GUARDIAN

Bracket

KYC Whitelist

Security Assessment

May 12th, 2025

Summary

Audit Firm Guardian

Prepared By Owen Thurm, 0xCiphky, Daniel Gelfand

Client Firm Bracket

Final Report Date March 12, 2025

Audit Summary

Bracket engaged Guardian to review the security of their kyc functionality. From the 17th of April to the 18th of April, a team of 3 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Confidence Ranking
Given the lack of critical issues detected and minimal code changes following the main review, Guardian assigns a Confidence Ranking of 5 to the protocol.
Guardian advises the protocol to consider periodic review with future changes.
For detailed understanding of the Guardian Confidence Ranking, please see the rubric on the following page.

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

📊 Code coverage & PoC test suite: https://github.com/GuardianOrg/bracketfi-fuzz

Guardian Confidence Ranking

Confidence Ranking	Definition and Recommendation	Risk Profile
5: Very High Confidence	Codebase is mature, clean, and secure. No High or Critical vulnerabilities were found. Follows modern best practices with high test coverage and thoughtful design. **Recommendation:** Code is highly secure at time of audit. Low risk of latent critical issues.	0 High/Critical findings and few Low/Medium severity findings.
4: High Confidence	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. **Recommendation:** Suitable for deployment after remediations; consider periodic review with changes.	0 High/Critical findings. Varied Low/Medium severity findings.
3: Moderate Confidence	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. **Recommendation:** Address issues thoroughly and consider a targeted follow-up audit depending on code changes.	1 High finding and ≥ 3 Medium. Varied Low severity findings.
2: Low Confidence	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. **Recommendation:** Post-audit development and a second audit cycle are strongly advised.	2-4 High/Critical findings per engagement week.
1: Very Low Confidence	Code has systemic issues. Multiple High/Critical findings (≥5/week), poor security posture, and design flaws that introduce compounding risks. Safety cannot be assured. **Recommendation:** Halt deployment and seek a comprehensive re-audit after substantial refactoring.	≥5 High/Critical findings and overall systemic flaws.

Table of Contents

Project Overview

Project Summary

Project Name	Bracket
Language	Solidity
Codebase	https://github.com/bracket-fi/core-contracts
Commit(s)	Initial commit(s): 6f08bf8705217890da46a9584211f4bd21ed2df6 Final commit: Pending

Audit Summary

Delivery Date	March 12, 2025
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0
● High	1	0	0	0	0	1
● Medium	2	0	0	0	0	2
● Low	1	0	0	0	0	1
● Info	6	0	0	2	0	4

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
H-01	Incorrect stethAmount Used	Logical Error	● High	Resolved
M-01	KycWhitelist Cannot Be Deployed	Logical Error	● Medium	Resolved
M-02	Incorrect STETH Address	Logical Error	● Medium	Resolved
L-01	Missing Whitelisted From Check	Validation	● Low	Resolved
I-01	Transfer Optimization	Documentation	● Info	Resolved
I-02	Unnecessary Payable Modifier	Best Practices	● Info	Resolved
I-03	Misleading ETHToBrktETH Event Data	Events	● Info	Resolved
I-04	Missing VanityNavUpdated Event	Events	● Info	Acknowledged
I-05	Unnecessary Paused Check	Gas Optimization	● Info	Resolved
I-06	Hardcoded RocketDepositPool	Best Practices	● Info	Acknowledged

H-01 | Incorrect stethAmount Used

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

Description

In the `_lidoEthToBrktETH` function the return value from the submit function is used as the `stEth` amount gained from the submit call.

However the return value represents the amount of `stEth` shares, not the amount of `stEth` which were gained from the submit call.

As a result a significant portion of `stEth` will be left in the router contract and the user will not receive `bracketEth` for this amount.

Recommendation

Use the difference between the `stEth` balance before and after calling the submit function to wrap in the `wstEth` contract.

Resolution

Bracket Team: The issue was resolved in commit c23fed5.

M-01 | KycWhitelist Cannot Be Deployed

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

Description

The `KYCWhitelist` contract cannot be successfully deployed because the initializing functions used in the constructor use an `onlyInitializing` modifier.

Recommendation

Refactor the `KYCWhitelist` contract so that the constructor is the initializer or make a separate initializer function.

Resolution

Bracket Team: The issue was resolved in commit 7b70cb2.

M-02 | Incorrect STETH Address

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

Description

The `STETH` address in the `Config` contract is incorrect as it points to the `STETH` implementation contract (`0x17144556fd3424EDC8Fc8A4C940B2D04936d17eb`) rather than the correct proxy contract (`0xae7ab96520de3a18e5e111b5eaab095312d7fe84`).

Recommendation

Update the STETH address to be the correct proxy address.

Resolution

Bracket Team: The issue was resolved in commit ca93c8a.

L-01 | Missing Whitelisted From Check

Category	Severity	Location	Status
Validation	● Low	RebasingToken.sol: 81	Resolved

Description

In the `_update` function there is no validation that the from address is whitelisted on transfer. This allows unwhitelisted addresses to transfer to whitelisted addresses which may be unexpected.

Recommendation

Consider adding validation that the from address is also whitelisted in the `_update` function.

Resolution

Bracket Team: The issue was resolved in commit 70793e1.

I-01 | Transfer Optimization

Category	Severity	Location	Status
Documentation	● Info	RebasingToken.sol: 78	Resolved

Description

The _update function performs _clearDeposit in all cases, even when the _update invocation is for a mint or burn call.

There are no significant issues from this as the _clearDeposit call will always early return for minting and burning updates.

However to avoid any hidden introduction of issues in the future and to optimize gas expenditure, this unexpected execution should be avoided.

Recommendation

Consider only invoking the _clearDeposit function when the from address and to address are both nonzero, indicating that the update action is a legitimate transfer.

Resolution

Bracket Team: The issue was resolved in commit 6717a3d.

I-02 | Unnecessary Payable Modifier

Category	Severity	Location	Status
Best Practices	● Info	BrktEthRouter.sol: 47	Resolved

Description

The `wethToBrktETH` function includes a `payable` modifier yet does not handle `msg.value`.

Recommendation

Remove the `payable` modifier from the `wethToBrktETH` function.

Resolution

Bracket Team: The issue was resolved in commit 566e943.

I-03 | Misleading ETHToBrktETH Event Data

Category	Severity	Location	Status
Events	● Info	BrktEthRouter.sol: 42, 53	Resolved

Description

The `ETHToBrktETH` event emission in the `ethToBrktETH` and `wethToBrktETH` functions emits the minimum acceptable `brktEth` amount rather than the actual minted `brktEth` amount from the action. This may be misleading for consumers of the `ETHToBrktETH` event.

Recommendation

Consider if the actual minted `brktEth` amount should be emitted in the `ETHToBrktETH` event.

Resolution

Bracket Team: The issue was resolved in commit b0f4119.

I-04 | Missing VanityNavUpdated Event

Category	Severity	Location	Status
Events	● Info	BracketVault.sol	Acknowledged

Description

The updateNav function updates the vanityNav but does not emit a VanityNavUpdated event. This may mislead indexers and consumers of the VanityNavUpdated event if they are not aware of this behavior.

Recommendation

Consider if this behavior is intended, if not, consider emitting the VanityNavUpdated event in the updateNav function.

Resolution

Bracket Team: Acknowledged.

I-05 | Unnecessary Paused Check

Category	Severity	Location	Status
Gas Optimization	● Info	BrktEthRouter.sol: 75	Resolved

Description

In the lidoLimit function the isStakingPaused function is queried on the steth contract. If staking is paused the result of the limit is 0.

However this logic is already included in the lido underlying getCurrentStakeLimit function and therefore is unnecessary in the lidoLimit function.

Recommendation

Consider removing the unnecessary isStakingPaused logic in the lidoLimit function.

Resolution

Bracket Team: The issue was resolved in commit b677547.

I-06 | Hardcoded RocketDepositPool

Category	Severity	Location	Status
Best Practices	● Info	BrktEthRouter.sol	Acknowledged

Description

The RocketDepositPool contract is hardcoded as a constant in the BrktEthRouter contract. However the latest RocketDepositPool is allowed to change in the RocketStorage contract.

Recommendation

Be aware of this possibility and consider adding a setter function for the RocketDepositPool contract in case it were to be updated by Rocket Pool.

Resolution

Bracket 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