LayerEdge - Staking Audit Report

Submitted by: @YourHandleHere

Contest: Sherlock Audit Contest #952

Date: May 23, 2025

Summary of Findings

Total issues discovered: **6**

Severity levels: 1 High, 5 Medium

1. [High] Missing Upgrade Authorization Guard

Location: `UUPSUpgradeable` implementation

Description: The `_authorizeUpgrade` function is unprotected. Anyone could upgrade the implementation contract.

Impact: Full contract takeover.

Recommendation: Add `onlyOwner` modifier to `_authorizeUpgrade()`.

---

2. [Medium] ERC20 `transfer()` Return Value Not Checked

Location: Transfers using `IERC20.transfer(...)`

Description: The return value of token transfers is not checked.

Impact: If token fails silently, logic may assume success.

Recommendation: Always check the return value:

```solidity
require(token.transfer(to, amount), "Transfer failed");
```

---

3. [Medium] Unrestricted FenwickTree Updates

Location: `FenwickTree.sol`

Description: Any user may call `update(...)` without access controls.

Impact: Manipulation of staking tiers.

Recommendation: Restrict access or implement validation.

---

4. [Medium] Partial Use of Reentrancy Guard

Location: Functions with external calls (e.g. staking, withdrawing)

Description: Not all functions use `nonReentrant` modifier.

Impact: Potential reentrancy via custom ERC20 tokens.

Recommendation: Add `nonReentrant` to all functions involving external transfers.

---

5. [Medium] Unsafe WETH9 Implementation

Location: `WETH9.sol` fork

Description: The fallback function accepts ETH with no access control or events.

Impact: Hidden ETH deposits; potential misuse.

Recommendation: Use modern OpenZeppelin WETH or add proper controls.

---

6. [Medium] Initializer May Be Called Again

Location: `initialize()` method

Description: Initialization guard not enforced.

Impact: Risk of reinitialization by attacker if not deployed properly.

Recommendation: Use `initializer` modifier and verify proper use of OpenZeppelin's Initializable pattern.
