SAULIDITY

2022 | SMART CONTRACT SECURITY ANALYSIS

PREPARED BY

Saulidity

PRESENTED TO

SynthWave Optimizer

2022

SAULIDITY

SECURITY REPORT

Smart Contract Audit

saulidity.com

Saulidity

@Saulidity

DISCLAIMER

This report does not constitute financial advice, and Saulidity is not accountable or liable for any negative consequences resulting from this report, nor may Saulidity be held liable in any way. You agree to the terms of this disclaimer by reading any part of the report. If you do not agree to the terms, please stop reading this report immediately and delete and destroy any and all copies of this report that you have downloaded and/or printed. This report was entirely based on information given by the audited party and facts that existed prior to the audit. Saulidity and its auditors cannot be held liable for any outcome, including modifications (if any) made to the contract(s) for the audit that was completed. No modifications have been made to the contract(s) by the Saulidity team, but if it does, it will be indicated explicitly. The audit does not include the project team, website, logic, or tokenomics, but if it does, it will be indicated explicitly. The security is evaluated only on the basis of smart contracts only. There were no security checks performed on any apps or activities. There hasn't been a review of any product codes. It is assumed by Saulidity that the information and materials given were not tampered with, censored, or misrepresented. Even if this report exists and Saulidity makes every effort to uncover any security flaws, you should not rely completely on it and should conduct your own independent research. Saulidity hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Saulidity, for any amount or kind of loss or damage that may result to you or any other person or any kind of company, community, association and institution. Saulidity is the exclusive owner of this report, and it is published by Saulidity. Without Saulidity's express written authorization, use of this report for any reason other than a security interest in the individual contacts, or use of sections of this report, is forbidden.

Table of Contents

Saulidity

Saulidity is a renowned cybersecurity firm specializing in the analysis and development of Smart contracts. Saulidity, as a full-service security organization, can help with a variety of audits and project development.

In a market where confidence and trust are key, a genuine project may simply increase its user base enormously with an official audit performed by Saulidity.

Introduction

For a thorough understanding of the audit, please read the entire document.

The goal of the audit was to find any potential smart contract security problems and vulnerabilities.
The information in this report should be used to understand the smart contract's risk exposure and as a guide to improving the smart contract's security posture by addressing the concerns that were discovered.

During our audit, we conducted a thorough inquiry using automated analysis and manual review approaches.

The security specialists did a complete study independently of one another in order to uncover any security issues in the contracts as comprehensively as feasible. For optimum security and professionalism, all of our audits are undertaken by at least two independent auditors.

The audit was carried out on contracts that had not yet been deployed. The project's website, logic, or tokenomics have not been vetted by the Saulidity team.

Scope

We analyze smart contracts for both well-known and more specific vulnerabilities.

Here are some of the most well-known vulnerabilities that are taken into account but not limited to:

- Reentrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Style guide violation
- Transfer forwards all gas
- ERC20 API violation
- Compiler version not fixed
- Unchecked external call - Unchecked math
- Unsafe type inference
- Implicit visibility level

Audit & Project Information

Project Name	Synthwave Optimizer
Contract Name	NativeFarm.sol Strategy_V3_PCS.sol
Report ID	SAUL47100 V1.0
Website	synthwave.finance
Contact	RetroDefiKing
Contact Information	TG @RetroDefiKing
Code language	Solidity

Summary Table

SEVERITY	FOUND
Critical	0
High	0
Medium	6
Low	4
Lowest / Code Style / Optimized Practice	5

Executive Summary

ACCORDING TO THE ANALYSIS,

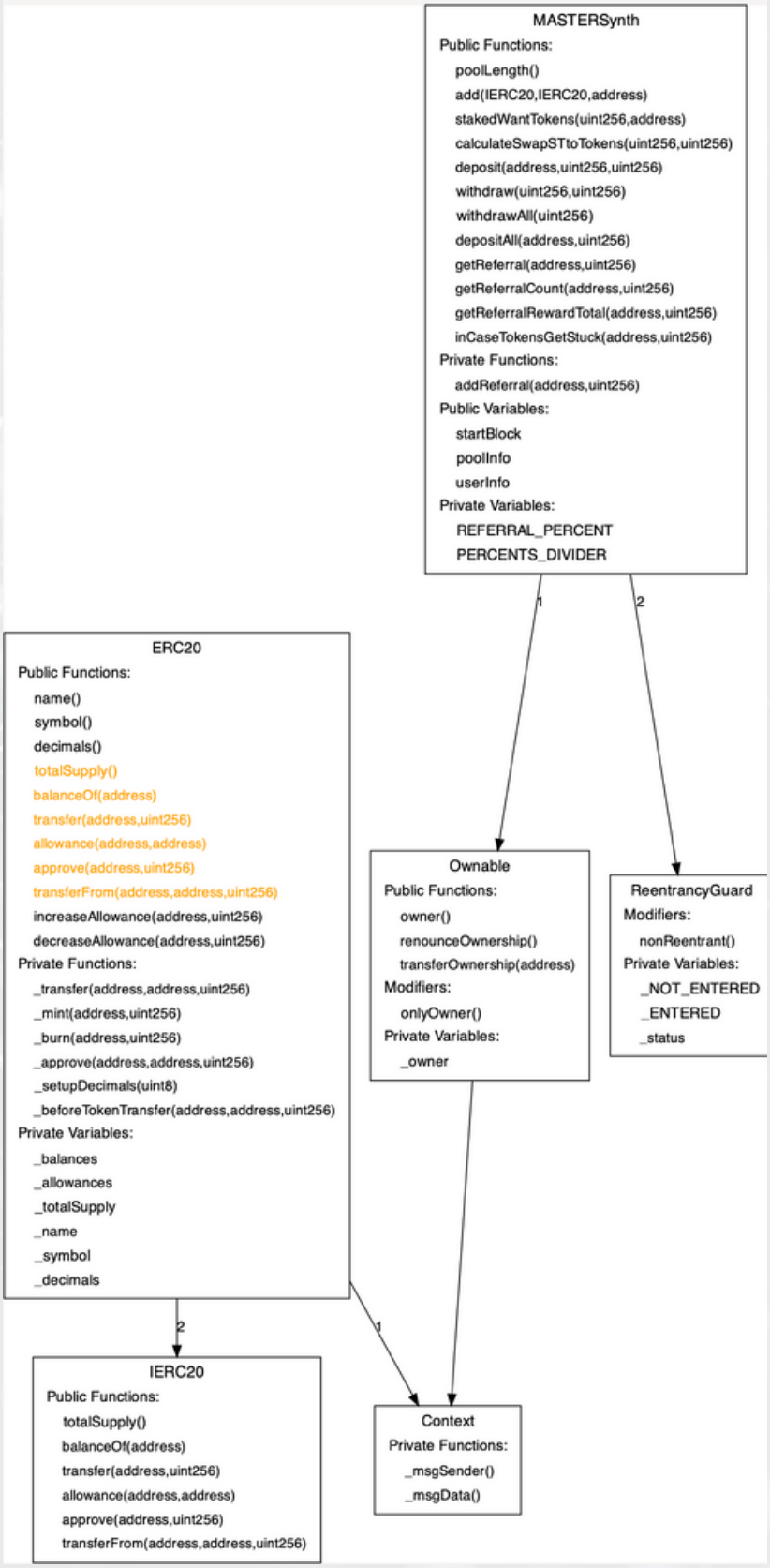
6 MEDIUM, 4 LOW, 5 LOWEST/OPTIMIZATION SEVERITY ISSUES WERE FOUND.

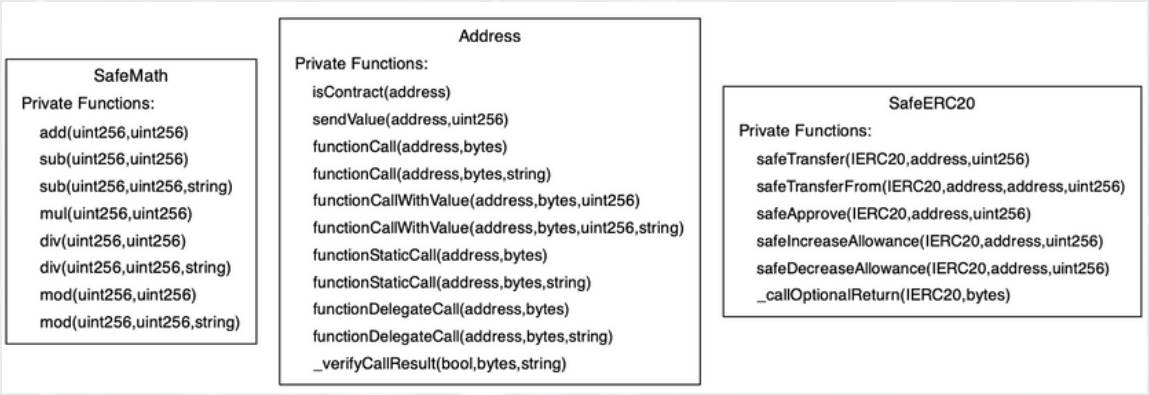
IT SHOULD BE NOTED THAT IMPORTANT FINDINGS SHOULD BE ACKNOWLEDGED AND IF POSSIBLE, MITIGATED BEFORE DEPLOYMENT

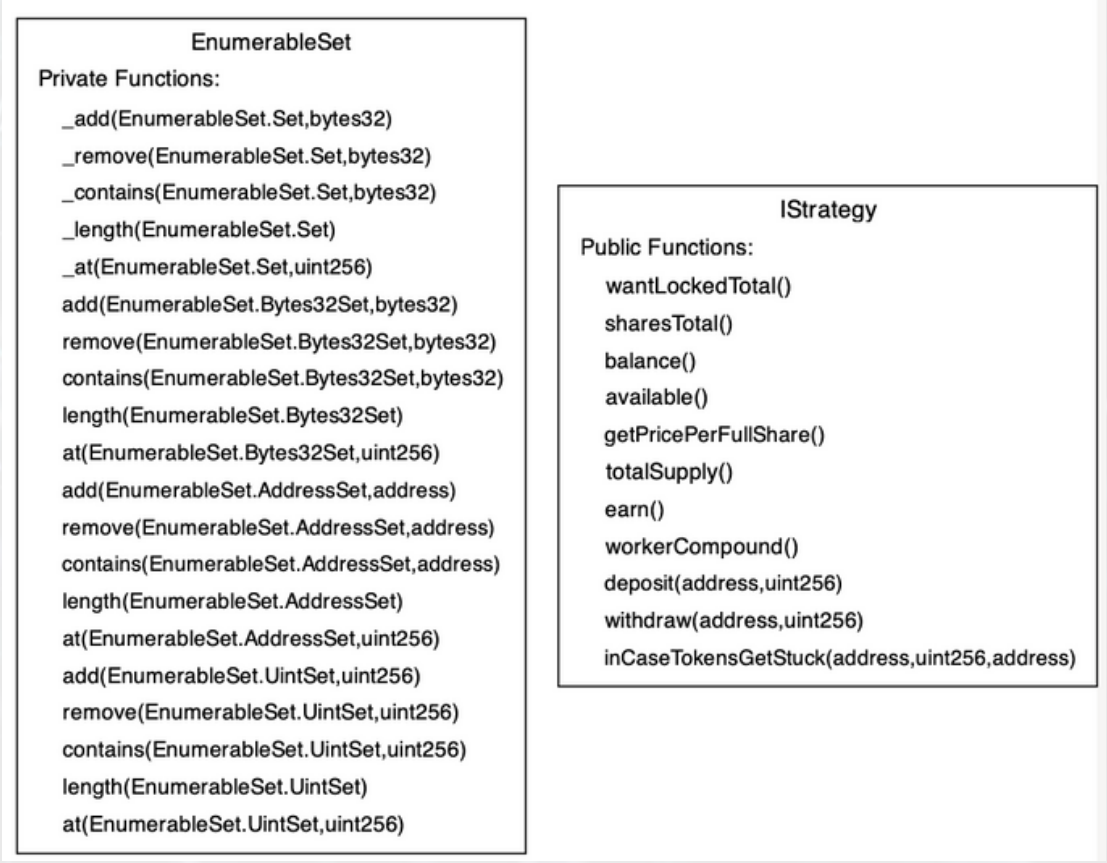
ALL ISSUES FOUND DURING AUTOMATED ANALYSIS WERE MANUALLY REVIEWED, AND FINDINGS ARE PRESENTED IN THE ANALYSIS SECTION.

Inheritance

NativeFarm.sol




SafeMath
Private Functions:
add(uint256,uint256)
sub(uint256,uint256)
sub(uint256,uint256,string)
mul(uint256,uint256)
div(uint256,uint256)
div(uint256,uint256,string)
mod(uint256,uint256)
mod(uint256,uint256,string)
Address
Private Functions:
isContract(address)
sendValue(address,uint256)
functionCall(address,bytes)
functionCall(address,bytes,string)
functionCallWithValue(address,bytes,uint256)
functionCallWithValue(address,bytes,uint256,string)
functionStaticCall(address,bytes)
functionStaticCall(address,bytes,string)
functionDelegateCall(address,bytes)
functionDelegateCall(address,bytes,string)
_verifyCallResult(bool,bytes,string)
SafeERC20
Private Functions:
safeTransfer(IERC20,address,uint256)
safeTransferFrom(IERC20,address,address,uint256)
safeApprove(IERC20,address,uint256)
safeIncreaseAllowance(IERC20,address,uint256)
safeDecreaseAllowance(IERC20,address,uint256)
_callOptionalReturn(IERC20,bytes)


EnumerableSet
Private Functions:
_add(EnumerableSet.Set,bytes32)
_remove(EnumerableSet.Set,bytes32)
_contains(EnumerableSet.Set,bytes32)
_length(EnumerableSet.Set)
_at(EnumerableSet.Set,uint256)
add(EnumerableSet.Bytes32Set,bytes32)
remove(EnumerableSet.Bytes32Set,bytes32)
contains(EnumerableSet.Bytes32Set,bytes32)
length(EnumerableSet.Bytes32Set)
at(EnumerableSet.Bytes32Set,uint256)
add(EnumerableSet.AddressSet,address)
remove(EnumerableSet.AddressSet,address)
contains(EnumerableSet.AddressSet,address)
length(EnumerableSet.AddressSet)
at(EnumerableSet.AddressSet,uint256)
add(EnumerableSet.UintSet,uint256)
remove(EnumerableSet.UintSet,uint256)
contains(EnumerableSet.UintSet,uint256)
length(EnumerableSet.UintSet)
at(EnumerableSet.UintSet,uint256)
IStrategy
Public Functions:
wantLockedTotal()
sharesTotal()
balance()
available()
getPricePerFullShare()
totalSupply()
earn()
workerCompound()
deposit(address,uint256)
withdraw(address,uint256)
inCaseTokensGetStuck(address,uint256,address)

Inheritance

Strategy_V3_PCS.sol

StrategyV3_PCS

Public Functions:

- balance()
- available()
- getPricePerFullShare()
- deposit(address,uint256)
- farm()
- withdraw(address,uint256)
- earn()
- convertDustToEarned()
- pause()
- unpause()
- setControllerFee(uint256)
- setbuyBackRate(uint256)
- setCompoundFee(uint256)
- setGov(address)
- setBuybackRouterAddress(address)
- workerCompound()
- inCaseTokensGetStuck(address,uint256,address)

Private Functions:

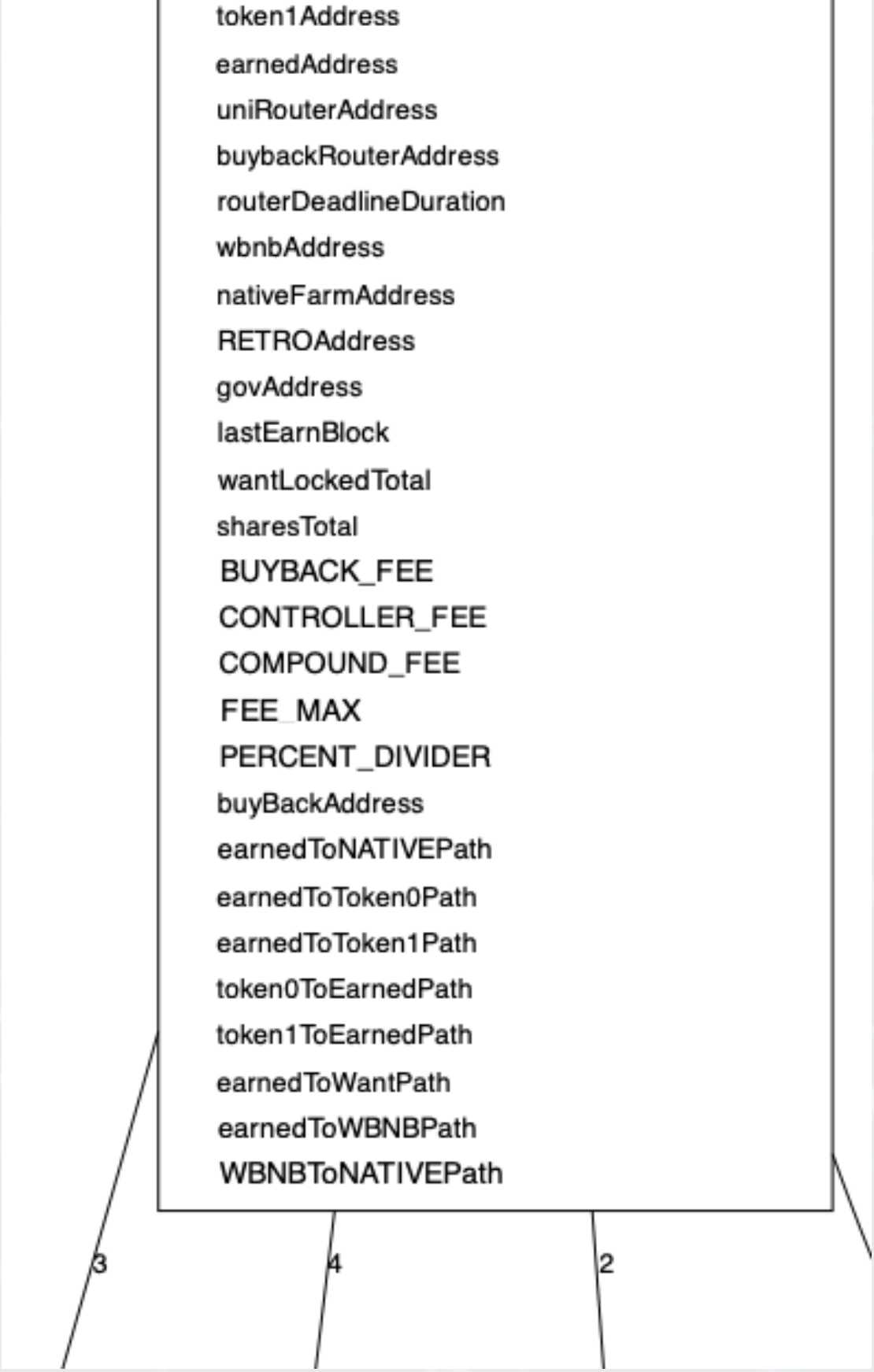
- _farm()
- buyBack(uint256)
- distributeFees(uint256)

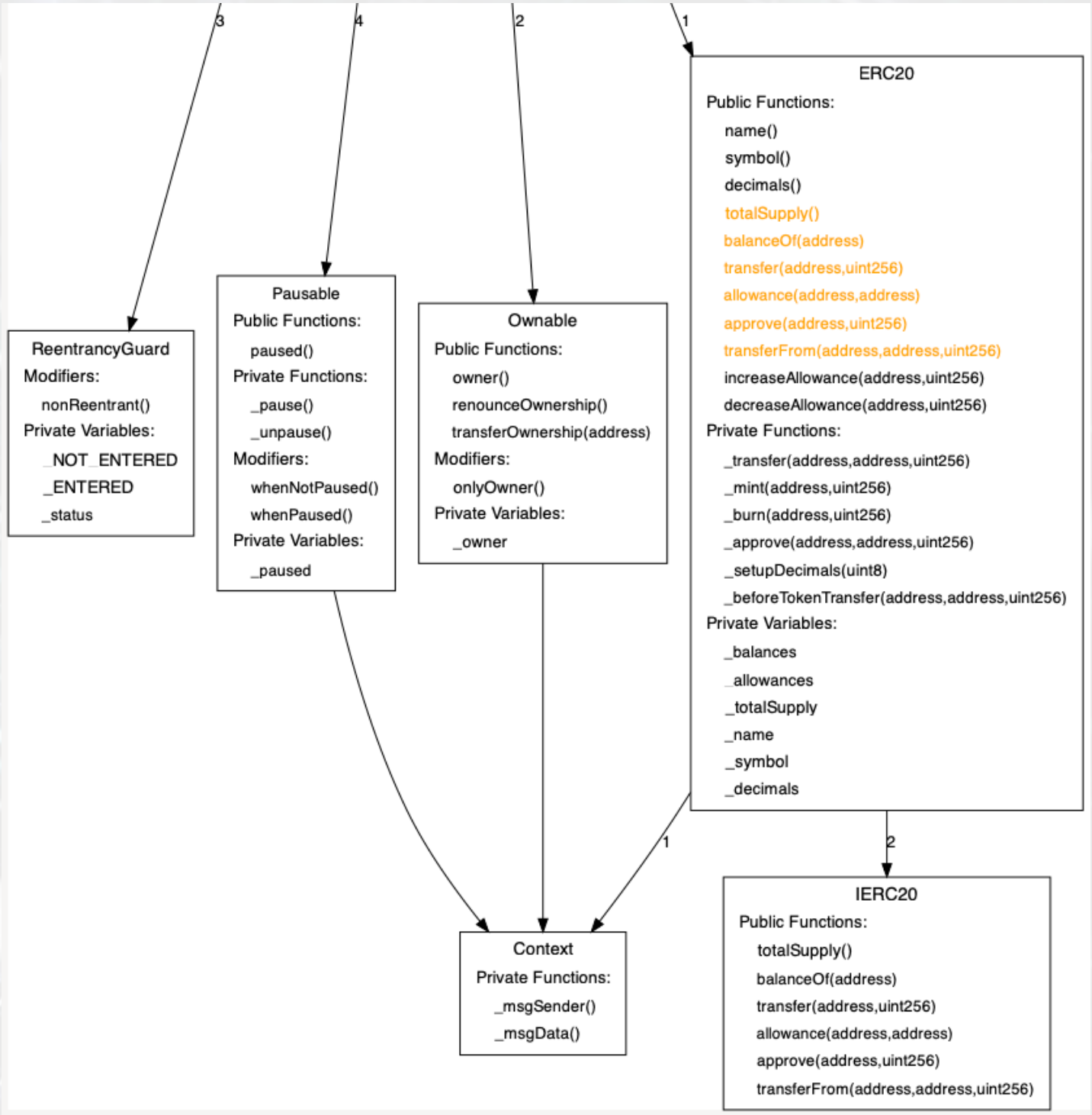
Modifiers:

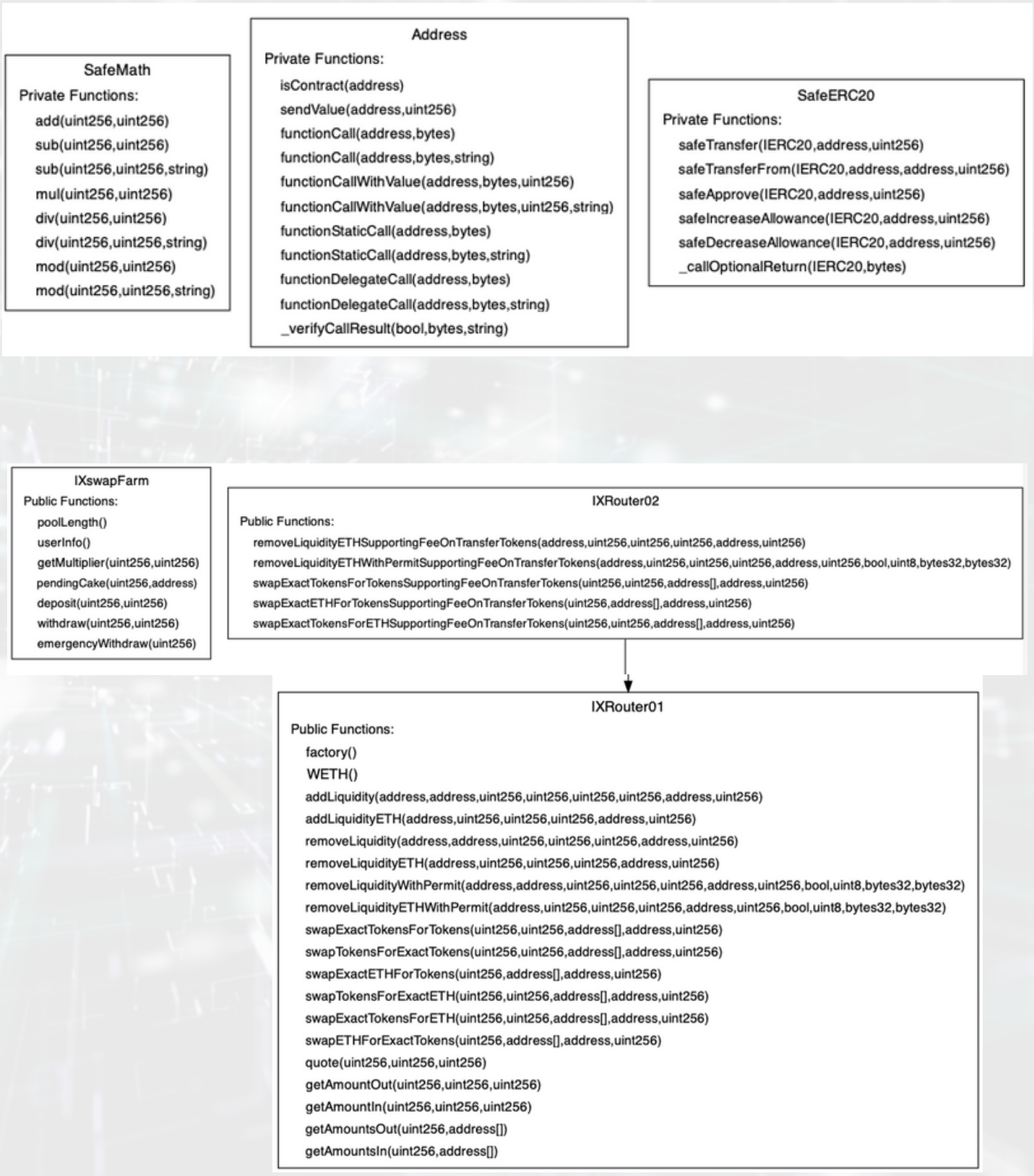
- onlyAllowGov()

Public Variables:

- isSingleVault
- isAutoComp
- farmContractAddress
- pid
- wantAddress
- token0Address

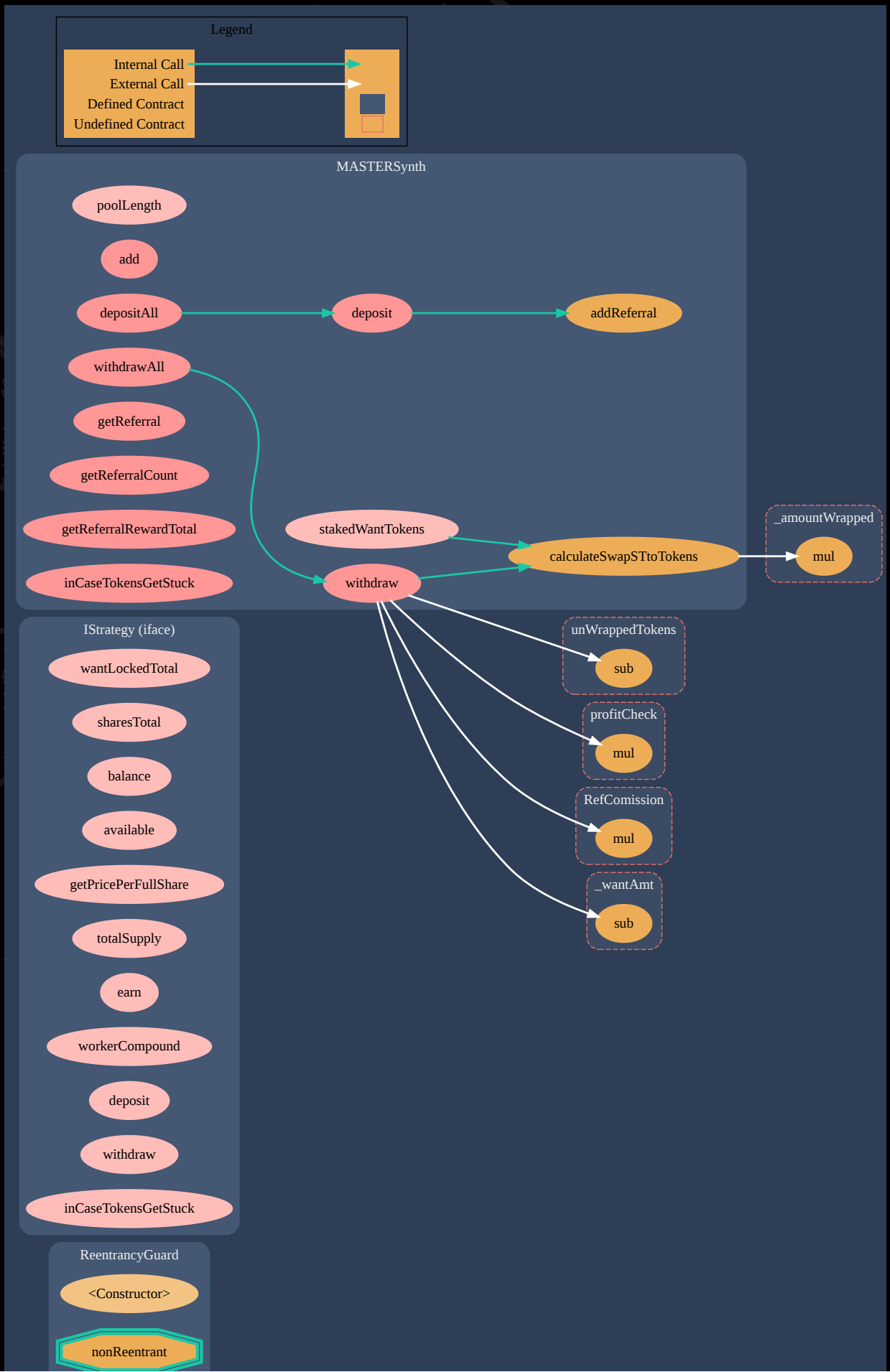

token1Address
earnedAddress
uniRouterAddress
buybackRouterAddress
routerDeadlineDuration
wbnbAddress
nativeFarmAddress
RETROAddress
govAddress
lastEarnBlock
wantLockedTotal
sharesTotal
BUYBACK_FEE
CONTROLLER_FEE
COMPOUND_FEE
FEE_MAX
PERCENT_DIVIDER
buyBackAddress
earnedToNATIVEPath
earnedToToken0Path
earnedToToken1Path
token0ToEarnedPath
token1ToEarnedPath
earnedToWantPath
earnedToWBNBPath
WBNBToNATIVEPath
3
4
2

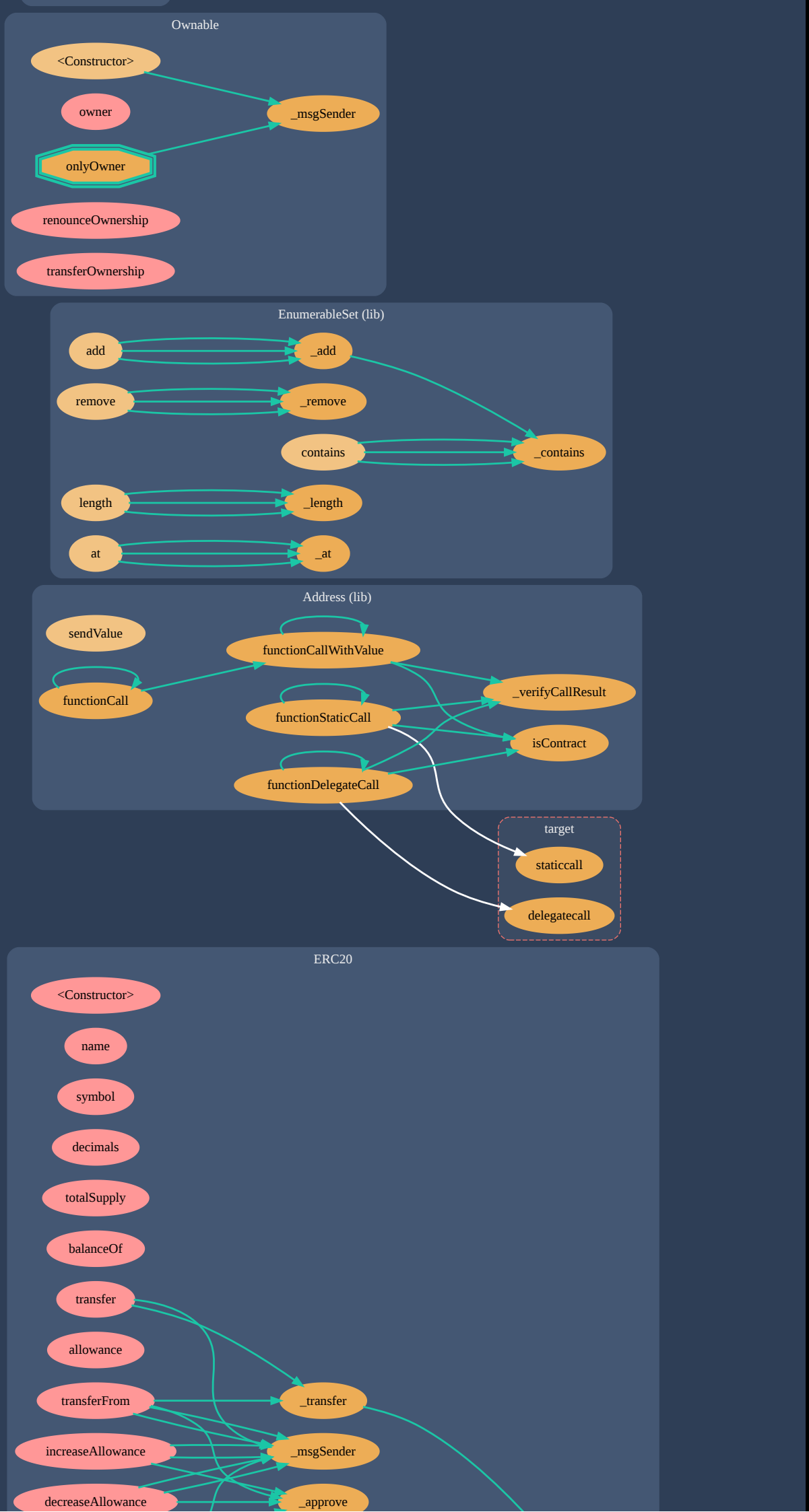

3
4
2
1
ReentrancyGuard
Modifiers:
nonReentrant()
Private Variables:
_NOT_ENTERED
_ENTERED
_status
Pausable
Public Functions:
paused()
Private Functions:
_pause()
_unpause()
Modifiers:
whenNotPaused()
whenPaused()
Private Variables:
_paused
Ownable
Public Functions:
owner()
renounceOwnership()
transferOwnership(address)
Modifiers:
onlyOwner()
Private Variables:
_owner
ERC20
Public Functions:
name()
symbol()
decimals()
totalSupply()
balanceOf(address)
transfer(address,uint256)
allowance(address,address)
approve(address,uint256)
transferFrom(address,address,uint256)
increaseAllowance(address,uint256)
decreaseAllowance(address,uint256)
Private Functions:
_transfer(address,address,uint256)
_mint(address,uint256)
_burn(address,uint256)
_approve(address,address,uint256)
_setupDecimals(uint8)
_beforeTokenTransfer(address,address,uint256)
Private Variables:
_balances
_allowances
_totalSupply
_name
_symbol
_decimals
1
2
Context
Private Functions:
_msgSender()
_msgData()
IERC20
Public Functions:
totalSupply()
balanceOf(address)
transfer(address,uint256)
allowance(address,address)
approve(address,uint256)
transferFrom(address,address,uint256)

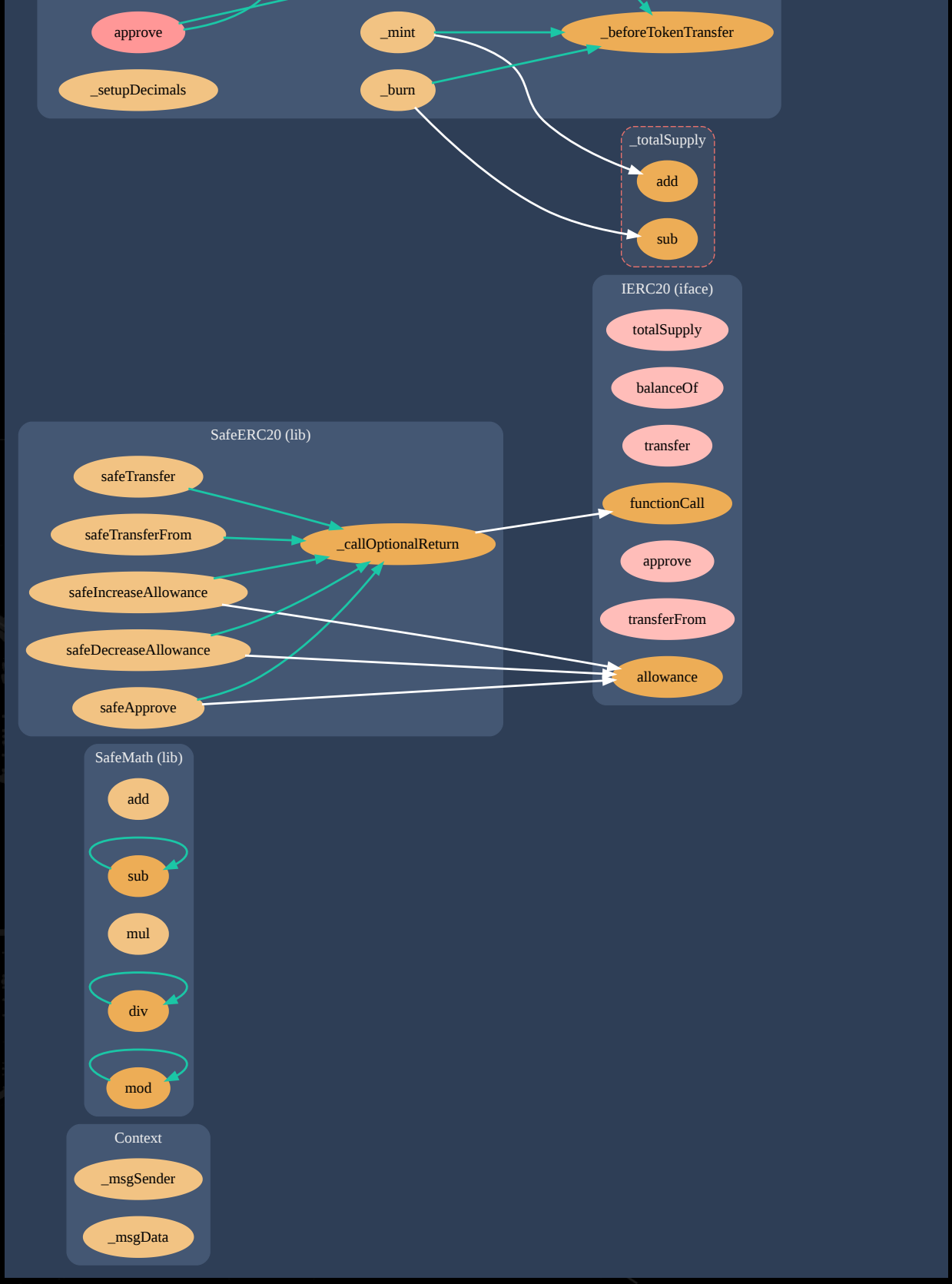

SafeMath
Private Functions:
add(uint256,uint256)
sub(uint256,uint256)
sub(uint256,uint256,string)
mul(uint256,uint256)
div(uint256,uint256)
div(uint256,uint256,string)
mod(uint256,uint256)
mod(uint256,uint256,string)
Address
Private Functions:
isContract(address)
sendValue(address,uint256)
functionCall(address,bytes)
functionCall(address,bytes,string)
functionCallWithValue(address,bytes,uint256)
functionCallWithValue(address,bytes,uint256,string)
functionStaticCall(address,bytes)
functionStaticCall(address,bytes,string)
functionDelegateCall(address,bytes)
functionDelegateCall(address,bytes,string)
_verifyCallResult(bool,bytes,string)
SafeERC20
Private Functions:
safeTransfer(IERC20,address,uint256)
safeTransferFrom(IERC20,address,address,uint256)
safeApprove(IERC20,address,uint256)
safeIncreaseAllowance(IERC20,address,uint256)
safeDecreaseAllowance(IERC20,address,uint256)
_callOptionalReturn(IERC20,bytes)
IXswapFarm
Public Functions:
poolLength()
userInfo()
getMultiplier(uint256,uint256)
pendingCake(uint256,address)
deposit(uint256,uint256)
withdraw(uint256,uint256)
emergencyWithdraw(uint256)
IXRouter02
Public Functions:
removeLiquidityETHSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256)
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint256)
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
IXRouter01
Public Functions:
factory()
WETH()
addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
removeLiquidityETH(address,uint256,uint256,uint256,address,uint256)
removeLiquidityWithPermit(address,address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
removeLiquidityETHWithPermit(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
swapExactTokensForTokens(uint256,uint256,address[],address,uint256)
swapTokensForExactTokens(uint256,uint256,address[],address,uint256)
swapExactETHForTokens(uint256,address[],address,uint256)
swapTokensForExactETH(uint256,uint256,address[],address,uint256)
swapExactTokensForETH(uint256,uint256,address[],address,uint256)
swapETHForExactTokens(uint256,address[],address,uint256)
quote(uint256,uint256,uint256)
getAmountOut(uint256,uint256,uint256)
getAmountIn(uint256,uint256,uint256)
getAmountsOut(uint256,address[])
getAmountsIn(uint256,address[])

Call Graph

NativeFarm.sol

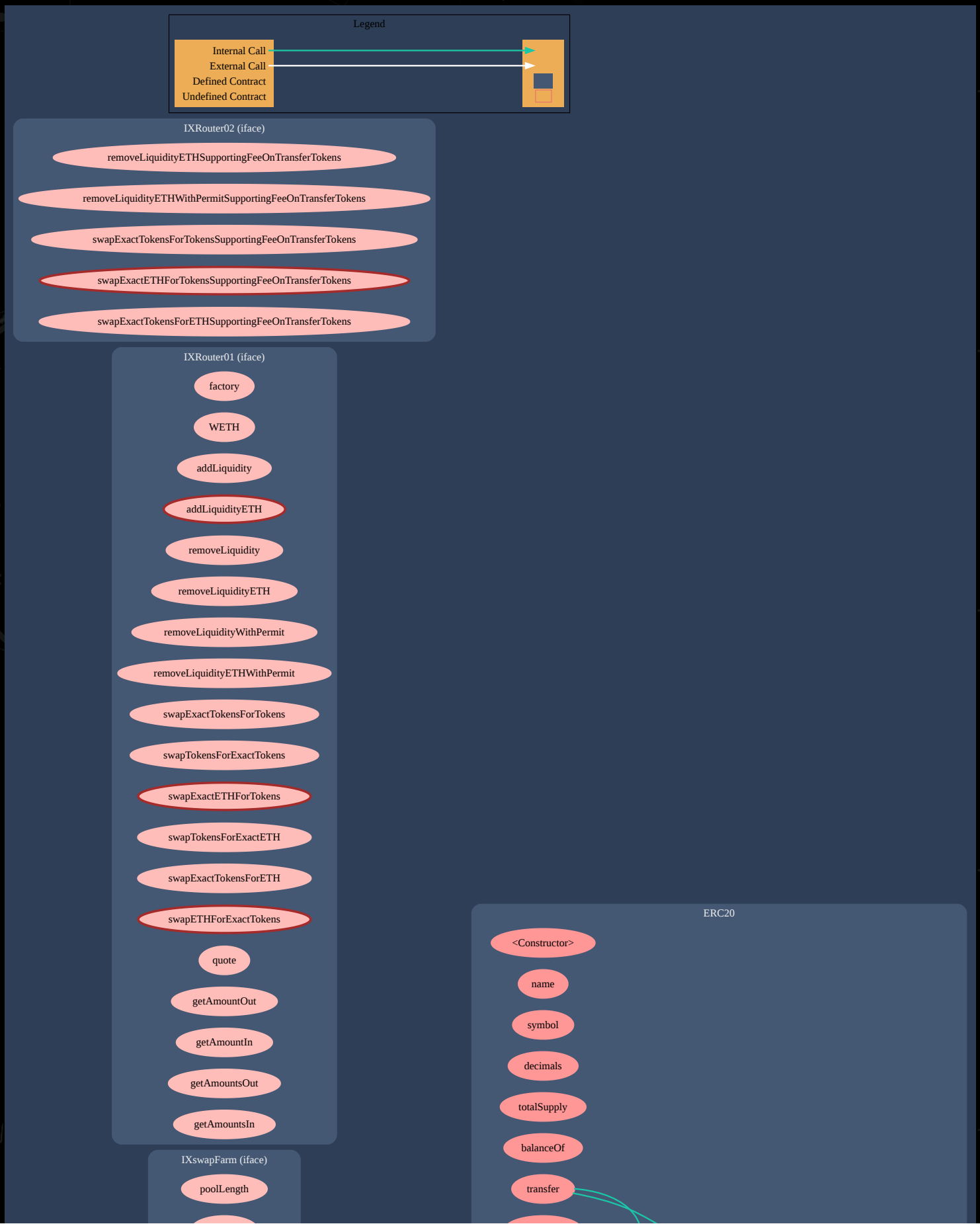


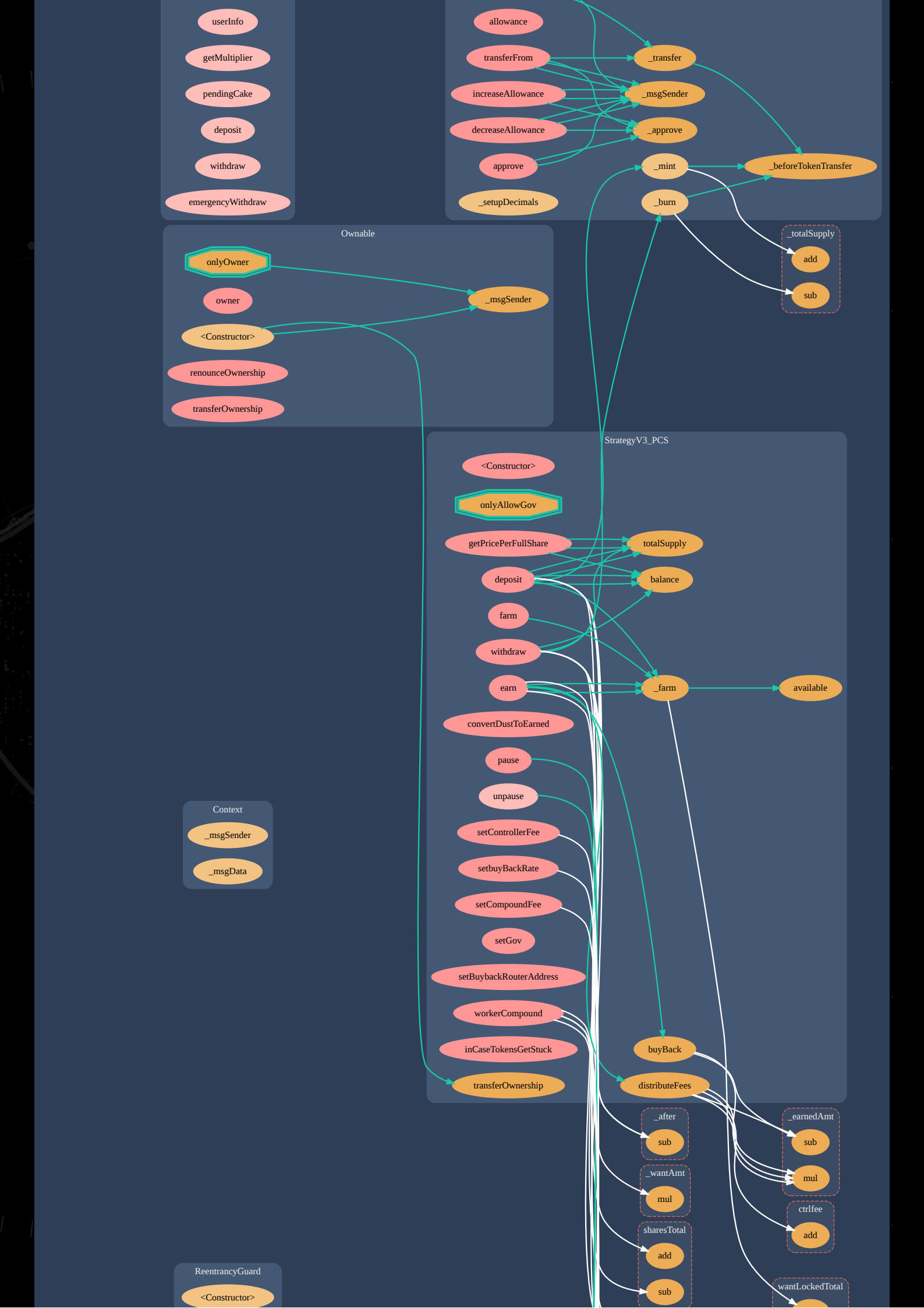

Ownable
<Constructor>
owner
_msgSender
onlyOwner
renounceOwnership
transferOwnership
EnumerableSet (lib)
add
_add
remove
_remove
contains
_contains
length
_length
at
_at
Address (lib)
sendValue
functionCallWithValue
functionCall
_verifyCallResult
functionStaticCall
isContract
functionDelegateCall
target
staticcall
delegatecall
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transfer
allowance
transferFrom
_transfer
increaseAllowance
_msgSender
decreaseAllowance
_approve

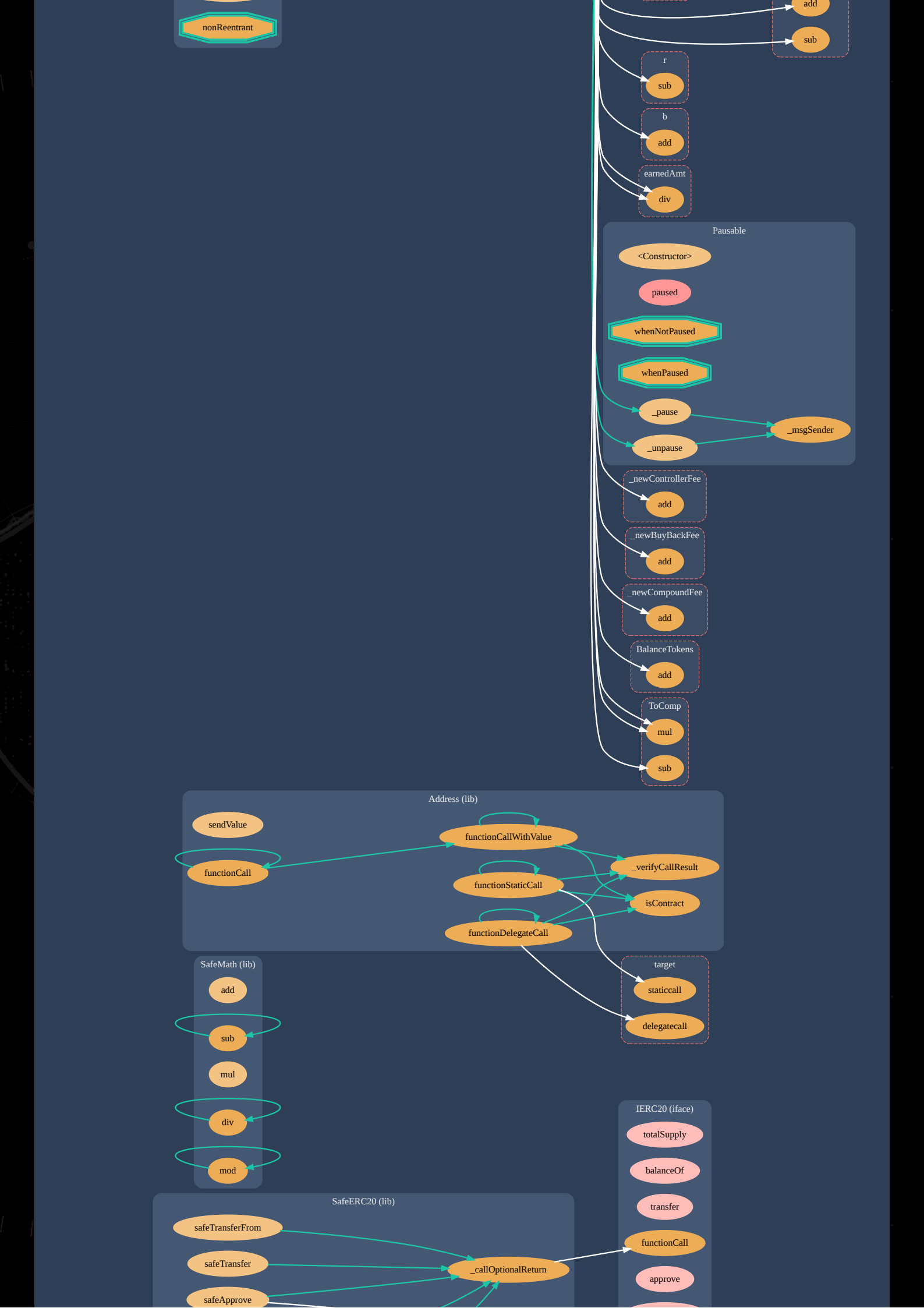

approve
_mint
_beforeTokenTransfer
_setupDecimals
_burn
_totalSupply
add
sub
IERC20 (iface)
totalSupply
balanceOf
transfer
functionCall
approve
transferFrom
allowance
SafeERC20 (lib)
safeTransfer
safeTransferFrom
_callOptionalReturn
safeIncreaseAllowance
safeDecreaseAllowance
safeApprove
SafeMath (lib)
add
sub
mul
div
mod
Context
_msgSender
_msgData

Call Graph

Strategy_V3_PCS.sol




userInfo
getMultiplier
pendingCake
deposit
withdraw
emergencyWithdraw
allowance
transferFrom
increaseAllowance
decreaseAllowance
approve
_setupDecimals
_transfer
_msgSender
_approve
_mint
_burn
_beforeTokenTransfer
_totalSupply
add
sub
Ownable
onlyOwner
owner
<Constructor>
renounceOwnership
transferOwnership
_msgSender
StrategyV3_PCS
<Constructor>
onlyAllowGov
getPricePerFullShare
totalSupply
deposit
balance
farm
withdraw
earn
_farm
available
convertDustToEarned
pause
unpause
setControllerFee
setbuyBackRate
setCompoundFee
setGov
setBuybackRouterAddress
workerCompound
inCaseTokensGetStuck
transferOwnership
buyBack
distributeFees
Context
_msgSender
_msgData
_after
sub
_earnedAmt
sub
mul
_wantAmt
mul
ctrlfee
add
sharesTotal
add
sub
wantLockedTotal
ReentrancyGuard
<Constructor>


nonReentrant
add
sub
r
sub
b
add
earnedAmt
div
Pausable
<Constructor>
paused
whenNotPaused
whenPaused
_pause
_unpause
_msgSender
_newControllerFee
add
_newBuyBackFee
add
_newCompoundFee
add
BalanceTokens
add
ToComp
mul
sub
Address (lib)
sendValue
functionCall
functionCallWithValue
functionStaticCall
functionDelegateCall
_verifyCallResult
isContract
target
staticcall
delegatecall
SafeMath (lib)
add
sub
mul
div
mod
IERC20 (iface)
totalSupply
balanceOf
transfer
functionCall
approve
SafeERC20 (lib)
safeTransferFrom
safeTransfer
safeApprove
_callOptionalReturn


safeIncreaseAllowance
safeDecreaseAllowance
transferFrom
allowance

Analysis

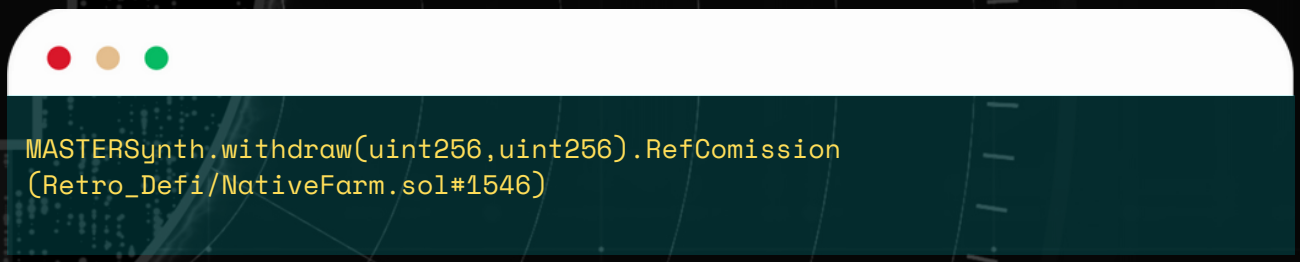
NativeFarm.sol

Issue: Local variable that is not initialized

Severity: **Medium**

Location: L1546

Description: Uninitialized local variable.



Comment: We recommend initializing the variable. If a variable is supposed to be initialized to zero, do so clearly.

Analysis

NativeFarm.sol

Issue: Privileged Ownership of inCaseTokensGetStuck()

Severity: **Medium**

Location: L1653

Description: Function inCaseTokensGetStuck at the aforementioned line allows the owner to drain tokens from the contract.

Comment: The team should assess the design to ensure that the danger of centralization is kept as low as possible by removing the function.,

Analysis

NativeFarm.sol

Issue: Return that is unused

Severity: **Medium**

Location: L1500-1575

Description: The return value of an external call is not stored in a local or state variable.

```
MASTERSynth.deposit(address,uint256,uint256) (Retro_Defi/NativeFarm.sol#1500-1530)
ignores return value by IStrategy(poolInfo[_pid].strat).deposit(msg.sender,_wantAmt)
(Retro_Defi/NativeFarm.sol#1520)
MASTERSynth.withdraw(uint256,uint256) (Retro_Defi/NativeFarm.sol#1533-1598) ignores
return value by IStrategy(poolInfo[_pid].strat).withdraw(msg.sender,_wantAmt)
(Retro_Defi/NativeFarm.sol#1575)
```

Comment: We recommend making sure that all the return values of the function calls are used.

Analysis

NativeFarm.sol

Issue: Division before multiplication

Severity: Low

Location: L1533-1598

Description: Integer division may result in a truncation. As a result, executing a multiplication before division can help to minimize accuracy loss in some cases.

```
MASTERSynth.withdraw(uint256,uint256) (Retro_Defi/NativeFarm.sol#1533-1598)
performs a multiplication on the result of a division:
 -RefComission = profitCheck.mul(REFERRAL_PERCENT).div(PERCENTS_DIVIDER)
(Retro_Defi/NativeFarm.sol#1552-1554)
 -RefComission = (RefComission.mul(totalSupply)).div(wantLockedTotal)
(Retro_Defi/NativeFarm.sol#1555-1557)
```

Comment: We recommend to consider doing multiplication first and then division.

Analysis

NativeFarm.sol

Issue: State variables that could be declared constant

Severity: Lowest / Code Style / Optimized Practice

Location: L1441-1445

Description: In order to save gas, constant state variables could be declared constant.

```
MASTERSynth.REFERRAL_PERCENT (Retro_Defi/NativeFarm.sol#1441) should be
constant
MASTERSynth.startBlock (Retro_Defi/NativeFarm.sol#1445) should be constant
```

Comment:

We recommend adding the constant attributes to state variables that do not change.

Analysis

NativeFarm.sol

Issue: Possibility to declare public function as external.

Severity: Lowest / Code Style / Optimized Practice

Location: L1464-1470, L1600-1603, L1605-1608, L1626-1633, L1635-1642, L1644-1651, L1653-1658

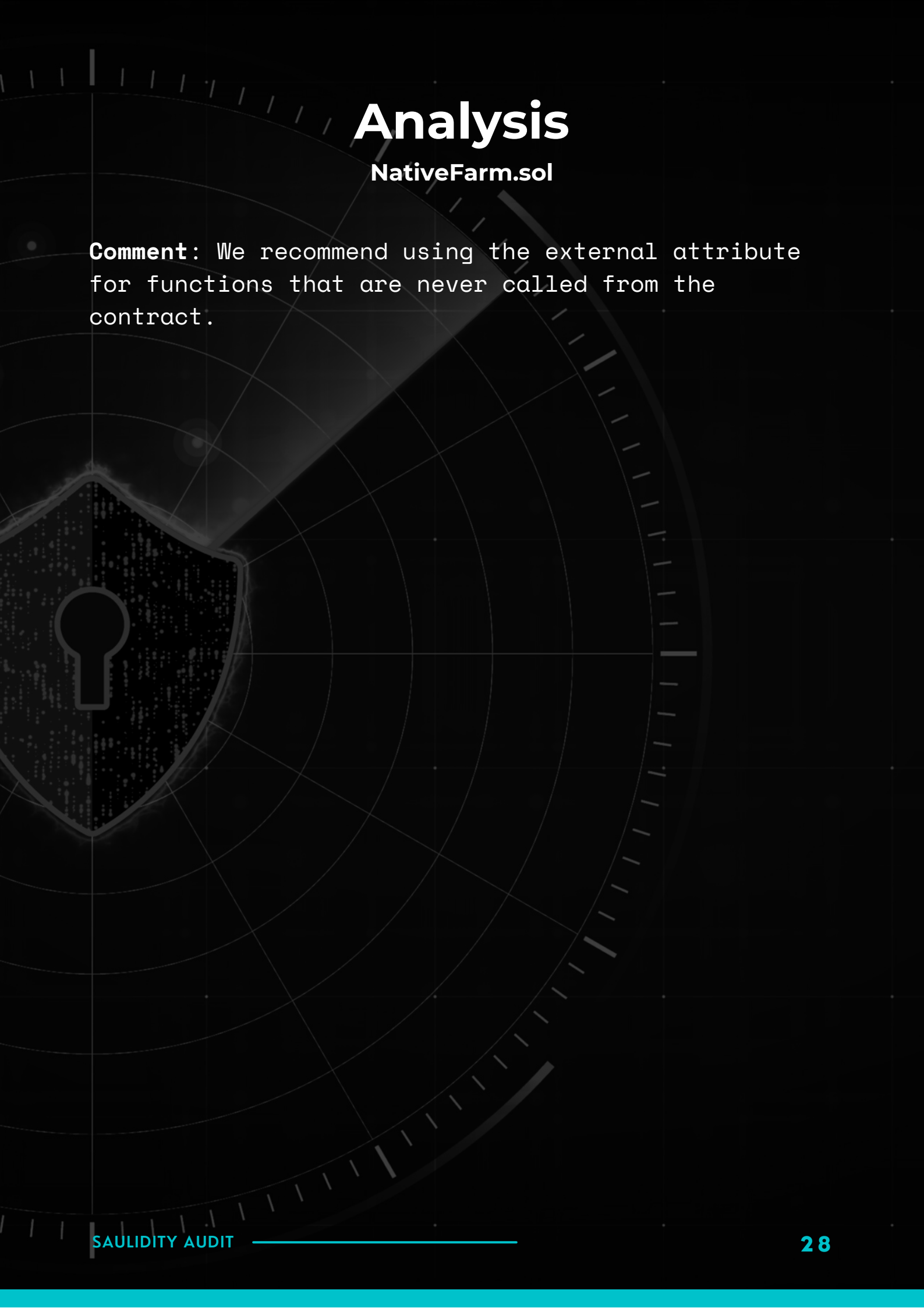
Description: In order to save gas, public functions that are never called by the contract could be declared external.

```
(IERC20,IERC20,address) should be declared external:
 - MASTERSynth.add(IERC20,IERC20,address) (Retro_Defi/NativeFarm.sol#1464-1470)
withdrawAll(uint256) should be declared external:
 - MASTERSynth.withdrawAll(uint256) (Retro_Defi/NativeFarm.sol#1600-1603)
depositAll(address,uint256) should be declared external:
 - MASTERSynth.depositAll(address,uint256) (Retro_Defi/NativeFarm.sol#1605-1608)
getReferral(address,uint256) should be declared external:
 - MASTERSynth.getReferral(address,uint256) (Retro_Defi/NativeFarm.sol#1626-1633)
getReferralCount(address,uint256) should be declared external:
 - MASTERSynth.getReferralCount(address,uint256) (Retro_Defi/NativeFarm.sol#1635-1642)
getReferralRewardTotal(address,uint256) should be declared external:
 - MASTERSynth.getReferralRewardTotal(address,uint256) (Retro_Defi/NativeFarm.sol#1644-1651)
inCaseTokensGetStuck(address,uint256) should be declared external:
 - MASTERSynth.inCaseTokensGetStuck(address,uint256) (Retro_Defi/NativeFarm.sol#1653-1658)
```

Analysis

NativeFarm.sol

Comment: We recommend using the external attribute for functions that are never called from the contract.



Analysis

NativeFarm.sol

Issue: Gas optimization

Severity: Lowest / Code Style / Optimized Practice

Location: General

Description:

The amount of gas needed to run the code depends on how large the pool is; if the pool is too large, a denial-of-service event may result in gas consumption that is higher than the block gas limit.

Comment: Limit the pool's size or update in a consistent manner.

Analysis

Strategy_V3_PCS.sol

Issue: Local variable that is not initialized

Severity: **Medium**

Location: L1855-1856

Description: Uninitialized local variable.

```
StrategyV3_PCS.distributeFees(uint256).compfee (Retro_Defi/Strategy_V3_PCS.sol#1856)
StrategyV3_PCS.distributeFees(uint256).ctrlfee (Retro_Defi/Strategy_V3_PCS.sol#1855)
```

Comment: We recommend initializing the variable. If a variable is supposed to be initialized to zero, do so clearly.

Analysis

Strategy_V3_PCS.sol

Issue: Privileged Ownership of inCaseTokensGetStuck()

Severity: **Medium**

Location: L1976

Description: Function inCaseTokensGetStuck at the aforementioned line allows the owner to drain tokens from the contract.

Comment: The team should assess the design to ensure that the danger of centralization is kept as low as possible by removing the function.,

Analysis

Strategy_V3_PCS.sol

Issue: Return that is unused

Severity: **Medium**

Location: L1695-1785

Description: The return value of an external call is not stored in a local or state variable.

```
StrategyV3_PCS.earn() (Retro_Defi/Strategy_V3_PCS.sol#1695-1785) ignores return value
by
IXRouter02(uniRouterAddress).addLiquidity(token0Address,token1Address,token0Amt,token1
Amt,0,0,address(this),now + routerDeadlineDuration)
(Retro_Defi/Strategy_V3_PCS.sol#1770-1779)
```

Comment: We recommend making sure that all the return values of the function calls are used.

Analysis

Strategy_V3_PCS.sol

Issue: Missing event for access control

Severity: Low

Location: L1951-1953

Description: Events that are missing for critical access control parameters.

```
StrategyV3_PCS.setGov(address) (Retro_Defi/Strategy_V3_PCS.sol#1951-1953) should emit an event for:
 - govAddress = _govAddress (Retro_Defi/Strategy_V3_PCS.sol#1952)
```

Comment: It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

Analysis

Strategy_V3_PCS.sol

Issue: Missing zero address check

Severity: Low

Location: L1951-1952, L1955-1959

Description: Lack of zero address validation.

```
StrategyV3_PCS.setGov(address)._govAddress (Retro_Defi/Strategy_V3_PCS.sol#1951) lacks
a zero-check on :
  - govAddress = _govAddress (Retro_Defi/Strategy_V3_PCS.sol#1952)
StrategyV3_PCS.setBuybackRouterAddress(address)._buybackRouterAddress
(Retro_Defi/Strategy_V3_PCS.sol#1955) lacks a zero-check on :
  - buybackRouterAddress = _buybackRouterAddress (Retro_Defi/Strategy_V3_PCS.sol#1959)
```

Comment: We recommend checking that the address is not zero.

Analysis

Strategy_V3_PCS.sol

Issue: Missing Events Arithmatic

Severity: Low

Location: L1602-1638, L1654-1689, L1927-1933, L1935-1941, L1943-1949

Description: Events for critical arithmetic parameters are missing.

```
StrategyV3_PCS.deposit(address,uint256) (Retro_Defi/Strategy_V3_PCS.sol#1602-1638)
should emit an event for:
 - wantLockedTotal = wantLockedTotal.add(_wantAmt)
(Retro_Defi/Strategy_V3_PCS.sol#1634)
StrategyV3_PCS.withdraw(address,uint256) (Retro_Defi/Strategy_V3_PCS.sol#1654-1689)
should emit an event for:
 - wantLockedTotal = wantLockedTotal.sub(r) (Retro_Defi/Strategy_V3_PCS.sol#1684)
StrategyV3_PCS.setControllerFee(uint256) (Retro_Defi/Strategy_V3_PCS.sol#1927-1933)
should emit an event for:
 - CONTROLLER_FEE = _newControllerFee (Retro_Defi/Strategy_V3_PCS.sol#1932)
StrategyV3_PCS.setbuyBackRate(uint256) (Retro_Defi/Strategy_V3_PCS.sol#1935-1941)
should emit an event for:
 - BUYBACK_FEE = _newBuyBackFee (Retro_Defi/Strategy_V3_PCS.sol#1940)
StrategyV3_PCS.setCompoundFee(uint256) (Retro_Defi/Strategy_V3_PCS.sol#1943-1949)
should emit an event for:
 - COMPOUND_FEE = _newCompoundFee (Retro_Defi/Strategy_V3_PCS.sol#1948)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-
events-arithmetic
```

Comment: We recommend emitting an event for critical parameter changes.

Analysis

Strategy_V3_PCS.sol

Issue: Possibility to declare public function as external.

Severity: **Lowest / Code Style / Optimized Practice**

Location: L596-1599 L1640-1642, L1695-1785, L1874-1917, L1919-1921, L1927-1933, L1935-1941, L1943-1949, L1951-1953, L1955-1960, L1976-1984

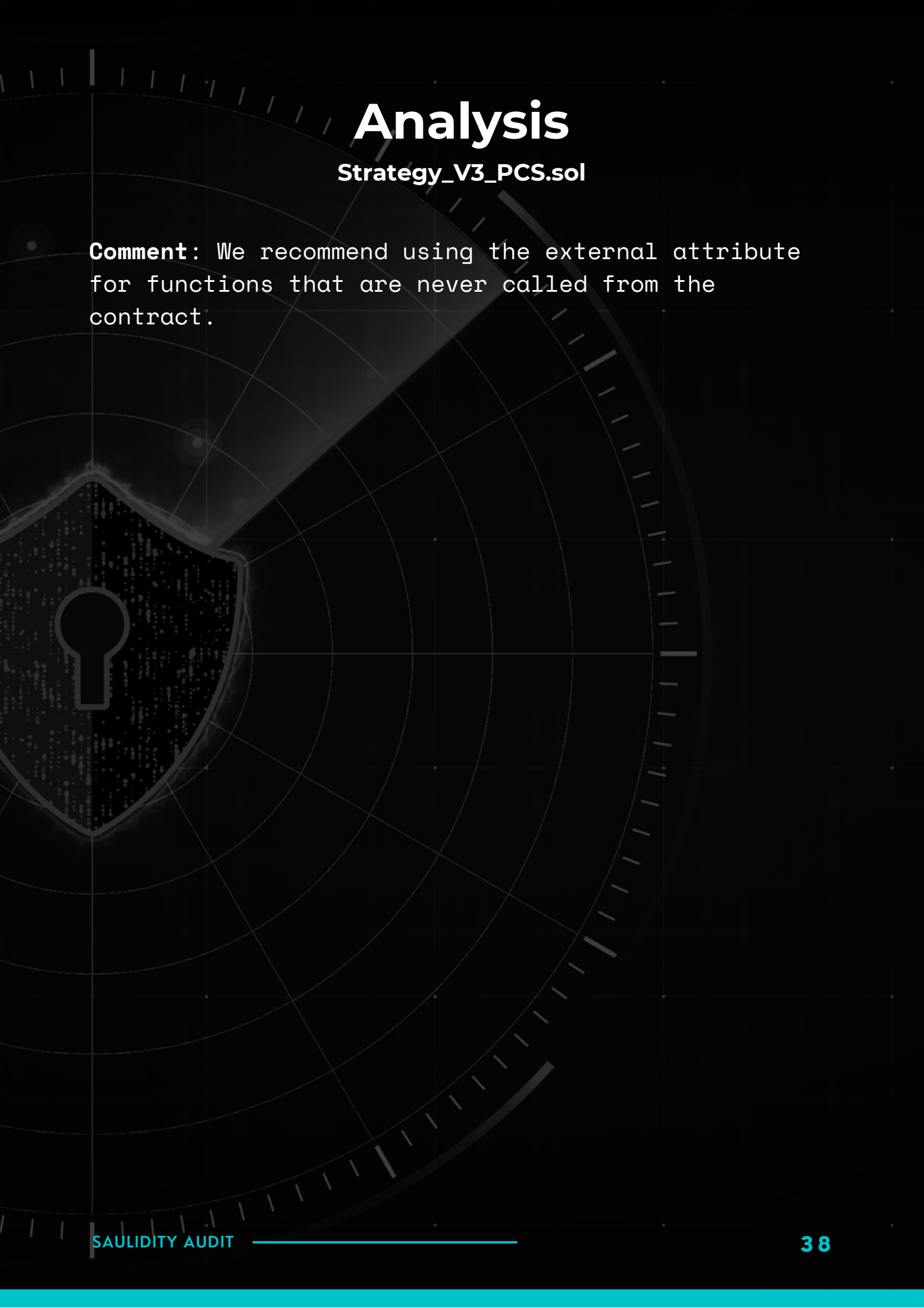
Description: In order to save gas, public functions that are never called by the contract could be declared external.

```
getPricePerFullShare() should be declared external:
 - StrategyV3_PCS.getPricePerFullShare() (Retro_Defi/Strategy_V3_PCS.sol#1596-1599)
farm() should be declared external:
 - StrategyV3_PCS.farm() (Retro_Defi/Strategy_V3_PCS.sol#1640-1642)
earn() should be declared external:
 - StrategyV3_PCS.earn() (Retro_Defi/Strategy_V3_PCS.sol#1695-1785)
convertDustToEarned() should be declared external:
 - StrategyV3_PCS.convertDustToEarned() (Retro_Defi/Strategy_V3_PCS.sol#1874-1917)
pause() should be declared external:
 - StrategyV3_PCS.pause() (Retro_Defi/Strategy_V3_PCS.sol#1919-1921)
setControllerFee(uint256) should be declared external:
 - StrategyV3_PCS.setControllerFee(uint256) (Retro_Defi/Strategy_V3_PCS.sol#1927-1933)
setbuyBackRate(uint256) should be declared external:
 - StrategyV3_PCS.setbuyBackRate(uint256) (Retro_Defi/Strategy_V3_PCS.sol#1935-1941)
setCompoundFee(uint256) should be declared external:
 - StrategyV3_PCS.setCompoundFee(uint256) (Retro_Defi/Strategy_V3_PCS.sol#1943-1949)
setGov(address) should be declared external:
 - StrategyV3_PCS.setGov(address) (Retro_Defi/Strategy_V3_PCS.sol#1951-1953)
setBuybackRouterAddress(address) should be declared external:
 - StrategyV3_PCS.setBuybackRouterAddress(address)
(Retro_Defi/Strategy_V3_PCS.sol#1955-1960)
workerCompound() should be declared external:
 - StrategyV3_PCS.workerCompound() (Retro_Defi/Strategy_V3_PCS.sol#1962-1974)
inCaseTokensGetStuck(address,uint256,address) should be declared external:
 - StrategyV3_PCS.inCaseTokensGetStuck(address,uint256,address)
(Retro_Defi/Strategy_V3_PCS.sol#1976-1984)
```

Analysis

Strategy_V3_PCS.sol

Comment: We recommend using the external attribute for functions that are never called from the contract.



Analysis

Strategy_V3_PCS.sol

Issue: State variables that could be declared constant

Severity: Lowest / Code Style / Optimized Practice

Location: L1468-1489

Description: In order to save gas, constant state variables could be declared constant.

```
StrategyV3_PCS.FEE_MAX (Retro_Defi/Strategy_V3_PCS.sol#1487) should be
constant
StrategyV3_PCS.PERCENT_DIVIDER (Retro_Defi/Strategy_V3_PCS.sol#1489) should be
constant
StrategyV3_PCS.isAutoComp (Retro_Defi/Strategy_V3_PCS.sol#1460) should be
constant
StrategyV3_PCS.isSingleVault (Retro_Defi/Strategy_V3_PCS.sol#1459) should be
constant
StrategyV3_PCS.routerDeadlineDuration (Retro_Defi/Strategy_V3_PCS.sol#1472)
should be constant
StrategyV3_PCS.uniRouterAddress (Retro_Defi/Strategy_V3_PCS.sol#1468-1469)
should be constant
StrategyV3_PCS.wbnbAddress (Retro_Defi/Strategy_V3_PCS.sol#1474) should be
constant
```

Comment:
We recommend adding the constant attributes to state variables that do not change.

Privileges and Notes

-Different tokens kinds might be added by the project team for staking.

-Use of SafeMath (or other comparable safe functions) to stop overflow problems.

-Making a deposit to the contract or ending it early are both free of fee.

-An emergency withdraw event is present but it is not emitted and there is no function.

-The team can pause deposits into the strategy contract and rebalance them without any delay.

-Tokens from users will be sent to team-created strategy contract in order to generate yield.

Testing Standards

The goal of the audit was to find any potential smart contract security problems and vulnerabilities.
The information in this report should be used to understand the smart contract's risk exposure and as a guide to improving the smart contract's security posture by addressing the concerns that were discovered.

The blockchain platform is used to deploy and execute smart contracts. The platform, its programming language, and other smart contract-related applications all have vulnerabilities that may be exploited. As a result, the audit cannot ensure the audited smart contract(s) explicit security.Audits can't make statements or warranties on security of the code.It also cannot be deemed an adequate assessment of the code's utility and safety, bug-free status, or any statements of the smart contract.
While we did our best in completing the study and publishing this report, it is crucial to emphasize that you should not rely only on it; we advocate all projects doing many independent audits and participating in a public bug bounty program to assure smart contract security.

Testing Standards

1. Gather all relevant data.

2. Perform a preliminary visual examination of all documents and contracts.

3. Find security holes with specialist tools & manual review with independent experts.

4. Create and distribute a report.

SAULIDITY

Smart Contract
Audit

saulidity.com
Saulidity
@Saulidity