Espresso

Security Review

Cantina Managed review by:

Nirvan Tyagi, Lead Security Researcher
Mridul Garg, Security Researcher

May 6, 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	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

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 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

The Espresso Network is a shared source of truth that provides strong confirmations on transaction ordering and data across chains. The confirmations provided by HotShot are additive, meaning that rollups can keep giving their users pre-confirmations using their own existing sequencer.

From Mar 11th to Mar 23rd the Cantina team conducted a review of Espresso-monorepo on commit hash 9781caeb. The team identified a total of **22** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	5	3	2
Low Risk	15	14	1
Gas Optimizations	0	0	0
Informational	2	0	2
Total	**22**	**17**	**5**

3 Findings

3.1 Medium Risk

3.1.1 Rogue Key Attacks Against BLS Aggregate Signature Verification

Severity: Medium Risk

Context: bls_over_bn254.rs#L152-L154

Description: The approach taken in this implementation is secure if either (1) the aggregated messages are unique, or (2) verification keys are checked to have a proof of knowledge of the secret signing key. Otherwise, the scheme would be susceptible to "rogue key attacks".

Recommendation: In addition to proofs of knowledge, there are a few other variants of BLS aggregate/multi signatures that protect against rogue key attacks. See the resources below for more information on options:

- BLS MultiSigs.
- BLS IRTF Docs.

Espresso: Acknowledged.

Cantina Managed: Acknowledged. Espresso assured the team that in the larger Espresso system at another stage, BLS verification keys are being validated using proof-of-possession.

3.1.2 Signatures Validated to be On-Curve

Severity: Medium Risk

Context: bls_over_bn254.rs#L156

Description: For all verification methods, `verify`, `aggregate_verify`, and `multi_sig_verify`, if the signature is not checked to be on-curve (or in the appropriate large prime-order) subgroup of the elliptic curve then various "low-order" attacks can be used to leak information about the secret key.

Recommendation: For BN254, a subgroup check in G1 is not needed, but an inexpensive on-curve check is still important.

Espresso: Fixed in commit 2e9bd531.

Cantina Managed: Fix verified.

3.1.3 Software Execution Timing Side Channel of Secret Signing Key

Severity: Medium Risk

Context: bls_over_bn254.rs#L445

Description: The group scalar multiplication that makes up the core cryptographic operation of BLS signature is implemented using the `arkworks` library which currently does not support constant-time elliptic curve / field arithmetic. Fine-grained timing of this operation can lead to leakage of the bits of the singing key.

Recommendation: If the Espresso system admits a channel for untrusted users to time this operation, consider switching to a constant-time group scalar multiplication implementation.

Espresso: Acknowledged. The proposed fix was not applicable to BN254. Timing side channels still exist, but may likely be difficult to exploit in the context of the larger system.

Cantina Managed: Acknowledged.

3.1.4 Parameters and Payload Length Uncommitted Leading to Ambiguous Recover Behavior

Severity: Medium Risk

Context: avid_m.rs#L350

Description: Certain metadata like `param` including the `recovery_threshold` and `total_weights` and, more importantly, `payload_byte_len` are not part of the commitment. Instead these values are passed along in an unauthenticated manner as part of the shares. For example, metadata like `payload_byte_len` and `num_polys` are read from the zeroth share.

Consider the following concern: if `payload_byte_len` is not committed does that mean that the data can be arbitrarily truncated during decoding which might have implications for downstream execution? Currently, this function allows for arbitrary truncation.

Even without arbitrary truncation, there is a concern around ambiguous truncation of padded payloads. When the payload is not a multiple of `field_bytes_len`, the encoding pads with zeros during dispersal (See avid_m.rs#L182). Currently, the unauthenticated `payload_byte_len` is the only way to tell how to recover the correct data, i.e., truncate the correct number of padded zeros.

Recommendation: The following can be implemented to prevent ambiguous recovery:

- Adding `params` and `payload_byte_len` to the commitment.
- Verifying that the shares include vectors with the appropriate number of evaluations (`num_polys` evaluations in each share vector where `num_polys` is computed from `payload_byte_len` and `recovery_threshold`) in `verify_share`.

Espresso: Fixed in commit 8583aee8.

Cantina Managed: Fix verified. Espresso informed us that it was undesirable to change the commitment due to downstream serialization issues. The client opted to change the padding to a deterministic variable length padding scheme so that truncation is unambiguous. Depending on the trust model of how the VID scheme is being used within the larger Espresso system, there may still be concerns around parameters and payload length not being authenticated. The team is unable to verify without a broader review of the Espresso system.

3.1.5 Merkle Path Verification Can Equivocate on Depth of Proof

Severity: Medium Risk

Context: internal.rs#L127

Description: Verification of Merkle tree proofs only check up to proof length. Importantly, this means that proofs for leaves can verify even if the path is not of height depth. This can be problematic as it may be possible to have multiple values verify for a specific position at different heights of the path.

Take, for example, the `FixedLenPoseidon2Hash DigestAlgorithm` (implemented in prelude.rs#L60). The `digest_leaf algorithm` places the leaf element in the second to last position and the position in the last position: Consider a node at depth i with proof for `leaf_val` at depth i+1:

```
node = (default..., leaf_val, pos)
```

You could produce a proof of this `leaf_val` as the value for position `pos`. However, if $i+1$ is not max depth, then it may be possible to produce another valid proof for `pos`. Consider:

```
leaf_val = H(default..., leaf_val', pos)
```

Then one could produce another accepting proof for `leaf_val'` at depth i+2 for `pos` (if `pos` traversal path matched up with second to last position) for node:

```
node = (default..., leaf_val', pos)
```

Recommendation: There are a couple options that can be taken to mitigate:

- (Option 1) Enforce that all leaf nodes for successful lookups appear at max depth equals `height`.
- (Option 2) Set a different hashing approach for leaf nodes so that they cannot be confused for internal nodes, e.g., enforce leaves are hashed as: `H("LEAF", default..., leaf, pos)`, and internal nodes are hashed as `H("INTERNAL", ...)`.

Espresso: Fixed in commit 1d1c75b3.

Cantina Managed: Fix verified.

3.2 Low Risk

3.2.1 Domain Separation for Hash Function

Severity: Low Risk

Context: bls_over_bn254.rs#L443

Description: Domain separation in hash functions that are reasoned as random oracles during security proofs can be useful to prevent certain types of replay attacks within complex systems. Here, a cipher suite identifier is used as a domain separator.

Ethereum, for example, uses the following to calculate the domain hash (see building_-blocks/signatures/#domain-separation-and-forks):

> The domain, in turn, is calculated by the compute_domain() function which combines one of ten domain types with a mash-up of the fork version and the genesis validators root.

Recommendation: A more expressive domain separation functionality including context of signature use may be appropriate depending on the primitive's use within the larger Espresso system.

Espresso: Acknowledged.

Cantina Managed: Acknowledged.

3.2.2 `Poseidon2` Hash Structs not bound to use `Poseidon2`

Severity: Low Risk

Context: crhf.rs#L15-L22, crhf.rs#L57-L60

Description: In `FixedLenPoseidon2Hash` and `VariableLenPoseidon2Hash`, the documentation misleadingly states that this sponge-based CRHF uses the Poseidon2 permutation. However, there is nothing about these structs that require use of the Poseidon2 permutation. It will use the Poseidon2 permutation only if it is parameterized by a Poseidon2 Sponge `S`.

Recommendation: To prevent misuse of these primitives, i.e., a developer accidentally instantiating the CRHF with a different sponge, add a dummy trait `PoseidonSponge` and require `S: Sponge<U = F> + PoseidonSponge`. Add dummy implementations of `PoseidonSponge` to existing implementations.

Espresso: Fixed in PR 782. Merged into main, will be on our next tag release.

Cantina Managed: Fix verified.

3.2.3 Different from 4x4 MDS Matrix from Reference Implementation

Severity: Low Risk

Context: external.rs#L8-L11

Description: The 4x4 MDS matrix used here differs from that of the Horizen Labs reference implementation. Instead, it matches a different MDS matrix choice made by Plonky3. This means that hash evaluations will not match that of Horizen Labs.

Recommendation: If matching the reference is desired, switch the encoded MDS matrix as noted.

Espresso: Added code comment in commit 08ee11e8.

Cantina Managed: Fix verified.

3.2.4 Different Behavior for Length 4 States from Reference Implementation

Severity: Low Risk

Context: external.rs#L56

Description: When state length `T=4`, this implementation applies the 4x4 matrix M and then completes the circulant matrix (doubling the state). This matches the Plonky3 implementation, but differs from the

Horizen Labs reference implementation and the paper's description (Sec 5.1 of `https://eprint.iacr.org/2023/323.pdf`), in which for `T=4`, the circulant matrix is not applied.

Recommendation: If matching the reference is desired, switch the behavior for `T = 4` as noted.

Espresso: Yes, we will stick to plonky3's choice for now, we will clarify in the comments. Fixed in commit 08ee11e8.

Cantina Managed: Fix verified.

3.2.5 Out-of-bounds Check on Shares Provided During VID Recover

Severity: Low Risk

Context: avid_m.rs#L350

Description: The `shares` vector is indexed without checking if it is empty.

Recommendation: Include bounds check.

Espresso: Fixed in commit 8583aee8.

Cantina Managed: Fix verified.

3.2.6 Out-of-bounds Check on Payload Provided During VID Recover

Severity: Low Risk

Context: avid_m.rs#L356

Description: The `payload` vector is indexed without checking if it is empty.

Recommendation: Include bounds check.

Espresso: Fixed in commit 8583aee8.

Cantina Managed: Fix verified.

3.2.7 Out-of-bounds Check if Range and Payload Length Do Not Match in Recover

Severity: Low Risk

Context: avid_m.rs#L386

Description: The `payload` vector is indexed by the input range and also by assuming it is of length `num_polys`. If either of these assumptions are wrong, an out-of-bounds panic will occur.

Recommendation: Include bounds check.

Espresso: Fixed in commit 8583aee8.

Cantina Managed: Fix verified.

3.2.8 Summation of Weights May Overflow During VID Recover

Severity: Low Risk

Context: avid_m.rs#L358

Description: The `collected_weights` summation may overflow from adversarial input. Specifically, `.sum()` may overflow:

```
// **Panics:**

///
/// When calling `sum()` and a primitive integer type is being returned, this
/// method will panic if the computation overflows and debug assertions are
/// enabled.
```

Recommendation: Include overflow check during summation.

Espresso: Fixed in commit 8583aee8.

Cantina Managed: Fix verified.

3.2.9 Validity Check for Collected Weights does not Consider Overlapping Ranges

Severity: Low Risk

Context: avid_m.rs#L367

Description: For recover to succeed, the number of unique collected weights must be greater than the recovery threshold. The current check does not check the uniqueness condition which would be violated for overlapping weight ranges.

Recommendation: Include a uniqueness check or non-overlapping check.

Espresso: Fixed in commit 8583aee8.

Cantina Managed: Fix verified.

3.2.10 Memory Overflow from Malicious Range Input

Severity: Low Risk

Context: avid_m.rs#L374-L378

Description: A vector of length the size of weight of share is created during `recover`. If the input weight range is chosen maliciously to be large, memory can be overloaded.

Recommendation: Perform some sanity bound-checking on range lengths.

Espresso: Fixed in commit 8583aee8.

Cantina Managed: Fix verified.

3.2.11 Test Case Does Not Properly Unwrap Result

Severity: Low Risk

Context: avid_m.rs#L225-L235, avid_m.rs#L447

Description: The assertion in the test case checks if the result `is_ok()`. However, a failed verification will still return an ok result.

Recommendation: Check whether the value inside of the first ok result is ok to determine whether verification succeeded or failed.

Espresso: Fixed in commit 8583aee8.

Cantina Managed: Fix verified.

3.2.12 Out-of-bounds Check for Fixed Length Hash Input Size

Severity: Low Risk

Context: prelude.rs#L76

Description: Implementation of `DigestAlgorithm` for `FixedLenPoseidon2Hash` assumes `INPUT_SIZE >= 2` and indexes into array without checking bounds.

Recommendation: Include bounds check.

Espresso: Fixed in PR 778.

Cantina Managed: Fix verified.

3.2.13 Out-of-bounds Check for Merkle Tree Proof Arity

Severity: Low Risk

Context: internal.rs#L136

Description: Implementation of Merkle tree verification assumes proof vectors include witness of appropriate arity and indexes into array without checking bounds.

Recommendation: Include bounds check.

Espresso: Fixed in PR 778.

Cantina Managed: Fix verified.

3.2.14 Out-of-bounds Check on Shares In Namespaced Recover

Severity: Low Risk

Context: namespaced.rs#L154

Description: The `shares` vector is indexed without checking if it is empty.

Recommendation: Include bounds check.

Espresso: Fixed in PR 3128.

Cantina Managed: Fix verified.

3.2.15 Out-of-bounds Check on Additional Shares in Namespaced Recover

Severity: Low Risk

Context: namespaced.rs#L169

Description: The shares vector is indexed without checking if it is of appropriate length.

Recommendation: Include bounds check.

Espresso: Fixed in PR 3128.

Cantina Managed: Fix verified.

3.3 Informational

3.3.1 Signing Optimization to Avoid Extra Group Scalar Multiplication

Severity: Informational

Context: bls_over_bn254.rs#L119

Description: During message signing, the public verification key is generated from the signing key. This is done because a subcall to `KeyPair::sign` is made in which the API requires a `KeyPair` including both a signing key and verification key. The actual signing logic only requires the signing key; generation of the verification key is unneeded.

Recommendation: There are two API changes that can be made to avoid this extra cost:

- (Option 1) Adapt `KeyPair::sign` to instead operate as `SignKey::sign`.
- (Option 2) Change `SigningKey` to be `KeyPair`.

Espresso: Acknowledged.

Cantina Managed: Acknowledged.

3.3.2 Batch Merkle Tree Lookups for Efficient Lookups Over Contiguous Ranges

Severity: Informational

Context: avid_m.rs#L308

Description: Merkle lookup proofs for a consecutive range of leaves can be made more efficient than simply providing a separate Merkle path for each leaf in the range. Intuitively, the first set of nodes in the Merkle paths for consecutive leaves are redundant and can be omitted as they can be computed from the leaves themselves.

Recommendation: Switch to batch lookup proofs for contiguous ranges.

Espresso: Acknowledged.

Cantina Managed: Acknowledged.