Base Bridge Referral Fees

Security Review

Cantina Managed review by:

Alireza Arjmand, Lead Security Researcher

Akshay Srivastav, Security Researcher

December 21, 2025

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity level	**Impact: High**	**Impact: Medium**	**Impact: Low**
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings are a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project’s overall security (Gas and Informational findings).

2 Security Review Summary

Base is a secure, low-cost, builder-friendly Ethereum L2 built to bring the next billion users onchain.

From Dec 10th to Dec 12th the Cantina team conducted a review of flywheel on commit hash 6004a0f8. The team identified a total of **4** issues:

Issues Found

Severity	**Count**	**Fixed**	**Acknowledged**
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	1	0	1
Gas Optimizations	0	0	0
Informational	3	2	1
Total	**4**	**2**	**2**

2.1 Scope

The security review had the following components in scope for flywheel on commit hash 6004a0f8:

```
src/hooks/BridgeReferralFees.sol
```

2.2 Cantina Managed Team Statement

The Cantina Managed team would like to thank the Coinbase team for their clear communication, thorough code walkthroughs, and responsiveness throughout the review process. One issue that was identified during the review was intentionally hidden. No other major issues were identified in the changes introduced.

3 Overview

The Cantina Managed team reviewed flywheel PR 155, which introduces changes to `BridgeReferralFees.sol`. The review focused on reducing the likelihood of reverts during fee and payout handling in Flywheel's bridge referral flow, while improving robustness and developer ergonomics.

3.1 Summary of Changes

This PR is primarily aimed at making `BridgeReferralFees` execution as non-reverting as possible, prioritizing graceful handling of edge cases over strict enforcement. The intent is to ensure Flywheel continues to function reliably even when external inputs or dependencies behave unexpectedly.

Several revert paths were removed or mitigated. Zero `bridgedAmount` scenarios no longer cause a revert and instead pass through silently, relying on Flywheel to filter out no-op payouts. All calls into `BuilderCodes` are now wrapped in `try/catch`, ensuring that failures in builder code processing result in fees being dropped rather than the entire transaction reverting. Additional protections were added to prevent arithmetic edge cases by constraining fee basis points through type-level enforcement and by adjusting the fee calculation logic to safely handle potential overflow conditions.

Finally, the PR improves developer ergonomics by replacing the `bytes32` builder code representation with a simpler `string` based interface, reducing configuration footguns and making integrations easier to reason about.

4 Findings

4.1 Low Risk

4.1.1 Non-Deterministic Bridged and Fee Amounts

Severity: Low Risk

Context: BridgeReferralFees.sol#L72-L112

Description: The fee and payout amounts prepared in `_onSend` are derived from the campaign's onchain balance at execution time, rather than from values that can be deterministically known or enforced by the sender on the source side of the bridge.

Specifically, `bridgedAmount` is calculated as the current balance of the campaign minus previously allocated fees, and `feeAmount` is derived as a percentage of that value. This logic is implemented in `BridgeReferralFees.sol`.

Because `bridgedAmount` depends on the live balance of the campaign contract, it can be externally influenced after the sender initiates the bridge but before `_onSend` executes.

Impact:

- `bridgedAmount` can be increased via unsolicited donations to the campaign contract. While it cannot be decreased, this still breaks the assumption that the payout amount is predictable or agreed upon by the sender.
- Fees may be skipped entirely if the builder code processing fails, resulting in a zero fee scenario.

As a result, the receiving contract on the destination chain may be forced to accept a larger payout than expected. If the target contract enforces strict amount checks or assumes exact values, the send transaction may revert, leading to unexpected failures or reduced composability.

This behavior is not necessarily a security vulnerability but can be problematic for integrations that rely on deterministic transfer amounts.

Recommendation: Consider documenting this behavior clearly or constraining payout assumptions in downstream integrations. If stronger guarantees are desired, alternative designs could derive payout amounts from sender provided data rather than live balances.

This issue does not necessarily need to be resolved but should be considered in use cases where exact payout amounts are required.

Coinbase: Acknowledged.

Cantina Managed: Acknowledged.

4.2 Informational

4.2.1 Failing test cases

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: On running the test suite with latest foundry version (`forge v1.5.0-stable`), multiple fuzz test cases appear to be failing.

For instance, the `test_success_emptyBuilderCode_zeroFees` in `test/unit/hooks/BridgeReferralFees/onSend.t.sol` is failing when the `user == address(bridgeReferralFeesCampaign)`.

To resolve that test case, this change was needed:

```
  function test_success_emptyBuilderCode_zeroFees(uint256 bridgedAmount, address user)
   ↪  public {
      bridgedAmount = bound(bridgedAmount, 1, type(uint128).max);
      vm.assume(user != address(0));
+     vm.assume(user != address(bridgeReferralFeesCampaign));

      usdc.mint(bridgeReferralFeesCampaign, bridgedAmount);
```

```
        string memory emptyCode = "";
        bytes memory hookData = abi.encode(user, emptyCode, uint8(0));

        uint256 userBalanceBefore = usdc.balanceOf(user);

        flywheel.send(bridgeReferralFeesCampaign, address(usdc), hookData);

        assertEq(usdc.balanceOf(user), userBalanceBefore + bridgedAmount, "User should
        ↪  receive full amount");
        assertEq(usdc.balanceOf(bridgeReferralFeesCampaign), 0, "Campaign should be
        ↪  empty");
    }
```

Recommendation: Consider running the test suite with latest foundry version and resolve any failing test cases.

Coinbase: Fixed in PR 155.

Cantina Managed: Fix verified.

4.2.2 Prioritize using `abi.decode` for decoding `bytes` data for consistency and readability

Severity: Informational

Context: BridgeReferralFees.sol#L124, BridgeReferralFees.sol#L166

Description: The `_onDistributeFees` and `_onUpdateMetadata` functions perform explicit typecast on `bytes hookData` to convert them to `string`. While other hook functions of `BridgeReferralFees` use `abi.decode` for parsing input `hookData`. This inconsistency in coding reduces the code readability for users and external reviewers.

Recommendation: Consider using `abi.decode` in all functions for decoding `bytes` data to enhance the consistency and readability of codebase.

Coinbase: Acknowledged.

Cantina Managed: Acknowledged.

4.2.3 Unused and redundant code

Severity: Informational

Context: BridgeReferralFees.sol#L19, BridgeReferralFees.sol#L36-L37

Description: The `BridgeReferralFees` hook contains minor implementation issues that affect code clarity and maintainability.

- BridgeReferralFees.sol#L37: The `ZeroBridgedAmount` custom error is declared but never used within the contract.
- BridgeReferralFees.sol#L19: The `NATIVE_TOKEN` address is redundantly defined instead of reusing `Constants.NATIVE_TOKEN`.

Recommendation: Remove unused error declarations and replace redundant constants with shared definitions such as `Constants.NATIVE_TOKEN` to improve maintainability and readability.

Coinbase: Fixed in commit a5056e3c.

Cantina Managed: Verified fix in commit a5056e3c.