MultiSigWallet Audit Report

Version 1.0

Cyfrin.io

September 21, 2025

MultiSig Audit Report

Vinay

September 21, 2025

Prepared by: Vinay
Lead Auditor: - Vinay

Table of Contents

Protocol Summary

- Role-based multisig wallet with timelock.
- Up to 5 signers; 3 confirmations required.
- Owner-only can propose transactions; signers confirm/revoke.
- Timelock based on ETH value:
  - <1 ETH → no delay
  - 1–10 ETH → 1 day
  - 10–100 ETH → 2 days
  - ≥ 100 ETH → 7 days
- Tracks confirmations per signer; prevents double execution.
- Uses OpenZeppelin Ownable, AccessControl, ReentrancyGuard.
- Executes ETH via safe call, CEI pattern.
- Centralized proposal and signer management; minor gas/audit optimizations possible.

Disclaimer

The Vinay's team makes all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the findings provided in this document. A security audit by the team is not an endorsement of the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the Solidity implementation of the contracts.

Risk Classification

		Impact		
		High	Medium	Low
	High	H	H/M	M
Likelihood	Medium	H/M	M	M/L
	Low	M	M/L	L

We use the CodeHawks severity matrix to determine severity. See the documentation for more details.

Audit Details

Scope

```
./src/
#-- MultiSigTimelock.sol
```

Roles

- **Owner**: Can add/remove signers and propose transactions.
- **Signers**: Can confirm, revoke, and execute transactions.

Executive Summary

Issues found

Severity	Number of issues found
High	1
Medium	0
Low	1
Gas	1
Info	1
Total	4

Findings

High

[H-1] Centralization Risk in `grantSigningRole` (Owner Can Add Malicious Signers)

Description: The `grantSigningRole` function allows the contract owner to arbitrarily assign signing roles to new accounts. Since up to 5 signers are allowed and only 3 confirmations are required, the owner could add malicious or colluding accounts to reach the REQUIRED_CONFIRMATIONS threshold without actual multisig consensus. This undermines the decentralized trust model expected of a multisig wallet.

Impact: - The multisig effectively degrades into a centralized wallet controlled by the owner.

- Collusion between the owner and newly added signers can bypass intended protections.
- Users relying on distributed signer trust can lose funds.

Proof of Concept: - Owner deploys the wallet, automatically becoming the first signer.

- Owner calls grantSigningRole repeatedly to add 2 additional malicious accounts.
- With 3 signers now controlled by the owner, any transaction can be confirmed and executed unilaterally, bypassing the need for distributed approvals.

Recommended Mitigation: - Require signer consensus (e.g., majority of existing signers must approve before a new signer is added).

- Alternatively, restrict signer management to a separate governance process or timelocked proposal.
- Emit strong warnings if full decentralization is not intended.

Low

[L-1] Pending Transaction Confirmation Not Updated When Signer Is Removed (Logic Issue)

Description: When a signer is removed via `revokeSigningRole`, the contract does not update confirmations on pending transactions that the signer had previously approved. As a result, a removed signer's confirmations still count toward REQUIRED_CONFIRMATIONS, potentially allowing a transaction to execute with fewer active signers than intended.

Impact: - The multisig confirmation logic becomes inaccurate, giving removed signers indirect influence.

- Could lead to premature execution of transactions without full active signer consensus.

Proof of Concept: - Signer Alice confirms a pending transaction txnId = 1.

- Owner removes Alice using revokeSigningRole.
- The transaction's confirmation count still includes Alice's approval.

- Only 2 active signers remain, but REQUIRED_CONFIRMATIONS = 3. The transaction can still execute if another single signer confirms, bypassing intended multisig security.

Recommended Mitigation: - Iterate over all pending transactions and clear the removed signer's signatures, decrementing the confirmation count where necessary.

- For scalability, consider tracking only active transactions or using a more gas-efficient data structure to avoid iterating all transactions on every signer removal.

Informational

[I-1] No Balance Check or Proposer Tracking in `_proposeTransaction` (Informational)

Description: The `_proposeTransaction` function does not check whether the proposed value is less than or equal to the contract balance at the time of proposal. Additionally, The `_proposeTransaction` function is restricted to onlyOwner, preventing signers from proposing transactions themselves. This centralizes transaction initiation and may reduce flexibility in multisig operations.

Impact: - Users may propose transactions that cannot be executed later due to insufficient contract balance. No direct security risk since execution time balance checks still exist.

- Only the owner can propose transactions, which reduces the decentralized nature of the multisig.
- Lack of proposer tracking reduces transparency for auditing and governance.

Recommended Mitigation: - Consider allowing all signers to propose transactions.

- Add a proposer field in the Transaction struct to store the address of the signer who proposed the transaction.

Gas

[G-1] Redundant `nonReentrant` Modifier (Gas Optimization)

Description: Some functions, like grantSigningRole, revokeSigningRole, and proposeTransaction, are marked nonReentrant but do not perform external calls. The modifier adds unnecessary gas overhead without providing additional security.

Impact: - Increased gas costs for routine operations.

- No actual security benefit for functions without external interactions.

Recommended Mitigation: - Remove nonReentrant from functions that do not transfer ETH or call external contracts.

- Keep nonReentrant only on functions like executeTransaction that interact with external addresses.