Comprehensive Audit Report: TetuPawnShop.sol

Contract Name: TetuPawnShop.sol
Repository: Tetu Contracts
Audit Date:21 April 2025
Auditor: Aayush Jha (aayushjhaaudits)

---

1. Executive Summary

This audit examines the redeem() function in TetuPawnShop.sol, identifying a critical vulnerability due to improper state management. The issue allows potential collateral manipulation, fund loss, and reentrancy-like exploits despite the nonReentrant modifier.

Key Findings

Severity	Issue	Status
Critical	Improper state change in redeem() before transfers	Fix Required
Medium	Lack of explicit checks on transfer amounts	Recommende d
Low	Event logging could be more detailed	Optional

---

2. Scope of Audit

Focus Areas

- redeem() function logic
- State transition risks
- Reentrancy & front-running vulnerabilities
- Collateral handling security

Exclusions

- Other functions in TetuPawnShop.sol
- External dependencies (e.g., IERC20, nonReentrant)

3. Detailed Findings

3.1 Critical: Premature State Change in redeem()

Description

The _endPosition() function is called before token transfers, violating Checks-Effects-Interactions (CEI) pattern.

Vulnerable Code

solidity
Copy
Download

```
function redeem(uint id) external nonReentrant override {
    // ... checks ...
    _endPosition(pos);  // ❌ State changed before transfers
    uint toSend = _toRedeem(id);
    IERC20(...).safeTransferFrom(...);
    _transferCollateral(...);
    _returnDeposit(id);
}
```

Impact

- Repeated redemptions (if combined with other exploits)
- Collateral theft if state is manipulated mid-execution
- Broken atomicity (partial execution risks)

Proof of Concept (PoC)

An attacker could:

1. Call redeem() in a malicious contract.
2. Exploit the state change before transfers complete.
3. Front-run or re-trigger the function (if nonReentrant is bypassed).

Recommended Fix

solidity

```
function redeem(uint id) external nonReentrant override {
    // ... checks ...
    uint toSend = _toRedeem(id);
    IERC20(...).safeTransferFrom(...);  // ✅ Interactions first
    _transferCollateral(...);
    _returnDeposit(id);
    _endPosition(pos);  // ✅ State change last
}
```

3.2 Medium: Missing Validation on toSend

Issue

No explicit check ensures toSend > 0, risking gas waste or unintended behavior.

Fix

solidity

```
require(toSend > 0, "TPS: Zero redemption amount");
```

3.3 Low: Insufficient Event Data

Issue

The PositionRedeemed event lacks critical details (e.g., toSend amount).

Improvement

solidity

```
emit PositionRedeemed(_msgSender(), id, toSend);
```

---

4. Risk Assessment

Issue	Likelihood	Impact	Severity
Premature _endPosition()	Medium	High	Critical
Missing toSend check	Low	Medium	Medium
Incomplete event logging	Low	Low	Low

---

5. Recommendations

Critical Fixes

✅ Move _endPosition() after transfers to follow CEI pattern.

Enhancements

- 🔹 Add require(toSend > 0) to prevent zero-value redemptions.
- 🔹 Enrich event logs with redemption amounts.

Future Considerations

- 🔸 Fuzz testing for edge cases in redemption logic.
- 🔸 Static analysis to detect similar CEI violations.

---

6. Conclusion

The redeem() function contains a critical vulnerability due to improper state management. Immediate fixes are required to prevent fund loss and collateral manipulation.
Audit Status: Completed
Final Severity Rating: Critical
Recommended Action: Patch before next deployment

Appendices

A. Test Cases

- Verify redeem() fails if _endPosition() is called early.
- Ensure toSend > 0 check rejects invalid redemptions.

B. References

- Consensys CEI Pattern
- SWC-107 Reentrancy

Signed,
Auditor: Aayush Jha
Date:21 April 2025