GA GUARDIAN

GMX

GMX Crosschain V2.2 6

Security Assessment

July 26th, 2025

Summary

Audit Firm Guardian

Prepared By Owen Thurm, CuriousApple, Wafflemakr, Cosine, Osman Ozdemir, Michael Lett

Client Firm GMX

Final Report Date July 26, 2025

Audit Summary

GMX engaged Guardian to review the security of their GMX Crosschain architecture. From the 3rd of July to the 21st of July, a team of 6 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 4 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: **Arbitrum, Avalanche**

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

📊 Code coverage & PoC test suite: https://github.com/GuardianOrg/gmx-syntheticsgmxcrosschain3-fuzz

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-synthetics/tree/main/contracts
Commit(s)	Initial commit: 7676b7b544ef37b0fbd2446aea932e7c51c7aaa8

Audit Summary

Delivery Date	July 26, 2025
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	1	0	0	0	0	1
High	2	0	0	1	0	1
Medium	3	0	0	2	0	1
Low	18	0	0	12	0	6
Info	0	0	0	0	0	0

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
C-01	Edge Oracle Uses Invalid Decimals	Logical Error	Critical	Resolved
H-01	Imbalanced Impact Caps Cause Unpayable Lent	Logical Error	High	Resolved
H-02	Price Changes Leave Unbacked Lent Amount	Gaming	High	Acknowledged
M-01	Max Impact Configuration Risk	Warning	Medium	Acknowledged
M-02	Subaccount Action Replay	Warning	Medium	Resolved
M-03	Lent Amount Buildup	Warning	Medium	Acknowledged
L-01	Lent Amount Left Due To Rounding	Warning	Low	Acknowledged
L-02	Incorrect srcChainId Emitted	Events	Low	Acknowledged
L-03	Unintended Reverts With Zero Transfer Tokens	DoS	Low	Resolved
L-04	Permits Allowed For Multichain Actions	Warning	Low	Acknowledged
L-05	Price Divergence Allows For A Large Lent Value	Warning	Low	Acknowledged
L-06	Execution Price Inaccuracy	Warning	Low	Acknowledged
L-07	Negative Impact Caps Ignored	Warning	Low	Acknowledged

Findings & Resolutions

ID	Title	Category	Severity	Status
L-08	Impact Pool Distribution Risk	Warning	Low	Acknowledged
L-09	Risk Free Trade Opportunity With Malicious Reverts	Warning	Low	Acknowledged
L-10	userNonce Values Can Be Reused	Warning	Low	Acknowledged
L-11	Nonexistent srcChainId Is Not Validated	Validation	Low	Acknowledged
L-12	DomainSeparator Duplication	Warning	Low	Acknowledged
L-13	Spread Reduction Factor Unused	Warning	Low	Acknowledged
L-14	Assumed Shared Decimals Of 6	Warning	Low	Resolved
L-15	Missing CLAIMABLE_COLLATERAL_DELAY Config	Warning	Low	Resolved
L-16	Unnecessary toBytes32	Superfluous Code	Low	Resolved
L-17	Unexpected Keys Are Settable	Configuration	Low	Resolved
L-18	Missing Reentrancy Guards	Warning	Low	Resolved

C-01 | Edge Oracle Uses Invalid Decimals

Category	Severity	Location	Status
Logical Error	● Critical	EdgeDataStreamProvider.sol	Resolved

Description

The `EdgeDataStreamProvider` contract computes the price decimals in the following way:

```
int256 floatMultiplier = int256(30) + report.expo;
if (floatMultiplier < 0) {
revert Errors.InvalidEdgeDataStreamExpo(report.expo);
}
uint256 adjustedBidPrice = report.bid * 10 ** (uint256(floatMultiplier));
uint256 adjustedAskPrice = report.ask * 10 ** (uint256(floatMultiplier));
```

However, this yields invalid prices which are in 30 decimals, rather than being able to produce a product of 30 decimals when multiplied by the token amount. Using the example response from the Chaos documentation:

```
{
"feedId": "ETHUSD",
"price": 200000000000,
"ts": 1653509453000,
"expo": -8,
"signature": "0x1234...5678",
"recoveryId": 1
}
```

bid: 200000000000
Expo = -8
result = 200000000000 * 1e22 = 2,000e30
However the Ether price in this case should be 2,000e12.

Recommendation

Use a token decimal adjustment in the Edge oracle to correctly compute prices with decimals that allow the token decimals + price decimals to be 30 total decimals for USD units.

Resolution

GMX Team: Resolved.

H-01 | Imbalanced Impact Caps Cause Unpayable Lent

Category	Severity	Location	Status
Logical Error	High	DecreasePositionCollateralUtils.sol	Resolved

Description

Positive impact is capped for each increase and decrease action, using the max positive impact factor. However the max negative impact factor is applied to the summation of the pending impact from increase and the impact realized during decrease.

As a result of this asymmetry in the capping performed on positive vs negative impact there can be more net positive impact paid out than the amount that is paid. Previously this was not an issue as the impact pool capping would prevent impact amount from being lent beyond what can be covered by the open positions.

However with the removal of this capping, the lent amount will remain after all positions are closed and will require payment from the admin to make the market whole and allow positive impact to occur in the future.

Consider the following scenario:
• New market, 0 OI
• Open Long size 100, get -10 units of pending impact
• Open Short size 100, get 10 units of pending impact > Capped to 5 immediately
• Close Long size 100, -10 pending impact + -10 impact realized on decrease > Capped to 100 * 0.05 = -5 negative impact actually paid
• Close Short size 100, 5 pending impact + 10 impact realized on decrease > Decrease impact is capped on it's own > 5 + 5 = 10 total positive impact paid out
• Only 5 negative impact paid, but 10 impact paid out leaving a lent amount which sticks around

Recommendation

Consider capping the pending impact and decrease impact separately to mimic the magnitude of impact that is applied to increase actions.

Resolution

GMX Team: Resolved.

H-02 | Price Changes Leave Unbacked Lent Amount

Category	Severity	Location	Status
Gaming	High	Global	Acknowledged

Description

Now that the positive impact amount realized on decrease is not constrained to the amount in the impact amalgamation, it is possible to realize an unpayable lent amount which remains after all positions close due to price changes.

Consider the scenario where one user opens a position at a higher price and closes it at a lower price (ignore PnL and focus on impact logic):

• Fresh market, 0 OI

• User A opens Long at price $100 and pays -$100 in price impact, e.g. 1 index token as pending impact

• Price goes to $50

• User A closes their Long at price $50 and receives +$100 in price impact, now 2 index tokens

• User A's net impact is 1 index token, creating a lent amount while closing the only position in the market

This unbacked lent amount must be paid by the admin and can build up significantly over time, especially in volatile markets.

Recommendation

Consider reverting to the original price impact capping mechanism which limits the amount of positive impact to what is available in the impact amalgam.

Resolution

GMX Team: Acknowledged.

M-01 | Max Impact Configuration Risk

Category	Severity	Location	Status
Warning	Medium	Global	Acknowledged

Description

Because of the way that price impact calculation is done, there is a risk of leaving an unbacked lent amount after all positions are closed when the max impact factors are updated.

Consider the following scenario:

• Impact factor caps are at 5%

• User A Opens a position size 100 and receives 10 negative impact, which is capped to 5

• Impact factor caps are raised to 10%

• User A Closes their position entirely and receives 10 positive impact, which is now uncapped at 10

• There is a net unpaid lent amount of 5 and no other positions are open

In this scenario the admin will have to cover the impact that is left as lent.

Recommendation

Be aware of this risk and consider this when making max impact configurations.

Resolution

GMX Team: Acknowledged.

M-02 | Subaccount Action Replay

Category	Severity	Location	Status
Warning	Medium	SubaccountRouterUtils.sol	Resolved

Description

Because the `domainSeparator` for sub account actions is based on the `srcChainId`, a signed subaccount approval could be used across Arbitrum and Avalanche if the `MultichainSubaccountRouter` was deployed to the same address.

Furthermore the sub account approval signature can be used across the Multichain router on one chain and the gelato relay router on another chain, with the same `srcChainId` being validated for the signature on both.

Recommendation

Be sure that the router addresses are different across chains, or consider adding the `block.chainid` as a salt to the actions being signed.

Resolution

GMX Team: Resolved.

M-03 | Lent Amount Buildup

Category	Severity	Location	Status
Warning	Medium	Global	Acknowledged

Description

Due to the nature of the price impact exponential calculations and the capping logic performed on decrease, when more than one position is involved there are many cases where there can be an unpaid lentAmount which remains after all positions close. Even if price and price impact factors remain constant and there is no distribution of the impact pool.

The following scenario achieves this by getting one short position to realize a net negative impact that is above the max price impact factor threshold such that it gets capped, while the opposite long position is not capped on it's positive impact.

The main reason the long position is not capped while the short position is capped is because of the exponential nature of the price impact calculations. Whereby closing 75% of a position by size actually incurs >85% of the impact made available by the position's size.

For example:
Only position in a market with size 890,000, closes by 650,000 size.
890,000 diff ^ 1.6 = 3303878065.25 * 1e24/1e30 = 3303.87806525
250,000 diff ^ 1.6 = 433215526.972 * 1e24/1e30 = 433.215526972
> price impact paid = 3303.87806525 - 433.215526972 = 2870.66253828
2870.66253828 / 3303.87806525 = 0.86887665996 86% of the price impact in the market is realized.
Vs.
650,000 / 890,000 = 0.73033707865 73% of the size and therefore proportionalPendingImpact is realized.

Recommendation

If the price impact exponent factor is kept as 1e30 then this behavior does not arise and cause an unbacked lent amount. However due to other previously mentioned factors such as changing price and changing price impact factors it should be expected that an unbacked lent amount may consistently build up and need to be repaid. This should be considered when assigning the maximum lent configurations.

Resolution

GMX Team: Acknowledged.

L-01 | Lent Amount Left Due To Rounding

Category	Severity	Location	Status
Warning	Low	Global	Acknowledged

Description

Throughout the GMX contracts the price impact for negative impact is rounded up and the positive impact is rounded down, as a result in many cases after all positions close the impact pool will be left with a lent amount of a few wei due to this rounding.

This may be unexpected and may need to be paid down by the admin after a significant period of time.

Recommendation

There is no inherent risk to this rounding, and the current rounding direction is in the best interest of the protocol. Simply be aware of this behavior.

Resolution

GMX Team: Acknowledged.

L-02 | Incorrect srcChainId Emitted

Category	Severity	Location	Status
Events	Low	DecreaseOrderUtils.sol	Acknowledged

Description

Throughout the GMX contracts the `srcChainId` of 0 is emitted when a Multichain `transferIn` is initiated by a non-multichain signed transaction.

However in the decrease flows and swap execution flows which are executed on Arbitrum there are several instances where a nonzero `srcChainId` is emitted.

This may be misleading for consumers of the `srcChainId` emitted in the `emitMultichainTransferIn` event.

Recommendation

Consider if a `srcChainId` of 0 should be used in all of the `recordTransferIn` invocations during both swap and decrease order execution.

Resolution

GMX Team: Acknowledged.

L-03 | Unintended Reverts With Zero Transfer Tokens

Category	Severity	Location	Status
DoS	Low	Global	Resolved

Description

During the regular withdrawal and GLV withdrawal flows the secondary output token from a swap will remain nonzero even when the secondary token output amount is zero.

As a result, in the `bridgeOutFromController` function execution, this may cause a zero token transfer to occur for the secondary token.

In this case for tokens that revert on zero transfers this may cause an unexpected failure of a bridge out action attached to a deposit or withdrawal.

Recommendation

Be aware of this edge case and consider skipping a `_bridgeOut` invocation if the amount to bridge for either token is zero.

Resolution

GMX Team: Resolved.

L-04 | Permits Allowed For Multichain Actions

Category	Severity	Location	Status
Warning	Low	BaseGelatoRelayRouter.sol	Acknowledged

Description

In the `withRelay` modifier the `_handleTokenPermits` function is invoked whether the router is multi chain or not.

However in the multi chain case there is no way to use the approval which would be made by a permit.

Recommendation

To avoid users making unnecessary approvals and exposing themselves to unnecessary risk by approving the router, the `_handleTokenPermits` function should explicitly early return if it is a multi chain router.

Resolution

GMX Team: Acknowledged.

L-05 | Price Divergence Allows For A Large Lent Value

Category	Severity	Location	Status
Warning	Low	Global	Acknowledged

Description

There are several measures in place to prevent the lent amount from becoming a large portion of the pool value calculation.

However in rare cases where there is large price diversion between the index token and both the long and short tokens.

Even just between the index/long token and the short token, the lent amount value can still become a large portion of the pool value calculation.

Recommendation

Be aware of this risk and monitor the pool lent amount percentage carefully to ensure that this edge case can be corrected by the admin if it were to arise.

Resolution

GMX Team: Acknowledged.

L-06 | Execution Price Inaccuracy

Category	Severity	Location	Status
Warning	Low	DecreasePositionCollateralUtils.sol	Acknowledged

Description

Now that the `totalImpactUsd` is capped in entirety by the max position impact factor on decrease, there is an added case of inaccuracy to the execution price calculation and comparison on decrease.

The comparison assumes that the individual price impact on decrease is capped solely by the max position impact factor once.

Now that it is capped by the max position impact factor in tandem with the `proportionalPendingImpact` the final impact and thus the resulting actual execution price can differ, even if it is not capped by the `capPositiveImpactUsdByPositionImpactPool` function.

Recommendation

Be aware of this additional edge case that causes an execution price calculation discrepancy. It could be more easily corrected than the `capPositiveImpactUsdByPositionImpactPool` inaccuracy edge case by using the min(`maxImpact - proportionalPendingImpact`, `decreaseImpact`). However it is likely better to acknowledge this edge case at this time.

Resolution

GMX Team: Acknowledged.

L-07 | Negative Impact Caps Ignored

Category	Severity	Location	Status
Warning	Low	PositionUtils.sol	Acknowledged

Description

In the `getExecutionPriceForIncrease` and `getExecutionPriceForDecrease` functions there is no accounting for the negative impact capping which will occur during the execution of the decrease order.

This may be unexpected for users, especially on decrease orders, when technically their order could have been executed when accounting for a reduction in negative impact that they receive.

Recommendation

Be aware of this behavior and be sure to document it for users.

Resolution

GMX Team: Acknowledged.

L-08 | Impact Pool Distribution Risk

Category	Severity	Location	Status
Warning	Low	Global	Acknowledged

Description

Since the impact pool distribution reduces the amount readily available in the impact pool to pay out positive impact, this increases the lent amount which is created when users realize positive impact.

Because of this behavior, the impact pool distribution creates a gap of unbacked lent amount which remains after all positions have closed and would need to be covered by the admin.

For example:

• User A opens Long and pays 10 impact into the pool

• Impact pool distributes down to 2

• User A closes and receives 10 positive impact

• lent amount is 8

Recommendation

Be aware of this risk and consider keeping the impact distribution rate set to 0.

Resolution

GMX Team: Acknowledged.

L-09 | Risk Free Trade Opportunity With Malicious Reverts

Category	Severity	Location	Status
Warning	Low	Global	Acknowledged

Description

During the `bridgeOutFromController` flow during deposit actions there is a potentially risky call to `getRevertMessage` in the `_bridgeOut` function. In the event that the `bridgeOutFromController` invocation reverts with maliciously crafted revert data that uses the "Error(string)" selector. The `getRevertMessage` function has a subtle erroneous behavior that can enable risk free trade actions from taking place.

The following assembly block:

```
assembly {
result := add(result, 0x04)
}
```

Corrupts the length field of the result bytes, making the length of the entire `result` object appear very long. Typically an out of bounds pointer panic revert would occur when attempting to decode string from a bytes object where the string purports that its length is longer than that of the bytes object which contains it. However since the length of the entire `result` object has been corrupted, the out of bounds panic revert does not occur.

This enables a malicious actor to force the `getRevertData` function to decode a string which claims to be tens of thousands of bytes long, causing an out of gas error and potentially preventing the keepers from executing the order. An attacker may be able to leverage this behavior to create a risk free swap opportunity with a swap on a deposit/withdrawal action, where they do not allow the keeper to execute the order for a period of time, and then allow the order to be executed when price has moved in their favor.

There is no untrusted external call which could create the malicious revert data necessary to cause this subtle behavior to become an issue. However there are external calls to the configured executor and DVN addresses for the underlying Stargate pool. Furthermore for future cross-chain providers this may pose an issue.

Recommendation

There is no immediate risk at this time, furthermore the `executionGas` calculations and management in `executeDeposit` and `executeWithdrawal` should help to address this vector.

Resolution

GMX Team: Acknowledged.

L-10 | userNonce Values Can Be Reused

Category	Severity	Location	Status
Warning	Low	Global	Acknowledged

Description

Now that the signature validation is performed based on digests, yet using a `userNonce` for uniqueness, the `userNonce` values can be repeated across actions. This may be misleading as nonces are usually intended to be unique in signatures.

Recommendation

Consider if this should be the expected behavior and document it as such for users.

Resolution

GMX Team: Acknowledged.

L-11 | Nonexistent srcChainId Is Not Validated

Category	Severity	Location	Status
Validation	Low	LayerZeroProvider.sol	Acknowledged

Description

In the `_decodeLzComposeMsg` function the `eidToSrcChainId` lookup is used to convert the LZ message's `srcEid` to a `chainId`.

However if the src chain is not supported this `eidToSrcChainId` will return an unexpected 0 chain id. This will simply emit a misleading event and allow the deposit to occur from an unsupported chain. This may also lead to unexpected issues with composed actions.

Recommendation

Consider validating that the resulting `srcChainId` from the `eidToSrcChainId` lookup is not 0 in the `_decodeLzComposeMsg` function.

Resolution

GMX Team: Acknowledged.

L-12 | DomainSeparator Duplication

Category	Severity	Location	Status
Warning	Low	Global	Acknowledged

Description

Given that the `getDomainSeparator` function returns a domain separator based upon what chain the action was signed on, rather than the chain which houses the contract verifying the signature, it would be possible to have the same domain separator value on Arbitrum and Avalanche if a multichain router contract were deployed to the same address on both chains.

Recommendation

There is an extra layer of protection against such a replay in that the actions have a `desChainId` which is included in the signature, however out of an abundance of caution it should be ensured that no Multichain router contracts have the same address across chains.

Resolution

GMX Team: Acknowledged.

L-13 | Spread Reduction Factor Unused

Category	Severity	Location	Status
Warning	Low	EdgeDataStreamProvider.sol	Acknowledged

Description

The `EdgeDataStreamProvider` does not use the `_getDataStreamSpreadReductionFactor` adjustment like the `ChainlinkDataStreamProvider` does.

As a result there may be a non-trivial difference between the result reported from either data stream provider, it may be possible for a malicious actor to arbitrage the difference between these providers, though they cannot directly specify which provider is used to execute their non-atomic actions.

Recommendation

Consider if the `EdgeDataStreamProvider` should be using the `_getDataStreamSpreadReductionFactor` adjustment.

Resolution

GMX Team: Acknowledged.

L-14 | Assumed Shared Decimals Of 6

Category	Severity	Location	Status
Warning	Low	LayerZeroProvider.sol	Resolved

Description

In the `bridgeOut` function the `_removeDust` function assumes that if the token is an 18 decimal token then it uses 6 shared decimals.

This may lead to unexpected outcomes when the token has 18 local decimals but shared decimals not equal to 6.

Recommendation

Ensure that only tokens with 6 shared decimals are supported for bridging with the `LayerZeroProvider`.

Resolution

GMX Team: Resolved.

L-15 | Missing CLAIMABLE_COLLATERAL_DELAY Config

Category	Severity	Location	Status
Warning	Low	Global	Resolved

Description

Configuration for the `CLAIMABLE_COLLATERAL_DELAY` value is missing.

Recommendation

Be sure to configure `CLAIMABLE_COLLATERAL_DELAY` before the contracts are live.

Resolution

GMX Team: Resolved.

L-16 | Unnecessary toBytes32

Category	Severity	Location	Status
Superfluous Code	Low	Cast.sol	Resolved

Description

The overloaded version of the `toBytes32` function that accepts a string input is unused.

Recommendation

Consider removing this version of the `toBytes32` function.

Resolution

GMX Team: Resolved.

L-17 | Unexpected Keys Are Settable

Category	Severity	Location	Status
Configuration	Low	Config.sol	Resolved

Description

In the `Config.sol` file the `MULTICHAIN_BALANCE`, `POSITION_LAST_SRC_CHAIN_ID` and `GMX_DATA_ACTION` keys are whitelisted as an `allowedBaseKey`, however these value should not be directly settable as it pertains to the accounting of the system balances or the key's value is not ever used.

Recommendation

Consider if this is the expected behavior, if not, remove the `MULTICHAIN_BALANCE` and/or `POSITION_LAST_SRC_CHAIN_ID` and `GMX_DATA_ACTION` keys whitelisting from the Config file.

Resolution

GMX Team: Resolved.

L-18 | Missing Reentrancy Guards

Category	Severity	Location	Status
Warning	Low	Global	Resolved

Description

The `executeDepositFromController` and `executeWithdrawalFromController` functions are missing a `nonReentrant` modifier.

Recommendation

Although no re-entrancy path is immediately obvious, out of an abundance of caution a `nonReentrant` modifier should be added to the `executeDepositFromController` and `executeWithdrawalFromController` functions.

This also establishes a safeguard for future use-cases of these functions.

Resolution

GMX Team: Resolved.

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