Case Study: Zero-Trust Infrastructure with Defense-in-Depth (6 Security Layers) and VLAN Micro-Segmentation

> Anonymized portfolio / case study version. Product and customer names, domains, hostnames, and public addresses have been removed; the architecture patterns are unchanged. Work sample: IT systems integration / network and security architecture.

About This Work Sample

This case study documents the network, firewall, and segmentation architecture of a server infrastructure that I designed, built, and operate myself on a dedicated bare-metal server (Proxmox VE 9.1 as the hypervisor, 35+ running guests across VMs and LXC containers). It demonstrates how to build a Zero-Trust infrastructure along enterprise patterns: a **two-tier firewall cascade (Edge → Core)** running as two separate chains for LAN and DMZ, strict VLAN micro-segmentation, isolated transit paths with no Layer-2 lateral movement, a dedicated hypervisor host firewall with one-way trust, out-of-band management, plus an admin VPN and site-to-site connectivity.

The underlying product name, customer references, public IP addresses, domains, hostnames, and business strategy have been deliberately removed. What is shown is exclusively my own infrastructure network architecture. The internal RFC 1918 address ranges are kept generic (e.g. `10.0.x` / an internal `/29`), since the segmentation scheme makes the architecture visible without exposing specific hosts.

The platform whose infrastructure is documented here is a multi-tenant SaaS platform (product anonymized). Status: in operation / self-operated (pre-launch).

1. Executive Summary

Architecture Principles

The network infrastructure follows a **Zero-Trust architecture** with six independent security layers (Defense-in-Depth). The compromise of one layer does not mean the compromise of the system as a whole.

1. **Firewall cascade (2 tiers)** — a two-tier chain of Edge Firewall (Tier 1, perimeter, OPNsense) → Core Firewall (Tier 2, policy enforcement, pfSense), built separately for LAN and for DMZ. Two tiers, not more.
2. **PVE host firewall (hypervisor protection)** — `firewalld` running directly on the Proxmox host (not the guests' VM firewall); the default zone `public` is default-deny, with only WireGuard UDP exposed publicly. Management (SSH, Proxmox UI) is reachable exclusively through the tunnel (`firewalld` zone `trusted`).
3. **VLAN micro-segmentation (segmentation)** — most VLANs are their own security zone with a `/29` subnet (user/DEV networks `/24`), default-deny, one hypervisor bridge per transit, and no Layer-2 lateral movement.
4. **Management-plane isolation (isolation)** — out-of-band, with no routing path from the management plane to the production VLANs.
5. **Mandatory VPN (access)** — WireGuard / IPsec; without a tunnel there is no access to management or internal zones.
6. **Key authentication (auth)** — SSH and hypervisor access by public key only.

One-Way Trust (the central principle)

The Proxmox host **hosts** the firewall VMs but **does not rely on them for its own protection**. The host has its own host firewall (`firewalld`) and its own out-of-band management. If a firewall VM fails or is compromised, the host stays locked down. The management VPN connection terminates **at the host itself**, not via the firewall VMs. Trust flows in one direction only: the host protects itself, independently of the firewalls it hosts.

Quick Facts

Metric	Value
Platform	Bare-metal dedicated server (Hetzner), Proxmox VE 9.1 as hypervisor
Running guests	35+ (VMs + LXC)
Location	Germany (location anonymized)
Remote site	Second site, via site-to-site VPN (IPsec)
Public IPv4 addresses	4 total: 3 on the host (each its own Hetzner MAC: PVE host OOB, Edge LAN, Edge DMZ) + 1 on the separate Cloud VPS Shield
Firewall tiers	2 (Edge → Core), as separate chains for LAN and DMZ
Firewall VMs	4 (Edge Firewall LAN/DMZ = OPNsense, Core Firewall LAN/DMZ = pfSense)
Host firewall	`firewalld` directly on the Proxmox host (default-deny, only WireGuard UDP exposed)
LAN VLANs	50+ (table below = excerpt), mostly `/29` , default-deny
DMZ VLANs	Minimal (edge services, ingress gateway), 48 filter rules on the Core DMZ firewall

Metric	Value
Cloud VPS Shield	Separate provider location, its own public IP, reverse ingress via WireGuard to the Core DMZ
Secret management	HashiCorp Vault
IaC	Ansible
Admin VPN (LAN)	WireGuard, terminated on the Core LAN firewall (including port 443 for restrictive networks)
Management VPN	WireGuard, terminated at the Proxmox host (bypasses the firewall VMs)
Site-to-site VPN	IPsec to the remote site
Database security	PostgreSQL with Row-Level Security (RLS)
Security monitoring	Wazuh SIEM, Graylog, Prometheus / Grafana / Loki
Out-of-band management	Provider KVM + a separate host management channel over the management VPN

2. The Two-Tier Firewall Cascade in Two Separate Chains

Terminology: 2 Firewall Tiers, 6 Security Layers

The **firewall cascade is two-tier** — Edge Firewall (Tier 1) → Core Firewall (Tier 2). This is deliberate: a technical reviewer should find exactly two firewall tiers, not some invented number. The **"6"** refers to the six distinct protection mechanisms from Section 1 (Defense-in-Depth), not to firewall stages.

Key Principle: Two Independent Uplink Chains

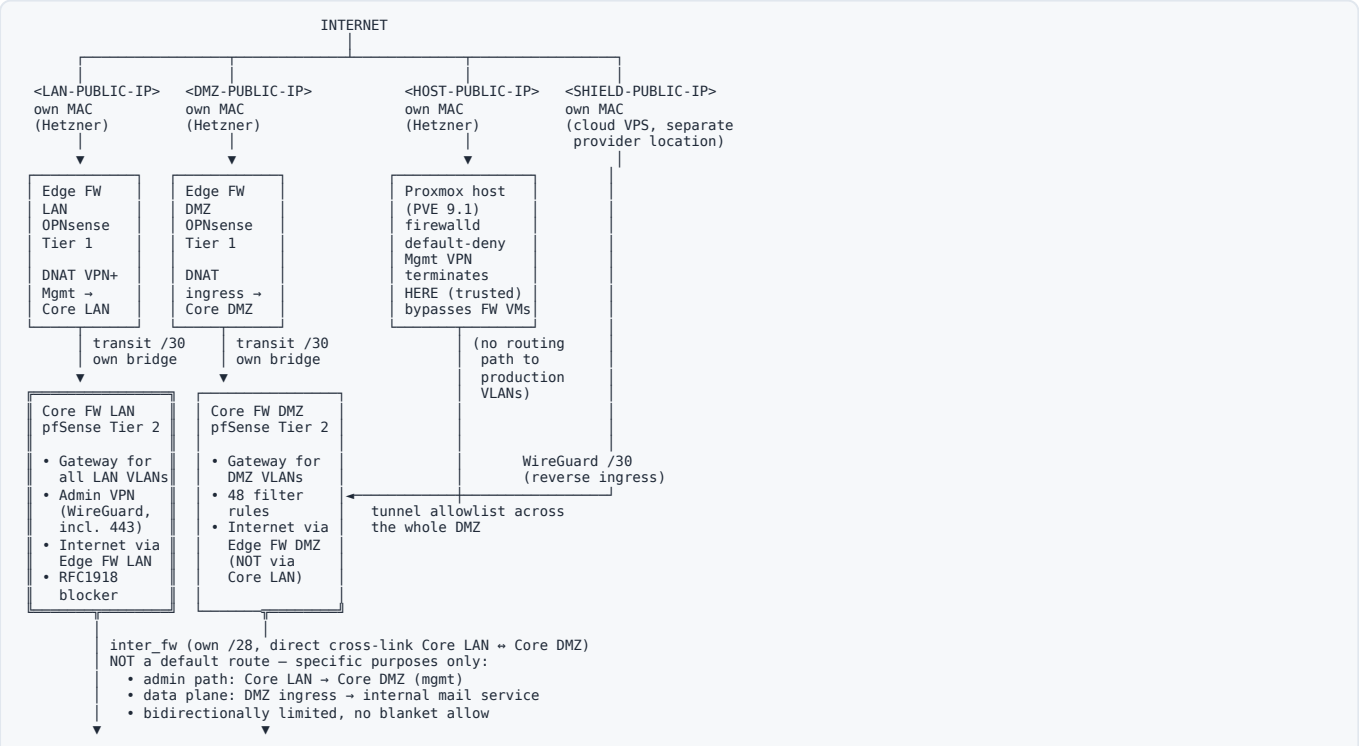
LAN and DMZ each have their own edge firewall, with its own public IP and its own hypervisor bridge. Each zone's internet access comes exclusively from its own chain. The Core LAN firewall is **not** the internet gateway for the DMZ. The only thing connecting the two Core firewalls is a direct cross-link (`inter_fw` , its own `/28`) for specific data exchange (admin path, mail stream to the internal mail service).

Firewall Role Convention

Used consistently throughout this document (hostnames anonymized, roles kept generic):

Role	Tier	Product	Zone
Edge Firewall LAN	Tier 1 (perimeter)	OPNsense	LAN chain
Core Firewall LAN	Tier 2 (policy enforcement, admin hub)	pfSense	LAN chain
Edge Firewall DMZ	Tier 1 (perimeter)	OPNsense	DMZ chain
Core Firewall DMZ	Tier 2 (policy enforcement)	pfSense	DMZ chain

Topology Diagram



```
    LAN VLANs           DMZ VLANs
   (50+, mostly /29)   (edge services,
   default-deny         ingress gateway)


Cloud VPS Shield (separate provider, own public IP):
   • 2nd public entry point into the DMZ (redundancy to the Edge DMZ chain)
   • attached to the Hetzner vSwitch (private L2, 10.10.0.0/16)
   • tunnels via WireGuard /30 to the Core DMZ, tunnel allowlist across the DMZ
   • reverse-proxy shield: ip_forward=0, no L3 DNAT → a clean L7 shield
   • Live: Postfix as an outbound smarthost (internal → internet, bound to the tunnel IP)
   • Inbound MX (internet → internal): planned — still to be wired
   • HAProxy TLS-hardened (TLS 1.2+) installed, web publishing prepared

Proxmox host is NOT a production data path — a self-contained management channel:
   • Own host firewall (firewalld), default zone public = default-deny
   • Management VPN (WireGuard) terminates AT THE HOST (firewalld zone trusted)
   • Fully bypasses the firewall VMs (one-way trust)
   • No routing path to the production VLANs
   • Can manage VMs (start/stop/migrate), not enter their production networks
   • Emergency access: provider KVM console (in case of total failure)
```

Key Points About the Topology

1. **Two parallel, independent internet uplinks** — the LAN chain and the DMZ chain each have their own public IP (with its own MAC) and their own edge firewall. No zone gets its internet through the other zone.
2. **`inter_fw` (the cross-link, its own `/28`) is not an uplink.** It exists exclusively for: - Admin access from the Core LAN firewall to the Core DMZ firewall as a management path - The data plane for inbound mail: DMZ ingress → internal mail service (internal `/29`) - No blanket DMZ↔LAN passthrough; every rule is individually explicit.
3. **Star-shaped management topology** centered on the Core LAN firewall as the central admin hub. It can administer every other firewall without using production transits or public IPs. Admin traffic stays strictly separated from production traffic.
4. **Failure behavior:** if the Edge LAN firewall fails, the LAN is cut off from the internet, but the DMZ keeps running (and vice versa). There is no shared single point of failure at the uplink. In addition, the Cloud VPS Shield provides a second, redundant public DMZ entry point.
5. **The Proxmox host is fully isolated from the production network (one-way trust)** — its own host firewall (`firewalld`), its own management VPN that terminates at the host itself and bypasses the firewall VMs. The host has neither a routing path to the production VLANs nor any reliance on the firewalls it hosts for its own protection. If a firewall VM is compromised, the host stays locked down.

The Two Tiers and Four Firewall VMs in Detail

#	Component	Role	Public IP	Instance type
T1-LAN	**Edge Firewall LAN — OPNsense**	Tier 1 perimeter for **LAN traffic**, destination NAT for VPN/mgmt to the Core LAN firewall, WAN bogon/private blocking, **internet uplink for the LAN chain**	`<LAN-PUBLIC-IP>`	KVM VM, own MAC (Hetzner)
T2-LAN	**Core Firewall LAN — pfSense**	Tier 2 policy enforcement, gateway for all LAN VLANs, terminates the admin VPN, receives internet exclusively via the Edge LAN firewall, central admin hub	Internal RFC 1918 (`/29`)	KVM VM (internal NICs: Proxmox auto-MAC)
T1-DMZ	**Edge Firewall DMZ — OPNsense**	Tier 1 perimeter for **DMZ traffic**, destination NAT to the ingress gateway, **internet uplink for the DMZ chain (independent of the LAN)**	`<DMZ-PUBLIC-IP>`	KVM VM, own MAC (Hetzner)
T2-DMZ	**Core Firewall DMZ — pfSense**	Tier 2 policy enforcement, gateway for DMZ VLANs, 48 filter rules, receives internet exclusively via the Edge DMZ firewall, administrable from the Core LAN firewall via `inter_fw`	Internal RFC 1918 (`/29`)	KVM VM (internal NICs: Proxmox auto-MAC)

The associated **host firewall (`firewalld`)** and **out-of-band management** are not a firewall tier but their own security layers (Layer 2 + Layer 4 from Section 1), and are deliberately kept separate — see one-way trust.

---

3. Provider Infrastructure

Dedicated Server

Field	Value
Platform	Bare-metal dedicated server (Hetzner)
Location	Germany (location anonymized)
Hypervisor	Proxmox VE 9.1
Guests	35+ running VMs and LXC containers

Field	Value
CPU features	VT-x + nested virtualization enabled

Public IP Assignment, One MAC per IP

The provider assigns each additional IP via its own MAC address (Hetzner "additional IP with its own MAC," anti-spoofing at the upstream router level). When the VM is created, the MAC is pinned to the vNIC. Public NICs carry Hetzner MACs; internal NICs carry Proxmox auto-MACs.

Public IP	MAC assignment	Assigned to
`<HOST-PUBLIC-IP>`	own MAC (Hetzner)	Proxmox host (out-of-band management)
`<LAN-PUBLIC-IP>`	own MAC (Hetzner)	Edge Firewall LAN (OPNsense)
`<DMZ-PUBLIC-IP>`	own MAC (Hetzner)	Edge Firewall DMZ (OPNsense)
`<SHIELD-PUBLIC-IP>`	own MAC (separate provider)	Cloud VPS Shield (reverse ingress)

Security Rationale

MAC-bound IP filtering is **a defense layer upstream of my own firewall**. Without the assigned MAC, a packet never even reaches the VM. This prevents:

- **IP spoofing** by other customers in the same provider subnet
- **MAC flooding** in the provider backbone
- **Failover attacks** in IP takeovers without a change of control

---

4. Hypervisor Bridge Layout

Bridge Concept: One Bridge per Trust Zone / Transit / Admin Path

No shared network between firewalls. Every firewall-to-firewall link is its **own hypervisor bridge**. In addition, there are **dedicated admin bridges** between the Core LAN firewall and the other firewalls for management access, separate from the production transit.

WAN Connectivity + Physical

Bridge	Type	VLAN-aware	Ports / CIDR	Purpose
`eno1`	Physical NIC	—	—	Provider uplink (main NIC)
`vmbr0`	Linux bridge	**Yes**	Port `eno1`	WAN bridge; carries all public IPs to the edge VMs via MAC binding

Rescue / Maintenance Bridges (a dedicated "service port" per firewall)

Each firewall has a dedicated **rescue bridge** to which a maintenance VM can be attached directly in an emergency. This is analogous to the physical setup where, when a network problem occurs, you plug a laptop directly into the switch/router to get access. If the production transit fails (a rule misconfiguration, an interface going down, etc.), direct access is still possible through the rescue bridge — without having to route through the production chain.

Bridge	Type	Purpose
`edge_lan`	Linux bridge	Rescue port for the Edge LAN firewall — a maintenance VM can be attached when the normal transit is broken
`edge_dmz`	Linux bridge	Rescue port for the Edge DMZ firewall
`core_lan`	Linux bridge	Rescue port for the Core LAN firewall
`core_dmz`	Linux bridge	Rescue port for the Core DMZ firewall

In normal operation: the rescue bridges have no active members (no VM attached), but are always up and ready.

In an emergency: the admin quickly spins up a maintenance VM (e.g. Debian Live), attaches it to the rescue bridge of the affected firewall, manually assigns an IP from the same subnet, and gains direct Layer-2 access to the impaired firewall — completely independent of production traffic.

Parallel to the physical setup: just like a "console port" or "out-of-band management port" on enterprise switches (Cisco IOS console, Juniper management Ethernet). It works even when the production data path is down.

Main Transit (Edge → Core, production traffic)

Bridge	Type	Members	Purpose	Transit net
`br_lan01`	Linux bridge	Edge FW LAN + Core FW LAN	Transit **Edge LAN ↔ Core LAN**	own `/30`

Bridge	Type	Members	Purpose	Transit net
`br_dmz01`	Linux bridge	Edge FW DMZ + Core FW DMZ	Transit **Edge DMZ ↔ Core DMZ**	own `/30`

Internal VLAN Trunks

Bridge	Type	VLAN-aware	Purpose
`vmbr2_VLAN`	Linux bridge	**Yes** (802.1Q trunk)	LAN VLAN trunk — the Core LAN firewall routes; all LAN service VMs attach here with a VLAN tag
`dmz_vlan01`	Linux bridge	**Yes** (802.1Q trunk)	DMZ VLAN trunk — the Core DMZ firewall routes; the DMZ ingress VM attaches here with a VLAN tag

Cross-Zone + Admin Bridges (star-shaped around the Core LAN firewall as the admin hub)

Bridge	Type	Members	Purpose
`inter_fw`	Linux bridge	Core FW LAN ↔ Core FW DMZ	**Cross-link** (own `/28`) for the admin path + mail stream (DMZ ingress → internal mail service)
`lan_opn`	Linux bridge	Core FW LAN ↔ Edge FW LAN (internal interface)	**Admin bridge**: the Core LAN firewall administers the Edge LAN firewall's GUI/SSH without traversing the production transit
`dmz_opn`	Linux bridge	Core FW LAN ↔ Edge FW DMZ (internal interface)	**Admin bridge**: the Core LAN firewall administers the Edge DMZ firewall's GUI/SSH the same way

Why the Separate Admin Bridges (`lan_opn` , `dmz_opn` , `inter_fw`)

The Core LAN firewall acts as the **central admin hub**. So that an admin does not have to manage each edge firewall over its production transit, or worse over its public IP, every edge firewall has a **dedicated admin NIC** connected only to the Core LAN firewall:

```
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)
```

This has three advantages:

1. **Admin traffic is completely separated from production traffic** — if the transit goes down (e.g. a config error), the admin still gets in.
2. **Production rule sets on the transits need no management exceptions** — cleanly separable.
3. **Smaller attack surfaces** — the admin bridges are fully internal hypervisor bridges, not routable from the outside, and each runs point-to-point between exactly two VMs.

vNIC Mapping of the 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` (cross-link), `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` (cross-link), `core_dmz` (rescue)

Note on Reading the Addressing

Gateway IPs always end in `.1` and are router interfaces on the respective Core firewall — there is no device running there, it is purely the routing gateway. Hosts have higher IPs (`.2` , `.3` , `.4` , …). Specific host IPs are kept generic in this work sample.

---

5. VLAN Structure

5.1 LAN Zone (behind the Core LAN Firewall)

All VLANs sit behind the Core LAN firewall and are strictly segmented by zone rules (50+ VLANs, mostly `/29`, default-deny). Subnet convention: `/29` for 6 hosts (sufficient for microservices), `/24` for the four larger zones (DEV, SIEM, Chat, Search), `/28` for ADMIN, `/30` for the quarantine VLAN. Specific subnets are shown generically (`10.0.x`); the VLAN scheme remains visible. (The table below is a representative excerpt; 50+ VLANs in total.)

Status legend: ✓ actively deployed · ◯ reserved (VLAN exists, no services) · ⚙ planned (roadmap)

Base

Name	Subnet	Purpose	Status
LAN	`10.0.x/24`	Standard LAN (clients, IoT base)	✓

MGMT (Management Infrastructure)

Name	Subnet	Purpose	Status
MGMT	internal `/29`	Hypervisor / Proxmox management (internal mgmt IP)	✓
MGMT_VAULT	internal `/29`	HashiCorp Vault (secret manager)	✓
MGMT_BACKUP	internal `/29`	Backup server	⚙ planned
MGMT_VAULT_SEAL	internal `/29`	Key custodian for Vault auto-unseal (compromise isolation kept separate from Vault, vTPM mandatory)	✓

ADMIN (Admin Workstations + Dev Tools)

Name	Subnet	Purpose	Status
ADMIN	internal `/28`	Admin tooling: web terminal, remote desktop gateway, remote management (bastion)	✓
ADMIN_IAC	internal `/29`	Ansible (IaC)	⚙ planned
ADMIN_GIT	internal `/29`	Internal Git server	⚙ planned
ADMIN_CI	internal `/29`	CI/CD runners	⚙ planned
ADMIN_REGISTRY	internal `/29`	Container registry	⚙ planned

SVC (Internal Services)

Name	Subnet	Purpose	Status
SVC_PROXY	internal `/29`	Internal reverse proxy	✓
SVC_MAIL	internal `/29`	Internal mail service (internal SMTP/IMAP)	✓
SVC_QUEUE	internal `/29`	Message queue	⚙ planned
SVC_DNS	internal `/29`	Client DNS (split-horizon resolver + ad blocking)	✓
SVC_AUTH_AD	internal `/29`	AD domain controller (isolated), backend for directory auth	✓
SVC_DNS_SERVICES	internal `/29`	Services DNS (containers/VMs), separated from the client resolver to avoid hairpinning	✓
SVC_PROXY_INT	internal `/29`	Internal routing proxy (default-deny to all other LAN hosts)	✓
SVC_CERT	internal `/29`	ACME client for wildcard certificates via DNS-01, push pattern (no inbound)	✓
SVC_MAIL_TIER	internal `/29`	Dedicated mail foundation (Postfix + Dovecot + DKIM + spam filtering), defense layering	⚙ planned
SVC_MAIL_ARCHIVE	internal `/29`	Tamper-proof mail archive (WORM, mTLS + RBAC), compromise containment kept separate from the mail service	⚙ planned

DB (Databases — never directly public)

Name	Subnet	Purpose	Status
DB_POSTGRESQL	internal `/29`	PostgreSQL (central, internal apps), Row-Level Security	✓
DB_MYSQL	internal `/29`	MySQL/MariaDB	◯ reserved
DB_MONGODB	internal `/29`	MongoDB	◯ reserved
DB_REDIS	internal `/29`	Redis cache / session store (native VM, not a container)	✓

APP (Application Tier)

Name	Subnet	Purpose	Status
APP_WEB	internal `/29`	Web frontend / landing	✓
APP_API	internal `/29`	Backend API (NestJS)	✓
APP_CONTAINER	internal `/29`	Docker host (microservices via macvlan)	✓
APP_WORKER	internal `/29`	Background workers	✓
APP_CORE	internal `/29`	Central core service	✓

STOR (Storage Systems)

Name	Subnet	Purpose	Status
STOR_DATA	internal `/29`	Primary storage (NFS/iSCSI)	⚙ planned
STOR_BACKUP	internal `/29`	Backup storage (Borg/Restic)	⚙ planned
STOR_S3	internal `/29`	S3-compatible object store (WORM for the mail archive)	⚙ planned

DEV / STAGE / SANDBOX

Name	Subnet	Purpose	Status
DEV	`10.0.x/24`	Development environment (incl. pentest VM)	✓
STAGE	internal `/29`	Staging	⚙ planned
SANDBOX	internal `/29`	Genuinely isolated sandbox (no routing path to production VLANs)	✓

MON / SEC

Name	Subnet	Purpose	Status
MON	internal `/29`	Infrastructure metrics (Prometheus / Grafana / Loki)	✓
SEC_SIEM	internal `/24`	SIEM stack (Wazuh: Manager, Indexer, Dashboard, fileserver, client) — larger host count	✓
SEC_LOGS	internal `/29`	Log aggregation (Graylog)	✓

Special

Name	Subnet	Purpose	Status
VPN_ADMIN	internal `/29`	Admin VPN subnet (the Core LAN firewall VPN terminates here)	✓
TRANSIT	internal `/30`	Internal transit	◯ reserved
QUARANTINE	internal `/29`	Dedicated quarantine VLAN (blackhole, no routing path)	✓

VPN Tunnel Endpoints

Endpoint	Termination	Purpose
Management VPN	Proxmox host (`firewalld` zone `trusted`)	Host OOB management, bypasses the firewall VMs
Admin VPN (LAN)	Core FW LAN (WireGuard, incl. port 443)	Admin access to the LAN VLANs
Site-to-site	Core FW LAN (IPsec)	Remote site connectivity

5.2 DMZ Zone

The DMZ is deliberately kept minimal: only edge services and an ingress gateway. It is cleanly separated from the LAN zone (its own addressing scheme). **48 filter rules** apply on the Core DMZ firewall. DMZ publishing runs via the Edge DMZ chain and/or via the Cloud VPS Shield.

Name	Purpose	Status
DMZ_INGRESS	Ingress gateway / edge services of the DMZ	✓
DMZ_MAIL	Inbound mail reception (behind the reverse/stream path)	✓
DMZ_PROXY	Reverse/stream path for mail ports to the internal mail service	⚙ planned

Why the DMZ stays minimal: Zero-Trust design. Web publishing runs over the Cloud VPS Shield as a reverse ingress (see Section 9), and mail protocols (SMTP/IMAP/POP3) over a minimal reverse/stream path. Every function exposed in the DMZ is explicitly bounded; everything else is default-deny.

5.3 Transit and Management Networks

Name	Subnet	Peers	Purpose
`br_lan01`	own `/30`	Edge FW LAN ↔ Core FW LAN	Transit Edge LAN ↔ Core LAN (own bridge)
`br_dmz01`	own `/30`	Edge FW DMZ ↔ Core FW DMZ	Transit Edge DMZ ↔ Core DMZ (own bridge)
`inter_fw`	own `/28`	Core FW LAN ↔ Core FW DMZ	Direct cross-link (admin path + mail stream)
Shield tunnel	own `/30`	Cloud VPS Shield ↔ Core FW DMZ	WireGuard reverse ingress, tunnel allowlist across the DMZ
vSwitch	`10.10.0.0/16`	Cloud VPS Shield (private L2)	Hetzner vSwitch (Layer-2 connectivity for the shield)

---

6. Firewall Rule Patterns

6.1 Transit Rule Set (a uniform pattern on all firewall transits)

Every firewall-to-firewall interface carries the same base set (reference implementation on `br_lan01`):

#	Action	Direction	Protocol	Source	Destination	Port	Purpose
00	block	in	any	bogons	any	any	Anti-spoofing: block bogon networks
01	block	in	any	NOT PeerIP	TransitNet	any	Anti-spoofing: only the peer IP may be the source
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 replies (IPsec + WG)
41	pass	in	esp	PeerIP	any	—	IPsec ESP
50	pass	in	tcp	PeerIP	TransitNet	OPN_MGMT_PORTS	Management (GUI/SSH from the peer)
90	pass	in	tcp	PeerIP	any	any	Default-allow TCP outbound (internet access)
99	block	in	any	any	RFC1918	any	Catch-all: no bridge hops into RFC 1918
100	block	in	any	any	any	any	Default-deny (explicit, with

#	Action	Direction	Protocol	Source	Destination	Port	Purpose
							logging)

6.2 OPNsense Aliases (built-in + user-defined)

Existing (OPNsense built-in): - `bogons` — IPv4 bogons (excluding RFC 1918, automatically maintained) - `__opt1_network` — automatically = the `br_lan01` transit net

User-defined: - `CORE_FW_LAN` — host IP of the Core LAN firewall on the transit - `VPN_IPSEC_PORTS` — ports 500, 4500 - `VPN_WG_STANDARD` — port 51820 - `VPN_WG_CUSTOM` — several obfuscated custom ports (incl. 443 for restrictive networks) - `VPN_ALL_UDP_PORTS` — all VPN UDP ports bundled - `WEB_OUTBOUND_PORTS` — ports 80, 443 - `RFC1918` — 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16 - `OPN_MGMT_PORTS` — ports 22, 443

6.3 Destination NAT on the Edge LAN Firewall (WAN)

Forward	Protocol	Dst port	Target	Target port	Purpose
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 for restrictive networks)
4	ESP	—	CORE_FW_LAN	—	IPsec ESP (non-NAT-T fallback)

6.4 Destination NAT on the Edge DMZ Firewall (WAN)

Forward	Protocol	Dst port	Target	Target port	Purpose
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

Each forward uses "Register rule" (OPNsense generates the WAN pass rule automatically).

---

7. Asymmetric Trust Policy

Principle

The Core LAN firewall may initiate anything outbound, but nobody may initiate back toward it. Connections are stateful — replies come back automatically, but no independent "outside-to-inside" connections are allowed. The same one-way principle applies between the host and the firewall VMs: the host protects itself and does not depend on the firewalls.

Trust Matrix

From → To	Allowed?	Rationale
Internet → Edge FW LAN	Only dedicated ports (DNAT to VPN/mgmt)	Minimal attack vector
Internet → Edge FW DMZ	Only dedicated ports (DNAT to ingress)	Minimal attack vector
Edge FW LAN → Core FW LAN	No (default-deny)	Isolate an edge compromise
Edge FW DMZ → Core FW DMZ	No (except via state)	Isolate an edge compromise
Core FW LAN → Edge FW LAN	Yes (LAN chain internet access, mgmt)	State-based return traffic
Core FW DMZ → Edge FW DMZ	Yes (DMZ chain internet access, mgmt)	State-based return traffic, independent of the LAN
Core FW LAN → Core FW DMZ (via inter_fw)	Yes (admin path, specific mgmt ports)	Cross-link for admin access
Core FW DMZ → Core FW LAN (via inter_fw)	No (default-deny)	The DMZ must not initiate back into the LAN
DMZ ingress → internal mail service (via inter_fw)	**Yes** (minimal, mail ports only)	The only permitted cross-zone data-plane rule
Internal mail service → internet	Yes (outbound relay)	Mail sending
Cloud VPS Shield → Core FW DMZ (WireGuard)	Yes (reverse ingress, tunnel allowlist across the DMZ)	2nd public DMZ entry point
Core FW DMZ → Cloud VPS Shield	Yes (state return traffic in the tunnel)	Reverse-proxy replies
Admin VPN → Core FW LAN	Yes (public key + restrictive allowlist)	Admin entry into the LAN
Management VPN → Proxmox host	Yes (`firewalld` zone `trusted` , public key)	OOB management, bypasses the firewall VMs

From → To	Allowed?	Rationale
Remote site → LAN (S2S VPN)	Yes (specific)	Domain join, directory, storage access
Remote site → DMZ	No	The DMZ stays internal-only
Proxmox host → production VLANs	**No routing path exists**	The host is out-of-band; it can only lifecycle-control VMs, not enter their networks
Firewall VM → Proxmox host	**No trust path**	One-way trust: the host does not rely on the firewalls it hosts

Zone Access Matrix (Admin VPN)

Per VPN client group, the Core LAN firewall defines which LAN VLANs are reachable:

Admin group	Access to
`admin-full`	All 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 Paths

The architecture has **two separate VPN paths** with different termination points — this is part of the one-way trust.

Management VPN — terminated at the Proxmox Host

- **WireGuard**, terminated directly on the Proxmox host (`firewalld` zone `trusted`).
- **Fully bypasses the firewall VMs.** Only WireGuard UDP is exposed publicly on the host; SSH and the Proxmox UI are reachable exclusively through the tunnel.
- Purpose: host out-of-band management (VM lifecycle: start/stop/migrate). No routing path to the production VLANs.

Admin VPN (LAN) — terminated on the Core LAN Firewall

- **WireGuard** (standard port + obfuscated custom ports, incl. **port 443** for restrictive networks), plus IPsec.
- The connection comes in via the Edge LAN firewall (destination NAT) to the Core LAN firewall.
- The admin client receives an IP from `VPN_ADMIN` ; a zone access matrix controls the reachable LAN VLANs per group.

```
Admin client (internet)
  | UDP WireGuard (51820 / custom / 443) or IPsec
  ▼
Edge FW LAN (OPNsense, Tier 1) — destination NAT
  | transit /30 (own bridge)
  ▼
Core FW LAN (pfSense, Tier 2) — VPN terminator
  | client IP from VPN_ADMIN, policy routing per group
  ▼
Permitted LAN VLANs (per the zone access matrix)
```

Authentication

- WireGuard: public-key authentication (one dedicated key per client).
- SSH and hypervisor access by public key only (Security Layer 6).
- Logging of all VPN sessions to the SIEM stack (Wazuh).

---

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

Principle: a Self-Operated Reverse Ingress Instead of CDN Dependency

Rather than a commercial CDN/tunnel service, I run my **own Cloud VPS Shield**: a VPS at a **separate provider location** with its own public IP. It forms a reverse ingress / L7 shield in front of my own infrastructure — the self-hosted alternative to Cloudflare, with no data flowing through a third-party edge.

Connectivity

- The shield is attached to the **Hetzner vSwitch** (private Layer 2, `10.10.0.0/16`) and tunnels via **WireGuard** `/30` to the Core DMZ firewall. The tunnel allowlist spans the entire DMZ.
- `ip_forward=0` **, no L3 DNAT** — the shield is a clean reverse-proxy shield (terminating/proxying at Layer 7), not a transparent router.
- It is intended as the **second public entry point into the DMZ** (redundancy to the Edge DMZ chain) — inbound still to be wired; outbound is already live.

Current Status

Function	Status
Outbound mail smarthost (internal → internet)	**Live** (bound to the tunnel IP, `mynetworks` relay)
Inbound MX (internet → internal)	**Planned** — no public listener yet
HAProxy (TLS 1.2+ hardened)	Installed, web publishing **prepared** (no bound frontend yet)
Reverse-proxy shield (`ip_forward=0` , no L3 DNAT)	Active

Security Arguments

- No open production port on my own Hetzner infrastructure for the services published through the shield — the only path is the WireGuard tunnel.
- TLS termination/hardening on the shield (TLS 1.2+), no cleartext in any tunnel-unfit segment.
- Full data sovereignty: no traffic over a third-party CDN edge, yet still an upstream protection and publishing layer.
- Redundancy: a second public DMZ entry point alongside the Edge DMZ chain.

---

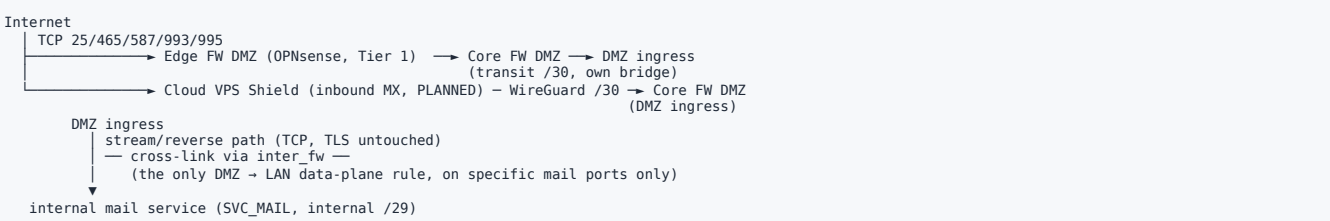
10. Mail Path (outbound live · inbound planned)

Purpose

Outbound (live): Internal mail servers relay their outgoing mail through the WireGuard tunnel via the Cloud VPS Shield to the internet (Postfix bound to the tunnel IP, `mynetworks` relay) — a clean, reputable sender IP; internal infrastructure stays hidden.

Inbound (target state, still to be wired): Inbound mail protocols cannot be handled by a pure web reverse proxy, so a classic reverse/stream path in the DMZ is planned. Two redundant entry points are intended — the Edge DMZ chain and the Cloud VPS (inbound MX) — reaching the internal mail service through the DMZ. The DMZ firewall rules (25/465/587/993) are prepared; the inbound MX is **not** yet activated.

Architecture



Important: the mail stream from the DMZ ingress to the internal mail service runs over the **cross-link** `inter_fw` . The rule is tightly scoped to the mail ports and to exactly these two internal hosts — everything else cross-zone is default-deny.

Rule on the Core LAN Firewall (the only DMZ→LAN exception)

```
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. Directory and Auth Layer

The authentication strategy follows Security Layer 5 (mandatory VPN) and Security Layer 6 (key authentication):

- **SSH and hypervisor access by public key only.** No password login on the management plane.
- **Mandatory VPN:** without a tunnel there is no access to management or internal zones — the management VPN at the host, the admin VPN on the Core LAN firewall.
- **AD domain controller isolated** in its own VLAN (`SVC_AUTH_AD`) as the directory backend for Windows clients (domain join) at the remote site over the site-to-site VPN. The DC has no routing path to the production VLANs beyond the explicitly permitted directory ports.
- **Secret management** via HashiCorp Vault (`MGMT_VAULT`), with a separate key custodian for auto-unseal (`MGMT_VAULT_SEAL` , vTPM mandatory, compromise isolation).

The sandbox and quarantine VLAN are **genuinely isolated** (no routing path to production VLANs); the quarantine VLAN is designed as a blackhole.

---

12. Site-to-Site VPN — Remote Site

Overview

The second site is connected to the primary site via **IPsec site-to-site VPN**. Traffic is routed bidirectionally and transparently, as though it were a single network.

Site	Role
Primary	Central infrastructure (LAN + DMZ)
Remote	Home office / remote workstations

VPN Configuration

Parameter	Value
VPN type	IPsec (IKEv2)
Terminator (primary)	Core FW LAN (internal RFC 1918, exposed via destination NAT on the Edge LAN firewall)
Terminator (remote)	Local pfSense at the remote site
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 Examples on the Core LAN Firewall

Source	Destination	Purpose
Remote network	AD domain controller (`SVC_AUTH_AD`)	Domain join + LDAP
Remote network	`STOR_DATA`	File access to central storage
LAN	Remote network	Monitoring checks against the remote site

---

13. Attack Scenarios and Containment

Scenario A: Edge LAN firewall (OPNsense) compromised

- **Entry:** via a 0-day in the OPNsense web GUI or a destination NAT misconfiguration
- **Blast radius:** the Edge LAN firewall VM
- **Containment:** the Core LAN firewall blocks Edge→Core (default-deny). The DMZ is untouched. The admin VPN keeps running (terminated on the Core LAN firewall, not on the edge firewall). The Proxmox host stays locked down (one-way trust, its own `firewalld`).
- **Recovery:** restore the VM from backup or redeploy. The MAC binding at the provider persists; the new VM gets the same MAC.

Scenario B: Edge DMZ firewall (OPNsense) compromised

- **Entry:** via inbound mail port NAT
- **Blast radius:** the Edge DMZ firewall VM
- **Containment:** the Core DMZ firewall blocks Edge→Core DMZ (state return traffic only). The LAN is untouched; the only DMZ→LAN rule is DMZ ingress→mail service on specific mail ports.

Scenario C: DMZ ingress (mail reverse path) compromised

- **Entry:** via a mail protocol exploit
- **Blast radius:** the ingress VM
- **Containment:** the Core LAN firewall permits only the mail ports DMZ ingress→mail service. No access to other LAN VLANs, no internet except state responses.

Scenario D: Cloud VPS Shield compromised

- **Entry:** an attacker takes over the VPS at the separate provider
- **Blast radius:** the shield VPS
- **Containment:** the shield reaches the infrastructure only over the WireGuard tunnel with its allowlist; `ip_forward=0` and the absence of L3 DNAT prevent transparent onward routing. The Core DMZ firewall filters all tunnel traffic. Rebuild the VPS + rotate the WireGuard key.

Scenario E: Core LAN firewall compromised

- **Entry:** via an admin VPN credential leak or a CVE
- **Blast radius:** Core LAN + admin hub — affects all LAN VLANs, but not directly the DMZ chain and not the host
- **Containment:**
- Out-of-band via the management VPN at the host + the provider KVM allows rescue at the host level
- The rescue bridge `core_lan` allows direct maintenance access to the Core LAN firewall VM without going through the compromised transit
- Vault secrets isolated in `MGMT_VAULT`

- Backups for a full rebuild
- The DMZ chain keeps running (its own independent uplink via the Edge DMZ firewall); the host stays locked down (one-way trust)

Scenario F: Provider account hijacked

- **Entry:** social engineering / password leak
- **Blast radius:** the server could be reinstalled, the MAC binding changed
- **Containment:** 2FA + emergency PIN on the provider account, support escalation, offline backup of the VM backups to external media (planned).

Scenario G: Transit down / interface misconfiguration

- **Entry:** admin error, interface crash, MTU mismatch on a production transit
- **Blast radius:** the production traffic of one zone is affected, the firewall itself healthy
- **Containment:**
- The rescue bridges (`edge_lan` , `edge_dmz` , `core_lan` , `core_dmz`) allow a maintenance VM to be attached directly
- The admin fixes the interface config via rescue access, with no need for the provider KVM
- Fast recovery (minutes, not hours)

---

14. Backup & Recovery

Backup Concept

```
VMs (all):
  │
  ├── Proxmox Backup Server (PBS) — isolated bridge, dedicated backup network
  │   └── Off-site, encrypted, restore-tested, no L2 path to production VLANs
  │
  ├── Application data:
  │   ├── PostgreSQL: pg_dump → backup storage, encrypted with a Vault key
  │   ├── Redis: AOF snapshots → backup storage
  │   ├── Docker volumes: Restic → backup storage
  │   └── Mail: native backup + WORM archive → object store (object lock)
  │
  └── Off-site replication
```

The **Proxmox Backup Server** is attached to its **own isolated bridge** in a separate backup network, is replicated **off-site**, **encrypted**, and **restore-tested** — and has **no Layer-2 path** to the production VLANs.

RTO / RPO

Category	RTO	RPO
Full system (all VMs)	< 4 hours	24 hours
PostgreSQL (point-in-time)	< 1 hour	15 minutes (WAL archiving)
Mail	< 2 hours	24 hours
Configuration (firewall XML)	< 15 minutes	Before every change (manual export)

DR Drill

Quarterly: a test restore of a random VM from the Proxmox Backup Server, verifying that it works.

---

15. BSI IT-Grundschutz Mapping

BSI building block	Description	Implementation
SYS.1.3	Servers under Linux	Debian on all VMs
SYS.1.5	Virtualization	Proxmox VE 9.1, dedicated bridges per transit, MAC binding per public IP, own host firewall (`firewalld`)
SYS.1.6	Containerization	Docker in an isolated VLAN (`APP_CONTAINER`), own Docker daemon network; alongside LXC guests on the hypervisor
SYS.1.8	Storage solutions	Object store (WORM via object lock) + Proxmox Backup Server in separate networks
SYS.2.1	General client	Admin workstations in `ADMIN` , `/28` subnet
SYS.2.2.3	Windows client	AD domain join for Windows clients at the remote site (via S2S VPN), DC isolated in its own VLAN
NET.1.1	Network architecture	Zone segmentation (Edge/Core × LAN/DMZ), VLAN micro-segmentation, Defense-in-Depth (6 security layers)

BSI building block	Description	Implementation
NET.1.2	Network management	pfSense + OPNsense + SIEM (Wazuh)
NET.3.1	Routing	Core LAN firewall as the central router, policy routing per VLAN
NET.3.2	Firewall	Two-tier Edge/Core cascade (2 tiers) in two separate chains + host `firewalld`, default-deny, asymmetric trust policy, stateful
NET.3.3	VPN	Two separate VPN paths (management VPN at the host, admin VPN on the Core LAN firewall) with WireGuard + public key; site-to-site IPsec (AES-256-GCM, DH-14)
APP.2.2	Active Directory	AD domain controller isolated in its own VLAN, no routing path to production VLANs beyond permitted directory ports
APP.3.1	Web applications	Web publishing via the Cloud VPS Shield (reverse ingress, TLS 1.2+)
APP.3.2	Web server	Web frontend in `APP_WEB` + internal reverse proxy
APP.3.6	DNS server	Split-horizon resolver internally + ad blocking, separate services resolver
APP.4.3	Databases	PostgreSQL (RLS) / MySQL / MongoDB / Redis in separate VLANs, `/29` micro-segment
APP.5.3	Email	Internal mail service + DMZ ingress + Cloud VPS Shield (outbound relay; inbound MX planned) + WORM archive
CON.1	Cryptographic concept	HashiCorp Vault (`MGMT_VAULT`), AES-256-GCM for PII in the application databases
CON.3	Data backup	Proxmox Backup Server + Restic/Borg + object-lock WORM
CON.8	Software development	Git server (`ADMIN_GIT`) + CI runners (`ADMIN_CI`) + IaC via Ansible (`ADMIN_IAC`)
OPS.1.1.2	Backup/restore	Proxmox Backup Server + quarterly DR drill
OPS.1.1.5	Logging	Metrics (Prometheus/Grafana/Loki) + log aggregation (Graylog) + SIEM (Wazuh)
DER.1	Detection	Wazuh SIEM + threat intel
ORP.4	Identity management	Public-key auth for SSH/hypervisor, mandatory VPN, isolated AD domain controller as the directory backend

16. Naming Convention

Virtual Machines

- **Firewall VMs (functional, roles kept generic):** Edge Firewall LAN (OPNsense), Edge Firewall DMZ (OPNsense), Core Firewall LAN (pfSense), Core Firewall DMZ (pfSense)
- **Service VMs (server-numbered):** `<role>01` — e.g. `api01`, `core-service01`, `docker01`, `postgres01`, `redis01`, `vault01`

VLANs

`<category>_<purpose>` (UPPERCASE) — e.g. `SVC_MAIL`, `APP_API`, `DMZ_INGRESS`

Firewall Rules

`{prio} {type}: {description}` — e.g. `00 Anti-Spoof: block bogons inbound`, `40 VPN replies Core → Internet`

17. Production Status and Verification

Verification point	Result
Core firewall dashboard gateway transit (Edge↔Core `/30`)	Online, 0% loss, RTT < 5 ms
Edge firewall transit interface IP correct (`/30`)	Confirmed
WireGuard clients from outside (admin VPN, incl. 443)	Connection succeeds via destination NAT to the Core LAN firewall
Management VPN at the Proxmox host	Active, terminated at the host (`firewalld` zone `trusted`), bypasses the firewall VMs
Host `firewalld` default zone `public`	Default-deny, only WireGuard UDP exposed

Verification point	Result
IPsec tunnel to the remote peer	Active, handshake < 3 min
Cloud VPS Shield WireGuard tunnel to the Core DMZ	Up; Postfix outbound smarthost live (bound to the tunnel IP); inbound MX planned
Edge→Core initiated (test: `curl`)	Correctly blocked (asymmetric trust)
Core→Edge initiated (test: `curl`)	Successful (Core→Edge management permitted)
`inter_fw` cross-link (`/28`) in production	Up, rules active

---

18. Audit / Compliance References

This architecture is deliberately structured to support the following compliance requirements:

- **GDPR Art. 32 (security of processing):** encryption, access control, availability (Sections 7, 13, 14)
- **BSI IT-Grundschutz:** building-block mapping in Section 15
- **ISO 27001 Annex A:** access control (A.9), cryptography (A.10), communications security (A.13), operations security (A.12)
- **NIS2:** risk management, incident handling (logging/SIEM), supply chain
- **GoBD (for the mail archive):** WORM storage via object lock, 10-year retention

---

Competency Summary (for the Portfolio)

This work sample demonstrates practical competence in:

- **Firewall architecture:** a two-tier Edge/Core cascade (2 tiers) across two independent uplink chains for LAN and DMZ, plus a dedicated hypervisor host firewall (`firewalld`) with one-way trust; an asymmetric Zero-Trust trust policy, default-deny with explicit logging, stateful filtering, and a uniform transit rule set across OPNsense and pfSense.
- **Defense-in-Depth (6 security layers):** firewall cascade, host `firewalld`, VLAN micro-segmentation, management-plane isolation, mandatory VPN, and key authentication as six independent mechanisms — deliberately distinct from the number of firewall tiers.
- **Network segmentation:** VLAN micro-segmentation into `/29` zones (MGMT, ADMIN, SVC, DB, APP, STOR, DEV/STAGE, MON/SEC), one bridge per transit to prevent Layer-2 lateral movement, a star-shaped admin-bridge topology around a central admin hub, a genuinely isolated sandbox + blackhole quarantine.
- **Virtualization & resilience:** Proxmox VE 9.1 bridge layout (35+ guests), rescue / out-of-band bridges analogous to physical console ports, full isolation of the host management channel from the production network, one-way trust between host and firewall VMs.
- **Reverse ingress without CDN dependency:** a self-operated Cloud VPS Shield (separate provider, WireGuard reverse ingress, HAProxy TLS 1.2+, `ip_forward=0`) as a Cloudflare alternative and a second, redundant DMZ entry point.
- **VPN & remote connectivity:** two separate VPN paths (management VPN at the host, admin VPN on the Core LAN firewall incl. port 443), site-to-site IPsec between two sites, a zone access matrix per admin group, public-key auth.
- **Perimeter minimization:** a minimal DMZ (only edge services + ingress gateway, 48 filter rules), MAC-bound IP filtering per public IP at the provider level as an upstream defense layer, Vault secret management.
- **Compliance awareness:** consistent BSI IT-Grundschutz mapping, GDPR / ISO 27001 / NIS2 / GoBD alignment, documented RTO/RPO, and quarterly DR drills.