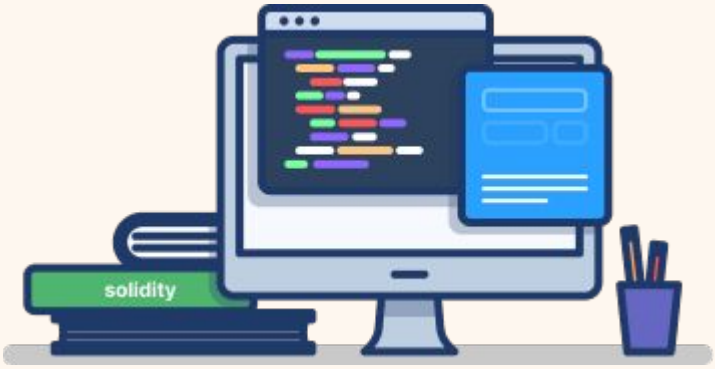
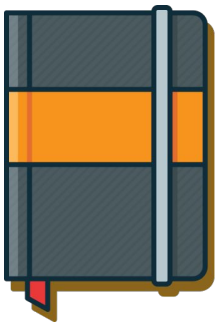
Coinbae Audit

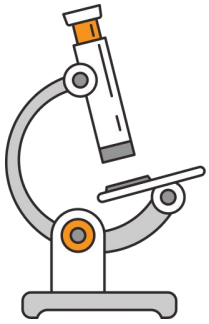


Pud.Fi MasterChef Contract March 2021

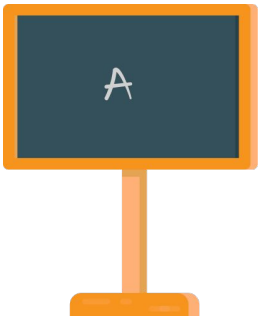
Contents



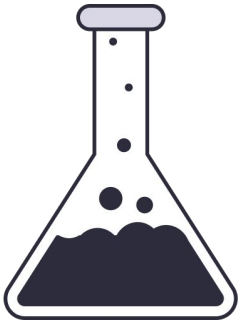
Introduction, 2



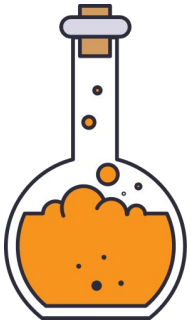
Scope, 3



Synopsis, 5



Low Severity, 9



Medium Severity, 10

Team, 15

Introduction

Audit:

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

https://etherscan.io/address/0x15Bee180BB39eE5c0166E63313C33984376930Db#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 MasterChef Contract
Website: https://pud.fi/
MasterChef was the master of pud. He now governs over Pud. He can make Pud and he is a fair guy.

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

43	43	0
Identified	Confirmed	Critical

0	12	5
High	Medium	Low

Optimization issues identified: **26**

Analysis:
https://etherscan.io/address/0x15Bee180BB39eE5c0166E63313C33984376930Db#code

Risk:
Low

Optimization issues identified:

External-function

add(uint256,IERC20,bool,address) should be declared external:
- MasterChef.add(uint256,IERC20,bool,address)
(contracts/MasterChef.sol#104-126)

set(uint256,uint256,bool) should be declared external:
- MasterChef.set(uint256,uint256,bool)
(contracts/MasterChef.sol#129-141)

withdraw(uint256,uint256) should be declared external:
- MasterChef.withdraw(uint256,uint256)
(contracts/MasterChef.sol#268-302)

emergencyWithdraw(uint256) should be declared external:
- MasterChef.emergencyWithdraw(uint256)
(contracts/MasterChef.sol#305-324)

dev(address) should be declared external:
- MasterChef.dev(address) (contracts/MasterChef.sol#337-340)

setTreasury(address) should be declared external:
- MasterChef.setTreasury(address)
(contracts/MasterChef.sol#343-345)

setPudPerBlock(uint256) should be declared external:
- MasterChef.setPudPerBlock(uint256)
(contracts/MasterChef.sol#347-351)

setStartBlock(uint256) should be declared external:
- MasterChef.setStartBlock(uint256)
(contracts/MasterChef.sol#353-360)

Optimization issues identified:

External-function

setBonusEndBlock(uint256) should be declared external:
- MasterChef.setBonusEndBlock(uint256)
(contracts/MasterChef.sol#362-364)

setDevFundDivRate(uint256) should be declared external:
- MasterChef.setDevFundDivRate(uint256)
(contracts/MasterChef.sol#366-369)

setPoolStrategy(uint256,address) should be declared external:
- MasterChef.setPoolStrategy(uint256,address)
(contracts/MasterChef.sol#376-381)

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)

Optimization issues identified:

External-function

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)

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

owner() should be declared external:
- Ownable.owner() (contracts/lib/ownable.sol#35-37)

renounceOwnership() should be declared external:
- Ownable.renounceOwnership() (contracts/lib/ownable.sol#54-57)

transferOwnership(address) should be declared external:
- Ownable.transferOwnership(address) (contracts/lib/ownable.sol#63-67)

mint(address,uint256) should be declared external:
- PudToken.mint(address,uint256) (contracts/pud-token.sol#9-11)

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)

Reentrancy-events:

Reentrancy in MasterChef.deposit(uint256,uint256) (contracts/MasterChef.sol#226-265):
External calls:
- updatePool(_pid) (contracts/MasterChef.sol#230)
  - pud.mint(devaddr,pudReward.div(devFundDivRate)) (contracts/MasterChef.sol#215)
  - pud.mint(address(this),pudReward) (contracts/MasterChef.sol#216)
- safePudTransfer(msg.sender,pending) (contracts/MasterChef.sol#236)
  - pud.transfer(_to,pudBal) (contracts/MasterChef.sol#330)
  - pud.transfer(_to,_amount) (contracts/MasterChef.sol#332)
- pool.lpToken.safeTransferFrom(address(msg.sender),pool.strategy,_amount) (contracts/MasterChef.sol#244-248)

Event emitted after the call(s):
- Deposit(msg.sender,_pid,_amount) (contracts/MasterChef.sol#264)

Low Issues

Reentrancy-events:

Reentrancy in MasterChef.emergencyWithdraw(uint256)
(contracts/MasterChef.sol#305-324):
External calls:
- IStrategy(pool.strategy).withdraw(r) (contracts/MasterChef.sol#313)
- pool.lpToken.safeTransfer(address(msg.sender),r)
(contracts/MasterChef.sol#320)
Event emitted after the call(s):
- EmergencyWithdraw(msg.sender,_pid,user.shares)
(contracts/MasterChef.sol#321)

Reentrancy in MasterChef.withdraw(uint256,uint256)
(contracts/MasterChef.sol#268-302):
External calls:
- updatePool(_pid) (contracts/MasterChef.sol#273)
- pud.mint(devaddr,pudReward.div(devFundDivRate))
(contracts/MasterChef.sol#215)
- pud.mint(address(this),pudReward)
(contracts/MasterChef.sol#216)
- safePudTransfer(msg.sender,pending)
(contracts/MasterChef.sol#281)
- pud.transfer(_to,pudBal) (contracts/MasterChef.sol#330)
- pud.transfer(_to,_amount) (contracts/MasterChef.sol#332)
- IStrategy(pool.strategy).withdraw(r) (contracts/MasterChef.sol#290)
- pool.lpToken.safeTransfer(address(msg.sender),r)
(contracts/MasterChef.sol#297)
Event emitted after the call(s):
- Withdraw(msg.sender,_pid,r) (contracts/MasterChef.sol#301)

Medium Issues

Divide-before-multiply:

MasterChef.pendingPud(uint256,address) (contracts/MasterChef.sol#162-185) performs a multiplication on the result of a division:
-pudReward = multiplier.mul(pudPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (contracts/MasterChef.sol#175-178)
-accPudPerShare = accPudPerShare.add(pudReward.mul(1e12).div(lpSupply)) (contracts/MasterChef.sol#179-181)

MasterChef.updatePool(uint256) (contracts/MasterChef.sol#196-223) performs a multiplication on the result of a division:
-pudReward = multiplier.mul(pudPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (contracts/MasterChef.sol#210-213)
-pool.accPudPerShare = pool.accPudPerShare.add(pudReward.mul(1e12).div(lpSupply)) (contracts/MasterChef.sol#219-221)

Reentrancy-no-Eth:

Reentrancy in MasterChef.add(uint256,IERC20,bool,address) (contracts/MasterChef.sol#104-126):
External calls:
- massUpdatePools() (contracts/MasterChef.sol#111)
  - pud.mint(devaddr,pudReward.div(devFundDivRate)) (contracts/MasterChef.sol#215)
  - pud.mint(address(this),pudReward) (contracts/MasterChef.sol#216)
State variables written after the call(s):
- poolInfo.push(PoolInfo(_lpToken,_allocPoint,lastRewardBlock,0,_strategy,0)) (contracts/MasterChef.sol#116-125)
- totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/MasterChef.sol#115)

Medium Issues

Reentrancy-no-Eth:

Reentrancy in MasterChef.deposit(uint256,uint256)
(contracts/MasterChef.sol#226-265):
External calls:
- updatePool(_pid) (contracts/MasterChef.sol#230)
- pud.mint(devaddr,pudReward.div(devFundDivRate))
(contracts/MasterChef.sol#215)
- pud.mint(address(this),pudReward)
(contracts/MasterChef.sol#216)
- safePudTransfer(msg.sender,pending)
(contracts/MasterChef.sol#236)
- pud.transfer(_to,pudBal) (contracts/MasterChef.sol#330)
- pud.transfer(_to,_amount) (contracts/MasterChef.sol#332)
-
pool.lpToken.safeTransferFrom(address(msg.sender),pool.strategy,_amo
unt) (contracts/MasterChef.sol#244-248)
State variables written after the call(s):
- pool.totalShares = pool.totalShares.add(shares)
(contracts/MasterChef.sol#262)
- user.shares = user.shares.add(shares)
(contracts/MasterChef.sol#260)
- user.rewardDebt = user.shares.mul(pool.accPudPerShare).div(1e12)
(contracts/MasterChef.sol#261)

Reentrancy in MasterChef.emergencyWithdraw(uint256)
(contracts/MasterChef.sol#305-324):
External calls:
- IStrategy(pool.strategy).withdraw(r) (contracts/MasterChef.sol#313)
- pool.lpToken.safeTransfer(address(msg.sender),r)
(contracts/MasterChef.sol#320)
State variables written after the call(s):
- user.shares = 0 (contracts/MasterChef.sol#322)
- user.rewardDebt = 0 (contracts/MasterChef.sol#323)

Medium Issues

Reentrancy-no-ETH:

Reentrancy in MasterChef.set(uint256,uint256,bool)
(contracts/MasterChef.sol#129-141):
External calls:
- massUpdatePools() (contracts/MasterChef.sol#135)
- pud.mint(devaddr,pudReward.div(devFundDivRate))
(contracts/MasterChef.sol#215)
- pud.mint(address(this),pudReward)
(contracts/MasterChef.sol#216)
State variables written after the call(s):
- poolInfo[_pid].allocPoint = _allocPoint
(contracts/MasterChef.sol#140)
- totalAllocPoint =
totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint)
(contracts/MasterChef.sol#137-139)

Reentrancy in MasterChef.setPoolStrategy(uint256,address)
(contracts/MasterChef.sol#376-381):
External calls:
- IStrategy(pool.strategy).harvest() (contracts/MasterChef.sol#378)
- IStrategy(pool.strategy).withdrawAll(_strategy)
(contracts/MasterChef.sol#379)
State variables written after the call(s):
- pool.strategy = _strategy (contracts/MasterChef.sol#380)

Medium Issues

Reentrancy-no-ETH:

Reentrancy in MasterChef.withdraw(uint256,uint256)
(contracts/MasterChef.sol#268-302):
External calls:
- updatePool(_pid) (contracts/MasterChef.sol#273)
- pud.mint(devaddr,pudReward.div(devFundDivRate))
(contracts/MasterChef.sol#215)
- pud.mint(address(this),pudReward)
(contracts/MasterChef.sol#216)
- safePudTransfer(msg.sender,pending)
(contracts/MasterChef.sol#281)
- pud.transfer(_to,pudBal) (contracts/MasterChef.sol#330)
- pud.transfer(_to,_amount) (contracts/MasterChef.sol#332)
State variables written after the call(s):
- pool.totalShares = pool.totalShares.sub(_shares)
(contracts/MasterChef.sol#284)
- user.shares = user.shares.sub(_shares)
(contracts/MasterChef.sol#282)
- user.rewardDebt = user.shares.mul(pool.accPudPerShare).div(1e12)
(contracts/MasterChef.sol#283)

Unused Return:

MasterChef.safePudTransfer(address,uint256)
(contracts/MasterChef.sol#327-334) ignores return value by
pud.transfer(_to,pudBal) (contracts/MasterChef.sol#330)

MasterChef.safePudTransfer(address,uint256)
(contracts/MasterChef.sol#327-334) ignores return value by
pud.transfer(_to,_amount) (contracts/MasterChef.sol#332)

MasterChef.setPoolStrategy(uint256,address)
(contracts/MasterChef.sol#376-381) ignores return value by
IStrategy(pool.strategy).withdrawAll(_strategy)
(contracts/MasterChef.sol#379)

Audit Team

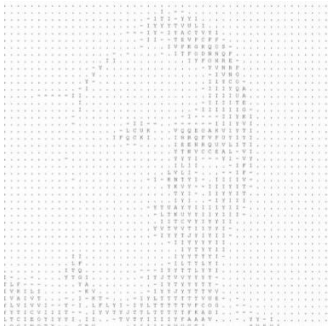
Team Lead: Eelko Neven

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.

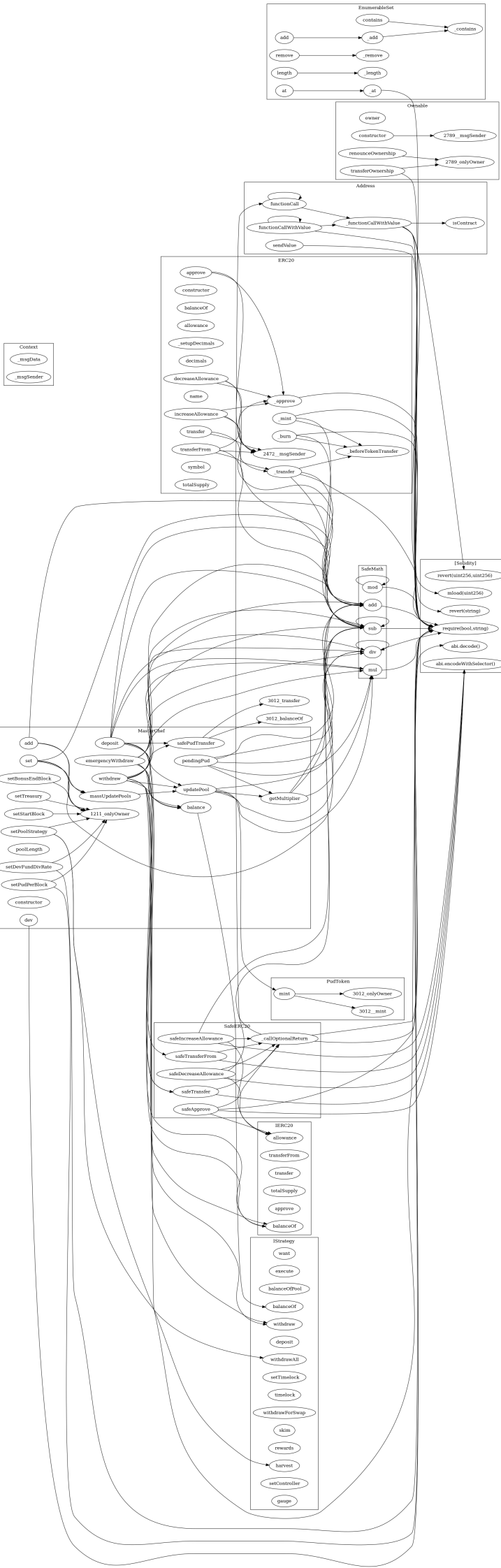
He has extensive knowledge of the Solidity programming language and this is why he loves to do Defi and other smartcontract reviews.



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.


EnumerableSet
contains
_contains
add
_add
remove
_remove
length
_length
at
_at
Ownable
owner
constructor
2789__msgSender
renounceOwnership
2789_onlyOwner
transferOwnership
Address
functionCall
functionCallWithValue
functionCallWithValue
isContract
sendValue
ERC20
approve
constructor
balanceOf
allowance
_setupDecimals
decimals
decreaseAllowance
name
approve
increaseAllowance
mint
transfer
burn
beforeTokenTransfer
transferFrom
2472__msgSender
symbol
_transfer
totalSupply
Context
_msgData
msgSender
SafeMath
mod
add
sub
div
mul
[Solidity]
revert(uint256,uint256)
mload(uint256)
revert(string)
require(bool,string)
abi.decode()
abi.encodeWithSelector()
3012_transfer
3012_balanceOf
MasterChef
add
set
setBonusEndBlock
setTreasury
setStartBlock
setPoolStrategy
poolLength
setDevFundDivRate
setPudPerBlock
constructor
dev
deposit
emergencyWithdraw
withdraw
massUpdatePools
1211_onlyOwner
safePudTransfer
pendingPud
updatePool
balance
getMultiplier
PudToken
mint
3012_onlyOwner
3012__mint
SafeERC20
safeIncreaseAllowance
safeTransferFrom
safeDecreaseAllowance
safeTransfer
safeApprove
callOptionalReturn
IERC20
allowance
transferFrom
transfer
totalSupply
approve
balanceOf
IStrategy
want
execute
balanceOfPool
balanceOf
withdraw
deposit
withdrawAll
setTimelock
timelock
withdrawForSwap
skim
rewards
harvest
setController
gauge