Flying Tulip PUT (ftPUT) Audit Report

mkXploit

January 15, 2026

Flying Tulip PUT (ftPUT) Audit Report

Version 0.1

mkXploit

January 16, 2026

Flying Tulip PUT (ftPUT) Audit Report

mkXploit

January 15, 2026

Flying Tulip PUT (ftPUT) Audit Report

Prepared by: mkXploit Lead Auditors:

- [mxXploit] https://github.com/mkXploit

Assisting Auditors:

- None

Table of contents

See table

About mkXploit

mkXploit is a smart contract security auditor specializing in identifying vulnerabilities and strengthening the security posture of decentralized applications.

With deep expertise in Solidity, EVM mechanics, and blockchain security patterns, I conduct comprehensive audits across DeFi protocols, NFT platforms, DAOs, and other Web3 applications.

Disclaimer

The mkXploit team makes all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the the findings provided in this document. A security audit by the team is not an endorsement of the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the solidity implementation of the contracts.

Risk Classification

		Impact		
		High	Medium	Low
	High	H	H/M	M
Likelihood	Medium	H/M	M	M/L
	Low	M	M/L	L

Audit Details

The findings described in this document correspond the following commit hash:

`193074610e69365161610de3163d7955312e7557`

Scope

ftPUT @ 193074610e69365161610de3163d7955312e7557 - ftPUT/contracts/cb/CircuitBreaker.sol - ftPUT/contracts/FlyingTulipOracle.sol - ftPUT/contracts/ftACL.sol - ftPUT/contracts/ftYieldWrapper.sol - ftPUT/contracts/interfaces/IAaveOracle.sol - ftPUT/contracts/interfaces/IAavePoolAddressesProvider.sol - ftPUT/contracts/interfaces/IAavePoolInstance.sol - ftPUT/contracts/interfaces/ICircuitBreaker.sol - ftPUT/contracts/interfaces/IERC20MetadataBurnable.sol - ftPUT/contracts/interfaces/IFlyingTulipOracle.sol - ftPUT/contracts/interfaces/IftACL.sol - ftPUT/contracts/interfaces/IftPut.sol - ftPUT/contracts/interfaces/IftYield.sol - ftPUT/contracts/interfaces/IftYieldWrapper.sol - ftPUT/contracts/interfaces/IStrategy.sol - ftPUT/contracts/pFT.sol - ftPUT/contracts/PutManager.sol - ftPUT/contracts/strategies/AaveStrategy.sol

Audit Details

The findings described in this document correspond the following commit hash:

`22bbbb2c47f3f2b78c1b134590baf41383fd354f`

Protocol Summary

Cash-secured PUT option product where users deposit collateral (USDC, etc.) to receive pFT NFTs that can later be exercised for collateral return or invalidated for FT redemption, while their principal earns yield in strategies (e.g., Aave) that accrues to the treasury.

Roles

PutManager

- **msig**: Admin - pause, upgrade, add collateral, withdraw capital
- **configurator**: Sale control - enable/disable sale, manage FT liquidity, set caps
- **ftACL**: Optional whitelist gating

ftYieldWrapper

- **yieldClaimer / subYieldClaimer**: Deploy/claim yield, maintain strategies
- **strategyManager**: Add/remove/reorder strategies
- **treasury**: Receive yields, confirm strategy changes
- **putManager / depositor**: Deposit/withdraw wrapper shares

pFT

- **Only PutManager**: Mint/burn NFTs

Trust: msig fully trusted (can upgrade), others partially trusted to their specific functions.

Executive Summary

Issues found

Severity	Number of issues found
High	0
Medium	0
Low	3

Severity	Number of issues found
Info	0
Total	3

Findings

[L-1] withdrawUnderlying Burns More Than Value Transferred

Description: The `EthenaSUSDeStrategy::withdrawUnderlying()` function burns `amount` shares but may transfer sUSDe shares worth up to 1 wei less due to rounding.

Impact: Users lose up to 1 wei (~$0.000001) per withdrawal. Violates documented "exact withdrawal" invariant.

Note: This appears intentional per code comments to prevent overpayment, but contradicts documentation.

Recommendation: Either: 1. Update documentation to reflect 1 wei tolerance is intentional

[L-2] Missing Bounds Validation on ftPerUSD Updates

Description: `FlyingTulipOracle::setftPerUSD()` only validates non-zero, lacking min/max bounds.

Impact: Operational risk - msig typo could set incorrect rate: - Setting to 1 instead of 1e8 → users get 1/100M of expected FT - Setting to type(uint64).max → users get excessive FT

Recommendation:

```
uint64 constant MIN_FT_PER_USD = 1e6;  // 0.01 FT per USD
uint64 constant MAX_FT_PER_USD = 1e13; // 100,000 FT per USD

function setftPerUSD(uint64 newFtPerUSD) external onlyMsig {
    if (newFtPerUSD == 0 ||
        newFtPerUSD < MIN_FT_PER_USD ||
        newFtPerUSD > MAX_FT_PER_USD) {
        revert ftOracleError();
    }
    // ...
}
```

[L-3] Unused State Variable in Queue Management

Description: The `head` variable in `EthenaSUSDeStrategy::claimQueued()` is set but never used for validation or logic.

Impact: Dead code, misleading variable name, wasted gas on writes.

Recommendation: Either remove the variable or use it for validation:

```
function claimQueued(uint256 id) external ... {
    if (id >= tail) revert InvalidQueueId();
    // ...
    delete queue[id];
    if (id == head) {
        while (head < tail && queue[head] == address(0)) head++;
    }
}
```
