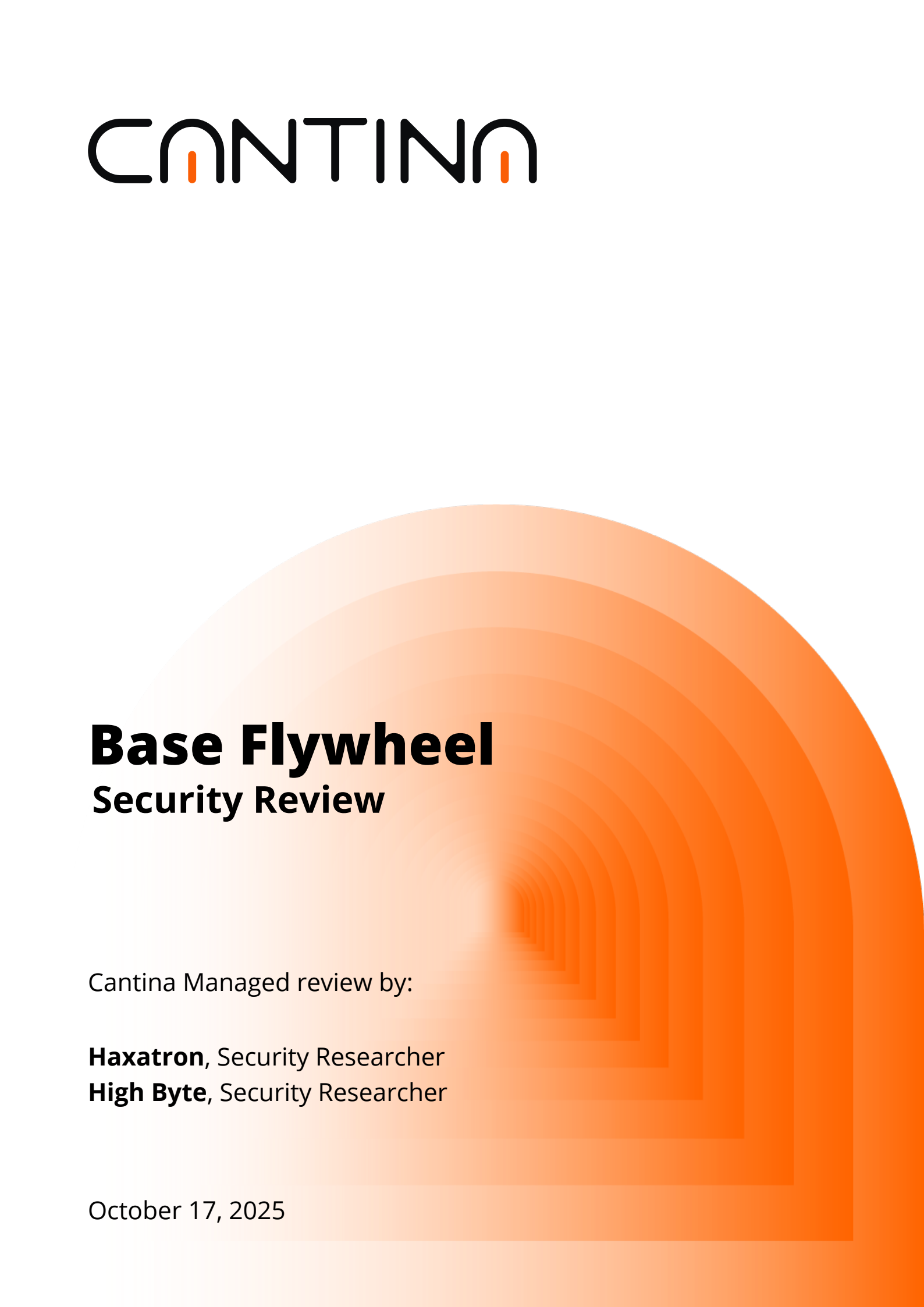


Base Flywheel
Security Review

Cantina Managed review by:

Haxatron, Security Researcher
High Byte, Security Researcher

October 17, 2025

Contents

1 Introduction **2**
- 1.1 About Cantina . . . 2
- 1.2 Disclaimer . . . 2
- 1.3 Risk assessment . . . 2
  - 1.3.1 Severity Classification . . . 2

2 Security Review Summary **3**

3 Findings **4**
- 3.1 High Risk . . . 4
  - 3.1.1 Advertiser can bypass the attribution window by changing `ACTIVE` -> `FINALIZED` . . . 4
- 3.2 Medium Risk . . . 6
  - 3.2.1 Lack of minimum attribution window allows advertiser to immediately close the campaign . . . 6
  - 3.2.2 Advertiser can frontrun `onReward` by disabling a conversion config . . . 7
  - 3.2.3 Changes to the attribution provider fee immediately take effect . . . 7
- 3.3 Low Risk . . . 8
  - 3.3.1 `BuilderCodes` EIP712 domain name and version cannot change . . . 8
  - 3.3.2 No deadline check for `BuilderCodes` registration signatures . . . 9
  - 3.3.3 Allocated funds can be accidentally locked if campaign is set to `FINALIZED` . . . 10
  - 3.3.4 Rebase tokens can cause funds to be locked on a negative rebase . . . 10
  - 3.3.5 Skip failed payments instead of reverting . . . 11
  - 3.3.6 `attributionProvider` can block advertiser from updating deadline . . . 11
- 3.4 Gas Optimization . . . 11
  - 3.4.1 Consider converting the triple mapping to single mapping . . . 11
- 3.5 Informational . . . 12
  - 3.5.1 Consider making the 32-byte maximum builder code length explicit . . . 12
  - 3.5.2 Only owner can withdraw funds in `SimpleRewards` deviating from README . . . 12
  - 3.5.3 Allowed state transitions for `SimpleRewards` and `CashbackRewards` deviates from README . . . 13
  - 3.5.4 Consider adding a check that `isActive` is true before disabling a conversion config . . . 14
  - 3.5.5 Consider adding a check that allowed publisher refcode is in the registry in `AdConversion.onCreateCampaign` . . . 15
  - 3.5.6 Consider adding validation that the `maxRewardBasisPoints` cannot exceed `10_000` . . . 15
  - 3.5.7 Consider setting `maxRewardBasisPoints = 10_000` in `_validatePaymentReward` if `maxRewardBasisPoints` is `0` . . . 15
  - 3.5.8 Fund withdrawals can occur before `FINALIZED` in `SimpleRewards` and `CashbackRewards`, deviating from README . . . 16
  - 3.5.9 Consider changing default hook behavior to abstract or no-op . . . 17
  - 3.5.10 Merge duplicated code . . . 17
  - 3.5.11 Safeguard hooks by enforcing modifier . . . 17
  - 3.5.12 Add events index . . . 18

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 Aug 24th to Oct 8th the Cantina team conducted a review of flywheel on commit hash 71eda492. The team identified a total of **23** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	1	1	0
Medium Risk	3	2	1
Low Risk	6	5	1
Gas Optimizations	1	1	0
Informational	12	7	5
Total	**23**	**16**	**7**

3 Findings

3.1 High Risk

3.1.1 Advertiser can bypass the attribution window by changing `ACTIVE` -> `FINALIZED`

Severity: High Risk

Context: README.md#L778-L785, Flywheel.sol#L383-L403, AdConversion.sol#L411-L440

Description: The attribution window can be bypassed by the advertiser by updating from `INACTIVE` → `FINALIZED` or `ACTIVE` → `FINALIZED`. To understand how this can occur we first discuss the allowed state transitions for `AdConversion` in the README:

Discussion on State Transitions:

State	Who Can Transition To	Available Functions	Special Behaviors
INACTIVE	• ACTIVE: Attribution Provider only; • FINALIZING: Attribution Provider or Advertiser	None	Security: No party can pause active campaigns (ACTIVE → INACTIVE blocked)
ACTIVE	• FINALIZING: Attribution Provider or Advertiser	`reward()` only	Live campaign processing conversions
FINALIZING	• FINALIZED: Attribution Provider (any time), Advertiser (after deadline)	`reward()` only	Sets attribution deadline based on campaign's configured duration (max 180 days)
FINALIZED	None (terminal state)	None	Only Advertiser can withdraw remaining funds

Here is the implementation of the restrictions in the code:

```
/// @notice Updates the status of a campaign
///
/// @param campaign Address of the campaign
/// @param newStatus New status of the campaign
/// @param hookData Data for the campaign hook
function updateStatus(address campaign, CampaignStatus newStatus, bytes calldata hookData)
    external
    nonReentrant
    onlyExists(campaign)
{
    CampaignStatus oldStatus = _campaigns[campaign].status;
    if (
        newStatus == oldStatus // must update status
            || oldStatus == CampaignStatus.FINALIZED // cannot update from finalized
            || (oldStatus == CampaignStatus.FINALIZING && newStatus != CampaignStatus.FINALIZED) // finalizing
            ↪  can only update to finalized
    ) revert InvalidCampaignStatus();

    _campaigns[campaign].hooks.onUpdateStatus(msg.sender, campaign, oldStatus, newStatus, hookData);
    _campaigns[campaign].status = newStatus;
    emit CampaignStatusUpdated(campaign, msg.sender, oldStatus, newStatus);
}
```

```
/// @inheritdoc CampaignHooks
function onUpdateStatus(
    address sender,
    address campaign,
    Flywheel.CampaignStatus oldStatus,
    Flywheel.CampaignStatus newStatus,
    bytes calldata hookData
) external override onlyFlywheel {
    // Prevent ACTIVE → INACTIVE transitions for ALL parties (no one can pause active campaigns)
    if (oldStatus == Flywheel.CampaignStatus.ACTIVE && newStatus == Flywheel.CampaignStatus.INACTIVE) {
        revert Unauthorized();
    }

    // Attribution provider can perform other valid state transitions
    if (sender == state[campaign].attributionProvider) return;
```

```
    // Otherwise only advertiser allowed to update status
    if (sender != state[campaign].advertiser) revert Unauthorized();

    // Advertiser always allowed to start finalization delay
    if (newStatus == Flywheel.CampaignStatus.FINALIZING) {
        state[campaign].attributionDeadline = uint48(block.timestamp) + state[campaign].attributionWindow;
        emit AttributionDeadlineUpdated(campaign, state[campaign].attributionDeadline);
        return;
    }

    // Advertiser only allowed to finalize, but only if delay has passed
    if (newStatus != Flywheel.CampaignStatus.FINALIZED) revert Unauthorized();
    if (state[campaign].attributionDeadline > block.timestamp) revert Unauthorized();
}
```

From the code we see the following restrictions:

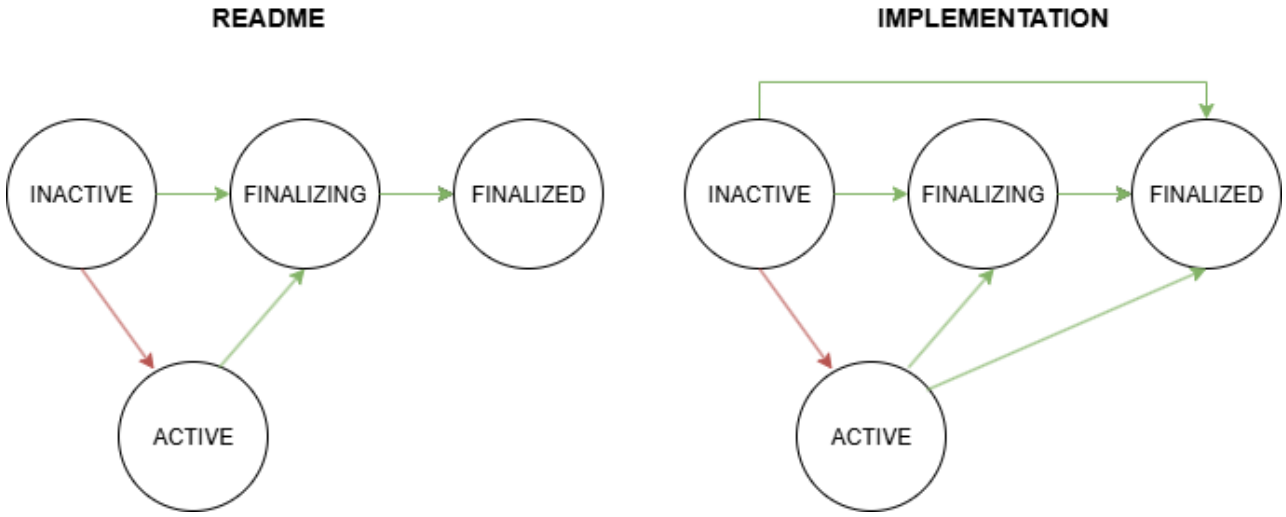
For both AP and Advertiser:

- `newStatus == oldStatus`: The status must be updated.
- `oldStatus == CampaignStatus.FINALIZED`: `FINALIZED` cannot be updated from.
- `(oldStatus == CampaignStatus.FINALIZING && newStatus != CampaignStatus.FINALIZED)`: If `FINALIZING`, must update to `FINALIZED`.
- `oldStatus == Flywheel.CampaignStatus.ACTIVE && newStatus == Flywheel.CampaignStatus.INACTIVE`: `ACTIVE` → `INACTIVE` is disallowed.

For advertiser only:

- `newStatus == Flywheel.CampaignStatus.FINALIZING`, if transition to `FINALIZING` set the attribution deadline.
- `newStatus != Flywheel.CampaignStatus.FINALIZED`, if transition is not to `FINALIZING`, then it must be to `FINALIZED` and the attribution deadline must pass: `state[campaign].attributionDeadline > block.timestamp` evaluates to `false`.

We can observe the following state transition graph to see how the implementation differs from the README. Red edges represent that only the AP can perform the transition while green edges represent both the AP and advertiser can perform the transition.



If we notice the above, the restriction does not account for the fact that the advertiser can transition from `INACTIVE` → `FINALIZED` or `ACTIVE` → `FINALIZED`. Hence, a malicious advertiser can transition from `INACTIVE` → `FINALIZED` or `ACTIVE` → `FINALIZED` as seen above.

When this occurs, the `attributionDeadline` remains 0 in the state. Then, the condition `state[campaign].attributionDeadline > block.timestamp` will evaluate to `false` and the revert will not occur, this will allow a malicious advertiser to immediately close a campaign and retrieve all the funds without allowing for the attribution provider to fully attribute all the recipients, allowing them to get free advertisement work without paying for it.

Recommendation: The fix for this issue is similar the fix for the other issue by implementing the changes in the common `updateStatus` function in the `Flywheel` contract with the addition of one more change that should only be present in the `AdConversion` hook. More precisely, the following minimal changes can be used to correct the state transition graph:

For the `Flywheel` contract:

- `(oldStatus == CampaignStatus.FINALIZING && newStatus != CampaignStatus.FINALIZED)` should be changed to `(oldStatus != CampaignStatus.FINALIZING && newStatus == CampaignStatus.FINALIZED)`, this means that if the new status is `FINALIZED`, then the old status MUST BE `FINALIZING`. This deletes the `INACTIVE` → `FINALIZED` and `ACTIVE` → `FINALIZED` edges and restores `FINALIZING` → `INACTIVE` and `FINALIZING` → `ACTIVE` edges.
- Then to delete `FINALIZING` → `INACTIVE` edge, add `(oldStatus == CampaignStatus.FINALIZING && newStatus == CampaignStatus.INACTIVE)`.

For the `AdConversion` contract:

- To delete `FINALIZING` → `ACTIVE` edge (note that after the changes above, this will only be allowed for AP), add `(oldStatus == CampaignStatus.FINALIZING && newStatus == CampaignStatus.ACTIVE)`.

Coinbase: Fixed in PR 90.

Cantina Managed: Fix verified.

3.2 Medium Risk

3.2.1 Lack of minimum attribution window allows advertiser to immediately close the campaign

Severity: Medium Risk

Context: AdConversion.sol#L216-L231

Description: When creating a `AdConversion` campaign, the `campaignAttributionWindow` determines how long the campaign must be in `FINALIZING` state before an advertiser can close the campaign by transitioning it to `FINALIZED`. The problem is that there is no minimum limit on what this value can be, the code only checks that it is a multiple of `days` and it cannot be more than `180 days`.

```
/// @inheritdoc CampaignHooks
function onCreateCampaign(address campaign, bytes calldata hookData) external override onlyFlywheel {
    (
        address attributionProvider,
        address advertiser,
        string memory uri,
        string[] memory allowedPublisherRefCodes,
        ConversionConfigInput[] memory configs,
        uint48 campaignAttributionWindow
    ) = abi.decode(hookData, (address, address, string, string[], ConversionConfigInput[], uint48));

    // Validate attribution deadline duration (if non-zero, must be in days precision)
    if (campaignAttributionWindow % 1 days != 0) revert InvalidAttributionWindow(campaignAttributionWindow);

    // Validate attribution window is between 0 and 6 months (180 days)
    if (campaignAttributionWindow > 180 days) revert InvalidAttributionWindow(campaignAttributionWindow);
```

Thus, an advertiser could set the `campaignAttributionWindow = 0` which bypasses all checks. With `campaignAttributionWindow = 0`, an advertiser can instantly transition a campaign from `ACTIVE` → `FINALIZING` → `FINALIZED` and retrieve all the funds without allowing for the attribution provider to fully attribute all the recipients, allowing them to get free advertisement work without paying for it.

This issue is Medium severity and not High severity as the attribution window being 0 would probably be a red flag to potential users and attribution providers interacting with the malicious advertiser.

Recommendation: Although this seems intentional judging from the comment `(if non-zero, must be in days precision)`, we recommend adding a minimum check for the `campaignAttributionWindow` parameter to prevent the above scenario.

Coinbase: This was actually intentional. Attribution providers will be able to see the configured window and choose whether or not to set campaign to ACTIVE. There are some cases where having no delay is helpful for operations and trust can fill in the gap.

Coinbase: Acknowledged. This was actually intentional. Attribution providers will be able to see the configured window and choose whether or not to set campaign to ACTIVE. There are some cases where having no delay is helpful for operations and trust can fill in the gap.

Cantina Managed: Acknowledged.

3.2.2 Advertiser can frontrun `onReward` by disabling a conversion config

Severity: Medium Risk

Context: AdConversion.sol#L315-L326

Description: An advertiser can frontrun an `onReward` call and cause it to fail by disabling a conversion config via `disableConversionConfig`, this will cause `isActive` to be false and a revert to occur in `onReward`:

```
/// @inheritdoc CampaignHooks
function onReward(address attributionProvider, address campaign, address payoutToken, bytes calldata hookData)
    external
    override
    onlyFlywheel
    returns (Flywheel.Payout[] memory payouts, Flywheel.Allocation[] memory fees)
{
        // ...
        if (configId != 0) {
            if (configId > conversionConfigCount[campaign]) {
                revert InvalidConversionConfigId();
            }

            ConversionConfig memory config = conversionConfigs[campaign][configId];
            if (!config.isActive) revert ConversionConfigDisabled();
            // ...
        }
```

This can be used by a malicious advertiser to deny payouts to recipients and cause honest attribution provider calls to constantly fail and therefore waste gas. Even in the case of honest advertisers disabling conversion config, since the attribution provider call may be delayed, the call might fail without the honest advertiser intending it to fail. A workaround does exist where an attribution provider can set `configId = 0` to bypass any conversion config checks although this might cause the data to be incorrect.

Recommendation: Remove the `isActive` check in `onReward` and have the offchain attribution provider check it instead. Note that this was the same approach taken by the previous iteration of the Flywheel contracts.

Coinbase: Fixed in PR 100. Replaced with existence check instead.

Cantina Managed: Fix verified.

3.2.3 Changes to the attribution provider fee immediately take effect

Severity: Medium Risk

Context: AdConversion.sol#L282-L285

Description: In the `AdConversion` contract, advertisers can choose their own attribution providers who will perform attributions for them. The assumption is that the advertiser generally trusts the attribution provider to submit correct attribution data and payout amounts. One of the factors advertisers may take into account when choosing the provider is the attribution provider fee, which is the fee taken from every payout that goes to the provider for submitting attributions.

However, even an honest attribution provider might update their fee via `setAttributionProviderFee`, this updates the `attributionProviderFees` mapping:

```
/// @notice Sets the fee for an attribution provider
///
/// @param feeBps The fee in basis points (0 to 10000, where 10000 = 100%)
///
/// @dev Only the attribution provider themselves can set their fee
function setAttributionProviderFee(uint16 feeBps) external {
    if (feeBps > MAX_BPS) revert InvalidFeeBps(feeBps);

    uint16 oldFeeBps = attributionProviderFees[msg.sender];
```

```
    attributionProviderFees[msg.sender] = feeBps;

    emit AttributionProviderFeeUpdated(msg.sender, oldFeeBps, feeBps);
}
```

During `onReward`, the attribution provider fee % is read from this same `attributionProviderFees` mapping:

```
/// @inheritdoc CampaignHooks
function onReward(address attributionProvider, address campaign, address payoutToken, bytes calldata hookData)
    external
    override
    onlyFlywheel
    returns (Flywheel.Payout[] memory payouts, Flywheel.Allocation[] memory fees)
{
    // ...

    // Get the fee from the stored attribution provider fees
    uint16 feeBps = attributionProviderFees[attributionProvider];
    // ...

        // Deduct attribution fee from payout amount
        uint256 attributionFee = (attributions[i].conversion.payoutAmount * feeBps) / MAX_BPS;
        feeAmount += attributionFee;
        uint256 netAmount = attributions[i].conversion.payoutAmount - attributionFee;
    // ...
}
```

As such any updates to the `attributionProviderFees` takes effect immediately, even if in mid-campaign. This may not be ideal for advertisers or users.

Recommendation: Consider caching the attribution provider fee at the time of campaign creation in `onCreateCampaign` and use that cached fee throughout the campaign.

Note that if such a change were to be implemented an advertiser can cache zero fees for an attribution provider who isn't ready for attribution and hence has not yet set their fee, so you might also want to check that the attribution provider is ready via some `bool` stored in a mapping (ie. `providerIsReady`).

Coinbase: We are fixing by having the advertiser passing in attribution provider free Bps so that AP and Advertiser can come to an agreement on fees. If AP doesn't like the fee, they can not make campaign ACTIVE. This allows AP and different advertisers to come to different agreements easily.

Coinbase: Fixed in PR 91.

Cantina Managed: Fix verified.

3.3 Low Risk

3.3.1 `BuilderCodes` EIP712 domain name and version cannot change

Severity: Low Risk

Context: BuilderCodes.sol#L324-L331

Description: The `BuilderCodes` uses Solady's `EIP712` to implement EIP-712-compliant signatures. Since it is an upgradeable contract, the domain name and version returned by the `_domainNameAndVersion` may subject to change in the future:

```
/// @notice Returns the domain name and version for the referral codes
///
/// @return name The domain name for the referral codes
/// @return version The version of the referral codes
function _domainNameAndVersion() internal pure override returns (string memory name, string memory version) {
    name = "Builder Codes";
    version = "1";
}
```

The problem is that since the `BuilderCodes` contract does not override `_domainNameAndVersionMayChange` function to return `true`, Solady will instead use the cached domain name and version when calculating the domain separator:

```
function _buildDomainSeparator() private view returns (bytes32 separator) {
    // We will use `separator` to store the name hash to save a bit of gas.
```

```
    bytes32 versionHash;
    if (_domainNameAndVersionMayChange()) {
        (string memory name, string memory version) = _domainNameAndVersion();
        separator = keccak256(bytes(name));
        versionHash = keccak256(bytes(version));
    } else {
        separator = _cachedNameHash;
        versionHash = _cachedVersionHash;
    }
    // ...
```

Hence, updating the domain name and version on a contract upgrade to invalidate previous signatures would not be possible.

Recommendation: Override `_domainNameAndVersionMayChange` to return `true` if you foresee that the domain name and version will change, as suggested by the Solady library.

```
/// @dev Returns if `_domainNameAndVersion()` may change
/// after the contract has been deployed (i.e. after the constructor).
/// Default: false.
function _domainNameAndVersionMayChange() internal pure virtual returns (bool result) {}
```

Coinbase: Fixed in PR 93.

Cantina Managed: Fix verified.

3.3.2 No deadline check for `BuilderCodes` registration signatures

Severity: Low Risk

Context: BuilderCodes.sol#L122-L147

Description: The `BuilderCodes` contract allows users to register custom builder codes using the signature of someone with `REGISTER_ROLE`.

```
/// @notice Registers a new referral code in the system with a signature
///
/// @param code Custom builder code for the builder code
/// @param initialOwner Owner of the builder code
/// @param payoutAddress Default payout address for all chains
/// @param registrar Address of the registrar
/// @param signature Signature of the registrar
function registerWithSignature(
    string memory code,
    address initialOwner,
    address payoutAddress,
    address registrar,
    bytes memory signature
) external {
    // Check registrar has role
    _checkRole(REGISTER_ROLE, registrar);

    // Check signature is valid
    bytes32 structHash =
        keccak256(abi.encode(REGISTRATION_TYPEHASH, keccak256(bytes(code)), initialOwner, payoutAddress));
    if (!SignatureCheckerLib.isValidSignatureNow(registrar, _hashTypedData(structHash), signature)) {
        revert Unauthorized();
    }

    _register(code, initialOwner, payoutAddress);
}
```

Since there are no deadline checks, an already issued signature for a specific builder code in the contract will never expire and will be indefinitely valid. The absence of a deadline check does not represent a direct security risk but is something you might want to consider adding.

Recommendation: Consider adding a deadline check, if necessary.

Coinbase: Fixed in PR 93.

Cantina Managed: Fix verified.

3.3.3 Allocated funds can be accidentally locked if campaign is set to `FINALIZED`

Severity: Low Risk

Context: Flywheel.sol#L257-L282, Flywheel.sol#L309-L323, Flywheel.sol#L366-L381

Description: Funds passing through Flywheel can flow through main two pathways, a one-step transfer via `reward()` or a two-step transfer via `allocate()` then `distribute()`. The problem is that two-step transfers, both `allocate()` and `distribute()` calls can only occur during the `ACTIVE` or `FINALIZING` states in Flywheel. Once set to the `FINALIZED`, none of these calls can be made. This presents a problem where funds that have been allocated via `allocate()` but were not distributed via `distribute()` will be locked in the contract when the status is transitioned to `FINALIZED`. The funds are permanently locked because `allocate()` will increase the `totalReserved` by the allocated amount.

```
function allocate(address campaign, address token, bytes calldata hookData)
    external
    nonReentrant
    onlyExists(campaign)
    acceptingPayouts(campaign)
    returns (Allocation[] memory allocations)
{
    // ...
    _assertTotalReservedSolvency(campaign, token, totalReserved[campaign][token] + totalAmount);
}
```

Then not even `withdrawFunds` can be called to withdraw the allocated amount, as the `_assertTotalReservedSolvency` function prevents the total funds in the campaign from going below the `totalReserved` which includes the allocated but not distributed funds.

```
function withdrawFunds(address campaign, address token, bytes calldata hookData)
    external
    nonReentrant
    onlyExists(campaign)
{
    Payout memory payout = _campaigns[campaign].hooks.onWithdrawFunds(msg.sender, campaign, token, hookData);
    (address recipient, uint256 amount) = (payout.recipient, payout.amount);
    Campaign(campaign).sendTokens(token, recipient, amount);
    emit FundsWithdrawn(campaign, token, recipient, amount, payout.extraData);
    _assertTotalReservedSolvency(campaign, token, totalReserved[campaign][token]);
}
```

Recommendation: If you consider this a problem to be fixed, a possible solution would be to allow `distribute()` calls even in `FINALIZED` state to allow allocated funds to be distributed even after a campaign has been `FINALIZED`.

Coinbase: Fixed in PR 107. On finalization, instead of forcing entities to `deallocate` every undistributed allocation they previously made, the better devx is to just auto-deallocate everything implicitly by not counting allocated payouts in the solvency check for `withdrawFunds`.

Cantina Managed: Fix verified.

3.3.4 Rebase tokens can cause funds to be locked on a negative rebase

Severity: Low Risk

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

Description: Rebase tokens are tokens that can have their balances changed outside of a transfer, one notable example is stETH (Lido staked ETH). In the `Flywheel` contracts, any token can be used and decided upon by the owner of the campaign. On a negative rebase of any such token, users balances can be decreased. This can cause the token balance stored in the `Campaign` contract to decrease below the reserved amounts, making it impossible to retrieve the full reserved amount from the `Campaign` contract. This might cause tokens to be stuck as the `_assertTotalReservedSolvency` invariant will always fail and cause a revert.

A workaround would be to batch deallocate tokens until the reserved amounts fall below the actual balance after the negative base. Another way would be to then top up the `Campaign` contract so that its balance meets the reserved amount. Both of these workarounds would require a lot of hassle which is not ideal.

Recommendation: If you believe this is a problem that should be fixed, one solution would be to have a token blocklist / allowlist preventing the use of these tokens. Another solution would be to warn users of the potential consequences of using such tokens in the documentation.

Coinbase: Addressed in documentation update in PR 95.

Cantina Managed: Acknowledged.

3.3.5 Skip failed payments instead of reverting

Severity: Low Risk

Context: CashbackRewards.sol#L107

Description/Recommendation: `_validatePaymentReward` can revert, but perhaps it should return boolean instead so you could process any other valid rewards. Any call would include lots of calldata, as well as execution ops, so small failure shouldn't require re-executing.

Coinbase: Fixed in PR 98.

Cantina Managed: Fix verified.

3.3.6 `attributionProvider` can block advertiser from updating deadline

Severity: Low Risk

Context: AdConversion.sol#L424-L435

Description/Recommendation: If called by `attributionProvider` then we won't reach the deadline update part due to early return, but it will succeed in setting `FINALIZING` status. Since you can also transition `FINALIZING` → `FINALIZED` it means the `attributionProvider` can dos `advertiser` from changing the deadline.

- `Flywheel.sol`:

```
    || (oldStatus == CampaignStatus.FINALIZING && newStatus != CampaignStatus.FINALIZED) // finalizing can
    ↪  only update to finalized
```

Coinbase: Will fix by setting attribution deadline even if attribution provider submits status transition. Fixed in PR 90.

Cantina Managed: Fix verified. Now deadline is udpated for both on `ACTIVE` → `FINALIZING`.

3.4 Gas Optimization

3.4.1 Consider converting the triple mapping to single mapping

Severity: Gas Optimization

Context: Flywheel.sol#L277

Description/Recommendation: The mapping `pendingPayouts[campaign][token]` could be cached as follows:

```
mapping(bytes32 key => uint256 amount) storage _pendingPayouts = pendingPayouts[campaign][token];

for (/*...*/) {
    // ...
    _pendingPayouts[key] += amount;
    // ...
}
```

A alternative approach is making `pendingPayouts` mapping from a single key which is `keccak256(abi.encode(key1, key2, key3))`.

Coinbase: Fixed in PR 96.

Cantina Managed: Fix verified.

3.5 Informational

3.5.1 Consider making the 32-byte maximum builder code length explicit

Severity: Informational

Context: BuilderCodes.sol#L268-L276

Description: When registering a custom builder code, the `_register` will call `toTokenId` to convert the builder code `string` to a `uint256` to determine the ERC721 token ID to mint.

```
/// @notice Converts a referral code to a token ID
///
/// @param code Builder code to convert
///
/// @return tokenId The token ID for the referral code
function toTokenId(string memory code) public pure returns (uint256 tokenId) {
    if (!isValidCode(code)) revert InvalidCode(code);
    return uint256(LibString.toSmallString(code));
}
```

It does this via Solady `LibString.toSmallString` which reverts if the length of the `string` is more than 32 bytes. Hence, there is actually an implicit 32-byte maximum length for a builder code due to the check in the Solady library. However, it can be made more explicit through documentation and comments or even an earlier check in the `BuilderCodes` contract that reverts if a given code is above 32 bytes.

Recommendation: Consider making the 32-byte maximum builder code length explicit by adding documentation and code comments or even an earlier check in the `BuilderCodes` contract that reverts if a given code is above 32 bytes.

Coinbase: Fixed in PR 93.

Cantina Managed: Fix verified.

3.5.2 Only owner can withdraw funds in `SimpleRewards` deviating from README

Severity: Informational

Context: README.md#L794, SimpleRewards.sol#L138-L148

Description: The README states that for the `SimpleRewards` contract, only the Manager can withdraw funds.

> SimpleRewards: Manager withdraws funds.

This is contrary to the code present in the `SimpleRewards` contract:

```
/// @inheritdoc CampaignHooks
function onWithdrawFunds(address sender, address campaign, address token, bytes calldata hookData)
    external
    virtual
    override
    onlyFlywheel
    returns (Flywheel.Payout memory payout)
{
    if (sender != owners[campaign]) revert Unauthorized();
    return (abi.decode(hookData, (Flywheel.Payout)));
}
```

Recommendation: Either fix the documentation if it is incorrect or the code to use `onlyManager` if the code is incorrect. Note that if the code is incorrect, the `CashbackRewards` has to be updated to override the `onWithdrawFunds` to include the owner check, as `CashbackRewards` currently inherits `onWithdrawFunds` function from `SimpleRewards` and the documentation states that the owner is to withdraw funds from `CashbackRewards`.

Coinbase: README will be corrected. Fixed in PR 95.

Cantina Managed: Fix verified.

3.5.3 Allowed state transitions for `SimpleRewards` and `CashbackRewards` deviates from README

Severity: Informational

Context: README.md#L787-L794, Flywheel.sol#L383-L403

Description: According to the README, the following are the allowed state transitions for `SimpleRewards` and `CashbackRewards`:

State	Who Can Transition	Available Functions	Special Behaviors
INACTIVE	• ACTIVE: Manager only • FINALIZING: Manager only	None	Initial/paused state
ACTIVE	• INACTIVE: Manager only • FINALIZING: Manager only	reward(), allocate(), deallocate(), distribute()	CashbackRewards: Payment must be collected in AuthCaptureEscrow
FINALIZING	• ACTIVE: Manager only • FINALIZED: Manager only	reward(), allocate(), deallocate(), distribute()	Grace period before closure
FINALIZED	None (terminal state)	None	CashbackRewards: Owner withdraws fundsSimpleRewards: Manager withdraws funds

Here is the implementation of the restrictions in the code.

```
/// @notice Updates the status of a campaign
///
/// @param campaign Address of the campaign
/// @param newStatus New status of the campaign
/// @param hookData Data for the campaign hook
function updateStatus(address campaign, CampaignStatus newStatus, bytes calldata hookData)
    external
    nonReentrant
    onlyExists(campaign)
{
    CampaignStatus oldStatus = _campaigns[campaign].status;
    if (
        newStatus == oldStatus // must update status
            || oldStatus == CampaignStatus.FINALIZED // cannot update from finalized
            || (oldStatus == CampaignStatus.FINALIZING && newStatus != CampaignStatus.FINALIZED) // finalizing
            ↪  can only update to finalized
    ) revert InvalidCampaignStatus();

    _campaigns[campaign].hooks.onUpdateStatus(msg.sender, campaign, oldStatus, newStatus, hookData);
    _campaigns[campaign].status = newStatus;
    emit CampaignStatusUpdated(campaign, msg.sender, oldStatus, newStatus);
}
```

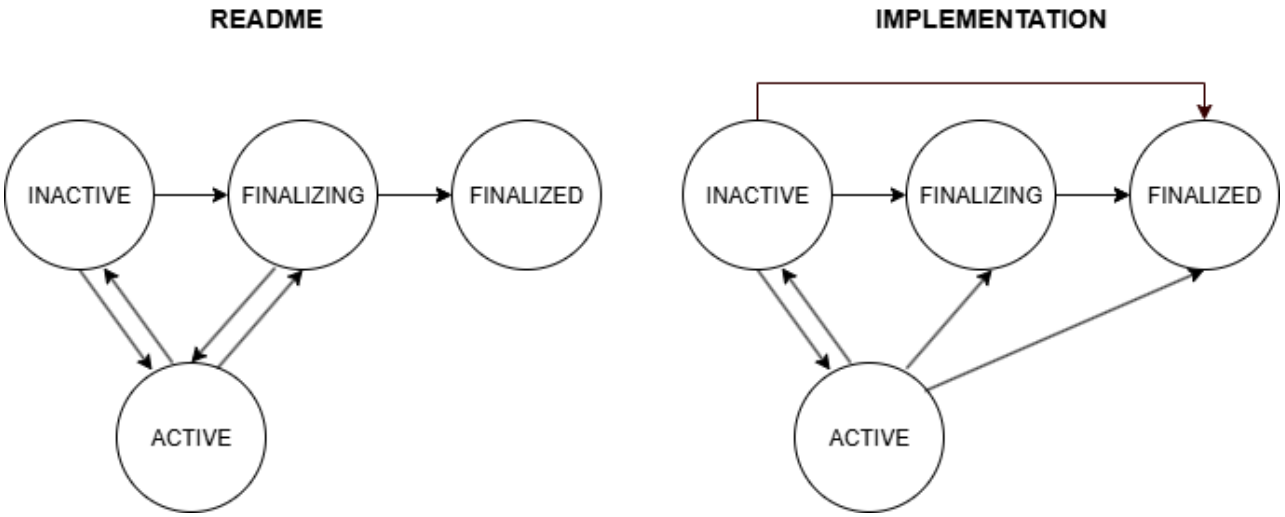
From the code we can see there are three restrictions:

- `newStatus == oldStatus`: The status must be updated.
- `oldStatus == CampaignStatus.FINALIZED`: `FINALIZED` cannot be updated from.
- `(oldStatus == CampaignStatus.FINALIZING && newStatus != CampaignStatus.FINALIZED)`: If `FINALIZING`, must update to `FINALIZED`.

This deviates from the README in the following ways:

- `INACTIVE` → `FINALIZED` and `ACTIVE` → `FINALIZED` are allowed, when they should not be.
- `FINALIZING` → `ACTIVE` is disallowed, when they should be.

The following illustration shows the difference between the state transition graphs between the README and the current implementation.



Recommendation: If the README is incorrect, then it should be corrected.

However, if the README is correct, then the following minimal changes should be made to align the implementation with the README:

- `(oldStatus == CampaignStatus.FINALIZING && newStatus != CampaignStatus.FINALIZED)` should be changed to `(oldStatus != CampaignStatus.FINALIZING && newStatus == CampaignStatus.FINALIZED)`, this means that if the new status is `FINALIZED`, then the old status MUST BE `FINALIZING`. This deletes the `INACTIVE` → `FINALIZED` and `ACTIVE` → `FINALIZED` edges and restores `FINALIZING` → `INACTIVE` and `FINALIZING` → `ACTIVE` edges.
- Then to delete `FINALIZING` → `INACTIVE` edge, add `(oldStatus == CampaignStatus.FINALIZING && newStatus == CampaignStatus.INACTIVE)`.

Coinbase: README will be corrected. Fixed in PR 95.

Cantina Managed: Fix verified.

3.5.4 Consider adding a check that `isActive` is true before disabling a conversion config

Severity: Informational

Context: AdConversion.sol#L495-L510

Description: An already disabled conversion config (`isActive = false` can be passed to `disableConversionConfig`) without reverting.

```
/// @notice Disables a conversion config for a campaign
/// @param campaign Address of the campaign
/// @param configId The ID of the conversion config to disable
/// @dev Only advertiser can disable conversion configs
function disableConversionConfig(address campaign, uint16 configId) external {
    if (msg.sender != state[campaign].advertiser) revert Unauthorized();

    if (configId == 0 || configId > conversionConfigCount[campaign]) {
        revert InvalidConversionConfigId();
    }

    // Disable the config
    conversionConfigs[campaign][configId].isActive = false;

    emit ConversionConfigStatusChanged(campaign, configId, false);
}
```

Recommendation: Add a check that `isActive` is true before disabling a conversion config.

Coinbase: Fixed in PR 100.

Cantina Managed: Fix verified.

3.5.5 Consider adding a check that allowed publisher refcode is in the registry in `AdConversion.onCreateCampaign`

Severity: Informational

Context: AdConversion.sol#L245-L252

Description: Refcodes that are not registered in the `BuilderCodes` registry can be passed to `onCreateCampaign` allowlist without reverting.

```
/// @inheritdoc CampaignHooks
function onCreateCampaign(address campaign, bytes calldata hookData) external override onlyFlywheel {
    // ...

    // Set up allowed publishers mapping if allowlist exists
    if (hasAllowlist) {
        uint256 count = allowedPublisherRefCodes.length;
        for (uint256 i = 0; i < count; i++) {
            allowedPublishers[campaign][allowedPublisherRefCodes[i]] = true;
            emit PublisherAddedToAllowlist(campaign, allowedPublisherRefCodes[i]);
        }
    }
    // ...
}
```

Recommendation: Add a check that the allowed publisher refcode is in the registry in `AdConversion.onCreateCampaign`.

Coinbase: Acknowledged. If the refcode isn't active by the time attribution happens then rewards will be rejected so validation is already happening here. We don't want to make protocol too restrictive.

Cantina Managed: Acknowledged.

3.5.6 Consider adding validation that the `maxRewardBasisPoints` cannot exceed `10_000`

Severity: Informational

Context: CashbackRewards.sol#L82-L91

Description: There is no input validation on the `maxRewardBasisPoints` which determines the maximum % collected in the `AuthCaptureEscrow` that can be allocated or distributed via `CashbackRewards`.

```
/// @inheritdoc CampaignHooks
function onCreateCampaign(address campaign, bytes calldata hookData) external override onlyFlywheel {
    (address owner, address manager, string memory uri, uint16 maxRewardBasisPoints_) =
        abi.decode(hookData, (address, address, string, uint16));
    owners[campaign] = owner;
    managers[campaign] = manager;
    campaignURI[campaign] = uri;
    maxRewardBasisPoints[campaign] = uint256(maxRewardBasisPoints_);
    emit CampaignCreated(campaign, owner, manager, uri);
}
```

That means that the campaign owner can set `maxRewardBasisPoints` that exceeds `10_000` which does not make sense as it would mean users are getting higher cashbacks than what they paid.

Recommendation: It might make sense to add validation that `maxRewardBasisPoints` cannot exceed `10_000`.

Coinbase: We have potential use cases that would desire a maximum over 100% so will leave unbounded for now. Data is immutable and visible to all parties so not concerned of abuse here.

Cantina Managed: Acknowledged.

3.5.7 Consider setting `maxRewardBasisPoints = 10_000` in `_validatePaymentReward` if `maxRewardBasisPoints` is `0`

Severity: Informational

Context: CashbackRewards.sol#L246-L267

Description: In the `CashbackRewards._validatePaymentReward` function, if `maxRewardBasisPoints = 0`, the reward amount validation is simply skipped.

```
/// @dev Validates a payment reward and returns the payment info hash
///
/// @param paymentReward The payment reward
/// @param campaign The campaign address
/// @param token The campaign token
/// @param operation The type of operation being performed
function _validatePaymentReward(
    PaymentReward memory paymentReward,
    address campaign,
    address token,
    RewardOperation operation
) internal view returns (bytes32 paymentInfoHash) {
    // ...
    // Early return if no max reward percentage is configured
    uint256 maxRewardBps = maxRewardBasisPoints[campaign];
    if (maxRewardBps == 0) return paymentInfoHash;
    // ...
    // Check total reward amount doesn't exceed the max allowed reward for this payment
    uint120 totalRewardAmount = previouslyRewardedAmount + paymentReward.payoutAmount;
    uint120 maxAllowedRewardAmount = uint120(paymentAmount * maxRewardBps / BASIS_POINTS_100_PERCENT);
    if (totalRewardAmount > maxAllowedRewardAmount) {
        revert RewardExceedsMaxPercentage(
            paymentInfoHash, maxAllowedRewardAmount, totalRewardAmount - maxAllowedRewardAmount
        );
    }
}
```

This means that if `maxRewardBasisPoints = 0`, campaign manager can set any cashback reward amounts they want, including ones that exceeding the amount of payment tracked in the `AuthCaptureEscrow`.

Recommendation: If `maxRewardBasisPoints = 0`, validation should continue but `maxRewardBasisPoints` should be set to `10_000` to allow the manager to cashback up to 100% of the payments tracked in `AuthCaptureEscrow`, but not any more than that.

Coinbase: Acknowledged. This is intentional, but we agree that `0` is quite permissive. Will probably not change.

Cantina Managed: Acknowledged.

3.5.8 Fund withdrawals can occur before `FINALIZED` in `SimpleRewards` and `CashbackRewards`, deviating from README

Severity: Informational

Context: README.md#L496-L511, Flywheel.sol#L366-L381, SimpleRewards.sol#L138-L148

Description: The README suggests that fund withdrawals can occur only when the campaign has reached `FINALIZED` state.

Aspect	AdConversion	CashbackRewards	SimpleRewards
...	...	...	
Fund Withdrawal	Advertiser only (FINALIZED + deadline)	Owner only (FINALIZED)	Manager only (FINALIZED)
...	...	...	

However, this isn't true for `SimpleRewards` and `CashbackRewards`, both `Flywheel.withdrawFunds` and `SimpleRewards.onWithdrawFunds` (which `CashbackRewards` inherits `onWithdrawFunds` from) do not check for the `FINALIZED` state.

```
/// @notice Withdraw tokens from a campaign
///
/// @param campaign Address of the campaign
/// @param token Address of the token to withdraw
/// @param hookData Data for the campaign hook
function withdrawFunds(address campaign, address token, bytes calldata hookData)
    external
    nonReentrant
```

```
    onlyExists(campaign)
{
    Payout memory payout = _campaigns[campaign].hooks.onWithdrawFunds(msg.sender, campaign, token, hookData);
    (address recipient, uint256 amount) = (payout.recipient, payout.amount);
    Campaign(campaign).sendTokens(token, recipient, amount);
    emit FundsWithdrawn(campaign, token, recipient, amount, payout.extraData);
    _assertTotalReservedSolvency(campaign, token, totalReserved[campaign][token]);
}
```

```
/// @inheritdoc CampaignHooks
function onWithdrawFunds(address sender, address campaign, address token, bytes calldata hookData)
    external
    virtual
    override
    onlyFlywheel
    returns (Flywheel.Payout memory payout)
{
    if (sender != owners[campaign]) revert Unauthorized();
    return (abi.decode(hookData, (Flywheel.Payout)));
}
```

Recommendation: If the README is incorrect, it should be corrected. Otherwise, the check for `FINALIZED` state should be present in the `withdrawFunds` flow for both the `SimpleRewards` and `CashbackRewards` contracts, note that the `AdConversion` contract is not affected as it does contain a check for `FINALIZED` state.

Coinbase: README will be corrected. Fixed in PR 95.

Cantina Managed: Fix verified.

3.5.9 Consider changing default hook behavior to abstract or no-op

Severity: Informational

Context: CampaignHooks.sol#L155-L163

Description/Recommendation: There is no point of campaign hook that reverts on things such as status update. Seems like you would never want that, so it is better to simply not implement and enforce implementation by inheritors, or as no-op by default.

Coinbase: Fixed in PR 94.

Cantina Managed: Fix verified.

3.5.10 Merge duplicated code

Severity: Informational

Context: Flywheel.sol#L245

Description/Recommendation: The `_allocateFees` shares most logic with the `allocate` function, can be merged into single internal function.

Coinbase: Will not change as we are satisfied with current readability.

Cantina Managed: Acknowledged.

3.5.11 Safeguard hooks by enforcing modifier

Severity: Informational

Context: CampaignHooks.sol#L48-L55

Description/Recommendation: All virtual functions here use `onlyFlywheel` modifier, but whenever someone overrides this function they have to include this modifier again if they want to preserve this behavior. There is a better way to implicitly enforce this, all `external` functions should be non-virtual and have this modifier and call respective `internal virtual` functions. For example:

```
function onReward(/*...*/) external onlyFlywheel {
  return _onReward(/*...*/);
}
```

```
function _onReward(/*...*/) internal virtual { } // devs override this more safely
```

Coinbase: Fixed in PR 94.

Cantina Managed: Fix verified.

3.5.12 Add events index

Severity: Informational

Context: Flywheel.sol#L113

Description/Recommendation: Consider indexing `token` and `recipient` as well. At very slightly higher cost this will allow much easier lookups of payments by users and tools, for analytics for instance.

Coinbase: Will not implement for now. Common production-grade indexing tools don't benefit from indexing parameters so we just want to do the minimum viable filtering which would be per-campaign.

Cantina Managed: Acknowledged.