THE ARCADIA GROUP

Audit of the Evolution Finance Contracts

a report of findings

innovative fortuna iuvat

Jan 11th, 2021

Table of Contents

Document Info

Client	Evolution Finance
Title	Smart Contract Audit of Evolution Finance Contracts
Approved By	Rasikh Morani

Contact

For more information on this report, contact us below

Rasikh Morani
rasikh@arcadiamgroup.com
https://t.me/thearcadiagroup

Executive Summary

A Representative Party of Evolution Network ("Evolution") engaged The Arcadia Group ("Arcadia"), a software development, research, and security company, to conduct a review of the following Evolution Finance smart contracts in the provided repo at Commit #bc844785d23dc52b1812ecaacde906db3b90c63d.

Constants.sol
Freezable.sol
EvoToken.sol
Festaked.sol
FestakedRewardContinuation.sol
FestakedWithoutReward.sol
IFestaked.sol
IFestakeRewardManager.sol
IFestakeWithdrawer.sol
MandatoryLocked.sol
OpenEndedRewardManager.sol
RewardDistributor.sol
ConstantRatioReshapableERC20.sol
IncreasingRatioReshapableERC20.sol
ReshapableERC20.sol
LiquidityAdder.sol
Locker.sol
PriceBaseLiquidityUnlockable.sol
StakeDevBurnTaxable.sol
Taxable.sol
Sweepable.sol
SweepBackToLiq.sol

There were 7 issues found, 0 of which were deemed to be ‘critical’, and 1 of which were rated as ‘high’.

The audit is then followed by a second review of the code at commit **#fa38a70f8b1c3216accb51c15f3cbbea432051ab**. All the issues found by the audit were reviewed and discussed between the development team and the auditing team. Additionally at request of Evolution Finance, we would like to verify that in our review of the EvoToken.sol, there was no active mint function, with the _mint function (which can only be called once at deployment) being the only such function.

Arcadia also reviewed the updated code at commit #fa38a70f8b1c3216accb51c15f3cbbea432051ab.

- **Issues EVN-1, EVN-2, EVN-3, EVN-4, and EVN-6 are fixed by the team in the updated code.**
- **Issue EVN-5 is fixed in all files, except that the transferFrom function call in WrappedERC20.sol is still present in the code without checking for deflationary token transfer. However, as discussed with the team, this contract will only be used for wrapping ETH, which should not cause the issue with deflationary token. Thus, this issue is considered as resolved.**
- **Recommendation EVN-7 has been discussed between the auditing team and the development team. This is a purely post-contract deployment issue.**

Severity Rating	Number Of Original Occurrences	Number Of Remaining Occurrences
Critical	0	0
High	1	0
Medium	3	0
Low	0	0
Notice	2	0
Informational	0	0

Arcadia completed the reviews using various methods primarily consisting of dynamic and static analysis. This process included a line by line analysis of the in-scope contracts, optimization analysis, analysis of key functionalities and limiters, and reference against intended functionality.

Findings

1. Deflationary Token in safeAddLiquidity function

- EVN-1
- Severity: Medium
- Likelihood: Medium
- Impact: High

- Target: LiquidityAdder.sol
- Category: Transfer Token
- Finding Type: Dynamic
- Lines: 30-53

Function `safeAddLiquidity` calls safeTransferFrom and transferFrom of arbitrary token without checking the actual token amount received. This is because there are many deflationary tokens where fees are applied on token transfer, causing the actual token amount received is different from the expected amount.

```
token.safeTransferFrom(msg.sender, address(this), tokenAmount);
targetTok.transferFrom(msg.sender, address(this), targetAmount);
```

Action Recommended: Check the actual token amount transferred from the transaction sender to the contract. Here’s a possible solution for transferFrom token.

```
uint256 balBefore = targetTok.balanceOf(address(this));
targetTok.transferFrom(msg.sender, address(this), targetAmount);
uint256 actualReceived = targetTok.balanceOf(address(this)).sub(balBefore);
```

2. Function safeAddLiquidityAndStake will fail

- EVN-2
- Severity: High
- Impact: High

- Target: LiquidityAdder.sol
- Category: Stake
- Finding Type: Dynamic
- Lines: 55-69

Function safeAddLiquidityAndStake of contract LiquidityAdder will fail when calling `stakeFor`. This is because the liquidity token ownership belongs to the `to` parameter address.

```
function safeAddLiquidityAndStake(
        IFestaked stake,
        IERC20 token,
        uint256 tokenAmount,
        uint256 targetAmount,
        uint256 minTokenAmount,
        uint256 minTargetAmount,
        address to,
        uint256 deadline) public
    returns (bool) {
        require(stake.getToken() == address(targetToken), "LA: Stake and token don't match");
        (uint256 tau, uint256 tou, uint256 liq) = safeAddLiquidity(token, tokenAmount, targetAmount,
minTokenAmount,
            minTargetAmount, to, deadline);
        return stake.stakeFor(msg.sender, liq);
    }
```

Action Recommended: The function should work as follows:

- Call `safeAddLiquidity` with making liquidity token ownership belonging to the contract

```
(uint256 tau, uint256 tou, uint256 liq) = safeAddLiquidity(token,
tokenAmount, targetAmount, minTokenAmount, minTargetAmount,
address(this), deadline);
```

- Approve allowing for stake to transfer LP token to from the contract
- call `stakeFor` function

3. **Local parameter current in addNewPool function is not used**

- EVN-3
- Severity: Notice
- Impact: None

- Target: RewardDistributor.sol
- Category: Unused parameters
- Finding Type: Dynamic
- Line 21

Local parameter `current` is unused and should be removed from the code.

```
    RewardPools memory current = rewardPools;
```

Action Recommended: Remove the local parameter.

4. Ensure no overflow in function updateRewardDistributionForPools

- EVN-4
- Severity: Medium
- Likelihood: Low
- Impact: Medium

- Target: Invest.sol
- Category: Math
- Finding Type: Dynamic
- Lines: 46

Function updateRewardDistributionForPools could have potential overflow in compute the sum of ratios.

```
function updateRewardDistributionForPools(bytes calldata poolRatios)  onlyOwner()
   external returns (bool) {
       uint sum = 0;
       uint len = rewardReceivers.length;
       for (uint i = 0; i < len; i++) {
           sum += toUint8(poolRatios, i);
       }
       require(sum == 100, "ReardD: ratios must add to 100");
       rewardPools.pools = poolRatios;
       return true;
   }
```

5. **Deflationary Token**

- EVN-5
- Severity: Medium
- Likelihood: Medium
- Impact: High

- Target: ReshapableERC20.sol,WrappedERC20.sol,FeStaked.sol
- Category: Deflationary token transfer
- Finding Type: Dynamic

All transferFrom tokens do not take deflationary tokens into account.

```
        IERC20(token).safeTransferFrom(from, address(this), amount);


        wrappedToken.safeTransferFrom(msg.sender, address(this), _amount);


        IERC20(token).safeTransferFrom(allower, receiver, amount);
```

Action Recommended:

- Use the solution described in Issue EVN-1 to check the actual receive token amount.

6. Function addReward can remove parameter withdrawableAmount

- EVN-6
- Severity: Notice
- Likelihood: High
- Impact: None

- Target: OpenEndedRewardManager.sol
- Category: Informational
- Finding Type: Dynamic
- Lines: 66

Function addReward(uint256 rewardAmount, uint256 withdrawableAmount) of contract `OpenEndedRewardManager` can remove parameter withdrawableAmount as it must be always 0.

```
function addReward(uint256 rewardAmount, uint256 withdrawableAmount)
   external override returns (bool) {
       require(withdrawableAmount == 0, "OERM: withdrawableAmount > 0 not supported");
       require(rewardAmount != 0, "OERM: rewardAmount not supported");
       rewardToken.safeTransferFrom(msg.sender, address(this), rewardAmount);
       _addMarginalReward();
   }
```

Action Recommended: Remove the redundant parameter

Recommendations

7. **Contract owner for all contracts should be a multisignature wallet**

- PLZ-7
- Severity: Recommendation
- Impact: Recommendation

- Target:All
- Category: Recommendation
- Finding Type: Dynamic

This issue is related to the post contract deployment and operation. The ownership of all contracts after deployment and configuration should be transferred to a multisignature wallet or a governance wallet. This is to ensure that no one has the super power to change anything maliciously to the contract without being approved by others.

Conclusion

Arcadia identified some issues that occurred at git hash #bc844785d23dc52b1812ecaacde906db3b90c63d and reviewed the updated code at git hash with the remediation at **#fa38a70f8b1c3216accb51c15f3cbbea432051ab.**

Severity Rating	**Number Of Original Occurrences**	**Number Of Remaining Occurrences**
Critical	0	0
High	1	0
Medium	3	0
Low	0	0
Notice	2	0
Informational	0	0

Disclaimer

While best efforts and precautions have been taken in the preparation of this document, The Arcadia Group and the Authors assume no responsibility for errors, omissions, or damages resulting from the use of the provided information. Additionally, Arcadia would like to emphasize that the use of Arcadia's services does not guarantee the security of a smart contract or set of smart contracts and does not guarantee against attacks. One audit on its own is not enough for a project to be considered secure; that categorization can only be earned through extensive peer review and battle testing over an extended period.