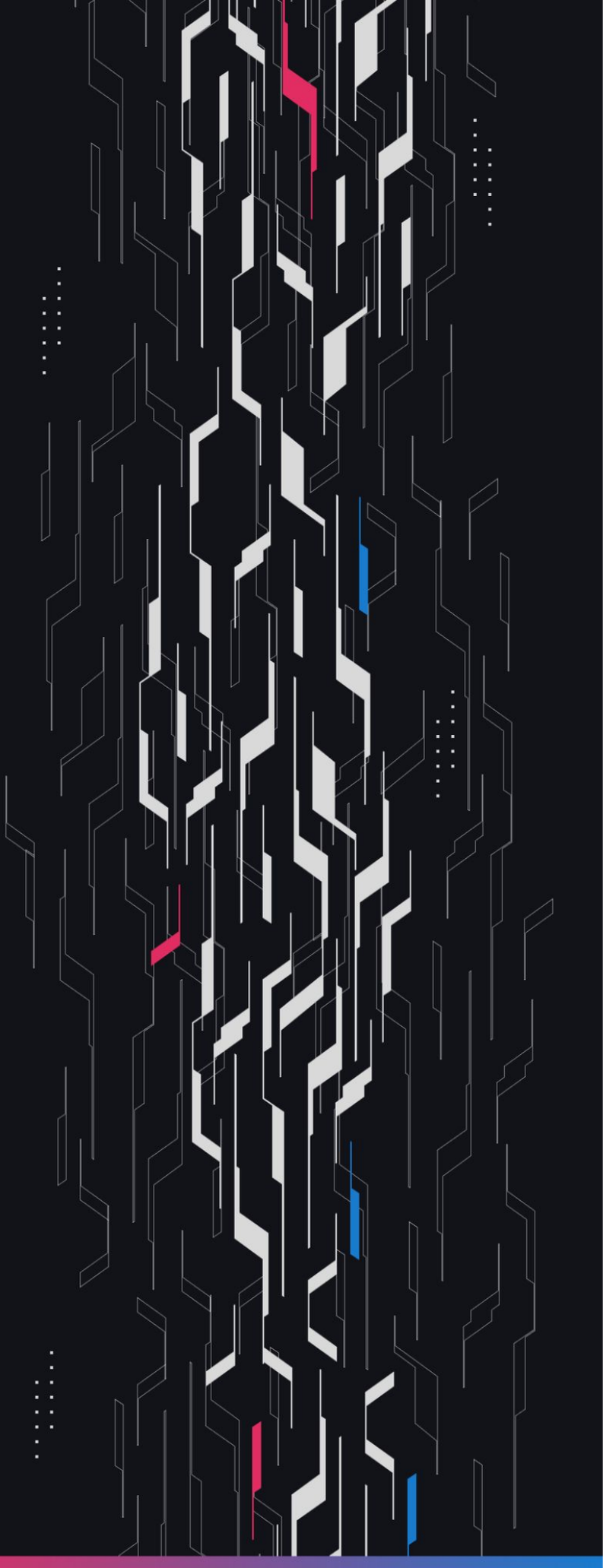
GA GUARDIAN

GMX

Gasless Transactions

Security Assessment

February 27th, 2025



Summary

Audit Firm Guardian

Prepared By Owen Thurm, Daniel Gelfand, Kiki, Wafflemakr, Ladboy

Client Firm GMX

Final Report Date February 27, 2025

Audit Summary

GMX engaged Guardian to review the security of their Gasless transactions support. From the 27th of January to the 3rd of February, a team of 4 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Issues Detected Throughout the engagement 5 High/Critical issues were uncovered and promptly remediated by the GMX team.

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

🔗 Blockchain network: **Arbitrum**

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

Table of Contents

Project Overview

Project Summary

Project Name	GMX
Language	Solidity
Codebase	https://github.com/gmx-io/gmx-synthetics
Commit(s)	Initial commit: 2a77cc76b0ab597578fc0f6640f6e325ca255bb2 Final commit: 5b68fd45f0efe790d6b48ed6fa7d1b35ac3dd862

Audit Summary

Delivery Date	February 27, 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	5	0	0	0	0	5
Medium	4	0	0	2	0	2
Low	18	0	0	10	0	8

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	Atomic Swaps With Normal Fees	Gaming	High	Resolved
H-02	Missing Update Order Fee Logic	Logical Error	High	Resolved
H-03	Missing decreasePositionSwapType Validation	Logical Error	High	Resolved
H-04	Lacking Cancellation Receiver Validation	Validation	High	Resolved
H-05	callbackContract Siphons Refunds	Validation	High	Resolved
M-01	removeSubaccount Cannot Be Gasless	Logical Error	Medium	Resolved
M-02	Unused Permit DoS	DoS	Medium	Resolved
M-03	subAccount Is Not Fully Refunded In Top Up	Logical Error	Medium	Acknowledged
M-04	Swap Occurs Without Market Update	Logical Error	Medium	Acknowledged
L-01	Subaccounts Access To Funds Are Not Partitioned	Unexpected Behavior	Low	Acknowledged
L-02	Comingling Of Gelato And GMX Fee	Logical Error	Low	Acknowledged
L-03	Unused Enum	Superfluous Code	Low	Resolved
L-04	Lacking Feature Validation	Validation	Low	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
L-05	Permit Frontrunning	Frontrunning	Low	Resolved
L-06	Bypassing Subaccount Disabled Feature	Logical Error	Low	Acknowledged
L-07	Missing Swap Path Validation	Validation	Low	Resolved
L-08	Missing MaxFee Validation For Relayer	Validation	Low	Resolved
L-09	removeSubaccount Marked As Payable	Modifiers	Low	Resolved
L-10	Duplicated orderKey Events	Unexpected Behavior	Low	Resolved
L-11	Nonce Dependence Prevents Subsequent Actions	DoS	Low	Acknowledged
L-12	Redundant Required Fee Check	Superfluous Code	Low	Acknowledged
L-13	Missing validateMarketTokenBalance Validation	Validation	Low	Acknowledged
L-14	Tokens Which Do Not Support Permit	Validation	Low	Acknowledged
L-15	Unnecessary Wnt Transfers To/From OrderVault	Superfluous Code	Low	Resolved
L-16	Account Can Steal executionFee From subAccount	Warning	Low	Acknowledged
L-17	Account Can Avoid Refunding Subaccount	Logical Error	Low	Acknowledged

Findings & Resolutions

ID	Title	Category	Severity	Status
L-18	SubAccount Can Make Account Lose Funds	Warning	Low	Acknowledged

H-01 | Atomic Swaps With Normal Fees

Category	Severity	Location	Status
Gaming	High	BaseGelatoRelayRouter.sol: 202	Resolved

Description

The introduction of fee swaps to pay the executor allows for users to do arbitrarily large swaps in a single transaction without first initiating this action on chain.

Before the introduction of gasless transactions this ability was not allowed on the GMX Exchange and it was explicitly disallowed for atomic withdrawals to include swaps.

Gasless transactions unintentionally introduces a way for users to achieve single transaction swaps by using a higher than needed fee swap input amount and receiving the excess amount back.

Furthermore, the swap made in this instance uses the normal swap fees as computed by the `SwapUtils.swap` function. There is no application of the `AtomicSwap` fee pricing.

Recommendation

Consider removing the fee swap feature so that the ability for users to perform swaps in a single transaction is not unintentionally introduced with this update.

Otherwise, keep the feature and limit the size of swap that is allowed to be carried out from this, and furthermore charge the higher `AtomicSwap` fee rate for these swaps.

Resolution

GMX Team: Resolved.

H-02 | Missing Update Order Fee Logic

Category	Severity	Location	Status
Logical Error	High	BaseGelatoRelayRouter.sol: 106	Resolved

Description

In the `_updateOrder` function there is no logic to support sending wnt tokens to the order vault to cover any fee increase related to the update of an order.

This means that any order updates which result in a higher `executionFee` being required will not be possible through gasless transactions.

Recommendation

Include logic to transfer additional `executionFee` wnt value to the `OrderVault` to be able to successfully update orders which would have an increased `executionFee`.

Resolution

GMX Team: Resolved.

H-03 | Missing decreasePositionSwapType Validation

Category	Severity	Location	Status
Logical Error	High	GelatoRelayRouter.sol: 134	Resolved

Description

In the `_getCreateOrderStructHash` the `decreasePositionSwapType` value is not included in the resulting hash and therefore cannot be validated.

The caller of the `createOrder` function can then decide the value of the `_getCreateOrderStructHash` on the params.

This can be used to cause loss or unexpected outcomes for the user as their position and integrating addresses may not have support for a particular receiving token.

Recommendation

Include the `params.decreasePositionSwapType` in the `_getCreateOrderStructHash` function.

Resolution

GMX Team: Resolved.

H-04 | Lacking Cancellation Receiver Validation

Category	Severity	Location	Status
Validation	High	SubaccountRouter.sol: 96	Resolved

Description

The `SubaccountRouter` contract lacks validation for the cancellation receiver in the `createOrder` function.

This way sub accounts can potentially create orders with a large `initialCollateralDeltaAmount` on them and cancel them with the `cancellationReceiver` as their own address to drain the main account's assets.

Recommendation

Consider adding the `cancellationReceiver` validation to the `createOrder` function in the `SubaccountRouter` contract.

Resolution

GMX Team: Resolved.

H-05 | callbackContract Siphons Refunds

Category	Severity	Location	Status
Validation	High	SubaccountGelatoRelayRouter.sol: 79	Resolved

Description

In the `createOrder` function the `receiver` and `cancellationReceiver` addresses are validated against the account to ensure that funds are not sent to an address that is not controlled by the position owner.

However there is no validation on the callback contract which will take a higher priority than the `cancellationReceiver` or `receiver` when the `executionFee` is refunded.

Recommendation

If it is important for the protocol to ensure that sub accounts cannot siphon the refund fee this way, consider validating that the provided callback contract is the saved callback contract for the account and market.

Otherwise be aware of this issue and clearly document it for users and integrators of the sub account feature.

Resolution

GMX Team: Resolved.

M-01 | removeSubaccount Cannot Be Gasless

Category	Severity	Location	Status
Logical Error	Medium	SubaccountGelatoRelayRouter.sol: 141	Resolved

Description

The `removeSubaccount` function is not outfitted to be a gasless transaction function. The function is missing the `_handleRelay` logic and does not have a `onlyGelatoRelay` modifier.

Recommendation

Outfit the `removeSubaccount` function so that it can be used by the Gelato relayer in a gasless manner.

Resolution

GMX Team: Resolved.

M-02 | Unused Permit DoS

Category	Severity	Location	Status
DoS	Medium	BaseGelatoRelayRouter.sol: 271	Resolved

Description

In the `_handleTokenPermits` function if the existing allowance for the spender is greater than the required amount then the permit signature is not used.

However this allows for a malicious DoS in the future because the permit signature is now public and the corresponding nonce has not been used.

The next time the same signer wishes to use the corresponding nonce for a permit to the same token a malicious actor can submit the permit they signed in the past to use the corresponding nonce before the user's newly signed permit is executed.

This can be carried out for future interactions with the GMX system or any other permit application for the token.

Recommendation

Consider always using the permit signature no matter if it is unnecessary and instead determine if the permit is necessary at the UI level.

Resolution

GMX Team: Resolved.

M-03 | subAccount Is Not Fully Refunded In Top Up

Category	Severity	Location	Status
Logical Error	Medium	SubaccountRouter.sol: 222	Acknowledged

Description

When determining how much to top up it the contract calculates the amount of gas used as follows:

`uint256 nativeTokensUsed = (startingGas - gasleft()) * tx.gasprice +  executionFee`

The issue with this is that there is ample logic and gas consumed after this calculation. None of which will be considered for refund. Meaning that a sub account will always get an insufficient refund in terms of the additional top up logic.

Recommendation

Add a `topUpgas` variable to the calculation which contains the amount of gas that will be consumed after the calculation.

Resolution

GMX Team: Acknowledged.

M-04 | Swap Occurs Without Market Update

Category	Severity	Location	Status
Logical Error	Medium	BaseGelatoRelayRouter.sol: 216	Acknowledged

Description

Before any action is executed that a user can take GMX will call `updateFundingAndBorrowingState` to update the borrowing rate of traders in the underlying pool.

However currently a swap through the gelato routers will occur without making any update which means that subsequent borrowing fee rate calculations are incorrect between the period of the swap and the next position update. Causing an incorrect amount of borrowing fees to be charged.

Recommendation

Consider updating the borrowing and funding state with `updateFundingAndBorrowingState` before a swap occurs in a market.

Resolution

GMX Team: Acknowledged.

L-01 | Subaccounts Access To Funds Are Not Partitioned

Category	Severity	Location	Status
Unexpected Behavior	Low	Global	Acknowledged

Description

An account can have many sub accounts all of which will have access to the same funds. Because of this users may end up having more funds traded then expected.

Recommendation

Consider giving the user control over how much a sub account can spend, or document to users that all sub accounts share the same total allowance

Resolution

GMX Team: Acknowledged.

L-02 | Comingling Of Gelato And GMX Fee

Category	Severity	Location	Status
Logical Error	Low	BaseGelatoRelayRouter.sol 300	Acknowledged

Description

The execution fee and gelato fee come from the same source, `fee.feeAmount`. That means if a user wants to use USDC to pay gelato when creating an order they will have to pay a (swap) fee to pay a (gelato) fee.

It also means that `executionFee` for GMX will not be sufficient at times since for `createOrder` it is just sending over the residual amount, so even in cases where there is no swap there is still the case where Gelato fee increases and reduces the residual amount that can be used for GMX `executionFee`.

Recommendation

Consider separating the execution fee and the gelato fee so that users can provide `wnt` for the execution fee without paying a fee on it.

Resolution

GMX Team: Acknowledged.

L-03 | Unused Enum

Category	Severity	Location	Status
Superfluous Code	Low	Errors.sol: 426	Resolved

Description

In the `Errors.sol` file the `SignatureType` enum is not used throughout the codebase.

Recommendation

Remove the unused `SignatureType` or implement it's intended use-case.

Resolution

GMX Team: Resolved.

L-04 | Lacking Feature Validation

Category	Severity	Location	Status
Validation	Low	Global	Resolved

Description

There is currently no feature to deactivate gasless transactions in the event that they need to be.

Recommendation

Consider adding a gasless transactions wide feature that can be disabled, and when disabled no gasless transactions can be executed.

Resolution

GMX Team: Resolved.

L-05 | Permit Frontrunning

Category	Severity	Location	Status
Frontrunning	Low	BaseGelatoRelayRouter.sol: 274	Resolved

Description

In the `_handleTokenPermits` function a permit is made to an arbitrary token. However since the permit signature must exist in the transaction calldata it is visible in the mempool for chains like Avalanche which have a public mempool.

This allows malicious actors to frontrun the relayer and submit this permit in a separate transaction, causing the relayer's execution of the action to fail.

Recommendation

Either ensure that the relayer is using a private RPC or consider adding logic to continue if the permit action fails, since in the case where a malicious actor has frontrun the permit the approved value has already been given.

Resolution

GMX Team: Resolved.

L-06 | Bypassing Subaccount Disabled Feature

Category	Severity	Location	Status
Logical Error	Low	SubaccountGelatoRelayRouter.sol: 157	Acknowledged

Description

The `SubaccountGelatoRelayRouter` allows users to create, update and cancel orders using a `subaccount` and relaying transaction with Gelato. All subaccount GMX interactions are done directly using the `SubaccountUtils`, skipping the `SubaccountRouter`.

During `_handleSubaccountAction`, it will validate if the subaccount feature is enabled for the `SubaccountGelatoRelayRouter`.

Therefore, if the sub account feature is disabled for `SubaccountRouter`, it will still be available using. `SubaccountGelatoRelayRouter`. This can create unexpected scenarios, if both features are not disabled at the same time.

Recommendation

Document this scenario and make sure that both subaccount features are disabled at the same time.

Resolution

GMX Team: Acknowledged.

L-07 | Missing Swap Path Validation

Category	Severity	Location	Status
Validation	Low	BaseGelatoRelayRouter.sol	Resolved

Description

The `_swapFeeTokens` function does not perform any validation on the length of the swap path provided. This does not adhere to the maximum swap length allowed on the GMX exchange for canonical orders.

Recommendation

Consider validating the swap path in the `_swapFeeTokens` function to avoid any unexpected behaviors.

Resolution

GMX Team: Resolved.

L-08 | Missing MaxFee Validation For Relayer

Category	Severity	Location	Status
Validation	Low	BaseGelatoRelayRouter.sol: 307	Resolved

Description

The router contracts will pay the Gelato relayer a fee during `_transferRelayFee`. To ensure there is a heightened control over the fees, Gelato suggests to use `_transferRelayFeeCapped` with a `maxFee` that will prevent transactions to be executed if the relayer fee is above a certain limit.

Recommendation

Introduce a `maxFee` parameter, either in the function call, or as a state variable, and use `_transferRelayFeeCapped` instead.

Resolution

GMX Team: Resolved.

L-09 | removeSubaccount Marked As Payable

Category	Severity	Location	Status
Modifiers	Low	SubaccountGelatoRelayRouter.sol: 145	Resolved

Description

The `payable` modifier is added to the `removeSubaccount` function, however this function does not do anything with the accepted Ether value.

Recommendation

Remove the `payable` modifier from the `removeSubaccount` function.

Resolution

GMX Team: Resolved.

L-10 | Duplicated orderKey Events

Category	Severity	Location	Status
Unexpected Behavior	Low	BaseGelatoRelayRouter.sol	Resolved

Description

In the swap for the fees for `_swapFeeTokens` the `orderKey` provided is the same as the order key that will be created.

The order that is created could also be a swap order and thus there may be confusion about which events correspond to the order execution and which correspond to the fee swap for the order.

Recommendation

Confirm whether or not this is expected behavior. If it is not or the behavior should be made more clear, consider assigning a different order key which is unique to, but different from, the order being swapped for in the fee swapped.

Resolution

GMX Team: Resolved.

L-11 | Nonce Dependence Prevents Subsequent Actions

Category	Severity	Location	Status
DoS	Low	Global	Acknowledged

Description

In the Relay Signature verification logic for `_validateCall` and `_handleSubaccountApproval` a monotonically increasing nonce is used to validate actions by a signer.

However since the message nonce execution must be in order if multiple messages were to be signed and one message were to fail execution then it would prevent all subsequent messages from being executed.

The message could fail execution for a number of reasons including but not limited to a deadline invalidation, a swap failure, or an errantly missing approval.

Recommendation

Consider using unique nonces which are not required to be monotonically increasing so that if one message fails it does not prevent messages signed with a higher nonce from being executed.

Resolution

GMX Team: Acknowledged.

L-12 | Redundant Required Fee Check

Category	Severity	Location	Status
Superfluous Code	Low	BaseGelatoRelayRouter.sol: 304	Acknowledged

Description

The contract validates if the `requiredRelayFee` sent by Gelato Router is greater than the output amount returned from `swapFeeTokens`.

However, the function will still revert without that validation, during `_transferRelayFee` during the token transfer if the relay params did not specify the correct amount.

Similarly, if the remaining fee tokens (residual) are not enough to cover the `executionFee`, the transaction will revert in the `orderHandler` call.

Recommendation

Avoid redundant token balance checks as they are already handled during transfer functions.

Resolution

GMX Team: Acknowledged.

L-13 | Missing validateMarketTokenBalance Validation

Category	Severity	Location	Status
Validation	Low	BaseGelatoRelayRouter.sol: 202	Acknowledged

Description

In the `_swapFeeTokens` function there is no `validateMarketTokenBalance` validation after the swap. However this validation is performed throughout the codebase whenever a swap occurs to validate that no market balances were perturbed by a swap.

Recommendation

Add `validateMarketTokenBalance` validation after the swap is invoked in the `_swapFeeTokens` function.

Resolution

GMX Team: Acknowledged.

L-14 | Tokens Which Do Not Support Permit

Category	Severity	Location	Status
Validation	Low	BaseGelatoRelayRouter.sol: 252	Acknowledged

Description

The `_handleTokenPermits` function does not validate that each target token supports the permit functionality.

This lack of validation can lead to unexpected behaviors if the permit function is routed to a token contract's fallback function instead of the expected permit function, which does not exist on that token. No particularly malicious behaviors on popular tokens has been identified at this time.

However with many tokens being supported on the GMX system and more to be added in the future, out of an abundance of caution it may be appropriate to validate that the tokens used with the `_handleTokenPermits` function are a whitelisted for permit use within GMX.

Recommendation

Consider adding validation to ensure that tokens used in the `_handleTokenPermits` function do indeed support permit functionality and are allowed by a whitelisted list.

Resolution

GMX Team: Acknowledged.

L-15 | Unnecessary Wnt Transfers To/From OrderVault

Category	Severity	Location	Status
Superfluous Code	Low	BaseGelatoRelayRouter.sol: 299	Resolved

Description

The `_handleRelayFee` will handle Gelato relay fee payment, as well as GMX `executionFee`. However, when the fee token is `wnt`, it will perform unnecessary transfers from/to the `orderVault`:

• transfer tokens from user to `OrderVault` (`_handleRelayFee`)
• transfer out `feeAmount` from `OrderVault` (`_swapFeeTokens`)
• transfer relay fee to collector (`_transferRelayFee`)

Recommendation

Do not transfer fee tokens from user to `orderVault` if the `fee.feeToken = wnt` and instead send them to the router contract. Then, avoid the call to `_swapFeeTokens` and only call `_transferRelayFee` as the tokens are already in the router.

Resolution

GMX Team: Resolved.

L-16 | Account Can Steal executionFee From subAccount

Category	Severity	Location	Status
Warning	Low	SubaccountRouter.sol	Acknowledged

Description

An account can steal funds from the subaccount by manipulating the top up amount.
The way this would be done is an account would signal to the sub account to create an order.

Then the account will frontrun the sub account and decrease the top up amount, this will truncate the amount the sub account should be refunded to only a dust amount.

Next the account will cancel the order where the execution fee will be sent to the cancelation receiver (account). If the subaccount is a bot or has any automation this attack would be easy to repeat stealing funds each time.

Recommendation

Consider putting a small time lock on `setSubaccountAutoTopUpAmount` and `setMaxAllowedSubaccountActionCount`

Resolution

GMX Team: Acknowledged.

L-17 | Account Can Avoid Refunding Subaccount

Category	Severity	Location	Status
Logical Error	Low	SubaccountRouter.sol: 218	Acknowledged

Description

When an account does not have enough funds or allowance to repay the subaccount in the `_autoTopUpSubaccount` function it will return early. An account can leverage this by frontrunning a sub account and revoking allowance or transferring funds.

Leading to the sub account not being compensated for paying the execution fee. An account can do this repeatedly to avoid all execution fee costs.

Recommendation

Instead of returning early either revert, or include an additional parameter where the sub account can choose if they want to waive the refund.

Resolution

GMX Team: Acknowledged.

L-18 | SubAccount Can Make Account Lose Funds

Category	Severity	Location	Status
Warning	Low	BaseGelatoRelayerRouter.sol	Acknowledged

Description

The user can generate a sub account, the sub account action is managed by sub account router. However, in the newly added code, the sub account, if compromised, can directly make the main account lose money by executing a swap.

The swap logic let the main account transfer the `fee.feeToken` and swap the fee token to output WETH token to pay for the relayer fee. The swap has minimum slippage control (`minOutputAmount` is `_getFee()`), so the main account can suffer from slippage loss when executing the swap.

The sub account can also compose a long swap path market pass to force the main account pay high gelato relayer fee.

Recommendation

Document such attack vector if the sub account is compromised. Validate the `fee.feeSwapPath` and the amount of fee token (input token) the sub account can be used to swap for the WETH relayer fee.

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