CANTINA

Coinbase: SignatureDiscountValidator.sol

Security Review

Cantina Managed review by:
R0bert, Lead Security Researcher
0xWeiss, Security Researcher

November 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 Nov 6th to Nov 7th the Cantina team conducted a review of basenames on commit hash fb15e223. The team identified a total of **3** issues:

Issues Found

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

2.1 Scope

The security review had the following components in scope for basenames on commit hash fb15e223:

```
src/L2/discounts/SignatureDiscountValidator.sol
```

3 Findings

3.1 Informational

3.1.1 Missing event for signer rotations

Severity: Informational

Context: SignatureDiscountValidator.sol#L35-L39

Description: The `SignatureDiscountValidator.setSigner` function mutates the signer without emitting an event and the constructor also initializes the signer silently. Off-chain services (discount portals, caching layers, monitoring dashboards) cannot detect signer rotations or invalidate cached signatures promptly without polling storage directly, which both increases operational risk and delays revocations.

Recommendation: Declare `event SignerUpdated(address indexed oldSigner, address indexed newSign` and emit it in the constructor and `setSigner`, passing the previous value so downstream systems can react to signer rotations.

Coinbase: Acknowledged. Won't fix as we're expecting this to be used rarely (~1/yr maximally). Additionally we wouldn't expect to listen to this event since we would be the party executing the change in the first place. We can't come up with an external party that would care to listen to this event.

Cantina Managed: Acknowledged.

3.1.2 Incorrect NatSpec Comment for Signature Expiration Boundary Condition

Severity: Informational

Context: SybilResistanceVerifier.sol#L19

Description: `SybilResistanceVerifier.verifySignature` treats a signature as valid when the packed `expires` equals the current block timestamp because it only reverts when `expires < block.timestamp`.

```
if (expires < block.timestamp) revert SignatureExpired();
```

On the flipside, the NatSpec comment for `SignatureExpired` states, "Thrown when the signature expiry date >= block.timestamp,":

```
/// @notice Thrown when the signature expiry date >= block.timestamp.
   error SignatureExpired();
```

implying equality should already be rejected. The NatSpec is wrong as signatures should still be valid when their expiry date is equal to `block.timestamp`. The mismatch between documentation and behavior risks engineers relying on the wrong edge condition.

Recommendation: Update the NatSpec to reference only an expired signature when `block.timestamp` is bigger than the expiry date:

```
- /// @notice Thrown when the signature expiry date >= block.timestamp.
+ /// @notice Thrown when the signature expiry date < block.timestamp.
```

Coinbase: Fixed in commit 791f5a01.

Cantina Managed: Fixed.

3.1.3 Discount signatures scoped per validator instance

Severity: Informational

Context: SignatureDiscountValidator.sol#L48-L51

Description: `SignatureDiscountValidator.isValidDiscountRegistration` verifies only `(claimer, expires, signature)` and does not receive the `discountKey` being claimed. In theory, if multiple discount keys were mapped to the same validator, one signed authorization could be replayed across tiers because the signature payload omits the specific key. However, the deployment plan

explicitly assigns a single `discountKey` per validator/signer pair, so there is no path for a signature to ”jump” to a different tier. Under that operational constraint the previously identified risk is not exploitable and should be treated as informational documentation of the coupling assumption instead of an active issue.

Recommendation: Maintain (and ideally enforce in configuration/tests) the invariant that each validator instance serves exactly one `discountKey`. If future requirements ever call for multiple tiers behind the same validator, revisit this design and include the key or equivalent salt inside the signed payload to prevent cross-tier reuse.

Coinbase: Acknowledged. This is known and each validator will be ever assigned a single `discountKey`.

Cantina Managed: Acknowledged.