CANTINA

Base Flywheel and Builder Codes

Security Review

Cantina Managed review by:

Optimum, Lead Security Researcher

Akshay Srivastav, Security Researcher

RustyRabbit, Security Researcher

January 7, 2026

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 Sep 9th to Sep 15th the Cantina team conducted a review of flywheel on commit hash 676c37c1. The team identified a total of **15** issues:

Issues Found

Severity	**Count**	**Fixed**	**Acknowledged**
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	6	5	1
Gas Optimizations	2	0	2
Informational	7	4	3
Total	**15**	**9**	**6**

3 Findings

3.1 Low Risk

3.1.1 `InvalidCode` error uses empty value for `toCode()` function

Severity: Low Risk

Context: BuilderCodes.sol#L292-L296

Description: When converting a `tokenId` to a code the `normalizeSmallString` returns a value that differs from the direct cast to `bytes32` (most likely upper/lower case differences) the `InvalideCode(code)` error is returned. However at this point the `code` is empty. It is also confusing that the error indicates the output (`code`) is incorrect rather than the more typical indication that the input (`tokenId`) is incorrect.

Recommendation: Either create an additional Error indicating an invalid `tokenId` and use that or ensure the `code` is the direct cast of the tokenId or the value returned by `fromSmallString`.

Coinbase: Will switch lines 293/294. See PR 118.

Cantina Managed: Fixed in PR 118.

3.1.2 Buyers of `BuilderCodes` tokens might lose payouts due to missing updating payout address

Severity: Low Risk

Context: BuilderCodes.sol#L178

Description: The `updatePayoutAddress()` function allows the owner of a `BuilderCodes` token to specify the address that will receive payouts. Since these tokens are transferable, ownership can change. However, when a token is transferred, the payout address does not automatically update. This means the new owner must explicitly set a new payout address; otherwise, payouts may continue to be sent to the previous owner.

Recommendation: Currently, the responsibility of updating the payout address lies entirely with the user. In practice, many users are likely to overlook this step, creating a risk of funds being misdirected. While there is no perfect solution that covers all edge cases, two possible approaches could improve safety:

1. Automatic Update on Transfer. Override `ERC721._update()` to automatically call `_updatePayoutAddress()` with the new owner address.
    - Pro: Ensures the payout address always matches the current owner.
    - Con: If the new owner is a contract that cannot receive payouts, this may lead to failures or lost funds.
2. Maintain Current Design but Improve Awareness. Leave the logic unchanged but clearly document and communicate the limitation to users and potential buyers.
    - Pro: Avoids compatibility issues with contracts or specialized payout setups.
    - Con: Relies on user diligence, increasing the chance of errors.

Coinbase: Acknowledged. Will document.

Cantina Managed: Acknowledged.

3.1.3 `createCampaign()` is front-runnable which might cause confusion for callers and potentially lost funds

Severity: Low Risk

Context: Flywheel.sol#L241

Description: The `createCampaign()` function uses the `CREATE2` opcode to deploy new instances of the `Campaign` contract. However, the salt used for contract creation does not include `msg.sender`, which makes the function front-runnable.

An attacker could frontrun a legitimate call to `createCampaign()` by submitting an identical transaction with the same salt. While the attacker does not gain direct benefits-since the deployed contract would be the same as the intended one-the frontrunning causes the original caller's transaction to revert. This behavior can lead to several issues:

1. If `createCampaign()` is called by another contract as part of a broader operation, the entire transaction may revert unexpectedly.
2. A caller may have already pre-computed the address of the new campaign and transferred funds there. If their `createCampaign()` transaction fails, they may believe their funds are lost.
3. The caller may perceive the failed transaction as a denial-of-service (DoS) attack against their campaign creation attempts.

Recommendation: To mitigate this issue, incorporate `msg.sender` into the salt when creating the campaign. This ensures uniqueness per caller and prevents front-running collisions:

```
campaign = Clones.cloneDeterministic(.
    campaignImplementation,
    keccak256(abi.encode(hooks, nonce, hookData, msg.sender)).
);
```

Coinbase: Fixed in commit 8b884ca9.

Cantina Managed: Fixed.

3.1.4 Reverting token transfers can undo payout batches

Severity: Low Risk

Context: Campaign.sol#L41-L49

Description: When payouts to multiple recipients are batched, the complete batch will revert if one of them reverts. For instance this can happen when one of the recipients is on the USDC blocklist.

Note that the `manager/attributionProvider` does not always know the recipient in advance and thus may not preemptively be able to censor those recipients to avoid the issue. The manager would then have to determine which recipient is causing the issue and resubmit the batch.

Recommendation: Consider returning an array with the status of each of the payments (and fee) rather than revering the batch. Alternatively (or additionally) use some retry mechanism caching the payment info so the user can withdraw when the issue has been resolved.

Coinbase: Fixed by the following combination of PRs:

- PR 112: Add failed fee fallback mechanism.
- PR 121: Make `BridgeRewards` accept bridged amount in `calldata`.
- PR 125: Update `BridgeRewards` logic (reverts change to using `calldata` above).

We acknowledge that payouts can block other payouts. We do not think this has the same exposure to DoS risk.

Cantina Managed: Fixed for the fee distribution.

3.1.5 Excessive fees can prevent users from receiving bridged funds immediately

Severity: Low Risk

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

Description: The `BridgeRewards` hook contract limits the maximum `feeBps` that can be applied. If this is exceeded the payout reverts which causes the user's funds to be locked in the bridge until the issue is resolved.

Recommendation: Consider capping the fee to it's maximum rather than reverting in this case so the user immediately receive their funds.

Coinbase: Fixed in PR 116.

Cantina Managed: Fixed verified.

3.1.6 Non-existent builder codes can prevent users from receiving bridged funds immediately

Severity: Low Risk

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

Description: When an incorrect builder code is used while bridging user funds the transaction reverts and the user are locked in the bridge until the bridge operator resolves the issue.

Recommendation: An incorrect builder token is the bridge's error and should not lead to user inconvenience. For better UX consider putting the inconvenience on the builder by handling it as a payout without builder code and transfering all of the funds to the user.

Coinbase: Fixed in PR 116.

Cantina Managed: Fixed verified.

3.2 Gas Optimization

3.2.1 Unnecessary call to hook contracts when updating metadata

Severity: Gas Optimization

Context: Flywheel.sol#L448-L449

Description: When updating a hook contract's metadata the updated URI is queried in a separate call in `campaignURI` This is an unnecessary call as the `onUpdateMetadata()` function could return the URI immediately.

Recommendation: Consider modifying `onUpdateMetadata()` to reurn the newly modified URI in order to save the cost of an extra call to the same hook contract.

Coinbase: Discussed and will leave it as is.

Cantina Managed: Acknowledged.

3.2.2 Unused imports

Severity: Gas Optimization

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

Description:

```
// CashbackRewards.sol
import {CampaignHooks} from "../CampaignHooks.sol";
```

```
// Flywheel.sol
import {SafeERC20, IERC20} from "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";
```

`CampaignHooks` is not used in `CashbackRewards` and `SafeERC20` is not used in `Flywheel`.

Recommendation: Consider removing these unused imports to reduce the contract size.

Coinbase: Acknowledged.

Cantina Managed: Acknowledged.

3.3 Informational

3.3.1 `owner_` is not used in the `Adconversion constructor`

Severity: Informational

Context: AdConversion.sol#L190

Description: The constructor of `AdConversion` takes an `owner_` as parameter but it is never used and the contract isn't `Ownable` as there is no owner-only functionality provided by the contract.

Recommendation: Remove the `owner_` from the constructor definition.

Coinbase: Fixed in PR 115.

Cantina Managed: Fixed.

3.3.2 `_onCreateCampaign` does not check existence before overwriting

Severity: Informational

Context: CashbackRewards.sol#L91-L99

Description: When `Flywheel`creates a new campaign and calls `onCreateCampaign()` on the `CashbackRewards`, `SimpleRewards` and `AddConversion` hook contracts they do not check if the campaign is already registered, in which case they should revert instead of silently overwriting the existing campaign data as they currently do.

Although the `Flywheel` contract always generates a new campaign address `CashbackRewards`, `SimpleRewards` and `AddConversion` should not rely on how they are used to perform correctly.

Also note that due to versioning of `Flywheel` the need may arise for hook contracts to support multiple instances of (different versions of) `Flywheel`. Therefor it would make sense to add the `flywheel` address as a parameter of the internal hook functions like `_onCreateCampaign()`.

Recommendation: Consider checking whether the indicated campaign address is already regstered and revert in case it is.

Additionally consider modifying the internal interface to include the calling `flywheel` contract to allow the implementation to separate campaign storage per instance of `flywheel`. Although `msg.sender` can always be used for this it would make for a cleaner integration as 3rd party implementers are encouraged to inherit from `CampaignHooks`.

Coinbase: Acknowledged.

Cantina Managed: Acknowledged.

3.3.3 Edge case for referral code mismatch in case of reorgs

Severity: Informational

Context: PseudoRandomRegistrar.sol#L57-L59

Description: The referral codes generated for consecutive registrations could be swapped between two users when a block is reorged and the order of their transactions is swapped.

The code generated is based on the following parameters:

- `nonce` (sequence number tracked by the register function).
- Block timestamp.
- Block hash of the previous block.
- `prevrandao`.

In case of a reorg `prevrandao` can be the same for the original block and the reorged block. On some chains like Arbitrum `prevrandao` is always a fixed value. The block timestamp of both blocks has a high probability of being equal and the block hash of the previous block will be the same for the first reorged block.

Therefor if the order of two register transaction are swapped users could end up with swapped codes after the reorg versus what they received in their original confirmation.

Recommendation: Although this is en extreme edge case you might consider to further reduce the likelihood of this happening by including `msg.sender`, `tx.origin`. As these still do not offer any guarantee (for third party contracts and account abstraction) adding a (non-sequential) randomness parameter to the `generate()` function can additionally be considered for these situations.

Coinbase: Acknowledged. We're okay with this risk.

Cantina Managed: Acknowledged.

3.3.4 `BuilderCodes.updateMetadata`: Missing `tokenId` existence check

Severity: Informational

Context: BuilderCodes.sol#L160-L162

Description: The `BuilderCodes.updateMetadata` function is missing `_requireOwned(tokenId)` check due to which a `MetadataUpdate` event can be emitted for a non-existent `tokenId` or a `tokenId` which will be minted in future.

Recommendation:

```diff
  function updateMetadata(uint256 tokenId) external onlyRole(METADATA_ROLE) {
+     _requireOwned(tokenId);
      emit MetadataUpdate(tokenId);
  }
```

Coinbase: Fixed in PR 118.

Cantina Managed: Fix verified.

3.3.5 Out-of-order alphanumeric characters

Severity: Informational

Context: BuilderCodes.sol#L47, PseudoRandomRegistrar.sol#L16

Description: These constant variables have misplaced `n` and `o` characters:

1. `ALLOWED_CHARACTERS` in `BuilderCodes`.
2. `ALPHANUMERIC` in `PseudoRandomRegistrar`.

Recommendation: Switch the places of `n` and `o` characters.

```diff
- 0123456789abcdefghijklmonpqrstuvwxyz
+ 0123456789abcdefghijklmnopqrstuvwxyz
```

Coinbase: Fixed in PR 115.

Cantina Managed: Fix verified.

3.3.6 Potential out-of-gas issue in future hooks implementations

Severity: Informational

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

Description: The `Flywheel.createCampaign()` function allows callers to specify an arbitrary `hooks` contract. This introduces a potential risk: if the specified hook contract writes to its own storage inside `_onCreateCampaign()`, the stored data may accumulate over time.

If other functions in the system later iterate over this storage, the increasing data size could cause those calls to run out of gas and revert, effectively leading to a denial of service.

Recommendation: Document this potential pitfall and clearly warn implementers of hook contracts to avoid unbounded storage writes in `_onCreateCampaign()`.

Coinbase: Acknowledged.

Cantina Managed: Acknowledged.

3.3.7 Missing `foundry.lock` file

Severity: Informational

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

Description: This repository does not include a `foundry.lock` file, which ensures deterministic builds by locking dependency versions. Without it, contributors may build with different dependency versions, risking inconsistent behavior or security issues.

Impact: Recent `npm` supply-chain attacks showed how compromised packages can introduce wallet-hijacking malware. Unlocked dependencies can expose projects to similar risks.

Recommendation: Commit `foundry.lock` to version control and pin dependencies to ensure reproducible and secure builds.

Coinbase: Fixed in PR 115.

Cantina Managed: Fixed by implementing the reviewer's recommendation.