CLAWDIT

Finding the void in your code

Security Audit Report

GLMR/MOVR ’ Base Migration Bridge

Moonbeam Foundation

Version: 1.2

Date: June 29, 2026

Lead Auditor: CLAWDIT Security

Report ID: CLAWDIT-2026-003

Table of Contents

1. Executive Summary 3
2. Scope & Methodology 4
3. Findings Summary 5
4. Detailed Findings 6
5. Gas Optimizations 21
6. Recommendations 22
7. Disclaimer 23

1. Executive Summary

CLAWDIT reviewed the Moonbeam Foundation's GLMR/MOVR ’ Base migration bridge — a lock-release system that moves GLMR and MOVR from Moonbeam to Base via an immutable, fixed-supply MigratedFixedToken paired with a pre-funded, UUPS-upgradeable MoonbeamAxelarBridge that delivers strictly by transfer and can never mint. Source authentication relies on Axelar gateway validation plus a pinned peer (chain + address) check and a per-nonce replay guard, with CEI ordering and a claimable fallback so a consumed nonce can never lose funds. No critical or high severity issues were identified; supply-conservation, replay, reentrancy, and access-control invariants held throughout. This report applies a trust-boundary-aware severity model: findings reachable only after a trusted boundary is crossed — a compromised Axelar validator set or a compromised owner key — are capped at Medium, since no unprivileged actor can reach them under honest operation. This report reflects the June 29, 2026 re-review addendum, which extended scope to the migrator front-end (Next.js dApp) and its server-side APIs. Every finding below has been reviewed and acknowledged by the client, including the scheduled go-live hardening — moving bridge ownership to a Timelock-behind-Safe and calling lockPeers() to permanently freeze the cross-chain peer — that closes the report's most significant centralization concern (GMB-008).

0	0	4	17	8
CRITICAL	HIGH	MEDIUM	LOW	INFO

Audit Details

Client	Moonbeam Foundation
Project	GLMR/MOVR ’ Base Migration Bridge
Chain	Base / Moonbeam
Audit Date	June 29, 2026
Lead Auditor	CLAWDIT Security

2. Scope & Methodology

Contracts in Scope

MoonbeamAxelarBridge

`0xEdaD5417237deC27e0eC2163EDc4c756f42cac74 (proxy) / 0xB527efa58A6E042E64f35d16d1dE28123390cb0E (impl)`

Chain: Base | 604 LOC

MigratedFixedToken

`TBD`

Chain: Base | 33 LOC

GLMR

`TBD`

Chain: Base | 18 LOC

MOVR

`TBD`

Chain: Base | 19 LOC

AxelarInterfaces

`N/A`

Chain: Base / Moonbeam | 79 LOC

Repository

`Commit: 7740836`

Methodology

- Manual code review
- Automated analysis (Foundry forge build + test, forge lint)
- Static analysis (Slither — slither-check-upgradeability for UUPS storage-layout safety, slither-check-erc for ERC-20/2612)
- Cross-chain GMP source-authentication & replay analysis (Axelar)
- Float / supply-conservation invariant analysis (stateful fuzz)

- CEI / reentrancy analysis
- Front-end / dApp source review (React/Next.js app-router, viem, wagmi, Axelar GMP SDK) — calldata-encoder, signing-flow and chain/account-guard analysis, plus server-side API review (SSRF, input validation, RPC trust)
- Function-selector & ABI verification (Foundry cast sig) and snapshot-vs-live on-chain reconciliation (cast call / cast storage against Base 8453 & Moonbeam 1284)
- Deployment verification — Basescan source-verification status + Sourcify check + ERC-1967 implementation-pointer reconciliation

Tools Used

- Foundry (forge build + test + lint)
- Slither
- Manual review

3. Findings Summary

ID	TITLE	SEVERITY	STATUS
GMB-011	Rate cap is bypassed by the unthrottled claimable bucket (claimDelivery)	Medium	Acknowledged
GMB-016	No fast emergency pause under the mandated owner=Timelock model — incident response inherits the timelock delay	Medium	Acknowledged
GMB-020	Live deployment holds the full float while go-live hardening is not engaged	Medium	Acknowledged
GMB-001	totalDelivered over-counts queued and failed deliveries	Low	Acknowledged
GMB-002	Cross-chain adminReclaim double-pay is procedurally gated, not oracle-enforced	Low	Acknowledged
GMB-003	adminReclaim must precede the post-close sweep or make-whole becomes off-chain only	Low	Acknowledged
GMB-004	Source-chain name is matched case-sensitively (operational footgun)	Low	Acknowledged
GMB-008	Retained owner powers (peer-repoint, recoverMigratedToken, setEcosystemFund, UUPS) — centralization / full-float-drain trust assumption	Low	Acknowledged
GMB-012	Over-float delivery poisons the claimable fence and bricks the permissionless remainder sweep	Low	Acknowledged
GMB-014	No decimal normalization: amount forwarded verbatim assumes source-asset and migrated-token decimals are equal	Low	Acknowledged
GMB-015	sweepToken bypasses the totalClaimable fence and can seize queued claimers' reserve	Low	Acknowledged
GMB-017	Deploy-lifecycle test never reaches owner=Timelock and never calls lockPeers() — the GMB-008 mitigation is asserted, not tested	Low	Acknowledged
GMB-018	Fork tests silently self-skip without --fork-url and their setup comments are inaccurate; the positive gateway-approval path is mock-only	Low	Acknowledged
GMB-021	Deployed implementation storage layout vs the audited source (in-place upgrade safety)	Low	Acknowledged

GMB-022	Deployed remediated implementation is unverified on Basescan	Low	Acknowledged
GMB-005	rateCap does not bound a single delivery larger than the cap	Informational	Acknowledged
GMB-006	No address(0) recipient guard in _execute (defense-in-depth)	Informational	Acknowledged
GMB-007	deliveryStatus can report status 2 (queued) for an unrelated nonce	Informational	Acknowledged
GMB-010	No on-chain cap binds cumulative deliveries to the float (lock-release trust assumption)	Informational	Acknowledged
GMB-013	Unused, misleading AxelarExecutable abstract contract in AxelarInterfaces.sol	Informational	Acknowledged
GMB-019	Zodiac OpsModule lets a single sub-threshold key call an onlyOwner function (pause) — module-enablement footgun	Informational	Acknowledged
GMB-FE-001	No wallet chain-id verification before signing the lock (wrong-chain robustness gap)	Low	Acknowledged
GMB-FE-002	Server-side request forgery in /api/gas-oracle via the client-controlled srcRpc (and bridge) parameters	Medium	Acknowledged
GMB-FE-003	wallet_addEthereumChain hardcodes nativeCurrency ETH for every chain, including Moonbeam (native GLMR)	Low	Acknowledged
GMB-FE-004	Recipient address validated by length/charset only — no EIP-55 checksum guard	Low	Acknowledged
GMB-FE-005	Transaction path uses window.ethereum directly, diverging from the wagmi-connected account/provider	Low	Acknowledged
GMB-FE-006	Crossing inspector reports the SENDER's Base balance as the recipient's, and labels a global aggregate as a per-crossing match	Low	Acknowledged
GMB-FE-007	Stale claim/mint-model code and comments remain in the live distribute-model front-end	Informational	Acknowledged
GMB-FE-008	Live contract addresses duplicated across five+ modules with no single source of truth	Informational	Acknowledged

4. Detailed Findings

GMB-011 Rate cap is bypassed by the unthrottled claimable bucket (claimDelivery)

Medium | Acknowledged

`MoonbeamAxelarBridge.sol:336-368, 372-390`

DESCRIPTION

The claimable mapping serves two roles: the failed-transfer fallback and the rate-cap overflow queue. When a delivery breaches the rolling window, _execute queues the amount in claimable, but claimDelivery has no rate gate, so the rate-deferred amount can be settled in the same block. Because both execute() and claimDelivery() are top-level external calls, an attacker contract can bundle every execute() and the final claimDelivery into a single transaction, delivering far more than the documented per-window ceiling in one block. There is no inter-transaction window in which an operator could pause between the queuing and the claim.

IMPACT

The rate cap is the only on-chain blast-radius control for the catastrophic cross-chain scenario — a compromised Axelar validator set or a forged/spoofed GMP. Because the bypass is atomic, the cap provides zero reaction time: it functions only as a monitoring tripwire, not the documented per-window outflow ceiling, and the real on-chain bound on such a drain becomes perTxCap times the number of approved messages — effectively the entire float. It is not reachable under honest Axelar operation, where no unprivileged actor can produce a delivery.

RECOMMENDATION

Gate claimDelivery by the same rolling window using partial settlement: settle up to the remaining window headroom and leave the remainder claimable for the next window. Also correct the 'settle in a later window' NatSpec, which did not match the unthrottled behaviour.

GMB-016 No fast emergency pause under the mandated owner=Timelock model — incident response inherits the timelock delay

Medium | Acknowledged

`MoonbeamAxelarBridge.sol:482 (pause), 404 / 422 (whenPaused recovery)`

DESCRIPTION

pause() is a plain onlyOwner function, and the two incident-response functions (adminReclaim, recoverMigratedToken) are onlyOwner whenPaused. The mandated mainnet end-state is owner=Timelock-behind-Safe, under which pause() is itself a timelocked operation (schedule -> wait MIN_DELAY -> execute). An emergency pause therefore arrives only after MIN_DELAY (2 days in the test configuration), and the whenPaused recovery functions are reachable only after that. There is no separate guardian/pauser role, and a Zodiac OpsModule fast-pause cannot pause a Timelock-owned bridge once the Safe is no longer owner.

IMPACT

The contract's only on-chain defense against an in-progress drain is pause, but under the mandated ownership it arrives ~MIN_DELAY late. Together with GMB-011 (the rate cap is bypassable atomically) and the owner-power peer-repoint drain in GMB-008, the float is left unprotected for up to MIN_DELAY once the underlying trigger (Axelar compromise or an owner-key event) occurs.

RECOMMENDATION

Add a dedicated pauser/guardian role that can pause() without the Timelock delay (e.g. the 4/7 Safe retains pause while the Timelock holds upgrade), or document that pause is delayed under owner=Timelock and that the float is unprotected during MIN_DELAY.

GMB-020 Live deployment holds the full float while go-live hardening is not engaged

Medium | Acknowledged

`Base proxy 0xEdaD5417237deC27e0eC2163EDc4c756f42cac74 (runtime state)`

DESCRIPTION

At re-review the live bridge is unpaused, holds the full ~1.237B GLMR float and has already delivered, yet none of the hardening is engaged: guardian()==0x0 (GMB-016's fast-pause has no actor), peersLocked()==false (GMB-008's peer-repoint drain is live), owner() is the 4/7 Safe directly rather than the mandated Timelock, and both blast-radius controls are off (rateCap()==0, perTxCap()==type(uint128).max). Disposition (2026-06-29): acknowledged — partial hardening is done and verified on-chain (a guardian fast-pause is set on both legs, and the Moonbeam bridge was moved off a single hot EOA to a 4/7 Gnosis Safe, the same council as Base), and the client has committed to a documented, on-chain-verifiable go-live schedule: blast-radius caps and lockPeers() at migration start, owner'Timelock at T+30d.

IMPACT

With no on-chain blast-radius bound and no peer lock, a single 4/7 multisig action can upgrade the implementation, repoint the source-auth peer, or sweep — redirecting or draining the live float — with no timelock delay. This was the dispositive residual risk for any external party relying on the bridge prior to the documented go-live schedule; risk is bounded by the float and the multisig ownership during the interim window.

RECOMMENDATION

Execute the documented go-live ceremony on mainnet: setGuardian(Safe), set a meaningful rateCap/ratePeriod and lower perTxCap, transferOwnership to the Timelock and acceptOwnership, then call lockPeers(). Re-confirm each control via on-chain reads.

GMB-001 totalDelivered over-counts queued and failed deliveries

Low | Acknowledged

`MoonbeamAxelarBridge.sol:334`

DESCRIPTION

In _execute, totalDelivered += amount executes before the rate-cap early-return and before the transfer try/catch. When a delivery is queued claimable (rate-cap breach, transfer returns false, or transfer reverts), the same amount is recorded in both totalDelivered and totalClaimable. Until/unless the recipient calls claimDelivery, totalDelivered overstates the tokens that actually left the float; a never-claimed queue overstates it permanently. This is inconsistent with windowDelivered, which is incremented only on a successful transfer. Re-review: verified — totalDelivered and the rate window are now debited only on a successful transfer.

IMPACT

Accounting/transparency inaccuracy. issuanceSnapshot() (totalDelivered, totalClaimable, totalSent) feeds dashboards; the delivered figure can exceed the float that actually moved by the outstanding queued amount. No loss of funds and no impact on the float invariant.

VULNERABLE CODE

```
// current (line 334, before rate-cap return and transfer):
delivered[nonce] = true;
totalDelivered += amount;
...
if (windowDelivered + amount > rateCap) { claimable[to] += amount; ... return; }

// recommended: count only actual outflow
try migratedToken.transfer(to, amount) returns (bool ok) {
 if (ok) { totalDelivered += amount; if (rateCap > 0) windowDelivered += amount; ... }
 else { claimable[to] += amount; ... }
}
```

RECOMMENDATION

Move totalDelivered += amount into the successful-transfer branch alongside windowDelivered += amount; or introduce a separate totalQueued counter and document totalDelivered as gross.

GMB-002 Cross-chain adminReclaim double-pay is procedurally gated, not oracle-enforced

Low | Acknowledged

`MoonbeamAxelarBridge.sol:422-442`

DESCRIPTION

adminReclaim returns a stuck locker's funds on the source chain, but the source contract cannot observe whether the Base side already delivered (or will deliver) the corresponding nonce. If a nonce is reclaimed on the source and its GMP later executes on Base, the user is paid twice. The guard is procedural: adminReclaim is whenPaused, pays only the recorded sender the recorded amount, is idempotent per nonce, and the runbook requires confirming on Axelarscan that the message is unexecuted and unexecutable before reclaiming.

IMPACT

If the operator reclaims a nonce that subsequently delivers on Base, the float on Base over-pays by that amount (a double-pay), reducing the remainder ultimately swept to the ecosystem fund. Bounded per-nonce and gated behind pause + manual verification.

RECOMMENDATION

Retain the whenPaused + idempotent + recorded-sender guards. The runbook should mandate (1) pausing both legs and (2) confirming delivered(nonce)==false on Base and that the Axelar command is unapproved/gas-unpaid, before any reclaim. Prefer topUpGas (which delivers) over adminReclaim for merely under-gassed messages.

GMB-003 adminReclaim must precede the post-close sweep or make-whole becomes off-chain only

Low | Acknowledged

`MoonbeamAxelarBridge.sol:508-525`

DESCRIPTION

sweep(to) / sweepToken(...) move the entire source-side balance after migrationEnd. A NATIVE lock retains its amount in the contract to back a future adminReclaim; if a known-stuck lock is not reclaimed before the sweep, the funds backing it leave the contract and adminReclaim then reverts (insufficient balance), so the make-whole can only be performed off-chain from the swept remainder. This is the intended sunset behavior (un-migrated value flows to the ecosystem fund).

IMPACT

Operational ordering hazard, not a vulnerability. Mis-ordering forces an off-chain make-whole rather than an on-chain one; funds are not lost to an attacker.

RECOMMENDATION

Document and enforce (operationally) that all known-stuck nonces are reclaimed before any sweep, and add this explicit ordering step to BRIDGE_MANUAL_OPS.md. Consider a pre-sweep monitoring alert on outstanding lockReceipts with reclaimed==false.

GMB-004 Source-chain name is matched case-sensitively (operational footgun)

Low | Acknowledged

`MoonbeamAxelarBridge.sol:325, 592-593`

DESCRIPTION

_execute compares the inbound sourceChain to the stored peerChain case-sensitively via keccak256 (_eq), whereas the peer address is compared case-insensitively (_eqLowerHex). The deployed peerChain string must therefore match the exact Axelar axelarId casing (e.g. 'Moonbeam', 'base'). A casing mismatch makes every inbound delivery revert with UnexpectedSourceChain, stranding all deliveries until updatePeer is corrected — which is impossible after lockPeers (would require a UUPS upgrade). Re-review: verified — source chain is now compared case-insensitively (_eqLower).

IMPACT

If misconfigured at go-live, all deliveries on that leg revert. Not exploitable by a third party, but a high-impact deploy-time mistake, especially once lockPeers() has frozen updatePeer().

RECOMMENDATION

Verify peerChain against the official Axelar config per chain before unpause and before lockPeers(). Add a deploy-script assertion / go-live checklist step comparing peerChain to the canonical axelarId. Optionally normalize chain-name casing on both sides.

GMB-008 Retained owner powers (peer-repoint, recoverMigratedToken, setEcosystemFund, UUPS) — centralization / full-float-drain trust assumption

Low | Acknowledged

`MoonbeamAxelarBridge.sol:404-408, 444-449, 499-503, 588`

DESCRIPTION

The owner can repoint the delivery token (recoverMigratedToken, whenPaused), change the remainder-sweep recipient (setEcosystemFund, anytime), repoint the source-auth peer before lockPeers() (updatePeer), and upgrade the implementation (UUPS). These are intentionally retained for deadlock recovery and rotation. The strongest concern is the peer-repoint: before lockPeers() a compromised owner can point peerAddress at an attacker contract that then emits a real (gateway-approved) GMP delivering the entire float to itself — a full-float drain, live, with no upgrade. This is the report's worst theft vector. Disposition (2026-06-29): acknowledged. The peer-repoint power is intentionally retained for pre-launch wiring correction; the client accepts it with a scheduled mitigation. Risk is already reduced — both legs are now owned by a 4/7 Safe (the prior single-EOA Moonbeam owner is gone) — and lockPeers() is scheduled on both bridges at migration start to permanently freeze the peer and close the vector for the live window.

IMPACT

A compromised or malicious owner could drain the entire float by repointing the peer before lockPeers(), redirect future remainder sweeps, or (while paused) swap the delivery token. Rated Low (centralization) because it requires a compromised or malicious owner and is mitigated by owner = 4/7 Gnosis Safe behind a Timelock on mainnet, born-paused, and calling lockPeers() at go-live.

RECOMMENDATION

Deploy with owner = Timelock-behind-Safe before mainnet go-live and call lockPeers() immediately once both legs are pinned (this closes the full-float peer-repoint drain). Make the Timelock handoff and lockPeers() hard, tested go-live gates (GMB-017), not prose assertions. Monitor setEcosystemFund and upgrade events.

GMB-012 Over-float delivery poisons the claimable fence and bricks the permissionless remainder sweep

Low | Acknowledged

`MoonbeamAxelarBridge.sol:354-367, 529-537`

DESCRIPTION

If a delivery's amount exceeds the bridge's remaining float, the transfer reverts, is caught, and the full amount is queued claimable — over-crediting totalClaimable beyond the actual token balance. Two consequences follow: the over-credited claim can never settle (claimDelivery's transfer of that amount reverts), and sweepRemainderToEcosystem computes amt = bal > fence ? bal - fence : 0 with fence = totalClaimable, so once totalClaimable exceeds the balance it sweeps 0 permanently, stranding the residual float on the permissionless path. Reproduced in BridgeSweepFence.t.sol. Re-review: verified — the solvency ceiling now stops the over-crediting; the fence-poisoning PoC reverts InsolventQueue.

IMPACT

The permissionless remainder sweep is bricked and the totalClaimable counter is permanently corrupted (clearable only by upgrade); the underlying tokens remain owner-recoverable. The precondition is a delivery exceeding the float — under honest operation (float == total locked supply) it never triggers, hence Low severity.

RECOMMENDATION

Enforce the GMB-010 global ceiling so a delivery can never exceed the float; and/or make the sweep fence min(totalClaimable, balance) so a corrupted counter cannot brick the residual sweep.

GMB-014 No decimal normalization: amount forwarded verbatim assumes source-asset and migrated-token decimals are equal

Low | Acknowledged

`MoonbeamAxelarBridge.sol:285 (lock payload), 329-367 (_execute delivery)`

DESCRIPTION

lock() encodes the raw amount into the GMP payload and _execute() delivers that exact integer via migratedToken.transfer — the bridge performs zero decimal scaling. This is correct only if the source asset (consumed on the Moonbeam side) and the migrated token (delivered on Base) have identical decimals. The NATIVE GLMR leg (18 decimals) -> migrated GLMR (18) is fine. The xcMOVR (ESCROW) leg -> migrated MOVR must be confirmed 18<->18: if xcMOVR on Moonbeam carries different decimals than the 18-decimal migrated MOVR, every delivery is mis-scaled by 10^(delta). No on-chain guard exists, since the two assets live on different chains. Off-chain go-live gate: decimals equal before enabling the ESCROW/MOVR leg; GLMR is 18<->18, MOVR off at launch.

IMPACT

If source and migrated decimals ever differ, every migrant on that leg is systematically and silently mis-paid by a factor of 10^(delta) — catastrophic in either direction. Rated Low because the GLMR launch leg is 18<->18, MOVR is disabled at launch, and it is a deploy-time fact, not an attacker action. Escalates to Critical if the xcMOVR leg is enabled with a decimal mismatch.

RECOMMENDATION

Before enabling the MOVR (ESCROW) leg, verify decimals(xcMOVR) == decimals(migrated MOVR) == 18 and record it in the deploy checklist as a hard gate. For defense in depth, either assert the source asset's decimals at deploy and the migrated token's decimals on the dest, compared off-chain in the deploy script, or add explicit scaling in lock()/_execute keyed to a stored decimals delta.

GMB-015 sweepToken bypasses the total-Claimable fence and can seize queued claimers' reserve

Low | Acknowledged

`MoonbeamAxelarBridge.sol:517-525`

DESCRIPTION

sweepRemainderToEcosystem (the permissionless dest-remainder sweep) deliberately retains the total-Claimable fence so honest queued claimers (the failed-transfer / rate-cap fallback) can always be paid. sweepToken(token, to, amount) has no such fence: after migrationEnd the owner can call sweepToken(migratedToken, owner, balanceOf(bridge)) and take the tokens reserved for queued claimers, after which their claimDelivery reverts permanently with DeliveryStillFailing. This defeats the 'queued claims are always coverable' no-loss property that every other path upholds. Re-review: verified — sweepToken now caps a migratedToken sweep to balance - totalClaimable; the seizure PoC now reverts 'exceeds claimable fence'.

IMPACT

Precondition: a compromised or malicious owner AND block.timestamp >= migrationEnd(). It sits within the GMB-008 owner-trust envelope, hence Low — but it specifically breaks the headline no-loss guarantee for queued claimers.

VULNERABLE CODE

```
function sweepToken(address token, address to, uint256 amount) external onlyOwner nonReen-
trant
{
 if (to == address(0) || token == address(0)) revert ZeroAddress();
 if (block.timestamp < migrationEnd()) revert SweepNotYetOpen();
 if (token == address(migratedToken)) {
 uint256 bal = migratedToken.balanceOf(address(this));
 uint256 maxSweep = bal > totalClaimable ? bal - totalClaimable : 0;
 require(amount <= maxSweep, "exceeds claimable fence");
}
(bool ok, bytes memory ret) = token.call(abi.encodeWithSignature("transfer(ad-
dress,uint256)", to, amount));
 require(ok && (ret.length == 0 || abi.decode(ret, (bool))), "token sweep failed");
 emit SweptToken(token, to, amount);
}
```

RECOMMENDATION

In sweepToken, when token == address(migratedToken), cap amount to balanceOf(bridge) - totalClaimable (mirroring the sweepRemainderToEcosystem fence); or forbid sweeping the migrated token entirely and route the dest float exclusively through the fenced sweepRemainderToEcosystem.

GMB-017 Deploy-lifecycle test never reaches owner=Timelock and never calls lockPeers() — the GMB-008 mitigation is asserted, not tested

Low | Acknowledged

`test/MainnetDeployUnderSafe.t.sol; test/BridgeTimelockGovernance.t.sol`

DESCRIPTION

MainnetDeployUnderSafe presents itself as 'the exact ownership end-state the mainnet deploy produces' but ends owner=Safe-direct — the transferOwnership/acceptOwnership handoff to the Timelock is never exercised. lockPeers() is never called in any Safe/Timelock/Zodiac lifecycle test (only under a plain EOA). So every 'mainnet' deploy test leaves peersLocked=false — the exact GMB-008 state in which a compromised/colluding owner can repoint the peer and drain the whole float with no upgrade. Re-review: verified — a go-live ceremony test now exercises deploy -> setGuardian -> transferOwnership(Timelock) -> acceptOwnership -> lockPeers() -> guardian fast-pause end to end.

IMPACT

The latch (lockPeers) that mitigates the worst theft vector (GMB-008) and the Timelock handoff that GMB-008/GMB-016 depend on were asserted in prose but never demonstrated end-to-end. The code is sound; this was a test-assurance and deploy-procedure gap, not a code defect.

RECOMMENDATION

Add a deploy-lifecycle test that performs the real ceremony: deploy -> pin both peers -> transferOwnership to the Timelock -> acceptOwnership -> assert owner==Timelock -> call lockPeers() and assert peersLocked. Make lockPeers() and the Timelock handoff hard, tested go-live gates.

GMB-018 Fork tests silently self-skip without --fork-url and their setup comments are inaccurate; the positive gateway-approval path is mock-only

Low | Acknowledged

`test/BridgeBaseFork.t.sol:16-20, 38, 57; test/BridgeMoonbeamFork.t.sol:33-34, 54, 63`

DESCRIPTION

Both fork tests' header comments claim the fork is created in setUp() via createSelectFork, but there is no createSelectFork anywhere in test/. setUp() only reads forked = (block.chainid == expected); both test bodies early-return with emit log('SKIP') when not forked. Under a plain forge test (chainid 31337) real-gateway coverage runs only if --fork-url is passed. Consequently the positive delivery path (validateContractCall == true -> transfer) is exercised only against MockAxelar.validateContractCall, which returns true unconditionally; there is no relayed round-trip against a real approved Axelar command.

IMPACT

No on-chain vulnerability. However the 'live Base + Moonbeam fork tests — real Axelar gateway' characterization is only partially accurate: on a Base fork the real gateway genuinely rejects a forged command, but no test sets up a real approved command, and both fork tests no-op under a default forge test run.

RECOMMENDATION

Either make the fork tests create their own fork (vm.createSelectFork with an RPC env var) or fix the comments to state they require --fork-url and self-skip otherwise. Add a positive round-trip test that approves a command on the real gateway (or a faithful harness) so the validateContractCall==true -> transfer path is covered outside the always-true mock.

GMB-021 Deployed implementation storage layout vs the audited source (in-place upgrade safety)

Low | Acknowledged

`MoonbeamAxelarBridge.sol storage layout; live impl 0xB527efa58A6E042E64f35d16d1dE28123390cb0E`

DESCRIPTION

The originally audited source packed lockEnabled before closeAt, diverging from the deployed layout. The remediated source moves lockEnabled to an appended slot so closeAt occupies slot 3 offset 21, matching the live proxy. forge inspect storage-layout confirms slots 0-18 are byte-identical to the deployed layout, and the live proxy reads closeAt, owner, totalDelivered and float back consistently after the in-place upgrade.

IMPACT

A storage-layout mismatch on an in-place UUPS upgrade could corrupt closeAt or the claimable accounting and brick or misdirect the float. The aligned layout and consistent post-upgrade reads remove that risk for the in-scope source.

RECOMMENDATION

Resolved for the in-scope source. Residual bytecode-level confirmation depends on verifying the implementation on Basescan (see GMB-022).

GMB-022 Deployed remediated implementation is unverified on Basescan

Low | Acknowledged

```
Base impl 0xB527efa58A6E042E64f35d16d1dE28123390cb0E (proxy
0xEdaD5417237deC27e0eC2163EDc4c756f42cac74)
```

DESCRIPTION

At re-review the deployed implementation 0xB527efa5 was not verified on Basescan (only raw bytecode was exposed). The initial re-review linked the audited source to the live contract through behavioural on-chain evidence (ERC-1967 pointer plus consistent storage reads), not bytecode-level source verification. Re-review (2026-06-29): resolved — Basescan now reports the implementation as 'Contract: Verified' with an Exact Match (compiler version + optimization settings exposed); a third party can now confirm the audited source equals the deployed bytecode. Residual: the contract is not yet published to Sourcify — recommend completing that and re-verifying on every future upgrade.

IMPACT

Before verification, no third party (exchange or external reviewer) could independently confirm that the audited code was the code running over the ~1.237B GLMR float — typically a blocking item in institutional due diligence. This is now resolved on Basescan.

RECOMMENDATION

Publish the implementation and proxy to Sourcify as well, with the exact compiler settings and the remediated sources; re-verify on every subsequent upgrade.

GMB-005 rateCap does not bound a single delivery larger than the cap

Informational | Acknowledged

```
MoonbeamAxelarBridge.sol:337-356, 372-390
```

DESCRIPTION

If a delivery amount exceeds rateCap, windowDelivered + amount > rateCap is always true, so the delivery is permanently queued claimable. However claimDelivery is not rate-limited and settles the full amount immediately. The rate cap therefore bounds aggregate small deliveries within a window but not a single large delivery; the only hard per-delivery ceiling is perTxCap. Shares its root cause with GMB-011. Re-review: verified — claimDelivery now partial-settles a >cap claim across windows; material only when rateCap>0 (live rateCap==0, see GMB-020).

IMPACT

The rolling window is not a true blast-radius ceiling unless perTxCap <= rateCap. No funds at risk; perTxCap remains the enforced per-tx bound.

RECOMMENDATION

If the window is intended as a hard ceiling, enforce or document perTxCap <= rateCap (e.g. a check in setRateCap / lowerPerTxCap) and add a NatSpec note that claimDelivery intentionally bypasses the rate cap.

GMB-006 No address(0) recipient guard in _execute (defense-in-depth)

Informational | Acknowledged

`MoonbeamAxelarBridge.sol:329-367`

DESCRIPTION

_execute decodes (to, amount, nonce) from the GMP payload and does not check to != address(0). A transfer to address(0) reverts in OZ ERC20, is caught, and queues claimable[address(0)] += amount; claimDelivery(address(0)) would also revert, leaving the amount fenced but unclaimable (and, like GMB-012, poisoning the totalClaimable fence). This is not reachable in practice because lock() enforces recipient != 0 and the peer is pinned/trusted, so the payload's to is always non-zero. Re-review: verified — _execute reverts RecipientZero on to==address(0).

IMPACT

None under the trusted-peer model. Listed for defense-in-depth completeness.

RECOMMENDATION

Optionally add an explicit to == address(0) branch (skip or revert) to make the invariant local rather than relying on the source-side check + pinned peer.

GMB-007 deliveryStatus can report status 2 (queued) for an unrelated nonce

Informational | Acknowledged

`MoonbeamAxelarBridge.sol:569-573`

DESCRIPTION

deliveryStatus(nonce, recipient) returns 2 whenever claimable[recipient] > 0, regardless of which nonce produced the queued balance. A recipient with any queued amount will show 2 for an unrelated, never-delivered nonce whose true status is 0. Re-review: view-only/cosmetic; documented.

IMPACT

View-only; cosmetic dashboard inaccuracy. No on-chain effect.

RECOMMENDATION

Document that status 2 indicates the recipient has a queued balance (not that the specific nonce is queued), or track queued state per nonce if precise per-nonce status is required.

GMB-010 No on-chain cap binds cumulative deliveries to the float (lock-release trust assumption)

Informational | Acknowledged

`MoonbeamAxelarBridge.sol:317-368`

DESCRIPTION

The bridge enforces no invariant that totalDelivered + totalClaimable <= the migrated-token float. _execute will transfer or queue any amount the pinned peer emits. Solvency of the claimable fence (balanceOf(bridge) >= totalClaimable) therefore depends entirely on the off-chain guarantee that the source side never emits locks summing beyond the genesis supply — the documented 'max exposure = the float' model. A stateful invariant fuzz confirms solvency, value conservation, and the fence-sum hold under that lock<=supply assumption; once it is violated (over-emission), solvency breaks. This is the root cause of GMB-012 and the over-credit case of GMB-006. Re-review: verified — _queueClaimable reverts InsolventQueue past the float and un-consumes the GMP nonce (re-relayable).

IMPACT

A peer-authentication bug, an under-funded/misconfigured float, or a compromised pinned peer could silently under-collateralize queued claims. Inherent to lock-release bridges; bounded by the float and mitigated by pinned-peer auth and the born-paused / lockPeers go-live posture.

RECOMMENDATION

Enforce a global ceiling: revert (or queue with an alarm event) any delivery that would push totalDelivered + totalClaimable past the funded float, applied at both the direct-transfer and queue paths. At minimum, monitor balanceOf(bridge) >= totalClaimable on-chain via an alert.

GMB-013 Unused, misleading AxelarExecutable abstract contract in AxelarInterfaces.sol

Informational | Acknowledged

`AxelarInterfaces.sol:47-79`

DESCRIPTION

AxelarInterfaces.sol declares an abstract contract AxelarExecutable whose NatSpec states it is 'the base contract our destination-side bridge inherits.' The bridge does NOT inherit it: MoonbeamAxelarBridge reimplements execute() with its own gateway validation and its own NotApprovedByGateway error. AxelarExecutable is therefore unused dead code with a comment asserting an inheritance relationship that does not exist. Surfaced by Slither's unimplemented-functions detector and confirmed by manual review. Re-review: resolved — the misleading NatSpec was corrected.

IMPACT

No security impact — the in-scope execute() is correct and fork-verified against the real Axelar gateway. The risk is reviewer/maintainer confusion.

RECOMMENDATION

Delete AxelarExecutable (and its duplicate NotApprovedByGateway error) from AxelarInterfaces.sol, or correct the NatSpec to 'reference only; the bridge implements its own gateway-validated execute().' Keep only the IAxelarGateway / IAxelarGasService interfaces the bridge actually imports.

GMB-019 Zodiac OpsModule lets a single sub-threshold key call an onlyOwner function (pause) — module-enablement footgun

Informational | Acknowledged

`test/MoonbeamAxelarLifecycleZodiac.t.sol:43-58, 140, 167`

DESCRIPTION

In the lifecycle test, OpsModule.pauseBridge is onlyOwner where the module owner is a single key, and the module is enabled on the owner Safe; it pauses the bridge via the avatar without the 4/7 threshold. As wired it is narrow and safe (pause-only, Enum.Operation.Call not delegatecall, no escalation to upgrade/updatePeer/sweep), but it demonstrates the structural footgun: any module enabled on the owner Safe bypasses the threshold for whatever it can execute, and the bridge has no defense (it trusts owner()==Safe). A broader production module would be a single-key bypass of all owner powers.

IMPACT

None as wired (pause-only griefing). Listed because module governance is a recurring source of full-privilege bypass; a future, broader module enabled on the owner Safe would inherit all onlyOwner powers without the multisig threshold.

RECOMMENDATION

Bound and review every module enabled on the owner Safe; keep modules pause-only and Call-only; never enable a module that can reach upgrade/updatePeer/sweep/setEcosystemFund. Document the module allowlist as a go-live gate.

GMB-FE-001 **No wallet chain-id verification before signing the lock (wrong-chain robustness gap)** Low Acknowledged

`migrator/_axelar-bridge.tsx: switchChain() 922-949, send() 951-989, doBridge() 1087-1176`

DESCRIPTION

doBridge() calls await switchChain(sChain) and then send()s the lock with eth_sendTransaction carrying no chainId parameter, so the transaction executes on whatever chain the wallet is actually on. switchChain() only throws if the RPC call itself rejects; several wallets resolve wallet_switchEthereumChain without actually switching (pending user action, multi-wallet, or the chain being mid-add), and the user can also manually flip the network between the switch and the signature. The eth_estimateGas pre-flight in send() is the only guard, but for the NATIVE GLMR path the lock carries value = amount + gasValue to the SOURCE bridge address; if the wallet is on the wrong chain that address is almost always code-less there, so eth_estimateGas returns the bare 21000 transfer cost and does not revert — the pre-flight passes and the wallet pops a normal-looking transfer. Fixed & verified (2026-06-29): assertOnChain(c) reads eth_chainId and throws on mismatch; it is called both after switchChain() and again inside send() at signing time, so a value-bearing lock can no longer be signed on the wrong chain.

IMPACT

If the lock is signed on the wrong chain, the user broadcasts a native-value transfer to the source-bridge address on a chain where it almost always holds no code. A real loss requires the narrow combination of a small amount, the user coincidentally holding enough native coin on the wrong chain, and the target address being code-less there, hence Low — but the missing assertion was a genuine safety gap on a funds-moving action.

RECOMMENDATION

After switchChain(), read eth_chainId and assert it equals sChain.chainId immediately before calling send(); abort with a clear error otherwise. Prefer the wagmi connector's chain-aware send (with an explicit chainId) over a raw window.ethereum eth_sendTransaction, so the wallet itself rejects a wrong-chain signature.

GMB-FE-002 Server-side request forgery in /api/gas-oracle via the client-controlled srcRpc (and bridge) parameters

Medium | Acknowledged

`api/gas-oracle/route.ts: sourceRpc/bridge resolution ~55-90; reached from migrator/_axelar-bridge.tsx oracleGasValue() 1030-1062`

DESCRIPTION

/api/gas-oracle read srcRpc straight from the query string (sourceRpc = u.searchParams.get('srcRpc') ?? srcCfg?.rpc) and passed it to priceDeposit(), together with a likewise client-controlled bridge param used as the eth_call to. The route was unauthenticated and force-dynamic. priceDeposit performs on-chain getters (gasPolicy / quoteLock) via that RPC, i.e. a server-side eth_call against a caller-supplied URL — so any client could drive outbound server requests to a host of their choosing. Fixed & verified (2026-06-29, commit d2507ea): /api/gas-oracle now resolves sourceRpc strictly from the server-side registry (const sourceRpc = srcCfg?.rpc); the client srcRpc param is no longer read, and a client ?bridge= is honored only when knownBridge() matches a registry deployment. No client-controlled URL reaches priceDeposit.

IMPACT

A remote, unauthenticated caller could drive server-side requests to internal-only endpoints (the bridge-synchronizer on 127.0.0.1:8092, other localhost/RPC services) and cloud metadata (169.254.169.254), enabling internal network/port discovery and request smuggling from the server's network position. The route returned gas numbers derived from state rather than the raw upstream body, so direct data exfiltration was limited (semi-blind SSRF), but the internal-reachability and probing primitive was real.

RECOMMENDATION

Do not accept an RPC URL from the client. Resolve srcRpc strictly from the server-side registry by (vendor, chainId), ignoring any caller-supplied srcRpc. Validate bridge against the registry's known deployments and reject anything else. If a client override must remain, allowlist it to a fixed set of known RPC origins and block private/link-local/metadata ranges.

GMB-FE-003 wallet_addEthereumChain hard-codes nativeCurrency ETH for every chain, including Moonbeam (native GLMR)

Low | Acknowledged

`migrator/_axelar-bridge.tsx: switchChain() add-chain branch 936-946`

DESCRIPTION

When wallet_switchEthereumChain returns code 4902 (chain unknown to the wallet), switchChain() adds it with nativeCurrency: { name: 'ETH', symbol: 'ETH', decimals: 18 } for all chains. For Moonbeam (chainId 1284) the native currency is GLMR, not ETH; the function is generic and applies the same ETH literal to the source chain a GLMR migrant must add. Fixed & verified (commit d2507ea): ChainInfo carries a per-chain nativeCurrency (GLMR for Moonbeam 1284) and switchChain()'s wallet_addEthereumChain now passes c.nativeCurrency ?? ETH.

IMPACT

A first-time Moonbeam user who adds the network through the portal saw their GLMR balance and the lock's value field denominated as 'ETH'. On the NATIVE path the lock spends amount + gasValue of the native token, so a user mis-reading GLMR as ETH could badly misjudge what they are about to send. No direct theft, but a real comprehension/UX-safety defect on a funds-moving screen.

RECOMMENDATION

Carry the correct nativeCurrency per ChainInfo (GLMR for 1284, GLMR/MOVR/ETH as appropriate) and pass it into wallet_addEthereumChain, rather than a hardcoded ETH literal.

GMB-FE-004 Recipient address validated by length/charset only — no EIP-55 checksum guard

Low | Acknowledged

`migrator/_axelar-bridge.tsx: recip validation 915-918, used in doBridge() 1130 / encodeLockV2`

DESCRIPTION

The recipient is validated with /^0x[0-9a-fA-F]{40}$/ only. A mistyped address that still has 40 hex characters passes validation and is forwarded verbatim into the lock calldata. There is no EIP-55 checksum verification (viem getAddress / isAddress), which is the standard client-side defence against single-character recipient typos. Fixed & verified (commit d2507ea): recipient is validated with EIP-55 via viem getAddress (isChecksumValid); a mixed-case address failing the checksum is rejected (all-lower/all-upper still accepted, as expected).

IMPACT

A user editing the recipient (the field is pre-filled with their own wallet but editable) could enter a typo'd-yet-valid-length address and have the migrated GLMR delivered, irreversibly, to the wrong account. The contract's only recipient guard is the address(0) check (GMB-006), which does not catch a non-zero typo. Bounded by the user's own transferred amount, hence Low.

RECOMMENDATION

Validate the recipient with EIP-55 checksum (viem getAddress in a try/catch, or isAddress with strict checksum). Reject mixed-case addresses that fail the checksum; for all-lower/all-upper input, surface a confirmation of the checksummed form before locking.

GMB-FE-005 Transaction path uses window.ethereum directly, diverging from the wagmi-connected account/provider

Low | Acknowledged

`migrator/_axelar-bridge.tsx: switchChain()/send() read window.ethereum 922-989; account from wagmi useAccount 23`

DESCRIPTION

The displayed account (walletAddr) is sourced from wagmi useAccount(), but chain switching and the actual eth_sendTransaction are issued against the raw injected window.ethereum. With multiple injected wallets (the window.ethereum provider can differ from the one wagmi connected, and EIP-6963 multi-provider setups make window.ethereum ambiguous), the signing provider may not be the one backing the displayed account. Mitigated & verified (commit d2507ea): send() asserts the injected provider's active account (eth_accounts[0]) equals the displayed wagmi walletAddr before signing. Best-effort (falls through if eth_accounts is unsupported), backed by the hard chain guard; acceptable for the residual multi-wallet risk.

IMPACT

The chain switch and lock could be submitted against a different wallet than the one shown, undermining the 'what you see is what you sign' guarantee on a funds-moving action; impact is operational/confusion rather than direct theft, hence Low.

RECOMMENDATION

Route both the chain switch and the transaction through the wagmi/viem connector that produced walletAddr (e.g. wagmi useWalletClient / sendTransaction), so the provider, chain and account are consistent and EIP-6963-aware. Avoid reading window.ethereum directly for signing.

GMB-FE-006 Crossing inspector reports the SENDER's Base balance as the recipient's, and labels a global aggregate as a per-crossing match

Low | Acknowledged

`api/crossing-inspect/route.ts: recipient=lock.sender ~ recipientBaseWei block; 'reconcile' check (totalDelivered>=totalSent)`

DESCRIPTION

The inspector derives recipient = lock.sender (the on-chain lockReceipts(nonce) stores only the sender, not the chosen recipient) and then reads balanceOf(sender) on Base, surfacing it as 'In your wallet on Base'. When a user bridges to a different recipient than themselves (a supported flow), this shows the wrong party's balance. Separately, the 'Amounts match exactly' / reconcile check asserts totalDelivered >= totalSent — a global solvency aggregate across all crossings — but presents it per-nonce as if this crossing's amounts were reconciled. Fixed & verified (commit d2507ea): the inspector still reads lock.sender (the only on-chain recipient signal) but now labels it honestly — 'sender's balance on Base' (not 'your wallet') — and the reconcile row reads 'bridge totals reconcile (all crossings)' / 'bridge-wide' rather than a per-crossing match. The misleading framing is removed.

IMPACT

Delivery-truth display could mislead: a sender bridging to a third party might see a confirmation implying THEY received funds (or a false zero), and the 'amounts match' line could read green for a specific crossing on the strength of unrelated global totals. No fund movement is affected, but the inspector is the user's trust signal for 'did it actually arrive', so a misleading reading was a meaningful UX-integrity defect.

RECOMMENDATION

Where a distinct recipient is known (decode it from the BridgeOut event / dest Released log rather than the source lockReceipt), read balanceOf(actualRecipient) and label it accordingly; when only the sender is known, label the row 'sender balance' rather than 'your wallet'. Reframe the reconcile line as a global solvency indicator, or compute a genuinely per-crossing check for the 'amounts match' claim.

GMB-FE-007 Stale claim/mint-model code and comments remain in the live distribute-model front-end

Informational | Acknowledged

`migrator/_lib.ts (BALANCES/PROOFS Merkle verify); migrator/_axelar-bridge.tsx header 10-13 and rec() vendor 'mint' 1003-1010`

DESCRIPTION

The production model is distribute (lock on Moonbeam, deliver from a pre-funded float on Base; NATIVE GLMR, no mint, no claim). Yet the front-end still carries obsolete-model artifacts: _lib.ts imports a BALANCES/PROOFS snapshot and implements Merkle leaf/proof verification (a claim/airdrop pattern); the _axelar-bridge.tsx header documents an approve(bridge, amount) + burnFrom GLMR flow that is only the ESCROW/MOVR path, not the live native path; and GLMR-mainnet crossings are tagged vendor: 'mint' / bridgeId: 'glmr-moonbeam-base' so the tracker filters them in. Fixed & verified (commit d2507ea): the _axelar-bridge header now documents the live NATIVE distribute (lock-release) flow; the obsolete claim/mint copy was rewritten.

IMPACT

No direct security impact, but the dead claim/mint code and contradictory comments raised the risk that a future edit wires the wrong (obsolete) model or that the delivery tracker silently drops crossings if the 'mint' vendor string is changed. Documentation/code drift on a funds-moving surface.

RECOMMENDATION

Remove the unused Merkle/BALANCES/PROOFS claim path (or clearly quarantine it if a non-GLMR route still needs it), correct the header to describe the live NATIVE GLMR flow (msg.value = amount + gas, no approve/burnFrom), and replace the 'mint' vendor tag with a model-accurate constant documented in one place.

GMB-FE-008 Live contract addresses duplicated across five+ modules with no single source of truth

Informational | Acknowledged

`migration-data/base-mainnet.json; migrator/_axelar-bridge.tsx VENDOR_ROUTES 356-358; migrator/_verify-contracts.tsx 26,33-35; api/crossing-inspect/route.ts SRC/DEST; api/crossings/route.ts; api/treasury/route.ts`

DESCRIPTION

The bridge, token, and Safe addresses are hardcoded independently in at least five places (the env registry, VENDOR_ROUTES, the Verify-tab table, the crossing-inspect constants, and treasury). At this snapshot they are consistent and match the live on-chain deployment, so there is no drift today. The issue is structural: the Verify tab — which exists precisely so holders can trust-but-verify the addresses the UI calls — derives its addresses from a different literal than the code paths that actually build the transaction. Fixed & verified (commit d2507ea): bridge-registry exposes glmrMainnetBridge() as the single source of truth; VENDOR_ROUTES and api/crossing-inspect both import the live addresses from it, closing the cross-module drift.

IMPACT

On any future redeploy/migration of a leg, an update applied to some modules but not the Verify table (or vice versa) would show users an address the bridge does not actually call, defeating the verification feature and enabling a convincing wrong-address display. No current exploit; Informational.

RECOMMENDATION

Make the env registry (base-mainnet.json / bridge-registry) the single source of truth and have VENDOR_ROUTES, _verify-contracts, crossing-inspect, crossings and treasury all import from it. Add a build/test assertion that the Verify-tab addresses equal the addresses used to construct the lock.

5. Gas Optimizations

No significant gas optimization opportunities were identified.

6. Recommendations

1. [Implemented] (GMB-011) Gate claimDelivery by the rate-cap rolling window with partial settlement so the per-window outflow ceiling is actually enforced, not merely a tripwire.
2. [Implemented] (GMB-016) Add a guardian/pauser role that can pause() without the Timelock delay, so emergency response is immediate under the mandated owner=Timelock model.
3. [Implemented] (GMB-010, GMB-012) Enforce a global solvency ceiling binding totalDelivered + total-Claimable to the funded float at both the direct-transfer and queue paths; a local fence cap alone does not restore solvency.
4. [Implemented] (GMB-015) Fence sweepToken against the migrated-token float so it cannot seize the reserve owed to queued claimers.
5. [Pending — go-live gate] (GMB-014) Before enabling the MOVR/ESCROW leg, verify decimals(xcMOVR) == 18 as a hard go-live gate.
6. [Implemented] (GMB-004) Verify peerChain matches the canonical Axelar axelarId casing before unpause and before lockPeers().
7. [Scheduled — go-live: 4/7 Safe done; Timelock + lockPeers pending] (GMB-008) Deploy with owner = Timelock-behind-Safe, deploy a fresh proxy (never upgrade the live one — storage layout differs), and call lockPeers() immediately once both legs are pinned to close the peer-repoint drain.
8. [Implemented] (GMB-017) Add a deploy-lifecycle test that reaches owner=Timelock and calls lockPeers(), making both hard, tested go-live gates.
9. [Acknowledged — operational/runbook] (GMB-002, GMB-003) Strengthen the reclaim runbook: pause both legs and confirm the nonce is unexecuted/unexecutable before adminReclaim, and reclaim all known-stuck nonces before any sweep (add this ordering to BRIDGE_MANUAL_OPS.md).
10. [Implemented] (GMB-001) Move the totalDelivered increment into the successful-transfer branch so it reflects only actual outflow.
11. [Implemented] (GMB-005) If the rate window is intended as a hard ceiling, enforce or document perTxCap <= rateCap.
12. [Acknowledged — open test-assurance item] (GMB-018) Make the fork tests create their own fork (or correct the misleading comments) and add a positive real-gateway round-trip so the approval path is not mock-only.
13. [Implemented] (Documentation) Correct the storage-gap description in SELF_AUDIT.md to __gap 40->37 for 3 appended slots (the code is correct), and align the go-live ordering across the runbook and checklist (pin peers before unpause).

14. [Implemented] Front-end: assert eth_chainId == source chain after switchChain() and before signing the lock; prefer the wagmi/viem connector (with explicit chainId) over raw window.ethereum eth_sendTransaction (GMB-FE-001, GMB-FE-005).

15. [Implemented] Front-end API: never accept an RPC URL from the client — resolve srcRpc and the bridge address from the server-side registry and block private/metadata ranges (GMB-FE-002).

16. [Implemented] Front-end: validate the recipient with an EIP-55 checksum and confirm the checksummed form before locking; set the correct nativeCurrency per chain when adding a network (GMB-FE-004, GMB-FE-003).

17. [Implemented] Front-end: make the address registry the single source of truth for every module and add a test asserting the Verify-tab addresses equal the lock target; remove the obsolete claim/mint-model code and comments (GMB-FE-008, GMB-FE-007).

18. [Implemented] Front-end: review @/lib/bridge-logs (the calldata encoders) — out of the snapshot — to fully verify the lock(...) ABI encoding against the deployed MoonbeamAxelarBridge.

Code Quality Assessment

8.5/10

- Strong source authentication: Axelar validateContractCall + pinned peer (exact chain, lowercased address) + per-nonce replay guard, with CEI ordering (delivered[nonce] set before the external transfer)
- Robust no-loss design: failed/rate-capped deliveries are queued claimable rather than lost; the nonce is consumed exactly once
- Immutable token: MigratedFixedToken mints once at genesis with no mint/owner/pause/upgrade path — supply is fixed and provable
- nonReentrant on every state-mutating external entry; state mutated before external calls
- Born paused, lockEnabled direction switch, and one-way lockPeers() latch give a clean, fail-safe go-live posture
- UUPS storage layout is append-only and upgrade-safe: __gap is 40 -> 37 for exactly 3 appended slots
- The two Medium findings (GMB-011, GMB-016) concern the on-chain blast-radius controls, not the core value flow, which is sound
- Solvency rests on an off-chain lock<=supply assumption (GMB-010); a global on-chain ceiling would make it self-enforcing

7. Disclaimer

> **Important Notice**
>
> This audit report is provided "as is" and for informational purposes only. CLAWDIT makes no warranties, express or implied, regarding the findings, recommendations, or any other information provided in this report. The audit is not a guarantee of the absence of bugs or vulnerabilities. Smart contracts are inherently risky, and users should always do their own research before interacting with any smart contract.

CLAWDIT

Finding the void in your code

clawdit.xyz