Smart Contract Audit Report

1. Overview

- **Project Name:**
  BeraFarm

- **Contracts Audited:**

```
contracts/
├── main/
│   ├── access/
│   │   └── AccessController.sol
│   ├── events/
│   │   └── EventEmitter.sol
│   ├── fbgtController/
│   │   ├── FBGTController.sol
│   │   └── FBGTToken.sol
│   ├── fee/
│   │   └── FeeModule.sol
│   ├── oracle/
│   │   └── PriceOracle.sol
│   ├── registry/
│   │   ├── ProtocolRegistry.sol
│   │   └── VaultManagerConfig.sol
│   ├── strategy/
│   │   ├── bearn/
│   │   │   └── StrategyBearnHub.sol
│   │   ├── burrBear/
│   │   │   └── StrategyBurrBear.sol
│   │   ├── dolomite/
│   │   │   └── StrategyDolomite.sol
│   │   ├── euler/
│   │   │   └── StrategyEuler.sol
│   │   ├── hub/
│   │   │   └── StrategyBeraHub.sol
│   │   ├── infrared/
│   │   │   ├── InfraredStrategyDolomite.sol
│   │   │   ├── InfraredStrategyHub.sol
│   │   │   └── InfraredStrategyKodiakIsland.sol
│   │   └── kodiak/
│   │       ├── islands/
│   │       │   └── StrategyKodiakIslands.sol
│   │       └── swap/
│   │           ├── ISwapRouter.sol
│   │           └── KodiakSwapController.sol
│   ├── DataEmitter.sol
│   ├── RewardsDistributor.sol
│   ├── StrategyManager.sol
│   ├── TrancheVault.sol
│   └── TrancheVaultFactory.sol
└── tokens/
    └── TrancheToken.sol
```

- **Code base:** Provided by file
- **Chain:** Berachain
- **Date:** 04/09/2025
- **Commit Hash:** cc5eb62ade35b4fe755ccf1da9c0619e51558b08

2. Executive Summary

This audit aims to assess the security of the smart contracts for the BeraFarm protocol. We identified issues:

- High Risk (1)
- Medium Risk (2)
- Low Risk (4)
- Informational / Gas Optimization Suggestions (1)

All high and medium severity issues have been addressed /partially addressed /not addressed by the development team at the time of publishing this report.

3. Methodology

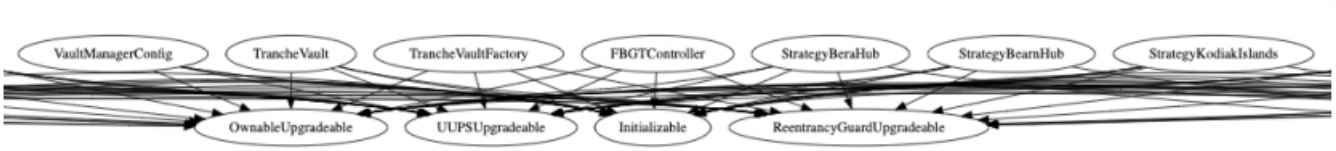
Our audit involved the following steps:

1. **Manual Code Review** – Analyzed logic for correctness, authorization, error handling, etc.
2. **Automated Testing** – Used static analysis tools such as Slither, MythX, or Hardhat plugins.
3. **Gas Usage Review** – Identified areas for optimization.
4. **Dependency Analysis** – Assessed third-party libraries (e.g., OpenZeppelin).

4. Risk Classification

Severity	Description
High	Critical vulnerabilities that can lead to loss.
Medium	Significant issues but not critical.
Low	Minor issues, often best practices.
Informational	No direct risk but useful suggestions.

5. Inheritance Graph



6. Findings

6.1.High Severity

[H-01] Division by Zero in Investment Update After Total Strategy Loss

- **Description:** After 100% strategy loss, the first _updateInvestment() call sets both vault investments to zero. Second call still passes guard condition (totalActualInvestments > 0) but attempts division by zero using totalCurrentInvestments = 0, permanently breaking all vault operations, including deposits and withdrawals.
- **Mitigation:** Add a safety check before division: if (totalCurrentInvestments > 0) around lines 635-636. If zero, skip division and maintain zero values, representing the correct state after total loss.
- File: contracts/main/TrancheVault.sol
- Line: 635-636
- **Status:** Fixed

6.2.Medium Severity

[M-01] Missing checks for slippage parameter

- **Description:** The deposit and withdraw functions accept _slippage and _lpSlippage parameters but never use them in the implementation. There's no range validation (could accept >100% slippage), and no actual slippage protection is provided. This creates a misleading interface where users believe they have slippage protection when they don't.
- **Mitigation:** Add validSlippage(_slippage) modifier for parameter validation or remove unused parameter and update the interface accordingly.
- File: contracts/main/strategy/bearn/StrategyBearnHub.sol
- Add Constants: Line ~41
- Add Modifier: Line ~68
- Functions to Update:
- deposit() - Lines 120-124
- withdraw() - Lines 137-141

- File: contracts/main/strategy/burrBear/StrategyBurrBear.sol
- Add Constants: Line ~45
- Add Modifier: Line ~68
- Functions to Update:
- deposit() - Lines 116-120
- withdraw() - Lines 133-137

- File: contracts/main/strategy/dolomite/StrategyDolomite.sol
- Add Constants: Line ~45
- Add Modifier: Line ~65
- Functions to Update:
- deposit() - Lines 158-162
- withdraw() - Lines 173-177

- File: contracts/main/strategy/euler/StrategyEuler.sol
- Add Constants: Line ~65
- Add Modifier: Line ~93
- Functions to Update:
- deposit() - Lines 142-146
- withdraw() - Lines 157-161

- File: contracts/main/strategy/infrared/InfraredStrategyHub.sol
- Add Constants: Line ~70
- Add Modifier: Line ~100
- Functions to Update:
- deposit() - Lines 149-153
- withdraw() - Lines 166-170

- File: contracts/main/strategy/infrared/InfraredStrategyDolomite.sol
- Add Constants: Line ~160
- Add Modifier: Line ~185
- Functions to Update:
- deposit() - Lines 169-173
- withdraw() - Lines 183-187

- File: contracts/main/strategy/infrared/InfraredStrategyKodiakIsland.sol
- Add Constants: Line ~45
- Add Modifier: Line ~70
- Functions to Update:
- deposit() - Lines 97-101
- withdraw() - Lines ~120-125

- File: contracts/main/strategy/kodiak/islands/StrategyKodiakIslands.sol
- Add Constants: Line ~45
- Add Modifier: Line ~70
- Functions to Update:
- deposit() - Lines 91-95
- withdraw() - Lines ~120-125
- **Status:** Fixed

6.3.Low Severity

[L-01] Floating pragma solidity version.

- **Description:** Adding the constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.
- File: All
- **Status:** Fixed

[L-02] Misleading Function Behaviour - setValidatorInfo Appends Instead of Replacing

Description: The setValidatorInfo function name and documentation suggest it replaces validator configuration, but the implementation only appends new validators to the existing array. This creates operational confusion where owners expect to update validator sets but actually accumulate validators, leading to unintended fund allocations and no cleanup mechanism.

- File: FBGTController.sol
- Line: 310-324
- **Status:** Fixed

[L-03] Inconsistent Error Handling

Description: Uses require(baseIndex != type(uint256).max, "BASE not in pool") instead of custom errors like the rest of the contract. This creates inconsistency in error handling patterns, higher gas costs for users, and deviates from established coding standards throughout the codebase.

Mitigation: Replace with custom error: if (baseIndex == type(uint256).max) revert ErrorLibrary.TokenNotInPool(); to maintain consistency and reduce gas costs.

- File:StrategyBeraborrowHub.sol
- Line: 309
- **Status:** Fixed

[L-04] Incorrect Token Amount Passed to Internal Withdrawal Function

Description: The withdraw function passes _params.strategyTokenAmount (vault tokens) to _withdraw() instead of the actual island LP token balance received from vault redemption. This causes incomplete withdrawals when vault exchange rates differ from 1:1, potentially leaving tokens stuck in the contract.

Mitigation: To mitigate this, replace line 233 with _withdraw(balance, _slippage, _lpSlippage); to use the actual island token balance received after vault redemption instead of the original vault token amount.

- File: StrategyBeraborrowHub.sol
- Line: 309
- **Status:** Fixed

6.4. Informational/Optimization Severity

[I-01] Remove unused code

- **Description:** Functions that are not used (dead-code).
- **Status:** Fixed

8. Conclusion

The smart contracts are generally well-written and follow modern best practices. After addressing the findings noted, the contracts should be considered safe for deployment, assuming no changes are made without further auditing.

9. Disclaimer

This audit report is not the final deployed version of the contract. So, any changes made after the audit will be considered out of the scope of the audit, and we will not be responsible for any changes related to that.