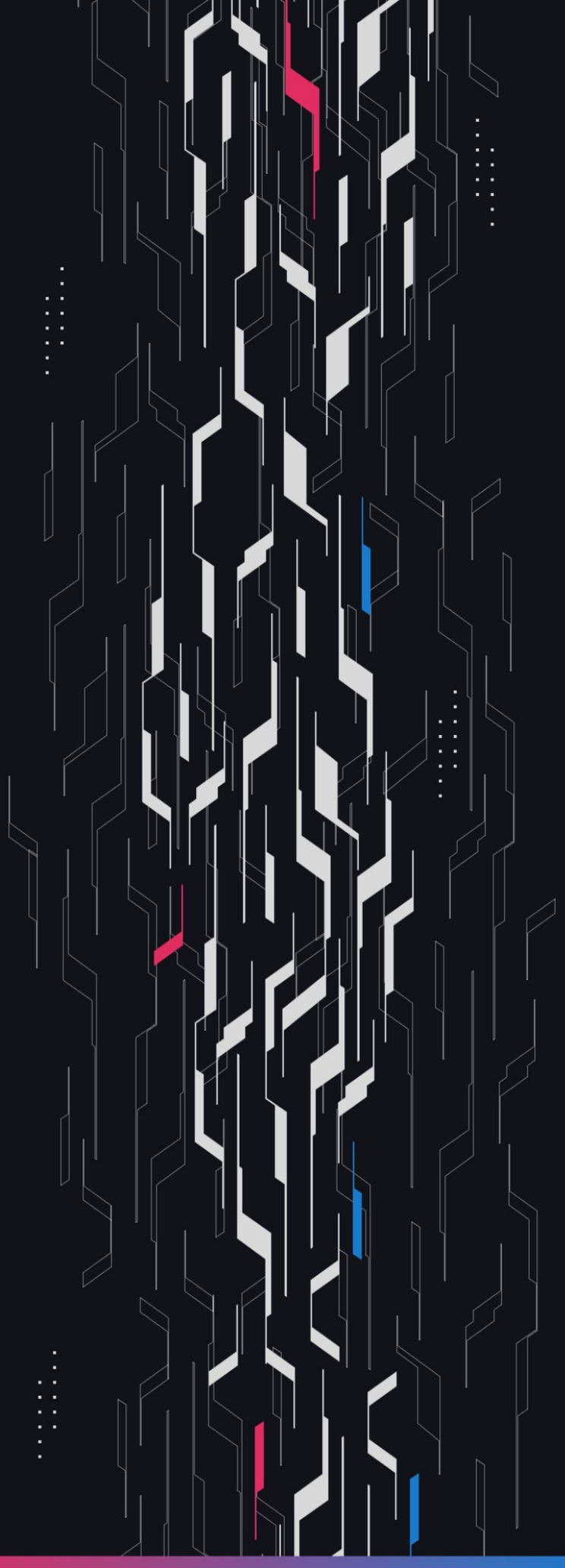
GA GUARDIAN

GMX

OFT Review

Security Assessment

September 19th, 2025



Summary

Audit Firm Guardian

Prepared By Owen Thurm, Robert Reigada, Ciphky

Client Firm GMX

Final Report Date September 19, 2025

Audit Summary

GMX engaged Guardian to review the security of their GMX OFT. From the 14th of August to the 18th of September, a team of 3 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Confidence Ranking

Given the lack of critical issues detected and minimal code changes following the main review, Guardian assigns a Confidence Ranking of 5 to the protocol. Guardian advises the protocol to consider periodic review with future changes. For detailed understanding of the Guardian Confidence Ranking, please see the rubric on the following page.

🔗 Blockchain network: **Crosschain**

✅ Verify the authenticity of this report on Guardian's GitHub: https://github.com/guardianaudits

📊 Code coverage & PoC test suite: https://github.com/GuardianOrg/gmx-token-lz-gmxoft-team1, https://github.com/GuardianOrg/gmx-token-lz-gmxoft-team2

Guardian Confidence Ranking

Confidence Ranking	Definition and Recommendation	Risk Profile
5: Very High Confidence	Codebase is mature, clean, and secure. No High or Critical vulnerabilities were found. Follows modern best practices with high test coverage and thoughtful design. **Recommendation:** Code is highly secure at time of audit. Low risk of latent critical issues.	0 High/Critical findings and few Low/Medium severity findings.
4: High Confidence	Code is clean, well-structured, and adheres to best practices. Only Low or Medium-severity issues were discovered. Design patterns are sound, and test coverage is reasonable. Small changes, such as modifying rounding logic, may introduce new vulnerabilities and should be carefully reviewed. **Recommendation:** Suitable for deployment after remediations; consider periodic review with changes.	0 High/Critical findings. Varied Low/Medium severity findings.
3: Moderate Confidence	Medium-severity and occasional High-severity issues found. Code is functional, but there are concerning areas (e.g., weak modularity, risky patterns). No critical design flaws, though some patterns could lead to issues in edge cases. **Recommendation:** Address issues thoroughly and consider a targeted follow-up audit depending on code changes.	1 High finding and ≥ 3 Medium. Varied Low severity findings.
2: Low Confidence	Code shows frequent emergence of Critical/High vulnerabilities (~2/week). Audit revealed recurring anti-patterns, weak test coverage, or unclear logic. These characteristics suggest a high likelihood of latent issues. **Recommendation:** Post-audit development and a second audit cycle are strongly advised.	2-4 High/Critical findings per engagement week.
1: Very Low Confidence	Code has systemic issues. Multiple High/Critical findings (≥5/week), poor security posture, and design flaws that introduce compounding risks. Safety cannot be assured. **Recommendation:** Halt deployment and seek a comprehensive re-audit after substantial refactoring.	≥5 High/Critical findings and overall systemic flaws.

Table of Contents

Project Overview

Project Summary

Project Name	GMX
Language	Solidity
Codebase	https://github.com/gmx-io/gmx-token-lz
Commit(s)	Initial commit: 15142bb3323a63da06b2f8660fbc0d035cd940bb Final commit: b8c6627e9c1f4bb6128b2de7694a84ba4f50c396

Audit Summary

Delivery Date	September 19, 2025
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
High	0	0	0	0	0	0
Medium	6	0	0	0	0	6
Low	0	0	0	0	0	0
Info	3	0	0	3	0	0

Audit Scope & Methodology

Scope and details:

GMX_Adapter.sol
GMX_MintBurnAdapter.sol
GMX_LockboxAdapter.sol
OverridableInboundRateLimiter.sol
IOverridableInboundRateLimiter.sol
IGMXMinterBurnable.sol
layerzero.config.ts

Audit Scope & Methodology

Vulnerability Classifications

Severity	Impact: *High*	Impact: *Medium*	Impact: *Low*
Likelihood: *High*	🔴 Critical	🟠 High	🟡 Medium
Likelihood: *Medium*	🟠 High	🟡 Medium	🟢 Low
Likelihood: *Low*	🟡 Medium	🟢 Low	🟢 Low

Impact

High Significant loss of assets in the protocol, significant harm to a group of users, or a core functionality of the protocol is disrupted.

Medium A small amount of funds can be lost or ancillary functionality of the protocol is affected. The user or protocol may experience reduced or delayed receipt of intended funds.

Low Can lead to any unexpected behavior with some of the protocol's functionalities that is notable but does not meet the criteria for a higher severity.

Likelihood

High The attack is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount gained or the disruption to the protocol.

Medium An attack vector that is only possible in uncommon cases or requires a large amount of capital to exercise relative to the amount gained or the disruption to the protocol.

Low Unlikely to ever occur in production.

Audit Scope & Methodology

Methodology

Guardian is the ultimate standard for Smart Contract security. An engagement with Guardian entails the following:

- Two competing teams of Guardian security researchers performing an independent review.
- A dedicated fuzzing engineer to construct a comprehensive stateful fuzzing suite for the project.
- An engagement lead security researcher coordinating the 2 teams, performing their own analysis, relaying findings to the client, and orchestrating the testing/verification efforts.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts. Comprehensive written tests as a part of a code coverage testing suite.
- Contract fuzzing for increased attack resilience.

Findings & Resolutions

ID	Title	Category	Severity	Status
M-01	Incorrect Enforced Options In Layerzero.config.ts File	Configuration	Medium	Resolved
M-02	SOLANA_V2_TESTNET Instead Of SOLANA_V2_MAINNET	Configuration	Medium	Resolved
M-03	Non-normalized Send Accounting Inflate Rate-limit	Rounding	Medium	Resolved
M-04	Unsafe Layer Zero Configuration	Configuration	Medium	Resolved
M-05	Asymmetric Capacity Accounting	Configuration	Medium	Resolved
I-01	Inbound Rate Limit Is Effectively “Net-Flow”	Warning	Info	Acknowledged
I-02	Rate Limits Are Not Applied Correctly	Validation	Info	Acknowledged

M-01 | Incorrect Enforced Options In Layerzero.config.ts File

Category	Severity	Location	Status
Configuration	Medium	layerzero.config.ts	Resolved

Description

In `layerzero.config.ts` the `enforcedOptions` tuple for some connections does not match the target chain type. The 5th element that `generateConnectionsConfig([...])` expects (per your sample) is ordered as:

```
[ /* Chain B enforcedOptions */, /* Chain A enforcedOptions */ ]
```

However, the current file mixes up Solana/EVM option sets and also uses a misleading comment ("Chain A enforcedOptions, Chain B enforcedOptions"), causing EVM chains to receive `SOLANA_ENFORCED_OPTIONS` and/or Solana to receive `EVM_ENFORCED_OPTIONS`. This miswiring makes the executor run with the wrong units (EVM: gas/wei vs. Solana: compute units/lamports), leading to under-provisioned execution, reverts, or stuck messages.

Examples (current vs. fixed)
Arbitrum (A, EVM) ↔ Avalanche (B, EVM) — must be EVM on both sides. Current (wrong — A gets Solana options):

```
[EVM_ENFORCED_OPTIONS, SOLANA_ENFORCED_OPTIONS]
```

Fixed (both sides EVM):

```
[EVM_ENFORCED_OPTIONS, EVM_ENFORCED_OPTIONS]
```

Solana (A) ↔ Arbitrum (B, EVM) — B must get EVM, A must get Solana. Current (wrong — B gets Solana, A gets EVM):

```
[SOLANA_ENFORCED_OPTIONS, EVM_ENFORCED_OPTIONS]
```

Fixed (match sample order [B, A]):

```
[EVM_ENFORCED_OPTIONS, SOLANA_ENFORCED_OPTIONS]
```

Solana (A) ↔ Avalanche (B, EVM) — already correct if using [B, A] order:

```
[ EVM_ENFORCED_OPTIONS, SOLANA_ENFORCED_OPTIONS ]
```

Reference: https://docs.layerzero.network/v2/developers/evm/technical-reference/simple-config

Recommendation

Correct the tuples to match the sample's order [Chain B, Chain A] and to use Solana options for Solana and EVM options for EVM.

Resolution

GMX Team: The issue was resolved in PR#3.

M-02 | SOLANA_V2_TESTNET Instead Of SOLANA_V2_MAINNET

Category	Severity	Location	Status
Configuration	Medium	layerzero.config.ts	Resolved

Description

In `layerzero.config.ts` the Solana contract entry is wired to testnet while the EVM peers are on mainnet.

This mixes environments inside the same connection matrix and will cause wiring failures, message delivery errors, or accidental testnet minting when operating from mainnet.

Recommendation

Point Solana to mainnet and load the mainnet OFT Store address:

```
const solanaContract: OmniPointHardhat = {
eid: EndpointId.SOLANA_V2_MAINNET, // ✅ mainnet EID
address: getOftStoreAddress(EndpointId.SOLANA_V2_MAINNET), // ✅ mainnet OFT Store
}
```

Resolution

GMX Team: The issue was resolved in PR#3.

M-03 | Non-normalized Send Accounting Inflate Rate-limit

Category	Severity	Location	Status
Rounding	Medium	GMX_Adapter.sol	Resolved

Description PoC

The adapter counts inbound and outbound volume using different units/rounding. On receive, the limiter charges the normalized amount in local denomination (LD) derived from the packet's shared-decimals (SD) payload:

```
super._outflow(_origin.srcEid, _toLD(_message.amountSD()));
```

but on `send` it credits the limiter with the raw `_amountLD`:

```
super._inflow(_dstEid, _amountLD);
```

In OFT, packets carry amounts in SD. Conversions are:

```
_toSD(_amountLD) = floor(_amountLD / 10^(LD−SD))
_toLD(_amountSD) = _amountSD * 10^(LD−SD)
```

Every inbound charge is therefore an exact multiple of the conversion rate, while outbound credits can include a remainder ("dust") < `10^(LD−SD)`.

For example, with `LD=18`, `SD=8`, `cv=10^10`: sending `_amountLD = cv + (cv−1) = 19,999,999,999` transmits only 1 SD, yet the limiter credits `19,999,999,999` on send but charges only `10,000,000,000` per matching receive.

Repeating this with high-dust amounts accumulates extra headroom inside the window, allowing more inbound packets than intended under a net-flow limiter.

Recommendation

Normalize send-side accounting to the same basis used on receive so both directions use identical units and rounding:

```
super._inflow(_dstEid, _toLD(_toSD(_amountLD)));
```

Resolution

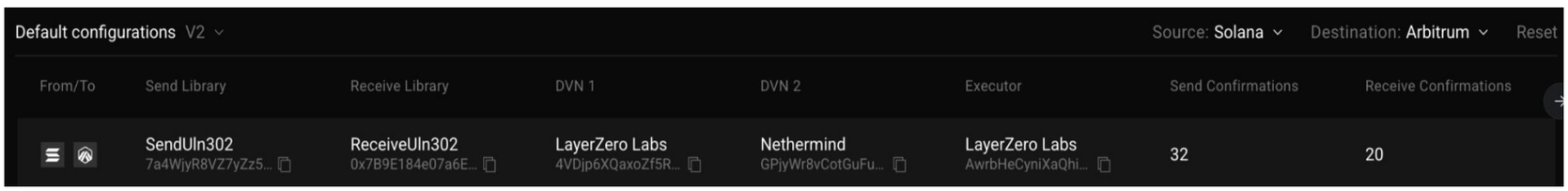
GMX Team: The issue was resolved in PR#2.

M-04 | Unsafe Layer Zero Configuration

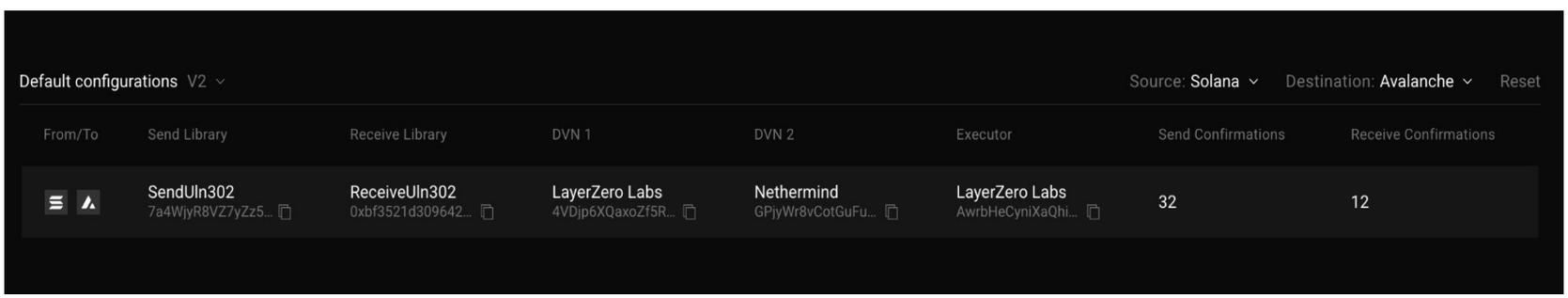
Category	Severity	Location	Status
Configuration	Medium	layerzero.config.ts	Resolved

Description

The current confirmation settings defined in the layerzero.config.ts file for Solana - Arbitrum are 32 for sending and 20 for receiving: [32, 20], // [A to B confirmations, B to A confirmations].
Given the default values used by LayerZero, defined here https://layerzeroscan.com/tools/defaults?version=V2, this is correct:



However for Solana - Avalanche, sending confirmations are set to 10 for sending and 12 for receiving: [10, 12], // [A to B confirmations, B to A confirmations]
The recommended defaults by LayerZero are 32 for sending and 12 for receiving:



Finally, for Arbitrum - Avalanche, sending confirmations are set to 20 while receiving confirmations are set to 12. These are exactly the recommended default values by LayerZero.

Recommendation

Update the Solana - Avalanche configuration and use the LayerZero default values: 32 confirmations for sending and 12 for receiving.

Resolution

GMX Team: The issue was resolved in PR#3.

M-05 | Asymmetric Capacity Accounting

Category	Severity	Location	Status
Configuration	Medium	GMX_Adapter.sol	Resolved

Description

`GMX_Adapter._debit` reduces the rate-limiter's in-flight amount on send unconditionally, even if the `_from` address is part of the `exemptAddresses`:

```
super._inflow(_dstEid, _amountLD);
```

while `_lzReceive` skips increasing in-flight for exempt recipients (and overridable GUIDs):

```
if (exemptAddresses[toAddress] guidOverrides[_guid]) {
super._outflow(_origin.srcEid, _toLD(_message.amountSD()));
}
```

This breaks symmetry. Any transfer whose destination `toAddress` is exempt will not add to in-flight on receive, yet it does decrease in-flight on send.

While it makes sense to not apply the rate limit on certain addresses when they receive tokens, these same addresses should, at the same time, not decrease the rate limit when they send tokens.

Recommendation

Consider updating the `_debit` function to not call `super._inflow` when the `_from` address is part of the `exemptAddresses` mapping in the destination chain.

Resolution

GMX Team: The issue was resolved in PR#4.

I-01 | Inbound Rate Limit Is Effectively “Net-Flow”

Category	Severity	Location	Status
Warning	Info	GMX_Adapter.sol	Acknowledged

Description

GMX_Adapter tries to enforce an inbound throttle by swapping LayerZero’s rate-limiter calls. LayerZero’s limiter uses a single bucket per EID (amountInFlight). _outflow(amount) increases the bucket. _inflow(amount) decreases it but floors at zero (returns headroom).

Because the adapter calls _inflow on sends and _outflow on receives, the bucket measures net = inbound − outbound within the window, not inbound alone. As a consequence an actor can alternate receive→send→receive to keep the bucket ≤ limit while pulling in more than the intended inbound cap.

Concrete example (window limit L = 1000 from chain A; start bucket = 0):
1. Receive 1000 from A → _outflow(+1000) → bucket = 1000
2. Send 1000 to A → _inflow(−1000) floors to 0 → bucket = 0
3. Receive 1000 from A → bucket = 1000
4. Send 1000 to A → bucket = 0
5. Receive 1000 from A → bucket = 1000

Within one window, the adapter received 3000, yet the bucket never exceeded 1000. This defeats an “inbound-only” throttle: total inbound allowed becomes L + outbound_in_same_window.

• If your policy goal is “throttle inbound mint/credit pressure regardless of outbound”, the current scheme fails: an actor can exceed the inbound cap as long as they also send roughly the same amount back out during the window.

• If your policy goal is “limit net bandwidth”, then the behavior is expected, but then it’s not an “inbound cap.”

Recommendation

Enforce a true inbound-only limit by not returning capacity on outbound sends. In practice, remove the _inflow call from _debit and keep the _outflow on _lzReceive only. This uses the limiter as an inbound-only bucket per srcEid.

Resolution

GMX Team: Acknowledged.

I-02 | Rate Limits Are Not Applied Correctly

Category	Severity	Location	Status
Validation	Info	RateLimiter.sol	Acknowledged

Description

The `_inflow()` function is designed to reduce the `amountInFlight` by the `_amount` passed to it (`inflowAmount`), with a lower bound of zero. Despite scenarios involving a total of 1000 units of outflow and 1000 units of inflow processed in the same timestamp, the remaining rate limit differs based on the order of operations.

This inconsistency proves that the rate limiter behaves differently when the sequence of inflow and outflow operations changes, even though the net effect is the same. Users performing the same net actions will experience different rate limits based solely on the sequence of their operations.

Moreover, it is expected that users will deliberately arrange operations to maximize their rate limits. A rate limiter should apply limits consistently, regardless of the order of operations, as long as the net effect is the same.

Recommendation

Instead of simply setting to 0 the `rl.amountInFlight` when `_amount > rl.amountInFlight` consider declaring `rl.amountInFlight` as an int256. This allows the rate limiter to accurately track the net balance of inflows and outflows, ensuring that the rate limit is applied consistently regardless of the order of operations.

By allowing `amountInFlight` to become negative, it is possible to account for scenarios where inflows exceed outflows within the rate limiting window. This approach ensures that the remaining rate limit (`amountCanBeSent`) reflects the true net activity over time, eliminating discrepancies caused by the sequence of inflow and outflow operations.

Resolution

GMX Team: Acknowledged.

Remediation Findings & Resolutions

ID	Title	Category	Severity	Status
M-01	amountReceivedLD Burnt Instead Of amountSentLD	Logical Error	Medium	Resolved
I-01	Lack Of A Double Step Transfer Ownership Pattern	Best Practices	Info	Acknowledged

M-01 | amountReceivedLD Burnt Instead Of amountSentLD

Category	Severity	Location	Status
Logical Error	Medium	GMX_MintBurnAdapter.sol	Resolved

Description

The `GMX_MintBurnAdapter`'s `_debit` implementation burns `amountReceivedLD` even though the OFT core deliberately splits `_debitView`'s output into two values: `amountSentLD` (the amount actually deducted on the source chain) and `amountReceivedLD` (the post-adjustment amount that should be minted/unlocked on the destination).

Right now `_debitView` returns identical numbers for both slots, but the API is designed so custom adapters can diverge them, for example to apply protocol fees, handle fee-on-transfer tokens, tweak dust rounding, or refund rate-limit penalties.

Once such a customization makes `amountSentLD` greater than `amountReceivedLD`, burning only `amountReceivedLD` leaves the excess tokens on the source chain while the remote side still mints the lower amount.

Over time that silently inflates the global supply relative to what is minted elsewhere, breaking the accounting guarantees of the OFT mesh.

Recommendation

Burn `amountSentLD`, in line with the upstream `MintBurnOFTAdapter`, so every unit deducted on the source chain is destroyed regardless of downstream adjustments.

If you want to keep the current behaviour until divergence is formally supported, enforce `require(amountSentLD = amountReceivedLD)` before burning to fail fast instead of letting inflation slip through silently.

Resolution

GMX Team: Resolved.

I-01 | Lack Of A Double Step Transfer Ownership Pattern

Category	Severity	Location	Status
Best Practices	Info	OverridableInboundRateLimiter.sol	Acknowledged

Description

The current ownership transfer process for the `OverridableInboundRateLimiter` which is inheriting from the `Ownable` contract involves the current owner calling the `transferOwnership` function:

```
function transferOwnership(address newOwner) public virtual onlyOwner {
require(newOwner = address(0), "Ownable: new owner is the zero address");
_transferOwnership(newOwner);
}
```

If the nominated EOA account is not a valid account, it is entirely possible that the owner may accidentally transfer ownership to an uncontrolled account, losing the access to all functions with the `onlyOwner` modifier.

Recommendation

It is recommended to implement a two-step process transfer ownership process where the owner nominates an account and the nominated account needs to call an `acceptOwnership` function for the transfer of the ownership to fully succeed.

This ensures the nominated EOA account is a valid and active account. This can be easily achieved by using OpenZeppelin's Ownable2Step contract.

Resolution

GMX Team: Acknowledged.

Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Guardian to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Guardian’s position is that each company and individual are responsible for their own due diligence and continuous security. Guardian’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Guardian is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Guardian

Founded in 2022 by DeFi experts, Guardian is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

To learn more, visit https://guardianaudits.com

To view our audit portfolio, visit https://github.com/guardianaudits

To book an audit, message https://t.me/guardianaudits