KittyFi Audit Report

Version 1.0

Owen Lee

August 12, 2024

KittyFi Audit Report

Owen Lee

August 12, 2024

Prepared by: Owen Lee. Lead Auditors: - Owen Lee.

Table of Contents

Protocol Summary

KittyFi is a EUR-pegged stablecoin protocol that enhances user collateral by earning yield through the Aave protocol. Unlike traditional stablecoin models where collateral remains idle, KittyFi actively deposits user collateral into Aave to generate interest. The earned yield is then redistributed equally among all users in the pool, thereby reducing the risk of liquidation. This approach ensures that collateral is not only used to back the stablecoin (KittyCoin) but also provides additional financial security for users.

Disclaimer

I, Owen Lee makes all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for 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

We use the CodeHawks severity matrix to determine severity. See the documentation for more details.

Audit Details

This protocol does not correspond to any commit Hash.:

Scope

```
src/KittyCoin.sol
src/KittyPool.sol
src/KittyVault.sol
```

Roles

- `User` - Performing deposit and withdrawal of collateral along with minting and burning of KittyCoin.
- `Meowntainer` - Responsible for performing executions to supply and withdraw collateral from Aave protocol on KittyVault.

Executive Summary

The audit of KittyFi identified two significant vulnerabilities that could impact user security and protocol stability.Overall, the audit has highlighted critical areas that require prompt remediation to ensure the protocol's reliability and security.

Issues found

Severity	Number of issues found
High	1
Medium	1
Low	0
Info	0
Gas Optimizations	0
Total	2

Findings

High

[H-1] Theft of collateral tokens with fewer than 18 decimals.

Description:

When converting the collateral from USD to EUR, the decimal for the minted `Kitty Coin` is fixed at 18, but the collateral decimal is not taken into account, which may not be 18, such as 8 for WBTC.

As a result, there is a huge error in the calculation of `getUserVaultMeowllateralInEuros` and `getTotalMeowllateralInAave` function, resulting in funds loss for users.

The `getUserVaultMeowllateralInEuros` function:

```

function getUserVaultMeowllateralInEuros(address _user) external view
    returns (uint256) {

        (, int256 collateralToUsdPrice, , , ) = i_priceFeed.
            latestRoundData();

        (, int256 euroPriceFeedAns, , ,) = i_euroPriceFeed.
            latestRoundData();


        uint256 collateralAns = getUserMeowllateral(_user).mulDiv(
            uint256(collateralToUsdPrice) * EXTRA_DECIMALS, PRECISION);
        return collateralAns.mulDiv(uint256(euroPriceFeedAns) *
            EXTRA_DECIMALS, PRECISION);
    }
```

The `getTotalMeowllateralInAave` function:

```

function getTotalMeowllateralInAave() public view returns (uint256) {
        (uint256 totalCollateralBase, , , , , ) = i_aavePool.
            getUserAccountData(address(this));

        (, int256 collateralToUsdPrice, , , ) = i_priceFeed.
            latestRoundData();


        return totalCollateralBase.mulDiv(PRECISION, uint256(
            collateralToUsdPrice) * EXTRA_DECIMALS);
    }
```

Impact:

The decimal of the collateral is not taken into account, which can result in a miscalculation of the result and a huge loss for the user. For example the `getUserVaultMeowllateralInEuros` function undervalues WBTC, and users who deposit 1WBTC but cannot borrow any `KittyCoin` stable coin.

Recommended Mitigation:

Taken into account the collateral decimals, provide a correct implementation of `getUserVaultMeowllateralInEuro` and `getTotalMeowllateralInAave` by not hardcoding it.

Medium

[M-1] Lack of essential stale check in `getUserVaultMeowllateralInEuros()` and `getTotalMeowllateralInAave()`

Description:

On the `KittyVault.sol` in both the `getUserVaultMeowllateralInEuros()` and `getTotalMeowllateralInAa` `()` we are using the `latestRoundData()`, but there is no check if the return value indicates stale data.

```
function getUserVaultMeowllateralInEuros(address _user) external view returns (uint256) {

@>         (, int256 collateralToUsdPrice, , , ) = i_priceFeed.latestRoundData();

@>         (, int256 euroPriceFeedAns, , ,) = i_euroPriceFeed.latestRoundData();

        uint256 collateralAns = getUserMeowllateral(_user).mulDiv(uint256(collateralToUsdPrice) * EXTRA_DECIMALS, PRECISION);
        return collateralAns.mulDiv(uint256(euroPriceFeedAns) * EXTRA_DECIMALS, PRECISION);
    }
```

```
function getTotalMeowllateralInAave() public view returns (uint256) {
        (uint256 totalCollateralBase, , , , , ) = i_aavePool.getUserAccountData(address(this));

@>         (, int256 collateralToUsdPrice, , , ) = i_priceFeed.latestRoundData();


        return totalCollateralBase.mulDiv(PRECISION, uint256(collateralToUsdPrice) * EXTRA_DECIMALS);
    }
```

Impact:

This could lead to stale prices which means that prices would be executed that don’t reflect the current pricing resulting in a potential loss of funds for the user and/or the protocol

Recommended Mitigation:

The `updatedAt` parameter should be returned from `latestRoundData()` and compare it to a staleness threshold.

```

function getUserVaultMeowllateralInEuros(address _user) external view
    returns (uint256) {

-        (, int256 collateralToUsdPrice, , , ) = i_priceFeed.
    latestRoundData();
+        (, int256 collateralToUsdPrice, , uint256 updatedAt,) =
    i_priceFeed.latestRoundData();


-       (, int256 euroPriceFeedAns, , ,) = i_euroPriceFeed.
    latestRoundData();
+       (, int256 euroPriceFeedAns, , uint256 updatedAt,) =
    i_euroPriceFeed.latestRoundData();


+       if (updatedAt < block.timestamp - 60 * 60 ) {
+       revert("stale price feed");
+       }

        uint256 collateralAns = getUserMeowllateral(_user).mulDiv(
            uint256(collateralToUsdPrice) * EXTRA_DECIMALS, PRECISION);
        return collateralAns.mulDiv(uint256(euroPriceFeedAns) *
            EXTRA_DECIMALS, PRECISION);
    }
```

```
function getTotalMeowllateralInAave() public view returns (uint256) {
        (uint256 totalCollateralBase, , , , , ) = i_aavePool.
            getUserAccountData(address(this));

-        (, int256 collateralToUsdPrice, , , ) = i_priceFeed.
    latestRoundData();
+        (, int256 collateralToUsdPrice, , uint256 updatedAt,) =
    i_priceFeed.latestRoundData();


+       if (updatedAt < block.timestamp - 60 * 60 ) {
+       revert("stale price feed");
+       }


```

```
        return totalCollateralBase.mulDiv(PRECISION, uint256(
            collateralToUsdPrice) * EXTRA_DECIMALS);
    }
```