GUARDIAN AUDITS

SMART CONTRACT SECURITY AUDIT OF

Dolomite

Summary

Audit Firm Guardian

Prepared By Daniel Gelfand, Owen Thurm, Kiki, ABA

Client Firm Dolomite

Final Report Date January 11, 2024

Audit Summary

Dolomite engaged Guardian to review the security of its GMX V2 module, allowing users to use GM tokens as collateral for borrowing on Dolomite. From the 1st of November to the 15th of November, a team of 4 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Issues Detected Throughout the engagement 6 High/Critical issues were uncovered and remediated by the Dolomite team. During the fix review, 3 Critical issues were uncovered in the code changes.

Security Recommendation Given the number of High and Critical issues detected, Guardian supports a thorough internal review and independent security review of the protocol at a finalized frozen commit.

Notice that the examined smart contracts are not resistant to internal exploit. 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/DolomitePoCs

Table of Contents

Project Overview

Project Summary

Project Name	Dolomite
Language	Solidity
Codebase	https://github.com/dolomite-exchange/dolomite-margin-modules
Commit(s)	1a91bbd4526681577ae829b00744ee5fc1c210bc

Audit Summary

Delivery Date	January 11, 2024
Audit Methodology	Static Analysis, Manual Review, Test Suite

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	2	0	0	0	1	1
High	4	0	0	0	0	4
Medium	13	0	0	2	1	10
Low	15	0	0	2	0	13

Audit Scope & Methodology

Vulnerability Classifications

Vulnerability Level	Classification
🔴 Critical	Easily exploitable by anyone, causing loss/manipulation of assets or data.
🟠 High	Arduously exploitable by a subset of addresses, causing loss/manipulation of assets or data.
🟡 Medium	Inherent risk of future exploits that may or may not impact the smart contract execution.
🟢 Low	Minor deviation from best practices.

Methodology

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.

Audit Scope & Methodology

ID	File	SHA-1 Checksum(s)
GMXVF	GmxV2IsolationModeVaultFactory.sol	344bb78d18798a68bcff123b12c57c666a76599a
GMXR	GmxV2Registry.sol	1bc372f55253ceaafd6ac5747b53bce3e2d59811
VAULT	GmxV2IsolationModeTokenVaultV1.sol	c45c9c43cb45c1c56a1911a5d19a14db5e210098
GMXWT	GmxV2IsolationModeWrapperTraderV2.sol	236b0f8e904367abfd0fde7671cd1fcbc0ee1667
GMXO	GmxV2MarketTokenPriceOracle.sol	a6792810c607ed3e18a2fd25e11c0ed5c5cf1624
GMXL	GmxV2Library.sol	e7cad31a3ff038abda813d9498f7ec478044303c
GMXUT	GmxV2IsolationModeUnwrapperTraderV2.sol	5019a89aa5a1d24db184009923369f9f90513dba
ITVF	IsolationModeTokenVaultV1WithFreezable.sol	af10c87271ded42c8ae81e15ebe7dd4ffc31e4e2
AIMUTI	AsyncIsolationModeUnwrapperTraderImpl.sol	925426e1e9d06b418504449c1fb4cfc6984878ea
UAIWT	UpgradeableAsyncIsolationModeWrapperTrader.sol	15a4634b785c8b33faa5eab88c919e97e720a516
FIVF	FreezableIsolationModeVaultFactory.sol	ddc9beb7947ab65e427e1c8ec745d8c464470151
AITB	AsyncIsolationModeTraderBase.sol	530aaf95aafe7346c870ecaa9a363e90ec877cca
GTPB	GenericTraderProxyBase.sol	fe98b9a7f9edf41f30467c050547f5a51e5f6e92

Findings & Resolutions

ID	Title	Category	Severity	Status
GLOBAL-1	DoS Callbacks Through Simple Transfer	DoS	Critical	Resolved
ITVF-1	Liquidations Prevented With Pending Action	DoS	Critical	Partially Resolved
GMXL-1	Withdrawals Apply _minOutputAmount To One Side	Logical Error	High	Resolved
GMXL-2	Redemptions Incorrectly Appear Unpaused	Logical Error	High	Resolved
AIMUTI-1	Severely Undercollateralized Positions Cannot Be Liquidated	Logical Error	High	Resolved
AIMUTI-2	Withdrawal Keys Misused by Differing Subaccount in Liquidations	DoS	High	Resolved
VAULT-1	Vault Owner Can Cancel Liquidation	Access Control	Medium	Resolved
GMXO-1	Withdrawal Not Necessarily 50-50	Logical Error	Medium	Resolved
UAIWT-1	Excess GM Not Partially Deposited On Supply Cap	Logical Error	Medium	Resolved
GMXUT-1	Can't Unfreeze Vault If Execution Interrupted	DoS	Medium	Resolved
GLOBAL-2	Lack Of Liquidation Incentives	Incentives	Medium	Resolved
GMXL-3	Withdrawals Fail When A Backing Token is Zero	Logical Error	Medium	Partially Resolved
UAIWT-2	Fund Transfer From Wrapper Trader Can Be Skipped	Logical Error	Medium	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
GMXO-2	Potentially Misleading PnL Factor	Oracle Risk	Medium	Resolved
GLOBAL-3	Vault Frozen On Single Account	Logical Error	Medium	Resolved
GMXO-3	Liquidator Can Force Liquidation	Protocol Manipulation	Medium	Resolved
FIVF-1	Liquidation Needs To Match Pending Output Token	Validation	Medium	Acknowledged
AITB-1	Unbounded Execution Fee For Deposits and Withdrawals	Validation	Medium	Resolved
GLOBAL-4	Sequencer May Experience Outages	Logical Error	Medium	Acknowledged
GLOBAL-5	Features May Be Disabled	Logical Error	Low	Resolved
GMXL-4	Inefficient Execution Fee Handling	Superfluous Code	Low	Resolved
UAIWT-3	Incorrect maxWei Limit Bypass Check	Logical Error	Low	Resolved
GMXL-5	Redundant Self Import	Superfluous Code	Low	Resolved
GMXO-4	GM Price Impact May Be Misrepresented	Oracle Risk	Low	Resolved
GTPB-1	Incorrect Interface Used	Typo	Low	Resolved
UAIWT-4	Dolomite Assumes Owner Can Handle GM	Documentation	Low	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
UAIWT-5	Inaccurate File Name	Typo	Low	Resolved
GMXO-5	Typo	Typo	Low	Resolved
GMXO-6	Inaccurate Comment	Documentation	Low	Resolved
GLOBAL-6	Wrapping Fails With Zero Output	Documentation	Low	Resolved
GLOBAL-7	Liquidations Cannot Include Pending Keys	Documentation	Low	Acknowledged
GMXIVF-1	Swap-only GMX Markets Are Unsupported	Unsupported Feature	Low	Resolved
VAULT-2	Stored Liquidation Fee Too Insufficient	Incentives	Low	Acknowledged
GMXO-7	Only Fee For Negative Price Impact Is Read	Oracle Risk	Low	Resolved

GLOBAL-1 | DoS Callbacks Through Simple Transfer

Category	Severity	Location	Status
DoS	● Critical	Global	Resolved

Description PoC

When the callback `afterWithdrawalExecution` is triggered, it is checked that the amount of GM tokens sent to GMX to be redeemed match the amount of GM actually redeemed: `Require.that(withdrawalInfo.inputAmount == _withdrawal.numbers.marketTokenAmount)`

GMX records how much GM needs to be withdrawn by comparing the balance of GM before and after the function `createWithdrawal` is called. An attacker can easily cause a mismatch between `withdrawalInfo.inputAmount` and `marketTokenAmount` by sending 1 wei of GM to the Withdrawal Vault before an unwrapping is initiated.

The short or long token will be stuck in the Unwrapper Trader, the vault will remain frozen prohibiting any user execution, and the Core protocol will still believe the user holds the GM collateral which no longer truly exists. This will affect any user's normal withdrawal as well as liquidations.

This attack is applicable to the `afterDepositCancellation` callback as well due to the following validation: `assert(_deposit.numbers.initialLongTokenAmount == 0 || _deposit.numbers.initialShortTokenAmount == 0);` An attacker can send 1 wei to the Deposit Vault prior to the call of the `createDeposit` function, causing a revert at this assertion.

Recommendation

Do not use a strict equality. Modify the validations such that the amounts of the event data are expected to be greater than or equal to the amounts Dolomite expects from its system. Furthermore, if two tokens are received from GMX upon cancellation, either deposit the unintended token into Dolomite if supported or send it to the Vault owner.

Resolution

Dolomite Team: The issue was resolved in commit 90b5b77.

ITVF-1 | Liquidations Prevented With Pending Action

Category	Severity	Location	Status
DoS	● Critical	IsolationModeTokenVaultV1WithFreezable.sol: 670	Partially Resolved

Description PoC

When performing a liquidation, the amount the user is to be liquidated for is validated with function `_validateWithdrawalAmountForUnwrapping`:

`Require.that(balance - (withdrawalPendingAmount + depositPendingAmount) > 0);`

A user can simply initiate a withdrawal for their entire balance, but specify a `minOutputAmount` of long/short token that is impossible to achieve with the provided amount of GM to withdraw. If GMX were to execute their withdrawal, the user would listen to the cancellation event, and reinitiate another unwrapping with the same parameters.

By doing so, the user prevents the Liquidator from passing any withdrawal amount greater than 0, as `balance = withdrawalPendingAmount` and `balance - (withdrawalPendingAmount + depositPendingAmount) == 0` which fails the above validation.

Furthermore, an attacker could actually perform this attack through self-liquidation, as a `minOutputAmount` is passed to the `prepareForLiquidation` function as well.

Recommendation

When a user attempts to initiate a deposit or withdrawal, verify whether the account is liquidatable. If so, prevent the action from being initiated. An edge case exists such that a user may becoming liquidatable when they already having a pending deposit/withdrawal, but this solution will prevent the continuous, malicious use of pending deposits/withdrawals to prevent liquidation.

Furthermore, consider restricting the `minOutputAmount` a liquidator can pass to prevent liquidation delay through self-liquidation.

Resolution

Dolomite Team: The issue was resolved in commits 843181e and bf66738.

Guardian: Ensure the `minOutputAmount` on `_params.extraData` is validated with function `_checkMinAmountIsNotTooLarge`. Furthermore, do not allow liquidatable users to initiate a wrapping and clearly document this behavior.

Additionally, validate that `_params.extraData` is 32 bytes long to prevent a malicious actor from padding the bytes and causing an OOG error when copying it into memory.

GMXL-1 | Withdrawals Apply _minOutputAmount To One Side

Category	Severity	Location	Status
Logical Error	High	GmxV2Library.sol: 151, 152	Resolved

Description

In the executeInitiateUnwrapping function the withdrawalParams are created with a _minOutputAmount that can only apply to either the shortToken output or the longToken output.

However withdrawals in GMX remove a split of the shortToken and longToken. The resulting shortToken and longToken are then subjected to the longTokenSwapPath and shortTokenSwapPath, the outputs of which are validated against the minLongTokenAmount and minShortTokenAmount respectively.

Therefore even though the outputs are swapped to the same token, take for example the short token, the short token that is directly removed from GMX will be validated against the entire _minOutputAmount and the short token that is received from the portion that was removed as the longToken and swapped to the shortToken will be validated against a minShortTokenAmount of 0.

This results in the withdrawal likely failing the minimum output validation as the portion of short token that is directly removed from GMX is unlikely to solely pass the _minOutputAmount validation. Additionally, there can be no minimum output that applies to the portion of funds that are swapped which is exactly the portion that ought to be validated.

Recommendation

Refactor the _minOutputAmount logic such that the minimum output can be split amongst the minLongTokenAmount and minShortTokenAmount.

Resolution

Dolomite Team: The issue was resolved in commit 5963990.

GMXL-2 | Redemptions Incorrectly Appear Unpaused

Category	Severity	Location	Status
Logical Error	High	GmxV2Library.sol: 236 - 239	Resolved

Description

The function isExternalRedemptionPaused is used to determine if GM withdrawals are currently paused, utilizing the current PnL-to-Pool Factors as one validation for a paused state.

The validation only compares against the maxPnlForAdl and maxPnlForWithdrawals with isLong = true, although the threshold factor may be different with isLong = false. This is incorrect as the shortPnlToPoolFactor should be compared against the MAX_PNL_FACTOR_FOR_WITHDRAWALS for isLong = false as in MarketUtils.validateMaxPnl

Furthermore, the condition verifies that the shortPnlToPoolFactor or the longPnlToPoolFactor should not exceed the maxPnlForWithdrawals, as GMX will revert on withdrawal execution when the threshold is passed. However, the check also looks at the maxPnlForAdl, which does not impact withdrawal execution on GMX.

As a result, redemptions may appear possible when the PnL-to-Pool Factor exceeds both the maxPnlForWithdrawals and maxPnlForAdl, although that is not the case and the withdrawal will fail.

Additionally, GMX can disable withdrawal creation, execution, or a market entirely which should also be included to verify that redemptions are paused. Because the market appears unpaused, users can modify their collateralization when liquidations aren't possible, or zap into more of the irredeemable GM tokens across Dolomite.

Recommendation

Fetch the maxPnlForWithdrawals with both isLong = true and isLong = false
For both long and short thresholds, update the validation to only check whether the shortPnlToPoolFactor or longPnlToPoolFactor exceeds the maxPnlForWithdrawals:

```
bool isShortPnlTooLarge = shortPnlToPoolFactor > int256(maxPnlForWithdrawalsShort);
bool isLongPnlTooLarge = longPnlToPoolFactor > int256(maxPnlForWithdrawalsLong);
```

In addition, verify that the market is enabled and withdrawal features are enabled through the Datastore.

Resolution

Dolomite Team: The issue was resolved in commit 11b095a.

AIMUTI-1 | Severely Undercollateralized Positions Cannot Be Liquidated

Category	Severity	Location	Status
Logical Error	High	AsyncIsolationModeUnwrapperTraderImpl.sol: 184-198	Resolved

Description PoC

During liquidation, an issue appears when the total available user amount to liquidate is equal to the input liquidation amount. The input liquidation amount is passed to the `liquidate` function from the `LiquidatorProxyV4WithGenericTrader` contract when liquidating a position.

Liquidation always creates a second set of call and sell actions. When the amounts are equal, the second call and sell action will be executed with 0 as input. This results in execution failing because the first set of actions clears the withdrawal position.

It is also worth mentioning that the account is not vaporizable in this state since it still holds a positive balance in the GM market.

Recommendation

In the `createActionsForUnwrapping` function from `AsyncIsolationModeUnwrapperTraderImpl` do not create a second call and sell action if the difference between the input amount and available amount is zero.

If the mentioned solution is implemented, 2 dummy actions that do not have any side-effects should be created as a workaround. This is needed to maintain compatibility with the liquidation proxy, which at this point already creates an action array using the length 4 for the liquidation unwrapping.

A different solution can be modifying the `LiquidatorProxyV4WithGenericTrader` to determine if this would be a 2 or 4 step liquidation.

Resolution

Dolomite Team: The issue was resolved in commit f434e31.

AIMUTI-2 | Withdrawal Keys Misused by Differing Subaccount in Liquidations

Category	Severity	Location	Status
DoS	High	AsyncIsolationModeUnwrapperTraderImpl.sol: 133-145	Resolved

Description PoC

When a liquidation is executed on an account from a vault with multiple accounts, a malicious actor can pass the withdrawal key belonging to an account that is different from the one being liquidated and block the vault.

Consider the scenario where a vault has accounts A and B:

By liquidating account A using account B's key, account B's withdrawal information is cleared. If account B has a withdrawal that needs to be retried, the execution will fail as the stored withdrawal information is empty.

Even if account B becomes liquidatable and uses account A's key to perform the liquidation, that will fail because `DolomiteMargin` would interpret that as a borrow increase in an unborrowable market. This happens when account A's withdrawal is for a larger amount that account B's.

The vault remains frozen and all operations from any sub-accounts are blocked. Such a hijacked liquidation can occur when both withdrawals have the same output token and are both retryable. The withdrawal can be retryable from either an on-going liquidation or from a failed withdrawal of a healthy position.

Recommendation

In the `_callFunction` function, verify that the stored `accountNumber` matches the `_accountInfo.number` but only if the call action was sent from a liquidation operation.

When sent from a liquidation operation, the `_accountInfo.number` variable holds the liquidatable account but when sent from a normal unwrapping it is the ZAP account number.

Resolution

Dolomite Team: The issue was resolved in commit f434e31.

VAULT-1 | Vault Owner Can Cancel Liquidation

Category	Severity	Location	Status
Access Control	Medium	GmxV2IsolationModeTokenVaultV1.sol: 91	Resolved

Description

In the `cancelWithdrawal` function, the vault owner can cancel liquidations which is essentially a withdrawal of the GM tokens. When a liquidator uses `prepareForLiquidation`, they trigger a forced withdrawal from the underwater vault.

The issue arises as the `cancelWithdrawal` function doesn't distinguish between user-initiated withdrawals and forced withdrawals (like liquidations). This allows a user to repeatedly cancel withdrawal attempts, causing a loss of funds for both the protocol (insolvency) and the liquidator as only the first liquidation execution fee is covered by the user.

Recommendation

Differentiate between normal withdrawals and those from liquidation. Restrict the vault owner from calling the `cancelWithdrawal` function when there is a pending withdrawal initiated by the `prepareForLiquidation` function.

Resolution

Dolomite Team: The issue was resolved in commit 20b002e.

GMXO-1 | Withdrawal Not Necessarily 50-50

Category	Severity	Location	Status
Logical Error	Medium	GmxV2MarketTokenPriceOracle.sol: 167-168	Resolved

Description

When calculating the swap price impact, it is assumed that withdrawing GM tokens provides 50% in the short token and the other 50% in long tokens. According to documentation, "Assume under the worst case, we liquidate 10% of the supply cap (which would entail a swap for half of that, 5%, to USDC (short token))."

However, that is not necessarily the case because long and short tokens are withdrawn with their value relative to the total pool value e.g. if the total pool value is $100 and $80 is from the long token, 80% of the withdrawn value will be in long tokens.

This assumption leads to an inaccuracy in the resulting price impact calculation, affecting the calculated price of the GM token in the `_getGmTokenPriceAfterPriceImpact` function.

Recommendation

Consider using the reader to get the current token ratios in the market and adjust the `wethAmountIn` accordingly.

Resolution

Dolomite Team: We have decided to no longer measure price impact but increase the liquidation penalty instead.

UAIWT-1 | Excess GM Not Partially Deposited On Supply Cap

Category	Severity	Location	Status
Logical Error	Medium	UpgradeableAsyncIsolationModeWrapperTrader.sol: 415-421	Resolved

Description

After a deposit is created, any extra GM tokens that are received are then deposited into Dolomite Margin for the user. If this amount would equal or surpass the maximum allowed value for a market, then it is entirely sent to the vault owner.

This is done in the `_depositIntoDefaultPositionAndClearDeposit` function from the `UpgradeableAsyncIsolationModeWrapperTrader` contract.

The issue is that if the maximum is exceeded, then the entire excess is wrongly sent to the the vault owner, instead of only the difference that causes the `maxWei` to be exceeded. The user may temporarily miss out on borrowing power as even a slight excess over cap leads to transferring the whole amount to the vault owner.

Recommendation

Modify the `_depositIntoDefaultPositionAndClearDeposit` function so that it sends only the excess that would not fit into the market to the vault owner and deposit the rest into Dolomite in the user's account.

Resolution

Dolomite Team: The issue was resolved in commit a490adb.

GMXUT-1 | Can't Unfreeze Vault If Execution Interrupted

Category	Severity	Location	Status
DoS	Medium	GmxV2IsolationModeUnwrapperTraderV2.sol: 124	Resolved

Description

During the unwrapping process, the vault is frozen by incrementing the mapping `_vaultToPendingAmountWeiMap` by `_amountDeltaWei.value`.

Once unwrapping concludes, the `_vaultToPendingAmountWeiMap` function is reduced by `_amountDeltaWei.value`, effectively unfreezing the vault.

However, the unwrapping process may fail, as acknowledged in the `afterWithdrawalExecution` function:

```
// @audit:  If GMX changes the keys OR if the data sent back is malformed (causing the above requires to
//          fail), this will fail. This will result in us receiving tokens from GMX and not knowing who they
//          are for, nor the amount. The only solution will be to upgrade this contract and have an admin
//          "unstuck" the funds for users by sending them to the appropriate vaults.
```

In the event of a failure in the `afterWithdrawalExecution` function, the protocol can recover the funds but is unable to unfreeze the vault. Consequently, the user remains unable to utilize their vault, including unwrapping any remaining funds.

Recommendation

Enable the Admin to invoke the `setVaultAccountPendingAmountForFrozenStatus` function, providing a means to unfreeze an account if execution is ever interrupted.

Resolution

Dolomite Team: The issue was resolved in commit cfd0a09.

GLOBAL-2 | Lack Of Liquidation Incentives

Category	Severity	Location	Status
Incentives	Medium	Global	Resolved

Description

According to the Dolomite whitepaper, “Liquidations forcefully repay any debt that is owed by a borrower by transferring an equivalent amount of collateral from the borrower to the liquidator, plus a liquidation penalty of 5%.”

The penalty is used as a reward for performing the liquidation and maintaining protocol solvency. However, the integration lacks a reward for liquidations in the modules, leaving no incentive for a user to trigger the `prepareForLiquidation` function if the unwrapping and swap into the Core protocol succeeds.

Recommendation

Consider providing the liquidator a reward in the output token for the liquidation.

Resolution

Dolomite Team: The issue was resolved in commit 54ab211.

GMXL-3 | Withdrawals Fail When A Backing Token is Zero

Category	Severity	Location	Status
Logical Error	Medium	GmxV2Library.sol: 295-300	Partially Resolved

Description

The `outputToken` and `secondaryOutputToken` are always validated to be equal after the execution of a withdrawal:

`_outputTokenAddress.value == _secondaryOutputTokenAddress.value`

However, the GMX `SwapUtils.swap` function does not alter the resulting token in the case that the input into the swap is 0. An input of 0 can occur if the GMX market is one-sided at the point of withdrawal execution e.g. market only has 1 ETH and no USDC deposited.

Another scenario this may occur in is if the amount being withdrawn is very small. This ultimately means that the validation will fail and the tokens will be stuck in the unwrapper trader.

Recommendation

Modify the check such that if the value of the withdrawal output amount is 0, then the `_outputTokenAddress` and the `_secondaryOutputTokenAddress` do not have to match.

Resolution

Dolomite Team: The issue was resolved in commit 0a8e6d3.

Guardian: Compare `_withdrawalInfo.outputToken` against the `_outputTokenAddress` if the requested output token is the long token. Otherwise compare it against the `_secondaryOutputTokenAddress`. Afterwards, if the other token's output is non-zero, ensure the two token addresses match.

UAIWT-2 | Fund Transfer From Wrapper Trader Can Be Skipped

Category	Severity	Location	Status
Logical Error	Medium	UpgradeableAsyncIsolationModeWrapperTrader.sol: 351	Resolved

Description

If a user receives more GM than the `minOutputAmount` they set, the excess GM has to be deposited into the Core protocol such that the Vault balance and Core balance align. However, this state assumes `_shouldSkipTransfer` has been set to `false` upon deposit creation in the call to function `IsolationModeTokenVaultV1WithFreezable.executeDepositIntoVault`:

```
else {
  Require.that(
    isVaultFrozen(),
    _FILE,
    "Vault should be frozen"
    );
   _setShouldVaultSkipTransfer(/* _shouldSkipTransfer = */ false);
}
```

It is possible to overwrite this pre-requisite state and set `_shouldSkipTransfer = true` through the function `UpgradeableAsyncIsolationModeUnwrapperTrader.callFunction` when the sender is an operator.

In this case, once the deposit is resolved and the `afterDepositExecution` callback is triggered, the fund transfer into the Vault would be skipped and the funds would remain stuck inside the wrapper trader.

Recommendation

Prior to calling `factory.depositIntoDolomiteMarginFromTokenConverter`, explicitly `_setShouldVaultSkipTransfer(/* _shouldSkipTransfer = */ false);`

Resolution

Dolomite Team: The issue was resolved in commit 0a8e6d3.

GMXO-2 | Potentially Misleading PnL Factor

Category	Severity	Location	Status
Oracle Risk	Medium	GmxV2MarketTokenPriceOracle.sol: 224	Resolved

Description

In the `GmxV2MarketTokenPriceOracle` contract the call to function `getMarketTokenPrice` uses the `MAX_PNL_FACTOR_FOR_WITHDRAWALS` PnL type to read the market token price from GMX. This is typically the most constrictive PnL Type, such that trader profit is capped to the smallest amount relative to the `MAX_PNL_FACTOR_FOR_TRADERS` and `MAX_PNL_FACTOR_FOR_DEPOSITS`.

Consequently, the resulting price of the market token will be higher when measured using the more constrictive `MAX_PNL_FACTOR_FOR_WITHDRAWALS`. This will ultimately cause a user's collateral to have a greater value than if the another type was used.

Out of an abundance of caution, it may be preferable to use the less constrictive `MAX_PNL_FACTOR_FOR_DEPOSITS`, so that the collateral is not optimistically valued by capping the PnL to a lower amount.

Recommendation

Consider using the less constrictive `MAX_PNL_FACTOR_FOR_DEPOSITS` to read the price of the GM token.

Resolution

Dolomite Team: The issue was resolved in commit 990d726.

GLOBAL-3 | Vault Frozen On Single Account

Category	Severity	Location	Status
Logical Error	Medium	Global	Resolved

Description

When a subaccount creates a deposit or withdrawal, the Vault is frozen to prevent misuse such as borrowing when the underlying funds are not present, putting the protocol at risk. While the Vault is frozen, all other subaccounts are unable to perform any actions, including depositing, withdrawing, and borrowing.

This poses a potential problem as a deposit or withdrawal may take a prolonged time to be executed, and the order cannot be cancelled for the `MIN_ORACLE_BLOCK_CONFIRMATIONS`. During this period, a subaccount that is close to liquidation is unable to deposit into the protocol and save their position.

Recommendation

Consider allowing users to deposit and withdraw if another subaccount is frozen, but not borrow. Otherwise, explicitly document to users that if one account is in a frozen state, all other accounts cannot perform Vault actions.

Resolution

Dolomite Team: The issue was resolved in commits 784b629 and f434e31.

GMXO-3 | Liquidator Can Force Liquidation

Category	Severity	Location	Status
Protocol Manipulation	Medium	GmxV2MarketTokenPriceOracle.sol: 183-187	Resolved

Description

When calculating the price of GM, the value is adjusted down by any negative price impact upon withdrawal. A liquidator can shift the price down on a position that is near liquidation by putting capital into GMX. This will alter the price that is calculated, and put the user in a liquidatable state.

A liquidator can use this to have first access to liquidating the user and get a unfair advantage compared to the other liquidators. Note that this manipulation can be done by non-liquidators as well to grief other users.

Recommendation

Document the behavior of GM's pricing to users and carefully monitor the pricing for manipulation. Furthermore, utilize higher liquidity pools to minimize price impact.

Resolution

Dolomite Team: We have adjusted the oracle to no longer user price impact.

FIVF-1 | Liquidation Needs To Match Pending Output Token

Category	Severity	Location	Status
Validation	Medium	FreezableIsolationModeVaultFactory.sol: 146-169	Acknowledged

Description

The `expectedConversionToken` validation is meant to ensure that the values of two different tokens aren't added and subtracted in the `_accountInfoToPendingAmountWeiMap` and `_vaultToPendingAmountWeiMap` mappings.

Due to this check, liquidations will fail if the `outputToken` does not match the `outputToken` on a pending deposit/withdrawal. Forcing liquidations to use a specific `outputToken` can lead to less long/short token being withdrawn on redemption by experiencing negative price impact when swapping to that particular `outputToken`.

Furthermore, because the Oracle assumes liquidations are typically performed from long token to short token, a user could further exacerbate the price impact mispricing by forcing a liquidation from short to long token instead.

Recommendation

Reconsider if the `expectedConversionToken` check is even necessary. The `_amountDeltaWei.value` is always in GM, so subtracting two different token values should not occur. However, this would require a change to the `_accountInfoToOutputTokenMap` as an account could be experiencing two different conversion tokens.

Resolution

Dolomite Team: Acknowledged.

AITB-1 | Unbounded Execution Fee For Deposits and Withdrawals

Category	Severity	Location	Status
Validation	Medium	AsyncIsolationModeTraderBase.sol	Resolved

Description

After deposit or withdrawal execution, the excess execution fee is refunded. However, this refund is inaccessible by the user initiating the wrapping or unwrapping, but rather held by the Dolomite Margin owner.

This can potentially cause asset loss as there are no limits on how much `msg.value` a user can forward as the execution fee. A user may prefer to first deposit for GM directly through GMX, as they are assured that they don't lose native tokens unnecessarily.

Recommendation

If the attribution of gas refunds is too constrictive with current size limits, consider bounding how much `msg.value` a user forwards for GMX execution.

Resolution

Dolomite Team: The issue was resolved in commit c1949b8.

GLOBAL-4 | Sequencer May Experience Outages

Category	Severity	Location	Status
Logical Error	Medium	Global	Acknowledged

Description

While the Arbitrum sequencer is down it is possible for a users position to go from healthy to undercollateralized. During this time the average user will not be able to rescue their position as they will not be able to submit orders directly through Arbitrum.

However, most liquidators will be automated and would be sophisticated enough to submit liquidation transactions through the delayed inbox on L1. When the sequencer is back online the transactions submitted through the delayed box will be executed first, meaning the position will be liquidated before the users have a chance to rescue their position.

Recommendation

Consider adding a grace period after outages to allow users some time to save their position when the sequencer is back online.

Resolution

Dolomite Team: Acknowledged.

GLOBAL-5 | Features May Be Disabled

Category	Severity	Location	Status
Logical Error	Low	Global	Resolved

Description

GMX may choose to disable certain features of its protocol, including but not limited to deposit execution. If this feature was paused, a Dolomite user would be able to create a deposit even when execution of any deposits isn't occurring.

As a result, a deposit will be created, unexecuted, and unable to be cancelled for the `MIN_ORACLE_BLOCK_CONFIRMATIONS` period. The user will face a period where their funds are inaccessible as a result.

Recommendation

Consider disallowing initiating wrappings when deposit creation or execution is disabled on GMX. Otherwise, clearly document this behavior to users and monitor when features are disabled.

Resolution

Dolomite Team: The issue was resolved in commit 38e9e52.

GMXL-4 | Inefficient Execution Fee Handling

Category	Severity	Location	Status
Superfluous Code	Low	GmxV2Library.sol: 85-88	Resolved

Description

The GMX V2 system accepts and stores the execution fee as wrapped native tokens. The `exchangeRouter.sendWnt` function simply wraps the `msg.value` sent into native tokens and transfers them to the vault. Therefore it is unnecessary to unwrap the wrapped native tokens only for the `sendWnt` function to wrap them again.

Recommendation

Transfer the execution fee into the GMX V2 system with the `exhangeRouter.sendTokens` function.

Resolution

Dolomite Team: The issue was resolved in commit 6d1f1d8.

UAIWT-3 | Incorrect maxWei Limit Bypass Check

Category	Severity	Location	Status
Logical Error	Low	UpgradeableAsyncIsolationModeWrapperTrader.sol: 415	Resolved

Description

When receiving excess GM after a deposit, if it surpasses or is equal to the maximum allowed value for a market, then it is entirely sent to the vault owner. The check incorrectly includes equality with maximum WEI, since Dolomite allows deposits up to the maximum WEI, but not exceeding it.

Recommendation

Modify the comparison in the if of the `_depositIntoDefaultPositionAndClearDeposit` function from `>= maxWei` to `> maxWei`.

Resolution

Dolomite Team: The issue was resolved in commit 6d1f1d8.

GMXL-5 | Redundant Self Import

Category	Severity	Location	Status
Superfluous Code	Low	GmxV2Library.sol: 24	Resolved

Description

In the `GmxV2Library.sol` file, the library itself is reimported redundantly.

Recommendation

Remove the self import from line 24.

Resolution

Dolomite Team: The issue was resolved in commit 6d1f1d8.

GMXO-4 | GM Price Impact May Be Misrepresented

Category	Severity	Location	Status
Oracle Risk	Low	GmxV2PriceOracle.sol: 173-180	Resolved

Description

The GM oracle calculates the price impact to appropriately adjust GM's price when the price impact is negative. However, the calculation is done using the short token as the output amount.

The short token being the output is not always the case, as a user can liquidate with the output token as the long token, which may result in an entirely different price impact than measured.
As a result, GM's resultant price will be misrepresented.

However, Dolomite does note that liquidations will generally be into USDC which is why price impact is measured from long to short token.

Recommendation

Consider adjusting the `_getAdjustedAccountValues` logic to take into account the output token during a liquidation and/or documenting this behavior to users.

Resolution

Dolomite Team: We have decided to remove the price impact calculation and instead increase the liquidation penalty.

GTPB-1 | Incorrect Interface Used

Category	Severity	Location	Status
Typo	Low	GenericTraderProxyBase.sol: 354-355	Resolved

Description

When calculating the actions length, the `traderType` is `IsolationModeWrapper` but the `IIsolationModeUnwrapperTrader` interface is used.

Recommendation

Use the `IIsolationModeWrapperTrader` interface for consistency.

Resolution

Dolomite Team: The issue was resolved in commit 297f43c.

UAIWT-4 | Dolomite Assumes Owner Can Handle GM

Category	Severity	Location	Status
Documentation	Low	UpgradeableAsyncIsolationModeWrapperTrader.sol: 422	Resolved

Description

In the case that the deposit of excess GM into the borrow account fails, and the market supply caps are exceeded, Dolomite transfers those GM tokens directly to the vault owner.

However, a Vault can be created for an arbitrary owner and Dolomite assumes that the owner can handle GM. A potential problem arises if the Vault's account is a contract which cannot support the transfer of GM, and those tokens will be locked.

Recommendation

Document to users this behavior to prevent unexpected loss of funds.

Resolution

Dolomite Team: The issue was resolved in commit d247fb8.

UAIWT-5 | Inaccurate File Name

Category	Severity	Location	Status
Typo	Low	UpgradeableAsyncIsolationModeWrapperTrader.sol: 52	Resolved

Description

The `_FILE` name is set to "IsolationModeWrapperTraderV2" which is not the actual name of the file. This differs from the standard within the `UpgradeableAsyncIsolationModeUnwrapperTrader.sol` contract where `_FILE = "UpgradeableUnwrapperTraderV2"`

Recommendation

Change it to `bytes32 private constant _FILE = "UpgradeableWrapperTraderV2";`

Resolution

Dolomite Team: The issue was resolved in commit 6d1f1d8.

GMXO-5 | Typo

Category	Severity	Location	Status
Typo	Low	GmxV2MarketTokenPriceOracle.sol: 163	Resolved

Description

There is a typo in the comment: “there’s on cap” should be “there’s no cap”.

Recommendation

Fix the typo as described above.

Resolution

Dolomite Team: The issue was resolved in commit 6d1f1d8.

GMXO-6 | Inaccurate Comment

Category	Severity	Location	Status
Documentation	Low	GmxV2MarketTokenPriceOracle.sol: 45	Resolved

Description

In the price oracle the documentation states: "/// @dev All of the GM tokens listed have, at-worst, 20 bp for the price deviation".

However, this conflicts with the constant `PRICE_DEVIATION_BP` which is set to 25 bp.

Recommendation

Update the comment to reflect the 25bp price deviation.

Resolution

Dolomite Team: The issue was resolved in commit 6d1f1d8.

GLOBAL-6 | Wrapping Fails With Zero Output

Category	Severity	Location	Status
Documentation	Low	Global	Resolved

Description

The call to function `swapExactInputForOutput` fails if the `_minOutputAmountWei` is 0. As a result, the user is unable to swap their tokens into market tokens without setting the `minOutputAmount` of market tokens, which is not a requirement for GMX deposits.

Recommendation

Clearly document the behavior that a user is required to specify a non-zero `minOutputAmount` for a deposit to be created.

Resolution

Dolomite Team: The issue was resolved in commit d247fb8.

GLOBAL-7 | Liquidations Cannot Include Pending Keys

Category	Severity	Location	Status
Documentation	Low	Global	Acknowledged

Description

While a deposit or withdrawal is pending, the order is yet to be marked as retryable. If a liquidation were to be triggered through the `LiquidatorProxyV4WithGenericTrader` contract with a pending key, it will ultimately call function `createActionsForWrapping` and fail due to the retryable validation.

This is important to note as multiple keys can be passed for a liquidation through the `ZapParams`, and one failing will prevent the other liquidations from occurring.

Recommendation

Clearly document that liquidations should exclude pending deposits/orders that are not marked as retryable to prevent liquidation failure.

Resolution

Dolomite Team: Acknowledged.

GMXIVF-1 | Swap-only GMX Markets Are Unsupported

Category	Severity	Location	Status
Unsupported Feature	Low	GmxV2IsolationModeVaultFactory.sol: 84	Resolved

Description

GMX supports markets which are solely used for swapping and no trading is allowed. In such markets, the index token is `address(0)`.

When the `VaultFactory` is being constructed, `INDEX_TOKEN_MARKET_ID = DOLOMITE_MARGIN().getMarketIdByTokenAddress(INDEX_TOKEN);` calls DolomiteMargin to fetch the `marketId` for the token, but it will revert on `Getters._requireValidToken` due to the zero address being the parameter.

Vaults cannot be created for swap-only market tokens and users will not be able to use these tokens as collateral for their borrow positions.

Recommendation

Explicitly document that swap-only markets will not be supported, or add extra handling when the index token is empty to prevent a revert when fetching the `marketId` and the Chainlink price for that market.

Resolution

Dolomite Team: The issue was resolved in commit d247fb8.

VAULT-2 | Stored Liquidation Fee Too Insufficient

Category	Severity	Location	Status
Incentives	Low	GmxV2IsolationModeTokenVaultV1.sol: 128-138	Acknowledged

Description

In order to open a borrow position on the GMX vault, users need to also submit in advance an execution fee that will be used in case of liquidation.

If for various reasons, such as network clogging, the fee must be increased, then issues appear because all existing opened positions will most likely not have enough fees deposited to cover liquidation. In this case liquidators would need to add the extra fee themselves, making the position less desirable to liquidate.

Recommendation

One solution is to have the wrapper provide the difference, where the team deposits the amount directly to it after increasing the fees.

Another solution is that the team perform these liquidations at a loss to themselves. This solution is a bit more gas intensive then the first suggestion but does not require the calculation of a deposit amount for all existing borrowing positions in advance.

Resolution

Dolomite Team: In this case, Dolomite will act as the liquidator of last resort and cover the cost ourselves.

GMXO-7 | Only Fee For Negative Price Impact Is Read

Category	Severity	Location	Status
Oracle Risk	Low	GmxV2MarketTokenPriceOracle.sol: 107	Resolved

Description

When reading the swap fee from GMX, `forPositiveImpact` is always set as `false`. This is okay as the fee is ultimately always valued with `forPositiveImpact = false` when withdrawing in GMX.

However, it should be made clear that the `getFeeBpByMarketToken` function will only return the largest swap fee, which is the one considering negative price impact.

Recommendation

Consider adding documentation for function `getFeeBpByMarketToken` so that it is clear it does not consider the positive price impact swap fee.

Resolution

Dolomite Team: The issue was resolved in commit 8dc257b.

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