GUARDIAN

GMX

Gasless Sponsored Calls - 2

Security Assessment

April 11th, 2025

Summary

Audit Firm Guardian

Prepared By Curiousapple, Wafflemakr, Cosine, Osman Ozedemir, Mark Jonathas, Michael Lett

Client Firm GMX

Final Report Date April 11, 2025

Audit Summary

GMX engaged Guardian to review the security of GMX's Gelato Sponsored Call Integration. From the 31st of March to the 4th of April, a team of 6 auditors reviewed the source code in scope.
All findings have been recorded in the following report.

For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

🔗 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-synthetics-team1, https://github.com/GuardianOrg/gmx-synthetics-team2, https://github.com/GuardianOrg/gmx-synthetics-fuzz

Table of Contents

Project Information

Smart Contract Risk Assessment

Addendum

Project Overview

Project Summary

Project Name	GMX
Language	Solidity
Codebase	https://github.com/gmx-io/gmx-synthetics (PR #139 : Gelato sponsored call)
Commit(s)	Initial commit: d3da8d10ab45f7cb5ac87011e7e946c1b627e7bb Final commit: 50e97983ccbfc8a8849c9eaf9d7eb319caf164d3

Audit Summary

Delivery Date	April 11, 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	1	0	0	1	0	0
● Low	11	0	0	5	0	6

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.

Invariants Assessed

During Guardian's review of GMX, fuzz-testing was performed on the protocol's main functionalities. Given the dynamic interactions and the potential for unforeseen edge cases in the protocol, fuzz-testing was imperative to verify the integrity of several system invariants.

Throughout the engagement the following invariants were assessed for a total of 10,000,000+ runs with a prepared fuzzing suite.

ID	Description	Tested	Passed	Remediation	Run Count
INC-01	Position size in USD should increase after successful increase position call.	✅	✅	✅	10M+
INC-02	Long Open Interest should increase after successful increase position call.	✅	✅	✅	10M+
INC-03	Collateral amount of position should increase after successful increase position call.	✅	✅	✅	10M+
INC-04	Collateral sum for longs should increase after successful increase position call.	✅	✅	✅	10M+
INCI-04	Collateral sum for shorts should increase after successful increase position call.	✅	✅	✅	10M+
DEC-01	Position size in USD should decrease after successful decrease position call.	✅	✅	✅	10M+
DEC-02	Collateral amount of position should decrease after successful decrease position call.	✅	✅	✅	10M+
DEC-03	Long Open Interest should decrease after successful decrease position call.	✅	✅	✅	10M+
DEC-04	Collateral sum for longs should decrease after successful decrease position call.	✅	✅	✅	10M+

Invariants Assessed

ID	Description	Tested	Passed	Remediation	Run Count
CLOSE-01	Position size in USD should be 0 after closing the position.	✅	✅	✅	10M+
CLOSE-02	Position size in tokens should be 0 after closing the position.	✅	✅	✅	10M+
CLOSE-03	Position collateral amount should be 0 after closing the position.	✅	✅	✅	10M+
CLOSE-04	Auto cancel order list should be empty after closing the position.	✅	✅	✅	10M+
CNCL-ORD-1	User should receive the same amount of long tokens he sent to create an order	✅	✅	✅	10M+
CNCL-ORD-2	User should receive the same amount of short tokens he sent to create an order	✅	✅	✅	10M+
SWP-01	Received token balance after swap should be equal to simulated amounts before swap	✅	✅	✅	10M+
GEN-1	Function call should not silently revert	✅	✅	✅	10M+
GEN-2	Fee should be covered and refund sent to the callback contract	✅	✅	✅	10M+
UPDT-01	RelayFeeAddress WNT increase should match gas spent	✅	✅	✅	10M+

Findings & Resolutions

ID	Title	Category	Severity	Status
M-01	Collateral Lost When Combining Swaps	Logical Error	● Medium	Acknowledged
L-01	Return Handling For Order Creation	Best Practices	● Low	Resolved
L-02	minPrice Used For maxRelayFeeSwapUsd	Logical Error	● Low	Resolved
L-03	Batch Orders Do Not Work As Expected	Logical Error	● Low	Acknowledged
L-04	Modifier Execution Order	Logical Error	● Low	Resolved
L-05	Fixed _getCalldataGas Parameters	Best Practices	● Low	Acknowledged
L-06	Over/Under Estimation	Logical Error	● Low	Resolved
L-07	Superfluous Return Statement	Best Practices	● Low	Resolved
L-08	Incorrect Comment: GelatoRelayRouter	Best Practices	● Low	Resolved
L-09	EIP-712 Signature Readability	Best Practices	● Low	Acknowledged
L-10	Subaccounts: Gas Griefing Risk	Trust Assumptions	● Low	Acknowledged
L-11	Less Premium Is Charged In Config	Logical Error	● Low	Acknowledged

M-01 | Collateral Lost When Combining Swaps

Category	Severity	Location	Status
Logical Error	● Medium	BaseGelatoRelayRouter.sol: 293	Acknowledged

Description PoC

The current implementation allows both external calls and internal swaps to be executed in the same transaction. Users can externally swap tokens and send them to orderVault as collateral, and later use atomic swaps to receive WNT in order to pay the execution and relay fees.

However, the unrecorded collateral in the orderVault is at risk if the atomic swap uses the same collateral for tokenIn. The following scenario could occur:

- User swaps ARB for USDC in external calls, sends USDC to orderVault.
- User swaps USDC to WNT using atomic swap, sending USDC to orderVault, and later executing SwapUtils.swap
- Now createOrder is triggered, and collateral is recorded with recordTransferIn
- cache.initialCollateralDeltaAmount will be 0, and user lost the USDC collateral

The issue relies on the fact that orderVault is a StrictBank, so when SwapUtils.swap calls params.bank.transferOut it also triggers _afterTransferOut, syncing the tokenBalances with the current bank balance (which includes the user's collateral).

Recommendation

Prevent users from combining external calls and internal atomic swaps within the same transaction, either through the UI or by implementing on-chain protections.

Resolution

GMX Team: Acknowledged.

L-01 | Return Handling For Order Creation

Category	Severity	Location	Status
Best Practices	● Low	GelatoRelayRouter.sol, BaseGelatoRelayRouter.sol	Resolved

Description

When creating an order using a standalone call, the newly created order's key is returned, which can be utilized for future updates or cancellations. However, when an order is created as part of a batch process, its key is neither validated nor returned during the execution of the batch.

Recommendation

Consider implementing a bytes array as the return value for the batch function. This array should include the newly generated keys for any `_createOrder` calls within the batch. If the batch contains only updates or cancellations, the function should return an empty array.

Resolution

GMX Team: Resolved.

L-02 | minPrice Used For maxRelayFeeSwapUsd

Category	Severity	Location	Status
Logical Error	● Low	BaseGelatoRelayRouter.sol: 283	Resolved

Description

The maximum relay fee swap size for sub accounts is capped, and execution reverts if the calculated USD amount exceeds maxRelayFeeSwapUsd.

However, the calculation for this check is currently performed using the min oracle price instead of the max oracle price.

To enhance safety and prevent potentially underestimating the swap size, it would be preferable to use the max oracle price during this calculation.

Recommendation

Consider using the max oracle price for the maxRelayFeeSwapUsd check to provide greater safety and ensure the cap is accurately enforced.

Resolution

GMX Team: Resolved.

L-03 | Batch Orders Do Not Work As Expected

Category	Severity	Location	Status
Logical Error	● Low	BaseGelatoRelayRouter.sol: 111C1-113C10	Acknowledged

Description

With the newly introduced batch functionality, users can now perform multiple actions in a single operation. GMX also intends to enable users to transfer the required tokens to the orderVault via external calls during the _handleRelayBeforeAction.

```
// External calls can be used to send tokens to OrderVault. In this case, initialCollateralDeltaAmount can be zero,
// and there is no need to call _sendTokens.
```

However, an issue arises when combining the 'batch orders' feature with 'token transfers via external calls' in the following scenario:

Consider a user batching two createOrders actions, each requiring 100 USDC as collateral. The total of 200 USDC will be transferred to the orderVault through external calls, after which the batch function will iterate _createOrder twice.

Since the unaccounted balance of the orderVault is treated as the initialCollateralDeltaAmount (as seen in the code here), the first order will receive the entire 200 USDC as collateral.

The second order, however, will receive 0 USDC, as the entire balance of the orderVault is allocated to the first order during its iteration.

Recommendation

Consider disabling token transfers via external calls for batch orders. Instead, enforce the transfer of tokens directly within the _createOrder function by specifying a non-zero initialCollateralDeltaAmount whenever a batch is used.

This restriction could be implemented at the contract level and also managed in the UI when constructing the batch for the user, ensuring smooth operation and preventing potential issues.

Resolution

GMX Team: Acknowledged.

L-04 | Modifier Execution Order

Category	Severity	Location	Status
Logical Error	● Low	GelatoRelayRouter, SubaccountGelatoRelayRouter.sol	Resolved

Description

Solidity serializes modifiers linearly, meaning the order of execution depends on how they are arranged.

For example, in the function signature: external withRelay(relayParams, account, false) nonReentrant returns (bytes32).
The withRelay modifier will be executed first, followed by nonReentrant.
Since nonReentrant is checked after _handleRelayBeforeAction and cleared before _handleRelayAfterAction, reentrancy is theoretically possible through both of these functions.
However, we couldn't identify a incentive for a normal user to sign off on actions that would enable such behavior.

Theoretically, a sub account might attempt reentrancy via the permit mechanism in _handleTokenPermits. Even so, the only effect would be an increase in gasUsed, resulting in the user paying more gas than necessary—without gaining any real advantage.

Additionally, _handleRelayAfterAction interacts with wnt tokens and hence doesn't allow arbitrary operations. That said, just because we haven't found a concrete exploit path today doesn't mean one won't emerge in the future.

Recommendation

Consider placing the nonReentrant modifier before withRelay for all relevant actions.

Resolution

GMX Team: Resolved.

L-05 | Fixed _getCalldataGas Parameters

Category	Severity	Location	Status
Best Practices	● Low	GasUtils.sol: 600-601	Acknowledged

Description

The `_getCalldataGas` function uses hardcoded parameters that cannot be changed after deployment. Specifically:

- `calldataLengthLimit` is fixed: `if (calldataLength > 50000) { ... }`
- `calldataCostPerByte` is also hardcoded: `uint256 txCalldataGasUsed = calldataLength * 10;`

This lack of configurability limits the protocol's ability to adapt to future changes in gas costs or desired behavior.

Recommendation

Consider making these parameters configurable via the datastore, so they can be adjusted post-deployment if necessary.

Resolution

GMX Team: Acknowledged.

L-06 | Over/Under Estimation

Category	Severity	Location	Status
Logical Error	● Low	GasUtils.sol: 559-560	Resolved

Description

To verify that there's minimal delta between what GMX is charged by Gelato and what users refund back to GMX in the case of a `sponsoredCall`, we fuzzed the following invariant: UPDT-01: WNT increase should match gas spent

- Invariant location
- Setup logic
- Foundry test case

Example observation

```
Charged:      385,982
Actual:       357,827
Delta:        28,155 (~7%)
```

We observed a delta in the range of +7–8%.
Initially, we attributed part of this to overestimation of calldata bytes (e.g., counting 0-bytes with 10 as a cost), which accounted for 14,562 units in our test case.

Subtracting this gives: 28,155 - 14,562 = 14,593

Later, we realized that:

- The base gas cost includes Gelato contract overhead (e.g., verification and external call)
- Also includes intrinsic Ethereum transaction cost
- However, in our fuzzing setup, we call updateOrder directly and measure gas around the external call, so these should be excluded

Subtracting both: 14,593 - 21,000 - 10,000 = -16,407

This shows that some overestimations (e.g., calldata cost) and underestimations (e.g., base cost) offset each other, making the final delta difficult to evaluate reliably in a test environment.

Recommendation

- Review the base gas cost configuration and consider increasing it — it is currently set to 40,000, which may not be sufficient to account for two WNT transfers, as intended. One WNT transfer will always occur.
- Deploy the the gasless routers to a testnet or in a trial phase on mainnet.
- Let a set of on-chain transactions execute under real parameters.
- Analyze the actual delta between what is charged and what is refunded, and adjust the estimation logic accordingly.

Given a list of on-chain transactions, we can assist in perfecting these estimations. On-chain measurements provide more accurate data compared to local test environments (like Foundry or Hardhat), which rely on assumptions that may not reflect production behavior.

Resolution

GMX Team: Resolved.

L-07 | Superfluous Return Statement

Category	Severity	Location	Status
Best Practices	● Low	BaseGelatoRelayRouter.sol: 205	Resolved

Description

The _handleRelayBeforeAction function includes a return statement that attempts to return the result of _handleRelayFee(), but neither of these functions has a defined return value

Recommendation

Remove the return statement.

Resolution

GMX Team: Resolved.

L-08 | Incorrect Comment: GelatoRelayRouter

Category	Severity	Location	Status
Best Practices	● Low	GelatoRelayRouter.sol: 26	Resolved

Description

The GelatoRelayRouter.batch function contains the following comment:

```
// @note all params except subaccount should be part of the corresponding struct hash
```

However, the GelatoRelayRouter does not contain any subaccount param.

Recommendation

Modify the comment so it refers to account instead.

Resolution

GMX Team: Resolved.

L-09 | EIP-712 Signature Readability

Category	Severity	Location	Status
Best Practices	● Low	RelayUtils.sol: 211	Acknowledged

Description

When signing transactions with wallets supporting EIP-712 (like MetaMask), users should see a structured, human-readable representation of all critical parameters. However, there are some issues with the current implementation:

• Parameters reduced to bytes32 hashes appear as unreadable hex strings in the signing interface

• Users have no way to verify the content of important parameters like relay fee information or subaccount permissions

• This creates blind spots where users must trust the frontend application completely

Recommendation

Expand all critical parameters into full EIP-712 typed structures

Resolution

GMX Team: Acknowledged.

L-10 | Subaccounts: Gas Griefing Risk

Category	Severity	Location	Status
Trust Assumptions	● Low	BaseGelatoRelayRouter.sol: 254-255	Acknowledged

Description

GMX has implemented various protections to minimize the potential damage that malicious subaccounts could cause to their associated primary accounts.

One such potential vector involves subaccounts deliberately increasing gas usage, leading to higher-than-necessary refunds paid by the main account to GMX.

We previously flagged a similar issue in our earlier review, where we discussed the possibility of padding calldata with empty data to artificially inflate gas usage.

A comparable scenario exists with token permits. Since permits are called on arbitrary tokens, a malicious subaccount could pass a poorly optimized or deliberately gas-heavy token, causing the `handleTokenPermits` logic to consume excessive gas.

While it’s true that the subaccount would be spending its own funds to inflict this kind of griefing on the main account—and would gain no tangible benefit from doing so—we’re raising this again given your decision to address the calldata padding issue in our prior review.

Recommendation

- Either treat this as an accepted trust assumption for subaccounts,

or

- Consider adding guardrails such as:
  - Limiting the gas forwarded for permit calls
  - Restricting the number of permits processed in `handleTokenPermits`

Resolution

GMX Team: Acknowledged.

L-11 | Less Premium Is Charged In Config

Category	Severity	Location	Status
Logical Error	● Low	general.ts: 132	Acknowledged

Description

In the config files, a 6% relay fee premium is applied to the gas cost. However, according to the Gelato docs, a fixed 10% fee premium is charged on both Arbitrum and Avalanche.

Recommendation

Consider setting the premium to 10% if GMX does not have a protocol-specific discount agreement with Gelato.

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 Audits

Founded in 2022 by DeFi experts, Guardian Audits is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian Audits 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