Case-Study: Zero-Trust-Netzwerk mit Defense-in-Depth (6 Schutzschichten) und VLAN-Mikro-Segmentierung

> Anonymisierte Portfolio-/Case-Study-Version. Produkt-/Kundenname, Domains, Hostnamen und öffentliche Adressen entfernt; Architektur-Patterns unverändert. Arbeitsprobe IT-Systemintegration / Netzwerk- und Security-Architektur.

Über diese Arbeitsprobe

Diese Case-Study dokumentiert die Netzwerk-, Firewall- und Segmentierungs-Architektur einer selbst aufgebauten und betriebenen Server-Infrastruktur auf einem dedizierten Bare-Metal-Server (Proxmox VE 9.1 als Hypervisor, 35+ laufende Guests aus VMs und LXC-Containern). Sie zeigt den Aufbau eines Zero-Trust-Netzwerks nach Enterprise-Mustern: eine **zweistufige Firewall-Kaskade (Edge → Core)** in zwei getrennten Ketten für LAN und DMZ, strikte VLAN-Mikro-Segmentierung, isolierte Transit-Pfade ohne Layer-2-Lateral-Movement, eine eigene Hypervisor-Host-Firewall mit Einbahn-Vertrauen, ein Out-of-Band-Management sowie Admin-VPN und Site-to-Site-Anbindung.

Der zugrunde liegende Produktname, Kundenbezüge, öffentliche IP-Adressen, Domains, Hostnamen und die Geschäftsstrategie wurden bewusst entfernt. Gezeigt wird ausschließlich die eigene Infrastruktur-Netzarchitektur. Die internen RFC-1918-Adressräume sind generisch gehalten (z.B. `10.0.x` / internes `/29`), da das Segmentierungs-Schema die Architektur sichtbar macht, ohne konkrete Hosts preiszugeben.

Die Plattform, deren Infrastruktur hier dokumentiert ist, ist eine Multi-Tenant-SaaS-Plattform (Produkt anonymisiert). Status: in Betrieb / Eigenbetrieb (Pre-Launch).

1. Executive Summary

Architektur-Prinzipien

Die Netzwerkinfrastruktur folgt einer **Zero-Trust-Architektur** mit sechs unabhängigen Schutzschichten (Defense-in-Depth). Die Kompromittierung einer Schicht bedeutet nicht die Kompromittierung des Gesamtsystems.

1. **Firewall-Kaskade (2 Tiers)** — eine zweistufige Kette Edge-FW (Tier 1, Perimeter, OPNsense) → Core-FW (Tier 2, Policy-Enforcement, pfSense), getrennt aufgebaut für LAN und für DMZ. Zwei Tiers, nicht mehr.
2. **PVE-Host-Firewall (Hypervisor-Schutz)** — `firewalld` direkt auf dem Proxmox-Host (nicht die VM-Firewall der Gäste), Default-Zone `public` = Default-Deny, öffentlich nur WireGuard-UDP offen. Management (SSH, Proxmox-UI) ausschließlich über den Tunnel (`firewalld` -Zone `trusted`).
3. **VLAN-Mikro-Segmentierung (Segmentierung)** — die meisten VLANs sind eine eigene Sicherheitszone mit `/29` -Subnetz (User-/DEV-Netze `/24`), Default-Deny, EINE Hypervisor-Bridge je Transit, kein Layer-2-Lateral-Movement.
4. **Management-Plane-Isolation (Isolation)** — Out-of-Band, kein Routing-Pfad von der Management-Ebene zu den Produktiv-VLANs.
5. **VPN-Zwang (Zugang)** — WireGuard / IPsec, ohne Tunnel kein Zugriff auf Management oder interne Zonen.
6. **Key-Authentifizierung (Auth)** — SSH- und Hypervisor-Zugang nur per Public Key.

Einbahn-Vertrauen (zentrales Prinzip)

Der Proxmox-Host **hostet** die Firewall-VMs, **verlässt sich für seinen eigenen Schutz aber nicht auf sie**. Der Host hat eine eigene Host-Firewall (`firewalld`) und ein eigenes Out-of-Band-Management. Fällt eine Firewall-VM aus oder wird sie kompromittiert, bleibt der Host dicht. Die Management-VPN-Verbindung terminiert **am Host selbst** und nicht über die Firewall-VMs. Vertrauen fließt nur in eine Richtung: Host → schützt sich selbst, unabhängig von den von ihm gehosteten Firewalls.

Quick Facts

Kennzahl	Wert
Plattform	Bare-Metal Dedicated-Server (Hetzner), Proxmox VE 9.1 als Hypervisor
Laufende Guests	35+ (VMs + LXC)
Standort	Deutschland (Standort anonymisiert)
Remote-Standort	zweiter Standort, via Site-to-Site-VPN (IPsec)
Public IPv4-Adressen	4 gesamt: 3 am Host (je eigene Hetzner-MAC: PVE-Host OOB, Edge-LAN, Edge-DMZ) + 1 am separaten Cloud-VPS-Shield
Firewall-Tiers	2 (Edge → Core), als getrennte Ketten für LAN und DMZ
Firewall-VMs	4 (Edge-FW LAN/DMZ = OPNsense, Core-FW LAN/DMZ = pfSense)
Host-Firewall	`firewalld` direkt auf dem Proxmox-Host (Default-Deny, nur WireGuard-UDP offen)
LAN-VLANs	50+ (Tabelle unten = Auszug), überwiegend `/29` , Default-Deny
DMZ-VLANs	minimal (Edge-Dienste, Ingress-Gateway), 48 Filter-Regeln auf der Core-DMZ-FW

Kennzahl	Wert
Cloud-VPS-Shield	separater Provider-Standort, eigene Public-IP, Reverse-Ingress per WireGuard zur Core-DMZ
Secret-Management	HashiCorp Vault
IaC	Ansible
Admin-VPN (LAN)	WireGuard, terminiert auf der Core-LAN-FW (inkl. Port 443 für restriktive Netze)
Management-VPN	WireGuard, terminiert am Proxmox-Host (umgeht die Firewall-VMs)
Site-to-Site-VPN	IPsec zum Remote-Standort
Datenbank-Sicherheit	PostgreSQL mit Row-Level-Security (RLS)
Security-Monitoring	Wazuh-SIEM, Graylog, Prometheus / Grafana / Loki
Out-of-Band-Management	Provider-KVM + separater Host-Management-Kanal über Management-VPN

2. Die zweistufige Firewall-Kaskade in zwei getrennten Ketten

Begriffsklärung: 2 Firewall-Tiers, 6 Schutzschichten

Die **Firewall-Kaskade ist zweistufig** — Edge-FW (Tier 1) → Core-FW (Tier 2). Das ist bewusst so gehalten: ein technischer Reviewer soll genau zwei Firewall-Tiers vorfinden und nicht eine erfundene Zahl. Die **„6"** bezieht sich auf die sechs unterschiedlichen Schutzmechanismen aus Abschnitt 1 (Defense-in-Depth), nicht auf Firewall-Stufen.

Wichtiges Prinzip: zwei unabhängige Uplink-Ketten

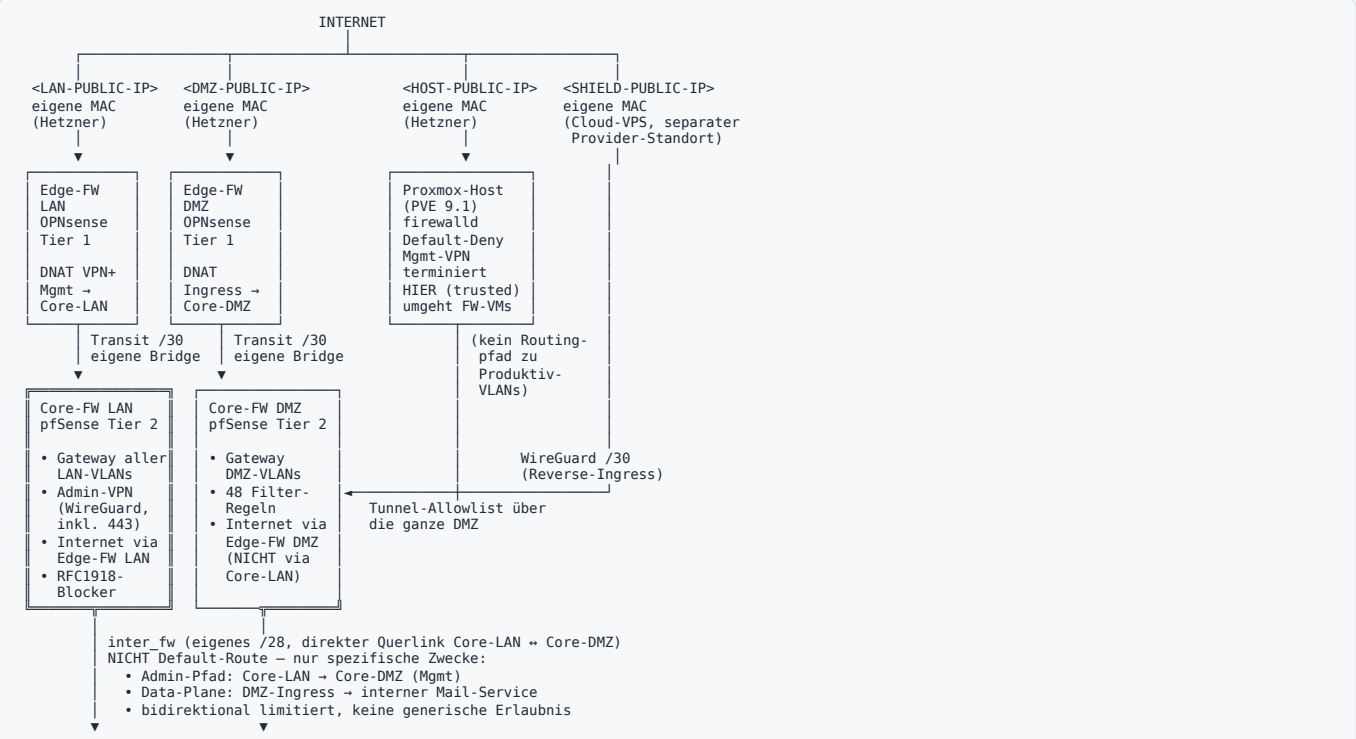
LAN und DMZ haben jeweils eine eigene Edge-Firewall mit eigener Public-IP und eigener Hypervisor-Bridge. Der Internet-Zugang jeder Zone kommt ausschließlich aus der eigenen Kette. Die Core-LAN-FW ist **nicht** der Internet-Durchgang für die DMZ. Zwischen beiden Core-FWs existiert nur ein direkter Querlink (`inter_fw` , eigenes `/28`) für spezifischen Datenaustausch (Admin-Pfad, Mail-Stream zum internen Mail-Service).

Firewall-Rollen-Konvention

Durchgängig verwendet in dieser Dokumentation (Hostnamen anonymisiert, Rollen generisch):

Rolle	Tier	Produkt	Zone
Edge-FW LAN	Tier 1 (Perimeter)	OPNsense	LAN-Kette
Core-FW LAN	Tier 2 (Policy-Enforcement, Admin-Hub)	pfSense	LAN-Kette
Edge-FW DMZ	Tier 1 (Perimeter)	OPNsense	DMZ-Kette
Core-FW DMZ	Tier 2 (Policy-Enforcement)	pfSense	DMZ-Kette

Topologie-Diagramm



```
    LAN-VLANs           DMZ-VLANs
   (50+, überw. /29)   (Edge-Dienste,
   Default-Deny         Ingress-Gateway)


Cloud-VPS-Shield (separater Provider, eigene Public-IP):
   • 2. öffentlicher Eingang in die DMZ (Redundanz zur Edge-DMZ-Kette)
   • hängt am Hetzner vSwitch (privates L2, 10.10.0.0/16)
   • tunnelt per WireGuard /30 zur Core-DMZ, Tunnel-Allowlist über die DMZ
   • Reverse-Proxy-Shield: ip_forward=0, kein L3-DNAT → sauberer L7-Shield
   • Live: Postfix als ausgehender Smarthost (intern → Internet, an Tunnel-IP gebunden)
   • Eingehender MX (Internet → intern): geplant — noch nachzuziehen
   • HAProxy TLS-gehärtet (TLS 1.2+) installiert, Web-Publishing vorbereitet

Proxmox-Host ist KEIN Produktiv-Datenpfad — eigenständiger Management-Kanal:
   • Eigene Host-Firewall (firewalld), Default-Zone public = Default-Deny
   • Management-VPN (WireGuard) terminiert AM HOST (firewalld-Zone trusted)
   • Umgeht die Firewall-VMs vollständig (Einbahn-Vertrauen)
   • Kein Routing-Pfad zu den Produktiv-VLANs
   • Kann VMs verwalten (Start/Stop/Migrate), nicht in ihre Produktiv-Netze
   • Notzugriff: Provider-KVM-Konsole (bei komplettem Ausfall)
```

Kernaussagen zur Topologie

1. **Zwei parallele, unabhängige Internet-Uplinks** — die LAN-Kette und die DMZ-Kette haben jeweils eine eigene Public-IP (mit eigener MAC) und eine eigene Edge-Firewall. Keine Zone bekommt Internet durch die andere Zone.
2. **`inter_fw` (Querlink, eigenes `/28`) ist kein Uplink.** Er existiert ausschließlich für: - Admin-Zugriff von der Core-LAN-FW auf die Core-DMZ-FW als Management-Pfad - Data-Plane für den eingehenden Mail-Stream: DMZ-Ingress → interner Mail-Service (internes `/29`) - Kein generischer DMZ↔LAN-Durchgang, jede Regel einzeln explizit.
3. **Sternförmige Management-Topologie** um die Core-LAN-FW als zentralen Admin-Hub. Sie kann jede andere Firewall administrieren, ohne Produktiv-Transits oder Public-IPs zu nutzen. Admin-Traffic bleibt strikt vom Produktiv-Traffic getrennt.
4. **Ausfallverhalten:** Fällt die Edge-FW LAN aus, ist das LAN vom Internet getrennt, aber die DMZ läuft weiter (und umgekehrt). Kein gemeinsamer Single-Point-of-Failure am Uplink. Zusätzlich bildet der Cloud-VPS-Shield einen zweiten, redundanten öffentlichen DMZ-Eingang.
5. **Der Proxmox-Host ist komplett vom Produktiv-Netz isoliert (Einbahn-Vertrauen)** — eigene Host-Firewall (`firewalld`), eigener Management-VPN, der am Host selbst terminiert und die Firewall-VMs umgeht. Der Host hat weder einen Routing-Pfad zu den Produktiv-VLANs noch verlässt er sich für seinen eigenen Schutz auf die von ihm gehosteten Firewalls. Bei einem Firewall-VM-Kompromiss bleibt der Host dicht.

Die zwei Tiers und vier Firewall-VMs im Detail

#	Komponente	Rolle	Public-IP	Instanz-Typ
T1-LAN	**Edge-FW LAN — OPNsense**	Tier-1-Perimeter für **LAN-Traffic**, Destination-NAT für VPN/Mgmt zur Core-LAN-FW, WAN-Bogons-/Private-Block, **Internet-Uplink der LAN-Kette**	`<LAN-PUBLIC-IP>`	KVM-VM, eigene MAC (Hetzner)
T2-LAN	**Core-FW LAN — pfSense**	Tier-2-Policy-Enforcement, Gateway für alle LAN-VLANs, terminiert den Admin-VPN, bekommt Internet ausschließlich über die Edge-FW LAN, zentraler Admin-Hub	intern RFC1918 (`/29`)	KVM-VM (interne NICs: Proxmox-Auto-MAC)
T1-DMZ	**Edge-FW DMZ — OPNsense**	Tier-1-Perimeter für **DMZ-Traffic**, Destination-NAT zum Ingress-Gateway, **Internet-Uplink der DMZ-Kette (unabhängig vom LAN)**	`<DMZ-PUBLIC-IP>`	KVM-VM, eigene MAC (Hetzner)
T2-DMZ	**Core-FW DMZ — pfSense**	Tier-2-Policy-Enforcement, Gateway für DMZ-VLANs, 48 Filter-Regeln, bekommt Internet ausschließlich über die Edge-FW DMZ, administrierbar von der Core-LAN-FW via `inter_fw`	intern RFC1918 (`/29`)	KVM-VM (interne NICs: Proxmox-Auto-MAC)

Die zugehörige **Host-Firewall (`firewalld`)** und das **Out-of-Band-Management** sind kein Firewall-Tier, sondern eigene Schutzschichten (Layer 2 + Layer 4 aus Abschnitt 1) und werden bewusst getrennt geführt — siehe Einbahn-Vertrauen.

---

3. Provider-Infrastruktur

Dedicated Server

Feld	Wert
Plattform	Bare-Metal Dedicated-Server (Hetzner)
Standort	Deutschland (Standort anonymisiert)
Hypervisor	Proxmox VE 9.1
Guests	35+ laufende VMs und LXC-Container

Feld	Wert
CPU-Features	VT-x + Nested Virtualization aktiviert

Public-IP-Zuordnung mit eigener MAC pro IP

Der Hoster vergibt jede zusätzliche IP über eine eigene MAC-Adresse (Hetzner „zusätzliche IP mit eigener MAC", Anti-Spoofing auf Upstream-Router-Ebene). Beim Erstellen der VM wird die MAC am vNIC fest eingetragen. Public-NICs tragen Hetzner-MACs, interne NICs Proxmox-Auto-MACs.

Public-IP	MAC-Zuweisung	Zugewiesen an
`<HOST-PUBLIC-IP>`	eigene MAC (Hetzner)	Proxmox-Host (Out-of-Band-Management)
`<LAN-PUBLIC-IP>`	eigene MAC (Hetzner)	Edge-FW LAN (OPNsense)
`<DMZ-PUBLIC-IP>`	eigene MAC (Hetzner)	Edge-FW DMZ (OPNsense)
`<SHIELD-PUBLIC-IP>`	eigene MAC (separater Provider)	Cloud-VPS-Shield (Reverse-Ingress)

Sicherheits-Argument

Das MAC-gebundene IP-Filtering ist **eine vorgelagerte Verteidigungsschicht vor der eigenen Firewall**. Ohne die zugewiesene MAC kommt ein Paket gar nicht zur VM. Das verhindert:

- **IP-Spoofing** durch andere Kunden im selben Provider-Subnetz
- **MAC-Flooding** im Provider-Backbone
- **Failover-Angriffe** bei IP-Übernahmen ohne Kontroll-Wechsel

---

4. Hypervisor-Bridge-Layout

Bridge-Konzept: eine Bridge pro Vertrauenszone / Transit / Admin-Pfad

Kein Shared-Netz zwischen Firewalls. Jeder Firewall-zu-Firewall-Link ist eine **eigene Hypervisor-Bridge**. Zusätzlich existieren **dedizierte Admin-Bridges** zwischen der Core-LAN-FW und den anderen Firewalls für den Management-Zugriff, getrennt vom Produktiv-Transit.

WAN-Anbindung + physisch

Bridge	Typ	VLAN-aware	Ports / CIDR	Zweck
`eno1`	Physical NIC	—	—	Provider-Uplink (Haupt-NIC)
`vmbr0`	Linux Bridge	**Yes**	Port `eno1`	WAN-Bridge, trägt alle Public-IPs über MAC-Bindung zu den Edge-VMs

Rescue-/Wartungs-Bridges (dedizierter „Service-Port" pro Firewall)

Jede Firewall hat eine dedizierte **Rescue-Bridge**, an die im Notfall eine Wartungs-VM direkt angeschlossen werden kann. Analog zum physischen Setup, bei dem man bei einem Netzwerk-Problem einen Laptop direkt an den Switch/Router hängt, um Zugriff zu bekommen. Wenn der Produktiv-Transit ausfällt (Regel-Fehlkonfiguration, Interface-Down etc.), ist über die Rescue-Bridge weiterhin ein direkter Zugang möglich — ohne durch die Produktiv-Kette routen zu müssen.

Bridge	Typ	Zweck
`edge_lan`	Linux Bridge	Rescue-Port für die Edge-FW LAN — Wartungs-VM anschließbar, wenn der normale Transit defekt ist
`edge_dmz`	Linux Bridge	Rescue-Port für die Edge-FW DMZ
`core_lan`	Linux Bridge	Rescue-Port für die Core-FW LAN
`core_dmz`	Linux Bridge	Rescue-Port für die Core-FW DMZ

Im Normalbetrieb: Die Rescue-Bridges haben keine aktiven Mitglieder (keine VM angeschlossen), sind aber immer aktiv und bereit.

Im Notfall: Der Admin erstellt kurzfristig eine Wartungs-VM (z.B. Debian-Live), hängt sie an die Rescue-Bridge der betroffenen Firewall, vergibt manuell eine IP aus demselben Subnetz und hat direkten Layer-2-Zugriff auf die gestörte Firewall — völlig unabhängig vom Produktiv-Traffic.

Parallele zum physischen Setup: wie ein „Console-Port" oder „Out-of-Band-Management-Port" an Enterprise-Switches (Cisco IOS-Console, Juniper-Management-Ethernet). Funktioniert auch, wenn der produktive Datenpfad down ist.

Haupt-Transit (Edge → Core, Produktiv-Traffic)

Bridge	Typ	Mitglieder	Zweck	Transit-Net
`br_lan01`	Linux Bridge	Edge-FW LAN + Core-FW LAN	Transit **Edge LAN ↔ Core LAN**	eigenes `/30`

Bridge	Typ	Mitglieder	Zweck	Transit-Net
`br_dmz01`	Linux Bridge	Edge-FW DMZ + Core-FW DMZ	Transit **Edge DMZ ↔ Core DMZ**	eigenes `/30`

Interne VLAN-Trunks

Bridge	Typ	VLAN-aware	Zweck
`vmbr2_VLAN`	Linux Bridge	**Yes** (802.1Q-Trunk)	LAN-VLAN-Trunk — die Core-FW LAN routet, alle LAN-Service-VMs hängen hier mit VLAN-Tag
`dmz_vlan01`	Linux Bridge	**Yes** (802.1Q-Trunk)	DMZ-VLAN-Trunk — die Core-FW DMZ routet, die DMZ-Ingress-VM hängt hier mit VLAN-Tag

Cross-Zone- + Admin-Bridges (sternförmig um die Core-LAN-FW als Admin-Hub)

Bridge	Typ	Mitglieder	Zweck
`inter_fw`	Linux Bridge	Core-FW LAN ↔ Core-FW DMZ	**Querlink** (eigenes `/28`) für Admin-Pfad + Mail-Stream (DMZ-Ingress → interner Mail-Service)
`lan_opn`	Linux Bridge	Core-FW LAN ↔ Edge-FW LAN (internes Interface)	**Admin-Bridge**: die Core-LAN-FW administriert die Edge-FW-LAN-GUI/SSH, ohne über den Produktiv-Transit zu laufen
`dmz_opn`	Linux Bridge	Core-FW LAN ↔ Edge-FW DMZ (internes Interface)	**Admin-Bridge**: die Core-LAN-FW administriert die Edge-FW-DMZ-GUI/SSH analog

Warum die separaten Admin-Bridges (`lan_opn`, `dmz_opn`, `inter_fw`)

Die Core-FW LAN fungiert als **zentraler Admin-Hub**. Damit ein Admin nicht jede Edge-Firewall über deren Produktiv-Transit oder gar über die Public-IP administrieren muss, hat jede Edge-Firewall eine **dedizierte Admin-NIC**, die nur mit der Core-LAN-FW verbunden ist:

```
Core-FW LAN (Admin-Hub)
  ├── via lan_opn   ──► Edge-FW LAN (OPNsense)   (GUI/SSH)
  ├── via dmz_opn   ──► Edge-FW DMZ (OPNsense)   (GUI/SSH)
  └── via inter_fw  ──► Core-FW DMZ (pfSense)    (GUI/SSH + Mail-Stream)
```

Das hat drei Vorteile:

1. **Admin-Traffic ist komplett vom Produktiv-Traffic getrennt** — bricht der Transit zusammen (z.B. Config-Fehler), kommt der Admin trotzdem drauf.
2. **Produktiv-Regelsätze auf den Transits brauchen keine Mgmt-Ausnahmen** — sauber trennbar.
3. **Kleinere Angriffsflächen** — die Admin-Bridges sind vollständig interne Hypervisor-Bridges, nicht von außen routbar, und je eine zwischen genau zwei VMs (Punkt-zu-Punkt).

vNIC-Zuordnung der Firewall-VMs (Bridge-Mapping)

VM	vNICs / Bridges
Edge-FW LAN	`vmbr0` (WAN via MAC), `br_lan01` (Transit), `lan_opn` (Admin), `edge_lan` (Rescue)
Core-FW LAN	`br_lan01` (Transit), `vmbr2_VLAN` (LAN-Trunk), `inter_fw` (Querlink), `lan_opn` + `dmz_opn` (Admin), `core_lan` (Rescue)
Edge-FW DMZ	`vmbr0` (WAN via MAC), `br_dmz01` (Transit), `dmz_opn` (Admin), `edge_dmz` (Rescue)
Core-FW DMZ	`br_dmz01` (Transit), `dmz_vlan01` (DMZ-Trunk), `inter_fw` (Querlink), `core_dmz` (Rescue)

Hinweis zur Interpretation der Adressierung

Gateway-IPs enden stets auf `.1` und sind Router-Interfaces auf der jeweiligen Core-FW — dort läuft kein Gerät, das ist nur das Routing-Gateway. Hosts haben höhere IPs (`.2`, `.3`, `.4` …). Konkrete Host-IPs sind in dieser Arbeitsprobe generisch gehalten.

---

5. VLAN-Struktur

5.1 LAN-Zone (hinter der Core-FW LAN)

Alle VLANs liegen hinter der Core-FW LAN und sind durch Zonen-Regeln streng segmentiert (50+ VLANs, überwiegend `/29`, Default-Deny). Subnetz-Konvention: `/29` für 6 Hosts (ausreichend für Microservices), `/24` für die vier größeren Zonen (DEV, SIEM, Chat, Search), `/28` für ADMIN, `/30` für das Quarantäne-VLAN. Konkrete Subnetze sind generisch dargestellt (`10.0.x`); das VLAN-Schema bleibt sichtbar. (Die folgende Tabelle ist ein repräsentativer Auszug; real 50+ VLANs.)

Status-Legende: ✓ aktiv deployed · ◯ reserviert (VLAN existiert, keine Services) · ⚙ geplant (Roadmap)

Base

Name	Subnetz	Zweck	Status
LAN	`10.0.x/24`	Standard-LAN (Clients, IoT-Basis)	✓

MGMT (Management-Infrastruktur)

Name	Subnetz	Zweck	Status
MGMT	internes `/29`	Hypervisor-/Proxmox-Management (interne Mgmt-IP)	✓
MGMT_VAULT	internes `/29`	HashiCorp Vault (Secret-Manager)	✓
MGMT_BACKUP	internes `/29`	Backup-Server	⚙ geplant
MGMT_VAULT_SEAL	internes `/29`	Key-Custodian für Vault-Auto-Unseal (Compromise-Isolation getrennt von Vault, vTPM-Pflicht)	✓

ADMIN (Admin-Workstations + Dev-Tools)

Name	Subnetz	Zweck	Status
ADMIN	internes `/28`	Admin-Tools: Web-Terminal, Remote-Desktop-Gateway, Remote-Management (Bastion)	✓
ADMIN_IAC	internes `/29`	Ansible (IaC)	⚙ geplant
ADMIN_GIT	internes `/29`	interner Git-Server	⚙ geplant
ADMIN_CI	internes `/29`	CI/CD-Runner	⚙ geplant
ADMIN_REGISTRY	internes `/29`	Container-Registry	⚙ geplant

SVC (interne Services)

Name	Subnetz	Zweck	Status
SVC_PROXY	internes `/29`	interner Reverse-Proxy	✓
SVC_MAIL	internes `/29`	interner Mail-Service (SMTP/IMAP intern)	✓
SVC_QUEUE	internes `/29`	Message-Queue	⚙ geplant
SVC_DNS	internes `/29`	Clients-DNS (Split-Horizon-Resolver + Ad-Blocking)	✓
SVC_AUTH_AD	internes `/29`	AD-Domain-Controller (isoliert), Backend für Verzeichnis-Auth	✓
SVC_DNS_SERVICES	internes `/29`	Services-DNS (Container/VMs), getrennt vom Clients-Resolver gegen Hairpin	✓
SVC_PROXY_INT	internes `/29`	interner Routing-Proxy (Default-Deny zu allen anderen LAN-Hosts)	✓
SVC_CERT	internes `/29`	ACME-Client für Wildcard-Zertifikate via DNS-01, Push-Pattern (kein Inbound)	✓
SVC_MAIL_TIER	internes `/29`	dedizierte Mail-Foundation (Postfix + Dovecot + DKIM + Spam-Filter), Defense-Layering	⚙ geplant
SVC_MAIL_ARCHIVE	internes `/29`	revisionssicheres Mail-Archiv (WORM, mTLS + RBAC), Compromise-Containment getrennt vom Mail-Service	⚙ geplant

DB (Datenbanken — nie direkt öffentlich)

Name	Subnetz	Zweck	Status
DB_POSTGRESQL	internes `/29`	PostgreSQL (zentral, interne Apps), Row-Level-Security	✓

Name	Subnetz	Zweck	Status
DB_MYSQL	internes `/29`	MySQL/MariaDB	◯ reserviert
DB_MONGODB	internes `/29`	MongoDB	◯ reserviert
DB_REDIS	internes `/29`	Redis Cache/Session-Store (native VM, kein Container)	✓

APP (Application-Tier)

Name	Subnetz	Zweck	Status
APP_WEB	internes `/29`	Web-Frontend / Landing	✓
APP_API	internes `/29`	Backend-API (NestJS)	✓
APP_CONTAINER	internes `/29`	Docker-Host (Microservices via macvlan)	✓
APP_WORKER	internes `/29`	Background-Worker	✓
APP_CORE	internes `/29`	zentraler Core-Service	✓

STOR (Storage-Systeme)

Name	Subnetz	Zweck	Status
STOR_DATA	internes `/29`	Primär-Storage (NFS/iSCSI)	⚙ geplant
STOR_BACKUP	internes `/29`	Backup-Storage (Borg/Restic)	⚙ geplant
STOR_S3	internes `/29`	S3-kompatibler Object-Store (WORM für Mail-Archiv)	⚙ geplant

DEV / STAGE / SANDBOX

Name	Subnetz	Zweck	Status
DEV	`10.0.x/24`	Development-Umgebung (inkl. Pentest-VM)	✓
STAGE	internes `/29`	Staging	⚙ geplant
SANDBOX	internes `/29`	echt isolierte Sandbox (kein Routing-Pfad zu Produktiv-VLANs)	✓

MON / SEC

Name	Subnetz	Zweck	Status
MON	internes `/29`	Infrastructure-Metriken (Prometheus / Grafana / Loki)	✓
SEC_SIEM	internes `/24`	SIEM-Stack (Wazuh: Manager, Indexer, Dashboard, Fileserver, Client) — größerer Host-Bedarf	✓
SEC_LOGS	internes `/29`	Log-Aggregation (Graylog)	✓

Special

Name	Subnetz	Zweck	Status
VPN_ADMIN	internes `/29`	Admin-VPN-Subnetz (Core-LAN-FW-VPN terminiert hier)	✓
TRANSIT	internes `/30`	internes Transit	◯ reserviert
QUARANTINE	internes `/29`	eigenes Quarantäne-VLAN (Blackhole, kein Routing-Pfad)	✓

VPN-Tunnel-Endpunkte

Endpunkt	Terminierung	Zweck
Management-VPN	Proxmox-Host (`firewalld` -Zone `trusted`)	Host-OOB-Management, umgeht die Firewall-VMs
Admin-VPN (LAN)	Core-FW LAN (WireGuard, inkl. Port 443)	Admin-Zugriff auf LAN-VLANs
Site-to-Site	Core-FW LAN (IPsec)	Anbindung Remote-Standort

5.2 DMZ-Zone

Die DMZ wird bewusst minimal gehalten: nur Edge-Dienste und ein Ingress-Gateway. Sie ist sauber von der LAN-Zone getrennt (eigenes Adress-Schema). Auf der Core-FW DMZ greifen **48 Filter-Regeln**. Das Publishing der DMZ erfolgt über die Edge-DMZ-Kette und/oder über den Cloud-VPS-Shield.

Name	Zweck	Status
DMZ_INGRESS	Ingress-Gateway / Edge-Dienste der DMZ	✓
DMZ_MAIL	eingehender Mail-Empfang (hinter Reverse-/Stream-Pfad)	✓
DMZ_PROXY	Reverse-/Stream-Pfad für Mail-Ports zum internen Mail-Service	⚙ geplant

Warum die DMZ minimal bleibt: Zero-Trust-Design. Web-Publishing läuft über den Cloud-VPS-Shield als Reverse-Ingress (siehe Abschnitt 9), Mail-Protokolle (SMTP/IMAP/POP3) über einen minimalen Reverse-/Stream-Pfad. Jede in der DMZ exponierte Funktion ist explizit begrenzt; alles andere ist Default-Deny.

5.3 Transit- und Management-Netze

Name	Subnetz	Peers	Zweck
`br_lan01`	eigenes `/30`	Edge-FW LAN ↔ Core-FW LAN	Transit Edge LAN ↔ Core LAN (eigene Bridge)
`br_dmz01`	eigenes `/30`	Edge-FW DMZ ↔ Core-FW DMZ	Transit Edge DMZ ↔ Core DMZ (eigene Bridge)
`inter_fw`	eigenes `/28`	Core-FW LAN ↔ Core-FW DMZ	direkter Querlink (Admin-Pfad + Mail-Stream)
Shield-Tunnel	eigenes `/30`	Cloud-VPS-Shield ↔ Core-FW DMZ	WireGuard-Reverse-Ingress, Tunnel-Allowlist über die DMZ
vSwitch	`10.10.0.0/16`	Cloud-VPS-Shield (privates L2)	Hetzner vSwitch (Layer-2-Anbindung des Shields)

---

6. Firewall-Regel-Patterns

6.1 Transit-Regelset (einheitliches Pattern auf allen Firewall-Transits)

Jedes Firewall-zu-Firewall-Interface trägt dasselbe Grundset (Referenz-Implementation auf `br_lan01`):

#	Action	Direction	Protocol	Source	Destination	Port	Zweck
00	block	in	any	bogons	any	any	Anti-Spoofing: Bogons-Netze blockieren
01	block	in	any	NOT PeerIP	TransitNet	any	Anti-Spoofing: nur Peer-IP darf Source sein
10	pass	in	icmp	PeerIP	any	—	Ping/MTU-Discovery
20	pass	in	udp	PeerIP	any	53	DNS UDP
21	pass	in	tcp	PeerIP	any	53	DNS TCP
22	pass	in	udp	PeerIP	any	123	NTP
30	pass	in	tcp	PeerIP	any	80, 443	HTTP/HTTPS (Updates, Cert-Renewal)
40	pass	in	udp	PeerIP	any	VPN_ALL_UDP_PORTS	VPN-Antworten (IPsec + WG)
41	pass	in	esp	PeerIP	any	—	IPsec ESP
50	pass	in	tcp	PeerIP	TransitNet	OPN_MGMT_PORTS	Management (GUI/SSH vom Peer)
90	pass	in	tcp	PeerIP	any	any	Default-Allow TCP outbound (Internet-Zugriff)
99	block	in	any	any	RFC1918	any	Catch-All: keine Bridge-Hops ins RFC1918
100	block	in	any	any	any	any	Default-Deny (explizit, mit

#	Action	Direction	Protocol	Source	Destination	Port	Zweck
							Logging)

6.2 OPNsense Aliases (intern + User-definiert)

Bestehend (OPNsense-intern): - `bogons` — IPv4-Bogons (ohne RFC1918, automatisch gepflegt) - `__opt1_network` — automatisch = `br_lan01` - Transit-Net

User-definiert: - `CORE_FW_LAN` — Host-IP der Core-FW LAN im Transit - `VPN_IPSEC_PORTS` — Port 500, 4500 - `VPN_WG_STANDARD` — Port 51820 - `VPN_WG_CUSTOM` — mehrere verschleierte Custom-Ports (inkl. 443 für restriktive Netze) - `VPN_ALL_UDP_PORTS` — alle VPN-UDP-Ports gebündelt - `WEB_OUTBOUND_PORTS` — Port 80, 443 - `RFC1918` — 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16 - `OPN_MGMT_PORTS` — Port 22, 443

6.3 Destination-NAT auf der Edge-FW LAN (WAN)

Forward	Protocol	Dst-Port	Target	Target-Port	Zweck
1	UDP	500, 4500	CORE_FW_LAN	500, 4500	IPsec IKE + NAT-T
2	UDP	51820	CORE_FW_LAN	51820	WireGuard Default
3	UDP/TCP	443 + Custom-Range	CORE_FW_LAN	443 + Custom-Range	WireGuard Custom (443 für restriktive Netze)
4	ESP	—	CORE_FW_LAN	—	IPsec ESP (Nicht-NAT-T-Fallback)

6.4 Destination-NAT auf der Edge-FW DMZ (WAN)

Forward	Protocol	Dst-Port	Target	Target-Port	Zweck
1	TCP	25	DMZ-Ingress	25	SMTP Incoming
2	TCP	465	DMZ-Ingress	465	SMTP Submission TLS
3	TCP	587	DMZ-Ingress	587	SMTP Submission STARTTLS
4	TCP	993	DMZ-Ingress	993	IMAP SSL
5	TCP	995	DMZ-Ingress	995	POP3 SSL

Jeder Forward mit „Register rule" (OPNsense erzeugt die WAN-Pass-Regel automatisch).

---

7. Asymmetrische Trust-Policy

Prinzip

Die Core-FW LAN darf alles nach außen initiieren, aber niemand darf zurück zu ihr initiieren. Verbindungen laufen stateful — Antworten kommen automatisch zurück, aber keine eigenständigen Verbindungen von „außen nach innen". Dasselbe Einbahn-Prinzip gilt zwischen Host und Firewall-VMs: Der Host schützt sich selbst und ist nicht auf die Firewalls angewiesen.

Trust-Matrix

Von → Nach	Erlaubt?	Begründung
Internet → Edge-FW LAN	Nur dedizierte Ports (DNAT zu VPN/Mgmt)	Minimaler Angriffsvektor
Internet → Edge-FW DMZ	Nur dedizierte Ports (DNAT zu Ingress)	Minimaler Angriffsvektor
Edge-FW LAN → Core-FW LAN	Nein (Default-Deny)	Edge-Kompromiss isolieren
Edge-FW DMZ → Core-FW DMZ	Nein (außer via State)	Edge-Kompromiss isolieren
Core-FW LAN → Edge-FW LAN	Ja (Internet-Zugang LAN-Kette, Mgmt)	State-basierter Rückverkehr
Core-FW DMZ → Edge-FW DMZ	Ja (Internet-Zugang DMZ-Kette, Mgmt)	State-basierter Rückverkehr, unabhängig vom LAN
Core-FW LAN → Core-FW DMZ (via inter_fw)	Ja (Admin-Pfad, spezifische Mgmt-Ports)	Querlink für Admin-Zugriff
Core-FW DMZ → Core-FW LAN (via inter_fw)	Nein (Default-Deny)	DMZ darf nicht zurück ins LAN initiieren
DMZ-Ingress → interner Mail-Service (via inter_fw)	**Ja** (minimal, nur Mail-Ports)	Einzige zugelassene Cross-Zone-Data-Plane-Regel
interner Mail-Service → Internet	Ja (outbound Relay)	Mail-Versand
Cloud-VPS-Shield → Core-FW DMZ (WireGuard)	Ja (Reverse-Ingress, Tunnel-Allowlist über DMZ)	2. öffentlicher DMZ-Eingang
Core-FW DMZ → Cloud-VPS-Shield	Ja (State-Rückverkehr im Tunnel)	Reverse-Proxy-Antworten
Admin-VPN → Core-FW LAN	Ja (Public-Key + restriktive Allowlist)	Admin-Einstieg ins LAN
Management-VPN → Proxmox-Host	Ja (`firewalld` -Zone `trusted` , Public-Key)	OOB-Management, umgeht die FW-VMs

Von → Nach	Erlaubt?	Begründung
Remote-Standort → LAN (S2S-VPN)	Ja (spezifisch)	Domain-Join, Verzeichnis, Storage-Access
Remote-Standort → DMZ	Nein	DMZ bleibt intern-only
Proxmox-Host → Produktiv-VLANs	**Kein Routing-Pfad vorhanden**	Host ist out-of-band; kann VMs nur lifecycle-steuern, nicht in ihre Netze
Firewall-VM → Proxmox-Host	**Kein Vertrauenspfad**	Einbahn-Vertrauen: der Host verlässt sich nicht auf die von ihm gehosteten Firewalls

Zone-Access-Matrix (Admin-VPN)

Pro VPN-Client-Gruppe definiert die Core-FW LAN, welche LAN-VLANs erreichbar sind:

Admin-Gruppe	Zugriff auf
`admin-full`	Alle LAN-VLANs + DMZ
`admin-dev`	DEV, STAGE, ADMIN_GIT, ADMIN_CI, APP_API
`admin-ops`	MON, SEC_SIEM, SEC_LOGS, MGMT, MGMT_VAULT, MGMT_BACKUP
`admin-mail`	SVC_MAIL, DMZ_PROXY

---

8. VPN-Pfade

Die Architektur kennt **zwei getrennte VPN-Pfade** mit unterschiedlichen Terminierungspunkten — das ist Teil des Einbahn-Vertrauens.

Management-VPN — terminiert am Proxmox-Host

- **WireGuard**, terminiert direkt auf dem Proxmox-Host (`firewalld` -Zone `trusted`).
- **Umgeht die Firewall-VMs vollständig.** Öffentlich ist am Host nur WireGuard-UDP offen; SSH und Proxmox-UI sind ausschließlich über den Tunnel erreichbar.
- Zweck: Host-Out-of-Band-Management (VM-Lifecycle: Start/Stop/Migrate). Kein Routing-Pfad zu den Produktiv-VLANs.

Admin-VPN (LAN) — terminiert auf der Core-FW LAN

- **WireGuard** (Standard-Port + verschleierte Custom-Ports, inkl. **Port 443** für restriktive Netze), zusätzlich IPsec.
- Verbindung kommt über die Edge-FW LAN (Destination-NAT) zur Core-FW LAN.
- Der Admin-Client bekommt eine IP aus `VPN_ADMIN` ; eine Zone-Access-Matrix steuert je Gruppe die erreichbaren LAN-VLANs.

```
Admin-Client (Internet)
  | UDP WireGuard (51820 / Custom / 443) bzw. IPsec
  ▼
Edge-FW LAN (OPNsense, Tier 1) — Destination-NAT
  | Transit /30 (eigene Bridge)
  ▼
Core-FW LAN (pfSense, Tier 2) — VPN-Terminator
  | Client-IP aus VPN_ADMIN, Policy-Routing je Gruppe
  ▼
Freigegebene LAN-VLANs (per Zone-Access-Matrix)
```

Authentifizierung

- WireGuard: Public-Key-Auth (pro Client ein eigener Key).
- SSH-/Hypervisor-Zugang ausschließlich per Public Key (Schutzschicht 6).
- Logging aller VPN-Sessions an den SIEM-Stack (Wazuh).

---

9. Cloud-VPS-Shield (Reverse-Ingress, „Cloudflare-Alternative")

Prinzip: selbst betriebener Reverse-Ingress statt CDN-Abhängigkeit

Anstelle eines kommerziellen CDN-/Tunnel-Dienstes wird ein **eigener Cloud-VPS-Shield** betrieben: ein VPS an einem **separaten Provider-Standort** mit eigener Public-IP. Er bildet einen Reverse-Ingress / L7-Shield vor der eigenen Infrastruktur — die selbst gehostete Alternative zu Cloudflare, ohne Datenfluss über einen Dritt-Edge.

Anbindung

- Der Shield hängt am **Hetzner vSwitch** (privates Layer-2, `10.10.0.0/16`) und tunnelt per **WireGuard** `/30` zur Core-FW DMZ. Die Tunnel-Allowlist reicht über die gesamte DMZ.
- **`ip_forward=0` , kein L3-DNAT** — der Shield ist ein sauberer Reverse-Proxy-Shield (terminiert/proxyt auf Layer 7), kein transparenter Router.
- Er ist als **zweiter öffentlicher Eingang in die DMZ** vorgesehen (Redundanz zur Edge-DMZ-Kette) — eingehend noch nachzuziehen, ausgehend bereits live.

Aktueller Stand

Funktion	Status
Ausgehender Mail-Smarthost (intern → Internet)	**Live** (an Tunnel-IP gebunden, `mynetworks`-Relay)
Eingehender MX (Internet → intern)	**geplant** — noch kein Public-Listener
HAProxy (TLS 1.2+ gehärtet)	installiert, Web-Publishing **vorbereitet** (noch kein gebundenes Frontend)
Reverse-Proxy-Shield (`ip_forward=0` , kein L3-DNAT)	aktiv

Sicherheitsargumente

- Kein offener Produktiv-Port auf der eigenen Hetzner-Infrastruktur für die über den Shield publizierten Dienste — der einzige Pfad ist der WireGuard-Tunnel.
- TLS-Termination/-Härtung auf dem Shield (TLS 1.2+), kein Klartext im Tunnel-untauglichen Bereich.
- Volle Datenhoheit: kein Traffic über einen Dritt-CDN-Edge, dennoch ein vorgelagerter Schutz- und Publishing-Layer.
- Redundanz: zweiter öffentlicher DMZ-Eingang neben der Edge-DMZ-Kette.

---

10. Mail-Pfad (ausgehend live · eingehend geplant)

Zweck

Ausgehend (live): Interne Mail-Server relayen ihre ausgehende Post über den WireGuard-Tunnel über den Cloud-VPS-Shield ins Internet (Postfix an die Tunnel-IP gebunden, `mynetworks`-Relay) — saubere, reputable Absender-IP, interne Infrastruktur bleibt verborgen.

Eingehend (Soll, noch nachzuziehen): Eingehende Mail-Protokolle lassen sich nicht über einen reinen Web-Reverse-Proxy abbilden, daher ein klassischer Reverse-/Stream-Pfad in der DMZ. Geplant sind zwei redundante Eingänge — die Edge-DMZ-Kette und der Cloud-VPS (eingehender MX) — die über die DMZ zum internen Mail-Service reichen. Die DMZ-Firewall-Regeln (25/465/587/993) sind vorbereitet; der eingehende MX ist noch **nicht** aktiv geschaltet.

Architektur

Wichtig: Der Mail-Stream vom DMZ-Ingress zum internen Mail-Service läuft über den **Querlink** `inter_fw` . Die Regel ist eng auf die Mail-Ports und genau diese beiden internen Hosts begrenzt — alles andere Cross-Zone ist Default-Deny.

Regel auf der Core-FW LAN (die einzige DMZ→LAN-Ausnahme)

```
pass in on inter_fw proto tcp from <DMZ-INGRESS> to <SVC_MAIL>
     port {25, 465, 587, 993, 995}
     keep state
     description "DMZ-Ingress → SVC_MAIL Mail-Stream"
```

---

11. Verzeichnis- und Auth-Schicht

Die Authentifizierungs-Strategie folgt Schutzschicht 5 (VPN-Zwang) und Schutzschicht 6 (Key-Authentifizierung):

- **SSH- und Hypervisor-Zugang ausschließlich per Public Key.** Kein Passwort-Login auf der Management-Ebene.
- **VPN-Zwang:** Ohne Tunnel kein Zugriff auf Management oder interne Zonen — Management-VPN am Host, Admin-VPN auf der Core-FW LAN.
- **AD-Domain-Controller isoliert** in einem eigenen VLAN (`SVC_AUTH_AD`) als Verzeichnis-Backend für Windows-Clients (Domain-Join) am Remote-Standort über das Site-to-Site-VPN. Der DC hat keinen Routing-Pfad zu den Produktiv-VLANs außerhalb der explizit erlaubten Verzeichnis-Ports.
- **Secret-Management** über HashiCorp Vault (`MGMT_VAULT`), mit getrenntem Key-Custodian für Auto-Unseal (`MGMT_VAULT_SEAL` , vTPM-Pflicht, Compromise-Isolation).

Sandbox und Quarantäne-VLAN sind **echt isoliert** (kein Routing-Pfad zu Produktiv-VLANs); das Quarantäne-VLAN ist als Blackhole ausgelegt.

---

12. Site-to-Site-VPN — Remote-Standort

Übersicht

Der zweite Standort ist über **IPsec Site-to-Site-VPN** mit dem Primär-Standort verbunden. Der Traffic wird bidirektional transparent geroutet, als wäre es ein einziges Netz.

Standort	Rolle
Primär	zentrale Infrastruktur (LAN + DMZ)
Remote	Home-Office / Remote-Workstations

VPN-Konfiguration

Parameter	Wert
VPN-Typ	IPsec (IKEv2)
Terminator Primär	Core-FW LAN (intern RFC1918, via Destination-NAT auf der Edge-FW LAN exponiert)
Terminator Remote	lokale pfSense am Remote-Standort
Auth	Pre-Shared Key + Certificates
Encryption	AES-256-GCM
PFS	DH Group 14 (2048-bit)
IKE-Phase-1	IKEv2, Mutual PSK
IKE-Phase-2	Tunnel-Mode, ESP

Policy-Beispiele auf der Core-FW LAN

Source	Destination	Zweck
Remote-Netz	AD-Domain-Controller (`SVC_AUTH_AD`)	Domain-Join + LDAP
Remote-Netz	`STOR_DATA`	File-Access auf zentralen Storage
LAN	Remote-Netz	Monitoring-Checks gegen Remote-Standort

---

13. Angriffs-Szenarien und Containment

Szenario A: Edge-FW LAN (OPNsense) kompromittiert

- **Eintritt:** via 0-Day im OPNsense-WebGUI oder Destination-NAT-Misconfig
- **Blast-Radius:** Edge-FW-LAN-VM
- **Containment:** Die Core-FW LAN blockt Edge→Core (Default-Deny). DMZ unberührt. Der Admin-VPN läuft weiter (terminiert auf der Core-FW LAN, nicht auf der Edge-FW). Der Proxmox-Host bleibt dicht (Einbahn-Vertrauen, eigene `firewalld`).
- **Recovery:** VM aus Backup wiederherstellen oder neu deployen. Die MAC-Bindung beim Provider bleibt, die neue VM bekommt dieselbe MAC.

Szenario B: Edge-FW DMZ (OPNsense) kompromittiert

- **Eintritt:** via eingehendes Mail-Port-NAT
- **Blast-Radius:** Edge-FW-DMZ-VM
- **Containment:** Die Core-FW DMZ blockt Edge→Core-DMZ (nur State-Rückverkehr). LAN unberührt; die einzige DMZ→LAN-Regel ist DMZ-Ingress→Mail-Service auf spezifischen Mail-Ports.

Szenario C: DMZ-Ingress (Mail-Reverse-Pfad) kompromittiert

- **Eintritt:** via Mail-Protokoll-Exploit
- **Blast-Radius:** Ingress-VM
- **Containment:** Die Core-FW LAN erlaubt nur die Mail-Ports DMZ-Ingress→Mail-Service. Kein Zugriff auf andere LAN-VLANs, kein Internet außer State-Responses.

Szenario D: Cloud-VPS-Shield kompromittiert

- **Eintritt:** Angreifer übernimmt den VPS am separaten Provider
- **Blast-Radius:** Shield-VPS
- **Containment:** Der Shield erreicht die Infrastruktur nur über den WireGuard-Tunnel mit Allowlist; `ip_forward=0` und kein L3-DNAT verhindern transparentes Weiterrouten. Die Core-FW DMZ filtert jeden Tunnel-Verkehr. Neuaufbau des VPS + Rotation des WireGuard-Keys.

Szenario E: Core-FW LAN kompromittiert

- **Eintritt:** via Admin-VPN-Credential-Leak oder CVE
- **Blast-Radius:** Core LAN + Admin-Hub — betrifft alle LAN-VLANs, aber nicht direkt die DMZ-Kette und nicht den Host
- **Containment:**
- Out-of-Band über Management-VPN am Host + Provider-KVM erlaubt Rescue auf Host-Ebene

- Rescue-Bridge `core_lan` erlaubt direkten Wartungs-Zugriff auf die Core-FW-LAN-VM, ohne durch den kompromittierten Transit zu müssen
- Vault-Secrets in `MGMT_VAULT` isoliert
- Backups für kompletten Rebuild
- DMZ-Kette läuft weiter (eigenständiger Uplink über die Edge-FW DMZ); der Host bleibt dicht (Einbahn-Vertrauen)

Szenario F: Provider-Account gekapert

- **Eintritt:** Social-Engineering / Passwort-Leak
- **Blast-Radius:** Server könnte neu installiert, MAC-Bindung geändert werden
- **Containment:** 2FA + Notfall-PIN auf dem Provider-Account, Support-Eskalation, Offline-Backup der VM-Backups auf externe Medien (geplant).

Szenario G: Transit ausgefallen / Interface-Fehlkonfiguration

- **Eintritt:** Admin-Fehler, Interface-Crash, MTU-Mismatch auf einem Produktiv-Transit
- **Blast-Radius:** Produktiv-Traffic einer Zone betroffen, Firewall selbst gesund
- **Containment:**
- Rescue-Bridges (`edge_lan` , `edge_dmz` , `core_lan` , `core_dmz`) erlauben das direkte Anhängen einer Wartungs-VM
- Admin fixt die Interface-Config via Rescue-Zugriff, ohne die Provider-KVM zu brauchen
- schnelle Wiederherstellung (Minuten, nicht Stunden)

---

14. Backup & Recovery

Backup-Konzept

```
VMs (alle):
  │
  ├── Proxmox Backup Server (PBS) — isolierte Bridge, eigenes Backup-Netz
  │   └── Off-Site, verschlüsselt, Restore-getestet, kein L2-Pfad zu Produktiv-VLANs
  │
  ├── Application-Daten:
  │   ├── PostgreSQL: pg_dump → Backup-Storage, verschlüsselt mit Vault-Key
  │   ├── Redis: AOF-Snapshots → Backup-Storage
  │   ├── Docker-Volumes: Restic → Backup-Storage
  │   └── Mail: Native-Backup + WORM-Archiv → Object-Store (Object-Lock)
  │
  └── Offsite-Replikation
```

Der **Proxmox Backup Server** hängt an einer **eigenen, isolierten Bridge** in einem separaten Backup-Netz, ist **off-site** repliziert, **verschlüsselt** und **Restore-getestet** — und hat **keinen Layer-2-Pfad** zu den Produktiv-VLANs.

RTO / RPO

Kategorie	RTO	RPO
Full-System (alle VMs)	< 4 Stunden	24 Stunden
PostgreSQL (Point-in-Time)	< 1 Stunde	15 Minuten (WAL-Archiving)
Mail	< 2 Stunden	24 Stunden
Configuration (Firewalls XML)	< 15 Minuten	vor jeder Änderung (manueller Export)

DR-Drill

Quartalsweise: Test-Restore einer zufälligen VM aus dem Proxmox Backup Server, Verifikation der Funktionsfähigkeit.

---

15. BSI IT-Grundschutz Mapping

BSI-Baustein	Beschreibung	Implementierung
SYS.1.3	Server unter Linux	Debian auf allen VMs
SYS.1.5	Virtualisierung	Proxmox VE 9.1, dedizierte Bridges pro Transit, MAC-Bindung pro Public-IP, eigene Host-Firewall (`firewalld`)
SYS.1.6	Containerisierung	Docker in isoliertem VLAN (`APP_CONTAINER`), eigenes Docker-Daemon-Network; daneben LXC-Guests auf dem Hypervisor
SYS.1.8	Speicherlösungen	Object-Store (WORM via Object-Lock) + Proxmox Backup Server in getrennten Netzen
SYS.2.1	Allgemeiner Client	Admin-Workstations in `ADMIN` , `/28` -Subnetz
SYS.2.2.3	Windows Client	AD-Domain-Join für Windows-Clients am Remote-Standort (via S2S-VPN), DC isoliert in eigenem VLAN

BSI-Baustein	Beschreibung	Implementierung
NET.1.1	Netzarchitektur	Zonen-Segmentierung (Edge/Core × LAN/DMZ), VLAN-Mikro-Segmentierung, Defense-in-Depth (6 Schutzschichten)
NET.1.2	Netzmanagement	pfSense + OPNsense + SIEM (Wazuh)
NET.3.1	Routing	Core-FW LAN als zentraler Router, Policy-Routing je VLAN
NET.3.2	Firewall	Zweistufige Edge/Core-Kaskade (2 Tiers) in zwei getrennten Ketten + Host- `firewalld`, Default-Deny, asymmetrische Trust-Policy, stateful
NET.3.3	VPN	Zwei getrennte VPN-Pfade (Management-VPN am Host, Admin-VPN auf der Core-FW LAN) mit WireGuard + Public-Key; Site-to-Site IPsec (AES-256-GCM, DH-14)
APP.2.2	Active Directory	AD-Domain-Controller isoliert in eigenem VLAN, kein Routing-Pfad zu Produktiv-VLANs außerhalb erlaubter Verzeichnis-Ports
APP.3.1	Webanwendungen	Web-Publishing über Cloud-VPS-Shield (Reverse-Ingress, TLS 1.2+)
APP.3.2	Webserver	Web-Frontend in `APP_WEB` + interner Reverse-Proxy
APP.3.6	DNS-Server	Split-Horizon-Resolver intern + Ad-Blocking, getrennter Services-Resolver
APP.4.3	Datenbanken	PostgreSQL (RLS) / MySQL / MongoDB / Redis in getrennten VLANs, `/29`-Micro-Segment
APP.5.3	E-Mail	interner Mail-Service + DMZ-Ingress + Cloud-VPS-Shield (ausgehender Relay; eingehender MX geplant) + WORM-Archiv
CON.1	Kryptokonzept	HashiCorp Vault (`MGMT_VAULT`), AES-256-GCM für PII in Applikations-Datenbanken
CON.3	Datensicherung	Proxmox Backup Server + Restic/Borg + Object-Lock-WORM
CON.8	Software-Entwicklung	Git-Server (`ADMIN_GIT`) + CI-Runner (`ADMIN_CI`) + IaC via Ansible (`ADMIN_IAC`)
OPS.1.1.2	Backup/Restore	Proxmox Backup Server + DR-Drill quartalsweise
OPS.1.1.5	Protokollierung	Metriken (Prometheus/Grafana/Loki) + Log-Aggregation (Graylog) + SIEM (Wazuh)
DER.1	Detektion	Wazuh-SIEM + Threat-Intel
ORP.4	Identitätsmanagement	Public-Key-Auth für SSH/Hypervisor, VPN-Zwang, isolierter AD-Domain-Controller als Verzeichnis-Backend

16. Naming Convention

Virtuelle Maschinen

- **Firewall-VMs (funktional, Rollen generisch):** Edge-FW LAN (OPNsense), Edge-FW DMZ (OPNsense), Core-FW LAN (pfSense), Core-FW DMZ (pfSense)
- **Service-VMs (server-nummeriert):** `<rolle>01` — z.B. `api01`, `core-service01`, `docker01`, `postgres01`, `redis01`, `vault01`

VLANs

`<Kategorie>_<Zweck>` (UPPERCASE) — z.B. `SVC_MAIL`, `APP_API`, `DMZ_INGRESS`

Firewall-Regeln

`{Prio} {Typ}: {Beschreibung}` — z.B. `00 Anti-Spoof: Bogons inbound blocken`, `40 VPN-Antworten Core → Internet`

17. Produktiv-Status und Verifikation

Verifikationspunkt	Ergebnis
Core-FW-Dashboard-Gateway Transit (Edge↔Core `/30`)	Online, 0% Loss, RTT < 5 ms
Edge-FW-Transit-Interface IP korrekt (`/30`)	bestätigt
WireGuard-Clients von außen (Admin-VPN, inkl. 443)	Verbindung erfolgreich über Destination-NAT zur Core-FW LAN

Verifikationspunkt	Ergebnis
Management-VPN am Proxmox-Host	aktiv, terminiert am Host (`firewalld` -Zone `trusted`), umgeht die FW-VMs
Host- `firewalld` Default-Zone `public`	Default-Deny, öffentlich nur WireGuard-UDP offen
IPsec-Tunnel zum Remote-Peer	aktiv, Handshake < 3 min
Cloud-VPS-Shield WireGuard-Tunnel zur Core-DMZ	up; Postfix ausgehender Smarthost live (an Tunnel-IP gebunden); eingehender MX geplant
Edge→Core initiiert (Test: `curl`)	korrekt geblockt (Asymmetric-Trust)
Core→Edge initiiert (Test: `curl`)	erfolgreich (Core→Edge-Management erlaubt)
`inter_fw` -Querlink (`/28`) produktiv	up, Regeln aktiv

18. Audit / Compliance-Referenzen

Diese Architektur ist bewusst so strukturiert, dass sie folgende Compliance-Anforderungen unterstützt:

- **DSGVO Art. 32 (Sicherheit der Verarbeitung):** Verschlüsselung, Zugangskontrolle, Verfügbarkeit (Abschnitte 7, 13, 14)
- **BSI IT-Grundschutz:** Baustein-Mapping in Abschnitt 15
- **ISO 27001 Annex A:** Zugangskontrolle (A.9), Kryptografie (A.10), Kommunikationssicherheit (A.13), Betriebssicherheit (A.12)
- **NIS2:** Risk-Management, Incident-Handling (Logging/SIEM), Supply-Chain
- **GoBD (für Mail-Archiv):** WORM-Speicherung via Object-Lock, 10-Jahre-Aufbewahrung

Kompetenz-Zusammenfassung (für das Portfolio)

Diese Arbeitsprobe demonstriert praktische Kompetenz in:

- **Firewall-Architektur:** zweistufige Edge/Core-Kaskade (2 Tiers) in zwei unabhängigen Uplink-Ketten für LAN und DMZ, dazu eine eigene Hypervisor-Host-Firewall (`firewalld`) mit Einbahn-Vertrauen; asymmetrische Zero-Trust-Trust-Policy, Default-Deny mit explizitem Logging, stateful Filtering, einheitliches Transit-Regelset auf OPNsense und pfSense.
- **Defense-in-Depth (6 Schutzschichten):** Firewall-Kaskade, Host- `firewalld` , VLAN-Mikro-Segmentierung, Management-Plane-Isolation, VPN-Zwang und Key-Authentifizierung als sechs unabhängige Mechanismen — bewusst getrennt von der Zahl der Firewall-Tiers.
- **Netzwerk-Segmentierung:** VLAN-Mikro-Segmentierung in `/29` -Zonen (MGMT, ADMIN, SVC, DB, APP, STOR, DEV/STAGE, MON/SEC), eine Bridge pro Transit zur Verhinderung von Layer-2-Lateral-Movement, sternförmige Admin-Bridge-Topologie um einen zentralen Admin-Hub, echt isolierte Sandbox + Blackhole-Quarantäne.
- **Virtualisierung & Resilienz:** Proxmox-VE-9.1-Bridge-Layout (35+ Guests), Rescue-/Out-of-Band-Bridges analog zu physischen Console-Ports, vollständige Isolation des Host-Management-Kanals vom Produktiv-Netz, Einbahn-Vertrauen Host ↔ Firewall-VMs.
- **Reverse-Ingress ohne CDN-Abhängigkeit:** selbst betriebener Cloud-VPS-Shield (separater Provider, WireGuard-Reverse-Ingress, HAProxy TLS 1.2+, `ip_forward=0`) als Cloudflare-Alternative und zweiter redundanter DMZ-Eingang.
- **VPN & Remote-Anbindung:** zwei getrennte VPN-Pfade (Management-VPN am Host, Admin-VPN auf der Core-FW LAN inkl. Port 443), Site-to-Site-IPsec zwischen zwei Standorten, Zone-Access-Matrix je Admin-Gruppe, Public-Key-Auth.
- **Perimeter-Minimierung:** minimale DMZ (nur Edge-Dienste + Ingress-Gateway, 48 Filter-Regeln), MAC-gebundenes IP-Filtering pro Public-IP auf Provider-Ebene als vorgelagerte Verteidigungsschicht, Vault-Secret-Management.
- **Compliance-Bewusstsein:** durchgängiges BSI-IT-Grundschutz-Mapping, DSGVO-/ISO-27001-/NIS2-/GoBD-Bezug, dokumentierte RTO/RPO und quartalsweise DR-Drills.