Invariant Labs

Prove it, then ship.

Security Report for

MultiSig Wallet

2025-11-03

Auditor:

Misbahu A.

Security Analysis Report: MultiSigWallet

Date: 2025-11-03

Analyst: Misbahu

Code: https://github.com/rebecacuevas/multi-sig-wallet/tree/main

X profile for firm: x.com/Invariants_

Executive Summary

This report details the analysis of the `MultiSigWallet` contract. The contract's internal logic for transaction confirmation and execution is sound and secure against common exploits like re-entrancy.

The single, **Critical** finding is a **design-level flaw** that violates a core operational invariant: the immutability of the owner-set and threshold creates a direct risk of **permanent fund loss** if a sufficient number of owners lose their keys.

Two Informational findings related to gas optimization and state management are also noted.

System Invariants

The system is a standard `MultiSigWallet` contract on Ethereum, deployed as an immutable contract. The analysis was performed against the following confirmed operational invariants.

- **INVARIANT-1 (Execution Threshold):** A transaction T can only be executed if and only if T.numConfirmations >= numConfirmationsRequired
- **INVARIANT-2 (Owner-Only):** Only an address A where isOwner[A] == true can call `submitTransaction`, `confirmTransaction`, `executeTransaction`, or `revokeConfirmation`.
- **INVARIANT-3 (Immutable Quorum):** The set of `owners` and the `numConfirmationsRequired` threshold are immutable post-deployment.
- **INVARIANT-4 (No Double Vote):** An owner cannot confirm a transaction txIndex more than once.
- **INVARIANT-5 (No Re-execution):** An executed transaction T (where T.executed == true) can never be executed again.
- **INVARIANT-6 (State-Before-Call):** The `executeTransaction` function sets `transaction.executed = true` *before* the external call, mitigating re-entrancy.
- **INVARIANT-7 (No Fund Lock):** It is impossible for the contract to enter a state where held funds (ETH) cannot be transferred out via a valid, executable transaction.

Findings

● C-01: [Critical] Violation of Invariant-7 Leads to Permanent Fund Lock

- **Description:** The contract violates **INVARIANT-7 (No Fund Lock)** due to its strict adherence to **INVARIANT-3 (Immutable Quorum)**. The `owners` list and `numConfirmationsRequired` are set only in the constructor and can never be modified.
- **Impact:** If a sufficient number of owners lose their private keys, the number of active, available owners may fall below `numConfirmationsRequired`. If this occurs, the check `transaction.numConfirmations >= numConfirmationsRequired` (line 126) can never be met for any transaction. All funds held by the contract will be **permanently and irrecoverably locked**.
- **Recommendation:** This immutable design is suitable only for short-term or low-value use cases. For any persistent treasury, a new contract must be deployed that includes owner-gated functions to add/remove owners and modify the confirmation threshold.

□ M-01: [Medium] No Findings

- No medium-severity findings were identified. The `revokeConfirmation` function, which could be used for gas griefing, is a necessary safety feature to prevent the execution of malicious or erroneous transactions and we think it is better classified as intended behavior.

● I-01: [Informational] Lack of Initial Confirmation on Submission

- **Description:** The `submitTransaction` function (lines 90-104) does not automatically add the submitter's confirmation. The `msg.sender` must send a second, separate transaction to `confirmTransaction` (lines 106-117) to add their vote.
- **Impact:** This is a UX and gas-inefficient design. It doubles the gas cost and operational steps required for the submitter of a transaction.
- **Recommendation:** Modify `submitTransaction` to also call `confirmTransaction` internally or otherwise register the submitter's vote, setting `numConfirmations` to 1 and `isConfirmed[txIndex][msg.sender]` to `true`.

● I-02: [Informational] No Transaction Pruning or Cancellation

- **Description:** There is no mechanism to cancel or remove a transaction from the `transactions` array. Transactions that are submitted but fail to gain a quorum of signatures remain in the array indefinitely.
- **Impact:** This leads to minor state bloat and clutters the contract's history. It does not pose a direct security risk but is operationally untidy.
- **Recommendation:** Consider adding an `ownerOnly` function to cancel a non-executed transaction (e.g., `cancelTransaction(uint256 _txIndex)`). This is a low-priority enhancement.