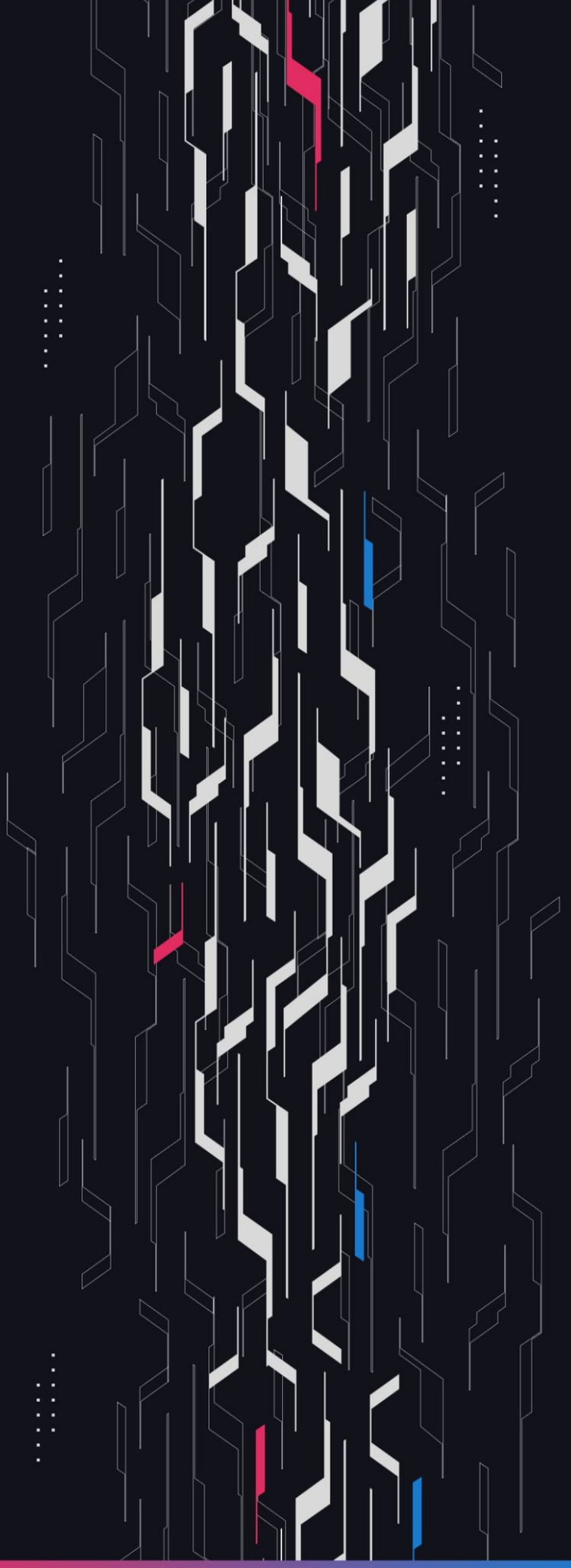
GA GUARDIAN

GMX

V2.1 Updates #2

Security Assessment

June 14th, 2024



Summary

Audit Firm Guardian

Prepared By Owen Thurm, Daniel Gelfand, Kiki, Mark Jonathas, Vladimir Zotov, Dogus Kose, 0x3b

Client Firm GMX

Final Report Date June 14th, 2024

Audit Summary

GMX engaged Guardian to review the security of updates to it's synthetic assets exchange. From the 3rd of June to the 6th of June, a team of 7 auditors reviewed the source code in scope. All findings have been recorded in the following report.

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

📊 Code coverage & PoC test suite: https://github.com/GuardianAudits/gmx-v2-1-fuzzing

Table of Contents

Project Overview

Project Summary

Project Name	GMX
Language	Solidity
Codebase	https://github.com/gmx-io/gmx-synthetics/
Commit(s)	b9abc254e75d34ee70057bb1b682fa61e4d5c7d5

Audit Summary

Delivery Date	June 14th, 2024
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	2	0	0	0	0	2
High	1	0	0	0	0	1
Medium	7	0	0	3	0	4
Low	8	0	0	5	0	3

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	Liquidations Prevented With updateOrder	Protocol Manipulation	Critical	Resolved
C-02	LimitSwaps Cannot Execute After Request Expiration	Logical Error	Critical	Resolved
H-01	Keeper's Not Remunerated for Cancellation Callback	Logical Error	High	Resolved
M-01	No way for user to add cancellation receiver	Logical Error	Medium	Resolved
M-02	Sequencer Outage Risks	Logical Error	Medium	Acknowledged
M-03	Callback Gas Validation Ignores 63/64 Rule	Logical Error	Medium	Resolved
M-04	AutoCancel Validation May DoS Order Creation	Logical Error	Medium	Acknowledged
M-05	Orders Which Close Positions May Be Censored	Logical Error	Medium	Resolved
M-06	Liquidation Gas Usage May Exceed Block Gas Limit	Protocol Manipulation	Medium	Acknowledged
M-07	Old Estimated Execution Base Gas Fee Used	Logical Error	Medium	Resolved
L-01	GMX config uses realtimeFeed instead of dataStream	Configuration	Low	Resolved
L-02	Users Pay Extra in Fees In Certain Markets	Documentation	Low	Acknowledged
L-03	Total AutoCancel Gas Supersedes Max Auto Cancels	Documentation	Low	Acknowledged

Findings & Resolutions

ID	Title	Category	Severity	Status
L-04	Callback And Refund Receiver Risks	Documentation	Low	Acknowledged
L-05	Incorrect Oracle Price Estimate	Logical Error	Low	Resolved
L-06	Inconsistent AutoCancel Validation	Unexpected Behavior	Low	Resolved
L-07	Optimal Usage Borrowing Can Remain Constant	Configuration	Low	Acknowledged
L-08	Incorrect Estimated Price Counts	Logical Error	Low	Acknowledged

C-01 | Liquidations Prevented With updateOrder

Category	Severity	Location	Status
Protocol Manipulation	● Critical	OrderHandler.sol: 114	Resolved

Description

createOrder checks every new decrease order for if it will pass over the max autoCancel gas limit using validateTotalCallbackGasLimitForAutoCancelOrders.

However the same check is missed inside updateOrder, enabling the user to:

1. Create 5 decrease orders with their max allowed callback gas, without putting them inside the autoCancel list.

2. Call updateOrder for all of these order with autoCancel variable set to true.
With this method users can bypass 5 million maximum callback gas limit for Auto Cancel orders and reach up to 10 million. Which will result in reverting liquidations because gas required to liquidate will bypass block gas limit in avalanche.

Recommendation

Add the validateTotalCallbackGasLimitForAutoCancelOrders validation inside updateOrder.

Resolution

GMX Team: Resolved.

C-02 | LimitSwaps Cannot Execute After Request Expiration

Category	Severity	Location	Status
Logical Error	● Critical	SwapOrderUtils: 37-45	Resolved

Description

When handling swap orders, the validation for the `requestExpirationPeriod` is meant to only be applied to `MarketSwaps`. Since it is applied to both swap types, it will revert for nearly all `LimitSwaps`.

This will occur because the majority of `LimitSwaps` will not be eligible to be executed until a later time has passed than the `REQUEST_EXPIRATION_TIME`.

Recommendation

Only perform this verification for `MarketSwaps`.

Resolution

GMX Team: Resolved.

H-01 | Keeper's Not Remunerated For Cancellation Callback

Category	Severity	Location	Status
Logical Error	High	OrderUtils.sol: 212-226	Resolved

Description

The callback gas amount is included inside an order's `executionFee`, however the `payExecutionFee` function will refund the not used part of `executionFee` to the user.

Since this unused part includes callback gas, the keeper will not be remunerated for the gas spent during the cancellation callback.

Recommendation

Change the places for order cancellation callback call and execution fee payment.

Resolution

GMX Team: Resolved.

M-01 | No way for user to add cancellation receiver

Category	Severity	Location	Status
Logical Error	Medium	OrderUtils.sol: 141	Resolved

Description

When an order gets cancelled there is a check to see if the order has a `cancellationReceiver`. If it does the funds will be sent there, if not then the `account` will get the funds. This works well, however there is no way for a user to add a `cancellationReceiver` when creating an order. Preventing the use of this feature.

Recommendation

Set the `cancellationReceiver` when creating an order and validate that the address used is valid.

Resolution

GMX Team: Resolved.

M-02 | Sequencer Outage Risks

Category	Severity	Location	Status
Logical Error	Medium	Global	Acknowledged

Description

The sequencer uptime check is performed only in: Atomic Withdrawal, Normal Withdrawal and Liquidations.

If sequencer is down, while it won't be possible to execute these functions, rest of the protocol will continue functioning if they don't have a priceFeed to check for reference price.

Recommendation

Localize the sequencer checks to exactly where the Chainlink Aggregator Price is used.

Resolution

GMX Team: Acknowledged.

M-03 | Callback Gas Validation Ignores 63/64 Rule

Category	Severity	Location	Status
Logical Error	Medium	CallbackUtils.sol: 72	Resolved

Description

validateGasLeftForCallback() verifies that the gas left in the transaction is enough to call the callback contract. However, validateGasLeftForCallback() checks gasLeft() and forgets to account that 1/64th of the gas is reserved when making an external call. Although this case is less likely to occur, it has the same impact as H-06.

Recommendation

Verify that the gasLeft() subtracted by the gas withheld from making an external call is greater than the callback gas limit.

Resolution

GMX Team: Resolved.

M-04 | AutoCancel Validation May DoS Order Creation

Category	Severity	Location	Status
Logical Error	Medium	Global	Acknowledged

Description

`MAX_TOTAL_CALLBACK_GAS_LIMIT_FOR_AUTO_CANCEL_ORDERS` can change according to gas requirements of the system/chain. If this value decreases however, the position holders that already have maximum amount of callback gas used for their autoCancel orders can not call decrease order because the call will revert with `MaxTotalCallbackGasLimitForAutoCancelOrdersExceeded`.

Recommendation

Before reducing this variable inform users about this problem and let them prepare their positions to handle with this case.
Additionally, this validation does not need to take place for `MarketDecrease` orders.

Resolution

GMX Team: Acknowledged.

M-05 | Orders Which Close Positions May Be Censored

Category	Severity	Location	Status
Logical Error	Medium	OrderHandler.sol: 201	Resolved

Description

If a `decreaseOrder` will close the position altogether, the `autoCancelList` will be cleared out during that order's execution. But gas provided by the keeper won't be checked if it is sufficient to handle the gas required for cancellations of these orders that are in `autoCancelList`.
If keepers don't provide enough gas for all auto-cancellation logic, since the gas provided is not validated to cover these auto-cancellations with `validateExecutionGas`, a position closing order created by the user will be cancelled instead of reverting. This can lead to the censoring of closing orders for users and can lead to unfair liquidations and loss of funds.

Recommendation

There are different possible solutions that comes with some caveats.
1- Query the position to see if order size is the entire position. If so, increase the result of `estimateExecuteOrderGasLimit` when the decrease order has auto cancel orders.
2- Before starting `clearAutoCancelOrders()` check if there is enough gas, if not revert such that order is not cancelled and the error is caught in the `_handleOrderError` function as a keeper mistake.
Note that both of these first two solutions has a griefing vector whereby someone can frontrun the execution transaction and update an order to be an auto-cancel one such that the required gas for both will change. Which can lead to revert for keeper's execution error.
3- Separating the logic of auto cancellation from order execution. Emitting an event after position is completely closed and letting keepers to call `clearAutoCancelOrders` in a seperate transaction can solve the problem in a safer way, which will also address high gas usage concerns. The caveat for this is the execution logic change itself.

Resolution

GMX Team: Resolved.

M-06 | Liquidation Gas Usage May Exceed Block Gas Limit

Category	Severity	Location	Status
Protocol Manipulation	Medium	ExecuteOrderUtils.sol: 119-133	Acknowledged

Description PoC

Based on the estimated gas usage configurations, the gas required to execute some decrease and liquidation orders may exceed the Avalanche block gas limit of 15,000,000 gas:

• Liquidation's gas usage itself: 4,000,000 (Decrease order gas limit)
• afterOrderExecution callback gas: 2,000,000
• Main payExecutionFee: 500,000
• 5 autoCancel order cancellation: 5 x 600.000 = 3,000,000
• 5 autoCancel order cancellation callback: 5,000,000
• 5 autoCancel payExecutionFee: 5 x 500,000 = 2,500,000
• In total = 17,100,000 which is 2,100,000 more than avalanche block gas limit.

However in practice, it is unlikely that a liquidation will consume 15,000,000 or more gas units, refer to the attached PoC where we show that the rough maximum gas usage for a liquidation is around 14,000,000 gas.

If liquidation execution can consume more than 15,000,000 gas units this would result in unliquidatable positions on the Avalanche network, which will introduce bad debt into the system.

Recommendation

Carefully consider this limit when making future code updates and modifying the `refundExecutionFeeGasLimit` as well as other gas configurations.

Resolution

GMX Team: Acknowledged.

M-07 | Old Estimated Execution Base Gas Fee Used

Category	Severity	Location	Status
Logical Error	Medium	Global	Resolved

Description

The `EXECUTION_GAS_FEE_BASE_AMOUNT` key has been replaced with an `EXECUTION_GAS_FEE_BASE_AMOUNT_V2_1` key to allow an increased base fee to be charged for additional gas expenditures in V2.1.

However the corresponding estimated fee which is required upon order creation is still based upon the `ESTIMATED_GAS_FEE_BASE_AMOUNT` which corresponds with the old estimated base gas fee amount.

As a result the estimated fee which users are required to pay upfront may be insufficient to cover the gas expenditure for order execution in the V2.1 system.

Recommendation

Consider implementing a `ESTIMATED_GAS_FEE_BASE_AMOUNT_V2_1` which corresponds to the `EXECUTION_GAS_FEE_BASE_AMOUNT_V2_1` value.

Resolution

GMX Team: Resolved.

L-01 | Config Uses realtimeFeed Instead Of dataStream

Category	Severity	Location	Status
Configuration	Low	config/tokens.ts	Resolved

Description

The file `config/tokens.ts` implements the configuration for all tokens and their oracles. In the GMX tokens, `realtimeFeedId` and `realtimeFeedDecimals` are used instead of the new `dataStreamFeedId` and `dataStreamFeedDecimals`.
This will cause the setup to fail/revert.

Recommendation

Change the names of the two variables.

Resolution

GMX Team: Resolved.

L-02 | Users Pay Extra in Fees In Certain Markets

Category	Severity	Location	Status
Documentation	Low	GasUtils.sol	Acknowledged

Description

`estimatedDepositOraclePriceCount()`, `estimatedWithdrawalOraclePriceCount()`, `estimateOrderOraclePriceCount()`, & `estimateShiftOraclePriceCount()` make an assumption that the long, short, and index tokens will all be different tokens. This is not always the case since index token, long token, and short token can be the same. The oracle price count is then multiplied by `EXECUTION_GAS_FEE_PER_ORACLE_PRICE` and added to the Keeper's fee. This will charge users unnecessary fees with each interaction to the protocol.

Recommendation

Store a variable that tracks the amount of tokens that have their prices set. Then when calculating the Keeper's fee, utilize this value.

Resolution

GMX Team: Acknowledged.

L-03 | Total AutoCancel Gas Supersedes Max Auto Cancels

Category	Severity	Location	Status
Documentation	Low	Global	Acknowledged

Description

The maximum amount of auto cancels multiplied by the max callback gas limit is greater than the max total callback gas limit for auto cancels. This can be an issue for users and integrators who are not aware of this caveat, and attempt to add the maximum amount of auto cancels to a position.

Recommendation

Be sure to document this behavior to alert users and integrators of this scenario.

Resolution

GMX Team: Acknowledged.

L-04 | Callback And Refund Receiver Risks

Category	Severity	Location	Status
Documentation	Low	Global	Acknowledged

Description

Because the funds are sent to the `account` instead of the callback contract when an order is cancelled it could be unexpected for users and integrating protocols, making it become difficult for the callback contract to handle these funds as they would receive the executionFee refund, but not the input token amount for deposits, withdrawals, or orders.

Recommendation

Document this behavior so integrators and users can build accordingly.

Resolution

GMX Team: Acknowledged.

L-05 | Incorrect Oracle Price Estimate

Category	Severity	Location	Status
Logical Error	Low	DepositUtils.sol: 136	Resolved

Description

createDeposit() calls estimatedWithdrawalOraclePriceCount(). The logic is the same as estimatedDepositOraclePriceCount(), so there is no impact, however the naming convention is wrong.

Recommendation

Switch estimatedWithdrawalOraclePriceCount() to estimatedDepositOraclePriceCount() in createDeposit().

Resolution

GMX Team: Resolved.

L-06 | Inconsistent AutoCancel Validation

Category	Severity	Location	Status
Unexpected Behavior	Low	OrderUtils.sol: 159	Resolved

Description

To verify update the auto cancel list an order must either be a `LimitDecrease` or `StopLossDecrease` with the new change to `updateAutoCancelList()`. However, `isDecrease()` is still used in `createOrder`. This will cause `MarketDecrease` orders to call `validateTotalCallbackGasLimitForAutoCancelOrders()` when it is unnecessary.

Recommendation

Use the same check from `updateAutoCancelList()` in `createOrder()`.

Resolution

GMX Team: Resolved.

L-07 | Optimal Usage Borrowing Can Remain Constant

Category	Severity	Location	Status
Configuration	Low	MarketUtils.sol: 2458	Acknowledged

Description

The `additionalBorrowingFactorPerSecond` in the `getKinkBorrowingFactor` function is initialized to 0 and is only changed if `aboveOptimalUsageBorrowingFactor` is less than or equal to `baseBorrowingFactor`. Therefore the `borrowingFactorPerSecond` will not grow since multiplication by 0 will cause `additionalBorrowingFactorPerSecond * diff / divisor` to be 0.

Recommendation

Consider verifying that the `aboveOptimalUsageBorrowingFactor` is always be greater than `baseBorrowingFactor` upon configuration.

Resolution

GMX Team: Acknowledged.

L-08 | Incorrect Estimated Price Counts

Category	Severity	Location	Status
Logical Error	Low	GasUtils.sol: 224, 234, 241	Acknowledged

Description

In the GasUtils file the oracle price estimation functions accept a swapCount and add this value to the resulting estimated oracle prices necessary. However the estimation does not accurately account for the prices required for swaps.
For example:

• Consider a deposit to the USDC/WETH market
• longTokenSwapPath = [USDT/ATOM, ATOM/DAI, DAI/WETH]
• shortTokenSwapPath = [SOL/WBTC, WBTC/ARB, ARB/USDC]

In the worst case, all 6 of these markets in the swap path have a different index token, and the deposit market has a unique index token as well. Yielding 7 prices necessary just for index tokens. Then all tokens used in the swapPaths are necessary: [USDT, ATOM, DAI, WETH, SOL, WBTC, ARB, USDC] which adds 8 more potential prices.

While the existing validation assumes that the worst case is 8 prices as mentioned in the comments, and 9 prices as the maximum returnable by the estimatedDepositOraclePriceCount and estimatedWithdrawalOraclePriceCount functions — the actual worst case is 15 total prices.

Recommendation

Multiply the swapCount by 2 in each of the estimatedDepositOraclePriceCount, estimatedWithdrawalOraclePriceCount, and estimateOrderOraclePriceCount functions in order to accurately represent the worst case amount of oracle prices required for the action.

Resolution

GMX Team: Acknowledged.

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