Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

# SMART CONTRACT

## Security Audit Report

| | |
|---|---|
| Customer: | ShibaNova Team |
| Website: | https://shibanova.io |
| Platform: | Binance Smart Chain |
| Language: | Solidity |
| Date: | June 14th, 2021 |

# Table of contents

Introduction ..... 4
Project Background ..... 4
Audit Scope ..... 4
Claimed Smart Contract Features ..... 6
Audit Summary ..... 7
Technical Quick Stats ..... 8
Code Quality ..... 9
Documentation ..... 9
Use of Dependencies ..... 9
AS-IS overview ..... 10
Severity Definitions ..... 25
Audit Findings ..... 25
Conclusion ..... 29
Our Methodology ..... 30
Disclaimers ..... 32
Appendix
- Code Flow Diagram ..... 33
- Slither Report Log ..... 42

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO PUBLIC AFTER ISSUES ARE RESOLVED.


**Email: audit@EtherAuthority.io**

# Introduction

EtherAuthority was contracted by the ShibaNova team to perform the Security audit of the ShibaNova Protocol smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on June 14th, 2021.

**The purpose of this audit was to address the following:**

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

# Project Background

ShibaNova is a daily dividend AMM (Automated Market Maker) & low fee DEX on the Binance Smart Chain. NOVA and sNOVA are their tokens as part of the Defi ecosystem.

# Audit Scope

| **Name** | **Code Review and Security Analysis Report for ShibaNova Protocol Smart Contracts** |
|---|---|
| **Platform** | **BSC / Solidity** |
| **File 1** | FeeManager.sol |
| **File 1 MD5 Hash** | 9216BBE9AD5B12BB158894D5AE572048 |
| **File 1 Online Code** | https://testnet.bscscan.com/address/0x9A50e1dCE6B5E4E6CDEab09A7aCb5068f8e61D91#code |
| **File 2** | MasterShiba.sol |
| **File 2 MD5 Hash** | 3FDCF9D1F05669A7F0182374C3F544C1 |
| **File 2 Online Code** | https://testnet.bscscan.com/address/0xcF6541a23547062468fDc3088677d762750C81bA#code |
| **File 3** | NovaToken.sol |
| **File 3 MD5 Hash** | 259F33CEB1213BCA04DEE83FCE6A09C3 |
| **File 3 Online Code** | https://testnet.bscscan.com/address/0x9F249308497344A9e23e2c269703aD92c80b9F86#code |

| | |
|---|---|
| **File 4** | SNovaToken.sol |
| **File 4 MD5 Hash** | 79776991EFBFC3AE577C046F03E2D87E |
| **File 4 Online Code** | https://testnet.bscscan.com/address/0x465d9BD104c7d7Baa98bF4f0EDF471b0d92D4e1c#code |
| **File 5** | Timelock.sol |
| **File 5 MD5 Hash** | 6A370EB579AC09EAF8265AB347655BD5 |
| **File 5 Online Code** | https://testnet.bscscan.com/address/0xb2B8C7fb1d58Abe082e105f52833eed59637501d#code |
| **File 6** | ShibaBonusAggregator.sol |
| **File 6 MD5 Hash** | 1C8A8D5286C4111643393E68A4FB3F3C |
| **File 6 Online Code** | https://testnet.bscscan.com/address/0x0bD0445d91F37076c43254a162e611198A5De02d#code |
| **File 7** | ShibaFactory.sol |
| **File 7 MD5 Hash** | 359AE6FC99D0EB67B7E71EBE14F21635 |
| **File 7 Online Code** | https://testnet.bscscan.com/address/0xC42c94349F9Cea97e1347EA22108FAc2931F6De7#code |
| **File 8** | ShibaPair.sol |
| **File 8 MD5 Hash** | 296DC25697CCCC1C07C6A211B3FABF25 |
| **File 8 Online Code** | https://testnet.bscscan.com/address/0xC42c94349F9Cea97e1347EA22108FAc2931F6De7#code |
| **File 9** | ShibaMoneyPot.sol |
| **File 9 MD5 Hash** | EB106964853AA66734DFE26D32666081 |
| **File 9 Online Code** | https://testnet.bscscan.com/address/0x745b5045f725a53bc0217568bAf8649F88d2De82#code |
| **File 10** | ShibaRouter.sol |
| **File 10 MD5 Hash** | 9DB3643926D8C73E93A49426EC13C54E |
| **File 10 Online Code** | https://testnet.bscscan.com/address/0x1720bA6c96c7E68Aa8417aBf55D6Bb08a0EB965A#code |

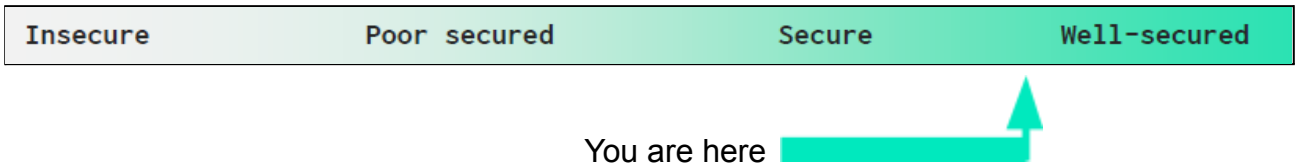
**PS: There are many libraries and external imported contracts, which are not part of the audit scope, and thus not considered in this audit SOP.**

## Claimed Smart Contract Features

| Claimed Feature Detail | Our Observation |
|---|---|
| FeeManager contract:<br>• Distribute fee to the moneypot and dev wallet<br>• Owner can set a money pot wallet, router wallet, team address, etc. | **YES, This is valid.** |
| MasterShiba contract:<br>• Initial emission rate: 1 Nova.<br>• Minimum emission rate: 0.5 Nova.<br>• Emission reduction rate per period in basis points: 2% | **YES, This is valid.** |
| Nova Token:<br>• Initial Supply: 0<br>• Minting: by MasterShiba only.<br>• 2% of every transfer amount will be burned. | **YES, This is valid.** |
| sNova Token:<br>• Default swap Penalty Max Period: 24 hrs<br>• Default swap Penalty Max Per SNova: 30% | **YES, This is valid. This is set at the time of contract deployment.** |
| Timelock contract:<br>• Grace period: 14 days<br>• Minimum delay: 0<br>• Maximum delay: 30 days | **YES, This is valid.** |

# Audit Summary

According to the **standard** audit assessment, Customer`s solidity smart contracts are **secured**. These contracts also have owner functions (described in the centralization section below), which does not make everything 100% decentralized. Thus, the owner must execute those smart contract functions as per the business plan.




We used various tools like MythX, Slither and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

**We found 0 critical, 0 high, 0 medium, 2 low and some very low level issues in smart contracts which are described in the audit finding section. We also revised the code and these issues are fixed / acknowledged by the ShibaNova team.**

## Technical Quick Stats:

| Main Category | Subcategory | Result |
|---|---|---|
| Contract Programming | Solidity version not specified | Passed |
| | Solidity version too old | Moderated |
| | Integer overflow/underflow | Passed |
| | Function input parameters lack of check | Passed |
| | Function input parameters check bypass | Passed |
| | Function access control lacks management | Passed |
| | Critical operation lacks event log | Moderated |
| | Human/contract checks bypass | Passed |
| | Random number generation/use vulnerability | Passed |
| | Fallback function misuse | Passed |
| | Race condition | Passed |
| | Logical vulnerability | Passed |
| | Other programming issues | Passed |
| Code Specification | Function visibility not explicitly declared | Passed |
| | Var. storage location not explicitly declared | Passed |
| | Use keywords/functions to be deprecated | Passed |
| | Other code specification issues | Passed |
| Gas Optimization | “Out of Gas” Issue | Passed |
| | High consumption ‘for/while’ loop | Moderated |
| | High consumption ‘storage’ storage | Passed |
| | Assert() misuse | Passed |
| Business Risk | The maximum limit for mintage not set | Passed |
| | “Short Address” Attack | Passed |
| | “Double Spend” Attack | Passed |

**Overall Audit Result:** **PASSED**

## Code Quality

This audit scope has 10 smart contracts. These smart contracts also contain Libraries, Smart contracts inherits and Interfaces. These are compact and well written contracts.

The libraries in the ShibaNova protocol are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the ShibaNova protocol.

The ShibaNova team has **not** provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Some code parts are **not well** commented on smart contracts.

## Documentation

We were given ShibaNova smart contracts code in the form of the files. The hashes of that code are mentioned above in the table.

As mentioned above, some code parts are **not well** commented. So it is difficult to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official website https://shibanova.io which provided rich information about the project architecture and tokenomics.

## Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects. And their core code blocks are written well.

Apart from libraries, its functions are used in external smart contract calls.

# AS-IS overview

## FeeManager.sol

**(1) Interface**

(a) IMoneyPot

(b) IShibaPair

(c) IShibaRouter01

(d) IShibaRouter02

(e) IBEP20

**(2) Inherited contracts**

(a) Context

(b) Ownable

**(3) Usages**

(a) using SafeMath for uint256;

(b) using SafeBEP20 for IBEP20;

(c) using Address for address;

**(4) Events**

(a) event Approval(address indexed owner, address indexed spender, uint value);

(b) event Transfer(address indexed from, address indexed to, uint value);

(c) event Mint(address indexed sender, uint amount0, uint amount1);

(d) event Burn(address indexed sender, uint amount0, uint amount1, address indexed to);

(e) event Swap( address indexed sender,uint amount0In,uint amount1In, uint amount0Out,uint amount1Out, address indexed to);

(f) event Sync(uint112 reserve0, uint112 reserve1);

(g) event Transfer(address indexed from, address indexed to, uint256 value);

(h) event Approval(address indexed owner, address indexed spender, uint256 value);

(i) event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

## (6) Functions

| Sl. | Functions | Type | Observation | Conclusion |
|---|---|---|---|---|
| 1 | removeLiquidityToToken | external | access only Owner | No Issue |
| 2 | swapBalanceToToken | external | access only Owner | No Issue |
| 3 | swapToToken | external | access only Owner | No Issue |
| 4 | updateShares | external | access only Owner | No Issue |
| 5 | setTeamAddr | external | access only Owner | No Issue |
| 6 | setupRouter | external | access only Owner | No Issue |
| 7 | setupMoneyPot | external | access only Owner | No Issue |
| 8 | distributeFee | external | access only Owner | Keep array length limited |
| 9 | owner | read | access only Owner | No Issue |
| 10 | renounceOwnership | write | access only Owner | No Issue |
| 11 | transferOwnership | write | access only Owner | No Issue |
| 12 | _transferOwnership | internal | Passed | No Issue |
| 13 | _msgSender | internal | Passed | No Issue |
| 14 | _msgData | internal | Passed | No Issue |

## MasterShiba.sol

**(1) Interface**

(a) IBEP20

(b) IMasterBonus

(c) IBonusAggregator

**(2) Inherited contracts**

(a) Ownable

(b) IMasterBonus

(c) IBonusAggregator

(d) ShibaBonusAggregator

(e) Context

(f) ShibaBEP20

(g) IBEP20

**(3) Struct**

(a) UserInfo: Information about UserInfo.

(b) PoolInfo: Information about PoolInfo.

**(4) Usages**

(a) using SafeMath for uint256;

(b) using SafeBEP20 for ShibaBEP20;

(c) using SafeBEP20 for IBEP20;

(d) using Address for address;

**(5) Events**

(a) event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

(b) event Transfer(address indexed from, address indexed to, uint256 value);

(c) event Approval(address indexed owner, address indexed spender, uint256 value);

(d) event Deposit(address indexed user, uint256 indexed pid, uint256 amount);

(e) event Withdraw(address indexed user, uint256 indexed pid, uint256 amount);

(f) event EmergencyWithdraw(address indexed user, uint256 indexed pid, uint256 amount);

(g) event EmissionRateUpdated(address indexed caller, uint256 previousAmount, uint256 newAmount);

## (6) Functions

| Sl. | Functions | Type | Observation | Conclusion |
|---|---|---|---|---|
| 1 | poolLength | external | Passed | No Issue |
| 2 | userBonus | read | Passed | No Issue |
| 3 | getMultiplier | write | Passed | No Issue |
| 4 | add | external | access only Owner | No Issue |
| 5 | set | external | access only Owner | No Issue |
| 6 | pendingNova | external | Passed | No Issue |
| 7 | updateEmissionRate | internal | Passed | No Issue |
| 8 | massUpdatePools | write | Passed | Keep array length limited |
| 9 | updatePool | write | Passed | No Issue |
| 10 | updateUserBonus | external | Passed | No Issue |
| 11 | deposit | external | Passed | No Issue |
| 12 | withdraw | external | Passed | No Issue |
| 13 | emergencyWithdraw | external | Passed | No Issue |
| 14 | getPoolInfo | external | Passed | No Issue |
| 15 | safeNovaTransfer | internal | Passed | No Issue |
| 16 | safeSNovaTransfer | internal | Passed | No Issue |
| 17 | dev | external | Passed | No Issue |
| 18 | setFeeAddress | external | access only Owner | No Issue |
| 19 | updateMinimumEmissionRate | external | access only Owner | No Issue |
| 20 | owner | read | Passed | No Issue |
| 21 | renounceOwnership | write | access only Owner | No Issue |
| 22 | transferOwnership | write | access only Owner | No Issue |
| 23 | _transferOwnership | internal | Passed | No Issue |
| 24 | getOwner | external | Passed | No Issue |

## NovaToken.sol

### (1) Interface

(a) IBEP20

### (2) Inherited contracts

(a) ShibaBEP20

(b) Context

(c) IBEP20

(d) Ownable

### (3) Usages

(a) using SafeMath for uint256;

(b) using Address for address;

### (4) Events

(a) event Transfer(address indexed from, address indexed to, uint256 value);

(b) event Approval(address indexed owner, address indexed spender, uint256 value);

(c) event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

### (5) Functions

| Sl. | Functions | Type | Observation | Conclusion |
|---|---|---|---|---|
| 1 | isOwner | read | Owner must be MasterShiba | No Issue |
| 2 | isSNova | internal | Passed | No Issue |
| 3 | setupSNova | external | Passed | No Issue |
| 4 | mint | external | Only should be by MasterShiba | No Issue |
| 5 | _transfer | internal | Passed | No Issue |
| 6 | getOwner | external | Passed | No Issue |
| 7 | name | read | Passed | No Issue |
| 8 | decimals | read | Passed | No Issue |
| 9 | symbol | read | Passed | No Issue |
| 10 | totalSupply | read | Passed | No Issue |
| 11 | burnSupply | read | Passed | No Issue |
| 12 | balanceOf | read | Passed | No Issue |
| 13 | transfer | write | Passed | No Issue |

| | | | | |
|---|---|---|---|---|
| **14** | allowance | read | Passed | No Issue |
| **15** | approve | write | Passed | No Issue |
| **16** | transferFrom | write | Passed | No Issue |
| **17** | increaseAllowance | write | Passed | No Issue |
| **18** | decreaseAllowance | write | Passed | No Issue |
| **19** | _transfer | internal | 2% burned | No Issue |
| **20** | mint | external | Passed | No Issue |
| **21** | _mint | internal | Passed | No Issue |
| **22** | _burn | internal | Passed | No Issue |
| **23** | _approve | internal | Passed | No Issue |
| **25** | _burnFrom | internal | Passed | No Issue |
| **26** | owner | read | Passed | No Issue |
| **27** | renounceOwnership | write | access only Owner | No Issue |
| **28** | transferOwnership | write | access only Owner | No Issue |
| **29** | _transferOwnership | internal | Passed | No Issue |
| **30** | _msgSender | internal | Passed | No Issue |
| **31** | _msgData | internal | Passed | No Issue |

## SNovaToken.sol

**(1) Interface**

(a) IBEP20

(b) IMoneyPot

**(2) Inherited contracts**

(a) Ownable

(b) ShibaBEP20

(c) Context

(d) IBEP20

**(3) Usages**

(a) using SafeMath for uint256;

(b) using Address for address;

**(4) Struct**

(a) HolderInfo: Details about HolderInfo;

(b) Checkpoint: Details about Checkpoint;

**(5) Events**

(a) event Transfer(address indexed from, address indexed to, uint256 value);

(b) event Approval(address indexed owner, address indexed spender, uint256 value);

(c) event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

(d) event DelegateChanged(address indexed delegator, address indexed fromDelegate, address indexed toDelegate);

(e) event DelegateVotesChanged(address indexed delegate, uint previousBalance, uint newBalance);

**(6) Functions**

| Sl. | Functions | Type | Observation | Conclusion |
|---|---|---|---|---|
| 1 | setupNova | external | access only Owner | No Issue |
| 2 | setupMoneyPot | external | access only Owner | No Issue |
| 3 | getPenaltyPercent | read | Passed | No Issue |
| 4 | swapToNova | external | Passed | No Issue |
| 5 | previewSwapNovaExpectedAmount | external | Passed | No Issue |
| 6 | _swapNovaAmount | internal | Passed | No Issue |
| 7 | _getAvgTransactionBlock | internal | Passed | No Issue |
| 8 | mint | external | access only Owner | No Issue |
| 9 | _transfer | internal | Passed | No Issue |
| 10 | delegates | read | Passed | No Issue |
| 11 | delegate | write | Passed | No Issue |
| 12 | delegateBySig | write | Handle signature securely | No Issue |
| 13 | getCurrentVotes | read | Passed | No Issue |
| 14 | getPriorVotes | read | Possibility of high gas consumption | Keep loop iterations limited |
| 15 | _delegate | internal | Passed | No Issue |
| 16 | _moveDelegates | internal | Passed | No Issue |
| 17 | _writeCheckpoint | internal | Passed | No Issue |
| 18 | safe32 | internal | Passed | No Issue |
| 19 | getChainId | internal | Passed | No Issue |
| 20 | getOwner | external | Passed | No Issue |
| 21 | name | read | Passed | No Issue |
| 22 | decimals | read | Passed | No Issue |
| 23 | symbol | read | Passed | No Issue |

| | | | | |
|---|---|---|---|---|
| **24** | totalSupply | read | Passed | No Issue |
| **25** | burnSupply | read | Passed | No Issue |
| **26** | balanceOf | read | Passed | No Issue |
| **27** | transfer | write | Passed | No Issue |
| **28** | allowance | read | Passed | No Issue |
| **29** | approve | write | Passed | No Issue |
| **30** | transferFrom | write | Passed | No Issue |
| **31** | increaseAllowance | write | Passed | No Issue |
| **32** | decreaseAllowance | write | Passed | No Issue |
| **33** | _transfer | internal | Passed | No Issue |
| **34** | mint | external | Passed | No Issue |
| **35** | _mint | internal | Passed | No Issue |
| **36** | _burn | internal | Passed | No Issue |
| **37** | _approve | internal | Passed | No Issue |
| **38** | _burnFrom | internal | Passed | No Issue |
| **39** | owner | read | Passed | No Issue |
| **40** | renounceOwnership | write | access only Owner | No Issue |
| **41** | transferOwnership | write | access only Owner | No Issue |
| **42** | _transferOwnership | internal | Passed | No Issue |
| **43** | _msgSender | internal | Passed | No Issue |
| **44** | _msgData | internal | Passed | No Issue |

## Timelock.sol

**(1) Usages**

(a) using SafeMath for uint;

**(2) Events**

(a) event NewAdmin(address indexed newAdmin);

(b) event NewPendingAdmin(address indexed newPendingAdmin);

(c) event NewDelay(uint indexed newDelay);

(d) event CancelTransaction(bytes32 indexed txHash, address indexed target, uint value, string signature, bytes data, uint eta);

(e) event ExecuteTransaction(bytes32 indexed txHash, address indexed target, uint value, string signature, bytes data, uint eta);

(f) event QueueTransaction(bytes32 indexed txHash, address indexed target, uint value, string signature, bytes data, uint eta);

**(6) Functions**

| Sl. | Functions | Type | Observation | Conclusion |
|---|---|---|---|---|
| 1 | setDelay | write | Passed | No Issue |
| 2 | acceptAdmin | write | Passed | No Issue |
| 3 | setPendingAdmin | write | Passed | No Issue |
| 4 | queueTransaction | write | Passed | No Issue |
| 5 | cancelTransaction | write | Passed | No Issue |
| 6 | executeTransaction | write | Passed | No Issue |
| 7 | getBlockTimestamp | read | Passed | No Issue |

## ShibaBonusAggregator.sol

**(1) Interface**

(a) IBonusAggregator

(b) IMasterBonus

**(2) Inherited contracts**

(a) Context

(b) Ownable

(c) IBonusAggregator

**(3) Usages**

(a) using SafeMath for uint256;

**(4) Events**

(a) event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

**(5) Functions**

| Sl. | Functions | Type | Observation | Conclusion |
|---|---|---|---|---|
| 1 | onlyVerifiedContract | modifier | Passed | No Issue |
| 2 | setupMaster | external | access only Owner | No Issue |
| 3 | addOrRemoveContractBonusSource | external | access only Owner | No Issue |

| | | | | |
|---|---|---|---|---|
| **4** | addUserBonusOnFarm | external | access only VerifiedContract | No Issue |
| **5** | removeUserBonusOnFarm | external | access only VerifiedContract | No Issue |
| **6** | getBonusOnFarmsForUser | external | Passed | No Issue |
| **7** | owner | read | Passed | No Issue |
| **8** | onlyOwner | modifier | Passed | No Issue |
| **9** | renounceOwnership | write | access only Owner | No Issue |
| **10** | transferOwnership | write | access only Owner | No Issue |
| **11** | _transferOwnership | internal | Passed | No Issue |
| **12** | _msgSender | internal | Passed | No Issue |
| **13** | _msgData | internal | Passed | No Issue |

## ShibaFactory.sol

**(1) Imports**

(a) IShibaFactory.sol

(b) ShibaPair.sol

**(2) Inherited contracts**

(a) IShibaFactory

**(3) Events**

(a) event PairCreated(address indexed token0, address indexed token1, address pair, uint);

**(4) Functions**

| Sl. | Functions | Type | Observation | Conclusion |
|---|---|---|---|---|
| **1** | owner | read | Passed | No Issue |
| **2** | allPairsLength | read | Passed | No Issue |
| **3** | createPair | write | Passed | No Issue |
| **4** | setFeeTo | write | Passed | No Issue |
| **5** | setFeeToSetter | write | Passed | No Issue |
| **6** | feeAmount | read | Passed | No Issue |
| **7** | setFeeAmount | write | Passed | No Issue |

## ShibaPair.sol

### (1) Imports

(a) IShibaPair.sol

(b) ShibaERC20.sol

(c) Math.sol

(d) UQ112x112.sol

(e) IERC20.sol

(f) IShibaFactory.sol

(g) IShibaCallee.sol

### (2) Inherited contracts

(a) IShibaPair

(b) ShibaERC20

### (3) Usages

(a) using SafeMath for uint;

(b) using UQ112x112 for uint224;

### (4) Events

(a) event Mint(address indexed sender, uint amount0, uint amount1);

(b) event Burn(address indexed sender, uint amount0, uint amount1, address indexed to);

(c) event Swap(address indexed sender, uint amount0In, uint amount1In, uint amount0Out, uint amount1Out, address indexed to );

(d) event Sync(uint112 reserve0, uint112 reserve1);

### (5) Functions

| Sl. | Functions | Type | Observation | Conclusion |
|---|---|---|---|---|
| 1 | getReserves | read | Passed | No Issue |
| 2 | _safeTransfer | private | Passed | No Issue |
| 3 | initialize | write | Passed | No Issue |
| 4 | _update | private | Passed | No Issue |
| 5 | _mintFee | private | Passed | No Issue |
| 6 | mint | write | Passed | No Issue |

| | | | | |
|---|---|---|---|---|
| **7** | burn | write | Passed | No Issue |
| **8** | swap | write | Passed | No Issue |
| **9** | skim | write | Passed | No Issue |
| **10** | sync | write | Passed | No Issue |

## ShibaMoneyPot.sol

**(1) Interface**

(h) IBEP20

**(2) Inherited contracts**

(c) Context

(d) Ownable

(e) IMoneyPot

**(3) Struct**

(a) TokenPot: Information about Token Pot.

(b) UserInfo: Information about User.

**(4) Usages**

(c) using SafeBEP20 for IBEP20;

(d) using SafeMath for uint256;

(e) using Address for address;

**(5) Events**

(e) event Transfer(address indexed from, address indexed to, uint256 value);

(f) event Approval(address indexed owner, address indexed spender, uint256 value);

(g) event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

## (6) Functions

| Sl. | Functions | Type | Observation | Conclusion |
|---|---|---|---|---|
| 1 | getRegisteredToken | external | Passed | No Issue |
| 2 | distributedMoneyPot | external | Passed | No Issue |
| 3 | isDividendsToken | external | Passed | No Issue |
| 4 | updateAddressWithoutReward | external | access only Owner | No Issue |
| 5 | updateFeeManager | external | access only Owner | No Issue |
| 6 | getRegisteredTokenLength | external | Passed | No Issue |
| 7 | getTokenAmountPotFromMoneyPot | external | Passed | No Issue |
| 8 | tokenPerBlock | external | Passed | No Issue |
| 9 | massUpdateMoneyPot | write | keep array length limited | No Issue |
| 10 | updateCurrentMoneyPot | external | Passed | No Issue |
| 11 | getMultiplier | internal | Passed | No Issue |
| 12 | _updateTokenPot | internal | Passed | No Issue |
| 13 | pendingTokenRewardsAmount | external | Passed | No Issue |
| 14 | updateSNovaHolder | external | keep array length limited | No Issue |
| 15 | harvestRewards | external | keep array length limited | No Issue |
| 16 | harvestReward | write | Passed | No Issue |
| 17 | depositRewards | external | Passed | No Issue |
| 18 | depositBonusRewards | external | access only Owner | No Issue |
| 19 | addTokenToRewards | external | access only Owner | No Issue |
| 20 | removeTokenToRewards | external | keep array length limited | No Issue |
| 21 | nextMoneyPotUpdateBlock | external | access only Owner | No Issue |
| 22 | addToPendingFromReserveTokenAmount | external | access only Owner | No Issue |
| 23 | safeTokenTransfer | internal | Passed | No Issue |
| 24 | owner | read | Passed | No Issue |
| 25 | onlyOwner | modifier | Passed | No Issue |
| 26 | renounceOwnership | write | access only Owner | No Issue |
| 27 | transferOwnership | write | access only Owner | No Issue |
| 28 | _transferOwnership | internal | Passed | No Issue |
| 29 | _msgSender | internal | Passed | No Issue |
| 30 | _msgData | internal | Passed | No Issue |

## ShibaRouter.sol

**(1) Interface**

(a) IERC20

(b) IShibaFactory

(c) IShibaPair

(d) IShibaRouter01

(e) IShibaRouter02

(f) IWETH

**(2) Inherited contracts**

(a) IShibaRouter02

**(3) Usages**

(a) using SafeMath for uint;

**(4) Events**

(a) event Approval(address indexed owner, address indexed spender, uint value);

(b) event Transfer(address indexed from, address indexed to, uint value);

(c) event PairCreated(address indexed token0, address indexed token1, address pair, uint);

(d) event Approval(address indexed owner, address indexed spender, uint value);

(e) event Transfer(address indexed from, address indexed to, uint value);

(f) event Mint(address indexed sender, uint amount0, uint amount1);

(g) event Burn(address indexed sender, uint amount0, uint amount1, address indexed to);

(h) event Swap(address indexed sender,uint amount0In,uint amount1In,uint amount0Out,uint amount1Out,address indexed to);

(i) event Sync(uint112 reserve0, uint112 reserve1);

## (5) Functions

| Sl. | Functions | Type | Observation | Conclusion |
|---|---|---|---|---|
| 1 | ensure | modifier | Passed | No Issue |
| 2 | receive | external | Passed | No Issue |
| 3 | _addLiquidity | internal | Passed | No Issue |
| 4 | addLiquidity | external | Passed | No Issue |
| 5 | addLiquidityETH | external | Passed | No Issue |
| 6 | removeLiquidity | write | Passed | No Issue |
| 7 | removeLiquidityETH | write | Passed | No Issue |
| 8 | removeLiquidityWithPermit | external | Passed | No Issue |
| 9 | removeLiquidityETHWithPermit | external | Passed | No Issue |
| 10 | removeLiquidityETHSupportingFeeOnTransferTokens | write | Passed | No Issue |
| 11 | removeLiquidityETHWithPermitSupportingFeeOnTransferTokens | external | Passed | No Issue |
| 12 | _swap | internal | Passed | No Issue |
| 13 | swapExactTokensForTokens | external | Passed | No Issue |
| 14 | swapTokensForExactTokens | external | Passed | No Issue |
| 15 | swapExactETHForTokens | external | Passed | No Issue |
| 16 | swapTokensForExactETH | external | Passed | No Issue |
| 17 | swapExactTokensForETH | external | Passed | No Issue |
| 18 | swapETHForExactTokens | external | Passed | No Issue |
| 19 | _swapSupportingFeeOnTransferTokens | internal | Passed | No Issue |
| 20 | swapExactTokensForTokensSupportingFeeOnTransferTokens | external | Passed | No Issue |
| 21 | swapExactETHForTokensSupportingFeeOnTransferTokens | external | Passed | No Issue |
| 22 | swapExactTokensForETHSupportingFeeOnTransferTokens | external | Passed | No Issue |
| 23 | quote | write | Passed | No Issue |
| 24 | getAmountOut | write | Passed | No Issue |
| 25 | getAmountIn | write | Passed | No Issue |
| 26 | getAmountsOut | read | Passed | No Issue |
| 27 | getAmountsIn | read | Passed | No Issue |

# Severity Definitions

| Risk Level | Description |
|---|---|
| Critical | Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc. |
| High | High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions |
| Medium | Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose |
| Low | Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution |
| Lowest / Code Style / Best Practice | Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored. |

# Audit Findings

## Critical

No critical severity vulnerabilities were found.

## High

No high severity vulnerabilities were found.

## Medium

No Medium severity vulnerabilities were found.

## Low

(1) Input validation missing in MasterShiba.sol

```
// Add a new lp to the pool. Can only be called by the owner.
function add(uint256 _allocPoint, IBEP20 _lpToken, uint256 _depositFeeBP, bool _isSNovaRewards, bool
    require(_depositFeeBP <= 400, "add: invalid deposit fee basis points");
    if (_withUpdate) {
        massUpdatePools();
    }
```

In the add() function, the token must never be added twice. So, there must be a condition to prevent that from happening by mistake.

**Resolution**: Please add a “require” condition which checks for any existing tokens. This issue is acknowledged by the ShibaNova team.

(2) Infinite loops possibility at multiple places:

```
// Update reward variables for all pools. Be careful of gas spending!
function massUpdatePools() public {
    uint256 length = poolInfo.length;
    for (uint256 pid = 0; pid < length; ++pid) {
        updatePool(pid);
    }
}
```

As seen in the AS-IS section, there are several places in the smart contracts, where array.length is used directly in the loops. It is recommended to put some kind of limits, so it does not go wild and create any scenario where it can hit the block gas limit.

**Resolution**: Please add some conditions which limits array length. This issue is acknowledged by the ShibaNova team.

## Very Low / Discussion / Best practices:

(1) Solidity version

```
pragma solidity 0.6.12;
```

Use the latest solidity version while contract deployment to prevent any compiler version level bugs.

**Resolution**: This issue is acknowledged.

(2) Event logs must be fired in place where the stats are being changed. for example:

- add function in MasterShiba.sol
- set function in MasterShiba.sol
- dev function in MasterShiba.sol
- setFeeAddress function in MasterShiba.sol
- updateMinimumEmissionRate function in MasterShiba.sol
- All functions of FeeManager contracts.

**Resolution**: This issue is acknowledged.

(3) Declare variables as constant, which never change. It is good practice and will save some gas.

File 7 of 7 : ShibaBEP20.sol

```
28
29      string private _name;
30      string private _symbol;
31      uint8 private _decimals;
32
```

**Resolution**: This issue is acknowledged.

(4) Consider specifying function visibility to “external” instead of “public”, if that function is not being called internally. It will save some gas as well.
https://ethereum.stackexchange.com/questions/32353/what-is-the-difference-between-an-internal-external-and-public-private-function/32464

**Resolution**: This issue is acknowledged.

# Centralization

These smart contracts have some functions which can be executed by Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

- removeLiquidityToToken: can be accessed by the FeeManager contract owner.
- swapBalanceToToken: can be accessed by the FeeManager contract owner.
- swapToToken: can be accessed by the FeeManager contract owner.
- updateShares: can be accessed by the FeeManager contract owner.
- setTeamAddr: can be accessed by the FeeManager contract owner.
- setupRouter: can be accessed by the FeeManager contract owner.
- setupMoneyPot: can be accessed by the FeeManager contract owner.
- distributeFee: can be accessed by the FeeManager contract owner.
- add: can be accessed by the MasterShiba contract owner.
- set: can be accessed by the MasterShiba contract owner.
- setFeeAddress:  can be accessed by the MasterShiba contract owner.
- updateMinimumEmissionRate: can be accessed by the MasterShiba owner.
- setupSNova:  can be accessed by the SNovaToken contract owner.
- mint:  can be accessed by the SNovaToken contract owner.
- setupMaster: can be accessed by the ShibaBonusAggregator contract owner.
- updateAddressWithoutReward: can be accessed by the ShibaMoneyPot owner.
- updateFeeManager: can be accessed by the ShibaMoneyPot contract owner.
- depositBonusRewards: can be accessed by the ShibaMoneyPot contract owner.
- removeTokenToRewards: can be accessed by the ShibaMoneyPot contract owner.
- addToPendingFromReserveTokenAmount: can be accessed by the ShibaMoneyPot contract owner.

# Conclusion

We were given contract codes. And we have used all possible tests based on giving objects as files. We observed some issues in the smart contracts and those are fixed/acknowledged in the smart contracts. **So it is good to go for the production.**

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high level description of functionality was presented in As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is "**Secured**".

# Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

**Manual Code Review:**

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

**Vulnerability Analysis:**

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

**Documenting Results:**

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

**Suggested Solutions:**

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

# Disclaimers

## EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

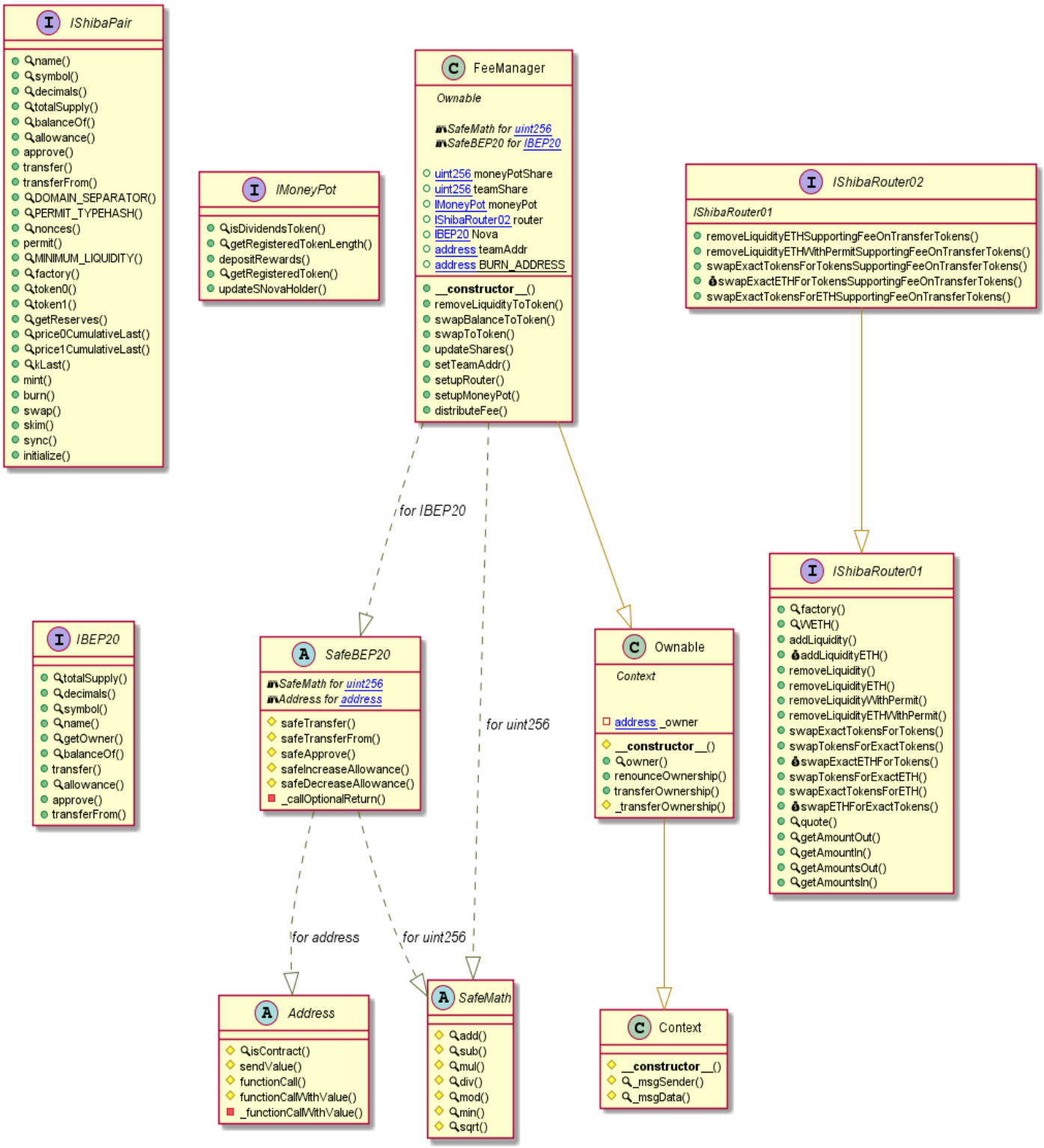
## Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

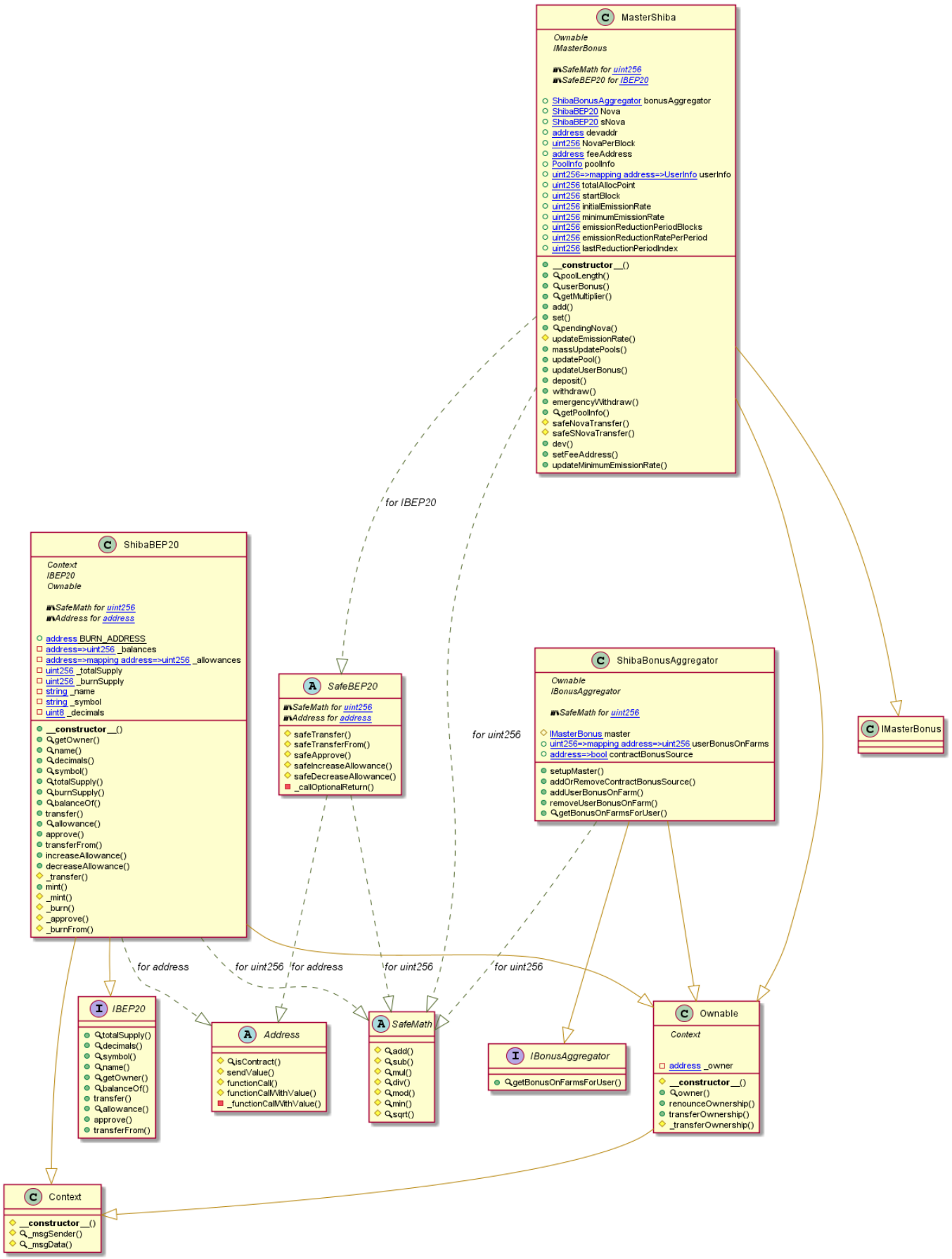
# Appendix

## Code Flow Diagram - ShibaNova Protocol

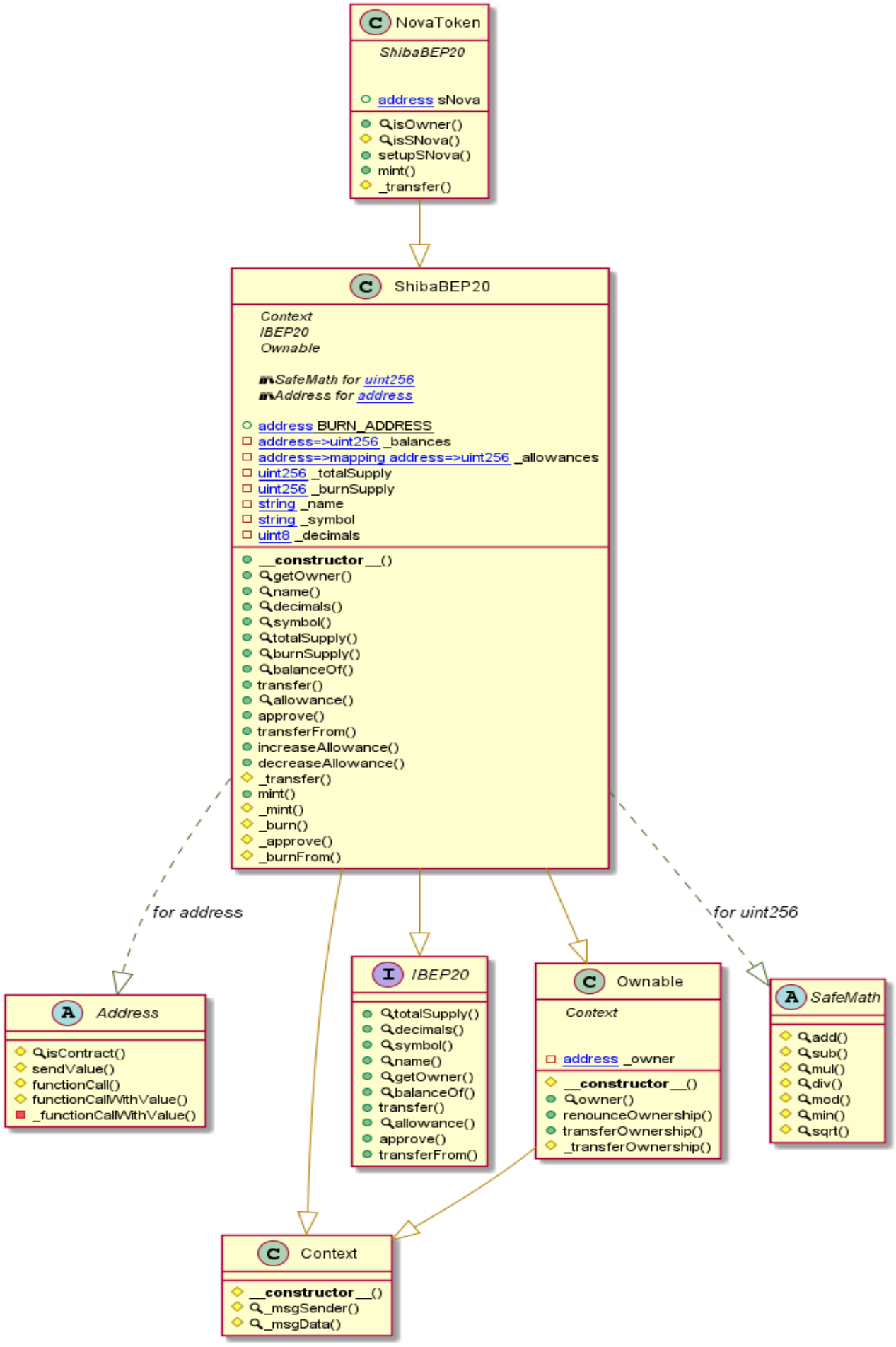
## FeeManager.sol Diagram



# MasterShiba.sol Diagram



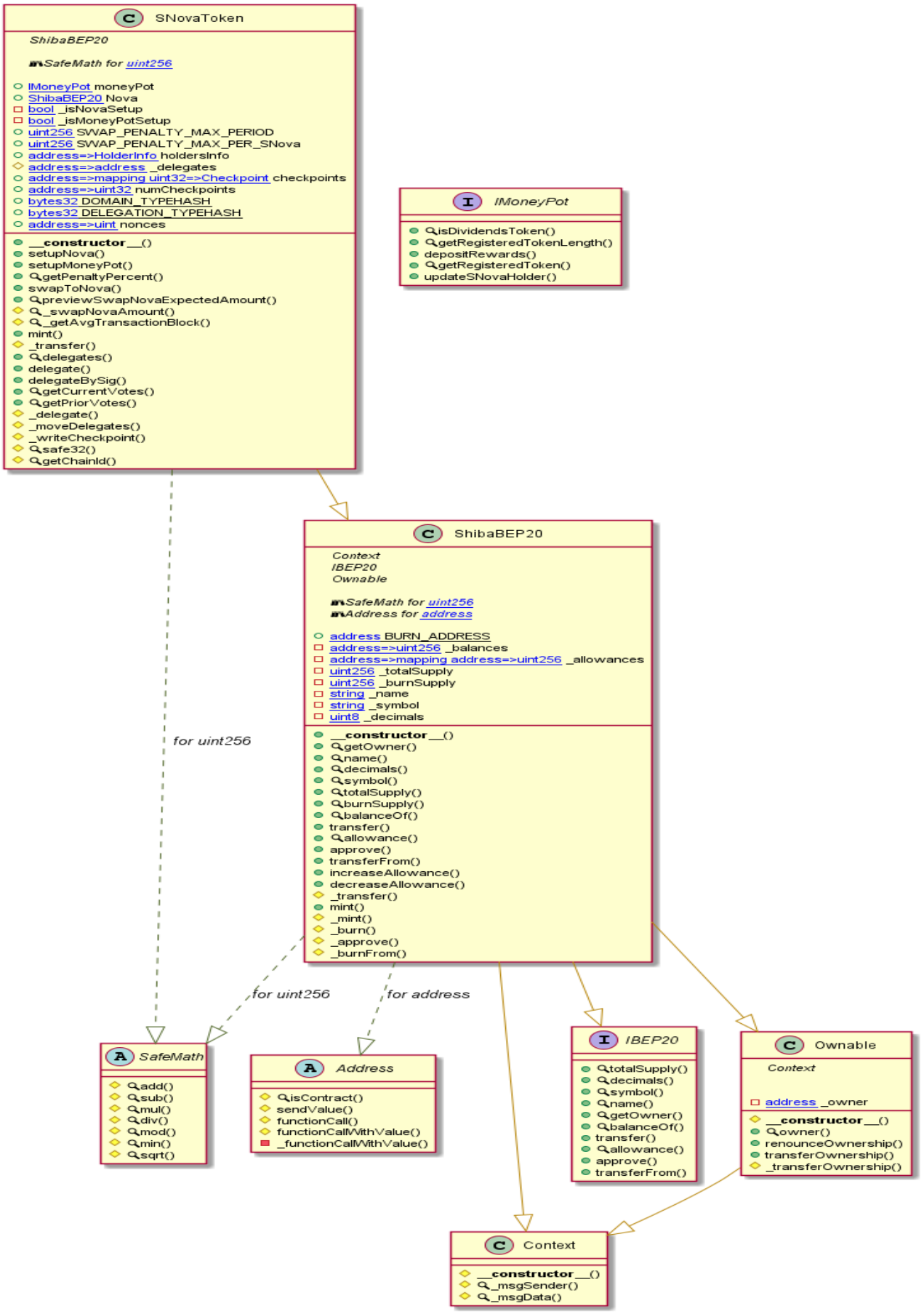
# NovaToken.sol Diagram



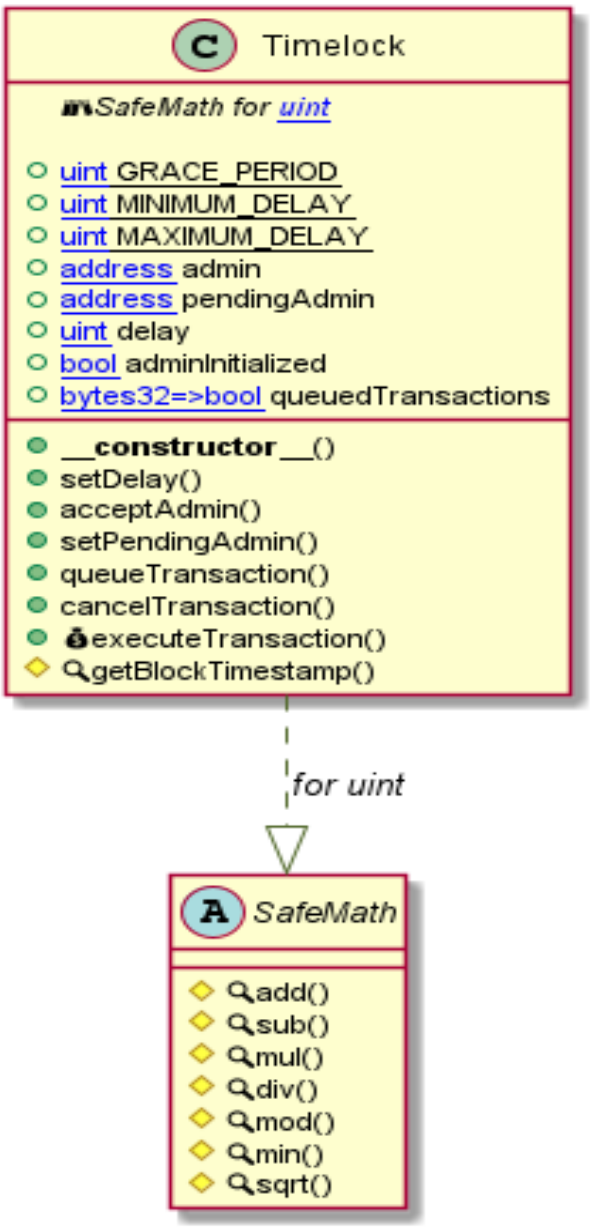



**Email: audit@EtherAuthority.io**

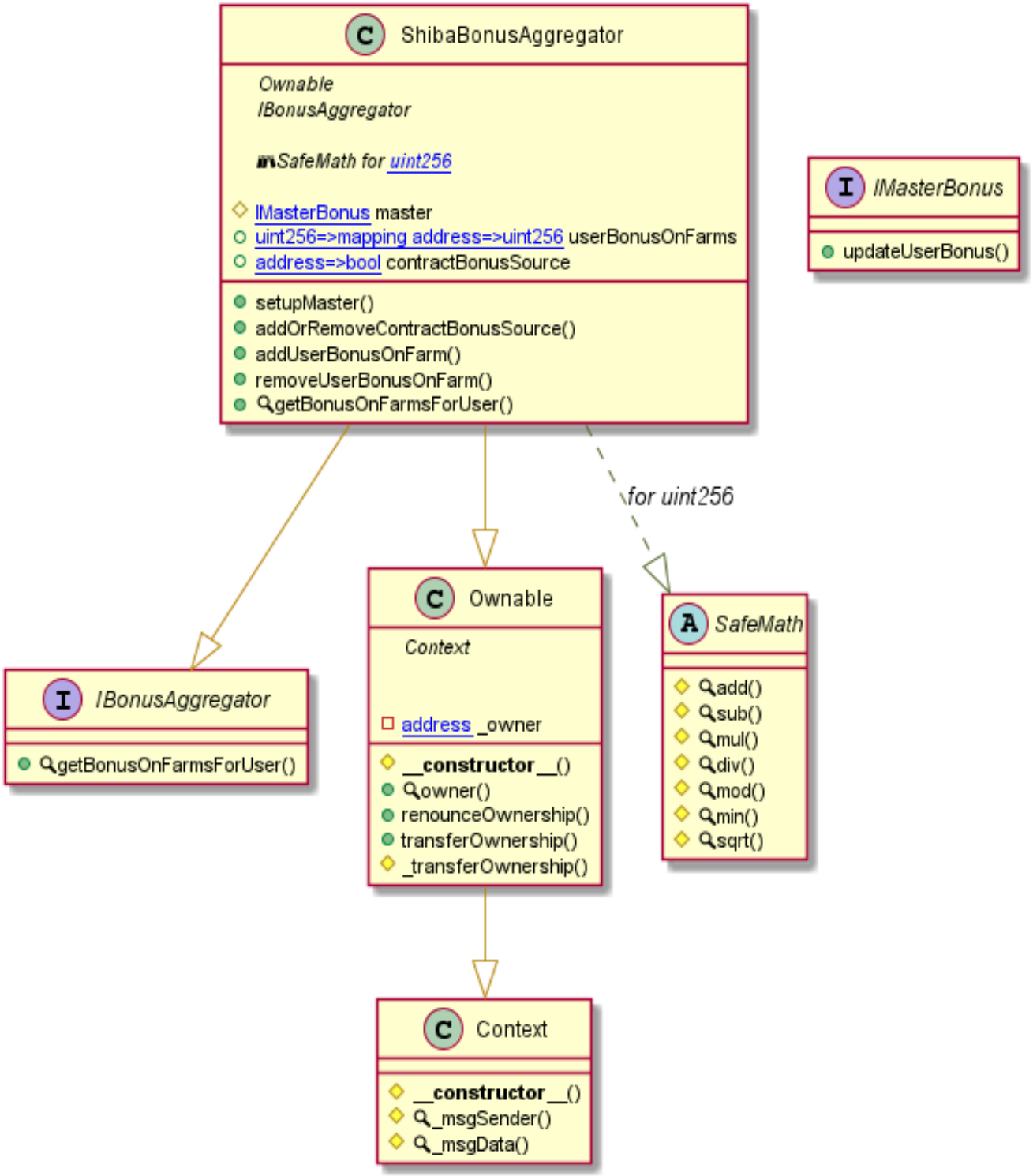
# SNovaToken.sol Diagram



# Timelock.sol Diagram



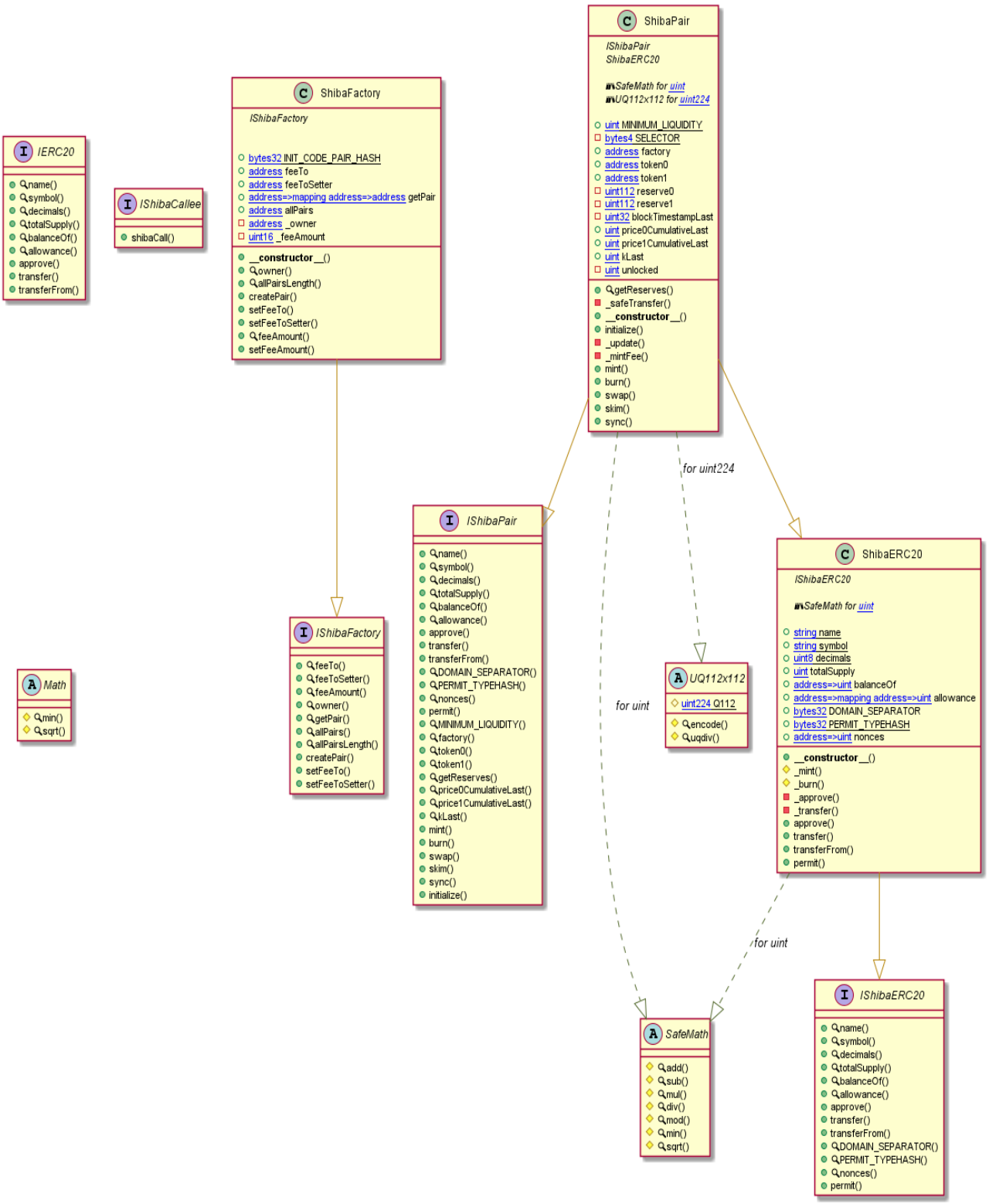
# ShibaBonusAggregator.sol Diagram



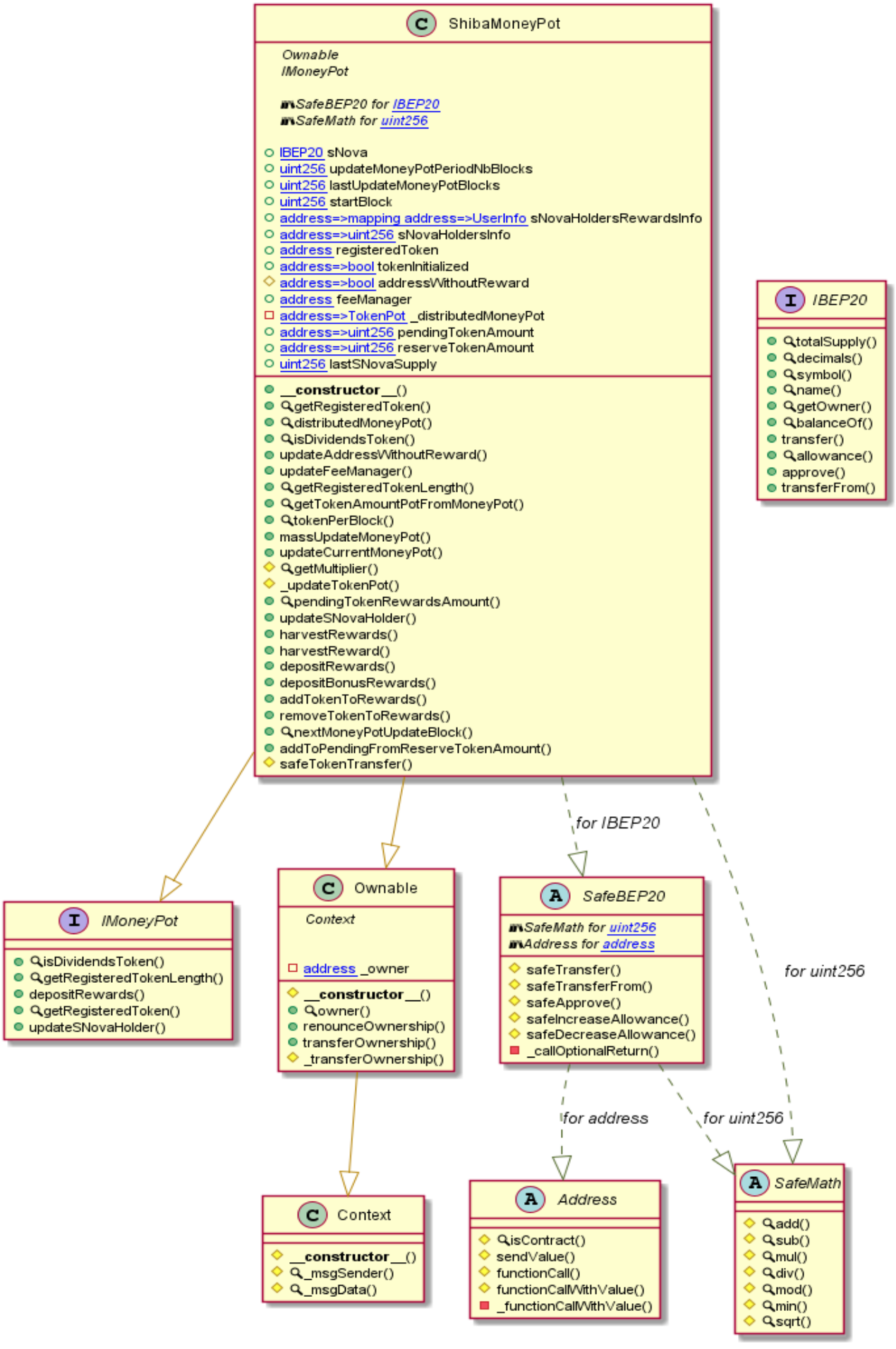



**Email: audit@EtherAuthority.io**

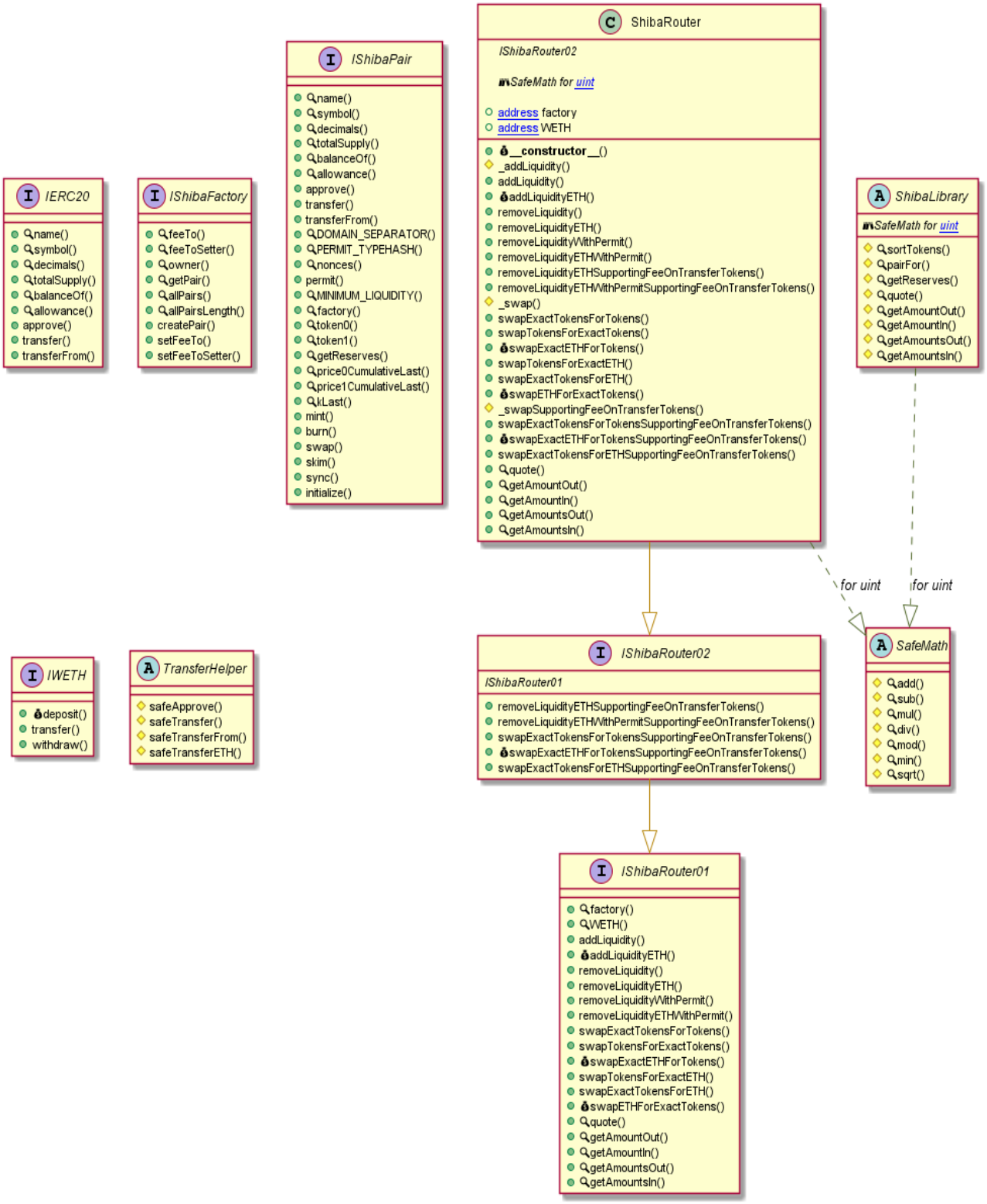
# ShibaFactory.sol Diagram



# ShibaMoneyPot.sol Diagram



# ShibaRouter.sol Diagram



# Slither Results Log

## Slither log >> FeeManager.sol

INFO:Detectors:
FeeManager.distributeFee() (FeeManager.sol#907-920) ignores return value by Nova.transfer(BURN_ADDRESS,Nova.balanceOf(address(this))) (FeeManager.sol#918)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
FeeManager.removeLiquidityToToken(address) (FeeManager.sol#858-866) ignores return value by _pair.approve(address(router),_amount) (FeeManager.sol#864)
FeeManager.removeLiquidityToToken(address) (FeeManager.sol#858-866) ignores return value by router.removeLiquidity(token0,token1,_amount,0,0,address(this),block.timestamp.add(100)) (FeeManager.sol#865)
FeeManager.swapBalanceToToken(address,address) (FeeManager.sol#868-876) ignores return value by IBEP20(_token0).approve(address(router),_amount) (FeeManager.sol#871)
FeeManager.swapBalanceToToken(address,address) (FeeManager.sol#868-876) ignores return value by router.swapExactTokensForTokens(_amount,0,path,address(this),block.timestamp.add(100)) (FeeManager.sol#875)
FeeManager.swapToToken(address,address,uint256) (FeeManager.sol#878-885) ignores return value by IBEP20(_token0).approve(address(router),_token0Amount) (FeeManager.sol#880)
FeeManager.swapToToken(address,address,uint256) (FeeManager.sol#878-885) ignores return value by router.swapExactTokensForTokens(_token0Amount,0,path,address(this),block.timestamp.add(100)) (FeeManager.sol#884)
FeeManager.distributeFee() (FeeManager.sol#907-920) ignores return value by _curToken.approve(address(moneyPot),_moneyPotAmount) (FeeManager.sol#913)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
FeeManager.constructor(IBEP20,address,uint256)._teamAddr (FeeManager.sol#850) lacks a zero-check on :
	- teamAddr = _teamAddr (FeeManager.sol#853)
FeeManager.setTeamAddr(address)._newTeamAddr (FeeManager.sol#894) lacks a zero-check on :
	- teamAddr = _newTeamAddr (FeeManager.sol#895)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
FeeManager.distributeFee() (FeeManager.sol#907-920) has external calls inside a loop: _curToken = IBEP20(moneyPot.getRegisteredToken(index)) (FeeManager.sol#910)
FeeManager.distributeFee() (FeeManager.sol#907-920) has external calls inside a loop: _amount = _curToken.balanceOf(address(this)) (FeeManager.sol#911)
FeeManager.distributeFee() (FeeManager.sol#907-920) has external calls inside a loop: _curToken.approve(address(moneyPot),_moneyPotAmount) (FeeManager.sol#913)
FeeManager.distributeFee() (FeeManager.sol#907-920) has external calls inside a loop: moneyPot.depositRewards(address(_curToken),_moneyPotAmount) (FeeManager.sol#914)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Address.isContract(address) (FeeManager.sol#76-87) uses assembly
	- INLINE ASM (FeeManager.sol#83-85)
Address._functionCallWithValue(address,bytes,uint256,string) (FeeManager.sol#184-210) uses assembly
	- INLINE ASM (FeeManager.sol#202-205)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.functionCall(address,bytes) (FeeManager.sol#131-133) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (FeeManager.sol#160-166) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (FeeManager.sol#174-182) is never used and should be removed
Address.sendValue(address,uint256) (FeeManager.sol#105-111) is never used and should be removed
Context._msgData() (FeeManager.sol#754-757) is never used and should be removed
SafeBEP20.safeApprove(IBEP20,address,uint256) (FeeManager.sol#394-408) is never used and should be removed

SafeBEP20.safeDecreaseAllowance(IBEP20,address,uint256) (FeeManager.sol#419-429) is never used and should be removed
SafeBEP20.safeIncreaseAllowance(IBEP20,address,uint256) (FeeManager.sol#410-417) is never used and should be removed
SafeBEP20.safeTransferFrom(IBEP20,address,address,uint256) (FeeManager.sol#378-385) is never used and should be removed
SafeMath.min(uint256,uint256) (FeeManager.sol#715-717) is never used and should be removed
SafeMath.mod(uint256,uint256) (FeeManager.sol#690-692) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (FeeManager.sol#706-713) is never used and should be removed
SafeMath.sqrt(uint256) (FeeManager.sol#720-731) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (FeeManager.sol#105-111):
- (success) = recipient.call{value: amount}() (FeeManager.sol#109)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (FeeManager.sol#184-210):
- (success,returndata) = target.call{value: weiValue}(data) (FeeManager.sol#193)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IShibaPair.DOMAIN_SEPARATOR() (FeeManager.sol#20) is not in mixedCase
Function IShibaPair.PERMIT_TYPEHASH() (FeeManager.sol#21) is not in mixedCase
Function IShibaPair.MINIMUM_LIQUIDITY() (FeeManager.sol#38) is not in mixedCase
Function IShibaRouter01.WETH() (FeeManager.sol#215) is not in mixedCase
Parameter FeeManager.removeLiquidityToToken(address)._token (FeeManager.sol#858) is not in mixedCase
Parameter FeeManager.swapBalanceToToken(address,address)._token0 (FeeManager.sol#868) is not in mixedCase
Parameter FeeManager.swapBalanceToToken(address,address)._token1 (FeeManager.sol#868) is not in mixedCase
Parameter FeeManager.swapToToken(address,address,uint256)._token0 (FeeManager.sol#878) is not in mixedCase
Parameter FeeManager.swapToToken(address,address,uint256)._token1 (FeeManager.sol#878) is not in mixedCase
Parameter FeeManager.swapToToken(address,address,uint256)._token0Amount (FeeManager.sol#878) is not in mixedCase
Parameter FeeManager.updateShares(uint256)._moneyPotShare (FeeManager.sol#887) is not in mixedCase
Parameter FeeManager.setTeamAddr(address)._newTeamAddr (FeeManager.sol#894) is not in mixedCase
Parameter FeeManager.setupRouter(address)._router (FeeManager.sol#898) is not in mixedCase
Parameter FeeManager.setupMoneyPot(IMoneyPot)._moneyPot (FeeManager.sol#902) is not in mixedCase
Variable FeeManager.Nova (FeeManager.sol#844) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (FeeManager.sol#755)" inContext (FeeManager.sol#745-758)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable IShibaRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (FeeManager.sol#220) is too similar to IShibaRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (FeeManager.sol#221)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
FeeManager.slitherConstructorConstantVariables() (FeeManager.sol#835-922) uses literals with too many digits:
- BURN_ADDRESS = 0x000000000000000000000000000000000000dEaD (FeeManager.sol#848)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
owner() should be declared external:
- Ownable.owner() (FeeManager.sol#789-791)
renounceOwnership() should be declared external:
- Ownable.renounceOwnership() (FeeManager.sol#808-811)
transferOwnership(address) should be declared external:
- Ownable.transferOwnership(address) (FeeManager.sol#817-819)

Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:FeeManager.sol analyzed (11 contracts with 75 detectors), 52 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration

//----------

## Slither log >> MasterShiba.sol

INFO:Detectors:
MasterShiba.pendingNova(uint256,address) (MasterShiba.sol#1117-1133) performs a multiplication on the result of a division:
-NovaReward = multiplier.mul(NovaPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (MasterShiba.sol#1124)
-accNovaPerShare = accNovaPerShare.add(NovaReward.mul(1e12).div(lpSupply)) (MasterShiba.sol#1125)
MasterShiba.updatePool(uint256) (MasterShiba.sol#1174-1197) performs a multiplication on the result of a division:
-NovaReward = multiplier.mul(NovaPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (MasterShiba.sol#1186)
-pool.accNovaPerShare = pool.accNovaPerShare.add(NovaReward.mul(1e12).div(lpSupply)) (MasterShiba.sol#1195)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in MasterShiba.add(uint256,IBEP20,uint256,bool,bool) (MasterShiba.sol#1083-1099):
External calls:
- massUpdatePools() (MasterShiba.sol#1086)
- Nova.mint(devaddr,devMintAmount) (MasterShiba.sol#1188)
- sNova.mint(address(this),NovaReward) (MasterShiba.sol#1190)
- Nova.mint(address(this),NovaReward) (MasterShiba.sol#1193)
State variables written after the call(s):
- poolInfo.push(PoolInfo(_lpToken,0,_allocPoint,lastRewardBlock,0,_depositFeeBP,_isSNovaRewards)) (MasterShiba.sol#1090-1098)
- totalAllocPoint = totalAllocPoint.add(_allocPoint) (MasterShiba.sol#1089)
Reentrancy in MasterShiba.deposit(uint256,uint256) (MasterShiba.sol#1222-1260):
External calls:
- updatePool(_pid) (MasterShiba.sol#1226)
- Nova.mint(devaddr,devMintAmount) (MasterShiba.sol#1188)
- sNova.mint(address(this),NovaReward) (MasterShiba.sol#1190)
- Nova.mint(address(this),NovaReward) (MasterShiba.sol#1193)
- safeSNovaTransfer(_user,pending) (MasterShiba.sol#1231)
- transferSuccess = sNova.transfer(_to,sNovaBal) (MasterShiba.sol#1332)
- transferSuccess = sNova.transfer(_to,_amount) (MasterShiba.sol#1334)
- safeNovaTransfer(_user,pending) (MasterShiba.sol#1234)
- transferSuccess = Nova.transfer(_to,NovaBal) (MasterShiba.sol#1320)
- transferSuccess = Nova.transfer(_to,_amount) (MasterShiba.sol#1322)
- pool.lpToken.safeTransferFrom(address(_user),address(this),_amount) (MasterShiba.sol#1239)
- pool.lpToken.safeTransfer(feeAddress,depositFee) (MasterShiba.sol#1246)
State variables written after the call(s):
- pool.lpSupply = pool.lpSupply.add(_bonusAmount) (MasterShiba.sol#1250)
- user.amount = user.amount.add(_amount).sub(depositFee) (MasterShiba.sol#1247)
- user.amountWithBonus = user.amountWithBonus.add(_bonusAmount) (MasterShiba.sol#1249)
Reentrancy in MasterShiba.deposit(uint256,uint256) (MasterShiba.sol#1222-1260):
External calls:
- updatePool(_pid) (MasterShiba.sol#1226)
- Nova.mint(devaddr,devMintAmount) (MasterShiba.sol#1188)
- sNova.mint(address(this),NovaReward) (MasterShiba.sol#1190)
- Nova.mint(address(this),NovaReward) (MasterShiba.sol#1193)
- safeSNovaTransfer(_user,pending) (MasterShiba.sol#1231)
- transferSuccess = sNova.transfer(_to,sNovaBal) (MasterShiba.sol#1332)
- transferSuccess = sNova.transfer(_to,_amount) (MasterShiba.sol#1334)
- safeNovaTransfer(_user,pending) (MasterShiba.sol#1234)

- transferSuccess = Nova.transfer(_to,NovaBal) (MasterShiba.sol#1320)
- transferSuccess = Nova.transfer(_to,_amount) (MasterShiba.sol#1322)
- pool.lpToken.safeTransferFrom(address(_user),address(this),_amount) (MasterShiba.sol#1239)
State variables written after the call(s):
- pool.lpSupply = pool.lpSupply.add(_bonusAmount_scope_0) (MasterShiba.sol#1255)
- user.amount = user.amount.add(_amount) (MasterShiba.sol#1252)
- user.amountWithBonus = user.amountWithBonus.add(_bonusAmount_scope_0) (MasterShiba.sol#1254)

Reentrancy in MasterShiba.set(uint256,uint256,uint256,bool,bool) (MasterShiba.sol#1102-1114):
External calls:
- massUpdatePools() (MasterShiba.sol#1105)
- Nova.mint(devaddr,devMintAmount) (MasterShiba.sol#1188)
- sNova.mint(address(this),NovaReward) (MasterShiba.sol#1190)
- Nova.mint(address(this),NovaReward) (MasterShiba.sol#1193)
State variables written after the call(s):
- poolInfo[_pid].allocPoint = _allocPoint (MasterShiba.sol#1108)
- poolInfo[_pid].depositFeeBP = _depositFeeBP (MasterShiba.sol#1109)
- poolInfo[_pid].isSNovaRewards = _isSNovaRewards (MasterShiba.sol#1110)
- totalAllocPoint = totalAllocPoint.sub(prevAllocPoint).add(_allocPoint) (MasterShiba.sol#1112)

Reentrancy in MasterShiba.updatePool(uint256) (MasterShiba.sol#1174-1197):
External calls:
- Nova.mint(devaddr,devMintAmount) (MasterShiba.sol#1188)
- sNova.mint(address(this),NovaReward) (MasterShiba.sol#1190)
- Nova.mint(address(this),NovaReward) (MasterShiba.sol#1193)
State variables written after the call(s):
- pool.accNovaPerShare = pool.accNovaPerShare.add(NovaReward.mul(1e12).div(lpSupply)) (MasterShiba.sol#1195)
- pool.lastRewardBlock = block.number (MasterShiba.sol#1196)

Reentrancy in MasterShiba.updateUserBonus(address,uint256,uint256) (MasterShiba.sol#1200-1219):
External calls:
- updatePool(_pid) (MasterShiba.sol#1203)
- Nova.mint(devaddr,devMintAmount) (MasterShiba.sol#1188)
- sNova.mint(address(this),NovaReward) (MasterShiba.sol#1190)
- Nova.mint(address(this),NovaReward) (MasterShiba.sol#1193)
- safeSNovaTransfer(_user,pending) (MasterShiba.sol#1208)
- transferSuccess = sNova.transfer(_to,sNovaBal) (MasterShiba.sol#1332)
- transferSuccess = sNova.transfer(_to,_amount) (MasterShiba.sol#1334)
- safeNovaTransfer(_user,pending) (MasterShiba.sol#1211)
- transferSuccess = Nova.transfer(_to,NovaBal) (MasterShiba.sol#1320)
- transferSuccess = Nova.transfer(_to,_amount) (MasterShiba.sol#1322)
State variables written after the call(s):
- pool.lpSupply = pool.lpSupply.sub(user.amountWithBonus) (MasterShiba.sol#1215)
- pool.lpSupply = pool.lpSupply.add(user.amountWithBonus) (MasterShiba.sol#1217)
- user.amountWithBonus = user.amount.mul(bonus.add(10000)).div(10000) (MasterShiba.sol#1216)
- user.rewardDebt = user.amountWithBonus.mul(pool.accNovaPerShare).div(1e12) (MasterShiba.sol#1218)

Reentrancy in MasterShiba.withdraw(uint256,uint256) (MasterShiba.sol#1263-1287):
External calls:
- updatePool(_pid) (MasterShiba.sol#1268)
- Nova.mint(devaddr,devMintAmount) (MasterShiba.sol#1188)
- sNova.mint(address(this),NovaReward) (MasterShiba.sol#1190)
- Nova.mint(address(this),NovaReward) (MasterShiba.sol#1193)
- safeSNovaTransfer(msg.sender,pending) (MasterShiba.sol#1272)
- transferSuccess = sNova.transfer(_to,sNovaBal) (MasterShiba.sol#1332)
- transferSuccess = sNova.transfer(_to,_amount) (MasterShiba.sol#1334)
- safeNovaTransfer(msg.sender,pending) (MasterShiba.sol#1275)
- transferSuccess = Nova.transfer(_to,NovaBal) (MasterShiba.sol#1320)
- transferSuccess = Nova.transfer(_to,_amount) (MasterShiba.sol#1322)
State variables written after the call(s):
- user.amount = user.amount.sub(_amount) (MasterShiba.sol#1279)
- user.amountWithBonus = user.amountWithBonus.sub(_bonusAmount) (MasterShiba.sol#1281)

Reentrancy in MasterShiba.withdraw(uint256,uint256) (MasterShiba.sol#1263-1287):
External calls:
- updatePool(_pid) (MasterShiba.sol#1268)
- Nova.mint(devaddr,devMintAmount) (MasterShiba.sol#1188)

```
            - sNova.mint(address(this),NovaReward) (MasterShiba.sol#1190)
            - Nova.mint(address(this),NovaReward) (MasterShiba.sol#1193)
        - safeSNovaTransfer(msg.sender,pending) (MasterShiba.sol#1272)
            - transferSuccess = sNova.transfer(_to,sNovaBal) (MasterShiba.sol#1332)
            - transferSuccess = sNova.transfer(_to,_amount) (MasterShiba.sol#1334)
        - safeNovaTransfer(msg.sender,pending) (MasterShiba.sol#1275)
            - transferSuccess = Nova.transfer(_to,NovaBal) (MasterShiba.sol#1320)
            - transferSuccess = Nova.transfer(_to,_amount) (MasterShiba.sol#1322)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (MasterShiba.sol#1282)
        State variables written after the call(s):
        - pool.lpSupply = pool.lpSupply.sub(_bonusAmount) (MasterShiba.sol#1283)
        - user.rewardDebt = user.amountWithBonus.mul(pool.accNovaPerShare).div(1e12)
(MasterShiba.sol#1285)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
ShibaBEP20.constructor(string,string).name (MasterShiba.sol#695) shadows:
        - ShibaBEP20.name() (MasterShiba.sol#711-713) (function)
        - IBEP20.name() (MasterShiba.sol#188) (function)
ShibaBEP20.constructor(string,string).symbol (MasterShiba.sol#695) shadows:
        - ShibaBEP20.symbol() (MasterShiba.sol#725-727) (function)
        - IBEP20.symbol() (MasterShiba.sol#183) (function)
ShibaBEP20.allowance(address,address).owner (MasterShiba.sol#766) shadows:
        - Ownable.owner() (MasterShiba.sol#316-318) (function)
ShibaBEP20._approve(address,address,uint256).owner (MasterShiba.sol#929) shadows:
        - Ownable.owner() (MasterShiba.sol#316-318) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
MasterShiba.constructor(ShibaBEP20,ShibaBEP20,ShibaBonusAggregator,address,address,uint256,uint256
)._devaddr (MasterShiba.sol#1032) lacks a zero-check on :
            - devaddr = _devaddr (MasterShiba.sol#1040)
MasterShiba.constructor(ShibaBEP20,ShibaBEP20,ShibaBonusAggregator,address,address,uint256,uint256
)._feeAddress (MasterShiba.sol#1033) lacks a zero-check on :
            - feeAddress = _feeAddress (MasterShiba.sol#1041)
MasterShiba.dev(address)._devaddr (MasterShiba.sol#1340) lacks a zero-check on :
            - devaddr = _devaddr (MasterShiba.sol#1342)
MasterShiba.setFeeAddress(address)._feeAddress (MasterShiba.sol#1345) lacks a zero-check on :
            - feeAddress = _feeAddress (MasterShiba.sol#1346)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in MasterShiba.deposit(uint256,uint256) (MasterShiba.sol#1222-1260):
        External calls:
        - updatePool(_pid) (MasterShiba.sol#1226)
            - Nova.mint(devaddr,devMintAmount) (MasterShiba.sol#1188)
            - sNova.mint(address(this),NovaReward) (MasterShiba.sol#1190)
            - Nova.mint(address(this),NovaReward) (MasterShiba.sol#1193)
        - safeSNovaTransfer(_user,pending) (MasterShiba.sol#1231)
            - transferSuccess = sNova.transfer(_to,sNovaBal) (MasterShiba.sol#1332)
            - transferSuccess = sNova.transfer(_to,_amount) (MasterShiba.sol#1334)
        - safeNovaTransfer(_user,pending) (MasterShiba.sol#1234)
            - transferSuccess = Nova.transfer(_to,NovaBal) (MasterShiba.sol#1320)
            - transferSuccess = Nova.transfer(_to,_amount) (MasterShiba.sol#1322)
        - pool.lpToken.safeTransferFrom(address(_user),address(this),_amount) (MasterShiba.sol#1239)
        - pool.lpToken.safeTransfer(feeAddress,depositFee) (MasterShiba.sol#1246)
        Event emitted after the call(s):
        - Deposit(_user,_pid,_amount) (MasterShiba.sol#1259)
Reentrancy in MasterShiba.emergencyWithdraw(uint256) (MasterShiba.sol#1290-1300):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),amount) (MasterShiba.sol#1298)
        Event emitted after the call(s):
        - EmergencyWithdraw(msg.sender,_pid,amount) (MasterShiba.sol#1299)
Reentrancy in MasterShiba.withdraw(uint256,uint256) (MasterShiba.sol#1263-1287):
        External calls:
        - updatePool(_pid) (MasterShiba.sol#1268)
            - Nova.mint(devaddr,devMintAmount) (MasterShiba.sol#1188)
            - sNova.mint(address(this),NovaReward) (MasterShiba.sol#1190)
```

- Nova.mint(address(this),NovaReward) (MasterShiba.sol#1193)
- safeSNovaTransfer(msg.sender,pending) (MasterShiba.sol#1272)
- transferSuccess = sNova.transfer(_to,sNovaBal) (MasterShiba.sol#1332)
- transferSuccess = sNova.transfer(_to,_amount) (MasterShiba.sol#1334)
- safeNovaTransfer(msg.sender,pending) (MasterShiba.sol#1275)
- transferSuccess = Nova.transfer(_to,NovaBal) (MasterShiba.sol#1320)
- transferSuccess = Nova.transfer(_to,_amount) (MasterShiba.sol#1322)
- pool.lpToken.safeTransfer(address(msg.sender),_amount) (MasterShiba.sol#1282)

Event emitted after the call(s):
- Withdraw(msg.sender,_pid,_amount) (MasterShiba.sol#1286)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address.isContract(address) (MasterShiba.sol#32-43) uses assembly
- INLINE ASM (MasterShiba.sol#39-41)

Address._functionCallWithValue(address,bytes,uint256,string) (MasterShiba.sol#140-166) uses assembly
- INLINE ASM (MasterShiba.sol#158-161)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.functionCall(address,bytes) (MasterShiba.sol#87-89) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (MasterShiba.sol#116-122) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (MasterShiba.sol#130-138) is never used and should be removed
Address.sendValue(address,uint256) (MasterShiba.sol#61-67) is never used and should be removed
Context._msgData() (MasterShiba.sol#282-285) is never used and should be removed
SafeBEP20.safeApprove(IBEP20,address,uint256) (MasterShiba.sol#607-621) is never used and should be removed
SafeBEP20.safeDecreaseAllowance(IBEP20,address,uint256) (MasterShiba.sol#632-642) is never used and should be removed
SafeBEP20.safeIncreaseAllowance(IBEP20,address,uint256) (MasterShiba.sol#623-630) is never used and should be removed
SafeMath.min(uint256,uint256) (MasterShiba.sol#561-563) is never used and should be removed
SafeMath.mod(uint256,uint256) (MasterShiba.sol#536-538) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (MasterShiba.sol#552-559) is never used and should be removed
SafeMath.sqrt(uint256) (MasterShiba.sol#566-577) is never used and should be removed
ShibaBEP20._burn(address,uint256) (MasterShiba.sol#906-913) is never used and should be removed
ShibaBEP20._burnFrom(address,uint256) (MasterShiba.sol#946-953) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (MasterShiba.sol#61-67):
- (success) = recipient.call{value: amount}() (MasterShiba.sol#65)

Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (MasterShiba.sol#140-166):
- (success,returndata) = target.call{value: weiValue}(data) (MasterShiba.sol#149)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter ShibaBonusAggregator.setupMaster(IMasterBonus)._master (MasterShiba.sol#383) is not in mixedCase
Parameter ShibaBonusAggregator.addOrRemoveContractBonusSource(address,bool)._contract (MasterShiba.sol#387) is not in mixedCase
Parameter ShibaBonusAggregator.addOrRemoveContractBonusSource(address,bool)._add (MasterShiba.sol#387) is not in mixedCase
Parameter ShibaBonusAggregator.addUserBonusOnFarm(address,uint256,uint256)._user (MasterShiba.sol#391) is not in mixedCase
Parameter ShibaBonusAggregator.addUserBonusOnFarm(address,uint256,uint256)._percent (MasterShiba.sol#391) is not in mixedCase
Parameter ShibaBonusAggregator.addUserBonusOnFarm(address,uint256,uint256)._pid (MasterShiba.sol#391) is not in mixedCase
Parameter ShibaBonusAggregator.removeUserBonusOnFarm(address,uint256,uint256)._user (MasterShiba.sol#397) is not in mixedCase
Parameter ShibaBonusAggregator.removeUserBonusOnFarm(address,uint256,uint256)._percent (MasterShiba.sol#397) is not in mixedCase
Parameter ShibaBonusAggregator.removeUserBonusOnFarm(address,uint256,uint256)._pid (MasterShiba.sol#397) is not in mixedCase
Parameter ShibaBonusAggregator.getBonusOnFarmsForUser(address,uint256)._user (MasterShiba.sol#402) is not in mixedCase

Parameter ShibaBonusAggregator.getBonusOnFarmsForUser(address,uint256)._pid (MasterShiba.sol#402) is not in mixedCase
Parameter ShibaBEP20.mint(address,uint256)._to (MasterShiba.sol#875) is not in mixedCase
Parameter ShibaBEP20.mint(address,uint256)._amount (MasterShiba.sol#875) is not in mixedCase
Parameter MasterShiba.userBonus(uint256,address)._pid (MasterShiba.sol#1073) is not in mixedCase
Parameter MasterShiba.userBonus(uint256,address)._user (MasterShiba.sol#1073) is not in mixedCase
Parameter MasterShiba.getMultiplier(uint256,uint256)._from (MasterShiba.sol#1078) is not in mixedCase
Parameter MasterShiba.getMultiplier(uint256,uint256)._to (MasterShiba.sol#1078) is not in mixedCase
Parameter MasterShiba.add(uint256,IBEP20,uint256,bool,bool)._allocPoint (MasterShiba.sol#1083) is not in mixedCase
Parameter MasterShiba.add(uint256,IBEP20,uint256,bool,bool)._lpToken (MasterShiba.sol#1083) is not in mixedCase
Parameter MasterShiba.add(uint256,IBEP20,uint256,bool,bool)._depositFeeBP (MasterShiba.sol#1083) is not in mixedCase
Parameter MasterShiba.add(uint256,IBEP20,uint256,bool,bool)._isSNovaRewards (MasterShiba.sol#1083) is not in mixedCase
Parameter MasterShiba.add(uint256,IBEP20,uint256,bool,bool)._withUpdate (MasterShiba.sol#1083) is not in mixedCase
Parameter MasterShiba.set(uint256,uint256,uint256,bool,bool)._pid (MasterShiba.sol#1102) is not in mixedCase
Parameter MasterShiba.set(uint256,uint256,uint256,bool,bool)._allocPoint (MasterShiba.sol#1102) is not in mixedCase
Parameter MasterShiba.set(uint256,uint256,uint256,bool,bool)._depositFeeBP (MasterShiba.sol#1102) is not in mixedCase
Parameter MasterShiba.set(uint256,uint256,uint256,bool,bool)._isSNovaRewards (MasterShiba.sol#1102) is not in mixedCase
Parameter MasterShiba.set(uint256,uint256,uint256,bool,bool)._withUpdate (MasterShiba.sol#1102) is not in mixedCase
Parameter MasterShiba.pendingNova(uint256,address)._pid (MasterShiba.sol#1117) is not in mixedCase
Parameter MasterShiba.pendingNova(uint256,address)._user (MasterShiba.sol#1117) is not in mixedCase
Parameter MasterShiba.updatePool(uint256)._pid (MasterShiba.sol#1174) is not in mixedCase
Parameter MasterShiba.updateUserBonus(address,uint256,uint256)._user (MasterShiba.sol#1200) is not in mixedCase
Parameter MasterShiba.updateUserBonus(address,uint256,uint256)._pid (MasterShiba.sol#1200) is not in mixedCase
Parameter MasterShiba.deposit(uint256,uint256)._pid (MasterShiba.sol#1222) is not in mixedCase
Parameter MasterShiba.deposit(uint256,uint256)._amount (MasterShiba.sol#1222) is not in mixedCase
Parameter MasterShiba.withdraw(uint256,uint256)._pid (MasterShiba.sol#1263) is not in mixedCase
Parameter MasterShiba.withdraw(uint256,uint256)._amount (MasterShiba.sol#1263) is not in mixedCase
Parameter MasterShiba.emergencyWithdraw(uint256)._pid (MasterShiba.sol#1290) is not in mixedCase
Parameter MasterShiba.getPoolInfo(uint256)._pid (MasterShiba.sol#1302) is not in mixedCase
Parameter MasterShiba.safeNovaTransfer(address,uint256)._to (MasterShiba.sol#1316) is not in mixedCase
Parameter MasterShiba.safeNovaTransfer(address,uint256)._amount (MasterShiba.sol#1316) is not in mixedCase
Parameter MasterShiba.safeSNovaTransfer(address,uint256)._to (MasterShiba.sol#1328) is not in mixedCase
Parameter MasterShiba.safeSNovaTransfer(address,uint256)._amount (MasterShiba.sol#1328) is not in mixedCase
Parameter MasterShiba.dev(address)._devaddr (MasterShiba.sol#1340) is not in mixedCase
Parameter MasterShiba.setFeeAddress(address)._feeAddress (MasterShiba.sol#1345) is not in mixedCase
Parameter MasterShiba.updateMinimumEmissionRate(uint256)._minimumEmissionRate (MasterShiba.sol#1349) is not in mixedCase
Variable MasterShiba.Nova (MasterShiba.sol#993) is not in mixedCase
Variable MasterShiba.NovaPerBlock (MasterShiba.sol#999) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (MasterShiba.sol#283)" inContext (MasterShiba.sol#273-286)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
ShibaBEP20.slitherConstructorConstantVariables() (MasterShiba.sol#669-954) uses literals with too many digits:
	- BURN_ADDRESS = 0x000000000000000000000000000000000000dEaD (MasterShiba.sol#673)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:

MasterShiba.emissionReductionPeriodBlocks (MasterShiba.sol#1017) should be constant
MasterShiba.emissionReductionRatePerPeriod (MasterShiba.sol#1019) should be constant
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
renounceOwnership() should be declared external:
- Ownable.renounceOwnership() (MasterShiba.sol#335-338)
transferOwnership(address) should be declared external:
- Ownable.transferOwnership(address) (MasterShiba.sol#344-346)
name() should be declared external:
- ShibaBEP20.name() (MasterShiba.sol#711-713)
decimals() should be declared external:
- ShibaBEP20.decimals() (MasterShiba.sol#718-720)
symbol() should be declared external:
- ShibaBEP20.symbol() (MasterShiba.sol#725-727)
totalSupply() should be declared external:
- ShibaBEP20.totalSupply() (MasterShiba.sol#732-734)
burnSupply() should be declared external:
- ShibaBEP20.burnSupply() (MasterShiba.sol#739-741)
balanceOf(address) should be declared external:
- ShibaBEP20.balanceOf(address) (MasterShiba.sol#746-748)
transfer(address,uint256) should be declared external:
- ShibaBEP20.transfer(address,uint256) (MasterShiba.sol#758-761)
allowance(address,address) should be declared external:
- ShibaBEP20.allowance(address,address) (MasterShiba.sol#766-768)
approve(address,uint256) should be declared external:
- ShibaBEP20.approve(address,uint256) (MasterShiba.sol#777-780)
transferFrom(address,address,uint256) should be declared external:
- ShibaBEP20.transferFrom(address,address,uint256) (MasterShiba.sol#794-806)
increaseAllowance(address,uint256) should be declared external:
- ShibaBEP20.increaseAllowance(address,uint256) (MasterShiba.sol#820-823)
decreaseAllowance(address,uint256) should be declared external:
- ShibaBEP20.decreaseAllowance(address,uint256) (MasterShiba.sol#839-846)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MasterShiba.sol analyzed (11 contracts with 75 detectors), 104 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration

//----------

## Slither log >> NovaToken.sol

INFO:Detectors:
ShibaBEP20.constructor(string,string).name (NovaToken.sol#575) shadows:
- ShibaBEP20.name() (NovaToken.sol#591-593) (function)
- IBEP20.name() (NovaToken.sol#207) (function)
ShibaBEP20.constructor(string,string).symbol (NovaToken.sol#575) shadows:
- ShibaBEP20.symbol() (NovaToken.sol#605-607) (function)
- IBEP20.symbol() (NovaToken.sol#202) (function)
ShibaBEP20.allowance(address,address).owner (NovaToken.sol#646) shadows:
- Ownable.owner() (NovaToken.sol#314-316) (function)
ShibaBEP20._approve(address,address,uint256).owner (NovaToken.sol#809) shadows:
- Ownable.owner() (NovaToken.sol#314-316) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
NovaToken.setupSNova(address)._sNova (NovaToken.sol#863) lacks a zero-check on :
- sNova = _sNova (NovaToken.sol#864)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Address.isContract(address) (NovaToken.sol#26-37) uses assembly
- INLINE ASM (NovaToken.sol#33-35)

Address._functionCallWithValue(address,bytes,uint256,string) (NovaToken.sol#134-160) uses assembly
- INLINE ASM (NovaToken.sol#152-155)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (NovaToken.sol#134-160) is never used and should be removed
Address.functionCall(address,bytes) (NovaToken.sol#81-83) is never used and should be removed
Address.functionCall(address,bytes,string) (NovaToken.sol#91-97) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (NovaToken.sol#110-116) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (NovaToken.sol#124-132) is never used and should be removed
Address.isContract(address) (NovaToken.sol#26-37) is never used and should be removed
Address.sendValue(address,uint256) (NovaToken.sol#55-61) is never used and should be removed
Context._msgData() (NovaToken.sol#182-185) is never used and should be removed
SafeMath.min(uint256,uint256) (NovaToken.sol#524-526) is never used and should be removed
SafeMath.mod(uint256,uint256) (NovaToken.sol#499-501) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (NovaToken.sol#515-522) is never used and should be removed
SafeMath.sqrt(uint256) (NovaToken.sol#529-540) is never used and should be removed
ShibaBEP20._burnFrom(address,uint256) (NovaToken.sol#826-833) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (NovaToken.sol#55-61):
- (success) = recipient.call{value: amount}() (NovaToken.sol#59)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (NovaToken.sol#134-160):
- (success,returndata) = target.call{value: weiValue}(data) (NovaToken.sol#143)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter ShibaBEP20.mint(address,uint256)._to (NovaToken.sol#755) is not in mixedCase
Parameter ShibaBEP20.mint(address,uint256)._amount (NovaToken.sol#755) is not in mixedCase
Parameter NovaToken.setupSNova(address)._sNova (NovaToken.sol#863) is not in mixedCase
Parameter NovaToken.mint(address,uint256)._to (NovaToken.sol#868) is not in mixedCase
Parameter NovaToken.mint(address,uint256)._amount (NovaToken.sol#868) is not in mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (NovaToken.sol#183)" inContext (NovaToken.sol#173-186)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
NovaToken.slitherConstructorConstantVariables() (NovaToken.sol#837-892) uses literals with too many digits:
- BURN_ADDRESS = 0x000000000000000000000000000000000000dEaD (NovaToken.sol#553)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
renounceOwnership() should be declared external:
- Ownable.renounceOwnership() (NovaToken.sol#333-336)
transferOwnership(address) should be declared external:
- Ownable.transferOwnership(address) (NovaToken.sol#342-344)
name() should be declared external:
- ShibaBEP20.name() (NovaToken.sol#591-593)
decimals() should be declared external:
- ShibaBEP20.decimals() (NovaToken.sol#598-600)
symbol() should be declared external:
- ShibaBEP20.symbol() (NovaToken.sol#605-607)
totalSupply() should be declared external:
- ShibaBEP20.totalSupply() (NovaToken.sol#612-614)
burnSupply() should be declared external:
- ShibaBEP20.burnSupply() (NovaToken.sol#619-621)
balanceOf(address) should be declared external:
- ShibaBEP20.balanceOf(address) (NovaToken.sol#626-628)
transfer(address,uint256) should be declared external:
- ShibaBEP20.transfer(address,uint256) (NovaToken.sol#638-641)
allowance(address,address) should be declared external:
- ShibaBEP20.allowance(address,address) (NovaToken.sol#646-648)
approve(address,uint256) should be declared external:

- ShibaBEP20.approve(address,uint256) (NovaToken.sol#657-660)
transferFrom(address,address,uint256) should be declared external:
- ShibaBEP20.transferFrom(address,address,uint256) (NovaToken.sol#674-686)
increaseAllowance(address,uint256) should be declared external:
- ShibaBEP20.increaseAllowance(address,uint256) (NovaToken.sol#700-703)
decreaseAllowance(address,uint256) should be declared external:
- ShibaBEP20.decreaseAllowance(address,uint256) (NovaToken.sol#719-726)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:NovaToken.sol analyzed (7 contracts with 75 detectors), 43 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration

//----------

## Slither log >> SNovaToken.sol

INFO:Detectors:
SNovaToken._swapNovaAmount(address,uint256) (SNovaToken.sol#924-932) uses a dangerous strict equality:
- penalty == 0 (SNovaToken.sol#927)
SNovaToken._writeCheckpoint(address,uint32,uint256,uint256) (SNovaToken.sol#1191-1209) uses a dangerous strict equality:
- nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (SNovaToken.sol#1201)
SNovaToken.getPenaltyPercent(address) (SNovaToken.sol#888-900) uses a dangerous strict equality:
- block.number == holderInfo.avgTransactionBlock (SNovaToken.sol#893)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
ShibaBEP20.constructor(string,string).name (SNovaToken.sol#576) shadows:
- ShibaBEP20.name() (SNovaToken.sol#592-594) (function)
- IBEP20.name() (SNovaToken.sol#207) (function)
ShibaBEP20.constructor(string,string).symbol (SNovaToken.sol#576) shadows:
- ShibaBEP20.symbol() (SNovaToken.sol#606-608) (function)
- IBEP20.symbol() (SNovaToken.sol#202) (function)
ShibaBEP20.allowance(address,address).owner (SNovaToken.sol#647) shadows:
- Ownable.owner() (SNovaToken.sol#314-316) (function)
ShibaBEP20._approve(address,address,uint256).owner (SNovaToken.sol#810) shadows:
- Ownable.owner() (SNovaToken.sol#314-316) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
SNovaToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (SNovaToken.sol#1055-1096) uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(now <= expiry,Nova::delegateBySig: signature expired) (SNovaToken.sol#1094)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (SNovaToken.sol#25-36) uses assembly
- INLINE ASM (SNovaToken.sol#32-34)
Address._functionCallWithValue(address,bytes,uint256,string) (SNovaToken.sol#133-159) uses assembly
- INLINE ASM (SNovaToken.sol#151-154)
SNovaToken.getChainId() (SNovaToken.sol#1216-1220) uses assembly
- INLINE ASM (SNovaToken.sol#1218)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (SNovaToken.sol#133-159) is never used and should be removed
Address.functionCall(address,bytes) (SNovaToken.sol#80-82) is never used and should be removed
Address.functionCall(address,bytes,string) (SNovaToken.sol#90-96) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (SNovaToken.sol#109-115) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (SNovaToken.sol#123-131) is never used and should be removed
Address.isContract(address) (SNovaToken.sol#25-36) is never used and should be removed

Address.sendValue(address,uint256) (SNovaToken.sol#54-60) is never used and should be removed
Context._msgData() (SNovaToken.sol#182-185) is never used and should be removed
SNovaToken._transfer(address,address,uint256) (SNovaToken.sol#968-986) is never used and should be removed
SafeMath.min(uint256,uint256) (SNovaToken.sol#524-526) is never used and should be removed
SafeMath.mod(uint256,uint256) (SNovaToken.sol#499-501) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (SNovaToken.sol#515-522) is never used and should be removed
SafeMath.sqrt(uint256) (SNovaToken.sol#529-540) is never used and should be removed
ShibaBEP20._burnFrom(address,uint256) (SNovaToken.sol#827-834) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (SNovaToken.sol#54-60):
	- (success) = recipient.call{value: amount}() (SNovaToken.sol#58)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (SNovaToken.sol#133-159):
	- (success,returndata) = target.call{value: weiValue}(data) (SNovaToken.sol#142)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter ShibaBEP20.mint(address,uint256)._to (SNovaToken.sol#756) is not in mixedCase
Parameter ShibaBEP20.mint(address,uint256)._amount (SNovaToken.sol#756) is not in mixedCase
Parameter SNovaToken.setupNova(ShibaBEP20)._Nova (SNovaToken.sol#872) is not in mixedCase
Parameter SNovaToken.setupMoneyPot(IMoneyPot)._moneyPot (SNovaToken.sol#878) is not in mixedCase
Parameter SNovaToken.getPenaltyPercent(address)._holderAddress (SNovaToken.sol#888) is not in mixedCase
Parameter SNovaToken.swapToNova(uint256)._amount (SNovaToken.sol#903) is not in mixedCase
Parameter SNovaToken.previewSwapNovaExpectedAmount(address,uint256)._holderAddress (SNovaToken.sol#919) is not in mixedCase
Parameter SNovaToken.previewSwapNovaExpectedAmount(address,uint256)._sNovaAmount (SNovaToken.sol#919) is not in mixedCase
Parameter SNovaToken.mint(address,uint256)._to (SNovaToken.sol#956) is not in mixedCase
Parameter SNovaToken.mint(address,uint256)._amount (SNovaToken.sol#956) is not in mixedCase
Variable SNovaToken.Nova (SNovaToken.sol#858) is not in mixedCase
Variable SNovaToken.SWAP_PENALTY_MAX_PERIOD (SNovaToken.sol#862) is not in mixedCase
Variable SNovaToken.SWAP_PENALTY_MAX_PER_SNova (SNovaToken.sol#863) is not in mixedCase
Variable SNovaToken._delegates (SNovaToken.sol#995) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (SNovaToken.sol#183)" inContext (SNovaToken.sol#173-186)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
SNovaToken.slitherConstructorConstantVariables() (SNovaToken.sol#849-1222) uses literals with too many digits:
	- BURN_ADDRESS = 0x000000000000000000000000000000000000dEaD (SNovaToken.sol#554)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
renounceOwnership() should be declared external:
	- Ownable.renounceOwnership() (SNovaToken.sol#333-336)
transferOwnership(address) should be declared external:
	- Ownable.transferOwnership(address) (SNovaToken.sol#342-344)
decimals() should be declared external:
	- ShibaBEP20.decimals() (SNovaToken.sol#599-601)
symbol() should be declared external:
	- ShibaBEP20.symbol() (SNovaToken.sol#606-608)
totalSupply() should be declared external:
	- ShibaBEP20.totalSupply() (SNovaToken.sol#613-615)
burnSupply() should be declared external:
	- ShibaBEP20.burnSupply() (SNovaToken.sol#620-622)
transfer(address,uint256) should be declared external:
	- ShibaBEP20.transfer(address,uint256) (SNovaToken.sol#639-642)
allowance(address,address) should be declared external:
	- ShibaBEP20.allowance(address,address) (SNovaToken.sol#647-649)
approve(address,uint256) should be declared external:
	- ShibaBEP20.approve(address,uint256) (SNovaToken.sol#658-661)
transferFrom(address,address,uint256) should be declared external:
	- ShibaBEP20.transferFrom(address,address,uint256) (SNovaToken.sol#675-687)

increaseAllowance(address,uint256) should be declared external:
- ShibaBEP20.increaseAllowance(address,uint256) (SNovaToken.sol#701-704)
decreaseAllowance(address,uint256) should be declared external:
- ShibaBEP20.decreaseAllowance(address,uint256) (SNovaToken.sol#720-727)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:SNovaToken.sol analyzed (8 contracts with 75 detectors), 55 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration

//----------

## Slither log >> Timelock.sol

INFO:Detectors:
Timelock.constructor(address,uint256).admin_ (Timelock.sol#213) lacks a zero-check on :
- admin = admin_ (Timelock.sol#217)
Timelock.setPendingAdmin(address).pendingAdmin_ (Timelock.sol#239) lacks a zero-check on :
- pendingAdmin = pendingAdmin_ (Timelock.sol#247)
Timelock.executeTransaction(address,uint256,string,bytes,uint256).target (Timelock.sol#272) lacks a zero-check on :
- (success,returnData) = target.call.value(value)(callData) (Timelock.sol#291)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in Timelock.executeTransaction(address,uint256,string,bytes,uint256) (Timelock.sol#272-297):
External calls:
- (success,returnData) = target.call.value(value)(callData) (Timelock.sol#291)
Event emitted after the call(s):
- ExecuteTransaction(txHash,target,value,signature,data,eta) (Timelock.sol#294)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Timelock.queueTransaction(address,uint256,string,bytes,uint256) (Timelock.sol#252-261) uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(eta >= getBlockTimestamp().add(delay),Timelock::queueTransaction: Estimated execution block must satisfy delay) (Timelock.sol#254)
Timelock.executeTransaction(address,uint256,string,bytes,uint256) (Timelock.sol#272-297) uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(getBlockTimestamp() >= eta,Timelock::executeTransaction: Transaction hasn't surpassed time lock) (Timelock.sol#277)
- require(bool,string)(getBlockTimestamp() <= eta.add(GRACE_PERIOD),Timelock::executeTransaction: Transaction is stale) (Timelock.sol#278)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
SafeMath.div(uint256,uint256) (Timelock.sol#107-109) is never used and should be removed
SafeMath.div(uint256,uint256,string) (Timelock.sol#123-133) is never used and should be removed
SafeMath.min(uint256,uint256) (Timelock.sol#172-174) is never used and should be removed
SafeMath.mod(uint256,uint256) (Timelock.sol#147-149) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (Timelock.sol#163-170) is never used and should be removed
SafeMath.mul(uint256,uint256) (Timelock.sol#81-93) is never used and should be removed
SafeMath.sqrt(uint256) (Timelock.sol#177-188) is never used and should be removed
SafeMath.sub(uint256,uint256) (Timelock.sol#46-48) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (Timelock.sol#60-69) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Timelock.executeTransaction(address,uint256,string,bytes,uint256) (Timelock.sol#272-297):
- (success,returnData) = target.call.value(value)(callData) (Timelock.sol#291)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
setDelay(uint256) should be declared external:
- Timelock.setDelay(uint256) (Timelock.sol#222-229)
acceptAdmin() should be declared external:

- Timelock.acceptAdmin() (Timelock.sol#231-237)
setPendingAdmin(address) should be declared external:
- Timelock.setPendingAdmin(address) (Timelock.sol#239-250)
queueTransaction(address,uint256,string,bytes,uint256) should be declared external:
- Timelock.queueTransaction(address,uint256,string,bytes,uint256) (Timelock.sol#252-261)
cancelTransaction(address,uint256,string,bytes,uint256) should be declared external:
- Timelock.cancelTransaction(address,uint256,string,bytes,uint256) (Timelock.sol#263-270)
executeTransaction(address,uint256,string,bytes,uint256) should be declared external:
- Timelock.executeTransaction(address,uint256,string,bytes,uint256) (Timelock.sol#272-297)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Timelock.sol analyzed (2 contracts with 75 detectors), 22 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration

//----------

## Slither log >> ShibaBonusAggregator.sol

INFO:Detectors:
Context._msgData() (ShibaBonusAggregator.sol#31-34) is never used and should be removed
SafeMath.div(uint256,uint256) (ShibaBonusAggregator.sol#212-214) is never used and should be removed
SafeMath.div(uint256,uint256,string) (ShibaBonusAggregator.sol#228-238) is never used and should be removed
SafeMath.min(uint256,uint256) (ShibaBonusAggregator.sol#277-279) is never used and should be removed
SafeMath.mod(uint256,uint256) (ShibaBonusAggregator.sol#252-254) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (ShibaBonusAggregator.sol#268-275) is never used and should be removed
SafeMath.mul(uint256,uint256) (ShibaBonusAggregator.sol#186-198) is never used and should be removed
SafeMath.sqrt(uint256) (ShibaBonusAggregator.sol#282-293) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Parameter ShibaBonusAggregator.setupMaster(IMasterBonus)._master (ShibaBonusAggregator.sol#319) is not in mixedCase
Parameter ShibaBonusAggregator.addOrRemoveContractBonusSource(address,bool)._contract (ShibaBonusAggregator.sol#323) is not in mixedCase
Parameter ShibaBonusAggregator.addOrRemoveContractBonusSource(address,bool)._add (ShibaBonusAggregator.sol#323) is not in mixedCase
Parameter ShibaBonusAggregator.addUserBonusOnFarm(address,uint256,uint256)._user (ShibaBonusAggregator.sol#327) is not in mixedCase
Parameter ShibaBonusAggregator.addUserBonusOnFarm(address,uint256,uint256)._percent (ShibaBonusAggregator.sol#327) is not in mixedCase
Parameter ShibaBonusAggregator.addUserBonusOnFarm(address,uint256,uint256)._pid (ShibaBonusAggregator.sol#327) is not in mixedCase
Parameter ShibaBonusAggregator.removeUserBonusOnFarm(address,uint256,uint256)._user (ShibaBonusAggregator.sol#333) is not in mixedCase
Parameter ShibaBonusAggregator.removeUserBonusOnFarm(address,uint256,uint256)._percent (ShibaBonusAggregator.sol#333) is not in mixedCase
Parameter ShibaBonusAggregator.removeUserBonusOnFarm(address,uint256,uint256)._pid (ShibaBonusAggregator.sol#333) is not in mixedCase
Parameter ShibaBonusAggregator.getBonusOnFarmsForUser(address,uint256)._user (ShibaBonusAggregator.sol#338) is not in mixedCase
Parameter ShibaBonusAggregator.getBonusOnFarmsForUser(address,uint256)._pid (ShibaBonusAggregator.sol#338) is not in mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (ShibaBonusAggregator.sol#32)" inContext (ShibaBonusAggregator.sol#22-35)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
owner() should be declared external:
- Ownable.owner() (ShibaBonusAggregator.sol#67-69)
renounceOwnership() should be declared external:
- Ownable.renounceOwnership() (ShibaBonusAggregator.sol#86-89)

transferOwnership(address) should be declared external:
- Ownable.transferOwnership(address) (ShibaBonusAggregator.sol#95-97)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ShibaBonusAggregator.sol analyzed (6 contracts with 75 detectors), 23 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration

//----------

## Slither log >> ShibaFactory.sol

INFO:Detectors:
Reentrancy in ShibaFactory.createPair(address,address) (ShibaFactory.sol#475-488):
External calls:
- IShibaPair(pair).initialize(token0,token1) (ShibaFactory.sol#483)
State variables written after the call(s):
- getPair[token0][token1] = pair (ShibaFactory.sol#484)
- getPair[token1][token0] = pair (ShibaFactory.sol#485)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
ShibaFactory.constructor(address,address,uint16).owner (ShibaFactory.sol#460) shadows:
- ShibaFactory.owner() (ShibaFactory.sol#467-469) (function)
- IShibaFactory.owner() (ShibaFactory.sol#60) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
ShibaFactory.setFeeAmount(uint16) (ShibaFactory.sol#504-511) should emit an event for:
- _feeAmount = _newFeeAmount (ShibaFactory.sol#510)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
ShibaFactory.constructor(address,address,uint16).owner (ShibaFactory.sol#460) lacks a zero-check on :
- feeToSetter = owner (ShibaFactory.sol#461)
- _owner = owner (ShibaFactory.sol#463)
ShibaFactory.constructor(address,address,uint16)._feeTo (ShibaFactory.sol#460) lacks a zero-check on :
- feeTo = _feeTo (ShibaFactory.sol#462)
ShibaFactory.setFeeTo(address)._feeTo (ShibaFactory.sol#490) lacks a zero-check on :
- feeTo = _feeTo (ShibaFactory.sol#492)
ShibaFactory.setFeeToSetter(address)._feeToSetter (ShibaFactory.sol#495) lacks a zero-check on :
- feeToSetter = _feeToSetter (ShibaFactory.sol#497)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in ShibaFactory.createPair(address,address) (ShibaFactory.sol#475-488):
External calls:
- IShibaPair(pair).initialize(token0,token1) (ShibaFactory.sol#483)
State variables written after the call(s):
- allPairs.push(pair) (ShibaFactory.sol#486)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in ShibaFactory.createPair(address,address) (ShibaFactory.sol#475-488):
External calls:
- IShibaPair(pair).initialize(token0,token1) (ShibaFactory.sol#483)
Event emitted after the call(s):
- PairCreated(token0,token1,pair,allPairs.length) (ShibaFactory.sol#487)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
ShibaERC20.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (ShibaFactory.sol#431-443)
uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(deadline >= block.timestamp,ShibaNovaSwap: EXPIRED) (ShibaFactory.sol#432)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
ShibaERC20.constructor() (ShibaFactory.sol#374-388) uses assembly
- INLINE ASM (ShibaFactory.sol#376-378)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

INFO:Detectors:
Math.min(uint256,uint256) (ShibaFactory.sol#129-131) is never used and should be removed
Math.sqrt(uint256) (ShibaFactory.sol#134-145) is never used and should be removed
SafeMath.div(uint256,uint256) (ShibaFactory.sol#252-254) is never used and should be removed
SafeMath.div(uint256,uint256,string) (ShibaFactory.sol#268-278) is never used and should be removed
SafeMath.min(uint256,uint256) (ShibaFactory.sol#317-319) is never used and should be removed
SafeMath.mod(uint256,uint256) (ShibaFactory.sol#292-294) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (ShibaFactory.sol#308-315) is never used and should be removed
SafeMath.mul(uint256,uint256) (ShibaFactory.sol#226-238) is never used and should be removed
SafeMath.sqrt(uint256) (ShibaFactory.sol#322-333) is never used and should be removed
ShibaERC20._burn(address,uint256) (ShibaFactory.sol#396-400) is never used and should be removed
ShibaERC20._mint(address,uint256) (ShibaFactory.sol#390-394) is never used and should be removed
UQ112x112.encode(uint112) (ShibaFactory.sol#345-347) is never used and should be removed
UQ112x112.uqdiv(uint224,uint112) (ShibaFactory.sol#350-352) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Function IShibaERC20.DOMAIN_SEPARATOR() (ShibaFactory.sol#44) is not in mixedCase
Function IShibaERC20.PERMIT_TYPEHASH() (ShibaFactory.sol#45) is not in mixedCase
Function IShibaPair.DOMAIN_SEPARATOR() (ShibaFactory.sol#88) is not in mixedCase
Function IShibaPair.PERMIT_TYPEHASH() (ShibaFactory.sol#89) is not in mixedCase
Function IShibaPair.MINIMUM_LIQUIDITY() (ShibaFactory.sol#106) is not in mixedCase
Variable ShibaERC20.DOMAIN_SEPARATOR (ShibaFactory.sol#366) is not in mixedCase
Parameter ShibaFactory.setFeeTo(address)._feeTo (ShibaFactory.sol#490) is not in mixedCase
Parameter ShibaFactory.setFeeToSetter(address)._feeToSetter (ShibaFactory.sol#495) is not in mixedCase
Parameter ShibaFactory.setFeeAmount(uint16)._newFeeAmount (ShibaFactory.sol#504) is not in mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Slither:ShibaFactory.sol analyzed (10 contracts with 75 detectors), 33 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
**//----------**

## Slither log >> ShibaMoneyPot.sol

INFO:Detectors:
ShibaMoneyPot._updateTokenPot(address) (ShibaMoneyPot.sol#766-807) performs a multiplication on the result of a division:
-tokenRewardsPerBlock = tokenPot.tokenAmount.div(updateMoneyPotPeriodNbBlocks) (ShibaMoneyPot.sol#780)
-tokenPot.accTokenPerShare = tokenPot.accTokenPerShare.add(tokenRewardsPerBlock.mul(multiplier).mul(1e12).div(lastSNovaSupply)) (ShibaMoneyPot.sol#781)
ShibaMoneyPot._updateTokenPot(address) (ShibaMoneyPot.sol#766-807) performs a multiplication on the result of a division:
-tokenRewardsPerBlock_scope_1 = tokenPot.tokenAmount.div(updateMoneyPotPeriodNbBlocks) (ShibaMoneyPot.sol#798)
-tokenPot.accTokenPerShare = tokenPot.accTokenPerShare.add(tokenRewardsPerBlock_scope_1.mul(multiplier_scope_0).mul(1e12).div(lastSNovaSupply)) (ShibaMoneyPot.sol#799)
ShibaMoneyPot.pendingTokenRewardsAmount(address,address) (ShibaMoneyPot.sol#812-837) performs a multiplication on the result of a division:
-tokenReward = _distributedMoneyPot[_token].tokenAmount.div(updateMoneyPotPeriodNbBlocks) (ShibaMoneyPot.sol#819)
-accTokenPerShare = (accTokenPerShare.add(tokenReward.mul(getMultiplier(lastRewardBlock,lastUpdateTokenPotBlocks.add(updateMoneyPotPeriodNbBlocks))).mul(1e12).div(lastSNovaSupply))) (ShibaMoneyPot.sol#823-825)
ShibaMoneyPot.pendingTokenRewardsAmount(address,address) (ShibaMoneyPot.sol#812-837) performs a multiplication on the result of a division:
-tokenReward = _distributedMoneyPot[_token].tokenAmount.div(updateMoneyPotPeriodNbBlocks) (ShibaMoneyPot.sol#819)

-accTokenPerShare =
accTokenPerShare.add(tokenReward.mul(getMultiplier(lastRewardBlock,block.number)).mul(1e12).div(lastS
NovaSupply)) (ShibaMoneyPot.sol#831-833)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
ShibaMoneyPot._updateTokenPot(address) (ShibaMoneyPot.sol#766-807) uses a dangerous strict equality:
- lastSNovaSupply == 0 (ShibaMoneyPot.sol#772)
ShibaMoneyPot.pendingTokenRewardsAmount(address,address) (ShibaMoneyPot.sol#812-837) uses a
dangerous strict equality:
- lastSNovaSupply == 0 (ShibaMoneyPot.sol#814)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in ShibaMoneyPot.depositRewards(address,uint256) (ShibaMoneyPot.sol#914-926):
External calls:
- IBEP20(_token).safeTransferFrom(msg.sender,address(this),_amount) (ShibaMoneyPot.sol#918)
State variables written after the call(s):
- pendingTokenAmount[_token] = pendingTokenAmount[_token].add(_amount)
(ShibaMoneyPot.sol#924)
Reentrancy in ShibaMoneyPot.harvestReward(address,address) (ShibaMoneyPot.sol#885-909):
External calls:
- safeTokenTransfer(_token,_sNovaHolder,sNovaHoldersRewardsInfo[_token][_sNovaHolder].pending)
(ShibaMoneyPot.sol#905)
- transferSuccess = token.transfer(_to,tokenBal) (ShibaMoneyPot.sol#993)
- transferSuccess = token.transfer(_to,_amount) (ShibaMoneyPot.sol#995)
State variables written after the call(s):
- sNovaHoldersRewardsInfo[_token][_sNovaHolder].pending = 0 (ShibaMoneyPot.sol#906)
- sNovaHoldersRewardsInfo[_token][_sNovaHolder].rewardDept =
holderBalance.mul(tokenPot.accTokenPerShare).div(1e12) (ShibaMoneyPot.sol#908)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
ShibaMoneyPot.constructor(IBEP20,address,address,uint256,uint256)._feeManager
(ShibaMoneyPot.sol#697) lacks a zero-check on :
- feeManager = _feeManager (ShibaMoneyPot.sol#703)
ShibaMoneyPot.updateFeeManager(address)._feeManager (ShibaMoneyPot.sol#727) lacks a zero-check on
:
- feeManager = _feeManager (ShibaMoneyPot.sol#729)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in ShibaMoneyPot.depositBonusRewards(address,uint256) (ShibaMoneyPot.sol#931-934):
External calls:
- IBEP20(_token).safeTransferFrom(msg.sender,address(this),_amount) (ShibaMoneyPot.sol#932)
State variables written after the call(s):
- reserveTokenAmount[_token] = reserveTokenAmount[_token].add(_amount)
(ShibaMoneyPot.sol#933)
Reentrancy in ShibaMoneyPot.depositRewards(address,uint256) (ShibaMoneyPot.sol#914-926):
External calls:
- IBEP20(_token).safeTransferFrom(msg.sender,address(this),_amount) (ShibaMoneyPot.sol#918)
State variables written after the call(s):
- reserveTokenAmount[_token] = reserveTokenAmount[_token].add(_amount)
(ShibaMoneyPot.sol#921)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Address.isContract(address) (ShibaMoneyPot.sol#34-45) uses assembly
- INLINE ASM (ShibaMoneyPot.sol#41-43)
Address._functionCallWithValue(address,bytes,uint256,string) (ShibaMoneyPot.sol#142-168) uses assembly
- INLINE ASM (ShibaMoneyPot.sol#160-163)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.functionCall(address,bytes) (ShibaMoneyPot.sol#89-91) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (ShibaMoneyPot.sol#118-124) is never used and
should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (ShibaMoneyPot.sol#132-140) is never used
and should be removed
Address.sendValue(address,uint256) (ShibaMoneyPot.sol#63-69) is never used and should be removed
Context._msgData() (ShibaMoneyPot.sol#191-194) is never used and should be removed

SafeBEP20.safeApprove(IBEP20,address,uint256) (ShibaMoneyPot.sol#403-417) is never used and should be removed
SafeBEP20.safeDecreaseAllowance(IBEP20,address,uint256) (ShibaMoneyPot.sol#428-438) is never used and should be removed
SafeBEP20.safeIncreaseAllowance(IBEP20,address,uint256) (ShibaMoneyPot.sol#419-426) is never used and should be removed
SafeBEP20.safeTransfer(IBEP20,address,uint256) (ShibaMoneyPot.sol#379-385) is never used and should be removed
SafeMath.min(uint256,uint256) (ShibaMoneyPot.sol#629-631) is never used and should be removed
SafeMath.mod(uint256,uint256) (ShibaMoneyPot.sol#604-606) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (ShibaMoneyPot.sol#620-627) is never used and should be removed
SafeMath.sqrt(uint256) (ShibaMoneyPot.sol#634-645) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (ShibaMoneyPot.sol#63-69):
- (success) = recipient.call{value: amount}() (ShibaMoneyPot.sol#67)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (ShibaMoneyPot.sol#142-168):
- (success,returndata) = target.call{value: weiValue}(data) (ShibaMoneyPot.sol#151)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter ShibaMoneyPot.distributedMoneyPot(address)._token (ShibaMoneyPot.sol#710) is not in mixedCase
Parameter ShibaMoneyPot.isDividendsToken(address)._tokenAddr (ShibaMoneyPot.sol#718) is not in mixedCase
Parameter ShibaMoneyPot.updateAddressWithoutReward(address,bool)._contract (ShibaMoneyPot.sol#723) is not in mixedCase
Parameter ShibaMoneyPot.updateAddressWithoutReward(address,bool)._unattributeDividends (ShibaMoneyPot.sol#723) is not in mixedCase
Parameter ShibaMoneyPot.updateFeeManager(address)._feeManager (ShibaMoneyPot.sol#727) is not in mixedCase
Parameter ShibaMoneyPot.getTokenAmountPotFromMoneyPot(address)._token (ShibaMoneyPot.sol#736) is not in mixedCase
Parameter ShibaMoneyPot.tokenPerBlock(address)._token (ShibaMoneyPot.sol#741) is not in mixedCase
Parameter ShibaMoneyPot.updateCurrentMoneyPot(address)._token (ShibaMoneyPot.sol#752) is not in mixedCase
Parameter ShibaMoneyPot.getMultiplier(uint256,uint256)._from (ShibaMoneyPot.sol#756) is not in mixedCase
Parameter ShibaMoneyPot.getMultiplier(uint256,uint256)._to (ShibaMoneyPot.sol#756) is not in mixedCase
Parameter ShibaMoneyPot.pendingTokenRewardsAmount(address,address)._token (ShibaMoneyPot.sol#812) is not in mixedCase
Parameter ShibaMoneyPot.pendingTokenRewardsAmount(address,address)._user (ShibaMoneyPot.sol#812) is not in mixedCase
Parameter ShibaMoneyPot.updateSNovaHolder(address)._sNovaHolder (ShibaMoneyPot.sol#843) is not in mixedCase
Parameter ShibaMoneyPot.harvestRewards(address)._sNovaHolder (ShibaMoneyPot.sol#874) is not in mixedCase
Parameter ShibaMoneyPot.harvestReward(address,address)._sNovaHolder (ShibaMoneyPot.sol#885) is not in mixedCase
Parameter ShibaMoneyPot.harvestReward(address,address)._token (ShibaMoneyPot.sol#885) is not in mixedCase
Parameter ShibaMoneyPot.depositRewards(address,uint256)._token (ShibaMoneyPot.sol#914) is not in mixedCase
Parameter ShibaMoneyPot.depositRewards(address,uint256)._amount (ShibaMoneyPot.sol#914) is not in mixedCase
Parameter ShibaMoneyPot.depositBonusRewards(address,uint256)._token (ShibaMoneyPot.sol#931) is not in mixedCase
Parameter ShibaMoneyPot.depositBonusRewards(address,uint256)._amount (ShibaMoneyPot.sol#931) is not in mixedCase
Parameter ShibaMoneyPot.addTokenToRewards(address)._token (ShibaMoneyPot.sol#939) is not in mixedCase
Parameter ShibaMoneyPot.removeTokenToRewards(address)._token (ShibaMoneyPot.sol#953) is not in mixedCase
Parameter ShibaMoneyPot.addToPendingFromReserveTokenAmount(address,uint256)._token (ShibaMoneyPot.sol#980) is not in mixedCase

Parameter ShibaMoneyPot.addToPendingFromReserveTokenAmount(address,uint256)._amount (ShibaMoneyPot.sol#980) is not in mixedCase
Parameter ShibaMoneyPot.safeTokenTransfer(address,address,uint256)._token (ShibaMoneyPot.sol#988) is not in mixedCase
Parameter ShibaMoneyPot.safeTokenTransfer(address,address,uint256)._to (ShibaMoneyPot.sol#988) is not in mixedCase
Parameter ShibaMoneyPot.safeTokenTransfer(address,address,uint256)._amount (ShibaMoneyPot.sol#988) is not in mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (ShibaMoneyPot.sol#192)" inContext (ShibaMoneyPot.sol#182-195)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
owner() should be declared external:
- Ownable.owner() (ShibaMoneyPot.sol#324-326)
renounceOwnership() should be declared external:
- Ownable.renounceOwnership() (ShibaMoneyPot.sol#343-346)
transferOwnership(address) should be declared external:
- Ownable.transferOwnership(address) (ShibaMoneyPot.sol#352-354)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ShibaMoneyPot.sol analyzed (8 contracts with 75 detectors), 60 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration

//----------

## Slither log >> ShibaRouter.sol

INFO:Detectors:
ShibaRouter.removeLiquidity(address,address,uint256,uint256,uint256,address,uint256) (ShibaRouter.sol#619-635) ignores return value by IShibaPair(pair).transferFrom(msg.sender,pair,liquidity) (ShibaRouter.sol#629)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
ShibaLibrary.getAmountsOut(address,uint256,address[]).i (ShibaRouter.sol#481) is a local variable never initialized
ShibaRouter._swap(uint256[],address[],address).i (ShibaRouter.sol#729) is a local variable never initialized
ShibaRouter._swapSupportingFeeOnTransferTokens(address[],address).i (ShibaRouter.sol#838) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
ShibaRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256) (ShibaRouter.sol#549-576) ignores return value by IShibaFactory(factory).createPair(tokenA,tokenB) (ShibaRouter.sol#559)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
ShibaRouter.constructor(address,address)._factory (ShibaRouter.sol#538) lacks a zero-check on :
- factory = _factory (ShibaRouter.sol#539)
ShibaRouter.constructor(address,address)._WETH (ShibaRouter.sol#538) lacks a zero-check on :
- WETH = _WETH (ShibaRouter.sol#540)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
ShibaRouter._swap(uint256[],address[],address) (ShibaRouter.sol#728-739) has external calls inside a loop: IShibaPair(ShibaLibrary.pairFor(factory,input,output)).swap(amount0Out,amount1Out,to,new bytes(0)) (ShibaRouter.sol#735-737)
ShibaRouter._swapSupportingFeeOnTransferTokens(address[],address) (ShibaRouter.sol#837-854) has external calls inside a loop: (reserve0,reserve1) = pair.getReserves() (ShibaRouter.sol#845)
ShibaRouter._swapSupportingFeeOnTransferTokens(address[],address) (ShibaRouter.sol#837-854) has external calls inside a loop: amountInput = IERC20(input).balanceOf(address(pair)).sub(reserveInput) (ShibaRouter.sol#847)
ShibaRouter._swapSupportingFeeOnTransferTokens(address[],address) (ShibaRouter.sol#837-854) has external calls inside a loop: pair.swap(amount0Out,amount1Out,to,new bytes(0)) (ShibaRouter.sol#852)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

INFO:Detectors:
SafeMath.div(uint256,uint256) (ShibaRouter.sol#336-338) is never used and should be removed
SafeMath.div(uint256,uint256,string) (ShibaRouter.sol#352-362) is never used and should be removed
SafeMath.min(uint256,uint256) (ShibaRouter.sol#401-403) is never used and should be removed
SafeMath.mod(uint256,uint256) (ShibaRouter.sol#376-378) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (ShibaRouter.sol#392-399) is never used and should be removed
SafeMath.sqrt(uint256) (ShibaRouter.sol#406-417) is never used and should be removed
TransferHelper.safeApprove(address,address,uint256) (ShibaRouter.sol#502-506) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in TransferHelper.safeApprove(address,address,uint256) (ShibaRouter.sol#502-506):
- (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (ShibaRouter.sol#504)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (ShibaRouter.sol#508-512):
- (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value)) (ShibaRouter.sol#510)
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256) (ShibaRouter.sol#514-518):
- (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (ShibaRouter.sol#516)
Low level call in TransferHelper.safeTransferETH(address,uint256) (ShibaRouter.sol#520-523):
- (success) = to.call{value: value}(new bytes(0)) (ShibaRouter.sol#521)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IShibaPair.DOMAIN_SEPARATOR() (ShibaRouter.sol#54) is not in mixedCase
Function IShibaPair.PERMIT_TYPEHASH() (ShibaRouter.sol#55) is not in mixedCase
Function IShibaPair.MINIMUM_LIQUIDITY() (ShibaRouter.sol#72) is not in mixedCase
Function IShibaRouter01.WETH() (ShibaRouter.sol#93) is not in mixedCase
Variable ShibaRouter.WETH (ShibaRouter.sol#531) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable IShibaRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (ShibaRouter.sol#98) is too similar to IShibaRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (ShibaRouter.sol#99)
Variable ShibaRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (ShibaRouter.sol#580) is too similar to ShibaRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (ShibaRouter.sol#553)
Variable ShibaRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (ShibaRouter.sol#580) is too similar to ShibaRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (ShibaRouter.sol#581)
Variable ShibaRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (ShibaRouter.sol#552) is too similar to ShibaRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (ShibaRouter.sol#553)
Variable IShibaRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (ShibaRouter.sol#98) is too similar to ShibaRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBDesired (ShibaRouter.sol#553)
Variable ShibaRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (ShibaRouter.sol#580) is too similar to IShibaRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (ShibaRouter.sol#99)
Variable ShibaRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (ShibaRouter.sol#552) is too similar to IShibaRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (ShibaRouter.sol#99)

Variable ShibaRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountADesired (ShibaRouter.sol#552) is too similar to ShibaRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (ShibaRouter.sol#581)
Variable IShibaRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (ShibaRouter.sol#98) is too similar to ShibaRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (ShibaRouter.sol#581)
Variable ShibaRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountAOptimal (ShibaRouter.sol#570) is too similar to ShibaRouter._addLiquidity(address,address,uint256,uint256,uint256,uint256).amountBOptimal (ShibaRouter.sol#565)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
quote(uint256,uint256,uint256) should be declared external:
- ShibaRouter.quote(uint256,uint256,uint256) (ShibaRouter.sol#919-921)
getAmountOut(uint256,uint256,uint256) should be declared external:
- ShibaRouter.getAmountOut(uint256,uint256,uint256) (ShibaRouter.sol#923-931)
getAmountIn(uint256,uint256,uint256) should be declared external:
- ShibaRouter.getAmountIn(uint256,uint256,uint256) (ShibaRouter.sol#933-941)
getAmountsOut(uint256,address[]) should be declared external:
- ShibaRouter.getAmountsOut(uint256,address[]) (ShibaRouter.sol#943-951)
getAmountsIn(uint256,address[]) should be declared external:
- ShibaRouter.getAmountsIn(uint256,address[]) (ShibaRouter.sol#953-961)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:ShibaRouter.sol analyzed (10 contracts with 75 detectors), 42 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration




**Email: audit@EtherAuthority.io**