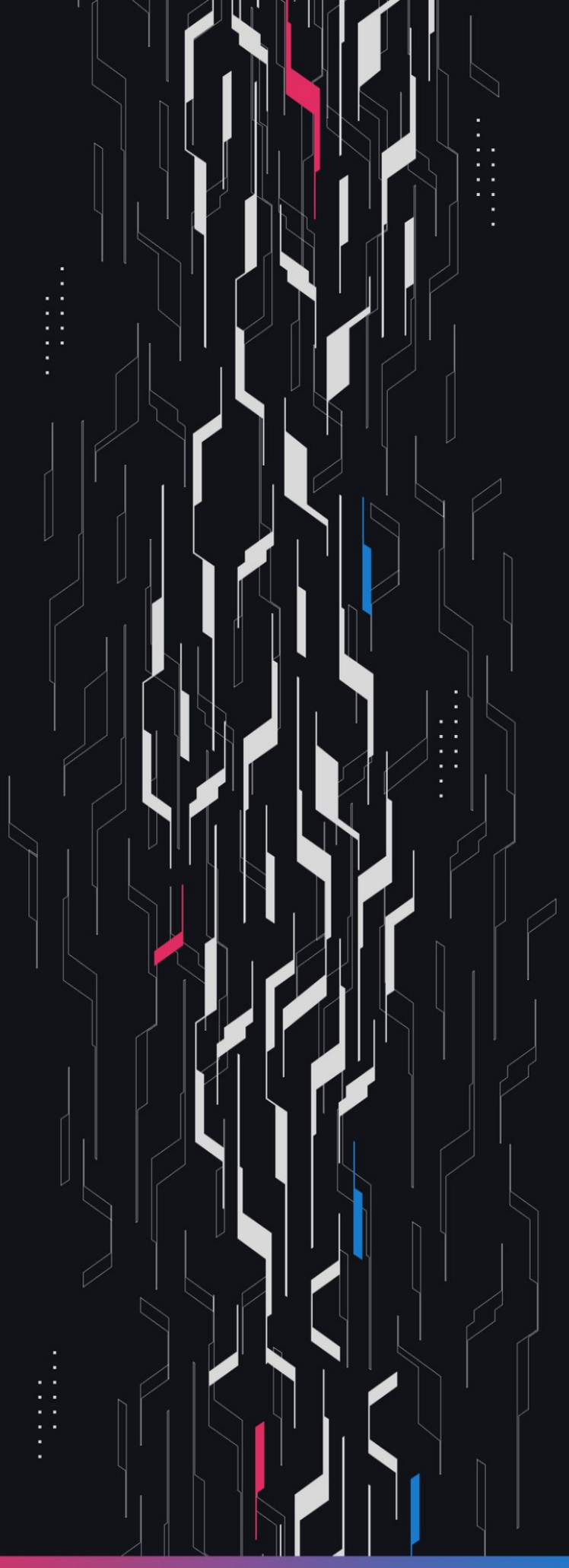
GA GUARDIAN

Umami

GMXV2 Position Manager

Security Assessment

January 26th, 2025



Summary

Audit Firm Guardian

Prepared By Daniel Gelfand, Kiki, Dogus Kose,

Wafflemakr, Cosine, Vladimir Zotov

Client Firm Umami

Final Report Date January 26, 2025

Audit Summary

Umami engaged Guardian to review the security of its GMXV2 position manager which is used as an external hedging mechanism. From the 26th of November to the 2nd of December, a team of 6 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Issues Detected Throughout the engagement 8 High/Critical issues were uncovered and promptly remediated by the Umami team.

Security Recommendation Given the number of High and Critical issues detected as well as additional code changes made after the main review, Guardian recommends that an independent security review of the protocol at a finalized frozen commit is conducted before deployment.

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

📊 Code coverage & PoC test suite: https://github.com/GuardianAudits/umamipositionmanager-fuzzing

Table of Contents

Project Overview

Project Summary

Project Name	Umami
Language	Solidity
Codebase	https://github.com/UmamiDAO/v3-vaults-generic
Commit(s)	Initial commit: 3298fe3ebb7154ac3016c560526fd1f01c0018b2 Final commit: c5e4d7d6c64db5b74029bbaed0f995de0d9ae7b1

Audit Summary

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

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	2	0	0	0	0	2
High	6	0	0	0	0	6
Medium	16	0	0	1	0	15
Low	16	0	0	3	0	13

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 Umami's GMXV2 Position Manager, fuzz-testing with Foundry 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 Foundry fuzzing suite.

ID	Description	Tested	Passed	Remediation	Run Count
POS-01	IncreaseRequests should only have member executed = true after order execution	✅	✅	✅	10M+
POS-02	AggregateVault USDC balance should decrease after increasePosition in Shorts	✅	✅	✅	10M+
POS-03	AggregateVault WETH balance should decrease after increasePosition in Longs	✅	✅	✅	10M+
POS-04	DecreaseRequest should only have member executed = true after order execution	✅	✅	✅	10M+
POS-05	Position key should be zero after decreasing to zero	✅	❌	✅	10M+
POS-06	Decreasing a position should not impact USDC TVL.	✅	❌	✅	10M+
POS-07	Decreasing a position should not impact WETH TVL.	✅	✅	✅	10M+
POS-08	Position size should be zero after close	✅	✅	✅	10M+
POS-09	Margin should be zero after close	✅	✅	✅	10M+

Invariants Assessed

ID	Description	Tested	Passed	Remediation	Run Count
POS-10	Position key should be zero after decreasing to zero	✅	✅	✅	10M+
POS-11	Closing a position should not impact USDC TVL	✅	✅	✅	10M+
POS-12	Closing a position should not impact WETH TVL	✅	✅	✅	10M+
POS-13	Margin should be increased after successful execution	✅	✅	✅	10M+
POS-14	Margin should be decreased after successful execution	✅	✅	✅	10M+
POS-15	Decreasing a margin should not impact USDC TVL	✅	✅	✅	10M+
POS-16	Decreasing a margin should not impact WETH TVL	✅	✅	✅	10M+
POS-17	Claim of the pending funding should not impact USDC TVL.	✅	✅	✅	10M+
POS-18	Claim of the pending funding should not impact WETH TVL.	✅	✅	✅	10M+
POS-19	Position amounts are not equal after close and decrease simulation	✅	✅	✅	10M+
DOS-01	Position margin should not revert	✅	✅	✅	10M+
DOS-02	Get position PNL call should not revert	✅	✅	✅	10M+

Findings & Resolutions

ID	Title	Category	Severity	Status
C-01	getPositionPnl Always Returns 0	Logical Error	Critical	Resolved
C-02	Insufficient Access Control On Callbacks	Access Control	Critical	Resolved
H-01	Claimable Collateral Cannot Be Claimed	Logical Error	High	Resolved
H-02	User Can Escape Cost Of Holding A Position	Logical Error	High	Resolved
H-03	Wrong Vault Benefits From Funding Fee Claims	Logical Error	High	Resolved
H-04	Stepwise Jump From Claimable Funds Omitted	Logical Error	High	Resolved
H-05	ADL Returns Native ETH	Logical Error	High	Resolved
H-06	GMX Callback Revert Due To Stale LLO Prices	Logical Error	High	Resolved
M-01	No Way Of Canceling Stuck Order	DoS	Medium	Resolved
M-02	Execution Feature Check Is Missing	DoS	Medium	Resolved
M-03	SequencerUpTime Check Is Missing	Validation	Medium	Resolved
M-04	Validations Perform With Zero Size Delta	Validation	Medium	Resolved
M-05	Missing Open Interest Validation	Logical Error	Medium	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
M-06	Missing Validation For Increase Position	Validation	Medium	Resolved
M-07	Position Size Not Validated	Validation	Medium	Resolved
M-08	Missing Referral Code And Refund Configuration	Logical Error	Medium	Resolved
M-09	Key Not Cleared When Closing Position	Logical Error	Medium	Resolved
M-10	DOS When Losses Exceed Size	DoS	Medium	Resolved
M-11	DoS If Funding Fee Claims Are Disabled	DoS	Medium	Resolved
M-12	Callback Contract Not Set	Logical Error	Medium	Resolved
M-13	Margin Calculations Revert With Disabled Market	Logical Error	Medium	Acknowledged
M-14	Stored CollateralDelta Can Be Manipulated	Logical Error	Medium	Resolved
M-15	Wrong Acceptable Price Used In GMX	Logical Error	Medium	Resolved
M-16	Incorrect Calculation For _sizeDelta	Logical Error	Medium	Resolved
L-01	Extracting Value With Increasing Orders	Logical Error	Low	Resolved
L-02	Missing Keeper Fee Logic	Logical Error	Low	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
L-03	Function Naming Pattern Broken	Warning	Low	Resolved
L-04	Stored Market Should Reflect Position's Market	Logical Error	Low	Resolved
L-05	Redundant Logic In positionMargin	Logical Error	Low	Resolved
L-06	Wrong isLong For PositionRequest Event Emission	Warning	Low	Resolved
L-07	Superfluous Code	Best Practices	Low	Resolved
L-08	Convention For Storage Location	Best Practices	Low	Acknowledged
L-09	Not Used Functions And TODO's	Warning	Low	Resolved
L-10	Unused Code Or Missing Implementation	Unused code	Low	Resolved
L-11	Stale Price From LLO	Warning	Low	Acknowledged
L-12	Duplicated Constant Param	Warning	Low	Resolved
L-13	Incorrect Natspec For Position Pnl	Documentation	Low	Resolved
L-14	Margin Only Orders Do Not Need acceptablePrice	Gas Optimization	Low	Resolved
L-15	Index Token Does Not Always Equal The Long Token	Validation	Low	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
L-16	Orders Allow High Slippage	Logical Error	Low	Acknowledged

C-01 | getPositionPnl Always Returns 0

Category	Severity	Location	Status
Logical Error	● Critical	GmxV2PositionManagerUtils.sol: 189	Resolved

Description

The `getPositionPnl` function is utilized to verify the current profit and loss of a position. However, when it calls GMX's `getPositionPnlUsd`, the `sizeDeltaUsd` parameter is set to 0.

The issue with this is that the `getPositionPnlUsd` function calculates profit and loss based on the amount of size change.

Therefore, in cases where the `sizeDelta` is zero, it indicates that no profit or loss has occurred for that portion of the position since the portion is 0.

This results in margin calculations being inaccurate by the amount of profit and loss the position currently has, impacting PPS.

Recommendation

To obtain the total profit and loss of the position, pass in the total size of the position.

Resolution

Umami Team: Resolved.

C-02 | Insufficient Access Control On Callbacks

Category	Severity	Location	Status
Access Control	● Critical	GmxV2PositionManager.sol: 572	Resolved

Description

The `afterOrderExecution` function is missing a check to see if the key is one from Umami. The issue with this is that arbitrary users can currently set Umami as the callback contract and execute the logic within this callback. Part of this logic is setting the current key.

If this were to change either by the attacker using a different collateral token or the opposite trading direction, the key would point to an empty position, resulting in the pps instantly decreasing by whatever the external position value is as well as making the actual external position unreachable without admin intervention.

To add to this any admin intervention can then be exploited, since re-adding a position would cause a stepwise jump in pps a user could deposit prior to the action and then redeem right after to extract value from the external position.

Recommendation

Validate that the key in the callback is from an order created by Umami.

Resolution

Umami Team: Resolved.

H-01 | Claimable Collateral Cannot Be Claimed

Category	Severity	Location	Status
Logical Error	High	GmxV2PositionManager.sol	Resolved

Description

The `ExchangeRouter.claimCollateral` is not implemented in the `GmxV2PositionManager` contract. Quoted from the GMX docs:

If negative price impact is capped, the additional amount would be kept in the claimable collateral pool, this needs to be manually claimed using the `ExchangeRouter.claimCollateral` function.

Recommendation

Implement the `claimCollateral` function.

Resolution

Umami Team: Resolved.

H-02 | User Can Escape Cost Of Holding A Position

Category	Severity	Location	Status
Logical Error	High	GmxV2PositionManager.sol	Resolved

Description

When calculating the value of the position the `GmxV2PositionManager` contract only takes into account the deposited collateral and the current PnL of the position. The calculation does not take Fees, discounts, funding & price impact into account. This will misprice the user's shares and can enable MEV opportunities as stepwise jumps in the share price will occur when the position is closed. Furthermore pending funding fees to be paid to the user and funding fees that have yet to be claimed but are no longer pending should be accounted for.

Recommendation

The position should be valued as if it is incurring all fees which would be levied upon it when it is completely closed as well as any pending borrowing and funding fees.

The GMX `Reader` contract has a function called `getPositionInfo` which returns the `totalCostAmount`. This variable includes all fees, discounts & funding charged to the user (not including any funding paid to the user) and can be used to calculate the real value of the position together with the `pnlAfterPriceImpactUsd` variable to include the price impact.

Firstly, query the `getPositionInfo` function on the GMX `Reader` contract to retrieve the `PositionInfo` result. The `PositionInfo` has several fields which is important to us including `PositionFundingFees funding` and `uint256 totalCostAmount`

Specifically for the funding fees we need to account for:

• `positionInfo.fees.funding.claimableLongTokenAmount` pending long token amount paid
• `positionInfo.fees.funding.claimableShortTokenAmount` pending short token amount paid
• `bytes32 key = Keys.claimableFundingAmountKey(market, token, account);` the already settled, but not yet claimed funding amount paid for each token

Resolution

Umami Team: Resolved.

H-03 | Wrong Vault Benefits From Funding Fee Claims

Category	Severity	Location	Status
Logical Error	High	GmxV2PositionManager.sol	Resolved

Description

Long tokens can collect both long and short funding fees. Because short funding fees come in the form of USDC, those funding fees will be credited to the USDC vault instead of the vault that actually has the position. Leading to a loss of yield for users who have deposited into the BTC vault.

Recommendation

When claiming the short funding fees for a long position swap the claimed USDC for the correct long token. This will ensure that the funding fees go to the correct vault. It is also important that this value while pending is also credited to the correct vault.

Resolution

Umami Team: Resolved.

Guardian Team: The newly introduced `_swapFunding` can revert which would ultimately prevent `afterOrderExecution` from settling, claiming funding fees, and updating the position state in the contract. Consider putting the swap in its own try-catch.

H-04 | Stepwise Jump From Claimable Funds Omitted

Category	Severity	Location	Status
Logical Error	High	GmxV2PositionManager.sol: 230	Resolved

Description

The `positionMargin` function fails to include claimable funding fees and claimable collateral. This omission creates opportunities for users to steal yield by timing their deposits before fee claims.

The vulnerability exists because when these fees/rewards are later claimed, they cause a step increase in the vault's total value, which directly impacts the PPS (Price Per Share).

This creates an exploitable scenario where users can:
1. Monitor positions for unclaimed fees/collateral
2. Deposit into the vault right before claims are processed
3. Capture a portion of the yield they didn't help generate
4. Exit with profits taken from legitimate long-term holders

The impact is severe because:
• Multiple claimable types are affected (funding, collateral)
• Claims/Keepers are predictable
• The attack requires no special permissions
• Profit potential scales with unclaimed amounts

Recommendation

Modify `positionMargin` to include both claimable funding fees and claimable collateral. These would be claimed with `claimFundingFees` and `claimCollateral` in the GMX `ExchangeRouter` respectively.

Resolution

Umami Team: Resolved.

Guardian Team: Claimable collateral is still not used in the `_positionMargin` function.

H-05 | ADL Returns Native ETH

Category	Severity	Location	Status
Logical Error	High	GmxV2PositionManager.sol	Resolved

Description PoC

During auto deleveraging (ADL) scenarios, GMX keepers will automatically close positions. According to the docs, When the pending profits exceed the market's configured threshold, profitable positions may be partially or fully closed.

The issue arises for the WETH vault, as ADL will return the remaining collateral in native ETH and not WETH.

Therefore, the AggregateVault.getVaultPPS will have an invalid state as only WETH balance is accounted for, creating a big step wise jump, allowing users to deposit/withdraw with an incorrect share price calculation.

Recommendation

After an ADL scenario, GMX will close the position and execute afterOrderExecution callback. Consider validating if order.flags.shouldUnwrapNativeToken is true, and either pause or wrap the native tokens into WETH.

Resolution

Umami Team: Resolved.

H-06 | GMX Callback Revert Due To Stale LLO Prices

Category	Severity	Location	Status
Logical Error	High	GmxV2PositionManager.sol: 556	Resolved

Description PoC

Protocol uses Chainlink LLO pricing for critical calculations during rebalance period, like `getVaultPPS` which fetches `GmxV2PositionManager.positionMargin`.

However, fetching LLO prices can revert if they are stale:

`if (priceDeets.lastUpdatedBlockNumber < minBlockNumber) revert PriceOutsideTolerance();`

Although `getVaultPPS` is safe as the prices are updated when rebalance period is opened and closed, this is not the case for the `GmxV2PositionManager.decreasePosition` which uses LLO pricing for PnL calculations.

Even if prices are updated just before decreasing a position, the GMX `afterOrderExecution` callback will likely revert when calculating position notional during `_updatePositionCache`.

This issue will prevent position data to be cached, specially the `key` parameter used to correctly calculate `positionMargin`.

Recommendation

Remove the `pos.size` calculation when caching the position notional during GMX `afterOrderExecution` callback, as the calculation is not used in the current implementation.

Resolution

Umami Team: Resolved.

M-01 | No Way Of Canceling Stuck Order

Category	Severity	Location	Status
DoS	Medium	GmxV2PositionManager.sol: 42	Resolved

Description

For a variety of reasons keepers may not execute an order in a timely manner or at times may never execute an order. This includes not canceling an order.

When this happens Umami has no functionality to cancel such an order themselves which means that the order along with any collateral provided will be stuck.

This also impacts the rebalance period as there is intended to be no pending orders when the rebalance period is closed.

Recommendation

Implement functionality for the keeper to call GMX's `cancelOrder` function.

Resolution

Umami Team: Resolved.

M-02 | Execution Feature Check Is Missing

Category	Severity	Location	Status
DoS	Medium	GmxV2PositionManager.sol	Resolved

Description

GMX is able to disable features, usually performed during updated. These features include `EXECUTE_ORDER_FEATURE_DISABLED`, which is crucial for the `GmxV2PositionManager`.

If orders are created but can't be executed, the contract does not have a way to cancel the order (and this feature could be disabled too).

Recommendation

Prevent orders to be created with GMX V2 if the feature is disabled using `FeatureUtils.validateFeature(DataStore(GMX_V2_DATA_STORE), Keys.executeOrderFeatureDisabledKey(address(ORDER_HANDLER),uint256(Order.OrderType.MarketDecrease)))`.

Resolution

Umami Team: Resolved.

M-03 | SequencerUpTime Check Is Missing

Category	Severity	Location	Status
Validation	Medium	OracleWrapper.sol	Resolved

Description

The `OracleWrapper.getChainlinkPrice` function does not check if the received price is stale and if the Arbitrum sequencer is up. Therefore the system will continue to work with outdated prices.

This can lead to accepting bad prices or unexpected DoS and wasted gas as the slippage check on the GMX side will revert.

Recommendation

Revert if the returned price is stale or if the sequencer is down.

Resolution

Umami Team: Resolved.

M-04 | Validations Perform With Zero Size Delta

Category	Severity	Location	Status
Validation	Medium	GmxV2PositionManagerUtils.sol: 115	Resolved

Description

`validateOpenInterestLimits` calls `validateReserve` even when the `sizeDelta` is potentially 0.

This is especially problematic since orders made to solely add collateral may be prevented from executing even if the validations won't fail on GMXV2, leading a position to be potentially liquidated and decreasing the vault's PPS.

Recommendation

Only perform `validateReserve`, `validateOpenInterestReserve` and `willPositionCollateralBeSufficient` if the `sizeDelta` is greater than 0, as in GMX's `increasePosition` function.

Resolution

Umami Team: Resolved.

M-05 | Missing Open Interest Validation

Category	Severity	Location	Status
Logical Error	Medium	GmxV2PositionManagerUtils.sol	Resolved

Description

The `GmxV2PositionManager` does not validate that the OI limits, reserve limits, and sufficient collateral checks will be passed upon order execution which may lead to invalid orders which will fail execution and delay hedging rebalance.

Recommendation

When calling `_increasePosition` call the `GmxV2PositionManagerUtils.validateOpenInterestLimits` function prior to creating the increase order.

In addition when calling `_decreasePosition` instead of using `GmxV2PositionManagerUtils.validateOpenInterestLimits`, utilize GMX's `willPositionCollateralBeSufficient` function to ensure collateral will be sufficient on decrease orders.

This is because `GmxV2PositionManagerUtils.validateOpenInterestLimits` has additional validations that are not needed on decrease orders.

Resolution

Umami Team: Resolved.

M-06 | Missing Validation For Increase Position

Category	Severity	Location	Status
Validation	Medium	GmxV2PositionManager.sol: 321	Resolved

Description

Keepers are allowed to open/increase GMX positions using `GmxV2PositionManager.increasePosition`.

During `_increasePosition`, the position `key` is calculated based on the contract's address, market, collateral and side (long or short).

However, there is no validation if there is already an active position and if the cached position key is the same as the one calculated. This will allow keepers to open a short position even if a long position is already active.

Recommendation

Make sure the cached position key matches the calculated key value when increasing position, only when the there is an active position managed.

Resolution

Umami Team: Resolved.

M-07 | Position Size Not Validated

Category	Severity	Location	Status
Validation	Medium	GmxV2PositionManager.sol	Resolved

Description

Function `increasePosition` does not validate that the collateral and size requested meets the minimum requirements in GMXV2. Consequently, an order can be created that will fail on execution, delaying the creation of a hedge.

Recommendation

Validate against `dataStore.getUint(Keys.MIN_COLLATERAL_USD).toInt256();` and `dataStore.getUint(Keys.MIN_POSITION_SIZE_USD)` on increase as in `PositionUtils.validatePosition`.

Resolution

Umami Team: Resolved.

M-08 | Missing Referral Code And Refund Configuration

Category	Severity	Location	Status
Logical Error	Medium	GmxV2PositionManager.sol	Resolved

Description

The GMX order handler will execute certain callbacks to the AggregateVault. These callbacks must be whitelisted using AggregateVault.updateDefaultHandlerContract with the function selector and the handler contract address.

However, the DeploySystem script is missing the refundExecutionFee handler setup, preventing the correct callback execution in GmxV2PositionManager. Additionally, the current deploy script does not set the referral code used when creating orders in GMX.

Once a referral code is set for an account GMX won't accept new referral codes, so when Umami attempts to change their referral code via the setReferralCode function, the referral rewards will still belong to the original code.

This can impact the protocol if the intention is to have different referral codes periodically or if there is any need to change the contract/address that is responsible for the referral rewards.

Any attempt to change the referral code would require deploying new contracts as the original contract is locked with the original code.

Recommendation

Accurately set the referral code when the contract is deployed and remove the ability to change it. Additionally, the address/contract needs to have the needed functionality to handle the referral rewards. Furthermore, add the refundExecutionFee selector to the handler in the deploy script.

Resolution

Umami Team: Resolved.

M-09 | Key Not Cleared When Closing Position

Category	Severity	Location	Status
Logical Error	Medium	GmxV2PositionManager.sol: 548	Resolved

Description

The `GmxV2PositionManager.afterOrderExecution` callback updates the cached position info after successfully increasing or decreasing a position.

The issue relies when a position is fully closed, as the `pos.key` is not deleted, to inform that the manager does not have an active position.

This will impact several part of the code like:

• `decreasePosition`, `closePosition`, `increaseMargin` and `decreaseMargin` check if there is an open position: `if (positionKey = bytes32(0)) revert NoPositionOpen();`

This prevents correct validation when the keeper is creating orders.

• `GmxV2PositionManager.getPositionPnl` call will revert. This may have greater impact if the functions is integrated by other contracts directly.

Recommendation

During `_updatePositionCache`, set key to `bytes32(0)` if position size is 0.

Resolution

Umami Team: Resolved.

M-10 | DOS When Losses Exceed Size

Category	Severity	Location	Status
DoS	Medium	GmxV2PositionManager.sol: 515	Resolved

Description

Because arbitrary users can add funds to the position at the time of creation due to the `OrderVault` recording all transfers in, as well as the fact that positions in general can be over leveraged. It is possible to create a position where more collateral than the position size is added.

Combining this with the way `_positionNotional` calculates the notional value of the position. It is possible for an underflow to occur where the losses (negative PnL) exceed the size of the position. When this happens callback functionality will revert.

This is especially problematic since the callback is how position data is updated and how funding fees are initially claimed.

Recommendation

Check if the losses exceed the size of the position and set the notional value to 0 to prevent the underflow.

Resolution

Umami Team: Resolved.

M-11 | DoS If Funding Fee Claims Are Disabled

Category	Severity	Location	Status
DoS	Medium	GMXV2PositionManager.sol	Resolved

Description

In order to successfully iterate through the `afterOrderExecution` function the funding fees needs to be successfully claimed. However, there will be times when the claim funding fees feature is disabled.

When this happens the `_claimFundingFees` function within `afterOrderExecution` will revert. Due to this revert the position will not be saved in state, this includes the positions key.

Without the key being stored any attempt to calculate PPS or reference the external position will revert, halting the protocol.

Recommendation

Call the `claimFundingFees` function within a try catch block to ensure that the protocol will not halt if the feature is disabled.

Resolution

Umami Team: Resolved.

M-12 | Callback Contract Not Set

Category	Severity	Location	Status
Logical Error	Medium	GmxV2PositionManager.sol	Resolved

Description

The callback contract is not set for the position manager. In cases where a position is liquidated or an ADL occurs, the keeper will reference whatever address is set for the callback contract for the designated account.

If no callback contract is set then it will set the callback contract to `address(0)` and skip the callback. If the callback is not called when a position is decreased or liquidated then the funding fees will be left unclaimed, and there will not be the opportunity to update the position's state.

Recommendation

Set the callback contract for the position manager by using GMX's `setSavedCallbackContract` function.

Resolution

Umami Team: Resolved.

M-13 | Margin Calculations Revert With Disabled Market

Category	Severity	Location	Status
Logical Error	Medium	Pricing.sol: 19	Acknowledged

Description

The position PnL calculation uses the `GMXPricing` library to fetch the market prices. However, it uses `MarketUtils.getEnabledMarket` which reverts if the market is disabled. Although disabling ETH/USD market seems not likely, the position manager might use other markets that could be disabled.

Due to the fact that `GmxV2PositionManager.positionMargin` uses the PnL calculations, the `getVaultPPS` will revert as well, breaking core functionality in the `AggregateVault`.

Recommendation

Consider using `MarketStoreUtils.get(dataStore, marketAddress)` instead of `getEnabledMarket` to avoid this revert. Additionally, make sure orders are not created with disabled markets.

Resolution

Umami Team: Acknowledged.

M-14 | Stored CollateralDelta Can Be Manipulated

Category	Severity	Location	Status
Logical Error	Medium	GmxV2PositionManager.sol: 377	Resolved

Description

Any donation or sitting funds in GMX will result in a larger `collateralDelta` for the GMX position than what Umami has stored. This is because GMX's `transferIn` function is based on balance change, not any parameter sent by the user.

This inaccuracy will impact the ability to accurately read the stored `collateralDelta` for the GMX position, which leads to inaccuracies in subsequent rebalance actions.

Recommendation

Get the actual Collateral Delta for the pending GMX position and store it instead of just the value passed into `_increasePosition`.

This can be done by querying the Reader getOrder function to see the actual increase amount which was recorded.

Resolution

Umami Team: Resolved.

M-15 | Wrong Acceptable Price Used In GMX

Category	Severity	Location	Status
Logical Error	Medium	GmxV2PositionManager.sol: 339	Resolved

Description

The acceptable price in both `_increasePosition` and `_decreasePosition` is based on the Chainlink price feed. However, when GMX uses the LLO, this can lead to situations where the acceptable price is different from the actual price.

This can cause orders to fail to execute or experience worse execution. For example, Chainlink has a deviation threshold 0.05% for BTC which means that for any `toleranceBps` set there is an additional 5bps slippage potentially unaccounted for.

Recommendation

Consider using the LLO when GMX uses LLO for the asset, or clearly document this behavior.

Resolution

Umami Team: Resolved.

M-16 | Incorrect Calculation For _sizeDelta

Category	Severity	Location	Status
Logical Error	Medium	GmxV2PositionManager.sol	Resolved

Description

When a keeper attempts to decrease a GMXV2 position using the `decreasePosition` function, they have the option to reduce the size of the position by a specified amount called `_usdNotional`.

In some cases it is possible that the losses of the position equal the size of the position. In this case There will be a division by 0 and revert when determining the decrease amount. `size * _usdNotional / notional`

Recommendation

To address this issue, consider validating that `notional` is not 0 prior the calculation `size * _usdNotional / notional`.

Resolution

Umami Team: Resolved.

L-01 | Extracting Value With Increasing Orders

Category	Severity	Location	Status
Logical Error	Low	GmxV2PositionManager.sol: 327	Resolved

Description PoC

When rebalance period is opened, requests executions are paused in `AggregateVault` to prevent deposits and withdrawals. However, `GmxV2PositionManager` is able to create orders in GMX outside of the rebalance period.

The issue relies during `_increasePosition`, when collateral is sent to `ORDER_VAULT`. This creates an invalid state window from the moment assets are sent until GMX keepers execute the order.

The `getVaultPPS` will return a lower value when the order is created, as `positionMargin` does not return the pending collateral, and once the order is executed, a stepwise jump in `getVaultPPS` occurs.

Keepers might want to only increase margin outside of the rebalance period, unaware of the invalid state this will generate. Even though malicious users can potentially profit from this state, this may also affect any user (i.e. withdrawal executed after margin increase order was requested).

Recommendation

Ensure and document that the external hedging position should only be modified during a rebalance period.

Resolution

Umami Team: Resolved.

L-02 | Missing Keeper Fee Logic

Category	Severity	Location	Status
Logical Error	Low	GMXV2PositionManager.sol	Resolved

Description

The `GMXV2PositionManager` does not have any logic to handle keepers fees. However, `FeeReserve` is imported into the contract as well as a Fee related event which none of it is used.

Recommendation

If it is intended to have keeper logic like there is in the V1 version it should be implemented, otherwise the `FeeReserve` import and event should be removed.

Resolution

Umami Team: Resolved.

L-03 | Function Naming Pattern Broken

Category	Severity	Location	Status
Warning	Low	GmxV2PositionManager.sol	Resolved

Description

The `GmxV2PositionManager` contract contains functions that deviate from a consistent naming convention, for example:

- `_getPositionDirectionAndSize` – Starts with an underscore but is not private/internal.
- `estimateExecuteOrderGasLimit` – Does not start with an underscore but is set to internal.

Recommendation

Stick to the Solidity naming conventions or establish and follow a consistent naming schema across the contract.

Resolution

Umami Team: Resolved.

L-04 | Stored Market Should Reflect Position's Market

Category	Severity	Location	Status
Logical Error	Low	GmxV2PositionManager.sol	Resolved

Description

When a position is decreased via the `_decreasePosition` function the market is stored as address(0) despite all the other parameters being set.

This inaccuracy will impact the ability to accurately read decrease position requests. This should not impact keepers and their ability to make future rebalances but should be fixed nonetheless.

Recommendation

Store the market address for the position when it is decreased.

Resolution

Umami Team: Resolved.

L-05 | Redundant Logic In positionMargin

Category	Severity	Location	Status
Logical Error	Low	GmxV2PositionManager.sol: 233-238	Resolved

Description

The `positionMargin` function includes an `if (margin = 0)` check that is not reachable as it is inside an `if (margin > 0)` condition. This code is redundant and can be removed.

Recommendation

Remove the `if (margin = 0) revert PositionLiquidated(_indexToken);` check.

Resolution

Umami Team: Resolved.

L-06 | Wrong isLong For PositionRequest Event Emission

Category	Severity	Location	Status
Warning	Low	GmxV2PositionManager.sol	Resolved

Description

The `PositionRequest` event emission in the `_decreasePosition` function incorrectly emits `true` for `isIncrease`.

Recommendation

For the `_decreasePosition` function emit `false` instead of `true`.

Resolution

Umami Team: Resolved.

L-07 | Superfluous Code

Category	Severity	Location	Status
Best Practices	Low	GmxV2PositionManager.sol: 321	Resolved

Description

In `_increasePosition()`, `key` is extracted via a call to `getPositionKey` function from GmxV2PositionManagerUtils.sol, but it is not used in the rest of the function/call flow.

Recommendation

Consider deleting specified unused line of code.

Resolution

Umami Team: Resolved.

L-08 | Convention For Storage Location

Category	Severity	Location	Status
Best Practices	Low	GmxV2PositionManagerStorage.sol	Acknowledged

Description

The `GmxV2PositionManagerStorage` uses a custom namespaced storage pattern, not following the convention specified in EIP-7201

Recommendation

Consider following the EIP-7201 convention for storage locations.

Resolution

Umami Team: Acknowledged.

L-09 | Not Used Functions And TODO's

Category	Severity	Location	Status
Warning	Low	GmxV2PositionManagerUtils.sol	Resolved

Description

In `GmxV2PositionManagerUtils` contract, there are functions which are not utilized in anywhere of the codebase and does not have a meaning itself such as `getAcceptablePrice`, `validateTokens` , `validateOpenInterestLimits` and `validateStableToken`.

These functions uses the vault address corresponds to GMX v1 and might be irrelevant in the current context. There are also Todo's in the contract that suggests contract is not fully ready.

Recommendation

Remove unnecessary functions and resolve TODO's.

Resolution

Umami Team: Resolved.

L-10 | Unused Code Or Missing Implementation

Category	Severity	Location	Status
Unused code	Low	Global	Resolved

Description

The following errors, events, params are never used in the code: `GmxV2PositionManager`

- `_increasePosition` function, memory param `acceptablePrice`
- `PositionRequestSettled`, `LiquidationResetError` and `FeeReserveFeeClaimed`
- `InsufficientBalance` and `UnknownAccount`
- `StorageViewer` import
- `IPositionRouter`, `IVault` and `IRouter` are gmx-v1 imports
- `_getUnrealisedFundingFees` not implemented
- `_getPositionFee` not implemented
- `marginFeeBasisPoints` not used `GmxV2PositionManagerUtils`
- `validateTokens` seems to be a function from gmx-v1 manager utils, but not used in gmx-v2
- `getAcceptablePrice` same as above.
- `validateStableToken` same as above

Recommendation

Consider removing these or implement them in the code if needed.

Resolution

Umami Team: Resolved.

L-11 | Stale Price From LLO

Category	Severity	Location	Status
Warning	Low	OracleWrapper.sol	Acknowledged

Description

`_setAndGetPriceLlo` function checks `observationsTimestamp` from Chainlink against `lloTimeTolerance`. According to Chainlink documentation `observationsTimestamp` is the latest timestamp for which price is applicable.

Hence currently contract allows using prices that exceeds `observationsTimestamp` with `LloTimeTolerance` amount.

Recommendation

Consider checking against `validFromTimestamp` to be safe against stale prices.

Resolution

Umami Team: Acknowledged.

L-12 | Duplicated Constant Param

Category	Severity	Location	Status
Warning	Low	GmxV2PositionManager.sol: 81	Resolved

Description

The `GmxV2PositionManager` uses both `BIPS` and `TOTAL_BPS` constants in the code. Although they are the same value, it can create confusion when adding new features and potentially greater impact if one constant is changed while the other is not.

Recommendation

Consider removing one constant and always using the same param, either `BIPS` or `TOTAL_BPS`.

Resolution

Umami Team: Resolved.

L-13 | Incorrect Natspec For Position Pnl

Category	Severity	Location	Status
Documentation	Low	GmxV2PositionManager.sol: 282	Resolved

Description

The `GmxV2PositionManager.getPositionPnl` natspec states: `Returns the realised and unrealised profit and loss (PnL) for the given index token`. However, only the unrealized PnL is returned.

Recommendation

Remove the `realised` PnL comment from natspec as this value is not returned.

Resolution

Umami Team: Resolved.

L-14 | Margin Only Orders Do Not Need acceptablePrice

Category	Severity	Location	Status
Gas Optimization	Low	GmxV2PositionManager.sol: 426	Resolved

Description

Keepers can either increase/decrease position size, collateral, or both. However, when only the collateral amount is modified, the the `acceptablePrice` is not verified. Therefore, the extra chainlink calls and storage reads can be avoided is `_sizeDelta` is 0.

Recommendation

Only calculate `acceptablePrice` if `_sizeDelta>0`.

Resolution

Umami Team: Resolved.

L-15 | Index Token Does Not Always Equal The Long Token

Category	Severity	Location	Status
Validation	Low	GmxV2PositionManager.sol	Resolved

Description

The `_getCollateralToken` function assumes that the long token is always the same as the index token but this is not always the case in GMX. For example DOGE/USD is backed by WETH/USDC.

Therefore the system is not able to handle such cases, this will lead to big issues if the protocol decides to add a GMX market where the index token and long token are not the same.

Recommendation

Ensure to never to add such markets for example with a check in the constructor or by documenting this behavior.

Resolution

Umami Team: Resolved.

L-16 | Orders Allow High Slippage

Category	Severity	Location	Status
Logical Error	Low	GmxV2PositionManager.sol: 427	Acknowledged

Description

When creating orders, the `acceptablePrice` is calculated based on a `toleranceBps`, set in the deployment script. Currently this value is initialized with 5% and suspiciously named `CONFIG_SWAP_SLIPPAGE_TOLERANCE`, although is not related to swaps.

Due to the fact that external hedging is done during rebalance periods and these are done approx. every 4 hours, users might anticipate this period, create long or short positions, so the `GmxV2PositionManager` position execution price incurs in higher slippage.

Recommendation

Consider reducing the `toleranceBps` to avoid high slippage scenarios.

Resolution

Umami 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