CPChain PoS: CpChainBase Security Audit Report

DogScan Security Team

August 4th, 2024

Contents

DogScan Security Audit Report

Project	CPChain PoS
Contract File	`CpChainBase.sol`
Source Path	src/core/pos/CpChainBase.sol
Commit	`b098dffd4589081b8a9996972cc044be552e321a`
Audit Date	August 4th, 2024
Report Version	1.0

1. Executive Summary

We conducted a comprehensive security audit of the `CpChainBase` contract. This contract serves as the base staking pool contract in the CPChain PoS system, responsible for managing user deposits, share allocation, and staker list maintenance.

The audit results show a generally good overall security posture with proper access control implementation that mitigates most security risks. One low-severity issue was identified involving a logical flaw in the staker management functionality.

The audit results revealed one low-severity issue primarily related to state management logic.

Overall Risk Rating: Low

We recommend the project team fix the identified state management issue to ensure perfect state consistency.

2. Audit Scope

The audit scope covers the complete functionality of the `CpChainBase` contract:

Contract Information:

- Contract Type: Base Staking Pool Contract
- Main Functions: User deposit management, share calculation, staker list maintenance

Key Audit Areas:

- Deposit and withdrawal mechanisms
- Share calculation logic
- Staker list management
- Access control patterns
- State variable synchronization

3. Audit Methodology

This audit employed a multi-agent AI security analysis framework specifically designed for smart contract security assessment:

1. Specialized Analysis Modules:

- **State Management Expert**: Reviews state variable consistency and synchronization logic
- **Access Control Expert**: Evaluates permission management and role control
- **Arithmetic Security Expert**: Examines share calculations and mathematical operations
- **Reentrancy Expert**: Analyzes potential reentrancy vectors
- **Code Quality Expert**: Evaluates code standards and best practices

2. Comprehensive Analysis:

- Static code analysis focused on state management
- Logic error identification
- Access control issue assessment
- State management inconsistency checks

4. Findings Summary

ID	Title	Severity	Status
L-01	State Corruption due to Flawed Logic in deleteStaker	**Low**	**Pending Fix**

5. Detailed Findings

[L-01] State Corruption due to Flawed Logic in deleteStaker

Severity: Low

Description The `deleteStaker` function fails to handle cases where the specified staker address is not found in the `stakerList` array. If an invalid address is passed, the function will still decrement the `stakerNumbers` counter without removing any element from `stakerList`. This creates a permanent desynchronization between the counter and the array's length.

Technical Details

```
function deleteStaker(address staker) public onlyStrategyManager {
    uint256 length = stakerList.length;

    for (uint256 i = 0; i < length; i++) {
        if (stakerList[i] == staker) {
            stakerList[i] = stakerList[stakerNumbers - 1];
            stakerList.pop();
            break;
        }
    }
    // Issue: This executes even if staker was not found
    stakerNumbers = stakerNumbers - 1;
}
```

Impact

- **State Inconsistency**: Creates permanent desynchronization between `stakerNumbers` and `stakerList.length`
- **Dependent Functionality Affected**: Other functions relying on accurate staker counts may be impacted
- **Data Integrity**: Compromises contract state integrity
- **Risk Mitigation**: Risk is mitigated by the `onlyStrategyManager` modifier, limiting access to privileged users only

Potential Scenarios

1. Strategy manager mistakenly passes a non-existent staker address

2. In multi-threaded environments, staker may have been deleted by another transaction
3. Frontend application passes incorrect address parameters

Recommendation Refactor the `deleteStaker` function to ensure state variables remain synchronized:

```
function deleteStaker(address staker) public onlyStrategyManager {
    uint256 length = stakerList.length;
    bool found = false;

    for (uint256 i = 0; i < length; i++) {
        if (stakerList[i] == staker) {
            stakerList[i] = stakerList[stakerNumbers - 1];
            stakerList.pop();
            found = true;
            break;
        }
    }

    // Only decrement counter if staker was actually found and removed
    if (found) {
        stakerNumbers = stakerNumbers - 1;
    }
}
```

Alternatively, use a safer approach:

```
function deleteStaker(address staker) public onlyStrategyManager {
    uint256 length = stakerList.length;

    for (uint256 i = 0; i < length; i++) {
        if (stakerList[i] == staker) {
            stakerList[i] = stakerList[length - 1];
            stakerList.pop();
            stakerNumbers = stakerNumbers - 1;
            return; // Return immediately after successful deletion
        }
    }

    // If we reach here, staker was not found
    revert("Staker not found in the list");
}
```

6. Architecture and Design Assessment

Design Strengths

1. **Good Access Control**: Implements strict access control, with almost all important operations restricted to the `cpChainDepositManager` address
2. **Reentrancy Protection**: Uses OpenZeppelin's upgradeable patterns and appropriate security measures
3. **Share Calculation Mechanism**: Implements virtual shares and balance offsets to prevent zero-value boundary issues
4. **Event Logging**: Appropriate event emissions facilitate off-chain monitoring

Key Architectural Issues

1. **High Centralization**: Almost all important operations depend on a single address (cpChainDepositManager)
2. **State Management Flaws**: The `deleteStaker` function contains logical flaws
3. **Trust Assumptions**: Users must highly trust the security of the `cpChainDepositManager` address

Systemic Risk Assessment

This contract serves as the base staking pool with the following characteristics:

1. **Centralized Trust Model**: Highly dependent on the correctness of the strategy manager role
2. **State Consistency Risk**: Although impact is limited, state inconsistencies may cause long-term issues
3. **Single Point of Failure**: The security of the `cpChainDepositManager` address is critical

7. Conclusion

This security audit identified **one low-severity issue** in the `CpChainBase` contract.

Key Findings Summary:

- **State Management Flaw**: Logical flaw in the `deleteStaker` function may cause state inconsistency

- **Good Access Control**: Contract properly implements access control, effectively mitigating security risks
- **Centralized Design**: Contract adopts centralized design around trusted strategy manager

Overall Risk Rating: Low

While the identified issue has low severity and is limited by privileged access, we recommend fixing it to ensure perfect state consistency. The contract's access control model is robust and effectively mitigates risks like reentrancy and unauthorized access.

Priority Remediation Recommendations:

1. **Medium Priority**: [L-01] Fix the logical flaw in the `deleteStaker` function
2. **Long-term Improvement**: Consider implementing additional state consistency checks

We recommend the development team address the identified low-severity issue as a precautionary measure to ensure perfect state consistency.