GUARDIAN

GMX

GMX Crosschain V2.2 5

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 9th of June to the 16th of June, 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, Avalance**

✅ 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: 894723b14b5d3fb944e06a463dd522be0b60b206

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	0	0	0	0	0	0
● High	1	0	0	0	0	1
● Medium	4	0	0	1	0	3
● Low	14	0	0	9	0	5
● 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
H-01	Bridged Withdrawals Fail	Logical Error	● High	Resolved
M-01	Price Impact Factor Gaming	Gaming	● Medium	Acknowledged
M-02	Incorrect Position Key Used	Logical Error	● Medium	Resolved
M-03	Incorrect Gas Utils Function	Logical Error	● Medium	Resolved
M-04	Same Actions Cannot Be Done	Validation	● Medium	Resolved
L-01	Lent Payback Is Not Rounded Up	Rounding	● Low	Resolved
L-02	Config Keeper May Init An Invalid Provider	Unexpected Behaviour	● Low	Resolved
L-03	Provider May Be Immediately Updated	Unexpected Behaviour	● Low	Acknowledged
L-04	Config Keeper Trusted To Set Providers	Trust Assumptions	● Low	Acknowledged
L-05	Lacking Error Validation	Validation	● Low	Resolved
L-06	Bridging Fee Is Paid Twice	Logical Error	● Low	Resolved
L-07	Documentation Regarding Withdrawals	Documentation	● Low	Acknowledged
L-08	Increased Signature Protection	Warning	● Low	Acknowledged

Findings & Resolutions

ID	Title	Category	Severity	Status
L-09	Execution Cost Greater Than Provided Fee	Configuration	● Low	Acknowledged
L-10	Missing Gas Limit Config Params	Configuration	● Low	Acknowledged
L-11	Order Cancels Do Not Update lastSrcChainId	Logical Error	● Low	Acknowledged
L-12	Loss Of Signature Cancelability And Sequentiality	Warning	● Low	Acknowledged
L-13	Unnecessary Reordering Of Transfer Steps	Best Practices	● Low	Acknowledged
L-14	Bridging Out Tokens From GLV Vault	Warning	● Low	Resolved

H-01 | Bridged Withdrawals Fail

Category	Severity	Location	Status
Logical Error	● High	ControllerUtils.sol: 63	Resolved

Description

GM and GLV withdrawals result in two output tokens, and the `bridgeOutFromController` function is called twice to bridge these tokens. Both calls to `bridgeOutFromController` use the same `withdrawal.dataList`.

The `dataList` contains the Stargate provider address, and that provider can only bridge a single specific token, which is `stargate.token()`. Therefore, it is not possible to bridge both `outputToken` and `secondaryOutputToken` using the same `dataList`.

Recommendation

One option to consider is including two different providers in the `dataList` during cross-chain withdrawals.

However, this would require different `dataList` decoding logic for deposits and withdrawals, as deposits require bridging only a single token.

Another option is to enforce `outputToken = secondaryOutputToken` when the user wants to withdraw and bridge out.

Resolution

GMX Team: Resolved.

M-01 | Price Impact Factor Gaming

Category	Severity	Location	Status
Gaming	● Medium	Global	Acknowledged

Description

Because the price impact is no longer capped to the amount available in the impact amalgam, but rather the maximum lent, it is possible that net positive impact is paid out to traders from the market up to the magnitude of the max lent.

This could arise and be potentially forced through a gaming of the price impact factors as they change based upon the market depth.

A sophisticated actor could observe that the market depth off-chain is now lower and open trades before the price impact factors are updated to bank a X amount of negative impact for creating an imbalance of A.

The sophisticated actor could then observe the price impact factors by GMX being updated and realize Y amount of positive impact for closing their position and removing the imbalance of A.

Recommendation

Be aware of this risk, to mitigate its probability the risk oracle price impact factor changes should not be large in magnitude compared to their previous factors. Furthermore be sure to keep the max lent usd values small so that the opportunity is limited.

Resolution

GMX Team: Acknowledged.

M-02 | Incorrect Position Key Used

Category	Severity	Location	Status
Logical Error	● Medium	OrderUtils.sol: 209	Resolved

Description

In the `_updatePositionLastSrcChainId` function the swap path is evaluated to find the collateral token of the position.

However for decrease orders this will produce an inaccurate result as the swap path has nothing to do with the collateral token of the position.

Recommendation

Only evaluate the resulting token from the order swap path for increase orders and use the `initialCollateralToken` for decrease orders.

Resolution

GMX Team: Resolved.

M-03 | Incorrect Gas Utils Function

Category	Severity	Location	Status
Logical Error	● Medium	LayerZeroProvider.sol: 471	Resolved

Description

The `_handleGlvWithdrawal` in `LayerZeroProvider.sol` uses `_validateGasLeft` to validate if the gas left is sufficient to cover the remaining GLV withdrawal creation flow.

However, it uses the `GasUtils` function `estimateCreateGlvDepositGasLimit` instead of the `estimateCreateGlvWithdrawalGasLimit`.

This can lead to incorrect gas validation and potentially causing the known compose censoring issue.

Recommendation

Use the correct `GasUtils` function for the `_handleGlvWithdrawal` flow:

`estimateCreateGlvWithdrawalGasLimit`

Resolution

GMX Team: Resolved.

M-04 | Same Actions Cannot Be Done

Category	Severity	Location	Status
Validation	● Medium	BaseGelatoRelayRouter.sol: 415	Resolved

Description

The nonce was removed from the relayParams to fix a previous issue, and the replay check is now performed based on the digest.

However, since there is no unique identifier, the structHash and the digest will be identical for the exact same actions.

This will result in the same signature being generated, causing the action to fail even if the user legitimately intends to perform it a second time.

Users must change something in the signature if they want to perform the same action with the same parameters.

The easiest option appears to be the deadline, allowing users to repeat the same action by signing with a different deadline value.

Recommendation

Document this behavior for users and inform them about how digests are generated. Alternatively, consider adding a user-provided salt to the relayParams to allow differentiation of actions, even when all other parameters are identical.

Resolution

GMX Team: Resolved.

L-01 | Lent Payback Is Not Rounded Up

Category	Severity	Location	Status
Rounding	● Low	PositionImpactPoolUtils.sol	Resolved

Description

In the `reduceLentAmount` function the `longTokenAmount` and `shortTokenAmount` computed to be paid by the caller is rounded down. Instead this amount should be rounded up to round in the favor of the GM market.

Recommendation

Consider rounding the `longTokenAmount` and `shortTokenAmount` values up to round in favor of the GM market.

Resolution

GMX Team: Resolved.

L-02 | Config Keeper May Init An Invalid Provider

Category	Severity	Location	Status
Unexpected Behavior	● Low	Config.sol: 59	Resolved

Description

In the initOracleProviderForToken function there is no validation that the provider is enabled with the isOracleProviderEnabledKey.

Furthermore there is no validation that the token is not address(0).

Recommendation

Consider validating that the provider is enabled with the isOracleProviderEnabledKey and that the token is not address(0).

Resolution

GMX Team: Resolved.

L-03 | Provider May Be Immediately Updated

Category	Severity	Location	Status
Unexpected Behavior	● Low	Config.sol	Acknowledged

Description

In the initOracleProviderForToken function the oracleProviderUpdatedAt value is not assigned, therefore the config keeper may immediately invoke the setOracleProviderForToken function to configure a new provider.

Recommendation

Consider updating the oracleProviderUpdatedAt value in the initOracleProviderForToken function.

Resolution

GMX Team: Acknowledged.

L-04 | Config Keeper Trusted To Set Providers

Category	Severity	Location	Status
Trust Assumptions	● Low	Config.sol	Acknowledged

Description

In the `setOracleProviderForToken` function the config keeper is able to reset the oracle provider for a token that is currently configured.

As a result a compromised Config Keeper may assign a supported provider that is incompatible with the specified token in order to DoS trades for a period of time and potentially carry out risk free trades or avoid liquidation.

It may also be possible to use a feed which reports an inaccurate price for the token in order to game the exchange.

Recommendation

Consider if the Config Keeper should be trusted to set the provider for an already configured token. If not, then consider only allowing the oracle provider for an already configured token to be configured through the time lock.

Resolution

GMX Team: Acknowledged.

L-05 | Lacking Error Validation

Category	Severity	Location	Status
Validation	● Low	RelayUtils.sol	Resolved

Description

In the validateSignature function, during the second attempt at signature verification the error returned by the tryRecover function is not validated to be the ECDSA.RecoverError.NoError result.

Recommendation

Out of an abundance of caution, consider validating that the error from the second tryRecover invocation is the ECDSA.RecoverError.NoError result.

Resolution

GMX Team: Resolved.

L-06 | Bridging Fee Is Paid Twice

Category	Severity	Location	Status
Logical Error	● Low	GlvWithdrawalUtils.sol, ExecuteWithdrawalUtils.sol	Resolved

Description

Two cross-chain bridging transactions occur during cross-chain withdrawals, regardless of the output tokens, and both transactions incur a bridging fee.

However, if `outputToken = secondaryOutputToken`, there's no need to bridge twice. Instead, the total output amount can be bridged in a single transaction, avoiding double fees.

Recommendation

Bridge `outputAmount + secondaryOutputAmount` in a single transaction if the output tokens are the same.

Resolution

GMX Team: Resolved.

L-07 | Documentation Regarding Withdrawals

Category	Severity	Location	Status
Documentation	● Low	GlvWithdrawalUtils.sol, ExecuteWithdrawalUtils.sol	Acknowledged

Description

Cross-chain withdrawals attempt to bridge both `outputToken` and `secondaryOutputToken`. However, there is no guarantee that these tokens are supported by Stargate, as the Stargate currently supports only a limited set of major tokens.

As a result, users attempting to bridge out the resulting output tokens may encounter unexpected failures.

Recommendation

Document which output tokens are supported by Stargate for bridging, and clearly inform users of these limitations.

Resolution

GMX Team: Acknowledged.

L-08 | Increased Signature Protection

Category	Severity	Location	Status
Warning	● Low	RelayUtils.sol	Acknowledged

Description

In the `validateSignature` function the minified digest is not marked as used, based on the idea that the original digest being marked as used is sufficient.

However there is some nonzero chance that a malicious actor is able to find a collision with the minified digest that allows the same signature to be submitted and successfully authenticated on behalf of the victim.

Recommendation

While the likelihood of this being a viable attack vector is extremely small, out of an abundance of caution the `minifiedDigest` which was signed should be marked as used as well.

This way if a collision were to ever occur it would simply prevent a user from making a certain signature, versus allowing for an unexpected action to be carried out on behalf of the user.

Resolution

GMX Team: Acknowledged.

L-09 | Execution Cost Greater Than Provided Fee

Category	Severity	Location	Status
Configuration	● Low	general.ts	Acknowledged

Description

During withdrawal creation, the handler verifies if the wnt tokens sent by the user are enough to cover the estimated gas spent during execution.

For withdrawals, this estimation uses the an estimated gas limit of 1_500_000 set in the config, plus some base amount and adjustments.

However, the withdrawal execution now includes two optional bridge out flows, that will break the gas limit estimation. Therefore, keepers will need to spend more gas than the execution fee supplied by the user, so they will prefer not to execute these.

The same applies to the glv withdrawal flow. Additionally, this may pose a notable issue on chains with lower block gas limits such as avalanche, in some cases opening up opportunities for risk free trades.

Recommendation

During our internal testing, the gas spent to execute a withdrawal with bridge out flows is above 2_700_000. We advice to conduct some testing with different params and increase both the withdrawalGasLimit and glvWithdrawalGasLimit to a more suitable value.

Resolution

GMX Team: Acknowledged.

L-10 | Missing Gas Limit Config Params

Category	Severity	Location	Status
Configuration	● Low	general.ts	Acknowledged

Description

The LayerZeroProvider now allows users to initiate a GM or GLV withdrawal order creation. These new flows also include the _validateGasLeft to prevent a previous issue regarding lzCompose censoring.

However, neither CREATE_WITHDRAWAL_GAS_LIMIT or CREATE_GLV_DEPOSIT_GAS_LIMIT are set in general config, so these values are 0 in the dataStore.

This opens up the risk of an actor invoking the lzCompose function, providing insufficient amount of gas and forcing the withdrawal transaction to fail.

Recommendation

Add the missing keys to the general config and be sure to include them in the updateGeneralConfigUtils script.

Resolution

GMX Team: Acknowledged.

L-11 | Order Cancels Do Not Update lastSrcChainId

Category	Severity	Location	Status
Logical Error	● Low	Global	Acknowledged

Description

The lastSrcChainId is updated on order creation, however when an order is canceled it is not considered for a lastSrcChainId update.

Recommendation

Include _updatePositionLastSrcChainId at the end of the cancel order flow.

Resolution

GMX Team: Acknowledged.

L-12 | Loss Of Signature Cancelability And Sequentiality

Category	Severity	Location	Status
Warning	● Low	BaseGelatoRelayRouter.sol: 411-412	Acknowledged

Description

As a fix for using nonces in non-atomic actions (crosschain), GMX switched to using digests to protect against replays.

However, this approach is applied globally across all actions, not just cross-chain ones. This comes with two implications:

1. Users can no longer cancel a signature by issuing a new one with the same nonce—unlike standard EVM-style signature workflows. A signed message remains valid until its deadline.

2. At time T=0, if a user signs multiple actions, they are no longer processed sequentially, which could be important in some trade setups.

Recommendation

If these trade-offs are acceptable, consider enforcing short deadlines close to the current timestamp. Otherwise, consider separating cross-chain non-atomic logic from the rest of the gasless/multichain action set.

Resolution

GMX Team: Acknowledged.

L-13 | Unnecessary Reordering Of Transfer Steps

Category	Severity	Location	Status
Best Practices	● Low	MultichainGlvRouter.sol: 49-50	Acknowledged

Description

To support multichain fee payments for atomic actions from lzCompose, GMX reordered _processTransferRequests to come before the WNT transfer.

However, this is no longer necessary, as the early return for excluded addresses in handleRelayFee was removed. This allows users to send tokens directly to the router when needed.

Recommendation

While there is no harm in the current order as well, we just want to make GMX aware that they can use handleRelayFee to transfer tokens to router similar to all gasless actions, and don’t have to use transfer requests specifically.

Resolution

GMX Team: Acknowledged.

L-14 | Bridging Out Tokens From GLV Vault

Category	Severity	Location	Status
Warning	● Low	GlvWithdrawalUtils.sol: 283	Resolved

Description

User may optionally bridge out tokens after withdrawing from GLV. However, the `executeGlvWithdrawal` uses the `srcChainId`, `receiver` and `dataList` from the `glvWithdrawal` struct.

If `receiver = glvWithdrawal.glv()`, `srcChainId = 0` and `dataList` contains the bridge action data, this will start a bridge out using the `glv` as the account.

Although user will lose all of its funds plus a previous `wnt` deposit to pay for the bridge fee, this is an unexpected scenario that might become significant if the `glv` address will eventually have non-zero multichain balance.

Recommendation

Consider passing an empty `dataList` array and `srcChainId = 0`, just like it's done in the `executeGlvDeposit` flow.

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