GUARDIAN AUDITS

SMART CONTRACT SECURITY AUDIT OF

Key Finance

Summary

Audit Firm: Guardian Audits

Client Firm: Key Finance

Prepared By: Owen Thurm, Daniel Gelfand

Final Report Date - May 1, 2023

Audit Summary

Key Finance engaged Guardian to review the security of its GMX rewards solution, GMX Key. From the 28th of March to the 10th of April, a team of 2 auditors reviewed the source code in scope. The auditing approach championed manual analysis to uncover novel exploits and verify intended behavior with ancillary verification from formal methods such as contract fuzzing. All findings and remediations 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.

Issues Detected Throughout the course of the audit numerous high impact issues were uncovered and promptly remediated by the Key Finance team. Several issues impacted the fundamental behavior of the protocol, Guardian believes these issues to be resolved. However numerous changes were made to the codebase after Guardian's review, for this reason Guardian supports an independent security audit of the protocol at a finalized frozen commit.

🔗 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	Key Finance
Language	Solidity
Codebase	https://github.com/KeyFinanceTeam/key-finance-contracts
Commit (final)	45302fa6f71ba65241c08e96883a3e7db15c1bd8

Audit Summary

Delivery Date	May 1, 2023
Audit Methodology	Static Analysis, Manual Review, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
High	3	0	0	0	0	3
Medium	9	0	0	2	0	7
Low	5	0	0	1	0	4

Audit Scope & Methodology

Scope

ID	File	SHA-1 Checksum(s)
ADM	Adminable.sol	dc2d2e8b9cbfee7524008057260ef56c8191c5f5
BTOK	BaseToken.sol	c738a59fa377f0672a0407f8fd9854a0b0c6811b
CFG	Config.sol	5f37e991c2d110801140321a9ec0201e122d932c
CNV	Converter.sol	d48a7eb00461c22f3e9a80b6d342f0ec124c7887
GKEY	GMXKey.sol	27a536536afc30eb49af9e4a5c05924e7e18cbe5
LPS	LPStaker.sol	b537640be0e253dfe6b943cc9a8df3ff7a8029fd
MKEY	MPKey.sol	c85f89bdf90d7bbf609f253ca78a46be60683601
RWH	RewardClaimHelper.sol	c1942f4493d624a597eda2c96bebe4f13e16b826
REW	Rewards.sol	31c07d7055265c7ef3d029db79f2988026db1e61
STK	Staker.sol	dc65be687e32e95d7020900b8de9b5fbf534fe68
TREC	TransferReceiver.sol	4d4aed6fd58d5c348d98a50c6c6ceb7092b8f162

Audit Scope & Methodology

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.

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.

Findings & Resolutions

ID	Title	Category	Severity	Status
LPS-1	Rewards May Be Stolen	Logical Error	High	Resolved
STK-1	Reward Compounds Are Sandwichable	Sandwich Attack	High	Resolved
TREC-1	Lost WETH Rewards Upon Upgrade	Lost Rewards	High	Resolved
GLOBAL-1	Centralization Risk	Centralization / Privilege	Medium	Acknowledged
TREC-2	Invalid Assumption	Logical Error	Medium	Resolved
TREC-3	Unexpected Rewards	Protocol Manipulation	Medium	Resolved
LPS-2	onERC721Received Reentrancy	Reentrancy	Medium	Resolved
STK-2	Users May Stake For Others	Unexpected Behavior	Medium	Resolved
ADM-1	Admin Role Changes Should Be Two Step	Unnecessary Risk	Medium	Resolved
BTOK-1	Dangerous Approve	Frontrunning	Medium	Resolved
LPS-3	Fee-On-Transfer Tokens	Compatibility	Medium	Acknowledged
REW-1	Chain Incompatibility	Compatibility	Medium	Resolved
GLOBAL-2	Custom Reverts	Optimization	Low	Acknowledged

Findings & Resolutions

ID	Title	Category	Severity	Status
GLOBAL-3	Use Standard ReentrancyGuard	Best Practices	Low	Resolved
REW-2	Redundant Transfers	Optimization	Low	Resolved
CNV-1	Inaccurate Comment	Documentation	Low	Resolved
GLOBAL-4	Lack of Events	Events	Low	Resolved

LPS-1 | Rewards May Be Stolen

Category	Severity	Location	Status
Logical Error	High	LPStaker.sol	Resolved

Description

The `uniswapV3Staker` contract which the `LPStaker` interacts with allows any arbitrary address to directly call the `unstakeToken` function and unstake for any depositor after the incentive key `endTime`.

When a deposit is unstaked from an incentive key directly from the `uniswapV3Staker`, those rewards will be incremented for the `LPStaker` contract, but not credited towards the user who staked. Therefore malicious stakers may unstake for other stakers and immediately claim their rewards as their own by unstaking through the `LPStaker` contract.

Recommendation

Consider using a modified version of the `uniswapV3Staker` where the depositor must always be the one to unstake. Otherwise be sure to manage the incentive keys extremely carefully and never allow an incentive key to reach its `endTime` while users have staked for it.

Resolution

Key Team: A modified version of the `uniswapV3Staker` was implemented.

STK-1 | Reward Compounds Are Sandwichable

Category	Severity	Location	Status
Sandwich Attack	● High	Staker.sol	Resolved

Description

There exists no fee or lockup period associated with staking to receive a portion of the rewards compounded during the `updateAllRewardsForTransferReceiverAndTransferFee` function.

A malicious actor may simply buy GMXKey and stake right before the `updateAllRewardsForTransferReceiverAndTransferFee` function to immediately accrue a portion of the collected rewards that were meant to be attributed to other stakers. The malicious actor can then immediately claim these rewards, unstake and sell GMXKey after the reward compound, therefore stealing rewards from other stakers.

Recommendation

Consider implementing a staking/unstaking fee or a “warmup period” where stakers cannot accrue rewards.

Resolution

Key Team: A new approach to rewards including reward periods was adopted.

TREC-1 | Lost WETH Rewards Upon Upgrade

Category	Severity	Location	Status
Lost Rewards	High	TransferReceiver.sol	Resolved

Description

During the account transfer process, the `RewardRouterV2` does not `claimForAccount` from the `feeGMXTracker`. Therefore any accrued WETH rewards will not be transferred to the new address upon upgrade of the `TransferReceiver`.

The `TransferReceiver` will then have no means of claiming these WETH rewards and injecting them into the `Rewards` system as the `amountToMint` calculation will underflow and revert since the receiver's staked balances have been transferred out.

Recommendation

Require WETH rewards to be claimed and injected into the `Rewards` system before initiating the `TransferReceiver` upgrade process.

Resolution

Key Team: The Rewards logic was updated to allow any remaining WETH to be collected.

GLOBAL-1 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Medium	Global	Acknowledged

Description

The admin address holds the ability to negatively impact the system in numerous ways, including but not limited to:

- Take all esGMX, sGMX and bnGMX via reserveSignalTransfer and signalTransfer.
- Use the withdrawTokens function to take any non-WETH ERC20 rewarded to the TransferReceiver.
- Lock all staked Uniswap V3 LP positions by pausing the LPStaker contract.
- Lock all GMXKey and MPKey stakes by pausing the Staker contract.
- Raise fees to 100% in the Rewards contract.

Recommendation

Ensure that the admin address is a multi-sig, optionally with a timelock for improved community trust and oversight. Attempt to limit the scope of the admin address permissions such as locking stakes and raising fees to 100%.

Resolution

Key Team: We have removed the pausable modifier for functions that would lock V3 positions and privileged addresses will be multi-sigs.

TREC-2 | Invalid Assumption

Category	Severity	Location	Status
Logical Error	Medium	TransferReceiver.sol	Resolved

Description

In the `acceptTransfer` function it is assumed that `A maximum of ~7% amount of GMX (as esGMX) would be added`, however that assumption does not hold in several cases.

A user could have removed their sGMX and only been left with esGMX and bnGMX or a user may have accepted a transfer from another account which perturbed this ratio.

Recommendation

Do not rely on this assumption holding and remove the comment. If it is paramount that only a small percentage of esGMX is added, add an explicit check.

Resolution

Key Team: The comment has been removed.

TREC-3 | Unexpected Rewards

Category	Severity	Location	Status
Protocol Manipulation	Medium	TransferReceiver.sol	Resolved

Description

The allowance is used to determine how much WETH to inject into the `Rewards` system and it is incremented based on the current balance of the `TransferReceiver`.

However the balance of the `TransferReceiver` can be inflated by transferring WETH directly to the `TransferReceiver` contract. Therefore rewards that are not explicitly from GMX are able to enter the `Rewards` system.

Additionally, it is possible that `privateTransferMode` is turned off for either esGMX or bnGMX in the future, which could also potentially perturb the `Rewards` system.

Recommendation

Consider if outside WETH should be included in the accounted rewards. If not, implement a before and after balance check when calling `rewardRouter.handleRewards` to get the actual WETH amount received from GMX.

Additionally, have a plan for the scenario where `privateTransferMode` is turned off for either `esGMX` or bnGMX.

Resolution

Key Team: The recommended before and after check was implemented.

LPS-2 | onERC721Received Reentrancy

Category	Severity	Location	Status
Reentrancy	Medium	LPStaker.sol	Resolved

Description

During the `unstakeAndWithdrawLpToken` function, the `msg.sender` may re-enter into the `onERC721Received` function upon the `withdrawToken` call by transferring the withdrawn Uniswap V3 LP NFT back to the `LPStaker`.

This reentrancy can yield an unexpected state where the token still exists in the `tokensStaked` list for the owner, but not in the `idToOwner` or `stakedIndex`. Such an unexpected state may have unintended consequences and effect frontend systems reading from the contract or third party systems built on top of the `LPStaker`.

Recommendation

Move the `withdrawToken` call to the end of the `for` loop to follow Check-Effects-Interactions. Alternatively, add a reentrancy check to the `onERC721Received` function.

Resolution

Key Team: Check-Effects-Interactions was adopted.

STK-2 | Users May Stake For Others

Category	Severity	Location	Status
Unexpected Behavior	Medium	Staker.sol	Resolved

Description

The `stake` function in the `Staker` contract allows users to stake for any address rather than just their own. This can cause unexpected consequences for contract systems interfacing with the `Staker` contract, especially if the necessary staking “warmup period” is implemented.

Recommendation

Reconsider if this feature is necessary, and if so carefully document it and consider its impacts when combined with the solution for STK-1.

Resolution

Key Team: Users can no longer stake for others in the Staker contract.

ADM-1 | Admin Role Changes Should Be Two Step

Category	Severity	Location	Status
Unnecessary Risk	Medium	Adminable.sol	Resolved

Description

As addressed in GLOBAL-1, the admin address carries numerous important abilities for the system.

However the `changeAdmin` function allows the admin address to be errantly transferred to the wrong address as it does not use a two-step transfer process.

Recommendation

Implement a two step “push” and “pull” admin transfer process. If it is desired to have a method to relinquish ownership, implement a separate function to do so.

Resolution

Key Team: The recommended push and pull transfer process was adopted.

BTOK-1 | Dangerous Approve

Category	Severity	Location	Status
Frontrunning	Medium	BaseToken.sol	Resolved

Description

The `BaseToken` only exposes the dangerous `approve` function rather than an additional alternative `increaseAllowance` function.

Recommendation

Implement an `increaseAllowance` function so that users may increase their allowances without risk of frontrunning.

Resolution

Key Team: The recommended `increaseAllowance` function was implemented.

LPS-3 | Fee-On-Transfer Tokens

Category	Severity	Location	Status
Compatibility	Medium	LPStaker.sol	Acknowledged

Description

The `LPStaker` contract is not compatible with fee-on-transfer tokens for the `rewardToken` as it relies on the `uint` returned from the `uniswapV3Staker` `claimReward` function to increment the reward mapping.

Fee on transfer tokens will cause this returned value to be inaccurate and potentially leave users unable to claim their rewards and potentially locked in the contract.

It should also be noted that rebase tokens or other balance altering tokens will not be accurately accounted for in a similar way.

Recommendation

Consider if fee-on-transfer, rebase, or any similar tokens should be supported. If so, add before and after balance checks for the `claimReward` function to measure the reward claimed accurately.

Resolution

Key Team: The `rewardToken` will never be a fee-on-transfer token.

REW-1 | Chain Incompatibility

Category	Severity	Location	Status
Compatibility	Medium	Rewards.sol	Resolved

Description

The `withdrawTo` function is used on WETH in the `_transferAsETH` function. This function is supported on Arbitrum, however it is not supported on the Avalanche C-chain or other networks that GMX may deploy on.

Recommendation

Do not use the `withdrawTo` function when deploying on Avalanche or other chains. Instead implement a method to receive Ether and relay it to the `to` address.

Resolution

Key Team: The `withdrawTo` function has been replaced with `withdraw`.

GLOBAL-2 | Custom Reverts

Category	Severity	Location	Status
Optimization	● Low	Global	Acknowledged

Description

Throughout the codebase `require` statements are used when instead custom errors may be implemented with `if` condition checks.

Recommendation

Replace `require` statements with `if` statements and custom error reverts to save gas.

Resolution

Key Team: Opted to keep the `require` statements.

GLOBAL-3 | Use Standard ReentrancyGuard

Category	Severity	Location	Status
Best Practices	● Low	Global	Resolved

Description

Throughout the codebase a non OpenZeppelin `ReentrancyGuard` contract is used. The custom `ReentrancyGuard` contract is inferior as it uses a `boolean _guard` storage variable.

Recommendation

Use the OpenZeppelin `ReentrancyGuard`.

Resolution

Key Team: Implemented the recommended OZ `ReentrancyGuard`.

REW-2 | Redundant Transfers

Category	Severity	Location	Status
Optimization	Low	Rewards.sol: 187	Resolved

Description

When claiming and updating a reward for a `TransferReceiver`, the fee is first transferred to the `TransferReceiver` before being transferred to the `msg.sender`.

Recommendation

Consider implementing a `feeTo` address parameter on the `updateAllRewardsForTransferReceiverAndTransferFee` function so that two redundant transfers are not needed.

Resolution

Key Team: The suggested `feeTo` address was implemented.

CNV-1 | Inaccurate Comment

Category	Severity	Location	Status
Documentation	● Low	Converter.sol: 177, 210	Resolved

Description

In the `completeConversion` and `completeConversionToMpKey` functions it is stated that `the sender's vesting tokens must be non-zero` however the sender's vesting tokens must be zero or else the `acceptTransfer` on the `RewardRouter` will revert.

Recommendation

Update the inaccurate comments.

Resolution

Key Team: The comment was updated.

GLOBAL-4 | Lack of Events

Category	Severity	Location	Status
Events	Low	Global	Resolved

Description

Throughout the codebase there are functions that alter the contract state in a significant way without emitting an event.

For example the `signalTransfer` and `reserveSignalTransfer` ought to emit an event for third party systems to be able to read.

Recommendation

Emit an appropriate event whenever a significant change is made in the contract system.

Resolution

Key Team: The suggested events were added.

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