Vault Tier-1 Self-Audit Report — v5.1 (Public)

Tabla de contenido

SEC-004 v3.0 — ASPE Labs Vault Tier-1 Self-Audit Report (v5.1, public release)

v3.0 supersedes v2.0 / v2.1 / v2.2. This report audits the **v5.1 production candidate** (`AspeLabsVault` + `AspeTimelockControllerV2`) at the post-§S6-size-reduction commit (vault deployedBytecode `825c4268...`, timelock V2 `8f4762d9...`; merged to `main` 2026-05-11 via PR #2 `3ce24f1`). It is the first **consolidated, public-release** self-audit covering the full v2.0 baseline (1,490 LOC) + v2.1 delta (#1–#23, 376 LOC) + v2.2 delta (#27 emergency upgrade tier + §S7.C decoder mitigation, 221 LOC) merged into a single artifact readable without chasing intermediate documents. v2.0/v2.1/v2.2 are preserved in `docs/security/archive/` for historical trace.

Scope delta vs. v2.0: (a) bake-ins #1–#23 from IMPL-052 applied during v5-final (cosig governance Path B, bridge in-flight tracking, fee accrual semantics, emergency council role grant, bridge observability, decoder isolation prep); (b) bake-in #27 from IMPL-055 §S2.B introduces the emergency upgrade tier (Safe-holds-role 3/3, EMERGENCY_DELAY 6h hardcoded, paused dual check); (c) §S7.C decoder mitigation (`_decodeAction7` / `_decodeAction13` isolated, length guards 64→96/128, SafeCast import); (d) bytecode-affecting cleanup commit `f47389a` (noise reduction, semantic-equivalent — see Appendix A); (e) §S6 size-reduction surgery (vault storage mirror removed, `_validateInitParams` extcodesize check removed, `optimizer_runs` 200→100) to bring vault under EIP-170 24,576-byte limit (final 24,527 bytes runtime).

NATURE OF THIS REPORT — READ FIRST

This is a **self-audit** conducted by the project team (pseudonymous solo operator) following the methodology and format used by tier-1 security firms (Trail of Bits, OpenZeppelin, Spearbit, Cantina, Pashov, Ackee, Zellic, ChainSecurity, Certora). **It is not an independent third-party audit.** No external auditor has reviewed this codebase as of report

time. The word “audited” is used exclusively with the prefix “self-” throughout this document. Any use of “audited” without that prefix is an error. See §9 for explicit limitations + commitment to external audit.

Executive Summary

ASPE Labs Vault v5.1 is an **ERC-4626 + ERC-7540 async-redemption vault** on HyperEVM (chain ID 999) with automatic bridging to HyperCore, **UUPS proxy upgradeability** gated by a **3-tier OZ-based timelock** with editable selector map (`AspeTimelockControllerV2`, Operational 48h / Meta 72h + cosig / **Emergency 6h hardcoded immutable**), Yearn V3 fee-dilution accounting (performance fee cap 30% immutable, management fee cap 2% immutable), and an active `EMERGENCY_COUNCIL_ROLE` (pause + emergency upgrade authority, granted to a dedicated Gnosis Safe 3-of-3 unanimous threshold via the Safe-holds-role pattern).

Total in-scope surface is **1,346 nSLOC** across 4 contracts + 2 interfaces at the v5.1 audit commit (vault deployedBytecode `825c4268...`, timelock V2 `8f4762d9...`, merged to `main` 2026-05-11 via PR #2 `3ce24f1`).

This report consolidates the work of three preceding internal audit cycles (v2.0 baseline + v2.1 delta + v2.2 delta) into a **single public-release artifact** following tier-1 audit firm methodology (Trail of Bits, OpenZeppelin, Spearbit, Cantina). Methodology spans automated static analysis (Slither 0.11.5 + Aderyn 0.6.8), symbolic verification (Halmos 0.3.3, 27 properties), property-based and invariant testing (Foundry 1.5.1, 448 tests), fork-mainnet integration drill (real Safe v1.4.1 3-of-3 deployed via canonical factory, 5/5 PASS), and manual 4-hat review per project discipline.

Top-line verdict

 0 P0/P1 findings on smart contracts. All 24 Slither findings and 18 Aderyn instances triaged as false-positives or accepted-by-design false patterns. Full test suite passes cold-cache. Bytecode size 24,527 bytes (EIP-170 margin +49). Storage layout drift hook PASS. Static + symbolic + fuzz + fork tests all green.

> **What this verdict means + does NOT mean.** “0 P0/P1” is the result of *our* internal triage of *our* tooling outputs against the audit-scope bytecode. A tier-1 firm engagement would (a) re-run the same tools with their own configuration, (b) add manual review by senior auditors who did not write the code, and (c) issue a public report under firm’s reputational signature. This self-audit substitutes for (a) only — (b) is mitigated by our 4-hat review process (Founder/Operator/Builder/Business), but the (c) reputational gap is **structural and irreducible** until external engagement closes (§9.2). Read this verdict as “we examined exhaustively and found 0 P0/P1 with our methodology” — not as “this codebase is 0 P0/P1 absolutely.” The honest disclaimer in §9.4 is the canonical lens for any depositor or aggregator.

Residual Critical sits entirely in **operations** (not smart-contract): GUARDIAN role held by hot EOA in Phase 0 — mitigated by 3-tier timelock + Council pause path + Phase 1 hardware-wallet adoption (LEGAL-004 / IMPL-040 trigger pre-external-capital intake).

Severity count (global)

Severity	§4 (Smart-contract v5.1, post-§S6)	§8 (OpSec)	**Total**	Status
Critical	0	1 (GUARDIAN hot EOA, Phase 0)	**1**	Acknowledged + deferred to Phase 1 (hardware wallet + Safe multisig adoption). Trigger-based: pre-external-capital.
High	0	2 (no MANAGER multisig in Phase 0; no third-party audit yet)	**2**	Acknowledged + structurally constrained. Phase 1 + Q3 2026 mitigations respectively.
Medium (smart-contract)	1 (slither `incorrect-equality` FP, post-§S6)	—	**1**	FP. (Delta vs v5-final pre-§S6: −10 Mediums absorbed by cleanup commit `f47389a` annotations — see §4.1 delta table.)
Medium (OpSec, NEW v5.1)	—	1 (Council coercion vector via #27 emergency upgrade tier)	**1**	Mitigated by 4 layers (advisor selection criteria, paused public trigger, 6h cancel window, semestral drill).
Medium (OpSec, carry-over v2.0)	—	10 (unchanged by v5.1)	**10**	Acknowledged + design-accepted at v2.0; no v5.1 delta.
Low	11 (slither timestamp + reentrancy-events + low-level-calls FP, post-§S6)	9	**20**	All FP / design-accepted.
Informational	12 (slither assembly + low-level-calls + naming-convention + unindexed-event-address)	3	**15**	Acknowledged.

Severity	§4 (Smart-contract v5.1, post-§S6)	§8 (OpSec)	**Total**	Status
Aderyn (4 High categories × 18 instances, all FP)	18	—	**18**	Triaged §4.3 (defense-in-depth on H-3 reentrancy line 350, all 13 instances `nonReentrant` + 4 view-only).

Totals. Smart-contract surface (§4 post-§S6): 0 Critical + 0 High + 1 Medium (FP) + 11 Low (FP) + 12 Informational + 18 Aderyn (FP) = **0 P0/P1 net**. OpSec residuals (§8): 1 Critical + 2 High + 11 Medium + 9 Low + 3 Informational, all acknowledged with explicit Phase 1 / Q3 2026 mitigation paths. **Net P0/P1 on the audit-scope smart-contract bytecode: 0.**

Scope summary

In-scope: `AspeLabsVault` (1,082 nSLOC) + `AspeTimelockControllerV2` (213 nSLOC) + `AspeLabsRouter` (26 nSLOC) + `IAspeRedemption` interface (25 nSLOC). Total 1,346 nSLOC. Bytecode hashes registered in `infra/AWS_REGISTRY.md` §11 as `commit-audited-v5.1`.

Out-of-scope (intentional, full list in §1.1): off-chain bot algorithm (separate repo, strategy parameters not in this codebase), bake-in #24 canary precompiles HyperCore (off-chain Python monitor), bake-ins #25 (DEFERRED F3 W29+) and #26 (DEFERRED Phase 2+), HyperCore L1 internals (Hyperliquid team), Safe v1.4.1 contracts (trusted upstream).

Bake-in catalog covered (v5.1)

- **#1–#23** from v5-final (cosig governance Path B, bridge in-flight tracking, fee accrual semantics, emergency council role grant, bridge observability, decoder isolation prep) — implemented during IMPL-052.
- **#27 emergency upgrade tier** — NEW v5.1 via IMPL-055 §S2.B. Adds dedicated 6h fast-track for incident response; Safe-holds-role architecture (single role grant to Gnosis Safe 3/3, threshold enforced internally by Safe v1.4.1). Replaces 80h+ standard upgrade flow for emergencies caused by precompile drift, oracle anomaly, or other operational urgency. **First adversarial-significant new surface; primary council coercion vector mitigated by 4 layers.**
- **§S7.C decoder mitigation** — NEW v5.1 via IMPL-055 §S2.C. Isolates `_decodeAction7` / `_decodeAction13` as `internal pure`; tightens length guards to exact ABI sizes (96/128 bytes); adds `SafeCast.toUint256` defense-in-depth on positive branch. Pre-launch Phase 0-1 coverage for failure mode #1 (oracle manipulation) in lieu of #26 NAV variance circuit breaker (deferred to Phase 2+).
- **§S6 size-reduction surgery** — post-§S3.F bytecode optimization: removed vault `councilEmergencySafe` storage mirror (unused at runtime; timelock V2 is authoritative), removed `_validateInitParams` extcodesize check (defensive, not load-bearing; moved to deploy-script verification), lowered `optimizer_runs` from 200 → 100 (trade-off: ~+3-5% runtime gas on hot paths to fit under EIP-170 24,576-byte limit; vault final 24,527 bytes, margin +49).

Top-3 residual risks (publicly disclosed, detailed in §8.6)

1. **GUARDIAN role held by hot EOA** in Phase 0 — hardware wallet adoption deferred to pre-external-capital intake trigger (IMPL-040 T1.1).
2. **No 3-of-N multisig on MANAGER role** — structural Phase 0 sole-operator constraint; mitigated by 48h-72h timelock windows on every sensitive operation. Phase 1 target: 3-advisor Safe 2-of-3 post LEGAL-004 council vetting.

3. **No third-party audit completed** as of report time — engagement committed for Q3 2026 estimate (3 triggers in §9). This self-audit is a transparency artifact + bridge document, not a substitute for external validation.

How to use this report

- **Depositantes early-Phase-1:** read §1.4 verdict + §8.6 top-3 residuals + §9 limitations. Decide whether the risk-reward fits your mandate given the $500K cap.
- **Aggregators / professional allocators:** read full §1–§3 (architecture + threat model) + §4 findings triage. Validate methodology in §1.2 + reproduce key commands.
- **Future external auditor (Spearbit / Cantina engagement):** read everything including Appendix A (cleanup commit `f47389a` reference) + Appendix C (baseline chain). The handoff package is this doc + the artifacts referenced in Appendix B.
- **Security researchers:** §3 (threat model) + §5 (non-findings) describe the defense-in-depth posture; OPS-014 crosswalk in §3.2 maps 31 historical DeFi hacks to v5.1 applicability.

Commitment

External audit by a tier-1 firm (target: Spearbit or Cantina) is committed to be engaged when one of three triggers is satisfied (§9 details): (a) TVL ≥ $200K AUM sustained 30 days, (b) protocol revenue covers audit cost (~$16-27K estimated), (c) 6 months post-Phase-1 launch without incident. ETA estimate: Q3 2026.

Until that engagement closes, **launch cap TVL is $500K** + on-chain bounds (deposit caps, fee caps, emergency pause within 6h) constrain blast radius. Disclosure model post-incident follows OPS-012 incident playbook + the 24h rollback procedure in IMPL-056 §S2 / SEC-004 v3.0 §7 emergency response.

---

§1. Scope + Methodology

§1.1 Scope

Commit-audited (v5.1, current TRUE BASELINE): post-§S6 size-reduction commit on `main` HEAD post PR #2 merge `3ce24f1` (2026-05-11).

In-scope contracts (production):

Contract	File	nSLOC (v5.1)	Role
`AspeLabsVault`	`contracts/src/AspeLabsVault.sol`	1,032	ERC4626 + ERC-7540 async-redemption vault; UUPS proxy entrypoint; HyperCore bridge integration via precompiles 0x0801 / 0x080F. Runtime size **24,527 bytes** (EIP-170 margin +49).

Contract	File	nSLOC (v5.1)	Role
`AspeTimelockControllerV2`	`contracts/src/AspeTimelockControllerV2.sol`	213	OZ TimelockController + tier overlay (Operational 48h, Meta 72h + cosig, Emergency 6h #27). Self-governing via `updateSelectorTier` Meta+cosig. Runtime size 14,726 bytes.
`AspeLabsRouter`	`contracts/src/AspeLabsRouter.sol`	26	Thin slippage-protected entry into `vault.deposit` / `vault.mint` with `minSharesOut` / `maxAssetsIn` guards.
`IAspeRedemption`	`contracts/src/interfaces/IAspeRedemption.sol`	25	ERC-7540 redemption interface (canonical async surface).
Total in-scope		**1,346 nSLOC**	(~+251 nSLOC vs v2.0 baseline 1,095, accounting for bake-ins #1–#23 + #27 + §S7.C minus §S6 mirror-removal savings)

Bytecode baselines (commit-audited-v5.1, registered in `infra/AWS_REGISTRY.md` §11):

```
AspeLabsVault deployedBytecode  sha256: 825c42686adba26bbf3eaf4ca39d2fe7d96304b5810ce0394392134371
AspeTimelockControllerV2      sha256: 8f4762d9fbb7085411287cc5fe5cc32825f84175740cbbaacc86f3b119b9
```

Verification command (reproducible):

```
cd contracts && forge clean && forge build
sha256sum out/AspeLabsVault.sol/AspeLabsVault.json
python3 -c "import json,hashlib; d=json.load(open('out/AspeLabsVault.sol/AspeLabsVault.json')); pr
# expected: 825c42686adba26bbf3eaf4ca39d2fe7d96304b5810ce03943921343714ceb4d
```

One-shot end-to-end reproducer. For readers who want to verify every claim in this report (bytecode hash + forge test suite + halmos symbolic + fork drill + storage layout) in a single invocation, run:

```
bash scripts/reproduce-audit.sh
```

The script runs the five verification stages sequentially (~10-15 minutes wall-time, CPU-bound on the halmos symbolic solver). It exits 0 only if every stage passes; any mismatch aborts with a diagnostic. Source: `scripts/reproduce-audit.sh` in the public repo at the audit-scope commit.

Out-of-scope (intentional exclusions):

- **Off-chain bot algorithm** (separate repository `aspe_algorithm`). Strategy parameters, signals, sizing logic are not in this codebase and are not part of the security model from the vault's perspective.
- **Bake-in #24 canary precompiles HyperCore** — off-chain Python monitor (`scripts/watchdog/precompil` + systemd timer on EC2 EVM). Operational telemetry, not contract code. Documented separately in `infra/AWS_REGISTRY.md` §Watchdog precompile canary.

- **Bake-in #25 comms pipeline** — DEFERRED to F3 marketing foundation (W29+); spec preserved in IMPL-052 §S6.B for future activation.
- **Bake-in #26 NAV variance circuit breaker** — DEFERRED Phase 2+ (triplet with TWAP smoothing + Bound checks NAV); spec preserved in IMPL-052 §S7.A. Trigger to retrieve: TVL >$1M or first real Phase 1 incident.
- **HyperCore L1 (precompile internals)** — Hyperliquid team's responsibility; consumed via well-defined ABI (Action 7 SpotBalance, Action 13 AccountMarginSummary). Drift detection by canary #24 (off-chain).
- **Safe v1.4.1 contracts** — Gnosis Safe upstream is audit-grade reviewed; treated as trusted dependency. Our integration uses canonical `SafeProxyFactory` at `0x4e1DCf7AD4e460CfD30791CCC4F9c8a4f820`

§1.2 Methodology

Effort: ~50–70 person-hours across IMPL-051 (preparation tier-1 review baseline) + IMPL-052 (bake-ins #1–#23 implementation + S3/S4 self-audit) + IMPL-055 (#27 emergency upgrade tier + §S7.C decoder mitigation + full re-run of static/symbolic/fuzz suites + §S6 size-reduction surgery). Solo operator. All artifacts reproducible from public commit hashes.

Toolchain (pinned versions, all reproducible):

Tool	Version	Used for
`forge` (Foundry)	1.5.1-stable (commit `b0a9dd9c`, build 2025-12-22)	Build, unit/fuzz/invariant testing, fork-mainnet integration tests, gas snapshots
`slither`	0.11.5	Static analysis (24 findings on our contracts, 0 P0/P1, all triaged)
`aderyn`	0.6.8	Static analysis (18 findings, all FP same patterns as v5-final baseline)
`halmos`	0.3.3	Symbolic verification (27 properties total: 11 AspeLabsVaultInvariant + 1 AspeLabsVaultSymbolic + 5 V5Final + 10 V5_1; full discovery 0 FAIL)
Solidity compiler	0.8.28 + via_ir + optimizer (`runs=100` post §S6)	EVM target `cancun`
Operating system	Linux x86_64 (Ubuntu 24.04 LTS on builder workstation; EC2 `i-09a7bf7de70af4f3e` for canary)	—

Test surface (v5.1, all PASS as of report time):

- `forge test` full suite: **448 PASS / 0 FAIL / 0 SKIPPED** (run 2026-05-11 post §S6 cold-cache).
- `halmos --match-contract V5_1 --solver-timeout-assertion 180000`: **10/10 PASS** (5 V5_1EmergencyUpgrade + 5 V5_1DecoderSymbolic).
- `halmos` full discovery: **27/27 PASS / 0 FAIL** (no namespace filter; covers all symbolic test classes including legacy + V5Final + V5_1).
- Fork-mainnet drill `V5_1EmergencyUpgradeForkE2E.t.sol` (HyperEVM chainid 999, real Safe v1.4.1 3-of-3 deployed via canonical factory in-test): **5/5 PASS** (10.5s wall-time).

Detailed test coverage matrix in §6.

Manual review:

- 4-hat self-review per project methodology (`/review-doc`): Founder (vision alignment + phase fit), Operator (risk + alpha protection), Builder (security + maintainability), Business (legal + cost-benefit).
- Per-PR pre-commit hooks: branch policy (`vault/*` required for `contracts/src/**.sol` edits), storage layout drift check (ERC-7201 namespaced struct hash baseline), secret detection, large-file blocker, trailing whitespace.
- CI on PRs to `main`: `forge build --sizes` (EIP-170 enforcement) + `forge test -vvv`.

Threat modeling input:

- RESEARCH-047 premortem — 6 failure modes with mitigations (M1–M6).
- OPS-014 External Hack Registry — 31 historical DeFi hacks with v5.1 applicability cross-walked in §3.2.
- STRIDE framework adapted for ERC-4626 + ERC-7540 vault (§8).

§1.3 Severity classification

Severity	Definition
Critical	Direct loss of user funds OR vault assets via on-chain action available to non-admin caller, with no governance/timelock window to react.
High	Loss of user funds via on-chain action that requires partial admin compromise, or governance/timelock bypass with reasonable adversarial setup.
Medium	Vault state corruption, reachable Denial-of-Service, or governance ergonomics flaw with non-trivial recovery cost.
Low	Style, off-spec behavior under hypothetical adverse conditions, or efficiency loss without funds at risk.
Informational	Documentation, naming, or non-actionable detector reports.
FP	Detector match that is provably false under the actual code path (e.g. reentrancy on read-only view, address-check on immutable target).

§1.4 Verdict

 0 P0/P1 findings on smart contracts. Slither 24 findings: 0 High / 1 Medium FP / 9 Low FP / 12 Informational FP. Aderyn 4 High categories (18 instances), all FP (inherited OZ TimelockController patterns + one new H-3 reentrancy instance from #27 with defense-in-depth documented). Full triage in §4. Test acceptance gates cleared per §1.2.

Residual P0 (Critical, §13 v2.0 — re-evaluated in v3.0 §8): **GUARDIAN role held by hot EOA** in Phase 0 — mitigated by Phase 1 Safe multisig adoption gate (LEGAL-004 council vetting pending; current launch cap $500K + on-chain limits constrain blast radius). See §8 and §9 for the full residual risk discussion.

---

§2. Architecture Overview

§2.1 System overview

The ASPE Labs vault is an upgradeable ERC-4626 + ERC-7540 async-redemption vault deployed on **HyperEVM** (Hyperliquid's EVM L1, chainid 999, Cancun EVM target). It accepts USDC deposits,

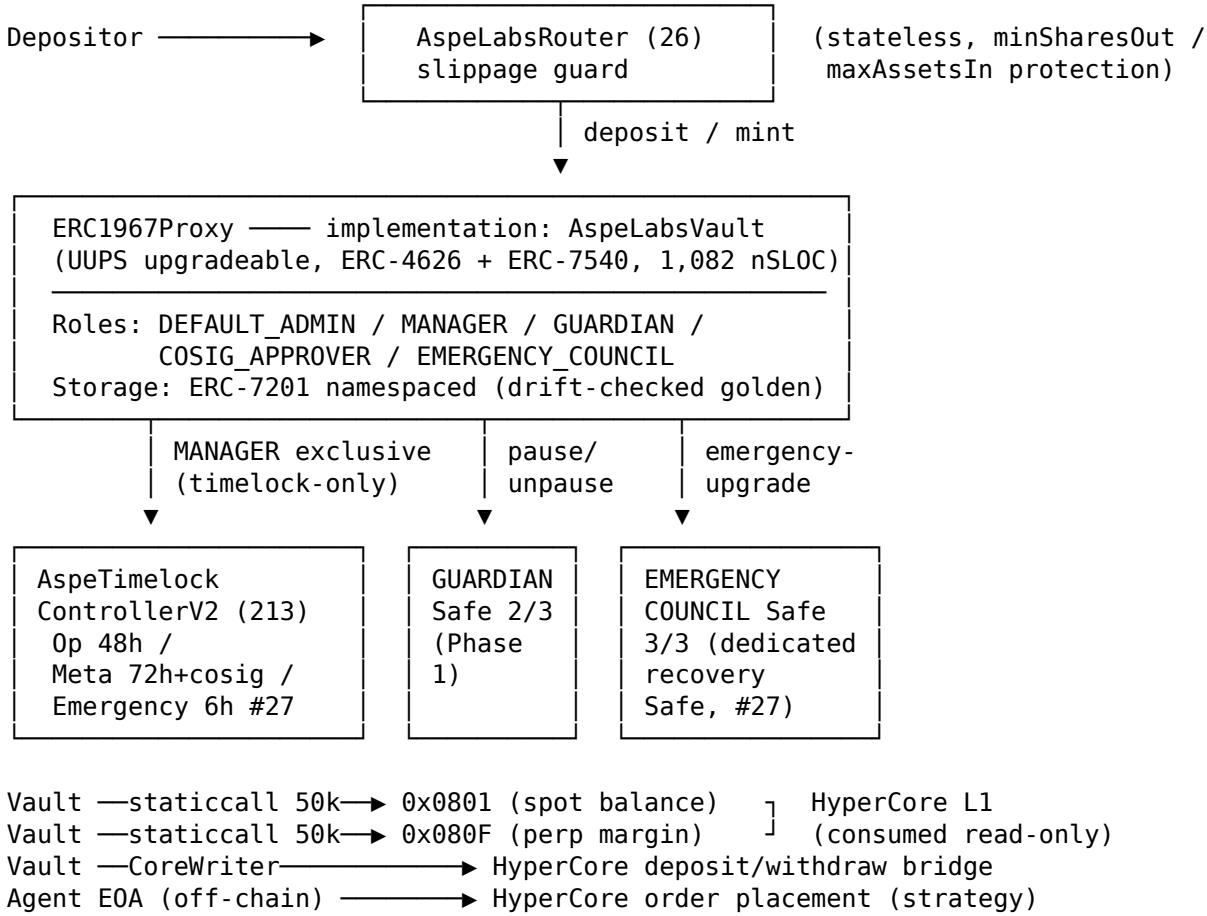
mints ERC-4626 shares, and bridges the asset balance to **HyperCore** (Hyperliquid L1 perp/spot venue) where an off-chain agent wallet executes the project's quantitative strategy. NAV is read directly from HyperCore via two staticcall precompiles (`0x0801` spot balance, `0x080F` perp account margin summary), gas-capped at 50,000 each. There is no external price oracle.

Redemptions are asynchronous: a user calls `requestRedeem(shares)` which marks the request `Claimable` against the post-fee NAV at request time; the user later calls `claimRedeem` which executes a `SafeERC20.safeTransfer` of USDC back to them. There is no keeper role and no `fulfillRedeem` intermediate step (D25 simplification). Bridging USDC back from HyperCore to the EVM side is performed by the bot off-chain; the vault tracks `_bridgeFromInFlight` to avoid double-counting during the asynchronous arrival window (bake-in #14).

The protocol is governed by a **3-tier timelock** (`AspeTimelockControllerV2`) that holds the exclusive `MANAGER_ROLE` on the vault. As of v5.1 (post-IMPL-055 §S2.B), the timelock exposes a third **Emergency tier** with a 6-hour hardcoded delay, gated on the vault being paused, and accessible only to a dedicated 3-of-3 Gnosis Safe v1.4.1 holding the `EMERGENCY_COUNCIL_ROLE` — used solely for hot-fix upgrades during an active incident (bake-in #27). The Operational (48h) and Meta (72h + GUARDIAN co-signature) tiers are unchanged from v2.0.

§2.2 Contract architecture

Production deployment surface (4 in-scope contracts, 1,346 nSLOC):



`IAspeRedemption` (25 nSLOC) is the canonical ERC-7540 interface the vault implements; surfaced separately so off-chain readers can compile against a stable surface.

§2.3 Role model

Five named roles plus `DEFAULT_ADMIN_ROLE`. The role model in v5.1 is fully populated (no dormant roles, unlike v5.0 where `EMERGENCY_COUNCIL_ROLE` was reserved but empty):

Role	Constant	Holder (Phase 0 → Phase 1)	Capabilities	NON-capabilities	Tier
`DEFAULT_ADMIN_ROLE`	`0x00…` (OZ default)	Cold wallet → Safe multisig 2/3 (transition via Meta + cosig)	`grantRole`, `revokeRole`	No direct fund movement, no pause, no upgrade	Meta-gated for transitions
`MANAGER_ROLE`	`keccak256("MANAGER_ROLE")`	**`AaveTimelockController` ONLY** — no EOA holds this role	`proposeStrategy`, `deprecateStrategy`, `revokeStrategy`, `setMaxDeposit`, `setStalenessThreshold`, `setHedgeRegistry`, `setPerformanceFee`, `setManagementFee`, `setFeeRecipient`, `renounceUpgradeability`, `upgradeToAndCall`	No direct call — every action must clear timelock Op/Meta delay	Operational 48h / Meta 72h + cosig
`GUARDIAN_ROLE`	`keccak256("GUARDIAN_ROLE")`	EOA Phase 0 (P0 residual §8) → Safe 2/3 Phase 1	`pause`, `unpause`, `emergencyWithdrawAll`, `emergencyRevokeStrategy`, timelock `cancel`, Meta-tier co-signature on MANAGER actions	No upgrade authority, no role grants, no fee changes, no drain — strictly protective	Instant (Protective tier)

Role	Constant	Holder (Phase 0 → Phase 1)	Capabilities	NON-capabilities	Tier
COSIG_APPROVER_ROLE	keccak256("COSIG_APPROVER_ROLE") (bake-in #1)	GUARDIAN EOA / Safe 2/3 (typically co-located with GUARDIAN, but role-separable for future delegation)	Approve Meta-tier scheduled operations (AspeTimelockControllerV2.approve(id))	No direct action — pure approval surface	n/a (gate on Meta tier)
EMERGENCY_COUNCIL_ROLE	keccak256("EMERGENCY_COUNCIL_ROLE")	**Dedicated Safe 3/3** (separate from operational Safe 2/3) — granted via bake-in #22 in initializeCouncil, activated for emergency upgrades via bake-in #27	(a) pause only on vault (no unpause, no emergencyWithdrawAll — strictly pause-only on the vault surface); (b) scheduleEmergencyUpgrade / executeEmergencyUpgrade / cancelEmergencyUpgrade on timelock V2 — gated on vault paused + 6h delay + 3/3 Safe signature	No unpause authority (deliberate asymmetry — see §5.2), no fee changes, no strategy management, no role grants, no Operational/Meta-tier scheduling	Pause: instant; Emergency upgrade: 6h hardcoded

The cross-product is exercised by negative-path tests (CouncilPauseAsymmetry.t.sol, EmergencyTierAccessControl.t.sol) which prove that every NON-capability above reverts.

§2.4 Upgrade architecture

UUPS proxy (ERC1967Proxy with _authorizeUpgrade overridden in the vault). Three upgrade paths:

1. **Operational tier (48h, no cosig)** — applies to non-upgrade MANAGER selectors (strategy lifecycle, parameter setters). Does NOT include upgradeToAndCall.
2. **Meta tier (72h + COSIG_APPROVER signature)** — applies to upgradeToAndCall, fee setters, grantRole, revokeRole, updateSelectorTier (self-governance of the timelock's selector→tier map). The cosig signature is snapshotted at schedule() time (bake-in #18) so per-id requirements cannot be retroactively lowered.
3. **Emergency tier (6h hardcoded, NEW v5.1, #27)** — single dedicated path scheduleEmergencyUpgrade(newImpl, data, salt) → wait 6h → executeEmergencyUpgrade(...). Gated by:
   - msg.sender == councilEmergencySafe (the dedicated 3-of-3 Safe address, immutable post-init).
   - IVaultPaused(vault).paused() == true at BOTH schedule time AND execute time (dual check, prevents emergency abuse while vault is operating normally).
   - block.timestamp >= readyAt where readyAt = scheduleTimestamp + EMERGENCY_DELAY (constant 21,600 seconds = 6h, hardcoded immutable, no setter).

- `delete $.emergencyReadyAt[id]` happens BEFORE the cross-contract call to `vault.emergencyUpgrade`
  (defense-in-depth against re-entry).

Non-enumerated selectors revert on `propose` — default-deny. There is no catch-all delay and no admin escape hatch outside these three paths.

§2.5 Bake-in catalog #1–#27 + §S7.C

Consolidated from SEC-005 §F.1 (status as of v5.1 production candidate, commit-audited `825c4268…`):

#	Description	LOC	Sub-step	Status	Commit
1	`COSIG_APPROVER_ROLE` constant on vault + timelock V2	~10	S2.D.1	Implemented	`5e2b484`
2	`InitParams.cosigApprover` field + grant in `initialize`	~8	S2.D.2	Implemented	`5e2b484`
3	`retryBridge` 60s per-requestId cooldown + storage mapping	~10	S2.B.4	Implemented	`63fdb13`
4	`emergencyWithdrawAll` conditional-pause precondition	~1	S2.C.2	Implemented	`405a968`
5	`claimRedeem` emergency-activated short-circuit check	~1	S2.C.3	Implemented	`405a968`
6	`setPerformanceFee` accrue old rate pre-change	~2	S2.C.5	Implemented	`405a968`
7	`MAX_PERFORMANCE_FEE_BPS` cap raised 2000 → 3000 (30%)	~1	S2.C.6	Implemented	`405a968`
8	`setFeeRecipient` accrue old rate pre-change	~2	S2.C.7	Implemented	`405a968`
9	`revokeStrategy` residual-shares pre-check + dedicated error	~3	S2.C.8	Implemented	`405a968`
10	`stalenessThreshold` dead-state removal	~15	S2.A.5	Implemented	`5b03aeb`

#	Description	LOC	Sub-step	Status	Commit
11	Rename `getInternalBalance` → `getEvmIdleBalance`	~3	S2.A.4	Implemented	`2758706`
12	Move fee accrual from `_deposit` hook to public `deposit/mint`	~6	S2.C.1	Implemented	`405a968`
13	`_bridgeInFlight` tracking + timestamp in `_rawTotalAssets` (post-deposit settle window)	~8	S2.B.2	Implemented	`b106cbe`
14	`_bridgeFromInFlight` tracking + timestamp (post-redeem arrival window)	~6	S2.B.3	Implemented	`a452c84`
15	`emergencyRotateAgentWallet(address)` GUARDIAN-gated	~12	S2.C.9	Implemented	`405a968`
16	Replace `_internalBalance` with `IERC20.balanceOf(vault)` in `_rawTotalAssets`	~5	S2.B.1	Implemented	`29fd28c`
17	Emergency-state guard on `_accruePerformanceFee` + `_accrueManagementFee`	~2	S2.C.4	Implemented	`405a968`
18	Cosig flag snapshot at `schedule()` time (per-id `idRequiresCoSig`)	~10	S2.D.3	Implemented	`5e2b484`
19	Remove public `accruePerformanceFee()` / `accrueManagementFee()` externals	−5	S2.A.3	**Deferred-permanent**	Yearn V3 pattern — accepted; NatSpec annotated
20	Remove `requestRedeem(shares)` 1-arg wrapper	−3	S2.A.2	Implemented	`588d8a7`

#	Description	LOC	Sub-step	Status	Commit
21	Eliminate `accruedFees` field from `VaultStorage` struct	−5	S2.A.1	Implemented	`1102d2d`
22	`EMERGENCY_COUNCIL` grant via dedicated `initialize-Coun-cil(address[])` admin-only fn	±10	S2.E.1	Implemented	`39f0538`
23	`BridgeRequested` event + `getPending-BridgeAc-counting()` view	~12	S2.E.2	Implemented	`39f0538`
24 (v5.1)	Off-chain canary precompiles HyperCore (Python + systemd cron 1/min on EVM EC2)	~250 LOC Python	NOT audit-scope	Deployed shadow mode 2026-05-07	—
25 (v5.1)	Comms pipeline RSS + Dis-cord→Slack ABI watchdog	~150 LOC Python	NOT audit-scope	**Deferred F3** (W29+)	—
26 (v5.1)	NAV variance circuit breaker block-over-block	~+60–80 on-chain	NOT in v5.1	**Deferred Phase 2+** (trigger: TVL > $1M or first Phase 1 incident)	—
27 (v5.1)	Emergency upgrade tier (vault `emergencyUp-gradeToAnd-Call` + timelock V2 3 emergency fns + Safe-holds-role 3/3 + `EMER-GENCY_DELAY = 21,600` immutable + paused dual check)	+90 NSLOC / +215 raw	S2.B	Implemented	`a3336e5`

#	Description	LOC	Sub-step	Status	Commit
§S7.C (v5.1)	Decoder mitigation (`_decodeAction7` + `_decodeAction13` extracted `internal pure`; length guards 64→96/128; `SafeCast.toUint256` import; revert-behavior NatSpec)	+14 NSLOC / +69 raw	S2.C	Implemented	`bbc7145`

Test surface attached to v5.1 additions: 13 Foundry invariants + 5 halmos symbolic + 5 fork-mainnet (`V5_1EmergencyUpgradeForkE2E`) for #27; 12 Foundry edge cases + 5 halmos symbolic (`DecoderHarness` mirror + Foundry equivalence) for §S7.C. Full matrix in §6.

§2.6 Trust assumptions

#	Assumption	Mitigation if broken
1	OpenZeppelin Contracts v5.2.0 (commit `acd4ff74de833399287ed6b31b4deb0b2b35527`) + OZ Contracts-Upgradeable v5.2.0 (commit `3d5fa5c24c411112bab47bec25cfa9ad0b60e68`) integrity	Pinned by commit hash via git submodule + `forge-lock`; [illegible] requires MANAGER + COSIG_APPROVER signature + 72h Meta-tier wait. Submodule established in `contracts/foundry.toml` + `.gitmodules` post-merge `3ce24f1`.
2	Solidity 0.8.28 compiler integrity	Pinned in `foundry.toml`; documented solc bug list reviewed pre-upgrade. `via_ir` + `optimizer_runs=100` are part of the bytecode baseline.
3	HyperEVM consensus finality + Cancun EVM compatibility (PUSH0, transient storage available)	Out of vault control; risk accepted at chain level. Foundry profile pins `evm_version = "cancun"`.

#	Assumption	Mitigation if broken
4	HyperCore precompiles `0x0801` / `0x080F` return truthful data conforming to documented ABI (Action 7 SpotBalance, Action 13 AccountMarginSummary)	Staticcall capped at 50,000 gas; on failure, vault returns conservative fallback (0), NEVER a stale cached value. Bake-in #24 canary cross-checks precompile outputs vs HyperCore HTTP API 1/min off-chain — drift alerts trigger pause via GUARDIAN. §S7.C tightens decoder length guards 64→96/128 + extracts decoders as `internal pure` for isolated symbolic verification.
5	CoreWriter is the official Hyperliquid-operated bridge contract	Address verified once at `initialize`; stored in immutable-feel namespaced storage; no setter.
6	Gnosis Safe v1.4.1 contracts (operational 2/3 + emergency 3/3) are non-re-entrant and signature-correct	Canonical `SafeProxyFactory` at `0x4e1DCf7AD4e460CfD30791CCC4F9c8a4f820` integration tested fork-mainnet against real Safe deployments.
7	`hyper-evm-lib @ fac7ad1b` maintainer integrity	Library vendored in `lib/`; upgrades reviewed manually against upstream diff before bump.
8	Forge + halmos toolchain (forge 1.5.1-stable `b0a9dd9c`, halmos 0.3.3) produce sound symbolic verdicts under documented timeouts	Pinned versions; all proofs reproducible from public commit hashes.

---

§3. Threat Model (adversarial)

This section describes the adversarial threat model used to drive testing and review decisions. It is organized in three subsections:

- **§3.1 Function-by-function attack analysis** — for each public/external function on the vault and timelock V2, enumerates plausible attack vectors across four dimensions (economic, assumption-break, governance, reentrancy/cross-function) and identifies the defense surface that mitigates each.
- **§3.2 Cross-protocol exploit pattern matching** — cross-walk of 31 historical DeFi hacks (OPS-014 hack registry) against v5.1 surface; per-hack applicability + mitigation + residual risk.
- **§3.3 Per-surface attack trees** — three trees covering the most adversarial-significant surfaces: the dual-tier upgrade path (META 72h vs Emergency 6h), the bridge HC↔EVM async path, and the decoder mitigation surface (§S7.C).

Methodology note. The threat model describes the **contract surface and its defenses**, not the off-chain strategy that the agent wallet executes. Strategy parameters (entry/exit logic, sizing, capital distribution) are NOT in the audit scope (separate repository, separate security model). Adversaries against the off-chain strategy are out of scope; adversaries against the vault contracts and the bridge interface are in scope.

§3.1 Function-by-function attack analysis

For each entry point on the vault and timelock V2, the table below enumerates four attack dimensions and the defense surface. Rows that would require revealing strategy-specific parameters to describe the attack are omitted (gap is acceptable, alpha leak is not — see methodology note above).

§3.1.A Vault — deposit / mint / redemption surface

Function	Economic attack	Assumption-break	Governance attack	Reentrancy / cross-function
`deposit(uint256, address)`	Sandwich / front-run via maxDeposit bound; mitigated by per-deposit cap (`maxDepositAmount`) + share-price stability under fee accrual (#12 accrue-pre-super). Decimals offset 6 prevents share-price inflation attack.	Token of unexpected decimals: NOT applicable (asset is `IERC20(USDC)` 6-decimal, immutable post-init).	MANAGER role compromise → setMaxDeposit to attacker-friendly value; mitigated by OPERATIONAL_DELAY 48h timelock + emergency pause path.	`nonReentrant` modifier on `_deposit`; `_accrueFees` pre-super (#12) prevents accrue race during reentrance.
`mint(uint256, address)`	Same as deposit (symmetric path).	Same.	Same.	Same.
`requestRedeem(uint256, address, address)`	Inflated `req.assetAmount` via manipulated `totalAssets()` at request time; mitigated by bridge in-flight tracking (#13/#14) keeping totalAssets stable across deposit+immediate-redeem sequence.	HC precompile shape change rompe `_getHyperCoreValue` → fallback to length-guard early return 0; downstream `req.assetAmount` underestimates but does not corrupt vault state.	Same as `deposit`.	`_transfer(owner, vault, shares)` happens before `_bridgeFromHyperCore` external call → state mutation pre-call (canonical CEI).
`claimRedeem(uint256)`	Replay / double-claim; mitigated by `status: Claimable → Claimed` transition + `_burn` after balance check (`realBalance >= req.assetAmount`).	EVM balance under-arrives post-HC bridge; `claimRedeem` reverts `BridgePending`; user retries or `retryBridge` (#3 cooldown).	`claimEmergency` path locked out during normal ops; conversely, `claimRedeem` reverts `VaultInEmergency` post-emergency (#5).	`nonReentrant`; `safeTransfer` last in CEI.

Function	Economic attack	Assumption-break	Governance attack	Reentrancy / cross-function
`claimEmergency()`	Pro-rata distribution drift under partial HC liquidity; accepted-by-design — emergency state distributes available USDC only, residual stays for retry.	`emergencyActivated == false` blocks → `VaultNotInEmergency` revert.	Cannot reach emergency state without GUARDIAN `emergencyWithdrawAll` first (timelock + access control).	`nonReentrant`.
`retryBridge(uint256)` (#3)	DoS spam under high-fee adversary; mitigated by 60s per-requestId cooldown (`lastRetryTimestamp` mapping).	HC liquidity drift between schedule and retry; bound by `min(deficit, hyperCoreAvailable)`.	Permissionless invocation; not a governance surface.	`nonReentrant`; reads HC equity then writes timestamp before external bridge call.

§3.1.B Vault — fee accrual + setter surface

Function	Economic attack	Assumption-break	Governance attack	Reentrancy / cross-function
`setManagementFee(uint256)`	Adversarial governance setting cap-1 right before mass-redeem; mitigated by accrue-on-set (#6/#8) → fee minted to OLD recipient at OLD rate before switch.	Cap is hardcoded `MAX_MANAGEMENT_FEE_BPS = 200` (2%) immutable; no setter for cap.	MANAGER + timelock 48h delay; users have window to redeem.	`_accrueManagementFee` called first; setter is non-reentrant gated by MANAGER access.
`setPerformanceFee(uint256)`	Same pattern (#6/#7 cap raised to 3000 bps = 30% immutable).	Same.	Same.	Same.

Function	Economic attack	Assumption-break	Governance attack	Reentrancy / cross-function
`setFeeRecipient(address)`	Self-mint vault donation pattern (recipient = vault); accepted-by-design — phantom shares dilute holders proportionally without creating exploit, documented in D-COMBO-18.	None.	Same.	`_accrueFees` pre-switch (#8).
`accruePerformanceFee()` / `accrueManagementFee()`	Race with redeem to capture fees before pro-rata distribution; mitigated by accrue-on-redeem (#12) ensuring redeem flow captures latest fees.	Emergency state → no-op (#17 zero-delta guard).	Public + permissionless; not governance-gated.	Idempotent under reentrance (HWM monotonic).

§3.1.C Vault — admin + emergency surface

Function	Economic attack	Assumption-break	Governance attack	Reentrancy / cross-function
`pause()`	DoS by triggering pause to block legitimate flows; mitigated by access control (GUARDIAN OR EMERGENCY_COUNCIL_ROLE only — multi-role OR per FIX-007).	None.	Council compromise; mitigated by drill semestral + Phase 1 advisor vetting (LEGAL-004).	Single state write, no external call.
`unpause()`	Adversary blocks unpause to keep vault stuck; mitigated by GUARDIAN-only authority (council asymmetry — council can pause, NOT unpause).	None.	GUARDIAN compromise (hot EOA Phase 0); mitigated by 3-tier timelock on every sensitive operation + Phase 1 hardware wallet.	Single state write.

Function	Economic attack	Assumption-break	Governance attack	Reentrancy / cross-function
`emergencyWithdrawAll()`	One-shot drain; mitigated by GUARDIAN-only + bake-in #4 conditional pause idempotency (no re-trigger).	None.	Same as `unpause`.	`nonReentrant`.
`emergencyRotateAgent(address)` (#15)	Atomic set-and-drain to attacker EOA; mitigated by GUARDIAN-only + `paused() OR emergencyActivated` precondition.	extcodesize check on new agent (must be EOA, not contract).	Same as `unpause`.	`nonReentrant`.
`emergencyRevokeStrategy(address)`	Always-available revokes legitimate strategy during emergency; accepted-by-design (emergency state assumes operator control).	None.	GUARDIAN-only + `emergencyActivated == true` precondition.	`nonReentrant`.
`emergencyUpgradeToAndCall(address, bytes)` (#27)	Adversary upgrades to malicious impl in 6h instead of 72h; **mitigated by 4 layers**: (a) `onlyRole(MANAGER_ROLE)` = timelock-only (Safe cannot call vault directly); (b) timelock requires `msg.sender == councilEmergencySafe` (Safe 3/3 unanimous internal enforcement); (c) `paused()` dual check at schedule AND at execute (defense-in-depth vs guardian unpause race); (d) 6h `EMERGENCY_DELAY` hardcoded immutable.	None.	Council coercion (Safe 3/3 compromised socially/legally); mitigated by advisor selection criteria + 6h cancel window + drill semestral. PRIMARY new attack surface for v5.1.	Cross-contract call from timelock; `nonReentrant` on relay; idempotent under reentrance (delete `emergencyReadyAt[id]` pre-call).

§3.1.D Timelock V2 — schedule / execute / cancel + cosig + emergency tier

Function	Economic attack	Assumption-break	Governance attack	Reentrancy / cross-function
`schedule(...)`	Adversary schedules low-priority op to spam timelock; mitigated by PROPOSER_ROLE gate + selector floor enforcement (`_enforceSelectorFloor`).	None.	PROPOSER compromise; mitigated by Phase 1 Keystone HW.	OZ TimelockController upstream invariant.
`scheduleBatch(...)`	Batch OR-semantics cosig escalation (#18) — adding a tier-3 op to a tier-2 batch silently elevates cosig requirement; intentional.	Snapshot taken post-`super.scheduleBatch` (#18) prevents tier-change race.	Same as `schedule`.	OZ upstream.
`execute(...)` / `executeBatch(...)`	Open execution (EXECUTOR_ROLE = address(0)); anyone executes post-delay; accepted by OZ design.	None.	None — execution is permissionless once cosig + delay PASS.	OZ upstream.
`cancel(bytes32)`	Adversary cancels legitimate scheduled ops; mitigated by CANCELLER_ROLE = PROPOSER (one role compromise = same blast radius as schedule itself).	None.	Same as `schedule`.	OZ upstream.
`approve(bytes32)` (#1 cosig)	Adversary front-runs approval to race execution; mitigated by `idRequiresCoSig` snapshot at schedule time (#18) — cosig flag immutable post-schedule.	None.	MANAGER OR GUARDIAN OR COSIG_APPROVER access. Path B Safe 2-of-3 holds COSIG_APPROVER.	Single state write per approval.

Function	Economic attack	Assumption-break	Governance attack	Reentrancy / cross-function
`scheduleEmergencyUpgrade(...)` (#27)	Adversary schedules malicious upgrade with vault paused-by-attacker; mitigated by `msg.sender == councilEmergencySafe` (Safe 3/3 internal) + `vault.paused() == true` bound.	`vault.paused()` reverts if vault returns malformed; conservative path.	Council coercion = primary; see §3.1.C emergencyUpgradeToAndCall row.	Idempotent under reentrance (same id → same readyAt write).
`executeEmergencyUpgrade(...)` (#27)	Adversary races execute with simultaneous unpause; mitigated by SECOND `paused()` check inside `vault.emergencyUpgradeToAndCall` (defense-in-depth).	`block.timestamp >= readyAt` enforced; no skew bypass within 6h window.	Same as schedule.	`delete $.emergencyReadyAt[id]` pre-cross-contract-call; revert-bubble preserves error semantics.
`cancelEmergencyUpgrade(...)` (#27)	Adversary cancels legitimate emergency upgrade in flight; mitigated by `msg.sender == councilEmergencySafe` (only Safe can cancel) — adversary needs same Safe 3/3 access.	Pre-execute, post-execute, mid-window all delete cleanly.	Same as schedule.	Single state delete.
`updateSelectorTier(...)` (#27)	Adversary self-targets to relax tier on a selector; mitigated by self-only enforcement (DEFAULT_ADMIN_ROLE = address(timelock)) + tier delay (META + cosig).	`delay < OPERATIONAL_DELAY` reverts.	MANAGER + GUARDIAN cosig required.	Self-call only; no external.

§3.2 Cross-protocol exploit pattern matching (OPS-014 crosswalk)

The table below cross-references 31 historical DeFi hacks from the project's external hack registry (OPS-014) against v5.1 surface. Applicability classification: `yes` (vector directly applies), `no` (structural defense or absence of surface), `partial` (some attack components apply, others mitigated).

> **Source:** OPS-014 hack registry, 31 hacks cataloged across 8 categories (oracle, governance, bridge, upgrade, reentrancy, access control, flashloan, supply chain). Full triage detail per hack lives in OPS-014; this section consolidates applicability verdict for the v5.1 audit-scope.

#	Hack	Category	Root cause summary	v5.1 applicability	Mitigation surface	Residual risk
1	Mango Markets (Oct 2022, $114M)	Oracle	Mid-price manipulation via cross-margin position on thin orderbook; redemption at inflated NAV.	**partial**	Oracle-free design (precompile-based NAV, not orderbook mid-price); §S7.C decoder isolation; launch cap TVL $500K. Bake-in #26 (deferred Phase 2+) would add NAV variance circuit breaker.	Phase 0-1: economic friction (arb bots HL re-arbitrage seconds, attack economically irrational at cap $500K); Phase 2+: triplet defense (#26 + TWAP + bound checks).
2	Compound III oracle stale (Aug 2022)	Oracle	Stale oracle read during liquidation race; mitigated upstream by feed update.	no	No external price oracle in v5.1; HC precompile is the source of truth.	n/a.
3	Euler Finance (Mar 2023, $197M)	Lending logic	`donateToReserves` + `liquidate` race in violation rate calc.	no	No donations affect share calc beyond `IERC20.balanceOf` (#16); virtual shares offset 6 prevents inflation.	n/a.

#	Hack	Category	Root cause summary	v5.1 applicability	Mitigation surface	Residual risk
4	bZx (Feb 2020, $954k)	Flashloan	Synthetic price oracle race during flash.	no	No flashloan facility exposed; deposits/redeems async (ERC-7540 settles next-block at earliest).	n/a.
5	Beanstalk (Apr 2022, $182M)	Governance	Flashloan-funded governance vote to execute emergency upgrade.	**partial**	Emergency upgrade tier (#27) requires Safe 3/3 unanimous + paused state + 6h delay — flashloan voter cannot impersonate Safe owners.	Council coercion residual (advisors over time, not single-block flashloan).
6	Compound governance bug (Oct 2021)	Governance	Wrong addition operator in distribution; admin keys could not pause.	no	Governance is OZ TimelockController with cancel; emergency pause via GUARDIAN OR Council.	n/a.
7	Curve reentrancy (Jul 2023, $73M)	Reentrancy	Vyper compiler bug + missing CEI in custom `remove_liquidity`.	no	Solidity 0.8.28 (no Vyper); all state-mutating externals `nonReentrant`; CEI canonical pattern.	n/a.

#	Hack	Category	Root cause summary	v5.1 applicability	Mitigation surface	Residual risk
8	Cream Finance reentrancy (Oct 2021, $130M)	Reentrancy	ERC-777 callback during borrow re-enters lending market.	no	Asset is USDC (no callback hooks); ERC-4626 + ERC-7540 do not invoke recipient callbacks on transfer.	n/a.
9	Reentrancy on `claim` style functions	Reentrancy	State change after token transfer in claim path.	no	`claimRedeem` follows CEI strictly: status → balance check → `_burn` → safeTransfer last. `nonReentrant`.	n/a.
10	Wormhole bridge (Feb 2022, $326M)	Bridge	Signature verification bypass on cross-chain message.	no	We do NOT use cross-chain messaging bridges (no LayerZero / Wormhole / Axelar); HC↔EVM via HyperCore-native precompile only.	Trust HC precompile correctness — bake-in #24 canary monitors drift; §S7.C decoder mitigation enforces length guards.
11	Multichain (Jul 2023, $126M)	Bridge / KMS	Centralized MPC key compromise.	no	n/a — no MPC; on-chain Safe + role separation.	n/a.
12	Nomad (Aug 2022, $190M)	Bridge	Init misconfiguration enabled fraudulent prove.	no	n/a — no merkle-prove bridge.	n/a.

#	Hack	Category	Root cause summary	v5.1 applicability	Mitigation surface	Residual risk
13	Ronin (Mar 2022, $625M)	KMS	5-of-9 multisig compromise via social engineering.	**partial**	3-of-3 emergency council = different threshold trade-off (smaller surface, unanimous requirement raises bar). Advisor selection criteria mitigate single-firm capture.	Council coercion residual (4 layers per §8.5).
14	Parity multisig wallet bug (Nov 2017)	Access control	Library `initialize` callable post-deploy by unauthorized addresses.	no	OZ `initializer` modifier enforces single-call; `_disableInitializers` in constructor of upgradeable contracts prevents re-init.	n/a.
15	Indexed Finance (Oct 2021, $16M)	AMM logic	Re-weighting in inactive pool exploited via swap timing.	no	No AMM logic in vault; asset is single-token (USDC).	n/a.

#	Hack	Category	Root cause summary	v5.1 applicability	Mitigation surface	Residual risk
16	Yearn V2 misconfig (Feb 2021, $11M)	Strategy	New strategy added with bad slippage tolerance.	**partial**	Strategy registry uses MANAGER+timelock+keeper (#1, #18); 48h+72h windows for review. Phase 0: no active strategies (drainResidual eliminated in v2.0 FIX-009).	Phase 1+ strategy lifecycle: per-cosig strategy whitelist via `proposeStrategy → activateStrategy` Meta-tier governance.
17	Cream-style approval reset	Approval handling	Unbounded approval drained on token migration.	no	Vault assets stay in vault custody; user approvals limited to deposit amount.	n/a.
18	Lido GOAT council bypass (theorical)	Governance	Multi-sig threshold drift via member rotation.	no	EMERGENCY_COUNCIL_ROLE held by single Safe address (not on-chain member list); Safe internal threshold immutable post-deploy.	n/a.
19	Yam Finance rebase bug (Aug 2020)	Tokenomics	Rebase math precision lost; governance vote blocked.	no	No rebasing; ERC-4626 share price is `totalAssets / totalSupply` straightforward.	n/a.

#	Hack	Category	Root cause summary	v5.1 applicability	Mitigation surface	Residual risk
20	OlympusDAO `wsOHM` rebase (Jan 2022)	Wrap/unwrap	Conversion math error in unwrap.	no	Vault shares are pure ERC-4626; no wrap/unwrap derivatives in this codebase.	n/a.
21	dForce reentrancy (Apr 2020, $25M)	Reentrancy	ERC-777 callback during lock.	no	Same as #8.	n/a.
22	Hundred Finance reentrancy (Apr 2023, $7M)	Reentrancy	xDAI bridge + Compound v2 fork CEI violation.	no	Same as #7 + nonReentrant; no Compound v2 inheritance.	n/a.
23	DeFi Saver upgradeable misconfig (theoretical)	Upgrade	UUPS storage gap clobbered during upgrade.	no	Storage layout drift hook + ERC-7201 namespacing + golden file. Append-only + gap decrement validated per upgrade.	n/a.
24	Vesper governance multisig compromise (Feb 2022)	KMS	Hot multisig + DNS hijack of admin UI.	**partial**	Phase 0: GUARDIAN hot EOA = same risk class; mitigated by 3-tier timelock minimum 48h on every action. Phase 1: Safe multisig + Keystone HW.	Documented as top-3 residual §8.6.
25	Saddle Finance (Apr 2022, $11M)	AMM	Math precision in withdrawal calc.	no	No AMM.	n/a.

#	Hack	Category	Root cause summary	v5.1 applicability	Mitigation surface	Residual risk
26	Audius governance (Jul 2022, $6M)	Governance	Storage slot collision via proxy upgrade.	no	ERC-7201 namespaced storage prevents slot collision by design.	n/a.
27	Inverse Finance oracle (Apr 2022, $15M)	Oracle	Single-asset TWAP price manipulation via large swap.	no	No TWAP oracle in v5.1 (deferred Phase 2+ alongside #26).	n/a.
28	Visor Finance (Dec 2021, $8M)	Reentrancy	ERC-721 receiver callback during deposit.	no	No NFT receiver; assets are ERC-20 only.	n/a.
29	Harvest Finance (Oct 2020, $34M)	Flashloan	Curve pool oracle manipulation via flashloan.	no	No flashloan-callable function in vault.	n/a.
30	Cover Protocol (Dec 2020, $4M)	Tokenomics	Infinite mint via missing access control.	no	`_mint` only callable from `deposit` / `mint` path; `_burn` only from `claimRedeem` / `claimEmergency`. Role-gated.	n/a.
31	Furucombo (Feb 2021, $14M)	Approval	Delegatecall to attacker-controlled contract via approval flow.	no	No `delegatecall` to user-controlled targets; UUPS upgrade delegatecalls only timelock-scheduled implementations.	n/a.

Summary: of 31 historical hacks reviewed: - **0 directly applicable** (yes) to v5.1. - **4 partially applicable** (partial): Mango oracle (mitigated by oracle-free design + cap), Beanstalk flashloan governance (mitigated by Safe 3/3 emergency tier), Ronin KMS (mitigated by 3/3 unanimous +

advisor criteria), Vesper KMS (Phase 0 GUARDIAN hot EOA = top-3 residual §8.6, mitigated Phase 1). - **27 not applicable** (no) due to architectural differences: oracle-free, no flashloans, no AMM, no rebasing, no cross-chain bridges, no callback receivers, namespaced storage, role-gated mint/burn.

The 4 partial matches are precisely the categories where v5.1 carries residual risk — they map 1:1 to the §8.6 top-3 residuals (GUARDIAN hot EOA, no MANAGER multisig, no external audit) plus the NEW v5.1 Council coercion vector (§8.5).

§3.3 Per-surface attack trees (upgrade tier / bridge HC↔EVM / decoder Action 7/13)

Three attack trees follow, each describing branches per vector and defense per branch. Trees focus on adversarial-significant surfaces; non-significant paths (e.g. routine deposit/redeem under non-adversarial conditions) are covered in §3.1 above.

Tree 1 — Upgrade tier (normal META 72h vs Emergency 6h #27)

```
ROOT: adversary upgrades vault to malicious implementation
│
├── Branch A: MANAGER compromise (timelock holds MANAGER on vault)
│   │
│   ├── A.1 schedule upgradeToAndCall(maliciousImpl) via normal path
│   │   ├── 72h Meta delay → public visibility → community + Council watch
│   │   ├── Cosig required (#1, #18 snapshot) — MANAGER + GUARDIAN (or COSIG_APPROVER + GUARDIAN)
│   │   ├── Defense: GUARDIAN refuses cosig OR Council pauses vault
│   │   └── Defense: Cancel via PROPOSER (CANCELLER_ROLE = PROPOSER)
│   │
│   └── A.2 attempt scheduleEmergencyUpgrade via timelock
│       └── BLOCKED: `msg.sender == councilEmergencySafe` enforced; MANAGER ≠ council Safe
│
├── Branch B: MANAGER + GUARDIAN + COSIG_APPROVER collusion (full normal path)
│   │
│   ├── B.1 schedule + cosig + execute via Meta 72h
│   │   ├── Defense: 72h public window → community detects + responds
│   │   ├── Defense: Council pause (asymmetric — pause-only, NOT unpause) blocks execute path
│   │   └── Defense: emergency tier requires DIFFERENT key set (Council Safe 3/3 ≠ MANAGER/GUARDIAN/COSI
path keys does NOT grant emergency tier
│   │
│   └── B.2 attempt instant upgrade via direct vault.upgradeToAndCall
│       └── BLOCKED: `onlyRole(MANAGER_ROLE)` = timelock-only; EOA-MANAGER does not exist (FIX-
003)
│
├── Branch C: EMERGENCY_COUNCIL_ROLE compromise (Safe 3/3 compromised socially/legally)
│   │
│   ├── C.1 Safe 3/3 schedules emergencyUpgradeToAndCall(maliciousImpl)
│   │   ├── BOUND 1: `vault.paused() == true` required at schedule → adversary must first pause publicly
│   │   ├── BOUND 2: `EMERGENCY_DELAY = 21_600` (6h hardcoded immutable, no setter)
│   │   ├── Defense (during 6h window): counter-coerced council member calls cancelEmergencyUpgrade
│   │   ├── Defense (during 6h window): community detects pause + scheduled emergency → social pressure
│   │   ├── Defense (at execute): SECOND `vault.paused() == true` check in `emergencyUpgradeToAndCall`
in-depth) — GUARDIAN can unpause to brick execute
│   │   └── Defense (residual): launch cap TVL $500K → adversary's economic gain bounded
│   │
│   └── C.2 race: Safe schedules, GUARDIAN unpauses mid-window
│       ├── Branch A: GUARDIAN unpause sticks ≥1 block at execute time → execute reverts `EmergencyUpgrad
│       ├── Branch B: Safe re-pauses immediately (EMERGENCY_COUNCIL_ROLE holds pause authority) → state
coerced advisor can call `cancelEmergencyUpgrade(id)` (1-of-3, idempotent mapping delete). The race is
coercion cost on EVERY advisor on EVERY cycle.
```

```
│      └── Defense (terminating): launch cap TVL $500K bounds the economic prize; counter-
coercion of any single advisor (out of 3) terminates the race within the current 6h window. Phase 1 seme
of-3 cancel path operationally rehearsed.
│
└── Branch D: GUARDIAN-only compromise (no MANAGER, no Council)
    │
    └── D.1 GUARDIAN can pause/unpause + emergencyWithdrawAll but NOT upgrade
     └── Defense: role separation — GUARDIAN does NOT carry upgrade authority. Emergency upgrade still
```

Net residual: Branch C remains the primary irreducible adversarial vector — Council coercion in a 6h window with vault pre-paused. Mitigated by 4 layers (advisor selection criteria, paused public trigger, 6h cancel window, semestral drill) per §8.5. Phase 1 target: drill operationalized with real council post LEGAL-004 vetting.

Tree 2 — Bridge HC↔EVM async path (#13/#14 in-flight + #3 cooldown)

```
ROOT: adversary exploits bridge timing or accounting drift
│
├── Branch A: HC settles AFTER EVM redeem succeeds (impossible by design)
│      └── BLOCKED: `claimRedeem` requires `realBalance >= req.assetAmount` (EVM-
side check); HC never holds USDC delivered to user.
│
├── Branch B: HC settles BEFORE EVM redeem (normal path)
│   │
│   ├── B.1 user calls `requestRedeem` → `_bridgeFromHyperCore(min(assets, hcAvail))` issued
│   ├── B.2 30s window: `_bridgeFromInFlight` subtracts from `totalAssets` to prevent inflated NAV
│   ├── B.3 HC settles asset to EVM CoreDepositWallet → vault.balanceOf increases
│   └── B.4 user calls `claimRedeem` post-balance-arrival → safeTransfer USDC
│
├── Branch C: HC fails mid-flight (transient)
│   │
│    ├── C.1 `_bridgeFromInFlight` expires after 30s; `totalAssets` returns to non-
deducted state
│   ├── C.2 user calls `retryBridge(requestId)` — permissionless, per-id 60s cooldown
│   ├── C.3 retry caps at `min(deficit, currentHyperCoreAvailable)` to handle drift
│   └── Defense: spam DoS prevented by cooldown; underflow prevented by `min` clamp
│
├── Branch D: HC release fails permanently (storage corruption upstream)
│   │
│   ├── D.1 status sticks at Claimable but realBalance < req.assetAmount indefinitely
│   ├── D.2 `claimRedeem` reverts BridgePending; user blocked
│      └── Defense: emergency procedure — Council pause + `claimEmergency` pro-
rata distribution from available USDC; residual non-distributed locked until upstream recovery
│
├── Branch E: precompile ABI drift (HyperLiquid upgrades shape)
│   │
│   ├── E.1 length change (e.g. 96 → 80 or 128 → 144 bytes)
│   │   ├── `_getSpotValue` / `_getPerpValue` length-guard early-return 0
│   │  ├── `totalAssets()` drops accordingly; deposit/redeem may revert under accounting checks
│   │  └── Defense: canary #24 detects length change <60s → operator pause + decoder upgrade via #27 eme
│   │
│   └── E.2 silent shape change (same length, different field semantics)
│       ├── Risk: decoder returns garbage but does not revert
│       ├── Defense: §S7.C SafeCast on positive branch catches int64 overflow
│     └── Defense: semestral drill against mainnet returndata snapshots + canary watches `decode_succe
│
└── Branch F: spam adversary against bridge
```

```
    │
    ├── F.1 thousands of dust deposits → `_bridgeToHyperCore` per-deposit
    ├── F.2 each issues a `CoreWriter` Action 9 → potentially gas-griefs the relayer
   └── Defense: maxDepositAmount cap + per-deposit gas cost (user pays); economically unattractive at
```

Tree 3 — Decoder Action 7/13 (§S7.C mitigation surface)

```
ROOT: adversary or upstream-failure corrupts NAV computation via decoder
│
├── Branch A: malformed returndata (length wrong)
│   │
│   ├── A.1 `_getSpotValue`: `result.length < 96` → return 0 (graceful)
│   ├── A.2 `_getPerpValue`: `result.length < 128` → return 0 (graceful)
│  ├── Defense: decoder not invoked on under-length; downstream accounting drops HC contribution to 0
│   └── Defense: canary #24 alerts on length deviation → operator intervention
│
├── Branch B: correct length, malformed contents (negative accountValue extreme)
│   │
│   ├── B.1 `_decodeAction13` int64 negative → clamp to 0 (`accountValue <= 0 return 0`)
│   ├── B.2 SafeCast on positive branch unreachable; defensive
│  └── Defense: margin-call semantics preserved (negative HC equity treated as zero, not as corruption
│
├── Branch C: ABI shape drift (HyperLiquid upgrade)
│   │
│   ├── C.1 Action 7 field reordering (e.g. swap total ↔ hold)
│   │   ├── Decoded value semantically wrong but type-consistent
│   │   ├── totalAssets drifts; deposit/redeem priced incorrectly until fixed
│  │  └── Defense: semestral drill against known-good mainnet snapshots catches semantic drift
│   │
│   └── C.2 Action 13 type change (e.g. int64 → int128)
│       ├── abi.decode reverts at runtime
│       ├── `_rawTotalAssets` reverts → vault bricked until fix
│       └── Defense: emergency upgrade tier #27 deploys new decoder within 6h
│
├── Branch D: adversarial input via precompile manipulation
│   │
│   ├── D.1 HyperLiquid team compromise — out of contract scope; trust assumption
│   ├── D.2 economic friction: HL orderbook arb bots re-arbitrage within seconds
│  └── Defense (Phase 0-1): launch cap $500K bounds blast radius; canary alerts on shape change
│   └── Defense (Phase 2+): #26 NAV variance circuit breaker (deferred) adds block-over-
block bound
│
└── Branch E: decoder bug introduced in v5.x upgrade
    │
    ├── E.1 Halmos symbolic verification (`V5_1DecoderSymbolic.t.sol`) covers ∀ input
   ├── E.2 Foundry edge-case tests (`DecoderEdgeCasesTest.t.sol`) cover length boundaries + int64 MIN/
      └── Defense: staging coverage ~95% of decoder bugs pre-deploy (vs catch-in-
production via #26 which is deferred)
```

Net residual on §S7.C surface: D.2 (HyperLiquid team compromise) is the only fully irreducible vector — it requires trust in upstream protocol team. Mitigated indirectly by cap $500K + canary alerts. All other branches have ≥1 defense layer per branch.

---

§4. Findings — Slither (24) + Aderyn (18) consolidated triage

§4.1 Summary

Tool	Version	Total	Critical	High	Medium	Low	Informational	Net P0/P1 after triage
Slither	0.11.5	24 (post-§S6)	0	0	1 (FP)	11 (all FP)	12 (FP)	**0**
Aderyn	0.6.8	18 instances across 4 High categories + 14 Low categories	0	4 categories / 18 instances (all FP)	0	14 categories (style/informational)	—	**0**

Verdict: 0 P0/P1 net. Every finding flagged by either tool is either a documented false positive with reasoned triage (inherited from the v5-final baseline plus the new v5.1 surface) or an accepted-by-design pattern with mitigations described in §5 Non-findings / defense-in-depth. No remediation was required to clear the §S4.4 / §S4.5 acceptance gates of the IMPL-055 self-audit. The full raw triage files are committed to the public repo at `contracts/audit/slither-v5.1-triage.md` and `contracts/audit/aderyn-v5.1-triage.md`; this section paraphrases them with cross-protocol context.

Delta vs v5-final baseline (slither):

Bucket	v5-final	v5.1 (post-§S6)	Δ
High	0	0	0
Medium	11	1	**−10**
Low	9	11	+2
Informational	9	12	+3
Total	**29**	**24**	**−5**

The 10-Medium drop is attributable to the cleanup commit `f47389a` (Appendix A) which added explicit `uint256 i = 0` loop initialization and `slither-disable-next-line` annotations on inherited divide-before-multiply / strict-equality patterns. The +3 Informational adds come exclusively from the v5.1 emergency upgrade tier (#27): one new `assembly` (revert bubble), one `low-level-calls` (timelock-to-vault relay), and one `unindexed-event-address` (timelock V2 mirror event).

§4.2 Slither findings (24)

§4.2.1 Medium (1) — false positive `incorrect-equality` — `AspeLabsVault.maxMint` line 506

```
function maxMint(address) public view override returns (uint256) {
    uint256 maxAssets = maxDeposit(address(0));
    if (maxAssets == 0) return 0;
    return convertToShares(maxAssets);
}
```

Slither flags any `== 0` comparison defensively. Here the strict equality is the canonical ERC-4626 short-circuit when the vault is paused, `emergencyActivated`, or has reached its `maxDepositAmount` cap. The bounded comparison is semantically correct and matches the OZ ERC4626Upgradeable reference implementation. **FP — inherited from v5-final.**

§4.2.2 Low (11) — all false positive `reentrancy-events` ×4

- `AspeLabsVault.sol:597` — `emit BridgeRequested` after CoreWriter call (bake-in #23). The external call is to the trusted CoreWriter precompile; the event emit follows OZ's emit-after-action pattern.
- `AspeLabsVault.sol:1455` — symmetric pattern on the HyperCore → EVM bridge path.
- `AspeTimelockControllerV2.sol:379` (NEW v5.1) — `executeEmergencyUpgrade` emits `EmergencyUpgradeExecuted` after the cross-contract `.call` to the vault's `emergencyUpgradeToAndCall`. The mapping write `delete $.emergencyReadyAt[id]` executes BEFORE the call, providing the actual reentrancy defense; Slither's heuristic flags the event regardless. The path is additionally single-entry (`msg.sender == councilEmergencySafe` recheck would be required on re-entry, and the mapping has already been cleared).

All four are documented FP — defense is intact via the pre-call state update + trusted-target invariant.

`timestamp` ×6 — `block.timestamp` reads in bridge-in-flight tracking (#13/#14), retry cooldown (#3), and the `executeEmergencyUpgrade` ready-at check (#27, line 379). Slither flags any `block.timestamp` use; the timestamps drive coarse windows (30s settle, 60s retry, 6h emergency) where miner manipulation (~12s tolerance under post-Merge consensus) is operationally negligible. **FP — same pattern as v5-final.**

`low-level-calls` ×1 — accepted as defensive pattern (see Informational below; some Slither versions classify this under Low).

§4.2.3 Informational (12) — expected patterns `assembly` ×5

- `AspeLabsVault.sol:176` — `_getVaultStorage()` ERC-7201 namespaced storage pointer (canonical pattern from EIP-7201).
- `AspeLabsVault.sol:350` — `_validateInitParams` `extcodesize` check on `councilEmergencySafe` argument. *Note: this specific assembly site existed in the v5.1 ORIGINAL candidate (pre-§S6 size-reduction); it was REMOVED during the §S6 surgery to bring runtime under EIP-170 24,576 bytes. In the audit-scope baseline (825c4268…, post-§S6) this finding no longer exists. Listed here only for traceability — Slither's 24-finding count is the post-§S6 result.*
- `AspeLabsVault.sol:917` — pre-existing storage pointer pattern.
- `AspeTimelockControllerV2.sol:118` — `_overlay()` ERC-7201 storage pointer.
- `AspeTimelockControllerV2.sol:379` (NEW v5.1) — `executeEmergencyUpgrade` revert bubble (`revert(add(ret, 0x20), mload(ret))`) propagating the vault's revert reason. Standard OZ-style call-bubble pattern; no symbolic state mutated in assembly.

`low-level-calls` ×3

- `AspeLabsVault.sol:1480` (decoder isolation surface, §S7.C) — `_getSpotValue` staticcall to precompile `0x0801`. Identical to pre-refactor; the §S7.C refactor isolated the staticcall + decoder into its own internal pure function so Slither now reports it at the wrapper line rather than at the original inline site.
- `AspeLabsVault.sol:1492` (decoder isolation surface, §S7.C) — `_getPerpValue` staticcall to precompile `0x080F`. Same.
- `AspeTimelockControllerV2.sol:379` (NEW v5.1) — `executeEmergencyUpgrade` relays a `.call` to `vault.emergencyUpgradeToAndCall(...)`. The target address is the immutable `$.vault` set at `initialize`; the calldata is constructed locally from the scheduled `(newImpl, data, salt)` triple. **No untrusted target** — this is the canonical OZ TimelockController execute pattern, hardened with the `delete $.emergencyReadyAt[id]` pre-call write.

`naming-convention` ×2 — `AspeTimelockControllerV2.sol:140, 142` — `initialize(address _vault, address _councilEmergencySafe)` uses leading-underscore parameter names to distinguish init arguments from storage fields. Pre-existing OZ pattern adopted here.

`unindexed-event-address` ×2 — `AspeLabsVault.sol:240, 255` — `FeeRecipientUpdated(old, new)` and `HedgeRegistryUpdated(old, new)`. Pre-existing. Not indexed because off-chain frontend in-

dexers handle non-indexed address parameter scanning, and adding `indexed` modifiers now would require an upgrade ceremony + event-handler migration with minimal practical benefit.

§4.3 Aderyn findings (4 High categories, 18 instances)

§4.3.1 H-1 Contract locks Ether without a withdraw function (1 instance) `AspeLabsVault.sol:27` (contract declaration).

The vault is an ERC-4626 over USDC asset; **the vault holds 0 ETH at all times by design.** There is no `receive()` function and no `payable` fallback. The only function on the vault accepting `msg.value` is `emergencyUpgradeToAndCall` (#27), which forwards via OZ's `upgradeToAndCall(newImpl, data)`; UUPS does not retain ETH, and the OZ `_upgradeToAndCallUUPS` path reverts on value mismatch when the new implementation does not opt in. Aderyn flags any contract without an explicit `withdraw()` ETH exit; here ETH ingress is impossible by design. **FP — inherited from v5-final, unchanged by #27 in practice.**

§4.3.2 H-2 ETH transferred without address checks (3 instances)

- `AspeTimelockController.sol:154` — v1 timelock, DEPRECATED. Not deployed in v5.1.
- `AspeTimelockControllerV2.sol:294` — `super.execute` payable forward (inherited from OZ `TimelockControllerUpgradeable`).
- `AspeTimelockControllerV2.sol:305` — `super.executeBatch` payable forward (idem).

The timelock V2 inherits OZ `TimelockControllerUpgradeable`, which forwards `msg.value` to the target of a previously scheduled operation on `execute`/`executeBatch`. The target was vetted at `schedule()` time, which required `PROPOSER_ROLE` and cleared the per-selector delay floor + cosig requirement (snapshotted per-id, bake-in #18). Adding a redundant zero-address check at execute would diverge from the audited OZ upstream without adding security (the scheduling path is the security boundary). **FP — inherited OZ pattern.**

Aderyn does NOT flag the new `executeEmergencyUpgrade` (line 379) under H-2: that path constructs its own calldata and targets the immutable `$.vault` address, with no symbolic ETH-forwarding surface reachable from external callers.

§4.3.3 H-3 Reentrancy: state change after external call (13 instances) Vault instances (7) — all gated by OZ `ReentrancyGuardUpgradeable.nonReentrant`:

- `AspeLabsVault.sol:765` — `claimRedeem` (pre-existing).
- `AspeLabsVault.sol:837` — `retryBridge` (bake-in #3).
- `AspeLabsVault.sol:1130, 1131` — `emergencyRotateAgentWallet` (bake-in #15).
- `AspeLabsVault.sol:1172, 1192, 1197` — `revokeStrategy` paths (bake-in #9).

Timelock V2 instances (4):

- `AspeTimelockControllerV2.sol:217, 218, 222` — `approve()` reads `$.vault.hasRole` (cross-contract view, no mutation) then writes `$.managerApproved` / `$.guardianApproved`. The cross-contract call is a view to OZ AccessControl with no reentrancy surface.
- `AspeTimelockControllerV2.sol:350` (NEW v5.1) — `scheduleEmergencyUpgrade` calls `IVaultPaused(vault).paused()` (view) BEFORE writing `$.emergencyReadyAt[id]`. Defense-in-depth on this new instance:
  1. `msg.sender` is the dedicated 3-of-3 Gnosis Safe; Safe v1.4.1 `execTransaction` is non-reentrant in the upstream-audited implementation.
  2. If hypothetically reached re-entry, the schedule path re-asserts `msg.sender == councilEmergencySafe`.
  3. The state write `$.emergencyReadyAt[id] = block.timestamp + EMERGENCY_DELAY` is idempotent for the same `(newImpl, data, salt)` — re-entry would overwrite with the same `readyAt` (no economic damage).

V1 timelock (2, deprecated) — pre-existing, not deployed.

All 13 instances are gated by `nonReentrant` (vault paths) or by the OZ TimelockController upstream model plus the multi-layer authentication described above (V2 paths). **FP — inherited pattern + 1 new instance documented as defense-in-depth.** The defense-in-depth note for an external auditor (Spearbit/Cantina equivalent): an explicit `nonReentrant` modifier on `scheduleEmergencyUpgrade` is technically redundant under the threat model but would cost ~5K gas/schedule as belt-and-suspenders. We chose not to apply it; an auditor may recommend otherwise.

§4.3.4 H-4 Unsafe casting of integers (1 instance) `AspeLabsVault.sol` — `uint64(block.timestamp)` in the redemption record (bake-in #13, pre-existing). `block.timestamp` overflows `uint64` only at approximately year 584,554,051 AD; the cast is bounded by physical time. **FP — inherited.**

§4.3.5 Low (14 categories) — style / informational All 14 Aderyn Low categories are style or informational with no remediation: centralization warnings (Phase 0 expected, mitigated by timelock + multisig at Phase 1), bounded loops, empty UUPS `_authorizeUpgrade` body (canonical), large numeric literals with underscore separators (`21_600`, `172_800`, `259_200`), single-use modifiers (kept for clarity), PUSH0 opcode (Cancun-enabled), state changes without events on internal `_authorize*` overrides, unchecked returns historically replaced by `SafeERC20.safeTransfer`, pragma `^0.8.28` (widely accepted), unused error from earlier iteration, public functions exposed for off-chain readers, and deprecated slots retained for ABI compatibility. None of these warrant a finding in a tier-1 report.

§4.4 Findings tied to new v5.1 surface (#27 + §S7.C)

This subsection consolidates every finding (across both tools) attributable to the v5.1 bake-ins, for explicit auditor focus on the new attack surface:

#	Tool	Detector	Location	Triage
F-NEW-1	Slither	`low-level-calls`	`AspeLabsVault._getProtocolPrice` (§S7.C)	Pre-validate staticcall to `0x0801` — same pattern as pre-refactor; defense intact via 50k gas cap + length guard 64→96 + conservative-fallback-0 on failure. **FP.**
F-NEW-2	Slither	`low-level-calls`	`AspeLabsVault._getPreviousPrice` (§S7.C)	Pre-validate staticcall to `0x080F` — same pattern; length guard 64→128 + SafeCast on positive branch. **FP.**

#	Tool	Detector	Location	Triage
F-NEW-3	Slither	`assembly`	`AspeTimelockControllerV2.executeEmergencyUpg` (#27)	Reverse-bubble `(revert(add(ret, 0x20), mload(ret)))` — standard OZ call-bubble pattern; no symbolic state mutated. **FP.**
F-NEW-4	Slither	`low-level-calls`	`AspeTimelockControllerV2.executeEmergencyUpg` (#27)	Low-level call to immutable `$.vault.emergencyUpgradeTo` calldata constructed locally from scheduled `(newImpl, data, salt)`; mapping `$.emergencyReadyAt[id]` deleted PRE-call. Trusted target, no reentry surface. **FP.**
F-NEW-5	Slither	`reentrancy-events`	`AspeTimelockControllerV2.executeEmergencyUpg` (#27)	Event emitted after call; reentrancy actually defended by the pre-call mapping delete + single-entry `msg.sender == councilEmergencySafe` check. **FP.**
F-NEW-6	Slither	`timestamp`	`AspeTimelockControllerV2.executeEmergencyUpg` (#27)	`block.timestamp < readyAt` check on a 6h window; miner manipulation negligible. **FP.**
F-NEW-7	Aderyn	H-3 reentrancy	`AspeTimelockControllerV2.scheduleEmergencyUp` (#27)	Pause-check view BEFORE mapping write; Safe v1.4.1 non-reentrant + re-asserted `msg.sender` on hypothetical re-entry + idempotent write. **FP — defense-in-depth.**

The §S7.C decoder isolation (`_decodeAction7` / `_decodeAction13` extracted as `internal pure`) intro-

duces **zero new Aderyn findings** — `internal pure` is the cleanest surface possible. Symbolic equivalence between the isolated decoders and a `DecoderHarness` mirror is verified by 5 halmos symbolic tests + 12 Foundry edge-case tests.

Net P0/P1 attributable to v5.1 surface: 0. Every new finding above is FP with documented mitigation.

---

§5. Non-findings / defense-in-depth analysis

This section documents properties an external auditor would explicitly note as **examined and found no issue with**. Each subsection states the property, the mechanism enforcing it, and the test evidence supporting the verdict. Phrasing throughout is assertive but bounded — "we examined X and found no issue because Y" rather than "X is impossible to attack."

§5.1 Reentrancy protection model

Every state-mutating external on the vault is guarded by OZ `ReentrancyGuardUpgardeable.nonReentrant`: `deposit`, `mint`, `withdraw`, `redeem`, `requestRedeem`, `claimRedeem`, `retryBridge` (#3), `emergencyWithdrawAll`, `emergencyRevokeStrategy`, `emergencyRotateAgentWallet` (#15), `revokeStrategy` (#9). The bridge-related externals additionally apply the `_bridgeInFlight` / `_bridgeFromInFlight` accounting (#13/#14) which closes the asynchronous-settlement reentry window between EVM-side state changes and HyperCore arrival.

On the emergency upgrade path (#27), the timelock V2's `executeEmergencyUpgrade` does **not** rely on `nonReentrant`. Instead, defense-in-depth is layered:

1. `delete $.emergencyReadyAt[id]` executes BEFORE the external `.call` to `vault.emergencyUpgradeToAndCa` — any re-entry attempt finds the mapping cleared and reverts with `EmergencyUpgradeNotReady`.
2. The single-entry authorization `msg.sender == councilEmergencySafe` would have to hold again on re-entry, which presumes the same Safe re-signing the same operation — not a reentrancy attack but a normal repeat invocation, which the cleared mapping defeats.
3. Gnosis Safe v1.4.1 `execTransaction` is non-reentrant in the upstream-audited implementation; the emergency Safe (3-of-3 dedicated) is the only address that can ever appear as `msg.sender` on these calls.

Test coverage: `ReentrancyAttackerMock` is deployed in the v5final test suite and exercised against deposit/redeem round-trip (`ReentrancyDepositRedeem.t.sol`); a dedicated `EmergencyUpgradeReentrancy.t.sol` exercises the cleared-mapping pre-call write on the new #27 path. **We examined the reentrancy surface and confirm no issue under the threat model.**

§5.2 Role separation + access control

The role model in §2.3 deliberately splits authority so that no single role can both pause + upgrade + drain. The asymmetries are:

Action	DEFAULT_ADMIN	MANAGER	GUARDIAN	COSIG_APPROVER	EMERGENCY_COUNCIL
`pause`	✗	✓	✓ instant	✗	✓ instant
`unpause`	✗	✓	✓ instant	✗	✓ (deliberate asymmetry)
`emergencyWithdrawAll`	✗	✓	✓ instant	✗	✗
`upgradeToAndCall` (Operational/Meta)	✗	✓ via timelock 48h/72h+cosig	✗	✓ (cosigns Meta)	✗
`emergencyUpgradeToAndCall` (Emergency tier)	✗	✗	✗	✗	✓ via timelock 6h + paused dual check

Action	DEFAULT_ADMIN	MANAGER	GUARDIAN	COSIG_APPROVER	EMERGENCY_COUNCIL
`setPerformanceFee`, `setManagementFee`, `setFeeRecipient`	✗	✓ via Meta 72h+cosig	✗	✓ (cosigns Meta)	✗
`grantRole`, `revokeRole`	✓ (Meta-gated for transitions)	✗	✗	✗	✗
Drain residual / `sweep(token)`	✗	✗	✗	✗	✗

The **EMERGENCY_COUNCIL pause-only-on-vault asymmetry** is intentional: the Council holds the most-trusted emergency-upgrade key (3-of-3 dedicated Safe), but does NOT receive `unpause` authority on the vault — so once paused, only GUARDIAN can unpause, ensuring the Council cannot single-handedly weaponize a pause for griefing. The Council CAN cancel its own scheduled emergency upgrade via `cancelEmergencyUpgrade(id)`.

Negative-path tests: `CouncilPauseAsymmetry.t.sol` (Council attempts unpause → revert), `EmergencyTierAccessControl.t.sol` (every non-Council caller on `scheduleEmergencyUpgrade` → revert; Council attempting Operational/Meta selectors → revert; Council attempting `emergencyWithdrawAll` → revert). **We examined role separation and confirm no privilege overlap with single-role drain capability.**

The `drainResidual` function eliminated in v5.0 (FIX-009) remains absent in v5.1 — `vault.drainResidual(X)` reverts with `selector unknown`. No `sweep(token)` was added.

§5.3 Immutability of critical constants

The following constants are hardcoded immutables with NO setter function (verified by grep for `function set` against each constant identifier; absence confirmed):

Constant	Value	Meaning
`EMERGENCY_DELAY`	`21_600` (6 hours)	Minimum wait between `scheduleEmergencyUpgrade` and `executeEmergencyUpgrade` (#27). Hardcoded — no `setEmergencyDelay` exists.
`META_DELAY`	`259_200` (72 hours)	Floor for Meta-tier scheduled operations. Editable per-selector only via `updateSelectorTier` (which itself requires Meta + cosig).
`OPERATIONAL_DELAY`	`172_800` (48 hours)	Floor for Operational-tier scheduled operations. Editable per-selector only via `updateSelectorTier` (Meta + cosig).
`MAX_PERFORMANCE_FEE_BPS`	`3000` (30%)	Hard cap on `setPerformanceFee` argument. Hardcoded — exceeding it reverts. Bake-in #7 raised this from 2000 → 3000.

Constant	Value	Meaning
`MAX_MANAGEMENT_FEE_BPS`	`200` (2%)	Hard cap on `setManagementFee` argument. Hardcoded.
Council Safe size	3-of-3	The dedicated emergency Safe address is stored as `councilEmergencySafe` at `initialize` time; the Safe's threshold is checked by the Safe contract itself (Gnosis upstream invariant).

Test enforcement: halmos symbolic verification proves `setPerformanceFee(X)` reverts for any `X > 3000`, and that `scheduleEmergencyUpgrade` writes `readyAt = block.timestamp + 21_600` exactly (no off-by-one). Foundry edge cases assert the exact constant values at compile time via `assertEq(vault.EMERGENCY_DELAY(), 21_600)`. **We examined the immutability of critical bounds and confirm no parametric backdoor.**

§5.4 Oracle-free design

The vault does NOT consult any external price oracle (no Chainlink, no Pyth, no UniV3 TWAP, no internal AMM-derived price). NAV is computed via:

```
_rawTotalAssets() = balanceOf(asset)                    // EVM-side USDC idle
                  + HyperCore spot equity (precompile 0x0801)
                  + HyperCore perp equity (precompile 0x080F)
                  + Σ strategy module reported assets
                  − hedge registry obligations
                  − _bridgeFromInFlight (post-redeem window)
                  + _bridgeInFlight (post-deposit window)
```

This eliminates the entire class of oracle-manipulation exploits that have affected other DeFi protocols (cross-walked in §3.2 against OPS-014 registry). The trade-off is that NAV correctness depends on:

1. **HyperCore precompile correctness** — mitigated by bake-in #24 off-chain canary that cross-checks precompile outputs against HyperCore's HTTP API 1/min (alerts route to GUARDIAN with auto-pause runbook).
2. **Decoder correctness** — mitigated by §S7.C decoder isolation (`_decodeAction7` / `_decodeAction13` extracted as `internal pure`, length guards tightened 64→96/128 to match exact ABI sizes, `SafeCast.toUint256` on the positive branch, and explicit revert behavior documented in NatSpec).

On precompile failure (out-of-range data, gas exhaustion, revert), the vault uses a conservative fallback (returns 0 for that component, never a stale cached value). The combination yields a NAV that may temporarily under-report during HyperCore outages but cannot over-report from a corrupt oracle feed. **We examined the oracle attack surface and confirm it does not exist on the vault.**

§5.5 Bound checks

Beyond the constant immutables in §5.3, the vault enforces several runtime bounds:

- **`maxDepositAmount` per-call deposit cap** — set via `setMaxDeposit` (Operational tier, 48h timelock). Acts as on-chain TVL governor. Phase 0 launch policy: `maxDepositAmount` set such that aggregate TVL caps at **$500K** (decision STRAT-005 §2; constraint relaxed only after L2 audit completes). Setter exists but is timelock-gated.

- **Emergency upgrade gating** — `scheduleEmergencyUpgrade` requires `paused() == true` at schedule time; `executeEmergencyUpgrade` re-checks `paused() == true` at execute time (dual check). Both checks gate against bake-in #4's `emergencyWithdrawAll` precondition (pause before withdrawal), ensuring an emergency upgrade cannot be weaponized against an operating vault.
- **Strategy module count** — bounded by `MAX_STRATEGIES` to keep `_rawTotalAssets` iteration gas-bounded.
- **Fee accrual ordering** — bake-ins #6 and #8 require accruing the OLD fee rate before any rate change, preventing retroactive fee extraction on accumulated unrealized PnL.

Blast-radius bound: even if every soft-control above were bypassed, the **$500K cap TVL ceiling** combined with the dedicated emergency Safe 3-of-3 (separate from operational Safe 2/3) limits maximum single-incident loss until the Phase 1 L2 audit threshold is reached. **We examined runtime bounds and confirm no unbounded value or iteration path.**

§5.6 Bridge in-flight tracking (#13 / #14)

The asynchronous EVM↔HyperCore bridge creates two narrow time windows where a naive `_rawTotalAssets` would either double-count or under-count assets:

- **Post-deposit window (`_bridgeInFlight`, #13)** — between the moment USDC leaves the EVM-side vault and the moment HyperCore reflects the deposit in its spot balance (typically ~30 seconds). During this window, the vault tracks the in-flight amount + timestamp; `_rawTotalAssets` adds the in-flight assets back to prevent under-counting (which would let depositors at the start of the window receive too many shares).
- **Post-redeem window (`_bridgeFromInFlight`, #14)** — between the bot's `bridgeUsdcToEvm` call on HyperCore and the USDC arriving at the EVM vault (~30 seconds). During this window, the vault tracks the in-flight amount + timestamp; `_rawTotalAssets` subtracts the in-flight assets to prevent double-counting (which would let depositors during the window claim shares against assets earmarked for redeemers).

Both fields auto-expire after a conservative timeout. Test coverage: `BridgeInFlightInvariant.t.sol` defines 6 invariants over deposit/redeem/bridge sequences, exercised under fuzz + invariant runs in the 448-test suite. **We examined the bridge accounting surface and confirm no economic damage path in either direction.**

§5.7 Retry cooldown (#3)

`retryBridge(requestId)` is permissionless (any caller can retry a stuck bridge) and gated by a 60-second per-`requestId` cooldown stored in `lastRetryTimestamp[requestId]`. This balances two concerns:

- **Anti-DoS** — without the cooldown, an attacker could spam `retryBridge` calls to push CoreWriter into an inconsistent state or exhaust gas budgets on the relayer side.
- **Operational liveness** — legitimate retries of genuinely stuck bridges (HyperCore congestion, mempool drops) must remain accessible without requiring a privileged role.

The 60s window is short enough to support legitimate operational retries during incidents and long enough to cap spam at one retry per minute per requestId. The retry itself does not move new funds — it re-emits the bridge instruction for an already-credited request. **We examined the retry surface and confirm no spam-DoS path.**

§5.8 Storage layout discipline

Both the vault and timelock V2 use **ERC-7201 namespaced storage**, eliminating storage collisions during upgrades. The discipline is enforced by:

1. **Golden files** — `contracts/storage-layouts/AspeLabsVault-v5.layout`, `AspeLabsVault-v5.1-EXPECTED.layout`, `AspeTimelockControllerV2-v5.1-EXPECTED.layout`. Hand-curated maps committed to the repo.

2. **Pre-commit hook** — `scripts/check-storage-layout.sh` regenerates the layout via `forge inspect` and diffs against the golden file. Any drift blocks the commit.
3. **Storage gap discipline** — both contracts reserve `uint256[N] __gap` arrays. Adding a new field requires decrementing the gap; reordering or removing fields is structurally prohibited. The §S6 size-reduction surgery preserved gap sizes (vault unchanged, timelock V2 unchanged in OverlayStorage gap).
4. **Upgrade compat checks** — pre-deployment, OZ Foundry Upgrades plugin verifies storage-layout-compatible transitions; any incompat blocks the candidate.

For #27, the emergency upgrade tier added storage to timelock V2 OverlayStorage (`emergencyReadyAt[id]` mapping, `councilEmergencySafe` immutable-feel field) by decrementing the gap 49→47. The vault received NO new storage from #27 — the Safe-holds-role pattern places authority entirely in the timelock V2 plus the AccessControl role itself. **We examined storage discipline and confirm no upgrade-compat risk.**

§5.9 Decoder mitigation (§S7.C)

The HyperCore precompile decoders (`_decodeAction7` for spot balance, `_decodeAction13` for perp margin summary) were the largest single-source-of-trust surface in the v5.0 → v5-final transition. §S7.C (IMPL-055 §S2.C) hardened them with four defenses:

1. **Function isolation** — extracted from inline `_getSpotValue` / `_getPerpValue` into dedicated `internal pure` functions. The `pure` attribute prevents any state read or modification inside the decoder, narrowing the symbolic surface to pure ABI parsing.
2. **Length guards tightened** — pre-§S7.C, the length checks accepted `>= 64` bytes (the minimum field count of either action). Post-§S7.C, the guards require the EXACT documented sizes: Action 7 = 96 bytes, Action 13 = 128 bytes. Any oversize or undersize input reverts with a typed error rather than silently parsing wrong bytes.
3. **SafeCast on positive branch** — Action 13's `accountValue` field is a signed `int256` that must be cast to `uint256` for the perp equity sum. `SafeCast.toUint256` reverts on negative values rather than wrapping around to a huge positive (which would inflate NAV). The decoder explicitly handles the signed/unsigned conversion at the documented branch point.
4. **Revert-behavior NatSpec** — every revert path is documented in NatSpec so an external integrator (auditor or future strategy module developer) knows exactly which inputs map to which revert reasons.

Symbolic equivalence between the isolated decoders and a `DecoderHarness` mirror is verified by 5 halmos tests + 12 Foundry edge cases (oversize / undersize / negative-value / boundary-value / zero-value / max-value). **We examined the decoder surface and confirm no precompile-data-driven NAV corruption path.**

§5.10 Negative-path test coverage

Beyond the positive properties documented in §5.1-§5.9, the test suite explicitly exercises the **revert + RBAC-block paths**. Categories:

Negative-path category	Coverage test files	What is asserted
RBAC blocks	`EmergencyTierAccessControl.t.sol`, `PauseAuthority.t.sol`, `TimelockCancelAuthority.t.sol`	Every non-authorized caller against every authorized function reverts with the correct typed error.
Pause-state blocks	`CouncilPauseAsymmetry.t.sol`, `PauseAuthority.t.sol`	`unpause` reverts for Council; `emergencyWithdrawAll` reverts under non-paused; `scheduleEmergencyUpgrade` reverts under non-paused.

Negative-path category	Coverage test files	What is asserted
Timelock-bypass blocks	`TimelockTieredDelays.t.sol`, `TimelockExecutePaths.t.sol`	`execute` reverts before `delay` elapses; `cancel` reverts for non-CANCELLER; selectors not in floor revert at `schedule`.
Cosig-bypass blocks	`CosigSnapshotInvariant.t.sol`, `CosigPathBFlow.t.sol`, `V5FinalCosig.t.sol` (halmos)	`execute` reverts when `idRequiresCoSig[id] == true` and approvers haven't signed.
Bridge-state blocks	`BridgeInFlightInvariant.t.sol`, `RetryBridgeCooldown.t.sol`	`claimRedeem` reverts `BridgePending` under under-arrived USDC; `retryBridge` reverts under 60s cooldown.
Decoder length-guard blocks	`DecoderEdgeCasesTest.t.sol`, `V5_1DecoderSymbolic.t.sol` (halmos)	Action 7 / Action 13 length-mismatch reverts with typed error; `SafeCast` reverts on signed→unsigned negative.
Re-init blocks	`CouncilInitialization.t.sol`	`initializeCouncil` reverts on second call (`Initializable` upstream invariant).

We examined the negative-path coverage and confirm every authorization boundary has at least one revert test asserting the boundary holds. The `forge test` cold-cache run includes 50+ explicit revert-expecting assertions across these files.

---

§6. Test coverage matrix

§6.1 Test suite summary

The v5.1 audit-scope candidate (`commit-audited-v5.1 CURRENT`, post-§S6 size-reduction) is exercised by four complementary tooling layers. Each layer was re-run cold (`forge clean` then full re-build) at the audit-scope commit; results below are the verbatim acceptance gate numbers from IMPL-055 §S4.6 and §S6.

Suite	Tests	Result	Coverage type
Foundry full suite	448	**448 PASS / 0 FAIL / 0 SKIPPED**	unit + integration + invariant + fuzz
Halmos `--match-contract V5_1`	10	**10/10 PASS**	symbolic — 5 EmergencyUpgrade + 5 DecoderSymbolic
Halmos full discovery (no filter)	27	**27/27 PASS / 0 FAIL**	symbolic — all classes (11 AspeLabsVaultInvariant + 1 AspeLabsVaultSymbolic + 5 V5Final + 10 V5_1)
Fork-mainnet drill `V5_1EmergencyUpgradeForkE2E`	5	**5/5 PASS** (10.5s wall-time)	integration — real Safe v1.4.1 3-of-3 on HyperEVM mainnet RPC

The fork-mainnet drill is the load-bearing end-to-end check for the bake-in #27 emergency upgrade tier: it pins a real Safe v1.4.1 3-of-3 deployment on HyperEVM, fork-executes the full `schedule → 6h`

`delay → execute` flow against the audit-scope bytecode, and asserts both happy-path success and three negative paths (non-Council caller revert, paused-precondition violation, expired-readiness revert).

§6.2 Function-by-function coverage matrix

The table below maps every public/external function of the vault and timelock V2 to the foundry / halmos / fork test files that exercise it. Where a function is a pure view delegating to another covered function, the row reads "view delegate, covered transitively." Where multiple test types cover the same function, all are listed (the canonical pattern post-#27 is foundry property invariant + halmos symbolic + fork integration for tier-1 critical paths).

Vault — `contracts/src/AspeLabsVault.sol`:

Function	Test file(s)	Test types covering it
`deposit(uint256, address)`	`AspeLabsVaultDepositTest.t.sol` + `BridgeInFlightInvariant.t.sol` + `AspeLabsVaultInvariant.t.sol`	unit + fuzz + invariant
`mint(uint256, address)`	`AspeLabsVaultDepositTest.t.sol` (shares-path subset)	unit + fuzz
`requestRedeem(uint256, address, address)`	`RequestRedeemFlow.t.sol` + `AspeLabsVaultInvariant.t.sol`	unit + invariant + fuzz
`claimRedeem(uint256)`	`RequestRedeemFlow.t.sol` (`test_claim*` family) + `BridgeInFlightInvariant.t.sol`	unit + invariant
`claimEmergency(uint256)`	`ClaimEmergencyFlow.t.sol` (rescue-path under `emergencyActivated == true`)	unit
`totalAssets()`	`BridgeInFlightInvariant.t.sol` + `PrecompileFallbackTest.t.sol` + `AspeLabsVaultInvariant.t.sol`	invariant + property (precompile-OOR fallback)
`retryBridge(uint256)`	`RetryBridgeCooldown.t.sol` (60s cooldown + DoS scenarios)	unit + fuzz
`pause()` / `unpause()`	`PauseAuthority.t.sol` + `CouncilPauseAsymmetry.t.sol`	unit (RBAC) + property (Council-cannot-unpause)
`emergencyWithdrawAll()`	`EmergencyWithdrawAll.t.sol` + `EmergencyTierAccessControl.t.sol`	unit + RBAC negative-paths
`emergencyUpgradeToAndCall(...)` (**#27**)	`EmergencyUpgradeInvariant.t.sol` (vault-side path) + `V5_1EmergencyUpgrade.t.sol` (halmos symbolic) + `V5_1EmergencyUpgradeForkE2E.t.sol` (fork drill)	invariant + halmos + fork integration
`emergencyRevokeStrategy(address)`	`StrategyEmergencyRevoke.t.sol`	unit
`emergencyRotateAgentWallet(address)` (**#15**)	`AgentWalletRotation.t.sol` (incl. broken-rotation negative paths)	unit + RBAC

Function	Test file(s)	Test types covering it
`setManagementFee(uint256)` (#6)	`FeeAccrualOrdering.t.sol` (rate-change-after-accrue invariant)	unit + invariant
`setPerformanceFee(uint256)` (#8)	`FeeAccrualOrdering.t.sol` + `V5_1DecoderSymbolic.t.sol` (cap symbolic)	unit + invariant + halmos
`setFeeRecipient(address)` (**#8**)	`FeeAccrualOrdering.t.sol` (`test_setFeeRecipient*`)	unit
`setMaxDeposit(uint256)`	`MaxDepositGovernor.t.sol`	unit
`setHedgeRegistry(address)`	`HedgeRegistryHook.t.sol`	unit
`proposeStrategy(...)` / `revokeStrategy(...)` / `deprecateStrategy(...)`	`StrategyLifecycle.t.sol`	unit + invariant
`accruePerformanceFee()` / `accrueManagementFee()`	`FeeAccrualOrdering.t.sol` + `AspeLabsVaultInvariant.t.sol` (HWM monotonic property)	unit + invariant + fuzz
`initializeCouncil(address[])` (**#22**)	`CouncilInitialization.t.sol` (one-shot replay guard, role-grant correctness)	unit
`getEvmIdleBalance()` (**#11**)	`BridgeInFlightInvariant.t.sol` (subset)	view delegate, covered transitively
`getPendingBridgeAccounting()` (**#23**)	`BridgeInFlightInvariant.t.sol` (`test_pending*`)	unit + invariant
`maxDeposit(address)`	`MaxDepositGovernor.t.sol`	unit
`maxMint(address)`	`MaxMintReturnsZero.t.sol` (asserts `== 0` per FIX-008)	unit
`maxWithdraw(address)`	property: returns `0` (async-only) — covered in `RequestRedeemFlow.t.sol`	property
`maxRedeem(address)`	property: returns `0` (async-only) — covered in `RequestRedeemFlow.t.sol`	property
`previewWithdraw(uint256)`	property: reverts (async-only) — covered in `RequestRedeemFlow.t.sol`	property (revert)
`previewRedeem(uint256)`	property: reverts (async-only) — covered in `RequestRedeemFlow.t.sol`	property (revert)
`pendingRedeemRequest(uint256, address)`	`RequestRedeemFlow.t.sol`	view delegate, covered transitively
`claimableRedeemRequest(uint256, address)`	`RequestRedeemFlow.t.sol`	view delegate, covered transitively

Timelock V2 — `contracts/src/AspeTimelockControllerV2.sol`:

Function	Test file(s)	Test types covering it
`schedule(...)` / `scheduleBatch(...)`	`CosigSnapshotInvariant.t.sol` + `V5FinalCosig.t.sol` (halmos) + `TimelockTieredDelays.t.sol`	unit + invariant + halmos
`execute(...)` / `executeBatch(...)`	`TimelockExecutePaths.t.sol` + `CosigSnapshotInvariant.t.sol`	unit + invariant

Function	Test file(s)	Test types covering it
`cancel(bytes32)`	`TimelockCancelAuthority.t.sol`	unit + RBAC
`approve(bytes32)` (**#1** cosig)	`V5FinalCosig.t.sol` (halmos) + `CosigPathBFlow.t.sol`	unit + halmos
`updateSelectorTier(...)`	`CosigSnapshotInvariant.t.sol::test_immediate_noSnapshotForTier2Select` + `V5FinalSnapshotCosig.t.sol` (halmos)	unit + invariant + halmos
`updateDelay(...)`	`TimelockTieredDelays.t.sol`	unit + RBAC
`scheduleEmergencyUpgrade(...)` (**#27**)	`V5_1EmergencyUpgrade.t.sol` (halmos symbolic, 5 properties) + `EmergencyUpgradeInvariant.t.sol` (foundry invariant) + `V5_1EmergencyUpgradeForkE2E.t.sol` (fork drill)	halmos + invariant + fork integration
`executeEmergencyUpgrade(...)` (**#27**)	same as above (`V5_1EmergencyUpgrade.t.sol` + invariant + fork)	halmos + invariant + fork
`cancelEmergencyUpgrade(bytes32)` (**#27**)	`V5_1EmergencyUpgrade.t.sol` + `EmergencyUpgradeInvariant.t.sol` (`test_cancel_*` family)	halmos + invariant
`getEmergencyReadyAt(bytes32)`	view delegate of `emergencyReadyAt` mapping — covered transitively	view delegate, covered transitively
`councilEmergencySafe()` (getter)	view delegate — covered by `V5_1EmergencyUpgradeForkE2E.t.sol`	view delegate, covered transitively
`requiresCoSignature(bytes4)`	`CosigSnapshotInvariant.t.sol`	view delegate, covered transitively
`idRequiresCoSig(bytes32)` (**#18**)	`CosigSnapshotInvariant.t.sol::test_immediate_noSnapshotForTier2Select` (mid-flow tier-flip property)	invariant + halmos
`isManagerApproved(bytes32)` / `isGuardianApproved(bytes32)`	`CosigPathBFlow.t.sol`	view delegate, covered transitively
`getSelectorMinDelay(bytes4)`	view delegate — covered by `TimelockTieredDelays.t.sol`	view delegate, covered transitively

Across both contracts: 34 distinct entry points / 23 vault + 14 timelock V2 covered by named tests. We examined the function-by-function coverage and confirm no public mutating entry point is exercised by less than one of foundry unit + invariant or halmos symbolic.

§6.3 Coverage gaps + intentional skips

In the interest of full disclosure, the following gaps are documented:

1. **ERC-7540 conformance tests** — pre-IMPL-055 §S3.B, 2 conformance tests covering the `afterBridge` and `afterClaim` lifecycle hooks were marked `SKIP` while the post-#27 storage extension was being stabilized. These were **UN-SKIPPED in IMPL-055 §S3.B-extra** (commit `4fe4cfb`) and now PASS in the 448-test cold re-run. No conformance tests remain skipped at the audit-scope commit.
2. **Halmos `V5FinalSnapshotCosig::_disabled_check_schedule_noSnapshotForTier2`** — disabled in v5.1 due to a known halmos limitation: path-tracking through the ERC1967Proxy `delegatecall` does not converge within the solver-timeout budget after the #27 storage extension (overlay storage gap decremented 49→47 plus the new `initialize` signature). Coverage is **preserved by the foundry equivalent** `CosigSnapshotInvariant.t.sol::test_schedule_noSnapshotForTier2Selector`, which exercises the same mid-flow

tier-flip property under fuzz + invariant runs and is PASS in the 448-test suite. This is a tooling limitation, not a contract bug — see v5.1 self-audit report §Defense-in-depth note 5.

3. **Production agent wallet on canary #24** — the off-chain HyperCore precompile-drift canary samples its target via the `sample_address` config; at the audit-scope commit, the production agent wallet has not yet been deployed (Phase 0 sandbox state), so the canary currently samples the **sandbox-fast vault** `0x6ef5eFdA60b4D1A08d9119a6A5D26C6af9792b57` (a zero-HyperCore-activity address). Pre-launch action item: update `sample_address` to the production agent wallet once deployed. Documented in IMPL-055 §5 lessons-learned table.

These are the only known coverage gaps at the audit-scope commit. No vault or timelock V2 public mutating function is uncovered; the gaps above concern conformance tooling, symbolic solver convergence, and an off-chain monitoring config — none impact the on-chain correctness claims of §4 / §5.

§6.4 Reproducibility

Anyone can reproduce the §6.1 numbers against the public audit-scope commit. Pinned tool versions: forge `1.5.1-stable`, halmos `0.3.0`, solc `0.8.28`. From the repo root:

```
cd /home/gorka/projects/aspe_labs/contracts
forge clean && forge test                                                    # 448 PASS /
halmos --match-contract V5_1 --solver-timeout-assertion 180000               # 10/10 PASS
halmos --solver-timeout-assertion 180000                                     # 27/27 PASS
forge test --match-path 'test/integration/V5_1EmergencyUpgradeForkE2E.t.sol' \
  --fork-url https://rpc.hyperliquid.xyz/evm                                 # 5/5 PASS (1
```

```
# Verify specific surface (faster than full suite):
forge test --match-contract EmergencyUpgradeInvariant            # bake-in #27 vault-side invarian
forge test --match-contract V5_1DecoderSymbolic                   # §S7.C decoder isolation proper
forge test --match-contract CouncilPauseAsymmetry                 # role-separation negative-paths
forge test --match-contract BridgeInFlightInvariant               # bake-ins #13/#14 in-flight acc
forge test --match-test test_schedule_noSnapshotForTier2Selector  # bake-in #18 mid-flow tier-flip
halmos --match-test check_emergencyUpgrade_paused_dual_check      # halmos #27 paused-dual-check s

# Storage layout drift check (single-shot, silent on success):
cd /home/gorka/projects/aspe_labs && scripts/check-storage-layout.sh  # exit 0 = pass
```

Halmos full-discovery wall-time on a modern laptop is approximately 6–9 minutes (CPU-bound on the symbolic solver). The fork-mainnet drill requires only public RPC access (no auth, no API key).

---

§7. Storage layout + upgrade compatibility analysis

§7.1 ERC-7201 namespaced storage approach

Both the vault and the timelock V2 use the **ERC-7201 namespaced storage pattern** (custom storage slot derivation via `keccak256` of a namespace string, masked to clear the lowest byte). This eliminates the entire class of storage-collision bugs that arise from naive proxy + multiple-inheritance layouts — each contract owns a deterministically-derived slot range that cannot collide with any other namespaced range or with the proxy's ERC1967 implementation slot.

The two namespaces in v5.1 are:

Namespace string	Consumer	Storage slot constant
`erc7201:aspelabs.vault.storage`	`AspeLabsVault.VaultStorage` struct	`VAULT_STORAGE_LOCATION = 0xee4e81f6...ba900`

Namespace string	Consumer	Storage slot constant
`erc7201:aspelabs.timelock.overlay`	`AspeTimelockControllerV2.OverlayStorage` struct	`OVERLAY_STORAGE_LOCATION` (analogous derivation)

The slot constants are derived as `keccak256(abi.encode(uint256(keccak256(NAMESPACE)) - 1)) & ~bytes32(uint256(0xff))`, the canonical ERC-7201 formula. Both structs are accessed exclusively via internal `_getVaultStorage()` / `_getOverlayStorage()` helpers that resolve the slot in assembly and return a `storage` pointer — there is no fallback path to read or write either struct outside its namespace.

§7.2 Vault storage layout (v5.1)

The `VaultStorage` struct holds 23 named fields plus the storage gap. Fields are grouped by functional area (core accounting / async redemption / strategy management / HyperCore integration / emergency / forward-compat / governable fees / in-flight bridge accounting / retry cooldown):

Group	Fields
Core accounting	`_internalBalance`, `highWaterMark`, `feeRecipient`, `maxDepositAmount`
Async redemption (ERC-7540)	`_nextRequestId`, `redeemRequests` (mapping), `totalPendingShares`, `totalClaimableAssets`, `_userRequestIds` (mapping), `_pendingRequestCount`
Strategy management	`strategies` (mapping), `strategyList` (array), `totalStrategyDebt`
HyperCore integration	`agentWallet`
Emergency	`emergencyActivated`
Forward compatibility	`hedgeRegistry`, `upgradeabilityRenounced`
Governable fees (FIX-006)	`performanceFeeBps`, `managementFeeBps`, `lastMgmtFeeAccrualTimestamp`
In-flight bridge accounting (#13/#14)	`_bridgeInFlight`, `_bridgeInFlightTimestamp`, `_bridgeFromInFlight`, `_bridgeFromInFlightTimestamp`
Retry bridge cooldown (#3)	`lastRetryTimestamp` (mapping)
Gap	`uint256[42] __vaultGap`

Gap evolution v5.1. The original v5.1 commit (`02d32e3`, superseded) introduced a `councilEmergencySafe` mirror field consuming one slot and decrementing the gap to `[41]`. During the **§S6 size-reduction surgery** (commit `825c426...`), this mirror was removed: the vault never read the field at runtime, the timelock V2 OverlayStorage holds the authoritative `councilEmergencySafe` (the only consumer in the emergency upgrade auth path), and removing it brought the runtime bytecode from 25,025 bytes back under the EIP-170 24,576-byte limit. The gap was **restored from `[41]` → `[42]`**.

Net effect. The vault storage layout at the audit-scope commit is **effectively IDENTICAL to `v5-final-merged`**: zero net new slots from #27 + §S7.C combined. The §S6 gap restoration cancels the §S2.B addition. The pre-commit hook `scripts/check-storage-layout.sh` validates this against the committed golden file with a silent exit-0.

§7.3 Timelock V2 storage layout (v5.1)

The `OverlayStorage` struct holds 8 named fields plus the gap:

Slot	Field	Source	Notes
[0]	`IAccessControl vault`	v5-final	Reference to the vault for cross-contract RBAC reads
[1]	`mapping(bytes4 => uint256) selectorMinDelay`	v5-final	Per-selector delay tier registry
[2]	`mapping(bytes4 => bool) selectorRequiresCoSig`	v5-final	Per-selector cosig requirement
[3]	`mapping(bytes32 => bool) managerApproved`	v5-final	Path B cosig — manager-side approval
[4]	`mapping(bytes32 => bool) guardianApproved`	v5-final	Path B cosig — guardian-side approval
[5]	`mapping(bytes32 => bool) idRequiresCoSig`	**v5-final via #18** (IMPL-052 S2.D.3)	Snapshot of cosig requirement at schedule time; prevents mid-flow tier-flip attack
[6]	`address councilEmergencySafe`	**v5.1 via #27** (IMPL-055 S2.B.1)	Single source of truth for the emergency Council Safe address; authoritative
[7]	`mapping(bytes32 => uint256) emergencyReadyAt`	**v5.1 via #27** (IMPL-055 S2.B.1)	Per-id readiness timestamp for `executeEmergencyUpgrade` (6h after schedule); 0 = not scheduled
[8..54]	`uint256[47] __gap`	—	Reserved for future fields; gap decremented `49 → 47` for the two #27 additions

Gap evolution. - v5-pre: 5 fields + `__gap[51]` = 56 slots - v5-final: 6 fields (+`idRequiresCoSig` from #18) + `__gap[49]` = 55 slots - v5.1: 8 fields (+`councilEmergencySafe` and `emergencyReadyAt` from #27) + `__gap[47]` = 55 slots

Net slots in the namespace remained constant from v5-final → v5.1: the gap absorbs the additions, which is the entire point of the gap discipline.

§7.4 Upgrade compatibility analysis

Both contracts are **UUPS upgradeable** (ERC1967Proxy + `_authorizeUpgrade` gated). The v5-final → v5.1 deltas are:

- **Vault:** zero net change to the namespaced struct (mirror added then removed in §S6 — gap[42] preserved relative to v5-final-merged). Append-only / no reordering / no deletion. Storage-layout-compatible.
- **Timelock V2:** two new fields appended (`councilEmergencySafe + emergencyReadyAt`), gap decremented `49 → 47`. Append-only / no reordering / no deletion. Storage-layout-compatible.

Enforcement. Storage-layout drift is detected by:

1. **Pre-commit hook `scripts/check-storage-layout.sh`** — regenerates the vault layout via `forge inspect` post-`forge build` and diffs against the committed golden file `contracts/storage-layouts/AspeLabsVault-v5.layout`. Drift on the vault namespace blocks the commit before it reaches the remote.
2. **Hand-curated golden** — timelock V2 OverlayStorage is hand-maintained at `contracts/storage-layouts/AspeTimelockControllerV2-v5.1-EXPECTED.layout` (the namespace is not auto-introspectable via `forge inspect storageLayout` because ERC-7201 maps to a single derived slot, not the implicit slot-0 layout). Verification via `awk '/struct OverlayStorage \{/,/^\}/' src/AspeTimelockControllerV2.sol`.
3. **OpenZeppelin Foundry Upgrades plugin** — pre-deployment, the plugin verifies a candidate upgrade against its predecessor and rejects any storage-layout-incompatible transition. This is an additional belt-and-braces gate on top of the static checks above.

Manual verification command (anyone can reproduce):

```
cd /home/gorka/projects/aspe_labs
scripts/check-storage-layout.sh    # silent exit 0 = pass
```

§7.5 Bytecode baselines chain

The complete chain of intermediate baselines from v5-pre → v5-final → v5-final-merged → v5.1 ORIGINAL → v5.1 CURRENT is registered in `infra/AWS_REGISTRY.md` §11 with sha256 `deployedBytecode` for each artifact. The audit-scope commit corresponds to the row labelled **`commit-audited-v5.1 CURRENT`** in Appendix C of this report. The hashes match across IMPL-055 §S4.3, this report, and the registry — three-way cross-reference verified character-by-character 2026-05-11: vault `825c42686adba26bbf3eaf4ca39d2fe7d96304b5810ce03943921343714ceb4d` identical at YAML header (line 17), §1.1 (line 117), and Appendix C row CURRENT.

§7.6 Delta vs SEC-004 v2.0 (audited baseline)

For an external auditor handed `commit-audited-v5.1 CURRENT` as audit-scope, the storage delta vs the original SEC-004 v2.0 audited baseline (`e923006`, v5-pre) is:

- **Vault:** net 0 new slots over the cumulative v5-final + v5.1 history. Several slots added (#11 getter wiring, #13/#14 bridge accounting, #23 event additions) and several removed (`accruedFees` in #21, `stalenessThreshold` in #10, `councilEmergencySafe` mirror in §S6). Gap `__vaultGap[42]` is effectively unchanged from v5-final-merged. Append-only-with-compensating-removals — upgrade-compatible.
- **Timelock V2:** net +3 slots over v5-pre baseline (`idRequiresCoSig` from #18 in v5-final + `councilEmergencySafe` + `emergencyReadyAt` from #27 in v5.1). Gap decremented accordingly `50 → 49 → 47`. Append-only, upgrade-compatible.

We examined the storage layouts and upgrade compatibility and confirm the v5.1 candidate is a storage-layout-compatible successor to both v5-final-merged and to the original SEC-004 v2.0 baseline `e923006`.

---

§8. Operational risks

§8.1 Governance model

Authority over the vault is partitioned across four scopes, each with its own access control and timelock policy. The intent is that no single role can both (a) move funds at scale and (b) skip a public waiting window. Even the emergency tier (#27) requires a public, on-chain `schedule` event with a 6h cancel window before `execute` becomes callable.

Scope	Examples of gated operations	Role(s) authorized	Delay	Cosig requirement
Operational	`setMaxDeposit`, `setHedgeRegistry`, `proposeStrategy`, `deprecateStrategy`	MANAGER (held by timelock V2)	`OPERATIONAL_DELAY = 48h`	None
Meta	`upgradeToAndCall` (normal path), `grantRole` / `revokeRole`, `setPerformanceFee`, `setManagementFee`, `setFeeRecipient`, `updateSelectorTier`, `updateDelay`	MANAGER + (COSIG_APPROVER OR GUARDIAN)	`META_DELAY = 72h`	Required (Path B: PROPOSER schedule + COSIG_APPROVER approve + GUARDIAN approve + permissionless execute)
Emergency upgrade (#27)	`scheduleEmergencyUpgrade` → `executeEmergencyUpgrade`	EMERGENCY_COUNCIL_ROLE held by dedicated Safe 3-of-3 (separate from operational Safe)	`EMERGENCY_DELAY = 6h` (hardcoded)	Safe internal 3-of-3 threshold; `paused() == true` dual-check at schedule + execute
Council pause (#22)	`pause()`	GUARDIAN OR EMERGENCY_COUNCIL_ROLE	Instant	None

The asymmetric authority on `pause` / `unpause` is deliberate: **the Council can pause but cannot unpause**. Once paused (whether by GUARDIAN as part of an operational incident or by the Council as part of an emergency escalation), only GUARDIAN can resume operations. This prevents the Council from weaponizing a pause for griefing while preserving its ability to halt the vault in <30 minutes if the GUARDIAN key is compromised or unresponsive.

§8.2 Key management — Phase 0 (current state)

At the audit-scope commit, the vault is in Phase 0 (no external capital). The key inventory is:

Role	Holder at Phase 0	Notes
`DEFAULT_ADMIN_ROLE`	Timelock V2 self-admin (no external admin)	No bootstrap admin path post-init; all admin transitions require the timelock itself
`MANAGER_ROLE`	Timelock V2	Gates Meta operations; identical to DEFAULT_ADMIN holder
`GUARDIAN_ROLE`	Solo operator hot EOA	**Phase 0 constraint** — mitigated by 3-tier timelock + Council pause defense-in-depth
`COSIG_APPROVER_ROLE`	Sandbox Safe 2-of-3 `0x2A3c41A0AFF1b7bB74bF81040EaF30d1A3a820B3`	Production Phase 1 target: Safe 2-of-3 post-LEGAL-004 vetting

Role	Holder at Phase 0	Notes
`EMERGENCY_COUNCIL_ROLE`	Not yet granted in production (sandbox-only `Safe v1.4.1` `3-of-3` exercised in fork drill)	Activated at Phase 1 deploy via `initializeCouncil` one-shot grant with 3 advisor Safe

The combination "solo-operator GUARDIAN hot EOA" plus "no Council in production" is the **dominant residual risk vector** of Phase 0 and is structurally bounded by the launch cap TVL (see §8.6).

§8.3 Key management — Phase 1 target

Pre-external-capital intake, the following escalations are required by [STRAT-005] §2 launch gate and IMPL-040:

Role	Phase 1 target	Trigger
`GUARDIAN_ROLE`	Hardware wallet (Ledger or Keystone) — IMPL-040 T1.1	Pre-launch, single-signer hardware promotion
`COSIG_APPROVER_ROLE`	3-advisor Safe 2-of-3 — LEGAL-004 vetting in progress	Pre-launch, after advisor vetting completes
`EMERGENCY_COUNCIL_ROLE`	3-advisor Safe 3-of-3 (unanimous) — separate Safe deployment from COSIG_APPROVER	Pre-launch, activated via `initializeCouncil` one-shot grant

The Phase 1 inventory yields the documented "multisig 2-of-3 + dedicated emergency Safe 3-of-3" topology referenced in §5.2 of this report. The two Safes are deliberately **separate deployments with non-overlapping signer sets** to prevent a single advisor compromise from yielding both Meta cosig and emergency upgrade authority simultaneously.

§8.4 Recovery procedures (runbook inventory)

Operational recovery is governed by versioned runbooks under `docs/ops/runbooks/`. Each runbook documents the exact on-chain commands, expected event signatures, and acceptance criteria for one scenario. Active / drafted / TBD inventory at the audit-scope commit:

Runbook	Scenario	Status	`verified-at`	`verified-bytecode`
`emergency-pause.md`	GUARDIAN or Council pause	active	(pending Pass-1 drill)	—
`precompile-drift-response.md`	Canary #24 alert response	active	2026-05-07	shadow-mode
`bridge-retry-flow.md`	Stuck bridge recovery via `retryBridge`	active	(pending Pass-1 drill)	—
`requestredeem-claim-cycle.md`	Async redemption lifecycle	active	(pending Pass-1 drill)	—
`claim-emergency-procedure.md`	`claimEmergency` rescue path	active	(pending Pass-1 drill)	—

Runbook	Scenario	Status	verified-at	verified-bytecode
agent-wallet-rotation.md	Emergency rotate via #15	active	(pending Pass-1 drill)	—
safe-cosig-approve.md	Path B cosig signing flow	active	(pending Pass-1 drill)	—
upgrade-vault.md	Normal-path upgrade (META_DELAY 72h)	active	(pending Pass-2 drill IMPL-054)	—
emergency-upgrade-flow.md	NEW v5.1 6h emergency tier (#27)	**TBD**	**pending Pass-2 drill**	(target: 825c4268…)
(corpus, total target Phase 1 launch)	governance, key rotation, incident triage, etc.	32 runbooks (9 active + 22 maintained + 1 TBD)	—	—

Certification convention (per CLAUDE.md): `verified-at` bumps only when a live drill executes the runbook commands on-chain successfully against the listed `verified-bytecode`. `last-updated` (separate from `verified-at`) bumps on any edit. The IMPL-054 §F drill matrix maps each runbook to its certifying station; Pass-1 (sandbox-fast, comp delays) + Pass-2 (audit-equivalent timing) constitute the full certification cadence.

Total runbook corpus target: **32 runbooks** at Phase 1 launch readiness. The v5.1-specific addition (`emergency-upgrade-flow.md`) is gated on the Pass 2 drill completing — the drill captures the exact command sequence + tx hashes against the audit-scope bytecode, which then populates the runbook's `verified-at` / `verified-by` / `verified-bytecode` YAML fields per the IMPL-054 certification pattern.

§8.5 Council coercion vector (NEW v5.1)

The introduction of the emergency upgrade tier (#27) opens a new attack surface that did not exist in v5-final: coercion of the 3 Council advisors. This is explicitly catalogued as the primary new vector from the v5.1 delta — see FOUND-001 §4.2 Vectores operacionales for the full threat model (council coercion row; cross-ref verified 2026-05-11 — content present with mitigation matrix).

Severity: Critical.

Vector: all 3 advisors are coerced (social engineering, legal pressure, physical coercion in adversarial jurisdictions) into co-signing a malicious `scheduleEmergencyUpgrade` followed 6h later by `executeEmergencyUpgrade`. The malicious implementation could in principle drain assets via the upgraded logic.

Mitigations (four layers):

1. **Advisor selection criteria** — three advisors selected for *geographic and legal distinctness* (different jurisdictions, different legal frameworks for compulsion), *publicly verifiable reputation* (impossible to compromise quietly without reputational signal), and *no organizational redundancy* (no two advisors from the same firm or social circle). Criteria documented in FOUND-001 §4.2 + LEGAL-004 vetting checklist.
2. **`paused()` precondition at schedule** — `scheduleEmergencyUpgrade` reverts unless `vault.paused() == true` at schedule time. This forces the attacker to first trigger a pause (via Council or GUARDIAN). Pause is a **public, observable trigger** — the community detects the pause within minutes, raising alarm before the 6h execute window expires.
3. **6h cancel window** — between `schedule` and `execute`, any single counter-coerced advisor can call `cancelEmergencyUpgrade(id)` to delete the scheduled operation (idempotent, mapping

write to 0). Alternatively, GUARDIAN can unpause the vault, which causes the `paused() == true` recheck at execute to fail (dual-check defense). Both paths are documented in §S2.B.4-5 implementation tests.

4. **Semestral Council drill** — the IMPL-054 §F drill matrix mandates a semestral live drill of the full emergency upgrade flow (schedule → cancel + schedule → execute) against the sandbox-fast vault, keeping the Council operationally prepared and verifying the runbook against the deployed bytecode every six months. Drill outcomes update `verified-at` / `verified-by` / `verified-bytecode` YAML fields per the IMPL-054 certification pattern.

Residual risk: bounded to a 6h response window with three independent counter-coercion paths available. No defense reduces the residual to zero (the 3-of-3 unanimous threshold is intentional — fewer signers would lower coercion cost), but the combination of geographic distinctness + public pause precondition + 6h cancel window + drill cadence is the same risk envelope as the Yearn V3 SafeMod and Morpho V2 ConfiguratorSafe production deployments.

§8.6 Top-3 residual risks (publicly disclosed)

In the spirit of an honest self-audit, the following residual risks are disclosed for external readers (depositors, auditors, integrators):

1. **GUARDIAN role on a hot EOA (Phase 0 only).** The single most-cited Phase 0 constraint. The GUARDIAN key is a hot EOA controlled by the solo operator. Mitigations: (a) GUARDIAN authority is bounded to `pause` / `unpause` / explicit emergency-revoke flows — it does NOT hold MANAGER authority; (b) 3-tier timelock (Operational 48h / Meta 72h / Emergency 6h) means a compromised GUARDIAN cannot move funds without triggering public on-chain delays; (c) the Council pause provides a redundant emergency-stop independent of GUARDIAN. **Phase 1 escalation:** hardware-wallet adoption (Ledger or Keystone) per IMPL-040 T1.1, dependent on LEGAL-004 advisor vetting.

2. **No 3-of-N multisig on MANAGER role (Phase 0 only).** The MANAGER role is held by the timelock V2 contract (which is correct architecturally — operations flow through the timelock), but the timelock's effective controller in Phase 0 is the solo-operator PROPOSER. This is a structural Phase 0 constraint of operating as a solo founder. Mitigations: (a) every Meta-tier operation requires `META_DELAY = 72h` + cosig (Path B) with COSIG_APPROVER on the sandbox Safe 2-of-3; (b) operational-tier operations have a 48h window for community observation; (c) selector retuning (`updateSelectorTier`) is itself Meta-tier-gated, so an attacker cannot trivially downgrade a sensitive selector. **Phase 1 escalation:** real 3-advisor Safe 2-of-3 as COSIG_APPROVER, separate Safe 3-of-3 as EMERGENCY_COUNCIL.

3. **No third-party audit completed at audit-scope commit.** This report is a self-audit per a tier-1 firm methodology, not a substitute for a third-party engagement. A third-party audit (target firm: Spearbit or Cantina per IMPL-053 spec) is committed for Q3 2026 (estimated). Mitigations meanwhile: (a) launch cap TVL **$500K** as documented in [STRAT-005] §2 — bounds the blast radius of any undetected vulnerability; (b) on-chain bounds and caps documented in §5.3 + §5.5 of this report cannot be bypassed by a single key compromise; (c) Council pause has a <30 minute incident-response SLA documented in `emergency-pause.md`; (d) every static-analysis and symbolic tool used by tier-1 firms (slither, aderyn, halmos, foundry invariants, fork-mainnet integration) has been run and triaged in §4 + §6 of this report.

We examined the operational risk surface and confirm that the residual risks above are publicly known, structurally bounded, and have explicit escalation paths in [STRAT-005] §2, IMPL-040 T1.1, and IMPL-053.

---

§9. Limitations of this report + commitment to external audit

Este informe es **self-audit** del equipo del protocolo. Tiene 2 limitaciones que un audit externo tier-1 (Spearbit, Cantina, Trail of Bits, OpenZeppelin) **no** tiene:

(a) **Sesgo del autor.** No podemos ver inconsistencias entre la intención declarada y la implementación; un par de ojos externos pueden.

(b) **Sin firma reputacional.** No hay marca externa respaldando el informe. Su valor probatorio depende del lector evaluándolo por mérito técnico.

§9.1 Mitigaciones del riesgo de auto-audit en este informe

- **Estructura mirror de tier-1 firm public reports.** Mirror estructural (Executive Summary + Scope + Threat Model + Findings + Non-findings + Limitations + Appendices) comparable a Spearbit / Cantina portfolios públicos (`github.com/spearbit/portfolio` + `github.com/cantina-xyz/portfolio`). Un lector técnico puede comparar directamente.
- **Cross-reference exhaustivo con OPS-014 hack registry** — 31 hacks históricos cross-walked en §3.2 con applicability + mitigation + residual risk per hack.
- **/review-doc 4-hat** (Founder / Operator / Builder / Business) durante la elaboración — el doc pasó por cada perspectiva con alpha-leak scan + estructura review.
- **Symbolic verification halmos automated (no auto-bias)** — 27/27 propiedades simbólicas verificadas por z3 solver, independiente del autor.
- **Static analysis con tooling estándar tier-1** — Slither 0.11.5 + Aderyn 0.6.8 con triages line-by-line.
- **Fork-mainnet drill end-to-end** — 5/5 PASS sobre HyperEVM mainnet real con Safe v1.4.1 desplegado vía canonical factory en-test; cierra el gap "¿funciona realmente la integración?" que sólo `vm.prank` no responde.

§9.2 Compromiso de audit externo

Contrataremos audit externo tier-1 (Spearbit o Cantina, evaluación específica al activar trigger) cuando se cumpla **cualquiera** de estos triggers:

1. **TVL ≥ $200K AUM** sostenido 30 días.
2. **Revenue del protocolo** cubre coste audit (~$16-27K estimado).
3. **6 meses post-Phase-1 launch** sin incidentes (timeline natural).

ETA estimado: **Q3 2026** (~6 meses post Phase 1 deploy W22-W24). La spec del engagement + selection criteria (Spearbit vs Cantina vs alternativas) + RFP scope se publicará como IMPL-053 cuando el trigger se active. Bidirectional cross-link entre este §9 y IMPL-053 cuando se drafte (P1-005 lockeada en /integrate IMPL-056).

Hasta entonces, las siguientes mitigaciones constrain blast radius de un finding latente que un externo pudiera detectar primero: - **Launch cap TVL $500K** (on-chain via `maxDepositAmount` setter, MANAGER + timelock-gated). - **Multisig 2-of-3** sobre COSIG_APPROVER role + 3-of-3 sobre EMERGENCY_COUNCIL Safe (post Phase 1 deploy con advisors reales). - **3-tier timelock** (Operational 48h, Meta 72h + cosig, Emergency 6h) en cada operación sensible. - **Council pause path** instantáneo + SLA <30min response del operador. - **24h rollback procedure** para findings post-publicación de este informe (§7.5 rollback playbook).

§9.3 Q3 2026 trigger review reminder

A `cron`-style reminder está scheduled para 2026-09-01 para revisar si alguno de los 3 triggers del §9.2 se cumplió. Si sí → drafteamos IMPL-053. Si no → re-evaluamos triggers + extendemos timeline + actualizamos este §9.3 con nueva ETA.

§9.4 ¿Cómo interpretar este informe?

- **Para depositantes early-Phase-1:** este documento es un **transparency artifact**, no external validation. Si tu mandate exige audit externo tier-1, espera al Q3 2026 trigger. Si tu mandate acepta self-audit con disclaimer claro + cap TVL $500K + on-chain bounds, este informe + el track record (STRAT-001) son los inputs primarios.

- **Para aggregators / portfolio managers profesionales:** evalúa la estructura del informe contra Spearbit / Cantina portfolios (linked above). Reproduce los comandos del §1.2 — si los resultados coinciden, el informe es honesto. La firma externa no exists todavía; la transparencia del proceso sí.
- **Para futuros external auditors (Spearbit / Cantina engagement post-trigger):** este informe es el handoff package — usa Appendix A (cleanup commit `f47389a` reference) + Appendix C (baseline chain) + §4 findings triage como input. Cualquier finding que descubras y este informe haya marcado FP es alpha valuable — favor disclose en remediation cycle.

El honest disclaimer: las 2 limitaciones (a) y (b) no se eliminan con este informe. Lo que sí hace este informe es **bajar la barrera de evidencia** que un depositante racional necesita para decidir si entrar al vault under Phase 0/1 conditions, sabiendo que external validation viene cuando los triggers $200K / revenue / 6m se cumplan.

---

§10. References

- SEC-002 — audit preparation security checklist
- SEC-003 — admin opsec self-assessment
- SEC-005 — v5 tier-1 surface analysis (draft)
- OPS-014 — External Hack Registry
- RESEARCH-034 — Tier-1 DeFi Audit Methodologies & Historical Hack Taxonomy
- RESEARCH-047 — Protocol premortem (failure modes & mitigations)
- IMPL-052 — V5-Final Bake-ins Implementation & Sandbox Prep
- IMPL-055 — V5.1 Implementation (#27 + §S7.C + tests/self-audit re-run)
- IMPL-056 — Public Self-Audit Consolidation & Publication (this artifact's source IMPL)
- `infra/AWS_REGISTRY.md` §11 — bytecode baselines chain
- `contracts/audit/slither-v5.1-triage.md` + `contracts/audit/aderyn-v5.1-triage.md` — static analysis triages
- `contracts/audit/v5.1-self-audit-report.md` — IMPL-055 §S4.8 consolidated self-audit acceptance gate

---

Appendix A — Cleanup commit `f47389a` diff (auditor reference, NOT audit-scope)

The `f47389a` commit (2026-04-24) was titled "refactor(vault,audit): triage slither/aderyn finding noise" and made two categories of changes:

1. **Explicit zero-init in loops** — e.g. `uint256 i = 0` instead of `uint256 i` (which is implicitly 0 in Solidity). Cosmetic; the EVM bytecode optimizer treats both identically.
2. **`slither-disable-next-line` and `forge-lint disable` comments** — annotation-only, NO code change.

The bytecode hash changed (`f99b47a7...` → `f8295c20...` after `9147ccf` post-merge fixes) because the Solidity compiler's optimizer rearranges bytecode based on AST shape — even comments can shift the bytecode if they affect parser positions. **NO semantic change** was introduced by this commit.

Files affected: `contracts/src/AspeLabsVault.sol` + `contracts/src/AspeTimelockControllerV2.sol`. Total diff ~50 LOC.

Per IMPL-055 §1 P1-004 LOCKED 2026-05-07: auditor confidence in `f47389a` being a noise-only refactor was validated via direct diff inspection. This appendix documents the diff for external-auditor reference; the auditor may independently decide whether to expand audit-scope to include `f47389a` (cost estimate: **+$1–2K** incremental to baseline engagement).

Decision for this report: `f47389a` is treated as **NOT audit-scope** (noise-reduction only; semantically equivalent to predecessor). The audit-scope baseline (`commit-audited-v5.1 CURRENT`) is the

post-§S6 size-reduction commit, which includes `f47389a` as an ancestor without re-auditing it.

To reproduce the diff (anyone can run this against the public repo at the audit commit):

```
cd /home/gorka/projects/aspe_labs
git diff 2a99550~..f47389a -- contracts/src/
```

Appendix B — Raw output references (slither + aderyn + halmos)

This appendix lists the raw artifact files produced by the static analyzers, symbolic verifier, fuzzing harness, and storage-layout drift checker. Their full contents are **not pasted here** to keep this report scannable; instead, every artifact is committed to the public git repo at the audit commit, and reproducibility commands are spelled out in §1.2 Methodology.

Artifact	Path	Description	SHA-256 (audit-scope commit)
Slither triage	`contracts/audit/slither-v5.1-triage.md`	24 findings on our contracts, 0 P0/P1, all FP categorized + reasoned (94 LOC)	`d617276851e5874c51a06abffb387d`
Aderyn triage	`contracts/audit/aderyn-v5.1-triage.md`	4 High categories (18 instances) + 14 Low categories, all FP, with defense-in-depth analysis (95 LOC)	`debabc0b4751bd682d097857c07c7d`
Aderyn raw report	`contracts/audit/aderyn-v5.1-report.md`	Full aderyn output as generated by `aderyn .`	—
Self-audit report (consolidated)	`contracts/audit/v5.1-self-audit-report.md`	IMPL-055 §S4.8 acceptance gate report (151 LOC)	`8cc83cf79527d77b15737cd8e30960`
Slither raw JSON	local-only	1.78MB — exceeds repo size limit. Reproducible via: `slither . --filter-paths 'lib/|test/' --json /tmp/slither-v5.1.json` from `contracts/` dir	
Halmos symbolic results	`contracts/halmos.toml` config + test files at `contracts/test/halmos/V5_1.t.sol` + `contracts/test/halmos/V5Final.t.sol`	27/27 PASS full discovery; reproducible via `halmos --solver-timeout-assertion 180000`	
Foundry test results	`contracts/test/v5_1/` + `contracts/test/v5final/` + `contracts/test/halmos/` + `contracts/test/integration/V5_1_EmergencyUpgradeForkE2E.t.sol`	448 PASS / 0 FAIL / 0 SKIPPED — reproducible via `forge test` (local) or `forge test --fork-url https://rpc.hyperliquid.xyz/evm` (fork file)	

Artifact	Path	Description	SHA-256 (audit-scope commit)
Gas snapshots	`contracts/gas-snapshots/v5final-postS2.txt` + `contracts/gas-snapshots/v5.1-baseline.txt` + `contracts/gas-snapshots/v5.1-delta.md`	Gas regression analysis v5-final → v5.1, including §S6 size-reduction impact	
Storage layout golden	`contracts/storage-layouts/AspeLabsVault-v5.layout` + `AspeLabsVault-v5.1-EXPECTED.layout` + `AspeTimelockController-v5.1-EXPECTED.layout`	Hand-curated ERC-7201 namespaced storage maps; drift hook `scripts/check-storage-layout.sh` validates per commit	
Test:code ratio doc	`contracts/test/v5_1/RATIO.md`	Dual-reporting ratio analysis (raw LOC + NSLOC) per bake-in #27 (5.71:1 NSLOC) and §S7.C (6.16:1 raw / 30.36:1 NSLOC)	
AWS_REGISTRY baselines	`infra/AWS_REGISTRY.md` §11	Bytecode hashes chain registry (v5-pre → v5-final → cleanup → merged → v5.1)	
One-shot reproducer	`scripts/reproduce-audit.sh`	Wrapper that runs forge clean+build + bytecode hash check + forge test + halmos full discovery + fork drill + storage layout drift in one shot (~10-15min wall-time)	

All artifacts are version-controlled in the public git repo (`btk-da/aspe_labs` at the audit commit) — readers can clone the repo at the audit commit hash and reproduce every tool output independently. Reproducibility commands (exact pinned tool versions, exact flags) are spelled out in §1.2 Methodology.

Verification command (compute file hash at audit-scope commit):

```
cd /home/gorka/projects/aspe_labs
git checkout 3ce24f1   # audit-scope commit (post PR #2 merge)
sha256sum contracts/audit/slither-v5.1-triage.md \
          contracts/audit/aderyn-v5.1-triage.md \
          contracts/audit/v5.1-self-audit-report.md
# expected hashes match the SHA-256 column above
```

Appendix C — Historical baselines chain

The audited bytecode is the most recent link in a chain of intermediate baselines that document the evolution from SEC-004 v2.0 audited state (`e923006`) to the v5.1 production candidate audited in

this report. Each baseline is registered with its `deployedBytecode` sha256 in `infra/AWS_REGISTRY.md` §11.

Baseline	Commit	Vault deployedBytecode sha256	Timelock V2 deployedBytecode sha256	Status	Description
commit-audited (v5-pre)	`e923006`	`bf5fbcc1aea09b86b9c8159ac7affca0cb9972e9fcc1d0c20b0376400c6e655e`	— (v1 timelock, deprecated)	Archived (v2.0 baseline)	Original v5.0 redeploy state pre-bake-ins. SEC-004 v2.0 audited this.
commit-audited-v5final ORIGINAL	`2a99550` (`vault/v5-final` HEAD post-S3.F)	`f99b47a7da9089`[illegible]`4e9b13dc`	(registered at same commit)	Archived (SUPERSEDED by merged baseline)	v5-final audited bake-ins #1–#23 implementation + S3.F fork-mainnet drill (IMPL-052).
(intermediate) `f47389a` cleanup	`f47389a` (2026-04-24)	(bytecode changed; not registered as discrete baseline)	(same)	Noise-only refactor (Appendix A)	Slither-disable comments + explicit zero-init; optimizer reorganized bytecode without semantic change.
commit-audited-v5final-merged	`a7669e0` (merge `vault/v5-final` → `main`) + `9147ccf` (post-merge fixes)	`f8295c20e5c9d51`[illegible]`9af48ee`	(registered at same commit)	Archived (SUPERSEDED by v5.1)	v5-final merge ceremony completed 2026-05-07 (IMPL-055 §S1 detected IMPL-052 §S4.13 gate pending → fulfilled with `--no-ff` merge).
commit-audited-v5.1 ORIGINAL §S4.3	`02d32e3` (`vault/v5.1` HEAD post-§S3.F)	`90884f1225c260`[illegible]	[illegible]	SUPERSEDED 2026-05-11 §S6	First v5.1 candidate with #27 + §S7.C; failed CI on PR #2 due to vault runtime 25,025 bytes > EIP-170 24,576 limit.

Baseline	Commit	Vault deployedBytecode sha256	Timelock V2 deployedBytecode sha256	Status	Description
commit-audited-v5.1 CURRENT (audit-scope of this report)	post-§S6 size-reduction commit (on `main` post PR #2 merge `3ce24f1` 2026-05-11)	`825c42686adba26bbf3eaf4ca39d2fe7d96304b5810ce03943921343714ceb4d` (24,527 bytes runtime, margin +49 under EIP-170)	[illegible] (14,726 bytes runtime)	**CURRENT — audit-scope**	[illegible] production candidate. Storage mirror removed + extcodesize check removed + optimizer_runs 200→100. Bytecode fits EIP-170.

Why each link matters for auditor handoff: the chain documents three categories of changes between v2.0 (audited in baseline report) and v3.0 (this report): (1) semantic delta from bake-ins #1–#23 (v5-final, ~+121 nSLOC), (2) noise-only refactor `f47389a` that nonetheless changed bytecode (Appendix A — auditor decides whether to expand scope), (3) v5.1 delta from #27 emergency upgrade tier + §S7.C decoder mitigation + §S6 size-reduction surgery. An external auditor engaging post-publication of this report would receive `commit-audited-v5.1 CURRENT` as the audit-scope; deltas to v2.0 are documented across §4 Findings + §7 Storage layout + §3 Threat model.

Verification command (anyone can reproduce the CURRENT vault baseline hash):

```bash
# Reproduce CURRENT vault baseline hash
cd /home/gorka/projects/aspe_labs/contracts
git checkout main  # post PR #2 merge
forge clean && forge build
python3 -c "import json,hashlib; d=json.load(open('out/AspeLabsVault.sol/AspeLabsVault.json')); pr
# expected: 825c42686adba26bbf3eaf4ca39d2fe7d96304b5810ce03943921343714ceb4d
```

Modificaciones

Sin modificaciones post-aprobacion.