Smart Contract Audit Report

- **Contract Name**: AgentDelegator
- **Audit Version**: 1.0
- **Audit Date**: 2025-04-17
- **Auditor**: DeepSeek-R1 with audit agent
- **GitHub / Code Reference**: https://github.com/verisense-network/aitonomy-solidity/tree/62cf075b79f25307b4f31cb17

1. Executive Summary

This audit evaluates the security and efficiency of the AgentDelegator contract, which facilitates delegated token withdrawals with signed approvals. The contract allows users to withdraw tokens by submitting messages signed the Aitonomy agents using TSS function of Verisense, which are verified to ensure authenticity. The audit focuses on:

- Security vulnerabilities
- Code correctness & best practices
- Gas efficiency optimizations

Key Findings:

☒ Critical issues fixed in previous versions (signature verification bypass in `batch_withdraw`)
☒ Reentrancy protection applied via `nonReentrant` modifier
☒ Input validation enforced (rewards length check, `messageBytes` decode checks)
☐ Minor gas optimization opportunities identified (**Low severity**)

Overall Assessment:

- **Critical Issues**: 0
- **High-Risk Issues**: 0
- **Medium-Risk Issues**: 1 (front-running due to external calls)
- **Low-Risk Issues**: 2

2. Detailed Findings

Fixed Issues (Previously Critical)

1. Signature Verification Bypass in `batch_withdraw`

- **Issue**: Previously, `_withdraw(rewards[i]._messageBytes, rewards[i]._messageBytes)` incorrectly passed `_messageBytes` as the signature, causing verification to fail unless the message coincidentally matched a valid signature.
- **Fix**: Now correctly passes `_signature` as the second argument.

Medium Severity Issues

1. Front-Running Due to External TokenContract Dependency

- **Description**: The `transfer()` call in `_withdraw()` is external, meaning a malicious `TokenContract` could attempt a reentrancy attack.
- **Mitigation**: The `nonReentrant` modifier prevents reentrancy, but a DoS attack (reverting transfers to block withdrawals) is still possible.
- **Recommendation**: Consider using `SafeERC20` from OpenZeppelin. If `TokenContract` is trusted and audited, this is acceptable.

2. Lack of EIP-712 Structured Data Signing

- **Description**: The signature scheme uses legacy `eth_sign ("\x19Ethereum Signed Message" prefix)`, which is less secure than EIP-712 for contract interactions.
- **Recommendation**: Upgrade to EIP-712 for better wallet UX and security.

Low-Severity Issues (Gas & Code Style)

1. Unbounded `user_tickets` Array Growth

- **Issue**: `user_tickets[destination].push(sequence)` grows indefinitely, increasing gas costs over time for `user_withdraws()`.
- **Recommendation**: Store only the latest sequence if historical data is not critical. Alternatively, emit events instead of storing all withdrawals on-chain.

2. Redundant Storage Write in `user_tickets_length`

- **Issue**: `user_tickets_length[destination]` is updated even if the array already tracks length.
- **Optimization**: Consider removing this mapping if `user_tickets[user].length` suffices.

3. Test Coverage & Edge Cases

Verified Scenarios:

☒ **Single withdrawal** (`withdraw()`) with valid signature.
☒ **Batch withdrawal** (`batch_withdraw()`) with multiple valid signatures.
☒ **Rejected withdrawals** (invalid signature, already claimed ticket, wrong user).
☒ **Reentrancy attempt blocked** (tested via malicious `TokenContract`).

Unverified Edge Cases (Manual Testing Recommended):

- Very large `rewards[]` array in `batch_withdraw()` (gas limit risks?).
- Edge-case `sequence` values (max `uint256`, zero, invalid encoding).

4. Recommendations

Severity	Recommendation
Medium	Implement **EIP-712** for structured signing (better security & UX).
Low	Optimize `user_tickets` storage by using events instead of arrays where possible.
Low	Remove redundant `user_tickets_length` if not needed.

5. Conclusion

The `AgentDelegator` contract is **secure** for production use, with fixes applied for previous major issues. Minor gas optimizations and UX improvements (EIP-712 adoption) are recommended but not critical.

Final Approval Status: **Approved** (With Minor Recommendations)

Auditor: DeepSeek-R1 with audit agent

Date: 2025-04-17