

# Firewall-Regeln für Security Server Setup

## Ursprüngliche Frage

**Frage:** "Für die Dienste die ich auf mein Server laufen will, welche Firewall regeln soll ich Zuerst bevor ich anfangen einzustellen, für mehr Sicherheit und keine Konflikte. Danke 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 1: Absolute Minimal-Konfiguration (ZUERST)

```
bash
```

*# 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
```

```
sudo ufw allow out 22/tcp comment 'SSH outbound'
```

```
sudo ufw allow in 22/tcp comment 'SSH inbound'
```

*# Aktivieren mit minimaler Konfiguration*

```
sudo ufw enable
```

## Warum so restriktiv?

- Verhindert, dass Services "heimlich" Verbindungen aufbauen
- Zwingt dich, jeden Service bewusst zu konfigurieren
- Verhindert DNS-Leaks während der Installation

## Phase 2: DNS-Grundlagen (für Package-Updates)

```
bash
```

```
# Temporär für System-Updates  
sudo ufw allow out 53/udp comment 'DNS queries'  
sudo ufw allow out 80/tcp comment 'HTTP updates'  
sudo ufw allow out 443/tcp comment 'HTTPS updates'
```

```
# System updaten  
sudo apt update && sudo apt upgrade
```

```
# Diese Regeln später wieder entfernen!
```

## Phase 3: Service-by-Service Freischaltung

### 3.1 Pi-hole ZUERST (DNS-Foundation)

```
bash  
  
# Nur das Minimum für Pi-hole  
sudo ufw allow in 53/udp comment 'DNS UDP for Pi-hole'  
sudo ufw allow in 80/tcp from 192.168.1.0/24 comment 'Pi-hole Admin'  
# NICHT Port 53 TCP - kommt später nach Tests
```

### 3.2 Unbound (nach Pi-hole Tests)

```
bash  
  
# Unbound upstream DNS  
sudo ufw allow out 853/tcp comment 'DNS-over-TLS upstream'  
# Erst nach erfolgreichen Pi-hole Tests
```

### 3.3 Tailscale (isoliert testen)

```
bash
```

```
# Tailscale in kontrollierter Umgebung
```

```
sudo ufw allow out 41641/udp comment 'Tailscale DERP'
```

```
sudo ufw allow out 3478/udp comment 'Tailscale STUN'
```

```
# Interface-spezifische Regeln später
```

## Phase 4: Kritische Sicherheitsoverrides

```
bash
```

```
# WICHTIG: Explizite Deny-Regeln für problematische Services
```

```
sudo ufw deny out 25/tcp comment 'Block SMTP spam'
```

```
sudo ufw deny out 587/tcp comment 'Block SMTP submission'
```

```
sudo ufw deny out 465/tcp comment 'Block SMTPS'
```

```
sudo ufw deny in 445/tcp comment 'Block SMB'
```

```
sudo ufw deny in 139/tcp comment 'Block NetBIOS'
```

```
sudo ufw deny in 135/tcp comment 'Block RPC'
```

```
# Rate Limiting SOFORT aktivieren
```

```
sudo ufw limit ssh
```

```
sudo ufw limit 53/udp
```

## Phase 5: Interface-basierte Isolation

```
bash
```

```
# Tailscale Interface isolieren (nach Installation)
```

```
sudo ufw allow in on tailscale0 from 100.64.0.0/10 comment 'Tailscale mesh'
```

```
sudo ufw deny in on tailscale0 from any to any comment 'Block external via Tailscale'
```

```
# Lokales Interface beschränken
```

```
sudo ufw allow in on eth0 from 192.168.1.0/24
```

```
sudo ufw deny in on eth0 from any to any comment 'Block external on LAN'
```

## Konfliktvermeidungs-Strategien

### 1. Service-Reihenfolge beachten:

```
bash
```

```
# NIEMALS gleichzeitig installieren:
```

```
# - Pi-hole + Unbound (Port 53 Konflikt)
```

```
# - Grafana + andere Web-Services (Port-Konflikte)
```

```
# - Suricata + Zeek (Interface-Konflikte)
```

### 2. Port-Mapping vorbereiten:

```
bash
```

```
# Alternative Ports definieren BEVOR Installation:
```

```
# Grafana: 3001 statt 3000 (falls Konflikt)
```

```
# Pi-hole Admin: 8080 statt 80 (falls Webserver läuft)
```

### 3. Logging für Debugging:

```
bash
```

```
sudo ufw logging on
```

```
# Überwache /var/log/ufw.log während jeder Service-Installation
```

## Test-Prozedur nach jeder Phase

```
bash
```

```
# Nach jeder Regel-Änderung:
```

```
sudo ufw status numbered
```

```
sudo netstat -tulnp
```

```
sudo ss -tulnp
```

```
nmap localhost # Interne Sicht
```

```
nmap [externe-IP] # Externe Sicht von anderem Rechner
```

## Warum diese Strategie?

1. **Verhindert Service-Interferenz:** Services können sich nicht "in die Quere kommen"
2. **Zwingt zu bewussten Entscheidungen:** Jeder Port muss explizit freigeschaltet werden
3. **Debugging wird einfacher:** Bei Problemen weißt du genau, welcher Service betroffen ist
4. **Zero-Trust von Anfang an:** Auch ausgehende Verbindungen müssen legitimiert werden
5. **Rollback-fähig:** Jede Phase kann einzeln rückgängig gemacht werden

## Erste Aktion

Starte mit der absolut minimalen Konfiguration und baue Service für Service auf. Das verhindert 90% aller Installationsprobleme!

**Antwort auf Nachfrage:** Phase 2 und dann 3.1

---

*Dokument erstellt für sichere Server-Konfiguration mit DNS-Security-Stack, VPN, IDS und Monitoring-Komponenten.*