0x Guard

Smart contracts security assessment

Final report

Tariff: Standard

Defender Finance Pools

December 2022

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for **Defender Finance Pools**.

The Genesis contract allows users to farm tokens in different pools.

The ShareTokenRewardPoolV2 contract allows users to farm shareTokens at a different speed for 18 months. The ShareTokenRewardPoolV2 contract may charge a fee of up to 1% for each deposit.

The code is available at the GitHub repository and was audited after the commit 63b83d0a992e815ee0f5ab2f0b07f2c0b698969c.

The inspected contracts are **Genesis.sol**, **ShareTokenRewardPoolV2.sol**.

Report Update.

The contract's code was updated according to this report and rechecked after the commit 3c8129d36f20d25cf9a1b60f4f265284dc5ac526.

Name	Defender Finance Pools
Audit date	2022-12-16 - 2022-12-16
Language	Solidity
Platform	Binance Smart Chain

Contracts checked

Name	Address
Genesis	
ShareTokenRewardPoolV2	

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

- Manually analyze smart contracts for security vulnerabilities
- Smart contracts' logic check

Known vulnerabilities checked

Title	Check result
Unencrypted Private Data On-Chain	passed
Code With No Effects	passed
Message call with hardcoded gas amount	passed
Typographical Error	passed
DoS With Block Gas Limit	passed
Presence of unused variables	passed
Incorrect Inheritance Order	passed
Requirement Violation	passed
Weak Sources of Randomness from Chain Attributes	passed
Shadowing State Variables	passed

Incorrect Constructor Name	passed
Block values as a proxy for time	passed
Authorization through tx.origin	passed
DoS with Failed Call	passed
Delegatecall to Untrusted Callee	passed
Use of Deprecated Solidity Functions	passed
Assert Violation	passed
State Variable Default Visibility	passed
Reentrancy	passed
Unprotected SELFDESTRUCT Instruction	passed
Unprotected Ether Withdrawal	passed
Unchecked Call Return Value	passed
Floating Pragma	passed
Outdated Compiler Version	passed
Integer Overflow and Underflow	passed
Function Default Visibility	passed

Classification of issue severity

High severity	High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.
Medium severity	Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.

Low severity Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

1. Blocking when calculating rewards (ShareTokenRewardPoolV2)
Status: Fixed

1. Rewards are calculated and updated by iterating over all reward periods (months). To do this, the period is first calculated, and then iteration is carried out over it.

When iterating over the last periods, the array goes beyond the boundaries, which will fail and block the execution of the function.

This behavior is possible due to two factors. First, while iterating, the loop may go beyond the array due to the '`i <= toMonth`' condition. Secondly, at each iteration step, the next element of the array is read `rewardInfos[i + 1].startTime`.

```
    function getDaoReward(uint256 _fromTime, uint256 _toTime) internal view returns
(uint256) {
        uint256 fromMonth = getMonthFrom(_fromTime);
        uint256 toMonth = getMonthFrom(_toTime);
        uint256 reward = 0;
        for (uint256 i = fromMonth; i <= toMonth; ++i) {
            uint256 timeFrom = _fromTime;
            uint256 timeTo = rewardInfos[i + 1].startTime > _toTime ? _toTime :
rewardInfos[i + 1].startTime;
            reward = reward +
timeTo.sub(timeFrom).mul(rewardInfos[i].rewardPerSecondForDao);
            _fromTime = timeTo;
        }

        return reward;
    }
```

The ussie occurs in functions getDaoReward(), getDevReward(), getUserReward(), updatePool().

2. Also, the getMonth() function can return a value of 19 or more for the last period (if the user claims a reward after the end of the reward period).

```
    function getMonth() public view returns (uint256) {
        if (block.timestamp < poolStartTime) return 0;
        return (block.timestamp - poolStartTime) / MONTH;
    }
```

In this case, a similar problem will happen in the functions deposit() (L401-L404) and withdraw() (L446-L449).

```
    function deposit(uint256 _pid, uint256 _amount) external {
        ...
        uint256 fromMonth = getMonthFrom(lastRewardTime);
        uint256 toMonth = getMonth();
        for (uint256 i = fromMonth; i <= toMonth; ++i) {
            user.rewardDebt[i] =
 user.amount.mul(pool.accRewardTokenPerShare[i]).div(1e18)
        }
        ...
    }
```

Recommendation: It is necessary to fix iteration over arrays.

Take into account that the total number of periods is 18, but the iteration starts from zero index. So the last element of the array has index 17.

Also, consider adding an emergencyWithdraw() function to withdraw the user's tokens in case of failure.

Medium severity issues

No issues were found

Low severity issues

1. Gas optimization (ShareTokenRewardPoolV2)

Status: Fixed

1. Variables `poolStartTime` and `poolEndTime` can be declared as `immutable` to save gas.

2. The `getUserReward()` function calculates rewards for all specified periods. The local variable `tokenSupply` is used to calculate accumulated rewards for each period. Since the variable `tokenSupply` does not change during the execution of the function, it is enough to define it only once, outside the loop. We recommend moving the L301 out from the for-loop.

3. The visibility of the functions `add()`, `set()` and `setDepositFeePercent()` can be declared `external` instead of `public` to save gas.

Conclusion

Defender Finance Pools Genesis, ShareTokenRewardPoolV2 contracts were audited. 1 high, 1 low severity issues were found.
1 high, 1 low severity issues have been fixed in the update.

We recommend writing tests to cover the founded issues.

Disclaimer

Guard