Pashov Audit Group

Trove Security Review

January 4th 2026 - January 5th 2026

Contents

1. About Pashov Audit Group

Pashov Audit Group consists of 40+ freelance security researchers, who are well proven in the space - most have earned over $100k in public contest rewards, are multi-time champions or have truly excelled in audits with us. We only work with proven and motivated talent.

With over 300 security audits completed — uncovering and helping patch thousands of vulnerabilities — the group strives to create the absolute very best audit journey possible. While 100% security is never possible to guarantee, we do guarantee you our team's best efforts for your project.

Check out our previous work here or reach out on Twitter @pashovkrum.

2. Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where we try to find as many vulnerabilities as possible. We can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

3. Risk Classification

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

Impact

- **High** - leads to a significant material loss of assets in the protocol or significantly harms a group of users
- **Medium** - leads to a moderate material loss of assets in the protocol or moderately harms a group of users
- **Low** - leads to a minor material loss of assets in the protocol or harms a small group of users

Likelihood

- **High** - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost
- **Medium** - only a conditionally incentivized attack vector, but still relatively likely
- **Low** - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive

4. About Trove

Trove is an token sale smart contract that allows users to participate using whitelisted ERC-20 tokens or native currency, while tracking individual and total contributions. It supports controlled refunds after the sale ends, and owner-managed fund withdrawal with strict lifecycle enforcement.

5. Executive Summary

A time-boxed security review of the **P123D456/sale-a** repository was done by Pashov Audit Group, during which **Shaka**, **DemoreXTess**, **Ch_301** engaged to review **Trove**. A total of **3** issues were uncovered.

Protocol Summary	
Project Name	Trove
Protocol Type	Tokensale
Timeline	January 4th 2026 - January 5th 2026

Review commit hash:

- 340b00281d1aa1a29297c8d3b4946bf0944775fe (P123D456/sale-a)

Scope

`Sale.sol`

6. Findings

Findings count

Severity	Amount
Low	3
Total findings	**3**

Summary of findings

ID	Title	Severity	Status
[L-01]	Fee on transfer tokens are not supported	Low	Acknowledged
[L-02]	Unbounded loops could lead to out-of-gas errors	Low	Acknowledged
[L-03]	Gas Griefing and DoS via Malicious Fallback in `refund()`	Low	Acknowledged

Low findings

[L-01] Fee on transfer tokens are not supported

When tokens are transferred in the `participate()` function, it is assumed that the amount transferred is equal to the amount received. This is not true for fee on transfer tokens, where a percentage of the transfer is taken as a fee and not received by the recipient.

It is recommended to either document the lack of support for fee on transfer tokens, or to modify the logic to account for the fees taken during transfers.

[L-02] Unbounded loops could lead to out-of-gas errors

The `whitelistedTokens()` and `participatedTokens()` functions use pagination to return token lists. This prevents the transaction from running out of gas when the lists are long.

However, the `refund()` and `totalParticipations()` functions do not use pagination and look through the entire list of tokens for a user or the total participations, respectively. If these lists are long, calling these functions could run out of gas.

Consider implementing pagination for these functions as well or limiting the maximum number of tokens that can be whitelisted.

[L-03] Gas Griefing and DoS via Malicious Fallback in `refund()`

The `refund()` function utilizes OpenZeppelin's `Address.sendValue` to return native tokens, which forwards all available gas to the recipient. A malicious user can participate with a negligible amount (e.g., 1 wei) and implement a gas-consuming loop in their contract's `receive()` function. When the `refundExecutor` attempts to process the refund, the attacker's fallback consumes all forwarded gas, causing the transaction to revert (Out of Gas). It is recommended to adopt a "Pull-over-Push" pattern, allowing users to claim their own refunds.