RUGDOG

SMART CONTRACT SECURITY AUDIT

Final report

Plan: Complex

MasterChefV2

April 2022

rugdog.net

the@rugdog.net

CONTENTS

INTRODUCTION

PancakeSwap MasterChef v2 is a new main staking contract for Farms while providing more flexibility for adjusting the $CAKE emissions, including CAKE pool, burn and other PancakeSwap products.

Name	MasterChefV2
Audit date	2022-04-29 - 2022-04-29
Language	Solidity
Network	Binance Smart Chain

CONTRACTS CHECKED

Name	Address
MasterChef	0xa5f8c5dbd5f286960b9d90548680ae5ebff07652

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

ATTACKS 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Outdated Compiler Version	✓ passed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 ISSUES

High severity	Issues leading to assets theft, locking or any other loss of assets or leading to contract malfunctioning.
Medium severity	Issues that can trigger a contract failure of malfunctioning.
Low severity	Issues that do now affect contract functionality. For example,

Low severity unoptimised gas usage, outdated or unused code.

ISSUES

High severity issues

1. Lack of input parameters validation (MasterChef)

The constructor lacks non-zero validation for the _MASTER_CHEF and _burnAdmin parameters.

```
constructor(
        IMasterChef _MASTER_CHEF,
        IBEP20 _CAKE,
        uint256 _MASTER_PID,
        address _burnAdmin
    ) public {
        MASTER_CHEF = _MASTER_CHEF;
        CAKE = _CAKE;
        MASTER_PID = _MASTER_PID;
        burnAdmin = _burnAdmin;
    }
```

Medium severity issues

1. Mass update pools may run of gas (MasterChef)

The function massUpdatePools() may run out of gas if a big number of pools are added.

```
/// @notice Update cake reward for all the active pools. Be careful of gas spending!
    function massUpdatePools() public {
        uint256 length = poolInfo.length;
        for (uint256 pid = 0; pid < length; ++pid) {
            PoolInfo memory pool = poolInfo[pid];
            if (pool.allocPoint != 0) {
                updatePool(pid);
            }
        }
    }
```

Low severity issues

1. Gas optimisations (MasterChef)

poolLength() can be declared external an external functions use less gas than public ones.

CONCLUSION

MasterChefV2 MasterChef contract was audited. 1 high, 1 medium, 1 low severity issues were found.

DISCLAIMER

STATIC ANALYSIS OUTPUT

INFO:Detectors:

MasterChefV2.pendingCake(uint256,address) (contracts/MasterChefV2.sol#234-251) performs a multiplication on the result of a division:

-boostedAmount = user.amount.mul(getBoostMultiplier(_user,_pid)).div(BOOST_PRECISION) (contracts/MasterChefV2.sol#249)

-boostedAmount.mul(accCakePerShare).div(ACC_CAKE_PRECISION).sub(user.rewardDebt) (contracts/MasterChefV2.sol#250)

MasterChefV2.pendingCake(uint256,address) (contracts/MasterChefV2.sol#234-251) performs a multiplication on the result of a division:

-accCakePerShare =
accCakePerShare.add(cakeReward.mul(ACC_CAKE_PRECISION).div(lpSupply)) (contracts/MasterChefV2.sol#246)

-cakeReward =
multiplier.mul(cakePerBlock(pool.isRegular)).mul(pool.allocPoint).div((totalRegularAllocPoint)) (contracts/MasterChefV2.sol#243-245)

MasterChefV2.updatePool(uint256) (contracts/MasterChefV2.sol#282-299) performs a multiplication on the result of a division:

-cakeReward =
multiplier.mul(cakePerBlock(pool.isRegular)).mul(pool.allocPoint).div(totalAllocPoint) (contracts/MasterChefV2.sol#290-292)

-pool.accCakePerShare =

pool.accCakePerShare.add((cakeReward.mul(ACC_CAKE_PRECISION).div(lpSupply))) (contracts/MasterChefV2.sol#293)

MasterChefV2.deposit(uint256,uint256) (contracts/MasterChefV2.sol#304-335) performs a multiplication on the result of a division:

-user.rewardDebt = user.amount.mul(multiplier).div(BOOST_PRECISION).mul(pool.accCakePerShare).div(ACC_CAKE_PRECISION) (contracts/MasterChefV2.sol#329-331)

MasterChefV2.withdraw(uint256,uint256) (contracts/MasterChefV2.sol#340-363) performs a multiplication on the result of a division:

-user.rewardDebt = user.amount.mul(multiplier).div(BOOST_PRECISION).mul(pool.accCakePerShare).div(ACC_CAKE_PRECISION) (contracts/MasterChefV2.sol#355-357)

MasterChefV2.updateBoostMultiplier(address,uint256,uint256) (contracts/MasterChefV2.sol#470-498) performs a multiplication on the result of a division:

-user.rewardDebt = user.amount.mul(_newMultiplier).div(BOOST_PRECISION).mul(pool.accCakePerShare).div(ACC_CAKE_PRECISION) (contracts/MasterChefV2.sol#488-490)

MasterChefV2.settlePendingCake(address,uint256,uint256) (contracts/MasterChefV2.sol#512-524) performs a multiplication on the result of a division:

-boostedAmount = user.amount.mul(_boostMultiplier).div(BOOST_PRECISION) (contracts/MasterChefV2.sol#519)

-accCake = boostedAmount.mul(poolInfo[_pid].accCakePerShare).div(ACC_CAKE_PRECISION) (contracts/MasterChefV2.sol#520)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

INFO:Detectors:

MasterChefV2.add(uint256,IBEP20,bool,bool) (contracts/MasterChefV2.sol#172-204) contains a tautology or contradiction:

 - require(bool,string)(_lpToken.balanceOf(address(this)) >= 0,None BEP20 tokens) (contracts/MasterChefV2.sol#178)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-contradiction

INFO:Detectors:

MasterChefV2.init(IBEP20) (contracts/MasterChefV2.sol#149-158) ignores return value by dummyToken.approve(address(MASTER_CHEF),balance) (contracts/MasterChefV2.sol#153)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

INFO:Detectors:

MasterChefV2.constructor(IMasterChef,IBEP20,uint256,address)._burnAdmin (contracts/MasterChefV2.sol#129) lacks a zero-check on :

- burnAdmin = _burnAdmin (contracts/MasterChefV2.sol#134)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

INFO:Detectors:

Reentrancy in MasterChefV2.init(IBEP20) (contracts/MasterChefV2.sol#149-158):

External calls:

- dummyToken.safeTransferFrom(msg.sender,address(this),balance) (contracts/MasterChefV2.sol#152)

- dummyToken.approve(address(MASTER_CHEF),balance) (contracts/MasterChefV2.sol#153)

- MASTER_CHEF.deposit(MASTER_PID,balance) (contracts/MasterChefV2.sol#154)

State variables written after the call(s):

- lastBurnedBlock = block.number (contracts/MasterChefV2.sol#156)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

INFO:Detectors:

Reentrancy in MasterChefV2.emergencyWithdraw(uint256) (contracts/MasterChefV2.sol#372-385):

External calls:

- lpToken[_pid].safeTransfer(msg.sender,amount) (contracts/MasterChefV2.sol#383)

Event emitted after the call(s):

- EmergencyWithdraw(msg.sender,_pid,amount) (contracts/MasterChefV2.sol#384)

Reentrancy in MasterChefV2.init(IBEP20) (contracts/MasterChefV2.sol#149-158):

External calls:

- dummyToken.safeTransferFrom(msg.sender,address(this),balance) (contracts/MasterChefV2.sol#152)

- dummyToken.approve(address(MASTER_CHEF),balance) (contracts/MasterChefV2.sol#153)

- MASTER_CHEF.deposit(MASTER_PID,balance) (contracts/MasterChefV2.sol#154)

Event emitted after the call(s):

- Init() (contracts/MasterChefV2.sol#157)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

INFO:Detectors:

SafeBEP20.safeApprove(IBEP20,address,uint256) (contracts/SafeBEP20.sol#51-68) is never used and should be removed

SafeBEP20.safeDecreaseAllowance(IBEP20,address,uint256) (contracts/SafeBEP20.sol#82-96) is never used and should be removed

SafeBEP20.safeIncreaseAllowance(IBEP20,address,uint256) (contracts/SafeBEP20.sol#70-80) is never used and should be removed

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

INFO:Detectors:

Pragma version^0.6.0 (contracts/SafeBEP20.sol#2) allows old versions

Pragma version>=0.4.0 (contracts/interfaces/IBEP20.sol#3) allows old versions

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

INFO:Detectors:

Parameter MasterChefV2.add(uint256,IBEP20,bool,bool)._allocPoint (contracts/MasterChefV2.sol#173) is not in mixedCase

Parameter MasterChefV2.add(uint256,IBEP20,bool,bool)._lpToken (contracts/MasterChefV2.sol#174) is not in mixedCase

Parameter MasterChefV2.add(uint256,IBEP20,bool,bool)._isRegular (contracts/MasterChefV2.sol#175) is not in mixedCase

Parameter MasterChefV2.add(uint256,IBEP20,bool,bool)._withUpdate (contracts/MasterChefV2.sol#176) is not in mixedCase

Parameter MasterChefV2.set(uint256,uint256,bool)._pid (contracts/MasterChefV2.sol#211) is not in mixedCase

Parameter MasterChefV2.set(uint256,uint256,bool)._allocPoint (contracts/MasterChefV2.sol#212) is not in mixedCase

Parameter MasterChefV2.set(uint256,uint256,bool)._withUpdate (contracts/MasterChefV2.sol#213) is not in mixedCase

Parameter MasterChefV2.pendingCake(uint256,address)._pid (contracts/MasterChefV2.sol#234) is not in mixedCase

Parameter MasterChefV2.pendingCake(uint256,address)._user (contracts/MasterChefV2.sol#234) is not in mixedCase

Parameter MasterChefV2.cakePerBlock(bool)._isRegular (contracts/MasterChefV2.sol#266) is not in mixedCase

Parameter MasterChefV2.updatePool(uint256)._pid (contracts/MasterChefV2.sol#282) is not in mixedCase

Parameter MasterChefV2.deposit(uint256,uint256)._pid (contracts/MasterChefV2.sol#304) is not in mixedCase

Parameter MasterChefV2.deposit(uint256,uint256)._amount (contracts/MasterChefV2.sol#304) is not in mixedCase

Parameter MasterChefV2.withdraw(uint256,uint256)._pid (contracts/MasterChefV2.sol#340) is not in mixedCase

Parameter MasterChefV2.withdraw(uint256,uint256)._amount (contracts/MasterChefV2.sol#340) is not in mixedCase

Parameter MasterChefV2.emergencyWithdraw(uint256)._pid (contracts/MasterChefV2.sol#372) is not in mixedCase

Parameter MasterChefV2.burnCake(bool)._withUpdate (contracts/MasterChefV2.sol#389) is not in mixedCase

Parameter MasterChefV2.updateCakeRate(uint256,uint256,uint256,bool)._burnRate (contracts/MasterChefV2.sol#408) is not in mixedCase

Parameter MasterChefV2.updateCakeRate(uint256,uint256,uint256,bool)._regularFarmRate (contracts/MasterChefV2.sol#409) is not in mixedCase

Parameter MasterChefV2.updateCakeRate(uint256,uint256,uint256,bool)._specialFarmRate (contracts/MasterChefV2.sol#410) is not in mixedCase

Parameter MasterChefV2.updateCakeRate(uint256,uint256,uint256,bool)._withUpdate (contracts/MasterChefV2.sol#411) is not in mixedCase

Parameter MasterChefV2.updateBurnAdmin(address)._newAdmin (contracts/

MasterChefV2.sol#436) is not in mixedCase

Parameter MasterChefV2.updateWhiteList(address,bool)._user (contracts/MasterChefV2.sol#447) is not in mixedCase

Parameter MasterChefV2.updateWhiteList(address,bool)._isValid (contracts/MasterChefV2.sol#447) is not in mixedCase

Parameter MasterChefV2.updateBoostContract(address)._newBoostContract (contracts/MasterChefV2.sol#456) is not in mixedCase

Parameter MasterChefV2.updateBoostMultiplier(address,uint256,uint256)._user (contracts/MasterChefV2.sol#471) is not in mixedCase

Parameter MasterChefV2.updateBoostMultiplier(address,uint256,uint256)._pid (contracts/MasterChefV2.sol#472) is not in mixedCase

Parameter MasterChefV2.updateBoostMultiplier(address,uint256,uint256)._newMultiplier (contracts/MasterChefV2.sol#473) is not in mixedCase

Parameter MasterChefV2.getBoostMultiplier(address,uint256)._user (contracts/MasterChefV2.sol#503) is not in mixedCase

Parameter MasterChefV2.getBoostMultiplier(address,uint256)._pid (contracts/MasterChefV2.sol#503) is not in mixedCase

Parameter MasterChefV2.settlePendingCake(address,uint256,uint256)._user (contracts/MasterChefV2.sol#513) is not in mixedCase

Parameter MasterChefV2.settlePendingCake(address,uint256,uint256)._pid (contracts/MasterChefV2.sol#514) is not in mixedCase

Parameter MasterChefV2.settlePendingCake(address,uint256,uint256)._boostMultiplier (contracts/MasterChefV2.sol#515) is not in mixedCase

Variable MasterChefV2.MASTER_CHEF (contracts/MasterChefV2.sol#62) is not in mixedCase

Variable MasterChefV2.CAKE (contracts/MasterChefV2.sol#64) is not in mixedCase

Variable MasterChefV2.MASTER_PID (contracts/MasterChefV2.sol#82) is not in mixedCase

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

INFO:Detectors:

MasterChefV2.slitherConstructorVariables() (contracts/MasterChefV2.sol#18-543) uses literals with too many digits:

- cakeRateToBurn = 643750000000 (contracts/MasterChefV2.sol#99)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

INFO:Detectors:

poolLength() should be declared external:

- MasterChefV2.poolLength() (contracts/MasterChefV2.sol#161-163)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

INFO:Slither:. analyzed (9 contracts with 75 detectors), 56 result(s) found

RUGDOG

WOOF!

rugdog.net

the@rugdog.net