GUARDIAN AUDITS

# SMART CONTRACT SECURITY AUDIT OF

# GMX

# Summary

**Audit Firm** Guardian

**Prepared By** Owen Thurm, Daniel Gelfand, 0xKato

**Client Firm** GMX

**Final Report Date** November 27, 2023

## Audit Summary

GMX engaged Guardian to review the security of the GMX V1 system. From the 10th of November to the 27th of November, a team of 3 auditors reviewed the source code in scope. All findings have been recorded in the following report.

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/GMXV1

# Table of Contents

# Project Overview

## Project Summary

| | |
|---|---|
| Project Name | GMX |
| Language | Solidity |
| Codebase | https://github.com/gmx-io/gmx-contracts |
| Commit(s) | 7461d1bf5c1d08f1e758f0b32f22b86d73ba7e4b |

## Audit Summary

| | |
|---|---|
| Delivery Date | November 27, 2023 |
| 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 | 3 | 3 | 0 | 0 | 0 | 0 |
| Medium | 10 | 10 | 0 | 0 | 0 | 0 |
| Low | 4 | 4 | 0 | 0 | 0 | 0 |

# 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.

# Findings & Resolutions

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| VEST-1 | Vested Amounts Increase Vesting Rate | Logical Error | High | Pending |
| GLOBAL-1 | Lending Protocols Exploited With GLP Mispricing | Protocol Manipulation | High | Pending |
| VLT-1 | Increased Insolvency Risk | Insolvency | High | Pending |
| VEST-2 | bonusRewards Lost During Account Transfer | Logical Error | Medium | Pending |
| VLTU-1 | Multiple Swaps More Favorable Than A Large Swap | Protocol Manipulation | Medium | Pending |
| RWTR-1 | Lost Rewards When All Users Unstake | Logical Error | Medium | Pending |
| VLT-2 | Lacking Liquidation Incentive | Incentives | Medium | Pending |
| VLT-3 | Blacklisted Addresses May Manipulate The Exchange | Blacklists | Medium | Pending |
| VPFEED-1 | Improper Aggregator Usage | Oracle Integration | Medium | Pending |
| GLPM-1 | cooldownDuration Bypassed | Protocol Manipulation | Medium | Pending |
| VLT-4 | liquidationFee Always Covered By GLP Holders | Logical Error | Medium | Pending |
| VLT-5 | Inaccurate Fee Amount Emitted | Logical Error | Medium | Pending |
| PRTE-1 | Users Unable To Cancel Their Orders If Leverage Disabled | Logic Error | Medium | Pending |

# Findings & Resolutions

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| VPFEED-2 | ammPrice Manipulation | Protocol Manipulation | Low | Pending |
| VLT-6 | Loss Amounts Rounded Down | Rounding | Low | Pending |
| VLT-7 | Misleading hasRealisedProfits Variable | Documentation | Low | Pending |
| OBOOK-1 | Unnecessary USDG Logic | Superfluous Code | Low | Pending |

# VEST-1 | Vested Amounts Increase Vesting Rate

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | High | Vester.sol: 356 | Pending |

## Description PoC

The `_getNextClaimableAmount` function relies on the vestedAmount to compute the additional `claimableAmount`. However the `vestedAmount` includes tokens that were previously vested, therefore the vesting rate is inreased for these previously vested tokens.

For example, if a user had previously vested 1,000 esGMX in year 1, they may deposit another 1,000 esGMX to vest in year 2 and vest these 1,000 tokens at a rate of 2,000 tokens / year. Therefore the user can fully vest their 1,000 tokens in only 6 months.

This perturbs the tokenomics of esGMX and allows accounts with large previously vested amounts to shrink or ultimately effectively bypass the vesting period.

## Recommendation

Consider refactoring the vesting rate logic such that the vesting rate of previously deposited tokens cannot effect the vesting rate of newly deposited tokens.

Perhaps by creating a Vest struct which contains the vesting information for a single deposit, and allowing users to create new Vests upon new deposits.

## Resolution

# GLOBAL-1 | Lending Protocols Exploited With GLP Mispricing

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Protocol Manipulation | High | Global | Pending |

## Description PoC

Users may decrease positions by directly calling the `Vault` contract which will avoid the global short average price update in the `ShortsTracker` contract. When a user decreases the size of their short position this ultimately leads to a mis-representation of the pending PnL for shorts, as the user's decrease has not been taken into account for the global short tracking.

As a result the pricing of GLP will temporarily factor in the user's PnL twice. Therefore if a user closes a short position that was in profit directly from the vault, the GLP price will see a stepwise decrease as the user's realized PnL is not erased from the pending PnL tracked through the `ShortsTracker`.

This stepwise decrease in the price of GLP poses a risk to any lending protocols using GLP as collateral. At minimum this could be leveraged to inflict bad debt on a protocol in a griefing attack.

In lending protocols with a "recovery mode" functionality, similar to Liquity, an attacker may be economically incentivized to cause this GLP mispricing and gain from it by triggering "recovery mode" on the lending protocol and being the first to liquidate many borrowing positions in a single transaction.

## Recommendation

Ensure there are no lending protocols accepting GLP as collateral that have a "recovery mode" or similar feature which could make this attack economically viable. And consider restricting the `decreasePosition` function on the Vault such that users cannot call it directly and must update the ShortsTracker every time.

## Resolution

# VLT-1 | Increased Insolvency Risk

| Category | Severity | Location | Status |
|---|---|---|---|
| Insolvency | High | Vault.sol | Pending |

## Description

Long positions are required to deposit the index token as collateral, and the value of that collateral is frozen upon sending it into the system. Therefore when the price of the index token decreases, the value of the user's collateral tokens deposited may become insufficient to cover their losses, however as the user's collateral balance was frozen upon position increase, the position may not be liquidated until it is deep in insolvency.

- The price of WETH is $5,000, the allowed maxLeverage is 20x.
- User A opens a 2x leverage position on WETH with 1 WETH as collateral.
- The price of WETH falls to $3,000
- User A's deposited 1 WETH collateral token is valued at $3,000
- User A's losses are $4,000
- User A's leverage is $10,000 / ($5,000 - $4,000) = 10x, therefore User A's position is not liquidatable.
- However user A's *real* leverage, considering the collateral they provided to the vault (or the vault *bought* from User A) is $10,000 / ($3,000 - $4,000) = -10x, the position is insolvent when considering the collateral provided by the user and cannot cover its losses.

The value of user A's collateral tokens deposited is only $3,000, which is not enough to cover the losses of the position. The position is insolvent, but technically not liquidatable. Therefore when the position is closed, the outstanding losses are eaten by the GLP pool. This way liquidations fail to effectively protect the protocol from insolvent positions.

## Recommendation

The purpose of liquidations is to protect LPers from taking on the losses of insolvent traders, freezing the value of a user's collateral at the time of deposit does not maintain this. Consider refactoring the leverage calculation such that the current value of the user's deposited collateral is taken into account, ultimately liquidating positions far before they can become insolvent.

## Resolution

# VEST-2 | bonusRewards Lost During Account Transfer

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | Medium | Vester.sol: 149 | Pending |

## Description

In the `Vester.transferStakeValues` function, the `bonusRewards` mapping value for the `_receiver` address is overwritten with the `bonusRewards` value from the `_sender` address.

Meanwhile in the `RewardRouterV2._validateReceiver` function, the receiver is not validated to have a `bonusRewards` value of 0.

Therefore the `_receiver` address may hold a nonzero `bonusRewards` amount, and this amount may be overwritten by the transfer.

## Recommendation

Add the `bonusRewards` from the sender to the receiver's bonus rewards rather than overwriting the values.

## Resolution

# VLTU-1 | Multiple Swaps More Favorable Than A Large Swap

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Protocol Manipulation | Medium | VaultUtils.sol: 145 | Pending |

## Description PoC

When a swap would cross the imbalance from more tokens than the `targetAmount` to less tokens then the target amount (or vice-versa), it is more favorable to use several smaller swaps than one large one.

This is because when a swap has a net balancing effect on the pool it is rewarded with a reduced `swapFee` where the reduction is based upon the size of the `initialDiff`.
However when a swap has a net imbalancing effect, even if a portion of the swap has a positive balancing effect, the entire averageDiff is treated as negative impact for the user.

## Recommendation

Consider refactoring the dynamic fee logic such that even when large swaps cross from one imbalanced side to another the fees are the same as if the user were to perform multiple smaller swaps.

## Resolution

# RWTR-1 | Lost Rewards When All Users Unstake

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | Medium | RewardTracker.sol: 273 | Pending |

## Description PoC

In the `_updateReward` function the distributor is forced to distribute pending rewards even if there are no staked tokens in the `RewardTracker`. In this case the pending rewards are not distributed to any users and are effectively lost.

These rewards may be rescued through the `withdrawToken` function, though these rewards ought to be either not distributed or saved until a user chooses to stake.

## Recommendation

Only invoke the `distributor.distribute()` function if there is a nonzero supply of staked tokens to distribute the rewards to. In this case there is an additional incentive for the first user to stake when the `totalSupply` is zero, as they will receive additional rewards that have been accumulating and not distributed.

If the additional incentive is not desired, consider allowing the `RewardTracker` contract to update the `lastDistributionTime` through the `updateLastDistributionTime` function, and update the `lastDistributionTime` without actually distributing the rewards when the totalSupply is 0.

## Resolution

# VLT-2 | Lacking Liquidation Incentive

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Incentives | Medium | Vault.sol: 720 | Pending |

## Description

When a position is solvently liquidated the `liquidationFeeUsd` is not paid to the `_feeReceiver` address, therefore there is no incentive for liquidators to liquidate users before they become insolvent. In fact the liquidator would prefer that the position become insolvent so that they may collect the `liquidationFeeUsd`.

At the moment liquidations are made by a trusted liquidator address, however the unfavorable incentive still applies to this liquidator as they stand to gain more in fees by allowing positions to become insolvent before liquidating them.

## Recommendation

Award a liquidation fee to the liquidator in all liquidation cases, especially to incentivize closing positions while they are still solvent.

## Resolution

# VLT-3 | Blacklisted Addresses May Manipulate The Exchange

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Blacklists | Medium | Vault.sol | Pending |

## Description PoC

If a user address is blacklisted for their collateral token, e.g. USDC, it is possible to use a potentially risk free trading strategy with a short and long position.

- User A is blacklisted for USDC.
- User A opens a short position backed by USDC and a long position backed by BNB, the index token for both is BNB and both positions have the same size.
- User A's short position cannot be liquidated until the position is insolvent, as any liquidation before this point would attempt to transfer USDC to the user.
- Therefore User A's losses on their short are capped to the amount of collateral for their position, however there is no cap for profits on their long.
- Price will gap over the short position's liquidation threshold and the delta between the liquidation's execution price and the user's insolvency price will be net profit for the user.

This trading strategy is however not guaranteed to be profitable due to position and borrowing fees, however it may prove to be profitable with higher amounts of leverage and in times of volatility.

## Recommendation

Keep the fees to a non-trivial amount and potentially raise them if someone is observed adopting this strategy.

## Resolution

# VPFEED-1 | Improper Aggregator Usage

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Oracle Integration | Medium | VaultPriceFeed.sol: 281, 309, 313 | Pending |

## Description

Throughout the `VaultPriceFeed` contract, the `latestAnswer` and `latestRoundData` functions of Chainlink aggregator feeds are consulted.

However the `latestAnswer` function is deprecated and the `latestRoundData` functions should use heartbeat checks to ensure price updates are occurring as expected.

## Recommendation

Use the latestRoundData functions with the appropriate heartbeat checks always.
https://docs.chain.link/data-feeds#check-the-timestamp-of-the-latest-answer

## Resolution

# GLPM-1 | cooldownDuration Bypassed

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Protocol Manipulation | Medium | GlpManager.sol: 234 | Pending |

## Description

In the `GlpManager` contract a `cooldownDuration` is imposed to prevent accounts from depositing and withdrawing from GLP in a short timeframe. However this validation is based upon the address of the `_account` who owns GLP.

A user may initiate an account transfer and transfer their staked GLP to a new account to be able to unstake and redeem this GLP in the same transaction in which the GLP was minted.

This may be leveraged with `GLOBAL-1` to allow for single-transaction arbitrages when the cooldown is nonzero and the price of GLP has a stepwise increase due to invalid shorts tracking.

## Recommendation

The `cooldownDuration` is configured to 0 in production, therefore single-transaction arbitrages of GLP stepwise price movements are possible by default. However even if the cooldown is configured it may be bypassed.

Be aware of this behavior, and consider updating the cooldown for receivers of account transfers.

## Resolution

# VLT-4 | liquidationFee Always Covered By GLP Holders

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | Medium | Vault.sol: 751, 752 | Pending |

## Description

In the `liquidatePosition` function there is an assumption that the liquidated amount is always sufficient to cover the liquidation fees. However lines 751 and 752 are only reachable when the position is insolvent and cannot cover their losses, `marginFees`, or the `liquidationFee`.

Therefore the liquidated amount can never cover the liquidation fee and this fee will always come from GLP holders' share of the `poolAmount`.

## Recommendation

Be aware that this assumption is invalid and be sure GLP holders are aware of the cost. Additionally, consider reducing the margin fees in order to cover the `liquidationFee` and an incentive for liquidators without charging the GLP holders.

## Resolution

# VLT-5 | Inaccurate Fee Amount Emitted

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | Medium | Vault.sol: 671, 680 | Pending |

## Description

In the `_reduceCollateral` function if the fee is larger than the `usdOut` then the fee is charged to the user's collateral rather than changing the `usdOutAfterFee`. In many cases the collateral will be reduced by the fee rather than the `usdOutAfterFee`. Such as a user decreasing a position at a loss without removing any collateral.

However in the `_decreasePosition` function the fee is assumed to always come from the `usdOutAfterFee` amount when the fee is computed for the `DecreasePosition` event emissions. This emitted fee will often errantly report that 0 fees were taken during the decrease, when in fact a nonzero fee amount was deducted from the collateral.

As a result any systems relying on this event emission will receive invalid data and could potentially be mislead.

## Recommendation

Either emit the fee that is charged in the `_reduceCollateral` function or return the actual fee amount that was charged from the `_reduceCollateral` function.

## Resolution

# PRTE-1 | Users Unable To Cancel Orders If Leverage Disabled

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Error | Medium | PositionRouter.sol: 623 | Pending |

## Description

In the event that the admin calls `setIsLeverageEnabled()` and sets `isLeverageEnabled` to false, users are no longer allowed to cancel their own orders due to `!isLeverageEnabled && !isKeeperCall { revert("403"); }` in `_validateExecutionOrCancellation`.

The users will need a keeper to cancel their order for them, which will result in either the keeper or the user losing the `executionFee`.

## Recommendation

Allow users to cancel their pending orders in the event that `isLeverageEnabled` is configured to false.

## Resolution

# VPFEED-2 | ammPrice Manipulation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Protocol Manipulation | Low | VaultPriceFeed.sol: 371 | Pending |

## Description

In the `VaultPriceFeed` contract the `ammPrice` is computed based upon the reserves of the DEX pair, however these values can be trivially manipulated in a single transaction with a flash loan. `isAmmEnabled`, is currently set to false so this risk can be safely ignored.

## Recommendation

Do not use the `ammPrice` feature under any circumstances as is.

## Resolution

# VLT-6 | Loss Amounts Rounded Down

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Rounding | Low | Vault.sol: 1007 | Pending |

## Description

In the `_reduceCollateral` function the `adjustedDelta` is rounded down even if the amount represents a loss.

In the event that the `adjustedDelta` variable represents a loss, the rounding down rounds in the trader's favor rather than the protocol's.

## Recommendation

Consider rounding up if the `adjustedDelta` amount represents a loss for the trader.

## Resolution

# VLT-7 | Misleading hasRealisedProfits Variable

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Documentation | Low | Vault.sol: 829 | Pending |

## Description

In the `getPosition` function the 6th entry in the returned tuple, which represents `hasRealisedProfits`, is true if `position.realisedPnl` is zero.

This may be misleading for systems relying on the `getPosition` function as a position can have been just opened and the `hasRealisedProfits` boolean will be true.

## Recommendation

Be sure to clearly document this potentially unexpected behavior.

## Resolution

# OBOOK-1 | Unnecessary USDG Logic

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Superfluous Code | Low | OrderBook.sol: 443 | Pending |

## Description

In the `validateSwapOrderPriceWithTriggerAboveThreshold` function USDG is treated as a token that is supported in the order swap path, however USDG cannot be traded in a swap order for several reasons:

1. Users have no way of obtaining USDG as it is solely held by the GlpManager contract.
2. USDG is not a whitelisted token in the vault contract, therefore any swap using USDG in the path would fail.
3. There is no price feed for USDG so any swap with USDG would fail to fetch an accurate price for the token.

## Recommendation

Consider removing the USDG logic from the `validateSwapOrderPriceWithTriggerAboveThreshold` function as it can never be used in a swap order.

## Resolution

# 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