GUARDIAN AUDITS

SMART CONTRACT SECURITY AUDIT OF

GMX

Summary

Audit Firm Guardian

Prepared By Owen Thurm, Daniel Gelfand

Client Firm GMX

Final Report Date October 23, 2023

Audit Summary

GMX engaged Guardian to review the security of its liquidity migration contracts from GLP to GMX V2 Markets. From the 9th of October to the 23th of October, a team of 2 auditors reviewed the source code in scope. All findings have been recorded in the following report.

🔗 Blockchain network: **Arbitrum, Avalanche**

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

📊 Code coverage & PoC test suite: https://github.com/GuardianAudits/GMX-POCS-10-09

Table of Contents

Project Overview

Project Summary

Project Name	GMX
Language	Solidity
Codebase	https://github.com/gmx-io/gmx-synthetics
Commit(s)	Initial Commit: 02319e7d2897167a1f033083d5b99eaefd468d7d Final Commit: 226fca743a64977f17f353de05d71cc03fc21a3f

Audit Summary

Delivery Date	October 23, 2023
Audit Methodology	Static Analysis, Manual Review, Test Suite

Vulnerability Summary

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

Audit Scope & Methodology

ID	File	SHA-1 Checksum(s)
GLPM	GlpMigrator.sol	dc982785a232d1c15259dbf78646d2f178d0a9b7
IGRR	IGlpRewardRouter.sol	9bdbb6ae9cd110fa8fadfd75afb1d554a80d9d1c
IGTL	IGlpTimelock.sol	de0d5ad0b0ac850d1ffba7c6a478e369a74f9e3a
IGVT	IGlpVault.sol	77cf63eb1fbe12a36e1fd6f90afd0aaab1fcd629
EXTH	ExternalHandler.sol	0316f26e6394c0061ab06421e74ba6e022e058f9

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
GLPM-1	Reduced Redemption Fees Gamed	Logical Error	High	Acknowledged
GLPM-2	USDC vs USDC.e	Validation	High	Resolved
GLPM-3	Additional Ether Lost	Logical Error	Medium	Resolved
GLPM-4	User Forced To Deposit Both Tokens	Logical Error	Medium	Resolved
GLPM-5	Reduced Burn Fee Can Be Larger Than Current	Validation	Medium	Resolved
GLPM-6	Lists With Different Lengths	Validation	Medium	Resolved
EXTH-1	Lack of Contract Existence Check	Low-Level Calls	Medium	Resolved
EXTH-2	Lack Of safeTransfer For Arbitrary Token	Logical Error	Medium	Resolved
GLPM-7	Inflexible executionFee	Optimization	Low	Resolved
GLPM-8	Migration Contracts Needs To Be Set As Handler	Access Control	Low	Resolved

GLPM-1 | Reduced Redemption Fees Gamed

Category	Severity	Location	Status
Logical Error	High	GlpMigrator.sol: 263	Acknowledged

Description

In the `_redeemGlp` function, users are allowed to make any arbitrary external calls with the `redemptionInfo.externalCallTargets` and `redemptionInfo.externalCallDataList`.

Therefore a user seeking to redeem GLP using the discounted redemption may do so as the external call is executed within the context of the `withReducedRedemptionFees` modifier.

Users may abuse the system in a similar way by simply providing an EOA address as the `redemptionInfo.receiver` rather than the `DepositVault`, or by using the subsequent external call to transfer out the redeemed tokens to their EOA.

Recommendation

Require that the `redeemedTokenAmount` (or at least a majority, accounting for potential fees & slippage) end up in the `DepositVault` contract.

This validation also serves as a safety net in the event that the provided `redemptionInfo.externalCallTargets` or `redemptionInfo.externalCallDataList` hold errors.

Resolution

GMX Team: Acknowledged and comment added in commit 2de90ca.

GLPM-2 | USDC vs USDC.e

Category	Severity	Location	Status
Validation	High	GlpMigrator.sol: 186-188	Resolved

Description

Currently GLP consists of a large amount of USDC.e (bridged USDC), which has a different token address than USDC. On the other hand, GMX V2 pools all use USDC.

Because a user is required to pass a `migrationItem.short.token` that matches the `cache.market.shortToken`, they cannot redeem with USDC.e as `tokenOut` since the short token validation will revert.

Rather, the user is forced to redeem for the limited amount of USDC directly so that they can deposit the USDC into the GMX V2 market.

Recommendation

In the case of USDC, modify the `InvalidShortTokenForMigration` check such that USDC.e can still pass and then be swapped for native USDC.

Furthermore, consider adding a state check after the external calls to check the token balances of the `depositVault`, to ensure the user has not mistakenly sent the USDC.e to the GMX V2 system and lost it.

Resolution

GMX Team: The `InvalidLongTokenForMigration` and `InvalidShortTokenForMigration` checks have been removed in commit 3a696d5.

GLPM-3 | Additional Ether Lost

Category	Severity	Location	Status
Logical Error	Medium	GlpMigrator.sol: 139	Resolved

Description

If the provided `msg.value` is greater than the `executionFee * migrationItems.length` then the excess Ether is not refunded to the user and can be used by the user who calls the migrate function next.

Recommendation

Add validation to ensure that `msg.value == executionFee * migrationItems.length`, otherwise refund any excess Ether to the caller.

Resolution

GMX Team: The recommendation has been implemented in commit 58312d4.

GLPM-4 | User Forced To Deposit Both Tokens

Category	Severity	Location	Status
Logical Error	Medium	GlpMigrator.sol: 165-191	Resolved

Description

In `RewardRouter.sol` for GMX V1, function `unstakeAndRedeemGlp()` requires that `_glpAmount > 0`:

`require(_glpAmount > 0, "RewardRouter: invalid _glpAmount");`

In the GlpMigrator contract, if a user wants to only redeem for a the long token in a market, and leaves the `GlpRedemption short` with `migrationItem.short.glpAmount = 0`, the migration will fail due to the above revert.

This is unexpected behavior as it forces users to not only populate the `migrationItem.short.token` to match the market's short token, but also set a miniscule amount of `glpAmount` to redeem for the short token to avoid migration failure.

The same behavior applies if a user wants to solely redeem for the short token in a market, and leave the long token untouched.

Recommendation

If the `glpAmount` for either the long or short token is 0, skip the call `unstakeAndRedeemGlp()` for that token.

Resolution

GMX Team: The recommendation has been implemented in commit 58312d4.

GLPM-5 | Reduced Burn Fee Can Be Larger Than Current

Category	Severity	Location	Status
Validation	Medium	GlpMigrator.sol: 73, 122, 125,	Resolved

Description

GMX is choosing to reduce the burn fee to further incentivize migration from GMX V1 to its latest GMX V2 system. However, there is no guarantee that the `reducedMintBurnFeeBasisPoints` is less than or equal to the current `mintBurnFeeBasisPoints`.

The burn fee can be increased to be larger than its current value, causing users to redeem less tokens than expected.

Recommendation

Inside modifier `withReducedRedemptionFees`, only update the burn fee in GMX V1 if the `_reducedMintBurnFeeBasisPoints` is smaller:

```
bool shouldUpdateFees = _reducedMintBurnFeeBasisPoints < mintBurnFeeBasisPoints;
```

Resolution

GMX Team: The recommendation has been implemented in commit 58312d4.

GLPM-6 | Lists With Different Lengths

Category	Severity	Location	Status
Validation	Medium	GlpMigrator.sol: 264-265	Resolved

Description

The `externalCallTargets` and `externalCallDataList` are used to call an external protocol with user-passed data.

However, `externalCallDataList` may have a different length than `externalCallTargets`, potentially causing an out-of-bounds error or the incorrect data being used for a particular target.

Similarly, `refundTokens` and `refundReceivers` may be different lengths, potentially causing an out-of-bounds error or funds being sent to an unintended receiver.

Recommendation

Add validation such that `externalCallTargets` and `externalCallDataList` are the same length and that `refundTokens` and `refundReceivers` are the same length:

`require(externalCallTargets.length == externalCallDataList.length)`

`require(refundTokens.length == refundReceivers.length)`

Resolution

GMX Team: The recommendation has been implemented in commit 3a696d5.

EXTH-1 | Lack of Contract Existence Check

Category	Severity	Location	Status
Low-Level Calls	Medium	ExternalHandler.sol: 51	Resolved

Description

The low-level `call` returns a success boolean of `true` if the target contract does not exist. As a result, the migration may not detect some failed external calls, leading to loss of funds for users.

Recommendation

Consider implementing a contract existence check prior to the `call`.

Resolution

GMX Team: The recommendation has been implemented in commit 3a696d5.

EXTH-2 | Lack Of safeTransfer For Arbitrary Token

Category	Severity	Location	Status
Logical Error	Medium	ExternalHandler.sol: 42	Resolved

Description

In the `makeExternalCalls` function, the arbitrary `refundToken` is transferred using the `transfer` function, however `safeTransfer` should be used to avoid potential loss if the token chooses to return `false` rather than reverting upon failure.

Recommendation

Prefer `safeTransfer` to `transfer`.

Resolution

GMX Team: The recommendation has been implemented in commit 3a696d5.

GLPM-7 | Inflexible executionFee

Category	Severity	Location	Status
Optimization	Low	GlpMigrator.sol: 210	Resolved

Description

The same `executionFee` is used for every `migrationItem` in the `migrationItems` list, however some migrations may require a smaller `executionFee` than others depending on if they are single token deposits.

Recommendation

Allow an individual `executionFee` to be specified on a `migrationItem` basis.

Resolution

GMX Team: The recommendation has been implemented in commit 58312d4.

GLPM-8 | Migration Contracts Needs To Be Set As Handler

Category	Severity	Location	Status
Access Control	Low	GlpMigrator.sol: 76	Resolved

Description

In order for `glpTimelock.setSwapFees()` to succeed, the GlpMigrator contract must be given the necessary access control to bypass the `onlyKeeperAndAbove` modifier in GMX V1.

Recommendation

Set the migration contract as a handler in the GLP Timelock contract.

Resolution

GMX Team: Confirmed the `GlpMigrator` will have the necessary privileges.

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