Firewall-Regeln für Security Server Setup

Ursprüngliche Frage

Frage: "Für die Dienste die ich auf meinem Server laufen will, welche Firewall-Regeln soll ich zuerst bevor ich anfange einstellen, für mehr Sicherheit und keine Konflikte. Denke zweimal nach."

System-Architektur umfasst:

- DNS-Security-Stack Implementation (Pi-hole, Unbound DNS-over-TLS)
- VPN und Zero-Trust Network Access (Tailscale)
- Intrusion Detection System (Suricata)
- Network Analysis Platform (Zeek)
- Advanced Security Features (DNS Sinkholing, Fail2ban)
- Monitoring und Alerting (Grafana Dashboard)
- Automation und Backup

Strategische Antwort: Präventive Firewall-Strategie

Phase 0: Vorbereitung und Planung

ſ		
	bash	

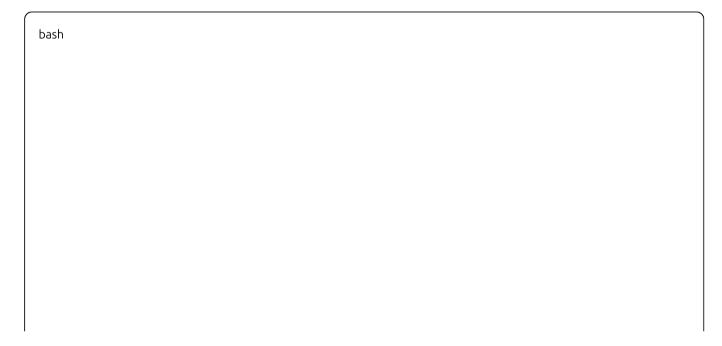
```
# System-Interface identifizieren
ip a
# Notiere dir die Interface-Namen (z.B. eth0, ens3, etc.)

# WICHTIG: Dein lokales Netzwerk identifizieren!
ip route | grep default
# Beispiel: Wenn dein Router 192.168.178.1 ist > verwende 192.168.178.0/24
# Wenn dein Router 192.168.1.1 ist > verwende 192.168.1.0/24
# Wenn dein Router 10.0.0.1 ist > verwende 10.0.0.0/24
# IMMER dein echtes Subnetz in ALLEN Regeln verwenden!

# Aktuelle Firewall-Regeln sichern (falls vorhanden)
sudo ufw status numbered > ~/ufw_backup_$(date +%Y%m%d).txt

# Sicherstellen, dass du alternativen Zugang hast (KVM/Console)
```

Phase 1: Absolute Minimal-Konfiguration (ZUERST)



```
# Komplettes Reset und härteste Einstellungen
sudo ufw --force reset
sudo ufw default deny incoming
sudo ufw default deny outgoing # <- WICHTIG: Auch ausgehend blockieren
sudo ufw default deny forward
sudo ufw default deny routed
# NUR absolute Notwendigkeiten
sudo ufw allow out on lo
sudo ufw allow in on lo
# Temporärer SSH-Zugang (wird später durch Tailscale ersetzt)
# NUR wenn du physischen Zugang verlieren könntest:
# sudo ufw allow in 22/tcp from 192.168.1.0/24 comment 'TEMP SSH local'
# Aktivieren mit minimaler Konfiguration
sudo ufw --force enable
```

Warum so restriktiv?

- Verhindert, dass Services "heimlich" Verbindungen aufbauen
- Zwingt dich, jeden Service bewusst zu konfigurieren
- Verhindert DNS-Leaks während der Installation
- Etabliert Zero-Trust von Anfang an

Phase 2: Tailscale Installation (PRIORITÄT für Remote-Zugang)

bash

```
# ===== WICHTIG: Installation-Strategie =====
# Die temporären Regeln sind NICHT für Konfliktvermeidung!
# Sie erlauben nur Downloads/Updates aus dem Internet.
# Ohne diese = keine Pakete können geladen werden (outgoing geblockt)
# Option A: EMPFOHLEN für Server-Setup
# Öffne die Ports EINMAL am Anfang für ALLE Installationen:
sudo ufw allow out 53/udp comment 'TEMP DNS UDP'
sudo ufw allow out 53/tcp comment 'TEMP DNS TCP'
sudo ufw allow out 80/tcp comment 'TEMP HTTP'
sudo ufw allow out 443/tcp comment 'TEMP HTTPS'
# → Lass diese offen während der gesamten Konfiguration
# → Lösche sie EINMAL am Ende nach ALLEN Installationen
# Option B: Nur wenn maximale Sicherheit wichtig ist
# Öffne/schließe für jede einzelne Installation (nervt aber)
# Tailscale installieren (falls noch nicht geschehen)
curl -fsSL https://tailscale.com/install.sh | sh
sudo tailscale up --ssh
# Tailscale-Konnektivität etablieren
sudo ufw allow out 41641/udp comment 'Tailscale DERP'
sudo ufw allow out 3478/udp comment 'Tailscale STUN'
sudo ufw allow out on tailscale0 to any comment 'Tailscale outbound'
# SSH NUR über Tailscale erlauben
sudo ufw allow in on tailscale0 to any port 22 proto tcp comment 'SSH via Tailscale only'
sudo ufw allow in on tailscale0 from 100.64.0.0/10 comment 'Tailscale subnet'
# TEMP Regeln entfernen → SIEHE "PHASE 7: FINALISIERUNG"
```

Phase 3: DNS-Foundation etablieren

3.1 Pi-hole Installation und Konfiguration

```
# WICHTIG: Ersetze 192.168.1.0/24 mit DEINEM Netzwerk!

# Beispiele: 192.168.178.0/24, 10.0.0.0/24, 192.168.0.0/24

# DNS für lokale Clients erlauben (Pi-hole)

sudo ufw allow in on eth0 to any port 53 proto udp from 192.168.1.0/24 comment 'DNS UDP Pi-hole local'

sudo ufw allow in on eth0 to any port 53 proto tcp from 192.168.1.0/24 comment 'DNS TCP Pi-hole local'

# Pi-hole Admin Interface (nur lokal)

sudo ufw allow in on eth0 to any port 80 proto tcp from 192.168.1.0/24 comment 'Pi-hole Admin local'

# Alternative: Pi-hole Admin nur über Tailscale

# sudo ufw allow in on tailscale0 to any port 80 proto tcp comment 'Pi-hole Admin Tailscale'
```

3.2 Unbound Integration (NACH Pi-hole Tests)

```
bash

# Unbound upstream DNS-over-TLS

sudo ufw allow out 853/tcp comment 'DNS-over-TLS upstream'

# Falls Unbound auf anderem Port läuft (z.B. 5335)

# Keine zusätzliche Firewall-Regel nötig (localhost only)
```

Phase 4: Monitoring und IDS

4.1 Suricata IDS

bash

Suricata braucht keine eingehenden Ports # Nur Promiscuous Mode auf dem Interface

Updates für Suricata-Regeln

sudo ufw allow out 443/tcp to any comment 'Suricata rule updates'

4.2 Zeek Network Monitoring

bash

Zeek läuft ebenfalls im Promiscuous Mode

Keine zusätzlichen Firewall-Regeln erforderlich

4.3 Grafana Dashboard

bash

Grafana nur über Tailscale

sudo ufw allow in on tailscale0 to any port 3000 proto tcp comment 'Grafana via Tailscale'

ODER mit alternativem Port bei Konflikt

sudo ufw allow in on tailscale0 to any port 3001 proto tcp comment 'Grafana alt port'

Phase 5: Sicherheits-Härtung

bash

```
#Rate Limiting aktivieren
sudo ufw limit in on tailscale0 to any port 22 proto tcp comment 'SSH rate limit'
sudo ufw limit in on eth0 to any port 53 comment 'DNS rate limit'

#Explizite Deny-Regeln für gefährliche Services
sudo ufw insert 1 deny out 25 comment 'Block SMTP'
sudo ufw insert 1 deny out 587 comment 'Block SMTP submission'
sudo ufw insert 1 deny out 465 comment 'Block SMTPS'
sudo ufw insert 1 deny in 445 comment 'Block SMB'
sudo ufw insert 1 deny in 139 comment 'Block NetBIOS'
sudo ufw insert 1 deny in 135 comment 'Block RPC'
sudo ufw insert 1 deny in 3389 comment 'Block RDP'

#Logging aktivieren
sudo ufw logging on
sudo ufw logging high #Für initiale Debugging-Phase
```

Phase 6: Fail2ban Integration

```
bash

# Fail2ban arbeitet mit iptables/ufw zusammen

# Konfiguration in /etc/fail2ban/jail.local

# Sicherstellen, dass Fail2ban UFW nutzt

sudo systemctl enable fail2ban

sudo systemctl start fail2ban

# Fail2ban-Status prüfen

sudo fail2ban-client status
```

Phase 7: FINALISIERUNG - Temporäre Regeln entfernen

```
bash
# ==== WICHTIG: ERST NACH ALLEN INSTALLATIONEN =====
# Wenn du ALLE Services installiert hast und alles läuft:
# 1. Check welche TEMP Regeln aktiv sind
sudo ufw status numbered | grep TEMP
# 2. Lösche alle temporären Regeln
sudo ufw delete allow out 53/udp
sudo ufw delete allow out 53/tcp
sudo ufw delete allow out 80/tcp
sudo ufw delete allow out 443/tcp
# 3. Verifiziere, dass keine TEMP Regeln mehr existieren
sudo ufw status | grep TEMP # Sollte leer sein
# Ab jetzt: System-Updates nur noch mit temporärem Öffnen/Schließen
```

Konfliktvermeidungs-Strategien

1. Installation-Reihenfolge (KRITISCH):

bash	

- 1. Tailscale (Remote-Zugang sichern)
- 2. Pi-hole (DNS-Foundation)
- 3. Unbound (DNS-Security)
- 4. Fail2ban (Brute-Force-Schutz)
- 5. Suricata ODER Zeek (nicht beide gleichzeitig!)
- 6. Grafana (Monitoring)

2. Port-Konflikt-Management:

3. Debugging und Logging:

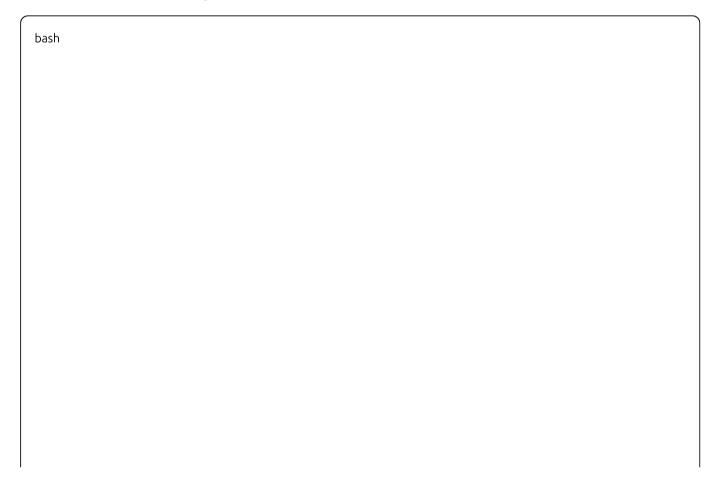
bash	

```
# Live-Monitoring der Firewall
sudo tail -f /var/log/ufw.log

# Geblockte Verbindungen analysieren
sudo grep -i 'block' /var/log/ufw.log | tail -20

# Service-spezifische Logs
journalctl -u tailscaled -f
journalctl -u pihole-FTL -f
```

Test-Prozedur nach jeder Phase



```
# Firewall-Status detailliert prüfen
sudo ufw status verbose
sudo ufw status numbered
# Offene Ports verifizieren
sudo ss -tulpn | grep LISTEN
sudo netstat -tulpn | grep LISTEN
# Externe Sicht testen (von anderem System)
nmap -sT <server-ip> # TCP scan
nmap -sU <server-ip> # UDP scan
# Tailscale-Konnektivität testen
tailscale status
tailscale ping <andere-tailscale-ip>
# DNS-Funktionalität testen
dig @localhost google.com
dig @localhost +tcp google.com
```

Rollback-Strategie

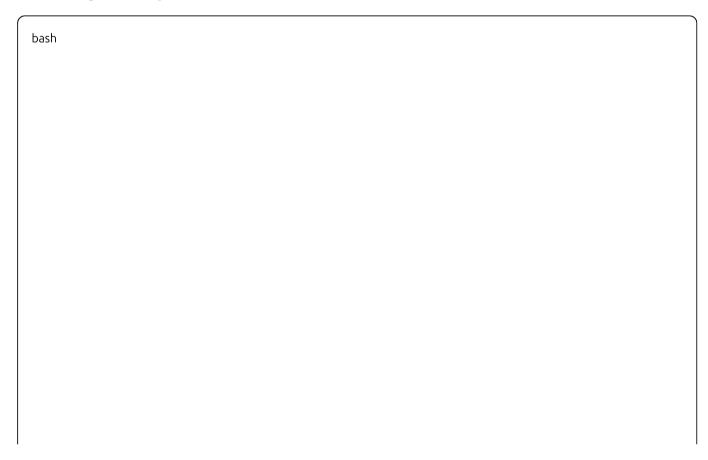
bash

```
# Bei Problemen: Spezifische Regel entfernen
sudo ufw status numbered
sudo ufw delete <nummer>

# Notfall: Firewall temporär deaktivieren
sudo ufw disable

# Kompletter Reset (nur im Notfall!)
sudo ufw --force reset
sudo ufw --force disable
```

Wartung und Updates



```
# Regelmäßige Überprüfung (wöchentlich)
sudo ufw show added
sudo ufw show raw
# Backup der Regeln
sudo ufw status numbered > ~/ufw rules $(date +%Y%m%d).txt
# Update-Prozedur für Services
# 1. Temporär Updates erlauben
sudo ufw allow out 80/tcp comment 'TEMP updates'
sudo ufw allow out 443/tcp comment 'TEMP updates'
# 2. Updates durchführen
sudo apt update && sudo apt upgrade
# 3. Temporäre Regeln wieder entfernen
sudo ufw delete allow out 80/tcp
sudo ufw delete allow out 443/tcp
```

Warum diese Strategie?

- 1. Tailscale-First: Sicherer Remote-Zugang hat Priorität
- 2. Service-Isolation: Jeder Dienst läuft in seiner eigenen "Sandbox"
- 3. Minimal Attack Surface: Nur notwendige Ports, nur notwendige Interfaces
- 4. **Defense in Depth**: Mehrere Sicherheitsebenen (UFW + Fail2ban + IDS)
- 5. Audit-Fähigkeit: Vollständiges Logging und Monitoring
- 6. **Rollback-Fähig**: Jede Änderung kann rückgängig gemacht werden

Checkliste vor Produktiv-Betrieb

Alle temporären Regeln entfernt?
SSH nur über Tailscale erreichbar?
DNS funktioniert lokal und über Pi-hole?
Monitoring (Grafana) nur über Tailscale?
Fail2ban aktiv und konfiguriert?
IDS (Suricata/Zeek) läuft und loggt?
Backup der finalen Firewall-Regeln erstellt?
Notfall-Zugang (KVM/Console) verfügbar?

Dokument optimiert für sichere Server-Konfiguration mit DNS-Security-Stack, Zero-Trust-VPN, IDS und Monitoring-Komponenten. Version: 2.0 - Mit Tailscale-First-Ansatz und verbesserter Struktur