Coinbae Audit

Pud.Fi Strategy Bacdai Contract March 2021

Contents

Introduction

Audit:

In March 2021 Coinbae's audit division performed an audit for the Pud.Fi Strategy Bacdai Contract.

https://etherscan.io/address/0x31dfcb1c5df01a27f8b0b5f9cd1585fe92c7970e#code

Pud.Fi

Pud.Fi takes DeFi to a whole new level of integrating technology and finance. Founded by Chocolate, together with several FinTech experts and enthusiasts, we have established a one of its kind Decentralized Finance (DeFi) protocol, a vault aggregation platform.

Scope of the audit:

The following Coinbae audit will cover assertions and property checking, ERC Standards, solidity coding best practices and conformity to the solidity style guide.

Overview:

Name: Pud.Fi Strategy Bacdai Contract
Website: https://pud.fi/

Audit Report Scope

Assertions and Property Checking:

1. Solidity assert violation.
2. Solidity AssertionFailed event.

ERC Standards:

1. Incorrect ERC20 implementation.

Solidity Coding Best Practices:

1. Outdated compiler version.
2. No or floating compiler version set.
3. Use of right-to-left-override control character.
4. Shadowing of built-in symbol.
5. Incorrect constructor name.
6. State variable shadows another state variable.
7. Local variable shadows a state variable.
8. Function parameter shadows a state variable.
9. Named return value shadows a state variable.
10. Unary operation without effect Solidity code analysis.
11. Unary operation directly after assignment.
12. Unused state variable.
13. Unused local variable.
14. Function visibility is not set.
15. State variable visibility is not set.

Solidity Coding Best Practices (Continued):

16. Use of deprecated functions: call code(), sha3(), ...
17. Use of deprecated global variables (msg.gas, ...).
18. Use of deprecated keywords (throw, var).
19. Incorrect function state mutability.
20. Does the code conform to the Solidity styleguide.

Convert code to conform Solidity style guide:

1. Convert all code so that it is structured accordingly the Solidity style guide.

Audit Report Scope

Categories:

High Severity:

High severity issues opens the contract up for exploitation from malicious actors. We do not recommend deploying contracts with high severity issues.

Medium Severity Issues:

Medium severity issues are errors found in contracts that hampers the effectiveness of the contract and may cause outcomes when interacting with the contract. It is still recommended to fix these issues.

Low Severity Issues:

Low severity issues are warning of minor impact on the overall integrity of the contract. These can be fixed with less urgency.

Optimization Issues:

Optimization issues are issues that pose no security risk or expose any underlying vulnerabilities, but instead make the contract more efficient.

Informational Issues:

Informational issues are issues that point to smart contract coding best practises.

Audit Report

22	22	0
Identified	Confirmed	Critical

0	6	2
High	Medium	Low

Optimization issues identified: **14**

Analysis:
https://etherscan.io/address/0x31dfcb1c5df01a27f8b0b5f9cd1585fe92c7970e#code

Risk:
Low

Optimization issues identified:

External-function

name() should be declared external:
- ERC20.name() (contracts/lib/erc20.sol#287-289)

symbol() should be declared external:
- ERC20.symbol() (contracts/lib/erc20.sol#295-297)

decimals() should be declared external:
- ERC20.decimals() (contracts/lib/erc20.sol#312-314)

totalSupply() should be declared external:
- ERC20.totalSupply() (contracts/lib/erc20.sol#319-321)

balanceOf(address) should be declared external:
- ERC20.balanceOf(address) (contracts/lib/erc20.sol#326-328)

transfer(address,uint256) should be declared external:
- ERC20.transfer(address,uint256) (contracts/lib/erc20.sol#338-341)

allowance(address,address) should be declared external:
- ERC20.allowance(address,address) (contracts/lib/erc20.sol#346-348)

approve(address,uint256) should be declared external:
- ERC20.approve(address,uint256) (contracts/lib/erc20.sol#357-360)

transferFrom(address,address,uint256) should be declared external:
- ERC20.transferFrom(address,address,uint256) (contracts/lib/erc20.sol#374-378)

increaseAllowance(address,uint256) should be declared external:
- ERC20.increaseAllowance(address,uint256) (contracts/lib/erc20.sol#392-395)

Optimization issues identified:

External-function

decreaseAllowance(address,uint256) should be declared external:
- ERC20.decreaseAllowance(address,uint256)
(contracts/lib/erc20.sol#411-414)

balanceOf() should be declared external:
- StrategyBase.balanceOf()
(contracts/strategies/strategy-base.sol#82-84)

harvest() should be declared external:
- StrategyBase.harvest() (contracts/strategies/strategy-base.sol#195)
- StrategyBasisFarmBase.harvest()
(contracts/strategies/strategy-basis-farm-base.sol#77-151)

execute(address,bytes) should be declared external:
- StrategyBase.execute(address,bytes)
(contracts/strategies/strategy-base.sol#199-233)

Low Issues

Shadowing-local:

ERC20.constructor(string,string).name (contracts/lib/erc20.sol#278) shadows:
- ERC20.name() (contracts/lib/erc20.sol#287-289) (function)

ERC20.constructor(string,string).symbol (contracts/lib/erc20.sol#278) shadows:
- ERC20.symbol() (contracts/lib/erc20.sol#295-297) (function)

Medium Issues

Unused-return:

StrategyBase._swapUniswap(address,address,uint256) (contracts/strategies/strategy-base.sol#236-267) ignores return value by UniswapRouterV2(univ2Router2).swapExactTokensForTokens(_amount,0,path,address(this),now.add(60)) (contracts/strategies/strategy-base.sol#260-266)

StrategyBase._swapUniswapWithPath(address[],uint256) (contracts/strategies/strategy-base.sol#269-286) ignores return value by UniswapRouterV2(univ2Router2).swapExactTokensForTokens(_amount,0,path,address(this),now.add(60)) (contracts/strategies/strategy-base.sol#279-285)

StrategyBase._swapSushiswap(address,address,uint256) (contracts/strategies/strategy-base.sol#288-319) ignores return value by UniswapRouterV2(sushiRouter).swapExactTokensForTokens(_amount,0,path,address(this),now.add(60)) (contracts/strategies/strategy-base.sol#312-318)

Medium Issues

Unused-return:

StrategyBase._swapSushiswapWithPath(address[],uint256) (contracts/strategies/strategy-base.sol#321-338) ignores return value by UniswapRouterV2(sushiRouter).swapExactTokensForTokens(_amount,0,path,address(this),now.add(60)) (contracts/strategies/strategy-base.sol#331-337)

StrategyBasisFarmBase.harvest() (contracts/strategies/strategy-basis-farm-base.sol#77-151) ignores return value by UniswapRouterV2(univ2Router2).addLiquidity(token1,token2,_token1,_token2,0,0,address(this),now + 60) (contracts/strategies/strategy-basis-farm-base.sol#127-136)

StrategyBasisFarmBase.harvest() (contracts/strategies/strategy-basis-farm-base.sol#77-151) ignores return value by IERC20(token1).transfer(IController(controller).treasury(),IERC20(token1).balanceOf(address(this))) (contracts/strategies/strategy-basis-farm-base.sol#139-142)

Audit Team

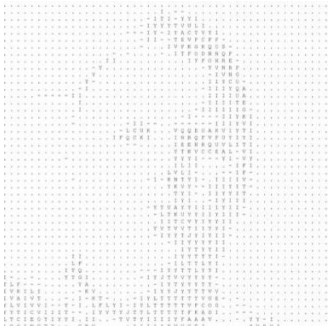
Team Lead: Eelko Neven

Eelko has been in the it/security space since 1991. His passion started when he was confronted with a formatted hard drive and no tools to undo it. At that point he started reading a lot of material on how computers work and how to make them work for others. After struggling for a few weeks he finally wrote his first HD data recovery program. Ever since then when he was faced with a challenge he just persisted until he had a solution.

This mindset helped him tremendously in the security space. He found several vulnerabilities in large corporation servers and notified these corporations in a responsible manner. Among those are Google, Twitter, General Electrics etc.

For the last 12 years he has been working as a professional security /code auditor and performed over 1500 security audits / code reviews, he also wrote a similar amount of reports.

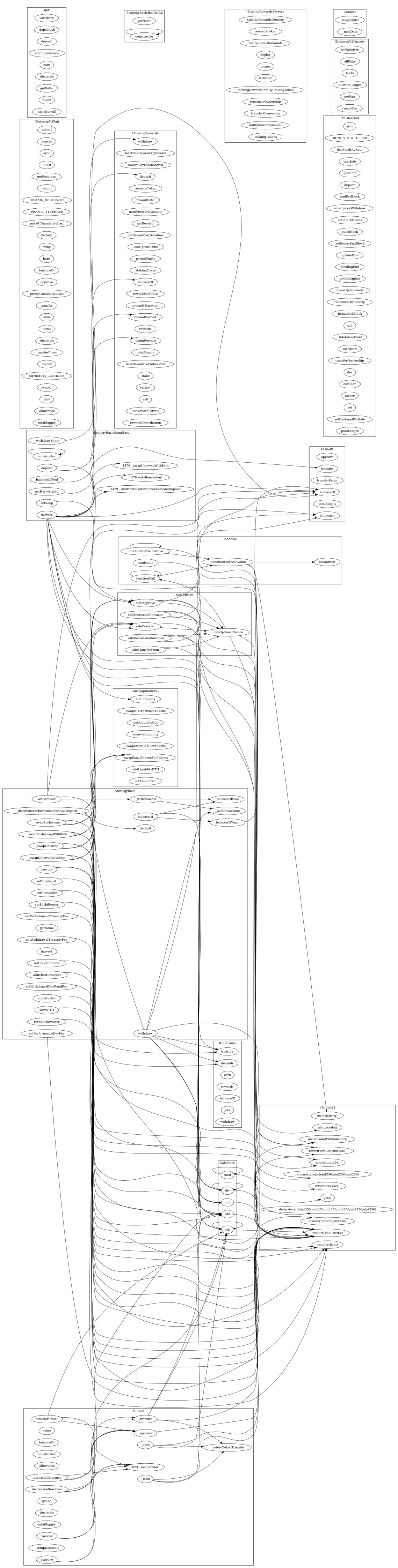
He has extensive knowledge of the Solidity programming language and this is why he loves to do Defi and other smartcontract reviews.



Email:
info@coinbae.com

Disclaimer

Coinbae audit is not a security warranty, investment advice, or an endorsement of Pud.Fi. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Pud.Fi team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.



Ijar
withdraw
depositAll
deposit
claimInsurance
earn
decimals
getRatio
token
withdrawAll
StrategyBasisBacDaiLp
getName
constructor
IStakingRewardsFactory
stakingRewardsGenesis
rewardsToken
notifyRewardAmounts
deploy
owner
isOwner
stakingRewardsInfoByStakingToken
renounceOwnership
transferOwnership
notifyRewardAmount
stakingTokens
Context
msgSender
_msgData
IUniswapV2Factory
feeToSetter
allPairs
feeTo
allPairsLength
getPair
createPair
IUniswapV2Pair
token1
nonces
sync
kLast
getReserves
permit
DOMAIN_SEPARATOR
PERMIT_TYPEHASH
price1CumulativeLast
factory
swap
burn
balanceOf
approve
price0CumulativeLast
transfer
skim
name
decimals
transferFrom
token0
MINIMUM_LIQUIDITY
symbol
mint
allowance
totalSupply
IStakingRewards
withdraw
lastTimeRewardApplicable
rewardPerTokenStored
deposit
rewardsToken
rewardRate
notifyRewardAmount
getReward
getRewardForDuration
lastUpdateTime
periodFinish
stakingToken
balanceOf
rewardPerToken
rewardsDuration
rewardEarned
rewards
claimReward
totalSupply
userRewardPerTokenPaid
stake
earned
exit
stakeWithPermit
rewardsDistribution
IMasterchef
pud
BONUS_MULTIPLIER
devFundDivRate
userInfo
poolInfo
deposit
pudPerBlock
emergencyWithdraw
setPudPerBlock
startBlock
setBonusEndBlock
updatePool
pendingPud
getMultiplier
massUpdatePools
renounceOwnership
bonusEndBlock
add
totalAllocPoint
withdraw
transferOwnership
dev
devaddr
owner
set
setDevFundDivRate
poolLength
StrategyBasisFarmBase
withdrawSome
constructor
deposit
balanceOfPool
getHarvestable
setKeep
harvest
1479__swapUniswapWithPath
1479_onlyBenevolent
1479__distributePerformanceFeesAndDeposit
IERC20
approve
transfer
transferFrom
balanceOf
totalSupply
allowance
Address
functionCallWithValue
sendValue
functionCall
_functionCallWithValue
isContract
SafeERC20
safeApprove
safeIncreaseAllowance
safeTransfer
safeDecreaseAllowance
safeTransferFrom
callOptionalReturn
UniswapRouterV2
addLiquidity
swapETHForExactTokens
getAmountsOut
removeLiquidity
swapExactETHForTokens
swapExactTokensForTokens
addLiquidityETH
getAmountsIn
StrategyBase
withdrawAll
distributePerformanceFeesAndDeposit
swapSushiswap
swapSushiswapWithPath
swapUniswap
swapUniswapWithPath
execute
setStrategist
setController
setSushiRouter
setPerformanceTreasuryFee
getName
setWithdrawalTreasuryFee
harvest
setUniv2Router2
whitelistHarvester
setWithdrawalDevFundFee
constructor
setWETH
revokeHarvester
setPerformanceDevFee
withdrawAll
balanceOf
deposit
withdraw
balanceOfPool
_withdrawSome
balanceOfWant
IController
treasury
devaddr
earn
rewards
balanceOf
jars
withdraw
[Solidity]
revert(string)
abi.decode()
abi.encodeWithSelector()
revert(uint256,uint256)
mload(uint256)
returndatacopy(uint256,uint256,uint256)
returndatasize()
gas()
delegatecall(uint256,uint256,uint256,uint256,uint256,uint256)
mstore(uint256,uint256)
require(bool,string)
require(bool)
SafeMath
mod
div
mul
add
sub
ERC20
transferFrom
name
balanceOf
constructor
allowance
increaseAllowance
decreaseAllowance
symbol
decimals
totalSupply
transfer
_setupDecimals
approve
_transfer
_approve
_burn
_mint
_beforeTokenTransfer
823__msgSender