Smart Contract Review

Task: PancakeSwap IFOPool.sol Repo

Introduction: The Pancakeswap IFO Pool is a decentralized application (Dapp) built on the Binance Smart Chain that allows users to stake their Cake Tokens in the Syrup pool to get ICake tokens that enables them to participate in the Initial Farm Offering of Newly launched Project tokens on the PancakeSwap Dex.

Contract - https://tinyurl.com/Pancake-contract

pragma solidity 0.6.12;
The Compiler version used to compile the contract by the EVM Compiler.

Imports Overview:

-Ownable.sol: This is an abstract contract setup to manage ownership and access control within other contracts, it defines the owner of a contract who has privileges to transfer ownership to another address and renounce ownership of the contract.

-SafeMath.sol: This is a solidity Library that provides a set of functions for performing arithmetic operations with overflow and underflow checks.

-SafeERC20.sol: This is a solidity Library that provides a set of functions for securely interacting with ERC20 tokens by mitigating against potential vulnerabilities like improper input validation and wrong handling of return values.

-Pausable.sol: This is an abstract contract that enables the Owner of the contract to pause and unpause specific functions within a smart contract.

-IMasterChef.sol: This is an interface that enables the contract it's imported into to interact with the MasterChef staking System, It provides a series of function declarations that users of the IFO pool can interact with to deposit, withdraw, stake etc.

Structure Definition:

The Contract inherits from ownable and pausable abstract contract, It attaches the library functions of SafeERC20 and SafeMath to IERC20 (Interface for ERC20 tokens) and Uint256 (Unsigned Integer).
The UserInfo struct contains unsigned integers; shares, lastDeposited, cakeAtlastUserAction, lastUserActionTime.
The UserIFOInfo struct contains unsigned integers; lastActionBalance, lastValidActionBalance, lastActionBlock, lastValidActionBlock, lastAvgBalance.

Enum IFOActions defines the actions of the user in the IFO Pool, they are Deposit or Withdraw.
token and receiptToken variables declare the Cake token and Syrup token respectively.
masterchef variable declares a reference to the MasterChef interface.
userInfo and userIFOInfo mapping declare key to value storage of address to the UserInfo and UserIFOInfo struct

Public State Variables:

These are a series of state variables defined within the contract - startBlock, endBlock, totalShares, lastHarvestedTime, admin, treasury, Max_Perfomance_Fee, Max_Call_Fee, Max_Withdraw_Fee, Max_Withdraw_Fee_Period, perfomanceFee, callFee, withdrawFee, withdrawFeePeriod.

Events:

These are a series of events to be emitted by the contract after a certain action has been performed.
Pause, Unpause, Deposit, Withdraw, Harvest, UpdateEndBlock, ZeroFreeIFO, UpdateStartAndEndBlock, UpdateUserIFO.

Constructor:

IERC20 interface with _token and _receiptToken for Cake and Syrup token,_masterChef, _admin, _treasury, _startBlock, _endBlock
A require statement to check the current block number, _startBlock and _endBlock, then assignment of state variables, An infinite approval on the Cake token,

Modifiers

onlyAdmin modifier checks that msg.sender is admin, notContract modifier checks that msg.sender is not a smart contract and msg.sender is tx.origin

Functions

Deposit: this is an external function with whenNotPaused and notContract modifiers, a uint256 _amount is passed to the function, It checks the amount deposited makes a series of checks, calls an instance of the struct and assigns it to a mapping and then updates the user deposit action and logs an event.

_isIFOAvailable: this is an internal view function that returns a boolean value, the function returns a value of true if the current block.number is greater than the startBlock indicating the IFO being available.

_isValidActionBlock: this is an internal view function that returns a boolean value, the function returns a value of true if the block.number is greater than or equal to the startBlock and the block.number is less than or equal to the endBlock.

_calculateAvgBalance: this is an internal view function that returns uint256 value of avgBalance, the variables _lastActionBlock, -_lastValidActionBlock, _lastActionBalance, _lastValidActionbalance, and _lastAvgBalance are passed to the function. The function checks the block of the last action and returns the average balance then makes some checks and calculates the average balance.

_updateUserIFO: this is an internal function, the amount and IFOActions enum are passed to the function.
It calls an instance of the UserIFOInfo struct and assigns it to a mapping, then calculates average balance, checks the user action and updates the average balance and logs an update event.

getUserCredit: this is an external view function that returns the average balance, it takes the user's address, It calls an instance of the UserIFOInfo struct and assigns it to a mapping, then makes some checks and returns the average balance.

withdrawAll: this is an external function that uses the notContract modifier, the function retrieves the user's share balance from the userInfo mapping and calls the withdraw function with the number of shares passed in as an argument.

emergencyWithdrawAll: this is an external function that uses the notContract modifier. The zeroFreeIFO and withdrawV1 functions are called, an emergency share withdrawal is performed.

_zeroFreeIFO: this is an internal function. It calls an instance of the UserIFOInfo struct and assigns it to a mapping, Then it sets all the variables from the struct to Zero. then an event is emitted.

withdraw: this is a public function that utilizes the notContract modifier, user's share is the given parameter, It calls an instance of the UserInfo struct and assigns it to a mapping, checks are performed on the shares, the amount is calculated, then user and total share is updated, the tokens are unstaked then withdrawal fee and user amount is calculated. The update function is called, the user amount is transferred and an event is logged.

withdrawV1: this is an internal function that takes the user's share as a parameter, It calls an instance of the UserInfo struct and assigns it to a mapping, Then it makes checks on the shares, then calculates the amount to be withdrawn and updates the user's and total shares, then it unstakes the token and transfer the withdrawal fee to the treasury, then it updates the user's action and transfers the current amount to the user and finally logs a withdrawal event.

harvest: this is an external function that utilizes the notContract and whenNotPaused modifier. The token balance is calculated. Then the tokens are unstaked and balance is updated, the performanceFee and callFee is calculated and then transferred to the treasury and user's address respectively, the earn function is called and a harvest event is logged.

setAdmin & setTreasury: these are external functions that uses the onlyOwner modifier, the address of the admin and treasury is passed to the functions respectively, a check is made to prevent zero address and the admin and treasury variable is updated.

setPerfomanceFee, setCallFee, setWithdrawFee: these are external functions that uses the onlyAdmin modifier, the performanceFee, callFee and withdrawFee is passed to the functions respectively and a check is made on the functions after which the variables are updated.

setWithdrawFeePeriod: this is an external function that uses the onlyAdmin modifier, the withdrawFeePeriod variable is passed to the function and a check is made on it, then it is updated.

updateStartAndEndBlocks: this is an external function that uses the onlyAdmin modifier, the startBlock and endBlock is passed to the function, a check is made, then the startBlock and endBlock is updated and an event is logged.

updateEndBlock: this is an external function that uses the onlyAdmin modifier, the new endblock is passed to the function, then a series of checks is performed. Then endBlock is updated and an event is logged.

emergencyWithdraw: this is an external function that uses the onlyAdmin modifier, that calls the emergencyWithdraw function from the masterchef contract, then makes a check and pauses the contract.

inCaseTokensGetStuck: this is an external function that uses the onlyAdmin modifier, the address of the stuck token is passed, then it compares the stuck token with the deposit and receipt token, then it queries the balance of the stuck token and transfers the amount to the admin.

pause: this is an external function that uses the onlyAdmin and whenNotPaused modifier, it calls the _pause function to set the contract in a paused state, then it emits an event to log the Pause action.

unpause: this is an external function that uses the onlyAdmin and whenPaused modifier, it calls the _unpause function to return the contract to normal functionalities, then it emits an event to log the action.

calculateHarvestCakeRewards: this is an external view function that returns a uint256, the pendingCake function is called from the masterchef contract to estimate the pending rewards, which is calculated and the current callFee is returned.

calculateTotalPendingCakeRewards: this is an external view function that returns a uint256, the pendingCake function is called from the masterchef contract to estimate the pending rewards, which is calculated and the total amount of cake rewards is returned.

getPricePerFullShare: this is an external view function that returns a uint256, the function calculates the price per full share based on the contract state and balance of tokens in it.

available: this is a public view function that returns a uint256, the function returns the balance of tokens in the contract.

balanceOf: this is a public view function that returns a uint256, the userInfo function is called to get information on the user's staking position, then it returns the user token balance in the contract with the amount added.

_earn: an internal function that checks the balance in the masterchef contract and calls the enterStaking function.

_isContract: this is an internal view function that returns a boolean, it initializes a variable, then performs a low level code and queries the code size in the address provided and returns the size to ascertain the type of address it received.

Findings

- There should be an approval revoke for the masterchef Contract on the token in case of an emergency in the deposit function
- There is no check if the startblock has passed such that if it has and the IFO has ended users might still be able to participate from the _isIFOAvailable function
- There is no check for the success or failure of the enterstaking function call in the earn function

Overall the code is structured properly and has the necessary checks to prevent unforeseen behaviors it also follows best practices of modular solidity smart contract development, the findings are certain checks that were missed.