macro

Seven Seas A-4

Security Audit

March 20, 2024
Version 1.0.0

Presented by 0xMacro

Table of Contents

Introduction

This document includes the results of the security audit for Seven Seas's smart contract code as found in the section titled ‘Source Code’. The security audit was performed by the Macro security team from April 1st to April 5th 2024.

The purpose of this audit is to review the source code of certain Seven Seas Solidity contracts, and provide feedback on the design, architecture, and quality of the source code with an emphasis on validating the correctness and security of the software in its entirety.

Disclaimer: While Macro’s review is comprehensive and has surfaced some changes that should be made to the source code, this audit should not solely be relied upon for security, as no single audit is guaranteed to catch all possible bugs.

Overall Assessment

The following is an aggregation of issues found by the Macro Audit team:

Severity	Count	Acknowledged	Won't Do	Addressed
Medium	4	-	-	4
Low	4	1	-	3
Code Quality	4	1	-	3
Informational	2	-	-	-

Seven Seas was quick to respond to these issues.

Specification

Our understanding of the specification was based on the following sources:

- Discussions with the Seven Seas team.
- Available documentation in the repository.

Trust Model, Assumptions, and Accepted Risks (TMAAR)

Architecture Roles:

- Manager:

  Intended to be the `manageVaultWithMerkleVerification` contract, which allows verified callers to manage the `BoringVault` with calls verified to be allowed via a set merkle tree.

- Minter:

  Intended to be set to be the `TellerWithMultiAssetSupport` contract to mint new shares at a rate provided by the minter by calling `enter()` on the `BoringVault` .

- Burner:

  Intended to be set to be the `TellerWithMultiAssetSupport` contract to burn shares in return for assets in the `BoringVault` at a rate provided by the burner.

- Internal Manager:

  Will be set to be the `manageVaultWithMerkleVerification` contract itself to call manageVaultWithMerkleVerification after receiving a balancer flashloan via the `receiveFlashloan()` function.

- Solver:

  Will be a contract that implements the `IAtomicSolver` interface, and controlled by an address with the strategist role. Can call `bulkDeposit()` and `bulkWithdrawal()` on the `TellerWithMultiAssetSupport` contract.

- Micromanager:

  Role intended for micro manager contracts such as `DexSwapperUManager` and `DexAggregatorUManager` to call `manageVaultWithMerkleVerification()` on the `ManagerWithMerkleVerification` contract.

Timelock Roles:

- Owner:

  This is an address that controls setting, updating, adding or removing important values for the `BoringVault` , `AccountantWithRateProviders` , `ManagerWithMerkleVerification` , `TellerWithMultiAssetSupport` , `DexAggregatorUManager` , and `DexSwapperUManager` contracts. It also controls the authority contract that determines the functions that roles can call throughout the protocol. This role is intended to be a timelocked multisig contract, with signers on the seven seas team as well as members of associated protocol.

EOA roles:

- Multisig:

  A multisig contract that has some administrative functions with the ability to more quickly interact with the protocol than the timelocked owner does. Can pause and unpause the `AccountantWithRateProviders` , `TellerWithMultiAssetSupport` , and `ManagerWithMerkleVerification` contracts as well as well as set allowed slippage, period and rate limit of the `DexSwapperUManager` and `DexAggregatorUManager` contracts. Controlled by signers on the seven seas team as well as members of associated protocol.

- Strategist multisig:

  Multisig that can call `refundDeposit()` on the `TellerWithMultiAssetSupport()` to return assets from a deposit that may have acted maliciously, like in cases of major arbitrage. Controlled by members of Seven Seas.

- Strategist:

  Main caller of `manageVaultWithMerkleVerification()` on the `ManagerWithMerkleVerification` contract, and may be able to call micro manager functions. Held by members of Seven Seas

- Update exchange rate:

  Role that allows setting the exchange rate in the `AccountantWithRateProviders` via the `updateExchangeRate()` function.

Trust Assumptions:

- It is assumed that all permissioned roles mentioned above will act in the best interest of the vaults it’s shareholders. Despite some control measures, they do have the ability to act outside this interest, so trust is required before investing in the vault.

Accepted Risks:

- Investing in a vault does not guarantee a positive return, and depending on the strategies employed and the state of the market, losses can occur.

Source Code

The following source code was reviewed during the audit:

- **Repository:** boring-vault
- **Commit Hash:** `939c77e25473dff3ed18fa104f004f7afd13452e`

Specifically, we audited the following contracts within this repository.

Contract	SHA256
src/atomic-queue/AtomicQueue.sol	`6c4bfc962181782185acbd638a2c2264f136d4901cf619078bc112c2a066e2fc`
src/atomic-queue/AtomicSolver.sol	`f54be47be35176cda73d5a5dcd8c6e925feedc66a31e6cdc0b24a2b7275da946`
src/atomic-queue/IAtomicSolver.sol	`0c561124046b7fe07f34b9df914b73dce833bb0316945a2d57575856b35defc4`
src/base/BoringVault.sol	`b0edb5d795fcb81124e46355670ea9ddf40b93690497c6b2dd524e58f6944a7d`
src/base/DecodersAndSanitizers/BaseDecoderAndSanitizer.sol	`59e9e2da41b76807eae5ad2fcc6adc45dd35dc974733c0b7266eb88979d9a293`
src/base/DecodersAndSanitizers/EtherFiLiquidDecoderAndSanitizer.sol	`c56d628745bc482373758d92666fc22b415634c02c345fb86616957d1f8800c5`
src/base/DecodersAndSanitizers/LidoLiquidDecoderAndSanitizer.sol	`12300763fd14e0c9e043753414d8fe1d8356bc0c7fd080cba53fdc9dd221fb85`
src/base/DecodersAndSanitizers/Protocols/AaveV3DecoderAndSanitizer.sol	`ee2cf9e87bb4877ec2578365d6c3fdabd8f7c799a7b57dcd14523fb03bba1d4f`
src/base/DecodersAndSanitizers/Protocols/AuraDecoderAndSanitizer.sol	`d89a171943ab7b21297ec40f2c2a48f83fba973c0ff0bbd738cd7b53f1eddf90`

Contract	SHA256
src/base/DecodersAndSanitizers/Protocols/ BalancerV2DecoderAndSanitizer.sol	c10d52e51e641658cc5915781103d1672a 3f6afabc1f763726b84c4cbdb6173a
src/base/DecodersAndSanitizers/Protocols/ ConvexDecoderAndSanitizer.sol	5d25a0715f1967451511d21cd4b0a9371b 590709ac22247a1aa2f0256a4e617a
src/base/DecodersAndSanitizers/Protocols/ CurveDecoderAndSanitizer.sol	c4a44c5a0ab672a7a04c20cff7ed41340e ef4fe81fb603d539bedea435714d14
src/base/DecodersAndSanitizers/Protocols/ ERC4626DecoderAndSanitizer.sol	1a167df426e44540eefd69249075d14d54 02a86a4558d425aa4e6b3ddd42d94e
src/base/DecodersAndSanitizers/Protocols/ EtherFiDecoderAndSanitizer.sol	9a2f3a42ac7a00f327755d308d7e96679c 3b6eef79dd6dac5ed61353ab9b37ac
src/base/DecodersAndSanitizers/Protocols/ GearboxDecoderAndSanitizer.sol	54f09e65d394c13c26e958ce42fc077e85 2f2a37ff9765823410683bd34f320e
src/base/DecodersAndSanitizers/Protocols/ LidoDecoderAndSanitizer.sol	67d2c958b914f1add251af6e14e02da15f ce81d0b9514a3fc071417817935f58
src/base/DecodersAndSanitizers/Protocols/ MorphoBlueDecoderAndSanitizer.sol	b00d406c21837d3a82a24b214be5776f36 5bbfa1b5403ed25c4d46f424fbb41a
src/base/DecodersAndSanitizers/Protocols/ NativeWrapperDecoderAndSanitizer.sol	fa41c17b07410dc15450569fc6ee9eb45c f189e19613fd376586837e62ae52dd
src/base/DecodersAndSanitizers/Protocols/ OneInchDecoderAndSanitizer.sol	18964becdb3b9c9e317990f6f1bccf72b4 4460d25b00f310eb3541e512ba5977
src/base/DecodersAndSanitizers/Protocols/ PendleRouterDecoderAndSanitizer.sol	c0208cccf5f0ebef4fba7636284dbe76bb f06248fcd0ac23e8eba8347f481515
src/base/DecodersAndSanitizers/Protocols/ UniswapV3DecoderAndSanitizer.sol	a474b26c2856c1fbc3727cb6f877c081b1 0c7349352c8ef7adf4eb4e2e92a13f

Contract	SHA256
src/base/Roles/AccountantWithRateProviders.sol	5afb5aa5b96fa71da3d1fc3564dd5b54459526b15f1ce2d1cc16e73d832ad93f
src/base/Roles/ManagerWithMerkleVerification.sol	358a56c2035f67158d02804d4d4b1966704ceb7a4232df8e1e7e1be80c8d9062
src/base/Roles/TellerWithMultiAssetSupport.sol	e6633b4fed0a5d5cf2c4ffc66905390a93f2e9cfd0858371e0b03f17d1fa63f7
src/helper/BoringVaultV0Lens.sol	cb5c00603aec0d26ac309ab2bb2807a049dd37e682f0ea4fa9dbc949b34bbf23
src/interfaces/AggregationRouterV5.sol	20d6520f87de266c02779d583dd5ebd899bddbdf97460d6020ebf93e41af0da6
src/interfaces/BalancerVault.sol	210c23c5dfa783273fb5f7cbdb912b5bedb762580a50cf65153aea37462ff6bb
src/interfaces/BeforeTransferHook.sol	a839387d607f8525e520e053cb818d09934d13fe29560efdfa9777835820c5da
src/interfaces/DecoderCustomTypes.sol	ec0a94b71df0e7a14282332b2dab72da26cb88056b04c03b51c132e2bf8966d2
src/interfaces/EtherFiLiquid1.sol	a1efd56fc6f451f7afa61f5ff1d26cbace8600548ab533b6957e4898e8b83da2
src/interfaces/IRateProvider.sol	209b0d3b89a45a2edad0120c699272223b9df0cdca924857bce3cbdd07e1c6d7
src/interfaces/IStaking.sol	da66b6453561c018b73bc788e19e798405ef48f88f470e0772eb95176a199414
src/interfaces/IUniswapV3Router.sol	07e54f895d2e96a922fd26ed7936c3ca432047815f40b9651d613cb73fcb8d81
src/interfaces/PriceRouter.sol	c284dee00354274fe8e460b089d9b5416d867943716b1eecf789ae907c6f45b7

Contract	SHA256
src/interfaces/RawDataDecoderAndSanitizerInterfaces.sol	29da7215cfe59a69bdd97c29b39e4ca3ed9278adc74870089b004953c1398d55
src/micro-managers/DexAggregatorUManager.sol	8a50ecd266f430e85847568cf475b69e900fb040d6ba8a6c12d8b84d5dfe20b6
src/micro-managers/DexSwapperUManager.sol	95304d5eb86d40a418e9ebf9d9efebd9e9a54f20b2d2ec71817a2231fd911e0f
src/migration/CellarMigrationAdaptor.sol	dee75c9908b677b7b7c7e10af8086e9ce6e7672feeefaf7438c4bdf9f4cf3b6a

Note: This document contains an audit solely of the Solidity contracts listed above. Specifically, the audit pertains only to the contracts themselves, and does not pertain to any other programs or scripts, including deployment scripts.

Issue Descriptions and Recommendations

Click on an issue to jump to it, or scroll down to see them all.

- ~~M-1~~ Vault manager cannot quickly pause strategist actions
- ~~M-2~~ Token addresses not verified for increasing or decreasing uniswapV3 liquidity
- ~~M-3~~ Bulk deposits and withdrawals missing supported asset check
- ~~M-4~~ Rate calculations may use the wrong decimals
- ~~L-1~~ No transaction deadline set
- ~~L-2~~ Manager flashloan functions authorization should be explicit and immutable
- ~~L-3~~ Teller minimum exchange rate update delay is inflexible
- L-4 Can unsafely borrow from protocols bringing vault close to liquidation
- Q-1 No max available functionality for ease of managing the vault
- ~~Q-2~~ Index relevant event parameters to allow services to easily query
- ~~Q-3~~ Roles intended for authorized functions are unclear
- ~~Q-4~~ Error missing relevant info making debugging difficult
- I-1 Manager safeguards do not entirely prevent malicious actions
- I-2 Vault manages are susceptible to MEV

Security Level Reference

We quantify issues in three parts:

1. The high/medium/low/spec-breaking **impact** of the issue:
   - How bad things can get (for a vulnerability)
   - The significance of an improvement (for a code quality issue)
   - The amount of gas saved (for a gas optimization)
2. The high/medium/low **likelihood** of the issue:
   - How likely is the issue to occur (for a vulnerability)
3. The overall critical/high/medium/low **severity** of the issue.

This third part – the severity level – is a summary of how much consideration the client should give to fixing the issue. We assign severity according to the table of guidelines below:

Severity	Description
(C-x) Critical	We recommend the client **must** fix the issue, no matter what, because not fixing would mean **significant funds/assets WILL be lost.**
(H-x) High	We recommend the client **must** address the issue, no matter what, because not fixing would be very bad, *or* some funds/assets will be lost, *or* the code's behavior is against the provided spec.
(M-x) Medium	We recommend the client to **seriously consider** fixing the issue, as the implications of not fixing the issue are severe enough to impact the project significantly, albiet not in an existential manner.
(L-x) Low	The risk is small, unlikely, or may not relevant to the project in a meaningful way. Whether or not the project wants to develop a fix is up to the goals and needs of the project.
(Q-x) Code Quality	The issue identified does not pose any obvious risk, but fixing could improve overall code quality, on-chain composability, developer ergonomics, or even certain aspects of protocol design.
(I-x) Informational	Warnings and things to keep in mind when operating the protocol. No immediate action required.
(G-x) Gas Optimizations	The presented optimization suggestion would save an amount of gas significant enough, in our opinion, to be worth the development cost of implementing it.

Issue Details

~~M-1~~ Vault manager cannot quickly pause strategist actions

TOPIC	STATUS	IMPACT	LIKELIHOOD
Protocol Design	Fixed	High	Low

The `ManagerWithMerkleVerification` contract allows permissioned addresses, typically strategists, to manage the `BoringVault` after verifying the calls are valid to make. However, in the case where a strategist starts to act against the best interest of the vault, there are no measures in place to quickly mitigate the damage. Currently the owner role can remove the merkle root associated with a strategist, but this action is behind a timelock and cannot act quickly if required.

Remediations to Consider

Allow pausing and unpausing strategists from managing the vault via a non-timelocked trusted role to allow quick action if a strategist goes rogue. Also consider rate limiting strategist actions, to reduce the potential impact of a rogue strategist.

~~M-2~~ Token addresses not verified for increasing or decreasing uniswapV3 liquidity

TOPIC	STATUS	IMPACT	LIKELIHOOD
Validation	Fixed	Medium	Low

In `UniswapV3DecoderAndSanitizer.sol` the functions `increaseLiquidity` and `decreaseLiquidty`, and `collect()`, the tokens in the pool associated with the provided tokenId are not checked to see if they are supported, unlike for the functions `exactInput()` and `mint()`. Although ownership of the provided token id is checked to be the vault, adding liquidity to this pool may be unauthorized seeing as anyone can create a lp position and transfer it to the vault. This allows strategists to take up unexpected lp positions, and or collect unexpected rewards.

Remediations to Consider

Check with the uniswap position manager which tokens are related to the token id, and add these tokens to the returned addressesFound array to be verified via the merkle tree.

~~M-3~~ Bulk deposits and withdrawals missing supported asset check

TOPIC	STATUS	IMPACT	LIKELIHOOD
Error Recovery	Fixed Fixed	Medium	Low

In `TellerWithMultiAssetSupport.sol` the `deposit()` and `depositWithPermit()` functions require the provided `depositAsset` is supported, by checking it with the `isSupported` mapping.

However, the functions `bulkDeposit()` and `bulkWithdraw()` do not make this check. Although both of these functions are permissioned, there may be multiple entities allowed to call these functions, including some select other vaults in the case of the `CellarMigrationAdaptor`, where ensuring the requested assets are allowed to be deposited and or withdrawn.

Remediations to Consider

Check the `isSupported` mapping if the `depositAsset` or `withdrawAsset` is supported before allowing bulk deposit or withdrawals.

~~M-4~~ Rate calculations may use the wrong decimals

TOPIC	STATUS	IMPACT	LIKELIHOOD
Pricing	Fixed	High	Low

The `AccountantWithRateProvider.sol` contract, allows setting an exchange rate of a Boring Vaults shares based in a base asset, using the base assets decimals. This contract also allows getting the exchange rate denominated in a provided quote asset, if it is supported, by calling getRateInQuote(), or a strategist can claimFees() in a provided asset. In both these cases the exchange rate needs to be converted from a rate of base/share to quote asset/share. However, some assets are pegged to the base asset, meaning their price is tied and assumed to be exchanged 1:1, but currently it is assumed that their decimals are also the same:

```
...
if (address(feeAsset) == address(base) || data.isPeggedToBase) {
    feesOwedInFeeAsset = state.feesOwedInBase;
}
...
```

Reference: AccountantWithRateProvider.sol#L284-286

```
 if (data.isPeggedToBase) {
      rateInQuote = accountantState.exchangeRate;
  }
```

Reference: AccountantWithRateProvider.sol#L325-327

If the decimals are different for the quote asset than the base it can lead to an inaccurate value. This is also the case for non-pegged assets, where the exchange rate is determined by calling `getRate()` on an associated rate provider, but is is assumed that the quote asset has the same decimals as the base asset as well, leading to the same issue.

Using inaccurate decimals for these functions results in inaccurate fees claimed or the rate returned, leading to either more or less fees collected, or more or less shares received when depositing. This can be remedied with refunding deposits if caught, and assuming a strategist is trusted could have inaccurate fees received covered or returned.

Remediations to Consider

Convert these values from the base assets decimals to the quote assets decimals for both pegged and non-pegged assets when claiming fees or getting the exchange rate in a quote asset.

~~L-1~~ No transaction deadline set

TOPIC	STATUS	IMPACT	LIKELIHOOD
Protocol Design	Fixed	Low	Low

When a transaction gets submitted to an RPC it is typically added to the mempool and can stay there for a indefinite amount of time, and can be delayed for a number of reasons. Sometimes malicious MEV/block builders may hold off executing a transaction until it suits them. Other times the gas price submitted may not be sufficient to compete to other demand, and execution may be held until competitive gas prices drop back down, but for time or condition sensitive transactions, this delay may result in the user no longer wanting to execute the transaction.

Protocols like UniswapV3 take an execution deadline as a parameter to give users control over this, however in the `DexSwapperUManager` micro manager, the deadline is set to the `block.timestamp` , which means it will always be a valid time, negating it set purpose:**

```
 IUniswapV3Router.ExactInputParams memory params = IUniswapV3Router.ExactInputI
    path: packedPath,
    recipient: boringVault,
    deadline: block.timestamp,
    amountIn: amountIn,
    amountOutMinimum: amountOutMinimum
});
```

Reference: DexSwapperUManager.sol#L115-L121

Remediations to Consider

Add an additional deadline parameter to `ManagerWithMerkleVerification.sol` 's `manageVaultWithMerkleVerification()` function, and require that the deadline hasn't been exceeded, to give control over the timing of vault managements. Alternatively, for protocol interactions that are known to be timing sensitive, require a deadline in its corresponding micro manager.

~~L-2~~ Manager flashloan functions authorization should be explicit and immutable

TOPIC	STATUS	IMPACT	LIKELIHOOD
Roles	Fixed Fixed	Medium	Low

In `ManagerWithMerkleVerification.sol` , the function `flashLoan()` is intended to be called by the `BoringVault` that it manages, after being initiated by manageVaultWithMerkleVerification(), which initiates a balancer flashloan where the balancer vault calls receiveFlashloan(). Both of these functions have a explicit intended caller that should not change, and these addresses are currently immutable values within the `ManagerWithMerkleVerification` contract. However, authorization is determined using the `onlyAuth` modifier which allows for the authorized addresses to change.

Remediations to Consider

Instead of using the `onlyAuth` modifier for `flashLoan()` and `receiveFlashloan()` , explicitly require the caller to be the `vault` and `balancerVault` addresses respectively.

~~L-3~~ Teller minimum exchange rate update delay is inflexible

TOPIC	STATUS	IMPACT	LIKELIHOOD
Protocol Design	Fixed Fixed	Low	Low

In `AccountantWithRateProviders.sol` , a permissioned entity is allowed to call updateExchangeRate() to update the vaults exchange rate, based on the value of assets in the vault per share. They are limited to only updating the exchange rate every so often and is defined by the `minimumUpdateDelayInHours` , otherwise attempting to update it before will can cause the vault to be paused:

```
if (
    currentTime < state.lastUpdateTimestamp + (state.minimumUpdateDelayInHours
    || newExchangeRate < currentExchangeRate.mulDivDown(state.allowedExchangeRa
) {
    // Instead of reverting, pause the contract. This way the exchange rate upd
    // to a better value, and pause it.
```

```
        state.isPaused = true;
    }
```

Reference: AccountantWithRateProviders.sol#L237-L244

This delay is limited to being defined in hours, which may not be as precise as a vault requires. Currently the `minimumUpdateDelayInHours` is packed in a storage slot as a unit8, but there is enough room in the slot to be up to a uint40 which could allow this delay to be reasonably expressed in seconds.

Remediations to Consider

Adjust the minimum delay to be defined in seconds to give more flexibility to the exchange rate updater.

L-4 Can unsafely borrow from protocols bringing vault close to liquidation

TOPIC	STATUS	IMPACT	LIKELIHOOD
Liquidation	Acknowledged	Medium	Low

BoringVaults are setup to potentially borrow assets from protocols like Morpho blue and Aave by posting collateral. Borrowing assets is risky, since the collateral put up has to be worth more than the value of the borrowed amount, otherwise the collateral will be liquidated. Most protocols ensure the value of collateral to loan, the health factor, is above the liquidation threshold, however setting an unhealthy health factor adds a lot of risk to the vault of losing assets from having the position liquidated, which may want to be prevented. The assumption is that strategists will act to the vaults benefit, but without additional checks in place a strategist may create a more risky loan than intended.

Remediations to Consider

Add micro managers that ensure positions maintain a sufficiently high health factor for all loans made to protocols.

RESPONSE BY SEVEN SEAS

> This issue will be addressed in future audits, by using a uManager.

Q-1 No max available functionality for ease of managing the vault

TOPIC	STATUS	QUALITY IMPACT
Quality of Life	Acknowledged	Low

In the prior vault managing protocol, cellar functions allowed strategists to specify they wanted to use all of a particular asset in the vault to do some action, which made interacting with the vault easier than providing exact amounts, which could vary throughout a set of rebalance actions. However, in this version that functionality is missing which is a quality of life decrease from the prior version.

Remediations to Consider

Add the ability for a strategist to specify the max amount of an asset the valut has available to use for relevant actions. Consider handling this in the sanitizers and decoders as an extra argument, and returning this additional data.

RESPONSE BY SEVEN SEAS

> This issue will be addressed in future audits.

~~Q-2~~ Index relevant event parameters to allow services to easily query

TOPIC	STATUS	QUALITY IMPACT
Events	Fixed	Low

Events like `deposit` in the `TellerWithMultiAssetSupport` contract emits relevant deposit info, like the `recipient` , the `depositAsset` which could be used by front-end applications or services to more easily query and display a users actions and assets if these values were indexed.

Remediations to Consider

Index relevant event parameters in events like the teller's `deposit` and the `BoringVault` 's `enter` and `exit` events.

~~Q-3~~ Roles intended for authorized functions are unclear

TOPIC	STATUS	QUALITY IMPACT
Authorization	Fixed	Low

Authorized functions throughout the boring vault repo use solmate's Auth's requiresAuth modifier to check if the sender is authorized to call the specific function. However, there are multiple potential roles that could be authorized but it is not explicitly stated which role is expected to call each function

Remediations to Consider

Add the intended role to the Natspec of each authorized function to make the intention more clear.

~~Q-4~~ Error missing relevant info making debugging difficult

TOPIC	STATUS	QUALITY IMPACT
Errors	Fixed	Low

In `ManagerWithMerkleVerification.sol`'s `manageVaultWithMerkleVerification()` strategists pass in data for function calls they would like the corresponding vault to make, along with a proof that each call is in a merkle tree and thus allowed. When each of these function calls is verified it calls `_verifyCalldata()` which will revert with `ManagerWithMerkleVerification__FailedToVerifyManageProof` if a call is not successfully verified.

However, this error does not return any information about the invalid function call and calls `manageVaultWithMerkleVerification()` may take multiple function calls, which may make it difficult for a strategist to determine which of their calls is invalid.

Remediations to Consider

Add additional parameters to the `ManagerWithMerkleVerification__FailedToVerifyManageProof` that make it clear which call has failed, making it easier for strategists to debug complicated manage calls.

I-1 Manager safeguards do not entirely prevent malicious actions

TOPIC	IMPACT
Informational	Informational ✳

Although every call a `BoringVault` makes is verified by the set merkle tree for a given caller, these verification checks are not a catch all to prevent malicious behaviour from a strategist. Users that invest in the vault still have to trust that the managers will act in the best interest of the vault and it’s shareholders.

RESPONSE BY SEVEN SEAS

> This issue will be addressed by introducing a more decentralized setup to sanitize rebalance TXs.

I-2 Vault manages are susceptible to MEV

TOPIC	IMPACT
MEV	Informational ✳

Since the vault does not check the value held in the vault before and after a call to manage, there could be protocol interactions that could be exploited via MEV, such as sandwich attacks or front-running, that may negatively impact the vault. It is important that this is considered and all calls to manage the vault be done so through a trusted private RPC to reduces the likelihood of being negatively effected by MEV.

RESPONSE BY SEVEN SEAS

> Rebalance TX will always be submitted using a trusted block builder.

Disclaimer