

Network Boot & Authentication Architecture for a Diskless Host Fleet — v2

802.1X EAP-TLS with TPM-anchored machine identity, MACsec link encryption, dm-verity-protected erofs root over NVMe/TCP, signed Unified Kernel Images, and an OCI release-artifact pipeline. Supersedes v1 (12 June 2026); this revision, 12 June 2026.

1. Objectives and Constraints

The fleet consists of diskless hosts that UEFI HTTP-boot a shared, signed Unified Kernel Image (UKI) and mount an immutable root filesystem over NVMe/TCP. The network enforces IEEE 802.1X port authentication with MACsec (802.1X-2010) link encryption. Design goals:

- **One generic boot artifact for the entire fleet.** No per-host builds; all host identity lives in hardware (TPM 2.0).
- **Non-exportable machine credentials.** The EAP-TLS private key is generated inside the TPM and never exists on disk or on the wire.
- **Continuous link security across the boot.** The MACsec session established in the initramfs survives the pivot to the real root, because the root filesystem itself rides on that link.
- **End-to-end integrity.** One Secure Boot signature on the UKI transitively covers kernel, initramfs, kernel cmdline, and — via the embedded dm-verity root hash — every block of the root filesystem.
- **Atomic, auditable releases.** Each fleet release is a single content-addressed OCI image coupling the signed UKI with its matching root image; rollout and rollback are one digest change in git.

2. Key Design Decisions

Concern	Decision	Rationale
EAP method	EAP-TLS (certificate-based)	Strongest standard machine auth; the MACsec CAK is derived from the EAP MSK automatically.
Private key storage	TPM 2.0, persisted at a fixed handle (e.g. 0x81010001) identical on every host	Persistent objects live inside the TPM — no on-disk metadata, nothing host-specific in the image.
Client certificate storage	TPM NV index (fixed index fleet-wide), read at boot with tpm2_nvread into /run	Certs are ~1–2 KB DER and fit standard NV space. File-based cert reference sidesteps known NM/wpa_supplicant PKCS#11-certificate bugs.
Supplicant	wpa_supplicant run directly in the initramfs as a systemd “root storage daemon” (argv[0] = @wpa_supplicant)	MKA keepalives must continue across switch-root; switches time out MKA peers after ~6 s of silence. NM is kept out of the authenticated link entirely.
Switch port mode	Low-impact / should-secure: pre-auth ACL permits DHCP + HTTPS to provisioning servers only; MACsec required after authentication	No firmware speaks EAPOL or MACsec, so a must-secure port can never boot a diskless host. Low-impact beats MAB: no MAC whitelist.
Boot artifact	Signed UKI (kernel + initramfs + cmdline in one PE binary), fetched directly by UEFI HTTP Boot over TLS	One file, one fetch, one signature covering the whole boot configuration including roothash=. No bootloader, no unsigned config files.
Root filesystem	Reproducible erofs image (lz4hc) + appended dm-verity hash tree, shared read-only NVMe/TCP namespace, tmpfs overlay for runtime state	Immutability is structural (no write paths); deterministic builds give a stable root hash; compression saves fabric bandwidth and MACsec cycles. Block-RO + verity makes per-host NVMe auth largely unnecessary.
Host NVMe identity	hostnqn/hostid derived at boot from the TPM certificate serial	NVMe identity is a function of the machine identity already proven via 802.1X; the image stays byte-identical.

Concern	Decision	Rationale
Release unit	FROM-scratch OCI image: signed UKI + rootfs.erofs + verity.json + checksums; deployments pinned by digest	Atomically couples UKI ↔ root namespace (a mismatched pair is an unbootable fleet); registry gives content-addressed, immutable, cosign-able storage.
Serving stack	Separate, independently patched images: SPDK nvme_tgt (userspace NVMe/TCP target) + nginx/caddy (HTTPS), fed by the artifact image via initContainer / image volume	Server software and fleet releases change on different cadences for different reasons; kernel nvmet cannot be genuinely containerised (configs control plane, host-kernel data path).

3. End-to-End Boot and Authentication Flow

#	Stage	What happens
1	Link up / pre-auth	Switch port in low-impact mode. Pre-auth ACL permits only DHCP and HTTPS to the serving stack. Host firmware gets a provisioning-VLAN DHCP lease.
2	UEFI HTTP Boot fetches the UKI	Firmware downloads fleet-<version>.efi directly (no shim/GRUB/config in the chain). Server TLS certificate validated against the CA enrolled in the TlsCaCertificate EFI variable (transport protection only).
3	Secure Boot verifies the UKI	One signature check covers kernel, initramfs, and the cmdline — including roothash= and all rd.* parameters. This is the integrity anchor for the entire chain.
4	Initramfs: identity assembly	Dracut hook reads the client certificate from the TPM NV index into /run (tpm2_nvread via /dev/tpmrm0). The private key is referenced at its persistent TPM handle; it never leaves the TPM.
5	802.1X EAP-TLS + MKA	@wpa_supplicant authenticates on the physical NIC (macsec_linux driver). RADIUS validates the chain, authorises the port, optionally assigns the production VLAN. MKA derives the CAK from the EAP MSK, negotiates SAKs, creates and keys macsec0.
6	Re-address on macsec0	DHCP runs on macsec0 (mandatory re-DHCP: any pre-auth lease is invalid after auth/VLAN change). All subsequent traffic is MACsec-encrypted.
7	NVMe/TCP connect	hostnqn/hostid are derived from the certificate serial; dracut's nvmm module connects the shared read-only namespace over macsec0.
8	dm-verity mapping + overlay root	systemd-veritysetup maps /dev/mapper/root from the namespace using roothash= and hash-offset= (data and hash tree share one device). The erofs root mounts read-only; rootovl adds a tmpfs overlay for runtime state. Any tampered block fails verification at read time.
9	switch-root	systemd's killing spree spares the @-prefixed supplicant (root-storage-daemon convention); the MKA session and NVMe/TCP connection never drop. Supplicant state lives in /run, which persists.
10	Steady state	Physical NIC and macsec0 are unmanaged in NetworkManager. The initramfs-launched supplicant owns the link for the life of the system, servicing MKA rekeys. NVMe/TCP keepalives ride the same uninterrupted link.

Why the supplicant must survive the pivot

With plain EAPOL, a supplicant dying at switch-root is harmless. With MACsec it is fatal in seconds: the kernel holds the current SAKs, but MKA peers expect keepalive MKPDUs roughly every 2 s and time out after ~6 s of silence, tearing down the secure association — and with it the link carrying the root filesystem. The systemd root-storage-daemon pattern (argv[0] beginning with “@”) exists for this class of problem and is the same mechanism iSCSI daemons use.

4. Identity and Trust Layout

Artifact	Location	Notes
EAP-TLS private key	TPM persistent handle, e.g. 0x81010001 (fleet-standard)	Generated in-TPM at enrolment; non-exportable. Not PCR-sealed by default — attests “this machine’s TPM”, not boot state.
Client certificate	TPM NV index, e.g. 0x01800001 (fleet-standard)	DER, ~1–2 KB. Also the source of the derived NVMe hostnqn/hostid.
RADIUS CA chain	File in the generic initramfs (inside the signed UKI)	Public, fleet-wide.
HTTPS Boot CA	TlsCaCertificate EFI variable (GUID fd2340d0-3dab-4349-a6c7-3b4f12b48eae), EFI_SIGNATURE_LIST format	Dedicated small boot-infrastructure CA. Variable is typically not write-protected — transport protection only; integrity rests on Secure Boot.
Secure Boot trust anchor	Fleet signing certificate in the firmware db (own PK/KEK/db hierarchy; see §6)	Enrolled once at host enrolment; never changes on the release path.
Root filesystem integrity	dm-verity root hash, embedded in the signed UKI cmdline	Stable across identical input trees (reproducible mkfs.erofs). Covered transitively by the UKI signature.
Fleet signing key	CI/HSM only — never on the Packer build host or inside the build VM	The crown jewel of the design: whoever holds it boots arbitrary code on every host.

5. Root Filesystem: erofs + dm-verity over NVMe/TCP

The root is a single shared read-only namespace, not per-host block devices. erofs is read-only by construction (no journal, no write paths — unlike ext4, which replays a dirty journal even on a “read-only” mount and would fault a verity device), packs the tree densely, builds reproducibly (fixed timestamps and UUID ⇒ identical input yields an identical verity root hash), and its lz4hc compression reduces both fabric I/O and MACsec encryption work on every read. The dm-verity hash tree is appended in-file at a 4 KiB-aligned offset, so data and hash share one namespace (systemd.verity_root_options=hash-offset=N). Runtime state lives on a tmpfs overlay (rootovl); per-host machine-id is generated into the overlay at boot.

Because the namespace is target-enforced read-only and every block is verity-verified end-to-end, per-host NVMe authentication (DH-HMAC-CHAP) and NVMe-TLS add little here and are omitted; MACsec already protects the access link. Fallback if erofs is unavailable on the target kernel: journal-less, reproducibility-pinned ext4 (mkfs.ext4 -O ^has_journal -d <tree>), then squashfs.

6. Secure Boot and UKI Lifecycle

Routine releases never touch firmware state: Secure Boot trusts the fleet signing *certificate*, enrolled once, and every release is simply signed with it.

Per-release pipeline (runs on the build host — the key never enters the build VM):

```
ukify build --linux vmlinuz --initrd initramfs.img \
  --cmdline "<single line incl. roothash= and hash-offset=>" \
  --signtool sbsign \
  --secureboot-private-key <CI/HSM key> \
  --secureboot-certificate <fleet cert> \
  --output fleet-<version>.efi
# Hard gate: roothash in the UKI cmdline MUST equal verity.json
# (a mismatched pair is an unbootable fleet; the build refuses).
```

Trust anchor (one-time decision)

- **Recommended — own the platform keys:** enrol fleet PK/KEK/db at host enrolment (sbctl, or Redfish/BMC Secure Boot key management at scale). UKIs load directly; no shim, no GRUB, no Microsoft

CA in the chain. **Caveat:** add-in NIC option ROMs are signed by the Microsoft Third-Party UEFI CA — retain that CA in db, or verify platforms HTTP-boot from built-in firmware drivers, or risk unbootable hardware.

- **Alternative — shim + MOK:** keep stock Secure Boot, enrol the fleet cert as a Machine Owner Key (one-time console interaction with MokManager per host), chain shim → signed systemd-boot → signed UKI. Less firmware surgery; reintroduces Microsoft and shim into the trust chain.

Rotation, revocation, rollback

- db and dbx are *authenticated variables*: updates signed by the fleet KEK can be pushed from the running OS (efi-updatevar / fwupd) or via Redfish — no console required. This is the mechanism for key rotation and for revoking compromised builds, and the reason to keep a PK/KEK/db hierarchy rather than one flat key.
- “Anything signed by us, forever” permits rollback to old vulnerable builds (an attacker serves an old signed UKI plus its matching old root image). Controls, in increasing strength: publish only current + previous versions; dbx-revoke superseded UKI hashes; a TPM NV monotonic counter checked early in boot for hard anti-rollback (likely disproportionate unless the threat model demands it).

7. Release Pipeline and Serving Stack

Build (Packer, qemu builder)

A throwaway build VM is kickstart-installed and provisioned (packages, the 95dot1x-macsec dracut module, diskless adaptations), then builds the fleet-generic initramfs with dracut `--no-hostonly` — mandatory, since the build VM only has virtio hardware; real NIC drivers and the macsec/nvme-tcp modules are asserted explicitly in dracut configuration. Post-processing on the build host: extract kernel + initramfs and root tree (libguestfs), mkfs.erofs (reproducible) + veritysetup format (hash-offset layout), then ukify + sbsign, then the OCI image build. The absence of a TPM in the build VM is by design — nothing host-specific exists at build time.

Release artifact (OCI image, FROM scratch)

Contains only `/artifacts/{fleet-<version>.efi, rootfs.erofs, verity.json, SHA256SUMS}`; the root hash is recorded as an image label. No shell, no distro, nothing executable. The image is cosign-signed for deployment-pipeline provenance (sbsign is for firmware; cosign is for the cluster). The registry provides content-addressed, immutable storage of every release.

Serving stack (separate, independently patched images)

- **SPDK nvmmf_tgt** serves rootfs.erofs as a read-only NVMe/TCP namespace on 4420. Userspace target $\Rightarrow$ genuinely containerised (kernel nvmet is configfs-driven with a host-kernel data path and a kernel-namespace backing-file pathing trap). Needs hugepages; host networking; tunable to one polling core for a read-mostly boot workload.
- **nginx/caddy** serves the UKI on 443 with the boot-infrastructure CA's certificate.
- **Wiring**: the artifact image runs as an initContainer copying `/artifacts` into a shared volume (or is mounted directly via the k8s image-volume source). Server configs reference fixed paths, so a release is purely an image-digest bump in the deployment manifest — one commit per fleet release, the single place coupling UKI $\leftrightarrow$ namespace. Rollback = revert the commit.
- **Availability**: this stack is the fleet's disk. NVMe/TCP hosts ride out target restarts via reconnect (`ctrl-loss-tmo`, default 600 s; the read-only root + page cache make brief stalls invisible), but outages beyond that window hang every host. Run replicas behind distinct discovery entries; do not serve a whole fleet from one unreplicated node. The HTTPS side matters only at boot; the NVMe side matters every second — same pod is fine, same failure domain all the way down is not.

Environment note: the cmdline (including the NVMe discovery address) is immutable once signed into the UKI. Use a discovery-service DNS name that resolves per environment to keep a single artifact across lab and production; otherwise per-environment UKI builds are required.

8. Enrolment (one-time per host)

1. Boot the host into a provisioning image over the pre-auth network path (same low-impact ACL).
2. Generate the EAP-TLS keypair inside the TPM; persist at the fleet-standard handle (`tpm2_evictcontrol`).
3. Produce a CSR with the host's designated CN/SAN; submit to the machine-identity CA.
4. Write the issued certificate to the fleet-standard NV index (`tpm2_nvdefine + tpm2_nvwrite`).
5. Enrol Secure Boot trust: fleet PK/KEK/db via sbctl or Redfish (or MOK via MokManager on the shim path).
6. Enrol the HTTPS Boot CA into the TlsCaCertificate EFI variable (`efivarfs` or BMC/Redfish).
7. Register the certificate identity with RADIUS / NAC, and — if the NVMe target filters hosts — add the derived hostnqn to the target allow-list (NQNs are a deterministic function of certificate serials).

After enrolment the host is stateless: replace releases at will; identity and trust survive in silicon and firmware NVRAM. Decommissioning = revoke the certificate, clear the handle/NV index, and remove the NQN from any allow-list.

9. Test Rig (Proxmox)

Proxmox bridges are dumb L2 — the authenticator is software. Per test “port”: an isolated vmbr (no physical NIC) joining the system-under-test to an authenticator VM running hostapd (driver=macsec_linux, ieee8021x=1; needs CONFIG_MACSEC + CONFIG_DRIVER_MACSEC_LINUX — expect to compile) with local FreeRADIUS. **Mandatory bridge tweak:** EAPOL’s destination MAC (01:80:C2:00:00:03) is bridge-filtered by default — set group_fwd_mask=8 on the test bridge. The SUT VM gets a swtpm-backed vTPM (provisioned by running the real enrolment flow) and OVMF, whose TlsAuthConfigDxe supports enrolling the boot CA, so the entire chain — HTTPS UKI fetch, Secure Boot verification, EAP-TLS from the vTPM, MKA, re-DHCP on macsec0, NVMe/TCP connect, verity root, @-daemon pivot survival — is testable with zero physical infrastructure. The serving stack runs as the same SPDK + HTTPS compose used in production, fed by the release artifact image. Known emulation gaps: pre-auth ACL semantics and RADIUS-driven dynamic VLAN assignment (test the re-DHCP logic by changing DHCP scope post-auth instead). MTU: MACsec adds 32 bytes — raise bridge MTU or accept macsec0 at ~1468 to avoid fragmenting root I/O.

10. Security Properties and Known Limitations

Property / limitation	Detail
Key non-exportability	Private key never exists outside the TPM; theft of any artifact yields no credential.
Whole-chain integrity	Firmware verifies one UKI signature; via the embedded roothash, that covers every root filesystem block read for the life of the system.
No boot-state attestation	The TPM key is auth-value protected, not PCR-sealed: 802.1X proves “this machine’s TPM”, not “this machine in a trusted boot state”. PCR policy or remote attestation is a separate design effort if required.
Pre-auth exposure	The UKI traverses the wire before MACsec exists. Mitigated by HTTPS transport and, decisively, Secure Boot verification; the pre-auth ACL bounds what an unauthenticated port reaches.
TlsCaCertificate is soft	Typically writable from a root OS context. Transport protection only; integrity rests on Secure Boot.
Signing-key custody	The fleet signing key boots arbitrary code on every host. CI/HSM custody, never on build infrastructure; rotation path exists via KEK-signed db updates.
Rollback window	Old signed UKIs remain bootable until dbx-revoked. Operational control (publish current+previous only) plus periodic dbx pushes for superseded builds.
Supplicant / serving stack as SPOFs	@wpa_supplicant death ⇒ MKA timeout (~6 s) ⇒ root link drops (watchdog it; recovery is reboot). Serving-stack outage beyond ctrl-loss-tmo hangs the fleet (replicate it).
NM/PKCS#11 fragility	Wired EAP-TLS with PKCS#11/TPM keys under NetworkManager has a history of defects (reports as recent as 2025); NM is deliberately excluded from the authenticated link.

11. Bench Verification Checklist

- Network policy is should-secure (low-impact), not must-secure, on fleet access ports; agree MACsec cipher suite, replay window, confidentiality vs integrity-only.
- RHEL wpa_supplicant: CONFIG_MACSEC enabled; TPM key reference method (OpenSSL provider handle: vs PKCS#11 URI) — prototype with pure wpa_supplicant first.
- erofs module present on the target kernel (modprobe erofs); ext4-^has_journal fallback wired into the pipeline if not.
- systemd.verity_root_options=hash-offset support in the shipped systemd/dracut; two-namespace fallback otherwise.
- Pivot behaviour: @wpa_supplicant survives switch-root; MKA continuity (no SAK renegotiation); NVMe/TCP connection rides through.

- Re-DHCP-after-auth ordering with production VLAN assignment in place.
- TlsCaCertificate enrolment per hardware model (efivarfs and Redfish); firmware hostname-verification behaviour against a valid-chain/wrong-name certificate.
- Secure Boot ownership per model: PK/KEK/db enrolment via Redfish; option-ROM dependency on the Microsoft Third-Party CA checked before removing it.
- TPM NV capacity and read-chunk behaviour on deployed TPM parts; fleet handle and NV index values frozen.
- Enrolment CA integration (online CSR vs batch); certificate lifetime/renewal flow; hostnqn allow-list feed from enrolment.
- MACsec offload validated with MKA on the target NIC/kernel if 10/25 GbE.
- CI smoke test green: qemu/Proxmox + swtpm + hostapd/FreeRADIUS (group_fwd_mask=8), asserting macsec0 up, verity root mapped, overlay rw, pivot survival.

Prepared with Claude (Anthropic) — design discussion capture v2, 12 June 2026. Companion repository: Packer pipeline (diskless.pkr.hcl), 95dot1x-macsec dracut module, extract-artifacts.sh and build-oci.sh post-processors, and CLAUDE.md. Technical claims around MKA timing, firmware behaviour, and distribution build options must be bench-verified per the checklist before committing the fleet design.