Pulse In Private Security Review

28.03.2025

Conducted by:

Kann, Lead Security Researcher

Ivan Fitro, Lead Security Researcher

Table of Contents

1 About Kann

Kann a Security Reseacher and Founder of Kann Audits.

2 About Ivan Fitro

Ivan Fitro a Security Reseacher and Founding SR in Kann Audits.

3 Disclaimer

Audits are a time, resource, and expertise bound effort where trained experts evaluate smart contracts using a combination of automated and manual techniques to identify as many vulnerabilities as possible. Audits can show the presence of vulnerabilities **but not their absence**.

4 Risk classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

4.1 Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - only a small amount of funds can be lost or a functionality of the protocol is affected.
- **Low** - any kind of unexpected behaviour that's not so critical.

4.2 Likelihood

- **High** - direct attack vector; the cost is relatively low to the amount of funds that can be lost.
- **Medium** - only conditionally incentivized attack vector, but still relatively likely.
- **Low** - too many or too unlikely assumptions; provides little or no incentive.

4.3 Actions required by severity level

- **Critical** - client **must** fix the issue.
- **High** - client **must** fix the issue.
- **Medium** - client **should** fix the issue.
- **Low** - client **could** fix the issue.

5 Executive summary

Overview

Project Name	Pulse In Private
Repository	https://github.com/alexbabits/pip
Commit hash	ce32639
Resolution	N/A
Documentation	https://pip-1.gitbook.io/pip-docs
Methods	Manual review

Scope

/src/Pip.sol
/circuits/height12/withdraw.circom

Issues Found

Critical risk	0
High risk	0
Medium risk	2
Low risk	0
Informational	0

6 Findings

6.1 Medium risk

6.1.1 Rescue Tokens: Implement a Rescue Funds Function for stucked funds

Severity: *Medium risk*

Description: In Pip.sol, when a user deposits, they receive a nullifier code which is required when withdrawing from a different address. However, since it is possible for a user to forget their nullifier code, the funds could become permanently stuck in the contract.

Recommendation: Add a rescueFunds function along with an on-chain view function to check whether the nullifier hash (used during withdrawal) has been consumed. This would enable the protocol to verify that the user genuinely forgot their nullifier code and facilitate the recovery of their funds.

Resolution: Acknowledged

6.1.2 Constant Fee for Relayer

Severity: *Medium risk*

Description: Gas consumption on the chain could increase—either temporarily during high congestion periods or permanently due to evolving network conditions. With the current fixed fee parameters, relayers may not receive sufficient incentives to cover their gas expenses. In some cases, a relayer might end up paying more in gas fees than the reward they receive. This imbalance not only discourages relayer participation but also forces new addresses to source gas from elsewhere in order to call the withdraw() function, which negatively impacts the protocol's overall usability and efficiency.

Recommendation: User-Defined Relayer Fee: Allow users to input a custom relayer fee when initiating a transaction. This approach gives users the flexibility to determine what they consider a fair incentive for the relayer, based on current network conditions and their own willingness to pay.

Dynamic Gas Fee Function: Alternatively, implement a function that can adjust the gas fee for relayers. This function would allow the protocol administrator (or even potentially the community through governance mechanisms) to update the relayer fee dynamically in response to changes in gas prices. This ensures that the fee remains competitive and sufficient to cover relayer costs.

Resolution: Fixed