GUARDIAN

GMX

JIT Review

Security Assessment

September 19th, 2025

Summary

Audit Firm Guardian

Prepared By Owen Thurm, Mark Jonathas

Client Firm GMX

Final Report Date September 19, 2025

Audit Summary

GMX engaged Guardian to review the security of their GMX JIT. From the 27th of August to the 18th of September, a team of 2 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Confidence Ranking

Given the number of non-critical issues detected and code changes following the main review, Guardian assigns a Confidence Ranking of 3 to the protocol. Guardian advises the protocol to address issues thoroughly and consider a targeted follow-up audit depending on code changes. For detailed understanding of the Guardian Confidence Ranking, please see the rubric on the following page.

🔗 Blockchain network: **Crosschain**

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

📊 PoC test suite: https://github.com/GuardianOrg/gmx-synthetics-team1-1756244992487, https://github.com/GuardianOrg/gmx-synthetics-fuzz-1756245101462

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
Commit(s)	Initial commit: 1be5f5166fff8cd6016a8ce9d7b57ab93ded3870 Final commit: 9abdfe178b35d0cd7bedcb31da311387ee77e253

Audit Summary

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

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0
● High	2	0	0	0	0	2
● Medium	12	0	0	4	1	7
● Low	33	0	0	19	0	14
● Info	0	0	0	0	0	0

Audit Scope & Methodology

Scope and details:

https://github.com/gmx-io/gmx-synthetics/pull/127

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	Keeper Fees Are Trapped	Logical Error	● High	Resolved
H-02	Limit Orders Require Out Of Date Prices	Logical Error	● High	Resolved
M-01	Missing Feature Validation	Validation	● Medium	Resolved
M-02	Orders Execution Fee Does Not Cover Shifts	Validation	● Medium	Acknowledged
M-03	Inaccurate Gas Validations	Validation	● Medium	Resolved
M-04	Users Only Pay Gas For Shifts On Failure	Logical Error	● Medium	Resolved
M-05	Misleading Simulation Results	Unexpected Behavior	● Medium	Resolved
M-06	poolToPnlRatio Abuse With JIT	Gaming	● Medium	Acknowledged
M-07	GLV Arbitraged With JIT	Gaming	● Medium	Partially Resolved
M-08	Missing Keeper Error	Validation	● Medium	Resolved
M-09	GLV Used To Exit Illiquid Markets	Gaming	● Medium	Acknowledged
M-10	Target Shifts Can Not Be Simulated	Logical Error	● Medium	Resolved
M-11	GLV Oracle Lag Causes Accounting Mismatch	Gaming	● Medium	Acknowledged

Findings & Resolutions

ID	Title	Category	Severity	Status
L-01	Misleading Simulations Because Of Shared Key	Warning	● Low	Acknowledged
L-02	ADLs Executed When Orders Disabled	Warning	● Low	Acknowledged
L-03	Unused getGlvTokenPrice Function	Superfluous Code	● Low	Resolved
L-04	Lacking Reentrancy Checks	Best Practices	● Low	Acknowledged
L-05	Lacking From And To Market Check	Validation	● Low	Resolved
L-06	Outdated Naming	Documentation	● Low	Resolved
L-07	Unexpected Behavior When GLV First Created	Warning	● Low	Acknowledged
L-08	Max Pnl Validation Missed	Validation	● Low	Resolved
L-09	GlvShift Interval DoS	DoS	● Low	Acknowledged
L-10	Unclear getPrimaryPrice Behavior	Documentation	● Low	Resolved
L-11	GLV Value Updates Are Incorrect	Logical Error	● Low	Acknowledged
L-12	Unexpected Key Replay	Unexpected Behavior	● Low	Resolved
L-13	Shifts Occur No Matter The Order Result	Unexpected Behavior	● Low	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
L-14	Lacking doExecuteGlvShift Reentrancy Guard	Reentrancy	● Low	Resolved
L-15	Order Execution Feature For Liquidation Contract	Unexpected Behavior	● Low	Acknowledged
L-16	Unnecessary GLV Price Timestamp Requirements	Unexpected Behavior	● Low	Acknowledged
L-17	Unexpected Order Cancellation	Unexpected Behavior	● Low	Acknowledged
L-18	JIT DoS’d With Atomic Actions	DoS	● Low	Acknowledged
L-19	JIT May Cause Some Orders To Fail	Unexpected Behavior	● Low	Acknowledged
L-20	Multiple Shifts Cannot Occur	Logical Error	● Low	Resolved
L-21	Increased Risk Of Arbitrage	Gaming	● Low	Acknowledged
L-22	JIT Short Term Risk Free Trades	Gaming	● Low	Resolved
L-23	JIT May Prevent Withdrawals	Warning	● Low	Acknowledged
L-24	GLV May Be Left Improperly Allocated	Warning	● Low	Acknowledged
L-25	Unnecessary ReentrancyGuard	Superfluous Code	● Low	Resolved
L-26	Multiple Shifts From A Market	Validation	● Low	Acknowledged

Findings & Resolutions

ID	Title	Category	Severity	Status
L-27	Unnecessary Library Usage	Gas Optimization	● Low	Resolved
L-28	Simulation Misses Reentrancy Guard Close	Unexpected Behavior	● Low	Resolved

H-01 | Keeper Fees Are Trapped

Category	Severity	Location	Status
Logical Error	● High	JitOrderHandler.sol	Resolved

Description

In the `executeOrderFromController` function the keeper is assigned as the `msg.sender`, however the caller of this function will be the `JitOrderHandler`. As a result the keeper fees will be sent to the `JitOrderHandler` contract and trapped.

The same will occur for the execution fee upon cancellation or freezing since the `_handleOrderError` assigns the `msg.sender` as the keeper always in the invocations to `OrderUtils.cancelOrder` and `OrderUtils.freezeOrder`.

Recommendation

Add a keeper parameter to the `executeOrderFromController` function.

Resolution

GMX Team: Resolved.

H-02 | Limit Orders Require Out Of Date Prices

Category	Severity	Location	Status
Logical Error	● High	JitOrderHandler.sol	Resolved

Description

When executing a LimitIncrease or StopIncrease order through the JitOrderHandler contract, the updatedAtTime of the GlvShift action is set to the orderUpdatedAtTime. However in the case of LimitIncrease or StopIncrease orders, the orderUpdatedAtTime will often be far in the past.

The logic for deposits and withdraws which will occur as a part of the GLV shift requires that the prices set in the oracle come from around the time of the action's updatedAtTime.

As a result, if a keeper were to execute this action with prices from the order's creation the prices would be invalid for the current market value of GM markets.

In practice, this simply reverts due to failing the oracle's maximum staleness checks, so LimitIncrease and StopIncrease orders cannot be executed with JIT.

Recommendation

Consider allowing the order keeper to provide two sets of prices, one for the Shift action and one for the order action, ensuring that prices are cleared after the shifts are executed and again after the order is executed.

This way the difference in price requirements can be met no matter what. The shift action should use the most up to date prices possible, using the block.timestamp as the updatedAtTime similar to atomic withdrawals.

While the order action may have specific requirements on when the prices are from depending on when the trigger price was hit.

Also notice that the simulateExecuteJitOrder function will have to be updated to handle two separate price submissions.

Resolution

GMX Team: Resolved.

M-01 | Missing Feature Validation

Category	Severity	Location	Status
Validation	● Medium	ExecuteGlvDepositUtils.sol	Resolved

Description

In the `_processMarketDeposit` function during a GLV deposit, there is no validation to ensure that the GM market has deposits enabled before executing the deposit.

This is in contrast to the way that Shift actions execute deposits using the `executeDepositFromController` function which validates the deposit feature.

In the `_processMarketWithdrawal` function during a GLV withdrawal, there is no validation to ensure that the GM market has withdrawals enabled before executing the withdrawal.

This is in contrast to the way that Shift actions execute deposits using the `executeWithdrawalFromController` function which validates the withdrawal feature.

Recommendation

Consider validating that the withdrawal feature is enabled during these functions, or using the `executeDepositFromController` and `executeWithdrawalFromController` functions on the respective handlers.

Resolution

GMX Team: Resolved.

M-02 | Orders Execution Fee Does Not Cover Shifts

Category	Severity	Location	Status
Validation	● Medium	JitOrderHandler.sol	Acknowledged

Description

Jit orders now require the execution of several shifts, the gas needed for which will be tracked by the `startingGas` recorded in the `executeJitOrder` function.

However the order execution fee is not increased for these orders, as a result the keeper will often subsidize the gas consumption of these shifts.

Recommendation

Be aware of this behavior and consider if it is acceptable for the keeper to subsidize the gas costs for these shifts.

Resolution

GMX Team: Acknowledged.

M-03 | Inaccurate Gas Validations

Category	Severity	Location	Status
Validation	● Medium	JitOrderHandler.sol	Resolved

Description

In the executeJitOrder function of the JitOrderHandler contract the validateExecutionGas and getExecutionGas validations are performed on the gasleft() value. However this gas value does not match the gas that is available in the executeOrderFromController mostly due to the 63/64 rule for the external call to the OrderHandler. Therefore the gas available to the _executeOrder function is not accurately validated.

Recommendation

Consider moving the gas validations into the executeOrderFromController function similarly to the executeOrder function such as the following:

```
function executeOrderFromController(
bytes32 key,
Order.Props memory order,
uint256 startingGas,
bool isSimulation
) external onlyController {
uint256 executionGas = gasleft();
uint256 estimatedGasLimit = GasUtils.estimateExecuteOrderGasLimit(dataStore, order);
GasUtils.validateExecutionGas(dataStore, executionGas, estimatedGasLimit);
executionGas = GasUtils.getExecutionGas(dataStore, executionGas);
try this._executeOrder{ gas: executionGas }(
key,
order,
msg.sender,
isSimulation
) {
} catch (bytes memory reasonBytes) {
_handleOrderError(key, startingGas, reasonBytes);
}
}
```

Resolution

GMX Team: Resolved.

M-04 | Users Only Pay Gas For Shifts On Failure

Category	Severity	Location	Status
Logical Error	● Medium	JitOrderHandler.sol	Resolved

Description

In the executeJitOrder function the startingGas is declared before the shifts are processed in order to track the amount of gas used by the shift actions for the JIT order.

However the startingGas value passed to the executeOrderFromController function is only used when the order execution fails.

Upon the successful completion of an order the startingGas that is re-declared at the beginning of the _executeOrder function is used, which does not track the gas usage of the previous shift actions.

As a result users with orders that successfully complete with JIT will not track the gas expenditure made for the shifts and may end up refunding users a portion of their execution fee which ought to have paid the keeper for the shift gas expenditure.

Recommendation

Consider updating the _executeOrder function to accept a startingGas value as a parameter rather than declaring it at the beginning of the function.

Resolution

GMX Team: Resolved.

M-05 | Misleading Simulation Results

Category	Severity	Location	Status
Unexpected Behavior	● Medium	JitOrderHandler.sol	Resolved

Description

In the `simulateExecuteJitOrder` function the shared `orderHandler.executeOrderFromController` function is used to execute the simulation.

However this function includes a try/catch which will prevent any error encountered during the order's execution from being raised.

This is in contrast to the existing `simulateExecuteOrder` function which does not include a try/catch.

Recommendation

Create a separate `simulateExecuteOrderFromController` function to be used for the `simulateExecuteJitOrder` function which does not include a try/catch.

Resolution

GMX Team: Resolved.

M-06 | poolToPnlRatio Abuse With JIT

Category	Severity	Location	Status
Gaming	● Medium	Global	Acknowledged

Description

Traders now have the ability to potentially influence where GLV shifts are made by creating large orders for distinct markets.

As a result, traders could potentially extract value by pushing GLV liquidity into markets that have a `pnlToPoolRatio` above the withdrawal or trader max pnl levels.

If a pool is above the withdrawal max `pnlToPoolRatio`, any user who holds GM tokens in that market is incentivized to create a large order and trigger JIT to force liquidity to shift into that market and allow them to withdraw their GM tokens at the expense of the GLV holders, who are now holding a nearly illiquid GM token.

If a pool is above the trader's max pnl `pnlToPoolRatio`, any user who has a position which is being profit capped is incentivized to open a large trade in that market to draw more liquidity in to decrease the `pnlToPoolRatio` and unlock more of their profits at the expense of the GLV holders.

Recommendation

Consider adding off-chain validations to avoid triggering Jit order executions for markets which have a `pnlToPoolRatio` above or even close to the withdrawal `pnlToPoolRatio`.

Resolution

GMX Team: Acknowledged.

M-07 | GLV Arbitraged With JIT

Category	Severity	Location	Status
Gaming	● Medium	JitOrderHandler.sol	Partially Resolved

Description

The JIT execution strategy can be triggered by users when they submit an order that exceeds the available liquidity in the pool. Given that users can trigger the exact market that JIT will move funds towards, it's possible for a user to frontrun the shift of a JIT action in order to benefit off of the liquidity amounts being deposited through price impact.

For example:
• GM A and B are in the same GLV
• The GLV holds 50 of the 100 GM A supply
• GM A has $10,000 WETH and $20,000 USDC
• GM B has $15,000 WETH and $15,000 USDC
• User A deposits $5,000 of WETH and $5,000 of USDC into GM B
• GM B now has $20,000 WETH and $20,000 USDC
• User A triggers a JIT action to shift from GM A to GM B by opening a large order for GM B
• GLV Shifts 50% of GM A into GM B
• GM B now has $25,000 WETH and $30,000 USDC, GLV experiences negative impact during this shift because GM B is now imbalanced
• User A withdraws $2,500 WETH and $7,500 USDC and receives positive impact because they have now balanced GM B back to $22,500 WETH and $22,500 USDC

There are maximum loss checks that take place upon GLV Shifts, however this only applies for a single GLV shift action. If an actor repeats this process many times over a period of time they can arbitrage much more than the maximum loss percentage over several shifts.

To make this strategy easily executable, users could continue to submit orders that trigger JIT but automatically fail their execution and are cancelled due to minimum output amount configurations on the increase order swap trivially failing to prevent the user from taking unwanted exposure for large orders.

Recommendation

Consider adding logic in the off-chain JIT keeper that prevents JIT actions from occurring when GLV Shifts have accumulated a certain threshold of losses due to price impact and other fees over a period of time. Furthermore, consider simulating the result of the JIT action and ensuring that the order is successfully executed and not cancelled or frozen by the JIT execution.

Resolution

GMX Team: Partially Resolved.

M-08 | Missing Keeper Error

Category	Severity	Location	Status
Validation	● Medium	BaseHandler.sol	Resolved

Description

In the OrderUtils.cancelOrder function the InsufficientGasForAutoCancellation error has been introduced which is raised instead of the InsufficientGasForCancellation error for non-autoCancel orders.

However the InsufficientGasForCancellation error has special handling where it causes a revert in the OrderHandler._handleOrderError function to prevent keepers from censoring orders.

The InsufficientGasForAutoCancellation is not included in the validateNonKeeperError validation and therefore allows keepers to censor orders by providing insufficient gas for the auto cancellation order.

Recommendation

Include the InsufficientGasForAutoCancellation error in the validateNonKeeperError validation.

Resolution

GMX Team: Resolved.

M-09 | GLV Used To Exit Illiquid Markets

Category	Severity	Location	Status
Gaming	● Medium	Global	Acknowledged

Description

With the new JIT feature there is a pathway for holders of GM tokens that are illiquid due to poolToPnl ratio to exit their GM holdings at the expense of GLV.

Consider the following scenario:

- User A holds 100 GM A
- GM A is at a pnlToPoolRatio of 30%
- The pnlToPoolRatioForWithdrawals is 20%
- GM A is currently locked out of withdrawals
- User A forces GLV to shift into GM A by creating a large trade in GM A
- The pnlToPoolRatio of GM A drops to 15% because User A's large position has no profit
- User A closes their large position
- User A withdraws their GM A tokens since now the pnlToPoolRatio allows them to do so
- As a result of User A's withdrawal, the pnlToPoolRatio goes back to 20%
- GLV is stuck with the now again illiquid GM A tokens

Recommendation

Be aware of this manipulation to exit illiquid GM markets. Consider adding validation to prevent such gaming based on the pnlToPoolRatio of markets that are being shifted into. No market that is the receiver of a GlvShift should be above the maximum withdrawalPnlToPoolRatio.

Resolution

GMX Team: Acknowledged.

M-10 | Target Shifts Can Not Be Simulated

Category	Severity	Location	Status
Logical Error	● Medium	ShiftHandler.sol	Resolved

Description

In the ShiftHandler contract the simulateExecuteShift function still has the onlyController modifier which means that users cannot simulate the execution of a particular shift.

Recommendation

Remove the onlyController modifier from the simulateExecuteShift function in the ShiftHandler contract.

Resolution

GMX Team: Resolved.

M-11 | GLV Oracle Lag Causes Accounting Mismatch

Category	Severity	Location	Status
Gaming	● Medium	GlvUtils.sol	Acknowledged

Description

When using the oracle price for the getGlvValue result, the GLV price is lagging from a few seconds ago. However the price of the underlying market tokens which are subsequently used in the deposit or withdrawal actions, e.g. to figure out how many market tokens to withdraw, is from the current block. This time differential between when the GLV price is measured from and when the underlying GM price is measured from results in mispricing of GLV which occurs over time. The most obvious case of this is when a GM market experiences a stepwise jump in price.

Though this is not necessary for this affect to take place over a period of time as small deviations build up and affect the GLV price. For the sake of example we will examine the effect on GLV price when an underlying GM market experiences a stepwise decrease in price resulting from the insolvent liquidation of a position. The stepwise decrease in price occurs from the system realizing in that single liquidation transaction that the position could not pay for it's total negative PnL.

Consider the following scenario:
• GM A is priced at $2 in block 100
• GLV holds just 5 GM A tokens
• GLV supply is 5
• GLV is worth $10 in block 100
• User A withdraws 2 GLV to GM A and their GLV withdrawal is processed in block 105 using the GLV price from block 100 via the oracle
• GLV is valued at $2 per GLV during the withdrawal, giving User A $4 of withdrawal value
• GM A is valued at $1 during the withdrawal, because it is using the valuation of the market token from block 105
• User A withdraws 4 GM A tokens from the GLV
• Now GLV price is 1 GM A token per 3 GLV = $0.33
• If User A had not done this withdrawal straddling the stepwise decrease in GM A price, the GLV price would rightfully be $1.

Notice that a distinct stepwise decrease does not have to happen, but if minor deviations between these two times contribute to shifts in price over a long period of time GLV holders will have their earnings markedly affected.

Recommendation

Ideally, the price of the GM token being deposited and withdrawn would be valued from the same point in time as the oracle price for GLV. Otherwise an imperfect solution would be to simply ensure that the block lag is consistently as small as possible for the GLV oracle, and hope that no notable stepwise jumps are straddled by a GLV action.

Resolution

GMX Team: Acknowledged.

L-01 | Misleading Simulations Because Of Shared Key

Category	Severity	Location	Status
Warning	● Low	ExchangeRouter.sol	Acknowledged

Description

The simulateExecuteLatestJitOrder function assumes that the current key belongs to an order, however since the key is shared across multiple actions such as orders, deposits, and withdrawals, the key may not belong to an order object.

This can lead to perturbed simulations when a deposit or withdrawal has been created after an order and the deposit or withdrawal is fetched from the dataStore in place of the latest order.

Recommendation

Be aware of this behavior and consider if it is desired in the simulation flow.

Resolution

GMX Team: Acknowledged.

L-02 | ADLs Executed When Orders Disabled

Category	Severity	Location	Status
Warning	● Low	AdlHandler.sol	Acknowledged

Description

The AdlHandler only validates if the executeAdlFeatureDisabledKey feature is disabled before executing an order with the ExecuteOrderUtils.executeOrder function.

This ignores the executeOrderFeatureDisabledKey feature and is in contrast to the liquidation handler which validates the executeOrderFeatureDisabledKey feature.

Recommendation

Consider if this is the expected behavior and consider also validating the executeOrderFeatureDisabledKey feature in the AdlHandler.

Resolution

GMX Team: Acknowledged.

L-03 | Unused getGlvTokenPrice Function

Category	Severity	Location	Status
Superfluous Code	● Low	GlvUtils.sol	Resolved

Description

The getGlvTokenPrice function which takes only 5 parameters defined here:

https://github.com/gmx-io/gmx-synthetics/blob/1be5f5166fff8cd6016a8ce9d7b57ab93ded3870/contracts/glv/GlvUtils.sol#L145 is not used throughout the codebase.

Recommendation

Consider if this version of the function should be used instead of the following one which is used in the GlvReader contract.

Resolution

GMX Team: Resolved.

L-04 | Lacking Reentrancy Checks

Category	Severity	Location	Status
Best Practices	● Low	OrderHandler.sol	Acknowledged

Description

In the new executeOrderFromController function there is no nonReentrant modifier to prevent re-entrancy back into the executeOrderFromController function.

For all functions on handler contracts which are not globalNonReentrant, they should be declared nonReentrant.

Ideally all globalNonReentrant functions in a handler contract are also marked nonReentrant, so that there can be no re-entrancies that begin with a globalNonReentrant function and enter into a simple nonReentrant function.

While not strictly necessary, the appropriate re-entrancy guards for non globalNonReentrant functions will further constrict the allowed execution paths to only expected pathways.

Recommendation

Firstly, mark the executeJitOrder function as nonReentrant. Secondly, consider making all handler functions nonReentrant in addition to globalNonReentrant so that the purely nonReentrant functions cannot be re-entered into from a globalNonReentrant function.

This extra validation comes with a cost of an additional storage write which will negatively impact gas costs. To combat this, instead a transient storage re-entrancy guard can be used.

Resolution

GMX Team: Acknowledged.

L-05 | Lacking From And To Market Check

Category	Severity	Location	Status
Validation	● Low	GlvShiftUtils.sol	Resolved

Description

The validateGlvShift function does not validate if the from market and to market are the same market.

This could lead to unexpected issues if the orderKeeper accidentally initiates a shift where the from market is the same market as the to market.

Recommendation

Consider adding a validation to the validateGlvShift function that prevents the from and to markets from being the same address.

Resolution

GMX Team: Resolved.

L-06 | Outdated Naming

Category	Severity	Location	Status
Documentation	● Low	GlvShiftUtils.sol	Resolved

Description

In the validatePriceImpact function, the function itself and several variables mention price impact. However the underlying key has been renamed to the glvShiftMaxLossFactorKey.

Recommendation

Consider renaming the validatePriceImpact function and the variables inside of it to be congruent with the new glvShiftMaxLossFactorKey name.

Resolution

GMX Team: Resolved.

L-07 | Unexpected Behavior When GLV First Created

Category	Severity	Location	Status
Warning	● Low	Global	Acknowledged

Description

Because GLV now uses an oracle for prices, there will be some lag between the current GLV price and the price reported by the oracle. One particular case which causes a significant difference between these is when a GLV has been created and receives its first deposit.

The initial deposit requires that a small amount of GLV is locked to set an initial non-trivial supply and backing. This is the first jump where GLV goes from being worth nothing to being worth something.

If the GLV oracle is already live at this point there may be unexpected behavior where in the block of the first deposit, and potentially in the subsequent blocks afterwards, the oracle continues to report GLV as being worth nothing while GLV is indeed worth something.

This may lead to unexpected edge cases which are not foreseen, for example this causes a divide by zero revert in the `usdToGlvTokenAmount` function.

Recommendation

Be sure to only create the oracle for a GLV and allow it to be used after the GLV is established with some amount of non-trivial TVL for a sufficient duration.

Resolution

GMX Team: Acknowledged.

L-08 | Max Pnl Validation Missed

Category	Severity	Location	Status
Validation	● Low	ExecuteGlvDepositUtils.sol	Resolved

Description

In the _processMarketDeposit function the validateMaxPnl validation only occurs if the deposit is a market token deposit.

However this validation could be performed even if the GLV deposit is not a market token deposit, since it is bad for GLV holders to receive a GM token which is above it's max pnl factor for withdrawals regardless of how the depositor obtains them.

There is a clear incentive for a depositor to use the GLV as exit liquidity if they are already holding these GM tokens, however it's worthwhile to always prevent the illiquid tokens from winding up in GLV when the max pnl factor for withdrawals is exceeded.

Recommendation

Consider performing the validateMaxPnl always in the _processMarketDeposit function.

Resolution

GMX Team: Resolved.

L-09 | GlvShift Interval DoS

Category	Severity	Location	Status
DoS	● Low	Global	Acknowledged

Description

When a GLV Shift occurs a cooldown is reset which prevents further GLV shifts from occurring for a given period of time.

With the JIT feature, if this cooldown period is configured for a non-trivial amount of time, users could prevent legitimate rebalances or other JIT actions by consistently placing large trades and then closing them shortly thereafter.

Recommendation

Be aware of this DoS vector and ensure that the cooldown period for GLV shifts is assigned with this in mind.

Resolution

GMX Team: Acknowledged.

L-10 | Unclear getPrimaryPrice Behavior

Category	Severity	Location	Status
Documentation	● Low	Oracle.sol	Resolved

Description

The getPrimaryPrice function implementation documentation describes that it reverts when the price is empty for the given token.

However the getPrimaryPrice function does not revert when the token is the zero address and the price is instead returned as the empty price.

Recommendation

Either make this behavior abundantly clear with a warning or consider changing the behavior of the _getPrimaryPrice function to revert if raiseOnEmpty is true and the provided token is the zero address.

Resolution

GMX Team: Resolved.

L-11 | GLV Value Updates Are Incorrect

Category	Severity	Location	Status
Logical Error	● Low	ExecuteGlvDepositUtils.sol	Acknowledged

Description

In the executeGlvDeposit function the getGlvValue function is used before and after the execution of the GLV deposit. Before the deposit the getGlvValue function is used to determine the GLV mint amount based on the current value of GLV and the value of the assets being deposited by the user.

After the deposit has been executed, the getGlvValue function is used a second time to query the updated value of GLV to emit with the emitGlvValueUpdated event.

However the forceCalculation boolean for this second invocation is false, and therefore it reports the same GLV price that was set by the oracle, showing no updated GLV value after the deposit.

Similarly during GLV withdrawals and shift flows the getGlvValue function is queried again to emit the updated value in the emitGlvValueUpdated event, though it inaccurately reports this because the forceCalculation is false.

Recommendation

Consider removing the emitGlvValueUpdated event from the GLV deposit, GLV withdrawal, and GLV shift flows.

Resolution

GMX Team: Acknowledged.

L-12 | Unexpected Key Replay

Category	Severity	Location	Status
Unexpected Behavior	● Low	JitOrderHandler.sol	Resolved

Description

In the JitOrderHandler _createGlvShift function the key of the GLV shift is based on the orderKey and the index of the shift action in the shift list. Keys are generally expected to be unique to each action and only correspond to one distinct action on the exchange.

However, it’s possible for the same orders to be executed multiple times through the JitOrderHandler in the event that an order is initially frozen and then unfrozen by the user updating the order.

As a result, the _createGlvShift can construct the same key for a corresponding GLV shift action, only this time the actual shift parameters of the from and to markets could be different.

This breaks the typical assumption that an action key is only executed once and that an action key pertains to one unique action on the exchange. In this case the ultimate effect is just that events will emit updates for this key multiple times across potentially different GLV shift actions.

Recommendation

Be aware of this behavior and if it is not desired, consider making the key unique to the GLV shift parameters so that the key must at least pertain to the same kind of shift. Or enforcing that it is unique to the JIT execution by hashing the block.timestamp with the key.

Resolution

GMX Team: Resolved.

L-13 | Shifts Occur No Matter The Order Result

Category	Severity	Location	Status
Unexpected Behavior	● Low	JitOrderHandler.sol	Resolved

Description

The JitOrderHandler relies on the executeOrderFromController for order execution. This executeOrderFromController function does not revert if the order fails to execute, and instead uses the _handleOrderError function of the OrderHandler to silently handle the cancellation or freezing of said order.

This results in shifts being executed and remaining executed in the event that an order executed through the JitOrderHandler fails execution. This may be unexpected since liquidity was moved without being utilized by the increase order.

Recommendation

Consider if this is the expected behavior. If not, consider reverting the executeOrderFromController in the event that the order execution fails.

Resolution

GMX Team: Resolved.

L-14 | Lacking doExecuteGlvShift Reentrancy Guard

Category	Severity	Location	Status
Reentrancy	● Low	GlvShiftHandler.sol	Resolved

Description

In the GlvShiftHandler there is no reentrancy guard for the doExecuteGlvShift function which can be called by the contract itself or from a controller contract.

A reentrancy guard could be added which would not clash with the globalNonReentrant of the executeGlvShift function and add some protection against any unexpected reentrancy into specifically the doExecuteGlvShift from outside controllers.

Recommendation

Consider adding a nonReentrant modifier to the doExecuteGlvShift function.

Resolution

GMX Team: Resolved.

L-15 | Order Execution Feature For Liquidation Contract

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

Description

The `LiquidationHandler` contract validates the `executeOrderFeatureDisabledKey` however it uses the `address(this)` which means that the feature enabled status will be validated with the `LiquidationHandler` contract as the context.

Therefore if the order feature is disabled for the `OrderHandler`, the order execution feature can still be enabled and orders can still be executed through the `LiquidationHandler`.

Recommendation

Be aware of this behavior.

Resolution

GMX Team: Acknowledged.

L-16 | Unnecessary GLV Price Timestamp Requirements

Category	Severity	Location	Status
Unexpected Behavior	● Low	Global	Acknowledged

Description

During the execution of a GLV deposit or a GLV withdrawal, the prices used by the oracle are required to be accurate for the market order timestamp rules.

This includes the GLV price which will be submitted to the oracle but not used for the withdrawal or deposit sub action processing.

The GLV price is unnecessarily subject to the rules of the index, long, and short token prices that will be used for the deposit or withdrawal actions.

This may cause issues in some cases where the most up to date price for GLV cannot be used to execute a GLV order that is somewhat stale, but still normally executable.

This gives more credence to the issue described in H-02, where in some cases keeper's may be required to submit a slightly outdated GLV price to meet these requirements.

The same applies to GLV shifts, where the order keeper may create a GLV shift and not immediately execute the request. In this case unnecessary constraints may apply to the GLV price submission upon execution of the GLV shift.

Recommendation

Keepers must be adept at understanding when to provide the GLV price from the data stream versus when to not provide the GLV price from the data stream when the latest most accurate price cannot satisfy the deposit and withdrawal market order price timestamp requirements.

Alternatively, similar to the solution of H-02, the computation could be forced for all GLV deposits and withdrawals and GLV shifts that are created in two-steps through the `GLVShiftHandler`.

Resolution

GMX Team: Acknowledged.

L-17 | Unexpected Order Cancellation

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

Description

In the OrderUtils.cancelOrder function the isSupportedOrder validation now optionally raises either the UnsupportedOrderType error or the UnsupportedOrderTypeForAutoCancellation error depending on if the cancellation being executed is an auto cancellation or not.

The UnsupportedOrderType error is specifically handled in the OrderHandler._handleOrderError function so that the execution reverts instead of going on to cancel the order. However the UnsupportedOrderTypeForAutoCancellation error is not explicitly handled this way.

This means that when an outdated autoCancel order using an old order type which is no longer supported is encountered, the order that triggers this autoCancellation will fail it’s execution and be cancelled.

The alternative behavior, if there were similar specific handling for the UnsupportedOrderTypeForAutoCancellation error, would be that the order fails execution and simply reverts instead of being cancelled.

Recommendation

This behavior may be fine, though it differs from the handling of other UnsupportedOrderType failures. Be aware of this inconsistency and consider if this is the desired behavior when an outdated unsupported order type is encountered during auto cancellation.

Resolution

GMX Team: Acknowledged.

L-18 | JIT DoS’d With Atomic Actions

Category	Severity	Location	Status
DoS	● Low	Global	Acknowledged

Description

Though there is minimal incentive for a perpetrator, it is possible to DoS the execution of a JIT action or the possibility of such execution by making atomic actions to swap or withdraw the necessary backing liquidity that would allow the JIT to take place and occur within the acceptable impact threshold configured.

Recommendation

Be aware of this DoS vector, though it is unlikely to provide any benefit to the perpetrator at a non-trivial cost.

Resolution

GMX Team: Acknowledged.

L-19 | JIT May Cause Some Orders To Fail

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

Description

If the increase order has a `swapPath` which goes through a GM market that the JIT action pulls liquidity from, the JIT action itself could cause the order execution to fail.

This may be unexpected and in some cases may unnecessarily cause orders to be cancelled rather than executed.

Recommendation

If there is ever an option between using a GM market which is not in the order's `swapPath` and using one which is in the order's `swapPath` for a JIT action, the GM market which is not in the order's `swapPath` should be used.

Resolution

GMX Team: Acknowledged.

L-20 | Multiple Shifts Cannot Occur

Category	Severity	Location	Status
Logical Error	● Low	GlvShiftUtils.sol	Resolved

Description

In the executeGlvShift function the validateGlvShiftInterval validation is performed to ensure that for a given GLV shifts are only executed at most once every glvShiftMinInterval.

However, in the context of a JIT order execution a list of shifts will be provided and all must be executed within the same GLV because the receiving GM market can only belong to one GLV token.

The list of shifts will not be able to be executed because the first shift action will set the glvShiftLastExecutedAtKey value to the current block.timestamp, causing the following shift executions to fail the validateGlvShiftInterval validation.

Recommendation

Consider either setting the glvShiftMinInterval to zero and retiring this validation, or allowing the validation to be bypassed by the JIT shift execution.

Resolution

GMX Team: Resolved.

L-21 | Increased Risk Of Arbitrage

Category	Severity	Location	Status
Gaming	● Low	Global	Acknowledged

Description

Since the GLV price is now determined by an oracle which carries a small delay, there is a risk of arbitraging the outdated oracle price relative to the real underlying value of GLV.

This is because users can directly observe the real value of GLV in real time onchain, and compare this to the reported price from the oracle.

This same staleness applies to all other actions on GMX, and typically oracle delays are not a large concern because of fees applied to actions which use the oracle reference price.

However on GLV deposits with GM tokens, there are no new fees which are levied on the user. This allows users to have much more leeway to create a profitable arbitrage if e.g. the oracle reports GM A as being worth $1.50, while it is currently worth $1.53.

Notice that this slightly better chance at realizing a profit from the slight oracle staleness only applies to users who are already holding GM tokens.

Recommendation

Consider requiring a lower tolerance for the maximum staleness of the GLV oracle if it is a `dataStream`, or use an aggregator with a very low deviation threshold.

Resolution

GMX Team: Acknowledged.

L-22 | JIT Short Term Risk Free Trades

Category	Severity	Location	Status
Gaming	● Low	JitOrderHandler.sol	Resolved

Description

Due to the validations around the execution of JIT actions, it may be possible to carry out a short term trade depending on the keeper logic.

For example, it may be too price impactful to carry out a JIT action to execute a large market order order at t = 100.

The user could observe that price moves in their favor by t = 110 and decide to make a swap in the GM from market to balance it, thus allowing the JIT action to be within the overall price impact constraints.

Then and only then is the JIT action carried out, using the prices adhering to the market order requirements at t = 100.

This attack class applies to any validations that would prevent the JIT action from taking place, such as lacking GLV liquidity to shift, too significant resulting OI imbalance, too high specific GM concentration in a GLV, etc…

Recommendation

Rather than waiting to see if conditions improve to allow a JIT action to occur, execute an order normally to cancel or freeze the order immediately.

Resolution

GMX Team: Resolved.

L-23 | JIT May Prevent Withdrawals

Category	Severity	Location	Status
Warning	● Low	Global	Acknowledged

Description

Because GLV withdrawals are two step actions there may be a certain amount of GM assets that have open requests to be withdrawn from a GLV by GLV holders.

If a `GLVShift` as a result of a JIT action occurs before these GLV withdrawals are executed then it may cause these withdrawals to fail as it removes the specific GM tokens that these withdrawals requested.

Recommendation

Be aware of this risk of DoSing in-flight withdrawals and consider having the keeper take into account open withdrawal requests before initiating a JIT action.

Resolution

GMX Team: Acknowledged.

L-24 | GLV May Be Left Improperly Allocated

Category	Severity	Location	Status
Warning	● Low	Global	Acknowledged

Description

If after a JIT action is executed a user closes their position shortly after it is successfully opened, the GLV may be left improperly allocated in a market that doesn't hold much open interest.

This will require another GLV Shift to shift back to a market with high utilization, which will ultimately cost more slippage upon the execution of this additional shift. Overtime this increased shifting may have a negative impact on GLV's value accrual.

Recommendation

Be aware of this increased cost of additional shifts when markets quickly lose open interest due to concentrated positions as a result of JIT.

Resolution

GMX Team: Acknowledged.

L-25 | Unnecessary ReentrancyGuard

Category	Severity	Location	Status
Superfluous Code	● Low	JitOrderHandler.sol	Resolved

Description

The JitOrderHandler contract inherits from the OpenZeppelin ReentrancyGuard abstract contract, however does not use any of the ReentrancyGuard features.

Instead the JitOrderHandler makes use of the globalNonReentrant modifier which does not come from the ReentrancyGuard contract.

Recommendation

Remove the ReentrancyGuard contract import and usage in the JitOrderHandler contract.

Resolution

GMX Team: Resolved.

L-26 | Multiple Shifts From A Market

Category	Severity	Location	Status
Validation	● Low	JitOrderHandler.sol	Acknowledged

Description

In the _processShifts function there is no validation that ensures that a shift from a fromMarket has not already occurred.

It doesn’t make sense to shift from the same fromMarket twice since this could occur in one larger shift, therefore this unexpected execution path could be limited to avoid unexpected behavior or logical issues from the keeper.

Recommendation

Consider adding validation that checks if a market exists in the shiftParamsList as a fromMarket more than one time and reverts if such a market is found.

Resolution

GMX Team: Acknowledged.

L-27 | Unnecessary Library Usage

Category	Severity	Location	Status
Gas Optimization	● Low	JitOrderHandler.sol	Resolved

Description

In the `JitOrderHandler` contract the following library usages are unnecessary:

- `using GlvDeposit for GlvDeposit.Props`
- `using GlvShift for GlvShift.Props`
- `using GlvWithdrawal for GlvWithdrawal.Props`

Recommendation

Remove these unnecessary library usages.

Resolution

GMX Team: Resolved.

L-28 | Simulation Misses Reentrancy Guard Close

Category	Severity	Location	Status
Unexpected Behavior	● Low	JitOrderHandler.sol	Resolved

Description

In the simulateExecuteJitOrder function the globalNonReentrant modifier comes before the withSimulatedOraclePrices modifier.

As a result the _globalNonReentrantAfter block is unreachable because the withSimulatedOraclePrices ending reverts before this is reached.

The net impact is just that the simulation may show an inaccurate amount of gas used, since there would be some gas refund for resetting the REENTRANCY_GUARD_STATUS to NOT_ENTERED.

Furthermore, this contradicts the ordering that has been used throughout the GMX codebase for other simulation functions.

Recommendation

Consider updating the ordering of the two modifiers such that the withSimulatedOraclePrices modifier comes before the globalNonReentrant modifier.

Resolution

GMX Team: Resolved.

Remediation Findings & Resolutions

ID	Title	Category	Severity	Status
M-01	GLV Shift Misses Feature Validation	Validation	● Medium	Resolved
L-01	Remaining ReentrancyGuard Import	Best Practices	● Low	Resolved
L-02	onlySelfOrController Unnecessary	Superfluous Code	● Low	Acknowledged
L-03	Integration Risk During Upgrade	Warning	● Low	Acknowledged
L-04	Increased Hedge Risk For GLV	Warning	● Low	Acknowledged
L-05	GLV Oracle Inaccuracy	Logical Error	● Low	Acknowledged

M-01 | GLV Shift Misses Feature Validation

Category	Severity	Location	Status
Validation	● Medium	GlvShiftUtils.sol	Resolved

Description

The executeGlvShift function directly uses the ShiftUtils.executeShift function to execute the underlying shift action, however does not validate if the executeShiftFeatureDisabledKey feature is disabled for the ShiftHandler address.

Therefore if shifts were to be disabled they could still be executed through the GLVShift action.

Recommendation

Consider implementing and using an executeShiftFromController function which similar to other functions, validates the feature and performs the shift action.

Resolution

GMX Team: Resolved.

L-01 | Remaining ReentrancyGuard Import

Category	Severity	Location	Status
Best Practices	● Low	JitOrderHandler.sol	Resolved

Description

In the `JitOrderHandler` contract there is a `ReentrancyGuard` import that is unused.

Recommendation

Consider removing this unused import.

Resolution

GMX Team: Resolved.

L-02 | onlySelfOrController Unnecessary

Category	Severity	Location	Status
Superfluous Code	● Low	Global	Acknowledged

Description

Technically, the `onlySelfOrController` modifier can simply be replaced by the `onlyController` modifier, since it is always used by contracts which will hold the controller role.

Recommendation

Consider removing the unnecessary `onlySelfOrController` modifier in favor of the already used `onlyController` modifier.

Resolution

GMX Team: Acknowledged.

L-03 | Integration Risk During Upgrade

Category	Severity	Location	Status
Warning	● Low	Global	Acknowledged

Description

During the upgrade period for the new GMX Shift and JIT updates, there may be multiple handlers active at the same time which means deposits, shifts, glv deposits, glv withdrawals, and orders could all have executions and callbacks coming from multiple handler contracts.

For some integrations this may pose an issue.

Recommendation

Be sure to alert integrations about this upgrade in advanced.

Resolution

GMX Team: Acknowledged.

L-04 | Increased Hedge Risk For GLV

Category	Severity	Location	Status
Warning	● Low	Global	Acknowledged

Description

With the introduction of JIT, there is an increased risk that GLV is exposed to unhedged market exposure, since often the act of opening a large trade that requires new liquidity to back it will create an imbalance in the market that may or may not be filled.

Recommendation

Be aware of this increased risk of unhedged exposure that may be brought to GLV, and consider adding validations in the keeper to ensure that the resulting state of the market will not be skewed above an acceptable threshold either net long or short.

Resolution

GMX Team: Acknowledged.

L-05 | GLV Oracle Inaccuracy

Category	Severity	Location	Status
Logical Error	● Low	Global	Acknowledged

Description

Throughout the different actions that can use the new GLV price oracle, none of the account for the GLV price oracle being used in the relevant estimateOraclePriceCount functions.

This means that actions submitted for glvDeposit, glvWithdrawal, and orders that would use JIT liquidity do not accurately account for the GLV price oracle being used.

In actions like glvDeposit and glvWithdrawal the GLV price oracle may replace several price oracles and therefore the oracle price count is over estimated.

For JIT orders the oracle price count would be under estimated since the GLV oracle price would be additional to the prices required by the normal order execution flow and oracle count estimation for the order.

Recommendation

Be aware of this inconsistency in the estimated oracle price count in GLV actions and JIT actions.

Resolution

GMX Team: Acknowledged.

Disclaimer

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

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

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

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

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

About Guardian

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

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

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

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