Threat model

Source: docs/threat-model.md · last edited 2026-07-13

Assets

- **Prompt & completion content** (primary) — e.g. patient records, legal documents, proprietary code.
- **Persisted tenant data** — KV cache, logs, RAG stores, fine-tunes / adapters.
- **Model weights / IP** (secondary) — protected as a side effect of enclave memory encryption.

Adversaries we defend against

Adversary	Defence
Honest-but-curious operator / insider (Liquify staff with host/root access)	Plaintext only inside the enclave; host cannot read enclave memory (CC mode). Operator sees ciphertext + metadata only.
Compromised host OS / hypervisor	CPU TEE (SEV-SNP/TDX) encrypts VM memory; GPU CC encrypts HBM + PCIe. Compromise of the host does not yield plaintext.
Man-in-the-middle / malicious gateway	Client encrypts to the *attested enclave key*, not to whatever terminates TLS. A MITM sees ciphertext; it cannot produce a valid attestation for a key it controls.
Enclave running modified/backdoored code	Measurement changes → not on the client's pinned allow-list → client refuses (verified live in `demo_tamper.py`, and through the hardware path: modified code ⇒ different MRTD/launch measurement).
Attestation forgery	Evidence must chain to the pinned silicon root — Intel's SGX Root CA or AMD's ARK. A forger without the vendor's key is rejected. On Intel this includes the step people skip: the quote body is signed by an *ephemeral* attestation key, which is worthless unless the Quoting Enclave's report vouches for it. We check that.
Replay / relay	Per-request nonce bound into the signed `report_data`; AEAD associated data binds ciphertext to the specific attested measurement. Hardware quotes carry no timestamp — freshness comes from the nonce, which is the stronger guarantee.
Debuggable enclave	A TD with the DEBUG attribute set, or an SEV-SNP guest whose policy allows DEBUG, lets the host read guest memory. The quote is perfectly valid; the environment is not confidential. The client rejects both.
Memory exfiltration via migration	An SEV-SNP guest policy permitting a migration agent can be asked to export the guest's memory. Rejected by default (`forbid_migration_agent`).
TCB downgrade	Client enforces a TCB floor. On AMD this is **componentwise** (bootloader/TEE/SNP/microcode), because a rollback of *one* component — e.g. microcode — is invisible to a single scalar and is a real attack.
Stale / wrong-chip endorsement key (AMD)	The VCEK's AMD extensions are checked against the report's `chip_id` and `reported_tcb`, so a genuine certificate issued for another chip or an older TCB cannot be paired with this report.
Less-privileged component attesting (AMD)	The report's VMPL must be 0 (`require_vmpl0`, default on). A component running at VMPL > 0 cannot present a report binding its own key as if it were the trusted monitor.

Adversary	Defence
Firmware pinning silently disabled (GPU)	Requiring GPU attestation with an *empty* firmware allow-list fails **closed** (refuses), not open — so an operator can't accidentally accept any VBIOS while believing attestation is enforced.
Cross-vendor confusion	The trust policy holds one entry per silicon platform. Evidence selects exactly one entry by backend, so AMD evidence is never judged against Intel's roots or allow-list, and evidence from an unlisted platform is refused before any crypto runs.
Other tenants	Per-request ephemeral keys; per-tenant at-rest keys (BYOK) so one tenant cannot read another's data or persisted state.

What we explicitly do NOT protect against (residual risk)

- **A genuine hardware TEE break or side-channel** (Intel/AMD/NVIDIA silicon flaw). The guarantee rests on the TEE being sound. This is the honest ceiling of the claim.
- **Malicious code that is nonetheless the *expected*, attested build.** Attestation proves *which* code runs, not that the code is bug-free. Mitigation: open source the enclave image + reproducible builds so the measured code is auditable.
- **Modified workload under an approved firmware measurement (TDX, on hardware).** On real TDX, MRTD typically measures only the initial TD memory (firmware/TDVF); the kernel, initrd, cmdline and often the application land in the runtime registers RTMR0–3. Pinning MRTD alone would let a host boot approved firmware with attacker-modified software and still produce a valid quote. **Mitigation (required before production): the reproducible build must establish which registers cover the workload, and the release manifest must pin them** via `PlatformPolicy.allowed_rtmrs`. Off-hardware this repo's simulator folds the app into MRTD, which hides the issue — it is flagged in code (`attest_tdx.WARN_MRTD_ COVERAGE`) and is a `TODO(hw)`.
- **Traffic analysis / metadata.** Request timing, sizes, tenant identity, and token counts are visible to the operator by design (needed for routing/billing). Content is not.
- **A malicious *client*.** Out of scope — we protect the client's data from us, not us from the client (rate-limiting/abuse controls are a separate concern).

Off-hardware caveats (development only)

Until the enclave runs on a real CVM, the **root of trust is simulated** and this is **not for real secrets**. Precisely what is and isn't simulated:

	Simulated off-hardware	Real, and tested today
Quote/report **bytes**	—	genuine TD Quote v4 and 1184-byte SEV-SNP layouts
Verification logic	—	production `TdxVerifier` / `SevSnpVerifier`: chain walking, QE binding, VCEK↔chip binding, endianness, `report_data`, debug/policy bits, TCB floors
Vendor PKI	stand-in Intel/AMD CAs generated locally	Intel SGX Root CA / AMD ARK on hardware

	Simulated off-hardware	Real, and tested today
Quote source	`LIQUIFY_HWSIM=1` builders	configfs-TSM / vendor ioctl on a CVM (`TODO(hw)`)
Live collateral	none	Intel PCS TCB info + QE identity + CRLs; AMD KDS VCEK fetch
GPU CC state	simulated NVIDIA report	NVIDIA on-GPU verifier + NRAS (Phase 2)

Consequences to be honest about:

- Anyone who can run the simulated PKI can mint "valid" evidence for it. This is harmless against a client pinning **real** vendor roots — the chain terminates at the wrong CA — so the residual risk is an *operator publishing a manifest that pins simulated roots*. The manifest records `simulated: true`, the roots say `SIMULATED` in their subject, and the SDK prints a warning on every run that loads one.
- The TCB floor currently comes from **our signed manifest**, not from Intel's or AMD's live TCB recommendation. A platform the vendor has since declared vulnerable would still pass if our manifest hasn't been updated. Closing this needs PCS/KDS collateral (Phase 1 remaining).
- `crypto.py` is a compact HPKE-*style* construction, not a certified RFC 9180 library.
- Transport is plain HTTP; RA-TLS is Phase 1.

The **verification logic and security properties** are real and tested — hardware swaps the simulated roots for silicon roots without changing the client SDK or wire format.

The trust story we sell (and can back up)

1. **Open-source client SDK** — customers verify attestation themselves; they don't take our word for it.
2. **Reproducible enclave builds + published measurements** — customers (or auditors) independently reproduce the measurement they pin.
3. **BYOK at-rest keys** — customers hold the root for persisted data; key revocation is provable deletion (crypto-shredding).
4. **Content-blind control plane** — routing, metering, and logging operate on metadata only; this is an architectural discipline, enforced in review.