitime Software, um den Nutzer zur Installation zu bewegen. Sie enthalten jedoch eine versteckte, bösartige Funktion. Nach der Ausführung installieren sie oft eine Hintertür (Backdoor), die Angreifern einen permanenten und unbemerkten Fernzugriff auf das System ermöglicht. Solche Remote Access Trojans (RATs) erlauben es Angreifern, Daten zu stehlen, das System zu überwachen oder es als Teil eines Botnetzes für weitere Angriffe zu missbrauchen.[6]

Ransomware:

Eine besonders profitable und schädliche Form ist die Ransomware. Diese Malware verschlüsselt die Dateien des Opfers, sodass auf sie nicht mehr zugegriffen werden kann. Anschließend wird eine Lösegeldforderung angezeigt, meist zahlbar in Kryptowährungen, um die Anonymität der Täter zu wahren. In den letzten Jahren hat sich das Vorgehen zur sogenannten "Double Extortion" (doppelte Erpressung) weiterentwickelt. Hierbei werden die Daten vor der Verschlüsselung zusätzlich vom System des Opfers auf Server der Angreifer kopiert. Wird das Lösegeld nicht gezahlt, drohen die Täter nicht nur mit der permanenten Zerstörung der Daten, sondern auch mit deren Veröffentlichung.[7]

Spyware:

Davon abzugrenzen ist Spyware, deren Hauptziel es ist, verdeckt Informationen über den Nutzer, dessen Verhalten und die auf dem System gespeicherten Daten zu sammeln. Eine der bekanntesten Unterarten sind Keylogger, die jeden Tastaturanschlag protokollieren. Auf diese Weise können Angreifer sensible Informationen wie Passwörter, Kreditkartennummern oder private Nachrichten mitschneiden und an ihre Server übermitteln.

Durch die Analyse von bekannten Malware-Signaturen, das Erkennen von Anomalien im Protokollverfahren und das Blockieren von Verbindungen zu IP-Adressen und Domains können IPS-Systeme diese Bedrohungen erkennen und unterbinden.

2.1.3 Exploits

Der Begriff Exploit bezeichnet ein spezifisches Software-Fragment, einen Datenblock oder eine Befehlssequenz, die gezielt einen Programmierfehler oder eine konzeptionelle Schwachstelle in einem Computersystem oder einer Anwendung ausnutzt. Es ist entscheidend zwischen der Schwachstelle, bei der es sich um einen passiven, latenten Fehler im Code handelt und dem Exploit, dem aktiven Werkzeug zur Ausnutzung dieses Fehlers, zu unterscheiden. Das Ziel eines Exploits ist es, erhöhte Rechte zu erlangen (Privilege Escalation), beliebigen Code auf dem Zielsystem auszuführen (Arbitrary Code Execution) oder einen Denial-of-Service-Zustand auszulösen. Dies macht Exploits zu einem primären Vektor für den erstmaligen Einbruch in ein Netzwerk.[9]

Der Prozess eines Angriffs, der auf der Ausnutzung von Schwachstellen basiert, lässt sich in die sieben Phasen der Cyber Kill Chain unterteilen, die den Ablauf aus der Perspektive des Angreifers strukturieren. Die erste Phase ist die **Reconnaissance**, in der ein Angreifer Informationen über das Ziel sammelt, um potenzielle Angriffsvektoren zu finden. In der zweiten Phase, der **Weaponization**, wird ein passender Exploit-Code mit einer Nutzlast (Payload) – dem eigentlichen Schadcode – gekoppelt. Anschließend erfolgt die **Delivery**, bei der der vorbereitete Exploit an das Zielsystem übermittelt wird, beispielsweise über eine Phishing-Mail oder einen verwundbaren Netzwerkdienst.

In der vierten Phase, der **Exploitation**, wird der Exploit-Code zur Ausführung gebracht und löst den Fehler in der verwundbaren Software aus. Dies ermöglicht die Ausführung des mitgelieferten Payloads. Darauf folgt die **Installation**, in der die Payload-Persistenz auf dem System etabliert wird, um nach einem Neustart noch auf dem Zielsystem vorhanden zu sein. In der sechsten Phase, **Command and Control** (C2), baut die installierte Schadsoftware eine ausgehende Verbindung zu einem vom Angreifer kontrollierten Server auf. Über diesen Kanal kann der Angreifer das System fernsteuern. In der letzten Phase, **Actions on Objectives**, verfolgt der Angreifer seine eigentlichen Ziele, wie zum Beispiel den Diebstahl von Daten, die Ausbreitung im internen Netzwerk, oder die Verschlüsselung von Systemen zur Erpressung von Lösegeld.[8]

Zero-Day-Exploit:

Die gefährlichste Kategorie sind Zero-Day-Exploits. Der Begriff „Zero-Day“ leitet sich aus der Perspektive des Softwareherstellers ab: Ab dem Moment, in dem ein Exploit für eine bisher unbekannte Schwachstelle aktiv ausgenutzt wird, hat der Hersteller null Tage Zeit gehabt, einen entsprechenden Sicherheitspatch zu entwickeln und bereitzustellen. Dieses Unkenntnis seitens der Verteidiger verschafft den Angreifern ein kritisches Zeitfenster, in dem ihre Angriffe mit sehr hoher Wahrscheinlichkeit erfolgreich sind.

Die besondere Gefahr von Zero-Day-Exploits liegt in ihrer Fähigkeit, traditionelle, signaturbasierte Schutzmechanismen wie die meisten Antivirenprogramme und Intrusion Prevention Systeme vollständig zu umgehen. Da der Angriff neu und unbekannt ist, existiert keine Signatur, kein Muster und kein Hashwert, nach dem ein solches System suchen könnte. Der Exploit ist für die Verteidigungslinie demnach quasi unsichtbar.

Die Abwehr von Zero-Day-Angriffen erfordert daher fortschrittlichere Sicherheitsstrategien, die über die reine Signaturerkennung hinausgehen. Dazu gehören verhaltensbasierte Analyse (Heuristik) und Anomalieerkennung, bei denen ein System nicht nach bekannten Mustern, sondern nach ungewöhnlichem und potenziell bösartigem Verhalten sucht. Eine weitere wichtige Technik ist das Sandboxing, bei dem verdächtiger Code in einer isolierten, virtuellen Umgebung ausgeführt wird, um seine Aktionen zu beobachten, ohne das eigentliche System zu gefährden.

2.2 IDS / IPS Grundlagen

Nach der detaillierten Betrachtung der vielfältigen Bedrohungslandschaft im vorherigen Kapitel, werden nun die Systeme näher beleuchtet, die entwickelt wurden, um Netzwerke vor genannten Angriffen zu schützen. Während klassische Firewalls den Verkehr primär anhand von Adressen und Ports (Schicht 3 und 4 des OSI-Modells) filtern, gehen moderne Schutzmechanismen einen entscheidenden Schritt weiter, indem sie den Inhalt des Datenverkehrs analysieren. Im Zentrum dieser erweiterten Sicherheitsarchitektur stehen Intrusion Detection- und Intrusion Prevention-Systeme (IDS/IPS).

In den folgenden Absätzen werden dazu die theoretischen Grundlagen dieser Technologien erklärt. Es wird die evolutionäre Entwicklung vom rein passiven Erkennen eines Angriffs (Detection) hin zum aktiven Verhindern (Prevention) nachgezeichnet, um die grundlegenden Unterschiede und die Motivation hinter der Entwicklung von IPS zu beleuchten. Anschließend wird das technische Funktionsprinzip der Deep Packet Inspection (DPI) erläutert, das es diesen Systemen überhaupt erst ermöglicht, den Inhalt von Datenpaketen zu analysieren. Abschließend folgt die Vorstellung der verschiedenen Erkennungsmethoden, auf deren Basis ein System die Entscheidung trifft, ob es sich um legitimen oder bösartigen Datenverkehr handelt.

2.2.1 Vom passiven Detektieren zum aktiven Verhindern

Die historischen Vorläufer moderner Angriffserkennungssysteme sind die Intrusion Detection Systeme (IDS). Sie entstanden aus der Notwendigkeit heraus, die Sicherheitsarchitekturen von Netzwerken zu ergänzen, da klassische Paketfilter-Firewalls allein nicht mehr ausreichten, um Angriffe auf höheren Protokollebenen zu erkennen. Das grundlegende Funktionsprinzip eines netzwerkbasierten IDS (NIDS) ist die der passiven Überwachung. Ein IDS wird im Netzwerk so implementiert, dass es eine Kopie des gesamten zu überwachenden Datenverkehrs erhält, ohne selbst Teil des aktiven Datenpfades zu sein. Technisch wird dies meist über einen sogenannten Mirror-Port (auch SPAN-Port genannt) an einem Netzwerk-Switch realisiert, der den gesamten Verkehr eines oder mehrerer anderer Ports auf den Port des IDS spiegelt.

Eine alternative, oft als zuverlässiger geltende Methode, ist der Einsatz eines dedizierten Network Taps, einer Hardware-Komponente, die direkt in die Leitung eingeschleift wird und eine verlustfreie Kopie des Verkehrs auskoppelt. Unabhängig von der Methode agiert das IDS somit stets wie ein Beobachter, der den Verkehr analysiert, ihn jedoch nicht direkt beeinflusst. Stellt das System eine potenzielle Bedrohung fest, ist seine primäre Aufgabe das Auslösen eines Alarms. Dieser Alarm kann in Form eines Eintrags in einer Log-Datei, einer Benachrichtigung per E-Mail oder einer Meldung an ein übergeordnetes Security Information and Event Mana-

gement (SIEM) System erfolgen, wo die Daten für eine weitere Analyse korreliert werden. Die Kernfunktion eines IDS ist also die reine Detektion; es beantwortet die Frage: „Passiert hier gerade etwas Böses?“ [10].

Definition Network Tap:

Ein Network TAP (Test Access Point) ist eine dedizierte Hardware-Komponente, die eine exakte, verlustfreie Kopie des gesamten Datenverkehrs von einer physischen Netzwerkverbindung erstellt. Das Gerät wird direkt in die Kabelverbindung, beispielsweise zwischen einem Router und einem Switch, eingeschleift und leitet den durchfließenden Verkehr passiv an einen oder mehrere Monitor-Ports weiter, an denen das Überwachungssystem angeschlossen ist. Im Gegensatz zu einem SPAN-Port eines Switches, der bei hoher Netzwerkauslastung unter Umständen Pakete verwirft, garantiert ein TAP die Weiterleitung allen Traffics, einschließlich fehlerhafter Pakete. Zudem sind die meisten TAPs ausfallsicher konzipiert.[13]

Die entscheidende Schwäche dieses passiven Ansatzes liegt in der systembedingten Latenz zwischen der Erkennung eines Angriffs und der Einleitung einer Gegenmaßnahme. Nachdem ein IDS einen Alarm ausgelöst hat, ist eine Reaktion erforderlich, um den Angriff zu stoppen. Diese Reaktion kann manuell durch einen Administrator erfolgen, der beispielsweise eine neue Regel in der Firewall konfiguriert, oder durch ein nachgeschaltetes, automatisiertes System. In der Zeit, die dieser Prozess unweigerlich in Anspruch nimmt, kann der Angriff sein Ziel bereits erreicht und erheblichen Schaden verursacht haben. Ein weiteres gravierendes, operatives Problem ist die sogenannte „Alert Fatigue“ (Alarm-Müdigkeit). Hochsensibel konfigurierte IDS können eine enorme Menge an Alarmen generieren, von denen viele Falschmeldungen (False Positives) sind. Administratoren werden von dieser Flut an Informationen überfordert, was dazu führen kann, dass sie kritische, echte Alarme übersehen oder weniger ernst nehmen.

Um diese kritische Schutzlücke zu schließen und die Reaktionszeit auf null zu reduzieren, wurden die Intrusion Prevention Systeme (IPS) entwickelt. Im Gegensatz zu einem IDS wird ein IPS aktiv und „in-line“ im Netzwerk platziert und behebt damit dessen grundlegende Schwäche. Diese architektonische Entscheidung ist fundamental: Das IPS fungiert als aktiver Kontrollpunkt, den der gesamte Datenverkehr passieren muss, um sein Ziel zu erreichen. Dies birgt jedoch auch neue Risiken: Ein IPS kann zu einem Leistungsengpass (Bottleneck) werden und stellt einen Single Point of Failure dar. Fällt das IPS aus, wird die gesamte Netzwerkverbindung unterbrochen, sofern keine Bypass-Mechanismen vorhanden sind.

Diese direkte Positionierung im Datenstrom ermöglicht es einem IPS, nicht nur zu erkennen, sondern auch unmittelbar und automatisiert zu handeln. Erkennt es einen Angriff, kann es eine Reihe von vordefinierten Aktionen durchführen. Die Palette dieser Reaktionsmöglichkeiten ist breit und reicht von dem einfachen Blockieren oder Verwerfen (Block/Drop) einzelner bössartiger Datenpakete über das aktive Zurücksetzen von TCP-Verbindungen (Reset), um eine Sitzung sauber zu beenden, bis hin zur reinen Alarmierung (Alert). Letztere kann dafür genutzt werden, um neue Regeln im Überwachungsmodus zu testen, ohne den produktiven Verkehr zu gefährden. [11, 12]

Darüber hinaus bieten viele Systeme erweiterte, oftmals herstellerabhängige Reaktionsmöglichkeiten. Dazu zählt beispielsweise die temporäre Sperrung der angreifenden IP-Adresse (Shun) oder die Umleitung des Angriffsverkehrs auf ein Ködersystem (Quarantäne/Sinkhole) zur weiteren Analyse und zur Sammlung von Informationen über den Angreifer. Die genaue Auswahl und die Granularität der verfügbaren Aktionen sind ein wesentliches Unterscheidungsmerkmal des jeweiligen Produkts und ein Indikator für seine technische Reife. Welche dieser Funktionen von den in dieser Arbeit untersuchten Systemen FortiGate und OPNsense im Detail unterstützt werden, wird im Rahmen ihrer jeweiligen Vorstellung in Kapitel 2.3 detailliert erläutert.

2.2.2 Funktionsprinzip: Deep Packet Inspection (DPI)

Die technologische Grundlage, die moderne Intrusion Prevention Systeme von klassischen Firewalls unterscheidet und ihre Effektivität überhaupt erst ermöglicht, ist die Deep Packet Inspection (DPI). Während traditionelle Paketfilter ihre Entscheidungen ausschließlich auf Basis der Header-Informationen der Schichten 3 und 4 des OSI-Modells (IP-Adressen, Ports) treffen, geht die DPI-Technologie entscheidende Schritte weiter. Selbst die fortgeschrittenere Stateful Packet Inspection (SPI), die den Zustand von Verbindungen verfolgt, analysiert nicht den eigentlichen Inhalt der übertragenen Daten. Die DPI hingegen untersucht nicht nur die Header, sondern auch die Nutzlast (Payload) der Datenpakete und kann den Datenstrom bis zur Anwendungsschicht (Schicht 7) analysieren, um so den Kontext und die Bedeutung der übertragenen Daten zu verstehen.

Der Funktionsprozess einer DPI-Engine ist hochkomplex und muss in Echtzeit ablaufen. Der erste Schritt in diesem Prozess ist die Paket-Normalisierung und Strom-Reassemblierung. Ein IPS empfängt den Netzwerkverkehr als eine Sequenz einzelner Datenpakete, die durch Fragmentierung oder durch die Netzwerkübertragung selbst in falscher Reihenfolge (Out-of-Order) eintreffen können. Die DPI-Engine muss diese Pakete daher anhand ihrer TCP-Sequenznummern korrekt sortieren und zu einem kohärenten, bidirektionalen Datenstrom zusammensetzen. Dieser Schritt ist kritisch, da viele Angriffstechniken gezielt versuchen, durch manipulierte Fragmentierung die Erkennung zu umgehen. Darauf aufbauend muss die Engine den reassemblierten Datenstrom dekodieren, um die genutzten Anwendungsprotokolle wie HTTP, SMB oder DNS zu identifizieren. Anschließend wird der Strom gemäß der Spezifikation des jeweiligen Protokolls geparkt, also in seine logischen Bestandteile zerlegt, um zwischen Kontrollinformationen und der eigentlichen Nutzlast zu unterscheiden.

Im Kern der DPI steht dann die eigentliche Inhaltsanalyse. Nachdem die Daten durch die vorherigen Schritte aufbereitet und in ihren protokollspezifischen Kontext gebracht wurden, werden sie an die eigentliche Analyse-Engine des IPS weitergeleitet. Die Aufgabe dieser Engine ist es, den Inhalt der Nutzlast zu bewerten und eine Entscheidung über dessen Zulässigkeit zu treffen. Hierfür greift das System auf verschiedene logische Verfahren zurück, um bösartige von gutartigen Daten zu unterscheiden. Die genaue Funktionsweise dieser zentralen Erkennungsmethoden, insbesondere der signaturbasierten und der anomaliebasierten Analyse, wird im nachfolgenden Kapitel 2.2.3 detailliert erläutert.

Die größte technische Hürde stellt jedoch der zunehmend verschlüsselte Datenverkehr (SSL/TLS) dar. Um die Nutzlast von HTTPS-Verbindungen zu inspizieren, muss das IPS eine als SSL/TLS-Inspektion bekannte Methode anwenden und als

autorisierter „Man-in-the-Middle“ agieren. Dabei fängt das IPS die Verbindungsanfrage des Clients ab, baut selbst eine Verbindung zum Zielserver auf und präsentiert dem Client ein eigenes zur Laufzeit generiertes Zertifikat. Damit dieser Prozess funktioniert, muss das Zertifikat der IPS-internen Certifikatsstelle auf allen Client-Systemen als vertrauenswürdig hinterlegt sein. Nur so kann das IPS den Verkehr transparent entschlüsseln, mit den beschriebenen Methoden analysieren und anschließend neu verschlüsselt zum Zielserver weiterleiten. Dieser Vorgang ist extrem rechenintensiv und verdeutlicht die Notwendigkeit spezialisierter Hardware in leistungsfähigen IPS-Systemen. Ohne die Fähigkeit der DPI, tief in die Nutzlast von Datenpaketen zu blicken, wäre ein Schutzsystem blind für nahezu alle modernen Angriffe auf der Anwendungsebene. [12, 10]

2.2.3 Zentrale Erkennungsmethoden

Nachdem die Deep Packet Inspection (DPI) den Netzwerkverkehr in einen analysierbaren, reassemblierten Datenstrom umgewandelt hat, beginnt der eigentliche Analyseprozess. An dieser Stelle kommen die Erkennungs-Engines des IPS zum Einsatz, die auf zwei fundamentalen, konzeptionell unterschiedlichen Paradigmen basieren: der signaturbasierten Erkennung, die auch als Missbrauchserkennung (Misuse Detection) bekannt ist, und der anomaliebasierten Erkennung (Anomaly Detection). Diese beiden Methoden bilden das logische Herzstück eines jeden IPS.

Signaturbasierte Erkennung:

Die signaturbasierte Erkennung stellt den fundamentalen und historisch älteren Ansatz zur Identifizierung von Cyber-Angriffen dar. Sie basiert auf einem fest definierten, vorab erstellten Wissensschatz über bekannte Bedrohungen und sucht im Netzwerkverkehr exakt nach den Mustern, die diesen Bedrohungen zugeordnet sind. Das zugrundeliegende Prinzip ist vergleichbar mit dem eines Virenschanners, der nach den eindeutigen "Fingerabdrücken" bekannter Viren sucht, jedoch auf der Ebene von Netzwerkprotokollen und Datenströmen.

Der Lebenszyklus einer Signatur beginnt mit der Entdeckung einer neuen Bedrohung. Sicherheitsforscher bei Herstellern wie Fortinet oder in Open-Source-Communities analysieren neue Malware oder die Vorgehensweise eines Exploits im Detail. Dabei extrahieren sie eindeutige, unveränderliche Merkmale. Dies kann eine bestimmte Zeichenfolge in der Nutzlast, eine charakteristische Abfolge von Befehlen oder ein spezifischer Wert in einem Protokoll-Header sein. Auf Basis dieser Analyse wird eine Signatur als formale Regel erstellt. Eine moderne Signatur ist dabei weit mehr als eine simple Zeichenkette. Sie ist ein mehrdimensionales Konstrukt, das typischerweise folgende Elemente umfasst:

- **Header-Bedingungen:**

Diese legen den Geltungsbereich der Regel fest und filtern den zu untersuchenden Verkehr vor. Hierzu gehören Protokoll (TCP, UDP, ICMP), Quell- und Ziel-IP-Adressen (oder ganze Subnetze) sowie Port-Nummern.

- **Inhalts-Muster (Payload Content):**

Der Kern der Signatur. Hier werden die eigentlichen böartigen Muster definiert, oft als Byte-Sequenzen oder als flexible reguläre Ausdrücke.

- **Kontextuelle Optionen:**

Diese verfeinern die Regel, um Falschmeldungen zu minimieren. So kann eine Regel beispielsweise festlegen, dass ein bestimmtes Muster nur dann als böartig gilt, wenn es in einem HTTP-POST-Request an einen bestimmten URL-Pfad auftritt oder wenn es nicht von einer bestimmten anderen Zeichenfolge gefolgt wird.

- **Metadaten:**

Jede Signatur wird mit zusätzlichen Informationen wie einer eindeutigen ID, einer Referenz zur entsprechenden CVE-Nummer (Common Vulnerabilities and Exposures), einer Beschreibung des Angriffs und einer Klassifizierung der Bedrohungsschwere versehen

Technisch werden zwei Arten von Signaturen unterschieden: atomare Signaturen, die ein einzelnes Paket auf ein Muster hin untersuchen, und zustandsbehaftete (stateful) Signaturen, die den Zustand einer gesamten Verbindung über mehrere Pakete hinweg verfolgen und erst dann anschlagen, wenn eine bestimmte Abfolge von Ereignissen eingetreten ist. Zustandsbehaftete Signaturen sind weitaus mächtiger, aber auch rechenintensiver. Um die immense Aufgabe zu bewältigen, Tausende dieser komplexen Regeln in Echtzeit auf einen Datenstrom mit Gigabit-Geschwindigkeit anzuwenden, werden die Signaturen beim Start des IPS in optimierte Datenstrukturen geladen. Dies ermöglicht einen simultanen Abgleich aller Muster in einem einzigen Durchlauf.

Der entscheidende Vorteil der signaturbasierten Erkennung liegt in ihrer hohen Präzision und Zuverlässigkeit. Ein positiver Treffer ist ein starker Indikator für einen tatsächlichen, bekannten Angriff, was die Rate an False Positives extrem niedrig hält. Die Nachteile sind jedoch ebenso nicht zu vernachlässigen. Die Methode ist per Definition reaktiv; sie kann einen Angriff erst erkennen, nachdem er analysiert und eine Signatur dafür erstellt und verteilt wurde. Gegen Zero-Day-Exploits ist sie daher wirkungslos. Zudem versuchen Angreifer gezielt, Signaturen durch Evasion-Techniken zu umgehen. Mittels Polymorphismus oder Metamorphismus wird der Schadcode bei jeder Infektion automatisch so verändert, dass sein "Fingerabdruck" variiert. Durch

Verschleierung, beispielsweise durch Kodierung oder Verschlüsselung der Nutzlast, wird versucht, den bösartigen Inhalt vor der DPI-Engine zu verbergen. Die Effektivität eines signaturbasierten IPS hängt somit direkt und unabdingbar von der Qualität, der Geschwindigkeit und der Intelligenz der Signatur-Updates des jeweiligen Herstellers oder der Community ab.

Anomaliebasierte Erkennung:

Die anomaliebasierte Erkennung verfolgt einen fundamental anderen, induktiven Ansatz. Statt nach bekannten Mustern des "Bösen" zu suchen, versucht sie, ein umfassendes Modell des "Normalen" zu erstellen und jede signifikante Abweichung von diesem Zustand als potenzielle Bedrohung zu identifizieren. Dieser Ansatz hat den theoretischen Hauptvorteil, auch bisher unbekannte Zero-Day-Angriffe, neue Malware-Varianten oder gezielte, individuelle Angriffe erkennen zu können, für die keine Signaturen existieren. Die zugrundeliegende Annahme ist, dass jede bösartige Aktivität zwangsläufig das normale Verhalten eines Systems oder Netzwerks stört und messbare Spuren hinterlässt.

Der Kernprozess ist die Erstellung und Pflege einer Baseline, eines digitalen Abbilds des Normalzustands. Dieser Prozess beginnt mit einer initialen Trainings- oder Lernphase, in der das System den Netzwerkverkehr über einen längeren Zeitraum passiv beobachtet. In dieser Phase kommen je nach Komplexität des Systems, unterschiedliche Verfahren zum Einsatz, von einfachen statistischen Mittelwerten bis hin zu komplexen Machine-Learning-Modellen. Die erfassten Parameter umfassen eine Vielzahl von Metriken, wie zum Beispiel die durchschnittliche und maximale Bandbreitennutzung pro Protokoll, die Latenzzeiten zu bestimmten Servern, die Größe von Paketen und die typischen Kommunikationsbeziehungen zwischen den Hosts im Netzwerk.

Nach der Trainingsphase geht das System in den aktiven Erkennungsmodus über und vergleicht den Echtzeit-Verkehr kontinuierlich mit der erlernten Baseline. Anomalien können dabei auf verschiedenen Ebenen festgestellt werden:

- **Protokoll-Anomalien:**

Dies ist die verlässlichste Form der Anomalieerkennung. Hierbei wird der Datenverkehr von einem strikten Protokoll-Parser analysiert, der auf den offiziellen RFC-Spezifikationen basiert. Jede Abweichung von der Norm – sei es ein ungültiges Flag im TCP-Header, eine fehlerhafte Zeichenkodierung in einer DNS-Anfrage oder die Verwendung eines veralteten Befehls in einer SMB-Sitzung – wird als Anomalie gemeldet. Viele Exploits verursachen solche subtilen Protokollverstöße bei dem Versuch, eine Schwachstelle auszunutzen.

- **Statistische Anomalien:**

Hierbei werden die aktuellen Metriken mit dem statistischen Modell der Baseline verglichen. Das System berechnet für beobachtete Ereignisse einen Anomalie-Score. Ein plötzlicher Anstieg der fehlgeschlagenen Login-Versuche auf einem Server, ein drastischer Anstieg des ausgehenden Verkehrs von einer Workstation mitten in der Nacht oder die Nutzung eines seltenen Protokolls durch einen Benutzer, der dies noch nie getan hat, würde den Score in die Höhe treiben. Überschreitet dieser Score einen vordefinierten Schwellenwert (Threshold), wird ein Alarm ausgelöst.

Die größte Herausforderung und der entscheidende Nachteil dieses Ansatzes ist die notorisch hohe Rate an Falschmeldungen (False Positives), da das "normale" Verhalten eines Netzwerks nicht statisch ist. Jede legitime, aber neue und unvorhergesehene Aktivität, wie beispielsweise die Einführung eines neuen Cloud-Dienstes, eine Systemwartung oder eine Marketing-Kampagne mit hohem Traffic, kann fälschlicherweise als Anomalie eingestuft werden. Dies führt zum operativen Problem der "Alert Fatigue", bei dem Administratoren durch eine Flut an Alarmen die Übersicht verlieren und die Sensitivität des Systems herunterregulieren, wodurch dessen Effektivität sinkt. Zudem besteht die Gefahr der "Baseline Pollution". Dies definiert die Lernphase in einem bereits kompromittierten Netzwerk, wodurch das bösartige Verhalten als Teil der Norm erlernt und zukünftig nicht mehr erkannt wird.

Aus diesen Gründen wird die anomaliebasierte Erkennung in der Praxis fast immer in einem hybriden Modell als Ergänzung zur signaturbasierten Erkennung eingesetzt. Sie dient als intelligentes Frühwarnsystem für neuartige Bedrohungen.

2.3 Vorstellung der Systeme

Nachdem die vorangegangenen Kapitel die theoretischen Grundlagen der Bedrohungslage und Funktionsweise von Intrusion Prevention Systemen erläutert haben, widmet sich dieses Kapitel der Vorstellung der beiden für den praktischen Vergleich ausgewählten Systeme. Die Auswahl fiel auf eine FortiGate-Firewall als prominenten Vertreter der kommerziellen NGFW und auf OPNsense als repräsentatives Beispiel für eine flexible, weit verbreitete Open-Source-Alternative.

Diese beiden Systeme wurden gezielt gewählt, da sie in ihren jeweiligen Segmenten eine hohe Marktrelevanz besitzen und fundamental unterschiedliche Philosophien in Bezug auf Architektur, Lizenzmodell und Verwaltung verkörpern. Um eine fundierte Vergleichsbasis zu schaffen, werden die beiden Systeme zunächst in separaten Unterkapiteln detailliert vorgestellt. Dabei wird auf ihre jeweilige Architektur, die technische Implementierung der IPS-Funktionalität und das zugrundeliegende Betriebsmodell eingegangen. Abschließend fasst eine synoptische Gegenüberstellung die zentralen Unterschiede tabellarisch zusammen, um eine klare Ausgangslage für die Konzeption der Testumgebung im nachfolgenden Kapitel zu schaffen.

2.3.1 Der kommerzielle Vertreter: FortiGate

Als kommerzieller Vertreter für den Systemvergleich wird die FortiGate VM des Herstellers Fortinet untersucht. Fortinet ist ein 1999 gegründetes, US-amerikanisches Unternehmen und zählt zu den globalen Marktführern im Bereich der Cybersicherheit. Die Unternehmensphilosophie manifestiert sich in der "Fortinet Security Fabric", einer Architektur, die darauf abzielt, eine breite, integrierte und automatisierte Sicherheit über die gesamte digitale Infrastruktur eines Unternehmens zu spannen. Das Kernstück dieser Architektur und das "Flaggschiff-Produkt" des Unternehmens ist die FortiGate Next-Generation Firewall (NGFW), die auf dem Betriebssystem FortiOS basiert.

Produktphilosophie und Architektur:

Die grundlegende Philosophie hinter der FortiGate-Plattform ist die der Konsolidierung von Sicherheitsfunktionen. Die Plattform basiert auf dem Prinzip des Unified Threat Management (UTM), dessen Ziel es ist, eine Vielzahl von ehemals separaten Sicherheitslösungen in einem einzigen System zu vereinen. Eine FortiGate integriert somit weit mehr als eine klassische Firewall. Sie vereint Funktionen wie Stateful Packet Inspection, VPN-Gateway, Intrusion Prevention (IPS), Antivirus-Scanning, Web- und DNS-Filterung sowie Application Control. All diese Sicherheitsdienste werden durch ein einziges, proprietäres Betriebssystem namens FortiOS gesteuert

und verwaltet. [15]

Hardware-Architektur vs. Virtuelle Umgebung

Ein fundamentaler architektonischer Aspekt der physischen FortiGate-Systeme ist der Einsatz von spezialisierter Hardware in Form von anwendungsspezifischen integrierten Schaltungen (ASICs), die unter Fortinet als Security Processing Units (SPUs) bezeichnet werden. Diese Prozessoren beschleunigen rechenintensive Aufgaben wie die Paketverarbeitung und die Inhaltsanalyse (Hardware-Offloading).

In der für diese Arbeit verwendeten FortiGate VM entfällt diese spezialisierte Hardware. Stattdessen werden alle sicherheitsrelevanten Berechnungen von der Standard-CPU des Virtualisierungs-Hosts ausgeführt. FortiOS ist darauf optimiert, diese Aufgaben effizient auf Standard-x86-Architekturen zu verteilen, um auch in einer virtualisierten Umgebung eine hohe Performance zu gewährleisten.[16]

Implementierung des IPS:

Die IPS-Funktionalität ist in der FortiGate VM kein globaler Dienst, sondern ein tief in die Verarbeitungslogik von Firewall-Richtlinien (Policies) integrierter Mechanismus. Diese richtlinienbasierte Architektur ist der Schlüssel zur granularen und performanten Anwendung von Sicherheitsmaßnahmen. Anstatt den gesamten Netzwerkverkehr pauschal zu überprüfen, wird die Inhaltsanalyse gezielt für definierte Datenströme aktiviert.

Die Konfiguration erfolgt über Security Profiles (Sicherheitsprofile). Ein Security Profile ist ein Bündel von Einstellungen für die verschiedenen Sicherheitsdienste wie Antivirus, Web-Filter, Application Control und IPS. Dafür wird innerhalb eines Security Profiles ein sogenannter IPS Sensor konfiguriert. In diesem Sensor wird detailliert festgelegt, welche Angriffs-Signaturen und -Kategorien (z.B. Web-Client, Malware, Botnet) für den zu schützenden Verkehr relevant sind und welche Aktion bei einem Treffer ausgeführt werden soll (Block, Monitor, Reset, etc.).

Die Stärke dieses Konzepts liegt in der Möglichkeit, beliebig viele, unterschiedliche IPS Sensoren für verschiedene Anwendungsfälle zu erstellen und diese dann gezielt den jeweiligen Firewall-Regeln zuzuweisen. Beispielsweise kann für eine Regel, die den Zugriff aus dem Internet auf einen öffentlichen Webserver erlaubt, ein sehr strenger IPS Sensor mit Fokus auf Web-Attacken (SQL-Injection, Cross-Site Scripting) hinterlegt werden. Gleichzeitig kann eine andere Regel, die den Surf-Verkehr interner Mitarbeiter ins Internet regelt, einen anderen IPS Sensor verwenden, der auf

den Schutz von Client-Anwendungen (Browser, Office-Programme) spezialisiert ist. Für hochperformanten Server-zu-Server-Verkehr im internen Netz kann hingegen bewusst auf ein IPS-Profil verzichtet werden, um die Latenz zu minimieren. Die Art und Weise, wie die Deep Packet Inspection (DPI) für diese Analyse durchgeführt wird, wird ebenfalls in der Firewall-Regel selbst festgelegt. Es kann pro Richtlinie den Inspection Mode (Inspektionsmodus) ausgewählt werden:

*Der schnelle Flow-based-Modus analysiert den Verkehr ohne vollständige Pufferung und ist der Standard für die Echtzeit-Prävention von Exploits im IPS.

*Der gründlichere Proxy-based-Modus puffert ganze Objekte, bevor er sie scannt, und wird oft in Kombination mit Antivirus-Profilen für Web- oder E-Mail-Verkehr genutzt.

Durch diese Kombination aus individuell konfigurierbaren IPS Sensoren und der Wahl des Inspektionsmodus pro Firewall-Regel ermöglicht die FortiGate-Architektur eine sehr flexible, kontextbezogene und ressourcenschonende Anwendung ihrer Intrusion-Prevention-Funktionen.

Ein wesentlicher Aspekt der kommerziellen Philosophie von Fortinet ist die Absicherung der Sicherheitsplattform selbst. Bei der hier verwendeten FortiGate VM greift dabei ein Modell der geteilten Verantwortung (Shared Responsibility). Fortinet ist für die Sicherheit des proprietären und gehärteten Betriebssystems FortiOS verantwortlich und liefert regelmäßig Sicherheitsupdates, um Schwachstellen in der virtuellen Appliance zu schließen und das Risiko einer direkten Kompromittierung der Firewall-Software zu minimieren.

Die Verantwortung für die Sicherheit der darunterliegenden Infrastruktur obliegt hingegen vollständig dem Anwender. Dies schließt die Härtung des Virtualisierungs-Hosts und die sichere Konfiguration des Hypervisors mit ein. Physische Angriffe, wie beispielsweise ein Evil-Maid-Angriff, stellen nun eine Bedrohung für den Virtualisierungs-Host dar, nicht mehr für eine dedizierte Appliance mit gehärtetem Gehäuse. Die Schutzmaßnahmen verlagern sich somit vom Firewall-Produkt selbst auf den Server und dessen physischen Standort. Im Gegensatz zu einer physischen Appliance liegt die Verantwortung für die Integrität der Hardware im virtuellen Setup ausschließlich beim Betreiber der Host-Systeme.

2.3.2 Die Open-Source-Alternative: OPNsense

Die Entstehungsgeschichte von OPNsense ist eng mit der Entwicklung anderer Open-Source-Firewall-Projekte verknüpft und für das Verständnis seiner Philosophie entscheidend. Das Projekt wurde Anfang 2015 als "Fork" des damals dominanten pfSense-Projekts ins Leben gerufen. Maßgeblich vorangetrieben wurde dieser Schritt von der

niederländischen Firma Deciso B.V. Die Gründe für die Abspaltung waren konkret und zielgerichtet: Zum Einen gab es Bedenken hinsichtlich der Lizenzpolitik und den Wunsch nach einer eindeutigen, liberalen 2-Klausel-BSD-Lizenz, um maximale Freiheit für private und kommerzielle Nutzer zu garantieren. Auch die Entwicklung von pfSense wurde als zunehmend geschlossen und intransparent wahrgenommen. Als Reaktion darauf setzte OPNsense von Beginn an auf eine offene Entwicklung und die aktive Beteiligung der Community. Außerdem lagen technische Gründe vor, insbesondere der Wunsch nach einer moderneren und besser strukturierten Codebasis, um die Wartbarkeit zu erhöhen und externe Beiträge zu erleichtern. Die Philosophie von OPNsense basiert seither konsequent auf diesen Gründungsprinzipien der Transparenz, Modularität und Sicherheit, gestützt auf das robuste FreeBSD-Betriebssystem. Zusätzliche Legitimität erhielt das junge Projekt, als der Gründer des ursprünglichen m0n0wall-Projekts, von dem einst pfSense abstammte, nach dessen Einstellung seiner Community den Wechsel zu OPNsense empfahl.[14]

Produktphilosophie und Architektur:

Im direkten Gegensatz zum integrierten All-in-One-Ansatz von Fortinet verkörpert OPNsense die Kernprinzipien der Open-Source-Welt: Modularität, Transparenz und Flexibilität. Die grundlegende Philosophie besteht darin, eine stabile und sichere Kernplattform für Routing und Firewalling bereitzustellen, die vom Anwender durch eine Vielzahl von Plugins je nach Bedarf erweitert werden kann. Die Offenlegung des gesamten Quellcodes schafft ein hohes Maß an Vertrauen und ermöglicht es der globalen Sicherheits-Community, den Code auf potenzielle Schwachstellen zu überprüfen.[14]

Die technische Basis von OPNsense ist das robuste und für seine Stabilität und Sicherheit bekannte Betriebssystem FreeBSD. Zusätzlich integriert OPNsense sicherheitsrelevante Features aus dem HardenedBSD-Projekt, um die Angriffsfläche des Basissystems weiter zu reduzieren. Die Verwaltung erfolgt, ähnlich wie bei FortiGate, primär über eine webbasierte grafische Benutzeroberfläche (GUI), die die Navigation innerhalb des Systems erleichtert.

Im Gegensatz zu einer geschlossenen Appliance liegt die Verantwortung für die Absicherung des Systems, auf dem OPNsense läuft, vollständig beim Anwender. Dies schließt sowohl die physische Sicherheit der Hardware als auch die Härtung des Betriebssystems ein. Ein physischer Angriff wie der Evil MaidAngriff, bei dem ein Angreifer mit kurzzeitigem, unbeaufsichtigtem Zugriff auf die Hardware eine Kompromittierung durchführt, stellt hier ein realistisches Szenario dar. Obwohl die Basis aus FreeBSD und HardenedBSD bereits ein hohes Sicherheitsniveau bietet, muss

der Anwender durch eigene Maßnahmen wie die Verschlüsselung der Festplatte, die Absicherung des BIOS/UEFI und die regelmäßige Installation von System- und Sicherheitsupdates die Integrität der gesamten Plattform gewährleisten. Dieser Aspekt stellt einen signifikanten Unterschied in der operativen Verantwortung im Vergleich zum kommerziellen Modell dar.

Hardware-Architektur:

Ein fundamentaler architektonischer Unterschied zu FortiGate ist die vollständige Hardware-Unabhängigkeit von OPNsense. Es handelt sich um eine reine Software-Distribution, die auf nahezu jeder Standard-x86-64-Architektur installiert werden kann. Dies bietet dem Anwender eine enorme Flexibilität bei der Wahl der Hardware-Plattform. Die Einsatzmöglichkeiten reichen von kleinen, energieeffizienten Embedded-Systemen für den Heim- oder Kleinbürogebrauch über virtuelle Maschinen in Rechenzentren, bis hin zu leistungsstarken Rack-Mount-Servern mit Multi-Gigabit-Netzwerkkarten für den Unternehmenseinsatz.

Diese Flexibilität hat jedoch zur Folge, dass die gesamte Verarbeitungs- und Analyse-Last auf der Standard-CPU des gewählten Systems liegt. OPNsense nutzt keine spezialisierten ASICs oder Hardware-Beschleuniger. Die Performance des Gesamtsystems, insbesondere der Durchsatz bei aktivierter Deep Packet Inspection, ist somit direkt und untrennbar von der Leistung der gewählten CPU, der Menge und Geschwindigkeit des Arbeitsspeichers (RAM) und der Qualität der Netzwerkkarten (NICs) abhängig. Die Verantwortung für die korrekte Dimensionierung der Hardware liegt somit vollständig beim Anwender.

Implementierung des Intrusion Prevention Systems:

Die Intrusion-Prevention-Funktionalität ist in OPNsense nicht Teil des Kernsystems, sondern wird als modulares Plugin realisiert. Der Anwender kann wählen, welche IPS-Engine er installieren möchte. Als Standard hat sich hierbei Suricata etabliert, eine hochperformante, multithreading-fähige und weit verbreitete Open-Source-IPS-Engine.

- **Implementierung als Dienst:**

Nach der Installation über die Plugin-Verwaltung läuft Suricata als eigenständiger Dienst auf dem OPNsense-System. Für die Prävention wird der Dienst im „Inline-Modus“ betrieben. In diesem Modus wird der gesamte Netzwerkverkehr einer oder mehrerer ausgewählter Netzwerkschnittstellen, beispielsweise der WAN-Schnittstelle, direkt durch die Suricata-Engine geleitet. Dies ist funk-

tional äquivalent zum Flow-based-Modus einer FortiGate und ermöglicht das Blockieren von böartigem Verkehr in Echtzeit.

- Die zentrale Rolle der Regel-Sets (Threat Intelligence):

Während bei FortiGate die Threat Intelligence als fertiger Service von den FortiGuard Labs bezogen wird, liegt die Verantwortung für die Auswahl, Konfiguration und Pflege der Erkennungsregeln bei OPNsense vollständig beim Administrator des Systems. Der Nutzer muss aktiv entscheiden, welche Regel-Sets er verwenden möchte. Gängige Optionen sind:

- * ET Open (Emerging Threats Open): Ein umfassendes Regel-Set, das von der Community gepflegt und kostenlos zur Verfügung gestellt wird. Es bietet eine große Basis für den Schutz vor einer breiten Palette von Bedrohungen.

- * ET Pro (Emerging Threats Pro): Eine kostenpflichtige, abonnementbasierte Version der Emerging-Threats-Regeln. Dieses Set bietet mehr Signaturen, eine schnellere Veröffentlichung von Regeln für neue Bedrohungen und wird durch professionellen Support unterstützt.

- * Cisco Talos Rulesets: Cisco Talos stellt die offiziellen Regeln für die IPS-Engine Snort bereit. Da Suricata in weiten Teilen kompatibel zu Snort ist, sind diese Regeln eine sehr populäre und hochwertige Alternative oder Ergänzung zu den ET-Regeln.

- Konfiguration und "Tuning":

Die reine Aktivierung eines Regel-Sets reicht oft nicht aus. Ein wesentlicher Teil der Arbeit des Administrators besteht im sogenannten "Tuning". Dabei müssen Regeln, die im spezifischen Netzwerkumfeld zu Falschmeldungen (False Positives) führen, identifiziert und gezielt deaktiviert oder angepasst werden. Dies erfordert ein tiefes Verständnis des eigenen Netzwerkverkehrs und der Funktionsweise des IPS und stellt einen signifikanten personellen Mehraufwand im Vergleich zu einer kommerziellen Lösung dar.

Betriebs- und Kostenmodell:

Das Betriebsmodell von OPNsense ist durch Offenheit und Eigenverantwortung geprägt. Die Software selbst, einschließlich aller Plugins, ist kostenlos und unterliegt keinen Lizenzgebühren. Die Total Cost of Ownership (TCO) setzt sich somit anders zusammen als bei einem kommerziellen Produkt. Die Hauptkostenpunkte sind die Anschaffung der Hardware und der oft unterschätzte personelle Aufwand für die initiale Konfiguration, die kontinuierliche Wartung, das Update-Management und

das Tuning der IPS-Regeln. Wiederkehrende Kosten entstehen nur, wenn man sich für optionale kommerzielle Regel-Sets oder einen professionellen Support-Vertrag, beispielsweise direkt von Deciso, entscheidet. Der primäre Support-Kanal ist jedoch die aktive Community über Foren, und die umfangreiche offizielle Dokumentation.

2.3.3 Synoptische Gegenüberstellung

Nachdem in den vorangegangenen Kapiteln die beiden für diese Arbeit ausgewählten Systeme einzeln vorgestellt wurden, dient dieser Abschnitt dazu, die gewonnenen Erkenntnisse in einem direkten Vergleich zu verdichten.

Die nachfolgende Tabelle stellt die zentralen konzeptionellen, technischen und operativen Eigenschaften der beiden Lösungen synoptisch gegenüber.

| Kriterium | FortiGate | OPNsense |
|--|--|--|
| I. Grundlegende Architektur & Philosophie | | |
| System-Typ | Virtuelle Security-Appliance (Software für Standard-Virtualisierungsumgebungen) | Software-Distribution, die auf Standard-Hardware installiert wird |
| Grundphilosophie | Unified Threat Management (UTM): Konsolidierung vieler Sicherheitsfunktionen in einem System | Modulare Plattform: Grundfunktionen sind enthalten, erweiterte Funktionen werden als Plugins hinzugefügt |
| Entwicklungsmodell | Proprietär / Closed-Source; Entwicklung erfolgt intern durch Fortinet | Open-Source; Entwicklung wird von der Firma Deciso B.V. und einer globalen Community vorangetrieben |
| Quellcode-Verfügbarkeit | Nicht verfügbar | Vollständig offen und einsehbar |
| Zielgruppe | Unternehmen jeder Größe, von KMU bis zu Großkonzernen und Providern | Primär KMU, Heimanwender, Bastler und Organisationen mit hohem Bedarf an Anpassbarkeit |
| II. Technische Implementierung des IPS | | |
| IPS-Engine | Proprietäre, in FortiOS tief integrierte Engine | Suricata (oder optional Snort) als eigenständiger Dienst/Plugin |
| Hardware-Beschleunigung | Nein , die gesamte Verarbeitungslast liegt auf der CPU des Virtualisierungs-Hosts | Nein , die gesamte Verarbeitungslast liegt auf der Standard-x86-CPU des Systems |

| Kriterium | FortiGate | OPNsense |
|---------------------------|--|---|
| Inspektionsmodi | Flow-based und Proxy-based als wählbare Modi pro Firewall-Regel | Inline-Modus (vergleichbar mit Flow-based), für Proxy-Funktionalität ist ein separater Dienst nötig |
| Standard-Inspektionsmodus | Flow-based für hohe Geschwindigkeit und geringe Latenz | Inline-Modus |
| Protokoll-Unterstützung | Sehr breite, durch FortiGuard Labs definierte und gepflegte Protokolldekodierer | Breite Protokoll-Unterstützung durch Suricata, abhängig von den aktivierten Regeln |
| SSL/TLS-Inspektion | Als integrierte, rechenintensive Kernfunktion verfügbar; Performance ist von der Host-CPU abhängig | Möglich, erfordert aber die manuelle Konfiguration eines separaten Web-Proxys (z.B. Squid) |
| Signatur-Erstellung | Erfolgt exklusiv durch das interne Sicherheitsteam der FortiGuard Labs | Erfolgt durch verschiedene Community-Projekte (z.B. Emerging Threats) und kommerzielle Anbieter |
| Signatur-Verteilung | Automatisiert über den FortiGuard Distribution Network (FDN) | Manuell oder per Cron-Job aus den vom Administrator ausgewählten Quellen |

III. Erkennungsmethoden & Intelligenz

| | | |
|---------------------------|---|--|
| Primäre Erkennungsmethode | Signaturbasierte Erkennung | Signaturbasierte Erkennung |
| Anomalieerkennung | Ja, integrierte heuristische und anomaliebasierte Methoden, insbesondere für DoS-Schutz | Limitiert auf strikte Protokoll-Validierung gemäß den Regeln; keine statistische Baseline-Analyse out-of-the-box |
| Zero-Day-Schutz | Limitiert; durch Heuristiken und proaktive Signaturen. Hauptschutz durch schnelles Ausrollen neuer Signaturen | Extrem limitiert; theoretisch durch Protokoll-Anomalien möglich, aber nicht der Fokus |

| Kriterium | FortiGate | OPNsense |
|--|--|--|
| Regel-Quellen | Exklusiv FortiGuard Labs | Vom Nutzer frei wählbar (z.B. ET Open, ET Pro, etc.) |
| Anpassbarkeit der Regeln | Limitiert auf das Aktivieren/Deaktivieren und Ändern der Aktion von bestehenden Signaturen | Vollständig; eigene Regeln können geschrieben und bestehende Regeln angepasst werden |
| IV. Verwaltung & Betrieb | | |
| Konfigurations-Interface | Zentrale, webbasierte GUI für alle Funktionen | Zentrale, webbasierte GUI, aber IPS ist ein eigener Konfigurationsbereich |
| Konfigurations-Paradigma | Richtlinien-zentriert: IPS wird als Profil auf Firewall-Regeln angewendet | Dienst-zentriert: IPS wird global auf Netzwerkschnittstellen aktiviert |
| Reporting & Logging | Umfassend und integriert, Anbindung an FortiAnalyzer möglich | Funktional, aber für tiefgehende Analyse oft die Anbindung an externe Log-Systeme (z.B. Elastic Stack) nötig |
| Update-Prozess | Vollautomatisiert für Signaturen; Firmware-Updates sind manuelle Eingriffe | Updates für das Basissystem, Plugins und Regel-Sets erfolgen getrennt und auf Initiative des Anwenders |
| Personalaufwand | Geringerer initialer Konfigurationsaufwand, aber laufende Lizenzverwaltung | Höherer initialer Konfigurations- und laufender Wartungsaufwand (Regel-Tuning) |
| V. Kommerzielles Modell & Support | | |
| Lizenzkosten (Software) | Ja, obligatorisch für IPS-Updates und andere Sicherheitsdienste | Nein, die Software ist kostenlos |
| Hardware-Kosten | Abhängig von den Ressourcen des Virtualisierungs-Hosts; keine direkten Appliance-Kosten | Abhängig von der vom Nutzer gewählten Standard-Hardware |

| Kriterium | FortiGate | OPNsense |
|---|--|---|
| Wiederkehrende Kosten | Hoch, durch jährliche Subscriptions für FortiGuard-Dienste und Support | Niedrig; optional für kommerzielle Regel-Sets oder Support-Verträge |
| Support-Modell | Offizieller Herstellersupport mit garantierten Reaktionszeiten (SLAs) | Primär über Community-Foren; kommerzieller Support ist optional bei Dritten erhältlich |
| Herstellerhaftung | Ja, im Rahmen des Support-Vertrags besteht eine klare Verantwortlichkeit | Nein, die Verantwortung für den Betrieb liegt vollständig beim Nutzer |
| VI. Performance & Skalierbarkeit | | |
| Leistungs-Flaschenhals | CPU-Leistung, RAM und Netzwerkanbindung des Host-Systems | CPU-Leistung, RAM-Geschwindigkeit der gewählten Hardware |
| Skalierbarkeit | Flexible vertikale Skalierung durch Zuweisung von Host-Ressourcen (vCPU, RAM); horizontale Skalierung durch Clustering | Hohe Flexibilität bei der vertikalen Skalierung durch Hardware-Upgrades; Clustering ist möglich |

3 Aufbau der Testumgebung

3.1 Physische und Virtuelle Komponenten

3.1.1 Test-Hardware

Hardware für OpnSense:

Für die Open-Source-Lösung OPNsense wurde im Sinne eines praxisnahen Vergleichs zunächst versucht, eine sehr ressourcenschonende Hardware-Plattform zu verwenden. Die initiale Wahl fiel auf einen kompakten Embedded-PC vom Typ PC Engines APU4, der in der Open-Source-Community häufig für den Aufbau von Routern und Firewalls genutzt wird. Die Installation auf diesem "headless" System (ohne Grafikschnittstelle) erfolgte über das offizielle serielle Image von OPNsense. Hierfür wurde ein bootfähiger USB-Stick mit dem Programm Rufus erstellt und die gesamte Installation über eine serielle Konsole via PuTTY gesteuert.

Bereits während der initialen Konfiguration des Intrusion Prevention Systems Suricata zeigte sich jedoch, dass die Hardware mit 4 GB RAM und der schwachen CPU mit 1GHz an ihre Leistungsgrenzen stieß und auch ohne Laden der Regel-Sets zu einer CPU-Auslastung von 100% führte. Das Laden der ausgewählten Regel-Sets mit über 35.000 Signaturen führte schließlich zu Absturz der gesamten Umgebung. Dies stellt einen stabilen Testbetrieb in Frage.

Um eine valide und performante Testdurchführung zu gewährleisten und die Fähigkeiten der OPNsense-Software fair bewerten zu können, wurde daher eine Umstellung auf eine leistungsfähigere Hardware-Plattform vorgenommen. Als finales Testsystem für OPNsense dient nun ein NRG Systems Mini-PC, der mit einem Intel(R) Core(TM) i7-4510U Prozessor, 8 GB DDR3L RAM und vier Gigabit-LAN-Anschlüssen ausgestattet ist. Diese Hardware-Aufwertung unterstreicht einen fundamentalen Aspekt der Open-Source-Philosophie: die Flexibilität, die Hardware an die wachsenden Anforderungen anzupassen, aber auch die Notwendigkeit einer sorgfältigen initialen Dimensionierung. Die Installation auf diesem System erfolgte unkompliziert über das Standard-ISO-Abbild.

Hardware für Virtualisierung:

Als Virtualisierungs-Host, der die zu schützenden Client-Systeme im LAN-Segment

beherbergt, dient ein kompakter Mini-PC vom Typ HP EliteDesk 800 G4 DM 35W. Das System ist mit einer Intel(R) Core(TM) i5-8500T CPU, mit einem 16 GB DDR4-Arbeitsspeicher ausgestattet. Als primärer Datenspeicher für die virtuellen Maschinen wurde eine 256 GB NVMe SSD gewählt.(siehe Abbildung 9.3)

Die Entscheidung für eine SSD anstelle einer mechanischen Festplatte (HDD) war für die Performance der Testumgebung entscheidend. In einer Virtualisierungsumgebung greifen mehrere VMs gleichzeitig auf den Speicher zu, was zu einer hohen Anzahl paralleler Lese- und Schreibvorgänge führt. Während eine HDD diese Anfragen aufgrund ihres mechanischen Aufbaus mit einem beweglichen Schreib-/Lesekopf nur sequenziell, also Schritt für Schritt, abarbeiten kann, ist eine SSD in der Lage, Tausende dieser Anfragen parallel zu bedienen. Dieser Vorteil verhindert einen Input/Output-Flaschenhals, der für die nachfolgenden Tests zu ungenauen Ergebnissen führen könnte.

Angreifer-Systems (Kali Linux):

Als Angreifer-System, das im ungesicherten WAN-Segment der Testumgebung agiert, wurde ein dedizierter Laptop vom Typ DELL Latitude 5520 verwendet. Das Gerät ist mit einem Intel(R) Core(TM) i7 Prozessor, 16 GB RAM und einer 500 GB SSD ausgestattet. Diese Hardware-Ressourcen wurden als mehr als ausreichend bewertet, um die für die Tests notwendigen Analyse- und Angriffswerkzeuge performant auszuführen.

Die Installation des Betriebssystems erfolgte mittels der offiziellen Installer ISO-Datei von Kali Linux (Version 2025.2). Um den Laptop von diesem Abbild zu starten, wurde mit dem Programm "Rufus" ein bootfähiger USB-Stick erstellt. Während des Installationsprozesses wurde die gesamte 500 GB SSD für Kali Linux partitioniert und ein Standardbenutzer für die Durchführung der Tests angelegt.

Ein entscheidender Schritt war die Netzwerkkonfiguration. Da sich der Angreifer-Laptop in einem isolierten, physischen Netzwerksegment (WAN_KALI) befindet, das nur mit dem WAN-Port der zu testenden Firewall verbunden ist, musste eine statische IP-Konfiguration vorgenommen werden. Dies wurde über die Kommandozeilen-Schnittstelle *nmcli* realisiert. Dem System wurde die IP-Adresse 192.168.107.40 /24 zugewiesen. Als Gateway wurde die IP-Adresse des WAN_KALI-Interfaces der Firewall (192.168.107.1) eingetragen, um eine korrekte Routing-Beziehung herzustellen.

Nach der erfolgreichen Grundinstallation wurde das System zunächst auf den neuesten Stand gebracht. Hierfür wurden über das Terminal die Befehle *sudo apt update* und *sudo apt full-upgrade -y* ausgeführt. Um sicherzustellen, dass alle für die ge-

planten Tests notwendigen Werkzeuge zur Verfügung stehen, wurde anschließend das Metapackage `kali-linux-default` installiert. Dieser Befehl (`sudo apt install -y kali-linux-default`) rüstet das System mit der Standard-Sammlung an Penetration-Testing-Tools aus, die unter anderem das Metasploit Framework, den Portscanner Nmap sowie den Paketgenerator hping3 umfasst.

3.1.2 Virtualisierungssoftware

Als Fundament für die Virtualisierung der Client-Systeme, die das LAN-Segment der Testumgebung abbilden, wurde die Open-Source-Plattform Proxmox Virtual Environment (VE) in der Version 8.2 gewählt. Die Entscheidung für einen dedizierten Bare-Metal-Hypervisor anstelle einer Desktop-Virtualisierungslösung wie Virtual-Box wurde getroffen, um eine höhere Performance durch den direkten, ressourcenschonenden Zugriff auf die Hardware des Test-Laptops zu gewährleisten. Proxmox VE bot zudem die notwendige, granulare Kontrolle über die virtuelle Vernetzung und eine integrierte Snapshot-Funktionalität. Dieses Feature war für die Durchführung der Tests essenziell, da sie es ermöglicht, die Testsysteme vor jedem Testdurchlauf in einen definierten, sauberen Ausgangszustand zurückzusetzen und so für reproduzierbare Ergebnisse zu sorgen.

Die Installation erfolgte über das offizielle ISO-Abbild direkt auf dem dafür vorgesehenen Test-Laptop, der ausschließlich als Host für die Client-Systeme dient. Hierfür wurde zunächst ein bootfähiger USB-Stick erstellt. Der Installationsprozess selbst gestaltet sich unkompliziert, erforderte jedoch bei der Netzwerkkonfiguration einzelne manuelle Anpassungen. Da der Laptop als Server im LAN-Segment hinter der physischen Firewall agieren soll, muss eine statische IP-Adresse, Subnetzmaske, Gateway (die LAN-IP-Adresse der Firewall), sowie der DNS-Server manuell in der Installationsroutine hinterlegt werden.

Aufgetretenes Problem:

Bei der Installation traten initial Konnektivitätsprobleme auf. Anschließend war die webbasierte Verwaltungsoberfläche von einem anderen Rechner im Netzwerk nicht erreichbar, obwohl die Netzwerkinformationen im Installer korrekt eingegeben wurden. Eine Analyse über die Kommandozeile des Servers zeigte, dass die Netzwerkkonfigurationsdatei `"/etc/network/interfaces"` nicht korrekt geschrieben wurde. Dieses Problem konnte durch eine manuelle Bearbeitung der Datei direkt auf der Server-Konsole und einen anschließenden Neustart des Netzwerk-Dienstes mittels `"systemctl restart networking.service"` behoben werden. Erst danach war der Proxmox-Host unter der korrekten statischen IP-Adresse erreichbar und die weitere Konfiguration konnte über die Weboberfläche erfolgen.

Ein zentrales Konzept für den Aufbau der Testumgebung in Proxmox ist dabei die Verwendung von virtuellen Bridges. Für den in dieser Arbeit realisierten Aufbau wurde eine einzige virtuelle Bridge (*vmbr0*) konfiguriert, die direkt an die physische Netzwerkkarte des Proxmox-Laptops gekoppelt ist. Das physische LAN-Kabel verbindet diese Netzwerkkarte mit dem LAN-Port der zu testenden, physischen Firewall. Alle in Proxmox erstellten Client-VMs werden dann mit ihren virtuellen Netzwerkkarten an diese Bridge angeschlossen, wodurch sie zusammen ein virtuelles LAN-Segment bilden. Dieses Segment ist physisch direkt und ausschließlich mit einem geschützten LAN-Port der Firewall verbunden.

3.1.3 Virtuelle Maschinen (VMs)

Die Client-Systeme, die das geschützte LAN-Segment bilden, sowie ein interner Angreifer wurden vollständig innerhalb der Proxmox Virtual Environment virtualisiert. Der grundlegende Prozess zur Erstellung jeder VM umfasst das Hochladen der entsprechenden ISO-Abbilder in den lokalen Speicher des Proxmox-Hosts. Anschließend wurden die virtuellen Maschinen über den Installationsassistenten mit den für ihre jeweilige Rolle und ihr Betriebssystem passenden Hardwareressourcen konfiguriert.

Ubuntu Desktop:

Um repräsentative Linux-Clients im geschützten LAN abzubilden, wurden zwei identische VMs mit Ubuntu 24.04 LTS Desktop erstellt. Diese dienen als Ziele für Angriffe und zur Durchführung von Tests wie der Malware-Erkennung oder der Überprüfung von Filterregeln. Beiden VMs wurden jeweils 4 CPU-Kerne, 4 GiB Arbeitsspeicher und eine 50 GiB Festplatte zugewiesen. Um sie im geschützten LAN-Segment zu platzieren, wurden beide VMs mit ihren virtuellen Netzwerkkarten an die interne Bridge *vmbr1* angebunden. (siehe Abbildung 9.4)

Kali-Linux:

Zur Simulation eines internen Angriffs wurde eine VM mit Kali Linux aufgesetzt, die ein kompromittiertes Endgerät im LAN repräsentiert. Der VM wurden 4 CPU-Kerne, 4 GiB Arbeitsspeicher und eine 50 GiB Festplatte zugewiesen. Entscheidend für diesen Testfall wurde die Netzwerkkarte der VM, identisch zu den Client-Systemen, an die interne Bridge *vmbr1* angebunden. (siehe Abbildung 9.5)

Dieser Aufbau eines flachen Netzwerks dient der praktischen Demonstration der Grenzen des reinen Perimeterschutzes. Wie erwartet, kann die physische Firewall den direkten Verkehr zwischen den VMs auf derselben Bridge nicht einsehen oder blockieren. Ein erfolgreicher interner Exploit demonstriert somit die kritische Schutzlücke, die bei alleiniger Konzentration auf den Perimeterschutz entsteht und unterstreicht

die Notwendigkeit weiterführender Sicherheitskonzepte wie Zero-Trust und interner Netzwerksegmentierung.

Windows 11:

Zur Simulation einer typischen Workstation in einem Unternehmensumfeld wurde eine VM mit Microsoft Windows 11 Enterprise aufgesetzt. Entsprechend den höheren Anforderungen des Betriebssystems wurden dieser VM 4 CPU-Kerne, 8 GiB Arbeitsspeicher und eine 70 GiB Festplatte zugewiesen. Auch diese VM wurde an die Bridge *vmb1* angebunden, um sie im LAN zu positionieren. (siehe Abbildung 9.6)

Besonderheit bei der Windows-Installation:

Aufgrund der spezifischen Anforderungen von Windows 11 mussten in den Systeminstellungen der VM das BIOS auf OVMF (UEFI) umgestellt und ein virtuelles Trusted Platform Module (TPM) 2.0 sowie eine EFI-Disk hinzugefügt werden. Im Gegensatz zu den Linux-basierten Systemen trat bei der Installation zudem ein Problem auf, da der Windows-Installer standardmäßig keine Treiber für die paravirtualisierte VirtIO-Hardware enthält. Dadurch konnte im Partitionierungs-Schritt der Installation keine Festplatte erkannt werden. Um dieses Problem zu lösen, musste die *virtio-win.iso*-Datei mit den Gast-Treibern heruntergeladen und der VM als zweites, virtuelles CD/DVD-Laufwerk zugewiesen werden. Im Windows-Setup konnten dann die Speichertreiber aus dem Verzeichnis `\viosstor\w11\amd` manuell geladen werden, woraufhin die virtuelle Festplatte erkannt wurde und die Installation fortgesetzt und abgeschlossen werden konnte.

3.2 Netzwerkarchitektur

3.2.1 Netzwerkarchitektur Testumgebung OPNsense

Die Netzwerktopologie gliedert sich in drei zentrale Bereiche. Über Port 4 der Firewall besteht die Anbindung an das Heimnetzwerk. Dieses Interface ist als WAN-Schnittstelle konfiguriert und bezieht seine IP-Adresse per DHCP, wodurch eine direkte Verbindung ins Internet gewährleistet ist. An Port 1 wurde der Mini-PC angeschlossen, auf dem Proxmox VE betrieben wird. Dieser dient als Hostsystem für die internen Client-VMs (Windows und Ubuntu). Die OPNsense agiert in diesem Segment als DHCP-Server und vergibt Adressen im Subnetz 192.168.101.0/24. In diesem Netz besitzt die Firewall selbst die Adresse 192.168.101.1, während der Proxmox-Host unter 192.168.101.10:8006 erreichbar ist.

Ein weiteres Teilnetz wurde über Port 2 realisiert und als Angreifernetz konfiguriert. Hier befindet sich ein physisch installierter Laptop mit Kali Linux, der über die Adresse 192.168.107.20 in das Subnetz 192.168.107.0/24 eingebunden ist. Die OPNsense übernimmt in diesem Netz die Gateway-Funktion und ist unter der Adresse

192.168.107.1 erreichbar. Dieses Segment dient ausschließlich der Durchführung von Angriffssimulationen auf die im Clientnetz befindlichen Systeme.

Durch diese Strukturierung konnte eine realistische Testumgebung geschaffen werden, die produktiven Netzwerkverkehr (Clientnetz - WAN) ebenso wie gezielte Angriffe (Kali - Clientnetz) abbildet. Die OPNsense übernimmt dabei die zentrale Sicherheitsfunktion und steht als einziger Inspektionspunkt in Mitten der Testumgebung.

3.2.2 Netzwerkarchitektur Testumgebung FortiGate VM

Diese Testumgebung wurde vollständig virtualisiert in Proxmox VE aufgebaut. Im Zentrum steht eine FortiGate-VM, die als virtuelle Firewall und IPS-Komponente zwischen den verschiedenen logischen Netzsegmenten vermittelt. Alle beteiligten Systeme, sowohl die Firewall als auch die angeschlossenen Test-Clients, laufen in diesem Fall als virtuelle Maschinen auf demselben Proxmox-Host.

Die FortiGate-VM ist mit drei virtuellen Netzwerkschnittstellen ausgestattet, die jeweils unterschiedlichen in Proxmox definierten Bridges zugeordnet sind. Das WAN-Interface (port1) ist mit der Bridge vmbr0 verbunden und erhält per DHCP eine Adresse aus dem Heimnetzwerk, wodurch die Internetanbindung gewährleistet wird. Das LAN-Interface (port2) ist an die Bridge vmbr1 gekoppelt und bildet das interne Clientnetz mit der Adresse 192.168.10.1/24. In diesem Segment befinden sich die Client-VMs (Windows und Ubuntu), die über den auf der FortiGate eingerichteten DHCP-Dienst automatisch mit Adressen versorgt werden. Das dritte Interface (port3) wurde der Bridge vmbr2 zugewiesen und als separates Kali-Testnetz (192.168.20.1/24) konfiguriert, in dem eine Kali-Linux-VM für die Durchführung gezielter Angriffe betrieben wird.

3.3 Konfiguration der Firewalls und IPS-Systeme

3.3.1 Konfiguration der FortiGate VM

Die Bereitstellung der FortiGate-VM erfolgte in mehreren Schritten, die sowohl die Einrichtung auf dem ProxmoxVE Server als auch die anschließende Konfiguration innerhalb des FortiGate-Betriebssystems umfasst. Ziel war die Integration der virtuellen Firewall in die zuvor entworfene Testumgebung, bestehend aus einem LAN-Segment für Clients, einem separaten Kali-Linux-Segment zur Durchführung von Angriffssimulationen sowie einer WAN-Anbindung.

Für die Installation der FortiGate-VM wurde ein von Fortinet bereitgestelltes Systemabbild (fortios.qcow2) genutzt. Da dieses Abbild zunächst auf den Hypervisor übertragen werden musste, erfolgte der Zugriff auf den Proxmox-Host über PuTTY.

Für die eigentliche Übertragung der Datei wurde das in PuTTY integrierte Tool `pscp` (PuTTY Secure Copy) eingesetzt. Der Kopiervorgang wurde über den Befehl `pscp fortios.qcow2 root@192.168.100.14:/var/lib/vz/images/` durchgeführt, wobei das FortiOS-Abbild vom lokalen Rechner in das von Proxmox verwendete Standardverzeichnis für virtuelle Festplatten (`/var/lib/vz/images/`) übertragen wurde.

Im Anschluss daran wurde über die Weboberfläche von ProxmoxVE eine neue VM erstellt. Während der Konfiguration wurde darauf verzichtet, eine leere Standardfestplatte anzulegen. Stattdessen wurde das zuvor hochgeladene QCOW2-Abbild direkt als Systemfestplatte eingebunden. Auf diese Weise startet die VM unmittelbar mit dem originalen FortiOS-System und eine separate Installation ist nicht mehr nötig.

Nach der erfolgreichen Einbindung des Systemabbilds wurde die virtuelle Maschine über die Proxmox-Weboberfläche erstellt und konfiguriert. Dabei wurde der Maschinentyp auf `q35` gesetzt, da dieser gegenüber älteren Varianten eine verbesserte Hardwarekompatibilität sowie höhere Performance bietet. Als Festplatten-Controller wurde `VirtIO-SCSI` gewählt, um eine optimale Leistung zu erzielen.[17]

Ein zentrales Element der Einrichtung war die Konfiguration der Netzwerkschnittstellen. Dazu wurden in Proxmox drei virtuelle Bridges erstellt, die jeweils einem logischen Teilnetz der Testumgebung entsprechen:

- `vmbr0`: Anbindung an das Heimnetzwerk und damit an das WAN
- `vmbr1`: internes LAN für die Client-Systeme
- `vmbr2`: separates Testnetz für die Kali-Linux-VM

Diese Bridges wurden anschließend jeweils den virtuellen Netzwerkkarten der FortiGate-VM zugeordnet. Damit war eine klare Trennung der Netzsegmente gewährleistet, die die spätere Abbildung realistischer Sicherheitsmechanismen erlaubt.(siehe Abbildung 9.7)

Nach Abschluss der Konfiguration konnte die VM gestartet werden. Der erste Zugriff erfolgte über die in Proxmox integrierte serielle Konsole, welche unmittelbar die Kommandozeile von FortiOS bereitstellt. Zunächst wurde das vom System geforderte Standardpasswort geändert, um die Basissicherheit der VM zu gewährleisten. Anschließend wurden die einzelnen Interfaces aktiviert und mit IP-Adressen versehen. Dabei wurde `port1` als WAN-Schnittstelle definiert und über DHCP an das Heimnetz angebunden. Für das interne LAN (`port2`) wurde die statische Adresse `192.168.10.1/24` vergeben, während das Kali-Testnetz (`port3`) die statische Adresse `192.168.20.1/24` erhielt.(siehe Abbildung 9.8)

Des Weiteren wurde auf den internen Schnittstellen ein DHCP-Server eingerichtet, sodass sowohl die Client-Systeme im LAN als auch die Kali-VM automatisch

mit IP-Adressen versorgt werden konnten. Für die Verwaltung der FortiGate-VM wurden auf dem LAN-Interface zusätzlich die Protokolle HTTPS, SSH und ICMP freigeschaltet, wodurch sowohl die spätere grafische Konfiguration über die Weboberfläche als auch eine kontinuierliche Erreichbarkeit zur Systemdiagnose sichergestellt wurde.(siehe Abbildung 9.8)

Für die Lizenzierung war zunächst eine Registrierung im Fortinet-Support-Portal erforderlich, da die Lizenzdatei dort für die spezifische VM-Instanz bereitgestellt wird.

Die heruntergeladene Lizenzdatei (.lic) wurde anschließend über die Weboberfläche der FortiGate-VM hochgeladen. Für die vollständige Aktivierung und Nutzung der sicherheitsrelevanten Dienste (IPS, Antivirus-Scanning, Webfiltering und Application Control) ist zwingend eine Kommunikation mit den Fortinet-Update-Servern erforderlich. Erst über diese Verbindung wird die Gültigkeit der Lizenz überprüft und es werden aktuelle Signaturdatenbanken heruntergeladen.

Da im Rahmen der Grundkonfiguration das WAN-Interface (port1) bereits per DHCP mit dem Heimnetz verbunden war und somit Zugriff auf das Internet besitzt, konnte die FortiGate-VM nach dem Upload der Lizenzdatei unmittelbar eine Verbindung zu den FortiGuard-Servern herstellen. Dadurch wurde die Lizenz automatisch aktiviert und die aktuellen Sicherheitsupdates heruntergeladen, was die VM in den vollen Funktionsumfang versetzte.

Abbildung 9.9 zeigt die definierte Default-Route, die sämtlichen ausgehenden Verkehr aus dem LAN über das WAN-Interface leitet. Zusätzlich wurde auf der FortiGate die DNS-Weiterleitung konfiguriert. Dadurch erhielten die Clients über DHCP automatisch die interne IP-Adresse der FortiGate als DNS-Server.

Im nächsten Schritt erfolgte die Erstellung zentraler Firewall-Policies. Um in den zu erstellenden Regeln die passenden Adressbereiche zu wählen, wurde für die LAN-Subnetz und KALI-Subnetz entsprechend adressen erstellt.(siehe Abbildung 9.10, 9.11) Zunächst wurde eine Policy "LAN_to_WAN" eingerichtet, die es dem LAN ermöglicht, ins Internet zu kommunizieren. Hierbei ist es besonders wichtig das NAT zu aktivieren.(siehe Abbildung 9.12) Auch Sicherheitsprofile wie Antivirus, Webfilter, IPS und SSL Inspection wurden aktiviert. Dabei wurden die default Profile der FortiGate ausgewählt. Im späteren Verlauf wird noch ein eigenes IPS-Profil und SSL-Inspection definiert und in die erstellten Policies übernommen.

Anschließend wurde die Regel erstellt, die den Verkehr vom Kali-Netz zum LAN zulässt. Diese Policy diente dazu, gezielt Angriffe aus der Kali-VM gegen die im LAN befindlichen Testsysteme zu initiieren und deren Abwehrmechanismen durch die FortiGate-VM zu überwachen.(siehe Abbildung 9.13) Analog dazu wurde eine

weitere Policy "KALI.to_WAN" erstellt, die es der Kali-VM erlaubt, auf das Internet zuzugreifen, um notwendige Tools für die Tests zu installieren.

Zur Verbesserung der Angriffserkennung wurde ein eigenes IPS-Profil erstellt. Dieses wurde nicht als Standardprofil übernommen, sondern individuell konfiguriert, um die Testumgebung gezielt abzusichern. Innerhalb des IPS-Profiles wurden Filter gesetzt, die sowohl Server- als auch Client-Signaturen mit hoher und kritischer Schwere (Severity) umfassten. Die Aktion wurde auf "Block" gesetzt, wodurch erkannte Angriffe unmittelbar unterbunden werden. (siehe Abbildung 9.14)

Darüber hinaus wurde zur Analyse verschlüsselter Verbindungen ein eigenes SSL-Inspection-Profil definiert. Im Rahmen dieses Profils kam die Option "Full SSL Inspection" zum Einsatz, sodass sämtliche verschlüsselte Verbindungen auf Anwendungsebene entschlüsselt und überprüft werden können. Als Zertifizierungsstelle wurde das von Fortinet bereitgestellte Zertifikat Fortinet_CA_SSL hinterlegt, welches auf allen Test-Clients installiert wurde, um Zertifikatswarnungen beim Zugriff auf HTTPS-Dienste zu vermeiden. Zusätzlich wurden ungültige, abgelaufene oder widerrufen Zertifikate standardmäßig blockiert, um potenzielle Risiken durch kompromittierte Verbindungen auszuschließen. Die Inspektion erstreckt sich auf alle sicherheitsrelevanten Protokolle (HTTPS, SMTPS, POP3S, IMAPS, FTPS, DNS over TLS), sodass neben Webverkehr auch E-Mail- und Dateitransfers überwacht werden können. Um die Nutzbarkeit der Testumgebung sicherzustellen und den Praxisbezug beizubehalten, wurden bestimmte sensible Kategorien, wie "Finance and Banking" oder "Personal Privacy" von der Entschlüsselung ausgenommen, da eine Aufschlüsselung dieser Inhalte sowohl aus rechtlicher Perspektive als auch in Hinblick auf den Datenschutz problematisch wäre. (siehe Abbildung 9.15)

Nachdem nun diese selbst-definierten Profile angelegt wurden, wurden sie den entsprechenden Policies zugewiesen. (siehe Abbildung 9.16)

3.3.2 Konfiguration von OPNsense

Die Konfiguration der OPNsense-Firewall wurde in zwei logischen Phasen durchgeführt. Zunächst wurde eine initiale Einrichtungsphase durchlaufen, in der dem System temporär ein Zugang zum Internet gewährt wurde. Dies war notwendig, um das Basissystem auf den neuesten Stand zu bringen, das für die Tests erforderliche Intrusion-Prevention-System-Plugin zu installieren und die benötigten Regel-Sets herunterzuladen. Nach Abschluss dieser Vorbereitungen wurde das System in die finale, isolierte Testkonfiguration überführt, in der es ausschließlich als Gateway zwischen dem Angreifer-Netz (WAN) und dem Client-Netz (LAN) fungiert.

Konfiguration des IPS (Suricata):

Die zentrale Schutzkomponente, das IPS, wurde durch die Installation des Plugins os-suricata über die Firmware-Verwaltung von OPNsense implementiert. Die anschließende Konfiguration des Dienstes erfolgte in mehreren Schritten. Zuerst wurden im Reiter "Download" die Regel-Sets ausgewählt und heruntergeladen.

Für die Konfiguration des Intrusion Prevention Systems wurde eine gezielte Auswahl an Regel-Sets getroffen, anstatt alle verfügbaren Regeln zu aktivieren. Dieser Ansatz verfolgt das Ziel, eine hohe Erkennungsrate für die relevanten Bedrohungsszenarien zu gewährleisten und gleichzeitig die Systemlast sowie das Risiko von Falschmeldungen (False-Positives) zu minimieren. Die Auswahl basiert hierbei auf einer mehrschichtigen Strategie.

Als Grundschutz dienen ausgewählte Kategorien des weit verbreiteten ËT Open-Regel-Sets. Dabei wurden die Kategorien aktiviert, die direkt die in dieser Arbeit untersuchten Bedrohungen abdecken: Erkennung von Software-Ausnutzungen, Schadsoftware-Kommunikation, Denial-of-Service-Angriffen und Netzwerk-Scans. Ergänzt wurde diese Auswahl um die Kategorien "attack_response" und "compromised", die Signaturen für bereits erfolgte Kompromittierungen enthalten.

Dieser breite, signaturbasierte Schutz wird durch spezialisierte, reputationsbasierte Bedrohungslisten des Non-Profit-Projekts abuse.ch ergänzt. Konkret wurden die "SSL Fingerprint Blacklist" und der "ThreatFox"-Feed aktiviert. Diese Listen enthalten keine generischen Angriffsmuster, sondern blockieren aktiv die Kommunikation mit in Echtzeit identifizierten, bösartigen Command-and-Control-Servern.

Die Konfiguration der OPNsense-Firewall wurde in zwei logischen Phasen durchgeführt. Zunächst wurde eine initiale Einrichtungsphase durchlaufen, in der dem System temporär ein Zugang zum Internet gewährt wurde. Dies war notwendig, um das Basissystem auf den neuesten Stand zu bringen, das für die Tests erforderliche Intrusion-Prevention-System-Plugin zu installieren und die benötigten Regel-Sets herunterzuladen. Nach Abschluss dieser Vorbereitungen wurde das System in die finale, isolierte Testkonfiguration überführt.

Die Konfiguration der Netzwerkschnittstellen erfolgte gemäß der in Kapitel 3.2 definierten logischen Netzwerkarchitektur. Der LAN-Schnittstelle wurde ihre Rolle als Gateway für das interne Netz zugewiesen und unter Services \hookrightarrow DHCPv4 \hookrightarrow [LAN] mit einem DHCP-Server zur automatischen Adressvergabe an die Client-VMs ausgestattet. Die WAN_KALI-Schnittstelle wurde als statischer Endpunkt für das Angreifer-Netz konfiguriert, wobei die systemeigenen Filter "Block private networks" und "Block bogon networks" für diese Schnittstelle bewusst deaktiviert wurden, da der angeschlossene Angreifer eine private IP-Adresse verwendet.

Konfiguration des IPS (Suricata):

Die zentrale Schutzkomponente, das IPS, wurde durch die Installation des Plugins `os-suricata` implementiert. Nach der Auswahl und dem Download der Regel-Sets "ET Open" und "abuse.ch" wurde der Dienst im Reiter "Settings" global aktiviert, in den "Inline Mode" (IPS) versetzt und angewiesen, den Verkehr auf der WAN_KALI-Schnittstelle zu überwachen. Um die Signaturen scharfzuschalten, wurde im Reiter "Policy" eine übergeordnete Richtlinie erstellt, die alle heruntergeladenen Regel-Sets anwies, bei einem Treffer die Standard-Aktion `drop` (aktiv blockieren) auszuführen. (siehe Abbildung 9.17)

Zur Vollständigkeit sei erwähnt, dass OPNsense neben der Verwaltung über die grafische Benutzeroberfläche auch erfahrenen Anwendern die Möglichkeit bietet, die IPS-Richtlinien direkt über die Kommandozeile zu verwalten. In den Konfigurationsdateien von Suricata können benutzerdefinierte Richtlinien geschrieben werden, um beispielsweise sehr granulare Ausnahmen zu definieren oder das Verhalten für einzelne Signaturen gezielt zu modifizieren. Für die im Rahmen dieser Arbeit durchgeführten Tests wurde auf diese Methode verzichtet, da der Fokus auf der standardmäßigen und für die meisten Anwender relevanten Konfiguration über die Weboberfläche lag.

Zuletzt wurden die Firewall-Regeln unter Firewall `↳ Rules` erstellt. Da OPNsense einer "Default Deny"-Strategie folgt, bei der jeglicher Verkehr, der nicht explizit erlaubt ist, blockiert wird, mussten spezifische Regeln erstellt werden, um den für die Tests notwendigen Datenfluss zu ermöglichen.

Für den eingehenden Testverkehr wurde auf dem WAN_KALI-Interface eine Regel angelegt, die den gezielten Angriff vom Kali-Laptop auf die zu testenden Dienste der Client-VMs erlaubt. Der Zweck dieser Regel ist es, den Angriffsverkehr kontrolliert an die IPS-Engine weiterzuleiten, die auf diesem Interface den Verkehr analysiert.

Für den ausgehenden Verkehr vom LAN ins WAN wurde aus Gründen der Vereinfachung im Testlabor eine "Pass-All"-Regel erstellt, die dem gesamten LAN net den Zugriff auf jedes beliebige Ziel erlaubt.

Hinweis: Es ist wichtig zu betonen, dass dies keine "Best-Practice"-Konfiguration für eine produktive Umgebung darstellt. In einem realen Unternehmensnetzwerk würde an dieser Stelle ein wesentlich restriktiveres Regelwerk nach dem Prinzip der geringsten Rechte zum Einsatz kommen. Dabei würden typischerweise Adress-Objekte oder Gruppen (z.B. "Buchhaltung", "Entwicklung", "Administration", "Drucker") erstellt werden. Anschließend würden für jede dieser Gruppen granulare Regeln definiert, die nur den Zugriff auf die für ihre Arbeit absolut notwendigen Dienste

und Ports erlauben (z.B. nur HTTP und HTTPS für Standard-Nutzer). Alle anderen ausgehenden Verbindungen würden durch die implizite "Deny-All"-Regel am Ende des Regelwerks blockiert. Für die klar definierte Testumgebung dieser Arbeit wurde auf diese granulare Aufteilung verzichtet, um den Fokus auf die Inbound-IPS-Funktionalität zu legen.

Damit das IPS auch in die Pakete von HTTPS Traffic schauen kann, benötigt man SSL-Inspection. Dafür kann man direkt auf der OPNsense das Plugin Squid herunterladen. Dies ist ein Service, der einen Web Proxy bereitstellt. Damit nun der Verkehr von den Clients nicht mehr direkt in das Wan gelangt, sondern an den Proxy geleitet wird, muss man entsprechende Network-Address-Translation (NAT)-Regeln erstellen. Der Squid Web Proxy hört per Default auf den Port 3128/3129. (siehe Abbildung 9.19) Damit wurden 2 NAT-Regeln erstellt, die den Verkehr über Port 80 und 443 auf Port 3128 und 3129 leiten. (siehe Abbildung ??)

Außerdem muss man ein Zertifikat erstellen, den Proxy dieses Zertifikat als Autorität zuweisen und das Zertifikat auf allen Clients installieren. Anschließend wurde durch Testen bestätigt, dass die Clients das installierte Zertifikat nutzen. Um jedoch auch aussagekräftig sagen zu können, ob der Verkehr nun über den Proxy läuft, musste man in den Log-Datei des Plugins Squid schauen. Dafür kann man sich in der Konsole auf der OPNsense anmelden und mit dem Befehl `tail -f /var/log/squid/access.log` live die letzten Zeilen der log-Datei sehen. (siehe Abbildung 9.20) Dies bestätigte, dass der gesamte Verkehr über den Web Proxy geht.

4 Testing

4.1 Definition der Testfälle

Um die Schutzwirkung und die Erkennungsfähigkeiten der beiden IPS-Lösungen – FortiGate VM und OPNsense – systematisch zu vergleichen, werden praxisnahe Testfälle aus verschiedenen Angriffskategorien definiert. Diese Kategorien spiegeln die in den theoretischen Grundlagen (Kapitel 2.1) beschriebene Bedrohungslage wider und zielen darauf ab, unterschiedliche Erkennungsmechanismen der Systeme zu prüfen und so ein breites Testbild aufzustellen.

4.1.1 Netzwerk-Scans (Reconnaissance)

In dieser Kategorie wird die Fähigkeit der IPS-Systeme getestet, verschiedene Methoden der Netzwerkaufklärung zu erkennen und zu blockieren. Solche Scans stellen oft den ersten Schritt eines gezielten Angriffs dar, weshalb ihre Erkennung fundamental für eine proaktive Verteidigung ist. Als Werkzeug kommt hierbei der Portscanner Nmap zum Einsatz, dessen verschiedene Modi gezielt unterschiedliche Erkennungsmechanismen der IPS-Systeme herausfordern sollen.

- **TCP-SYN-Scan (nmap -sS)**

Der Parameter -sS steht für "SYN-Scan" / "Stealth-Scan". Anstatt einen vollständigen TCP-Handshake (SYN → SYN/ACK → ACK) durchzuführen, sendet Nmap nur das initiale SYN-Paket. Antwortet der Zielport mit einem SYN/ACK, gilt er als offen. Antwortet er jedoch mit einem RST, ist der Port geschlossen. Der Angreifer bricht den Handshake nach dem SYN/ACK durch Senden eines RST-Pakets ab und hinterlässt so weniger Spuren in den Anwendungslogs des Zielsystems.

Dieser Scan wurde ausgewählt, da er die Standardmethode für schnelle und effiziente Portscans ist. Das Ziel ist es zu prüfen, ob das IPS diese weitverbreitete Technik anhand der hohen Anzahl von Verbindungsversuchen von einer Quelle zu vielen verschiedenen Ports erkennt und als bösartige Aufklärungsaktivität einstuft

- **Xmas-Scan (nmap -sX)**

Der Parameter -sX führt einen sogenannten "Xmas-Scan" durch. Dabei werden

TCP-Pakete versendet, bei denen die Flags FIN, PSH und URG gleichzeitig gesetzt sind. Der Name entstand wegen der unüblichen Flag-Kombination, die bei Überwachung in beispielsweise Wireshark oder tcpdump an eine bunte Weihnachtsbaumbeleuchtung erinnert. Laut RFC 793 sollten geschlossene Ports auf ein solches Paket mit einem RST-Paket antworten, während offene Ports es ignorieren.[19] Da diese Pakete gegen die TCP-Protokollspezifikation verstoßen, sollten sie von einer fortschrittlichen DPI-Engine als abnormal oder bösartig erkannt werden, selbst wenn keine spezifische Angriffs-Signatur existiert.

- **Fragmentierter Scan (nmap -f)**

Der Parameter -f weist Nmap an, die TCP-Header der Scan-Pakete in kleinere Fragmente (typischerweise 8-Byte-Segmente) aufzuteilen. Diese Technik zielt darauf ab, die Erkennungs-Engine eines Sicherheitssystems zu umgehen. Simple Paketfilter oder IPS-Systeme könnten Schwierigkeiten haben, alle Fragmente korrekt und in Echtzeit zusammenzusetzen, um die eigentliche Absicht zu erkennen. Dieser Test prüft gezielt die Effizienz der Paket-Reassemblierungsfunktion der DPI-Engine. Ein robustes IPS muss in der Lage sein, fragmentierte Angriffe zu rekonstruieren und als Scan zu identifizieren, anstatt die Fragmente einzeln als harmlos durchzulassen.

- **Scan mit Daten (nmap --data-length 25)**

Der Parameter --data-length 25 hängt an die gesendeten Scan-Pakete eine definierte Anzahl (hier 25 Bytes) an zufälligen Daten an. Standard-Portscan-Pakete enthalten normalerweise keine Nutzlast. Dieser Test wurde ausgewählt, um zu prüfen, ob das IPS auf Anomalien in der Paketstruktur reagiert. Durch das Hinzufügen einer unerwarteten Nutzlast wird der Fingerprint des Scans verändert. Es wird getestet, ob sich das IPS dadurch täuschen lässt oder ob es die Pakete dennoch korrekt als Teil eines Scans klassifiziert.

- **Decoy-Scan (nmap -D RND:10)**

Der Parameter -D aktiviert einen "Decoy-Scan", der die Herkunft des Scans verschleiern soll. RND:10 weist Nmap an, neben der echten IP-Adresse des Angreifers zehn weitere, zufällig generierte Quell-IP-Adressen in die Scan-Pakete einzutragen. Für das Zielsystem sieht es so aus, als würde der Scan von einer ganzen Gruppe von Rechnern ausgehen. Es wird geprüft, ob das System trotz der vielen gefälschten Quell-IP-Adressen in der Lage ist, den wahren Ursprung des Angriffs zu identifizieren und gezielt nur die reale IP-Adresse des Angreifers zu blockieren, anstatt auf die Köder hereinzufallen.

- **Schwachstellen Scan (nmap --script vuln)**

Der Befehl `--script vuln` weist die Nmap Scripting Engine (NSE) an, eine Sammlung von Skripten auszuführen, die aktiv versuchen bekannte Schwachstellen auf dem Zielsystem zu finden und teilweise auszunutzen. Dabei werden spezifische Anfragen gesendet, die charakteristisch für bestimmte Exploits sind. Er wurde ausgewählt, um die signaturbasierte Erkennung des IPS direkt zu prüfen. Es wird verifiziert, ob das IPS die Anfragen der Nmap-Skripte anhand bekannter Angriffsmuster erkennt und proaktiv blockiert, bevor eine Schwachstelle bestätigt werden kann.

4.1.2 Denial-of-Service (DoS)-Angriffe

In diesem Testfall wird ein Angriff auf der Anwendungsebene (Layer 7) simuliert, der darauf abzielt, die Ressourcen eines Webserver zu erschöpfen und ihn so für legitime Anfragen un erreichbar zu machen.

- **GoldenEye DoS Angriff**

Das Tool GoldenEye erzeugt eine hohe Last auf einem Webserver, indem es eine Flut von scheinbar legitimen HTTP-Anfragen mit ständig wechselnden Parametern sendet. Dies macht eine Abwehr auf reiner Netzwerkebene schwierig, da die Anfragen einzeln unauffällig wirken können.

Dieser Test wurde ausgewählt, um zu prüfen, ob das IPS in der Lage ist, einen Angriff auf der Anwendungsebene zu erkennen. Es soll verifiziert werden, ob das System das anomale Verhalten (hohe Frequenz von Anfragen von einer Quelle) als DoS-Angriff einstuft und die Quell-IP blockiert.

4.1.3 Exploits (Ausnutzung von Schwachstellen)

In dieser Kategorie wird die Kernkompetenz eines IPS geprüft: die Erkennung und Blockade von konkreten Exploits, die bekannte Software-Schwachstellen ausnutzen, um unautorisierten Zugriff oder Kontrolle über ein System zu erlangen.

- **EternalBlue (MS17-010)**

Mithilfe des Metasploit-Frameworks wird versucht, die bekannte Schwachstelle im SMB-Protokoll von Windows-Systemen auszunutzen. Dieser Exploit ermöglicht die Ausführung von beliebigem Code auf einem ungepatchten System. Dieser Test ist essenziell, da es sich um einen der wirkungsvollsten Exploits der letzten Jahre handelt. Es wird geprüft, ob das IPS die charakteristische Signatur des Angriffsverkehrs im SMB-Protokoll zuverlässig erkennt. Dabei ist nur von Bedeutung, ob das IPS den Exploit erkennt und nicht ob

EternalBlue Schaden am System anrichten kann.

- **Shellshock (CVE-2014-6271)**

Shellshock, offiziell als CVE-2014-6271 bekannt, ist eine kritische Sicherheitslücke in der weitverbreiteten Unix-Shell "Bash". Die Schwachstelle erlaubt es einem Angreifer, beliebige Befehle auf einem Zielsystem auszuführen, indem eine speziell präparierte Zeichenkette an eine Anwendung gesendet wird, die im Hintergrund auf die Bash-Shell zurückgreift. Der häufigste Angriffsvektor hierfür sind Webserver.

Dieser Exploit wurde als Testfall ausgewählt, da er ein klassisches Beispiel für einen hochkritischen Angriff auf der Anwendungsebene darstellt, der über den Standard-HTTP-Port läuft und für simple Paketfilter unsichtbar ist. Der Angriff besitzt eine sehr eindeutige und bekannte Signatur `(({ :; } ;))`, was ihn zu einem idealen Kandidaten macht, um die Effektivität der signaturbasierten Erkennung eines IPS zu überprüfen.

- **SQL-Injection (sqlmap)**

Mit dem Tool sqlmap wird automatisiert versucht, über Parameter einer Webanwendung bössartige SQL-Befehle an die dahinterliegende Datenbank zu senden. Dies bildet ebenfalls einen großen Angriffsvektor

Ziel ist die Überprüfung der Fähigkeit des IPS, diese gängige Web-Angriffstechnik anhand von typischen SQL-Schlüsselwörter und der speziellen/untypischen Syntax im HTTP-Anwendungsverkehr zu erkennen.

4.1.4 Passwortangriffe (Brute-Force)

- **Hydra SSH Brute-Force**

Dieser Testfall simuliert einen Brute-Force-Angriff auf den SSH-Dienst eines Servers mithilfe des Tools Hydra. Solche Angriffe, bei denen automatisiert eine große Anzahl von Passwörtern ausprobiert wird, gehören zu den häufigsten Bedrohungen für öffentlich erreichbare Dienste und stellen eine grundlegende Anforderung an moderne Schutzsysteme dar.

Dieser Test wurde ausgewählt, um nicht nur die reine Detektion, sondern vor allem die Qualität und Präzision der Erkennung durch das IPS zu bewerten. Das Ziel ist es zu überprüfen, ob das System die hohe Frequenz an fehlgeschlagenen Anmeldeversuchen von einer einzigen Quelle korrekt als kritischen Brute-Force-Angriff klassifiziert.

4.1.5 Malware-Erkennung

Dieser Testfall überprüft die grundlegende Fähigkeit der Systeme, bekannte und eindeutige Malware-Muster im Netzwerkverkehr zu identifizieren.

- **EICAR Testvirus**

Der EICAR-Testfile (European Institute for Computer Antivirus Research) ist kein echter Virus, sondern eine standardisierte, harmlose Zeichenfolge. Sie wurde als sicherer Test für die Funktionalität von Malware-Scannern entwickelt.

Es wird ausschließlich die Fähigkeit des Systems geprüft, eine bekannte, bösartige Signatur innerhalb eines Datenstroms zu erkennen und darauf zu reagieren. Die ungefährliche Natur der Datei gewährleistet dabei eine sichere und exakt reproduzierbare Testdurchführung, was eine Grundvoraussetzung für einen objektiven Vergleich ist.

Ziel des Tests ist es zu verifizieren, ob das IPS eine Deep Packet Inspection des Datenstroms durchführt, die bekannte EICAR-Signatur innerhalb der Nutzlast erkennt und daraufhin die konfigurierte Aktion, beispielsweise das Verwerfen des Pakets (drop), korrekt ausführt.

4.2 Durchführen der Tests

5 Auswertung

5.1 Analyse der Testergebnisse

5.2 Vergleich der IPS-Systeme

6 Fazit und Ausblick

7 Abkürzungsverzeichnis

BSI - Bundesamt für Sicherheit in der Informationstechnik

DoS - Denial of Service

DDoS - Distributed Denial of Service

KMU - Kleine und Mittelständige Unternehmen

IPS - Intrusion Prevention System

IDS - Intrusion Detection System

NGFW - Next Generation Firewall

VPN - Virtual Private Network

DPI - Deep Packet Inspection

TAP - Test Access Point

NIC - Netzwerkkarte

RAM - Arbeitsspeicher

GUI - Grafische Benutzeroberfläche

8 Selbstständigkeitserklärung

Ich versichere, dass ich die vorliegende Hausarbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form weder veröffentlicht, noch einer anderen Prüfungsbehörde vorgelegt.

Leipzig, 23. August 2025

Leon Baumgarten

Literaturverzeichnis

- [1] Bundesamt für Sicherheit in der Informationstechnik
Die Lage der IT-Sicherheit in Deutschland 2024
<https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2024.html>
Abgerufen am: 02.07.2025

- [2] Bundesamt für Sicherheit in der Informationstechnik
Viren und Würmer
https://www.bsi.bund.de/DE/Themen/Verbraucherinnen-und-Verbraucher/Cyber-Sicherheitslage/Methoden-der-Cyber-Kriminalitaet/Schadprogramme/Viren-und-Wuermer/viren-und-wuermer_node.html
Abgerufen am: 04.07.2025

- [3] Bitkom e.V.
Wirtschaftsschutz 2024
<https://www.bitkom.org/sites/main/files/2024-08/240828-bitkom-charts-wirtschaftsschutz-cybercrime.pdf>
Abgerufen am: 02.07.2025

- [4] *Was ist ein DDoS-Angriff?*
<https://www.cloudflare.com/de-de/learning/ddos/what-is-a-ddos-attack/>
Abgerufen am: 02.07.2025

- [5] Wesley M. Eddy
TCP SYN Flooding Attacks and Common Mitigations
<https://www.rfc-editor.org/rfc/rfc4987.html>
Abgerufen am: 03.07.2025

- [6] *Was ist Schadsoftware?*
<https://www.kaspersky.de/resource-center/definitions/what-is-malware>
Abgerufen am: 04.07.2025

- [7] European Union Agency for Cybersecurity
ENISA Threat Landscape 2023

- <https://www.enisa.europa.eu/publications/enisa-threat-landscape-2023>
Abgerufen am: 04.07.2025
- [8] Wikipedia (2025)
Cyber Kill Chain
https://de.wikipedia.org/wiki/Cyber_Kill_Chain
Abgerufen am: 04.07.2025
- [9] Wikipedia (2025)
Exploits
<https://de.wikipedia.org/wiki/Exploit> Abgerufen am: 04.07.2025
- [10] Claudia Eckert
IT-Sicherheit: Konzepte - Verfahren - Protokolle
Walter de Gruyter GmbH & Co KG, 2014
- [11] Open Information Security Foundation
Suricata Documentation – Rules.
<https://docs.suricata.io/en/latest/rules/intro.html#action>
Abgerufen am: 06.07.2025
- [12] Karen Scarfone, Peter Mell (NIST)
Guide to Intrusion Detection and Prevention Systems (IDPS)
<https://csrc.nist.gov/pubs/sp/800/94/final>
Abgerufen am: 06.07.2025
- [13] SANS Institute
The Difference between a TAP and a SPAN
SANS Institutet, 2014
- [14] Deciso B.V.
About the fork
<https://docs.opnsense.org/history/thefork.html>
Abgerufen am: 01.08.2025
- [15] Fortinet Inc.
FortiGate 70F Series
<https://www.fortinet.com/content/dam/fortinet/assets/data-sheets/fortigate-70f-series.pdf>
Abgerufen am: 01.08.2025
- [16] Fortinet Inc.
FortiGate 70F Series

<https://www.fortinet.com/de/products/next-generation-firewall>
Abgerufen am: 01.08.2025

- [17] Proxmox Server Solutions GmbH
Proxmox VE Administration Guide
<https://pve.proxmox.com/pve-docs/>
Abgerufen am: 22.08.2025

- [18] Fortinet Inc.
How to install FortiGate VM on Proxmox
<https://community.fortinet.com/t5/FortiGate/Technical-Tip-How-to-install-FortiGate-VM-on-Proxmox/ta-p/301097>
Abgerufen am: 21.08.2025

- [19] Defense Advanced Research Projects Agency
TRANSMISSION CONTROL PROTOCOL
<https://datatracker.ietf.org/doc/html/rfc793#section-3.9>
Abgerufen am: 22.08.2025

9 Abbildungsverzeichnis

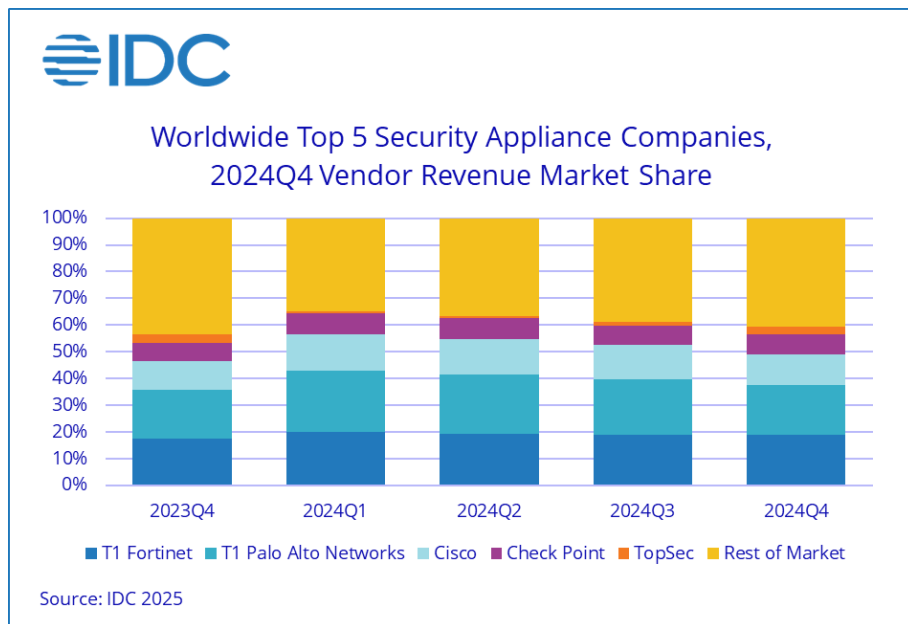


Abbildung 9.1: Marktanteil Fortinet

<https://my.idc.com/getdoc.jsp?containerId=prUS53243925>

```
# dmidecode 3.6
Scanning /dev/mem for entry point.
SMBIOS 3.0.0 present.
Table at 0xCFE97040.

Handle 0x0000, DMI type 0, 26 bytes
BIOS Information
    Vendor: coreboot
    Version: v4.19.0.1
    Release Date: 01/31/2023
    ROM Size: 8 MB
    Characteristics:
        PCI is supported
        PC Card (PCMCIA) is supported
        BIOS is upgradeable
        Selectable boot is supported
        ACPI is supported
        Targeted content distribution is supported
    BIOS Revision: 0.0
    Firmware Revision: 0.0

Handle 0x0001, DMI type 1, 27 bytes
System Information
    Manufacturer: PC Engines
    Product Name: apu4
    Version: 1.0
    Serial Number: 1543855
    UUID: Not Settable
    Wake-up Type: Reserved
```

Abbildung 9.2: Die Hardwarekomponenten der Firewall.

```
root@pve
-----
OS: Proxmox VE 8.4.0 x86_64
Host: HP EliteDesk 800 G4 DM 35W
Kernel: 6.8.12-9-pve
Uptime: 21 hours, 50 mins
Packages: 789 (dpkg)
Shell: bash 5.2.15
Terminal: /dev/pts/0
CPU: Intel i5-8500T (6) @ 3.500GHz
GPU: Intel CoffeeLake-S GT2 [UHD Graph
Memory: 8664MiB / 15776MiB
```

Abbildung 9.3: Die Hardwarekomponenten der ProxmoxVE-Umgebung.

| Virtuelle Maschine 100 (ubuntu01) auf Knoten 'pve' | | Keine Tags |
|--|--|--|
| Übersicht | Hinzufügen Entfernen Bearbeiten Disk-Aktion Zurücksetzen | |
| Konsole | | |
| Hardware | | |
| Cloud-Init | | |
| Optionen | | |
| Task History | | |
| Monitor | | |
| Backup | | |
| Replizierung | | |
| Snapshots | | |
| Firewall | | |
| Rechte | | |
| | Speicher | 4.00 GiB |
| | Prozessoren | 4 (1 sockets, 4 cores) [x86-64-v2-AES] |
| | BIOS | Standardeinstellung (SeaBIOS) |
| | Anzeige | Standardeinstellung |
| | Maschinentyp | Standardeinstellung (i440fx) |
| | SCSI Controller | VirtIO SCSI single |
| | CD/DVD Laufwerk (ide2) | local:iso/ubuntu-24.04.2-desktop-amd64.iso,media=cdrom,size=6194550K |
| | Laufwerk (scsi4) | local-lvm:vm-100-disk-0,discard=on,iotread=1,size=50G |
| | Netzwerkarte (net0) | virtio=BC:24:11:97:76:A9,bridge=vmbro |

Abbildung 9.4: Zuweisung der Hardwareressourcen für Ubuntu-VMs

| Virtuelle Maschine 102 (kali) auf Knoten 'pve' | | Keine Tags |
|--|--|---|
| Übersicht | Hinzufügen Entfernen Bearbeiten Disk-Aktion Zurücksetzen | |
| Konsole | | |
| Hardware | | |
| Cloud-Init | | |
| Optionen | | |
| Task History | | |
| Monitor | | |
| Backup | | |
| Replizierung | | |
| Snapshots | | |
| Firewall | | |
| Rechte | | |
| | Speicher | 4.00 GiB |
| | Prozessoren | 4 (1 sockets, 4 cores) [x86-64-v2-AES] |
| | BIOS | Standardeinstellung (SeaBIOS) |
| | Anzeige | Standardeinstellung |
| | Maschinentyp | Standardeinstellung (i440fx) |
| | SCSI Controller | VirtIO SCSI single |
| | CD/DVD Laufwerk (ide2) | local:iso/kali-linux-2025.2-installer-amd64.iso,media=cdrom,size=4373964K |
| | Laufwerk (scsi0) | local-lvm:vm-102-disk-0,discard=on,iotread=1,size=50G |
| | Netzwerkarte (net0) | virtio=BC:24:11:C3:81:94,bridge=vmbro |

Abbildung 9.5: Zuweisung der Hardwareressourcen für Kali-Linux-VM

| Virtuelle Maschine 104 (win11a) auf Knoten 'pve' | | Keine Tags |
|--|--|---|
| Übersicht | Hinzufügen Entfernen Bearbeiten Disk-Aktion Zurücksetzen | |
| Konsole | | |
| Hardware | | |
| Cloud-Init | | |
| Optionen | | |
| Task History | | |
| Monitor | | |
| Backup | | |
| Replizierung | | |
| Snapshots | | |
| Firewall | | |
| Rechte | | |
| | Speicher | 8.00 GiB |
| | Prozessoren | 4 (1 sockets, 4 cores) [x86-64-v2-AES] |
| | BIOS | OVMF (UEFI) |
| | Anzeige | Standardeinstellung |
| | Maschinentyp | pc-q35-9.2+pve1 |
| | SCSI Controller | VirtIO SCSI single |
| | Laufwerk (ide0) | local-lvm:vm-104-disk-1,discard=on,size=70G |
| | CD/DVD Laufwerk (ide1) | local:iso/virtio-win-0.1.271.iso,media=cdrom,size=709474K |
| | CD/DVD Laufwerk (ide2) | local:iso/26100.1742.240906-0331.ge_release_svc_refresh_CLIENTENTERPRISEVAL_OEMRET_x64FRE_de-de.iso,media=cdrom,size=5301356K |
| | Netzwerkarte (net0) | e1000=BC:24:11:8B:D5:A2,bridge=vmbro |
| | EFI-Disk | local-lvm:vm-104-disk-0,efitype=4m,pre-enrolled-keys=1,size=4M |
| | TPM-Status | local-lvm:vm-104-disk-2,size=4M,version=v2.0 |

Abbildung 9.6: Zuweisung der Hardwareressourcen für Windows11-VM

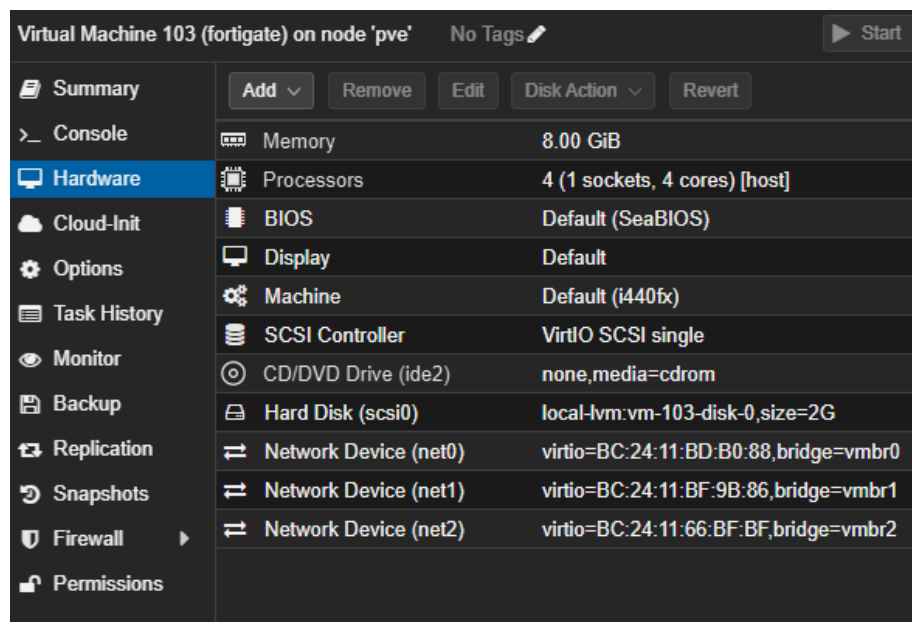


Abbildung 9.7: Konfiguration der FortiGate VM

```

FGVMSLTM25011489 # show system interface port2
config system interface
  edit "port2"
    set vdom "root"
    set ip 192.168.10.1 255.255.255.0
    set allowaccess ping https ssh
    set type physical
    set alias "LAN"
    set snmp-index 6
  next
end

FGVMSLTM25011489 # show system dhcp server
config system dhcp server
  edit 1
    set dns-service default
    set ntp-service local
    set default-gateway 192.168.10.1
    set netmask 255.255.255.0
    set interface "port2"
    config ip-range
      edit 1
        set start-ip 192.168.10.100
        set end-ip 192.168.10.200
      next
    end
  next
edit 2
  set dns-service default
  set default-gateway 192.168.20.1
  set netmask 255.255.255.0
  set interface "port3"
  config ip-range
    edit 1
      set start-ip 192.168.20.100
      set end-ip 192.168.20.200
    next
  end
next
end

```

Abbildung 9.8: Init-Konfiguration der Interfaces und DHCP

| | | | |
|-----------|---------------|-------------|---------|
| 0.0.0.0/0 | 192.168.100.1 | WAN (port1) | Enabled |
|-----------|---------------|-------------|---------|

Abbildung 9.9: Static Route zum WAN



| New Address | |
|----------------------------|---|
| Name | LAN_Subnet |
| Color |  <input type="button" value="Change"/> |
| Type | Subnet ▼ |
| IP/Netmask | 192.168.10.0/24 |
| Interface |  LAN (port2) ▼ |
| Static route configuration | <input checked="" type="checkbox"/> |
| Comments | <input type="text" value="Write a comment..."/> 0/255 |

Abbildung 9.10: Adressobjekt für das LAN-Subnetz












| New Address | |
|----------------------------|---|
| Name | KALI_Subnet |
| Color |  <input type="button" value="Change"/> |
| Type | Subnet ▼ |
| IP/Netmask | 192.168.20.0/24 |
| Interface |  KALI (port3) ▼ |
| Static route configuration | <input checked="" type="checkbox"/> |
| Comments | <input type="text" value="Write a comment..."/> 0/255 |

Abbildung 9.11: Adressobjekt für das Kali-Subnetz

| | |
|---|--|
| Name  | LAN_to_WAN |
| Type | Standard ZTNA |
| Incoming Interface |  LAN (port2) ▼ |
| Outgoing Interface |  WAN (port1) ▼ |
| Source |  LAN_Subnet ✕ + |
| IP/MAC Based Access Control  | + |
| Destination |  all ✕ + |
| Schedule |  always ▼ |
| Service |  ALL ✕ + |
| Action | ✓ ACCEPT  DENY |


Inspection Mode **Flow-based** Proxy-based

Firewall/Network Options


NAT ☒


IP Pool Configuration **Use Outgoing Interface Address** Use Dynamic IP Pool

Preserve Source Port ☐


Protocol Options **PROT** default ▼ 


Security Profiles

AntiVirus ☒ **AV** default ▼ 


Web Filter ☒ **WEB** default ▼ 

DNS Filter ☐

Application Control ☒ **APP** default ▼ 

IPS ☒ **IPS** default ▼ 

File Filter ☐

SSL Inspection **SSL** certificate-inspection ▼ 

Logging Options

Log Allowed Traffic ☒ **Security Events** All Sessions

Generate Logs when Session Starts ☐

Capture Packets ☐

Comments 0/1023

Enable this policy ☒

Abbildung 9.12: FortiGate VM Policy LAN_to_WAN

| | |
|-----------------------------|---------------|
| Name | KALI_to_LAN |
| Type | Standard ZTNA |
| Incoming Interface | KALI (port3) |
| Outgoing Interface | LAN (port2) |
| Source | KALI_Subnet |
| IP/MAC Based Access Control | |
| Destination | LAN_Subnet |
| Schedule | always |
| Service | ALL |
| Action | ACCEPT DENY |

Inspection Mode **Flow-based** Proxy-based

Firewall/Network Options

NAT ☐

Protocol Options **PROT** default

Security Profiles

AntiVirus ☒ **AV** default

Web Filter ☒ **WEB** default

DNS Filter ☐

Application Control ☒ **APP** default

IPS ☒ **IPS** default

File Filter ☐

SSL Inspection **SSL** certificate-inspection

Logging Options

Log Allowed Traffic ☒ Security Events **All Sessions**

Generate Logs when Session Starts ☐

Capture Packets ☐

Comments 0/1023

Enable this policy ☒

Abbildung 9.13: FortiGate VM Policy KALI_to_LAN

Name

Comments
0/255

Block malicious URLs
☐

IPS Signatures and Filters

+ Create New

Edit

Delete

| Details | Exempt IPs | Action | Packet Logging |
|--|------------|------------------|--------------------|
| <div>TGT Server</div> <div>TGT Client</div> <div>SEV <div><div></div><div></div><div></div><div></div><div></div></div></div> <div>SEV <div><div></div><div></div><div></div><div></div><div></div></div></div> <div>ACT Block</div> | | <div>Block</div> | <div>Enabled</div> |

1

Abbildung 9.14: FortiGate VM Custom IPS Sensor

Name

Comments 0/255

SSL Inspection Options

Enable SSL inspection of Multiple Clients Connecting to Multiple Servers
 Protecting SSL Server

Inspection method SSL Certificate Inspection Full SSL Inspection

CA certificate Fortinet_CA_SSL
 Download

Blocked certificates Allow Block View Blocked Certificates

Untrusted SSL certificates Allow Block Ignore View Trusted CAs List

Server certificate SNI check Enable Strict Disable

Enforce SSL cipher compliance ☐

Enforce SSL negotiation compliance ☐

RPC over HTTPS ☐

Protocol Port Mapping

Inspect all ports ☒

HTTPS ☒

SMTPS ☒

POP3S ☒

IMAPS ☒

FTPS ☒

DNS over TLS ☒

Exempt from SSL Inspection

Reputable websites ☐

Web categories Finance and Banking Health and Wellness Personal Privacy +

Addresses +

Log SSL exemptions ☐

SSH Inspection Options

SSH deep scan ☐

Common Options

Invalid SSL certificates Allow Block Custom

Log SSL anomalies ☒

Abbildung 9.15: FortiGate VM Custom SSL Inspection Profile

| ID | Name | From | To | Source | Destination | Schedule | Service | Action | NAT | Security Profiles | Log | Bytes |
|----|---------------|--------------|-------------|-------------|-------------|----------|---------|--------|----------|--|----------|-----------|
| 1 | LAN_to_WAN | LAN (port2) | WAN (port1) | LAN_Subnet | all | always | ALL | ACCEPT | Enabled | default IPS IPS-Testumgebung Deep-Inspection-Testumgebung | All | 66.85 MB |
| 2 | KALI_to_LAN | KALI (port3) | LAN (port2) | KALI_Subnet | LAN_Subnet | always | ALL | ACCEPT | Disabled | default IPS IPS-Testumgebung Deep-Inspection-Testumgebung | All | 0 B |
| 3 | KALI_to_WAN | KALI (port3) | WAN (port1) | KALI_Subnet | all | always | ALL | ACCEPT | Enabled | no-inspection | Disabled | 24.13 MB |
| 0 | Implicit Deny | any | any | all | all | always | ALL | DENY | | | Disabled | 123.73 kB |

Abbildung 9.16: FortiGate VM finale Policies

Settings

Download

Rules

User defined

Alerts

Schedule

advanced mode

General Settings

Enabled

☒

IPS mode

☒

Promiscuous mode

☐

Interfaces

LAN, WAN, WAN_KALI

Clear All

Select All

Detection

Pattern matcher

Hyperscan

Logging

Enable syslog alerts

☒

Enable eve syslog output

☐

Rotate log

Weekly

Save logs

4

Apply

Abbildung 9.17: Aktivierung des IDS und IPS

Services: Squid Web Proxy: Administration

General Proxy Settings

Forward Proxy

Proxy Auto-Config

Remote Access Control Lists

Support

advanced mode

Proxy interfaces

LAN

Clear All

Select All

Proxy port

3128

Enable Transparent HTTP proxy

☒

Enable SSL inspection

☒

Log SNI information only

☐

SSL Proxy port

3129

CA to use

OPNsense Inspection CA test

SSL no bump sites

Clear All

Copy

Paste

Text

Apply

Abbildung 9.18: Aktivierung des Squid Web Proxy mit SSL Inspection

| | Source | | | | Destination | | NAT | | | |
|--------------------------|-----------|-------|---------|-------|-------------|-------------|-----------|-------|---------------------------|--|
| <input type="checkbox"/> | Interface | Proto | Address | Ports | Address | Ports | IP | Ports | Description | |
| <input type="checkbox"/> | ! LAN | TCP | * | * | LAN address | 22, 80, 443 | * | * | Anti-Lockout Rule | |
| <input type="checkbox"/> | ↔ LAN | TCP | LAN net | * | * | 80 (HTTP) | 127.0.0.1 | 3128 | redirect traffic to proxy | |
| <input type="checkbox"/> | ↔ LAN | TCP | LAN net | * | * | 443 (HTTPS) | 127.0.0.1 | 3129 | redirect traffic to proxy | |

Abbildung 9.19: NAT Regeln zum Umleiten auf den Proxy

65

[illegible]

Abbildung 9.20: Live Log Einträge des Web Proxy