Security Assessment

Macaron

Jun 7th, 2021

Table of Contents

MCV-14 : Potential Reentrancy
MCV-15 : Variable Naming Convention
MCV-16 : Typos in Comments and Functions
MFC-01 : Compiler Error
MRC-01 : Ignored return values
MTC-01 : Delegation Power Not Moved Along with `transfer()`
MTC-02 : Incorrect Delegation Flow
MTC-03 : Timestamp Dependence
SPC-01 : Missing Emit Events
SPC-02 : Ignored return values
SPC-03 : Withdrawals to `controller` v.s. to `governance`
SPC-04 : Unnecessary `pid`
SPC-05 : Privileged Ownership
SPC-06 : Misleading implementation of `harvest()`
SPC-07 : Misleading parameter of `deposit()`
SPC-08 : No guarantee of `cakeSyrupToken` will be transferred back
SPC-09 : `deposit()` function doesn't deposit any real `baseToken`
SPC-10 : Gas Optimization on `pancakeRouter.swapExactTokensForTokens()`
SPC-11 : Potential Reentrancy
SPL-01 : Missing Emit Events
SPL-02 : Ignored return values
SPL-03 : Wrong Address Check for Ownership Transfer
SPL-04 : Withdrawals to `controller` v.s. to `governance`
SPL-05 : Privileged Ownership
SPL-06 : Misleading parameter of `deposit()`
SPL-07 : `deposit()` function doesn't deposit any real `baseToken`
SPL-08 : `withdraw()` doesn't burn `magicBoxToken`
SPL-09 : Potential Reentrancy
SPL-10 : Typos in Comments and Functions

Appendix

Disclaimer

About

Summary

This report has been prepared for Macaron smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis, and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Macaron
Platform	BSC
Language	Solidity
Codebase	https://github.com/macaronswap/macaron-contracts
Commits	<undefined>

Audit Summary

Delivery Date	Jun 07, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

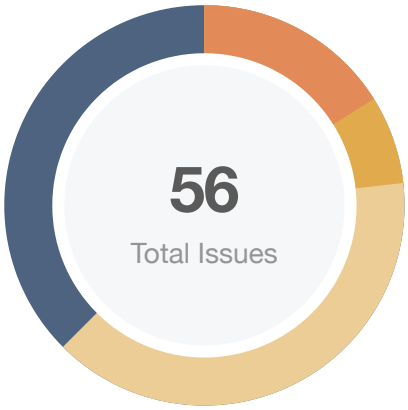
Vulnerability Summary

Total Issues	56
Critical	0
Major	9
Medium	4
Minor	22
Informational	21
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
CCC	ChocoChef.sol	12616e98c3efda5c94d7ecddaa98ccb693c8a833cccbaba203081851a244ee26
CFC	ChocoFall.sol	ca4e49b48c735b99c6a523c676ab25663e5a0c9cf594bb5290066b416edb6372
MFC	MacaronFactory.sol	f8561bced687870c111cb2ef4d5d728c1b80df7fd6a2f9c492b5fb799a04f39f
MRC	MacaronRouter.sol	28c8edfaae1dd6a75209f6a5b749cfdbe7c0ba0b420f8518b11ccdf59d6b2d94
MTC	MacaronToken.sol	2965b73146ec8215edc568d25fddcf13667e361dfbcf6cbfa987507f9f89f1e1
MBC	MagicBox.sol	797ab5892a263e73330db0e3aaae8bbf54eabc1bb2103ef0d493c3330e295118
MCV	MasterChefV2.sol	b91328e167a70e2819f8038480860381742cedf2c1daeb60b540eb9cfaf274f7
SPC	StrategyPancakeCake.sol	fb4c44d4e240db279202b15130552f10e7365ff07c24b2a25b6d3c371cf9ec29
SPL	StrategyPancakeCakeLP.sol	fc7e24f1e11a6172a2def7b00a181594cac01effa15023b28e1a00447a0be60e

Findings



Critical	**0** (0.00%)
Major	**9** (16.07%)
Medium	**4** (7.14%)
Minor	**22** (39.29%)
Informational	**21** (37.50%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
MCRN-01	`public` Functions Could Be Declared `external`	Gas Optimization	Informational	Resolved
MCRN-02	Zero Address Validation	Volatile Code	Informational	Resolved
CCC-01	Missing Reset Logic of `emergencyRewardWithdraw()`	Volatile Code	Minor	Partially Resolved
CCC-02	Missing Sanity Checks in `constructor()`	Coding Style	Informational	Resolved
CCC-03	Imprecise Arithmetic Operations Order	Mathematical Operations	Informational	Acknowledged
CCC-04	Potential Reentrancy	Volatile Code	Minor	Acknowledged
CFC-01	Delegation Power Not Moved Along with `transfer()`	Logical Issue	Major	Partially Resolved
CFC-02	Incorrect Delegation Flow	Logical Issue	Major	Resolved
CFC-03	Ignored return values	Logical Issue	Minor	Acknowledged
CFC-04	Timestamp Dependence	Volatile Code	Informational	Acknowledged
CFC-05	Inexistent Delegate Transfer	Logical Issue	Minor	Acknowledged
CFC-06	Typos in Comments and Functions	Coding Style	Informational	Acknowledged
MBC-01	Incorrect Delegation Flow	Logical Issue	Major	Resolved
MBC-02	Timestamp Dependence	Volatile Code	Informational	Acknowledged

ID	Title	Category	Severity	Status
MCV-01	Missing Emit Events	Gas Optimization	Informational	Acknowledged
MCV-02	Recommended Explicit Pool Validity Checks	Logical Issue	Informational	Resolved
MCV-03	Incompatibility With Deflationary Tokens	Logical Issue	Minor	Acknowledged
MCV-04	Over Minted Token	Logical Issue	Minor	Acknowledged
MCV-05	Syrup(Choco) Burn Issue	Logical Issue	Medium	Resolved
MCV-06	Centralized Control of Several State Variables	Logical Issue	Informational	Acknowledged
MCV-07	add() Function Not Restricted	Volatile Code	Major	Partially Resolved
MCV-08	Assignment Optimization	Gas Optimization	Informational	Acknowledged
MCV-09	Ignored return values	Logical Issue	Minor	Acknowledged
MCV-10	Wrong Address Check for Ownership Transfer	Volatile Code	Medium	Resolved
MCV-11	Risky Migrator Functionality	Logical Issue	Major	Acknowledged
MCV-12	Timestamp Dependence	Volatile Code	Informational	Acknowledged
MCV-13	Imprecise Arithmetic Operations Order	Mathematical Operations	Informational	Acknowledged
MCV-14	Potential Reentrancy	Volatile Code	Minor	Acknowledged
MCV-15	Variable Naming Convention	Coding Style	Informational	Acknowledged
MCV-16	Typos in Comments and Functions	Coding Style	Informational	Acknowledged
MFC-01	Compiler Error	Language Specific	Medium	Partially Resolved
MRC-01	Ignored return values	Logical Issue	Minor	Acknowledged
MTC-01	Delegation Power Not Moved Along with `transfer()`	Logical Issue	Major	Partially Resolved
MTC-02	Incorrect Delegation Flow	Logical Issue	Major	Resolved

ID	Title	Category	Severity	Status
MTC-03	Timestamp Dependence	Volatile Code	Informational	Acknowledged
SPC-01	Missing Emit Events	Gas Optimization	Informational	Acknowledged
SPC-02	Ignored return values	Logical Issue	Minor	Acknowledged
SPC-03	Withdrawals to `controller` v.s. to `governance`	**Centralization / Privilege, Volatile Code**	**Major**	**Partially Resolved**
SPC-04	Unnecessary `pid`	Volatile Code	Informational	Acknowledged
SPC-05	Privileged Ownership	**Centralization / Privilege**	**Minor**	**Partially Resolved**
SPC-06	Misleading implementation of `harvest()`	Logical Issue	Minor	Resolved
SPC-07	Misleading parameter of `deposit()`	Logical Issue	Minor	Partially Resolved
SPC-08	No guarantee of `cakeSyrupToken` will be transferred back	Logical Issue	Minor	Partially Resolved
SPC-09	`deposit()` function doesn't deposit any real `baseToken`	Logical Issue	Minor	Partially Resolved
SPC-10	Gas Optimization on `pancakeRouter.swapExactTokensForTokens()`	Gas Optimization	Informational	Resolved
SPC-11	Potential Reentrancy	Volatile Code	Minor	Acknowledged
SPL-01	Missing Emit Events	Gas Optimization	Informational	Acknowledged
SPL-02	Ignored return values	Logical Issue	Minor	Acknowledged
SPL-03	Wrong Address Check for Ownership Transfer	Volatile Code	Medium	Resolved
SPL-04	Withdrawals to `controller` v.s. to `governance`	**Centralization / Privilege, Volatile Code**	**Major**	**Partially Resolved**
SPL-05	Privileged Ownership	**Centralization / Privilege**	**Minor**	**Partially Resolved**

ID	Title	Category	Severity	Status
SPL-06	Misleading parameter of `deposit()`	Logical Issue	Minor	Partially Resolved
SPL-07	`deposit()` function doesn't deposit any real `baseToken`	Logical Issue	Minor	Partially Resolved
SPL-08	`withdraw()` doesn't burn `magicBoxToken`	Logical Issue	Minor	Partially Resolved
SPL-09	Potential Reentrancy	Volatile Code	Minor	Acknowledged
SPL-10	Typos in Comments and Functions	Coding Style	Informational	Acknowledged

MCRN-01 | `public` Functions Could Be Declared `external`

Category	Severity	Location	Status
Gas Optimization	Informational	Global	Resolved

Description

Functions listed here are never used in other contracts. Declaring functions as `external` could help save gas. ChocoChef.sol:

- `Ownable.renounceOwnership()` (ChocoChef.sol#89-92)
- `Ownable.transferOwnership(address)` (ChocoChef.sol#98-100)
- `ChocoChef.stopReward()` (ChocoChef.sol#768-770)
- `ChocoChef.setRewardEndBlock(uint256)` (ChocoChef.sol#772-774)
- `ChocoChef.setRewardPerBlock(uint256)` (ChocoChef.sol#776-779)
- `ChocoChef.deposit(uint256)` (ChocoChef.sol#833-856)
- `ChocoChef.withdraw(uint256)` (ChocoChef.sol#859-884)
- `ChocoChef.emergencyWithdraw()` (ChocoChef.sol#887-894)
- `ChocoChef.emergencyRewardWithdraw(uint256)` (ChocoChef.sol#897-900)
- `ChocoChef._unstakeAll()` (ChocoChef.sol#950-961)
- `ChocoChef.rewardDistribution(address)` (ChocoChef.sol#963-965)

ChocoFall.sol:

- `Ownable.renounceOwnership()` (ChocoFall.sol#92-95)
- `Ownable.transferOwnership(address)` (ChocoFall.sol#101-103)
- `MacaronToken.mint(address,uint256)` (ChocoFall.sol#871-874)
- `ChocoFall.mint(address,uint256)` (ChocoFall.sol#1110-1113)
- `ChocoFall.burn(address,uint256)` (ChocoFall.sol#1115-1118)
- `ChocoFall.safeMacaronTransfer(address,uint256)` (ChocoFall.sol#1131-1138)

MacaronRouter.sol:

- `MacaronRouter.quote(uint256,uint256,uint256)` (MacaronRouter.sol#750-752)
- `MacaronRouter.getAmountOut(uint256,uint256,uint256)` (MacaronRouter.sol#754-762)
- `MacaronRouter.getAmountIn(uint256,uint256,uint256)` (MacaronRouter.sol#764-772)
- `MacaronRouter.getAmountsOut(uint256,address[])` (MacaronRouter.sol#774-782)
- `MacaronRouter.getAmountsIn(uint256,address[])` (MacaronRouter.sol#784-792)

MagicBox.sol:

- `MagicBox.mint(address,uint256)` (MagicBox.sol#871-874)
- `MagicBox.burn(address,uint256)` (MagicBox.sol#876-879)

MasterChefV2.sol:

- `Ownable.renounceOwnership()` (MasterChefV2.sol#629-632)
- `Ownable.transferOwnership(address)` (MasterChefV2.sol#638-640)
- `MacaronToken.mint(address,uint256)` (MasterChefV2.sol#966-969)
- `ChocoFall.mint(address,uint256)` (MasterChefV2.sol#1205-1208)
- `ChocoFall.burn(address,uint256)` (MasterChefV2.sol#1210-1213)
- `ChocoFall.safeMacaronTransfer(address,uint256)` (MasterChefV2.sol#1226-1233)
- `MasterChef.updateMultiplier(uint256)` (MasterChefV2.sol#1579-1581)
- `MasterChef.add(uint256,IBEP20,bool,bool,ICakeStrategy,IBEP20)` (MasterChefV2.sol#1589-1610)
- `MasterChef.set(uint256,uint256,bool)` (MasterChefV2.sol#1613-1623)
- `MasterChef.setCakeStrategy(uint256,ICakeStrategy)` (MasterChefV2.sol#1639-1641)
- `MasterChef.setMacaronPerBlock(uint256)` (MasterChefV2.sol#1643-1646)
- `MasterChef.setMacaronPoolRewardRatio(uint256)` (MasterChefV2.sol#1648-1651)
- `MasterChef.setMigrator(IMigratorChef)` (MasterChefV2.sol#1654-1656)
- `MasterChef.migrate(uint256)` (MasterChefV2.sol#1659-1675)
- `MasterChef.deposit(uint256,uint256)` (MasterChefV2.sol#1735-1760)
- `MasterChef.withdraw(uint256,uint256)` (MasterChefV2.sol#1763-1788)
- `MasterChef.enterStaking(uint256)` (MasterChefV2.sol#1791-1809)
- `MasterChef.leaveStaking(uint256)` (MasterChefV2.sol#1812-1829)
- `MasterChef.emergencyWithdraw(uint256)` (MasterChefV2.sol#1832-1839)
- `MasterChef.dev(address)` (MasterChefV2.sol#1847-1850)
- `MasterChef.transferMacaronOwnership(address)` (MasterChefV2.sol#1853-1856)
- `MasterChef.transferChocoOwnership(address)` (MasterChefV2.sol#1859-1862)

StrategyPancakeCake.sol:

- `StrategyBase.setPancakeRouterPath(address,address,address[])` (StrategyPancakeCake.sol#813-819)
- `StrategyBase.claimReward()` (StrategyPancakeCake.sol#875)
- `StrategyPancakeCake.claimReward()` (StrategyPancakeCake.sol#1085-1087)
- `StrategyBase.balanceOf()` (StrategyPancakeCake.sol#897-899)
- `StrategyBase.setFarmingToken(address)` (StrategyPancakeCake.sol#922-924)

- `StrategyBase.setApproveRouterForToken(address,uint256)` (StrategyPancakeCake.sol#926-928)
- `StrategyBase.executeTransaction(address,uint256,string,bytes)` (StrategyPancakeCake.sol#935-958)
- `StrategyPancakeCake.initialize(address,address,address,address,address,address,address)` (StrategyPancakeCake.sol#999-1020)
- `StrategyPancakeCake.deposit(uint256)` (StrategyPancakeCake.sol#1026-1051)

StrategyPancakeCakeLP.sol:

- `StrategyBase.claimReward()` (StrategyPancakeCakeLP.sol#687)
- `StrategyPancakeCLP.claimReward()` (StrategyPancakeCakeLP.sol#839-841)
- `StrategyBase.balanceOf()` (StrategyPancakeCakeLP.sol#691-693)
- `StrategyBase.setrewardToken(address)` (StrategyPancakeCakeLP.sol#712-714)
- `StrategyBase.executeTransaction(address,uint256,string,bytes)` (StrategyPancakeCakeLP.sol#721-744)
- `StrategyPancakeCLP.initialize(address,uint256,address,address,address,address)` (StrategyPancakeCakeLP.sol#766-783)
- `StrategyPancakeCLP.deposit(uint256)` (StrategyPancakeCakeLP.sol#789-802)
- `StrategyPancakeCLP.balanceOfPoolPending()` (StrategyPancakeCakeLP.sol#848-850)
- `StrategyPancakeCLP.transferMagicBoxOwnership(address)` (StrategyPancakeCakeLP.sol#876-879)

Alleviation

Fixed in commit hash `bc931e1e568bdded3440922aea804f2ad68f3e1c`.

MCRN-02 | Zero Address Validation

Category	Severity	Location	Status
Volatile Code	Informational	Global	Resolved

Description

The listed functions are missing zero address validation when critical addresses are initialized or set.
MacaronRouter.sol:

- `MacaronRouter.constructor(address,address)._factory`
- `MacaronRouter.constructor(address,address)._WETH`

MasterChefV2.sol:

- `MasterChef.constructor(MacaronToken,ChocoFall,address,uint256,uint256)._devaddr`
- `MasterChef.dev(address)._devaddr`

StrategyPancakeCake.sol:

- `StrategyBase.setGovernance(address)._governance`
- `StrategyBase.setController(address)._controller`
- `StrategyBase.setTimelock(address)._timelock`
- `StrategyBase.setFarmingToken(address)._farmingToken`
- `StrategyBase.executeTransaction(address,uint256,string,bytes).target`
- `StrategyPancakeCake.initialize(address,address,address,address,address,address,address)._cakeMasterChef`
- `StrategyPancakeCake.initialize(address,address,address,address,address,address,address)._cakeSyrupToken`
- `StrategyPancakeCake.initialize(address,address,address,address,address,address,address)._macaron`
- `StrategyPancakeCake.setCakeMasterChefContract(address)._cakeMasterChef`

StrategyPancakeCakeLP.sol:

- `StrategyBase.setGovernance(address)._governance`
- `StrategyBase.setController(address)._controller`
- `StrategyBase.setTimelock(address)._timelock`
- `StrategyBase.setrewardToken(address)._rewardToken`

- `StrategyBase.executeTransaction(address,uint256,string,bytes).target`
- `StrategyPancakeCLP.initialize(address,uint256,address,address,address,address)._cakeMasterChef`
- `StrategyPancakeCLP.setCakeMasterChefContract(address)._cakeMasterChef`

Recommendation

Recommend applying `require` statements to make sure the critical state variables are not set to `address(0)`.

Alleviation

Fixed in commit hash `bc931e1e568bdded3440922aea804f2ad68f3e1c`.

CCC-01 | Missing Reset Logic of `emergencyRewardWithdraw()`

Category	Severity	Location	Status
Volatile Code	Minor	ChocoChef.sol: 897~898	Partially Resolved

Description

As in the function `emergencyRewardWithdraw()`, the reward token is transferred to `msg.sender`, the variables `rewardDebt` and `amount` for current `user` in the `userPool` are not updated.

Alleviation

[Macaron Team]: emergencyRewardWithdraw method only callable from owner for withdraw left reward after pool not using anymore. So user states don't need to update.

CCC-02 | Missing Sanity Checks in `constructor()`

Category	Severity	Location	Status
Coding Style	Informational	ChocoChef.sol: 759~760	Resolved

Description

In `constructor()` of contract ChocoChef, if `_isCLP` is true, the address of `_cakeChef` would be approved for max amount of `_stakingToken` and if `_isMaster` is also true, it would be approved for max `_syrup` token. However, the validity of the `_cakeChef` is not checked like how it is checked in `withdraw()/deposit()` of MasterChefV2.

In addition, the other state variables with typeof `address` are not checked for being non-zero addresses.

Alleviation

Fixed in commit hash `bc931e1e568bdded3440922aea804f2ad68f3e1c`.

CCC-03 | Imprecise Arithmetic Operations Order

Category	Severity	Location	Status
Mathematical Operations	Informational	ChocoChef.sol: 800~802, 818~820	Acknowledged

Description

In functions listed above in the locations section, there are divisions before multiplications, which would lead to precision loss.

Recommendation

Recommend reordering the math calculations to avoid precision loss.

CCC-04 | Potential Reentrancy

Category	Severity	Location	Status
Volatile Code	Minor	ChocoChef.sol: 833~857(ChocoChef.deposit()), 859~885(ChocoChef.withdraw()), 887~895(ChocoChef.emergencyWithdraw())	Acknowledged

Description

Functions listed in the locations section are risky to reentrancy attack, as there are state variable updated after external calls.

Recommendation

Recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts. Also, it is important making sure all the contracts behind interfaces are with valid addresses and being trusted.

CFC-01 | Delegation Power Not Moved Along with `transfer()`

Category	Severity	Location	Status
Logical Issue	Major	ChocoFall.sol: 661~662	Partially Resolved

Description

The voting power of delegation is not moved from token sender to token recipient along with the `transfer()`. Current `transfer()` is from `BEP20` protocol and don't invoke `_moveDelegates()`.

Recommendation

Recommend considering adopting a specific implementation of the standard that has a `_moveDelegates()` logic called upon transferring.

references: https://github.com/yam-finance/yam-protocol/blob/master/contracts/token/YAM.sol#L108

Alleviation

[Macaron Team]: This token should not transfer other address because it is proof of stake token. We do not want tokens to be collected and affect the voting results.

CFC-02 | Incorrect Delegation Flow

Category	Severity	Location	Status
Logical Issue	Major	ChocoFall.sol: 1117~1118(ChocoFall)	Resolved

Description

Whenever new tokens are minted, new delegates are moved from the zero address to the recipient of the minting process. However, whenever tokens are burned, new delegates are once again moved from the zero address to the recipient whereas delegates should be moved on the opposite way.

Recommendation

Recommend swapping the `address(0)` and `_from` to alleviate this issue. At its current state, it breaks the delegate mechanism and can also lead to a user being unable to mint/burn tokens in case the upper limit of a `uint256` is reached due to the SafeMath utilization in function `_moveDelegates()`.

Alleviation

Fixed in commit hash `bc931e1e568bdded3440922aea804f2ad68f3e1c`.

CFC-03 | Ignored return values

Category	Severity	Location	Status
Logical Issue	Minor	ChocoFall.sol: 1134~1135, 1136~1137	Acknowledged

Description

ChocoFall.sol:

- `ChocoFall.safeMacaronTransfer(address,uint256)` ignores return value by `macaron.transfer(_to,macaronBal)`
- `ChocoFall.safeMacaronTransfer(address,uint256)` ignores return value by `macaron.transfer(_to,_amount)`

MacaronRouter.sol:

- `MacaronRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256)` ignores return value by `IMacaronFactory(factory).createPair(tokenA,tokenB)`
- `MacaronRouter.removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)` ignores return value by `IMacaronPair(pair).transferFrom(msg.sender,pair,liquidity)`

MasterChefV2.sol:

- `ChocoFall.safeMacaronTransfer(address,uint256)` ignores return value by `macaron.transfer(_to,macaronBal)`
- `ChocoFall.safeMacaronTransfer(address,uint256)` ignores return value by `macaron.transfer(_to,_amount)`
- `MasterChef.add(uint256,IBEP20,bool,bool,ICakeStrategy,IBEP20)` ignores return value by `IBEP20(_lpToken).approve(address(_cakeStrategy),type()(uint256).max)`

StrategyPancakeCake.sol:

- `StrategyBase._swapTokens(address,address,uint256)` ignores return value by `pancakeRouter.swapExactTokensForTokens(_amount,0,path,address(this),now.add(1800))`
- `StrategyPancakeCake.deposit(uint256)` ignores return value by `IERC20(cakeSyrupToken).transfer(controller,_amount)`
- `StrategyPancakeCake._buyMacaronAndBurn(uint256,uint256)` ignores return value by `IERC20(macaron).transfer(address(0x000000000000000000000000000000000000dEaD),_macaronBal)`

- `StrategyPancakeCake.retireStrat()` ignores return value by `IERC20(baseToken).transfer(address(governance),baseBal)`

StrategyPancakeCakeLP.sol:

- `StrategyPancakeCLP.retireStrat()` ignores return value by `IERC20(clpToken).transfer(address(governance),baseBal)`

Recommendation

We recommend to check every function return values.

CFC-04 | Timestamp Dependence

Category	Severity	Location	Status
Volatile Code	Informational	ChocoFall.sol: 980~981, 1244~1245, 1244~1245	Acknowledged

Description

Functions listed in the locations section contain `require` statements using `block.timestamp`. Note the block time on testnet and mainnet BSC are different. Please understand the security risk level and trade-off of using `block.timestamp` or alias `now` as one of core factors in the contract.

Recommendation

Correct use of 15-second rule to minimize the impact caused by timestamp variance

CFC-05 | Inexistent Delegate Transfer

Category	Severity	Location	Status
Logical Issue	Minor	ChocoFall.sol: 1134, 1136	Acknowledged

Description

The transfer and transferFrom functions of the YAM project transfer delegates as well via overridence. The MacaronSwap implementation does not, leading to an inconsistency in the delegates of each address .

Recommendation

We advise that the transfer and transferFrom functions are properly overriden to also transfer delegates on each invocation from the sender of the funds to the recipient.

CFC-06 | Typos in Comments and Functions

Category	Severity	Location	Status
Coding Style	Informational	ChocoFall.sol: 712~713	Acknowledged

Description

MasterChefV2:

- Typo in comment body `points`
- Typo in comment body `multiplier`

StrategyPancakeCakeLP:

- Typo in comment body `governance`

ChochChef:

- Typo in comment body `points`
- Function name `setRewardToken` with wrong capital initials

Recommendation

We advise that the comment text and function name is corrected.

MBC-01 | Incorrect Delegation Flow

Category	Severity	Location	Status
Logical Issue	Major	MagicBox.sol: 878~879(MagicBox)	Resolved

Description

Whenever new tokens are minted, new delegates are moved from the zero address to the recipient of the minting process. However, whenever tokens are burned, new delegates are once again moved from the zero address to the recipient whereas delegates should be moved on the opposite way.

Recommendation

Recommend swapping the `address(0)` and `_from` to alleviate this issue. At its current state, it breaks the delegate mechanism and can also lead to a user being unable to mint/burn tokens in case the upper limit of a `uint256` is reached due to the SafeMath utilization in function `_moveDelegates()`.

Alleviation

Fixed in commit hash `bc931e1e568bdded3440922aea804f2ad68f3e1c`.

MBC-02 | Timestamp Dependence

Category	Severity	Location	Status
Volatile Code	Informational	MagicBox.sol: 979	Acknowledged

Description

Functions listed in the locations section contain `require` statements using `block.timestamp`. Note the block time on testnet and mainnet BSC are different. Please understand the security risk level and trade-off of using `block.timestamp` or alias `now` as one of core factors in the contract.

Recommendation

Correct use of 15-second rule to minimize the impact caused by timestamp variance

MCV-01 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	Informational	MasterChefV2.sol: 1654, 1579, 1847, 1639, 1648, 1643	Acknowledged

Description

Function that affect the status of sensitive variables should be able to emit events as notifications to customers. MasterChefV2.sol:

- `dev()`
- `setCakeStrategy()`
- `setMacaronPerBlock()`
- `setMacaronPoolRewardRatio()`
- `setMigrator()`
- `updateMultiplier()`

StrategyPancakeCake.sol:

- `StrategyBase.setGovernance()`
- `StrategyBase.setController()`
- `StrategyPancakeCake.setBuyBurnPercent()`

StrategyPancakeCakeLP.sol:

- `StrategyBase.setGovernance()`
- `StrategyBase.setController()`

Recommendation

Consider adding events for sensitive actions, and emit it in the function. For example,

```
event SetDev(address indexed user, address indexed _devaddr);
...
function dev(address _devaddr) public {
    ...
    devaddr = _devaddr;
    ...
    emit SetDev(msg.sender, _devaddr);
}
```

MCV-02 | Recommended Explicit Pool Validity Checks

Category	Severity	Location	Status
Logical Issue	Informational	MasterChefV2.sol: 1493	Resolved

Description

There's no sanity check to validate if a pool is existing. The current implementation simply relies on the implicit, compiler-generated bound-checks of arrays to ensure the pool index stays within the array range [0, poolInfo.length-1]. However, considering the importance of validating given pools and their numerous occasions, a better alternative is to make explicit the sanity checks by introducing a new modifier.

Recommendation

Apply necessary sanity checks to ensure the given `_pid` is legitimate by adding a new modifier validatePool to functions `set()`, `migrate()`, `deposit()`, `withdraw()`, `emergencyWithdraw()`, `pendingMacaron()` and `updatePool()`.

```
1 modifier validatePoolByPid(uint256 _pid) {
2     require (_pid < poolInfo . length , "Pool does not exist") ;
3     _;
4 }
```

Alleviation

Fixed in commit hash `bc931e1e568bdded3440922aea804f2ad68f3e1c`.

MCV-03 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Logical Issue	Minor	MasterChefV2.sol: 1762	Acknowledged

Description

The MasterChef contract operates as the main entry for interaction with staking users. The staking users deposit LP tokens into the MacaronSwap pool and in return get a proportionate share of the pool's rewards. Later on, the staking users can withdraw their assets from the pool. In this procedure, `deposit()` and `withdraw()` are involved in transferring users' assets into (or out of) the MacaronSwap protocol. When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged (and burned) transaction fee. As a result, this may not meet the assumption behind these low-level asset-transferring routines and will bring unexpected balance inconsistencies.

Recommendation

Regulate the set of LP tokens supported in Macaron and, if there is a need to support deflationary tokens, add necessary mitigation mechanisms to keep track of accurate balances.

MCV-04 | Over Minted Token

Category	Severity	Location	Status
Logical Issue	Minor	MasterChefV2.sol: 1728~1729	Acknowledged

Description

`updatePool()` function minted 100% + 10% (dev fee is included as 10% of the 100%) of total rewards.

Recommendation

We advise the client to fix to mint 100% of the block reward instead of 100% + 10%.

MCV-05 | Syrup(Choco) Burn Issue

Category	Severity	Location	Status
Logical Issue	Medium	MasterChefV2.sol: 1831	Resolved

Description

An exploit in the interaction between the `MasterChef` contract and the `ChocoFall` contract was abused by bad actors. Previously when `Macaron` was staked, an equal amount of `Choco` tokens would be minted. Once the `Macaron` was unstaked and withdrawn, the `Choco` tokens would be burned. The specific exploit here was that if a user used the `emergencyWithdraw()` function in the `MasterChef` contract to withdraw their staked `Macaron`, the corresponding `Choco` tokens would not be burnt as intended. This allowed bad actors to repeatedly mint more `Choco` tokens with their `Macaron` tokens.

Recommendation

Consider to make changes as following in `emergencyWithdraw()` function.

```
function emergencyWithdraw(uint256 _pid) public {
    PoolInfo storage pool = poolInfo[_pid];
    UserInfo storage user = userInfo[_pid][msg.sender];
    if(_pid == 0) {
        choco.burn(msg.sender, user.amount);
    }
    pool.lpToken.safeTransfer(address(msg.sender), user.amount);
    emit EmergencyWithdraw(msg.sender, _pid, user.amount);
    user.amount = 0;
    user.rewardDebt = 0;
}
```

Alleviation

Fixed in commit hash `bc931e1e568bdded3440922aea804f2ad68f3e1c`.

MCV-06 | Centralized Control of Several State Variables

Category	Severity	Location	Status
Logical Issue	Informational	MasterChefV2.sol: 1580~1581, 1639, 1643, 1648, 1654	Acknowledged

Description

There are several essential state variables limited by modifier `onlyOwner` in functions `updateMultiplier()`, `setMigrator()`, `setCakeStrategy()`, `setMacaronPerBlock()` and `setMacaronPoolRewardRatio()`. Need to paying extra attention to avoid the abuse of the privileged ownership, and thus avoid some critical changes without obtaining the consensus of the community. For example, in function `getMultiplier()`, it can alter the `BONUS_MULTIPLIER` variable and consequently the output of which is directly utilized for the minting of new macaron tokens.

Recommendation

Renounce ownership when it is the right timing, or gradually migrate to a timelock plus multisig governing procedure and let the community monitor in respect of transparency considerations.

MCV-07 | add() Function Not Restricted

Category	Severity	Location	Status
Volatile Code	Major	MasterChefV2.sol: 1589~1590	Partially Resolved

Description

The comment in line L122, mentioned // XXX DO NOT add the same LP token more than once. Rewards will be messed up if you do.

The total amount of reward eggReward in function `updatePool()` will be incorrectly calculated if the same LP token is added into the pool more than once in function `add()`.

However, the code does not reflect as the comment behaviors as there isn't any valid restriction on preventing this issue.

The current implementation is relying on the trust of the owner to avoid repeatedly adding same LP token to the pool, as the function will only be called by the owner.

Recommendation

Detect whether the given pool for addition is a duplicate of an existing pool. The pool addition is only successful when there is no duplicate. Using mapping of `addresses` -> `booleans`, which can restrict the same address being added twice.

Alleviation

[Macaron Team]: This is an autocompound contract so it should be run like that. We create this contract for live masterchef contract. This update needs two contracts changes. We will implement this future like you suggest.

MCV-08 | Assignment Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	MasterChefV2.sol: 1617~1618	Acknowledged

Description

The linked statement will only yield a different output stored to `totalAllocPoint` only if the condition of L1620 yields true .

Recommendation

As a result of the above, it is more optimal to move the assignment of L1617 to the if block of L1620.

MCV-09 | Ignored return values

Category	Severity	Location	Status
Logical Issue	Minor	MasterChefV2.sol: 1229~1230, 1231~1232, 1606~1607	Acknowledged

Description

ChocoFall.sol:

- `ChocoFall.safeMacaronTransfer(address,uint256)` ignores return value by `macaron.transfer(_to,macaronBal)`
- `ChocoFall.safeMacaronTransfer(address,uint256)` ignores return value by `macaron.transfer(_to,_amount)`

MacaronRouter.sol:

- `MacaronRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256)` ignores return value by `IMacaronFactory(factory).createPair(tokenA,tokenB)`
- `MacaronRouter.removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)` ignores return value by `IMacaronPair(pair).transferFrom(msg.sender,pair,liquidity)`

MasterChefV2.sol:

- `ChocoFall.safeMacaronTransfer(address,uint256)` ignores return value by `macaron.transfer(_to,macaronBal)`
- `ChocoFall.safeMacaronTransfer(address,uint256)` ignores return value by `macaron.transfer(_to,_amount)`
- `MasterChef.add(uint256,IBEP20,bool,bool,ICakeStrategy,IBEP20)` ignores return value by `IBEP20(_lpToken).approve(address(_cakeStrategy),type()(uint256).max)`

StrategyPancakeCake.sol:

- `StrategyBase._swapTokens(address,address,uint256)` ignores return value by `pancakeRouter.swapExactTokensForTokens(_amount,0,path,address(this),now.add(1800))`
- `StrategyPancakeCake.deposit(uint256)` ignores return value by `IERC20(cakeSyrupToken).transfer(controller,_amount)`
- `StrategyPancakeCake._buyMacaronAndBurn(uint256,uint256)` ignores return value by `IERC20(macaron).transfer(address(0x000000000000000000000000000000000000dEaD),_macaronBal)`

- `StrategyPancakeCake.retireStrat()` ignores return value by `IERC20(baseToken).transfer(address(governance),baseBal)`

StrategyPancakeCakeLP.sol:

- `StrategyPancakeCLP.retireStrat()` ignores return value by `IERC20(clpToken).transfer(address(governance),baseBal)`

Recommendation

We recommend to check every function return values.

MCV-10 | Wrong Address Check for Ownership Transfer

Category	Severity	Location	Status
Volatile Code	Medium	MasterChefV2.sol: 1854~1855, 1859~1860	Resolved

Description

Function `transferMacaronOwnership()` and `transferChocoOwnership()` in MasterChefV2 and `transferMagicBoxOwnership()` in StrategyPancakeCakeLP should have the `require` statement should check `newOwner != address(0)`, instead of `msg.sender`.

Alleviation

Fixed in commit hash `bc931e1e568bdded3440922aea804f2ad68f3e1c`.

MCV-11 | Risky Migrator Functionality

Category	Severity	Location	Status
Logical Issue	Major	MasterChefV2.sol: 1659~1660	Acknowledged

Description

`setMigrator()` function can set migrator contract to any contract that is implemented from `IMigratorChef` interface by owner. As result, the invocation of `migrator.migrate()` in function migrate() may bring dangerous effects as it is unknown to the user. However, the project may lose the ability to upgrade and migrate if `setMigrator()` and `migrate()` are removed. In addition, the migrator functionality itself is risky.

Recommendation

Recommend showing more transparency to the community and external users on how to prevent abusage of the migrate functionality.

Alleviation

[Macaron Team]: We will publish migrator contract as public on github.

MCV-12 | Timestamp Dependence

Category	Severity	Location	Status
Volatile Code	Informational	MasterChefV2.sol: 1075~1076, 1339~1340	Acknowledged

Description

Functions listed in the locations section contain `require` statements using `block.timestamp`. Note the block time on testnet and mainnet BSC are different. Please understand the security risk level and trade-off of using `block.timestamp` or alias `now` as one of core factors in the contract.

Recommendation

Correct use of 15-second rule to minimize the impact caused by timestamp variance

MCV-13 | Imprecise Arithmetic Operations Order

Category	Severity	Location	Status
Mathematical Operations	Informational	MasterChefV2.sol: 1695~1697, 1727~1731	Acknowledged

Description

In functions listed above in the locations section, there are divisions before multiplications, which would lead to precision loss.

Recommendation

Recommend reordering the math calculations to avoid precision loss.

MCV-14 | Potential Reentrancy

Category	Severity	Location	Status
Volatile Code	Minor	MasterChefV2.sol: 1589~1611(MasterChef.add()), 1735~1761(MasterChef.deposit()), 1832~1840(MasterChef.emergencyWithdraw()), 1791~1810(MasterChef.enterStaking()), 1812~1830(MasterChef.leaveStaking()), 1659~1676(MasterChef.migrate()), 1613~1624(MasterChef.set()), 1711~1733(MasterChef.updatePool()), 1763~1789(MasterChef.withdraw())	Acknowledged

Description

Functions listed in the locations section are risky to reentrancy attack, as there are state variable updated after external calls.

Recommendation

Recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts. Also, it is important making sure all the contracts behind interfaces are with valid addresses and being trusted.

MCV-15 | Variable Naming Convention

Category	Severity	Location	Status
Coding Style	Informational	MasterChefV2.sol: 1534	Acknowledged

Description

The linked variables do not conform to the standard naming convention of Solidity whereby functions and variable names utilize the format unless variables are declared as constant in which case they utilize the format.

Recommendation

We advise that the naming conventions utilized by the linked statements are adjusted to reflect the correct type of declaration according to the Solidity style guide.

MCV-16 | Typos in Comments and Functions

Category	Severity	Location	Status
Coding Style	Informational	MasterChefV2.sol: 1541	Acknowledged

Description

MasterChefV2:

- Typo in comment body `points`
- Typo in comment body `multiplier`

StrategyPancakeCakeLP:

- Typo in comment body `governance`

ChochChef:

- Typo in comment body `points`
- Function name `setRewardToken` with wrong capital initials

Recommendation

We advise that the comment text and function name is corrected.

MFC-01 | Compiler Error

Category	Severity	Location	Status
Language Specific	Medium	MacaronFactory.sol: 141~142	Partially Resolved

Description

`MacaronFactory.sol` is not able to be successfully compiled. In function `constructor()` of contract `MacaronERC20` of `MacaronFactory.sol`, the `chainId := chainid` should actually be `chainId := chainid()`. In addition, the `override` keywords are required for overriding public state variables, according to Solidity Documentation.

Recommendation

Recommend resolving the compiler errors and thoroughly test the contracts with test cases and testnet deployments.

Alleviation

The `chainId()` issue was fixed in commit hash `bc931e1e568bdded3440922aea804f2ad68f3e1c`, but there are still compiler errors reported on development environment.

MRC-01 | Ignored return values

Category	Severity	Location	Status
Logical Issue	Minor	MacaronRouter.sol: 390~391, 460~461	Acknowledged

Description

ChocoFall.sol:

- `ChocoFall.safeMacaronTransfer(address,uint256)` ignores return value by `macaron.transfer(_to,macaronBal)`
- `ChocoFall.safeMacaronTransfer(address,uint256)` ignores return value by `macaron.transfer(_to,_amount)`

MacaronRouter.sol:

- `MacaronRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256)` ignores return value by `IMacaronFactory(factory).createPair(tokenA,tokenB)`
- `MacaronRouter.removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)` ignores return value by `IMacaronPair(pair).transferFrom(msg.sender,pair,liquidity)`

MasterChefV2.sol:

- `ChocoFall.safeMacaronTransfer(address,uint256)` ignores return value by `macaron.transfer(_to,macaronBal)`
- `ChocoFall.safeMacaronTransfer(address,uint256)` ignores return value by `macaron.transfer(_to,_amount)`
- `MasterChef.add(uint256,IBEP20,bool,bool,ICakeStrategy,IBEP20)` ignores return value by `IBEP20(_lpToken).approve(address(_cakeStrategy),type()(uint256).max)`

StrategyPancakeCake.sol:

- `StrategyBase._swapTokens(address,address,uint256)` ignores return value by `pancakeRouter.swapExactTokensForTokens(_amount,0,path,address(this),now.add(1800))`
- `StrategyPancakeCake.deposit(uint256)` ignores return value by `IERC20(cakeSyrupToken).transfer(controller,_amount)`
- `StrategyPancakeCake._buyMacaronAndBurn(uint256,uint256)` ignores return value by `IERC20(macaron).transfer(address(0x000000000000000000000000000000000000dEaD),_macaronBal)`

- `StrategyPancakeCake.retireStrat()` ignores return value by `IERC20(baseToken).transfer(address(governance),baseBal)`

StrategyPancakeCakeLP.sol:

- `StrategyPancakeCLP.retireStrat()` ignores return value by `IERC20(clpToken).transfer(address(governance),baseBal)`

Recommendation

We recommend to check every function return values.

MTC-01 | Delegation Power Not Moved Along with `transfer()`

Category	Severity	Location	Status
Logical Issue	Major	MacaronToken.sol: 661~662	Partially Resolved

Description

The voting power of delegation is not moved from token sender to token recipient along with the `transfer()`. Current `transfer()` is from `BEP20` protocol and don't invoke `_moveDelegates()`.

Recommendation

Recommend considering adopting a specific implementation of the standard that has a `_moveDelegates()` logic called upon transferring.

references: https://github.com/yam-finance/yam-protocol/blob/master/contracts/token/YAM.sol#L108

Alleviation

[Macaron Team]: This token should not transfer other address because it is proof of stake token. We do not want tokens to be collected and affect the voting results.

MTC-02 | Incorrect Delegation Flow

Category	Severity	Location	Status
Logical Issue	Major	MacaronToken.sol: 1212~1213(ChocoFall)	Resolved

Description

Whenever new tokens are minted, new delegates are moved from the zero address to the recipient of the minting process. However, whenever tokens are burned, new delegates are once again moved from the zero address to the recipient whereas delegates should be moved on the opposite way.

Recommendation

Recommend swapping the `address(0)` and `_from` to alleviate this issue. At its current state, it breaks the delegate mechanism and can also lead to a user being unable to mint/burn tokens in case the upper limit of a `uint256` is reached due to the SafeMath utilization in function `_moveDelegates()`.

Alleviation

Fixed in commit hash `bc931e1e568bdded3440922aea804f2ad68f3e1c`.

MTC-03 | Timestamp Dependence

Category	Severity	Location	Status
Volatile Code	Informational	MacaronToken.sol: 980~981	Acknowledged

Description

Functions listed in the locations section contain `require` statements using `block.timestamp`. Note the block time on testnet and mainnet BSC are different. Please understand the security risk level and trade-off of using `block.timestamp` or alias `now` as one of core factors in the contract.

Recommendation

Correct use of 15-second rule to minimize the impact caused by timestamp variance

SPC-01 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	Informational	StrategyPancakeCake.sol: 909~912, 913~916, 1171~1174	Acknowledged

Description

Function that affect the status of sensitive variables should be able to emit events as notifications to customers. MasterChefV2.sol:

- `dev()`
- `setCakeStrategy()`
- `setMacaronPerBlock()`
- `setMacaronPoolRewardRatio()`
- `setMigrator()`
- `updateMultiplier()`

StrategyPancakeCake.sol:

- `StrategyBase.setGovernance()`
- `StrategyBase.setController()`
- `StrategyPancakeCake.setBuyBurnPercent()`

StrategyPancakeCakeLP.sol:

- `StrategyBase.setGovernance()`
- `StrategyBase.setController()`

Recommendation

Consider adding events for sensitive actions, and emit it in the function. For example,

```
event SetDev(address indexed user, address indexed _devaddr);
...
function dev(address _devaddr) public {
    ...
    devaddr = _devaddr;
    ...
    emit SetDev(msg.sender, _devaddr);
}
```

SPC-02 | Ignored return values

Category	Severity	Location	Status
Logical Issue	Minor	StrategyPancakeCake.sol: 892, 1036, 1118, 1164	Acknowledged

Description

ChocoFall.sol:

- `ChocoFall.safeMacaronTransfer(address,uint256)` ignores return value by `macaron.transfer(_to,macaronBal)`
- `ChocoFall.safeMacaronTransfer(address,uint256)` ignores return value by `macaron.transfer(_to,_amount)`

MacaronRouter.sol:

- `MacaronRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256)` ignores return value by `IMacaronFactory(factory).createPair(tokenA,tokenB)`
- `MacaronRouter.removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)` ignores return value by `IMacaronPair(pair).transferFrom(msg.sender,pair,liquidity)`

MasterChefV2.sol:

- `ChocoFall.safeMacaronTransfer(address,uint256)` ignores return value by `macaron.transfer(_to,macaronBal)`
- `ChocoFall.safeMacaronTransfer(address,uint256)` ignores return value by `macaron.transfer(_to,_amount)`
- `MasterChef.add(uint256,IBEP20,bool,bool,ICakeStrategy,IBEP20)` ignores return value by `IBEP20(_lpToken).approve(address(_cakeStrategy),type()(uint256).max)`

StrategyPancakeCake.sol:

- `StrategyBase._swapTokens(address,address,uint256)` ignores return value by `pancakeRouter.swapExactTokensForTokens(_amount,0,path,address(this),now.add(1800))`
- `StrategyPancakeCake.deposit(uint256)` ignores return value by `IERC20(cakeSyrupToken).transfer(controller,_amount)`
- `StrategyPancakeCake._buyMacaronAndBurn(uint256,uint256)` ignores return value by `IERC20(macaron).transfer(address(0x000000000000000000000000000000000000dEaD),_macaronBal)`

- `StrategyPancakeCake.retireStrat()` ignores return value by `IERC20(baseToken).transfer(address(governance),baseBal)`

StrategyPancakeCakeLP.sol:

- `StrategyPancakeCLP.retireStrat()` ignores return value by `IERC20(clpToken).transfer(address(governance),baseBal)`

Recommendation

We recommend to check every function return values.

SPC-03 | Withdrawals to `controller` v.s. to `governance`

Category	Severity	Location	Status
Centralization / Privilege, Volatile Code	**Major**	StrategyPancakeCake.sol: 829~830, 853~854, 866~867, 1160~1161	**Partially Resolved**

Description

There are many functions transfer `_asset` tokens and `baseToken`s to `governance` instead of `controller`, which result in conflict with the comment descriptions in L734 of StrategyPancakeCake and L553 of StrategyPancakeCakeLP:

```
/*
A strategy must implement the following calls;
- deposit()
- withdraw(address) must exclude any tokens used in the yield - Controller role -
withdraw should return to Controller
- withdraw(uint) - Controller | Vault role - withdraw should always return to vault
- withdrawAll() - Controller | Vault role - withdraw should always return to vault
- balanceOf()

Where possible, strategies must remain as immutable as possible, instead of updating
variables, we update the contract by linking it in the controller
*/
```

Functions involved:

- StrategyPancakeCake:
  - `withdraw(address)`
  - `withdraw(uint)`
  - `withdrawAll()`
  - `retireStrat()`
- StrategyPancakeCakeLP:
  - `skim()`
  - `skimCLP()`
  - `skimRewards()`
  - `withdraw(address)`
  - `withdraw(uint)`
  - `withdrawAll()`
  - `retireStrat()`

Alleviation

Partially fixed in commit hash `bc931e1e568bdded3440922aea804f2ad68f3e1c`: The listed functions are fixed:

- StrategyPancakeCake:
  - `withdraw(address)`
  - `withdraw(uint)`
  - `withdrawAll()`
- StrategyPancakeCakeLP:
  - `withdraw(address)`
  - `withdraw(uint)`
  - `withdrawAll()`

SPC-04 | Unnecessary `pid`

Category	Severity	Location	Status
Volatile Code	Informational	StrategyPancakeCake.sol: 1036~1037	Acknowledged

Description

If the only possible pid is 0, seems it would be unnecessary to maintain an array of `poolInfo` and some following logics like `massUpdatePools()`, `emergencyWithdraw()`, etc. with input parameters of `pid`. Those would be accepted for future development and for keeping the pancakeswap interface.

SPC-05 | Privileged Ownership

Category	Severity	Location	Status
Centralization / Privilege	**Minor**	StrategyPancakeCake.sol	**Partially Resolved**

Description

In contract StrategyPancakeCake and StrategyPancakeCakeLP, the modifiers `onlyGovernance` and `onlyAuth` are over privileged. Addresses allowed by the privileged modifiers could:

1. approve token allowances for specific spenders
2. set critical state variables like Pancake Router address, router path, governance/controller address, farming token, etc.
3. withdraw, deposit and harvest

without obtaining consensus of the community.

In addition, we noticed that TimeLock is already introduced in the contract, but it is not yet taking place in controlling the functions.

Recommendation

Renounce ownership when it is the right timing, or gradually migrate to a timelock plus multisig governing procedure and let the community monitor in respect of transparency considerations.

Alleviation

[Macaron Team]: We will give privileges to timelock.

SPC-06 | Misleading implementation of `harvest()`

Category	Severity	Location	Status
Logical Issue	Minor	StrategyPancakeCake.sol: 1089	Resolved

Description

`harvest()` usually used to withdraw or claim reward, the function implementation seems not include any harvest logic.

Recommendation

Consider rename `harvest` to `earn`, and implement a real harvest function.

Alleviation

Fixed in commit hash `bc931e1e568bdded3440922aea804f2ad68f3e1c`.

SPC-07 | Misleading parameter of `deposit()`

Category	Severity	Location	Status
Logical Issue	Minor	StrategyPancakeCake.sol: 1026, 1032	Partially Resolved

Description

The `_amount` parameter is misleading, `_stakeCake()` will stake all baseToken balance instead of `_amount`.

Recommendation

We recommend to pass `_amount` parameter also to `_stakeCake()`.

Alleviation

[Macaron Team]: This is an autocompound contract so it should be run like that.

SPC-08 | No guarantee of `cakeSyrupToken` will be transferred back

Category	Severity	Location	Status
Logical Issue	Minor	StrategyPancakeCake.sol: 1036	Partially Resolved

Description

In the declaration of state variable, L755, it is mentioned that `controller` is the address of `MacaronMasterChef`, and in function `deposit()` of `StrategyPancakeCake`(L1036), all balance of the `cakeSyrupToken` in `StrategyPancakeCake` contract is transferred to `MacaronMasterChef`.

However, there are no guarantees the `cakeSyrupToken` will be transferred back.

Recommendation

We recommend to implement `transferFrom()` inside `withdraw()` function logic, so as to make sure burn the `cakeSyrupToken` from the caller.

Alleviation

[Macaron Team]: This update needs two contracts changes. We will implement this future like you suggest.

SPC-09 | `deposit()` function doesn't deposit any real `baseToken`

Category	Severity	Location	Status
Logical Issue	Minor	StrategyPancakeCake.sol: 1026~1051	Partially Resolved

Description

`deposit()` function doesn't deposit any real `baseToken`, but `withdraw()` and `withdrawToController()` will transfer `baseToken` out of strategy contract.

Recommendation

Consider rename `deposit()` to `earn()` or `stake()`, or add `transferFrom()` to transfer in real `baseToken` inside `deposit()` function.

Alleviation

[Macaron Team]: This is an autocompound contract so it should be run like that. We create this contract for live masterchef contract. This update needs two contracts changes. We will implement this future like you suggest.

SPC-10 | Gas Optimization on `pancakeRouter.swapExactTokensForTokens()`

Category	Severity	Location	Status
Gas Optimization	Informational	StrategyPancakeCake.sol: 892~893	Resolved

Description

The last parameter of `pancakeRouter.swapExactTokensForTokens()` is used for the modifier `ensure()` to provide functionalities of `deadline` for some off-chain applications. In this case, the `now.add(1800)` are not actually adding a `1800 seconds` time lock, it is just bypass the `ensure` modifier check in PancakeRouter.

Recommendation

We would like to confirm if there are some time lock logic here by design. If not, `now.add(1800)` could be changed to `now` directly to save gas.

Alleviation

Fixed in commit hash `bc931e1e568bdded3440922aea804f2ad68f3e1c`.

SPC-11 | Potential Reentrancy

Category	Severity	Location	Status
Volatile Code	Minor	StrategyPancakeCake.sol: 999~1021(StrategyPancakeCake.initialize())	Acknowledged

Description

Functions listed in the locations section are risky to reentrancy attack, as there are state variable updated after external calls.

Recommendation

Recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts. Also, it is important making sure all the contracts behind interfaces are with valid addresses and being trusted.

SPL-01 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	Informational	StrategyPancakeCakeLP.sol: 699~702, 703~706	Acknowledged

Description

Function that affect the status of sensitive variables should be able to emit events as notifications to customers. MasterChefV2.sol:

- `dev()`
- `setCakeStrategy()`
- `setMacaronPerBlock()`
- `setMacaronPoolRewardRatio()`
- `setMigrator()`
- `updateMultiplier()`

StrategyPancakeCake.sol:

- `StrategyBase.setGovernance()`
- `StrategyBase.setController()`
- `StrategyPancakeCake.setBuyBurnPercent()`

StrategyPancakeCakeLP.sol:

- `StrategyBase.setGovernance()`
- `StrategyBase.setController()`

Recommendation

Consider adding events for sensitive actions, and emit it in the function. For example,

```
event SetDev(address indexed user, address indexed _devaddr);
...
function dev(address _devaddr) public {
    ...
    devaddr = _devaddr;
    ...
    emit SetDev(msg.sender, _devaddr);
}
```

SPL-02 | Ignored return values

Category	Severity	Location	Status
Logical Issue	Minor	StrategyPancakeCakeLP.sol: 868	Acknowledged

Description

ChocoFall.sol:

- `ChocoFall.safeMacaronTransfer(address,uint256)` ignores return value by `macaron.transfer(_to,macaronBal)`
- `ChocoFall.safeMacaronTransfer(address,uint256)` ignores return value by `macaron.transfer(_to,_amount)`

MacaronRouter.sol:

- `MacaronRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256)` ignores return value by `IMacaronFactory(factory).createPair(tokenA,tokenB)`
- `MacaronRouter.removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)` ignores return value by `IMacaronPair(pair).transferFrom(msg.sender,pair,liquidity)`

MasterChefV2.sol:

- `ChocoFall.safeMacaronTransfer(address,uint256)` ignores return value by `macaron.transfer(_to,macaronBal)`
- `ChocoFall.safeMacaronTransfer(address,uint256)` ignores return value by `macaron.transfer(_to,_amount)`
- `MasterChef.add(uint256,IBEP20,bool,bool,ICakeStrategy,IBEP20)` ignores return value by `IBEP20(_lpToken).approve(address(_cakeStrategy),type()(uint256).max)`

StrategyPancakeCake.sol:

- `StrategyBase._swapTokens(address,address,uint256)` ignores return value by `pancakeRouter.swapExactTokensForTokens(_amount,0,path,address(this),now.add(1800))`
- `StrategyPancakeCake.deposit(uint256)` ignores return value by `IERC20(cakeSyrupToken).transfer(controller,_amount)`
- `StrategyPancakeCake._buyMacaronAndBurn(uint256,uint256)` ignores return value by `IERC20(macaron).transfer(address(0x000000000000000000000000000000000000dEaD),_macaronBal)`

- `StrategyPancakeCake.retireStrat()` ignores return value by `IERC20(baseToken).transfer(address(governance),baseBal)`

StrategyPancakeCakeLP.sol:

- `StrategyPancakeCLP.retireStrat()` ignores return value by `IERC20(clpToken).transfer(address(governance),baseBal)`

Recommendation

We recommend to check every function return values.

SPL-03 | Wrong Address Check for Ownership Transfer

Category	Severity	Location	Status
Volatile Code	Medium	StrategyPancakeCakeLP.sol: 867~868	Resolved

Description

Function `transferMacaronOwnership()` and `transferChocoOwnership()` in MasterChefV2 and `transferMagicBoxOwnership()` in StrategyPancakeCakeLP should have the `require` statement should check `newOwner != address(0)`, instead of `msg.sender`.

Alleviation

Fixed in commit hash `bc931e1e568bdded3440922aea804f2ad68f3e1c`.

SPL-04 | Withdrawals to `controller` v.s. to `governance`

Category	Severity	Location	Status
Centralization / Privilege, Volatile Code	**Major**	StrategyPancakeCakeLP.sol: 625~626, 630~631, 634~635, 639~640, 665~666, 678~679	**Partially Resolved**

Description

There are many functions transfer `_asset` tokens and `baseToken`s to `governance` instead of `controller`, which result in conflict with the comment descriptions in L734 of StrategyPancakeCake and L553 of StrategyPancakeCakeLP:

```
/*
A strategy must implement the following calls;
- deposit()
- withdraw(address) must exclude any tokens used in the yield - Controller role -
withdraw should return to Controller
- withdraw(uint) - Controller | Vault role - withdraw should always return to vault
- withdrawAll() - Controller | Vault role - withdraw should always return to vault
- balanceOf()

Where possible, strategies must remain as immutable as possible, instead of updating
variables, we update the contract by linking it in the controller
*/
```

Functions involved:

- StrategyPancakeCake:
  - `withdraw(address)`
  - `withdraw(uint)`
  - `withdrawAll()`
  - `retireStrat()`
- StrategyPancakeCakeLP:
  - `skim()`
  - `skimCLP()`
  - `skimRewards()`
  - `withdraw(address)`
  - `withdraw(uint)`
  - `withdrawAll()`
  - `retireStrat()`

Alleviation

Partially fixed in commit hash `bc931e1e568bdded3440922aea804f2ad68f3e1c`: The listed functions are fixed:

- StrategyPancakeCake:
  - `withdraw(address)`
  - `withdraw(uint)`
  - `withdrawAll()`
- StrategyPancakeCakeLP:
  - `withdraw(address)`
  - `withdraw(uint)`
  - `withdrawAll()`

SPL-05 | Privileged Ownership

Category	Severity	Location	Status
Centralization / Privilege	**Minor**	StrategyPancakeCakeLP.sol	**Partially Resolved**

Description

In contract StrategyPancakeCake and StrategyPancakeCakeLP, the modifiers `onlyGovernance` and `onlyAuth` are over privileged. Addresses allowed by the privileged modifiers could:

1. approve token allowances for specific spenders
2. set critical state variables like Pancake Router address, router path, governance/controller address, farming token, etc.
3. withdraw, deposit and harvest

without obtaining consensus of the community.

In addition, we noticed that TimeLock is already introduced in the contract, but it is not yet taking place in controlling the functions.

Recommendation

Renounce ownership when it is the right timing, or gradually migrate to a timelock plus multisig governing procedure and let the community monitor in respect of transparency considerations.

Alleviation

[Macaron Team]: We will give privileges to timelock.

SPL-06 | Misleading parameter of `deposit()`

Category	Severity	Location	Status
Logical Issue	Minor	StrategyPancakeCakeLP.sol: 789, 796	Partially Resolved

Description

The `_amount` parameter is misleading, `_stakeCake()` will stake all baseToken balance instead of `_amount`.

Recommendation

We recommend to pass `_amount` parameter also to `_stakeCake()`.

Alleviation

[Macaron Team]: This is an autocompound contract so it should be run like that.

SPL-07 | `deposit()` function doesn't deposit any real `baseToken`

Category	Severity	Location	Status
Logical Issue	Minor	StrategyPancakeCakeLP.sol: 789	Partially Resolved

Description

`deposit()` function doesn't deposit any real `baseToken`, but `withdraw()` and `withdrawToController()` will transfer `baseToken` out of strategy contract.

Recommendation

Consider rename `deposit()` to `earn()` or `stake()`, or add `transferFrom()` to transfer in real `baseToken` inside `deposit()` function.

Alleviation

[Macaron Team]: This is an autocompound contract so it should be run like that. We create this contract for live masterchef contract. This update needs two contracts changes. We will implement this future like you suggest.

SPL-08 | `withdraw()` doesn't burn `magicBoxToken`

Category	Severity	Location	Status
Logical Issue	Minor	StrategyPancakeCakeLP.sol: 639, 665, 678	Partially Resolved

Description

Only `withdrawToController()` burns `magicBoxToken`, but not other withdraw functions.

Recommendation

We recommend to make sure all withdraw function will burn the `magicBoxToken` minted in `deposit()` function, and burn from the correct user address.

Alleviation

[Macaron Team]: Magicbox burn in masterchef, Thic contract working dependently with masterchef. This update needs two contracts changes. We will implement this future like you suggest.

SPL-09 | Potential Reentrancy

Category	Severity	Location	Status
Volatile Code	Minor	StrategyPancakeCakeLP.sol: 766~784(StrategyPancakeCakeLP.initialize()), 804~811(StrategyPancakeCakeLP._stakeCakeLP()), 721~745(StrategyBase.executeTransaction()), 639~646(StrategyBase.withdraw(address)), 665~676(StrategyBase.withdraw(uint)), 678~684(StrategyBase.withdrawAll()), 648~661(StrategyBase.withdrawToController())	Acknowledged

Description

Functions listed in the locations section are risky to reentrancy attack, as there are state variable updated after external calls.

Recommendation

Recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts. Also, it is important making sure all the contracts behind interfaces are with valid addresses and being trusted.

SPL-10 | Typos in Comments and Functions

Category	Severity	Location	Status
Coding Style	Informational	StrategyPancakeCakeLP.sol: 638~639, 834~835, 712~713	Acknowledged

Description

MasterChefV2:

- Typo in comment body `points`
- Typo in comment body `multiplier`

StrategyPancakeCakeLP:

- Typo in comment body `governance`

ChochChef:

- Typo in comment body `points`
- Function name `setRewardToken` with wrong capital initials

Recommendation

We advise that the comment text and function name is corrected.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.