GA GUARDIAN

GMX

Buybacks

Security Assessment

November 18th, 2024

Summary

Audit Firm Guardian

Prepared By Daniel Gelfand, Owen Thurm, Osman Ozdemir, Mark Jonathas

Client Firm GMX

Final Report Date November 18, 2024

Audit Summary

GMX engaged Guardian to review the security of its GMX buyback mechanism. From the 7th of October to the 14th of October, 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 4 High/Critical issues were uncovered and promptly remediated by the GMX team. Several issues impacted the fundamental behavior of the system, following their remediation Guardian believes the protocol to uphold the functionality described for the buyback mechanism.

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

🔗 Blockchain network: **Arbitrum, Avalanche**

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

Table of Contents

Project Overview

Project Summary

Project Name	GMX
Language	Solidity
Codebase	https://github.com/gmx-io/gmx-synthetics
Commit(s)	Initial commit: 7d3d1ff928b9e6f319c24f3ba9737e1bcd09532b Final commit: 2253e65800d9092712c3218b2185dd9e2440733f

Audit Summary

Delivery Date	November 18, 2024
Audit Methodology	Static Analysis, Manual Review, Test Suite

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	3	0	0	0	0	3
High	1	0	0	0	0	1
Medium	1	0	0	1	0	0
Low	8	0	0	6	0	2

Audit Scope & Methodology

Vulnerability Classifications

Severity	Impact: *High*	Impact: *Medium*	Impact: *Low*
Likelihood: *High*	🔴 Critical	🟠 High	🟡 Medium
Likelihood: *Medium*	🟠 High	🟡 Medium	🟢 Low
Likelihood: *Low*	🟡 Medium	🟢 Low	🟢 Low

Impact

High Significant loss of assets in the protocol, significant harm to a group of users, or a core functionality of the protocol is disrupted.

Medium A small amount of funds can be lost or ancillary functionality of the protocol is affected. The user or protocol may experience reduced or delayed receipt of intended funds.

Low Can lead to any unexpected behavior with some of the protocol's functionalities that is notable but does not meet the criteria for a higher severity.

Likelihood

High The attack is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount gained or the disruption to the protocol.

Medium An attack vector that is only possible in uncommon cases or requires a large amount of capital to exercise relative to the amount gained or the disruption to the protocol.

Low Unlikely to ever occur in production.

Audit Scope & Methodology

Methodology

Guardian is the ultimate standard for Smart Contract security. An engagement with Guardian entails the following:

- Two competing teams of Guardian security researchers performing an independent review.
- A dedicated fuzzing engineer to construct a comprehensive stateful fuzzing suite for the project.
- An engagement lead security researcher coordinating the 2 teams, performing their own analysis, relaying findings to the client, and orchestrating the testing/verification efforts.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts. Comprehensive written tests as a part of a code coverage testing suite.
- Contract fuzzing for increased attack resilience.

Findings & Resolutions

ID	Title	Category	Severity	Status
C-01	Expected Fee Calculation Skews Output	Logical Error	Critical	Resolved
C-02	WNT Fees Cannot Be Used For GMX Buyback	Validation	Critical	Resolved
C-03	BUYBACK_MAX_PRICE_AGE Cannot Be Set	DoS	Critical	Resolved
H-01	V1 Fees Cannot Be Claimed	DoS	High	Resolved
M-01	Lack Of Interval Caps	Validation	Medium	Acknowledged
L-01	Output Amount Can Be Inaccurate	Logical Error	Low	Resolved
L-02	No Access Control On Claim Fees	Access Control	Low	Acknowledged
L-03	SafeTransferFrom Should Be First	Best Practices	Low	Acknowledged
L-04	Discounted GMX Buys Not Supported	Warning	Low	Acknowledged
L-05	buyback Uses Wrong Oracle Modifier	Informational	Low	Acknowledged
L-06	Consider Adding Withdraw Function For Fee Tokens	Informational	Low	Acknowledged
L-07	Memory Variable Declared In For Loop	Gas Optimization	Low	Resolved
L-08	Save WNT As Immutable	Gas Optimization	Low	Acknowledged

C-01 | Expected Fee Calculation Skews Output

Category	Severity	Location	Status
Logical Error	● Critical	FeeHandler.sol: 187	Resolved

Description

The buyback mechanism aims to calculate how many fee tokens must be provided to the user for the batch amount of buyback tokens based on current oracle prices:

`uint256 expectedFeeTokenAmount = Precision.mulDiv(batchSize, feeTokenPrice, buybackTokenPrice);`

The issue is that `batchSize` represents a buyback token amount, and should be multiplied with the `buybackTokenPrice` to get the USD value of the batch, and then divide by the `feeTokenPrice` to calculate the worth in fee tokens.

However, the current logic multiplies by `feeTokenPrice` and divides by `buybackTokenPrice`, leading to drastically different outputs of the swap.

Consider the following scenario:
Batch size: 1 GMX
`feeToken` WBTC: $100,000
`buybackToken` GMX: $10

The current logic would return `1 GMX * $100,000 / $10 = 10,000` BTC to provide to the user for 1 GMX, draining the contract's available fees. This scenario also does not account for the varying decimal precision in the prices for each token, which also drastically modifies the expected output.

Recommendation

Modify the calculation to `uint256 expectedFeeTokenAmount = Precision.mulDiv(batchSize, buybackTokenPrice, feeTokenPrice);`

Resolution

GMX Team: The issue was resolved in commit 217e012.

C-02 | WNT Fees Cannot Be Used For GMX Buyback

Category	Severity	Location	Status
Validation	● Critical	FeeHandler.sol	Resolved

Description

Function `claimFees` validates that the fee token batch size is zero, which prevents GMX and WNT from being the provided `feeToken`.

This will prevent accumulated WNT fees from being used to buyback GMX tokens in the `FeeHandler`, since the available fee amounts will not be incremented for either buyback token. Consequently, the expected distribution percentages of fees will never be met.

Recommendation

Allow WNT fees to buyback GMX.

Resolution

GMX Team: The issue was resolved in commit 217e012.

C-03 | BUYBACK_MAX_PRICE_AGE Cannot Be Set

Category	Severity	Location	Status
DoS	● Critical	Config.sol	Resolved

Description

`BUYBACK_MAX_PRICE_AGE` is not included in the `allowedBaseKeys` of the Config contract, which prevents it from being set, even by the keeper, as the `_validateKey` function will revert.

As a result, `maxPriceAge` will be 0 in the `_getMaxFeeTokenAmount` function, causing all buybacks to revert unless they are executed at the exact same timestamp as the oracle update.

Recommendation

Set `allowedBaseKeys[Keys.BUYBACK_MAX_PRICE_AGE]` to true in the Config file.

Resolution

GMX Team: The issue was resolved in commit 33844d9.

H-01 | V1 Fees Cannot Be Claimed

Category	Severity	Location	Status
DoS	High	FeeHandler.sol: 71	Resolved

Description

When withdrawing fees from V1, `FeeHandler` calls the `withdrawFees` function of the `vaultGov`, which is a timelock. However, this will always revert since the `vaultGov.withdrawFees` has an `onlyAdmin` modifier, and the `FeeHandler` is not admin of the timelock.

Since there can be only one admin, other `onlyAdmin` functions like `setLiquidator`, `addExcludedToken`, `setInPrivateTransferMode` etc. cannot be called if the `FeeHandler` is set as admin.

Recommendation

Consider removing the V1 fee logic from the `claimFees` function. Instead, V1 fees should be regularly withdrawn and transferred to the `FeeHandler` by the V1 admin. Additionally, consider adding a separate admin function that updates the available fee amounts after each V1 fee transfer.

Resolution

GMX Team: The issue was resolved in commit 68bb101.

M-01 | Lack Of Interval Caps

Category	Severity	Location	Status
Validation	Medium	FeeHandler.sol	Acknowledged

Description

Each buyback token has a particular batch amount to limit how much can be sold by the user at a particular time.

However, nothing prevents a user from calling the `buyback` function multiples times and swapping multiples batches within a small time frame.

This can be used to take advantage of any price discrepancies between the Chainlink oracles and market prices.

Recommendation

Consider limiting how much the protocol can buy back within a specific time interval.

Resolution

GMX Team: In case there is a large fee paid all at once, e.g. due to pending borrowing fees, then it is possible that the user can swap multiple times receiving the max price impact acceptable. This is accepted to be allowed behaviour.

L-01 | Output Amount Can Be Inaccurate

Category	Severity	Location	Status
Logical Error	Low	FeeHandler.sol: 105	Resolved

Description

The current `getOutputAmount` function does not validate that `markets` are unique. Hence, if duplicates of a market were passed into the view function, the `availableFeeAmount` would be larger due to double-counting of available fees.

Ultimately, the function would return an inaccurate output amount and this can affect the behavior of the UI.

Recommendation

Consider checking for duplicates in the `markets` parameter in the frontend and document this behavior.

Resolution

GMX Team: Added a note/comment in the code for this.

L-02 | No Access Control On Claim Fees

Category	Severity	Location	Status
Access Control	Low	FeeHandler.sol: 63	Acknowledged

Description

Currently any user can call function `claimFees` to withdraw fees from the GMX vault or claim accumulated GMXV2 fees. This can lead to a user continuously claiming fees while buybacks are active to prevent the protocol from claiming fees for itself when necessary.

Recommendation

Consider added `onlyFeeKeeper` access control.

Resolution

GMX Team: It is intended to allow any user to claim the fees so that they can call the buyback function after.

L-03 | SafeTransferFrom Should Be First

Category	Severity	Location	Status
Best Practices	Low	FeeHandler.sol	Acknowledged

Description

In function `buyback`, the `safeTransferFrom` occurs after the fee token is transferred. However it is a best practice to transfer funds in before performing other state updates.

Recommendation

Consider performing `safeTransferFrom` right after the initial validations and before the fee token is transferred to the user.

Resolution

GMX Team: Acknowledged; will not fix.

L-04 | Discounted GMX Buys Not Supported

Category	Severity	Location	Status
Warning	Low	FeeHandler.sol	Acknowledged

Description

The existing buyback mechanism aims to apply a premium on GMX purchases:

`Precision.applyFactor(expectedFeeTokenAmount, maxPriceImpactFactor + Precision.FLOAT_PRECISION);`

This will increase how many fee tokens the user may receive per buyback token. For example, if the price of GMX is $100 and batch size is 10, max price impact is 0.3%, and fees for AAVE accumulate, the max worth of AAVE that would be swapped would be $1003.

However, it may be prudent to also have a mechanism to discount GMX purchases so that users cannot extract value when the oracle price has yet to be updated to reflect latest market prices, or even to future-proof the contract as buyback incentive mechanisms may change with future proposals.

Recommendation

Clearly document the premium behavior to users and consider adding the ability to discount the GMX buyback price such as:

`Precision.applyFactor(expectedFeeTokenAmount, Precision.FLOAT_PRECISION - maxDiscountFactor);`

Resolution

GMX Team: Acknowledged; Will not fix for now, may be added as a feature in the future if needed.

L-05 | buyback Uses Wrong Oracle Modifier

Category	Severity	Location	Status
Informational	Low	FeeHandler.sol: 91	Acknowledged

Description

`buyback()` allows any user to call it and does not use the typical keeper system seen throughout the GMX protocol. This allows a user to create an atomic swap. Although an atomic action is taking place, the modifier `withOraclePrices()` is used.

This allows users to pass in price data that has a larger deadline than if the modifier `withOraclePricesForAtomicAction()` was used.

Although `maxPriceAge` is validated, if it is set to too large of a value, then this can open up users to perform swaps with prices that deviate from the current market price.

Additionally, if it is intended to still use Chainlink Price Feed, then using `withOraclePrices()` may cause unexpected reverts for a provider that is expected to be a Chainlink Data Stream.

Recommendation

If using `withOraclePrices()` is intended, verify that `BUYBACK_MAX_PRICE_AGE` is not set to too large of a value. Otherwise, use `withOraclePricesForAtomicAction()` instead.

Resolution

GMX Team: `BUYBACK_MAX_PRICE_AGE` will likely be set to 30 seconds or lower, `withOraclePricesForAtomicAction` is not used because some collateral tokens may not have an on-chain feed, which would cause a revert and would result in these tokens not being swapped out.

L-06 | Consider Adding Withdraw Function For Fee Tokens

Category	Severity	Location	Status
Informational	Low	FeeHandler.sol	Acknowledged

Description

Fee tokens are transferred to the `FeeHandler` contract, and users can purchase these fee tokens by providing buyback tokens. The keeper has the ability to withdraw buyback tokens but does not have the ability to withdraw fee tokens.

If users do not utilize the buyback functionality, claimed fee tokens may have to remain in the `FeeHandler` contract for a long period of time and cannot be used for other purposes.

Recommendation

Consider adding a privileged withdraw function for fee tokens.

Resolution

GMX Team: Acknowledged, we will not add this feature for now, it can be added later on if needed.

L-07 | Memory Variable Declared In For Loop

Category	Severity	Location	Status
Gas Optimization	Low	FeeHandler.sol: 140	Resolved

Description

`getOutputAmount()` declares `feeAmounts`, which is a memory variable, inside of a for loop. In Solidity, declaring a memory variable inside of a for loop stores a new variable to memory on each iteration.

The first 22 words of memory are priced linearly, but the pricing is quadratic after that. This can lead to users having to pay substantially more in gas costs.

Recommendation

Declare `feeAmounts` outside of the for loop, and reassign in each iteration.

Resolution

GMX Team: The issue was resolved in commit 33844d9.

L-08 | Save WNT As Immutable

Category	Severity	Location	Status
Gas Optimization	Low	FeeHandler.sol: 193	Acknowledged

Description

`_incrementAvailableFeeAmounts()` gets the wrapped native token from storage. Since the wrapped native token will remain the same for each chain, it is safe to store it as an immutable variable. This will cut down on gas costs, since it will avoid reading from storage.

Recommendation

Store `WNT` as an immutable variable in the constructor similar to `gmx`.

Resolution

GMX Team: Acknowledged; Will not fix.

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