

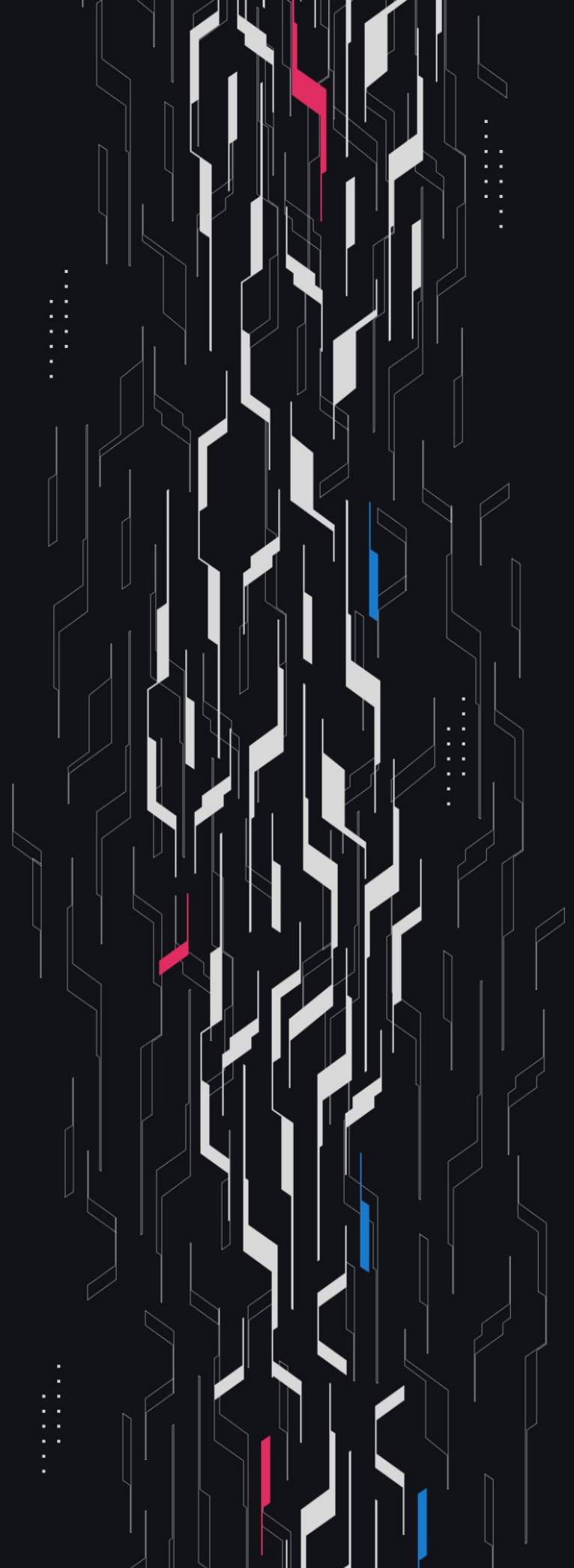
GA GUARDIAN

Bracket

**Bracket Wrapped
Vault Updates**

Security Assessment

September 19th, 2025



Summary

Audit Firm Guardian

Prepared By Owen Thurm, Daniel Gelfand

Client Firm Bracket

Final Report Date September 19, 2025

Audit Summary

Bracket engaged Guardian to review the security of their Bracket Wrapped Vault Updates. From the 12th of September to the 18th of September, a team of 2 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Confidence Ranking

Given the number of non-critical issues detected and code changes following the main review, Guardian assigns a Confidence Ranking of 3 to the protocol. Guardian advises the protocol to address issues thoroughly and consider a targeted follow-up audit depending on code changes. For detailed understanding of the Guardian Confidence Ranking, please see the rubric on the following page.

 Blockchain network: **Ethereum**

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

Guardian Confidence Ranking

Confidence Ranking	Definition and Recommendation	Risk Profile
5: Very High Confidence	<p>Codebase is mature, clean, and secure. No High or Critical vulnerabilities were found. Follows modern best practices with high test coverage and thoughtful design.</p> <p>Recommendation: Code is highly secure at time of audit. Low risk of latent critical issues.</p>	0 High/Critical findings and few Low/Medium severity findings.
4: High Confidence	<p>Code is clean, well-structured, and adheres to best practices. Only Low or Medium-severity issues were discovered. Design patterns are sound, and test coverage is reasonable. Small changes, such as modifying rounding logic, may introduce new vulnerabilities and should be carefully reviewed.</p> <p>Recommendation: Suitable for deployment after remediations; consider periodic review with changes.</p>	0 High/Critical findings. Varied Low/Medium severity findings.
3: Moderate Confidence	<p>Medium-severity and occasional High-severity issues found. Code is functional, but there are concerning areas (e.g., weak modularity, risky patterns). No critical design flaws, though some patterns could lead to issues in edge cases.</p> <p>Recommendation: Address issues thoroughly and consider a targeted follow-up audit depending on code changes.</p>	1 High finding and ≥ 3 Medium. Varied Low severity findings.
2: Low Confidence	<p>Code shows frequent emergence of Critical/High vulnerabilities (~2/week). Audit revealed recurring anti-patterns, weak test coverage, or unclear logic. These characteristics suggest a high likelihood of latent issues.</p> <p>Recommendation: Post-audit development and a second audit cycle are strongly advised.</p>	2-4 High/Critical findings per engagement week.
1: Very Low Confidence	<p>Code has systemic issues. Multiple High/Critical findings (≥ 5/week), poor security posture, and design flaws that introduce compounding risks. Safety cannot be assured.</p> <p>Recommendation: Halt deployment and seek a comprehensive re-audit after substantial refactoring.</p>	≥ 5 High/Critical findings and overall systemic flaws.

Table of Contents

Project Information

Project Overview 5

Audit Scope & Methodology 6

Smart Contract Risk Assessment

Findings & Resolutions 9

Addendum

Disclaimer 46

About Guardian 47

Project Overview

Project Summary

Project Name	Bracket
Language	Solidity
Codebase	https://github.com/bracket-fi/core-contracts
Commit(s)	Initial commit: dbc687a4fc86e3b006b02af46c16fba0526f49fa Final commit: d8185fdcc63f134397da1a91dd8a9f8487588cc1

Audit Summary

Delivery Date	September 19, 2025
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	8	0	0	0	0	8
● Medium	8	0	0	5	0	3
● Low	12	0	0	6	0	6
● Info	3	0	0	1	0	2

Audit Scope & Methodology

Scope and details:

<https://github.com/bracket-fi/core-contracts/compare/bec8a4f75517b0b14351864d379aadba962758d9..dbc687a4fc86e3b006b02af46c16fba0526f49fa>

Audit Scope & Methodology

Vulnerability Classifications

Severity	Impact: <i>High</i>	Impact: <i>Medium</i>	Impact: <i>Low</i>
Likelihood: <i>High</i>	● Critical	● High	● Medium
Likelihood: <i>Medium</i>	● High	● Medium	● Low
Likelihood: <i>Low</i>	● 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	Shares Value Transferred	Logical Error	● Critical	Resolved
C-02	Double Share Conversion Perturbs Accounting	Logical Error	● Critical	Resolved
H-01	Missing Pause Functions	Logical Error	● High	Resolved
H-02	Constructor Used For UUPS Contract	Logical Error	● High	Resolved
H-03	Missing Ownership Initialization	Logical Error	● High	Resolved
H-04	Balances Do Not Match Total Supply	Logical Error	● High	Resolved
H-05	Incongruent Rates Used In Wrapping	Logical Error	● High	Resolved
H-06	Upgrades Perturb Balances	Logical Error	● High	Resolved
M-01	Nav Cannot Be Updated From Zero	Logical Error	● Medium	Acknowledged
M-02	Nav Updates May Be Unintentionally Constricted	Validation	● Medium	Acknowledged
L-01	Zero Wraps Allowed	Validation	● Low	Resolved
L-02	Approval Mismatch	Unexpected Behavior	● Low	Resolved
L-03	Misleading Conversion Functions	Warning	● Low	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
L-04	Missing Closed Case	Best Practices	● Low	Resolved
L-05	Duplicated Active State	Unexpected Behavior	● Low	Acknowledged
I-01	Missing Initializers	Best Practices	● Info	Resolved
I-02	Dust Left In BracketWrappedVault.sol	Warning	● Info	Acknowledged

C-01 | Shares Value Transferred

Category	Severity	Location	Status
Logical Error	● Critical	BracketWrappedVault.sol	Resolved

Description

In the `unwrap` function the `transferFrom` invocation transfers the `amount` specified by the user, however this is a shares amount rather than a token amount.

Recommendation

Use the `assets` value in the `transferFrom` invocation instead.

Resolution

Bracket Team: The issue was resolved in commit [b5becff](#).

C-02 | Double Share Conversion Perturbs Accounting

Category	Severity	Location	Status
Logical Error	● Critical	BracketVaultV2.sol	Resolved

Description

The overridden `_update` function in the `RebasingToken` contract has been updated to include a conversion to shares for the amount being transferred. However the vault treats the amount being specified for the transfer/mint/burn action as a share amount.

Therefore in the `_clearDeposit` and `withdraw` functions when the `_mint` and `_burn` functions are called with values that have already been converted to shares, these values are doubly converted to shares thus perturbing the accounting system.

Recommendation

Consider removing the conversion to shares in the `_update` function, otherwise refactor the vault to treat the amounts passed into the transfer/mint/burn to be raw asset amounts rather than share amounts.

Resolution

Bracket Team: The issue was resolved in commit [4c089d6](#).

H-01 | Missing Pause Functions

Category	Severity	Location	Status
Logical Error	● High	BracketWrappedVault.sol	Resolved

Description

The BracketWrappedVault contract inherits from PausableUpgradeable without implementing functions to expose the pause and unpause functionality. As a result the wrap and unwrap functions cannot be paused.

Recommendation

Implement pause and unpause permissioned external functions.

Resolution

Bracket Team: The issue was resolved in commit [8e37544](#).

H-02 | Constructor Used For UUPS Contract

Category	Severity	Location	Status
Logical Error	● High	BracketWrappedVault.sol	Resolved

Description

In the BracketWrappedVault contract the constructor is used to initialize the contract with the correct vault, decimals, name and symbol.

However the BracketWrappedVault contract is a UUPS implementation contract, therefore this assignment should occur in the initialize function to instantiate the storage slots of the proxy contract being used with this implementation contract.

And the constructor of the implementation contract should use disableInitializers to disable the initialization of the UUPS implementation contract.

Recommendation

Move the constructor logic into an initialize function and use disableInitializers in the constructor of the BracketWrappedVault contract.

Resolution

Bracket Team: The issue was resolved in commit [a0409cb](#).

H-03 | Missing Ownership Initialization

Category	Severity	Location	Status
Logical Error	● High	BracketWrappedVault.sol	Resolved

Description

The BracketWrappedVault constructor fails to call the __Ownable_init function of the OwnableUpgradeable base contract.

As a result no owner is assigned for the protocol and upgrades using the UUPS functionality cannot take place.

Recommendation

In the new initialize function for the BracketWrappedVault contract, invoke the __Ownable_init function and assign the appropriate owner.

Resolution

Bracket Team: The issue was resolved in commit [a0409cb](#).

H-04 | Balances Do Not Match Total Supply

Category	Severity	Location	Status
Logical Error	● High	RebasingToken.sol	Resolved

Description

The `activeBalanceOf` has been changed to use the `vanityAssetsPerShare` function which maintains its original behavior of fluctuating the current reported balance of users by the `vanityNav`, however the `activeSupply` function still uses the `assetsPerShare` function which is now reliant on the reported `nav` from the previous epoch.

As a result the reported values from the summation of user balances and the `totalSupply` result will not agree when the `vanityNav` does not match the reported `nav` from the previous epoch.

Recommendation

Update the `activeSupply` function to use the `vanityAssetsPerShare` function instead of the `assetsPerShare` function if it is desired that the supply and balances of users float with the `vanityNav`.

Resolution

Bracket Team: The issue was resolved in commit [4c089d6](#).

H-05 | Incongruent Rates Used In Wrapping

Category	Severity	Location	Status
Logical Error	● High	BracketWrappedVault.sol	Resolved

Description

In the BracketWrappedVault the convertToShares and convertToAssets functions are used with the assetsPerShare rate to convert to and from shares.

However the assetsPerShare rate is not what will be used to determine how many shares ought to be transferred, instead the vanityAssetsPerShare rate is used in the _update function.

This will result in an unexpected amount of shares being transferred when the vanityNav does not match the nav of the last epoch.

Recommendation

Use the vanityAssetsPerShare instead of the assetsPerShare in the wrap and unwrap functions when converting between shares and assets.

Resolution

Bracket Team: The issue was resolved in commit [4c089d6](#).

H-06 | Upgrades Perturb Balances

Category	Severity	Location	Status
Logical Error	● High	RebasingToken.sol	Resolved

Description

In the `_update` function a conversion to shares by the `vanityAssetsPerShare` rate has been introduced.

This may be fine for new vaults which are being introduced, however for existing vaults this will immediately change the interpretation and valuation of entries in the balances mapping.

Recommendation

Do not perform this upgrade for vaults which already exist

Resolution

Bracket Team: Resolved.

M-01 | Nav Cannot Be Updated From Zero

Category	Severity	Location	Status
Logical Error	● Medium	NavUpdater.sol	Acknowledged

Description

In the `_getDiff` function the `_currentNav` is always used in the denominator of the basis points diff calculation.

As a result, if the nav ever goes to zero for any vault and is then updated to a nonzero value, the update will panic revert in the `_getDiff` function due to attempted division by 0.

Recommendation

Consider early returning a default value of 0 from the the `_getDiff` function to avoid division by zero in this case.

Resolution

Bracket Team: Acknowledged.

M-02 | Nav Updates May Be Unintentionally Constricted

Category	Severity	Location	Status
Validation	● Medium	NavUpdater.sol	Acknowledged

Description

The `maxBpsIncrease` value cannot be assigned higher than 10,000 basis points, meaning that any nav increase cannot reflect a doubling in the price of a bracket vault share.

This may inadvertently over-limit the appropriate amount of share value increase that can occur in a given epoch, since it could be reasonable for a fund to double its nav in a week long epoch in certain outlier circumstances.

Recommendation

Consider if the 10,000 basis point cap should hold for the increase validation, or if it should be higher.

Resolution

Bracket Team: Acknowledged.

L-01 | Zero Wraps Allowed

Category	Severity	Location	Status
Validation	● Low	BracketWrappedVault.sol	Resolved

Description

In the `wrap` and `unwrap` functions, amounts of zero are allowed which may lead to unexpected behavior. To limit the unexpected paths that can be taken these values should be validated against.

In the `wrap` function, even non-zero but very small `amount` values can round to a zero `share` amount, so in the `wrap` function the `share` value should be validated to be nonzero.

Recommendation

In the `wrap` function, validate that the `share` value is not zero, and in the `unwrap` function validate that the `amount` value is not zero.

Resolution

Bracket Team: The issue was resolved in commit [c870b1c](#).

L-02 | Approval Mismatch

Category	Severity	Location	Status
Unexpected Behavior	● Low	RebasingToken.sol	Resolved

Description

The approval amounts in the `rebasingToken` contract are tracked and spent based on an asset basis rather than a share basis which leads to a mismatch between the shares which can be transferred relative to the approval amount over time.

For example, if User A approves User B to spend 10 amount at $T = 1$ when the assets per share is 1, then User B can spend 10 shares from User A.

However if at $T = 2$ the assets per share is 2, User B can now only spend 5 shares from User A. This may simply be the expected behavior, since the value of the assets has gone up and so the approval is desired to be in the final amount rather than the shares amount.

However this creates notable behavior when rounding is taken into account, where small approval amounts can be spent without the transferring of actual assets.

For example, if the asset per share rate is $1.1e18$, a user is approved to spend $10e18$ tokens from User A, and they transfer 1 wei of assets from User A the approval will decrease by 1 wei without transferring any assets from User A since the resulting transferred shares value rounds to zero.

Recommendation

Overall these behaviors are innocuous but worth noting and confirming if they are the expected behavior.

Resolution

Bracket Team: The issue was resolved in commit [1afd8e3](#).

L-03 | Misleading Conversion Functions

Category	Severity	Location	Status
Warning	● Low	RebasingToken.sol	Resolved

Description

In the `RebasingToken` contract the `convertToShares` and `convertToAssets` functions which accept only a shares amount have been introduced and are not used anywhere in the codebase. This indicates that they are for usage by integrating protocols or users interacting with the vault.

In this case, it should be noted that these functions may be misleading, since they use the `assetsPerShare` value which is based on the per share rate of the last epoch, which is not what is reflected in the user balances by way of the `vanityAssetsPerShare` rate being used in the `activeBalanceOf` function.

Recommendation

Be aware of this potentially misleading behavior of the `convertToShares` and `convertToAssets` functions.

Resolution

Bracket Team: The issue was resolved in commit [4c089d6](#).

L-04 | Missing Closed Case

Category	Severity	Location	Status
Best Practices	● Low	KYCWhitelist.sol	Resolved

Description

In the `canDeposit` and `canWithdraw` functions there is no case that handles the `CLOSED` `ActionState` nor is there a default return value for the function if none of the explicit cases are matched.

Recommendation

Consider adding a default `return false` statement to silence the compiler warning and implicitly cover the `CLOSED` `ActionState` case.

Resolution

Bracket Team: The issue was resolved in commit [bb258c9](#).

L-05 | Duplicated Active State

Category	Severity	Location	Status
Unexpected Behavior	<div><div></div>Low</div>	NavUpdater.sol	Acknowledged

Description

In the NavUpdater contract there is an active state tracking for each vault, however this mirrors the active state tracking for each vault already done in the KYCWhitelist contract.

Recommendation

Consider if this state can be deduplicated to simply rely on the KYCWhitelist tracking of the active state of a vault.

Resolution

Bracket Team: Acknowledged.

I-01 | Missing Initializers

Category	Severity	Location	Status
Best Practices	● Info	BracketWrappedVault.sol	Resolved

Description

In the constructor for the BracketWrappedVault the initializer invocations are missing for the PausableUpgradeable, and UUPSUpgradeable contracts.

These initializer functions do nothing, however it is a best practice to call these initializer functions in the event that a future version of these base contracts in a future upgrade would have some logic.

Recommendation

Consider calling the initializer functions for the PausableUpgradeable, and UUPSUpgradeable contracts in the new BracketWrappedVault initializer function.

Resolution

Bracket Team: The issue was resolved in commit [a0409cb](#).

I-02 | Dust Left In BracketWrappedVault.sol

Category	Severity	Location	Status
Warning	● Info	BracketWrappedVault.sol	Acknowledged

Description

Due to the round down conversion in the `convertToAssets` function there may be cases where a dust amount of shares are left held by the `BracketWrappedVault` contract after all users have withdrawn.

Recommendation

Simply be aware of this behavior.

Resolution

Bracket Team: Acknowledged.

Remediation Findings & Resolutions

ID	Title	Category	Severity	Status
H-01	Vault Router Avoids KYC	Logical Error	● High	Resolved
H-02	Incorrect assetsPerShare Starting Value	Logical Error	● High	Resolved
M-01	Existing Allowances Change	Warning	● Medium	Acknowledged
M-02	Balances Do Not Match Total Supply	Logical Error	● Medium	Resolved
M-03	KYCWhitelist Storage Corruption	Logical Error	● Medium	Acknowledged
M-04	Ineffective Blacklist	Logical Error	● Medium	Resolved
M-05	Incorrect totalPendingShares Tracking	Logical Error	● Medium	Resolved
M-06	Whitelist Can Be Opened By Any Whitelisted User	Gaming	● Medium	Acknowledged
L-01	User Balances Stepwise Update	Warning	● Low	Acknowledged
L-02	Lock Frequency And Withdraw Delay Validation	Validation	● Low	Resolved
L-03	No Way To Update VanityNav	Warning	● Low	Acknowledged
L-04	Unused Modifier	Superfluous Code	● Low	Resolved
L-05	NavUpdater Deployment Warning	Warning	● Low	Acknowledged

Remediation Findings & Resolutions

ID	Title	Category	Severity	Status
L-06	Balances No Longer Include Inactive Amounts	Warning	<div><div></div>Low</div>	Acknowledged
L-07	totalSupply Rounding	Rounding	<div><div></div>Low</div>	Acknowledged
I-01	Unnecessary Import	Superfluous Code	<div><div></div>Info</div>	Resolved

H-01 | Vault Router Avoids KYC

Category	Severity	Location	Status
Logical Error	● High	VaultRouter.sol	Resolved

Description

The VaultRouter has been introduced to allow users to easily wrap WETH and deposit into the vault with.

However this contract allows users to bypass the KYC restriction because the BracketVault performs validation on the msg.sender in the deposit function, which would be the VaultRouter contract, which must be whitelisted to function.

Recommendation

Either perform the whitelist validation also in the VaultRouter contract or consider removing the VaultRouter contract entirely.

Resolution

Bracket Team: The issue was resolved in commit [89185a9](#).

H-02 | Incorrect assetsPerShare Starting Value

Category	Severity	Location	Status
Logical Error	● High	RebasingToken.sol	Resolved

Description

The BracketVault has a decimals amount of whatever the underlying token has and uses convertToShares to determine the amount of vault tokens to mint.

However the default value for the assetsPerShare is 1e18 when the epoch is 0, this leads to extreme precision loss and ultimately loss of funds for lower decimal vaults.

Recommendation

Report the default assetsPerShare as 10 * _decimals. Furthermore ensure that the updates for the nav for these vaults is in line with the decimals of the vault to avoid loss of funds and precision loss.

Resolution

Bracket Team: The issue was resolved in commit [81962b0](#).

M-01 | Existing Allowances Change

Category	Severity	Location	Status
Warning	● Medium	RebasingToken.sol	Acknowledged

Description

Previously, the `transferFrom` and `transfer` functions accepted `value` parameters that represented `share` values. However now they have been updated to accept `value` parameters that represent `asset` values.

Originally, the approvals consumed in `transferFrom` would be in shares, however now the approvals consumed in `transferFrom` are based on the provided `value` parameter in assets.

So the net effect on approvals of this upgrade for existing vaults is that the approval values made by users will be changed from share values to asset values.

Depending on the NAV of the vault this will either increase or decrease the amount transferable by approvals for users.

Recommendation

Be aware of this change and ensure it will not cause any significant unexpected changes for users. Users should be aware of this change as well.

Resolution

Bracket Team: Acknowledged.

M-02 | Balances Do Not Match Total Supply

Category	Severity	Location	Status
Logical Error	● Medium	RebasingToken.sol	Resolved

Description

The `balanceOf` function has been updated to only rely on the `activeBalance` and ignore the `inactiveBalance`.

This is however contradictory to the implementation of the `totalSupply` function which reports the sum of the `activeSupply` and `inactiveSupply` as the `totalSupply`.

As a result the sum of the individual user balances does not match the aggregate `totalSupply` reported by the `BracketVault`.

Recommendation

If it is desired to only factor in the active shares, then update the `totalSupply` function to only track the `activeSupply`.

Resolution

Bracket Team: The issue was resolved in commit [97c0951](#).

M-03 | KYCWhitelist Storage Corruption

Category	Severity	Location	Status
Logical Error	● Medium	KYCWhitelist.sol	Acknowledged

Description

The KYCWhitelist contract is a UUPSUpgradeable implementation contract, however storage variables have been added before the isUsedNonce mapping which will change the storage values that it is mapped to.

This will reset all of the usedNonce reporting and allow replay of these signatures.

Recommendation

Only append new storage variables after the isUsedNonce mapping.

Resolution

Bracket Team: Acknowledged.

M-04 | Ineffective Blacklist

Category	Severity	Location	Status
Logical Error	● Medium	Global	Resolved

Description

The KYCWhitelist now has a blacklist mode, whereby only blacklisted addresses are considered not whitelisted for deposits and withdrawals.

However this blacklist is easily bypassed by a blacklisted user by transferring their deposit tokens or bracket vault tokens to a non-blacklisted address they control and performing the desired actions with the vault.

Recommendation

Be aware of this loophole and consider performing transfer checks if the KYC whitelist is in blacklist mode.

Resolution

Bracket Team: The issue was resolved in commit [4ee243e](#).

M-05 | Incorrect totalPendingShares Tracking

Category	Severity	Location	Status
Logical Error	● Medium	BracketVaultV2.sol	Resolved

Description

In the BracketVault contract the totalPendingShares variable is tracked to represent the total sum of all pending shares which technically belong to users but have not yet been minted to their underlying balances mapping entry.

The use case for this variable is to ensure that the totalSupply reported is the same as the sum of the user balances. However the tracking of the totalPendingShares does not correctly allow this.

The totalPendingShares variable is adjusted for withdrawals, however this has nothing to do with the pending shares that should be minted to users via deposits and thus perturbs the totalSupply accounting.

For example, the totalSupply is based on the totalMintedShares() + totalNonMintedShares() result. totalNonMintedShares simply returns totalPendingShares for the BracketVault contract.

Whenever a withdrawal is initiated, it is both burned from the totalMintedShares and added to the totalPendingShares. This means there is no net change in the reported totalSupply, which does not agree with the underlying user balances tracking.

Furthermore, the totalPendingShares deduction by the withdrawalShares in the _processDepositsWithdrawals function is also invalid as this has no impact on the pending deposit shares of users.

Recommendation

Remove the deduction of withdrawalShares from the totalPendingShares in the _processDepositsWithdrawals function.

And remove the addition of shares to the totalPendingShares in the withdraw function.

Resolution

Bracket Team: The issue was resolved in commit [48e0fb6](#).

M-06 | Whitelist Can Be Opened By Any Whitelisted User

Category	Severity	Location	Status
Gaming	● Medium	Global	Acknowledged

Description

With the advent of EIP7702 accounts can now create code at their own address. This means that any whitelisted user can open up the whitelisted functionality to be used by non-whitelisted users.

Consider the following scenario:

- Assume protocol has a whitelist mapping, where only users who are whitelisted can call function A
- Bob is whitelisted for his address 0xFF
- Bob uses 7702 to set his 0xFF account code to:

```
contract {
  address victimSystem;
  function callThis(...) external {
    victimSystem.whitelistedFunction(...);
  }
}
```

- Now anyone can call the whitelisted function through calling callThis on 0xFF

Recommendation

Be sure to monitor whitelisted accounts to ensure that they do not add functionality that opens up the whitelist to non-whitelisted accounts. Otherwise consider requiring that msg.sender = tx.origin for the whitelist.

Resolution

Bracket Team: Acknowledged.

L-01 | User Balances Stepwise Update

Category	Severity	Location	Status
Warning	● Low	Global	Acknowledged

Description

The `assetsPerShare` function has been updated to use the NAV value of the last epoch rather than the `vanityNav` of the current epoch.

As a result, for existing vaults this will cause a stepwise jump in the balance reported to users and the reported `totalSupply`.

This may cause unexpected issues for integrations which may rely on this data and may be unexpected for users.

Recommendation

Be sure to document this change for integrators and users who may be impacted.

Resolution

Bracket Team: Acknowledged.

L-02 | Lock Frequency And Withdraw Delay Validation

Category	Severity	Location	Status
Validation	● Low	BracketVaultV2.sol	Resolved

Description

The lock frequency was previously hardcoded at 7 days, however now it is configurable by the admin lite role and upon initialization.

The vault could however benefit from some validations on the frequency to ensure that it is neither configured to long or too short.

For example, the BracketVault contract will stop functioning after `type(uint16).max` epochs, therefore an epoch time of 10 minutes is too low.

Similarly, there is no validation that the configuration of the `withdrawDelay` is greater than some minimum and less than some maximum.

Recommendation

Consider if validations should be introduced which ensure the `lockFrequency` and `withdrawDelay` is bounded between a maximum and minimum range on initialization and updates.

Resolution

Bracket Team: The issue was resolved in commit [4505682](#).

L-03 | No Way To Update VanityNav

Category	Severity	Location	Status
Warning	<div><div></div>Low</div>	BracketVaultV2.sol	Acknowledged

Description

With the new NavUpdater contract there is no way to update the vanityNav with the updateVanityNav function as this function is not invoked in the NavUpdater contract.

Recommendation

Be aware of this unreachable function.

Resolution

Bracket Team: Acknowledged.

L-04 | Unused Modifier

Category	Severity	Location	Status
Superfluous Code	● Low	NavUpdater.sol	Resolved

Description

In the NavUpdater contract the onlyActiveVault modifier is unused.

Recommendation

Consider implementing it's use or removing it.

Resolution

Bracket Team: The issue was resolved in commit [af232ad](#).

L-05 | NavUpdater Deployment Warning

Category	Severity	Location	Status
Warning	● Low	NavUpdater.sol	Acknowledged

Description

The NavUpdater has an open initialize function which can be called by anyone and used to grant the caller the default admin role of the contract.

If the NavUpdater has already received the NAV_UPDATER_ROLE for any vault instances a malicious actor could initialize it and unexpectedly start vaults or issue nav updates for existing vaults.

Recommendation

Be sure to initialize the NavUpdater contract before it receives any permissions. Furthermore, ideally the NavUpdater contract is initialized in the same transaction as it's deployment, or at least in the same deployment script.

Resolution

Bracket Team: Acknowledged.

L-06 | Balances No Longer Include Inactive Amounts

Category	Severity	Location	Status
Warning	● Low	Global	Acknowledged

Description

In the `RebasingToken` `balanceOf` function, the `inactiveBalanceOf` contributor has been removed, which will cause a stepwise change in the balances of users and Smart Contracts which have pending deposits.

This may be unexpected and could potentially cause issues in integrating systems.

Recommendation

Be sure to warn any integrators and users of this impending stepwise change.

Resolution

Bracket Team: Acknowledged.

L-07 | totalSupply Rounding

Category	Severity	Location	Status
Rounding	● Low	Global	Acknowledged

Description

After the balances and totalSupply mismatch is resolved from M-02, Balances Do Not Match Total Supply, there remains a rounding error which invalidates the sum of user balances = totalSupply invariant.

This is due to rounding which occurs upon converting between users shares and asset amounts on transfers and deposit/withdrawals and on the calculation of user balances and totalSupply.

Extensive fuzz testing has shown that the inaccuracy can only occur in the direction of sumOfUserBalances ≤ totalSupply, and is at maximum off on the order of thousands of wei.

Recommendation

Be aware of this rounding difference and invalidation of the sum of user balances = totalSupply invariant.

Resolution

Bracket Team: Acknowledged.

I-01 | Unnecessary Import

Category	Severity	Location	Status
Superfluous Code	● Info	BracketWrappedVault.sol	Resolved

Description

In the BracketWrappedVault contract the IERC20 interface is explicitly imported but not used.

Recommendation

Consider removing the IERC20 interface import.

Resolution

Bracket Team: The issue was resolved in commit [1962de4](#).

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

Founded in 2022 by DeFi experts, Guardian is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian 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>