Payy ZK Rollup

Security Review

Cantina Managed review by:

High Byte, Security Researcher
Jakub Heba, Associate Security Researcher

August 8, 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 Critical Risk . . . 4
3.1.1 Prover can inject arbitrary merkle trees during intermediate state . . . 4
3.2 Medium Risk . . . 4
3.2.1 Message parsing index miscalculation . . . 4
3.2.2 Failed burns result in permanent fund loss . . . 4
3.3 Low Risk . . . 5
3.3.1 Missing check for `publicInputs.length` . . . 5
3.3.2 Implement `removeProver` function . . . 5
3.3.3 `validFrom` can be set too far in the future . . . 5
3.3.4 Use constant for empty merkle root . . . 5
3.3.5 Ensure block height is strictly ascending . . . 6
3.3.6 Token transfer functions may not return boolean . . . 6
3.3.7 Validators do not sign data availability commitment . . . 6
3.4 Gas Optimization . . . 6
3.4.1 Simplify `substitutedBurns` mapping . . . 6
3.4.2 Reuse local variable . . . 7
3.4.3 `validatorsSet` can be a dynamic array instead of mapping . . . 7
3.5 Informational . . . 7
3.5.1 Ensure messages are zero when neither mint/burn . . . 7
3.5.2 Comment and code mismatch . . . 7

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

Payy is building onchain consumer banking by vertically integrating stablecoins, a privacy-preserving blockchain, global fiat ramps, payment cards and cross-chain DeFi — all accessible through a single interface for businesses and consumers.

From Jul 27th to Jul 31st the Cantina team conducted a review of payy-zk-rollup on commit hash 5cdaedf4. The team identified a total of **15** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	1	1	0
High Risk	0	0	0
Medium Risk	2	1	1
Low Risk	7	6	1
Gas Optimizations	3	3	0
Informational	2	2	0
Total	**15**	**13**	**2**

The Cantina Managed team reviewed Polybase Labs's `zk-rollup` holistically on commit hash 4a83f4be and concluded that all issues were addressed and no new vulnerabilities were identified.

3 Findings

3.1 Critical Risk

3.1.1 Prover can inject arbitrary merkle trees during intermediate state

Severity: Critical Risk

Context: main.nr#L80-L85

Description/Recommendation: It may seem that this should take care of validation:

```
assert(root == new_root, "New root is not valid");
```

but even that may not be sufficient. It is possible to inject arbitrary input commitments and even output commitments, as long as the final `root` generates to `new_root`, and the first round of input commitments are present in the `old_root`.

- In round $i = 0$, the inputs must match. the output commitments loop generates new merkle root, which can be any arbitrary path.
- In round $i = 1$, the inputs can now be forged due to arbitrary root set in round $i = 0$.
- In round $i = 2$, the inputs can also be forged. the output loop has to match the original `new_root`.

Polybase Labs: Fixed in PR 1885.

Cantina Managed: Fix verified.

3.2 Medium Risk

3.2.1 Message parsing index miscalculation

Severity: Medium Risk

Context: RollupV1.sol#L354-L375

Description: In the `verifyMessages` function of `RollupV1.sol`, there is a bug in how the message index is advanced for different transaction types. The function advances the index by only 1 for send transactions (`kind=1`) and padding transactions (`kind=0`), but advances by 6 for mint (`kind=2`) and burn (`kind=3`) transactions.

According to the ZK circuit design and the message structure, each UTXO transaction consistently produces 6 messages regardless of transaction type: `kind`, `note_kind`, `value`, `hash`, `burn_addr`, and the last one is not used. This causes the message parser to misalign after processing any `send` transaction, reading subsequent transaction data from incorrect offsets.

Recommendation: We recommend updating the `verifyMessages` function to advance the index by 6 for all transaction types. This ensures consistent message parsing alignment across all transaction types, matching the expected 6-message structure from the ZK circuits.

Polybase Labs: Fixed in commit e66ca7b5. The purpose of this is to only advance as many as needed - this has been asserted in the ZK now.

Cantina Managed: Fix verified.

3.2.2 Failed burns result in permanent fund loss

Severity: Medium Risk

Context: RollupV1.sol#L417-L429

Summary: In the `verifyBurn` function, when `executeBurn` is called and fails, the function continues processing without reverting, despite the user's funds already being burned on L2. The `executeBurn` function correctly returns a success flag indicating whether the transfer succeeded, but this return value is ignored. This creates a scenario where user funds are permanently destroyed - they have been removed from the L2 Merkle tree through the `burn` operation but never arrive on L1 due to the failed transfer. Common failure scenarios include tokens being paused, addresses being blacklisted by the token contract, or other transfer restrictions. Currently, there is no mechanism to recover these funds since they no longer exist on L2 and were not successfully transferred on L1. The design allows for systematic loss of user funds whenever L1 transfers fail for any reason.

Recommendation: We recommend implementing a proper failure handling mechanism for failed burns. Options include creating a retry mechanism where failed burns are stored and can be retried with different parameters, implementing a two-phase burn system where L2 burns are separate from L1 claims, allowing users to specify fallback addresses for burns, or implementing an escrow pattern where funds are held until successfully transferred. At minimum, failed burns should be tracked on-chain with events that include all necessary information for recovery.

Polybase Labs: Acknowledged.

Cantina Managed: Acknowledged.

3.3 Low Risk

3.3.1 Missing check for `publicInputs.length`

Severity: Low Risk

Context: RollupV1.sol#L310-L324

Description/Recommendation: There is a missing check for `publicInputs.length`. Perhaps the external call to `zkVerifiers.verifier` will do that but it can be modified to a version that doesn't, hence it is best check here, otherwise there's potential to set it to `length == 3` which may cause issues in the verifier.

Polybase Labs: Fixed in PR 1898.

Cantina Managed: Fix verified.

3.3.2 Implement `removeProver` function

Severity: Low Risk

Context: RollupV1.sol#L149-L151

Description/Recommendation: There is a method to `addProver` but no `removeProver`.

Polybase Labs: Fixed in PR 1896.

Cantina Managed: Fix verified.

3.3.3 `validFrom` can be set too far in the future

Severity: Low Risk

Context: RollupV1.sol#L729

Description/Recommendation: If you make a mistake and set really large `validFrom` you would lock yourself out from setting new validators. It is recommended to add safety checks, perhaps limit the bounds of how far in the future it can consider.

Polybase Labs: Fixed in PR 1908.

Cantina Managed: Fix verified.

3.3.4 Use constant for empty merkle root

Severity: Low Risk

Context: RollupV1.sol#L136

Description/Recommendation: There is a potential to insert backdoors by setting an arbitrary initial merkle root. It is advised to only allow for the constant root of an empty tree which is easier to verifier.

Polybase Labs: Fixed in PR 1900.

Cantina Managed: Fix verified.

3.3.5 Ensure block height is strictly ascending

Severity: Low Risk

Context: RollupV1.sol#L326

Description/Recommendation: Ensure the new block height is greater than the previous block height, ie. `require(height > blockHeight)`.

Polybase Labs: Fixed in PR 1899.

Cantina Managed: Fix verified.

3.3.6 Token transfer functions may not return boolean

Severity: Low Risk

Context: RollupV1.sol#L514-L521

Description/Recommendation: Note tokens may differ across chains so if you plan to deploy to new chains it might not return boolean which will lead to reverts.

Polybase Labs: Fixed in PR 1875.

Cantina Managed: Fix verified.

3.3.7 Validators do not sign data availability commitment

Severity: Low Risk

Context: RollupV1.sol#L657-L662

Description: In the `verifyValidatorSignatures` function, validators sign a message hash derived from `newRoot`, `height`, and `otherHashFromBlockHash`, but they do not sign the `commitHash` which represents the commitment to transaction data stored on the data availability layer. The `commitHash` is a critical component that ensures the underlying transaction data is available and can be retrieved from the DA layer.

This value is included in the `publicInputs` array and is validated as part of the ZK proof, but validators can currently sign state transitions without attesting to the availability of the underlying data. This breaks the security model of the rollup system where validators should be ensuring both the correctness of state transitions and the availability of transaction data.

The presence of the empty `verifyCommitHash` function suggests this was intended to be checked but the implementation is incomplete.

Recommendation: We recommend modifying the `getSignatureMessageHash` function to include the `commitHash` as part of the data that validators must sign. This ensures validators are attesting to both the state transition and the availability of the underlying transaction data.

Polybase Labs: Acknowledged. Won't fix.

Cantina Managed: Acknowledged.

3.4 Gas Optimization

3.4.1 Simplify `substitutedBurns` mapping

Severity: Gas Optimization

Context: RollupV1.sol#L89-L90

Description/Recommendation: The `substitutedBurns` mapping is complex and requires multiple hashes to generate storage key. It would be more efficient to rewrite something like this:

```
mapping (bytes32 => address) substitutedBurns;

 //such that:
bytes32 key = keccak256(abi.encode(hash, burnAddress, noteKind, amount));
substitutedBurns[key] = subtituteAdderss;
```

consider implementing internal function to access it.

Polybase Labs: Fixed in PR 1903.

Cantina Managed: Fix verified.

3.4.2 Reuse local variable

Severity: Gas Optimization

Context: RollupV1.sol#L419

Description/Recommendation: Use `substitutor` variable instead of reading same storage again.

Polybase Labs: Fixed in PR 1897.

Cantina Managed: Fix verified.

3.4.3 `validatorsSet` can be a dynamic array instead of mapping

Severity: Gas Optimization

Context: RollupV1.sol#L723-L727

Description/Recommendation: `validatorsSet` makes more sense as a dynamic array. Simply `validatorSet.push`, it natively manages the `.length` property, cheaper lookups.

Polybase Labs: Fixed in PR 1876.

Cantina Managed: Fix verified.

3.5 Informational

3.5.1 Ensure messages are zero when neither mint/burn

Severity: Informational

Context: main.nr#L48

Description/Recommendation: Include asserts that match this comment, this will prevent potential issues in case any of them are non-zero.

Polybase Labs: Fixed in PR 1877.

Cantina Managed: Fix verified.

3.5.2 Comment and code mismatch

Severity: Informational

Context: RollupV1.sol#L358-L359

Description/Recommendation: The comment doesn't match the code. This reads `uint256`, not just single byte.

Polybase Labs: Fixed in PR 1907.

Cantina Managed: Fix verified.