GUARDIAN AUDITS

SMART CONTRACT SECURITY AUDIT OF

Reliquary

~A Byte Masons joint.

Summary

Audit Firm	Guardian Audits
Client Firm	Byte Masons
Final Report Date	**Preliminary Report**

Audit Summary

After a line by line manual analysis and automated review, Guardian Audits has concluded that:

- Reliquary's smart contracts have a LOW RISK SEVERITY
- Reliquary's smart contracts have an **ACTIVE OWNERSHIP**
- Important operator privileges – `setEmissionSetter`, `addPool`, `modifyPool`
- Reliquary's smart contract operator has multiple "write" privileges. Centralization risk correlated to the active ownership is **MEDIUM**

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: **Fantom Opera**

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

Table of Contents

Project Overview

Project Summary

Project Name	Reliquary
Language	Solidity
Codebase	https://github.com/Byte-Masons/Reliquary
Commit	11a2e1343004d7571e283dd47b3bae4a03be8696

Audit Summary

Delivery Date	Preliminary Report
Audit Methodology	Static Analysis, Manual Review, Fuzzing, Penetration Tests

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0
● High	0	0	0	0	0	0
● Medium	6	6	0	0	0	0
● Low	3	3	0	0	0	0

Audit Scope & Methodology

Scope

ID	File	SHA-1 Checksum
DESC	NFTDescriptor.sol	FE53DAE384C389EA419BF8D27F9E57186ED2D3CA
RLC	Relic.sol	24447DEEF8245F0572BA2916E530FC66B6E6B254
RLQ	Reliquary.sol	8989F4BC607B8EF860F8B77896419AEEA4DA8575
RLD	ReliquaryData.sol	9EA083981F574B4DE1F9C2C1CB23052D507C9A02
REW	Rewarder.sol	8E8592E2FB3D8A6C871215ADC12C4B86D6B4062C

Methodology

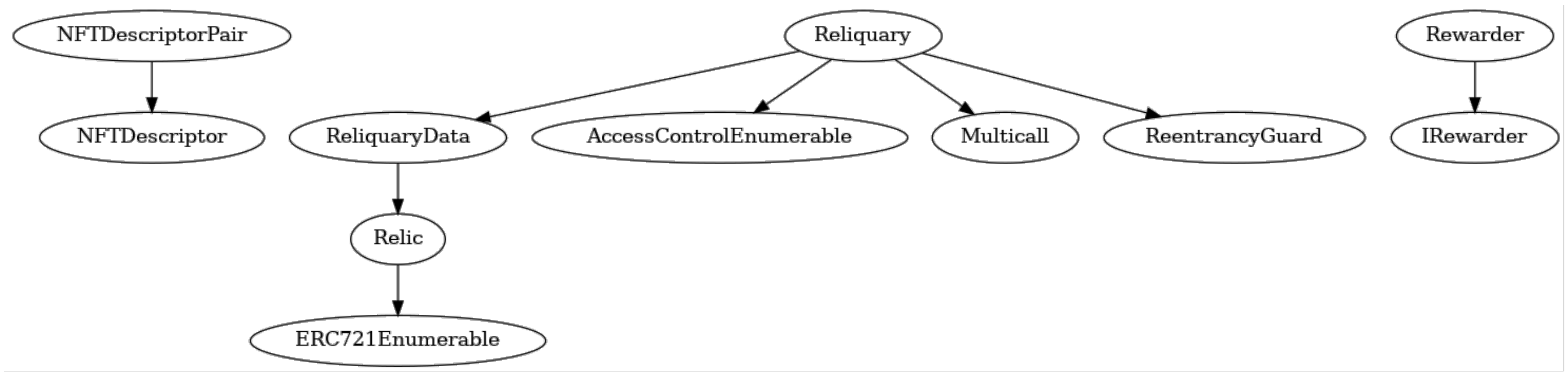
The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- 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.
- Contract fuzzing for increased attack resilience.
- 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.

Inheritance Graph



Findings & Resolutions

ID	Title	Category	Severity	Status
GLOBAL-1	Centralization Risk	Centralization / Privilege	● Medium	Unresolved
RLQ-1	Inefficient Reward Design	Tokenomics / Rewards	● Medium	Unresolved
RLQ-2	Withdrawal Maturity Setback	Tokenomics / Rewards	● Medium	Unresolved
RLQ-3	Burn-on-Transfer Tokens	Deflationary Tokens	● Medium	Unresolved
RLQ-4	Incorrect Truncation Avoidance	Logical Error	● Medium	Unresolved
RLQ-5	Unclear Naming	Code Documentation	● Low	Unresolved
RLQ-6	Redundant Pool ID Read	Optimization	● Low	Unresolved
DESC-1	Bad Decimal String UX	User Experience	● Low	Unresolved
REW-1	Lost Rewards Upon Withdrawal	Tokenomics / Rewards	● Medium	Unresolved

Global-1 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Medium	Global	Unresolved

Description

Elevated privilege leads to multiple DoS attack vectors:

- When a harvest is called, `_updatePosition` relies on there being enough OATH in the contract to cover the relic's accumulated rewards. If there is an insufficient amount of OATH, users would be unable to withdraw without giving up their entire rewards.

- The `operator` role is able to add and modify the `rewarder` for a given pool. If the `rewarder` lacks the `rewardToken` funds to pay out a relic or reverts in some other manner, users would be unable to harvest their rewards.

- If an operator sets an `emissionSetter` that specifies an emission rate greater than the upper bound of 6e18, the pool will not be able to be updated so no position would be able to be updated. Therefore deposits and regular withdrawals would be halted due to their reliance on `_updatePosition`.

Recommendation

Ideally only multi-sig addresses are granted the `operator` role and/or introduce a timelock for improved community oversight.

Resolution

RLQ-1 | Inefficient Reward Design

Category	Severity	Location	Status
Tokenomics / Rewards	● Medium	Reliquary.sol	Unresolved

Description

If userA has their funds deposited but does not call updatePosition for a significant period of time, the next time they update their position they receive rewards based on the level of their position the last time it was updated. In this case userA earns less than userB, who regularly calls updatePosition as soon as their relic's maturity reaches the next level and therefore received more rewards at the higher levels.

Recommendation

If this is not desired behavior, consider evaluating the current level of the user's relic in _updatePosition for distributing rewards, therefore removing the need to call updatePosition regularly. If this approach is taken, consider introducing logic that prevents users from simply holding their positions to receive all of their rewards based on a higher level, when in reality a portion of those rewards should have been weighted at a lower level.

Resolution

RLQ-2 | Withdrawal Maturity Setback

Category	Severity	Location	Status
Tokenomics / Rewards	● Medium	Reliquary.sol	Unresolved

Description

Because the position.entry increases on withdraw in _updateEntry, the position’s maturity decreases. As a result, when _updateLevel is called, a user’s position may be set back to a lower level and most likely a lower allocation. If the user’s whole position reached a certain level, there is no consequence to having the post-withdraw amount stay at that level.

Recommendation

If this is intended behavior, leave as is. Otherwise, do not allow for position.level to be decreased on withdraw until the whole position is withdrawn and the relic is burned.

Resolution

RLQ-3 | Burn-on-Transfer Tokens

Category	Severity	Location	Status
Deflationary Tokens	● Medium	Reliquary.sol	Unresolved

Description

Deposits and withdrawals are not capable of handling certain deflationary tokens. A burn-on-transfer token, complying with the `IERC20` interface of `_lpToken`, would make the `position.amount` seem higher than the true amount for a given relic. As a result, `emergencyWithdraw` would surely fail as there are not enough funds to cover the transfer unless someone else’s deposit is eaten into.

Recommendation

Verify whether support for burn-on-transfer tokens is desired. If so, compare balance after a transfer to the balance before transfer to see the amount that is received in the contract.

Resolution

RLQ-4 | Incorrect Truncation Avoidance

Category	Severity	Location	Status
Logical Error	● Medium	Reliquary.sol:508-512	Unresolved

Description

Lines 508 and 511-512 are mathematically equivalent, but they should be swapped in order to avoid truncation.

An example to demonstrate this:

If oldValue was 1 wei but addedValue was 1 ether, the else if branch on line 510 would be entered. weightOld would be rounded down to 0 and weightNew would be exactly 1e18. However, if the branch on line 508 were to be entered instead, you would get slightly less than 1e18 as the weightNew - providing a more accurate result.

Vice versa, if oldValue was 1 ether but addedValue was 1 wei, the if branch on line 508 would be entered. In this case, the weightNew produced would be 0 as the denominator is larger than the numerator. However, if line 510 was entered, the weightNew produced would be 1 - providing a more accurate result.

Recommendation

Swap the weight calculation logic between lines 508 and 511-512 in order to avoid the truncation inaccuracy.

Resolution

RLQ-5 | Unclear Naming

Category	Severity	Location	Status
Code Documentation	● Low	Reliquary.sol	Unresolved

Description

Although _poolBalance's notice states that it returns "The total deposits of the pool's token", it weighs the balance of each level of the pool by its corresponding allocation. Because it doesn't simply sum up the balance of each level, it doesn't reflect the total deposits. Rather, it reflects the balance of the pool adjusted by level allocation.

Recommendation

Update the comment to accurately reflect what the function is returning.

Resolution

RLQ-6 | Redundant Pool ID Read

Category	Severity	Location	Status
Optimization	● Low	Reliquary.sol: 397	Unresolved

Description

In line 397, `position.poolId` is accessed yet the value is already stored in the variable `poolId`.

Recommendation

Replace `position.poolId` with `poolId` for the transfer.

Resolution

DESC-1 | Bad Decimal String UX

Category	Severity	Location	Status
User Experience	● Low	NFTDescriptor.sol	Unresolved

Description

There are a few instance where `generateDecimalString` produces strings which may be further refined.

- If the decimals argument is set to 0, a number with an appended decimal point is produced such as “1.”.
- The function does not remove trailing zeroes. For example, `generateDecimalString(10, 2)` produces “0.10”.
- The function produces varying strings for equivalent numbers. For example, `generateDecimalString(10, 1)` produces “1.0” while `generateDecimalString(1, 0)` produces “1.”.

Recommendation

Verify that this is expected behavior. Otherwise, rectify the string generation using https://gist.github.com/wilsoncusack/d2e680e0f961e36393d1bf0b6faafba7 or Uniswap’s NFTDescriptor as a model.

Resolution

REW-6 | Lost Rewards Upon Withdrawal

Category	Severity	Location	Status
Tokenomics / Rewards	● Medium	Rewarder.sol	Unresolved

Description

If you deposit but then withdraw even a tiny amount before the `cadence` is reached, the `lastDepositTime` is reset to 0 and you must deposit again in order to be able to claim a deposit bonus.

Recommendation

Verify whether this is expected behavior. If not, either provide a weighted bonus or in the withdraw function `delete` the `lastDepositTime` only if `_claimDepositBonus` returns `true` or the user's new position amount is less than the minimum.

Resolution

Auditor’s Verdict

After a line by line manual analysis and automated review, Guardian Audits has concluded that:

- Reliquary’s smart contracts have a LOW RISK SEVERITY
- Reliquary’s smart contracts have an **ACTIVE OWNERSHIP**
- Important operator privileges – `setEmissionSetter`, `addPool`, `modifyPool`
- Reliquary’s smart contract operator has multiple “write” privileges. Centralization risk correlated to the active ownership is **MEDIUM**

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