CORROVERA
SECURITY

INDEPENDENT SECURITY ASSESSMENT · DEPLOYMENT AUDIT

ForestRoadVault
mainnet-v1

On-chain audit of the live Ethereum mainnet deployment: bytecode reproduction, role census, parameters, disclosure.

CLIENT / PROTOCOL
Forest Road — ForestRoadVault

CHAIN / DEPLOY BLOCK
Ethereum mainnet (1) · 25,768,251

SOURCE · FREEZE TAG
f1f1f47 · mainnet-v1-production-freeze-2026-08-16

SCOPE
contracts/src — 49 files · 18,244 lines

COMPILER · STATIC
solc 0.8.30 (runs 100) · Slither 0.11.6

REPORT DATE
2026-08-16

AI-ASSISTED AUDIT — NOT A CORROVERA MAXIMUM-ASSURANCE AUDIT · EVIDENCE FOR HUMAN REVIEW

Result: six findings — four Medium, one Low, one Informational — none of them a coding defect in the deployed bytecode. Every Medium concerns the *deployed configuration* or *what has been published about it*, which is the surface no prior round could examine because it did not exist until today.

// 1. What this document is, and is not

This is an **AI-assisted deployment audit** — the on-chain pass over the live deployment (bytecode, roles, parameters, capital, and the public record). It is one line of evidence, completed by the full fork-reproduced, adversarially cross-examined audit that supersedes it. Separately, the protocol carries maximum-assurance-grade adversarial evidence of *its own* (a 1,085/1,085-mutation sweep across five axes plus 126 fork-based exploitation tests, complete 2026-08-13, SEVERE 0 · MODERATE 0 · exploitable 0 — `docs/remediation/ATTACK_SURFACE_PLAN_2026-08-12.md` §6), over which this on-chain audit is a distinct, additional layer.

It is also **not a repeat of the 2026-08-04 review**. That review examined source at commit `d2ef15a`. Since then `contracts/src` has grown by **8,427 net lines across 42 files** and the protocol has been deployed. This audit's primary subject is the thing that has never been audited before: **the live deployment** — its bytecode, its role graph, its parameters, its capital, and the public record describing it.

What ran:

METHOD	COVERAGE
Independent bytecode reproduction	20 deployed contracts + 2 constructor-deployed helpers, byte-for-byte against locally compiled artifacts from the frozen tree
Exhaustive role census	276 events replayed from the deployment block; every reconstructed holder re-confirmed by live `hasRole`
Privilege negative-space sweep	7 principals × 18 modules × 17 roles = 2,142 `hasRole` probes
Full nullary state dump	every zero-argument `view`/`pure` function on every deployed module, decoded
Per-class configuration read	all five collateral classes, 14 getters each
Token distribution reconstruction	GROVE, sGROVE, USDfr, sUSDfr, by `Transfer` replay
ERC-7201 slot derivation	all 16 declared namespaces recomputed from the standard formula
Parameter conformance	every live parameter compared against the frozen `Config.sol` constant
Project test suite	2,130 tests; separate invariant campaign
Static analysis	Slither 0.11.6 over `src/`, 194 results triaged individually

METHOD	COVERAGE
Source review	FRGovernor, GroveVotesAggregator, MtmAtomicExecutor, AttestationOracle, AssessedImpairmentSource, GroveToken, CommitmentLedger in full; sUSDfr, RedemptionQueue, ReserveManager, CuratorModule, MintRedeemController, WaterfallEngine, DefaultManager, ClaimBridge, PointsModule on the paths reached by the findings
Register reconciliation	all 202 published findings parsed and used as an exclusion list

What did not run, and what that costs:

ABSENT	CONSEQUENCE
Model qualification benchmark	No lineage has measured competence at this task.
Generated fork reproduction	**No finding reaches `confirmed`.** Every finding below rests on source verification and direct chain reads, not an executed exploit.
Coverage-guided fuzzing (Echidna, Medusa) *by Corrovera*	Not run in this engagement. The mutation/exploitation axis is covered by the protocol's own 1,085-mutation sweep + 126 fork-exploitation tests (see above); this audit did not add independent fuzzing.
Symbolic execution / formal verification	Not executed. No property proven.
Independent adversarial cross-examination	Single-reviewer adjudication. The 2026-08-04 review submitted its lead finding to four independent lineages to refute; this one did not.
Live-fire governance rehearsal	No proposal was queued or executed against the deployed Governor.

A clean result in any section below is evidence that these methods at this depth found nothing there. It is not evidence of security.

Independence note. This audit consumed the project's own published findings register (202 entries) as an exclusion list. Where a finding restates or narrows a registered one, it says so and says what is new.

// 2. Executive summary

The deployed bytecode is exactly what the frozen source compiles to. That is not a formality — it was checked byte by byte, and **every single differing byte across all 22 contracts falls inside a solc-declared immutable slot and decodes to a known protocol address.** Zero unexplained bytes. The role graph is exactly what the deployment record claims. Every live parameter equals its frozen `Config.sol` constant. All 18 implementations have their initializers disabled. The deployer holds nothing, across 306 probed (module × role) pairs.

The engineering is not where the risk is. The risk is in three places the code cannot speak to:

First, there is one key quorum, not four. The deployment record separates "protocol authority" (the timelock) from "retained operational powers" (the ops Safe). On chain, the ops Safe, the treasury that holds 100% of GROVE, the fee recipient, and the approved curator are **four distinct Safe addresses with byte-identical**

owner sets and identical 2-of-4 thresholds — four EOA keys in total. Any two of them exercise every operational power immediately and command 1,000,000,000 votes against a 40,000,000 quorum, i.e. governance itself. The separation is nominal (F-01).

Second, once a proposal is queued it cannot be stopped, and no reader-facing document says so. The project knows this: ADR-0036 accepts it deliberately and designates it a **REQUIRED DISCLOSURE**. Yet the words "veto", "once queued", "queued proposal", "proposal guardian", ADR-0036, and even the two-day delay figure appear **nowhere** in the published documentation or the published findings register — the very register the deployment record tells readers to consult instead of the word "audited" (F-02).

Third, the loss cascade's first two layers are empty and nothing on chain requires them not to be. The curator is approved on all five classes and has posted **zero**; sGROVE coverage capacity is **zero**. `ClaimBridge` contains no reference to the curator module at all, and `WaterfallEngine.fund` consults none of it. The first-loss requirement is implemented as a **withdrawal ceiling on capital already posted**, never as an origination floor (F-03).

The protocol holds 100 USDC of seed capital and no user funds. Every finding is fixable before value moves, and F-03 in particular is exactly what the outstanding §8 acceptance gate exists to catch.

#	SEVERITY	STATUS	SUMMARY
F-01	**Medium**	New (extends internal `SAFE-CD-01`)	Four Safes, one 2-of-4 quorum: operations, governance, fees and first-loss capital all terminate at the same four keys. Absent from the published register
F-02	**Medium**	New	Queued governance is unstoppable; ADR-0036 designates this a required disclosure and no published surface makes it. The deployment record's `CANCELLER` wording implies a control that is unreachable
F-03	**Medium**	New	Origination and funding consult no junior capital. The first-loss requirement is a withdrawal ceiling, not an origination floor — and both cascade layers are currently empty
F-04	**Medium**	Carried forward, now live (`A-02`)	The degenerate attestation quorum shipped: threshold equals attester count equals 2, both keys unused and unfunded. Zero fault tolerance, ~10-day recovery
F-05	**Low**	New	Redemption settlement depends on two unfunded keys, and a missed epoch is lost throughput rather than deferred throughput
F-06	**Informational**	New	The mainnet production-entrypoint guard test is red on the working tree because the superseded manifest was archived after the freeze

// 3. F-01 — Four Safes, one signing quorum

Severity: Medium · Confidence: certain in the mechanism · Evidence: direct chain reads · Not reachable to `confirmed` without a key-custody attestation this audit cannot obtain

3.1 The finding

The deployment record's *Governance posture* section draws a line between authority and operations:

> Bootstrap authority is **surrendered**. The timelock holds `DEFAULT_ADMIN` and `UPGRADER` on every module. […] Retained **operational** powers, which are not protocol authority but are real: **opsAdmin** — 16 roles […]

Both halves are mechanically true and were independently confirmed. What the section does not say is that the two sides of the line are the same people.

Read from chain at block 25,768,817:

SAFE	ADDRESS	VERSION	THRESHOLD	HOLDS
opsAdmin	`0x297e88C997c2e0EDF70A5F817AAdcA2858Aa6c04`	1.4.1	2-of-4	16 operational roles
frTreasury	`0x0687a13c490B2573d4666fb3a7c21826a621215E`	1.4.1	2-of-4	1,000,000,000 GROVE (100% of supply), self-delegated
feeRecipient	`0x9Fa085bFd15e5f80D7df056e7cD654c1Dea4D297`	1.4.1	2-of-4	all protocol and performance fees
anchorCurator	`0x02C7608407E1A2f55E795bBf4Ee69A0F18a59066`	1.4.1	2-of-4	approved curator on all five classes

`getOwners()` on all four returns the identical set:

```
0x010B23B2f2A5F6bC4ce0c68A5f65c248c1129831
0x2d1da1037d2a3B7a49F343212c0b00714ae27975
0x7961523a81CC89bEb33f2F140255213a6433cDBd
0xe6E7d3C6379ad80De02F26CCC72605d0F70d5201
```

Set intersection across all four Safes: **4 of 4**. Distinct signer keys behind the entire protocol: **4**. No Safe has a module or a guard. All four are plain EOA signers.

3.2 What two keys can do

Immediately, through the ops Safe: pause `USDfr`, `sUSDfr`, `MintRedeemController`, `ReserveManager`, `RedemptionQueue`, `ClaimBridge`, `CuratorModule`, `WaterfallEngine`, `DefaultManager`, `AttestationOracle` and `sGROVE`; freeze any USDfr holder via `COMPLIANCE_ADMIN_ROLE`; originate facilities; fund them, distribute payments, declare defaults and realize losses via `SERVICER_ROLE` on `WaterfallEngine` and `DefaultManager`; and close redemption epochs via `SETTLEMENT_KEEPER_ROLE`.

After a governance cycle, through the treasury Safe: `Governor.quorum` is 40,000,000 votes; `GroveVotesAggregator.getVotes(frTreasury)` returns **1,000,000,000**. `proposalThreshold` is 1,000,000. The treasury therefore satisfies the proposal threshold 1,000× over and the quorum 25× over, and no other address holds a single GROVE. Voting delay 86,400 s + voting period 604,800 s + timelock 172,800 s = **864,000 s (10 days)** from proposal to arbitrary execution — including replacing every one of the 18 UUPS implementations, since the timelock holds `UPGRADER_ROLE` on all of them.

There is no opposing vote to lose, because there is no other voter.

3.3 Status against the register

The shared-owner condition is accepted internally as `SAFE-CD-01`, referenced in `docs/MAINNET_LAUNCH_RUNBOOK.md`, `docs/remediation/predeploy-validation/`, and ADR-0036 (which correctly cites it as the reason an "independent" proposal guardian was unavailable). `frontend/src/content/docs/security.md:136` discloses it in one line.

It is not in `frontend/src/content/audits.ts` — the 202-entry published findings register. The deployment record tells readers: "*read the register and the open findings rather than the word 'audited'.*" A reader who does exactly that does not learn this.

One of the four owner keys, `0x7961523a81CC89bEb33f2F140255213a6433cDBd`, has **transaction nonce 0** on mainnet: it has never signed anything. A 2-of-4 in which one key is operationally unproven is closer to a 2-of-3 in practice.

3.4 What would change the rating

Nothing in code. This is a key-custody and disclosure finding. Rating it lower would require either genuinely independent control domains, or the shared-quorum fact appearing in the register the deployment record points readers to. Rating it higher would require user funds to be present, which they are not yet.

// 4. F-02 — Queued governance is final, and no published document says so

Severity: Medium · Confidence: certain in the mechanism · Evidence: OpenZeppelin 5.4.0 source, live role state, exhaustive grep of the published surface

4.1 The mechanism

`FRGovernor` does not override `_validateCancel`. OpenZeppelin 5.4.0's base implementation is:

```
function _validateCancel(uint256 proposalId, address caller) internal view virtual returns (bool) {
    return (state(proposalId) == ProposalState.Pending) && caller == proposalProposer(proposalId);
}
```

`lib/openzeppelin-contracts-upgradeable/contracts/governance/GovernorUpgradeable.sol:818`

`cancel()` is the only external path to `_cancel`, and it reverts unless that predicate holds. A proposal that has been queued in the timelock is in `Queued`, never `Pending`. The `if (timelockId != 0) { $._timelock.cancel(timelockId); }` branch in `GovernorTimelockControlUpgradeable._cancel` is therefore **unreachable through any external call**.

Live state confirms the rest:

- `Timelock.CANCELLER_ROLE` → **Governor only** (exhaustive log replay; no other holder has ever been granted it, and no `RoleAdminChanged` event exists).
- `Timelock.EXECUTOR_ROLE` → **`address(0)`** — execution is permissionless once the delay elapses.

- `Timelock.getMinDelay()` → **172,800 s**.
- The counter-move — a proposal granting `CANCELLER_ROLE` to some other principal — itself takes 864,000 s, which is 5× the delay it would need to beat.

So the timelock's 48 hours is a **notice period, not a veto window**. The guardian pause does not help: pausing modules does not gate `upgradeToAndCall`, which is authorized by `UPGRADER_ROLE` on the timelock.

4.2 Why this is a finding and not a restatement of ADR-0036

It is not the mechanism that is new. ADR-0036 (*Queued governance is final; mainnet v1 has no proposal guardian*, accepted 2026-08-14) documents it with precision, including the 20,000-run address fuzz proving no address can cancel, and the observation that a reacting senior holder is still inside the 21-day redemption queue when the operation executes. That analysis is correct and this audit reproduced it independently.

What is new is that its own binding requirement has not been met. ADR-0036 §Consequences 2 reads:

> **2. Governance security now rests entirely on the vote. REQUIRED DISCLOSURE.** […] This must be disclosed plainly rather than implied — any public description of "timelock protection" that suggests a human circuit-breaker exists after a vote passes is false for mainnet v1.

Exhaustive grep of everything the site publishes — `frontend/src/content/docs/*.md` and `frontend/src/content/audits.ts` — returns **zero** occurrences of: `veto`, `once queued`, `queued proposal`, `circuit-break`, `proposal guardian`, `ADR-0036`, `0036`, or any statement of the timelock delay. The only "cannot be cancelled" copy on the site (`RedeemCard.tsx:369`) is about redemption queue entries, not governance.

Meanwhile the published governance page presents the timelock as precisely the circuit-breaker the ADR says must not be implied:

> Every economic / safety parameter. A compromised holder is the top of the threat model — hence the timelock + (prod) no-EOA rule.
> `frontend/src/content/docs/roles-and-governance.md:34-35`

And the deployment record itself frames `CANCELLER` as a live, controlled privilege:

> No authority role and no timelock `PROPOSER` / `CANCELLER` survives on the deployer or any named genesis principal
> `docs/deployments/mainnet-v1-2026-08-16-deployment-report.md:66`

That sentence is true and reads as reassurance about a control that does not function. A reader is more likely to conclude "the canceller is safely held" than "the canceller is unreachable and there is no veto at all".

4.3 Scope of the claim

This audit does **not** find that the decision was wrong. ADR-0036's reasoning — that a guardian held by the same key as the treasury protects nothing, and that the removed mechanism produced three findings in five days — is sound, and the measured surface reduction (6 branches → 0, 1 assembly block → 0) is real. The

finding is that an accepted decision's own disclosure obligation is outstanding on every surface a reader will actually reach.

4.4 One operational note

ADR-0036 §Consequences 3 records a footgun worth repeating in the operating procedure, because it is permanent: if `updateTimelock` is ever scheduled, the resulting operation can neither execute (it reverts unconditionally) nor be cancelled (no cancellation path), and occupies the timelock forever. Verified: `FRGovernor.sol:80–82`.

// 5. F-03 — The first-loss requirement is a withdrawal ceiling, not an origination floor

Severity: Medium · Confidence: high in the mechanism · Evidence: source-verified and confirmed against live state · Not reachable to `confirmed` without fork reproduction

5.1 The mechanism

The protocol's documented loss cascade has three layers: curator first-loss, then the sGROVE junior backstop, then senior burn. The first two are **advertised as protection and enforced as neither**.

`CuratorModule` computes a requirement from exposure:

```
function _requiredFirstLoss(CuratorStorage storage $, uint256 classId) private view returns (uint256) {
    uint256 exposure = $.registry.classExposure(classId);
    uint256 target = $.targets[classId];
    uint256 required = exposure < target ? exposure : target;
    uint256 marked = _markedFirstLoss($, classId);
    return marked > required ? marked : required;
}
```

`src/CuratorModule.sol:1404–1411`

and uses it in exactly one place — capping what a curator may take back out:

```
function _headroom(CuratorStorage storage $, uint256 classId, ClassPool storage pool) private view
returns (uint256) {
    uint256 required = _requiredFirstLoss($, classId);
    return pool.balance > required ? pool.balance - required : 0;
}
```

`src/CuratorModule.sol:1474–1481`, consumed at `src/CuratorModule.sol:546` inside `withdrawFirstLoss`.

`requiredFirstLoss` and `headroom` are referenced **nowhere outside `CuratorModule`**. Grepping the origination path for any curator concept returns **zero matches in the whole of `ClaimBridge.sol`**. `ClaimBridge.checkFundable` (`src/ClaimBridge.sol`) validates state, class activity, maturity, next-payment

date, the deal-identity attestations, and — for marked-to-market classes — mark freshness and LTV. It does not look at junior capital. `WaterfallEngine.fund` re-runs `checkFundable`, matches principal, charges the origination fee, and transitions the facility to `Active`. It never reads `CuratorModule`.

So the requirement binds a curator who has *already* posted. It does not bind anyone who wants to build exposure against an empty pool. There is no state in which origination reverts for want of first loss.

5.2 The live state makes this concrete

LAYER	LIVE VALUE AT BLOCK 25,768,817
`CuratorModule.poolBalance(1..5)`	`0, 0, 0, 0, 0`
`CuratorModule.firstLossTarget(1..5)`	`10,000,000e18` each
`CuratorModule.headroom(1..5)`	`0, 0, 0, 0, 0`
`CuratorModule.isApprovedCurator(class, anchorCurator)`	`true` for all five classes
`CuratorModule.postedOf(class, anchorCurator)`	`0` for all five classes
`sGROVE.totalStaked` / `coverageCapacity` / `coverageReserve`	`0` / `0` / `0`
`DefaultManager.impairmentBackstopCapacity()`	`0`
`CollateralRegistry.classExposure(1..5)`	`0, 0, 0, 0, 0`

The curator is approved and has posted nothing. The junior backstop has no stake and no reserve. **Cascade layers 1 and 2 are both empty**, and the only thing currently preventing a loss from landing on seniors is that no facility has been originated yet — a fact about the calendar, not about the code.

The codebase is aware the state is reachable: `WaterfallEngine.sol:515` reasons explicitly about "*default with curator pool 0 and sGROVE capacity 0*" in the interest-routing path. What is missing is anything that stops the book being built in it.

5.3 Why this unsettles an accepted finding

Register entry **F-01** (Medium, *Accepted*) — the curator first-loss round-trip from the 2026-08-04 review — was accepted on an actor-scope argument stated in the register itself:

> near-unreachable while Forest Road is sole curator, because **its first-loss capital is consumed before any senior loss** and franchise value dwarfs a single exit gain

That premise describes capital that does not currently exist. The acceptance is not wrong going forward, but it is **not yet established in the deployed state**, and it will not be until the anchor curator posts. This is worth recording against the register entry rather than leaving the acceptance to be read as already satisfied.

5.4 Recommendation

The cheapest correct fix is procedural and belongs in the outstanding **§8 acceptance**: require layer-1 and layer-2 funding *before* the first origination, and before any deposit path opens to users. A code-level fix — gating `WaterfallEngine.fund` on `CuratorModule.poolBalance(classId) >= _requiredFirstLoss(classId)` for the

post-funding exposure — is stronger but is a new invariant on a frozen tree and needs its own review. State whichever is chosen; do not leave the three-layer cascade described as protection while both funded layers are zero.

// 6. F-04 — The degenerate attestation quorum shipped to mainnet

Severity: Medium · Status: register entry `A-02` (Open), carried forward · Evidence: live chain reads plus source

Register entry `A-02` rated this against exactly this event:

> Rated against a **mainnet launch carrying this configuration forward**, which is the migration risk the deployment-reconciliation phase exists to catch

It was carried forward. Live:

- `ATTESTER_ROLE` holders: exactly two — `0x759C2Cd90594FdB114c5d37C5c3EA78a0033eD35` and `0xE2E544edEd1189b8A5BdA86A6F474011A397efb0`. Both plain EOAs.
- `AttestationOracle.initialize` sets a 2-of-n threshold for `CreditIssued`, `Valuation`, `PaymentReceived`, `DefaultDeclared`, `LossRealized`, `PastDueCured` and `TermsAmended`, and `setThreshold` floors those seven at 2 permanently (`src/AttestationOracle.sol:367-380`).
- **Threshold equals attester count equals 2. Fault tolerance is zero.**
- Both attester EOAs have **transaction nonce 0 and a 0 ETH balance**. Neither key has ever been exercised on mainnet, and neither can currently pay for a transaction.

Losing, rotating or simply failing to fund either key makes every value-moving attestation kind unsatisfiable. That halts valuations feeding `ReserveManager.totalBackingValue()`, payment distribution, default declaration, loss realization and past-due cure. Recovery requires `grantRole` on the oracle, whose `DEFAULT_ADMIN_ROLE` is the timelock — a **~10-day** governance cycle with no faster path.

Two things did improve and should be stated. The mainnet manifest records `attester2_DERIVED_FROM_DEPLOYER_KEY: false`, and the exhaustive sweep confirms neither attester holds any other role, and the deployer holds nothing. `SERVICER_ROLE` sits on the ops Safe, not on an attester. So `A-02`'s "the attester is also the servicer, one actor satisfies m-of-n in one transaction" half is **not** carried forward. What is carried forward is the zero-fault-tolerance half — and chain data cannot establish that two EOAs are independently custodied, so `A-02`'s signer-independence concern remains open on evidence rather than resolved.

// 7. F-05 — Exit liveness rests on two unfunded keys, and a missed epoch is lost, not deferred

Severity: Low · Evidence: source-verified and confirmed against live state

`RedemptionQueue.closeEpoch` is the only path that fills a queued redemption, and it is gated on `SETTLEMENT_KEEPER_ROLE`. The exhaustive census shows exactly two holders:

HOLDER	KIND	BALANCE	NONCE
`0x1Df7Adb9911e59d78c023B09E68711F97e05f4Cc` (queueKeeper)	EOA	**0 ETH**	**0**
`0x297e88C997c2e0EDF70A5F817AAdcA2858Aa6c04` (opsAdmin Safe)	Safe 2-of-4	0.0100 ETH	—

`Roles.SETTLEMENT_KEEPER_ROLE`'s own NatSpec requires exactly this two-holder posture — "a hot keeper and a manual ops-multisig backstop" — and warns that "a total keeper outage is bounded only by governance regranting the role". The posture is satisfied on paper. Neither holder can currently pay for a transaction of any size: the keeper has never transacted, and the ops Safe holds 0.01 ETH while also being the protocol's incident brake.

The second half is a design property worth stating to holders. The epoch budget is a **spot read**, not an accrual:

```
function availableLiquidity() public view returns (uint256) {
    QueueStorage storage $ = _storage();
    return Math.mulDiv($.reserves.idleReserve(), $.epochLiquidityBps, Config.BPS);
}
```

`src/RedemptionQueue.sol:575–578`

and a successful close resets the clock forward from *now*:

```
$.epochEndsAt = uint64(block.timestamp) + $.epochDuration;
```

`src/RedemptionQueue.sol:443`

So a day on which no epoch is closed does not bank its 167 bps for the next one. `Config.DEFAULT_EPOCH_LIQUIDITY_BPS`'s own comment characterises the parameter as `~1.67%/day ≈ 50%/30d`; that equivalence holds **only under daily keeper operation**. Realised exit capacity is an operational property of the keeper, not a property of the contract. (No published document currently states the 50%/30d figure to users, so this is a correctness note for the operating procedure and any future disclosure, not a live misstatement.)

The deployment record already lists Safe funding as an open item. This finding adds the keeper's own zero balance and the non-accrual property, neither of which is recorded there.

// 8. F-06 — The mainnet entrypoint guard test is red on the working tree

Severity: Informational · Evidence: reproduced

`forge test` on the deployment worktree reports **1,797 passed, 1 failed, 332 skipped** of 2,130. The single failure is:

```
[FAIL: vm.readFile: failed to read from ".../contracts/deployments/1.json": No such file or directory]
test_versionedGuardCannotOverwriteHistoricalManifestAndRefusesItsOwnSecondBroadcast()
  test/audit/MainnetProductionEntrypoint.t.sol
```

`git show mainnet-v1-production-freeze-2026-08-16:contracts/deployments/1.json` succeeds, so **the frozen commit is green** and the deployed bytecode is unaffected. The red is caused by the post-freeze staged rename of `contracts/deployments/1.json` to `contracts/deployments/archive/1-SUPERSEDED-20260814-block25690027.json`.

The consequence is small but specific: the test that exists to prove a versioned guard *cannot overwrite a historical manifest* is the one no longer running, on the tree an operator would actually execute, at the moment a superseded manifest was archived. Either re-point the fixture at the archive path or pin a fixture copy.

The separate invariant campaign is clean: **14 passed, 0 failed**, including the `SGroveHandler` stake/warp campaign. The 332 skips are pre-existing and were not investigated.

// 9. Verification results

Everything in this section came back clean. Read §1 before reading it as reassurance.

9.1 Bytecode reproduction — 22 of 22, zero unexplained bytes

Each deployed implementation's runtime code was fetched with `eth_getCode` and compared byte-for-byte against `deployedBytecode.object` from a local `forge build` of the frozen tree. Lengths matched exactly everywhere. Every differing byte was then located and classified against the artifact's own `immutableReferences` map:

CONTRACT	DIFFERING RUNS	COVERED BY A DECLARED IMMUTABLE SLOT	UNEXPLAINED BYTES
USDfr, sUSDfr, ComplianceRegistry, MintRedeemController, ReserveManager, RedemptionQueue, ClaimBridge, CollateralRegistry, CuratorModule, WaterfallEngine, AttestationOracle, PointsModule, GROVE, sGROVE, Governor	3 each	3 each	**0**
DefaultManager	7	7	**0**
AssessedImpairmentSource	6	6	**0**

CONTRACT	DIFFERING RUNS	COVERED BY A DECLARED IMMUTABLE SLOT	UNEXPLAINED BYTES
Timelock	0 (metadata-only trailer)	—	**0**
GroveVotesAggregator	9	9	**0**
MtmAtomicExecutor	8	8	**0**

Every 20-byte run decoded to a known protocol address: `UUPSUpgradeable.__self` (the implementation's own address) in the upgradeable modules; `sGROVE` / `GROVE` proxies in the aggregator; `DefaultManager` / `AttestationOracle` proxies in the executor.

`DefaultManager` carries two additional immutables, both deployed by its own implementation constructor and **absent from the deployment manifest**. Both were resolved and verified:

ADDRESS	IDENTIFIED AS	VERDICT
`0x45A06cc616e86aBd1681f20E2B274A516cCA7CDd`	`ConservativeImpairmentMath`	**EXACT byte match** (920 B)
`0xBD32AD1F92b70Ed3bD8301b1ce7A2748dA9fa267`	`CommitmentLedgerFactory`	**EXACT byte match** (6,212 B)

A third implicitly-deployed contract, the `CommitmentLedger` at `0xBfD49F9988125e67297BE0fF619F82d60c438322` (5,745 B, matching the local artifact length), appears as `DefaultManager.modules()[7]`. Its `manager()` returns the `DefaultManager` **proxy**, so only the proxy can write to it. **Integrators and monitoring should note that three live contracts in the trusted computing base are not listed in the deployment manifest.**

9.2 Proxy integrity — 18 of 18

- Every ERC-1967 implementation slot matches the declared implementation. **0 mismatches.**
- Every ERC-1967 admin slot and beacon slot is **zero** (correct for UUPS).
- Every proxy's `Initializable` slot reads `_initialized = 1`, `_initializing = 0` — initialized exactly once.
- **Every implementation reads `_initialized = 18446744073709551615 (type(uint64).max)`** — `_disableInitializers()` ran in all 18 constructors. The UUPS implementation-takeover class is closed.
- `proxiableUUID()` reverts through every proxy, which is the correct behaviour for OZ's `notDelegated` guard.
- Log replay shows exactly one `Upgraded` and one `Initialized(version=1)` per proxy, all within blocks 25,768,254–25,768,293. No unexpected upgrade has occurred.

9.3 Role census — exhaustive, not sampled

No module inherits `AccessControlEnumerable`, so `hasRole` alone can only confirm what an auditor already thought to ask. Instead, **all 276 events emitted by the deployed set since the deployment block were replayed** and the holder set reconstructed, then re-confirmed live.

- **80 live role grants** across the deployed set.

- **0 replay/live disagreements.**
- **0 unresolved role hashes** — every granted role hash resolves to a named constant.
- **0 `RoleAdminChanged` events** — every role's admin is `DEFAULT_ADMIN_ROLE`.
- `DEFAULT_ADMIN_ROLE` and `UPGRADER_ROLE` are held by the **timelock on all 17 role-bearing modules**; the timelock's own `DEFAULT_ADMIN` is itself.
- `CREDIT_ROLE`, `MINTER_ROLE`, `CONTROLLER_ROLE`, `LOSS_BURNER_ROLE` and `FEE_ACCOUNTING_ROLE` are held **only by protocol modules** — no EOA or Safe holds any of them.

Negative space was probed exhaustively rather than sampled — 7 principals × 18 modules × 17 roles = **2,142 `hasRole` calls**:

PRINCIPAL	ROLES HELD	
deployer `0x89575b…c11b`	**0 of 306**	bootstrap authority genuinely surrendered
frTreasury	**0 of 306**	holds no role; its power is the vote (F-01)
feeRecipient	**0 of 306**	
attester1 / attester2	1 each	`AttestationOracle.ATTESTER_ROLE` only
queueKeeper	1	`RedemptionQueue.SETTLEMENT_KEEPER_ROLE` only
opsAdmin	16	exactly the 16 the deployment record lists

The deployment record's governance-posture claims are therefore **confirmed as stated**. F-01 is not a contradiction of them; it is about what they omit.

9.4 Parameter conformance — every live value equals its frozen constant

PARAMETER	LIVE	CONFIG.SOL
Timelock `getMinDelay`	172,800	`TIMELOCK_MIN_DELAY = 2 days`
Governor `votingDelay` / `votingPeriod`	86,400 / 604,800	`1 days` / `GOV_VOTING_PERIOD = 7 days`
Governor `proposalThreshold` / quorum	1e24 / 4%	`GOV_PROPOSAL_THRESHOLD` / `GOV_QUORUM_FRACTION`
GROVE `totalSupply`	1e27	`GROVE_INITIAL_SUPPLY`
sGROVE unbonding / rewards duration	1,814,400 / 604,800	`21 days` / `SGROVE_REWARDS_DURATION`
sUSDfr performance / max	1000 / 2000 bps	`DEFAULT_` / `MAX_PERFORMANCE_FEE_BPS`
sUSDfr management / max / year	0 / 200 / 31,536,000	`DEFAULT_` / `MAX_MANAGEMENT_FEE_BPS`, `MANAGEMENT_FEE_YEAR`
WaterfallEngine protocol fee / max	1000 / 2000 bps	`DEFAULT_` / `MAX_PROTOCOL_FEE_BPS`
Queue epoch duration / liquidity	86,400 / 167 bps	`DEFAULT_EPOCH_DURATION` / `DEFAULT_EPOCH_LIQUIDITY_BPS`
Queue cooldown / min redemption	1,814,400 / 1e18	`DEFAULT_REDEEM_COOLDOWN` / `DEFAULT_MIN_REDEMPTION_VALUE`

PARAMETER	LIVE	CONFIG.SOL
sUSDfr yield vesting period	0	`DEFAULT_YIELD_VESTING_PERIOD`
Concentration (borrower / state / cap)	1500 / 2500 / 25e24	manifest, matched
DefaultManager cure / grace window, all 5 classes	86,400 / 1,814,400	`DEFAULT_MARGIN_CURE_WINDOW` / `DEFAULT_REDEEM_COOLDOWN`
Past-due weight, all 5 classes	5,000 bps	`DEFAULT_PAST_DUE_WEIGHT_BPS`

All five collateral classes are configured and active. **Zero drift.**

One live value deserves a pointer rather than a finding: `yieldVestingPeriod` is `0`, which is the configuration register entry **R14-02** (Medium, *Open*) is about. The finding there is that the acceptance is not written down, not that the value is wrong; it remains open and is now live.

9.5 Storage-layout integrity — 16 of 16

Every `@custom:storage-location erc7201:` annotation in `contracts/src` was recomputed from `keccak256(abi.encode(uint256(keccak256(ns)) - 1)) & ~bytes32(uint256(0xff))` and compared against the hardcoded constant. **16 verified correct, 0 mismatched** — including `AssessedImpairmentSource`, whose legacy-slot fallback correctly resolves to the canonical slot on this fresh deployment (`baseSource()` returns the `DefaultManager` proxy, proving the canonical slot is the populated one). This independently re-confirms the remediation of register entry `FRV-DSA-007`.

9.6 Access-control completeness

291 state-changing functions were enumerated across the deployed set and each classified. Every function that Slither or a modifier-only scan flags as unguarded was read in source; **all of them carry an explicit `msg.sender` check instead of a modifier**:

FUNCTION	GUARD		
`ReserveManager.depositUSDC`	`CONTROLLER_ROLE \	\	CREDIT_ROLE` checked in-body
`ReserveManager.recordExitPrepayment` / `consumeExitPrepayment`	`msg.sender == lossAbsorber`		
`ReserveManager.ratifyAndOpen` / `creditRecoveredIdleUSDC`	`_requireReserveLossAdmin()`		
`DefaultManager.absorbReserveLoss`	`msg.sender == reserves`		
`DefaultManager.drawForSeniorExit`	`msg.sender == controller`		
`sUSDfr.prepareRedemptionPricing`	`msg.sender == redemptionQueue`		
`PointsModule.onSharesTransfer` / `onUSDfrTransfer` / `onCuratorStakeChange` / `onCuratorLoss`	`msg.sender == vault / usdfrToken / curatorModule`		
`CuratorModule.postFirstLoss`	`approved[classId][msg.sender]`		

FUNCTION	GUARD
`CommitmentLedger.*`	`onlyManager`, live `manager()` = DefaultManager proxy

No missing access control was found. The genuinely permissionless state-changing surface is small and intentional: `AttestationOracle.attest` (a relay of threshold-signed data), `MtmAtomicExecutor.execute`, `DefaultManager.markPastDue` / `marginCall` / `liquidate` / `clearMarginCall`, `RedemptionQueue.requestRedeem` / `claim`, `CollateralRegistry.syncConcentrationBreaches`, `PointsModule.checkpoint` / `reconcile`, `sGROVE.stake` / `fundCoverage` / `notifyRewards`, `ReserveManager.recapitalize` / `reconcileIdleUSDC`, and the ERC-20/ERC-721/Governor standard surfaces.

9.7 Static analysis — 194 results, none material

Slither 0.11.6 over `src/` returned 12 High, 64 Medium, 66 Low, 52 Informational. Every High and Medium was read against source:

- **`arbitrary-send-erc20` (2 High)** — `ReserveManager.depositUSDC` and `recordPayment`. Both are role-gated in-body (§9.6), and the only standing approval in the tree is `MintRedeemController`'s `forceApprove(reserves, usdcAmount)` at `src/MintRedeemController.sol:371`, zeroed at line 388 in the same call. **False positive.**
- **`reentrancy-balance` (10 High)** — the balance-delta verification pattern the cascade uses deliberately, on `nonReentrant` functions. The only callback out of a protocol contract is the gas-capped, fail-open PointsModule hook, and `PointsModule`'s own external calls are three `view` reads (`balanceOf`, `postedOf`). **False positive.**
- **`uninitialized-local` (23), `unused-return` (23), `incorrect-equality` (13), `divide-before-multiply` (3)** — struct/array declarations assigned before use, `accrueFees()` return values deliberately discarded, exact-zero comparisons on integer accumulators, and floor-then-scale rounding that is intentional. **No material result.**

Slither found nothing that survives review. For a 18,244-line tree that is unusual, and it is consistent with sixteen prior review rounds.

9.8 Reproducibility of the frozen tree

ADR-0036 asserts that "*of the 19 production contracts, 18 are byte-identical to the tree that was swept and attacked*", with `FRGovernor` the sole exception. This was verified independently by diffing the swept tree object `fecf7d7d` against the freeze tag's `contracts/src` tree `9714bd1d`:

```
FRGovernor.sol | 90 +++++----------------------------------------
1 file changed, 13 insertions(+), 77 deletions(-)
```

The claim is exactly right. It is also the honest statement of a real gap, and the ADR makes it itself: `FRGovernor` is the one deployed contract carrying no mutation-sweep or fork-exploitation evidence, and ADR-0036's sequencing item 1 (deploy this Governor to a public testnet and run the governance suite against the live contract) has no completion record in `STATE.md` or the launch runbook. This audit read `FRGovernor` in full and found it to be a thin, standard OpenZeppelin composition with zero conditionals, zero role gates and one unconditional revert — which is what makes the absent campaign evidence tolerable rather than alarming. It is not the same thing as having run it.

// 10. Live state at the time of review

For the record, and because a reader six months from now will want to know what "the deployment" meant on day one:

USDfr total supply	100.000000 (all held by the sUSDfr vault)
sUSDfr total supply / assets	100 shares (24 dp) / 100 USDfr — exchange rate 1.000000
sUSDfr share holder	100% at `0x…dEaD` — seed shares permanently burned (inflation-attack mitigation)
ReserveManager idle USDC	100.000000 USDC; backing invariant holds, deficit 0
GROVE	1,000,000,000 — **100% at frTreasury**, self-delegated
sGROVE staked	0
Compliance allowlist	empty (`isAllowed` false for every principal probed); feeRecipient is protocol-exempt
Facilities originated	0; `CollateralRegistry.totalBookExposure` 0
Redemption queue	epoch 1, 0 requests, `availableLiquidity` 1.67 USDfr
Pause state	every pausable module **unpaused**
Attestations	none recorded; all valuation watermarks 0

No user funds are at risk today. Every Medium finding is addressable before the outstanding §8 acceptance admits value.

// 11. Recommendations, in the order they matter

1. **Publish the two disclosures.** Add `SAFE-CD-01` (shared four-owner quorum across all four Safes, including the anchor curator) and ADR-0036's queued-governance finality to `frontend/src/content/audits.ts`, the register the deployment record points readers to. Correct `roles-and-governance.md:34-35` so the timelock is not presented as the answer to a compromised holder without saying that a queued operation cannot be stopped. *(F-01, F-02)*
2. **Fund the loss cascade before the first origination**, and make that ordering an explicit §8 acceptance gate rather than an assumption. Record against register entry `F-01` that its actor-scope acceptance is not yet established on chain. *(F-03)*
3. **Fix the operational keys before value moves.** Fund the queue keeper and all four Safes; add a third attester so the 2-of-n threshold has fault tolerance; exercise the unused fourth Safe owner key and both attester keys once on mainnet so they are known-good before an incident. *(F-04, F-05)*
4. **Re-word the deployment record's `CANCELLER` sentence** so it states that the canceller is unreachable rather than that it is safely held. *(F-02)*

5. **List the three unmanifested contracts** — `ConservativeImpairmentMath`, `CommitmentLedgerFactory`, `CommitmentLedger` — in the deployment manifest and in monitoring. *(§9.1)*
6. **Re-point or pin the archived-manifest test fixture** so the versioned-entrypoint guard runs on the working tree. *(F-06)*
7. **Before the guardian pause is ever relied upon**, confirm in the operating procedure that it does not brake a queued upgrade or `grantRole`, and never schedule `updateTimelock`. *(F-02 §4.4)*

// 12. Reproduction

Every claim above is reproducible from the frozen tree and a mainnet archive endpoint.

```
git checkout mainnet-v1-production-freeze-2026-08-16
cd contracts && forge build --sizes          # local artifacts for the bytecode comparison
forge test                                    # 1797 pass / 1 environmental fail / 332 skip
forge test --match-path 'test/invariant/*'    # 14 pass
slither . --filter-paths "lib|test|script"    # 194 results, all triaged in §9.7
```

On-chain verification used batched `eth_getCode`, `eth_getStorageAt`, `eth_getLogs` and `eth_call` against chain 1 at block 25,768,817. The scripts are: `sweep_proxies.py` (ERC-1967 and `Initializable` slots), `verify_bytecode.py` and `diff_bytecode.py` (byte-for-byte comparison and immutable-slot classification), `role_census.py` (log replay and live confirmation), `principals.py` (Safe resolution), `state_dump.py` (full nullary ABI sweep), `per_class.py` (per-class configuration) and `distribution.py` (token distribution by `Transfer` replay).

Corrovera Security · *Independent minds. Corroborated truth.* A report is evidence for human review, not proof that software is secure.